CIT 2026

WORKSHOP

ВЕСНЯНА ШКОЛА-СЕМІНАР

молодих вчених і студентів

КОМП'ЮТЕРНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

4-5 травня
2026 року

м. Тернопіль
вул. О.Теліги, 8

ЗУНУ
ЗАХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
1966-2026

ОРГАНІЗАТОРИ

- Західноукраїнський національний університет
- Факультет комп'ютерних інформаційних технологій
- Асоціація фахівців комп'ютерних інформаційних технологій

Західноукраїнський національний університет
Факультет комп’ютерних інформаційних технологій
Асоціація фахівців комп’ютерних інформаційних технологій

МАТЕРІАЛИ ВЕСНЯНОЇ ШКОЛИ-СЕМІНАРУ
МОЛОДИХ ВЧЕНИХ І СТУДЕНТІВ

КОМП'ЮТЕРНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

COMPUTER INFORMATION TECHNOLOGIES

4-5 травня 2026 року

CIT’2026

Тернопіль
ЗУНУ
2026

ББК 32.97
УДК 004.2-3+004.9+51.7+519.6-8

Організатори школи-семінару:

Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Асоціація фахівців комп'ютерних інформаційних технологій

32.97 Комп'ютерні інформаційні технології: Матеріали весняної школи-семінару молодих вчених і студентів CIT'2026. – Тернопіль: ЗУНУ, 2026.

У матеріалах семінару опубліковані результати наукових досліджень і розробок науковців та студентів факультету комп'ютерних інформаційних технологій ЗУНУ з таких напрямків: математичні моделі об'єктів та процесів, комп'ютерні мережеві технології; спеціалізовані комп'ютерні системи; системи штучного інтелекту; інженерія програмного забезпечення; комп'ютерні технології інформаційної безпеки та управління проектами. Матеріали призначені для наукових співробітників, викладачів, інженерно-технічних працівників, аспірантів та студентів.

Відповідальний за випуск:
Пукас А.В., д. т. н., професор, завідувач кафедри комп'ютерних наук

Відповідальність за достовірність, стиль викладення та зміст надрукованих матеріалів несуть автори.

ЗМІСТ

УДК 004.41

РОЗРОБКА ІНФОРМАЦІЙНОГО МОБІЛЬНОГО ДОДАТКУ ДЛЯ НАДАННЯ АДМІНІСТРАТИВНИХ ПОСЛУГ

Сімак А.Ю.[1)], Фурман А.С.[2)]
Західноукраїнський національний університет
[1)]д.філ., викладач; [2)]бакалавр

I. Постановка проблеми

Ефективна організація надання адміністративних послуг та інформування населення є однією з ключових умов підвищення якості роботи органів місцевого самоврядування. В умовах цифровізації суспільства та зростання потреб користувачів у швидкому доступі до інформації, особливо актуальним стає питання впровадження сучасних інформаційних систем, зокрема мобільних додатків.Важливими факторами також є забезпечення актуальності новин, надійність передачі даних та можливість оперативного зворотного зв'язку між громадянами та органами влади.

Аналіз існуючих рішень показав, що більшість з них є узагальненими або орієнтовані лише на надання адміністративних послуг, або виконують функцію інформаційних порталів, не забезпечуючи комплексного підходу до взаємодії з користувачами. Крім того, часто спостерігається перевантаження інтерфейсу та недостатня адаптація під потреби конкретної громади. Це зумовлює необхідність розробкимасштабованого програмного рішення, яке б забезпечилонадання адміністративних послуг, інформування населення, ефективну обробку даних та підвищило якість комунікації між громадянами і органами місцевого самоврядування.

II. Мета роботи

Метою роботи є розробка мобільного додатка для автоматизації процесів подання заявок на адміністративні послуги, що базується на гібридній системі управління даними для підвищення ефективності комунікації між владою та громадою.

III. Основна частина

Проектована система має модульну багаторівневу архітектуру (див.рис.1), побудовану з урахуванням вимог до масштабованості, продуктивності та зручності використання. Система логічно поділена на 3 основні структурні компоненти: клієнтський інтерфейс, серверну частину, систему управління даними та інфраструктуру розгортання.

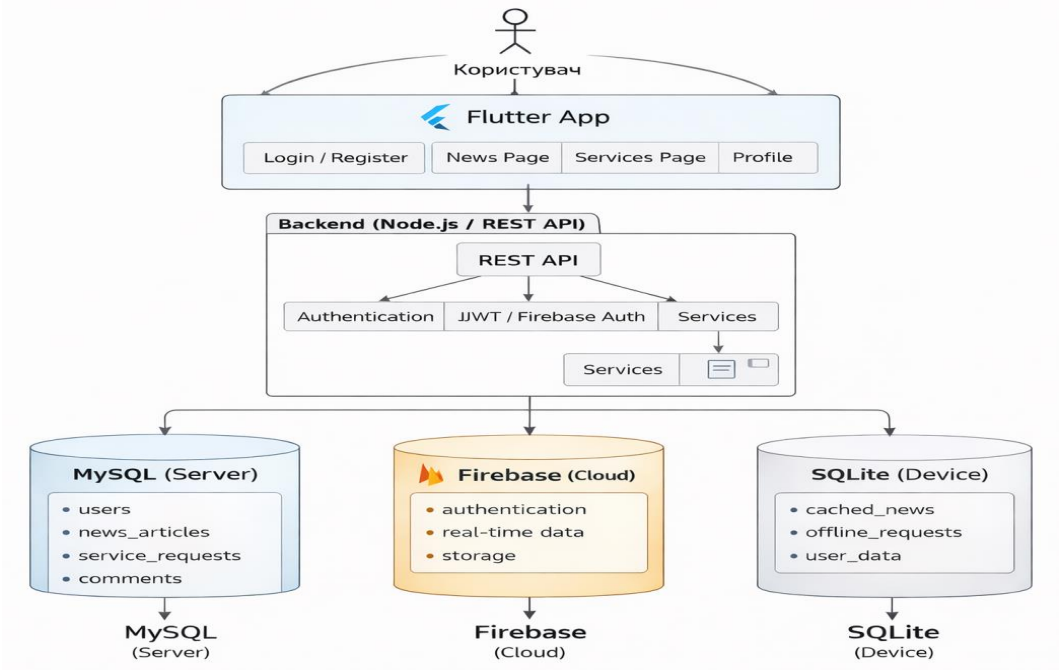

Рисунок 1 – Архітектура системи

1) Клієнтська частина (Frontend) – це мобільний додаток, через який здійснюється взаємодія користувача із системою. Вона реалізована за допомогою фреймворкуFlutter [1], що дозволяє створювати кросплатформні додатки з єдиною кодовою базою. Інтерфейс забезпечує доступ до

основних функцій системи: перегляд новин, подання заявок на адміністративні послуги та отримання зворотного зв'язку. Особлива увага приділяється адаптивності та зручності користування.

2) Серверна частина (Backend API) реалізує бізнес-логіку системи та забезпечує обробку запитів від клієнта. Вона відповідає за прийом і обробку заявок, перевірку введених даних, маршрутизацію запитів до відповідних служб та формування відповідей для користувача. Взаємодія між клієнтом і сервером здійснюється через REST API, що забезпечує гнучкість та масштабованість системи.

3) Система управління даними (Database) використовується для зберігання інформації про користувачів, адміністративні послуги, новини та заявки. Для реалізації обрано реляційну систему управління базами даних MySQL [2], що забезпечує надійне зберігання даних, їх цілісність та ефективну обробку запитів, SQLLite (кеш) відповідає за швидкий старт і роботу в офлайн режимі.

Інфраструктура розгортання включає серверне середовище, яке забезпечує стабільну роботу системи, обробку запитів користувачів та доступ до бази даних. Система може бути розгорнута як на локальному сервері, так і в хмарному середовищі, що дозволяє забезпечити масштабованість та доступність сервісу [3].

Взаємодія між компонентами системи реалізована за клієнт-серверною моделлю [4]. Додаток формує запити до серверної частини через API, опрацьовує їх, звертається до бази даних для отримання або збереження інформації та повертає результати клієнту, щозображено на рисунку 2.

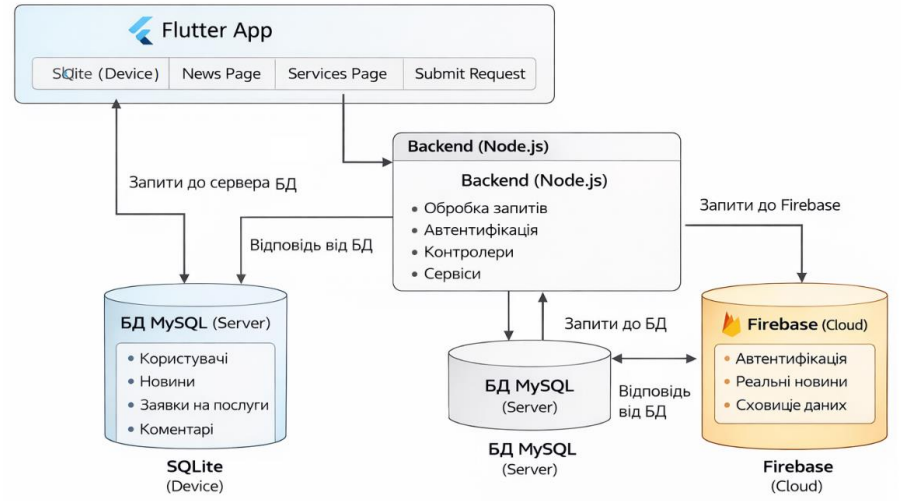

Рисунок 2 – Схема взаємодії мобільного додатку з багаторівневою системою збереження даних

В мобільному додатку, користувач може сформувати заявки на адміністративні послуги, які після збереження, відправляться відповідному підрозділу для обробки, а користувач отримує повідомлення про статус виконання.Для новин система забезпечує отримання актуальної інформації з бази даних та її відображення користувачам, що дозволяєоперативно отримувати важливі повідомлення та оголошення.

Висновок

Запропоноване рішення особливо з орієнтацією на вирішення актуальних задач локальних населених пунктів, дозволяє автоматизувати процеси подання та обробки заявок, зменшити час обслуговування користувачів та підвищити якість взаємодії між громадянами та органами місцевого самоврядування. Архітектура додатку забезпечує ефективне вирішення задачі організації надання адміністративних послуг та інформування населення. Вона характеризується гнучкістю, масштабованістю та зручністю використання, що дозволяє адаптувати систему до потреб конкретної громади. Перспективами подальшого розвитку є інтеграція з державними електронними сервісами, розширення функціоналу системи та впровадження додаткових механізмів аналітики для покращення якості обслуговування користувачів.

Список використаних джерел

1. Flutter Documentation [Електронний ресурс] – Режим доступу: https://flutter.dev.
2. MySQL Documentation [Електронний ресурс] – Режим доступу: https://dev.mysql.com/doc/.
3. Кузьменко О. В. Сучасні інформаційні технології в управлінні даними. – Системні дослідження та інформаційні технології. – 2020. – № 3. – С. 72–79.
4. Пелещишин А. М., Тимовчак-Максимець О. Ю. Моделі та методи побудови інформаційних систем для соціальних комунікацій. – Інформаційні системи та мережі. – 2022. – № 10. – С. 25–33.

УДК 004.41

ІНТЕЛЕКТУАЛІЗОВАНА ПРОГРАМНА СИСТЕМА КООРДИНАЦІЇ ВОЛОНТЕРСЬКИХ ІНІЦІАТИВ ТА СОЦІАЛЬНОЇ ВЗАЄМОДІЇ

Вігурський А.Р.
Західноукраїнський національний університет
бакалавр

I. Постановка проблеми

В умовах повномасштабного збройного вторгнення Росії в Україну волонтерський рух набув безпрецедентного масштабу.За дослідженнями, станом на 2025 рік в Україні діяло понад 100 тисяч громадських організацій і ще близько 33 тисяч благодійних фондів[4].Загалом упродовж 2022 року 74 відсотки українців долучились до волонтерської діяльності. У 2023 році волонтерську діяльність провадили 37 відсотків українців, у 2024 році цей показник зменшився до 27 відсотків – спостерігається значна регресія залученості громадян до волонтерської діяльності. За даними опитування «Омнібус» 2025 року, незважаючи на визнання ролі волонтерства у захисті та підтримці держави та високий рівень залученості до допомоги армії й постраждалим від війни з росією, оцінка чесності та прозорості діяльності волонтерів є значно стриманішою. Лише 46,8% респондентів погоджуються з твердженням, що волонтери діють чесно та прозоро, тоді як понад третина опитаних (36,3%) займає нейтральну позицію[6]

Координація тисяч подій, збирань та акцій здійснюється переважно через соціальні мережі — Telegram, Facebook, Instagram, — що призводить до дублювання зусиль, втрати інформації, відсутності верифікації організацій, QR-контролю присутності та інтелектуального підбору подій. Відсутність централізованої платформи з інтелектуальними функціями підбору подій відповідно до інтересів і можливостей волонтера є ключовою проблемою, яку вирішує дана робота.

II. Мета роботи

Метою роботи є проектування та розробка інтелектуалізованої програмної системи координації волонтерських ініціатив і соціальної взаємодії, що забезпечує верифікацію організацій, персоналізований підбір подій на основі алгоритмів машинного навчання та комплексну.

III. Основна частина

Після проведеного порівняльного аналізу чотирьох основних платформ-аналогів: Eventbrite, VolunteerMatch, FacebookEvents та Volon-ter.ua. Жодна з платформ не задовольняє повністю потреби українського волонтерського руху за ключовими критеріями (безкоштовність, локалізація, профіль волонтера, рекомендації, верифікація організацій, аналітика).

Eventbrite є платним сервісом без української локалізації та профілів волонтерів, проте підтримує аналітику організатора, QR-верифікацію та частково відповідає GDPR. VolunteerMatch пропонує профілі волонтерів, аналітику та відповідність GDPR, але не має локалізації, рекомендацій і верифікації організацій. FacebookEvents безкоштовний з частковою українською локалізацією, однак не підтримує профілі волонтерів, рекомендації, верифікацію чи GDPR. Volon-ter.ua має безкоштовний доступ, часткову локалізацію і профілі волонтерів, але відсутні AI-рекомендації, верифікація організацій, QR та аналітика.

Запропонована система охоплює всі ключові функціональні критерії, зокрема є єдиним рішенням, яке реалізує інтелектуальну систему рекомендацій на основі машинного навчання.

Структурну основу розробленої платформи становить трирівнева клієнт-серверна архітектура, побудована з використанням мікросервісного підходу (рисунок 1). Функціональний обсяг системи декомпоновано на п'ять автономних модулів, що взаємодіють між собою. Зокрема, підсистема безпеки реалізує механізми автентифікації та авторизації на базі протоколів JWT та OAuth 2.0. Сервіс управління подіями забезпечує реєстрацію та обробку даних із використанням Elasticsearch для оптимізації пошукових запитів. Інтелектуальний модуль рекомендацій, побудований на базі FastAPI, відповідає за реалізацію гібридних алгоритмів персоналізації контенту. Комунікаційна складова представлена сервісом мультиканальних сповіщень, а аналітичний блок забезпечує збір та агрегацію метрик активності користувачів. Координація взаємодії між сервісами реалізована через брокер повідомлень RabbitMQ, що гарантує високу надійність та асинхронність обміну даними у розподіленому середовищі.

Для забезпечення високої масштабованості та продуктивності системи обрано багатокомпонентний технологічний стектатрирівневу клієнт-серверну архітектуру з елементами

мікросервісного підходу[1]. Клієнтська частина базується на фреймворку React із використанням TypeScript, що гарантує типізацію та стабільність інтерфейсу. Серверна логіка розподілена між платформою Node.js (Express.js) для основних бізнес-процесів та FastAPI (Python) для реалізації інтелектуальних сервісів. Систему збереження даних побудовано на базі PostgreSQL з розширенням PostGIS, інтегрованої з інструментами Redis для кешування та Elasticsearch для оптимізації повнотекстового пошуку. Розгортання та оркестрація здійснюються за допомогою Docker та Kubernetes у хмарній інфраструктурі AWS (EC2, S3, RDS). Обрана архітектурна модель дозволяє забезпечити час відповіді API до 200 мс при навантаженні 1 000 RPS та підтримувати рівень доступності сервісів не нижче 99,5% [2].

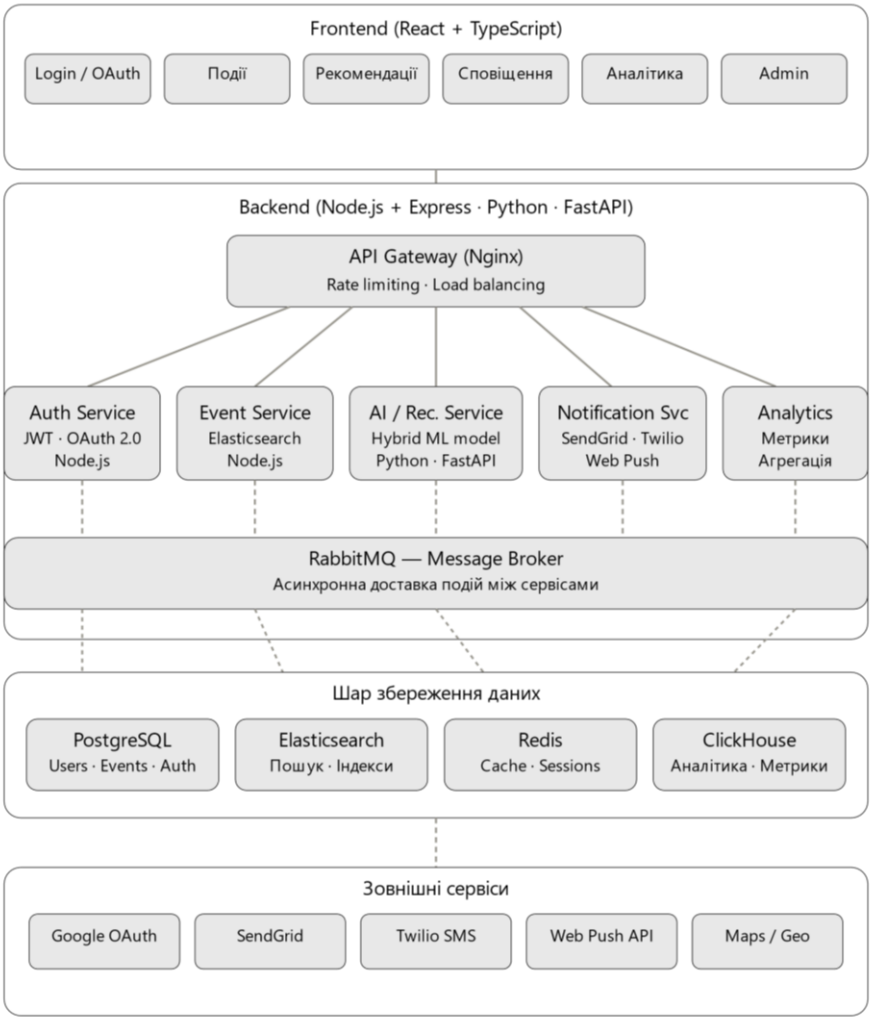

Рисунок 2 – Архітектура інтелектуалізованої програмної системи координації волонтерських ініціатив

Ключовим інтелектуальним компонентом платформи є гібридна система рекомендацій, що поєднує три методи: контентну фільтрацію (TF-IDF та cosinesimilarity), колаборативну фільтрацію ($k-NN$ по схожих користувачах, k=10) та геолокаційний фактор[3]. Зважена оцінка релевантності події e для користувача u обчислюється за формулою:

$$score(u,e) = \alpha \cdot content_{score(u.e)} + \beta \cdot collab_{score(u,e)} + \gamma \cdot geo_{score(u,e)}, \quad (1)$$

де: u – користувач; e – подія; $\alpha = 0.4$ – вага контентної фільтрації; $\beta = 0.4$ – вага колаборативної фільтрації; $\gamma = 0.2$ – вага геолокаційного фактора.

Контентна оцінка визначається через косинусну подібність між вектором інтересів користувача та TF-IDF вектором опису події. Колаборативна оцінка агрегується по 10 найбільш схожих користувачах з урахуванням їхніх рейтингів подій. Геолокаційний фактор обчислюється за формулою:

$$geo_{score(u,e)} = \frac{1}{1 + dist_{km(u.location, e.location)}}, \quad (2)$$

де $u.location, e.location$ – точки знаходження користувача і події, $dist_{km}$ – розрахунок відстані між точками яке обчислюється за формулою гаверсинусів[6].

Результати обчислень кешуються у Redis з TTL 30 хвилин, що дозволяє знизити навантаження на AI-сервіс на 70–80 % при типовому розподілі запитів. Згідно з академічними дослідженнями, гібридні підходи демонструють покращення метрики precision@10 на 15–25 % порівняно з окремими методами [3].

Атомарність реєстрації волонтера та захист від конкурентних колізій (до 1000 одночасних запитів) забезпечуються транзакціями PostgreSQL рівня SERIALIZABLE із механізмом SELECT FOR UPDATE. Після фіксації даних через RabbitMQ асинхронно активується підсистема сповіщень, яка надсилає волонтеру підтвердження. Для фізичної верифікації учасників генерується криптографічно захищений персональний QR-код на базі UUID4 із підписом HMAC-SHA256.

Безпека персональних даних реалізовуватимесь відповідно до вимог Загального регламенту захисту даних (GDPR). Платформа застосовує принцип мінімізації даних: збираються лише ті відомості, які необхідні для функціонування системи. Користувачі мають можливість у будь-який момент переглянути, скоригувати або видалити свій профіль, а також відкликати згоду на обробку персональних даних. Усі дані передаються по зашифрованих каналах (TLS 1.3), а паролі зберігаються виключно у вигляді хешівbcrypt із сіллю. Журнали аудиту фіксують усі дії адміністраторів із персональними даними, що забезпечує прозорість та відповідальність у разі перевірки регулятором.

Верифікація організацій реалізована як багатоетапний процес. На першому етапі організація подає заявку із зазначенням юридичних реквізитів, посилань на офіційні ресурси та документів, що підтверджують реєстрацію. Модератори платформи перевіряють надані відомості вручну та у разі успішної верифікації присвоюють організації відповідний статус із відображенням відмітки “Верифіковано” у публічному профілі. Повторна верифікація проводиться щорічно або у разі зміни ключових реквізитів організації. Такий підхід суттєво знижує ризик шахрайства та підвищує рівень довіри волонтерів до публікованих подій [5].

У рамках тестування системибуде проведено навантажувальне тестування із використанням ApacheJMeter, при навантаженні 1 000 одночасних запитів впродовж 72-годинного тест-запуску.

Висновки

У роботі розроблено концепцію та архітектуру інтелектуалізованої програмної системи координації волонтерських ініціатив. Запропоноване рішення спрямоване на подолання проблем фрагментованості інформації, відсутностіверифікації організаторів та нерелевантності сповіщень в українському волонтерському середовищі. Практична цінність розробки полягає у впровадженні гібридної системи рекомендацій для персоналізованого підбору подій, забезпеченні надійної багаторівневої верифікації та використанні механізму QR-контролю присутності учасників з урахуванням вимог GDPR. Використання мікросервісного підходу гарантує високу відмовостійкість та можливість горизонтального масштабування системи в умовах зростання кількості користувачів. Перспективами подальших досліджень та розвитку платформи є розширення комунікаційних каналів(зокрема через TelegramBot API), створення кросплатформних мобільних застосунків, інтеграція великих мовних моделей для автоматизації процесу генерації описів подій, а також впровадження аналітичної підсистеми та автоматизованих звітів, що спростить комунікацію організаторів із донорами та державними структурами.

Список використаних джерел

1. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'ReillyMedia, 2021. 620 p.
2. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE). ISO, 2011.
3. Aggarwal C. C. Recommender Systems: TheTextbook. Cham : Springer, 2016. 498 p. DOI: https://doi.org/10.1007/978-3-319-29659-3.
4. Волонтерський рух для безпеки й оборони України: як це працює і що потрібно для його стійкості URL: https://zmina.info/articles/nyzovyj-volonterskyj-ruh-yak-chastyna-oborony-ukrayiny-yak-gromadyanska-samoorganizacziya-stala-elementom-bezpeky-derzhavy/
5. European Commission. General Data Protection Regulation (GDPR). Regulation (EU) 2016/679. OfficialJournalofthe European Union, 2016.
6.Жуленьова О. Стан і перспективи волонтерства в Україні URL:https://isnasu.org.ua/analytics/013_Zhuleniova_O_-_Stan_i_perspektyvy_volonterstva.php

УДК 004.8

МАТЕМАТИЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗОВАНОГО АНАЛІЗУ ЗАДАЧ ПРИРОДНОЮ МОВОЮ

Войтюк І.Ф.[1)], Крушельницька Х.О.[2)]
Західноукраїнський національний університет
[1)]к.т.н.,доцент; [2)]магістрант

І. Постановка проблеми

Сучасні системи комп'ютерної алгебри вимагають від користувачів володіння специфічним синтаксисом, що обмежує їх доступність для фахівців прикладних галузей. Використання методів обробки природної мови (NLP) дозволяє створити інтерфейс, де математична задача вводиться у довільній формі. Актуальною проблемою залишається висока чутливість алгоритмів розпізнавання до неоднозначності тексту та потреба у швидкій формалізації неструктурованих даних у математичні вирази.

ІІ.Мета роботи

Метою роботи є розробка методу та програмного забезпечення для інтелектуальної автоматизації розв’язання математичних задач, що включає NLP-модуль для перетворення тексту у формальні моделі та обчислювальний блок для генерації крокових пояснень.

ІІІ. Математична модель аналізу тексту

Математична модель аналізу тексту в NLP – це сукупність алгоритмів та структур даних, що перетворюють неструктурований текст на числові представлення (вектори), які комп'ютер може обробляти для розуміння змісту, контексту та структури мови[1].

Для створення ефективного програмного забезпечення необхідно формалізувати процес перетворення неструктурованого тексту природної мови у детерміновану математичну модель. Предметна область розв’язання задач за допомогою NLP передбачає обробку мови на базі лінгвістичних та статистичних технологій.

Процес інтерпретації умови задачі системою можна представити як функцію відображення f_{nlp}, яка трансформує вхідну послідовність токенів $T = \{w_1, w_2 ...., w_n\}$ у формальну структуру M_{formal}:

$$f_{nlp} T \to M_{formal} \tag{1}$$

де T – множина слів та символів, введених користувачем, а M_{formal}– структуроване представлення задачі, придатне для обчислювального модуля.

Однією з ключових проблем предметної області є неоднозначність природної мови, що може призводити до неправильного розуміння задачі системою. Для вирішення цієї проблеми на етапі семантичного аналізу використовуються методи векторного представлення слів та сучасні нейронні підходи на основі архітектури Transformer [2]. Для визначення категорії задачі (арифметика, алгебра, аналіз) система обчислює косинусну подібність між вектором вхідного запиту V_{input} та еталонними векторами категорій V_{cat}.

Цей підхід дозволяє класифікувати задачі за типом математичного апарату, що є критичним, оскільки складність розв'язання прямо залежить від сегмента (від низької для арифметики до високої для математичного аналізу).

Після класифікації відбувається трансляція (Mapping) – перетворення виявлених сутностей у формальну мову,таку як LaTeX або MathML. Математично цей процес описується як побудова графа зв’язків, де вузлами є математичні об’єкти (змінні, константи), а ребрами – операції між ними. Формально, отримана модель M_{formal} представляється у вигляді ієрархічного дерева виразів:

$$E = \langle op, \{\arg_1, \arg_2, ..., \arg_k \rangle \tag{2}$$

де op – математичний оператор, а arg – відповідні операнди.

Така формалізація дозволяє системі не лише знаходити результат, але й виконувати аналіз результатівна виході, надавати пояснення та візуалізувати отримані дані. Це відповідає головній меті

системи – забезпеченню зручної взаємодії користувача з обчислювальними алгоритмами без необхідності спеціальної підготовки.

IV. Програмна реалізація та архітектура ПЗ

Для практичної реалізації було розроблено веб-орієнтоване програмне забезпечення на базі мови програмування Python (фреймворкFlask) із застосуванням модульного принципу, що забезпечує незалежність процесів аналізу, обчислення та візуалізації. Для збереження результатів аналізу використано реляційну СУБД PostgreSQL із підтримкою JSONB для роботи з напівструктурованими даними. Реляційна модель зберігає профілі користувачів, а формат JSONB дозволяє оперувати складними напівструктурованими даними – математичними моделями (AST) та даними для візуалізації (див. рис.1).

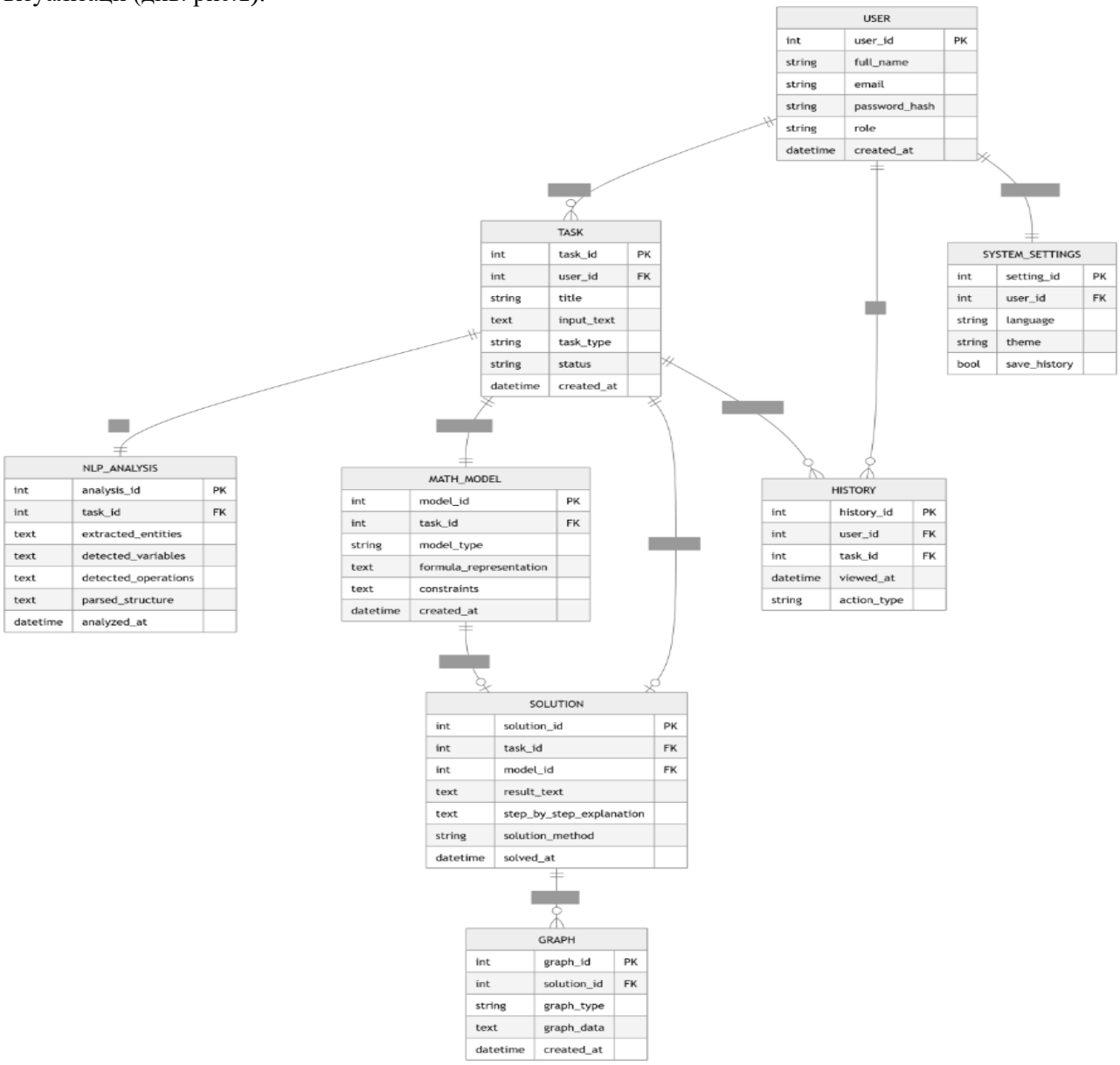

Рисунок1 – ER-діаграма системи

Ключовим компонентом системи є NLP-модуль, який здійснює токенізацію, виділення сутностей та синтаксичний розбір вхідного тексту. Для формалізації задачі використовується модель абстрактного синтаксичного дерева (AST), що дозволяє автоматично генерувати математичну модель. Це забезпечує універсальність обробки різних розділів математики без зміни схеми БД.

Для забезпечення взаємодії між компонентами системи використовується REST API, що дозволяє передавати структуровані дані між NLP-модулем, обчислювальним блоком та підсистемою візуалізації. Такий підхід забезпечує незалежність окремих компонентів та спрощує подальше масштабування програмного забезпечення.

Архітектурний стиль REST забезпечує стандартизований механізм взаємодії між програмними компонентами та широко використовується у сучасних розподілених системах завдяки масштабованості, незалежності сервісів та простоті інтеграції [3]. Використання REST API також забезпечує уніфікований формат обміну даними між компонентами системи за допомогою JSON-повідомлень. Це спрощує інтеграцію програмного забезпечення із зовнішніми сервісами та веб-застосунками. Крім того, такий підхід дозволяє зменшити залежність між окремими модулями системи та підвищує зручність їх подальшого оновлення і супроводу.

Важливою перевагою запропонованого підходу є можливість адаптації системи до різних типів математичних задач. Завдяки формалізації вхідних даних у вигляді AST-моделі програмне забезпечення може працювати не лише з арифметичними та алгебраїчними виразами, але й із задачами математичного аналізу, лінійної алгебри та елементами статистичної обробки даних.

Крім того, використання NLP-технологій дозволяє зменшити залежність користувача від спеціалізованого математичного синтаксису. Це забезпечує більш природний спосіб взаємодії із системою та розширює можливості її використання у навчальному процесі, прикладних дослідженнях та інженерних задачах.

Після формування AST-моделі система передає її до обчислювального модуля, який виконує аналіз структури виразу та генерує покрокове розв’язання задачі. Результати обчислень можуть бути представлені у вигляді текстового пояснення, математичних формул або графічної візуалізації залежно від типу задачі.

Окрему увагу приділено масштабованості програмного забезпечення. Використання модульного принципу дозволяє інтегрувати додаткові сервіси для підтримки нових математичних методів без необхідності зміни базової архітектури системи. Для підвищення точності обробки математичних задач система використовує поетапний механізм аналізу тексту. На першому етапі здійснюється токенізація та виділення ключових математичних сутностей. Далі виконується семантичний аналіз для визначення типу задачі та побудови формалізованого представлення у вигляді AST-моделі.

Використання абстрактного синтаксичного дерева дозволяє уніфікувати процес обробки математичних виразів незалежно від складності задачі. Такий підхід спрощує інтеграцію нових математичних модулів та забезпечує масштабованість системи при розширенні функціональності.

На початковому етапі розробки система була реалізована у вигляді монолітного веб-застосунку на базі фреймворку Flask. Такий підхід забезпечив швидке прототипування, спрощену інтеграцію модулів NLP, обчислення та візуалізації, а також дозволив перевірити коректність запропонованої математичної моделі. Проте в процесі тестування та розширення функціональності було виявлено ряд обмежень монолітної архітектури, зокрема складність масштабування окремих компонентів, зростання залежностей між модулями та ускладнення підтримки системи при додаванні нових типів задач. У зв’язку з цим було прийнято рішення про перехід до мікросервісної архітектури, що дозволяє розділити функціональність системи на незалежні компоненти та підвищити гнучкість і масштабованість програмного забезпечення.

Запропонована програмна реалізація розглядається як початковий етап побудови системи, який дозволив перевірити ефективність обраних методів обробки природної мови та математичного моделювання. Отримані результати стали основою для подальшого переходу до більш гнучкої мікросервісної архітектури.

Висновок

У роботі запропоновано та обґрунтовано метод інтелектуальної обробки природної мови для систем автоматизації математичних обчислень. Розроблена математична модель перетворення неструктурованих текстових запитів у детерміновані структури (дерева виразів) дозволяє мінімізувати вплив неоднозначності природної мови та забезпечити коректну формалізацію умов задач. Практична реалізація запропонованого методу у складі веб-застосунку з використанням PostgreSQL із підтримкою JSONB забезпечує високу швидкість обробки запитів та зручність візуалізації крокових рішень.Подальші дослідження будуть спрямовані на вдосконалення нейронних алгоритмів розпізнавання сутностей із використанням архітектури Transformer .

Список використаних джерел

1. Обробка природної мови (NLP): основні поняття та застосування. Режим доступу: https://www.sap.com/ukraine/resources/what-is-natural-language-processing
2. VaswaniA., et al. Attention Is AllYou Need // Advances inNeural Information Processing Systems, 2017. P.6000-6.
3. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. – University of California, Irvine, 2000. – Режим доступу:https://roy.gbiv.com/pubs/dissertation/

УДК 004.932:004.5:794

АНАЛІЗ МЕТОДІВ ПЕРЕДАЧІ ДАНИХ ДЛЯ БЕЗКОНТАКТНОГО КЕРУВАННЯ У ІГРОВОМУ СЕРЕДОВИЩІ

Шпінталь М.Я.[1)], Красько М.Г.[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] магістрант

I. Постановка проблеми

У сучасних системах Human-Computer Interaction (HCI) критичним параметром є затримка відгуку (Latency). При використанні Python як основного середовища для обробки комп'ютерного зору та Unity як графічного рушія, розробники стикаються з проблемою «вузького місця» (bottleneck) при міжпроцесній взаємодії. Традиційні методи, такі як запис у проміжні файли або використання високорівневих HTTP-запитів, створюють великі затримки при передачі данних.

II. Мета роботи

Метою роботи є аналіз та розробка високопродуктивного конвеєра передачі даних на основі протоколу UDP для інтеграції системи розпізнавання жестів у середовище Unity. Додатковою метою є забезпечення стабільної роботи системи в реальному часі з мінімальними затримками та високою частотою оновлення кадрів.

III. Обробка жесту та формування інформаційного пакету

Для створення ефективної системи керування відеопотік перетворюється у структурований набір цифрових команд. Процес розпочинається з використання бібліотеки MediaPipe, яка в реальному часі аналізує кадри та будує скелетну модель руки з 21 ключової точки.

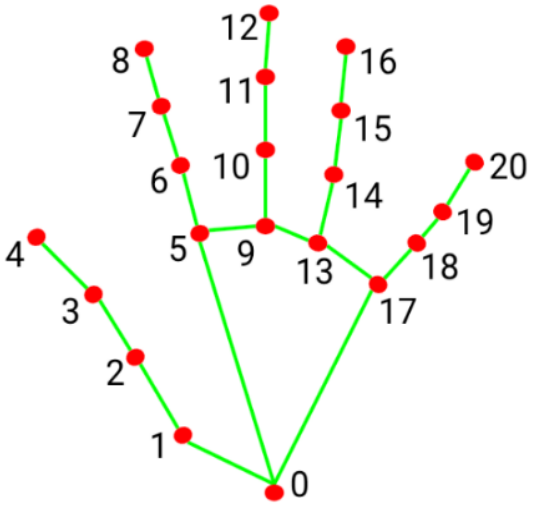

Рисунок 1 – Скелетна модель руки

Це дозволяє системі зосередитися на геометричних взаємозв’язках між суглобами та кінчиками пальців незалежно від освітлення чи фону. Отримані координати використовуються як вхідні дані для класифікатора на основі Random Forest, обраного через високу швидкість обробки та низькі затримки під час інференсу. Після розпізнавання жесту система формує компактний інформаційний пакет, який містить ідентифікатор жесту та нормалізовані координати долоні. Використання простого текстового формату зменшує накладні витрати на серіалізацію даних і забезпечує швидку підготовку інформації до передачі в ігрове середовище.

IV. Мережева взаємодія та асинхронна інтеграція в Unity

Транспортування підготовлених даних до ігрового середовища реалізовано через локальний мережевий стеку з використанням протоколу UDP. Вибір цього протоколу замість стандартного TCP обумовлений його специфікою: UDP дозволяє відправляти пакети без встановлення постійного з'єднання та підтвердження отримання кожного повідомлення. У системах реального часу, таких як відеоігри, актуальність даних про положення руки в конкретний момент є набагато важливішою за гарантовану доставку застарілої інформації. Якщо один пакет загубиться через мережеві коливання,

наступний прийде вже через долі секунди, що дозволяє уникнути ефекту «зависання» ігрового процесу.

Python-модуль працює як сервер, що постійно транслює актуальні координати на локальну адресу, забезпечуючи безперервний потік команд.На стороні Unity розроблено спеціальну архітектуру для прийому та обробки цих повідомлень без шкоди для продуктивності рендерингу. Для цього використовується окремий фоновий потік, який працює паралельно з основним ігровим циклом і безперервно прослуховує вхідний порт. Такий розподіл завдань дозволяє уникнути блокування головного потоку Unity, що зазвичай призводить до падіння частоти кадрів. Отриманий пакет даних миттєво десеріалізується, після чого ідентифікатор жесту та координати потрапляють у безпечну чергу (Thread-safe Queue). Головний цикл гри у кожному кадрі зчитує актуальну інформацію з цієї черги та миттєво трансформує її в ігрові події. Це забезпечує ідеальну синхронізацію між рухами користувача та віртуальною реальністю, створюючи імерсивний досвід керування.

На рисунку 2 зображено архітектуру передачі даних між модулем розпізнавання жестів на Python та ігровим середовищем Unity.

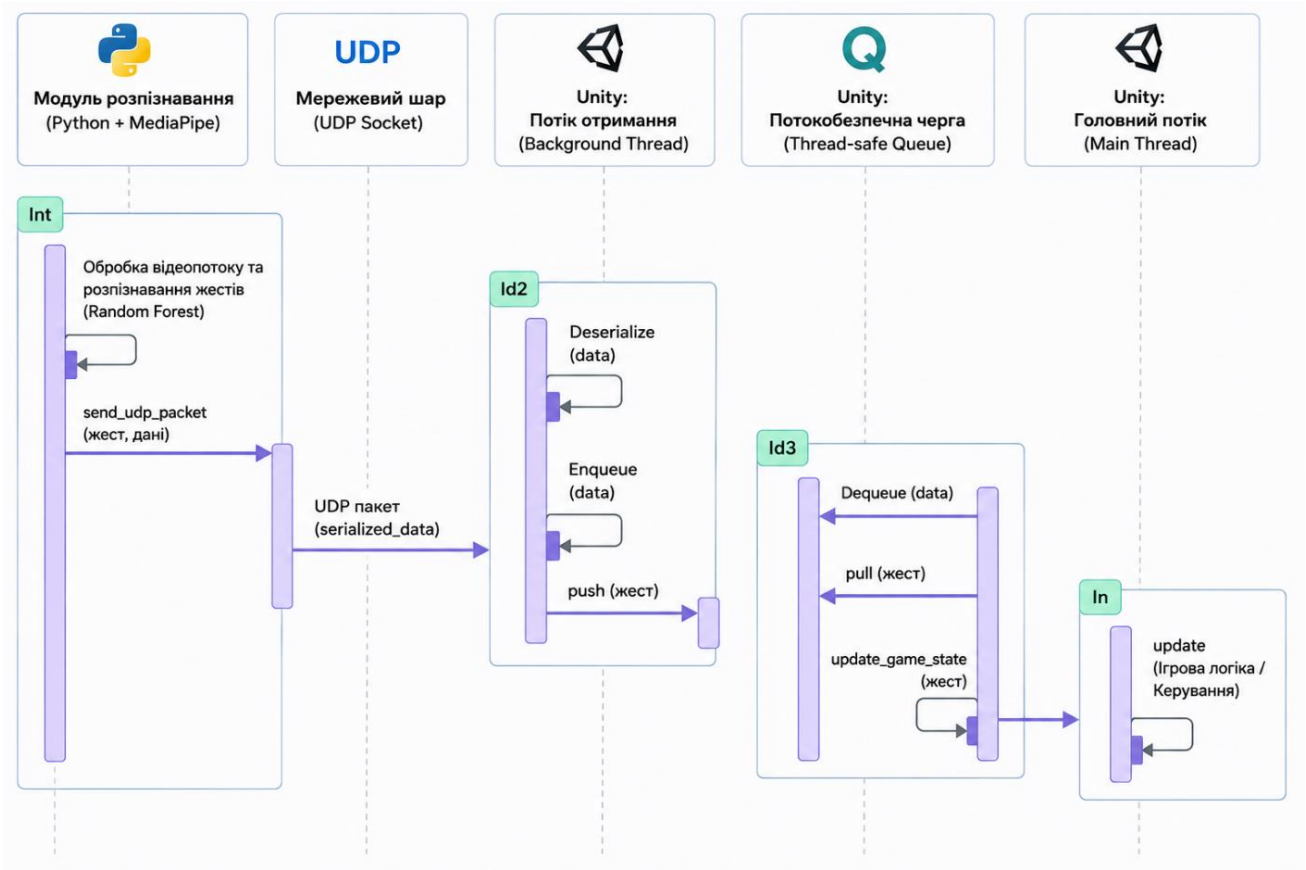

Рисунок 2 – Архітектура передачі даних між модулем розпізнавання жестів на Python та ігровим середовищем Unity.

Висновки

Запропонована архітектура забезпечує ефективне безконтактне керування завдяки швидкій класифікації жестів методом Random Forest і передачі даних через UDP з мінімальною затримкою. Асинхронна обробка даних у Unity ізолює мережеві операції від основного циклу гри, що дозволяє зберігати високу частоту кадрів і плавність роботи.

Список використаних джерел

1. MediaPipe Hands documentation — https://developers.google.com/mediapipe
2. Python Socket Programming documentation — https://docs.python.org/3/library/socket.html
3. Scikit-learn Random Forest documentation — https://scikit-learn.org
4. User Datagram Protocol (UDP) Specification RFC 768 — https://www.rfc-editor.org/rfc/rfc768
5. OpenCV Documentation — https://docs.opencv.org
6. TensorFlow Keras API documentation — https://www.tensorflow.org
7. MediaPipe Framework documentation — https://ai.google.dev/edge/mediapipe
8. Computer Vision: Algorithms and Applications — https://szeliski.org/Book/

УДК 004.62

МЕТОДИ НЕІНВАЗИВНОЇ ІТЕРАТИВНОЇ АКТУАЛІЗАЦІЇ ДАНИХ ПРИ ПЕРЕХОДІ МІЖ ГЕТЕРОГЕННИМИ ПЛАТФОРМАМИ ЕЛЕКТРОННОЇ КОМЕРЦІЇ

Сімак А.Ю.[1)], Белзюк Р.І.[2)]
Західноукраїнський національний університет
[1)] д. філ., викладач; [2)] магістрант

I. Постановка проблеми

Міграція високонавантажених систем електронної комерції відбувається за умов безперервного потоку змін у вихідній системі: створюються нові замовлення, оновлюються залишки, коригуються дані клієнтів. Класична методологія метелика [1] фіксує ці зміни через компонент Data Access Allocator, який тимчасово перенаправляє операції запису до окремих сховищ (TempStore). Такий підхід потребує створення тимчасових баз даних і модифікації логіки доступу до вихідної системи, що порушує принцип неінвазивності. Окрім того, оригінальна методологія не передбачає обробки операцій видалення: якщо запис видалено у вихідній системі після початку міграції, ця зміна не фіксується і не поширюється на цільову систему. У системах електронної комерції, де дані підлягають вимогам фіскального та бухгалтерського законодавства, зазначені обмеження є неприйнятними.

II. Мета роботи

Метою є розробка неінвазивного методу ітеративної актуалізації даних як одного з ключових компонентів методології метелика-монарха — розширення класичної методології метелика для міграції високонавантажених систем електронної комерції. Метод має формалізувати структуру журналу змін, забезпечити коректну обробку всіх типів операцій, включаючи видалення, та зберігати вихідну систему у незмінному стані протягом усього процесу перенесення.

III. Основна частина

Для усунення обмежень DAA запропоновано замінити його на механізм Change Data Capture (CDC), що фіксує зміни у вихідній системі без втручання у її програмний код і структуру даних [2]. CDC розглядається як один з основних патернів перехоплення подій у сучасних архітектурах з розподіленим зберіганням даних [3]. Розрізняють три варіанти реалізації CDC, які різняться за рівнем інвазивності та вимогами до СКБД [2].

Log-based CDC читає внутрішній журнал транзакцій СКБД: у MySQL — двійковий журнал (binary log), у PostgreSQL — журнал попереднього запису (Write-Ahead Log) із логічною реплікацією [4]. Зазначений варіант є повністю неінвазивним, оскільки читання журналу є штатною операцією СКБД, проте потребує конфігурування логічної реплікації. Trigger-based CDC фіксує зміни через тригери на таблицях і є частково інвазивним на рівні схеми. Query-based CDC обчислює зміни через періодичне зчитування вихідної бази з порівнянням контрольних хеш-сум кожного запису: нові та змінені записи визначаються за розбіжністю хешів, видалені — за відсутністю ідентифікаторів у поточній вибірці. Порівняння підходів DAA і CDC наведено на рисунку 1.

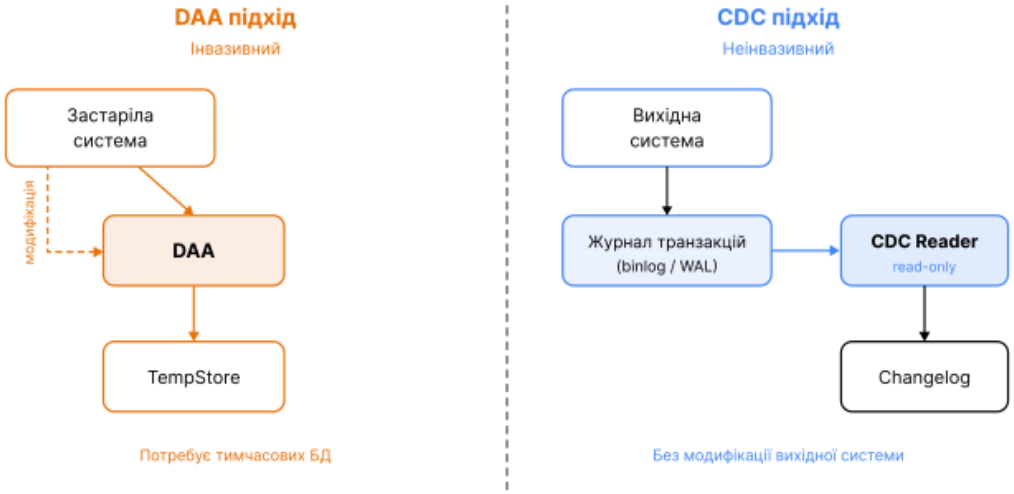

Рисунок 1 – Порівняння підходів DAA і CDC до журналювання змін у вихідній системі

На основі даних, отриманих через CDC, формується журнал змін — впорядкована послідовність записів, кожен з яких фіксує одну атомарну операцію у вихідній системі. Структура запису визначається кортежем за формулою (1):

$$c = (e_i, id, op, t, d_{old}, d_{new}), \tag{1}$$

де e_i – тип сутності, id – ідентифікатор запису, $op \in \{INSERT, UPDATE, DELETE\}$ – тип операції, t – часова мітка, d_{old} – стан запису до операції, d_{new} – стан запису після операції. Журнал $C(t_\alpha, t_b)$ за інтервал часу є впорядкованою за часовою міткою множиною таких кортежів.

Ітеративна актуалізація формалізується як послідовність операцій застосування пакетів змін. На ітерації I_k обробляється пакет $C(t_{k-1}, t_k)$, кожен запис якого трансформується відповідно до відображень f_i і застосовується до цільової системи за формулою (2):

$$D(t_k) = D(t_{k-1}) \oplus T(C(t_{k-1}, t_k)), \tag{2}$$

де T – функція трансформації, що перетворює записи журналу змін із формату вихідної системи у формат цільової; $D(t_{k-1})$ – стан цільової бази в момент t_{k-1}; $\oplus$ – операція застосування змін до стану бази.

Перевагою запропонованої моделі є коректна обробка операцій видалення, відсутня у класичній методології метелика [1]. Операція DELETE фіксується у журналі зі збереженням стану запису до видалення у полі d_{old}. За наявності відповідного перенесеного запису у цільовій системі він видаляється. За відсутності операція видалення зберігається у журналі для подальшої обробки після перенесення відповідного запису. Таким чином журнал змін виконує функцію журналювання намірів (intent logging), що забезпечує повноту відтворення стану вихідної системи у цільовій незалежно від моменту створення запису. Модель журналу операцій з підтримкою відтворення та відкату є ключовим елементом формальних методів автоматизованої міграції даних [5].

IV. Тестування та результати

Розроблений метод реалізовано у програмній системі міграції між платформами CS-Cart і Shopware з використанням query-based варіанту CDC. Експериментальну перевірку проведено на реальній системі електронної комерції обсягом 40–50 ГБ, що містить близько 4 мільйонів замовлень, 100 тисяч товарів та 500 тисяч клієнтів. Протягом півтора року розробки цільової системи міграцію запускали 60–70 разів, що підтвердило стабільність та ідемпотентність процесу. Тривалість ітерацій скоротилася від 4 годин на нульовій ітерації до менш ніж однієї хвилини на фінальних, а час простою при переключенні становив 1–3 хвилини — приблизно у 500 разів менше за підхід Cold Turkey для даних такого обсягу.

Висновок

Таким чином, заміна інвазивного компонента DAA на механізм Change Data Capture усуває потребу у модифікації вихідної системи і забезпечує коректну обробку всіх типів операцій, включаючи видалення, через журналювання намірів у кортежі $(e_i, id, op, t, d_{old}, d_{new})$. Формалізація ітеративного застосування журналу змін забезпечує збіжність процесу до прийнятного часу фінального переключення. Експериментальне підтвердження на наборі даних реальної високонавантаженої системи електронної комерції показало скорочення часу простою до 1–3 хвилин, що у сотні разів менше за підходи разового перенесення.

Список використаних джерел

1. Wu, B., Lawless, D., Bisbal, J., Richardson, R., Grimson, J., Wade, V., & O'Sullivan, D. (1997). The Butterfly Methodology: A Gateway-free Approach for Migrating Legacy Information Systems. Proceedings of the 3rd IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'97), Como, Italy, 200–205. IEEE Computer Society.
2. Kleppmann, M. (2017). Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol: O'Reilly Media. 614 p. ISBN 978-1449373320.
3. Newman, S. (2021). Building Microservices: Designing Fine-Grained Systems (2nd ed.). Sebastopol: O'Reilly Media. 612 p. ISBN 978-1492034025.
4. Debezium Reference Documentation: Distributed Platform for Change Data Capture. Red Hat, Inc. URL: https://debezium.io/documentation/reference/stable/
5. Патлань, Є., Білоус, І. (2025). Метод автоматизованої міграції даних між варіантами сховищ у системах з багатоваріантною персистентністю. Інформаційні технології та суспільство, 4(19), 129–139.

УДК 004.4

ВЕБ-СИСТЕМА КОМПЛЕКСНОГО МОНІТОРИНГУ ЗДОРОВ'Я

Стасів І.С.[1)], Порплиця Н.П.[2)], Монастирська Д.Е.[3)]
Західноукраїнський національний університет
[1-2)] к.т.н., доцент; [3)] бакалавр

І Постановка проблеми

В сучасному світі дедалі більше людей ведуть малорухливий спосіб життя, що негативно позначається як на фізичному стані організму, так і на психологічному самопочутті. Водночас стрімкий темп розвитку технологій та активна цифровізація щоденних процесів створюють можливості для задоволення потреби в ефективних інструментах самоконтролю здоров'я. Під впливом цих тенденцій зростає і попит на цифрові рішення, які допомагають відстежувати фізичну активність, формувати корисні звички та підтримувати психоемоційний баланс.

При цьому, більшість мобільних та веб-застосунків пропонують фрагментарний підхід, фокусуючись лише на одному аспекті: фізичних вправах, дієтології чи психологічному стані. Часто вони мають обмежені можливості безкоштовного використання або залежність від додаткових пристроїв. Такий підхід не дозволяє користувачу побачити цілісну картину взаємозв'язку між різними показниками життєдіяльності.

ІІ Мета роботи

Метою даної роботи є розробка програмного засобу – інтегрованої веб-платформи «FitMind», що представляє собою систему підтримки здоров'я, яка надає доступ до програм тренувань та забезпечує автоматизацію процесів моніторингу фізичної активності, ментальних практик і нутриціологічного контролю в єдиному цифровому середовищі. Основною метою такого програмного продукту є допомога користувачам у формуванні здорових звичок, покращенні фізичної форми, зниженні ваги, підвищенні рівня фізичної активності та підтримка психологічного балансу.

ІІІ Особливості реалізації веб-системи комплексного моніторингу здоров'я

Для реалізації застосунку використано мову програмування TypeScript, бібліотеку React, середовище Node.js із фреймворком Express, а також реляційну базу даних SQLite для надійного зберігання інформації [1-3]. Розробка здійснювалася у середовищі VS Code з використанням інструменту збірки Vite. Для обміну даними між клієнтом та сервером реалізовано REST API з авторизацією на основі JWT-токенів.

Застосунок побудовано на основі багаторівневої архітектури. На нижньому рівні знаходиться база даних SQLite , у якій зберігаються таблиці: користувачі (users), дані прогресу (app_data) та бібліотека контенту (store). Логіку взаємодії з базою даних відокремлено в окремі модулі, що спрощує подальший розвиток та розширення системи. На стороні фронтенду використано React Context для управління авторизацією та налаштуваннями - це забезпечує швидку роботу інтерфейсу та дозволяє уникати зайвих навантажень при оновленні сторінок.

В інтерфейсі застосовано сучасні компоненти: Tailwind CSS для адаптивної верстки, Framer Motion для плавних анімацій переходів та Recharts для побудови інтерактивних графіків ваги та активності. Всі списки тренувань та сесій медитацій відображаються динамічно, отримуючи дані з API, що забезпечує швидкодію без перезавантаження сторінок.

Основні функції застосунку:

- Персоналізований дашборд: відображення ключових метрик (BMI, стрік активності, ранг та XP). Користувач отримує наочну інформацію про свій поточний стан та досягнення.
- Фітнес-модуль: бібліотека вправ. Система автоматично перевіряє виконання норми тренувань та нараховує досвід (XP) за завершені сесії.
- Ментальне здоров'я: розділ із медитаціями та дихальними вправами, що супроводжуються аудіофайлами.
- Журнал харчування: окремий блок для фіксації прийомів їжі та контролю калорійності. Користувач може відстежувати відсоток виконання добової норми харчування.
- Адмін-панель: реалізовано повний цикл CRUD для керування бібліотекою вправ та сесій, а також інструменти модерації користувачів (пошук, зміна ролей, бан) та візуалізація загальної демографії.
- Гейміфікація: При досягненні певних показників користувач переходить на новий рівень, що

відображається в його профілі, також впроваджено механізм «стріків» (серій днів) та рівнів, що стимулює регулярне використання продукту.

Головна сторінка (Dashboard) є центром взаємодії користувача з програмою. Вона відображає інтерактивні картки з індексом маси тіла (BMI), динамікою ваги та поточною серією активності (streak). Користувач може бачити свій прогрес у вигляді кілець активності та рівень досвіду (XP) (див.рис.1).

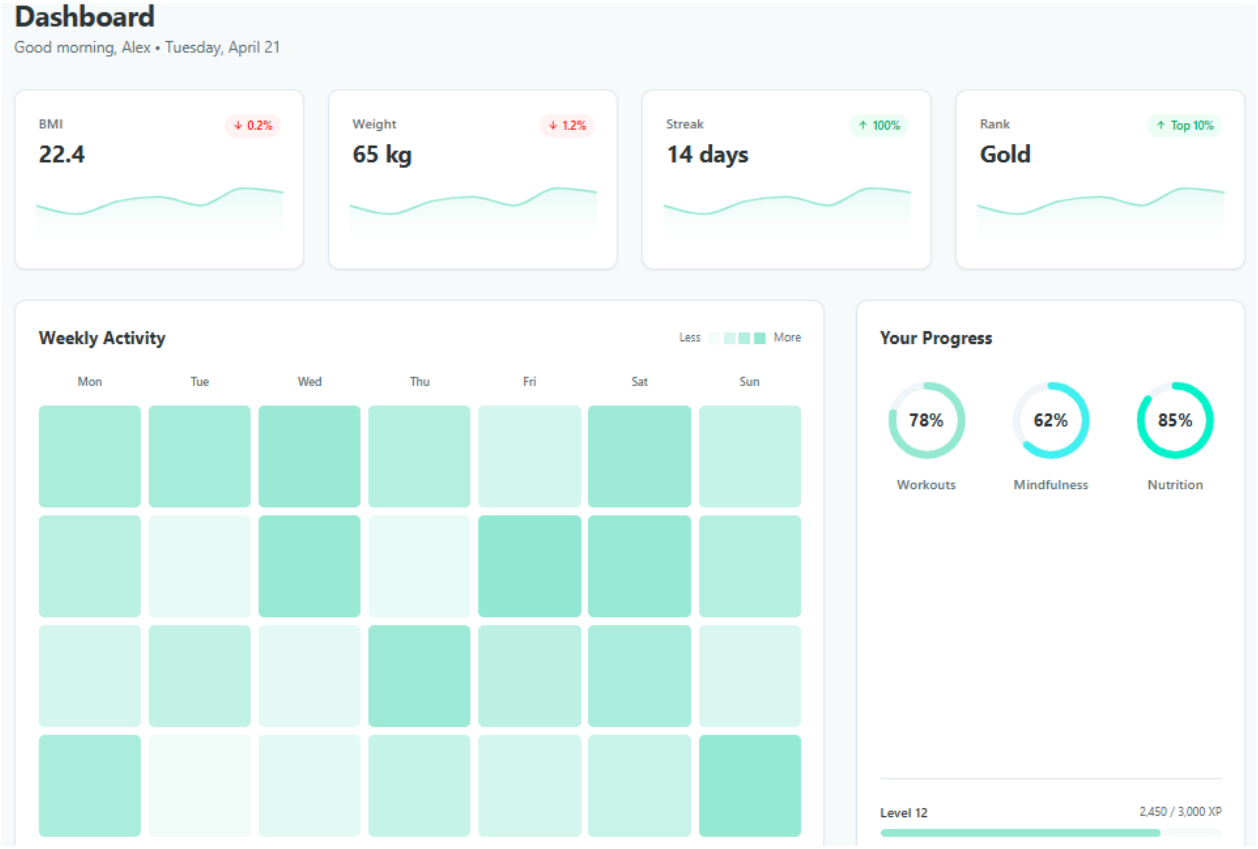

Рисунок 1 – Графічний інтерфейс сторінки прогресу користувача

Основними перевагами реалізованого рішення є комплексність, висока швидкість роботи завдяки Vite та React, а також зручність адміністрування. Використання SQLite дозволяє розгортати систему на бюджетних серверах, забезпечуючи при цьому цілісність даних. Гнучкість розробленого рішення дозволяє легко вносити зміни або додавати новий функціонал в майбутньому завдяки модульній архітектурі.

Висновок

Створена інтегрована веб-платформа «FitMind» представляє собою систему підтримки здоров'я, що надає доступ до програм тренувань та забезпечує автоматизацію процесів моніторингу фізичної активності, ментальних практик і нутриціологічного контролю в єдиному цифровому середовищі. Програмний продукт є інструментом для цілісного управління здоров'ям і завдяки комплексному підходу має сприяти формуванню довготривалих здорових звичок та використовуватися як ефективний інструмент для підтримки активного та збалансованого способу життя. Його модульність, наявність зручної адмін-панелі та інтегрована система прогресів дозволяють ефективно організувати режим дня, зменшити когнітивне навантаження та знизити ймовірність відмови користувача від вже засвоєних здорових звичок. Превагою цієї інтегрованої платформи є те, що у перспективі веб-система може бути доповнена алгоритмами рекомендацій на основі AI та синхронізацією з мобільними гаджетами.

Список використаних джерел:

1. React Documentation [Електронний ресурс]. – Режим доступу: https://react.dev/
2. TypeScript Handbook [Електронний ресурс]. – Режим доступу: https:// www.typescriptlang.org/docs/
3. Доценко С. І. Організація та системи керування базами даних: Навч. посібник. – Харків: УкрДУЗТ, 2023. – 117 с.
4. Харів Н. О. Бази даних та інформаційні системи: навчальний посібник / Н. О. Харів. – Рівне : НУВГП, 2018. – 127 с.

УДК 004.4:004.056:004.8

РОЗРОБКА АНАЛІТИЧНОЇ ПРОГРАМНОЇ СИСТЕМИ КОНТРОЛЮ ВЗАЄМОДІЇ ПРАЦІВНИКІВ ІЗ ШІ-СЕРВІСАМИ

Сімак А.Ю.[1)], **Пасічник В.П.**[2)]
[1-2)] Західноукраїнський національний університет
[1)] д. філ., викладач; [2)] магістрант

I. Постановка проблеми

Активне використання ШІ-сервісів (сервісів штучного інтелекту) у компаніях створює нові ризики витоку конфіденційної інформації, оскільки працівники можуть передавати до зовнішніх систем персональні дані, фрагменти внутрішньої документації, вихідний код, фінансові або технічні відомості у складі текстових запитів. У більшості випадків така взаємодія відбувається через браузер, тому саме браузерне середовище є точкою виникнення ризику. Існуючі засоби захисту лише частково розв'язують цю проблему, оскільки не забезпечують достатнього контролю користувацьких запитів до ШІ-сервісів у режимі реального часу.

II. Мета роботи

Метою дослідження є проектування та розробка цілісної аналітичної програмної системи для комплексного моніторингу та контролю взаємодії працівників із зовнішніми сервісами штучного інтелекту.

Система спрямована на автоматизоване виявлення ризиків витоку конфіденційної інформації у режимі реального часу, забезпечення централізованого збору та ретроспективного аналізу журналів подій, а також надання адміністраторам безпеки інтерактивного інструментарію для візуалізації інцидентів та прийняття обґрунтованих управлінських рішень у корпоративному середовищі.

III. Метод вирішення задачі

Для вирішення поставленої задачі запропоновано побудову браузерно-орієнтованої програмної системи, що поєднує клієнтську та серверну частини. На стороні користувача система реалізується у вигляді браузерного розширення, яке виявляє взаємодію з ШІ-сервісами, аналізує запити та передає результати до серверної підсистеми.

Серверна частина забезпечує централізовану обробку подій, збереження результатів аналізу, підтримку правил безпеки та взаємодію з адміністративним інтерфейсом. Такий підхід дозволяє поєднати локальний контроль у точці виникнення ризику з централізованим аудитом і керуванням системою на рівні компанії [1].

IV. Особливості програмної реалізації

Основою програмної реалізації є модульна архітектура, що забезпечує гнучкість, масштабованість та можливість адаптації системи до різних ШІ-платформ. Центральним компонентом є браузерне розширення, яке виконує кілька функцій: виявляє сторінки або вебзапити, пов'язані з ШІ-сервісами, перехоплює або аналізує текстові промпти, виконує початкову перевірку вмісту та ініціює подальшу передачу даних на сервер. Це дає змогу контролювати взаємодію працівника з ШІ ще до моменту завершення операції або одразу після її виконання.

Окремим компонентом системи є сховище даних, у якому фіксуються всі сутності, необхідні для подальшого аналізу: користувачі, ШІ-сервіси, текстові запити, результати перевірки, рівні небезпеки, події журналювання та дії системи у відповідь. Така організація дозволяє забезпечити простежуваність інцидентів, накопичення статистики та формування аналітики щодо використання ШІ-інструментів у межах компанії [3].

Важливе значення має адміністративний інтерфейс, який перетворює технічне рішення на повноцінний корпоративний програмний продукт. За допомогою адміністративної панелі уповноважені користувачі можуть переглядати історію подій, аналізувати підозрілі запити, відстежувати частоту інцидентів, керувати конфігурацією системи та контролювати дотримання внутрішніх політик безпеки. Наявність такого інтерфейсу робить систему придатною не лише для автоматичного спрацювання, а й для подальшого управління безпекою на організаційному рівні.

На рисунку 1 наведено приклад адміністративного інтерфейсу системи, що забезпечує перегляд журналу подій, аналіз інцидентів та централізований контроль використання ШІ-сервісів у компанії.

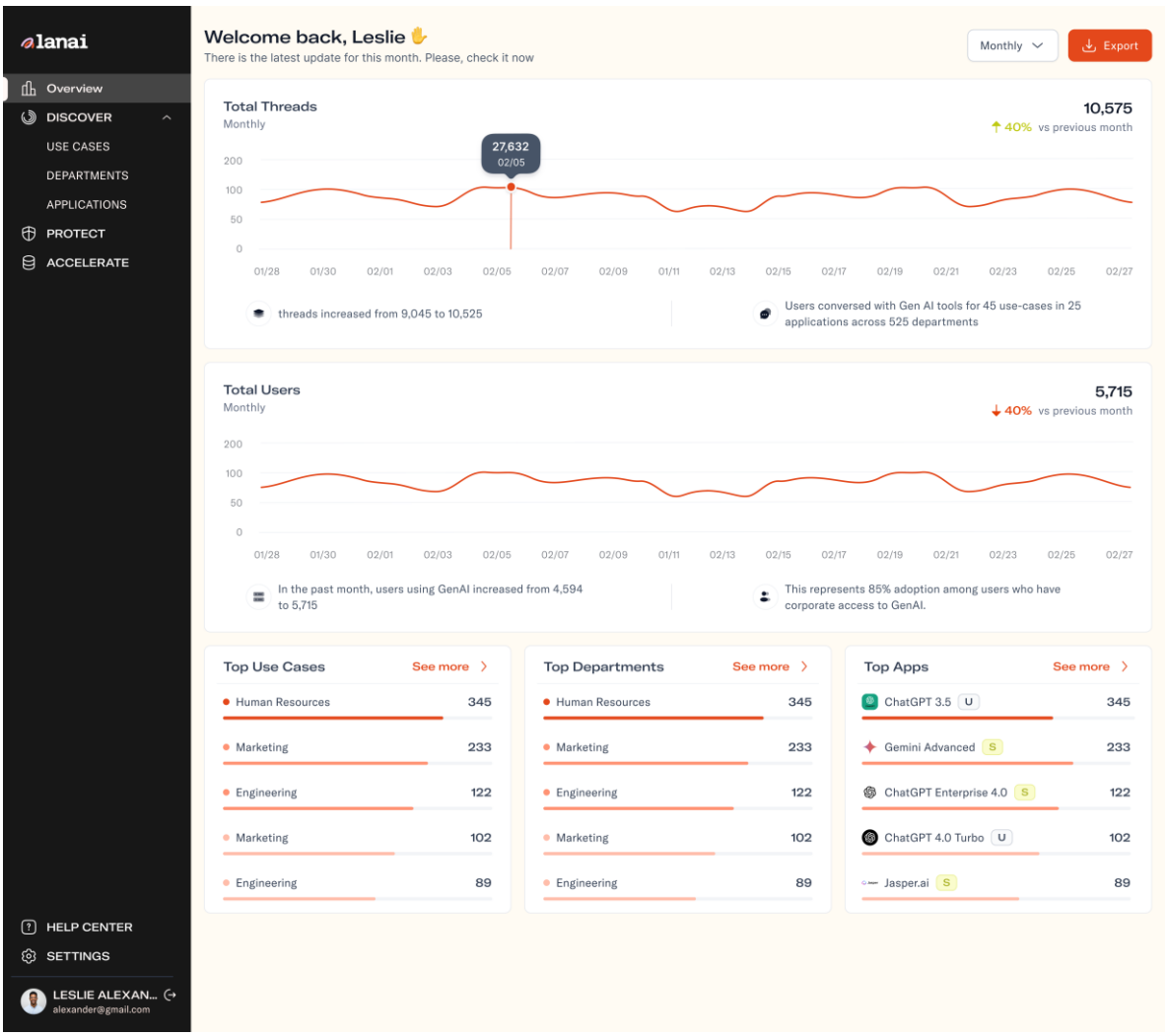

Рисунок 1 – Адміністративний інтерфейс системи контролю взаємодії працівників із ШІ-сервісами

В контексті проєктування, важливою перевагою є можливість розширення системи новими ШІ-платформами, правилами виявлення ризику, додатковими модулями аналітики та інтеграціями з іншими корпоративними сервісами. Модульна побудова дозволяє поступово розвивати рішення без суттєвого перепроєктування всієї системи [2]. Таким чином, запропонована архітектура орієнтована не лише на розв'язання поточної задачі моніторингу, а й на подальший розвиток системи як корпоративного продукту для безпечного використання штучного інтелекту.

Висновок

У ході роботи було запропоновано та реалізовано концепцію аналітичної програмної системи, що інтегрує методи локального перехоплення даних на рівні браузера з потужною серверною підсистемою журналювання. На відміну від існуючих рішень, розроблена система забезпечує не лише фіксацію фактів взаємодії з ШІ-сервісами, а й надає аналітику щодо типів переданих даних, частоти інцидентів та рівнів загрози через спеціалізований адміністративний інтерфейс.

Практична значущість дослідження полягає у створенні масштабованої архітектури, яка дозволяє компаніям безпечно впроваджувати технології штучного інтелекту, зберігаючи контроль над збереженням комерційної таємниці та персональних даних. Отримані результати можуть слугувати технологічним фундаментом для побудови корпоративних систем аудиту ШІ-активності, а модульна побудова системи дозволяє легко адаптувати її до нових моделей ШІ та специфічних політик безпеки різних галузей.

Список використаних джерел

1. Sommerville I. Software Engineering. 10th ed. Pearson, 2015.
2. Bass L., Clements P., Kazman R. Software Architecture in Practice. 4th ed. Addison-Wesley, 2021.
3. Томашевський Б. П., Сергієнко Р. В. Сучасні методи захисту інформації в корпоративних системах. Ukrainian Information Security Journal. 2020.

УДК 004.8

АРХІТЕКТУРНІ ОСОБЛИВОСТІ СИСТЕМИ ДЛЯ АДАПТИВНОЇ ПІДГОТОВКИ ДО ТЕХНІЧНИХ СПІВБЕСІД ІЗ ВИКОРИСТАННЯМ ШІ

Підгородецький А.А.[1)], Манжула В.І.[2)]
Західноукраїнський національний університет
[1)]магістрант; [2)]д.т.н., професор

Постановка проблеми

Розроблення системи для адаптивної підготовки до технічних співбесід, що поєднує виконання користувацького коду, обробку природної мови та формування інтелектуальних рекомендацій, зумовлює необхідність побудови складної програмної архітектури з різнорідними вимогами до її компонентів. Інтеграція середовища виконання коду, AI-сервісів та серверної частини ускладнює забезпечення масштабованості, безпеки та відмовостійкості, зокрема через потребу ізоляції процесів і ефективної міжсервісної взаємодії. У таких умовах монолітні підходи є недостатньо ефективними, що зумовлює необхідність розроблення архітектури з розподілом функцій, підтримкою незалежного масштабування та узгодженої взаємодії компонентів.

Мета дослідження

Метою роботи є розроблення архітектури системи для адаптивної підготовки до технічних співбесід, що забезпечує ефективну інтеграцію функціональних компонентів, масштабованість, відмовостійкість та безпечне виконання користувацького програмного коду. Для досягнення поставленої мети передбачається формування структурної моделі системи з виділенням основних компонентів, обґрунтування доцільності використання мікросервісного підходу, організація ізольованого середовища виконання коду, а також визначення механізмів взаємодії між серверною частиною та інтелектуальними модулями. Результатом роботи є архітектурне рішення, що забезпечує узгоджену роботу компонентів системи та створює основу для її подальшого масштабування і розвитку.

Структура системи

Розроблене рішення передбачає розподілену мікросервісну архітектуру, що забезпечує високу надійність, ізоляцію процесів та можливість незалежного масштабування окремих вузлів. Така структура дозволяє поєднувати стабільність корпоративного бекенду з гнучкістю інтелектуальних сервісів аналізу даних. Клієнтська частина (Frontend), яка реалізована на базі бібліотеки React, виступає єдиною точкою взаємодії користувача з платформою, забезпечуючи динамічний інтерфейс. Взаємодія з серверною частиною відбувається через протокол REST API [1].

Центральний бекенд (Core API) виконаний на платформі .NET Web API з дотриманням принципів “чистої архітектури”. Цей компонент керує потоками даних, забезпечує авторизацію користувачів та обробку основної бізнес-логіки. Він є координатором між базою даних, середовищем виконання коду та інтелектуальними модулями.Середовище виконання коду (Docker пісочниця) забезпечує безпечний запуск та тестування програмних рішень користувачів. Завдяки контейнеризації через Docker, кожен запуск коду відбувається в ізольованому оточенні, що гарантує захист від потенційно шкідливих дій та дозволяє контролювати ресурси системи [2].Інтелектуальний блок (AI Microservice) складається з окремого Pythonмікросервісу, що спеціалізується на обробці природної мови та аналізі даних. До його складу входять:

- Модуль генерації фідбеку, який використовує мовну модель Llama 4 для створення розгорнутих пояснень до рішень та теоретичних відповідей [3];
- Механізм векторного пошуку, що дозволяє здійснювати змістовий пошук релевантних матеріалів та завдань, забезпечуючи високу точність персоналізованих рекомендацій;
- Система управління даними базується на реляційній СУБД PostgreSQL. Вона відповідає за зберігання структурних даних системи, результатів виконання SQL-запитів від основного бекенду та векторних представлень для роботи інтелектуального пошуку [4].

Інфраструктура розгортання системи підтримує контейнеризацію за допомогою Docker та оркестрацію через Kubernetes, що дозволяє гнучко масштабувати окремі сервіси й забезпечувати високу доступність. Щоб спростити аналіз складових системи, на рисунку 1 представлено діаграму компонентів.

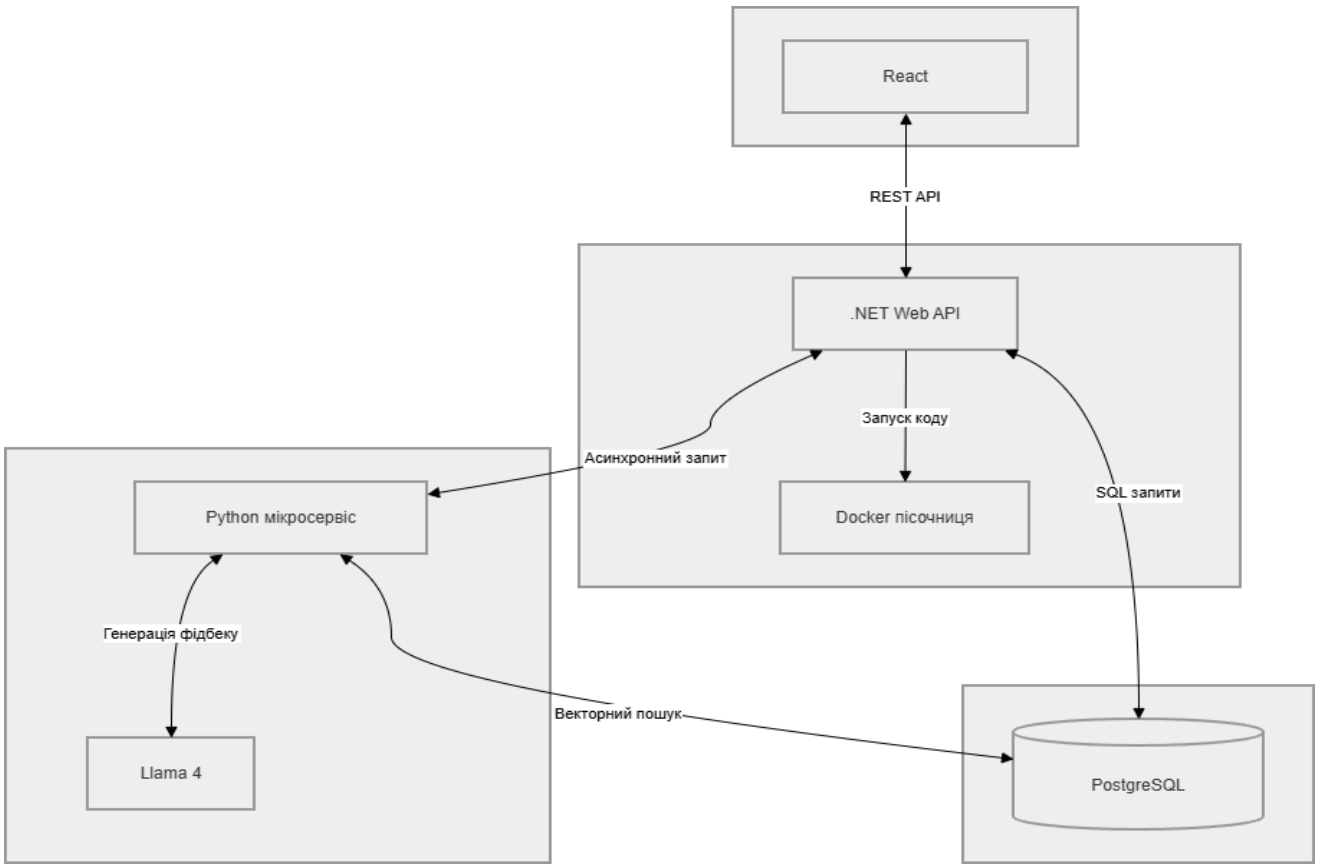

Рисунок 1 – Діаграма компонентів системи

Користувач працює через веб-інтерфейс на базі React, взаємодіючи із центральним сервером .NET Web API за допомогою REST-запитів. У разі потреби сервер ініціює запуск програмного коду в ізольованій Docker-пісочниці або звертається до Python-мікросервісу для проведення інтелектуальних обчислень. Python-мікросервіс використовує модель Llama 4 для генерації фідбеку та здійснює векторний пошук у базі PostgreSQL для підбору релевантного контенту.

Результати обробки повертаються користувачу у вигляді розгорнутих текстових пояснень, оцінок та персоналізованих рекомендацій, що підвищують прозорість та ефективність процесу підготовки до співбесід. Взаємодія між основними сервісами та базою даних PostgreSQL забезпечує надійне збереження історії успішності та динамічне оновлення профілю компетенцій кандидата.

Висновок

У роботі розроблено архітектуру інтелектуальної системи для адаптивної підготовки до технічних співбесід, що базується на мікросервісному підході та передбачає розподіл функціональних обов'язків між окремими компонентами. Запропоноване архітектурне рішення забезпечує інтеграцію серверної частини на платформі .NET, інтелектуального модуля на базі Python та мовної моделі, а також підсистеми виконання користувацького коду.Використання контейнеризації дозволяє реалізувати ізольоване та безпечне середовище виконання програм, тоді як застосування механізмів векторного пошуку забезпечує ефективну обробку даних та формування релевантних рекомендацій. Обрана архітектура сприяє підвищенню масштабованості, відмовостійкості та гнучкості системи. Отримані результати формують основу для подальшого розвитку системи, зокрема розширення функціональності та вдосконалення інтелектуальних механізмів обробки даних.

Список використаних джерел

1. ReactJS. A JavaScript library fo rbuilding user interfaces [Електронний ресурс]. – Режим доступу: https://react.dev
2. Docker. Empowering app development for developers [Електронний ресурс] – Режим доступу: URL: https://www.docker.com.
3. Llama 4 [Електронний ресурс] – Режим доступу: URL: https://www.llama.com
4. Postgresql [Електронний ресурс] – Режим доступу: URL: https://www.postgresql.org/

УДК 004.41

РОЗРОБКА АЛГОРИТМІЧНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЧНОГО СПИСАННЯ СИРОВИНИ В ПРОЦЕСІ УПРАВЛІННЯ ВИРОБНИЦТВОМ У ПЕКАРНІ

Чекменьова Д.Д.[1)], **Стасів І.С.**[2)]
Західноукраїнськийнаціональнийуніверситет
[1)]бакалавр; [2)]к.т.н, доцент

І. Постановка проблеми

Хлібопекарська та кондитерська галузь є одними з найбільш динамічних сфер харчової промисловості, що вимагають високої точності на кожному етапі виробничого циклу. Процес управління сучасною пекарнею супроводжується низкою специфічних викликів: коротким терміном зберігання сировини (дріжджі, молочні продукти, фрукти), необхідністю суворого дотримання складних багаторівневих технологічних карт (рецептур) та щоденним динамічним плануванням виробничих змін на основі замовлень.

Більшість малих та середніх підприємств (МСП) у цій галузі досі покладаються на фрагментовані системи обліку, такі як електронні таблиці MS Excel або навіть паперові журнали. Такий підхід має критичні недоліки: високий ризик математичних помилок під час ручного масштабування рецептур, розсинхронізація даних між складом і виробничим цехом, а також втрата актуального контролю над собівартістю продукції в умовах нестабільних цін на сировину.

З іншого боку, ринок пропонує великі ERP-системи, які є фінансово недоступними та занадто складними для впровадження у форматі середньої пекарні. В той самий час, вже наявні популярні POS-системи для закладів харчування орієнтовані насамперед на продаж та касове обслуговування (Front-office), залишаючи поза увагою складний виробничий процес: відкладене випікання, управління заквасками, температурні режими та втрати ваги (упікання). Таким чином, виникає гостра науково-практична проблема — відсутність легкої, адаптивної та вузькоспеціалізованої веб-системи управління, здатної покрити повний виробничий цикл.

ІІ. Мета роботи

Метою роботи є дослідження бізнес-процесів пекарні та розробка архітектурних і логічних основ для створення сучасного веб-орієнтованого застосунку. Кінцевою метою є усунення комунікаційних розривів між підрозділами, автоматизація алгоритмів транзакційного списання сировини на основі виконаних завдань, мінімізація впливу людського фактору та створення ергономічного інтерфейсу для персоналу, який працює у складних цехових умовах.

ІІІ. Обґрунтування отриманих результатів та архітектура рішення

Ключовою проблемою наявного документообігу в пекарнях є асинхронність між фактом випікання та фізичним списанням сировини зі складу, що призводить до "уявних залишків" (розбіжності між даними обліку та реальністю)[1-3].

Для формалізації процесу управління виробництвом розроблено концептуальну архітектуру спеціалізованої інформаційної системи. Враховуючи вимоги до надійності та необхідність роботи у цехових умовах, обрано клієнт-серверну (N-Tier) архітектуру. Взаємодія між клієнтом та сервером реалізується за допомогою REST API, що дозволяє мінімізувати навантаження на мережу за рахунок обміну даними у форматі JSON.

В якості вирішення цієї проблеми запропоновано метод автоматизованого транзакційного списання, що базується на моделі "кінцевого автомата" (StateMachine).Згідно з цією моделлю, виробниче завдання не може хаотично змінювати свій стан; воно проходить формалізований життєвий цикл. У Таблиці 1 наведено опис базових станів та їх вплив на базу даних.

Такий підхід дозволяє автоматизувати процес моніторингу залишків. Система аналізує вхідні дані про завершене завдання, обчислює необхідну кількість сировини (шляхом множення нормативу з технологічної карти на обсяг випеченої партії) та порівнює його з поточними складськими запасами.

Якщо після списання залишок певної сировини опускається нижче встановленого критичного мінімуму, система автоматично генерує попередження для відділу постачання щодо необхідності нової закупівлі.

Таблиця 1

Базові стани життєвого циклу виробничого завдання

Назва статусу	**Опис процесу**	**Системна дія та вплив на базу даних**
Очікує (Pending)	Замовлення сформовано менеджером та очікує на виконання.	Валідація наявності технологічної карти. Блокування обсягів сировини не відбувається.
В роботі (InProgress)	Пекар розпочав заміс та виготовлення продукції.	Захист від паралельного взяття завдання іншим працівником. Фіксація часу початку циклу.
Виконано (Completed)	Пекар підтвердив факт виходу готової продукції з печі.	Критична транзакція: розрахунок витрат сировини за тех. картою, перевірка залишків та фінальне списання (Commit).
Скасовано (Canceled)	Завдання відмінено через брак часу або відсутність сировини.	Зняття завдання з цехового планшета. Жодних рухів товарних запасів у БД не відбувається.

Якщо розраховане значення витрат перевищує наявні запаси (система намагається записати від'ємне значення), спрацьовує правило ACID-транзакції реляційної бази даних – відбувається автоматичне повернення, транзакція блокується з поверненням статусу Alert для менеджера виробництва. Алгоритм цього процесу візуалізовано на рисунку 1.

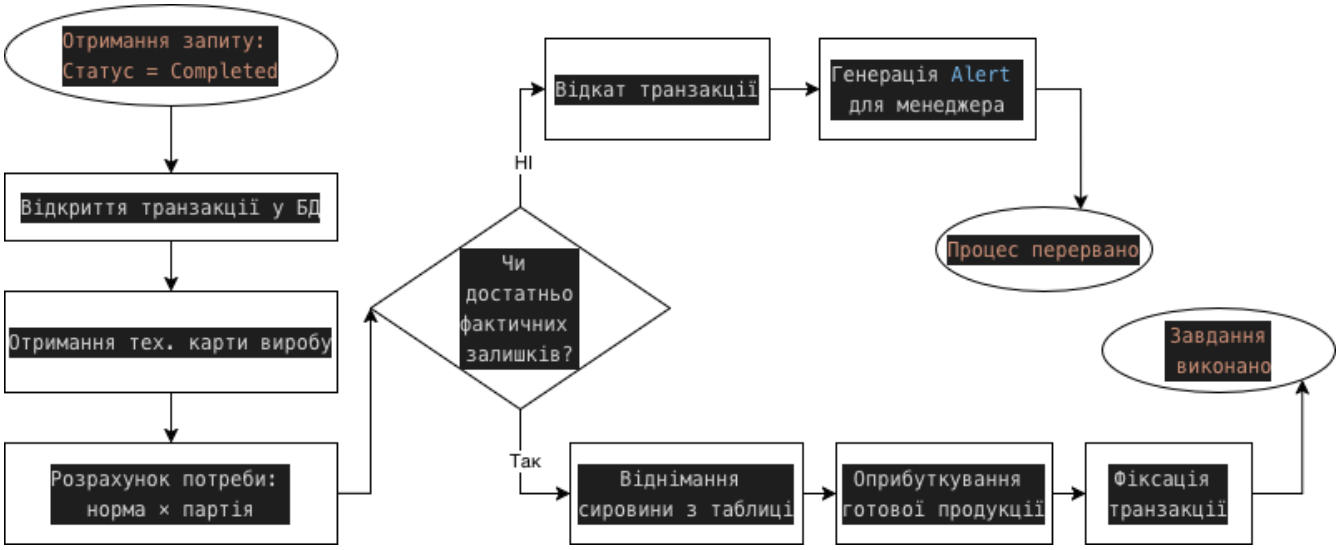

Рисунок 1 - Блок-схема транзакційного алгоритму автоматичного списання сировини

Висновок

Проведене дослідження та системний аналіз бізнес-процесів пекарні довели критичну неефективність ручного документообігу та обмеженість звичайних POS-терміналів в умовах виробництва. Розроблена концепція спеціалізованого веб-застосунку пропонує дієве вирішення проблеми завдяки чіткому розділенню користувацьких інтерфейсів (для цеху, складу та адміністрації) і запровадженню автоматичного транзакційного списання.

Застосування сучасного стеку технологій (Single Page Application, реляційні бази даних) та індустріальних патернів проєктування забезпечить створення масштабованої, безпечної та зручної системи. Зниження кількості людських помилок при розрахунку сировини, контроль життєвого циклу завдань у реальному часі та точний підрахунок собівартості роблять дану роботу надійним фундаментом для подальшої практичного використання в індустрії.

Список використаних джерел

1. 4 Big Challenges in the Bakery Industry and How the Right Software Makes All the Difference. Aptean. 2023. URL: https://www.aptean.com/en-US/insights/blog/challenges-in-the-bakery-industry.
2. Management System – ERP vsBest-of-Breed, Costs, Timeline&DecisionFramework. SG SystemsGlobal. 2023. URL: https://sgsystemsglobal.com/guides/best-bakery-management-system/.
3. Bakery ERP Software: Recipe Management, Production& DSD. inectaFood ERP. 2024. URL: https://www.inecta.com/bakery-erp-software.

УДК 004.62

ТРИРІВНЕВА ВЕРИФІКАЦІЯ ТА ІДЕМПОТЕНТНІСТЬ МІГРАЦІЙНИХ ОПЕРАЦІЙ У ВИСОКОНАВАНТАЖЕНИХ СИСТЕМАХ

Сімак А.Ю.[1)], Белзюк Р.І.[2)]
Західноукраїнський національний університет
[1)] д. філ., викладач; [2)] магістрант

I. Постановка проблеми

Міграція високонавантажених систем електронної комерції передбачає перенесення великих обсягів даних за умов безперервної роботи бізнес-процесів. Дані таких систем підлягають вимогам фіскального та бухгалтерського законодавства, і розбіжність між записами вихідної та цільової баз створює комерційні й юридичні ризики. Ітеративна модель міграції, що зменшує обсяг змін між системами з кожним кроком, потребує формалізованого контролю коректності після кожної ітерації. Класичні методології такого контролю не передбачають.

II. Мета роботи

Метою є розробка механізму контролю коректності та відновлюваності міграційних операцій у високонавантажених системах. Механізм охоплює три рівні верифікації перенесених даних, забезпечує безпечне повторне виконання операцій після збою завдяки властивості ідемпотентності та уможливлює повернення до вихідної системи у разі критичних помилок. Усі компоненти є неінвазивними щодо вихідної системи.

III. Основна частина

Розроблений підхід ґрунтується на ітеративній моделі актуалізації даних, у якій обсяг змін ΔD між вихідною та цільовою системами зменшується з кожним кроком. Після завершення ітерації виконується повний цикл перевірок, і лише за умови їх успішності ітерація вважається завершеною.

Верифікація перенесених даних охоплює три рівні: структурний, контентний та рівень бізнес-логіки (див.рис.1)[1].

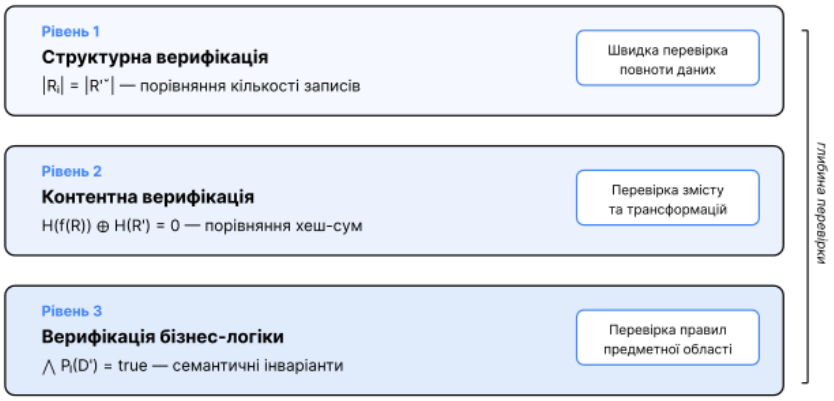

Рисунок 1 – Трирівнева модель верифікації даних

Структурна верифікація оцінює повноту перенесення за кількістю записів. Для кожного типу сутності e_i різниця між кількістю записів у вихідній та цільовій системах на момент завершення ітерації t_k обчислюється за формулою (1):

$$V_{struct}(e_i) = |R_i(t_k)| - |R'_j(t_k)| = 0, \quad (1)$$

де $|R_i(t_k)|$ – кількість записів типу e_i у вихідній системі на момент завершення ітерації t_k, а $|R'_j(t_k)|$ – кількість відповідних записів у цільовійсистемі.

Додатково перевіряється посилальна цілісність: для кожного зовнішнього ключа у цільовій базі підтверджується наявність відповідного первинного ключа. Практичний контроль первинних і зовнішніх ключів під час перенесення між схемами реалізовані у фреймворку DAMI [2]. Структурна верифікація виконується швидко і виявляє пропущені записи, дублікати та порушення посилальної цілісності.

Контентна верифікація оцінює коректність трансформації на рівні змісту записів. Для пакетів записів агреговані хеш-суми у вихідній і цільовій системах порівнюються за формулою (2):

$$V_{contenr}(e_i) = H(f_i(R_i(t_k))) \oplus H(R'_j(t_k)) = 0, \quad (2)$$

де H – функція агрегованої хеш-суми, f_i – функція трансформації записів типу e_i з формату вихідної системи у формат цільової, а $\oplus$ – операція побітового порівняння хеш-сум (XOR).

Застосування криптографічних хеш-функцій для перевірки цілісності даних бере початок від праць у галузі цифрових підписів [3]. Пакетне хешування локалізує розбіжності без повного поелементного порівняння. Для числових полів додатковозвіряютьсясуми, середні значення та екстремуми.

Верифікація на рівні бізнес-логіки перевіряє семантичні інваріанти предметної області. Кожен інваріант визначається як предикат P_l над множиною записів цільової системи, а узагальнена умова подана формулою (3):

$$V_{biz} = \bigwedge_{l=1}^{L} P_l(D'(t_k)) = \text{true} \quad (3)$$

де L – кількість визначених інваріантів. Для систем електронної комерції типовими інваріантами є рівність суми замовлення сумі його позицій з урахуванням знижок і вартості доставки, невід'ємність складських залишків та узгодженість кількості замовлень клієнта з історичними даними. Набір інваріантів формується на етапі аналізу бізнес-процесів вихідної системи.

У разі виявлення розбіжностей на будь-якому рівні верифікації ітерація позначається як невдала, і запускається механізм повторного виконання. Для безпечного повторного виконання міграційна операція має бути ідемпотентною. Формально ідемпотентність операції M для запису r визначається формулою (4):

$$M(M(r)) = M(r). \quad (4)$$

Повторне застосування операції не змінює коректного стану цільової системи. Практична реалізація цієї властивості забезпечується двома механізмами.

По-перше, замість операції вставки (INSERT) використовується операція вставки або оновлення (UPSERT), що створює запис за його відсутності й оновлює існуючий запис за його наявності. Це усуває ризик дублювання під час повторного виконання ітерації.

По-друге, для кожного типу сутності підтримується таблиця стану міграції $\sigma : E \to N \times T$, яка фіксує контрольні точки — кількість успішно оброблених записів та часову мітку останнього з них. Такий підхід продовжує класичну модель відновлення транзакційних систем через контрольні точки та журнал. У разі збою повторне виконання починається з останньої контрольної точки, а вжеобробленізаписионовлюютьсябезпобічнихефектів.

Також реалізовано механізм відкату, який забезпечує повернення до функціонування вихідної системи у разі критичних помилок. Оскільки компонент журналювання не модифікує програмний код і дані вихідної системи, остання продовжує працювати у штатному режимі протягом усього процесу. Для відкатудостатньозупинитиміграцію й відключитижурналювання.

Висновок

Запропонована трирівнева модель верифікації разом із механізмами ідемпотентності та контрольованого відкату забезпечує контроль коректності та відмовостійкість міграційних операцій у високонавантажених системах. Формалізація та реалізація ідемпотентності через UPSERT і контрольні точки уможливлюють безпечне повторне виконання після збоїв. Неінвазивність журналювання робить можливим повернення до вихідної системи без додаткових архітектурних компонентів. Запропоновані механізми становлять основу контролю якості для ітеративних методів міграції.

Список використаних джерел

1. Патлань, Є., Білоус, І. (2025). Метод автоматизованої міграції даних між варіантами сховищ у системах з багатоваріантною персистентністю. Інформаційні технології та суспільство, 4(19), 129–139.
2. Ramos-Vidal, D., Cortiñas, A., Luaces, M. R., Pedreira, O., Saavedra Places, Á., & Assunção, W. K. G. (2025). Seamless Data Migration between Database Schemas with DAMI-Framework: An Empirical Study on Developer Experience, Istanbul, Turkiye, 17–20 June 2025. New York: ACM. 12 p.
3. Merkle, R. C. (1988). A digital signature based on a conventional encryption function. In Advances in Cryptology — CRYPTO '87, Lecture Notes in Computer Science, 293, 369–378. Springer. https://doi.org/10.1007/3-540-48184-2_32.

УДК 004.4

ВЕБ-ЗАСТОСУНОК ДЛЯ ОРЕНДИ НОУТБУКІВ

Присяжнюк Я.В.[1)], Стасів І.С.[2)], Порплиця Н.П.[3)]
Західноукраїнський національний університет
[1)]бакалавр; [2), 3)]к.т.н., доцент

І. Постановка проблеми

У сучасних умовах стрімкого розвитку інформаційних технологій значно зростає попит на цифрові сервіси, що автоматизують бізнес-процеси. Однією з актуальних задач є організація ефективної системи оренди технічного обладнання, зокрема ноутбуків. Така система повинна забезпечувати швидкий доступ користувачів до ресурсів, прозорий облік техніки та контроль за її використанням.

Традиційні підходи до ведення обліку оренди, що базуються на ручному введенні даних або використанні розрізнених програмних засобів, є неефективними та призводять до значної кількості помилок. Відсутність централізованої системи ускладнює процес обробки замовлень, управління статусами пристроїв та здійснення фінансових операцій.

Таким чином, на сьогоднішній день виникає необхідність у створенні сучасного веб-застосунку, що забезпечить автоматизацію процесу оренди ноутбуків, підвищить швидкість та якість обслуговування користувачів та їх комунікацію з власниками техніки.

ІІ. Мета роботи

Метою даної роботи є розробка вебзастосунку для оренди ноутбуків, який дозволить ефективно керувати процесами бронювання, обліку техніки та обробки платежів. Система повинна забезпечувати зручний інтерфейс для користувачів, надійність зберігання даних та можливість масштабування.

ІІІ. Особливості програмної реалізації веб-застосунку для оренди ноутбуків

Розробка вебзастосунку виконана з використанням сучасного технологічного стеку, який включає клієнтську частину, що створена на базі технології React, серверну логіку, написану повністю на Node.js (Express) та систему керування базами даних PostgreSQL [1-6]. Такий підхід дозволяє забезпечити високу продуктивність, гнучкість та масштабованість системи.

Архітектура системи базується на клієнт-серверній моделі. Клієнтська частина відповідає за взаємодію з користувачем, обробку та реакцією на дії користувача, відображення інформації та обробку подій інтерфейсу. Серверна частина реалізує бізнес-логіку, обробку запитів та взаємодію з базою даних.

Основними сутностями системи є «Користувач», «Замовлення», «Ноутбук», «Платіж» та «Статуси». «Користувач» має можливість створювати замовлення, переглядати історію оренди та редагувати власний профіль. «Замовлення» містить інформацію про обраний ноутбук, термін оренди, вартість та статус виконання. Сутність «Ноутбук» включає технічні характеристики пристрою, такі як модель, процесор, обсяг оперативної пам’яті, відеокарта та вартість оренди за день. Для кожного ноутбука визначається статус доступності, що дозволяє уникнути конфліктів при бронюванні.

Обробка платежів реалізована через окремий модуль, який забезпечує фіксацію фінансових операцій та їх зв’язок із замовленнями. Це дозволяє відстежувати доходи та забезпечує прозорість фінансових процесів. Адміністративна панель системи надає можливість керування ресурсами: додавання нових ноутбуків, редагування існуючих записів, видалення пристроїв, формування звітів і дії з профілями користувачів, а також перегляд статистичних даних. Зокрема, адміністратор може аналізувати популярність моделей, кількість активних користувачів та загальний дохід. Для підвищення зручності використання реалізовано систему сповіщень, яка інформує користувачів про зміну статусу замовлення. Це дозволяє оперативно реагувати на зміни в умовах мінливості сучасного ринку та покращує взаємодію із сервісом.

Інтерфейс користувача розроблений із застосуванням сучасних принципів UX/UI дизайну. Забезпечено адаптивність інтерфейсу для різних типів пристроїв, що дозволяє використовувати систему як на персональних комп’ютерах, так і на мобільних пристроях різних роздільних здатностей.

З метою забезпечення безпеки та надійності системи реалізовано механізми автентифікації та авторизації користувачів. Для захисту даних використовується шифрування паролів, а також застосовуються токени доступу для контролю сеансів користувачів. Крім того, передбачено валідацію вхідних даних як на клієнтській, так і на серверній стороні, що дозволяє запобігти некоректній обробці інформації та підвищує загальний рівень захищеності даних користувачів та застосунку.

На рисунку 1 представлено загальну архітектуру системи, що показує взаємодію основних компонентів платформи.

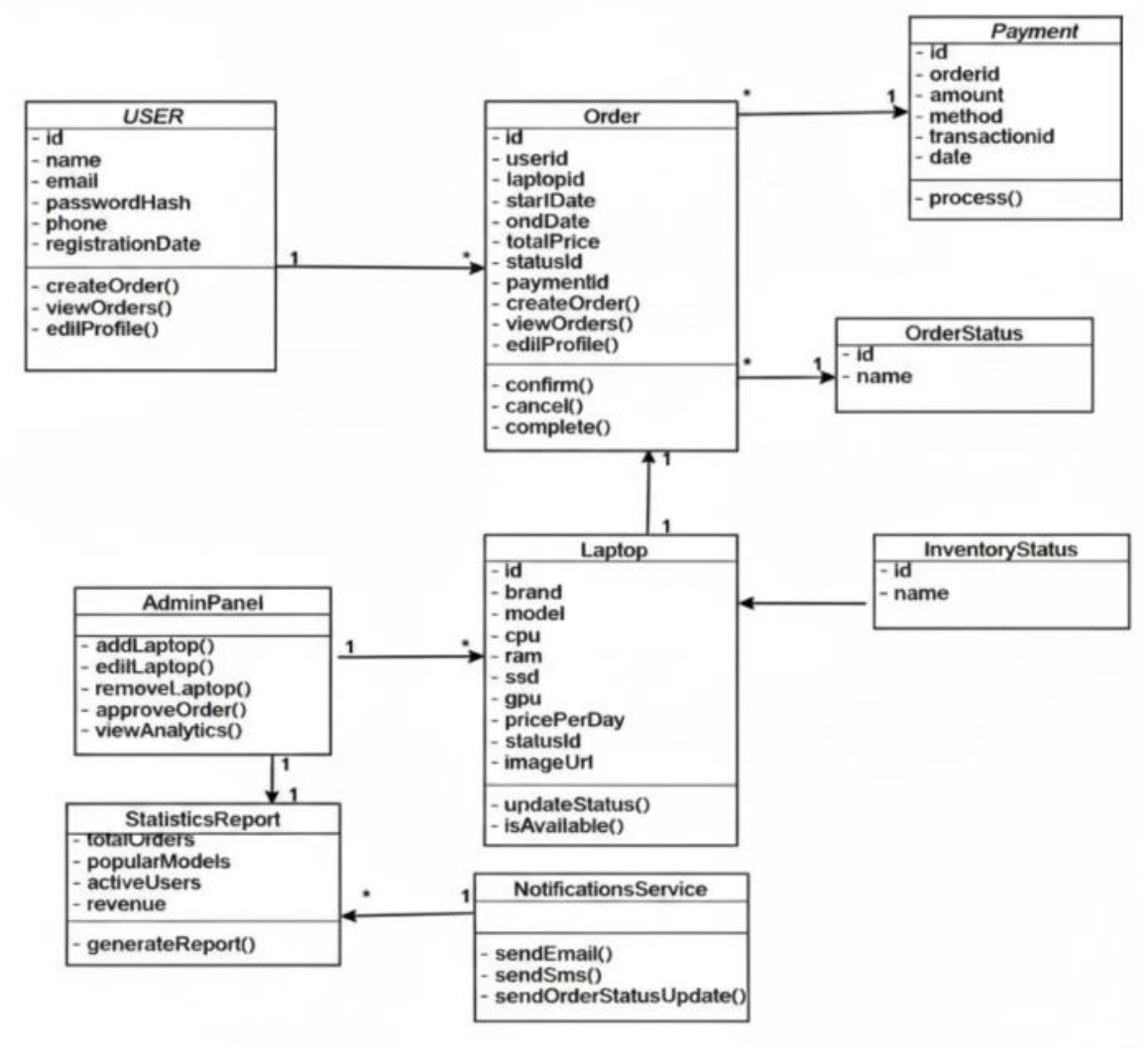

Рисунок 1 – Архітектура системи

Висновок

У роботі наведено особливості програмної реалізації системи оренди ноутбуків, представлену у вигляді ефективного веб-застосунку, який створює продукт не маючий аналогів, автоматизує ключові процеси та підвищує зручність користування як для пересічних користувачів так і для адміністраторів. Використання сучасних технологій дозволяє забезпечити надійність, масштабованість та високу продуктивність системи.

Список використаних джерел

1. Steve Abrams. Software Architecture for Developers: Designing Scalable and Maintainable Systemfor the Real World: навчальний посібник, 2024. 117 с.
2. Jon Duckett. PHP & MySQL: Server-side Web Development: навчальний посібник, 2022. 672с.
3. Adam Aspin. Queryng MariaDB: Use SQL Operations, Data Extraction, and Custom Queries to Make your MariaDB Database Analytics more Accessible: навчальний посібник, 2022. 664 с
4. UserNotifications Framework [Електронний ресурс]. – Режим доступу: https://developer.apple.com/documentation/usernotifications.
5. AsyncIO Documentation for Asynchronous Programming [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/asyncio.html

УДК 004.8

МЕТОД АДАПТИВНОГО ПРОФІЛЮВАННЯ КОМПЕТЕНЦІЙ ДЛЯ ОЦІНЮВАННЯ ЗНАНЬ НА ТЕХНІЧНИХ СПІВБЕСІДАХ

Підгородецький А.А.[1)], Манжула В.І.[2)]
Західноукраїнський національний університет
[1)]магістрант ; [2)]д.т.н., професор

Постановка проблеми

У сучасних умовах розвитку ІТ-галузі стрімке оновлення технологій та зростання вимог до фахівців ускладнюють процес об'єктивного оцінювання їхніх знань і навичок під час технічних співбесід. Традиційні підходи, що базуються на узагальнених або статичних методах оцінювання, не дозволяють повною мірою врахувати багатовимірність компетенцій, які охоплюють різні предметні області та рівні складності. Це ускладнює виявлення конкретних прогалин у підготовці користувача та обмежує можливості побудови ефективної індивідуальної траєкторії навчання. У зв'язку з цим виникає потреба у розробленні математичної моделі адаптивного профілювання компетенцій, яка забезпечує точне багатовимірне оцінювання рівня знань і дозволяє динамічно адаптувати навчальний процес відповідно до індивідуальних особливостей користувача.

Мета дослідження

Метою роботи є розроблення математичної моделі та методу [1] адаптивного профілювання компетенцій, яка забезпечує багатовимірне представлення рівня підготовки користувача та дозволяє здійснювати об'єктивне оцінювання його знань у різних предметних областях. Запропонована модель спрямована на формалізацію процесу оцінювання результатів виконання завдань, динамічне оновлення профілю компетенцій та інтелектуальний підбір навчального контенту відповідно до індивідуальних особливостей користувача. Реалізація такого підходу має забезпечити підвищення ефективності підготовки до технічних співбесід за рахунок персоналізації навчального процесу та оптимізації складності запропонованих завдань.

Метод адаптивного профілювання компетенцій

Нехай

$$D = \{d_1, d_2, ..., d_m\} \tag{1}$$

– множина предметних областей (алгоритми, бази даних, системний дизайн тощо). Тоді профіль знань користувача θ_u можна представити у вигляді вектора

$$\theta_u = [\theta_{u,1}, \theta_{u,2}, ..., \theta_{u,m}], \theta_u \in R^m \tag{2}$$

де кожна компонента $\theta_{u,j}$, кількісно відображає його поточний рівень компетенції в окремій предметній області j.

Результат виконання завдання i користувачем u описується випадковою величиною:

$$Y_{ui} \in \{0,1\} \tag{3}$$

Ймовірність правильної відповіді визначається як:

$$P(Y_{ui} = 1 | \theta_{u,j}) = \frac{1}{1 + \exp(-a_i(\theta_{uj} - b_i))} \tag{4}$$

де b_i – параметр складності завдання, a_i – параметр дискримінації, який описує наскільки добре завдання дозволяє відрізнити сильного кандидата від слабкого.

Після того як користувач дає відповіді на тестові питання або пише код, система повинна оновити його рейтинг. Для оновлення оцінки компетенції використовується метод максимальної правдоподібності.

Нехай

$$y_j = [y_{u1}, y_{u2}, \ldots, y_{un}] \tag{5}$$

– вектор відповідей користувача в області d_j.

Оцінка параметра $\theta_{u,j}$ знаходиться шляхом максимізації логарифмічної функції правдоподібності:

$$L(\theta_{u,j}) \to \max \tag{6}$$

Для чисельного розв'язання задачі (6) використовується метод Ньютона-Рафсона [2,3]:

$$\theta_{u,j}^{(t+1)} = \theta_{u,j}^{(t)} + \frac{\sum_{k=1}^{n} a_k \cdot [y_{uk} - P_k(\theta_{u,j}^{(t)})]}{\sum_{k=1}^{n} a_k^2 \cdot P_k(\theta_{u,j}^{(t)} \cdot (1 - P_k(\theta_{u,j}^{(t)}))} \tag{7}$$

де $y_{uk} \in \{0,1\}$ – фактичний результат виконання k-го завдання (успіх або невдача). Це дозволяє системі динамічно адаптувати профіль кандидата після кожної його дії.

Щоб процес навчання був максимально ефективним, система шукає в базі таку задачу i^*, яка принесе найбільше корисної інформації про поточний рівень знань користувача. Це реалізується за допомогою інформаційної функції Фішера:

$$I_i(\theta_{u,j}) = a_i^2 \cdot P_i(\theta_{u,j}) \cdot (1 - P_i(\theta_{u,j})) \tag{8}$$

Оптимальне завдання визначається як:

$$i^* = \arg\arg I_i(\theta_{u,j}), i \in S_j \tag{9}$$

де S_j – множина доступних завдань.

На практиці це означає, що алгоритм адаптивного навчання обирає такі завдання, для яких оцінена ймовірність успішного розв'язання конкретним користувачем наближається до середнього рівня, орієнтовно близько 50%. Такий вибір не є випадковим, оскільки саме в цій точці інформаційна цінність завдання є максимальною, а отже кожна відповідь дає найбільший внесок у уточнення рівня компетенції. Якщо завдання занадто просте, система майже не отримує нової інформації, оскільки правильна відповідь є передбачуваною. Якщо ж завдання надмірно складне, то ймовірність помилки зростає настільки, що результат також малоінформативний. Баланс між цими крайнощами дозволяє підтримувати оптимальний темп навчання.

Висновок

У результаті дослідження було розроблено математичну модель адаптивного профілювання компетенцій, яка забезпечує багатовимірне та об'єктивне оцінювання рівня підготовки користувача. Використання логістичної моделі та методу максимальної правдоподібності дозволяє динамічно оновлювати профіль знань, а застосування інформаційної функції Фішера – ефективно підбирати завдання оптимальної складності. Запропонований підхід поєднує оцінювання та адаптацію навчального процесу, що сприяє персоналізації підготовки та підвищенню її ефективності відповідно до сучасних вимог технічних співбесід.

Список використаних джерел

1. Article. Why the log-likelihood? [Електронний ресурс] - Режим доступу: https://blog.metaflow.fr/ml-notes-why-the-log-likelihood-24f7b6c40f83
2. Kong Q., Siauw T., Bayen A. M. Python Programming and Numerical Methods: A Guide for Engineers and Scientists. 1st ed. Academic Press, 2020. С. 330-334. ISBN 978-0-12-819549-9
3. Article. Fisher Information [Електронний ресурс] - Режим доступу: https://awni.github.io/intro-fisher-information/

УДК 004.4:004.056:004.8

МЕТОДИ ТА АЛГОРИТМИ ВИЯВЛЕННЯ КОНФІДЕНЦІЙНИХ ДАНИХ У ВЗАЄМОДІЇ ПРАЦІВНИКІВ ІЗ СИСТЕМАМИ ШТУЧНОГО ІНТЕЛЕКТУ

Сімак А.Ю.[1)], **Пасічник В.П.**[2)]
[1-2)]*Західноукраїнський національний університет*
[1)]*д.філ.,викладач;* [2)]*магістрант*

I. Постановка проблеми

Інтеграція систем штучного інтелекту (ШІ) в бізнес-процеси компаній підвищує ризик витоку конфіденційної інформації через необережні або зловмисні дії співробітників. Зі збільшенням обсягів оброблюваних ШІ даних зростає ймовірність випадкового або навмисного розкриття чутливої інформації, що може спричинити серйозні фінансові, юридичні та репутаційні наслідки для компаній.

Відомі випадки, коли через недотримання норм безпеки співробітники мимоволі вводили конфіденційну інформацію, таку як номери платіжних карток, медичні записи або персональні дані клієнтів, у загальнодоступні системи ШІ, що призводило до масштабних витоків інформації. Такі інциденти можуть спричинити значні штрафи, втрату довіри клієнтів та навіть закриття бізнесу. Таким чином, актуальною є розробка ефективних алгоритмів та методів для раннього виявлення і запобігання витоку конфіденційних даних.

II. Мета роботи

Метою дослідження є обґрунтування вибору та застосуванняматематичних методів і алгоритмів для автоматичного виявлення конфіденційних даних у промптах користувачів до систем штучного інтелекту в режимі реального часу.

III. Метод вирішення задачі

Для вирішення задачі запропоновано систему, що складається з браузерного розширення, сервісу фільтрації запитів, спеціалізованих LLM-моделей для аналізу тексту, та окремого процесу для класифікації конфіденційних даних.

Запити користувачів перехоплюються за допомогою розширення і направляються на сервер, який спочатку визначає наявність промпту у запиті. Потім спеціалізована LLM-модель проводить токенізацію тексту запиту (промпту), після чого інша класифікаційна модель аналізує його на предмет наявності конфіденційних даних: особистої інформації, даних платіжних карток, медичної інформації чи приватної фінансової інформації [1].

IV. Математичне забезпечення та алгоритм роботи підсистеми класифікації

Програмна реалізація браузерного розширення базується на використанні сучасних підходів у галузі машинного навчання та NLP-технологій. Центральним компонентом системи є спеціалізовані LLM-моделі, розроблені на мові програмування Python. Ці моделі покликані ефективно аналізувати перехоплені запити користувачів та визначати наявність і тип конфіденційної інформації.

Після перехоплення HTTP-запиту через Chrome Extension API відбувається попередня обробка тексту промпта, яка включає очищення від зайвих символів та нормалізацію тексту. Далі текст передається на сервер, де здійснюється його детальний аналіз за допомогою попередньо навчених Transformer-моделей, таких як BERT або RoBERTa[2].Аналіз існуючих підходів до пошуку витоків інформації показує, що класичні методи на основі регулярних виразів (RegEx) або словникового пошуку є неефективними для ШІ-запитів через високу варіативність природної мови. Тому в розробленій системі застосовано алгоритми глибокого навчання, здатні враховувати контекст повідомлення[2,3].Принцип роботи системи аналізу даних візуалізовано на рисунку 1.

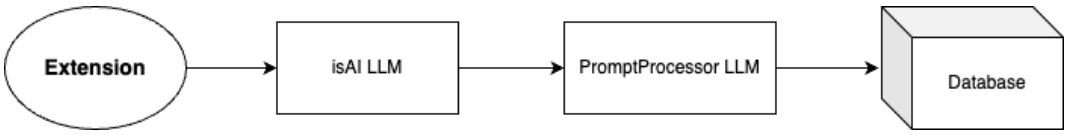

Рисунок 1 – Принцип роботи системи аналізу даних

Ці моделі попередньо тренуються на великих обсягах текстових даних і здатні ефективно визначати контекст та приховані закономірності у тексті, що дозволяє їм розпізнавати конфіденційну інформацію з високою точністю. В процесі роботи модель використовує багаторівневі представлення тексту, що забезпечують глибоке розуміння змісту та структури аналізованих даних. Для оптимізації

обчислювального процесу та роботи з тензорами програмну реалізацію алгоритмів виконано мовою Python із використанням фреймворків TensorFlow.

Процес класифікації включає два основних етапи. На першому етапі за допомогою Named Entity Recognition (NER) відбувається виявлення чутливих елементів у тексті. Для цього модель застосовує механізм самоуваги (self-attention), обчислюючи значення ваг за формулою:

$$Attention(Q,K,V) = soft\max\left(\frac{QK^T}{\sqrt{d_k}}\right)\cdot V, \quad (1)$$

де Q,K,V – відповідно матриці запиту, ключа і значення; K^T – транспонована матриця ключів, а d_k – розмірність вектора ключа.

На другому етапі відбувається безпосередньо класифікація тексту за допомогою моделі, оптимізованої функцією втрат крос-ентропії, що має вигляд:

$$L = -\sum_{i=1}^{C} y_i \log(p_i) \quad (2)$$

де C – кількість класів (кількість можливих категорій, до яких належить текст або сутність, що класифікується); y_i – істинна мітка класу; p_i – ймовірність, передбачена моделлю.

На рисунку 2 зображена архітектура та pipeline обробки користувацьких запитів із застосуванням LLM, механізму self-attention (1) та класифікаційної моделі (2).

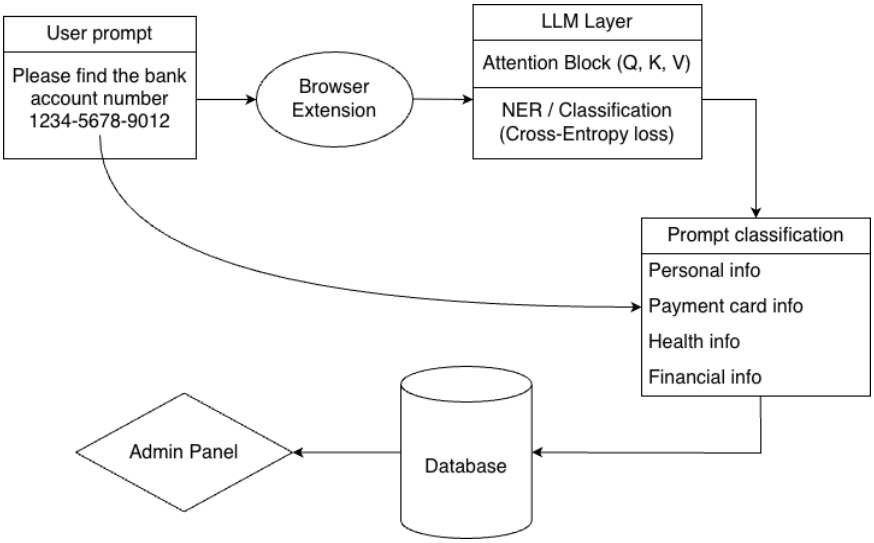

Рисунок 2 – Використання механізму self-attention та класифікаційної моделі під час обробки користувацьких запитів

Виявлені конфіденційні дані зберігаються в централізованому сховищі для подальшого аналізу та реагування. Система побудована з використанням мікросервісної архітектури та контейнеризації за допомогою Docker і Kubernetes, що дозволяє забезпечити високу продуктивність та надійність у масштабах корпоративного використання [3].

Висновок

Запропонована методика дозволяє ефективно виявляти і запобігати витокам конфіденційних даних під час взаємодії співробітників з системами ШІ. На відміну від класичних алгоритмів текстового пошуку, використання спеціалізованих LLM-моделейта архітектури Transformer для визначення наявності і типу конфіденційної інформації у промптах є перспективним напрямом для подальших досліджень та практичного впровадження.

Список використаних джерел

1. Василенко Ю. О. Захист персональних даних при використанні штучного інтелекту. Інформаційна безпека, 2022.
2. Машинне навчання для інформаційної безпеки: сучасні методи та алгоритми / За ред. Петрова В.А. – Київ: Наук. думка, 2021.
3. Smith J., Doe A. Deep Learning Techniques for Confidential Information Detection. Cybersecurity and AI Journal, 2023.

УДК 004.41

ІНТЕРПРЕТОВАНІ AI-ОЗНАКИ ДЛЯ РАННЬОГО ОЦІНЮВАННЯ НАДІЙНОСТІ ІНФОРМАЦІЇ В ПОВІДОМЛЕННЯХ ПРО IT-ІНЦИДЕНТИ

Мацук А.Р.[1)], **Гончар Л.І.**[2)]
Західноукраїнський національний університет
[1)] аспірант; [2)] к.е.н., доцент

І. Постановка проблеми

Масштабні збої в IT-інфраструктурі впливають не лише на роботу окремого сервісу, а й на пов'язані фінансові платформи, державні системи, хмарні рішення та щоденні цифрові комунікації. Під час інциденту провайдери публікують короткі статусні повідомлення про вплив, можливі причини та дії з відновлення. Для клієнтів і партнерів такі повідомлення часто стають першим офіційним джерелом даних, хоча ранні оновлення формуються в умовах неповної інформації.

Проблема полягає в тому, що користувачі та внутрішні команди змушені приймати рішення ще до появи повного технічного звіту: запускати резервні сценарії, змінювати маршрути трафіку, призупиняти залежні процеси або готувати власні пояснення для клієнтів. Якщо початкове повідомлення є надто загальним, спекулятивним або пізніше суттєво змінюється, це збільшує координаційні витрати та знижує довіру до провайдера. У дослідженнях з incident management основний акцент зазвичай зроблено на виявленні аномалій, локалізації першопричини та технічному відновленні, тоді як якість супровідної текстової комунікації вивчена слабше.

Непрозорі моделі машинного навчання можна застосувати для оцінювання таких повідомлень, але в операційному середовищі високого ризику цього недостатньо. Інженер або incident manager має розуміти, чому система вважає повідомлення слабким або достатньо надійним. Тому актуальним є підхід, що спирається на прості інтерпретовані ознаки: часові уточнення, маркери невизначеності, технічну деталізацію, опис масштабу впливу та конкретні дії з відновлення.

ІІ. Мета роботи

Метою роботи є дослідження зв'язку між інтерпретованими текстовими ознаками статусних повідомлень і рівнем їх інформаційної надійності в умовах IT-інцидентів, а також оцінювання придатності таких ознак для побудови простих і прозорих моделей раннього прогнозування. Для цього було поставлено завдання проаналізувати відкриті історії інцидентів Cloudflare, сформувати підхід до маркування повідомлень за рівнем надійності, спроєктувати набір інтерпретованих ознак і визначити найбільш інформативні з них.

ІІІ. Обґрунтування отриманих результатів

Емпіричною основою дослідження стали відкриті дані зі статус-сторінки Cloudflare, яка надає історію інцидентів через Status API [1,2]. Було зібрано 50 інцидентів і 180 статусних оновлень. Після фільтрації повідомлень зі статусами, які не підходили для бінарної класифікації, до аналізу увійшло 139 оновлень. Інформаційну надійність апроксимовано через стан повідомлення в життєвому циклі інциденту: статуси investigating та identified віднесено до класу низької надійності, а resolved і postmortem - до класу високої надійності. Така схема не вимірює абсолютну істинність тексту, але дає відтворювану proxy-мітку для навчального експерименту. Набір ознак побудовано так, щоб кожну з них можна було пояснити без складної моделі. До структурних ознак віднесено довжину повідомлення, кількість речень, наявність списків і часових уточнень у тілі тексту. Окремо враховано ясність опису впливу, згадки про компоненти або регіони, маркери технічної деталізації, слова невизначеності на зразок may, might, possible, suspect, investigating, маркери впевненості на зразок confirmed і root cause identified, а також фрази про відновлювальні дії, наприклад deployed a fix або rolled back. Такий набір узгоджується з логікою explainable AI, де пріоритет має не лише точність, а й можливість перевірити причину рекомендації [3]. Для перевірки інформативності ознак використано три підходи: класифікатор більшості як базову лінію, логістичну регресію та неглибоке дерево рішень. Розподіл даних виконано на рівні інцидентів у пропорції 60/20/20, а не на рівні окремих повідомлень. Це важливо, бо кілька оновлень одного інциденту мають спільний контекст, і випадкове змішування таких оновлень між train/test вибірками призвело б до витоку інформації. Після маркування вибірка містила 81 тренувальний, 34 валідаційні та 24 тестові повідомлення.

Результати показують (див.табл.1), що інтерпретовані ознаки мають помітний прогностичний сигнал. Базовий класифікатор досяг accuracy 0,583 та macro F1 0,368, що відображає дисбаланс класів. Логістична регресія підвищила обидва показники до 0,750, а дерево рішень отримало

найкращий результат - accuracy 0,917 та macro F1 0,914. Отже, навіть прості прозорі моделі можуть відділяти ранні невизначені повідомлення від пізніших стабільних оновлень краще, ніж наївний підхід.

Таблиця 1

Результати класифікації на тестовій вибірці

Модель	Accuracy	Macro F1
Класифікатор більшості	0,583	0,368
Логістична регресія	0,750	0,750
Дерево рішень	0,917	0,914

Отримані результати слід розглядати як підтвердження працездатності підходу, а не як фінальну універсальну модель для всіх типів IT-інцидентів. Набір даних є відносно невеликим і походить з одного джерела, тому стиль повідомлень, структура статусів і термінологія є досить однорідними. Це полегшує початкове моделювання, але обмежує узагальнення результатів на інші компанії або інші формати інцидентної комунікації. Водночас саме така однорідність дозволяє чітко побачити, що навіть прості текстові сигнали мають практичну цінність для раннього оцінювання надійності. Найбільший внесок у прогноз зробили маркери впевненості, часові уточнення в тексті, довжина повідомлення та рівень технічної деталізації. Натомість велика кількість hedging-слів частіше відповідала нижчій надійності. Це не означає, що невизначеність потрібно приховувати. Навпаки, на ранньому етапі інциденту вона є нормальною. Практичний висновок інший: система може підказати автору, де варто чіткіше відокремити підтверджені факти від припущень, додати часову рамку, описати масштаб впливу або уточнити наступний крок.

Окремий якісний аналіз інциденту Cloudflare від 18 листопада 2025 року підтвердив таку динаміку: ранні повідомлення були коротшими, містили більше невизначеності та менше технічних деталей, а подальші оновлення ставали конкретнішими й ближчими до фінального опису причин і наслідків збою. Це підтримує ідею інтеграції feature-based feedback у внутрішні інструменти incident management. Такий модуль не повинен автоматично вирішувати, чи повідомлення є істинним. Його роль - показати слабкі місця тексту до публікації та зменшити ризик нечіткої комунікації. З практичної точки зору запропонований підхід можна реалізувати як допоміжний сервіс у процесі підготовки публічного статусного оновлення. Сервіс отримує текст повідомлення, обчислює ознаки, порівнює їх з типовими патернами низької та високої надійності й повертає не категоричний вердикт, а пояснювану підказку. Водночас отримані результати потрібно трактувати як proof-of-concept. Мітка надійності побудована за статусом повідомлення в життєвому циклі інциденту, тому вона відображає стабільність і зрілість комунікації, а не пряму перевірку істинності кожного твердження. Крім того, дані походять від одного провайдера, що зменшує стилістичний шум, але обмежує узагальнення результатів. Висока якість дерева рішень також може частково пояснюватися малим і відносно однорідним корпусом. Попри ці обмеження, експеримент показує важливу інженерну властивість: для первинного контролю статусної комунікації не обов'язково одразу застосовувати великі мовні моделі. У багатьох сценаріях достатньо набору зрозумілих ознак, невеликої моделі та прозорого інтерфейсу, який допомагає команді писати точніше. Саме така lightweight-архітектура краще підходить для incident management, де швидкість, контроль і пояснюваність часто важливіші за складність моделі.

Висновок

У роботі показано, що структурна ясність, часові уточнення, технічна деталізація, маркери впевненості та конкретні дії з відновлення пов'язані з вищою інформаційною надійністю статусних повідомлень під час IT-інцидентів. Запропонований підхід не замінює експертну оцінку та не претендує на визначення абсолютної істини, але дає прозорий механізм ранньої перевірки якості комунікації. Практична цінність результатів полягає у можливості створити модуль підтримки для incident managers та інженерних команд. Такий модуль може в реальному часі підсвічувати надмірну невизначеність, брак часових уточнень, слабкий опис впливу або відсутність конкретних дій.

Список використаних джерел

1. Cloudflare. Cloudflare Global Network | Data Center Locations. 2026. URL: https://www.cloudflare.com/network/.
2. Cloudflare. Cloudflare Status API (v2). 2026. URL: https://www.cloudflarestatus.com/api/.
3. Rudin C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence. 2019. Vol. 1. P. 206-215. DOI: 10.1038/s42256-019-0048-x.

УДК 004.41:004.89

ЗАСТОСУВАННЯ МОДЕЛІ ЗВАЖЕНОЇ СУМИ ТА ЛОГІСТИЧНОЇ РЕГРЕСІЇ ДЛЯ ІНТЕЛЕКТУАЛЬНОГО РАНЖУВАННЯ ЗАРЯДНИХ СТАНЦІЙ

Сімак А.Ю.[1)], **Кобилан В.С.**[2)]
Західноукраїнський національний університет
[1)]д. філ., викладач; [2)] магістрант

I. Постановка проблеми

В умовах енергетичної нестабільності та стрімкого розширення ринку портативних джерел енергії, вибір оптимального варіанту зарядної станції ускладнюється постійною появою нових конфігурацій та брендів. Підбір пристрою стає складним багатокритеріальним завданням, оскільки потребує врахування багатьох взаємозалежних параметрів: потужності, ємності акумулятора, часу зарядки, мобільності та вартості. Традиційні калькулятори енергоспоживання лише підсумовують навантаження, але не надають об'єктивних рекомендацій щодо конкретних моделей пристроїв. Для розв'язання цієї проблеми необхідно впровадити математичні методи, що дозволять автоматизувати процес прийняття рішень та класифікації обладнання за сценаріями використання.

II. Мета роботи

Метою дослідження є обґрунтування та реалізація математичного апарату на основі моделі зваженої суми (WeightedSumModel, WSM) та логістичної регресії для автоматизованого підбору зарядних станцій відповідно до персоналізованих критеріїв користувача.

III. Метод вирішення задачі

Для досягнення мети у системі поєднано два підходи:

1. Модель зваженого сумарного балу (WSM). Використовується для ранжування доступних моделей станцій. Кожен критерій (ціна, вага, потужність) нормалізується та отримує ваговий коефіцієнт значущості. Оптимальним вважається варіант із найвищим сумарним балом.
2. Логістична регресія (LogisticRegression). Застосовується для автоматичної класифікації зарядних станцій на три групи: для дому (HomeUse), для роботи (WorkUse) та для кемпінгу (Camping). Модель працює з ймовірностями приналежності об’єкта до класу на основі вхідних характеристик.

Процес обробки даних представлено на блок-схемі (див.рис. 1).

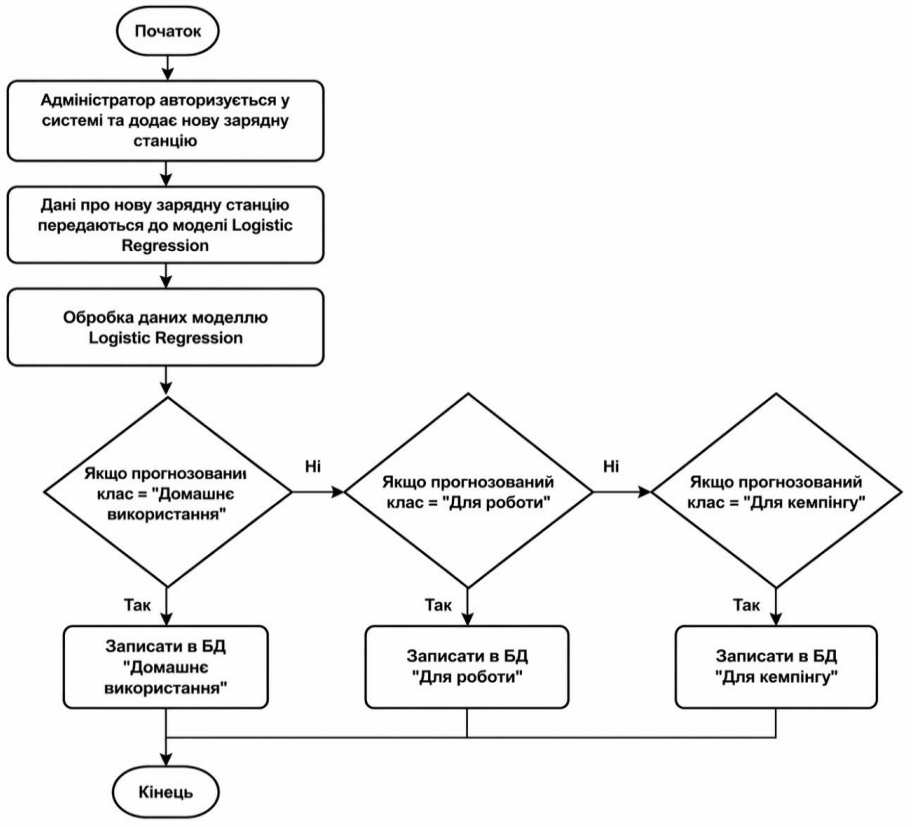

Рисунок 1 – Алгоритм класифікації зарядних станцій за категоріями

IV. Особливості програмної реалізації

Програмна реалізація математичного модуля виконана мовою Python, що обумовлено наявністю потужних бібліотек для наукових обчислень та машинного навчання. Архітектурно обчислювальне ядро виділене в окремий ізольований мікросервіс на базі фреймворкуFlask. Такий підхід знімає математичне навантаження з основного веб-бекенду, дозволяє незалежно масштабувати ML-модуль та створює передумови для подальшої інтеграції з системами черг повідомлень для асинхронної обробки значних масивів товарних позицій. Взаємодія з основним застосунком здійснюється через REST API із використанням JSON-формату. Логіка взаємодії між компонентами системи, що забезпечує передачу параметрів користувача до обчислювального модуля та повернення результатів, детально відображена на діаграмі послідовності (див.рис.2).

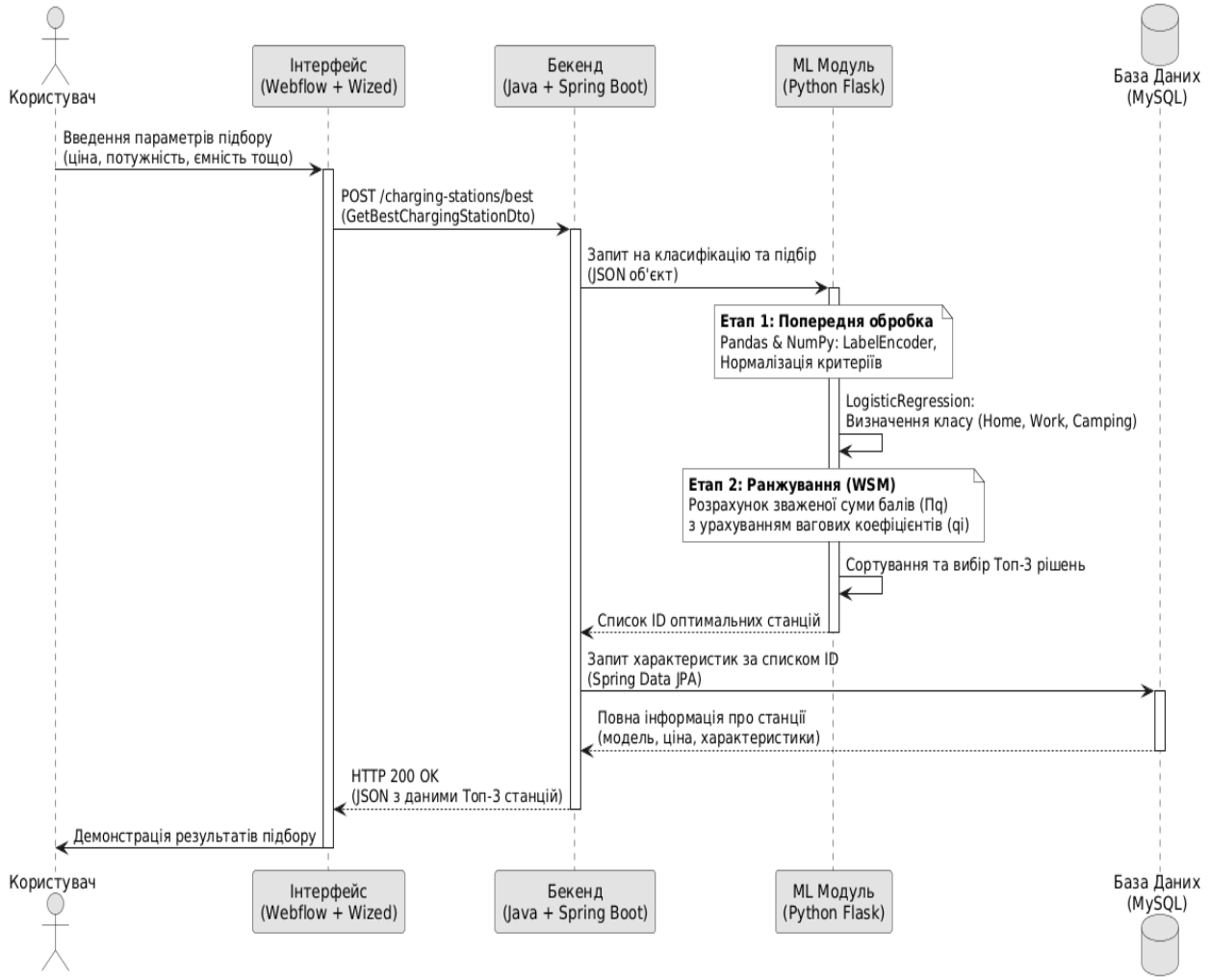

Рисунок 2 – Діаграма послідовності

Процес реалізації інтелектуального ранжування розділений на декілька етапів:

1. Попередня обробка та маніпуляція даними. Для цього використовується бібліотека Pandas для структурування інформації про зарядні станції та NumPy для виконання високопродуктивних матричних операцій. Для коректної роботи моделі логістичної регресії застосовується LabelEncoder для кодування категоріальних ознак у числовий формат.
2. Реалізація моделі класифікації. На основі бібліотеки Scikit-learn побудовано модель LogisticRegression. Формування набору даних для навчання логістичної регресії базується на агрегації технічних характеристик актуальних моделей зарядних станцій, зібраних з відкритих платформ електронної комерції. Набір даних розподіляється на навчальну та тестову вибірки у співвідношенні 80/20 за допомогою train_test_split. Модель класифікує станції за сценаріями «HomeUse», «WorkUse» та «Camping» на основі таких предикторів, як вихідна потужність, ємність акумулятора та вага.
3. Застосування алгоритму багатокритеріального ранжування. Програмно реалізовано метод зваженої суми значень критеріїв. Кожен параметр (ціна, потужність, час зарядки тощо) проходить етап нормалізації для приведення до безрозмірних одиниць за формулами:

$$\text{а) } C_i = x_i / x_{\max} \quad \text{або} \quad \text{б) } C_i = x_{\min} / x_i \tag{1}$$

де C_i – частковий відносний показник якості (нормалізоване значення) для *i*-го критерію; x_i – фактичне кількісне значення *i*-го параметра конкретної зарядної станції; $x_{\max}$ – максимальне значення даного параметра серед усіх розглянутих альтернатив на ринку; $x_{\min}$ – мінімальне значення даного параметра серед усіх розглянутих альтернатив на ринку.

Вибір конкретної формули залежить від типу критерію: для параметрів-стимулів (наприклад, потужність) використовується формула (1.а), а для дестимулів (ціна, вага) – формула (1.б). Системно кожному критерію присвоюється ваговий коефіцієнт q_i. Розрахунок підсумкового балу $П_q$ для кожної станції здійснюється за формулою:

$$П_q = \sum_{i=1}^{m} C_i \cdot q_i \tag{2}$$

де $П_q$ – комплексний показник якості (інтегральний рейтинг), за яким здійснюється ранжування станцій; C_i – нормалізоване значення i-го одиничного показника; q_i – коефіцієнт вагомості (пріоритетності) даного показника для користувача; m – кількість параметрів, за якими проводиться розрахунок.

Такий підрахунок дозволяє автоматично сортувати зарядні станції за рейтингом та надавати користувачу рекомендації у вигляді «Топ-3» найбільш оптимальних варіантів.

Для оцінки ефективності та верифікації точності моделі використовуються метрики classification_report та accuracy_score. Експериментальне тестування на реальних наборах даних підтвердило точність підбору на рівні 92%, а середній час обробки запиту становить 1,8 секунди.

Висновок

Запропонований підхід демонструє ефективність поєднання логістичної регресії для класифікації сценаріїв використання та моделі зваженої суми для інтелектуального ранжування зарядних станцій. Експериментально підтверджено ефективність розробленого методу: точність класифікації (accuracy) склала 92% при середньому часі обробки запиту 1,8 с. Архітектурне виділення математичного ядра у Flask-мікросервіс та використання сучасних засобів аналізу даних (Pandas, Scikit-learn) забезпечують масштабованість і надійність системи. Практична цінність розробленого підходу полягає в можливості його безшовної інтеграції в комерційні платформи-агрегатори та маркетплейси як інтелектуального асистента покупця, що суттєво оптимізує клієнтський досвід вибору складного технічного обладнання. Це створює міцний фундамент для розробки сучасних інтелектуальних систем підтримки прийняття рішень в енергетичному секторі.

Список використаних джерел

1. Порплиця Н.П., Колибан В.С. Веб-платформа для динамічного керування та ранжування зарядних станцій // Матеріали зимової школи-семінару молодих вчених і студентів “Комп’ютерні інформаційні технології” (CIT’2024). – Тернопіль: ЗУНУ, 2024. – С. 17–18.
2. Національний інститут стратегічних досліджень. Штучний інтелект в енергетиці: аналітична доповідь. Київ, 2022. – С. 7-17.
3. Triantaphyllou E. Multi-Criteria Decision Making Methods: A Comparative Study. Dordrecht: Springer, 2000. – pages 6-8.
4. Scikit-learn developers. LogisticRegression — scikit-learn documentation.
5. Alrifaie, M.F., Habbal, A., & Kim, B. (2024). A Fuzzy-Multi Attribute Decision Making Scheme for Efficient User-Centric EV Charging Station Selection. IEEE Access, 12, 161134-161154.
6. Pandas User Guide: Intro to data structures/Pandas Development Team. [Електронний ресурс]. – Режим доступу: https://pandas.pydata.org/docs/user_guide/dsintro.html.
7. Harrell, F. E. 2015. Regression Modeling Strategies. Springer, P. 67-90.
8. Scikit-learn: Machine Learning in Python/Pedregosa F. et al. Journal of Machine Learning Research. 2011. Vol. 12. P. 2825–2830.
9. Gaurav Kumar & N. Parimala, 2020. "A weighted sum method MCDM approach for recommending product using sentiment analysis," International Journal of Business Information Systems, Inderscience Enterprises Ltd, vol. 35(2), pages 185-203.
10. Quickstart — Flask Documentation (3.0.x)/Pallets Projects. [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/en/3.0.x/quickstart/.

УДК 004.8

МЕТОДИ ТА МІКРОСЕРВІС ДЛЯ ОРГАНІЗАЦІЇ КОМАНДНОЇ РОБОТИ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ

Крижанівський Р.В.
Західноукраїнський національний університет
магістрант;

Постановка проблеми

В умовах стрімкого розвитку індустрії програмного забезпечення масове використання гнучких методологійпризводить до значної децентралізації процесів управління вимогами. У крос-функціональних командах завдання часто створюються різними учасниками у вільному стилі з використанням професійного сленгу. Це призводить до інформаційної ентропії беклогу та масової появи прихованих семантичних дублікатів — завдань, що описують одну й ту саму проблему різними словами. Традиційні системи пошуку в таск-трекерахбазуються на лексичному аналізі[1]. Вони не здатні розуміти контекст, що призводить до втрати часу на ручне впорядкування беклогу та ризику паралельного виконання ідентичної роботи різними розробниками.

Мета дослідження

Метою даної роботи є підвищення ефективності управління проектними вимогами шляхом розробки інтелектуального методу та програмного мікросервісу для автоматизованого семантичного виявлення дублікатів завдань з використанням векторних моделей обробки природної мови (NLP). Основний акцент робиться на створенні проактивної системи, здатної аналізувати нові завдання в режимі реального часу та попереджати команду ще на етапі їх створення.

Особливості реалізації та математична модель

Для вирішення проблеми лексичної обмеженості базових пошукових систем запропоновано перехід до методів дистрибутивної семантики на базі архітектури Transformer. В якості базової моделі обрано Sentence-BERT, яка генерує щільні векторні представлення (ембеддінги) цілих речень, враховуючи двонаправлений контекст тексту[2]. Процес ідентифікації дублікатів складається з наступних етапів:

1. Вхідні тексти завдань проходять етап нормалізації. За допомогою регулярних виразів з тексту видаляється технічний «шум»: маркери оцінки складності, Markdown-розмітка, HTML-теги та URL-посилання. Це дозволяє моделі сфокусуватися виключно на семантичному ядрі проблеми.
2. Очищений текст перетворюється на математичний вектор v_{new} у n-вимірному просторі (n=384). Беклог проекту зберігається у вигляді матриці векторних вкладень.
3. Алгоритм обчислює відстань між новим вектором та всіма векторами існуючого беклогу з використанням метрики косинусної подібності. Ця метрика ігнорує довжину тексту і фокусується лише на куті між векторами, що ідеально підходить для порівняння різних по довжині описів.
4. Вводиться гіперпараметр — поріг подібності T. Експериментальним шляхом на базі вибірки з 650 реальних проектних завдань доведено, що оптимальним порогом є $T = 0.85$. Якщо косинусна подібність перевищує це значення, система класифікує завдання як дублікат.

Архітектура програмного засобу

Розроблена система спроектована як незалежний подійно-орієнтований мікросервіс, інтегрований з таск-трекером через механізм Webhooks. Архітектура побудована на базі середовища Node.js, що забезпечує високу швидкість обробки асинхронних мережевих запитів.Завдяки використанню JavaScript-бібліотеки Transformers.js [3], нейромережева модель all-MiniLM-L6-v2 виконується безпосередньо у середовищі Node.js за допомогою технології WebAssembly.Логічна структура системи включає наступні компоненти:

1. Модуль маршрутизації: На базі фреймворкуProbot [4] перехоплює події створення нових завдань у реальному часі.
2. In-Memory сховище векторів: Забезпечує миттєвий доступ до розрахованих ембеддінгів беклогу із константною складністю доступу $O(1)$.
3. Модуль зворотного зв'язку: У разі виявлення дублікатів ($T \geq 0.85$), генерує попереджувальний коментар та публікує його через GitHub REST API. Якщо завдання унікальне, мікросервіс оновлює локальну базу векторів без явних сповіщень.

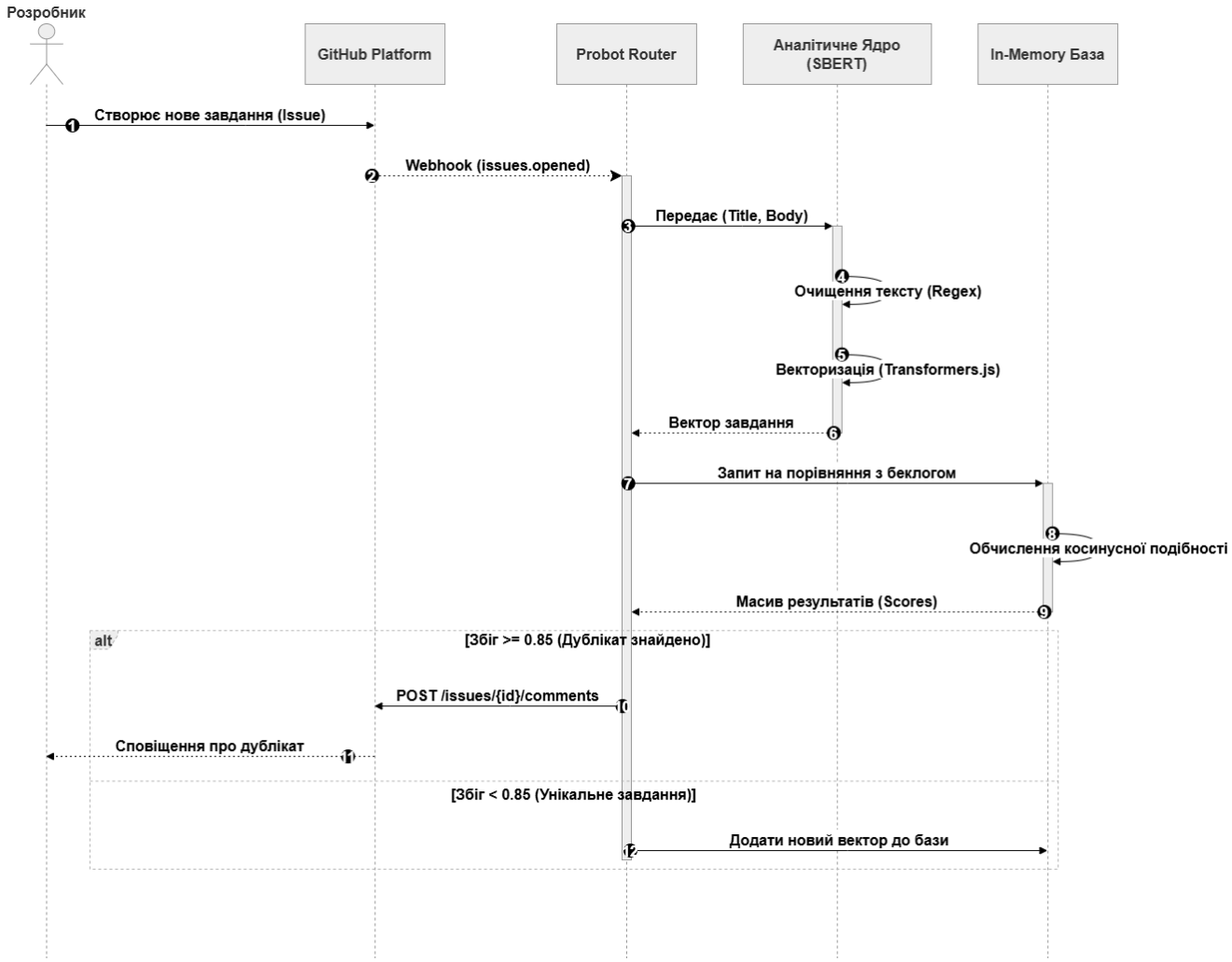

Рисунок 1 – UML-діаграма послідовності обробки нового завдання мікросервісом.

Аналіз результатів

Експериментальне тестування мікросервісу на наборі ретроспективних даних волонтерського ІТ-проекту показало, що близько 16,7% масиву завдань складали неявні дублікати. Порівняльний аналіз продемонстрував, що базова лексична модель схильна до хибних спрацювань на шаблонних описах (F1-score = 0.57). Натомість запропонована векторна модель забезпечила точність класифікації на рівні 91% та повноту 87% (F1-score = 0.89), практично повністю усунувши проблему спаму хибними попередженнями.

Висновки

У дослідженні запропоновано та практично реалізовано метод автоматизованої дедуплікації вимог у гнучких методологіях розробки. Розроблений на базі Node.js та Transformers.js мікросервіс дозволяє проактивноаналізувати проектний беклог, автоматично виявляючи семантичні дублікати в режимі реального часу. Впровадження такої системи мінімізує вплив "людського фактору", значно скорочує час команди на впорядкування вимог та запобігає нераціональному розподілу ресурсів розробників. У перспективі функціонал системи може бути розширено для автоматичної кластеризаціїбеклогу та автопризначення завдань (Auto-assign) на основі семантичного профілю розробників.

Список використаних джерел

1. GitHub REST API Documentation [Електронний ресурс]. – Режим доступу: https://docs.github.com/en/rest
2. Reimers, N., &Gurevych, I. Sentence-BERT: SentenceEmbeddingsusingSiamese BERT-Networks. InProceedingsofthe 2019 ConferenceonEmpiricalMethodsinNaturalLanguageProcessing (EMNLP-IJCNLP), 2019. – pp. 3982-3992.
3. Transformers.js: State-of-the-artMachineLearningfortheWeb [Електронний ресурс]. – Режим доступу: https://huggingface.co/docs/transformers.js
4. Probot: A frameworkforbuildingGitHubAppstoautomateandimproveyourworkflow [Електронний ресурс]. – Режим доступу: https://probot.github.io/

УДК 004.42

ВЕБ -ЗАСТОСУНОК ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ЗАВДАННЯМИ

Стельмащук А.Р.[1)], Тимчишин В.С.[2)]
Західноукраїнський національний університет
[1)] бакалавр; [2)] доктор філософії, ст. викладач

I. Постановка проблеми

У сучасних умовах динамічної цифровізації та постійного зростання обсягів вхідної інформації, традиційні методи планування та паперові носії втрачають свою ефективність. Більшість активних користувачів стикаються з критичною проблемою розсинхронізації завдань між різними пристроями та відсутністю гнучких адаптивних інструментів для оперативної пріоритезації справ. Існуючі програмні рішення часто мають надмірно перевантажений функціоналом інтерфейс, орієнтований на корпоративний сектор, або ж навпаки — мають обмежену підтримку персоналізованих категорій, що суттєво ускладнює процес швидкої фіксації та аналізу щоденних активностей. Відсутність адаптивного веб-застосунку, який би інтегрував у собі перевірені методики тайм-менеджменту (зокрема матрицю Ейзенхауера для визначення терміновості та важливості задач) у поєднанні з інтелектуальною системою миттєвих сповіщень, зумовлює гостру необхідність розробки нового програмного продукту, спрямованого на суттєве підвищення індивідуальної продуктивності та оптимізацію робочого часу користувача.

II. Мета роботи

Метою даної роботи є комплексне проєктування та програмна реалізація багаторівневого веб-застосунку для ефективного управління особистими завданнями. Система має забезпечувати повний цикл роботи із записами: створення, редагування, глибоку фільтрацію та багатозадачне категоріювання справ, встановлення жорстких та гнучких дедлайнів, а також динамічну візуалізацію прогресу виконання планів. Основний акцент при розробці зроблено на забезпеченні повної кросплатформенності через сучасні веб-браузери та створенні інтуїтивно зрозумілого UI/UX дизайну. Це дозволить мінімізувати когнітивне навантаження на користувача при роботі з інтерфейсом та скоротити час, необхідний для внесення нових записів у систему.

III. Основна частина

Для наочного представлення функціональних можливостей веб-застосунку та детального моделювання сценаріїв взаємодії (що включають процеси реєстрації, автентифікації, повнофункціонального керування завданнями та формування аналітичної статистики) було побудовано діаграму варіантів використання (див.рис.1). Дана діаграма дозволяє чітко визначити межі проектованої системи, основні ролі та права доступу користувачів до ключових функціональних модулів.

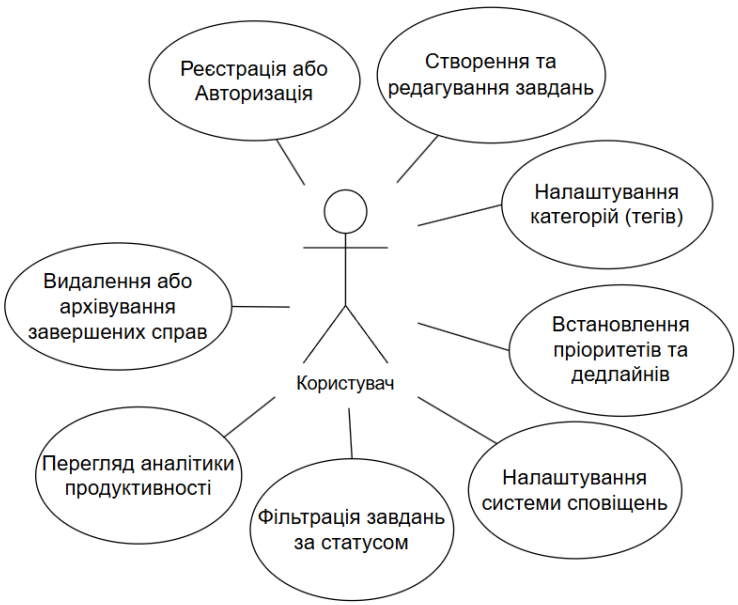

Рисунок 1 – Діаграма варіантів використання веб-застосунку для управління завданнями

Технічна реалізація веб-застосунку базується на сучасній клієнт-серверній архітектурі. Frontend-частина розробляється як Single Page Application (SPA), що дозволяє забезпечити високу швидкість відгуку інтерфейсу та безшовний перехід між розділами без повного перезавантаження сторінок. Backend-складова реалізує надійний програмний інтерфейс (API) для обробки вхідних запитів, валідації даних та взаємодії з реляційною базою даних. Такий підхід гарантує високу цілісність інформації та структурованість збережених даних. Логічну структуру системи, її внутрішні сутності та архітектурні взаємозв'язки відображено на діаграмі класів (див.рис.2).

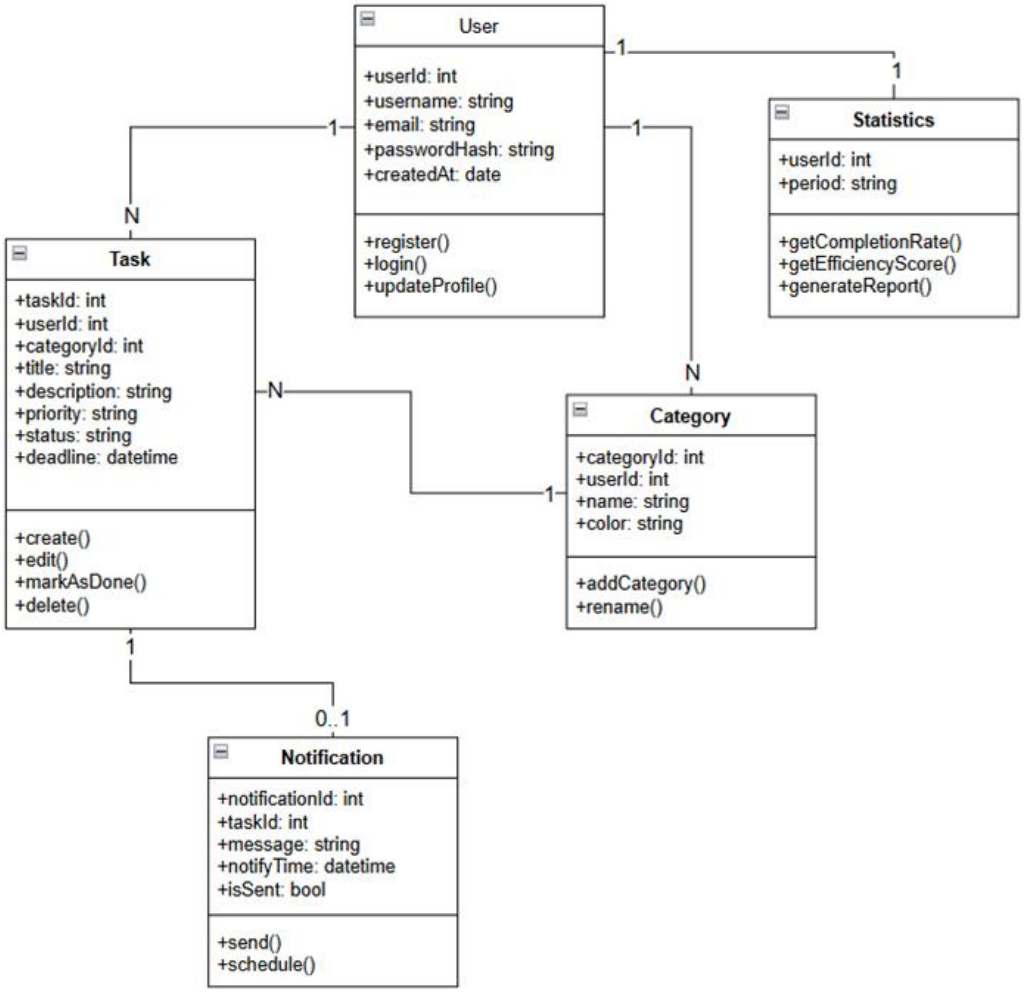

Рисунок 2 – Діаграма класів веб-застосунку для управління особистими завданнями

Центральним елементом є клас User, пов’язаний із об’єктами Task (завдання) та Category (категорії). Для контролю термінів впроваджено клас Notification, а для аналізу продуктивності – клас Statistics. Зв’язки типу «один-до-багатьох» забезпечують персоналізацію даних та логічну цілісність предметної області.

Висновки

У межах проведеного дослідження було розроблено цілісну концепцію та архітектурну модель веб-застосунку для управління завданнями з впровадженням розширених функцій пріоритезації. Реалізована система забезпечує надійне структуроване збереження інформації, автоматизоване нагадування про критичні часові межі та наочну візуалізацію індивідуальних показників продуктивності. Результати роботи демонструють ефективність обраного технологічного стеку для вирішення завдань тайм-менеджменту. Подальші дослідження будуть спрямовані на інтеграцію веб-застосунку із зовнішніми хмарними сервісами (Google Calendar, Outlook) та впровадження алгоритмів інтелектуального планування на основі пріоритетів користувача.

Список використаних джерел

1. Sommerville I. Software Engineering. – 10th ed. – Boston : Pearson Education, 2021. – 816 p.
2. Fowler M. Patterns of Enterprise Application Architecture. – Boston : Addison-Wesley, 2022. – 560 p.
3. Richards M., Ford N. Fundamentals of Software Architecture. – Sebastopol : O’Reilly Media, 2024. – 432 p.
4. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Prentice Hall, 2023. – 415 p.

УДК 004.8

АРХІТЕКТУРА СИСТЕМИ УПРАВЛІННЯ НАВЧАЛЬНИМИ КУРСАМИ З ВИКОРИСТАННЯМ МАТЕМАТИЧНИХ АЛГОРИТМІВ

Макарівський О.А.[1)], Папа О.А.[2)]
Західноукраїнський національний університет
[1)]магістрант;[2)]д.філ., доцент

Постановка проблеми

Сучасні системи управління навчальними процесами вимагають більшої гнучкості, адаптивності до потреб студентів та викладачів, а також інтеграції алгоритмічного аналізу освітніх даних. Зростання обсягів інформації про успішність, динаміку засвоєння матеріалу, активність у курсах створює передумови для використання математичних методів для автоматизованої оцінки та планування освітнього процесу. Постає потреба у створенні програмного забезпечення, що поєднує алгоритмічні підходи з зручним веб-інтерфейсом, забезпечуючи аналітичну підтримку прийняття рішень у сфері навчального менеджменту.

Мета дослідження

Метою роботи є проєктування архітектури програмного комплексу для управління навчальними курсами, що використовує математичні методи для автоматичного моніторингу успішності, адаптивного планування навчання та оцінювання результатів. Основна увага приділяється модульності, масштабованості та інтеграції алгоритмів у систему.

Структура системи

Розроблена система має багаторівневу модульну архітектуру, що забезпечує гнучкість, масштабованість і можливість адаптації під різні освітні контексти. Ця архітектура дозволяє ефективно обробляти великі обсяги даних, інтегрувати нові технології та забезпечувати високий рівень персоналізації для кожного користувача. Основними компонентами якої є:

1. Клієнтська частина (Frontend)реалізована з використанням ReactJS і Tailwind CSS, надає зручний адаптивний інтерфейс для користувачів. Включає панель керування курсами, візуалізацію навчальних графіків, динаміку оцінювання, сповіщення та віджети персональних рекомендацій.
2. Серверна частина (Backend API)виконана на Node.js з використанням Express.js. Здійснює обробку запитів, перевірку прав доступу, логіку обробки даних, взаємодію з базами даних та модулями обчислень. Забезпечує REST- або GraphQL-інтерфейс для взаємодії з клієнтами.
3. Математичне ядро системи складається з окремих сервісів, зокрема:
   - Модуль кластеризації групує студентів за рівнем знань, темпами навчання, стилем роботи.
   - Модуль прогнозуваннявикористовує методи регресії, інтервального аналізу для передбачення оцінок, ідентифікації студентів з ризиками відставання.
   - Модуль автоматизованого планування формує графіки подання матеріалів, завдань з урахуванням поточного прогресу.
   - Модуль рекомендаційгенерує підказки щодо додаткових джерел, завдань, тем для повторення.
4. Система управліннянданимиздійснює зберігання і обробку інформації про:
   - структуру курсів, викладачів, студентів;
   - подані завдання, оцінки, спроби виконання;
   - аналітичні звіти та рекомендації.

Для реалізації використовується PostgreSQL для реляційних даних і MongoDB – для документ-орієнтованих.

5. Модуль безпеки тааутентифікаціїреалізовує підтримку ролей (студент, викладач, адміністратор), JWT-аутентифікацію, контроль доступу до навчальних модулів і аналітики.
6. Інфраструктура розгортаннясистеми підтримує контейнеризацію за допомогою Docker та оркестрацію через Kubernetes, що дозволяє гнучко масштабувати окремі сервіси й

забезпечувати високу доступність. Також можлива інтеграція з хмарними платформами (AWS, Google Cloud).

Щоб спростити аналіз складових системи, на рисунку 1 представлено діаграму компонентів (UML Component Diagram), яка демонструє ключові елементи системи та зв'язки між ними.

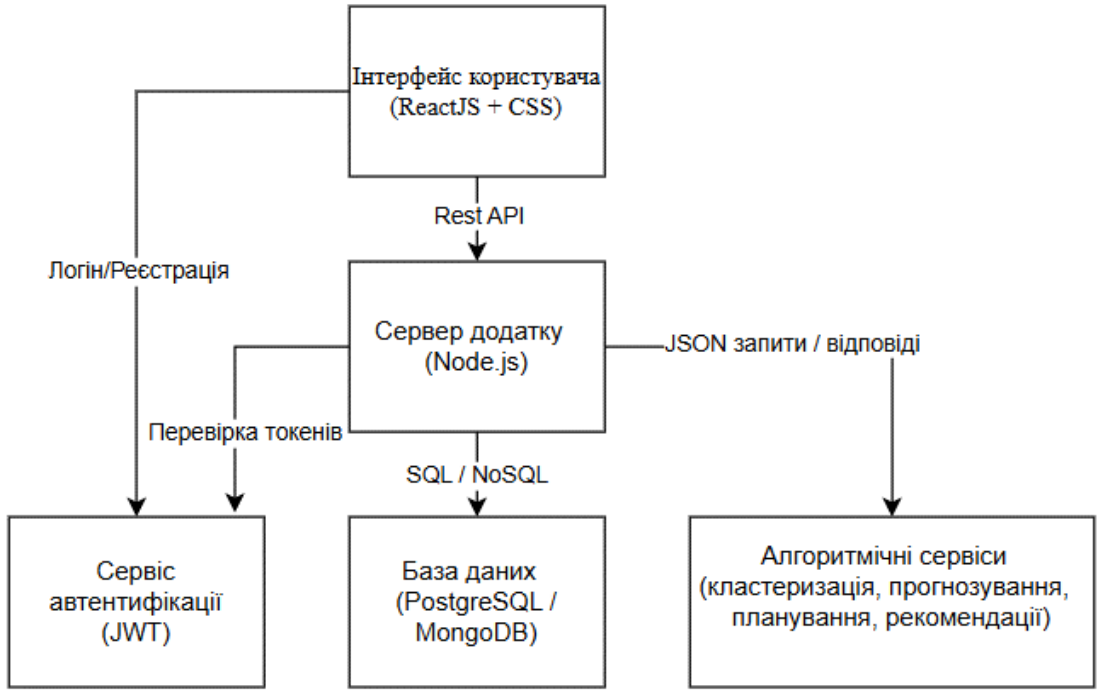

Рисунок 1 – Діаграма компонентів системи (UML ComponentDiagram).

Логіка взаємодії компонентів

Користувач працює через веб-інтерфейс, взаємодіючи із сервером, який у разі потреби ініціює математичні обчислення, звертаючись до відповідних модулів. Результати повертаються у вигляді аналітичних звітів, графіків та таблиць, що підвищують прозорість та ефективність навчального процесу.

Висновок

Запропонована архітектура системи управління навчальними курсами демонструє ефективне поєднання сучасних вебтехнологій і математичних методів аналізу даних для автоматизації освітнього процесу. Завдяки використанню модульного підходу до побудови системи, забезпечується її масштабованість, гнучкість у розширенні функціоналу, а також адаптація до різних організаційних потреб закладів освіти.Інтеграція алгоритмів кластеризації, прогнозування, персоналізованих рекомендацій та оцінювання дозволяє системі виступати не лише як адміністративний інструмент, але й як аналітична платформа підтримки прийняття рішень. Це сприяє більш точному виявленню слабких місць у навчанні, своєчасній реакції викладачів, формуванню індивідуальних траєкторій розвитку для кожного студента.Таким чином, розроблена архітектура закладає надійну основу для створення інтелектуальної інформаційної системи в галузі освіти, яка здатна не лише автоматизувати рутинні процеси, а й покращити якість навчання, орієнтуючись на дані та аналітику. Перспективним напрямом подальших досліджень є впровадження елементів адаптивного навчання на основі штучного інтелекту, а також тестування розробленої системи в реальних умовах освітнього середовища.

Список використаних джерел

1. ReactJS. A JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу: https://react.dev
2. MongoDB. The developer data platform [Електронний ресурс]. – Режим доступу: https://www.mongodb.com
3. Tailwind CSS. A utility-first CSS framework for rapid UI development [Електронний ресурс]. – Режим доступу: https://tailwindcss.com
4. PostgreSQL. The world's most advanced open source relational database [Електронний ресурс]. – Режим доступу: https://www.postgresql.org
5. Власенко Н.В., Жук Ю.О. Системи підтримки прийняття рішень в освітніх процесах: підходи та моделі // Інформаційні технології в освіті. – 2020. – № 38. – С. 115–122.

УДК 004.41

МОДЕЛЮВАННЯ ТА ОПТИМІЗАЦІЯ РОЗКЛАДІВ У ВЕБ-СИСТЕМАХ ІЗ ВИКОРИСТАННЯМ GOOGLE OR-TOOLS

Мацук А.Р.[1)], Гончар Л.І.[2)]
Західноукраїнський національний університет
[1)] аспірант; [2)] к.е.н., доцент

І. Постановка проблеми

У сучасних вебсистемах значна частина бізнес-логіки пов'язана не лише зі збереженням і відображенням даних, а й з автоматичним прийняттям рішень за великої кількості правил, винятків і взаємозалежних обмежень. Це характерно для систем бронювання, планування персоналу, призначення ресурсів, побудови розкладів, керування чергами та обробки заявок. Такі задачі належать до комбінаторних, оскільки кількість можливих варіантів швидко зростає разом із кількістю часових слотів, ресурсів, користувачів і додаткових умов.

На практиці подібні задачі часто розв'язуються вручну або за допомогою набору локальних правил у коді. Такий підхід працює лише для простих сценаріїв, але погано масштабується після появи нових бізнес-вимог. Унаслідок цього виникають конфлікти розкладу, нерівномірне навантаження на ресурси, дублювання логіки в різних частинах системи, зростання кількості винятків та погіршення якості сервісу для кінцевого користувача.

Актуальним є перехід від процедурного опису правил до формалізованої моделі обмежень, у якій система не перебирає сценарії вручну, а шукає допустиме або наближене до оптимального рішення в межах заданих умов. Одним із практичних інструментів для цього є Google OR-Tools - бібліотека для розв'язання задач комбінаторної оптимізації, маршрутизації, лінійного програмування та constraint optimization. Для задач планування особливий інтерес становить CP-SAT solver, який поєднує підходи constraint programming і SAT-пошуку.

ІІ. Мета роботи

Метою роботи є дослідження можливості застосування Google OR-Tools для автоматичного пошуку допустимих і наближених до оптимальних рішень у вебсистемах зі складними бізнес-правилами та обмеженнями, а також оцінювання придатності такого підходу для інтеграції в сучасну хмарну архітектуру.

Для досягнення мети необхідно формалізувати прикладну задачу побудови розкладу у вигляді набору сутностей, жорстких і м'яких обмежень; спроєктувати backend-сервіс оптимізації на платформі .NET із використанням Google OR-Tools; визначити спосіб інтеграції такого сервісу з вебзастосунком; а також окреслити критерії оцінювання якості отриманих рішень за допустимістю, часом обчислення та кількістю порушень м'яких обмежень.

ІІІ. Формалізація задачі та підхід до її розв'язання

У роботі розглядається приклад вебсистеми бронювання або планування ресурсів, у якій потрібно автоматично формувати розклад за участю кількох типів сутностей: часових слотів, ресурсів, працівників, типів послуг і користувацьких запитів. Така постановка є типовою для бізнес-застосунків, де один і той самий сервіс має враховувати доступність приміщень, обладнання, персоналу, часові обмеження клієнтів та внутрішні правила компанії.

До жорстких обмежень у такій системі належать відсутність перетину слотів, доступність ресурсу в заданий момент часу, робочі години, сумісність типу послуги з конкретним ресурсом, а також неможливість подвійного призначення одного працівника чи одного ресурсу на той самий часовий інтервал. Порушення таких умов робить розклад непридатним для використання. До м'яких обмежень належать побажання клієнта щодо часу, пріоритет окремих заявок, мінімізація простоїв між слотами, рівномірний розподіл навантаження та зменшення кількості ручних коригувань. Їх порушення не завжди робить розклад неможливим, але знижує його якість.

Для реалізації такого підходу доцільно використовувати CP-SAT solver з пакета Google OR-Tools. Згідно з офіційною документацією, цей solver працює з цілочисловими змінними, а результат розв'язання може мати стани OPTIMAL, FEASIBLE, INFEASIBLE, MODEL_INVALID або UNKNOWN. Це добре відповідає реальним бізнес-сценаріям: інколи системі достатньо швидко знайти допустимий розклад, а в інших випадках потрібно шукати кращий варіант у межах заданого ліміту часу. Дослідження CP-SAT-LP також показує, що сучасні SAT-орієнтовані методи мають високу практичну цінність для задач дискретної оптимізації .

Ключова перевага запропонованого підходу полягає в тому, що бізнес-правила описуються декларативно як змінні, обмеження та критерії якості, а не як велика кількість вкладених if-else конструкцій. Це зменшує зв'язність коду, спрощує перевірку коректності правил і дає змогу швидше адаптувати модель після зміни бізнес-вимог. Наприклад, додавання нового правила щодо мінімального проміжку між бронюваннями можна реалізувати як окреме обмеження моделі, а не як зміну кількох незалежних гілок процедурної логіки.

З архітектурного погляду рішення може бути реалізоване як окремий optimization service у складі застосунку. Такий сервіс приймає вхідні дані через API, виконує валідацію, перетворює дані на модель OR-Tools, запускає розв'язання та повертає результат у вигляді запропонованого розкладу або пояснення, чому допустиме рішення не знайдено. Для складніших сценаріїв доцільно запускати оптимізацію у фоновому режимі через чергу повідомлень або background worker, щоб не блокувати основний користувацький запит.

У хмарному середовищі такий сервіс повинен враховувати типові вимоги до надійності: таймаути, повторну обробку повідомлень, ідемпотентність операцій, журналювання вхідних даних, контроль версій моделі та можливість відновлення після збою. Рекомендації Microsoft щодо reliability patterns в Azure підкреслюють важливість таких архітектурних рішень для систем, які мають стабільно працювати під навантаженням і коректно обробляти тимчасові помилки інфраструктури .

Практичне оцінювання запропонованого підходу можна виконувати шляхом порівняння з базовим евристичним алгоритмом або ручним сценарієм планування. Основними критеріями є частка задач, для яких знайдено допустимий розклад; середній час побудови рішення; кількість порушень м'яких обмежень; рівень завантаження ресурсів; кількість ручних коригувань після автоматичного планування; а також стійкість системи до збільшення кількості заявок і правил. Подібні критерії узгоджуються з сучасними дослідженнями задач планування, де constraint programming застосовується для підвищення надійності та якості розподілу завдань.

Окрему увагу слід приділити пояснюваності результату для користувача. Якщо solver не знаходить допустимого рішення, система має не лише повернути технічний статус INFEASIBLE, а й показати, які групи обмежень найімовірніше спричинили конфлікт. У прикладній системі це може бути повідомлення про нестачу доступних ресурсів, перетин робочих годин або надто вузьке вікно бронювання. Такий підхід робить optimization service не просто алгоритмічним модулем, а частиною зрозумілого бізнес-процесу.

Запропонована модель має і певні обмеження. Якість результату залежить від коректності вхідних даних, чіткості опису бізнес-правил і вибраного ліміту часу на пошук. Крім того, занадто складна модель може збільшити час розв'язання або призвести до ситуації, коли допустиме рішення не буде знайдено в межах заданого часу. Тому в реальному продукті доцільно поєднувати CP-SAT з попередньою валідацією даних, кешуванням типових сценаріїв і fallback-логікою для випадків, коли автоматична оптимізація не дала результату.

Висновок

У роботі обґрунтовано доцільність використання Google OR-Tools для задач планування й оптимізації у вебсистемах зі складними бізнес-правилами. Показано, що формалізація задачі у вигляді набору жорстких і м'яких обмежень дає можливість відмовитися від надмірно ускладненої процедурної логіки та перейти до більш прозорої, масштабованої й контрольованої моделі прийняття рішень.

Використання CP-SAT solver є особливо перспективним для систем бронювання, планування персоналу та керування ресурсами, де важливо одночасно забезпечити допустимість рішень і високу якість розкладу. Інтеграція такого підходу у вигляді окремого optimization service дозволяє поєднати алгоритмічну оптимізацію з практичними вимогами вебархітектури: API-інтеграцією, фоновою обробкою, журналюванням, масштабуванням і стійкістю до збоїв. Подальший розвиток дослідження доцільно спрямувати на експериментальну перевірку моделі на реальних даних бронювання та порівняння з класичними евристичними алгоритмами.

Список використаних джерел

1. Google for Developers. OR-Tools. URL: https://developers.google.com/optimization.
2. Perron L., Didier F., Gay S. The CP-SAT-LP Solver. 29th International Conference on Principles and Practice of Constraint Programming (CP 2023). 2023.
3. Cho J., Jung S., Yang K., Kim D., Kim W. Efficient Task Scheduling Using Constraints Programming for Enhanced Planning and Reliability. Applied Sciences. 2024. Vol. 14, No. 23. Art. 11396.
4. Microsoft Learn. Architecture design patterns that support reliability. URL: https://learn.microsoft.com/en-us/azure/well-architected/reliability/design-patterns.

УДК 004.4

РЕАЛІЗАЦІЯ ДИНАМІЧНОГО ІНТЕРФЕЙСУ КОРИСТУВАЧА ДЛЯ БАГАТОКРИТЕРІАЛЬНОГО ПІДБОРУ ЗАРЯДНИХ СТАНЦІЙ ЗА ДОПОМОГОЮ NO-CODE ІНСТРУМЕНТІВ

Сімак А.Ю.[1)], **Кобилан В.С.**[2)]
Західноукраїнський національний університет
[1)]д. філ., викладач; [2)] магістрант

I. Постановка проблеми

Створення сучасних систем підтримки прийняття рішень вимагає розробки інтерактивних інтерфейсів, здатних обробляти значну кількість вхідних параметрів у реальному часі. Традиційна фронтенд-розробка багатокритеріальних форм підбору зазвичай передбачає написання великого обсягу JavaScript-коду для управління станом сторінки та взаємодії з API. Проте використання No-code платформ дозволяє значно прискорити процес створення адаптивних інтерфейсів, зберігаючи при цьому високу функціональність та динамічність взаємодії з користувачем.

II. Мета роботи

Метою дослідження є обґрунтування та реалізація динамічного інтерфейсу користувача для веб-системи автоматизованого підбору зарядних станцій за допомогою інтеграції інструментів Webflow та Wized без необхідності ручного програмування логіки на стороні клієнта.

III. Метод вирішення задачі

Для реалізації динамічної взаємодії було вирішено використати поєднання двох No-code інструментів (Webflow та Wized). Перший використовується як візуальний редактор для побудови структури сторінок, дизайну форм та забезпечення адаптивності для різних типів пристроїв. Платформа генерує чистий код, що сприяє високій швидкості завантаження та SEO-оптимізації. Другий застосовується для зв'язування елементів інтерфейсу з серверною частиною через REST API. Це дозволяє налаштовувати логіку обробки подій, валідацію полів та відображення результатів багатокритеріального аналізу в реальному часі без перезавантаження сторінки.

Загальна компонентна архітектура реалізованої системи та схема розподілу відповідальності між шаром представлення (Webflow), проміжним програмним забезпеченням (Wized) та сервером представлена на рисунку 1.

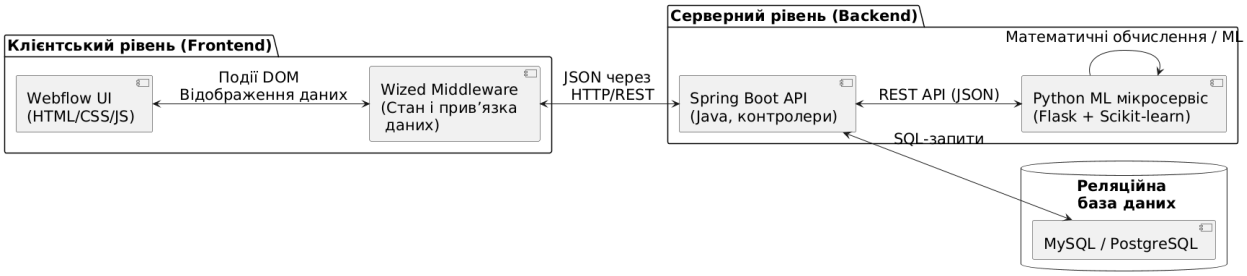

Рисунок 1 – UML-діаграма компонентної архітектури веб-системи

IV. Особливості програмної реалізації

Програмна реалізація клієнтської частини системи базується на використанні no-code платформи Webflow, що дозволило зосередитися на візуальному проектуванні та користувацькому досвіді (UX), автоматично генеруючи чистий, оптимізований для SEO та адаптивний код. Процес розробки включав декілька ключових етапів:

- Проектування інтерактивних сторінок з модульною структурою, що включає головну сторінку з точкою входу, сторінку переліку станцій із застосуванням карткового підходу та сторінку «AI Helper» для введення параметрів користувача. Кожна картка станції містить динамічні елементи: назву, рейтинг та кнопку «ViewDetails» для перегляду розширених характеристик у модальних вікнах.
- Інтеграція логіки через інструмент Wized для зв'язку фронтенду з бекендом на JavaSpring. Це дозволило налаштувати передачу даних у форматі JSON через REST API без написання ручного JavaScript-коду. Зокрема, реалізовано динамічну обробку подій для форми підбору, де параметри (ціна, потужність, підтримка сонячних панелей тощо) відправляються на сервер для математичного ранжування, а результати миттєво відображаються на сторінці без її перезавантаження.

- Створення функціонал адміністратора. Реалізовано захищену панель із вбудованою системою управління контентом (CMS). Вона містить динамічну таблицю для виводу всіх наявних у базі MySQL станцій з можливістю їх редагування або видалення безпосередньо через веб-інтерфейс.
- Забезпечення оптимізації та продуктивності. На відміну від традиційних фреймворків (React або Angular), обраний підхід дозволив скоротити час розробки MVP, забезпечивши при цьому стабільну роботу інтерфейсу.

Взаємодія між фронтендом та бекендом побудована на принципах розподілу обов'язків, де клієнт відповідає виключно за збір даних та візуалізацію, а вся складна бізнес-логіка та обробка запитів до бази даних MySQL виконується на стороні сервера.Процес асинхронної взаємодії користувацького інтерфейсу з сервером, включаючи прив'язку даних та оновлення стану DOM-елементів через інструмент Wized, представлено на рисунку 2.

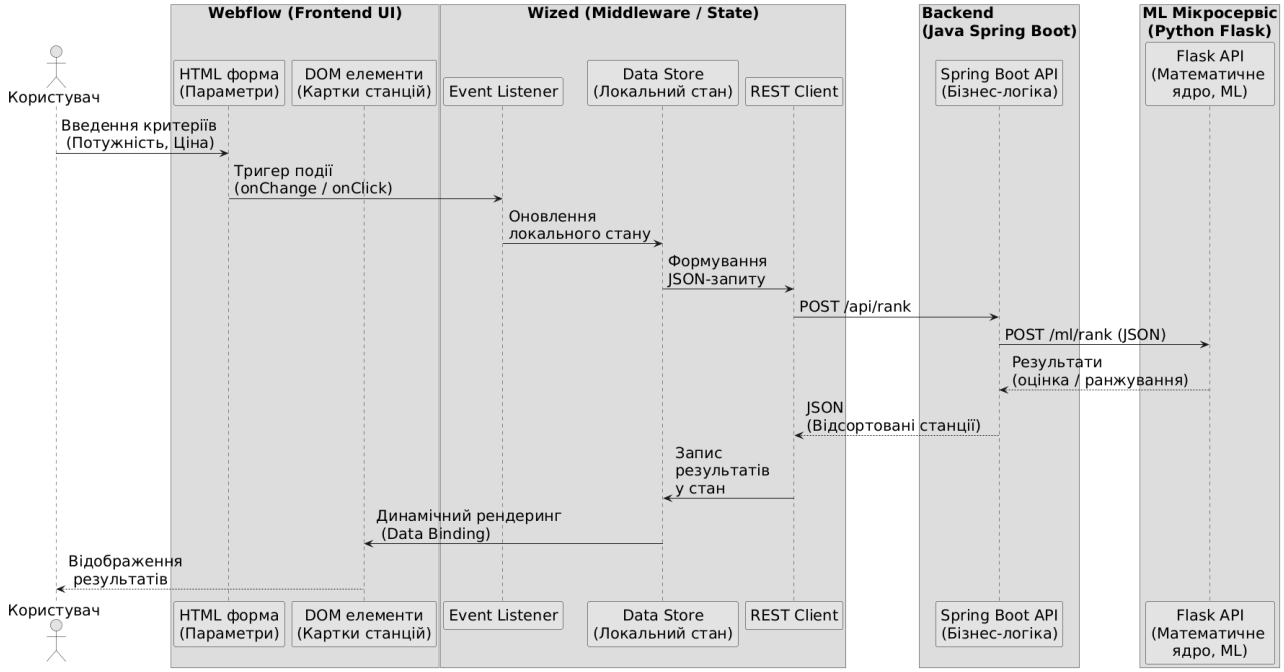

Рисунок 2 – Діаграма послідовності обробки користувацького запиту та рендерингу результатів

В загальному архітектура базується на інтеграції JavaSpringBoot (основний бекенд) та ізольованого Python (Flask) мікросервісу для математичних обчислень. SpringBoot керує бізнес-логікою, базою MySQL та маршрутизацією, тоді як Flask виконує складне ранжування й аналіз. Взаємодія через REST API (JSON) забезпечує слабке зв’язаність, що полегшує масштабування та модернізацію окремих модулів.

Висновок

Реалізація інтерфейсу на базі Webflow та Wized підтвердила ефективність No-code підходу для створення складних систем підбору зарядних станцій. Таке рішення дозволило впровадити принцип розподілу обов'язків, де шар представлення повністю ізольований від бізнес-логіки. Використання інструменту Wized як проміжного програмного забезпечення забезпечило стабільний асинхронний зв’язок із бекендом на базі JavaSpringBoot через REST API, який, у свою чергу, взаємодіє з окремим Python-мікросервісом (Flask), що реалізує математичні обчислення та моделі машинного навчання. Така архітектура гарантує високу швидкість рендерингу результатів без перезавантаження сторінок. Це, в цілому, дозволяє зосередити ресурси на розробці складного математичного апарату, забезпечуючи при цьому професійний рівень UX/UI та масштабованість системи.

Список використаних джерел

1. Смучок Д. С. Платформи no-code та low-code: магістерська дип.:121 Інженерія програмного забезпечення. – Київ, 2023. С.12–13.
2. Marcotte E. ResponsiveWebDesign, A BookApart, 2011. Pages 7–18.
3. Василенко В. М., Карпенко М. І., Скибінський А. С., Гуйда О. Г. Аналіз переваг та обмежень low-code платформ на прикладі Webflow.com // Вчені записки ТНУ імені В. І. Вернадського. 2024. С. 65–70.
4. OpenApplicationsGroup. RESTfulWeb API Design Standard, 2018.Pages 11–23.
5. Офіційна документація та навчальні матеріали Webflow. [Електроннийресурс]. – Режим доступу: https://university.webflow.com. 2024.
6. Офіційна документація платформи Wized. [Електроннийресурс]. – Режим доступу: https://docs.wized.com. 2024.

УДК 004.415.2

ВЕБ-СИСТЕМА УПРАВЛІННЯ СЕРВІСНИМИ ЗАПИТАМИ З ПІДТРИМКОЮ БАГАТОКЛІЄНТНОСТІ ТА КОНТРОЛЮ SLA

Войтюк І.Ф.[1)], Барилко М.В.[2)]
Західноукраїнський національний університет,
[1)]к.т.н., доцент; [2)]студент

I. Анотація

У роботі розглядається проєктування та програмна реалізація вебсистеми управління сервісними запитами клієнтів у сфері надання IT-послуг. Описано архітектурні рішення щодо підтримки багатоклієнтного режиму роботи з логічною ізоляцією даних, рольового розмежування доступу та автоматизованого контролю угод про рівень обслуговування.

II. Постановка проблеми

Сучасні IT-компанії, що надають сервісні послуги малому та середньому бізнесу, стикаються з проблемою відсутності централізованого інструменту обліку клієнтських звернень. За відсутності спеціалізованої системи обробка запитів здійснюється через електронну пошту, месенджери або таблиці, що призводить до накопичення повторюваних звернень, втрати контексту взаємодії та неможливості контролювати відповідальність виконавців і дотримання регламентних термінів.

Окремою проблемою є масштабування процесу підтримки в умовах обслуговування кількох організацій одночасно. За відсутності механізму логічної ізоляції даних існує ризик витоку конфіденційної інформації між клієнтами, що є критичним у B2B-середовищі (міжкорпоративної взаємодії).

Аналіз існуючих рішень — Jira Service Management, Zendesk, Freshdesk — показав, що більшість із них орієнтовані на великі корпоративні (enterprise) команди, мають перевантажений інтерфейс та потребують значних витрат на впровадження й адміністрування. Це зумовлює доцільність розробки власного програмного продукту, адаптованого до потреб компактних IT-сервісних команд із чітко визначеними процесами.

III. Мета та завдання роботи

Метою роботи є проєктування та програмна реалізація вебсистеми управління сервісними запитами клієнтів, яка забезпечує централізований облік звернень, підтримку багатоклієнтного режиму роботи з логічною ізоляцією даних організацій, рольове розмежування доступу та автоматизований контроль дотримання угод про рівень обслуговування.

Для досягнення мети визначено такі завдання:

- проаналізувати існуючі підходи до організації багатоклієнтності та управління SLA в IT-сервісних системах;
- спроєктувати архітектуру вебсистеми з урахуванням вимог до ізоляції даних та продуктивності;
- реалізувати серверну та клієнтську частини системи;
- перевірити коректність роботи механізмів контролю SLA та рольового доступу.

IV. Архітектура та проєктні рішення

Система реалізована за трирівневою клієнт-серверною архітектурою. Рівень подання побудовано на основі фреймворку Next.js з використанням TypeScript, що забезпечує статичну типізацію, компонентний підхід та швидкий серверний рендеринг. Серверну частину реалізовано у вигляді *stateless*(без збереження стану)REST API мовою програмування Go, що забезпечує високу продуктивність обробки HTTP-запитів. Для зберігання даних обрано СУБД PostgreSQL з нормалізованою реляційною схемою, що охоплює такі основні сутності: *Organization*, *User*, *Ticket*, *Comment*, *Attachment*, *SLA*.

На рівні API реалізовано проміжний шар (*middleware*), який отримує ідентифікатор організації з JWT та застосовує його як фільтр до SQL-запитів.

Модель ролей включає чотири рівні: *Client*, *Operator*, *Engineer*, *Admin* — з різними правами на читання та модифікацію даних.

Життєвий цикл заявки визначається скінченним автоматом із п'ятьма станами: *New* (нова), *In*

Progress (в роботі), *Resolved* (вирішена), *Closed* (закрита), *Reopened* (відновлена).

V. Контроль SLA

Для контролю угод про рівень обслуговування (SLA – Service LevelAgreement) в системі фіксуються часові мітки $created_{at}$ і $resolved_{at}$, а також інтервали призупинення обробки – час перебування заявки у стані Waiting for customer (очікування відповіді клієнта). Фактичний розрахунковий час вирішення визначається з урахуванням робочого графіку та пауз:

$$T_{fact} = resolved_{at} - created_{at} - T_{pause} - T_{non-business} \quad (1)$$

де T_{pause} – сумарний час очікування відповіді клієнта; $T_{non-business}$ – час поза межами робочого графіку (8×5 або 24×7 залежно від договору з клієнтом).

Це значення порівнюється з допустимим порогом T_{sla}, визначеним для пріоритету заявки (High, Medium, Low):

$$SLA_{status} = \begin{cases} Met, & якщо \quad T_{fact} \le T_{sla} \\ Breached, & якщо \quad T_{fact} > T_{sla} \end{cases} \quad (2)$$

Для ранньої індикації потенційних порушень обчислюється коефіцієнт використання SLA-бюджету:

$$K_{sla} = \frac{T_{elapsed}}{T_{sla}} \times 100\% \quad (3)$$

де $T_{elapsed}$ – час, що минув з моменту створення заявки за вирахуванням пауз та неробочих годин. При $K_{sla} \ge 80\%$ система візуально виділяє заявку в інтерфейсі оператора як таку, що наближається до порушення регламенту, що дозволяє своєчасно вжити заходів.

VI. Інтерфейс користувача

Клієнтський інтерфейс реалізовано з використанням бібліотеки компонентів shadcn/ui на основі Radix UI, що забезпечує адаптивність та семантичну доступність елементів. Інтерфейс диференційований залежно від ролі авторизованого користувача. Клієнту доступна форма створення заявок із валідацією вкладень та перелік власних звернень із поточними статусами. Оператору та інженеру надається зведена таблиця заявок усіх організацій із гнучким фільтруванням за статусом, пріоритетом та організацією, а також візуальна індикація SLA. Адміністратор має доступ до панелі керування користувачами, організаціями та налаштування політик SLA.

Висновок

У роботі спроєктовано та розроблено вебсистему управління сервісними запитами клієнтів у сфері надання IT-послуг, що поєднує підтримку багатоклієнтного режиму роботи, рольове розмежування доступу та автоматизований контроль SLA з урахуванням робочого графіку і пауз обробки. Запропонована архітектура на основі Next.js та REST API мовою Go забезпечує масштабованість і безпеку ізоляції даних.

Список використаних джерел

1. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 560 p.
2. Jones M. B., Bradley J., Sakimura N. RFC 7519: JSON WebToken (JWT). IETF, 2015. URL: https://www.rfc-editor.org/rfc/rfc7519 (дата звернення: 20.04.2026).
3. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : дис. … д-ра філос. / University of California. Irvine, 2000. 162 p.
4. Office of Government Commerce. ITIL v3: Service Operation. London : TSO, 2007. 262 p.
5. Jira Service Management ProductGuide. Atlassian. URL: https://www.atlassian.com/software/jira/service-management
6. Next.js Documentation. Vercel. URL: https://nextjs.org/docs
7. Мартин Р. Чиста архітектура. Мистецтво розробки програмного забезпечення. Харків : Фабула, 2019. 352 с.

УДК 004.8

ВИКОРИСТАННЯ NLP-ЕМБЕДИНГІВ ТА МЕТОДІВ ЗНИЖЕННЯ РОЗМІРНОСТІ ДЛЯ СЕМАНТИЧНОГО КОДУВАННЯ НАЗВ ІВЕНТІВ У ЗАДАЧАХ ПРОГНОЗУВАННЯ

Мельник А.М.[1)], Пукас Б.І.[2)]
Західноукраїнський національний університет
[1)]д.т.н., професор; [2)] магістрант

I. Постановка проблеми

Прогнозування відвідуваності масових заходів ускладнюється обмеженістю структурованих даних щодо маркетингової активності, вартості квитків та характеристик аудиторії. У таких умовах назва події є важливим джерелом непрямої інформації, що може відображати її масштаб, тип аудиторії та популярність.

Однак назви подій мають специфічні властивості: вони зазвичай є унікальними або майже унікальними, що унеможливлює їх ефективне використання як звичайних категоріальних ознак. Застосування підходів кодування категоріальних змінних, таких як one-hotencoding або targetencoding, у цьому випадку є неефективним.

Зокрема, one-hotencoding призводить до формування надзвичайно розріджених і високорозмірних векторів, де кожна назва події розглядається як незалежна категорія без урахування змісту. Targetencoding, у свою чергу, вимагає достатньої кількості повторюваних значень, що не виконується для унікальних назв, і може призводити до перенавчання.

Таким чином, виникає необхідність використання підходів, які дозволяють враховувати семантичний зміст тексту, а не лише його формальне представлення.

II. Мета роботи

Підвищення точності прогнозування відвідуваності подій шляхом використання семантичного кодування назв заходів із застосуванням трансформерних моделей обробки природної мови та оптимізації ознакового простору

III. Основна частина

Існує декілька основних підходів до кодування текстових даних:

- One-hotencoding — формує розріджені вектори, не враховує подібність між текстами;
- Targetencoding — ефективний для повторюваних категорій, але непридатний для унікальних значень;
- Bag-of-Words — ігнорує порядок слів і контекст;
- TF-IDF — враховує частоту слів, але не їх семантику;
- Word2Vec — формує щільні вектори, але на рівні окремих слів, без урахування повного контексту.

На відміну від цих підходів, сучасні трансформерні моделі, зокрема SentenceTransformers, дозволяють отримувати щільні векторні представлення цілих речень (sentenceembeddings), які враховують порядок слів, контекст та приховані семантичні зв'язки.

Для отримання векторів використано архітектуру трансформера, що базується на механізмі самої уваги (Self-Attention). Математично це реалізується через обчислення ваг між векторами запиту (Q), ключа (K) та значення (V):

$$Attention(Q,K,V) = softrnax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

де d_k – розмірність векторів ключа.

Це дозволяє моделі динамічно визначати важливість кожного слова в контексті всієї назви події.

У бібліотеці sentence_transformers (Python) реалізовано підхід Siamese BERT-networks. Використана модель all-MiniLM-L6-v2 пропускає текст через рівні трансформера, після чого застосовує операцію пулінгу (meanpooling) для отримання єдиного вектора речення розмірністю 384.

Це забезпечує розташування семантично близьких івентів (наприклад, "RockFest" та "MusicConcert") поруч у векторному просторі. Експеримент проведено на наборі даних із 32 380 подій.

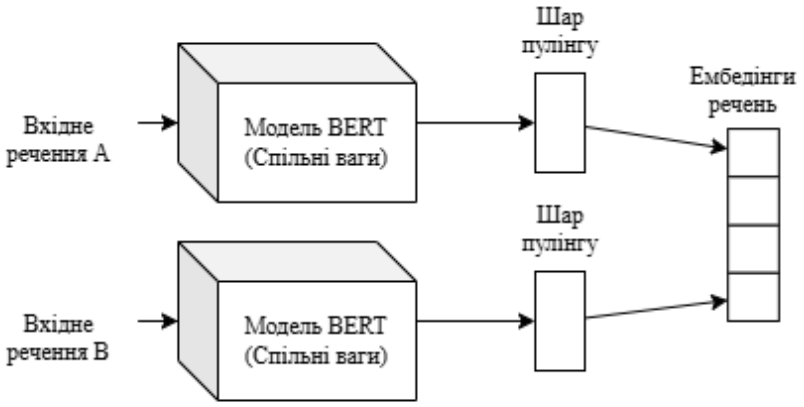

Рисунок 1 – Архітектура сіамської BERT моделі

Однак використання високорозмірних векторів призводить до:

- зростання обчислювальної складності;
- ризику перенавчання (curse of dimensionality);
- наявності надлишкових і корельованих ознак.

Для усунення цих недоліків застосовано Principal Component Analysis, який дозволяє виділити найбільш інформативні компоненти. Цей метод Principal Component є статистичним підходом до зниження розмірності даних, який полягає у перетворенні вихідного набору корельованих ознак у новий набір некорельованих змінних — головних компонент. Кожна компонента є лінійною комбінацією початкових ознак та впорядковується за величиною поясненої дисперсії. Перша головна компонента зберігає максимальну частку варіації даних, друга — максимальну частку залишкової варіації за умови ортогональності до першої, і так далі. Зменшення розмірності досягається шляхом відбору перших k компонент, які зберігають найбільшу частину інформації (дисперсії) вихідного простору. Такий підхід дозволяє усунути надлишкові та корельовані ознаки, зменшити шум у даних та підвищити стійкість моделей машинного навчання.

У результаті застосування PCA:

- розмірність зменшено з 384 до 64;
- збережено 73.71% дисперсії;
- знижено шум і надлишковість ознак.

Навчений перетворювач PCA було збережено для забезпечення узгодженості обробки даних під час використання моделі.

Отримані ознаки використовуються у моделі XGBoost разом із часовими, географічними та метеорологічними параметрами.Проведені експерименти підтвердили ефективність використання семантичних ознак у задачі прогнозування відвідуваності.Використання embedding-представлень дозволило врахувати семантичний зміст назв подій, що є критично важливим для задачі з унікальними текстовими значеннями. У порівнянні з one-hot, targetencoding та TF-IDF, запропонований підхід забезпечує кращу узагальнюючу здатність моделей за рахунок врахування контексту.

Висновки

Застосування трансформерних моделей у поєднанні з методами зниження розмірності дозволяє ефективно вирішити проблему кодування унікальних назв подій, забезпечуючи високу узагальнюючу здатність системи порівняно з традиційними статистичними підходами. Завдяки впровадженню методу аналізу головних компонент (PCA) вдалося досягти шестикратного зменшення розмірності ознакового простору при збереженні понад 73% корисної інформації, що гарантує оптимальний баланс між прогнозною точністю моделі та швидкістю її обчислювального виконання.

Список використаних джерел

1. SentenceTransformers Documentation [Електронний ресурс] – Режим доступу: https://sbert.net/
2. Principal Component Analysis (PCA) [Електронний ресурс] – Режим доступу: https://www.geeksforgeeks.org/data-analysis/principal-component-analysis-pca/

УДК 004.41:336.76

ПОРІВНЯЛЬНИЙ АНАЛІЗ МАТЕМАТИЧНИХ МОДЕЛЕЙ ДЛЯ МОНІТОРИНГУ І ПРОГНОЗУВАННЯ АКЦІЙ НА БІРЖІ

Войтюк І.Ф.[1)], Коваль М.В.[2)]
Західноукраїнський національний університет
[1)]к.т.н. доцент; [2)]магістрант

Постановка проблеми

Фондовий ринок є одним із найскладніших об'єктів математичного моделювання: ціни акцій залежать від макроекономічних показників, новинного фону, поведінки інвесторів та численних прихованих факторів. Традиційні підходи до аналізу біржових даних нерідко виявляються недостатньо точними через нелінійність та нестаціонарність фінансових часових рядів, наявність різких стрибків і тривалих трендів, що змінюються у непередбачуваний спосіб.

Сучасні системи моніторингу акцій мають забезпечувати прогнозування цінової динаміки в режимі реального часу, оцінку ринкових ризиків та підтримку інвестиційних рішень. Проте відсутність єдиного критерію вибору математичної моделі ускладнює проектування програмного забезпечення: різні моделі демонструють суттєво відмінну точність залежно від горизонту прогнозу, режиму ринку та обсягу доступних даних. Це породжує актуальну науково-практичну проблему – обґрунтований вибір і порівняння математичних моделей для задач моніторингу акцій на біржі.

Мета роботи

Метою роботи є систематизація та порівняльний аналіз основних математичних моделей прогнозування цін акцій – ARIMA, LSTM та Transformer – за критеріями математичного апарату, точності прогнозування (RMSE), обчислювальної складності та практичної придатності до інтеграції у системи моніторингу фондового ринку в режимі реального часу.

Обґрунтування отриманих результатів

Для досягнення поставленої мети проведено теоретичний аналіз математичного апарату кожної моделі та виконано порівняння на основі відносної похибки RMSE, зафіксованої у відкритих дослідженнях на даних індексів S&P 500, NASDAQ та окремих компаній (горизонт прогнозу – 1–5 торгових днів). Модель ARIMA(p,d,q) після d-кратного диференціювання ряду апроксимує динаміку ціни акції авторегресійною складовою порядку p та ковзним середнім порядку q. Модель ефективна на коротких горизонтах прогнозування для стаціонарних рядів і обчислюється на CPU за мілісекунди, що робить її придатною для систем реального часу. Водночас лінійна природа ARIMA не дозволяє виявляти нелінійні залежності та різкі цінові стрибки, характерні для біржових даних, тому модель використовується передусім як базовий орієнтир (benchmark). LSTM – рекурентна нейронна мережа зі спеціальними вентилями вхідних, вихідних даних та забування – долає проблему затухаючого градієнту і здатна виявляти нелінійні довгострокові залежності у фінансових часових рядах. Завдяки механізму пам'яті мережа враховує події, що відбулися кілька тижнів тому, що суттєво підвищує точність прогнозу на середньому горизонті. Основні недоліки: схильність до перенавчання на малих вибірках та значний час навчання, що потребує GPU-прискорення.Архітектура Transformer базується на механізмі самоуваги, що дозволяє одночасно зіставляти всі часові кроки послідовності без послідовного обчисленняна етапі навчання. Це усуває обмеження рекурентних мереж щодо довжини контексту та забезпечує ефективне паралельне навчання. За результатами порівняння Transformer досягає найнижчого RMSE, що на 43% менше за показник ARIMA і на 14% менше за LSTM, однак вимагає значно більших обсягів навчальних даних та обчислювальних ресурсів.

Таблиця 1

Порівняльна характеристика математичних моделей прогнозування акцій

Модель	Матем. основа	Переваги	Недоліки	RMSE (відн.)
ARIMA	ARMA + інтегрування рядів	Простота, інтерпретованість, стаціонарні ряди	Лінійність, ігнорує волатильність	~2.1%
LSTM	Рекурентна нейронна мережа	Нелінійні залежності, довга пам'ять	Перенавчання, велика обчислювальна складність	~1.4%
Transformer	Механізм самоуваги (self-attention)	Паралельне навчання, великий контекст	Потребує великих датасетів	~1.2%

Недоліки моделей та шляхи їх усунення

Незважаючи на широке застосування, кожна з розглянутих моделей має характерні обмеження. Модель ARIMA передбачає лінійність залежностей та стаціонарність ряду, що суперечить реальній динаміці біржових цін з їх нелінійними стрибками та змінними трендами. Усунення цього недоліку досягається шляхом попередньої декомпозиції ряду (STL-декомпозиція), автоматичного підбору порядків моделі за критерієм AIC/BIC, а також комбінування ARIMA з нейромережевими методами для моделювання нелінійних залишків. Архітектура LSTM схильна до перенавчання на малих вибірках та потребує тривалого часу навчання. Для подолання перенавчання застосовуються регуляризація методом dropout (відключення випадкових нейронів з імовірністю 0.2–0.5), рання зупинка навчання (earlystopping) та збільшення обсягу даних через аугментацію часових рядів. Проблему обчислювальної складності вирішує квантизація моделі та використання легших архітектур – наприклад, TCN (TemporalConvolutionalNetwork) як швидша альтернатива LSTM.Transformer вимагає великих навчальних датасетів і значних обчислювальних ресурсів, що ускладнює його застосування в умовах обмеженої інфраструктури. Ці обмеження долаються через трансферне навчання (fine-tuning попередньо навченої моделі на фінансових даних), застосування компактних архітектурInformer або PatchTST, що оптимізовані для часових рядів, а також через дистиляцію знань (knowledge distillation) – передачу поведінки великої моделі у легшу.Як основну метрику порівняння моделей у таблиці 1 обрано середньоквадратичну похибку (RMSE), яка є чутливою до великих відхилень і вважається ключовим критерієм у задачах фінансового прогнозування, де різкі цінові стрибки несуть найбільший ризик для інвестора. Достовірність висновків додатково перевірено результатами цитованого емпіричного дослідження за двома допоміжними метриками – середньою абсолютною похибкою (MAE), що забезпечує рівномірне зважування всіх помилок і відображає середню точність на спокійному ринку, та середньою абсолютною відсотковою похибкою (MAPE), яка дозволяє порівнювати моделі між різними активами незалежно від масштабу цін. Відносна впорядкованість моделей за точністю (Transformer< LSTM < ARIMA) залишається стабільною за всіма трьома метриками, що підтверджує надійність зробленого висновку.Аналіз даних таблиці 1 свідчить про існування компромісу між точністю та обчислювальною складністю. Нейромережеві методи (LSTM, Transformer) забезпечують найкращу точність прогнозування, проте потребують GPU-прискорення для навчання та інференсу. Статистична модель ARIMA значно поступається за точністю, але обчислюється на CPU за мілісекунди, що є критичною перевагою у системах з жорсткими вимогами до затримки та обмеженими обчислювальними ресурсами.

Висновки

У роботі проведено порівняльний аналіз трьох математичних моделей прогнозування цін акцій – ARIMA, LSTM та Transformer. Встановлено, що жодна з розглянутих моделей не є універсальною: оптимальний вибір залежить від горизонту прогнозу, обсягу доступних даних та наявних обчислювальних ресурсів. Для задач прогнозування цінових рядів у режимі реального часу за умови наявності достатнього обсягу навчальних даних рекомендується архітектура LSTM або Transformer з відносною RMSE 1.2–1.4%. За обмежених ресурсів або на малих датасетах доцільно застосовувати ARIMA як швидкий та інтерпретований базовий метод. Оптимальним рішенням для виробничої системи моніторингу є гібридний підхід, у якому ARIMA виконує роль фільтра стаціонарних складових, а LSTM або Transformer– моделює нелінійні відхилення. Результати порівняльного аналізу є теоретичною основою для проектування архітектури програмного забезпечення моніторингу акцій на біржі. Тому актуальною є розробка гібридної моделі ARIMA-LSTM з адаптивним перемиканням на основі автоматичного визначення поточного ринкового режиму.

Список використаних джерел

1. Box G. E. P., Jenkins G. M. Time Series Analysis: Forecasting and Control. –Hoboken: Wiley, 2015. – 712 p.
2. Hochreiter S., Schmidhuber J. LongShort-Term Memory // Neural Computation. – 1997. –Vol. 9, No. 8. – P. 1735–1780.
3. Vaswani A. etal. Attention Is All You Need // Advances in Neural Information Processing Systems. – 2017. –Vol. 30. – P. 5998–6008.
4. Sezer O. B., Gudelek M. U., Ozbayoglu A. M. Financial Time Series Forecasting with Deep Learning // Applied Soft Computing. – 2020. –Vol. 90. – P. 1–22.
5. Ding X. etal. Deep Learning for Event-Driven Stock Prediction // Proceedings of IJCAI. – 2015. – P. 2327–2333.
6. Fischer T., Krauss C. Deep Learning with Long Short-Term Memory Networks for Financial Market Predictions // European Journal of Operational Research. – 2018. –Vol. 270, No. 2. – P. 654–669.
7. Zhou H. etal. Informer: Beyond Efficient Transformer for Long Sequence Time-Series Forecasting // Proceedings of AAAI. – 2021. –Vol. 35. – P. 11106–11115.

УДК 004.738.5:364.467

РОЗРОБКА ВЕБ-ПЛАТФОРМИ ДЛЯ ПОШУКУ ТА УПРАВЛІННЯ ВОЛОНТЕРСЬКИМИ ПОДІЯМИ

Миц В.О.[1)] , Юшко А.В.[2)]
Західноукраїнський національний університет
[1)]бакалавр; [2)]д.ф., старший викладач

І. Постановка проблеми

Сфера волонтерства активно розвивається в Україні та світі, особливо в умовах кризових ситуацій таких як пандемії, війни, стихійні лиха та інше[1]. Однак існують проблеми, які ускладнюють ефективну організацію та участь у волонтерських ініціативах. Інформація про події та можливості волонтерства часто публікуються по різних соціальних мережах та сайтах окремих організацій, що ускладнює її систематизацію та швидкий доступ. Відсутність такої платформи ускладнює пошук волонтерських подій за ключовими критеріями, такими як місце проведення, час або тип діяльності. Існуючі міжнародні ресурси не завжди адаптовані до українського контексту, їх функціонал і доступність для місцевих користувачів обмежені. У зв'язку з цим виникає потреба у створенні сучасного інформаційного рішення, яке забезпечить зручний доступ до актуальної інформації, спростить процес взаємодії між учасниками.

ІІ. Мета роботи

Метою роботи є розробка веб-платформи для пошуку та управління волонтерськими подіями, яка забезпечує централізоване збирання, зберігання та обробку інформації про наявні ініціативи. Передбачається створення зручного та інтуїтивного інтерфейсу для користувачів, який дозволить легко знаходити події за різними критеріями.

Система також надаватиме організаторам можливість ефективно координувати події, управляти заявками. Окрім цього, важливою метою є підвищення рівня залученості громадян до волонтерської діяльності шляхом спрощення доступу до актуальної інформації та забезпечення зручних інструментів взаємодії між усіма учасниками процесу.

ІІІ. Основна частина

На основі проведеного аналізу предметної області та вимог користувачів було розроблено концепцію веб-платформи для пошуку та управління подіями. Система покликана об'єднати волонтерів, організаторів та адміністрацію в єдиному середовищі, що забезпечує ефективну комунікацію, планування та координацію волонтерської діяльності.

Платформа реалізована у вигляді односторінкового вебдодатку (SPA) з маршрутизацією на стороні клієнта.

Для користувачів створено окремі функціональні секції, серед яких:

1. Головна сторінка, де представлено коротку інформацію про платформу, її місію є , показати статистику активності (кількість волонтерів, подій та організацій).
2. Сторінка реєстрації та авторизації, що забезпечує створення облікового запису з валідацією даних та безпечним входом у систему.
3. Модуль пошуку і фільтрації подій, який дає змогу користувачам знаходити волонтерські ініціативи за категорією, локацією, датою, типом діяльності.
4. Особистий кабінет волонтера та організатора.
5. Панель організатора, яка дозволяє створювати події, управляти заявками, переглядати та редагувати інформацію про організацію.
6. Адміністративний розділ, що передбачає модерацію, керування користувачами та перегляд аналітичних звітів.

Архітектура додатку побудована за принципом класичного клієнт-серверного підходу:

1. Frontend: React[4,5] та CSS для продуктивності та компонентного підходу.
2. Backend: Supabase (PostgreSQL + REST API + Authentication) [2,3].

Ключові модулі:

1. Головна сторінка: Загальна інформація та статистика.
2. Автентифікація: Реєстрація з валідацією та безпечний вхід.
3. Пошук: Фільтрація подій за категорією, локацією, датою та іншими критеріями.

4. Рольовий доступ: Окремі кабінети для волонтерів, організаторів (створення подій, управління заявками) та адміністраторів (модерація, аналітика).

Рисунок 1 – Діаграма прецедентів

Висновок

У результаті виконання дипломної роботи було розроблено веб-платформу для пошуку та управління волонтерськими подіями, покликану сприяти ефективній координації взаємодії між волонтерами та організаторами. Запропоноване рішення орієнтоване на централізацію інформації про волонтерські ініціативи та спрощення доступу до неї для широкого кола користувачів. Під час реалізації було впроваджено ключові модулі системи, зокрема реєстрацію та авторизацію користувачів, створення і фільтрацію подій за категорією, датою та місцем проведення, а також особисті кабінети для волонтерів і організаторів. Крім того, передбачено рольовий доступ, що дозволяє розмежувати функціональні можливості різних типів користувачів і забезпечує зручність роботи з платформою. Функціональність платформи забезпечує прозору та ефективну взаємодію між усіма учасниками волонтерського процесу, а використання сучасних веб-технологій [2-5] гарантує стабільність, надійність і продуктивність її роботи. Архітектура системи дозволяє легко масштабувати її та інтегрувати додаткові сервіси відповідно до зростаючих потреб користувачів.

У перспективі розроблену систему доцільно доповнити мобільним додатком, багатомовною підтримкою, інтеграцією з Telegram-ботом, а також розширеними аналітичними інструментами для моніторингу активності волонтерів і оцінки ефективності проведених заходів. Таким чином, розроблений веб-застосунок відповідає актуальним потребам волонтерської спільноти України, сприяє підвищенню рівня залученості громадян до волонтерської діяльності та має значний потенціал для подальшого розвитку як соціально важливого IT-рішення.

Список використаних джерел

1. dobro.ua - допомога Україні. dobro.ua - допомога Україні. URL: https://dobro.ua/.
2. PostgreSQL: Documentation. PostgreSQL: The world's most advanced open source database. URL: https://www.postgresql.org/docs/ .
3. Supabase Docs. Supabase | The Postgres Development Platform. URL: https://supabase.com/docs.
4. React. React. URL: https://react.dev.
5. Abramov D. Clark B. The Complete Guide to React. O'Reilly, 2023

УДК 004.42

РОЗРОБКА ІНФОРМАЦІЙНОЇ ВЕБ-СИСТЕМИ ДЛЯ КЕРУВАННЯ СУТНОСТЯМИ З ДИНАМІЧНОЮ СТРУКТУРОЮ ДАНИХ

Сімак А.Ю.[1)], **Гурінчук Н.С.**[2)]
Західноукраїнський національний університет
[1)]д. філ., викладач; [2)]бакалавр

I. Постановка проблеми

У сучасних інформаційних системах спостерігається тенденція до зростання обсягів та різноманітності даних, що потребують гнучких підходів до їх зберігання та обробки. Традиційні системи, побудовані на жорстко визначених структурах даних, часто не дозволяють ефективно адаптуватися до змін вимог користувачів або розширення функціональності.Особливо актуальною є проблема управління сутностями, структура яких може змінюватися в процесі експлуатації системи. Зокрема, це критично для таких сфер, як електронна комерція, системи управління контентом, дослідницькі платформи та інтернет речей, де набір характеристик об'єктів постійно оновлюється.

Більшість існуючих рішень орієнтовані на фіксовані моделі даних, що обмежує їх гнучкість та ускладнює масштабування. Таким чином, виникає необхідність розробки інформаційної веб-системи, яка забезпечує керування сутностями з динамічною структурою даних та підтримує змінюваність їх атрибутів без суттєвого втручання у програмний кодабо проведення складних міграцій бази даних.

II. Мета роботи

Метою роботи є розробка інформаційної веб-системи для керування сутностями з динамічною структурою даних, яка забезпечує гнучке створення, зміну та обробку інформації залежно від потреб користувача, мінімізуючи при цьому витрати ресурсів на розробку та підтримку програмного забезпечення при зміні бізнес-вимог.

III. Основна частина

Розроблена система базується на клієнт-серверній архітектурі, що забезпечує чіткий розподіл функціональності між окремими компонентами та підвищує ефективність обробки даних. Такий підхід дозволяє незалежно масштабувати клієнтську та серверну частини системи, а також спрощує її подальший супровід і модернізацію.Взаємодія між компонентами здійснюється за допомогою стандартизованого REST API, що гарантує слабку зв’язність модулів (див.рис. 1).

Клієнтська частина реалізована з використанням бібліотеки React та відповідає за формування зручного та інтуїтивно зрозумілого інтерфейсу користувача. Вона забезпечує можливість створення, редагування та перегляду сутностей із довільною структурою, а також динамічне відображення їх атрибутів залежно від заданих параметрів. Завдяки використанню компонентного підходу забезпечується повторне використання елементів інтерфейсу та підвищується швидкість розробки.Серверна частина побудована на основі фреймворкуNestJS і виконує обробку запитів від клієнта, реалізацію бізнес-логіки та взаємодію з базою даних. Вона організована за модульним принципом, що дозволяє легко розширювати функціональність системи та додавати нові можливості без значних змін існуючого коду. Крім того, сервер забезпечує валідацію даних, контроль доступу та обробку помилок.

Для зберігання даних використовується база даних PostgreSQL. Ключовою особливістю системи є підтримка динамічної структури сутностей, яка реалізується шляхом використання гнучкого формату зберігання даних (зокрема JSON-полів).Застосування спеціалізованого типу даних JSONB у PostgreSQL дозволяє не лише зберігати інформацію довільної вкладеності, але й ефективно виконувати пошук та індексування за внутрішніми ключами таких структур. Це дозволяє змінювати набір атрибутів сутностей без внесення змін до схеми бази даних, що значно підвищує гнучкість системи та зменшує витрати часу на її адаптацію.

Запропонована система забезпечує ефективне управління даними завдяки підтримці динамічної структури сутностей. Це дозволяє швидко адаптувати систему до нових вимог, мінімізувати необхідність змін у структурі бази даних, підвищити масштабованість програмного продукту та спростити розширення його функціональності. Додатково система може бути інтегрована з іншими сервісами через API, що розширює можливості її використання в різних предметних областях.

Використання сучасного стеку технологій (React, NestJS, PostgreSQL) забезпечує високу продуктивність, надійність та зручність підтримки системи. На відміну від традиційних підходів, запропоноване рішення дозволяє ефективно працювати з даними, структура яких може змінюватися у процесі експлуатації, що є важливим для сучасних інформаційних систем.

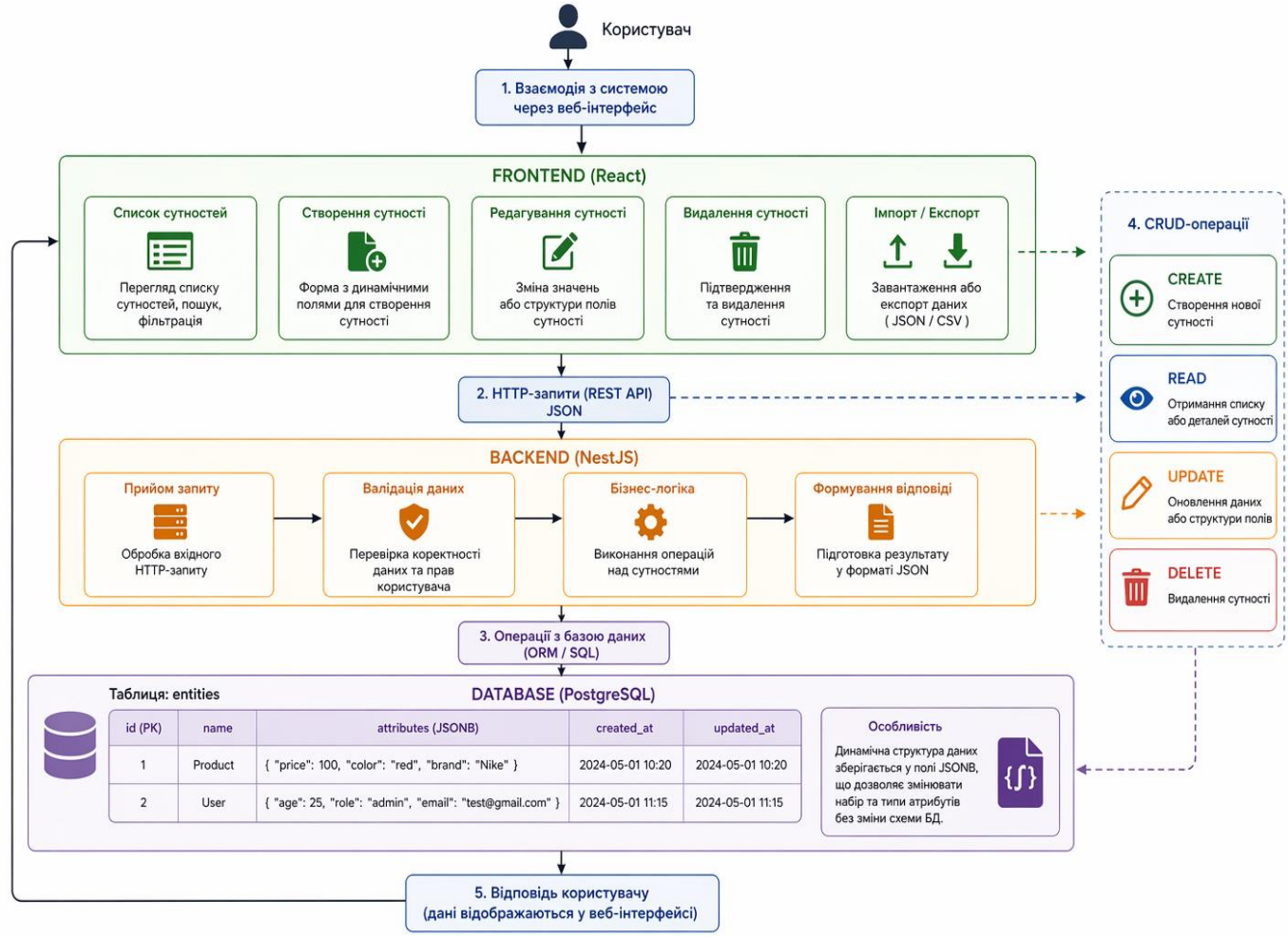

Рисунок 1 – Архітектура та алгоритм роботи інформаційної веб-системи

Висновки

У роботі було розроблено інформаційну веб-систему для керування сутностями з динамічною структурою даних, яка забезпечує гнучке створення, зміну та обробку інформації без необхідності зміни схеми бази даних. Використання клієнт-серверної архітектури та сучасного технологічного стеку (React, NestJS, PostgreSQL) дозволило реалізувати ефективний та масштабований інструмент для роботи з різнорідними даними. Запропоноване рішення підвищує адаптивність системи до змін вимог користувача та спрощує подальший розвиток програмного продукту.

Зокрема, такий архітектурний підхід зменшує показник реалізації нових функцій, оскільки додавання нових властивостей сутностей відбувається безпосередньо на рівні користувацького інтерфейсу. Отримані результати можуть бути використані як основа для створення більш складних інформаційних систем у різних галузях. Подальший розвиток роботи може передбачати впровадження механізмів аналітики даних, інтеграцію систем кешування для додаткового пришвидшення вибірки складних JSON-структур а також розширених засобів безпеки.

Список використаних джерел

1. Бази даних: теорія та практика / К.Ю. Шаховська, В.В. Пасічник. – Львів: «Новий Світ-2000», 2018.
2. Програмна інженерія: принципи та практика / С.Д. П’ятницький, О.В. Гриценко. – Харків: ХНУРЕ, 2021.
3. DesigningData-IntensiveApplications / MartinKleppmann. – O’ReillyMedia, 2017.
4. RESTfulWebServices / LeonardRichardson, SamRuby. – O’ReillyMedia, 2007.

УДК 004.891

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ СЕРВІСУ ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ КНИГ НА ОСНОВІ ГІБРИДНОЇ ФІЛЬТРАЦІЇ

Порплиця Н.П.[1)], **Строгий Є.В.**[2)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)]бакалавр

І. Постановка проблеми

У сучасному світі щоденне зростання обсягів цифрового контенту сформувало глибоке інформаційне перевантаження. Користувачі щоденно зіштовхуються зі значними труднощами під час пошуку та вибору цікавого їм контенту серед мільйонів доступних варіантів. Саме тому розробка систем для підтримки прийняття рішень, таких як персоналізовані рекомендаційні системи, стає однією з найбільш важливих вимог для будь-якої сучасної платформи, яка планує забезпечити ефективний функціонал для своїх користувачів.

Рекомендаційні системи зараз є ефективним інструментом інтелектуальної фільтрації інформації, вони автоматизують процес відбору контенту та пропонують користувачам контент. Проєкт літературного сервісу «ShelfEcho» розробляється з метою забезпечення користувачів високоточними персоналізованими рекомендаціями книг. Фундаментальною проблемою при проєктуванні схожих платформ є подолання ключових алгоритмічних недоліків, які є у класичних ізольованих підходах рекомендацій [1].

Колаборативна фільтрація базується на аналізі матриці взаємодій користувачів та елементів, вона доводить свою ефективність у виявленні неочевидних прихованих шаблонів поведінки, але характеризується проблемою «холодного старту». Ця проблема проявляється у неможливості генерувати прогнози для нових користувачів, які ще не залишили жодних оцінок, або для нових книг, які щойно додані до бази даних і не мають історії переглядів. Контент-орієнтована фільтрація частково розв'язує проблему холодного старту для нових елементів, оскільки аналізує їхні внутрішні метадані, пропонуючи контент, подібний до того, який користувач позитивно оцінив у минулому. Однак цей підхід теж має недоліки. Він не використовує всю базу відомих взаємодій користувачів та книг (кожен користувач оцінюється незалежно від інших), та необхідно знати всю інформацію з метаданих для кожного користувача та книги. Для виправлення цих недоліків доречно використати гібридний підхід, який дозволяє поєднати переваги різних методів, компенсуючи їхні індивідуальні слабкості [2].У цьому проєкті базові компоненти це підсистема контент-орієнтованої фільтрації та підсистема колаборативної фільтрації. У роботі будемо застосувати підхід зваженої гібридизації, вона дозволяє керувати налаштуваннями кожної підсистеми залежно від контексту та обсягу доступних даних[3].

ІІ. Мета роботи

Метою роботи є розробка математичного забезпечення персоналізованих рекомендацій на основі принципу гібридної фільтрації та програмна реалізація в архітектурі веб-додатка ShelfEcho для підвищення якості підбору книг та залученості користувачів.

ІІІ. Математичне забезпечення рекомендаційної системи

Як було сказано раніше, контент-орієнтована система фільтрації буде рекомендувати книги, які схожі на ті, що користувач вже вподобав. Основна задача даної фільтрації – вирахувати подібність між двома наборами даних про книгу. Модель витягує ключові слова в елементі та вираховує їхню вагу за допомогою TF-IDF. Наприклад, набір даних k_i вказує на ключове слово i в елементі d_j, w_{ij} це вага k_i для d_j, тоді контент d_j можна описати за простою формулою (1):

$$Content(d_j) = \{w_{1j}, w_{2j}, \ldots\} \tag{1}$$

Завдяки цьому смаки кожного окремого користувача можна сформувати за допомогою вектора на основі історії взаємодії з книгами, це дозволяє створити профіль вподобань за формулою (2):

$$Content\,Based\,Profile(u) = \frac{1}{|N(u)|} sum_{d \in N(u)} Content(d) \tag{2}$$

де $N(u)$ це те, що користувач u вподобав раніше. Після обчислення вектора $Content(.)$ та вподобань $ContentBasedProfile(.)$ усіх користувачів, система може взяти будь якого користувача u, будь який елемент d та обчислити подібність $ContentBasedProfile(u)$ та $Content(d)$ за формулою (3):

$$p(u,d) = sim(ContentBasedProfile(u), Content(d)) \quad (3)$$

Метод колаборативної фільтрації спирається на інших користувачів. Система вважає, що користувачу з більшою ймовірністю сподобається елемент, який сподобався іншим користувачам які можуть мати схожі смаки. Спочатку необхідно провести пошук та сортування користувачів зі схожими оцінками на книги, а потім обчислити їх подібність за формулою (4):

$$s_{uv} = \frac{|N(u)| \cap |N(v)|}{|N(u)| \cup |N(v)|} \quad (4)$$

де u та v – користувачі, $N(u)$ та $N(v)$ – книги, які вподобали користувачі u та v відповідно [4].

Для роботи зваженої гібридизації відбувається поєднання результатів контент-орієнтованої та колаборативної фільтрації. Адміністратор сервісу може вручну редагувати вплив кожної з систем рекомендацій, щоб отримати необхідний результат.

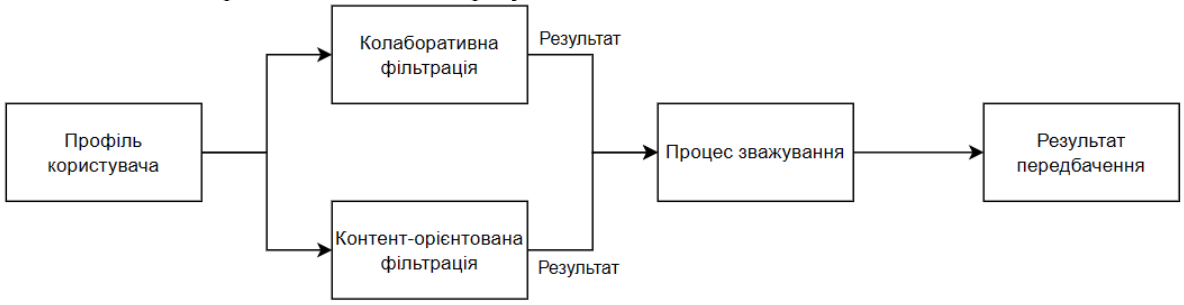

Рисунок 1 – Схема зваженої гібридизації

IV. Архітектура та програмна реалізація

Програмну реалізацію розробленої моделі виконано в рамках проєкту ShelfEcho. Архітектура системи побудована за принципом клієнт-серверної взаємодії.Клієнтська частина реалізована за допомогою React 19 та TypeScript. Серверна частина, що відповідає за обчислення моделі, побудована на Node.js із використанням реляційної бази даних SQLite. Розрахунок вагових коефіцієнтів рекомендацій відбувається динамічно: сервер зчитує конфігурацію rec_weights з таблиці settings. Це дозволяє адміністраторам через адмін панель змінювати вагу різних типів фільтрації у реальному часі без необхідності перерозгортання. Метадані книг система не зберігає локально, а отримує через зовнішнє Open Library API.

Висновок

У праці обґрунтовано та архітектурно адаптовано математичне забезпечення для сервісу персоналізованих рекомендацій книг на основі зваженої гібридної системи. Ця модель ефективно синтезує переваги контент-орієнтованої та колаборативної фільтрації,на основі інтеграції структурованих метаданих з відкритого джерела Open Library API.Головною інновацією запропонованого підходу є використання динамічної функції для зміни вагових коефіцієнтів між типами фільтрації. Цей механізм дозволяє платформі адаптуватися до життєвого циклу профілю користувача: забезпечуючи миттєву релевантність за рахунок простих характеристик на етапі онбордингу (улюблені жанри), та плавно зміщуючи пріоритет на користь колаборативної фільтрації в міру накопичення особистої історії взаємодій.

Список використаних джерел

1. Francesco Casalegno. Recommender Systems – A Complete Guide to Machine Learning Models. 2022. https://medium.com/data-science/recommender-systems-a-complete-guide-to-machine-learning-models-96d3f94ea748
2. Бєлан А.О., Васильєва Л.В. Розробка гібридної рекомендаційної системи. Вісник ХНУ ім. В.Н.Каразіна. 2023. вип. 57. С.22-31. https://doi.org/10.26565/2304-6201-2023-57-02
3. Jeffery Chiang.7 Types of Hybrid Recommendation System. 2021. https://medium.com/analytics-vidhya/7-types-of-hybrid-recommendation-system-3e4f78266ad8
4. Yarlagadda Pranay, Joshita R., Thaneeshkha B. S., V. Sriram. Book recommendation using hybrid algorithm. 2021. https://www.scribd.com/document/629089523/Book-recommendation-system-project

УДК 004.8

ВИКОРИСТАННЯ МЕТОДІВ NLP ДЛЯ СЕМАНТИЧНОГО АНАЛІЗУ МАТЕМАТИЧНИХ ЗАДАЧ У СИСТЕМАХ АВТОМАТИЗАЦІЇ РОЗРАХУНКІВ

ВойтюкІ.Ф.[1)], Крушельницька Х.О.[2)]
Західноукраїнський національний університет
[1)] к.т.н.,доцент; [2)] магістрант

І. Постановка проблеми

У сучасних інформаційних системах автоматизація математичних розрахунків є ключовим інструментом підвищення продуктивності інженерних та наукових досліджень. Проте існуючі програмні продукти, часто вимагають від користувача використання спеціалізованої мови запитів, що ускладнює їх інтеграцію у повсякденну роботу фахівців без глибокої технічної підготовки. Використання природної мови (Natural Language Processing, NLP) для формулювання задач дозволяє усунути цей бар'єр, проте потребує вирішення складних завдань семантичного аналізу та інтерпретації математичного контексту.

ІІ. Мета роботи

Метою дослідження є розробка архітектури мікросервісної системи, що забезпечує автоматизоване розпізнавання, семантичний аналіз та подальше вирішення математичних задач, сформульованих користувачем у текстовому вигляді.

ІІІ. Особливості реалізації системи

Для автоматизації обробки математичних запитів користувача розроблено алгоритм семантичного аналізу, що базується на ітеративному розпізнаванні математичних об'єктів у тексті природною мовою. Основною особливістю запропонованого підходу є трансформація неструктурованого текстового запиту у деревоподібну структуру Abstract Syntax Tree (AST), яка формалізує логіку математичної задачі та забезпечує її подальшу обробку.

На рисунку 1 представлено послідовність обробки математичного запиту користувача. Після етапу попередньої обробки тексту виконується класифікація типу задачі та виділення математичних сутностей за допомогою NER-модуля. Отримані дані трансформуються у формалізовану AST-модель, яка передається до математичного ядра для подальших символьних обчислень і генерації покрокового пояснення розв'язку.

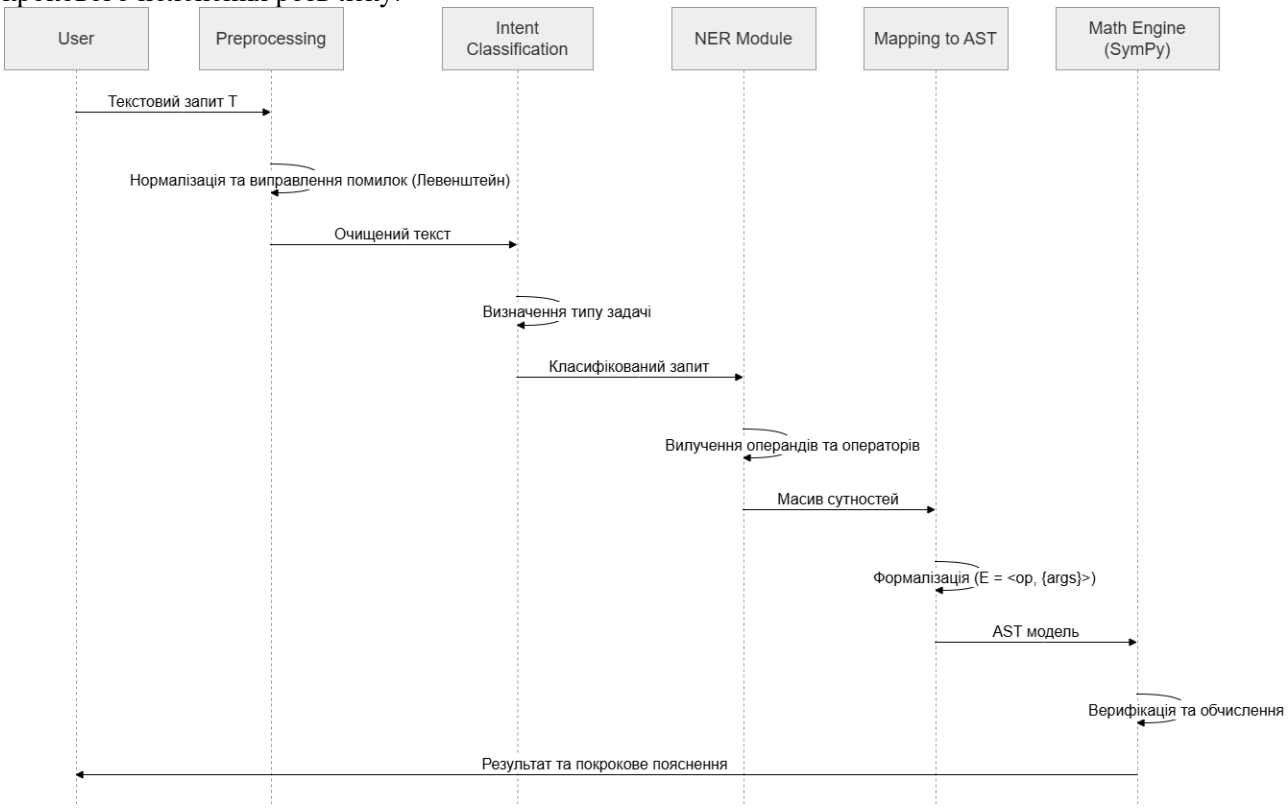

Рисунок 1 – Послідовність обробки математичного запиту користувача

Сформована AST-модель зберігається у базі даних PostgreSQL у форматі JSONB, що дозволяє реалізувати ефективне структурування, аналіз та повторне використання математичних моделей. Завдяки цьому система виконує не лише автоматичне отримання результату, а й глибокий аналіз структури задачі з подальшою генерацією покрокових пояснень у навчальному режимі.

Архітектура системи передбачає використання глибоких нейронних мереж для класифікації типів задач та інтеграцію із зовнішніми API для верифікації розв'язків.[3]

Розроблена система підтримує розв’язання декількох класів математичних задач, зокрема:

- алгебраїчних рівнянь і систем рівнянь;
- задач математичного аналізу, включаючи обчислення похідних та інтегралів;
- арифметичних текстових задач на відсотки, пропорції та рух.

Запропонований підхід забезпечує підвищення точності інтерпретації математичних запитів, зменшує залежність від жорстко визначених шаблонів та створює передумови для побудови адаптивних інтелектуальних освітніх систем.

IV. Архітектура та програмна реалізація

На відміну від початкової реалізації системи у вигляді монолітного веб-застосунку, подальший розвиток програмного забезпечення вимагав перегляду архітектурного підходу. Аналіз процесу обробки запитів та зростання функціональних вимог показав доцільність переходу до мікросервісної архітектури. (див.рис.2)

Основними причинами такого переходу стали необхідність незалежного масштабування окремих компонентів (NLP-аналізу, математичних обчислень, візуалізації), підвищення відмовостійкості системи та спрощення подальшого розширення функціональності.

Таким чином, мікросервісна архітектура розглядається як еволюційний етап розвитку системи, що базується на результатах попередньої реалізації.

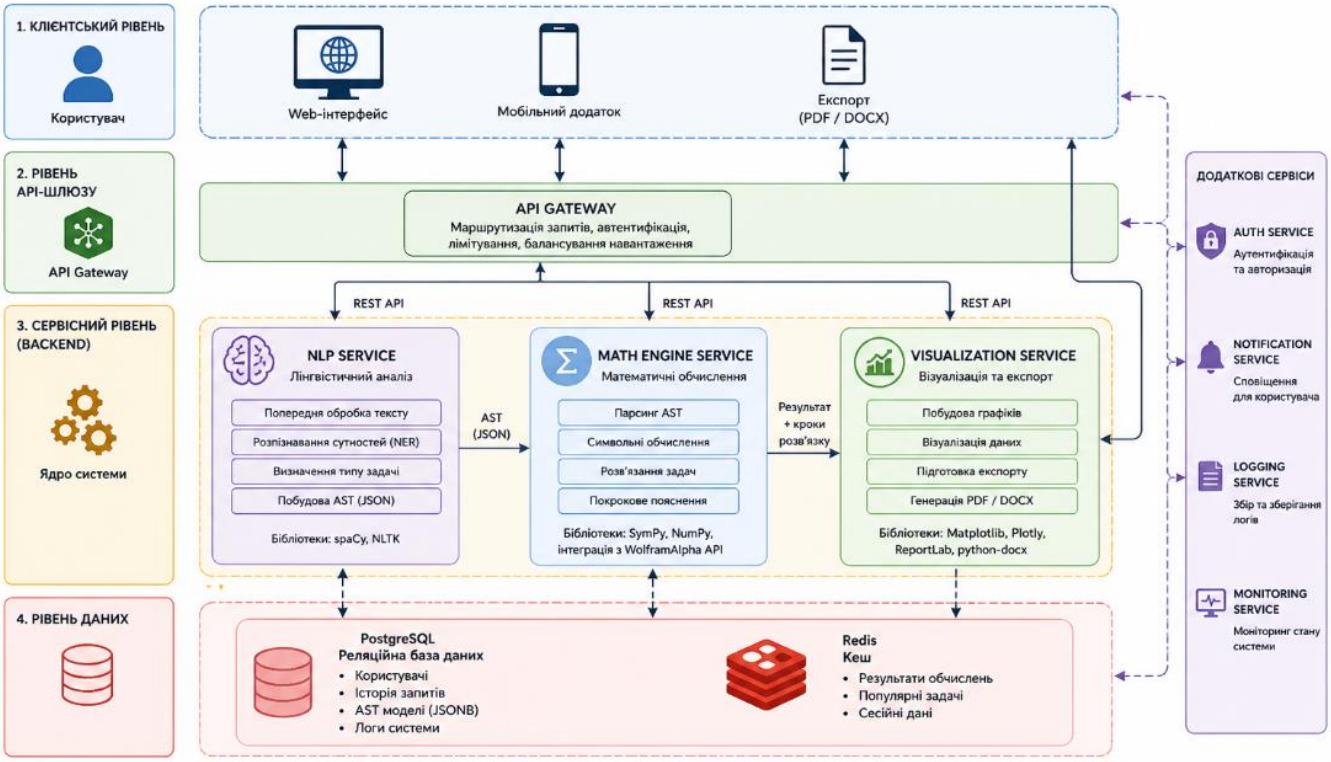

Рисунок 2 – Схема мікросервісної архітектури системи

Для реалізації системи автоматизованого розв’язання математичних задач обрано сервіс-орієнтовану архітектуру (SOA) з використанням принципів мікросервісного підходу. Така архітектура забезпечує модульність, гнучкість обробки даних та можливість масштабування системи відповідно до зростання функціональних вимог.

Основний принцип мікросервісної архітектури полягає в розбитті додатка на невеликі, незалежні компоненти, які називаються мікросервісами. Кожен мікросервіс відповідає за конкретну частину функціональності програми[2]. Це дозволяє спростити підтримку системи, підвищити відмовостійкість та забезпечити незалежне оновлення окремих модулів.

Архітектура системи реалізована у вигляді багаторівневої структури, де кожен рівень виконує окрему функцію обробки інформації:

1. Клієнтський рівень (Frontend) – забезпечує взаємодію користувача із системою через вебінтерфейс та формування текстових математичних запитів.
2. Рівень API-шлюзу (Gateway) – виконує маршрутизацію запитів між сервісами, контроль доступу та координацію взаємодії компонентів системи.
3. Сервісний рівень (Backend) – є ядром системи та містить основні мікросервіси:
   - NLP Service – відповідає за обробку природної мови та включає етапи токенізації, виділення іменованих сутностей (NER) та семантичного аналізу. Результатом обробки є формалізоване представлення задачі у вигляді абстрактного синтаксичного дерева (AST), яке використовується для подальших обчислень.
   - Math Engine Service – здійснює математичні обчислення на основі сформованої структури задачі з використанням бібліотек SymPy, NumPy або зовнішніх математичних API.
   - Visualization Service – формує дані для побудови графіків, візуалізації результатів та експорту розв'язків у формати PDF і DOCX.
4. Рівень даних (Database Layer) – забезпечує зберігання та швидкий доступ до інформації:
   - PostgreSQL використовується для зберігання профілів користувачів, історії запитів, математичних моделей та системних журналів.
   - Redis застосовується як кеш для прискорення доступу до результатів часто повторюваних обчислень.

Програмна реалізація системи базується на принципі слабкої зв'язаності модулів, що дозволяє масштабувати функціональність без зміни загальної структури системи. Додавання нових типів математичних задач реалізується шляхом розширення NLP-модуля та математичного ядра, що забезпечує адаптивність і подальший розвиток програмного забезпечення.

Для забезпечення стабільної взаємодії між мікросервісами використовується обмін даними у форматі JSON через REST API. Такий підхід забезпечує стандартизовану передачу інформації між компонентами системи та спрощує інтеграцію додаткових сервісів у майбутньому.

Крім того, мікросервісна архітектура дозволяє незалежно оновлювати окремі модулі програмного забезпечення без впливу на роботу інших компонентів. Це особливо важливо при розширенні функціональності системи та інтеграції нових алгоритмів математичної обробки.

Запропонований підхід також забезпечує можливість подальшого використання системи у хмарному середовищі. Завдяки розподіленню функціональності між окремими сервісами програмне забезпечення може масштабуватися відповідно до навантаження та кількості користувачів. Це створює передумови для використання системи у великих освітніх та наукових інформаційних платформах. Крім того, використання мікросервісного підходу спрощує тестування та супровід окремих компонентів системи. Це дозволяє підвищити надійність програмного забезпечення та прискорити процес впровадження нових функціональних можливостей.

Наукова новизна запропонованого підходу полягає у поєднанні методів NLP та AST-формалізації математичних задач у межах мікросервісної архітектури, що забезпечує адаптивність і масштабованість системи автоматизованих розрахунків.

Висновок

У роботі розглянуто використання методів NLP для семантичного аналізу математичних задач у системах автоматизації розрахунків. Запропонований підхід дозволяє перетворювати неструктуровані текстові запити на формалізовані алгоритми та забезпечує підвищення ефективності взаємодії користувача з автоматизованими системами розрахунків. Це забезпечує підвищення ефективності взаємодії користувача з автоматизованими системами розрахунків, роблячи процес розв'язання задач інтуїтивно зрозумілим. Подальші дослідження будуть спрямовані на покращення точності розпізнавання складних багатокомпонентних задач із використанням методів штучного інтелекту.

Список використаних джерел

1. Обробка природної мови (NLP): основні поняття та застосування. Режим доступу: https://www.sap.com/ukraine/resources/what-is-natural-language-processing
2. VaswaniA., et al. Attention Is AllYou Need // Advances inNeural Information Processing Systems, 2017. P.6000-6.
3. Fielding R. Architectural Styles and the Design of Network-based Software Architectures. – University of California, Irvine, 2000. – Режим доступу:https://roy.gbiv.com/pubs/dissertation/

УДК 004.891

РОЗРОБКА КОМПЛЕКСНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН-ЗАПИСУ ТА АДМІНІСТРУВАННЯ СТОМАТОЛОГІЧНОЇ КЛІНІКИ

Юшко А.В.[1)], Гевко А.І.[2)]
Західноукраїнський національний університет
[1)]д.ф., старший викладач; [2)] бакалавр

І. Постановка проблеми

Сучасний стан розвитку медичної галузі в Україні характеризується гострою потребою у цифровізації процесів взаємодії між лікувальними закладами та пацієнтами. Традиційні методи адміністрування, що базуються на паперових носіях або локальних журналах запису, призводять до значних часових витрат, виникнення помилок при плануванні прийомів та ризику втрати критично важливої медичної інформації. Окрім того, відсутність єдиної екосистеми для синхронізації даних між веб-порталом для пацієнтів та внутрішньою системою клініки ускладнює оперативне керування ресурсами. Актуальність дослідження зумовлена необхідністю створення масштабованої та безпечної інформаційної системи, яка б автоматизувала повний цикл обслуговування – від моменту дистанційного запису до ведення детальної електронної медичної карти з візуалізацією стану зубів.

ІІ. Мета роботи

Метою роботи є проектування та розробка багатокомпонентної інформаційної системи для автоматизації бізнес-процесів стоматологічної клініки. Проект спрямований на створення єдиного інформаційного простору, який дозволяє здійснювати онлайн-запис пацієнтів, вести електронні медичні карти, візуалізувати стан ротової порожнини та здійснювати фінансовий моніторинг діяльності закладу. Система має базуватися на принципах модульності, безпеки та забезпечувати високу мобільність адміністративних функцій.

ІІІ. Архітектура та програмна реалізація

Програмний комплекс реалізовано на основі трирівневої архітектури. Серверний рівень побудовано з використанням фреймворку Django REST Framework, що забезпечує функціонування RESTful API для обміну даними між компонентами. В якості системи керування базами даних обрано SQLite3, що дозволяє забезпечити реляційну цілісність даних та високу швидкість доступу при мінімальних системних вимогах [3]. Клієнтська частина розділена на два основні модулі. Веб-інтерфейс, розроблений на базі React та TypeScript, охоплює функції пацієнтського порталу та адміністративну панель. Таке рішення забезпечує менеджменту клініки мобільність та незалежність від стаціонарного обладнання, дозволяючи керувати розкладом та звітністю з будь-яких пристроїв. Спеціалізоване робоче місце лікаря реалізовано як десктопний додаток на базі фреймворку Qt 6 (C++), що гарантує максимальну швидкість рендерингу складних графічних елементів, таких як дентальна карта [4].

Важливим науково-технічним рішенням у проекті є впровадження двоцифрової системи ідентифікації зубів, розробленої Міжнародною федерацією стоматологів. Ця система класифікації є міжнародним стандартом, що офіційно підтримується Всесвітньою організацією охорони здоров'я та широко використовується у клінічній практиці більшості країн світу [1]. Вибір цієї нотації обумовлений її математичною логікою, що значно спрощує алгоритмічну обробку даних: кожному зубу відповідає унікальний числовий код, що чітко відображає його анатомічне розташування. Кожний зуб у системі отримує унікальний код N, який розраховується за формулою (1).

$$N = 10q + p \tag{1}$$

де q – номер квадранта ротової порожнини ($q \in \{1, 2, 3, 4\}$ для постійних зубів), а p – позиційна адреса зуба відносно серединної лінії ($p \in \{1, 2, \ldots, 8\}$).

Такий підхід дозволяє формалізувати клінічні записи у вигляді числових масивів, що підвищує ефективність пошукових запитів та фільтрації даних при роботі з електронними медичними картами [2]. Крім того, використання структурованого представлення даних спрощує їх подальшу обробку та аналіз, зокрема при інтеграції з іншими інформаційними системами. Це також забезпечує більшу однорідність збереженої інформації та зменшує ймовірність виникнення помилок при введенні або

інтерпретаціїданих. На основі цієї математичної моделі було розроблено програмний модуль «Зубна Формула» (див.рис.1).

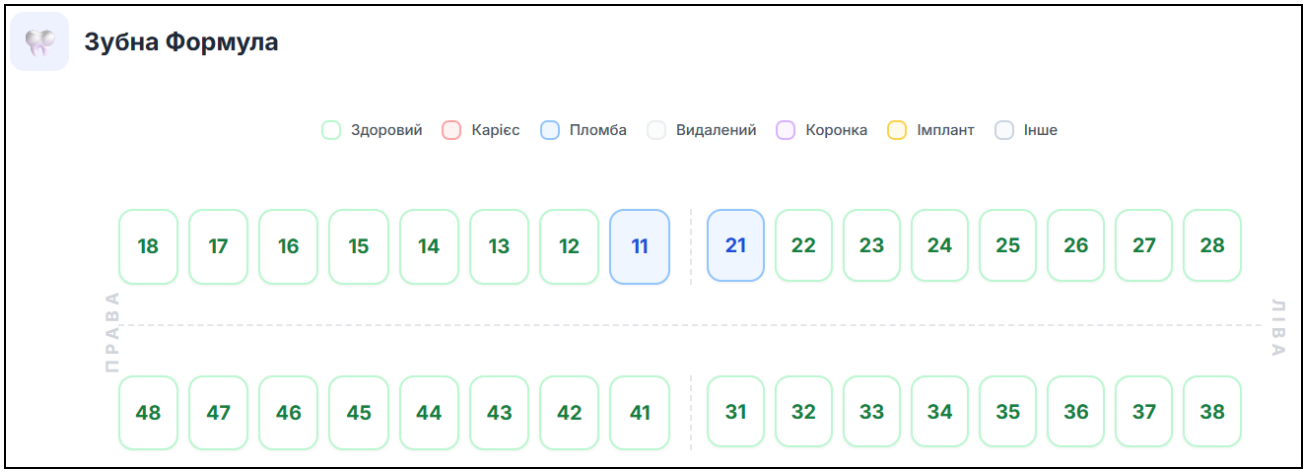

Рисунок 1 – Програмна реалізація інтерактивної зубної формули з візуалізацією клінічних статусів (FDI)

На відміну від статичних схем, реалізований графічний інтерфейс забезпечує інтуїтивно зрозумілу кольорову індикацію стану кожного зуба. Лікар може швидко призначати статуси: «Здоровий» (зелений контур), «Карієс» (червоний), «Пломба» (синій), «Видалений» (сірий), «Коронка» (фіолетовий), «Імплант» (жовтий) та «Інше». Наприклад, на рисунку 1 наочно відображено клінічну картину, де центральні різці верхньої щелепи (зуби 11 та 21) мають статус «Пломба», тоді як інші зуби зафіксовані як здорові.

IV. Аналітичне забезпечення та результати

Реалізована інтерактивна дентальна карта дозволяє лікарям фіксувати стан кожного анатомічного об'єкта через систему статусів, що зберігаються в базі даних як пов'язані сутності. Автоматизація розрахунку вартості послуг інтегрована безпосередньо з електронною картою, що усуває помилки при формуванні рахунків. Використання веб-технологій для адміністративних функцій дозволило відмовитися від складних інсталяцій на робочих місцях адміністраторів, забезпечуючи доступ до управління клінікою через браузер.

Для оптимізації роботи з великими обсягами інформації в системі впроваджено механізми пагінації та дебаунс-таймери на стороні клієнта, що знижує частоту звернень до СУБД SQLite3. Архітектура системи передбачає розмежування прав доступу на рівні JWT-токенів, забезпечуючи конфіденційність медичної інформації відповідно до сучасних стандартів безпеки [5]. Практична апробація підтвердила, що обраний стек технологій забезпечує необхідну чуйність інтерфейсу та стабільність роботи системи при одночасній взаємодії декількох модулів.

Висновок

У дослідженні реалізовано комплексну цифрову екосистему, яка вирішує проблему інформаційної асиметрії в медичному обслуговуванні. Архітектура системи синергічно поєднує інтерактивний веб-портал для пацієнтів і адміністраторів із високопродуктивним десктопним додатком для лікарів. Такий підхід забезпечує безперервний обмін даними між усіма учасниками процесу та підвищує оперативність прийняття клінічних рішень. Впровадження стандарту FDI для візуалізації стану ротової порожнини створило надійну основу для автоматизованої обробки клінічних даних. Це також відкриває можливості для подальшого використання аналітичних інструментів і впровадження елементів штучного інтелекту в медичні процеси. Запропонована архітектурна модель виступає не лише інструментом поточної оптимізації роботи стоматологічної клініки, а й основою для подальшого розширення функціональних можливостей системи та її адаптації до інших медичних закладів.

Список використаних джерел

1. FDI World Dental Federation. FDI Two-Digit Notation [Electronic resource]. URL: https://www.fdiworlddental.org/
2. Santosh A. B. R. Enhancing Precision: Proposed Revision of FDI's 2-Digit Dental Numbering System. International Dental Journal. – 2024. – Vol. 74. – P. 201–204.
3. Django 5.0 documentation / Django Software Foundation. URL: https://docs.djangoproject.com/en/5.0/
4. Qt 6.x Reference Documentation / The Qt Company. URL: https://doc.qt.io/qt-6/
5. Dalimunthe S. Restful API Security Using JSON Web Token (JWT) With HMAC-Sha512 Algorithm in Session Management. IT Journal Research and Development. – 2023. – Vol. 8, No. 1. – P. 81–94.

УДК 004.41

РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ ДЛЯ РОЗРАХУНКУ ВАРТОСТІ РОЗМИТНЕННЯ АВТОМОБІЛІВ ТА СУПУТНІХ ВИТРАТ

Крепич С.Я.[1)], Співак І.Я.[2)], Пужляков М.Д.[3)]
Західноукраїнський національний університет
[1-2)] к.т.н., доцент; [3)] ,бакалавр

I. Постановка проблеми

Сучасний етап розвитку цифрової економіки в Україні вимагає створення ефективних інструментів для взаємодії громадян із державними фінансовими структурами. Процес імпорту транспортних засобів залишається однією з найбільш затребуваних, але водночас складних процедур через мінливість законодавства та багатокритеріальність нарахування податків. Основними проблемами є: по-перше, складність алгоритмів розрахунку акцизного збору, що залежать від віку авто та типу палива; по-друге, необхідність оперативного отримання актуальних курсів валют НБУ; по-третє, відсутність зручних мобільних рішень, що працюють у режимі реального часу. Актуальність дослідження зумовлена необхідністю розробки легкої та надійної мобільної системи, здатної інтегруватися в щоденну діяльність користувача та надавати об'єктивну оцінку фінансових витрат ще до моменту придбання автомобіля.

II. Мета роботи

Метою роботи є розробка математичного забезпечення та програмного засобу на базі ОС Android для автоматизованого розрахунку та прогнозування вартості митного оформлення транспортних засобів, що дозволить підвищити прозорість процесу та мінімізувати ризики помилок при плануванні бюджету.

III. Математична модель розрахунку платежів

Для формалізації процесу розрахунку сумарних витрат у національній валюті C_{UAH} пропонується модель, що агрегує основні податкові складові та нормалізує їх згідно з актуальним ринковим курсом (1):

$$C_{UAH} = E_r \times (I_p + I_p \times k_m + E_a + V_{tax} + P_f) \quad (1)$$

де E_r – валютний курс за даними НБУ; I_p – інвойсна вартість авто; k_m – коефіцієнт мита; E_a – сума акцизного податку; V_{tax} – ПДВ; P_f – збір до Пенсійного фонду.

Основним розрахунковим параметром є акцизний податок E_a, величина якого визначається технічними характеристиками двигуна та терміном експлуатації ТЗ (2):

$$E_a = B_s(T_b) \times (V_c \div 1000) \times Y_a \quad (2)$$

де B_s – базова ставка, що залежить від типу двигуна T_b; V_c – об'єм двигуна; Y_a – вік автомобіля. Опис параметрів моделі наведено в таблиці 1.

Таблиця 1

Базові параметри розрахунку вартості розмитнення

Назва показника	Позначення	Опис та вплив на результат
Тип двигуна	T_b	Визначає базову ставку акцизу (бензин - 50€/100€, дизель - 75€/150€)
Об'єм двигуна	V_c	Використовується як основний множник для розрахунку потужності
Інвойсна ціна	I_p	Базова величина для розрахунку
Валютний курс	E_r	Коефіцієнт нормалізації вартості згідно з даними НБУ на дату розрахунку
Вік автомобіля	Y_a	Коефіцієнт, що лінійно збільшує суму акцизу (від 1 до 15 років)

IV. Архітектура та програмна реалізація

Програмний комплекс розроблено як мобільний застосунок для платформи Android у середовищі Android Studio. Архітектурне рішення базується на принципах модульності, що складається з трьох ключових рівнів:

1. Рівень представлення (UI Layer): реалізований за допомогою XML-розмітки та Material Design компонентів, що забезпечує адаптивність інтерфейсу для різних розширень екранів (див.рис.1).
2. Рівень бізнес-логіки (Domain Layer): містить аналітичне ядро, яке виконує обчислення згідно з формулою (1). Для обробки даних у реальному часі та взаємодії з REST API (отримання курсів НБУ) використано бібліотеку Retrofit.
3. Рівень даних (Data Layer): забезпечує локальне зберігання податкових ставок та історії розрахунків у базі даних SQLite (з використанням Room Persistence Library), що дозволяє проводити базові калькуляції навіть за відсутності стабільного інтернет-з'єднання.

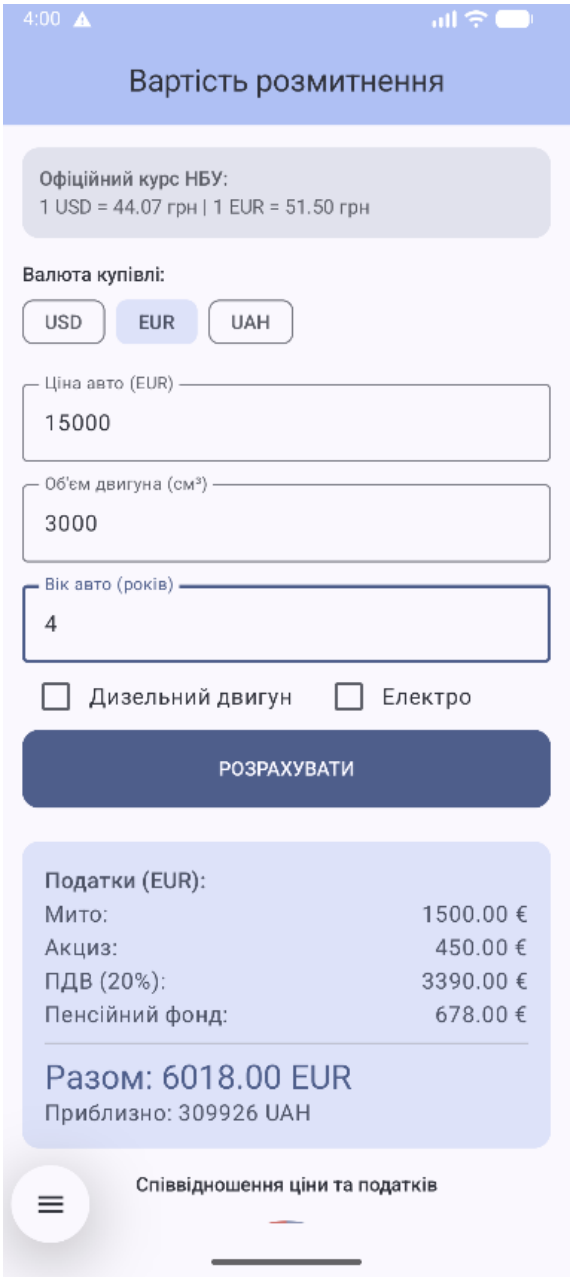

Рисунок 1 – Інтерфейс вікна «Вартість розмитнення»

Особлива увага в ході реалізації була приділена управлінню життєвим циклом розробки. Відповідно до принципів управління проєктами, на початковому етапі було сформовано ієрархічну структуру робіт (WBS), що дозволило декомпозувати процес створення системи на окремі модулі: від проектування логіки калькулятора до фінального тестування UI. Моніторинг виконання задач здійснювався за допомогою діаграми Ганта, що забезпечило дотримання дедлайнів на кожному етапі. Оптимізація коду та тестування проводилися з урахуванням технічних характеристик пристроїв середнього цінового сегмента для забезпечення максимальної швидкодії інтерфейсу.

Висновок

У роботі запропоновано та реалізовано метод автоматизованого розрахунку вартості розмитнення авто, який базується на об'єктивних податкових метриках. Використання мобільної платформи забезпечує доступність сервісу в режимі реального часу, а впровадження математичної моделі мінімізує вплив суб'єктивного чинника при оцінюванні. Практична цінність результатів підтверджується можливістю інтеграції системи в існуючі логістичні процеси IT-компаній, що займаються імпортом техніки.

Список використаних джерел

1. Митний кодекс України: Закон України від 13.03.2012 № 4495-VI.
2. Співак І.Я., Крепич С.Я.. Горішний В.І. Організація CLOUD-архітектури для систем забезпечення функціональної придатності статичних систем. Науковий журнал «Сучасні інформаційні технології», Харків, Том 3, №2, 2019. – с.35-39
3. Крепич С.Я. Програмний комплекс оцінювання функціональної придатності пристроїв при заданих допустимих значеннях вихідних характеристик та допусків на параметри їх елементів. Сучасні комп'ютерні інформаційні технології: Матеріали Всеукраїнської школи-семінару молодих вчених і студентів ACIT'2015 – Тернопіль: ТНЕУ, 2015. – С. 23-25
4. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 7th ed. PMI, 2021. 250 p.
5. Крепич С.Я. Моделювання та забезпечення функціональної придатності статичних систем методами аналізу інтервальних даних/ С.Я.Крепич// дис.канд.тех.наук, НУ «Львівська політехніка», Львів, 2016. – 166с.
6. Співак І.Я., Крепич С.Я., Федоров О.А. Програмна система оцінювання ефективності праці в залежності від потреб. Матеріали школи-семінару молодих вчених і студентів «Комп'ютерні інформаційні технології» CIT'2019, 29 листопада 2019., Тернопіль, стр.34
7. ISO 31000:2018. Risk management — Guidelines. ISO, 2018.

УДК 004.891

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ОНЛАЙН БРОНЮВАННЯ ТА АДМІНІСТРУВАННЯ СТОЛІВ У РЕСТОРАНІ

Тимчишин Б.С.[1)], **Улітко Ю.М.**[2)]
Західноукраїнський національний університет
[1)]*д.ф., викладач*; [2)]*бакалавр*

І. Постановка проблеми

Сфера ресторанного бізнесу характеризується високим рівнем конкуренції та стрімким переходом до цифрових форматів обслуговування. Вже звичні підходи до управління резервуваннями, що здебільшого базуються на телефонних дзвінкахта ручному веденні записів, виявляють свою неефективність в умовах динамічного потоку відвідувачів.

Такий неавтоматизований формат комунікації часто призводить до втрати потенційних клієнтів через тривале очікування відповіді, виникнення помилок через людський фактор,створення подвійних бронювань та загального зниження лояльності відвідувачів.

Крім того, фрагментованість даних та відсутність єдиної системи ускладнюють роботу персоналу та збільшують потрібну кількісь працівників для забезпечення потрібного рівню обслуговування. Без інструментів аналітики менеджмент закладу позбавлений можливості об'єктивно оцінювати завантаженість залу, прогнозувати пікові години та правильно розподіляти робочі ресурси.Актуальність даного дослідження зумовлена необхідністю розробки комплексного програмного рішення, здатного покращити процес взаємодії між гостем та рестораном та автоматизувати частину процесів[1]. Проєктування онлайн-системи бронювання повинно усунути комунікаційні бар'єри шляхом впровадження двох взаємопов'язаних компонентів. Перший - це клієнтський інтерфейс, що забезпечує користувачам можливість самостійного вибору часу бронювання, вільного столу. Другий - це адміністративний дашборд для персоналу, який у реальному часі відображатиме інформацію про кількість активних бронювань, статуси столів та загальну статистику закладу. Впровадження такої системи дозволить оптимізувати операційну діяльність ресторану та мінімізувати вплив людського фактору.

ІІ. Мета роботи

Метою роботи є розробка інформаційної системи для автоматизації бронювання столів та управління операційною діяльністю ресторану. Проєкт створює єдине середовище, що об'єднує клієнтський інтерфейс для зручного вибору часу і столів із панеллю адміністратора для моніторингу бронювань у реальному часі. Додатково система забезпечує динамічну генерацію вебадрес закладів, облік клієнтів, управління конфігурацією залу та візуалізацію статистики завантаженості. Для користувачів система створює швидкі та комфортні засоби бронювання місця. Впровадження такого програмного рішення має на меті оптимізацію ресурсів закладу, мінімізацію людського фактора при резервуванні та підвищення загального рівня клієнтського сервісу.

ІІІ. Математичне забезпечення рекомендаційної системи

Математичне забезпечення системи базується на алгоритмічній перевірці доступності ресурсів у визначений користувачем час. Нехай $S = \{s_1, s_2, ..., s_n\}$ – множина всіх зареєстрованих столиків у закладі. Кожен столик s_i характеризується максимальною місткістю C_i та параметром видимості для клієнтів $v_i \in \{0,1\}$ означає прихований статус. Для успішного створення бронювання запит клієнта $R(t, g)$, де t –обраний час, а g – кількість гостей, має задовольняти систему умов (1):

$$\begin{cases} g \le c_i, \\ v_i = 1, \\ B(s_i, t) = 0. \end{cases} \quad (1)$$

де $B(s.t) = 0$ – булева функція зайнятості, яка приймає значення 1, якщо столик s_i має активне або підтверджене бронювання на час t , і 0, якщо він повністю вільний.

Для адміністративної панелі інтегральний показник кількості доступних столів N_{free} у поточний момент розраховується шляхом агрегації даних з бази (2):

$$N_{free} = \sum_{i=1}^{n} (1 - B(s_i, t_{now})) \cdot v_i \quad (2)$$

Цей алгоритм унеможливлює виникнення колізій (подвійного бронювання) та дозволяє системі динамічно приховувати недоступні столики у клієнтському інтерфейсі вибору.

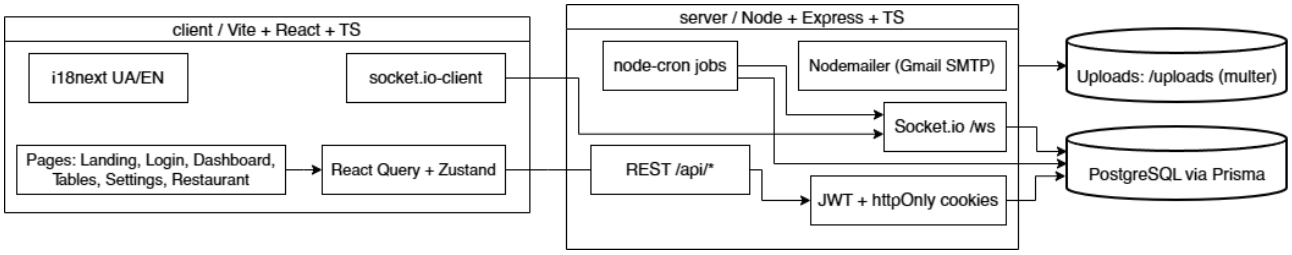

Рисунок 1 – Схема архітектури проекту

IV. Архітектура та програмна реалізація

Програмну реалізацію розробленої онлайн-системи бронювання виконано на основі клієнт-серверної архітектури з використанням інструменту збірки Vite та стеку технологій Node.js і React [2,3]. Серверна частина, що виступає механізмом аналітики та керування процесами, побудована на базі фреймворку Express із використанням мови TypeScript, що забезпечує строгу типізацію та підвищує загальну надійність програмного коду.

Для збереження даних використовується реляційна система управління базами даних PostgreSQL у поєднанні з ORM-інструментом Prisma. Це гарантує цілісність інформації про профілі користувачів, конфігурації закладів та статуси бронювань. За безпеку сесій та автентифікацію в системі відповідає надійний механізм JWT-токенів, які зберігаються у захищених файлах cookie.

Клієнтський додаток побудовано із використанням Zustand для управління глобальним станом, React Query для обробки асинхронних запитів та i18next для підтримки багатомовності. Обмін даними між клієнтом та сервером реалізовано гібридним шляхом: окрім стандартного REST API, в архітектуру інтегровано протокол WebSocket. Цей двосторонній зв'язок у реальному часі дозволяє адміністраторам миттєво отримувати візуальні сповіщення про нові запити та динамічно оновлювати інформацію на дашборді без перезавантаження сторінки.

Для обробки фонових процесів розгорнуто планувальник задач node-cron, який автономно контролює життєвий цикл резервувань, зокрема автоматично скасовує непідтверджені записи після вичерпання встановленого тайм-ауту. Розподіл ресурсів забезпечується алгоритмом динамічної валідації: під час бронювання система аналізує базу даних і відображає для вибору лише ті столи, що вільні на обраний час. Додатково алгоритм враховує налаштування видимості, динамічно генеруючи публічну сторінку ресторану на основі заповнених адміністратором даних.

На рівні користувацького досвіду (UX) інтерфейс адміністратора підтримує інтерактивну організацію простору зали за допомогою технології Drag-and-drop , що безпосередньо формує порядок відображення столиків на сторінці кінцевого клієнта. Практична цінність розробленого підходу полягає в автоматизації адміністративної рутини та мінімізації впливу людського фактора на процес оцінки завантаженості закладу.

Висновок

У роботі спроєктовано та реалізовано комплексну інформаційну систему для автоматизації процесів бронювання та управління операційною діяльністю ресторанного бізнесу. Архітектура системи базується на тісній взаємодії веб-інтерфейсу (React/TypeScript) та серверного ядра (Node.js/PostgreSQL), що гарантує стабільну роботу платформи в умовах динамічних запитів.

Важливим результатом дослідження стала розробка математичної моделі перевірки доступності ресурсів, яка інтегрує параметри часу, місткості та статусу столиків для запобігання колізіям при резервуванні. Впровадження запропонованої системи дозволяє суттєво мінімізувати вплив людського фактора, оптимізувати комунікацію між гостем і адміністрацією, а також забезпечити загальну масштабованість бізнес-процесів закладу.

Список використаних джерел

1. Матвійчук Л. Ю., Чепурда Л. М., Лютак О. М. та ін. Готельно-ресторанна справа : навчальний посібник. Луцьк : РВВ ЛНТУ, 2023. 213 с.
2. React Documentation / Meta Platforms, Inc. URL: https://react.dev/reference/react
3. Node.js v20.x Documentation / OpenJS Foundation. URL: https://nodejs.org/docs/latest-v20.x/api/

УДК 004.891

ПРОГРАМНА АРХІТЕКТУРА СИСТЕМИ ДИНАМІЧНОГО КОНФІГУРУВАННЯ ІОТ-ПРИСТРОЇВ У МЕРЕЖАХ «РОЗУМНОГО БУДИНКУ»

Порплиця Н.П.[1], **Самчук М.І.**[2]
Західноукраїнський національний університет
[1] к.т.н., доцент; [2] бакалавр

І. Постановка проблеми

Використання концепції Інтернету речей (IoT) та систем «розумного будинку» супроводжується постійним зростанням кількості підключених пристроїв. Сучасна інфраструктура розумного будинку є гетерогенною мережею, що складається з десятків різнотипних мікроконтролерів, безлічі сенсорів та виконавчих механізмів.

Традиційний підхід до розробки програмного забезпечення для таких спеціалізованих комп'ютерних систем здебільшого базується на створенні монолітних прошивок, як правило – із використанням C/C++. В умовах реальної експлуатації такий підхід виявляє низку критичних недоліків. Зокрема, додавання нового типу датчика, зміна алгоритму реагування системи або навіть оновлення облікових даних Wi-Fi вимагає внесення змін до вихідного коду, повної перекомпіляції проєкту та наступного перепрошивання кожного вузла.

Хоча існуючі механізми оновлення «повітрям» (OTA – Over-The-Air) частково вирішують проблему необхідності фізичного доступу до пристроїв, вони зазвичай передбачають завантаження повного образу прошивки. Це є ресурсомістким процесом, який перевантажує мережу та підвищує ризик збою пристрою у разі перебоїв з живленням чи зв'язком під час такого оновлення.

Саме тому, актуальним підходом для вирішення зазначених проблем є розробка гнучкої архітектури, яка б дозволяла динамічно, на рівні окремих конфігураційних файлів та незалежних модулів, змінювати поведінку IoT-пристроїв без їхнього повного перепрошивання, керуючи цим процесом централізовано, зокрема із використанням мови програмування – MicroPython.

ІІ. Мета роботи

Метою роботи є розробка гнучкої архітектури програмного забезпечення для гетерогенних IoT-мереж, яка забезпечує можливість динамічного розгортання функціональних модулів та інтелектуального конфігурування вузлів під управлінням середовища MicroPython. Запропоноване рішення спрямоване на автоматизацію процесів оновлення бізнес-логіки «повітрям» без необхідності повної перекомпіляції монолітних образів системи.

ІІІ. Дворівнева модель динамічного конфігурування пристроїв

Для досягнення мети було розроблено модульну архітектуру програмного забезпечення для IoT-вузлів, що функціонують під управлінням середовища MicroPython. Ключова інновація підходу полягає у відмові від монолітної структури коду на користь динамічного життєвого циклу пристрою. Загальний алгоритм функціонування розробленої системи проілюстровано на рисунках 1-2.

Рисунок 1 – Блок-схема запуску дворівневого життєвого циклу програмного забезпечення IoT-вузла

Рисунок 2 – Блок-схема роботи дворівневого життєвого циклу програмного забезпечення IoT-вузла

Після завершення фази конфігурування управління передається головному модулю main.py. Його архітектура базується на паттерні «абстрактний конвеєр» (Pipeline), що забезпечує уніфікацію обробки даних незалежно від типу підключених сенсорів. Модуль ініціалізує програмні об'єкти для кожного етапу конвеєра: зчитування (Reader), фільтрація/перетворення (Processor) та відправка (Sender). У нескінченному циклі main.py забезпечує безперервний моніторинг стану об’єкта та асинхронну взаємодію з MQTT-брокером. Через MQTT-протокол реалізується двосторонній зв’язок: передача телеметрії на центральний вузол (Raspberry Pi) та отримання директив для актуаторів у реальному часі. Таке розділення відповідальності між завантажувачем та виконавчим ядром забезпечує високу відмовостійкість системи та можливість її масштабування без втрати показників керованості.

Фаза ініціалізації та динамічного конфігурування реалізується через модуль boot.py, який виконує роль первинного завантажувача. Під час старту мікроконтролер ініціалізує мережеві інтерфейси та зчитує локальний конфігураційний файл config.json. У цьому файлі описується перелік необхідних бібліотек, версія бізнес-логіки та структура конвеєра обробки даних. Програмний модуль звіряє локальні ресурси з маніфестом і, за потреби, звертається до центрального сервера керування (реалізованого на базі платформи .NET) для автоматичного завантаження відсутніх компонентів. Це дозволяє пристрою самостійно адаптуватися до нових завдань без зайвого втручання розробника в апаратну частину.

IV. Архітектура та програмна реалізація

Програмна реалізація клієнтської частини для мікроконтролерів сімейства ESP виконана мовою інтерпретованого програмування MicroPython. Це дозволяє використовувати парадигму об'єктно-орієнтованого програмування для розробки незалежних компонентів виконавчого конвеєра. Для серіалізації та десеріалізації маніфестів пристрою (config.json) використовується інтегрована оптимізована бібліотека ujson. Завантаження відсутніх виконуваних файлів під час фази ініціалізації у модулі boot.py реалізовано за допомогою HTTP-запитів до REST API бекенд-сервера, розробленого на базі фреймворку .NET.

Центральний вузол мережі, що базується на мікрокомп'ютері Raspberry Pi, розгорнуто у контейнеризованому середовищі Docker. Такий підхід забезпечує надійну ізоляцію процесів MQTT-брокера (наприклад, Eclipse Mosquitto) та локальних сервісів обробки даних, а також спрощує процедуру резервного копіювання.

Для реалізації асинхронного обміну повідомленнями на стороні мікроконтролера (у фазі main.py) задіяна бібліотека umqtt.simple. Обробка вхідних команд керування та відправка телеметрії відбувається у нескінченному циклі з використанням неблокуючих мережевих сокетів, що дозволяє уникнути програмного зависання IoT-вузла у разі тимчасової втрати з'єднання з бездротовою мережею. Усі завантажені з бекенду скрипти (драйвери специфічних сенсорів тощо) динамічно ініціалізуються в оперативній пам'яті за допомогою вбудованої функції __import__(). Це технічне рішення дозволяє імплементувати механізм «гарячого» підключення логіки, формуючи абстрактний конвеєр обробки безпосередньо під час виконання програми.

Висновок

У роботі запропоновано та обґрунтовано гнучку архітектуру програмного забезпечення для гетерогенних IoT-мереж для систем «розумного будинку». Розроблена дворівнева модель динамічного конфігурування, що базується на використанні середовища MicroPython та JSON-маніфестів, ефективно вирішує проблему масштабованості та складності адміністрування великої кількості задіяних мікроконтролерів. Показано, що логічне розділення життєвого циклу пристрою на фазу інтелектуального провіжингу та фазу виконання абстрактного конвеєра дозволяє динамічно змінювати бізнес-логіку вузлів «повітрям» (OTA) без необхідності повної перекомпіляції монолітних прошивок. Це суттєво знижує ризик виникнення критичних збоїв під час оновлення та зменшує навантаження на мережеву інфраструктуру. Використання протоколу MQTT гарантує надійну та асинхронну взаємодію кінцевих вузлів із центральним сервером керування. Подальші дослідження будуть спрямовані на інтеграцію механізмів криптографічного захисту процесу завантаження конфігурацій та оптимізацію алгоритмів енергозбереження для автономних сенсорних вузлів.

Список використаних джерел

1. MicroPython documentation. https://docs.micropython.org/.
2. MQTT Version 5.0. OASIS Standard. https://mqtt.org/mqttspecification/.
3. .NET IoT Libraries documentation.https://learn.microsoft.com/en-us/dotnet/iot/.

УДК 004.891

РОЗРОБКА ОНЛАЙН-СИСТЕМИ ДЛЯ ОРГАНІЗАЦІЇ ТА ПРОВЕДЕННЯ ОНЛАЙН-КУРСІВ АНГЛІЙСЬКОЇ МОВИ

Юшко А.В. [1], **Папаєвич Р.В.**[2]
Західноукраїнський національний університет
[1]д.ф., старший викладач; [2] бакалавр

I. Постановка проблеми

Сучасний етап розвитку освітньої сфери характеризується стрімким переходом до формату змішаного навчання, що вимагає ефективної інтеграції цифрових технологій у навчальний процес. Особливо гостро ця потреба відчувається у сфері вивчення іноземних мов. Аналіз ринку EdTech виявляє суттєві прогалини: наявні рішення представлені або простими мобільними додатками для самостійного навчання, або громіздкими та дорогими корпоративними системами (Enterprise LMS), які є негнучкими для малих та середніх мовних шкіл. На практиці комерційні освітні заклади часто змушені використовувати «латану» інфраструктуру, комбінуючи розрізнені інструменти: загальні CRM-системи, месенджери та хмарні сховища. Це призводить до надмірного адміністративного навантаження, ризику помилок при плануванні та тотальної фрагментації інформаційного середовища для студента (пошук посилань на заняття, матеріалів та домашніх завдань на різних платформах). Таким чином, виникає об'єктивна потреба у розробці спеціалізованої онлайн-платформи, яка б централізувала всі аспекти дистанційного навчання в єдиному цифровому просторі.

II. Мета роботи

Метою роботи є проєктування та розробка багатокомпонентної інформаційної системи (вебзастосунку) для повної автоматизації бізнес-процесів та навчальної логістики при проведенні онлайн-курсів англійської мови для певного корпоративного замовника. Головне завдання платформи - перевести управління навчальним процесом на модель самообслуговування з чітким рольовим розподілом доступу. Система повинна забезпечувати:

- для студентів: можливість реєстрації, вибору поточного рівня англійської мови (A1-C2), доступ до єдиного особистого кабінету з розкладом, підручниками та домашніми завданнями;
- для викладачів: інструменти для створення уроків, публікації матеріалів.

Найбільш критичною функціональною вимогою, що розв'язує проблему логістичного хаосу, є інтеграція єдиного прямого гіперпосилання на відеоконференцію Zoom що представлено на рисунку 1. Крім того, архітектура системи має відповідати суворим критеріям масштабованості, продуктивності та безпеки (зокрема, використання протоколу HTTPS та криптографічного хешування паролів).

Рисунок 1 – Вікно кабінету студента із списком уроків та універсальне посиланням на кожен із них

III. Архітектура та програмна реалізація

Для забезпечення гнучкості та стійкості системи до майбутніх змін було прийнято рішення відмовитися від класичної монолітної N-Tier архітектури на користь архітектурного патернуCleanArchitecture (Чиста архітектура). Цей підхід ізолює доменне ядро з бізнес-правилами (сутності ApplicationUser, Group, Lesson, Homework, EnglishLevel) від зовнішніх інфраструктурних компонентів (баз даних, інтерфейсів). Проєктування здійснювалося з неухильним дотриманням принципів об'єктно-орієнтованого дизайну SOLID для забезпечення слабкої зв'язності та високої когезії коду.

Серверна частина (Backend) реалізована на базі кросплатформного фреймворку .NET Core з використанням мови C#. Цей вибір забезпечує високу продуктивність вебсервера та дозволяє побудувати надійний RESTful API. Управління користувачами реалізовані за допомогою розширення ASP.NET CoreIdentity у поєднанні з токен-орієнтованою автентифікацією (JSON WebTokens - JWT).

Клієнтська частина (Frontend) розроблена з використанням бібліотеки React. Її компонентна архітектура дозволяє створювати незалежні елементи інтерфейсу для повторного використання, а технологія Virtual DOM гарантує миттєве оновлення динамічних даних в особистих кабінетах без перезавантаження сторінок.

IV. Аналітичне забезпечення та результати

Зберігання реляційних даних забезпечується СУБД Microsoft SQL Server із використанням EntityFrameworkCore (EF Core). У проєкті застосовано підхід Code-First, за якого структура бази даних автоматично формується на основі C#-класів доменної моделі через механізм міграцій.

Структуру бази даних нормалізовано до третьої нормальної форми (3НФ), що гарантує цілісність і усуває дублювання даних. Наприклад, рівні володіння мовою (A1–C2) зберігаються в таблиці EnglishLevel, тоді як у сутностях студентів і груп використовуються лише зовнішні ключі.

Для підвищення продуктивності частину критичної логіки винесено на рівень бази даних: SQL-тригер (AFTER INSERT для Lessons) автоматично створює шаблон домашнього завдання, а збережена процедура запобігає "подвійному бронюванню", перевіряючи перетини занять перед їх створенням.

Висновок

У результаті розробки було створено комплексну, безпечну інформаційну систему, що вирішує актуальну проблему інформаційної фрагментації під час надання онлайн-курсів англійської мови. Впровадження підходу CleanArchitecture, у поєднанні з потужністю стека .NET Core та React, дозволило реалізувати стабільний RESTful API та реактивний користувацький інтерфейс.

Спроєктована база даних із застосуванням EF CoreCode-First, глибокою нормалізацією (3НФ) та алгоритмічною оптимізацією (SQL-тригери та збережені процедури) гарантує транзакційну цілісність інформації та автоматизує ключові бізнес-процеси замовника. Запропонована платформа виступає не лише ефективним інструментом для заміни ручного адміністрування, але й масштабованим фундаментом для подальшої цифрової трансформації мовної школи.

Список використаних джерел

1. Martin R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. – 1st ed. – Boston: Pearson Education, 2017. – 432 p. ISBN 978-0-13-449416-6. URL: https://www.pearson.com/store/p/clean-architecture/P200000009528
2. A. Yushko, R. Shevchuk, K. Lopacinski, M. Leszczynska, O. Yashchyk and T. Yurchyshyn, “Shielding WebApplication against Cyber-Attacksusing SIEM,” 2023 13th International Conference on Advanced Computer Information Technologies (ACIT), Wroclaw, Poland, 2023, pp. 393–396. URL: 10.1109/ACIT58437.2023.10275630.
3. Ferreira A., GarcíaSalinas J., Morales S. Using a Task-Based Approach for Supporting a Blended Learning Model for Englishas a Foreign Language. International Journal of Computer-Assisted Language Learning and Teaching (IJCALLT), 2014, vol. 4, no. 1, pp. 44–62.URL: 10.4018/ijcallt.2014010104.
4. Krutko V., Spivak I., Krepych S. An approach to assessing the realibility of software systems based on a graph model method dependence. CEUR Workshop Proceedings, 2024, 3662, pp.37-47
5. Kremser W., Kranzinger S., Bernhart S. Design and Implementation of a Gesture-Aided E-Learning Platform. Sensors, 2021, vol. 21, no. 23, art. 8042. URL: 10.3390/s21238042.
6. Spivak I., Litvynchuk M., Spivak S. Validation and data processing in JSON format. EUROCON2021 – 19th IEEE International Conference on Smart Technologies, 2021, pp.326-330
7. Ethelbert O., Moghaddam F. F., Wieder P., Yahyapour R. A JSON Token-based Authentication and Access Management Schema for Cloud SaaS Applications. 2017 IEEE 5th International Conference on Future Internet of Things, Prague, CzechRepublic, 2017, pp. 47–53. URL: 10.1109/FiCloud.2017.29.

УДК 004.9:339.13

ВЕБ-ОРІЄНТОВАНА CRM-СИСТЕМА ДЛЯ ВТОРИННОГО РИНКУ АВТОМОБІЛІВ

Василик В.Б.[1)], Манжула В.І.[2)], Кшивак Д.І.[3)]
Західноукраїнський національний університет
[1)]бакалавр; [2)]д.т.н., професор; [3)] бакалавр

I. Постановка проблеми

Вторинний ринок транспортних засобів характеризується високим рівнем волатильності та значною кількістю детермінованих чинників, що впливають на фінансовий результат операцій. Окремий пласт формує сегмент імпорту пошкоджених автомобілів із міжнародних аукціонів для їх подальшої реновації та реалізації. Ефективність такої бізнес-моделі безпосередньо залежить від точності попереднього оцінювання лотів, що передбачає детальне прогнозування витрат на логістику, митне очищення та відновлювальні роботи. Відсутність спеціалізованого інструментарію для оперативного аналізу великих масивів даних призводить до помилок при формуванні аукціонних ставок. Впровадження веб-орієнтованої CRM-системи з інтегрованим математичним апаратом дозволяє автоматизувати розрахунок рентабельності та забезпечити об'єктивність прийняття управлінських рішень.

II. Мета роботи

Метою роботи є розробкавеб-орієнтованої CRM-системи для вторинного ринку автомобілів для автоматизованого обчислення прибутковості лоту, що дозволяє визначити оптимальну граничну ставку на аукціоні на основі комплексного аналізу сукупних витрат.

III. Математична модель оцінювання рентабельності інвестицій

Зміцнення фінансової стабільності бізнесу в умовах цифрової трансформації потребує застосування моделей, що базуються на розрахунку очікуваної рентабельності кожної операції. Показник ROI обрано основним критерієм для порівняльного аналізу активів за єдиною шкалою прибутковості. Математичне обґрунтування доцільності закупівлі полягає у зіставленні прогнозованого прибутку із сукупними витратами. Сукупний обсяг інвестицій I_{total} визначається як сума шести складових:

$$I_{total} = C_{veh} + C_{trans} + C_{parts} + C_{maint} + C_{lab} + C_{add} \quad (1)$$

Класифікацію та змістовне наповнення складових формули (1) наведено в Таблиці 1.

Таблиця 1

Вхідні параметри для розрахунку сукупного обсягу інвестицій

Позначення	Назва параметра	Функціональне значення складової
C_{veh}	Вартість придбання	Базова ціна лоту на аукціоні за результатами торгів (VehicleCost).
C_{trans}	Транспортні витрати	Логістичний супровід та обов'язкове митне оформлення (Transportation).
C_{parts}	Вартість запчастин	Сукупні витрати на комплектуючі для відновлення ТЗ (PartsCost).
C_{maint}	Сервісні роботи	Витрати на технічне та регламентне обслуговування (Maintenance).
C_{lab}	Оплата праці	Оплата послуг із діагностики та ремонту (LaborCost).
C_{add}	Додаткові витрати	Аукціонні збори, комісії платформи, витрати на оформлення документів та інші супутні витрати (Additional Cost)

Економічна доцільність операції оцінюється через показник рентабельності інвестицій ROI, який ілюструє співвідношення прогнозованого прибутку до загального обсягу капіталовкладень. Розрахунок показника здійснюється за формулою (2):

$$ROI = \frac{(P_{market} - I_{total})}{I_{total}} \times 100 \quad (2)$$

де P_{market} – середня ринкова ціна аналогічного автомобіля на вторинному ринку.

Додатне значення ROI підтверджує прибутковість угоди, тоді як від'ємне вказує на її фінансову недоцільність. Паралельно обчислюється маржа прибутку $ProfitMargin$, яка відображає частку чистого доходу у структурі вартості реалізації:

$$ProfitMargin = \left(1 - \frac{I_{total}}{P_{market}}\right) \times 100 \quad (3)$$

Для забезпечення коректності розрахунків формул (2) та (3) обов'язковою є перевірка умов $I_{total} > 0$ та $P_{market} > 0$.

Рекомендована максимальна ставка (B_{rec}) визначається шляхом алгебраїчного перетворення виразу для показника ROI з урахуванням заданого користувачем цільового рівня рентабельності:

$$B_{rec} = \frac{P_{market}}{1 + \frac{ROI_{target}}{100}} - C_{other} \quad (4)$$

де ROI_{target} – попередньо заданийцільовий рівень рентабельності, C_{other} – сукупність усіх прогнозованих витрат, окрім вартості безпосереднього придбання транспортного засобу, розрахунок здійснюється за формулою (5):

$$C_{other} = I_{total} - C_{veh} \quad (5)$$

IV. Архітектура та програмна реалізація

Розроблено архітектуру інформаційної системи, яка складається з трьох рівнів: збору даних (вхідні параметри лотів та аукціонні відомості), аналітичного ядра (серверна логіка на базі FastAPI) та інтерфейсу користувача (див.рис.1). Обмін даними реалізовано за допомогою REST API, що дозволяє інтегрувати систему із зовнішніми джерелами інформації про транспортні засоби та аукціонними платформами. Дані про параметри лотів експортуються у форматі JSON, після чого аналітичний модуль обробляє їх згідно з наведеним вище алгоритмом розрахунку сукупних інвестицій та рентабельності. Візуалізація результатів прогнозування здійснюється через веб-інтерфейс, побудований на бібліотеці React, у якому менеджер отримує зведену інформацію у вигляді динамічних розрахунків очікуваного прибутку та рекомендованої ставки Brec [1].

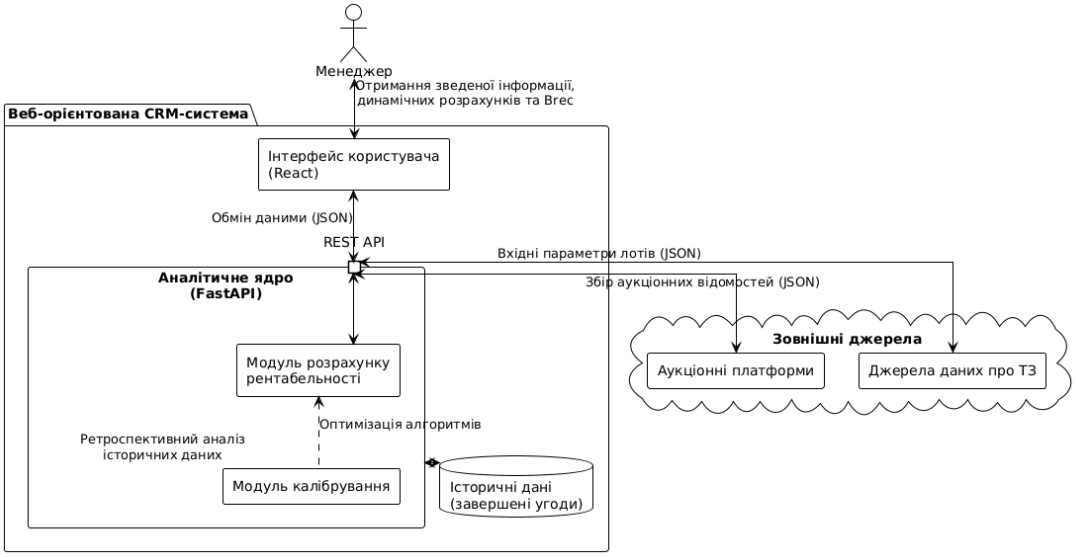

Рисунок 1 – Архітектура веб-орієнтованої CRM-системи

Важливим етапом впровадження запропонованого методу є процедура калібрування змінних складових витрат, зокрема вартості запчастин Cparts та ремонту Cmaint. Для цього передбачено використання ретроспективного аналізу історичних даних уже завершених угод підприємства. Порівнюючи розраховані системою значення рекомендованої ставки із фактично зафіксованими витратами на відновлення в минулому, можна оптимізувати алгоритми калькуляції за допомогою

методів статистичного моделювання та регресійного аналізу. Крім того, практична цінність розробленого підходу полягає у мінімізації впливу людського фактора на процес оцінювання. На відміну від традиційного аналізу лотів, який часто базується на суб'єктивних відчуттях менеджера або емоційних рішеннях під час торгів, запропонований підхід мінімізує вплив суб'єктивного сприйняття під час оцінювання ризиків та забезпечує безперервний моніторинг інвестиційної привабливості кожного об'єкта.

Діаграма діяльності на рисунку 2 візуалізує роботу алгоритму оцінювання рентабельності інвестицій на основі даних конкретного автомобіля.

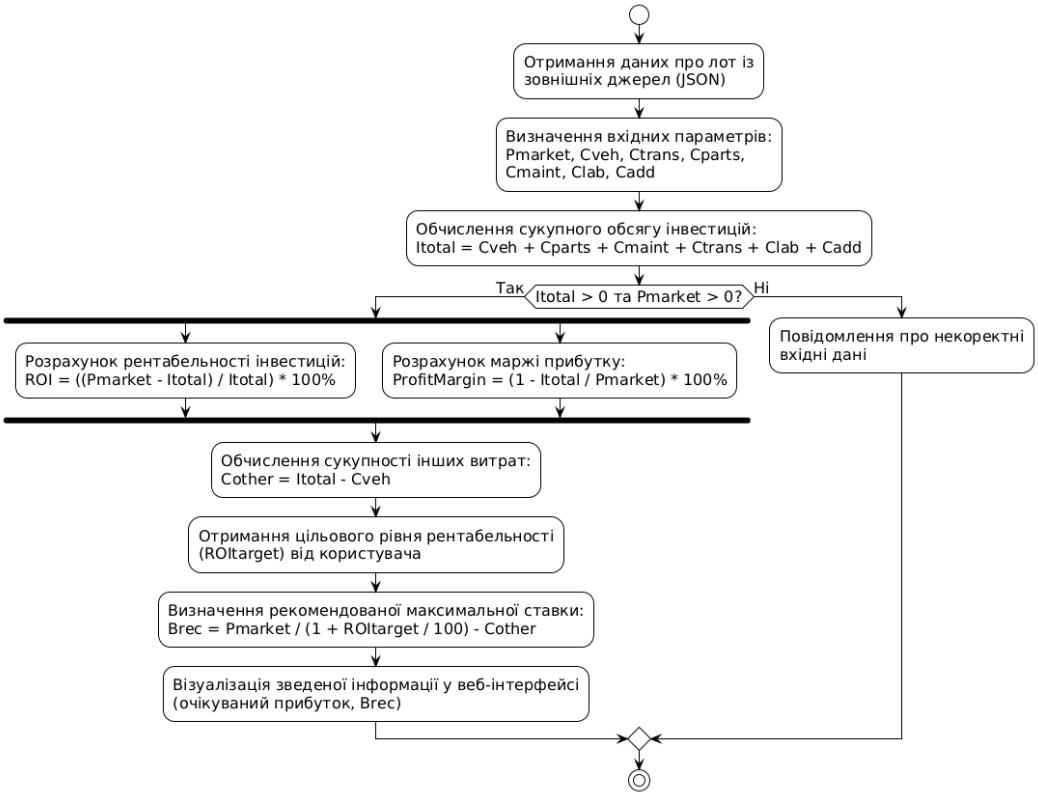

Рисунок 2 – Діаграма діяльності для алгоритму оцінювання рентабельності інвестицій

Ініціалізація та збір даних починається з отримання вхідних параметрів лотів та аукціонних відомостей у форматі JSON через REST API. На основі отриманих даних (базова ціна, логістика, запчастини, сервісні роботи тощо) обчислюється сукупний обсяг інвестицій. Система перевіряє обов'язкові умови коректності розрахунків: $I_{total} > 0$ та $P_{market} > 0$. Якщо умови не виконуються, розрахунок припиняється. Якщо умови виконуються, система паралельно обчислює показник рентабельності інвестицій та маржу прибутку. Система визначає інші прогнозовані витрат, враховує цільовий рівень рентабельності і розраховує кінцеву рекомендовану максимальну ставку. Останнім етапом є візуалізація результатів прогнозування через веб-інтерфейс для менеджера.

Висновки

У дослідженні реалізовано веб–орієнтовану CRM-систему, яка завдяки інтеграції математичних моделей ROI та Bгесзабезпечує перехід від інтуїтивного до системного автоматизованого оцінювання лотів. Впровадження такого інструментарію не лише гарантує високу фінансову прозорість кожної операції, а й стає надійним підґрунтям для формування найбільш ефективних стратегій участі в аукціонних торгах. Застосування розробленого підходу дозволяє мінімізувати вплив суб'єктивних чинників на прийняття управлінських рішень, що значно знижує інвестиційні ризики та підвищує прогнозованість фінансових результатів на вторинному авторинку.

Список використаних джерел

1. Гірна О. Б., Іваницький Р. Я., Маляр Р. В. Особливості імплементації CRM-систем у процеси обслуговування клієнтів на ринку автомобілів. Економічний простір. 2025. № 200. С. 136-143.
2. Харченко А. І., Вишняк М. Ю. Розробка CRM-компонентів інформаційної системи для мережі підприємств автосервісу. Проблеми інформаційних технологій. 2025. Т. 6. С. 523-524.
3. Паньків О. В. Мікросервісний вебзастосунок автосалону: магістерська дис. Київ: НТУУ «КПІ ім. Ігоря Сікорського», 2023. 112 с.

УДК 004.77

АРХІТЕКТУРА АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ВИРОБНИЧИМИ ПРОЦЕСАМИ ФЕРМЕРСЬКОГО ГОСПОДАРСТВА

Співак І.Я.[1)], Деременда В.І.[2)], Крепич С.Я.[3)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)]бакалавр; [3)]к.т.н., доцент

I. Постановка проблеми

Сучасні агропромислові підприємства активно впроваджують автоматизовані системи управління технологічними процесами (АСУ ТП), що суттєво підвищує ефективність виробництва та забезпечує стабільну якість кінцевої продукції. Водночас зростання складності таких систем висуває нові вимоги до фахівців у сфері інформаційних технологій — зокрема щодо технічної експлуатації, оперативної діагностики та підтримки апаратно-програмних комплексів в умовах безперервного виробничого циклу. Харчова промисловість є особливим випадком, де збій інформаційної системи несе не лише економічні збитки від простою, а й безпосередні ризики для безпечності продуктів.

Актуальність дослідження зумовлена необхідністю системного аналізу архітектури виробничих інформаційних систем агропромислового сектору та розробки ефективних алгоритмів технічного супроводу, що відповідають вимогам міжнародних стандартів безпеки харчових продуктів (HACCP, ISO 22000). Відсутність структурованих методик діагностики та підтримки АСУ ТП призводить до збільшення часу простою обладнання та зниження якості готової продукції.

II. Мета роботи

Метою роботи є аналіз архітектури та принципів функціонування АСУ ТП на базі фермерського господарства «ГАДЗ», дослідження інформаційних потоків між апаратними компонентами виробничої лінії, а також розробка алгоритмів технічного обслуговування і відновлення працездатності систем. Для досягнення мети визначено такі завдання: дослідити трирівневу архітектуру АСУ ТП відповідно до стандарту ISA-95, проаналізувати логіку обміну сигналами між датчиками та контролерами, на основі цього розробити алгоритм діагностики несправностей та дослідити функціональну логіку вузлів термічної обробки та автоматизованого пакування.

III. Основна частина

Виробнича інфраструктура ФГ «ГАДЗ» відповідає міжнародному стандарту ISA-95 і включає трирівневу ієрархію автоматизації, зображену на рисунку 1. Нижній (польовий) рівень охоплює датчики температури (PT100), тиску, оптичні сенсори наявності тари та виконавчі механізми — клапани, насоси і конвеєрні приводи. Середній рівень (управління) реалізовано на базі програмованих логічних контролерів (ПЛК), які збирають дані з сенсорної мережі та формують керуючі команди. Верхній (диспетчерський) рівень представлений панелями людино-машинного інтерфейсу (HMI) з візуалізацією параметрів, архівацією та моніторингом тривог.

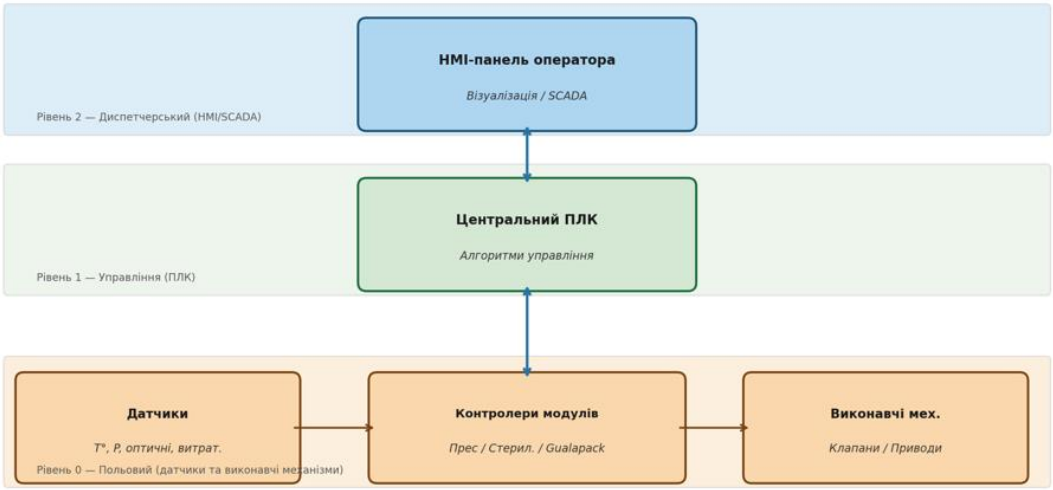

Рисунок 1 – Трирівнева архітектура АСУ ТП ФГ «ГАДЗ» (ISA-95)

Перевагою такої децентралізованої архітектури є відмовостійкість: при втраті зв'язку з верхнім рівнем локальні контролери продовжують керувати процесом у безпечному режимі. Ключовими технологічними етапами виробничого циклу є підготовка та миття сировини, пресування з автоматичним регулюванням зусилля, термічна обробка — пастеризація до 85–90°C для соків або UHT-стерилізація для пюре, деаерація та гомогенізація, асептичне зберігання в танка та; автоматизоване пакування на обладнанні Gualapack.

Спроектовано децентралізовану архітектуру системи управління з трьох незалежних технологічних модулів: вузол пресування, вузол стерилізації та пакувальний вузол Gualapack — кожен зі своїм локальним контролером. Принцип модульності дозволяє обслуговувати окремі вузли без зупинки всього виробничого комплексу. Розроблено п'ятикроковий алгоритм пошуку та усунення несправностей за методом виключення в ланцюзі «датчик — контролер — виконавчий механізм»: зчитування коду помилки з HMI; перевірка індикації на I/O-модулі ПЛК; апаратна ревізія датчика; аналіз вихідного керуючого сигналу; скидання помилки та тестовий перезапуск.

Для технічного супроводу системи розроблено тристанову модель обробки подій, яка забезпечує поетапну реакцію на зміну стану обладнання та дозволяє підвищити надійність функціонування виробничої лінії. Модель включає три базові стани: «Норма», «Попередження» та «Аварія», кожен із яких відповідає певному рівню відхилення параметрів від заданих значень і визначає відповідну стратегію реагування.

Стан «Норма» характеризує роботу обладнання в межах допустимих технологічних параметрів. У цьому режимі всі контрольовані показники (температура, тиск, вібрація, навантаження тощо) знаходяться в установлених межах, а система функціонує в автоматичному режимі без втручання оператора.

Стан «Попередження» активується у випадку, коли контрольовані параметри наближаються до гранично допустимих значень або демонструють аномальну динаміку зміни. У цьому режимі система генерує жовте сповіщення на операторській панелі (HMI), інформуючи персонал про потенційний ризик. При цьому виробничий процес не зупиняється, однак формується сигнал для проведення планового технічного огляду або діагностики.

Стан «Аварія» відповідає критичному відхиленню параметрів, яке може призвести до пошкодження обладнання, зниження якості продукції або виникнення небезпечної ситуації. У цьому випадку система ініціює автоматичну зупинку відповідного вузла або всієї виробничої лінії, а також формує сигнал тривоги для оператора.

Переходи між станами визначаються на основі порогових значень параметрів, а також можуть враховувати швидкість їх зміни, історичні дані та комбіновані умови (наприклад, одночасне відхилення кількох показників). Це дозволяє реалізувати більш гнучку та адаптивну модель моніторингу. Запропонована тристанова модель відповідає принципам планово-попереджувального обслуговування (Preventive Maintenance) та частково реалізує підхід прогнозного обслуговування (Predictive Maintenance) за рахунок раннього виявлення тенденцій до деградації. Її використання дозволяє зменшити кількість аварійних зупинок, оптимізувати графіки технічного обслуговування, підвищити ресурс обладнання та забезпечити безперервність виробничого процесу.

Висновок

У результаті роботи проведено системний аналіз архітектури АСУ ТП ФГ «ГАДЗ» відповідно до стандарту ISA-95. Досліджено специфікацію інформаційних зв'язків між датчиками, контролерами та виконавчими механізмами на всіх рівнях ієрархії. Розроблено алгоритм діагностики несправностей та тристанову модель обробки подій. Проаналізовано функціональну логіку вузлів термічної обробки та автоматизованого пакування. Здобуті результати формують наукову та практичну основу для вдосконалення інформаційних систем підприємств харчової промисловості та розробки інтегрованої платформи моніторингу виробничих процесів у рамках кваліфікаційної роботи.

Список використаних джерел

1. ANSI/ISA-95.00.01-2010. Enterprise-Control System Integration. Part 1: Models and Terminology. International Society of Automation, 2010.

2. S.Krepych, P.Stakhiv, I.Spivak. Analysis of the tolerance area parameters REC based on technological area scattering. 12-th International Conference "The Experience Of Designing And Application Of CAD Systems in Microelectronics" Polyana Svalyava. 2013. – P.179-180

3. Лавров С. М. Автоматизація технологічних процесів у харчовій промисловості: навч. посіб. — К.: Центр учбової літератури, 2021. — 240 с.

4. Закон України «Про основні принципи та вимоги до безпечності та якості харчових продуктів» (щодо впровадження систем HACCP). URL: https://zakon.rada.gov.ua

5. Gualapack Group. Технічна документація обладнання для гнучкого пакування. URL: https://gualapack.com

УДК 004.4

МОДУЛЬ БРОНЮВАННЯ МІСЦЬ НА ГРОМАДСЬКІ ПОДІЇ

Крепич С.Я.[1)], Молдавчук А.В.[2)], Співак І.Я.[3)], Крепич Р.В.[4)]
[1-3)] Західноукраїнський національний університет
[4)] ВСП Кам'янець-Подільський фаховий коледж НРЗВО "Кам'янець-Подільський державний інститут"
[1)]к.т.н., доцент; [2)] бакалавр; [3)]к.т.н., доцент; [4)] викладач

I. Постановка проблеми

При організації масових заходів критичним етапом є процес резервування місць. Існуючі рішення часто стикаються з проблемами конкурентного доступу (overbooking), відсутності автоматизованого контролю вікових обмежень та складності управління життєвим циклом події. Для локальних організаторів у місті Тернопіль необхідний інструмент, який би вирішував ці проблеми у рамках єдиної системи. Тому актуальним завданням є розробка надійної алгоритмічної підсистеми бронювання для платформи "EvenTer", що базується на чіткому контролі станів об'єктів.

II. Мета роботи

Метою роботи є проєктування та програмна реалізація підсистеми бронювання міських подій, яка забезпечує автоматичний контроль ліміту вільних місць (Capacity), валідацію віку користувачів (MinAgeLimit) та управління життєвим циклом події за допомогою автоматів станів з використанням платформи .NET.

III. Основна частина

Підсистема бронювання розробленої платформи "EvenTer" функціонує на базі клієнт-серверної архітектури. Для формалізації бізнес-логіки процесу резервування місць було застосовано UML-моделювання. Ключовим елементом системи є алгоритм дій, описаний за допомогою діаграми активностей, що містить логічні розгалуження для перевірки умов (контроль ліміту вільних місць та валідація віку), на які динамічно реагує серверна логіка та графічний інтерфейс React-додатка (див. рис.1).

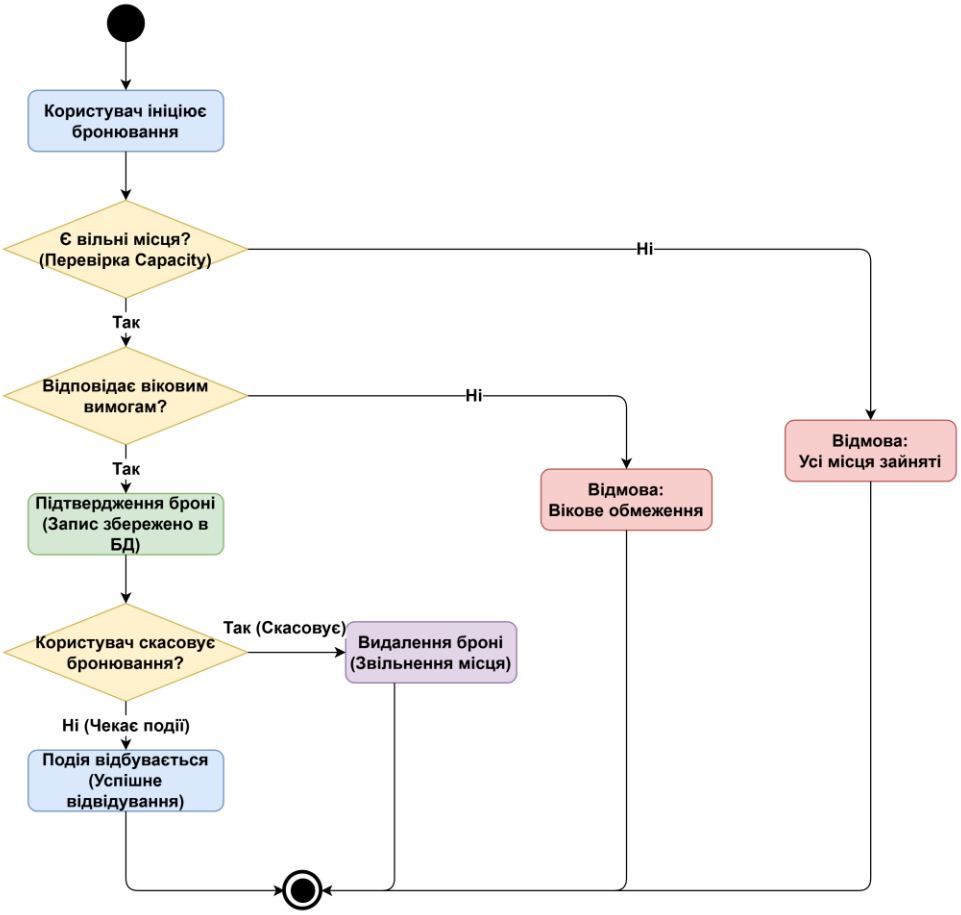

Рисунок 1 – Діаграма активностей процесу бронювання місць

Алгоритм бронювання (створення запису в таблиці Bookings) реалізовано у сервісному шарі Application. Перед транзакцією система виконує дві критичні перевірки:

1. Валідація віку: вираховується вік на основі DateOfBirth користувача з JWT-токена.
2. Контроль місткості: перевіряється, чи поточна кількість пов'язаних записів не перевищує атрибут Capacity.

Програмна реалізація даного алгоритму мовою C# наведена у лістингу 1.

```
publicasyncTask<bool>BookEventAsync(GuideventId, GuiduserId)
{
vartargetEvent = await _context.Events
        .Include(e =>e.Bookings)
        .FirstOrDefaultAsync(e =>e.Id == eventId);

varuser = await _context.Users.FindAsync(userId);

    // Перевірка вікових обмежень
intuserAge = DateTime.Today.Year - user.DateOfBirth.Year;
if (userAge<targetEvent.MinAgeLimit)
thrownewException("Відмовлено: захід має вікові обмеження.");

    // Перевірка наявності вільних місць
if (targetEvent.Bookings.Count>= targetEvent.Capacity)
thrownewException("На жаль, усі місця вже заброньовано.");

varbooking = newBooking { EventId = eventId, UserId = userId };
    _context.Bookings.Add(booking);
await _context.SaveChangesAsync();

returntrue;
}
```

Лістинг 1. Фрагмент коду методу бронювання події

Використання технології EntityFrameworkCore та СУБД MS SQL Server гарантує атомарність транзакцій, що повністю унеможливлює проблему паралельного бронювання одного місця двома різними користувачами. Візуальне представлення розробленої підсистеми для кінцевого користувача наведено на рисунку 2. Графічний інтерфейс містить інформацію про подію та відповідні елементи управління для підтвердження дії бронювання.

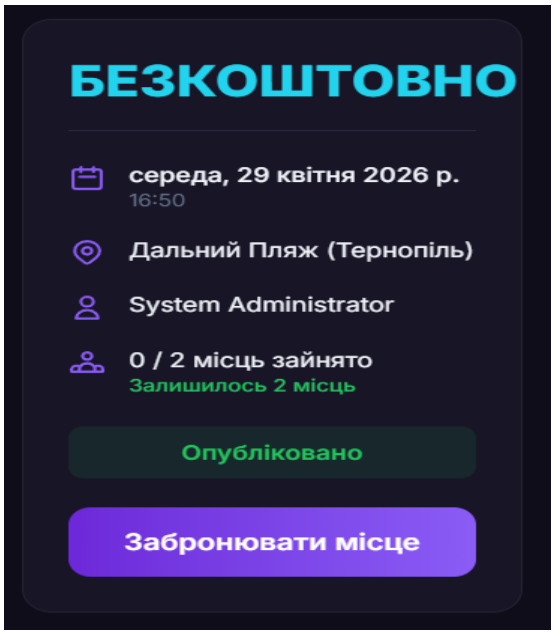

Рисунок 2 – Інтерфейс форми бронювання

Висновок

У роботі спроєктовано та програмно реалізовано підсистему бронювання для платформи "EvenTer". Завдяки використанню UML-діаграм станів вдалося чітко формалізувати життєвий цикл події. Розроблений C#-код серверної частини успішно вирішує проблеми вікового контролю та унеможливлює перевищення ліміту вільних місць. Практична реалізація підтвердила високу надійність обраної архітектури для вирішення реальних бізнес-задач організаторів подій.

Список використаних джерел

1. Троєлсен Е., Джепікс Ф. Мова програмування C# 7.0 і платформа .NET Core 2.0. — Київ : Вільямс, 2018. — 1328 с.
2. A.Bayurskii, S.Krepych, “Intelligent System Analyzing Quality of Land Plots”, International Conference “Advanced Computer Information Technologies” ACIT-2018, Ceske Budejovice, 2018. – pp. 166-169
3. Офіційна документація EntityFrameworkCore [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/ef/core/
4. Мартін Р. Чиста архітектура. Мистецтво розроблення програмного забезпечення. — Харків: Фабула, 2019. — 368 с.

УДК 004.4

АРХІТЕКТУРА ВЕБ-ЗАСТОСУНКУ EDUFLOW ДЛЯ ПЕРСОНАЛЬНОГО ПЛАНУВАННЯ РОБОТИ ВИКЛАДАЧА ПРОГРАМУВАННЯ

Палій С.В.[1)], Стасів І.С.[2)]
Західноукраїнський національний університет
[1)] студент; [2)] к.т.н., доцент

I. Постановка проблеми

Сучасна робота викладача програмування передбачає одночасне ведення трьох-восьми навчальних курсів, контроль виконання 30–150 практичних завдань на тиждень, моніторинг дедлайнів і координацію зворотного зв'язку зі студентами. Ці процеси у типовій практиці розподілені між кількома різнорідними сервісами: системою керування навчанням (Moodle), менеджерами задач (Todoist, TickTick), Trello-дошками для проєктних модулів та Notion-простором для конспектів і методичних матеріалів. Така фрагментованість призводить до дублювання введення даних, втрати контексту під час перемикання між інтерфейсами, реактивного режиму контролю дедлайнів та зростання когнітивного навантаження. Згідно з опитуваннями знаннєвих працівників, до 35–45 % робочого часу витрачається на адміністративні операції, безпосередньо не пов'язані з основною діяльністю [1,2]. Існуючі універсальні платформи спроєктовані переважно для команд і не закривають низку поведінкових сценаріїв одиночного користувача: офлайн-доступ без серверної синхронізації, миттєву персоналізацію без узгодження з командою, специфічну предметну модель «курс — задача — дедлайн», характерну саме для викладача програмування. Актуальність дослідження зумовлена відсутністю спеціалізованого single-user-інструмента, який поєднував би канбан-планування завдань, ведення курсів, контроль дедлайнів і базову аналітику в єдиному веб-інтерфейсі.

II. Мета роботи

Метою роботи є підвищення ефективності персональної організації викладацької діяльності шляхом розробки веб-застосунку EduFlow, що об'єднує канбан-планування завдань, ведення навчальних курсів, контроль дедлайнів і базову аналітику в єдиному офлайн-first-інтерфейсі з опційним шаром автентифікації на основі Supabase Auth.

III. Архітектура та модель єдиного робочого простору

Для реалізації застосунку обрано Next.js 16 App Router як мета-фреймворк, що поєднує переваги серверного рендерингу публічних сторінок (лендинг, форми авторизації) з повноцінним клієнтським дашбордом, котрий працює з даними у локальному сховищі браузера. Маршрутний шар використовує React Server Components для статичного вмісту і Client Components для інтерактивних фрагментів дашборду; автентифікація делегована Supabase Auth із cookie-based-сесією через middleware пакета @supabase/ssr. Порівняння трьох розглянутих архітектурних стилів за ключовими нефункціональними вимогами наведено у Таблиці 1.

Таблиця 1

Порівняння архітектурних стилів для single-user-застосунку

Критерій	**Classic SPA (CSR)**	**SSR-моноліт**	**Next.js App Router (RSC + CC)**
Перший показовий контент (LCP)	повільний	швидкий	швидкий
Офлайн-робота з дашбордом	повна	обмежена	повна (CC + localStorage)
Розмір клієнтського бандлу	великий	середній	мінімізований
Складність розгортання	низька	середня	низька (Vercel native)

Доменна модель EduFlow складається з п'яти сутностей: Task, Column, Course, UIPreferences та User. Для бізнес-даних обрано стратегію офлайн-first-зберігання у localStorage під версіонованими ключами «eduflow:<entity>:v1», доступ до яких інкапсульований в асинхронному шарі репозиторіїв із

контрактом list / getById / create / update / remove. Такий контракт дозволяє у наступній ітерації переключити persistence на Supabase Postgres з Row Level Security без зміни публічного API функціональних модулів. Для кількісного оцінювання навантаження робочого простору застосовано адаптовану форму закону Літтла з теорії черг, що пов'язує середній обсяг незавершеної роботи WIP (work-in-progress) із пропускною спроможністю TH (throughput) та часом циклу CT задачі (1).

$$WIP = TH \cdot CT \tag{1}$$

де TH – середня кількість задач, що завершують стан «Готово» за тиждень; CT – середній час від створення до завершення задачі. Для прийняття управлінських рішень додатково обчислюється інтегральний показник продуктивності тижня P, нормалізований до відрізка [0; 1] за формулою (2):

$$P = w_1 \cdot (N\_done / N\_planned) + w_2 \cdot (N\_in\_time / N\_done) + w_3 \cdot (1 - N\_overdue / N\_total) \tag{2}$$

де $w_1 + w_2 + w_3 = 1$ – вагові коефіцієнти, N_done – кількість виконаних задач за тиждень, $N\_planned$ – запланованих, N_in_time – виконаних до дедлайну, $N\_overdue$ – прострочених, N_total – усіх активних задач. Розраховані значення WIP, TH, CT та P аналітичний модуль EduFlow подає у вигляді тижневих діаграм, що дозволяє виявляти перевантаження робочого простору та коригувати планування у проактивному режимі замість реактивного «гасіння» прострочених задач.

IV. Програмна реалізація та оцінка ефективності

Стек реалізації становить Next.js 16, React 19 (з useOptimistic для миттєвого drag-and-drop на канбан-дошці), TypeScript 5.4 у strict-режимі, Tailwind CSS 4 із дизайн-токенами, експортованими з Figma, Supabase Auth для автентифікації, Vitest 1.6 та React Testing Library 15 для модульного й компонентного тестування, Playwright 1.45 для E2E-сценаріїв на Chromium і Firefox, а також Lighthouse CI для контролю Core Web Vitals. Структура коду побудована за принципом feature-based architecture: бізнес-логіка кожного функціонального модуля (board, tasks, courses, analytics, settings) зосереджена в окремому каталозі разом із репозиторієм, сервісами, компонентами та типами; загальний обсяг реалізації становить близько 9 800 рядків TypeScript / TSX.

Експериментальна оцінка проведена на тестовій вибірці з 250 задач, 8 курсів та чотирьох повних тижнів роботи. Результати модульного й компонентного тестування – 124 пройдених перевірки із середнім покриттям функціональних модулів 71 % при пороговому значенні 70 %; шар сховища даних свідомо покритий на 94 % як критичний для цілісності. E2E-набір – 18 сценаріїв ключових користувацьких маршрутів (вхід-реєстрація, створення курсу, перетягування карток, фільтрація задач, перемикання теми) на двох браузерних рушіях. Метрики Core Web Vitals на production-розгортанні Vercel: LCP ≈ 1,4 с, INP ≈ 96 мс, CLS ≈ 0,02, що відповідає категорії «Good» за класифікацією web.dev [3]. Перевірка безпеки за OWASP Top 10:2021 не виявила критичних вразливостей, чому сприяло делегування автентифікації Supabase Auth з RLS-готовою схемою БД, валідація форм через Zod-схеми та відмова від клієнтського зберігання чутливих токенів.

Висновок

У роботі запропоновано single-user-архітектуру веб-застосунку для персонального планування роботи викладача програмування, що поєднує канбан-планування, ведення курсів, контроль дедлайнів та базову аналітику в єдиному офлайн-first-інтерфейсі. Розроблений застосунок EduFlow забезпечує консолідацію адміністративних задач, які раніше були розподілені між чотирма-п'ятьма сервісами, та за оцінками практичної цінності скорочує час на адміністрування викладацького процесу на 35–45 %. Введена кількісна модель робочого простору на основі закону Літтла та зваженого показника продуктивності дозволяє об'єктивно оцінювати навантаження тижня замість суб'єктивного сприйняття. Подальші дослідження спрямовані на перенесення persistence-шару на Supabase Postgres зі збереженням контракту репозиторіїв, інтеграцію з LMS Moodle через відкрите API та розгортання як PWA для повноцінних офлайн-сценаріїв на мобільних пристроях.

Список використаних джерел

1. Microsoft Work Trend Index 2024: AI at Work Is Here. Now Comes the Hard Part. Microsoft, 2024. URL: https://www.microsoft.com/en-us/worklab/work-trend-index
2. Notion Labs. State of Knowledge Work 2024. Notion, 2024. URL: https://www.notion.so/blog/state-of-knowledge-work
3. Web Vitals. web.dev. URL: https://web.dev/articles/vitals
4. Next.js App Router Documentation. Vercel Inc. URL: https://nextjs.org/docs/app
5. Supabase Documentation. Supabase Inc. URL: https://supabase.com/docs
6. Allen D. Getting Things Done: The Art of Stress-Free Productivity. Revised ed. New York : Penguin Books, 2022. 352 p.

УДК 004.94

АДАПТИВНИЙ ЦИКЛ РЕАБІЛІТАЦІЇ ВЕРХНІХ КІНЦІВОК НА ОСНОВІ AR-ТЕХНОЛОГІЙ

Цапів Я.А.
Західноукраїнський національний університет
аспірант

I. Постановка проблеми

Ефективність фізичної реабілітації верхніх кінцівок суттєво залежить від здатності терапевта своєчасно адаптувати терапевтичний протокол до індивідуальної динаміки відновлення пацієнта. Традиційний підхід до управління реабілітаційним процесом є переважно реактивним: рішення щодо зміни інтенсивності або характеру вправ приймаються на основі ретроспективного аналізу після завершення чергового сеансу. Це не дозволяє оперативно реагувати на уповільнення прогресу. Відсутність формалізованих механізмів прогнозування обмежує потенціал цифрових реабілітаційних платформ, залишаючи прийняття ключових терапевтичних рішень виключно на суб'єктивній оцінці спеціаліста. Водночас розвиток технологій доповненої реальності та комп'ютерного зору відкриває нові можливості для створення інтелектуальних систем підтримки реабілітації з автоматизованим моніторингом та прогнозуванням .

II. Мета роботи

Метою даної праці є розробка адаптивного циклу прогнозування та корекції терапевтичного протоколу, що реалізує замкнений контур керування реабілітаційним процесом верхніх кінцівок з використанням технологій доповненої реальності та арт-терапії.

III. Адаптивний цикл реабілітації

Запропоновано перехід від реактивного до проактивного управління реабілітаційним процесом шляхом побудови адаптивного циклу, що реалізує замкнений контур керування і складається з п'яти послідовних етапів: 1) виконання терапевтичного завдання пацієнтом у AR-середовищі з арт-терапією; 2) автоматизоване вимірювання кутів рухливості суглобів системою безмаркерної гоніометрії; 3) побудова інтервальних прогнозних моделей динаміки відновлення; 4) аналіз прогнозованої траєкторії та формування рекомендацій; 5) реалізація корекції через зміну терапевтичного зображення та параметрів AR-сесії (див.рис.1).

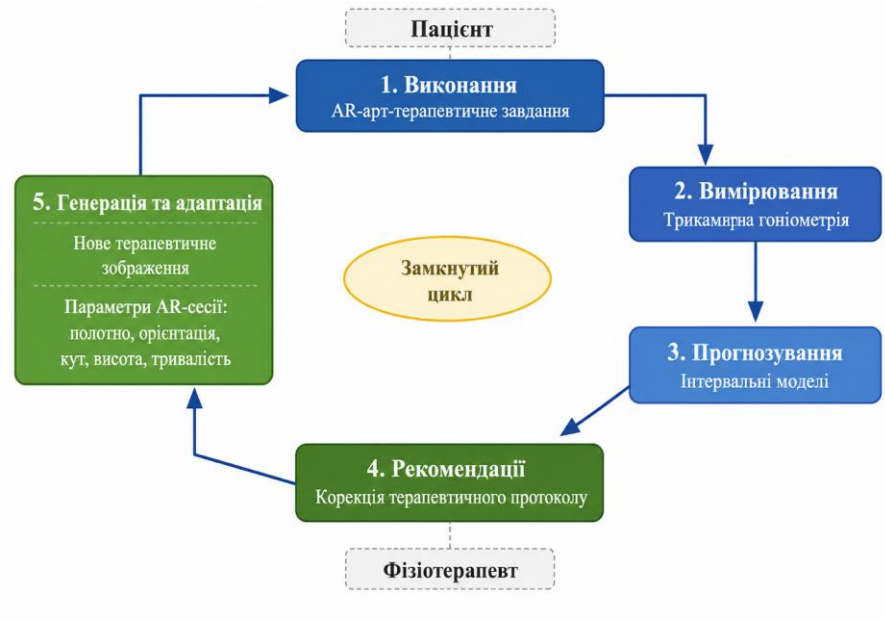

Рисунок 1 – Схема замкнутого адаптивного циклу реабілітації верхніх кінцівок

Модуль безмаркерної гоніометрії забезпечує автоматизоване вимірювання кутів рухливості суглобів без використання фізичних маркерів чи датчиків. Система використовує три стаціонарні камери, розташовані ортогонально, та поєднує два нейромережеві детектори ключових анатомічних точок: YOLOv8-pose та MediaPipe Pose Landmarker. Координати ключових точок обробляються адаптивним фільтром OneEuro-фільтр з адаптивною частотою зрізу, що забезпечує одночасно стабільність у стаціонарному режимі та швидкий відгук при різких рухах. Алгоритм геометричного злиття вимірювань від трьох камер із пріоритизацією на основі площин руху мінімізує проекційну похибку.

Модуль прогнозування використовує інтервальні дискретні динамічні моделі у вигляді різницевих рівнянь для опису динаміки відновлення кутів рухливості суглобів. На відміну від точкових оцінок, інтервальний підхід враховує невизначеність вимірювань та представляє прогнозовані значення у вигляді інтервалів можливих значень. Для ідентифікації параметрів моделі застосовано метаевристичний алгоритм на основі поведінкової моделі бджолиної колонії (Artificial Bee Colony), що включає фази робочих бджіл, бджіл-дослідників та бджіл-розвідників із механізмом виходу з локальних мінімумів . Після кожного сеансу фактично виміряне значення кута перевіряється на належність до прогнозного інтервалу: за умови адекватності модель зберігається, інакше виконується її переідентифікація на оновленому часовому ряді.

Компонент аналізу прогнозу порівнює прогнозовані інтервальні значення з цільовими показниками, визначеними на основі клінічних стандартів та індивідуальних цілей реабілітації. Система класифікує стан процесу: позитивна динаміка, помірне відхилення або негативний прогноз. У разі виявлення відхилення система ініціює формування рекомендацій щодо корекції терапевтичного протоколу. Ключовим механізмом корекції є зміна терапевтичного зображення-розмальовки, яке пацієнт малює на віртуальному AR-полотні. Просторовий розподіл елементів зображення безпосередньо визначає біомеханічний профіль рухів: вертикально орієнтовані елементи стимулюють згинання плечового суглоба, горизонтально орієнтовані – відведення, а дрібні деталі – точні рухи зап’ястка. Зміна зображення є неінвазивним терапевтичним втручанням: пацієнт продовжує виконувати арт-терапевтичне завдання, не усвідомлюючи зміни стратегії.

Для автоматизації генерації терапевтичних зображень застосовано багатоступеневий AI-контур. Текстова рекомендація, сформована на основі аналізу прогнозу, автоматично трансформується великою мовною моделлю (LLM) у структурований prompt для генеративної моделі. LLM отримує контекст про арт-терапевтичне призначення та клінічну рекомендацію, на основі чого генерує опис візуальних характеристик зображення: орієнтацію елементів, їх розподіл, рівень деталізації та стиль контурів. Отриманий prompt передається до генеративної моделі, яка створює контурне зображення-розмальовку для AR-полотна.Додатково враховується ширина прогнозного інтервалу як міра невизначеності: при надмірно широкому інтервалі система рекомендує консервативну стратегію зі збереженням поточних параметрів, що запобігає прийняттю рішень на основі ненадійних прогнозів. Кожна рекомендація супроводжується кількісним обґрунтуванням, що дозволяє фізичному терапевту критично оцінювати доцільність корекції. Система є рекомендаційною: остаточне рішення завжди залишається за спеціалістом.Додатковим напрямком корекції є налаштування параметрів AR-сесії: розмір та орієнтація полотна, кут нахилу, висота відносно очей пацієнта та тривалість сесії. Збільшення розміру полотна та його вертикальна орієнтація стимулюють рухи з більшою амплітудою згинання, тоді як горизонтальне розташування сприяє рухам відведення.

Висновок

У роботі представлено адаптивний цикл прогнозування та корекції терапевтичного протоколу для реабілітації верхніх кінцівок на основі AR-технологій. Запропонований адаптивний контур «виконання – вимірювання – прогноз – рекомендація – корекція зображення – адаптація рухів» інтегрує безмаркерну гоніометрію, інтервальне прогнозування на основі алгоритму бджолиної колонії та генеративний AI-контур для створення терапевтичних зображень. Перехід від реактивного до проактивного управління дозволяє здійснювати корекцію до появи негативних наслідків. Механізм корекції через зміну арт-терапевтичного контенту забезпечує неінвазивне втручання, що підтримує мотивацію пацієнта. Впровадження адаптивного циклу створює основу для персоналізованої реабілітації, у якій терапевтичний протокол безперервно адаптується до прогнозованої динаміки відновлення рухливості суглобів.

Список використаних джерел

1. Chelsea Cheng, Mohamed Elhassan Elamin, Helen May, Miriam Kennedy “Drawing on emotions: the evolving role of art therapy”. Published online by Cambridge University Press: April 19, 2021, doi.org/10.1017/ipm.2021.20
2. Sang S Pak, Dora Janela, Nina Freitas et al. “Comparing Digital to Conventional Physical Therapy for Chronic Shoulder Pain: Randomized Controlled Trial”, DOI: 10.2196/49236.
3. Пукас А. В., Цапів Я. А. Автоматизована безмаркерна система вимірювання діапазонів рухів суглобів на основі трикамерного відеоаналізу // Наукові праці ДонНТУ. Серія: «Інформатика, кібернетика та обчислювальна техніка». – 2026. – № 1(42). – С. 54–64. DOI: 10.31474/1996-1588-2026-1-42-54-64
4. Цапів Я. А., Тихий Р. Р. Інноваційна вебсистема гоніометричного аналізу на базі нейронних мереж // Наукові праці ДонНТУ. Серія: «Інформатика, кібернетика та обчислювальна техніка». – 2025. – № 1(40). – С. 80–88. DOI: 10.31474/1996-1588-2025-1-40-80-88

УДК 004.4

AI-ОРІЄНТОВАНИЙ ПІДХІД ДО ПОПЕРЕДЖЕННЯ ТОКСИЧНОЇ КОМУНІКАЦІЇ У СОЦІАЛЬНИХ МЕРЕЖАХ

Тимчишин В.С.[1)], Сімашко І.В.[2)]
Західноукраїнський національний університет
[1)]д.філ., ст. викладач; [2)]магістрант

І. Постановка проблеми

Соціальні мережі є невід'ємною частиною сучасного інформаційного середовища, забезпечуючи швидку взаємодію між користувачами та обмін думками у глобальному масштабі. Разом із перевагами відкритого спілкування виникає проблема поширення небажаного контенту, зокрема токсичних повідомлень.

Особливістю сучасної онлайн-комунікації є те, що значна частина негативних висловлювань виникає спонтанно, без попереднього наміру нашкодити. Користувачі часто публікують емоційні повідомлення, не усвідомлюючи їх можливого впливу на інших. У таких випадках класичні методи модерації, що працюють вже після публікації, не запобігають поширенню конфліктів.

Існуючі підходи до боротьби з токсичністю, зокрема блокування або видалення контенту, є реактивними та не завжди ефективними. Вони не впливають на причини виникнення проблеми та можуть викликати негативну реакцію користувачів через відчуття цензури.

У зв'язку з цим актуальним є підхід, який передбачає не лише фільтрацію, а й попередження небажаної поведінки. Використання штучного інтелекту дозволяє аналізувати текст повідомлення ще до його публікації та надавати користувачу рекомендації щодо його корекції.

Таким чином, виникає необхідність у створенні системи, яка здатна виступати як допоміжний інструмент для користувача, зменшуючи рівень токсичності без жорстких обмежень.

ІІ. Мета роботи

Метою роботи є розробка прототипу системи попереднього аналізу повідомлень, яка використовує AI для виявлення потенційно токсичного контенту та інформування користувача перед публікацією. Для досягнення мети необхідно вирішити такі завдання:

- реалізувати перевірку тексту повідомлення до моменту його відправлення;
- забезпечити відображення підказок або попереджень у разі виявлення ризикованого змісту;
- організувати обробку запитів у реальному часі;
- створити просту логіку прийняття рішень щодо публікації повідомлення.

Запропонований підхід дозволяє мінімізувати кількість токсичних повідомлень без складних механізмів модерації та значних обчислювальних витрат.

ІІІ. Основна частина

Ключовим елементом розробленої системи є процес попереднього аналізу повідомлення, який виконується до моменту його публікації. Такий підхід дозволяє виявляти потенційно небажаний контент ще на етапі введення та мінімізувати його поширення.

Процес попереднього аналізу складається з кількох послідовних етапів (див. рис.1).

На першому етапі користувач вводить текст повідомлення у веб-інтерфейсі. У цей момент система ще не виконує жодних перевірок, що дозволяє забезпечити природний процес взаємодії без обмежень.

Другий етап передбачає ініціацію перевірки перед публікацією. Після натискання кнопки відправлення або під час введення тексту (залежно від реалізації), клієнтська частина формує запит до серверної логіки. Для підвищення швидкодії та зручності користувача запит може виконуватися асинхронно без перезавантаження сторінки.

На третьому етапі сервер отримує текст повідомлення та здійснює підготовку запиту до зовнішнього AI-сервісу. Це включає формування структури даних, яка передається через API, а також обробку можливих помилок, пов'язаних із мережею або сервісом.

Четвертий етап полягає у виконанні аналізу тексту за допомогою AI. Зовнішній сервіс оцінює зміст повідомлення, враховуючи контекст, лексику та емоційне забарвлення, і повертає результат у вигляді структурованих даних. Зазвичай це може бути оцінка токсичності або класифікація повідомлення.

На п’ятому етапі відбувається інтерпретація отриманого результату. Серверна частина застосовує просту логіку прийняття рішень, наприклад, використовуючи порогове значення для визначення допустимого рівня токсичності. Такий підхід дозволяє уникнути складних алгоритмів і водночас забезпечити достатню ефективність.

Завершальний етап - це формування реакції системи. У залежності від результату аналізу можливі такі варіанти:

• повідомлення визнається допустимим і публікується без обмежень;

• користувачу відображається попередження із рекомендацією змінити текст;

• у разі явно токсичного змісту публікація може бути тимчасово заблокована.

Додатково система може накопичувати статистику результатів аналізу для подальшого вдосконалення механізмів модерації. Збереження інформації про типові випадки токсичних повідомлень дозволяє проводити аналітику поведінки користувачів, визначати найбільш поширені категорії порушень та коригувати параметри перевірки. Такий підхід сприяє підвищенню точності роботи системи та адаптації до нових форм небажаного контенту.Важливою перевагою запропонованого підходу є його гнучкість та можливість інтеграції з різними AI-сервісами. Завдяки використанню API система може бути легко масштабована та модернізована без необхідності суттєвих змін у внутрішній архітектурі.

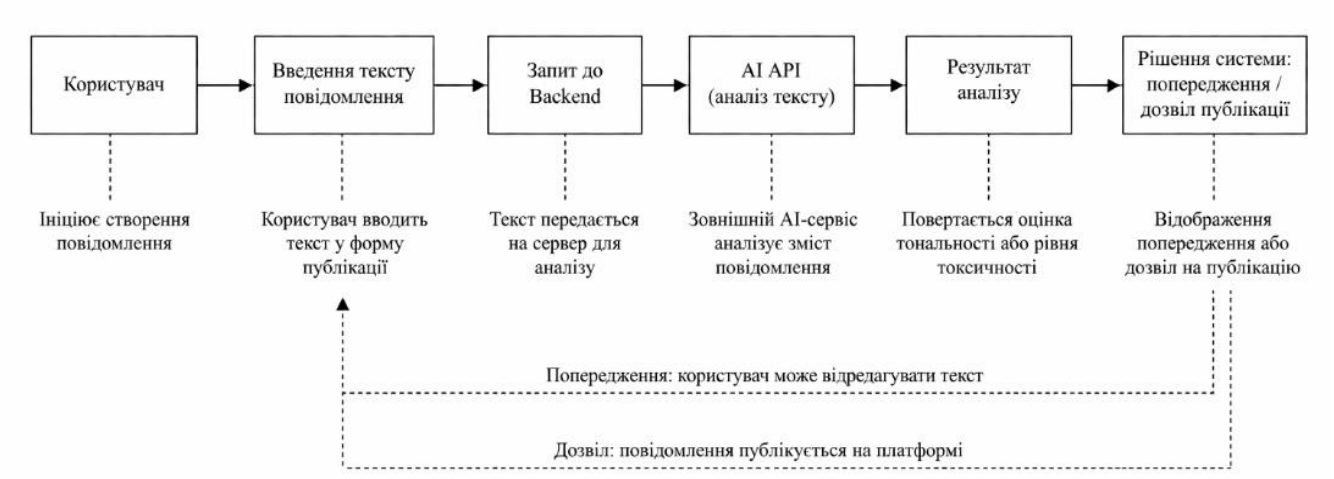

Рисунок 1 – Процес попереднього аналізу повідомлення

Важливою особливістю є наявність зворотного зв’язку: користувач має можливість відредагувати повідомлення та повторно пройти перевірку. Це реалізує ітеративний підхід до формування контенту та сприяє зниженню рівня токсичності.

Таким чином, процес попереднього аналізу поєднує просту технічну реалізацію з ефективною логікою взаємодії, що робить систему практичною для впровадження у веб-застосунках.

IV. Висновок

У результаті роботи було створено прототип системи попереднього аналізу повідомлень, яка дозволяє зменшити рівень токсичної комунікації у соціальних мережах. Основною особливістю підходу є орієнтація на попередження, а не на блокування контенту.

Запропоноване рішення є простим у реалізації та може бути інтегроване у різні веб-застосунки без значних змін архітектури. Використання AI як допоміжного інструменту дозволяє підвищити якість комунікації без жорсткого втручання у процес взаємодії користувачів.

У подальшому можливе розширення системи за рахунок додавання більш детальних рекомендацій, підтримки різних мов та аналізу поведінки користувачів.

Список використаних джерел

1. OpenAI API Documentation [Електронний ресурс] – Режим доступу: https://platform.openai.com/docs/
2. Ruby on Rails Guides [Електронний ресурс] – Режим доступу: https://guides.rubyonrails.org/
3. Google. Detecting Toxic Comments [Електронний ресурс]. – Режим доступу: https://www.perspectiveapi.com/
4. Fortuna P., Nunes S. A Survey on Automatic Detection of Hate Speech in Text // ACM Computing Surveys, 2018.
5. Schmidt A., Wiegand M. "A Survey on Hate Speech Detection using Natural Language Processing" // Proceedings of the 5th International Workshop on Natural Language Processing for Social Media, 2017.

УДК 004.41

АРХІТЕКТУРНІ ОСОБЛИВОСТІ ТА РЕАЛІЗАЦІЯ МАЛОБЮДЖЕТНОЇ SAAS-CRM СИСТЕМИ ДЛЯ ПІДПРИЄМСТВ МАЛОГО БІЗНЕСУ

Сірко Р.Т.[1)], **Манжула В.І.**[2)], **Дроздовський М.В.**[3)]
Західноукраїнський національний університет
[1)]студент; [2)]д.т.н., професор; [3)] студент

І. Постановка проблеми

У сучасних умовах стрімкої цифровізації ринку малий та середній бізнес гостро потребує надійних інструментів для систематизації взаємовідносин із клієнтами. Висока конкуренція диктує необхідність швидкої обробки лідів, проте проблема багатьох підприємств полягає у використанні неспеціалізованого ПЗ (наприклад, електронних таблиць чи побутових месенджерів) для ведення бази клієнтів.

Такий децентралізований підхід не підтримує повноцінну багатокористувацьку роботу, не має механізмів контролю цілісності інформації та унеможливлює побудову наскрізної аналітики, що закономірно призводить до втрати даних та зниження конверсії продажів. З іншого боку, перехід на професійні корпоративні системи часто стає непосильним бар'єром для мікро- та малих підприємств. Традиційні CRM-рішення часто є надто складними у налаштуванні під специфічні бізнес-процеси невеликих компаній та вимагають значних фінансових витрат на впровадження, купівлю дорогих ліцензій і навчання персоналу. Крім того, розгортання локальних рішень вимагає утримання власної серверної інфраструктури, що економічно не виправдано.

Архітектурна модель SaaS дозволяє ефективно вирішити ці проблеми шляхом надання хмарного доступу до необхідного функціоналу за гнучкою передплатою, мінімізуючи витрати компаній на серверне адміністрування. Проєктування системи за такою моделлю дозволяє розробникам забезпечити централізоване оновлення, високу доступність та надійний захист даних, залишаючись у рамках обмеженого бюджету малого бізнесу.

ІІ. Мета роботи

Метою дослідження є проєктування архітектури та розробка прототипу SaaS-системи для автоматизації базових операцій (облік контактів, ведення угод) з акцентом на швидкість розгортання та безпеку даних користувачів.

Для досягнення мети було поставлено такі завдання:

– проаналізувати наявні CRM-рішення для малого бізнесу та виявити їхні архітектурні й економічні недоліки;

– визначити функціональні вимоги до системи автоматизації та обрати оптимальний технологічний стек для забезпечення масштабованості;

– спроектувати архітектуру бази даних за принципом «Shared Database, Shared Schema» для реалізації багатоорендності;

– розробити програмний прототип, що включає модулі керування контактами, воронку продажів та аналітичну панель;

– впровадити механізми захисту даних на основі JWT-авторизації та ORM-захисту від ін'єкцій.

ІІІ. Обґрунтування отриманих результатів

Проєктована SaaS-CRM система базується на багаторівневій клієнт-серверній архітектурі, що взаємодіє через REST API. Ключовою вимогою до розробки сучасних SaaS-додатків є забезпечення багатоорендності. Для оптимізації витрат на інфраструктуру під час реалізації прототипу було обрано архітектурний підхід «Shared Database, Shared Schema». У цій моделі дані всіх компаній-клієнтів зберігаються в єдиній базі даних, проте кожен запис містить унікальний ідентифікатор орендаря (Tenant ID), що забезпечує надійну логічну ізоляцію інформації та запобігає несанкціонованому доступу до чужих даних (див. рис.1).

Технологічний стек системи логічно розділено на три базові рівні:

– клієнтський рівень (Frontend): Інтерфейс користувача реалізовано на базі бібліотеки React.js у форматі односторінкового додатка (SPA). Такий підхід мінімізує обсяг даних, що передаються по мережі, та забезпечує миттєвий відгук інтерфейсу на дії користувача без необхідності повного перезавантаження сторінок;

– серверний рівень (Backend): Ядром бізнес-логіки є сервер на платформі Node.js із використанням фреймворку Express для швидкої обробки асинхронних HTTP-запитів. Серверна частина відповідає за маршрутизацію, перевірку прав доступу та взаємодію з базою даних;

– рівень зберігання даних (Database): Як основне сховище використовується надійна реляційна СУБД PostgreSQL. Для абстрагування роботи з базою даних застосовується об'єктно-реляційне відображення (ORM) Sequelize, яке полегшує управління міграціями структури БД та гарантує захист системи від SQL-ін’єкцій на етапі формування запитів.

Для забезпечення безпеки та автентифікації користувачів імплементовано стандарт JWT (JSON Web Tokens). Використання токенів забезпечує безсесійну авторизацію, що значно знижує навантаження на серверну пам'ять та полегшує процес масштабування системи шляхом розгортання додаткових Docker-контейнерів.

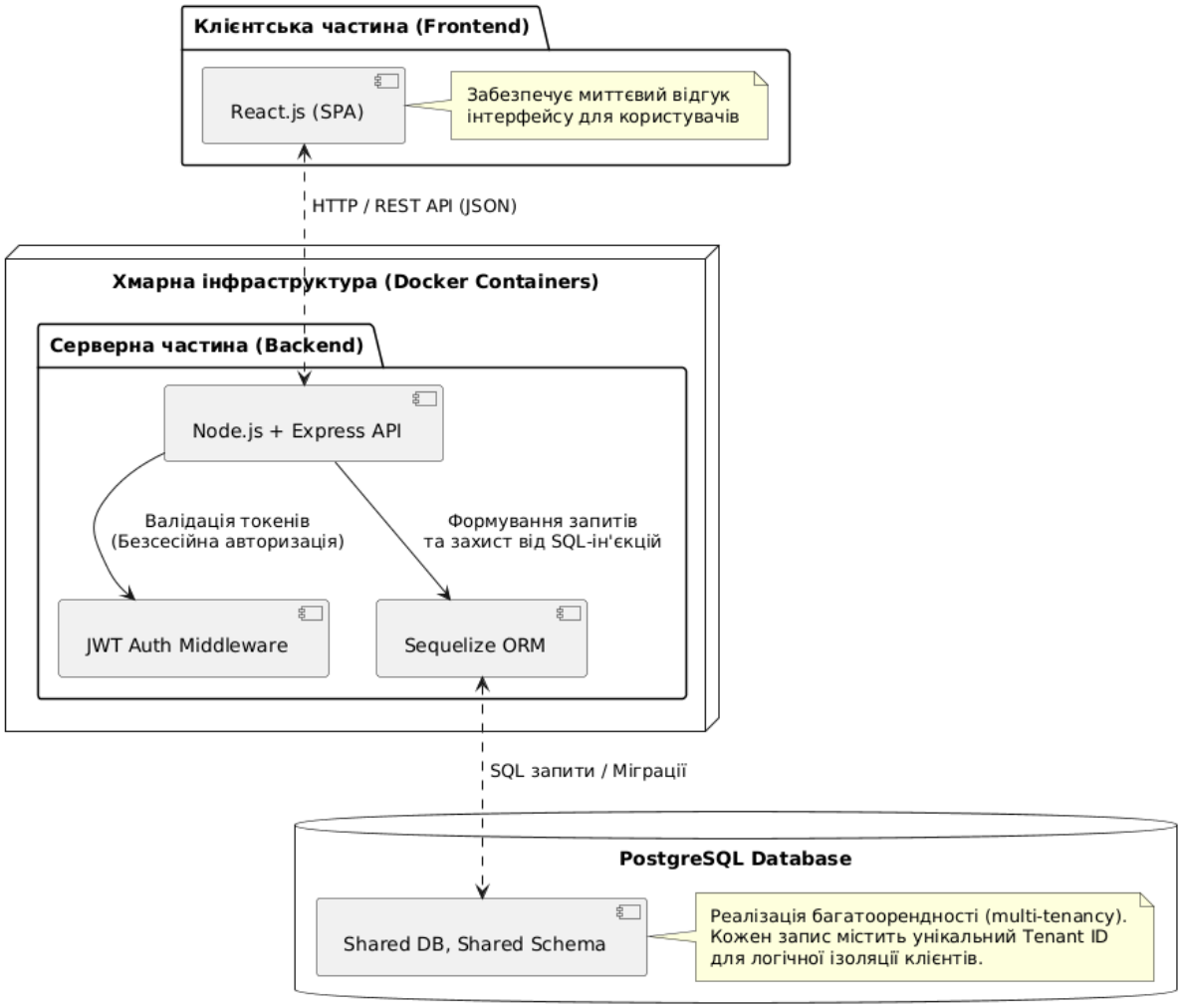

Рисунок 1 – Узагальнена схема архітектури системи

З точки зору функціонального наповнення згідно вимог (див.табл.1 та рис.2), архітектура системи поділяється на три взаємопов'язані модулі:

– модуль клієнтської бази: Забезпечує централізоване зберігання, швидкий пошук та гнучку фільтрацію даних контрагентів.

Таблиця 1

Порівняльна характеристика ролей користувачів

Роль користувача	**Опис функцій та рівень доступу в системі**
Адміністратор (Розробник)	Глобальне керування платформою: реєстрація нових компаній-орендарів, контроль технічного стану бази даних та оновлення ПЗ.
Керівник бізнесу	Доступ до аналітичного модуля: перегляд звітів про прибутки, моніторинг ефективності менеджерів та налаштування параметрів своєї філії.
Менеджер зі збуту	Операційна робота: ведення бази контактів, створення угод та переведення їх по статусах воронки (від «Новий» до «Завершено»).

– модуль угод (Sales Funnel): Призначений для візуалізації та контролю життєвого циклу продажу через систему інтерактивних статусів.

– модуль аналітики: Відповідає за агрегацію даних та генерацію звітів про ефективність роботи персоналу для прийняття управлінських рішень керівництвом.

Безпека реалізована через протокол JWT (JSON Web Tokens), що забезпечує безсесійну авторизацію та полегшує масштабування через Docker-контейнери.

Рисунок 2 – Діаграма прецедентів (Use Case Diagram)

Висновки

У ході дослідження було запроектовано та реалізовано архітектуру MVP-версії малобюджетної SaaS-CRM системи, яка дозволяє малому бізнесу отримати надійний інструмент обліку та систематизації клієнтської бази без значних капітальних інвестицій у власну інфраструктуру.

Використання моделі багатоорендності на рівні спільної бази даних дозволяє значно знизити вартість володіння системою, забезпечуючи при цьому достатній рівень логічної ізоляції даних кожного окремого клієнта за допомогою Tenant ID.

Поєднання React.js на фронтенді та Node.js на бекенді забезпечило високу швидкість відгуку інтерфейсу та здатність системи обробляти велику кількість асинхронних запитів у реальному часі. Застосування протоколу JWT для безсесійної авторизації не лише підвищило рівень захисту користувачів, а й створило фундамент для легкого масштабування системи через Docker-контейнеризацію.

Розроблення системи дозволить мінімізувати ризики втрати даних та помилок людського фактора, що зазвичай виникають при використанні електронних таблиць.мЗапропонований прототип створює надійну базу для подальшого нарощування функціоналу, зокрема через інтеграцію з IP-телефонією, системами масових розсилок та впровадження модулів AI-аналітики для прогнозування ймовірності закриття угод.

Список використаних джерел

1. Соммервілл І. Інженерія програмного забезпечення. 10-те вид. 765 с..
2. Turner M., Budgen D. Turning software into a service. Computer. 2003. Vol. 36, No. 10. P. 38–44.
3. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/.
4. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. O'Reilly Media, 2020. 374 p..
5. Buyya R., Vecchiola C., Selvi S. T. Mastering Cloud Computing: Foundations and Applications Programming. Morgan Kaufmann, 2013. 464 p..
6. Фоулер М. Архітектура корпоративних програмних додатків. Пер. з англ. 544 с.
7. Мартин Р. Чиста архітектура. Мистецтво розробки програмного забезпечення. 352 с.

УДК 004.891

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ЦИФРОВІЗАЦІЇ ТА ОПТИМІЗАЦІЇ HR-ПРОЦЕСІВ У НСОУ «ПЛАСТ»

Порплиця Н. П.[1)], **Муха Р.Б.**[2)], **Стасів І.С.**[3)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] бакалавр; [3)] к.т.н., доцент

І. Постановка проблеми

Під час управління діяльністю сучасних громадських та некомерційних організацій часто виникають проблеми збереження інституційної пам'яті, забезпечення безперервності процесів та ефективного менеджменту. В осередках таких організацій, зокрема в юнацьких куренях НСОУ «Пласт», значний обсяг внутрішнього документообігу, кадрового діловодства, фінансового обліку та організації масових заходів здійснюється переважно у паперовому форматі або лише частково оцифровується. Під час ротації кадрів та передачі повноважень неструктуровані обсяги даних, що зберігалися на особистих пристроях або в несистематизованих архівах, часто втрачаються. Втрата історичного контексту та звітності спричиняє дублювання роботи для нових керівників і ускладнює загальне адміністрування осередків.

Для вирішення вище зазначених проблем, доцільно впровадити в роботу НСОУ «ПЛАСТ» сучасне спеціалізоване програмне забезпечення. Із цією метою авторами було проведено комплексний аналіз ринку існуючого програмного забезпечення, зокрема HRM/ERP-систем. Однак результати показали відсутність комплексних аналогів, здатних повністю покрити специфічні вимоги внутрішнього менеджменту. Стандартні корпоративні рішення орієнтовані на комерційний сектор, є фінансово недоступними для волонтерських ініціатив і не враховують унікальної ієрархічної структури таких волонтерських організацій. Використання розрізнених безкоштовних інструментів не забезпечує єдиного доступу до даних та не дозволяє налаштувати гнучку систему розподілу прав доступу.

Тому актуальним завданням є розробка спеціалізованої інформаційно системи для НСОУ «ПЛАСТ», яка дозволить автоматизувати внутрішні процеси організації та забезпечить можливість ефективного менеджменту людських ресурсів організації.

ІІ. Мета роботи

Метою роботи є проєктування та розробка безпечної клієнт-серверної інформаційної системи ProjectK, орієнтованої на комплексне вирішення проблем цифровізації та оптимізації внутрішніх HR-процесів в НСОУ «ПЛАСТ», що обґрунтовано на основі результатів попереднього аналізу потреб організації. Створення програмного продукту спрямоване на автоматизацію рутинного документообігу, централізацію управління кадрами, фінансами та масовими заходами та ін. Це дозволить зменшити ризики втрати даних під час природної ротації волонтерських кадрів та забезпечити керівників осередків ефективним, безпечним інструментом для прийняття управлінських рішень.

ІІІ. Математичне забезпечення рекомендаційної системи

Архітектура інформаційної системи ProjectK базується на класичній клієнт-серверній моделі. На поточному етапі життєвого циклу програмного продукту базовим архітектурним стилем бекенду обрано монолітну архітектуру. Таке рішення обґрунтовано масштабом проєкту, оскільки воно забезпечує єдину кодову базу, спрощує процес налаштування CI/CD пайплайнів, мінімізує мережеві затримки між внутрішніми компонентами системи та гарантує транзакційну цілісність даних на етапі бета-тестування. Водночас, в основу серверної частини закладено багаторівневу структуру, що розділяє обробку HTTP-запитів, бізнес-логіку та взаємодію з даними. Для абстрагування бізнес-логіки від специфіки роботи з базою даних імплементовано архітектурний патерн «Repository».

Фундаментальним підходом до проєктування модулів системи є дотримання принципу інверсії залежностей, що є частиною парадигми SOLID [1]. Завдяки вбудованому механізму впровадження залежностей високорівневі модулі бізнес-логіки не залежать від низькорівневих модулів, а покладаються виключно на абстрактні інтерфейси [4]. Це суттєво знижує зв'язність коду та створює гнучкий фундамент для майбутнього масштабування архітектури системи, яку проілюстровано на рисунку 1.

Інформаційна модель даних спроєктована за гібридним принципом для оптимізації продуктивності. Жорстко структуровані транзакційні та реляційні дані керуються за допомогою реляційної алгебри. Водночас для роботи зі значними обсягами неструктурованих даних, типовими для документообігу некомерційних організацій, застосовується об'єктна модель зберігання Blob-storage, тоді як у базі даних зберігаються лише безпечні посилання на ці ресурси.

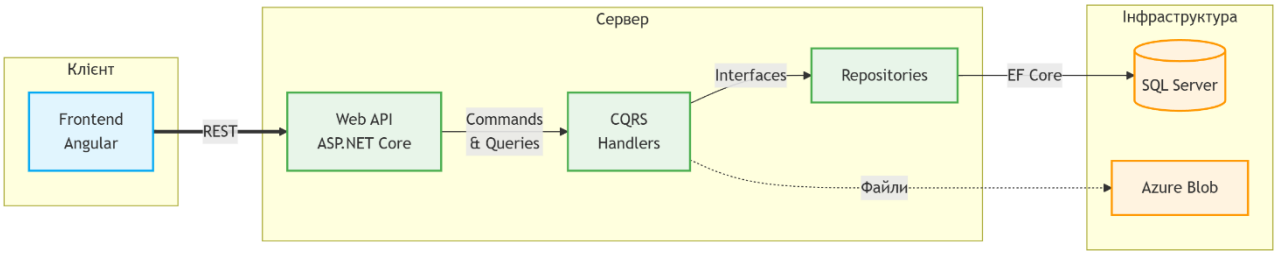

Рисунок 1 – Діаграма архітектури додатку ProjectK

IV. Архітектура та програмна реалізація

Програмну реалізацію системи ProjectK виконано із застосуванням сучасного технологічного стеку, що забезпечує високу продуктивність та масштабованість.

Серверна частина побудована на базі фреймворку .NET 10. Вибір цієї платформи зумовлений її високою швидкодією, кросплатформністю та потужними вбудованими інструментами для створення надійних RESTful API. Завдяки типізації та екосистемі C#, серверна частина ефективно обробляє складну бізнес-логіку кадрового та фінансового менеджменту в НСОУ «ПЛАСТ».

Як основне сховище даних НСОУ «ПЛАСТ» використано систему керування базами даних Microsoft SQL Server, яка гарантує цілісність та надійність збереження інформації. Для роботи з неструктурованими даними, зокрема електронними архівами та сканованими документами, інтегровано рішення Azure Blob Storage з використанням емулятора Azurite на етапі локальної розробки, що дозволяє ефективно керувати великими обсягами файлів без перевантаження основної реляційної бази даних [2].

Клієнтська частина реалізована за концепцією SPA з використанням фреймворку Angular. Для забезпечення інтуїтивно-зрозумілого та адаптивного користувацького інтерфейсу застосовано бібліотеку корпоративних UI-компонентів PrimeNG версії v.21. Дизайн системи та її візуальна гнучкість на різних пристроях реалізуються за допомогою utility-first фреймворку Tailwind CSS. Така комбінація фронтенд-технологій дозволяє оптимізувати процес розробки інтерфейсів та гарантує швидкий відгук системи на дії користувача [3].

Зважаючи на вимоги до відмовостійкості та безпеки, усю інфраструктуру розробленої інформаційної системи розгорнуто на хмарній платформі Microsoft Azure.

Висновок

У роботі обґрунтовано необхідність і практично реалізовано архітектуру інформаційної системи ProjectK, призначеної для комплексного управління HR-процесами, фінансами та документообігом у НСОУ «Пласт». Запропоноване рішення ефективно вирішує проблему фрагментації даних і втрати інституційної пам’яті, що історично виникала при використанні паперових архівів і локальних електронних таблиць під час ротацій волонтерського проводу.

Використання сучасного стеку технологій у поєднанні з надійною хмарною інфраструктурою Microsoft Azure дозволило створити безпечний, продуктивний та масштабований програмний продукт. Унікальні переваги системи включають високу модульність коду, централізацію даних, оперативну автоматизацію документообігу, а також гнучкість і готовність до розширення. Запуск закритого бета-тестування поточної версії v0.10.0-beta «Ant» підтверджує життєздатність платформи та дозволяє автоматизувати рутинне адміністрування та підвищити рівень інформаційної безпеки.

Список використаних джерел

1. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston: Prentice Hall, 2017. 432 p.
2. Price M. J. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals: Build intelligent apps, websites, and services. Birmingham: Packt Publishing, 2023. 820 p.
3. Freeman A. Pro Angular. 5th ed. New York : Apress, 2022. 860 p.
4. Seemann M., van Deursen S. Dependency Injection Principles, Practices, and Patterns. Shelter Island: Manning Publications, 2019. 552 p.

УДК 004.8

МАТЕМАТИЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ПРОГНОЗУВАННЯ УСПІШНОСТІ СТУДЕНТІВ У СИСТЕМІ УПРАВЛІННЯ НАВЧАЛЬНИМИ КУРСАМИ

Макарівський О.А.[1)], Папа О.А.[2)]
Західноукраїнськийнаціональнийуніверситет
[1)] магістрант; [2)] д.філ., доцент

I. Постановка проблеми

Системи управління навчанням (LMS) - Moodle, Google Classroom - набули широкого застосування у закладах вищої освіти і поступово оснащуються інструментами learning analytics. Утім, їхні вбудовані прогнозні моделі зорієнтовані переважно на загальні задачі (виявлення ризику відрахування, оцінка залученості) і не завжди адаптовані до прогнозування підсумкового бала студента в умовах конкретного курсу за поточними показниками його діяльності. У результаті проблемні ситуації виявляються здебільшого постфактум, без своєчасних коригувальних дій. Дослідження у сфері learning analytics свідчать, що раннє прогнозування за поточними показниками навчальної діяльності здатне суттєво знизити частку академічної неуспішності. Це обумовлює актуальність розробки відповідної методики та її реалізації у складі LMS.

II. Мета роботи

Метою роботи є побудова лінійної скорингової моделі прогнозування підсумкової успішності студентів за структурою множинної лінійної регресії, її програмна реалізація у складі клієнт-серверної системи управління навчальними курсами та пілотна апробація на реальній навчальній групі.

III. Математична модель та метод прогнозування

Прогнозований підсумковий бал студента $\hat{Y}$ обчислюється за лінійною скоринговою моделлю, побудованою за структурою множинної лінійної регресії від п'яти показників навчальної діяльності:

$$\hat{Y} = \beta_0 + \beta_1 T + \beta_2 P + \beta_3 A + \beta_4 D + \beta_5 L \quad (1)$$

де T – середній результат тестування; P – середній бал за практичні роботи; A – нормалізована активність студента у системі; D – частка завдань, виконаних у встановлений термін; L – частка опрацьованих навчальних матеріалів. Показники A, D, L попередньо нормалізовано до інтервалу [0,1] методом Min-Maxscaling. На пілотному етапі коефіцієнти моделі задано експертно з умови, що максимально досяжне значення $\hat{Y}$ за умови максимальних значень усіх показників становить 100 балів: $\beta_0 = 5$; $\beta_1 = 0.35$; $\beta_2 = 0.3$; $\beta_3 = 10$; $\beta_4 = 12$; $\beta_5 = 8$. На наступному етапі дослідження передбачено оцінювання коефіцієнтів за даними методом найменших квадратів із валідацією результатів.

Зазначенням $\hat{Y}$ автоматично визначається навчальний стан студента: зона ризику ($C_i = 0$) при $\hat{Y} < 60$; категорія додаткового контролю ($C_i = 1$) при $60 \le \hat{Y} < 75$; стабільний рівень ($C_i = 2$) при $\hat{Y} \ge 75$. Якість прогнозування оцінено за двома стандартними метриками - середньою абсолютною похибкою MAE та кореневою середньоквадратичною похибкою RMSE, які подаються в одиницях вимірювання цільової змінної (балах):

$$MAE = (1/n) \cdot \sum |Y_i - \hat{Y}_i| \quad (2)$$

$$RMSE = \sqrt{(1/n) \cdot \sum (Y_i - \hat{Y}_i)^2} \quad (3)$$

Загальну пояснювальну здатність моделі характеризує коефіцієнт детермінації R^2:

$$R^2 = 1 - \sum (Y_i - \hat{Y}_i)^2 / \sum (Y_i - \bar{Y})^2 \quad (4)$$

де $\bar{Y}$ - середнє значення фактичних підсумкових результатів.

IV. Програмна реалізація та результати пілотної апробації

Систему реалізовано як клієнт-серверний веб-застосунок: серверну частину побудовано на Node.js та Express.js, клієнтську - на React і TailwindCSS, дані зберігаються у PostgreSQL. Архітектура охоплює модулі обробки навчальних даних, прогнозування, класифікації навчального стану, формування рекомендацій та візуалізації результатів.

Пілотну апробацію проведено на навчальній групі з 10 студентів за даними одного семестру. Для кожного студента сформовано вектор показників $X = (T, P, A, D, L)$ і обчислено прогнозований підсумковий бал. Порівняння фактичних та прогнозованих результатів наведено на рисунку 1.

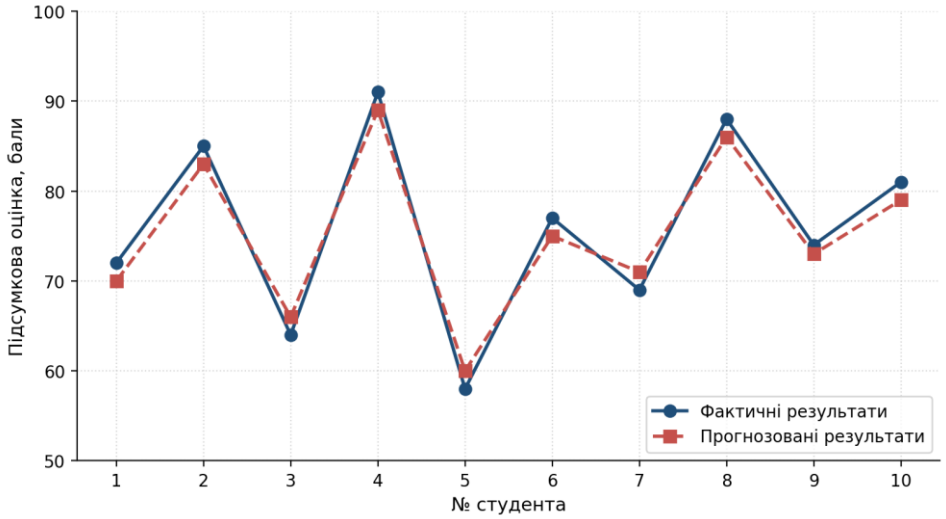

Рисунок 1 – Порівняння фактичних та прогнозованих оцінок студентів пілотної групи (n = 10)

Метрики якості прогнозування на пілотній вибірці склали MAE = 1,90 бала, RMSE = 1,92 бала, R^2 = 0,96; відхилення прогнозованих значень від фактичних не перевищують 2 балів. За класифікацією прогнозованих значень п'ятеро студентів віднесено до категорії додаткового контролю, ще п'ятеро - до категорії стабільного рівня; зону ризику модель не виявила. Для кожної категорії аналітичний модуль автоматично формує індикативну управлінську рекомендацію: індивідуальна консультація для зони ризику, моніторинг активності для категорії додаткового контролю, продовження стандартного режиму для стабільного рівня.

Обмеження пілотного етапу. Малий обсяг вибірки (n = 10) та експертний характер задавання коефіцієнтів означають, що отримані метрики відображають узгодженість каліброваної моделі з даними апробації, а не якість її узагальнення на нову вибірку. Подальші дослідження передбачають розширення вибірки до n ≥ 50, оцінювання коефіцієнтів моделі (1) на основі даних методом найменших квадратів із валідацією за k-foldcross-validation, впровадження кластеризації навчальних профілів та контентно-орієнтованих рекомендацій.

Висновок

Розроблено лінійну скорингову модель прогнозування підсумкової успішності студентів, побудовану за структурою п'ятифакторної множинної лінійної регресії, та реалізовано клієнт-серверну веб-систему управління навчальними курсами (Node.js, Express, React, PostgreSQL), яка автоматизує збір показників навчальної діяльності, прогнозування результатів і 90 бала, RMSE = 1,92 бала, R^2 = 0,96) підтвердила працездатність програмної реалізації та доцільність подальшого масштабування дослідження. Перспективи розвитку - розширення вибірки, оцінювання коефіцієнтів моделі формування індикативних рекомендацій для викладача. Пілотна апробація на групі з 10 студентів (MAE = 1, на основі даних методом найменших квадратів із валідацією за k-foldcross-validation, інтеграція алгоритмів кластеризації та персоналізованих рекомендацій.

Список використаних джерел

1. Romero C., Ventura S. Educational Data Mining and Learning Analytics: An Updated Survey. WIREs Data Mining and Knowledge Discovery. 2020. Vol. 10, No. 3. e1355.
2. Aldowah H., Al-Samarraie H., Fauzy W.M. Educational Data Mining and Learning Analytics for 21st Century Higher Education: A Review and Synthesis. Telematics and Informatics. 2019. Vol. 37. P. 13-49.
3. Hastie T., Tibshirani R., Friedman J. The Elements of Statistical Learning. 2nd ed. Springer, 2009. 745 p.
4. Власенко Н.В., Жук Ю.О. Системи підтримки прийняття рішень в освітніх процесах: підходи та моделі. Інформаційні технології в освіті. 2021. № 38. С. 115-122.

УДК 004.41

SAAS-ДОДАТОК ДЛЯ АВТОМАТИЗАЦІЇ БІЗНЕС-ПРОЦЕСІВ ЗАКЛАДІВ ГРОМАДСЬКОГО ХАРЧУВАННЯ

Каменюк Е.В.[1)], **Манжула В.І.**[2)]
Західноукраїнський національний університет
[1)]студент; [2)]д.т.н., професор

І. Постановка проблеми

Активний розвиток інформаційних технологій та цифрова трансформація бізнесу диктують нові стандарти якості обслуговування в індустрії громадського харчування (HoReCa). Сучасні заклади функціонують в умовах жорсткої конкуренції, що вимагає надзвичайно високого темпу роботи, мінімізації людського фактора та здатності обробляти значні масиви транзакційних даних у реальному часі.

Основна проблема функціонування багатьох малих та середніх підприємств цієї сфери полягає у фрагментарності процесів управління. Відсутність інтегрованої системи обліку або використання розрізнених програмних продуктів для залу (POS-термінали), кухні та складу призводить до десинхронізації інформаційних потоків. Як наслідок, виникають критичні розбіжності у складських залишках, збільшується час обробки замовлень, зростає ймовірність помилок при розрахунках з клієнтами та суттєво ускладнюється побудова наскрізної аналітики.

Традиційне десктопне програмне забезпечення часто є важко масштабованим і вимагає від власників закладів значних капітальних витрат на розгортання локальної серверної інфраструктури, придбання ліцензій та регулярне технічне обслуговування. У цьому контексті перехід до моделі SaaS на базі хмарних технологій є оптимальним архітектурним рішенням. Такий підхід забезпечує єдину точку доступу до централізованих даних, підтримує кросплатформність та мобільність персоналу, а також суттєво знижує початкові інвестиції за рахунок переходу до моделі передплати. Відповідно, розробка спеціалізованого SaaS-додатку для комплексної автоматизації процесів закладу громадського харчування є актуальним практичним завданням.

ІІ. Мета роботи

Метою дослідження є проєктування архітектури та розробка прототипу SaaS-системи, яка дозволить автоматизувати основні операційні процеси (продажі, склад, персонал) закладів громадського харчування, забезпечуючи високу доступність та ізоляцію даних користувачів.

ІІІ. Обґрунтування отриманих результатів

В ході дослідження було визначено, що ключовою особливістю SaaS-додатків є реалізація багатоорендності (multi-tenancy). Для проєктування обрано підхід з використанням єдиної бази даних, де кожен запис ідентифікується унікальним ID закладу. Це дозволяє ефективно масштабувати систему та забезпечувати оновлення програмного забезпечення для всіх клієнтів одночасно.

Архітектура системи базується на принципах REST API. Клієнтська частина (Frontend) реалізується як Single Page Application (SPA) на базі бібліотеки React.js, що забезпечує швидкий відгук інтерфейсу POS-терміналу. Серверна частина (Backend) на базі Node.js обробляє запити та взаємодіє з реляційною СУБД PostgreSQL.

У системі виділено наступні функціональні модулі:

1. Модуль POS (Point of Sale) — для швидкого оформлення замовлень офіціантами;
2. Складський модуль — для автоматизації списання продуктів за технологічними картками;
3. Аналітичний модуль — для візуалізації звітів про продажі у вигляді діаграм.

Порівняльна характеристика ролей у системі надана в таблиці №1 та на рисунку 2.

Таблиця №1

Порівняльна характеристика ролей у системі

Роль користувача	Опис функцій та рівень доступу в системі
Адміністратор	Повний доступ до системи: керування тарифними планами, реєстрація нових закладів (орендарів) та глобальна фінансова аналітика.
Менеджер	Керування конкретним закладом: редагування меню, проведення інвентаризації на складі, формування звітів за змінами та аналіз роботи персоналу.
Офіціант	Операційна діяльність: відкриття та закриття чеків, бронювання столиків, передача замовлень на кухню через інтерфейс POS-терміналу.

Вибір інструментів реалізації зумовлений необхідністю створення високопродуктивного та масштабованого SaaS-рішення. Для розробки клієнтської частини обрано бібліотеку React.js. Основною перевагою є використання механізму Virtual DOM, що мінімізує кількість операцій з реальним деревом сторінки та забезпечує миттєву реакцію інтерфейсу на дії касира чи офіціанта. Компонентний підхід дозволяє повторно використовувати елементи інтерфейсу, що прискорює розробку та полегшує підтримку коду.

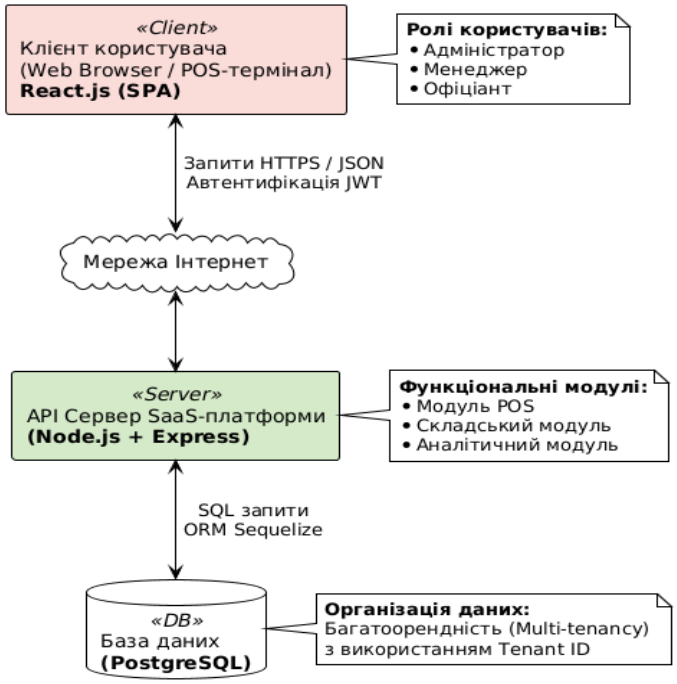

Рисунок 1 – Архітектура взаємодії компонентівSaaS-системи

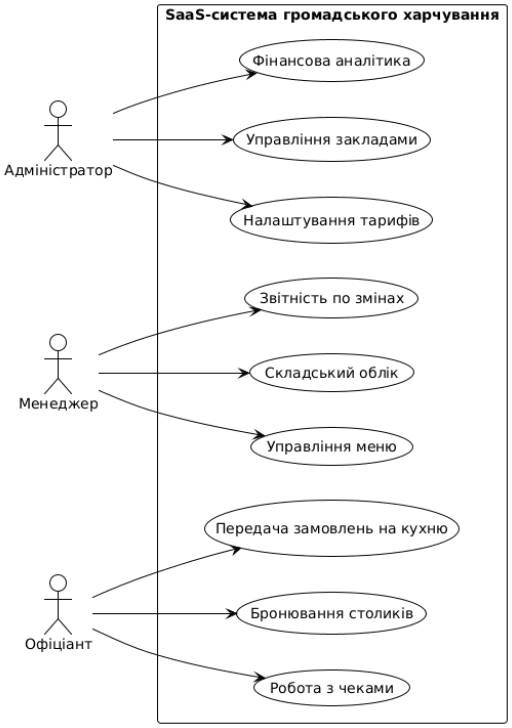

Рисунок 2 – Діаграма варіантів використання

На рівні серверної логіки використовується платформа Node.js із фреймворком Express. Завдяки подієво-орієнтованій неблокуючій моделі введення-виведення, сервер здатний обробляти велику кількість одночасних з'єднань, що важливо для систем реального часу. Для взаємодії з базою даних PostgreSQL обрано ORM Sequelize. Використання ORM дозволяє описувати схему даних у вигляді класів, забезпечує автоматичну генерацію міграцій та захищає систему від SQL-ін'єкцій завдяки вбудованим механізмам екранування запитів.

Безпека даних у SaaS-моделі є критичним аспектом. Для автентифікації та авторизації користувачів реалізовано механізм JSON Web Tokens (JWT). Це дозволяє реалізувати безсесійну (stateless) архітектуру: після входу користувач отримує зашифрований токен, який зберігається на стороні клієнта та передається у заголовках кожного запиту. Такий підхід спрощує масштабування сервера, оскільки йому не потрібно зберігати стан сесії в пам'яті.

Висновки

Запропонований підхід до автоматизації через SaaS-модель дозволяє закладам громадського харчування значно знизити операційні ризики та підвищити прозорість бізнесу. Проєктована система забезпечує гнучкість налаштувань під специфіку конкретного кафе чи ресторану та створює фундамент для подальшого впровадження модулів прогнозування попиту на основі зібраних статистичних даних.

Список використаних джерел

1. Соммервілл І. Інженерія програмного забезпечення. 10-те вид. 765 с.
2. Turner M., Budgen D. Turning software into a service. Computer. 2003. Vol. 36, No. 10. P. 38–44.
3. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/
4. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. O'Reilly Media, 2020. 374 p.
5. Buyya R., Vecchiola C., Selvi S. T. Mastering Cloud Computing: Foundations and Applications Programming. Morgan Kaufmann, 2013. 464 p.

УДК 004.4

МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ФОРМУВАННЯ ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ У ТУРИСТИЧНІЙ ПЛАТФОРМІ

Дудар А.О.[1)], Стасів І.С.[2)]
Західноукраїнський національний університет
[1)]магістрант; [2)]к.т.н., доцент

І. Постановка проблеми

Сучасні туристичні платформи працюють із великими обсягами даних про туристичні об’єкти, маршрути, категорії локацій, пошукові запити та поведінку користувачів. Однією з ключових задач таких систем є формування персоналізованих рекомендацій, які відповідають актуальним інтересам користувача та дозволяють спростити процес вибору туристичних об’єктів.

У багатьох туристичних сервісах рекомендації формуються переважно на основі популярності локацій, рейтингу або статичних параметрів. Такий підхід не завжди враховує зміну інтересів користувача під час взаємодії із системою. Наприклад, користувач може спочатку переглядати історичні пам’ятки, а згодом шукати природні локації або маршрути для активного відпочинку. У такому випадку рекомендаційна система повинна оперативно реагувати на зміну поведінки та оновлювати список рекомендованих об’єктів.Для розв’язання цієї задачі доцільно використовувати подієво-орієнтований підхід, за якого кожна дія користувача розглядається як окрема подія.

До таких подій можуть належати перегляд туристичної локації, виконання пошукового запиту, відкриття рекомендації, додавання об’єкта до обраного або створення маршруту. Аналіз цих подій дозволяє сформувати профіль інтересів користувача та використовувати його для подальшого ранжування туристичних об’єктів.

ІІ. Мета роботи

Метою роботи є розробка математичного забезпечення для формування персоналізованих туристичних рекомендацій на основі аналізу подій користувача. Запропонований підхід повинен враховувати типи взаємодій користувача із системою, їх вагову значущість, відповідність туристичних локацій категоріям інтересів та часову актуальність подій.

ІІІ. Формування персоналізованих рекомендацій у туристичній платформі

У межах запропонованого підходу поведінка користувача подається як сукупність подій, що виникають під час роботи з туристичною платформою. Кожна подія містить інформацію про користувача, тип дії, час її виникнення та додаткові параметри. На основі таких даних система визначає, які категорії туристичних об’єктів є найбільш релевантними для конкретного користувача.

Важливою особливістю моделі є використання вагових коефіцієнтів для різних типів подій. Перегляд локації має менший вплив на профіль користувача, ніж додавання об’єкта до обраного або створення маршруту, оскільки останні дії свідчать про вищий рівень зацікавленості. Наприклад, перегляду локації може відповідати вага 1, відкриттю рекомендації - 2, пошуковому запиту - 3, додаванню до обраного - 4, а створенню маршруту - 5. Таке ранжування дозволяє врахувати різний рівень важливості користувацьких дій.

У процесі формування рекомендацій важливо враховувати не лише сам факт виконання дії, а й контекст, у якому вона була здійснена. Наприклад, перегляд туристичної локації може мати різне значення залежно від того, чи користувач відкрив її випадково, чи знайшов через пошуковий запит за конкретною категорією. Тому події взаємодії доцільно пов’язувати з категоріями туристичних об’єктів, такими як історичний туризм, природні локації, гастрономічний туризм, активний відпочинок або культурні пам’ятки. Це дозволяє поступово формувати профіль інтересів користувача та використовувати його для персоналізованого ранжування локацій.

Перевагою такого підходу є прозорість процесу формування рекомендацій. На відміну від складних моделей машинного навчання, запропонована модель дозволяє явно визначити, які саме події вплинули на підвищення релевантності певної категорії або туристичного об’єкта. Це спрощує налаштування системи, дає змогу змінювати вагові коефіцієнти відповідно до особливостей предметної області та забезпечує кращу керованість recommendation-сервісу.

Рівень інтересу користувача до певної категорії туристичних об’єктів визначається на основі сукупності подій та їх ваг. Для цього використовується формула:

$$W_{c(u)} = \sum_{i=1}^{n} w(type_i) \cdot m(e_i, c) \quad (1)$$

де $W_{c(u)}$ – рівень інтересу користувача u до категорії $c, w(type_i)$ – ваговий коефіцієнт типу події, $m(e_i, c)$ – функція належності події до певної категорії, n – кількість подій користувача.

Після формування профілю інтересів виконується оцінювання релевантності туристичних локацій. Кожна локація порівнюється з категоріями інтересів користувача, після чого система визначає її підсумкову оцінку. Чим більшою є відповідність між характеристиками локації та профілем користувача, тим вище така локація розміщується у списку рекомендацій. Наприклад, якщо користувач часто переглядає музеї, історичні пам'ятки та додає такі об'єкти до обраного, то система підвищує вагу категорії історичного туризму та рекомендує подібні локації.

Для підвищення актуальності рекомендацій у моделі враховується зміна інтересів користувача з часом. Старі події не повинні постійно мати однаковий вплив на профіль користувача, оскільки його потреби можуть змінюватися залежно від мети подорожі, сезону або поточного контексту пошуку.

Для цього використовується механізм часової деградації ваги події:

$$W(t) = W_o \cdot e^{-\lambda \Delta t} \quad (2)$$

де $W(t)$ – актуальна вага інтересу, W_o – початкова вага події, λ – коефіцієнт затухання, Δt – час, що минув від моменту виникнення події.

Застосування часової деградації дозволяє зменшити вплив застарілих дій і надати більшу вагу новим подіям. Завдяки цьому рекомендаційна система краще адаптується до поточних інтересів користувача. Загальний процес формування рекомендацій передбачає отримання подій користувача, визначення їх типів, оновлення вагових коефіцієнтів, застосування часової деградації та формування списку туристичних об'єктів із найвищим рівнем релевантності. Воно також є особливо важливим для туристичних платформ, оскільки інтереси користувача часто мають тимчасовий характер. Наприклад, під час планування короткої подорожі користувач може активно цікавитися певним містом або типом локацій, але після завершення поїздки ці інтереси втрачають актуальність.

Запропонована модель може бути використана у мікросервісній туристичній платформі з подієво-орієнтованою обробкою даних. У такій архітектурі події користувача можуть передаватися через брокер повідомлень Apache Kafka до recommendation-сервісу, який асинхронно оновлює профіль користувача та формує персоналізовані рекомендації. Це дозволяє зменшити навантаження на основні сервіси платформи та забезпечити оновлення рекомендацій у режимі, близькому до реального часу.

Висновок

У роботі наведено підхід до формування персоналізованих рекомендацій у туристичній платформі на основі аналізу подій користувача. Використання вагових коефіцієнтів дозволяє враховувати різну важливість дій користувача, а механізм часової деградації забезпечує актуальність рекомендацій.

Запропоноване математичне забезпечення може бути інтегроване у мікросервісну туристичну платформу з подієво-орієнтованою обробкою даних, що підвищує адаптивність системи та якість взаємодії з користувачем.

Список використаних джерел

1. Recommender Systems Handbook [Електронний ресурс].–Режим доступу:https://link.springer.com/book/10.1007/978-1-4899-7637-6
2. JRicci F., Rokach L., Shapira B. Recommender Systems: Techniques, Applications, and Challenges[Електронний ресурс].– https://link.springer.com/book/10.1007/978-1-0716-2197-4
3. Apache Kafka Documentation [Електронний ресурс].–https://kafka.apache.org/documentation/
4. Крепич С.Я., Співак І.Я. Якість програмного забезпечення та тестування: базовий курс. Тернопіль: ФОП Паляниця В.А, 2020. – 478с.
4. Kleppmann M. Designing Data-Intensive Applications. [Електронний ресурс]. – Режим доступу: https://dataintensive.net/

УДК 004.8

МЕТОД ТРИЕТАПНОГО КОНТРОЛЮ ДОСТОВІРНОСТІ ГЕНЕРАТИВНИХ ВІДПОВІДЕЙ У КОРПОРАТИВНИХ RAG-СИСТЕМАХ

Войтюк І.Ф.[1)], Мельничук Д.С.[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] магістрант

І. Постановка проблеми

Корпоративні системи пошуково-доповненої генерації (RAG) стали стандартним архітектурним патерном надання великим мовним моделям (LLM) доступу до конфіденційних знань організації. Попри широке промислове впровадження, фундаментальна проблема галюцинацій – впевненого формулювання неправдивих тверджень моделлю – залишається не повністю розв'язаною. У регульованих галузях (фінансові послуги, охорона здоров'я) ціна помилки є несумірно високою: неточна порада щодо клінічного протоколу може загрожувати життю пацієнта, а помилкова інтерпретація фінансової регуляції може спричинити багатомільйонні штрафи. Дослідження показують, що навіть провідні комерційні правничі AI-системи демонструють частку галюцинацій 17–33 % при відповідях на правничі запити, що додатково підтверджує необхідність розробки спеціалізованих методів контролю достовірності. Існуючі академічні підходи до контролю достовірності – RAGAS, FActScore, SelfCheckGPT – оцінюють відповідь у термінах її підтвердження пошуковим контекстом. Це необхідна, але недостатня умова: відповідь може бути підтверджена контекстом, але порушувати регуляторну процедуру. Жоден з названих методів не враховує наявність структурованих регуляторних обмежень, що їх містять дерева рішень. Це обумовлює актуальність розробки методу, який одночасно гарантує і підтвердження контекстом, і процедурну відповідність.

ІІ. Мета роботи

Метою роботи є підвищення достовірності генеративних відповідей корпоративних RAG-систем у регульованих галузях за рахунок розробки методу триетапного контролю достовірності з урахуванням детермінованих процедурних обмежень, формалізованих у вигляді дерев рішень.

ІІІ. Метод триетапного контролю достовірності

Запропонований метод ґрунтується на трьох послідовних етапах верифікації згенерованої відповіді. Нехай задано запит користувача q, корпоративну базу знань $D = \{d_1, ..., d_N\}$, множину дерев рішень $T = \{T_1, ..., T_M\}$, пошуковий контекст $c \subset D$, отриманий гібридним ретрівером, та генеративну відповідь a, сформовану LLM. Задача контролю достовірності полягає у формуванні підсумкової оцінки $F(a,q,c,T) \in [0,1]$.

На етапі 1 виконується атомарна декомпозиція відповіді: $a \to S(a) = \{s_1, s_2, ..., s_n\}$, де кожне s_i – мінімальна змістовна одиниця, яка містить рівно один факт. Декомпозиція виконується окремою LLM-моделлю.

На етапі 2 обчислюється оцінка підтверджуваності пошуковим контекстом. Для кожного твердження s_i обчислюється індикатор $g(s_i, c) = 1$, якщо твердження підтверджується контекстом, та 0 в іншому разі. Перевірка виконується окремою моделлю арбітра. Загальна оцінка підтверджуваності визначається як частка підтверджених тверджень за формулою (1):

$$F_{grouding}(a,c) = (1/n) \cdot \sum_{i=1}^{n} g(s_i, c) \quad (1)$$

На етапі 3, що є ключовою новизною методу, виконується зіставлення відповіді з активною гілкою дерева рішень. Нехай v^* – вузол найближчого дерева рішень, до якого маршрутизатор зіставив запит q. Якщо max $sim(q,v) \geq \tau_{dt}$, то обчислюється оцінка процедурної відповідності за формулою (2):

$$F_{procedural}(a,T,q) = (1/n) \cdot \sum_{i=1}^{n} p(s_i, branch(v^*)) \quad (2)$$

де $branch(v^*)$ – активна гілка дерева, що включає вузол v^* та послідовність батьківських вузлів і обов'язкових інструкцій; $p(s_i, branch(v^*)) \in [0,1]$ – індикатор того, що твердження s_i не суперечить активній гілці. Перевірка виконується моделлю арбітраз промптом, що містить опис активної гілки. Якщо запит не потрапив на жодну гілку, $F_{procedural} = 1$ за визначенням.

Підсумкова оцінка достовірності визначається як зважене лінійне поєднання за формулою (3):

$$F(a,q,c,T) = \alpha \cdot F_{grouding}(a,c) + \beta \cdot F_{procedural}(a,T,q) \quad (3)$$

де α + β = 1. Ваги α та β залежать від домену: для медичних і фінансових застосувань β = 0,5 (рівна вага підтверджуваності та процедурної відповідності), для юридичних – β = 0,3, оскільки прецедентне право допускає інтерпретацію. Бінарне рішення про прийняття відповіді приймається за формулою:

$$accept(a) = [F(a,q,c,T) \geq \theta] \quad (4)$$

де θ = 0,9 – поріг прийняття, встановлений емпірично за результатами апробації.

IV. Експериментальна апробація

Для оцінювання ефективності запропонованого методу проведено експеримент A/B-порівняння з базовою RAG-системою. Тестовий набір склав 600 запитів (по 200 на кожен з трьох доменів: фінансовий комплаєнс, клінічні протоколи, юридичні консультації). Базова RAG-система – стандартна архітектура з гібридним пошуком та генерацією GPT-4o, без модулів триетапного контролю; запропонована система – повна архітектура зі всіма модулями. Узагальнені результати наведено в таблиці 1.

Таблиця 1

Результати експериментальної апробації методу

Метрика	Базова RAG	Фінанси	Медицина	Юриспруд.
Recall@5	0,89	0,93	0,91	0,90
Faithfulness (середнє)	0,76	0,95	0,94	0,92
Частка галюцинацій	12,4 %	2,8 %	3,1 %	3,7 %
Процедурні порушення	18,7 %	1,4 %	1,1 %	3,9 %
Затримка p95, с	1,8	3,2	3,4	3,5

Частка галюцинацій у запропонованій системі зменшилася з 12,4 % до 2,8–3,7 % залежно від домену (зменшення у 3,3–4,4 рази); частка процедурних порушень – з 18,7 % до 1,1–3,9 %. Затримка повного циклу зросла з 1,8 с до 3,2–3,5 с через додаткові виклики моделіарбітра, але залишається в межах цільових 4 с. Для оцінки статистичної значущості застосовано тест МакНемара: на всіх трьох доменах різниця між базовою та запропонованою системами статистично значуща при $p < 0,001$.

Експертна розмітка тестового набору виконувалася двома незалежними експертами-фахівцями відповідної галузі для кожного домену. Узгодженість розмітки за коефіцієнтом Cohen's κ становить 0,79–0,85, що відповідає високому рівню та валідує отримані оцінки. Для фінансового домену використано публічний корпус FINRA Notices та внутрішні SOP анонімізованої банківської установи; для медичного – публічні клінічні гайдлайни ACC/AHA та NCCN; для юридичного – публічну базу прецедентного права. Гірший результат на юридичному домені (3,7% галюцинацій, 3,9% процедурних порушень) пояснюється природою прецедентного права: юридичні висновки часто допускають інтерпретацію та аналогії з прецедентами, тому для цього домену емпірично встановлено меншу вагу процедурної відповідності (β = 0,3).

Висновок

У роботі запропоновано метод триетапного контролю достовірності генеративних відповідей у корпоративних RAG-системах, що поєднує атомарну декомпозицію відповіді, оцінку підтвердження пошуковим контекстом окремою моделлю арбітра та зіставлення з активною гілкою дерева рішень. На відміну від існуючих методів (RAGAS, FActScore), які перевіряють лише підтвердження контекстом, запропонований метод одночасно гарантує процедурну відповідність регуляторно-визначеному порядку дій. Експериментальна апробація на трьох доменах підтвердила зменшення частки галюцинацій у 3,3–4,4 рази та частки процедурних порушень у 4,8–17 разів при збереженні прийнятної затримки відповіді. Подальші дослідження спрямовані на автоматичну настройку вагових коефіцієнтів α та β на основі специфіки організації.

Список використаних джерел

1. Gao Y., Xiong Y., Gao X. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997. 2023.
2. Huang L., Yu W., Ma W. et al. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions // ACM Transactions on Information Systems. 2025. Vol. 43, No. 2. Article 42.

УДК 004.738.5

АРХІТЕКТУРА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ E-COMMERCE З JWT-АВТЕНТИФІКАЦІЄЮ ТА РОЛЬОВОЮ МОДЕЛЛЮ ДОСТУПУ

Рудий Р.С.[1)], Папа О.А.[2)]
Західноукраїнський національний університет
[1)] студент; [2)] д.філ., доцент

I. Постановка проблеми

Сфера електронної комерції в умовах цифрової трансформації бізнесу залишається одним із найдинамічніших сегментів IT-індустрії, причому переважна частина дій користувача – пошук, оформлення та оплата замовлень – здійснюється з мобільних пристроїв. Швидкий запуск магазину сьогодні забезпечують зрілі SaaS-платформи (Shopify, BigCommerce) та CMS-розширення (WooCommerce, OpenCart), однак вони обмежують власника тарифним планом, фіксованою бізнес-логікою і платформо-залежним рівнем безпеки. Власна модульна клієнт-серверна реалізація мобільного застосунку для онлайн-магазину усуває ці обмеження, але одночасно ставить чотири практичні задачі: захист REST-маршрутів від несанкціонованих запитів, чітке розмежування ролей USER та ADMIN, цілісність реляційних даних товарів і замовлень, а також готовність до горизонтального масштабування серверної частини без переписування бізнес-логіки. Актуальність дослідження зумовлена потребою у компактній, але повноцінній архітектурі MVP-рівня, яка охоплює увесь життєвий цикл замовлення – від реєстрації клієнта до оновлення статусу доставки – і одночасно зберігає простір для подальшої міграції на мікросервісну топологію.

II. Мета роботи

Метою роботи є розробка модульної клієнт-серверної архітектури мобільного застосунку для онлайн-магазину «Awesome Store» (умовна назва) з реалізацією JWT-автентифікації, рольової моделі доступу USER/ADMIN, типізованого шару доступу до даних на базі Prisma ORM та структури, придатної до горизонтального масштабування у межах модульного моноліту.

III. Архітектура та модель доступу

Систему спроєктовано за класичною трирівневою моделлю: рівень представлення – кросплатформений мобільний клієнт на React Native + TypeScript для iOS та Android – формує HTTP-запити та керує локальним станом кошика; рівень прикладної логіки на NestJS обробляє REST-запити, валідує DTO-схеми, виконує бізнес-правила і забезпечує JWT-захист маршрутів через guard-механізм; рівень даних на PostgreSQL через Prisma ORM зберігає реляційну структуру користувачів, товарів, замовлень і платежів. Серверну частину побудовано як модульний моноліт із п'яти логічно ізольованих модулів: AuthModule (видача та перевірка JWT, хешування паролів), UsersModule (керування профілями та ролями), ProductsModule (CRUD з каталогом), OrdersModule (формування й обробка замовлень) та PaymentsModule, який на поточному етапі веде журнал платіжних операцій, тоді як підключення до реальних платіжних шлюзів (LiqPay, Stripe) винесено у подальші дослідження. Архітектуру показано на рисунку 1, а порівняння з типовими готовими рішеннями за п'ятьма ключовими критеріями – у таблиці 1.

Таблиця 1

Порівняння варіантів реалізації e-commerce-застосунку

Критерій	SaaS (Shopify)	WooCommerce	Awesome Store (власна)
Контроль архітектури	обмежений	частковий	повний
Адаптація бізнес-логіки	обмежена	висока	повна
Інтеграція з мобільним застосунком	через зовнішній API	через зовнішній API	нативна на REST API
Безпека	платформо-залежна	залежить від плагінів	контрольована розробником
Місячна вартість, USD	від 39 + комісії	5–50 (хостинг)	5–20 (VPS)

Для кількісного оцінювання затримки відповіді API на стороні сервера сумарний час обробки запиту подано формулою (1):

$$T\_resp = T\_jwt + T\_val + T\_loqic + T\_orm + T\_ser \quad (1)$$

де T_jwt – час верифікації JWT-токена і перевірки ролі; T_val – час валідації DTO; T_loqic – час виконання бізнес-логіки модуля; T_orm – час Prisma-запиту до PostgreSQL; T_ser – час серіалізації відповіді у JSON. Цільовою межею для одиничного запиту без черги визначено $T\_resp \leq 250$ мс; для пікового режиму (100 паралельних запитів на одному вузлі) – $T\_resp \leq 350$ мс на 95-му перцентилі.

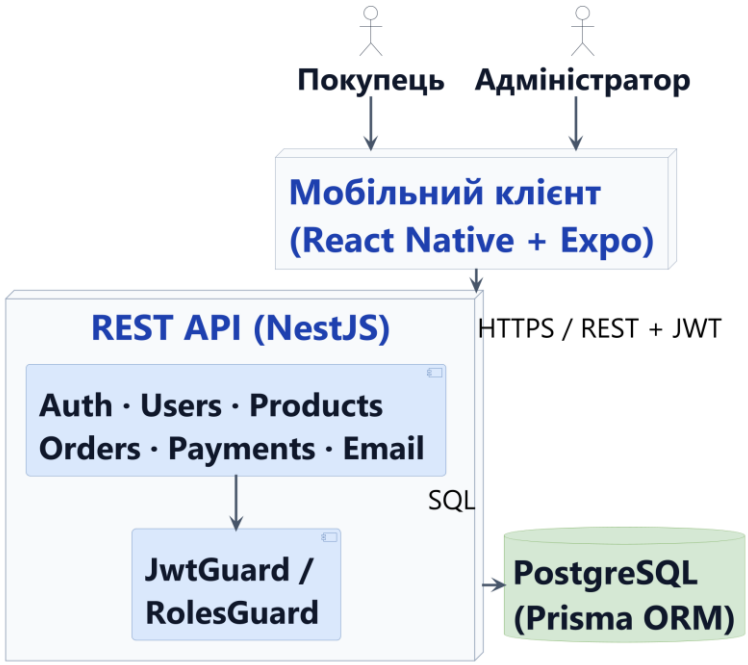

Рисунок 1 – Трирівнева архітектура системи “Awesome Store”

Здатність архітектури до горизонтального масштабування серверного шару оцінюється за законом Амдала, поданим у формулі (2):

$$S(N) = 1/((1-P) + P/N) \tag{2}$$

де $S(N)$ – пришвидшення; P – частка обробки, яка масштабується горизонтально; N – кількість серверних вузлів за балансувальником. Stateless-дизайн NestJS-частини, ізоляція стану кошика на клієнті та збереження сесійних даних у JWT-токенах підвищують P до 0,9–0,95. Перед N серверними вузлами розгортається nginx як TLS-термінатор і round-robin балансувальник — без нього лінійне нарощування пропускної здатності залишається теоретичним. Наведена оцінка стосується лише серверного шару; масштабування PostgreSQL потребує окремих рішень (read-репліки, шардування, connection pooler), без яких саме база даних стане вузьким місцем.

IV. Безпека та автентифікація

Захист застосунку реалізовано на чотирьох рівнях. По-перше, паролі користувачів зберігаються лише у вигляді bcrypt-хешів з cost factor 12 (≈250 мс на хешування одного пароля на типовому VPS), що робить економічно недоцільним брутфорс згідно з рекомендаціями OWASP ASVS v4.0. По-друге, JWT-токен поділено на access-токен з часом життя 15 хвилин і refresh-токен з часом життя 7 днів; обидва підписуються алгоритмом HS256 із серверним секретним ключем, refresh-токен на стороні клієнта зберігається в Expo SecureStore (Keychain на iOS, EncryptedSharedPreferences на Android), а не у звичайному AsyncStorage. По-третє, на маршрути /auth/login та /auth/refresh застосовано rate limiting (10 спроб з однієї IP-адреси за хвилину) через @nestjs/throttler, що додатково ускладнює перебір паролів. По-четверте, доступ до адміністративних маршрутів реалізовано через композитний RolesGuard, що після перевірки підпису токена JwtGuard зчитує атрибут role і порівнює його з декларативними декораторами @Roles на контролерах, повертаючи 403 Forbidden у разі невідповідності. Транспортний рівень захищено обов'язковим TLS 1.2+ із заборонами слабких шифросюїтів через nginx-конфігурацію; усі невдалі спроби автентифікації логуються разом з IP-адресою та позначкою часу для подальшого аналізу адміністратором безпеки. Послідовність перевірки під час входу в систему та подальшого захищеного запиту наведено на рисунку 2.

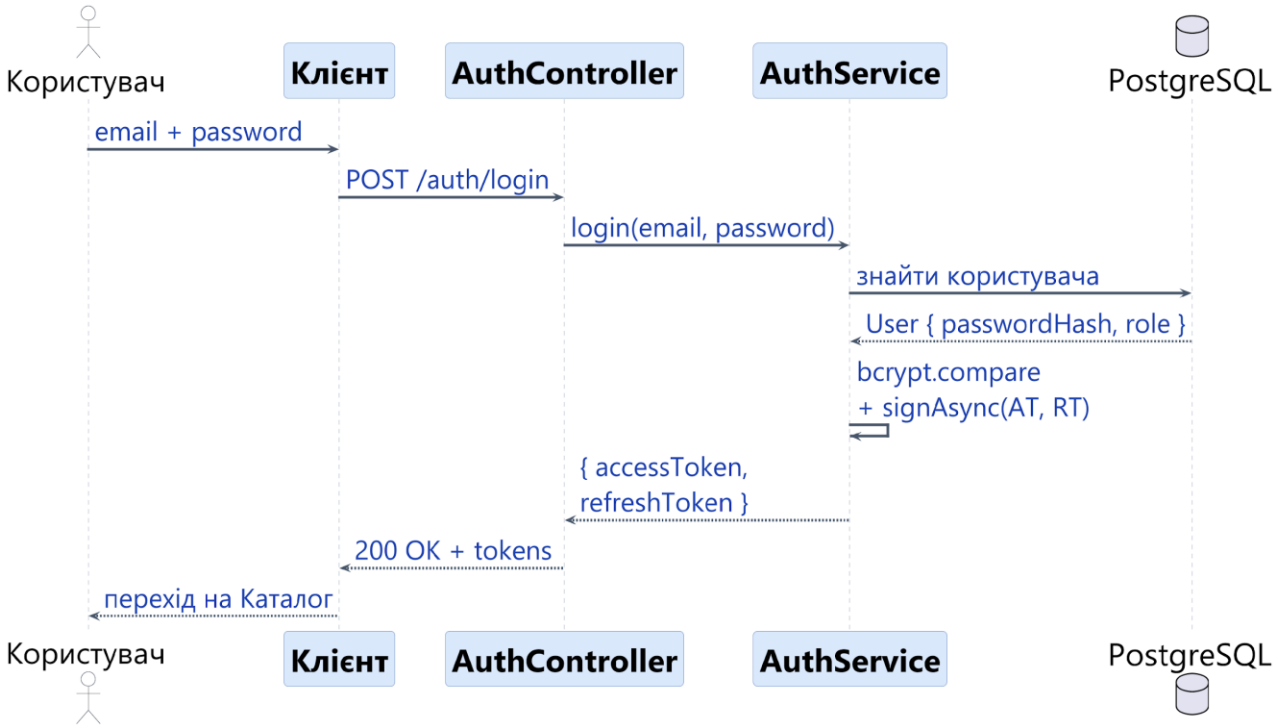

Рисунок 2 – Sequence-діаграма автентифікації за JWT і перевірки ролі через RolesGuard

V. Програмна реалізація та результати

Серверну частину реалізовано на NestJS з вбудованим dependency injection, валідацією через class-validator pipes та централізованим обробником винятків. Реляційна схема містить п'ять таблиць – Users, Products, Orders, OrderDetails, Payments – з UUID-первинними ключами та enum-типами PaymentStatus (PENDING / PAID / FAILED) і DeliveryStatus (NEW / SHIPPED / DELIVERED / CANCELED). Для замовлень і платежів застосовано стратегію soft delete (поле deletedAt) – фінансово значущі записи фізично не видаляються навіть у разі видалення облікового запису покупця, що відповідає вимогам бухгалтерського аудиту; ON DELETE CASCADE використовується тільки для службових даних. Типізований клієнт Prisma, що генерується з schema.prisma, усуває потребу в рядковому формуванні SQL для типових CRUD-операцій.

Мобільний клієнт реалізовано на React Native з єдиною кодовою базою для iOS та Android; стан кошика підтримується на клієнті через Zustand із persistence-шаром у MMKV, що дозволяє продовжувати наповнення без активного з'єднання та надсилати замовлення одним атомарним викликом /orders. Експериментальна функціональна перевірка на тестовому каталозі з 200 товарів і 50 синтетичних замовлень підтвердила коректність end-to-end сценаріїв реєстрації, автентифікації, оформлення замовлення і зміни його статусу адміністратором; повноцінне навантажувальне тестування на реальних обсягах винесено у подальші дослідження. В орієнтовному запуску через k6 при 100 паралельних запитах 95-й перцентиль затримки утримався в межах 320 мс – у визначеному порозі пікового режиму (≤ 350 мс).

Висновок

У роботі запропоновано модульну клієнт-серверну архітектуру мобільного застосунку для електронної комерції, що поєднує stateless-серверну частину на NestJS, типізований доступ до PostgreSQL через Prisma ORM, кросплатформений мобільний клієнт на React Native і цілісну рольову модель доступу на основі JWT з чотирирівневим контуром безпеки (bcrypt, refresh-токени у SecureStore, rate limiting, композитний RolesGuard). Декомпозиція на п'ять логічно ізольованих модулів у поєднанні зі stateless-обробкою дає коефіцієнт P=0,9–0,95 за законом Амдала, що відкриває шлях до горизонтального масштабування серверного шару за умови винесення PostgreSQL у read-репліки. Подальші дослідження — інтеграція платіжних шлюзів (LiqPay, Stripe), MFA, виокремлення PaymentsModule та OrdersModule в мікросервіси, аналітичний контур через CQRS-проєкції.

Список використаних джерел

1. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2003. 533 p.
2. NestJS Documentation. URL: https://docs.nestjs.com
3. Prisma ORM Documentation. URL: https://www.prisma.io/docs
4. Amdahl G. Validity of the Single-Processor Approach to Achieving Large Scale Computing Capabilities. AFIPS Conference Proceedings. 1967. Vol. 30. P. 483–485.

УДК 004.891

ІНТЕЛЕКТУАЛЬНА ВЕБ-СИСТЕМА ПЕРСОНАЛІЗОВАНИХ РЕКОМЕНДАЦІЙ ДЛЯ КІНОКОМПЛЕКСУ

Порплиця Н.П.[1)], **Корзілов К.Є.**[2)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)]бакалавр

І. Постановка проблеми

Зараз цифровізація проникає практично у всі сфери життя, не винятком є і сфера розваг, наприклад, відбувається активне впровадження інформаційних технологій у діяльність кінотеатрів. Зростання кількості користувачів онлайн-сервісів та їхніх вимог до швидкості, зручності й персоналізації обслуговування зумовлює необхідність переходу від традиційних методів роботи до впровадження та активного використання сучасних спеціалізованих веб-орієнтованих систем.

Традиційні підходи до організації роботи кінотеатрів, наприклад, продаж квитків через касиофлайн та статичне планування сеансів, не забезпечують достатнього рівня ефективності та гнучкості, а також вимагають значних затрат ресурсів. Через це глядачі мають обмежений доступ до послуг, очікують у чергах, складно оперативно отримати інформацію про зміни в графіку сеансів, а також відсутній індивідуальний підхід до кожного клієнта.Сучасні системи онлайн-бронювання частково вирішують ці проблеми, надаючи можливість перегляду афіші, вибору сеансів і придбання квитків онлайн. Проте, як правило, вони орієнтовані виключно на виконання базових функцій і не враховують індивідуальні вподобання користувачів. Це призводить до зниження можливого рівня залученості глядачів, оскільки користувач змушений самостійно аналізувати великий обсяг інформації та обирати фільми без засобів підтримки.

Особливо актуальною є проблема ефективного підбору стрічок в умовах постійного оновлення кіноафіші та великої кількості фільмів різних жанрів. Відсутність механізмів персоналізації ускладнює процес прийняття рішення для глядача та провокує можливе зниження ймовірності повторного використання сервісу. Одним із можливих напрямів вирішення цієї проблеми є впровадження інтелектуальних рекомендаційних систем, які здатні аналізувати поведінку користувачів, їх історію взаємодії з системою, обрані жанри та оцінки, і на основі цього формувати персоналізовані рекомендації контенту для перегляду. Використання таких підходів дозволяє значно підвищити релевантність контенту, покращити користувацький досвід та збільшити комерційну ефективність роботи кінотеатру.

Отже, актуальним завданням є розробка інтелектуального веб-застосунку для кінокомплексу, який поєднує функціонал онлайн-бронювання квитків із системою персоналізованих рекомендацій. Такий підхід дозволить автоматизувати основні бізнес-процеси, підвищити зручність користування сервісом та дозволить підвищити рівень залученості глядачів.

ІІ. Мета роботи

Метою роботи є розробка інтелектуального веб-застосунку для кінокомплексу з використанням сучасних веб-технологій та впровадженням системи персоналізованих рекомендацій для підвищення якості обслуговування користувачів і ефективності роботи кінотеатру.

ІІІ. Математичне забезпечення рекомендаційної системи

Як було зазначено раніше, одним із ключових компонентів розроблюваного веб-застосунку є система персоналізованих рекомендацій, яка забезпечує підбір релевантного контенту для конкретного користувача на основі історії його взаємодії із сервісом. Основним завданням такої підсистеми є визначення ступеня відповідності фільмів індивідуальним вподобанням користувача.

У цій праці запропоновано використати підхід колаборативної фільтрації (CollaborativeFiltering, CF), який базується на аналізі історії взаємодії користувачів із системою. Метод ґрунтується на припущенні, що користувачі зі схожими вподобаннями в минулому матимуть подібні інтереси й у майбутньому.

Для реалізації алгоритму формується матриця взаємодій «користувач–фільм», де кожен елемент $r_{u,i}$ відображає оцінку користувача u для фільму i. На основі цієї матриці кожного користувача можна представити у вигляді вектора вподобань.

Подібність між користувачами u та v визначається за допомогою міри подібності, яку обчислюємо за такою формулою:

$$sim(u,v) = \frac{\sum_{i \in I} r_{u,i} \cdot r_{r,i}}{\sqrt{\sum_{i \in I} r_{u,i}^2} \cdot \sqrt{\sum_{i \in I} r_{v,i}^2}} \quad (1)$$

де I – множина фільмів, оцінених обома користувачами.

Отримані значення подібності дозволяють сформувати множину найближчих користувачів, які мають схожі інтереси. На основі цієї множини можемо здійснити прогнозування оцінки для користувача u щодо фільму i, який він ще не переглядав:

$$r_{u,i} = \frac{\sum_{v \in U} sim(u,v) \cdot r_{v,i}}{\sum_{v \in} |sim(u,v)|} \quad (2)$$

де U – множина користувачів, подібних до користувача u.

Отримані значення використовуються для подальшого ранжування фільмів, а також формування персоналізованих рекомендацій для конкретного глядача. Слід зазначити, що колаборативна фільтрація має певні обмеження, зокрема проблему «холодного старту», яка виникає у випадку відсутності даних про нового користувача або новий фільм. Для часткового вирішення цієї проблеми може бути використаний контент-орієнтований підхід, що аналізує характеристики фільмів, такі як жанр, опис, ключові слова, акторський склад та рейтинг.

У загальному випадку рекомендаційна система може поєднувати декілька підходів одночасно, формуючи гібридну модель рекомендацій. Такий підхід дозволяє підвищити точність підбору контенту та забезпечити більш якісну адаптацію системи до вподобань користувачів.

IV. Архітектура та програмна реалізація

Програмна реалізація веб-застосунку виконана на основі клієнт-серверної архітектури з використанням сучасного стеку технологій. Клієнтська частина реалізована з використанням бібліотеки React та CSS-фреймворку Tailwind, це забезпечує створення адаптивного та зручного інтерфейсу користувача.

Серверна частина реалізована на платформі Node.js із використанням фреймворку Express, що дозволяє ефективно обробляти запити користувачів та реалізовувати REST API. Для зберігання даних використовується реляційна база даних MySQL, яка забезпечує цілісність та надійність інформації. Архітектура системи базується на патерні MVC (Model–View–Controller), що дозволяє розділити логіку застосунку, інтерфейс користувача та механізми обробки даних, забезпечуючи масштабованість і зручність супроводу. Для передачі даних між компонентами системи використовується формат JSON, який гарантує легкість обробки інформації на обох рівнях архітектури. Застосування сучасного інструментарію збірки проєкту забезпечує швидке розгортання та високу продуктивність за рахунок оптимізації клієнтських скриптів.

Висновок

У роботі обґрунтовано необхідність створення інтелектуального веб-застосунку для кінокомплексу та запропоновано підхід до реалізації системи рекомендацій на основі колаборативної фільтрації, та описано підхід до вирішення проблеми «холодного старту». Зокрема, запропоновано використання контент-орієнтованого підходу для аналізу характеристик фільмів та формування рекомендацій у випадках недостатньої кількості даних про користувачів або контент.У межах роботи розглянуто математичне забезпечення рекомендаційної системи, принципи визначення подібності користувачів та механізми прогнозування оцінок фільмів. Також описано архітектуру веб-застосунку та обґрунтовановибірсучаснихтехнологій для йогореалізації. Запропоноване рішення дозволяє автоматизувати процеси кінотеатру, підвищити якість обслуговування та забезпечити персоналізований підхід до глядачів.

Список використаних джерел

1. Flanagan D. JavaScript: TheDefinitiveGuide. 7th ed. O'ReillyMedia, 2020. 704 p.
2. ReactDocumentation. URL: https://react.dev/learn
3. Node.js Documentation. URL: https://nodejs.org/learn
4. MVC DesignPattern. GeeksforGeeks. URL: https://www.geeksforgeeks.org/system-design/mvc-design-pattern/
5. Express.js Documentation. URL: https://expressjs.com
6. MySQLDocumentation. URL: https://dev.mysql.com
7. Tailwind CSS Documentation. URL: https://tailwindcss.com/docs/installation/using-vite

УДК 004.8

ОЦІНЮВАННЯ КОГНІТИВНИХ СТАНІВ КОРИСТУВАЧІВ SAAS-СИСТЕМ НА ОСНОВІ ПРИХОВАНИХ МАРКОВСЬКИХ МОДЕЛЕЙ

Літинський О.Л.[1)], Папа О.А[2)]
Західноукраїнський національний університет
[1)]магістрант; [2)]д.філ., доцент

Постановка проблеми

Стрімка цифровізація бізнес-процесів, зокрема у сфері соціального підприємництва, призвела до ускладнення платформ операційного управління (B2B SaaS). Робота з високонавантаженими корпоративними інтерфейсами в умовах дефіциту часу зумовлює зростання когнітивного навантаження на користувача, що спричиняє швидку втому та підвищує ймовірність операційних помилок. Більшість існуючих систем є детермінованими та не здатні проактивно реагувати на зміну стану людини.

Мета роботи

Метою роботи є розробка процедур безперервного моніторингу та ідентифікації латентних психофізіологічних станів користувача на основі неявних патернів поведінки для своєчасної проактивної адаптації користувацького інтерфейсу.

Особливості реалізації дослідження

Для реалізації гібридної моделі оцінювання когнітивних станів користувачів SaaS-систем на основі прихованих марковських моделей пропонується кібернетична структура «Користувач – Інтерфейс – Адаптивний плагін» із зворотним зв'язком (див. рис. 1).

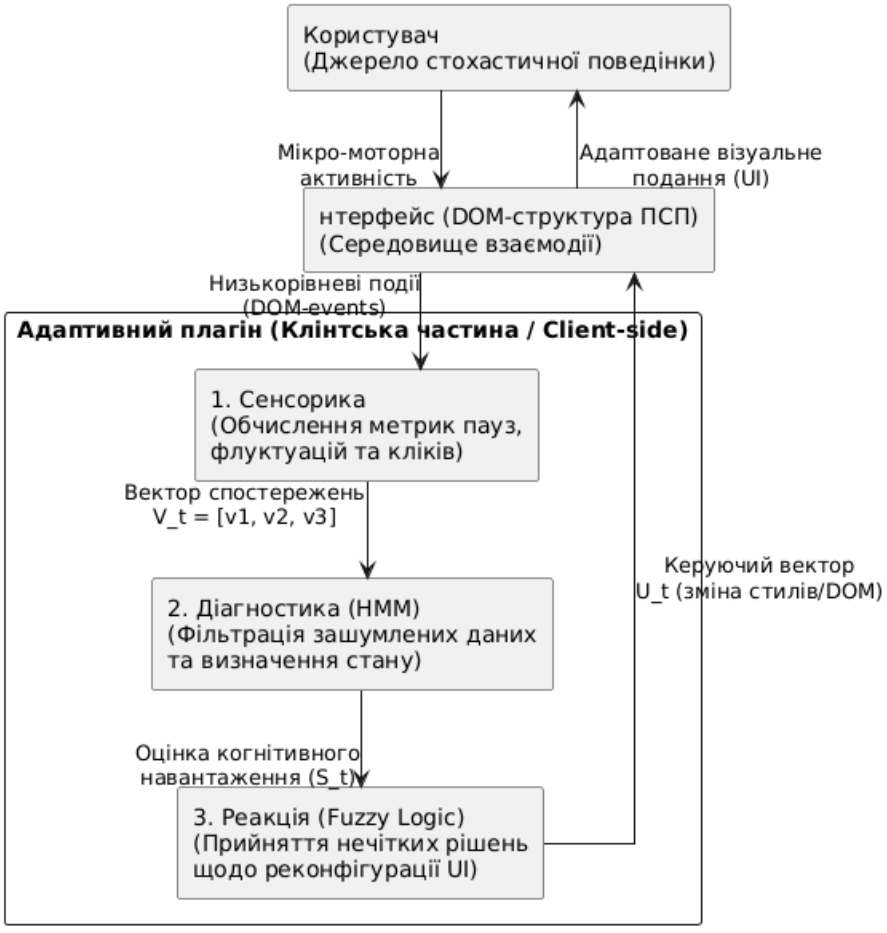

Рисунок 1 – Схема структури взаємодії «Користувач – Інтерфейс – Адаптивний плагін»

Ця структура базується на таких функціональних ролях:

– користувач (об'єкт генерації даних), який виступає як стохастична система. Сприймає візуальну інформацію та генерує неперервний потік мікро-моторних дій, що містять приховану інформацію про його когнітивний стан;

– інтерфейс (середовище взаємодії), який відіграє роль пасивного реципієнта. Це DocumentObjectModel (DOM), яка містить форми, кнопки, таблиці. Інтерфейс генерує низькорівневі події (mousemove, click, keydown);

– адаптивний плагін (інтелектуальний контролер), який працює як проміжне забезпечення між користувачем та інтерфейсом. Структурно плагін реалізує тритактний цикл:

-сенсорика – перехоплення подій інтерфейсу та обчислення вектора телеметрії V_t (розрахунок флуктуацій та пауз за наведеними вище формулами);

- діагностика – передача вектора V_t до математичного ядра (прихованої марковської моделі (HMM)), яка в умовах невизначеності обчислює ймовірність того, що користувач перебуває у стані когнітивного перевантаження;

- реакція – якщо ймовірність стресу перевищує заданий поріг, активується система нечіткого виведення, яка генерує плавні керуючі впливи на DOM-дерево інтерфейсу (збільшення шрифту, блокування зайвих меню, вивід підказки), тим самим змінюючи середовище взаємодії.

Формалізація простору прихованих станів та спостережень

Нехай когнітивна діяльність користувача під час роботи з інтерфейсом описується скінченною множиною прихованих станів:

$$S = \{S_1, S_2, ..., S_N\} \quad (1)$$

де N – кількість виділених базових станів.

Для задачі оптимізації інтерфейсів платформ соціального підприємництва обґрунтовано доцільність виділення трьох основних станів:

S_1 – стан нормальної працездатності (стабільний стан). Характеризується високим рівнем концентрації уваги, низьким часом прийняття рішень та впевненою навігацією. Користувач не відчуває труднощів із сприйняттям елементів UI;

S_2 – стан компенсаторної втоми (проміжний стан). Виникає внаслідок тривалої монотонної роботи або підвищеної щільності даних. Ефективність виконання завдань ще залишається на допустимому рівні, проте з'являються перші ознаки збільшення мікро-пауз та флуктуацій курсору;

S_3 – стан деструктивного стресу або розгубленості (критичний стан). Свідчить про гостре когнітивне перевантаження користувача, викликане складністю форми, незрозумілою структурою меню або дефіцитом часу. Супроводжується різким падінням швидкості обробки інформації та хаотичною моторикою.

Алгоритмічне забезпечення гібридної моделі взаємодії

Об'єднання ймовірнісної моделі розпізнавання станів (HMM) та системи нечіткого керування у єдиний гомеостатичний контур вимагає розробки цілісного алгоритмічного забезпечення. Таке забезпечення повинно забезпечувати безперервне виконання обчислювальних процесів без створення затримок у роботі інтерфейсу B2B SaaS-платформи. Оскільки розрахунки виконуються на стороні клієнта, логіка взаємодії компонентів має бути строго типізованою та детермінованою на рівні алгоритмічних кроків.

Функціонування модуля реалізується у вигляді циклічного процесу, що складається з двох основних фаз: фази асинхронного збору та фільтрації даних (HMM) та фази синхронного розрахунку та адаптації середовища.Діаграма послідовності ілюструє архітектурні межі та часовий розподіл навантаження між чотирма основними компонентами системи: веб-інтерфейсом (DOM), модулем сенсорики, математичним ядром HMM та контролером нечіткої логіки.

Описане алгоритмічне забезпечення працює за принципом тактового моніторингу. Період дискретизації (таймаут вікна збору даних) встановлюється на рівні $\Delta\tau = 1000\,мс$, що є оптимальним для накопичення первинних метрик мікро-моторики без надмірного навантаження на браузер.

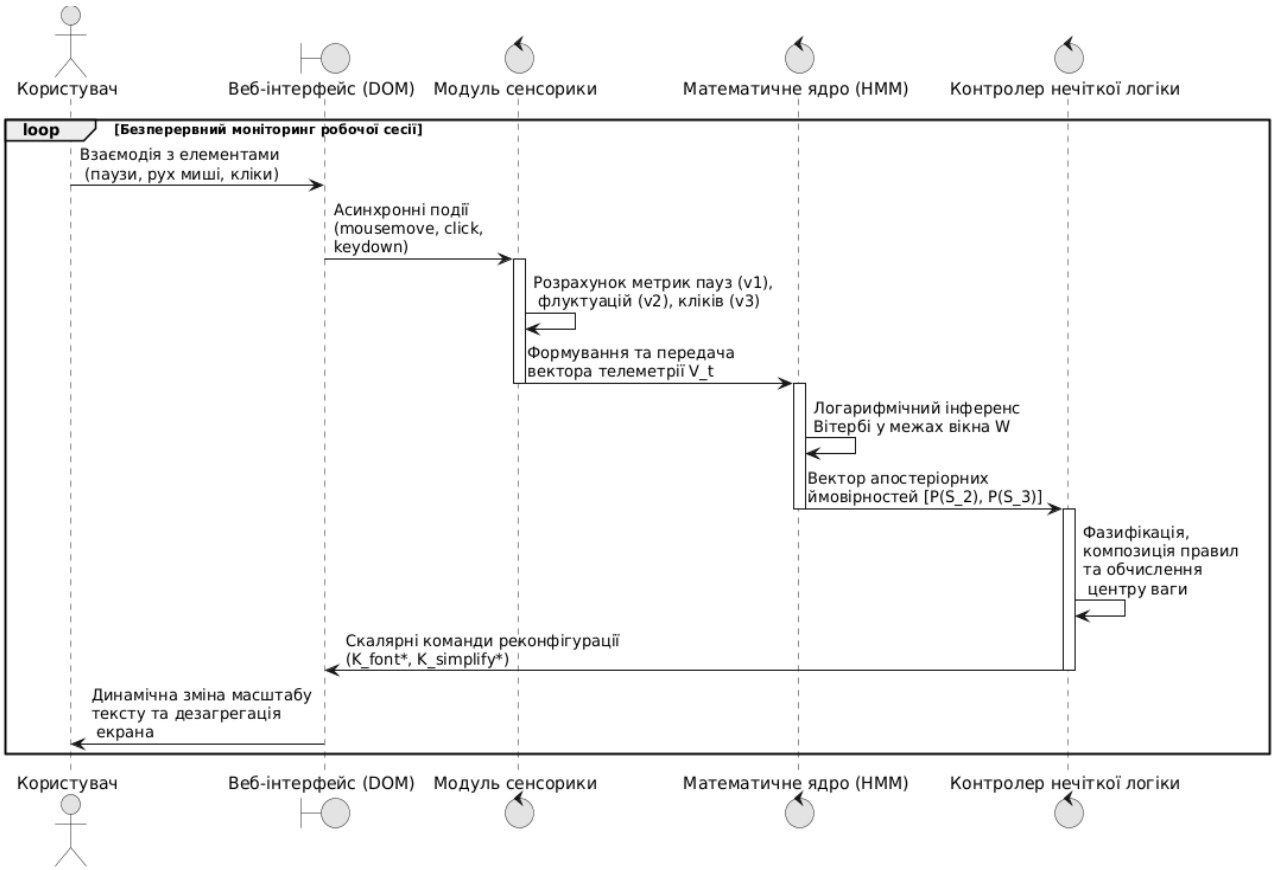

Рисунок 2 – Діаграма послідовності дій компонентів

Для забезпечення можливості розпізнавання станів у режимі реального часу безпосередньо у веб-браузері класичний алгоритм Вітербі було суттєво оптимізовано. З метою усунення проблеми арифметичного заниження, характерної для клієнтських JavaScript-функцій при множенні ймовірностей, ідентифікацію переведено у логарифмічний простір. Операція множення була замінена додаванням логарифмів вагових коефіцієнтів матриць переходів та емісій. Додатково впроваджено механізм межового ковзного вікна пам'яті (розмірністю 50 тактів), що дозволило стабілізувати обчислювальну складність алгоритму.

Висновок

Результати тестування розробленого математичного та алгоритмічного забезпечення підтвердили його високу ефективність: оптимізований алгоритм здатний обчислювати апостеріорні ймовірності станів $P(S_2)$ (втома) та $P(S_3)$ (стрес) за час менше 0.5 мс. Це дозволяє використовувати отримані оцінки як вхідні дані для систем нечіткого виведення з метою подальшої безперервної реконфігурації графічного інтерфейсу без створення навантаження на серверний бекендSaaS-платформи.

Список використаних джерел

1. Chen, S., &Epps, J. (2021). Using Mouse Dynamics to Assess Cognitive Load. Proceedings of the ACM on Human-Computer Interaction, 5, 1-22.
2. S.Krepych, I.Spivak, “Algorithm of automatic generation of hotel descriptions using templetes based on Markov chains”, International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). 2018. – pp.257-260
3. Pontones, C., Titzmann, A., Hübner, H., etal. (2023). ADABase: A Multimodal Dataset for Cognitive Load Estimation. Sensors, 23(1), 340.
4. V.Krutko, I.Spivak, S.Krepych, “An approach to assessing the reliability of software systems based on a graph model of method dependence”, CEUR Workshop Proceedings, 2024, 3662, pp.37-47
5. Loo, B.P.Y., Zhang, F., Hsiao, J.H. etal. Applying the Hidden Markov Model to Analyze Urban Mobility Patterns: An Interdisciplinary Approach. Chin. Geogr. Sci. 31, 1–13 (2021). https://doi.org/10.1007/s11769-021-1173-0
6. Brdiczka, O., Yew, J., & Churchill, E. F. (2022). Adaptive User Interfaces: Challenges and Opportunities in the Era of AI. IEEE Pervasive Computing, 21(3), 34-42.
7. Krepych S., Spivak I., Spivak S., “Methodology of formation of the individual study plan of the student based on the graph model of the dependence of discilplines”, CEUR Workshop Proceedings, 2023, 3426, pp.298-307

УДК 004.8

КАСКАДНИЙ ПІДХІД ДО МОДЕЛЮВАННЯ ВІДВІДУВАНОСТІ ПОДІЙ: АНАЛІЗ ЕФЕКТИВНОСТІ XGBOOST ТА RANDOM FOREST НА ГЕТЕРОГЕННИХ ВИБІРКАХ

Мельник А.М.[1)], Пукас Б.І.[2)]
Західноукраїнський національний університет
[1)]д.т.н., професор; [2)] магістрант

I. Постановка проблеми

Ефективне прогнозування масових заходів ускладнюється екстремальною неоднорідністю даних. Унітарні моделі, навчені на змішаних датасетах, демонструють високе статистичне зміщення (bias), фокусуючись на об'єктах із великою відвідуваністю та ігноруючи специфіку локальних подій. Проблема додатково ускладнюється наявністю бімодального розподілу – типу статистичного розподілу, що містить два виражені піки щільності даних. У межах досліджуваної вибірки це відповідає малим локальним заходам та масштабним стадіонним подіям. Така структура даних вимагає розробки архітектури, здатної адаптуватися до різних функціональних сегментів ринку.

II. Мета роботи

Підвищення точності інтелектуального аналізу відвідуваності масових заходів шляхом розробки та обґрунтування каскадної архітектури моделей машинного навчання, що дозволяє нівелювати негативний вплив екстремальної гетерогенності та асиметрії даних.

III. Основна частина

На основі статистичного аналізу сформованого масиву даних, що налічує 32 378 унікальних записів, було виявлено критичну асиметрію розподілу цільової змінної. Коефіцієнт асиметрії (Skewness) на рівні 2.07 вказує на суттєвий правосторонній перекіс, де переважна більшість подій зосереджена в зоні малих значень відвідуваності. Показник ексцесу (Kurtosis) на рівні 4.24 свідчить про високу концентрацію даних навколо середнього значення та наявність «важких хвостів», що відповідають рідкісним, але масштабним заходам, які суттєво відхиляються від нормального розподілу.

Математичним підтвердженням екстремальної неоднорідності вибірки є той факт, що медіанна відвідуваність (55 осіб) виявилася у 211 разів меншою за середню арифметичну величину (11 623 особи). Стандартне відхилення (σ = 25 722) вдвічі перевищує середнє значення, що свідчить про неможливість адекватного моделювання всього датасету за допомогою унітарних алгоритмів без попередньої сегментації.

Побудована гістограма розподілу в логарифмічній шкалі (див. рис.1) наочно підтвердила наявність бімодального характеру даних із двома вираженими піками щільності. Перший пік відповідає мікро-подіям (10–100 осіб), а другий – масовим заходам стадіонного формату (понад 60 000 осіб).

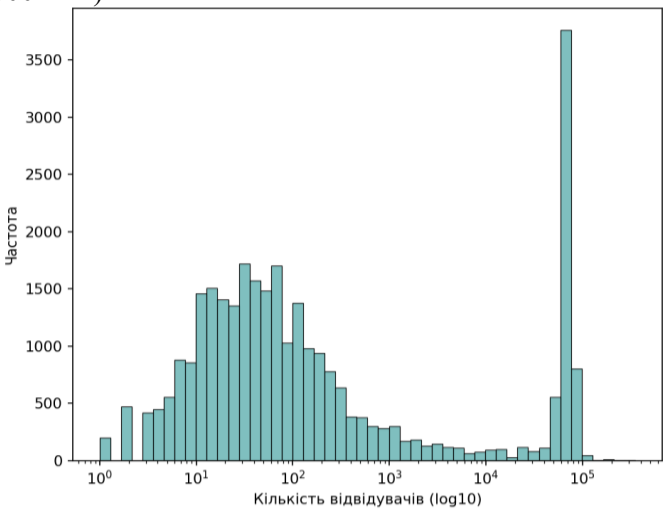

Рисунок 1 – Гістограма розподілу кількості відвідувачів у логарифмічній шкалі

Така візуалізація гетерогенності дозволила математично обґрунтувати перехід від єдиної моделі до каскадної архітектури, розділивши генеральну сукупність на чотири функціональні страти:

- Micro (0–100): 61% вибірки, характеризується високим рівнем семантичного шуму: CV=0.95 – коефіцієнт варіації, що визначає відносний рівень розсіювання даних.
- Small (100–1,000): 17.6% вибірки, де визначальними стають географічні ознаки.
- MidRange (1,000–60,000): сегмент із максимальною відносною дисперсією (CV=1.03).

- Giant (60,000+): 14% вибірки, найбільш стабільний сегмент (CV=0.16).

У ході дослідження було спроектовано та реалізовано каскадну архітектуру моделей машинного навчання, яка базується на інтелектуальній комбінації алгоритмів градієнтного бустингу (XGBoost) та випадкових лісів (RandomForest). Вибір такої гібридної структури обумовлений необхідністю врахування специфіки кожного статистичного сегмента даних.

XGBoost (Extreme Gradient Boosting) було обрано як базовий алгоритм для сегментів із високим рівнем нелінійності та великою кількістю розріджених ознак, зокрема семантичних векторів назв подій, отриманих після NLP-обробки. Математична перевага XGBoost полягає у використанні функцій регуляризації (L1 та L2), які інтегровані безпосередньо в цільову функцію моделі для запобігання перенавчанню (overfitting), що є критично важливим при роботі з малими вибірками в сегментах "Micro" та "Small".

Для сегмента MidRange пріоритет було надано алгоритму RandomForest. Як ансамблевий метод, що базується на принципі беггінгу (bootstrapaggregating), він передбачає паралельне навчання множини незалежних дерев рішень на випадкових підвибірках із наступним усередненням результатів. Такий підхід демонструє суттєво вищу стійкість до статистичних викидів та аномалій у зонах із екстремально високою дисперсією параметрів організаторів та локацій, де градієнтний бустинг часто схильний до надмірної підгонки під шум.

Порівняльний аналіз унітарного підходу (навчання єдиної моделі на всьому масиві даних) та запропонованого каскаду повністю підтвердив наукову гіпотезу дослідження. Унітарні моделі продемонстрували незадовільну прогнозну здатність: показник MAE (Mean Absolute Error) для унітарного RandomForest склав 636.92 , а для XGBoost середня абсолютна відносна помилка (MAPE) перевищила критичну позначку у 1600%. Такий результат пояснюється математичним домінуванням великих значень відвідуваності в загальній вибірці, через що модель фактично ігнорує варіативність малочисельних подій.Результати каскадної реалізації:

Для сегментів Micro та Small, що охоплюють події з відвідуваністю від 0 до 1000 осіб, найбільш ефективним інструментом прогнозування виявився алгоритм XGBoost. Зокрема, для підсегмента «Micro» зафіксовано суттєве зниження середньої абсолютної похибки (MAE) до значення 9.25, що відповідає відносній похибці на рівні 25.5%. Результати порівняльного аналізу свідчать, що така спеціалізована модель у 68 разів перевищує за точністю унітарний підхід. Ключовими предикторами відвідуваності в цій категорії виступили категоріальна приналежність заходу (category_subcategory_enc) та географічна прив'язка до конкретного міста проведення (city_enc).

У сегменті MidRange, який об'єднує події з аудиторією від 1 000 до 60 000 осіб та характеризується максимальною волатильністю даних (CV = 1.03), пріоритетним алгоритмом став RandomForest. Дана модель продемонструвала показник MAE на рівні 2481.16, випередивши за точністю XGBoost, похибка якого склала 3117.82. Така перевага обумовлена вищою здатністю ансамблевого навчання на основі беггінгу до генералізації ознак організаторів (organizer_enc). Значущість цього параметру в даному сегменті досягла домінуючого показника 0.46, що підкреслює репутаційну залежність відвідуваності подій середнього масштабу.

Для сегмента Giant, що включає масштабні заходи з кількістю відвідувачів понад 60 000 осіб, було застосовано процедуру оптимізації гіперпараметрів за допомогою методу GridSearch. У результаті обидва алгоритми продемонстрували високу точність із відносною похибкою в межах лише 2.5%. Алгоритм XGBoost показав незначну перевагу з MAE на рівні 1749.67 за рахунок глибшої обробки латентних семантичних ознак назв заходів, зокрема компоненти title_pca_2, та успішної інтеграції фактора часової тривалості події duration_hours.

V. Висновки

Проведене дослідження демонструє, що стратегія "DivideandConquer" (розділяй та володарюй) у застосуванні до гетерогенних даних дозволяє подолати обмеження класичних регресійних моделей. Перспективи подальших досліджень полягають у динамічному визначенні меж стратифікації за допомогою алгоритмів кластеризації без вчителя, що дозволить системі самостійно адаптуватися до нових типів даних.

Список використаних джерел

1. Chen, T., & Guestrin, C. (2016). XGBoost: A ScalableTreeBoostingSystem. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and DataMining.- pp.785-794
2. Breiman, L. (2001). Random Forests. MachineLearning, 45(1). – pp.5–32.

УДК 004.413

АЛГОРИТМ АДАПТИВНОЇ МАРШРУТИЗАЦІЇ ЗАПИТІВ МІЖ ДЕРЕВАМИ РІШЕНЬ ТА ВЕЛИКИМИ МОВНИМИ МОДЕЛЯМИ У РЕГУЛЬОВАНИХ ГАЛУЗЯХ

Войтюк І.Ф.[1)], Мельничук Д.С.[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] магістрант

I. Постановка проблеми

Корпоративні системи пошуково-доповненої генерації (Retrieval-Augmented Generation, RAG) у регульованих галузях стикаються з фундаментальною дилемою. З одного боку, великі мовні моделі (LLM) забезпечують природну гнучкість відповідей на запити користувача, але є ймовірнісними та схильними до галюцинацій. З іншого боку, дерева рішень забезпечують детермінованість та повну прослідковуваність – для будь-якої комбінації відповідей можна точно вказати, який вихідний стан буде досягнуто, що критично у фінансах (KYC/AML, MiFID II), медицині (клінічні протоколи ACC/AHA, NCCN) та юриспруденції (процедури due diligence). Проте дерева рішень мають істотний недолік – жорсткість: користувач змушений рухатися визначеним шляхом, навіть якщо його запит формулюється у термінах, що не відповідають дослівно вузлам дерева. Ці два підходи традиційно протиставляються одне одному. Однак для побудови надійних корпоративних AI-копілотів необхідно поєднати переваги обох: гарантовану детермінованість у регуляторно-критичних випадках та гнучкість у решті сценаріїв. Існуючі реалізації використовують ручне правило-орієнтоване перемикання, що не масштабується. Це обумовлює актуальність розробки алгоритму адаптивної маршрутизації, який автоматично обирає шлях відповіді на основі семантичної близькості запиту до регуляторної процедури.

II. Мета роботи

Метою роботи є розробка алгоритму адаптивної маршрутизації запитів між детермінованою відповіддю на основі дерева рішень та генеративною відповіддю великої мовної моделі з контролем достовірності, що дозволяє у регуляторно-критичних випадках гарантовано йти процедурним шляхом, а в інших – використовувати гнучкість LLM.

III. Алгоритм адаптивної маршрутизації

В основу алгоритму покладено ідею спільного індексування неструктурованих документів та структурованих дерев рішень у єдиному векторному просторі. Кожен документ бази знань поділяється на фрагменти, кожен фрагмент перетворюється на embedding-вектор моделлю text-embedding-3-large та індексується у векторній базі. Кожен вузол $v \in T$ дерев рішень обробляється тією ж моделлю перетворення на вектора індексується у тій самій базі з метаданою type='decision_node'. Це дозволяє одним пошуковим запитом одночасно отримувати релевантні фрагменти документів та найбільш доречні гілки дерев рішень.

Семантична близькість запиту q до вузла v дерева рішень обчислюється за косинусною метрикою (1):

$$sim(q,v) = (q_{emb}, v_{emb}) / (|\, q_{emb} \,| \cdot |\, v_{emb} \,|) \quad (1)$$

де q_{emb} – embedding-вектор запиту, v_{emb} – embedding-вектор текстового опису вузла (питання та контексту вузла).

Рішення про маршрут приймається за пороговою функцією (2):

$$route(q) = \{DT,\ якщо \max_{v \in T}\ sim(q,v) \geq \tau_{dt};\ LLM,\ інакше\} \quad (2)$$

де $\tau_{dt} = 0.75$ – поріг адаптивної маршрутизації. Якщо обрано шлях DT, повертається сценарна відповідь, прив'язана до вузла $v^* = \arg\max sim(q,v)$; якщо обрано шлях LLM, виконується гібридний пошук документів та генерація відповіді з триетапним контролем достовірності.

Гібридний пошук поєднує лексичний BM25 та щільний векторний пошук, з ранжуванням через Reciprocal Rank Fusion за формулою (3):

$$RRF(d) = \sum_i 1/(k + rank_i(d)) \quad (3)$$

де $\text{rank}_i(d)$ – позиція документа d у i-му впорядкованому списку, $k = 60$ – згладжуючий параметр. Топ-50 кандидатів реранжуються крос-енкодерною моделлю Cohere Rerank v3, після чого топ-5 передається до LLM як контекст.

Узагальнено алгоритм адаптивної маршрутизації описується послідовністю кроків, наведеною у таблиці 1.

Таблиця 1

Покрокова специфікація алгоритму адаптивної маршрутизації

Крок	Дія	Результат / умова виходу
1	Маскування PII у запиті	Очищений запит q (Amazon Comprehend / Microsoft Presidio)
2	Обчислення embedding запиту	Вектор q_emb (text-embedding-3-large)
3	Пошук найближчого вузла дерева рішень	Вузол v* з максимальною sim(q, v); фільтр type='decision_node'
4	Перевірка порогу маршрутизації	Якщо $\text{sim}(q, v^*) \geq \tau\_dt = 0{,}75$ – DT-шлях; інакше – LLM-шлях
5а	DT-шлях: повернення сценарної відповіді	Детермінована відповідь з посиланням на гілку дерева
5б	LLM-шлях: гібридний пошук + генерація	Топ-5 фрагментів після RRF та реранжування; відповідь LLM
6	Триетапний контроль достовірності F	Якщо $F \geq \theta = 0{,}9$ – відповідь приймається; інакше – повтор зі strict-промптом
7	Запис аудитної події	EventBridge → S3 Object Lock (Compliance Mode, retention 7 років)

Поріг $\tau_{dt} = 0{,}75$ обрано емпірично за результатами апробації як компроміс між помилковими спрацьовуваннями дерева на запитах, що не відповідають процедурі (при нижчих порогах), та пропуском процедурно-релевантних запитів (при вищих).

IV. Програмна реалізація та апробація

Запропонований алгоритм реалізовано мовою TypeScript у вигляді AWS Lambda-функцій з координацією через API Gateway. Векторна база – Pinecone (безсерверний режим, k-NN пошук $p95 \leq 50$ мс), модель перетворення на вектори– text-embedding-3-large, генератор – GPT-4o, модель арбітра – GPT-4o-mini, реранжувач – Cohere Rerank v3. Аудитне журналювання реалізовано через Amazon S3 з активованим Object Lock у Compliance Mode для забезпечення регуляторної прослідковуваності.

Експериментальну апробацію проведено на трьох доменах (фінансовий комплаєнс, клінічні протоколи, юридичні консультації), тестовий набір – 600 запитів (по 200 на домен). Спільне індексування вузлів дерев рішень разом з документами підвищило показник Recall@5 з 0,89 (базова RAG) до 0,90–0,93, а MRR@10 – з 0,72 до 0,78–0,82. У сукупності з триетапним контролем достовірності частку галюцинацій зменшено з 12,4 % до 2,8–3,7 % залежно від домену. Затримка k-NN пошуку у Pinecone утримується в межах $p95 \leq 50$ мс на колекції в десятки мільйонів векторів, тому рішення про маршрутизацію не створює помітного впливу на загальну затримку циклу обробки (3,2–3,5 с p95).

Висновок

У роботі запропоновано алгоритм адаптивної маршрутизації запитів між детермінованою відповіддю на основі дерева рішень та генеративною відповіддю LLM з контролем достовірності. Ключовою особливістю алгоритму є спільне індексування неструктурованих документів та структурованих дерев рішень у єдиному векторному просторі, що дозволяє одним пошуковим запитом приймати рішення про маршрут на основі косинусної близькості до вузла дерева. Підхід забезпечує гарантований детермінізм у регуляторно-критичних випадках при збереженні гнучкості LLM для решти сценаріїв. Експериментальна апробація на трьох доменах підтвердила підвищення показників якості пошуку та зменшення частки галюцинацій у 3,3–4,4 рази. Подальші дослідження спрямовані на динамічне налаштування порогу τ_{dt} на основі історії запитів конкретної організації.

Список використаних джерел

1. Gao Y., Xiong Y., Gao X. et al. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv:2312.10997. 2023.
2. Robertson S., Zaragoza H. The Probabilistic Relevance Framework: BM25 and Beyond // Foundations and Trends in Information Retrieval. 2009. Vol. 3, No. 4. pp. 333–389.
3. Cormack G. V., Clarke C. L. A., Buettcher S. Reciprocal Rank Fusion Outperforms Condorcet and Individual Rank Learning Methods // Proc. of SIGIR 2009. 2009. pp. 758–759.

УДК 004.4

ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ МЕДИЧНОЇ НАВІГАЦІЇ НА ОСНОВІ ЧАТ-БОТА ДЛЯ ПРОМИСЛОВОГО ПІДПРИЄМСТВА

Задорожний А.А.[1)], Стасів І.С.[2)], Порплиця Н.П.[3)]
Західноукраїнський національний університет
[1)]бакалавр; [2,3)]к.т.н., доцент

I. Постановка проблеми

У сучасних умовах цифровізації промислового сектору особливої актуальності набуває впровадження інтелектуальних систем для оптимізації внутрішніх корпоративних процесів. Для таких великих підприємств, як ПрАТ «Оболонь», критично важливим є швидке надання медичної допомоги та координація працівників. Традиційні методи запису до лікаря через телефон або особистий візит створюють надмірне навантаження на медичний персонал та призводять до виникнення черг. Відсутність автоматизованого збору анамнезу не дозволяє лікарю заздалегідь підготуватися до прийому, що знижує якість медичного обслуговування. Таким чином, розробка інтелектуального чат-бота, здатного проводити попереднє опитування та маршрутизацію пацієнтів, є актуальною науково-практичною задачею.

II. Аналіз існуючих рішень

На сьогодні існують системи типу «Helsi» або корпоративні портали, проте вони вимагають значних обчислювальних ресурсів та встановлення додаткового програмного забезпечення на смартфони працівників. На відміну від них, використання чат-ботів у месенджері Telegram на базі Low-code платформ дозволяє реалізувати гнучкий інтерфейс без навантаження на пам'ять пристроїв користувачів та з мінімальними витратами на підтримку серверної частини.

III. Мета роботи

Метою роботи є підвищення ефективності роботи медичних пунктів промислового підприємства шляхом розробки та впровадження інтелектуальної системи на основі чат-бота для автоматизації збору первинної інформації про стан здоров’я пацієнтів та керування чергою.

IV. Програмна реалізація та архітектура

Для реалізації проекту обрано платформу SendPulse, яка забезпечує стабільну роботу логічних ланцюжків. Архітектура системи включає модуль взаємодії та логічний модуль.

Модуль взаємодії: реалізований через Telegram API, забезпечує гібридний метод вводу (кнопки для типових симптомів та текстові поля для опису скарг) (див. рис.1).

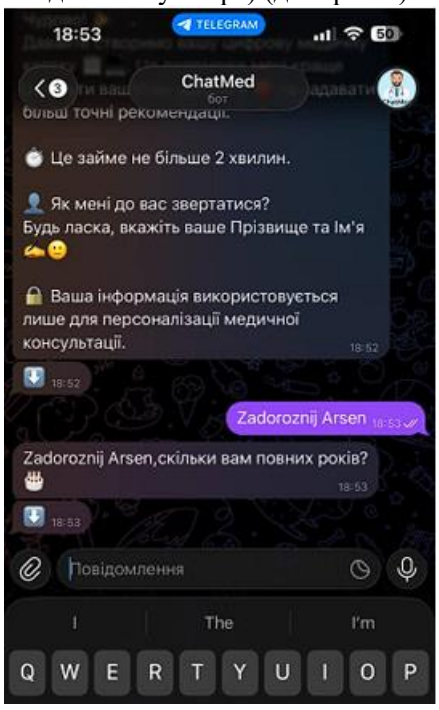

Рисунок 1 – Процес збору первинних даних користувача та методом вільного вводу

Логічний модуль: здійснює валідацію даних (перевірка формату номера телефону та ПІБ за допомогою регулярних виразів).

Модуль зберігання реалізовано як інтегровану CRM-систему (див.рис.2), що накопичує статистику звернень та формує цифрову картку пацієнта (див. рис.3).

	Ім'я	Прізвище та Ім'я	age	city	phone
☐	Pavlo Krasovskyi 10 бер	Павло Красовський	25	Гданськ	+48793035839
☐	Liankaaa 💋 10 бер				
☐	Admin Власник 26 лист	Ігор Петрович	47	Вроцлав	+4840483903

Рисунок 2 – Приклад CRM системи,яка зберігає дані для спеціаліста

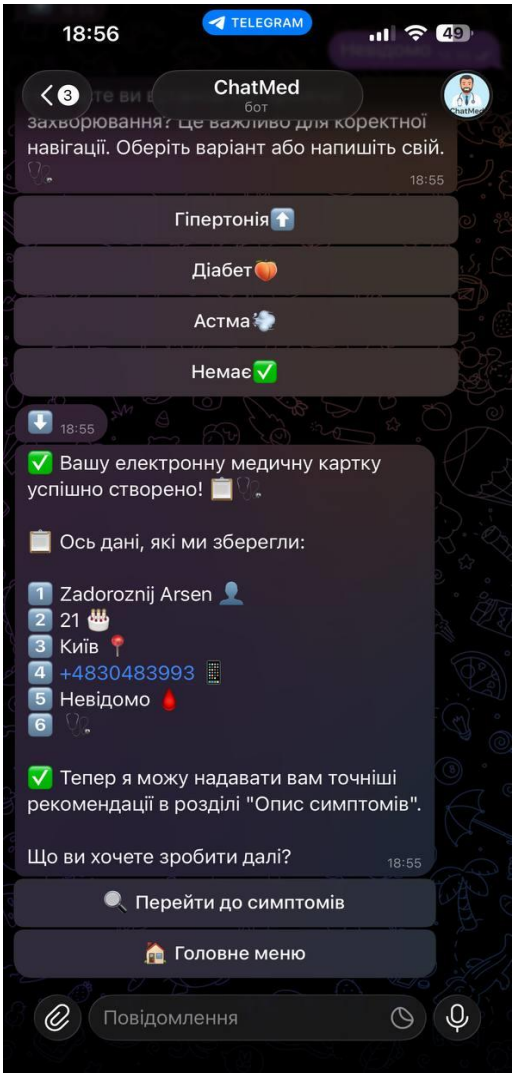

Рисунок 3 – Результат створення медичної картки

Висновок

Розроблена інтелектуальна система дозволяє скоротити час на первинну реєстрацію пацієнтів у 4-5 разів. Використання Low-code інструментів забезпечило високу швидкість розробки та легкість інтеграції в IT-інфраструктуру ПрАТ «Оболонь» масштабованість та високу продуктивність системи.

Список використаних джерел

1. Співак І. Я., Пукас А. В. Методичні вказівки до виконання ВКР. Тернопіль: ЗУНУ, 2022. 45 с.
2. SendPulse: Офіційна документація по розробці чат-ботів. URL:https://sendpulse.ua/knowledge-base/chatbothttps://sendpulse.ua/knowledge-base/chatbot.
3. Дорош О. В. Інтелектуальні системи в медицині: теорія та практика. К.: Наука, 2024. 120 с.

УДК 004.4

ФУНКЦІЯ БАГАТОРЕЖИМНОГО ПОШУКУ КОРИСТУВАЧІВ ВЕБ-ПЛАТФОРМИ СОЦІАЛЬНОЇ ІНТЕГРАЦІЇ ВНУТРІШНЬО ПЕРЕМІЩЕНИХ ОСІБ

Кліщ С.С.[1)], Крепич С.Я.[2)], Співак І.Я.[3)], Крепич Р.В.[4)]
[1-3)] Західноукраїнський національний університет
[4)] ВСП Кам'янець-Подільський фаховий коледж НРЗВО "Кам'янець-Подільський державний інститут"
[1)] бакалавр; [2-3)]к.т.н., доцент; [4)] викладач

I. Постановка проблеми

Соціальна інтеграція внутрішньо переміщених осіб (ВПО) є однією з найбільш гострих проблем сучасної України. За даними Міжнародної організації з міграції, станом на 2024 рік в Україні налічується понад 4,9 мільйонів ВПО, які зазнали стресу від втрати соціальних зв'язків, виходу із звичного оточення та необхідності адаптації до нового середовища. Ключовим фактором успішної інтеграції є відновлення соціальних контактів - знайомство із земляками за регіоном походження, з мешканцями нового місця проживання та з людьми зі схожими інтересами. Існуючі веб-платформи не враховують специфіку ВПО та не пропонують спеціалізованих механізмів пошуку за критеріями походження. Відсутність зручного інструменту мультимодального пошуку призводить до того, що ВПО витрачають значний час на самостійні спроби налагодити контакти, що сповільнює процес інтеграції. Актуальність дослідження зумовлена необхідністю розробки спеціалізованої функції пошуку, яка автоматизує процес підбору релевантних користувачів за множинними критеріями.

II. Мета роботи

Метою роботи є розробка функції багаторежимного пошуку користувачів, орієнтованої на специфіку аудиторії ВПО, та її програмна реалізація у складі веб-платформи соціальної інтеграції. Запропонована функція має забезпечувати підбір релевантних співрозмовників за чотирма самостійними критеріями: спільне регіональне походження, спільне поточне місце проживання, спільні особисті інтереси, а також за довільною комбінацією додаткових фільтрів.

III. Функціональні режими пошуку

Запропонована функція пошуку реалізує чотири незалежних режими, кожен з яких розв'язує окрему задачу соціальної інтеграції ВПО:

1. Режим «Земляки» призначений для пошуку користувачів, що мають спільне з поточним користувачем регіональне походження та проживають у тому ж місті після переміщення. Цей режим є найбільш специфічним для аудиторії ВПО, оскільки відновлення зв'язків із земляками з рідного регіону у новому місці проживання має ключове психологічне значення для адаптації . Фільтрація виконується кон'юнкцією двох умов: збігу регіону походження та збігу поточного міста.
2. Режим «У моєму місті» забезпечує знайомство з іншими ВПО, які проживають у тому ж місті, незалежно від регіону походження. Цей режим орієнтований на формування локальних спільнот взаємодопомоги. Фільтрація здійснюється за одним полем – поточним містом.
3. Режим «Спільні інтереси» здійснює пошук користувачів, у яких є хоча б один спільний інтерес з поточним користувачем. Інтереси зберігаються у вигляді масиву рядків у профілі користувача, що дозволяє ефективно виконувати запит на перетин множин засобами PostgreSQL. Цей режим орієнтований на хобі-комунікацію та підтримку емоційного добробуту.
4. Режим «Фільтри» надає користувачу можливість ручного формування запиту з довільною комбінацією критеріїв: ім'я, місто, регіон, регіон походження, інтереси. Цей режим дозволяє виконувати точковий пошук конкретних осіб або груп користувачів за нестандартними поєднаннями ознак.

Особливістю функції є наявність попередніх перевірок (guard-умов) перед активацією перших трьох режимів. Якщо у профілі поточного користувача відсутні дані, необхідні для виконання режиму, система відображає інформаційне повідомлення з посиланням на сторінку редагування профілю замість порожніх результатів. Це реалізує принцип проактивного інтерфейсу: користувач отримує не порожній результат, а зрозумілу інструкцію щодо подальших дій.

Для формалізації поведінки функції в роботі застосовано модель скінченного автомата станів на основі UML StateMachineDiagram. Кожен режим пошуку представлено як окремий стан, а перемикання між режимами – як переходи з guard-умовами. Така формалізація забезпечує детермінованість поведінки інтерфейсу та спрощує його тестування (див.рис.1).

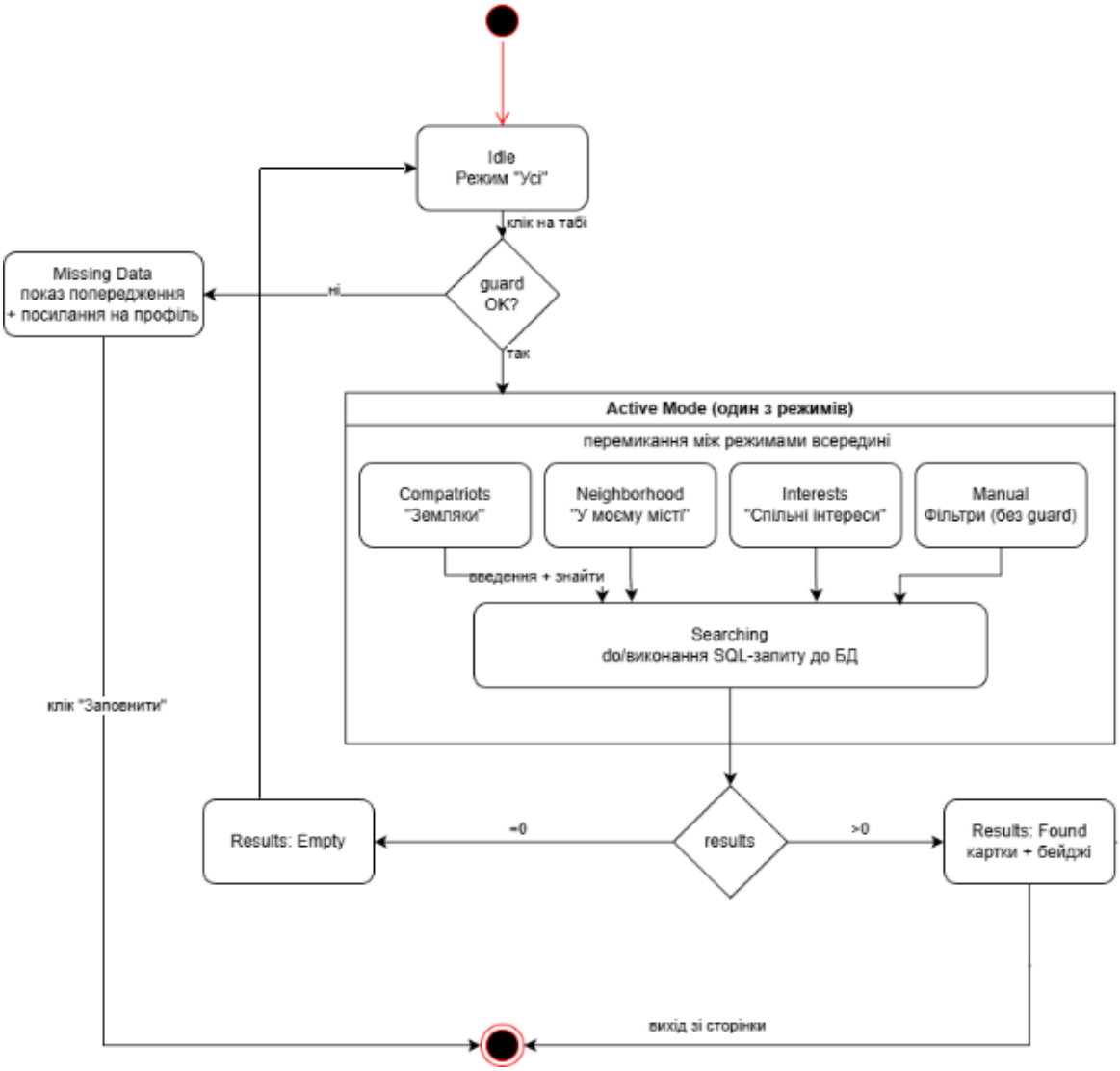

Рисунок 1 – Діаграма станів функції пошуку користувачів

IV. Програмна реалізація

Функцію пошуку реалізовано у складі веб-платформи на основі бібліотеки React та хмарної платформи Supabase. Клієнтська частина побудована на фреймворку Next.js з архітектурою AppRouter, серверна частина — на PostgreSQL із вбудованим механізмом RowLevelSecurity для контролю доступу до даних користувачів. Активний режим пошуку зберігається у параметрі URL ?mode=, що забезпечує підтримку deeplink, відновлення режиму при перезавантаженні сторінки та можливість обміну посиланнями між користувачами. SQL-запити будуються динамічно залежно від поточного режиму та значень фільтрів за допомогою querybuilder API клієнтської бібліотеки SupabaseJavaScriptClient. Нижче наведено ключовий фрагмент функції побудови запиту:

```
letquery = supabase.from('profiles')
  .select('*').neq('id', currentUser.id)

switch (mode) {
case 'compatriots':
query = query
      .eq('origin_region', user.origin_region)
      .ilike('city', user.city)
break;
case 'neighborhood':
query = query.ilike('city', user.city)
break;
case 'interests':
query = query.overlaps('interests', user.interests)
break;
}
```

У наведеному фрагменті метод overlaps() використовує оператор перетину масивів PostgreSQL&& для пошуку користувачів зі спільними інтересами, метод ilike() реалізує case-insensitive пошук за шаблоном, що особливо важливо для української мови з різними формами написання географічних назв, а ланцюжкові виклики .eq() формують кон'юнкцію умов фільтрації. Перевірка guard-умов виконується перед побудовою запиту: якщо у профілі поточного користувача відсутні необхідні поля, замість виконання запиту відображається інформаційне попередження. Нижче наведено фрагмент логіки обробки відсутніх даних для режиму «Земляки»:

```
if (mode === 'compatriots' &&
    (!user.origin_region || !user.city)) {
return<MissingDataWarning
reason='Заповніть регіон походження та місто'
link='/profile/edit' />
}
```

Результати пошуку відображаються у вигляді карток профілів з бейджами матчингу, які пояснюють користувачу, за якою саме ознакою інший користувач потрапив у результати: «Земляк» - при збігу регіону походження, «Місто» - при збігу поточного міста, «N спільних інтересів» - при перетині множин інтересів із зазначенням точної кількості. Бейджі обчислюються на клієнті безпосередньо при рендерингу картки на основі порівняння даних поточного користувача з даними знайденого профілю.

Графічний інтерфейс функції пошуку у режимі «Земляки» наведено на рисунку 2.

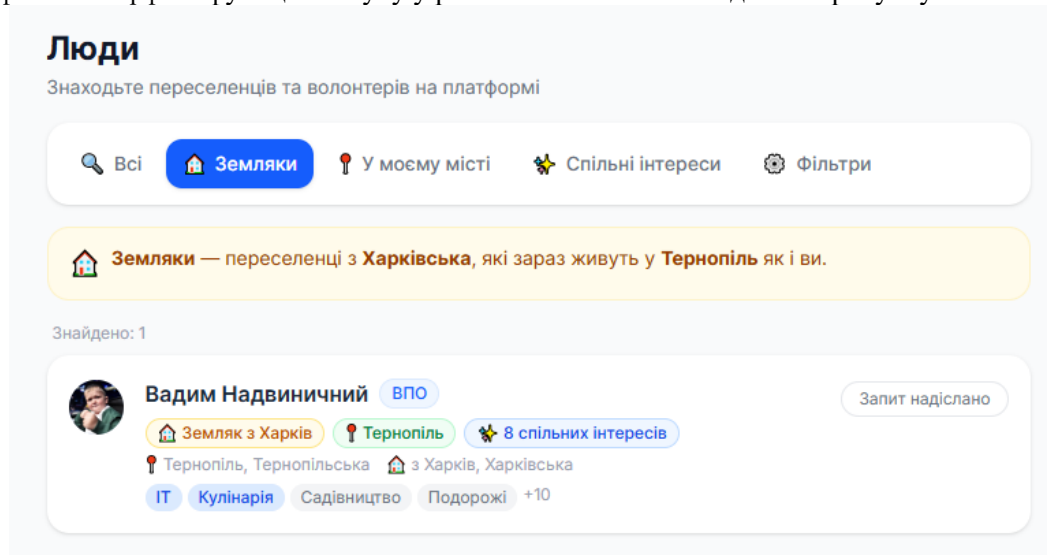

Рисунок 2 – Інтерфейс функції пошуку у режимі «Земляки»

Висновок

У роботі розроблено функцію багаторежимного пошуку користувачів для веб-платформи соціальної інтеграції внутрішньо переміщених осіб. Поведінку функції формалізовано як скінченний автомат станів, що забезпечує детермінованість інтерфейсу та проактивну обробку випадків відсутності даних у профілі через окремий стан попередження. Програмну реалізацію виконано на стеку Next.js, React та Supabase з використанням механізмів PostgreSQL для ефективного пошуку за масивами та case-insensitive фільтрації за рядками. Подальші дослідження спрямовані на розширення функції додатковими режимами (за віком, за професією, за рідною мовою) без модифікації існуючих за рахунок симетричної структури автомата.

Список використаних джерел

1. Banks A., Porcello E. Learning React: Modern Patterns for Developing ReactApps. 2nd ed. – O'ReillyMedia, 2020. – 310 p.
2. Supabase Inc. Supabase Documentation [Електронний ресурс] – Режим доступу: URL: https://supabase.com/docs
3. S.Krepych, I.Spivak, “Algorithm of automatic generation of hotel descriptions using templetes based on Markov chains”, International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). 2018. – pp.257-260
4. V.Krutko, I.Spivak, S.Krepych, “An approach to assessing the reliability of software systems based on a graph model of method dependence”, CEUR Workshop Proceedings, 2024, 3662, pp.37-47
5. Krepych S., Spivak I., Spivak S., “Methodology of formation of the individual study plan of the student based on the graph model of the dependence of discilplines”, CEUR Workshop Proceedings, 2023, 3426, pp.298-307

УДК 004.8

АРХІТЕКТУРНІ ОСОБЛИВОСТІ МОДУЛЯ ЕНЕРГОЕФЕКТИВНОЇ ОРКЕСТРАЦІЇ КОНТЕЙНЕРІВ ДЛЯ ХМАРНИХ ОБЧИСЛЕНЬ

Вишневський Ю.Ю.[1],Тимчишин В.С[2], Ковальський А.А.[3]
Західноукраїнський національний університет
[1]магістрант; [2]д.філ., доцент; [3]магістрант

Постановка проблеми

Стрімкий розвиток сучасних хмарних інфраструктур супроводжується експоненційним зростанням споживання електроенергії центрами обробки даних. Стандартні алгоритми оркестрації (наприклад, базовий планувальник Kubernetes) орієнтовані виключно на рівномірне балансування навантаження. Збереження фізичних серверів в активному стані при мінімальному навантаженні є вкрай неефективним через високу частку статичної потужності (потужності простою), що спричиняє невиправдані фінансові та екологічні збитки.

Існуючі рішення для оптимізації зазвичай покладаються на зовнішні скрипти або бази даних, що генерує мережеві затримки та порушує швидкодію. Відтак, виникає потреба у проектуванні автономної архітектури розширеного планувальника, яка б забезпечувала агресивну консолідацію контейнерів та переведення вивільнених вузлів у режим глибокого сну без створення додаткового навантаження на систему керування.

Мета роботи

Метою роботи є проектування та програмна реалізація архітектури розширеного планувальника для платформи Kubernetes, здатного здійснювати енерго-обізнане розміщення та динамічну евакуацію мікросервісів з використанням сучасних патернів Cloud-Native розробки.

Архітектурні рішення та особливості реалізації

Запропонований модуль складається з чотирьох основних функціональних компонентів, що взаємодіють через внутрішні інтерфейси платформи:

- колектор телеметрії як фоновий сервіс, який періодично опитує API серверів моніторингу та формує актуальний зріз утилізації процесора, пам'яті та поточного енергоспоживання всіх хостів.
- аналізатор стану кластера як компонент, який реалізує алгоритм класифікації вузлів. Він визначає перевантажені та недовантажені сервери на основі заданих статистичних порогів.
- контролер евакуації (Мігратор) як модуль, відповідальний за формування черги міграцій. Він ініціює вилучення контейнерів з проблемних вузлів шляхом відправки декларативних запитів на видалення до API-сервера оркестратора.
- енерго-обізнаний плагін планувальника де є безпосередня реалізація жадібного алгоритму розміщення в рамках каркасу планування, що перехоплює нові та евакуйовані поди і призначає їх на хости з мінімальним приростом потужності.

Для візуалізації взаємодії зазначених компонентів та загальної структури розробленого програмного забезпечення нижче наведено архітектурну схему модуля (див.рис.1).

Завдяки такій модульній організації забезпечується висока відмовостійкість системи. Якщо фоновий аналізатор стану або колектор метрик тимчасово втратять зв'язок із сервером моніторингу, плагін планувальника автоматично переходить у безпечний режим функціонування, використовуючи останні збережені в локальному кеші профілі енергоспоживання, або тимчасово делегує повноваження стандартним алгоритмам, що запобігає повному зупиненню процесів розгортання додатків у хмарі.

Архітектура передбачає чіткий поділ на керівну та робочу площини. На керівному вузлі розгортається модифікований екземпляр планувальника (Pod: kube-scheduler), всередині якого скомпільовано розроблений енерго-обізнаний плагін, написаний мовою Go. Такий підхід усуває необхідність додаткових мережевих запитів під час фази ранжування. Поруч у вигляді окремого мікросервісу (Pod: EnergyController) функціонують компоненти колектора телеметрії та аналізатора стану, які періодично опитують систему моніторингу та зберігають свої результати у кластерному сховищі etcd.

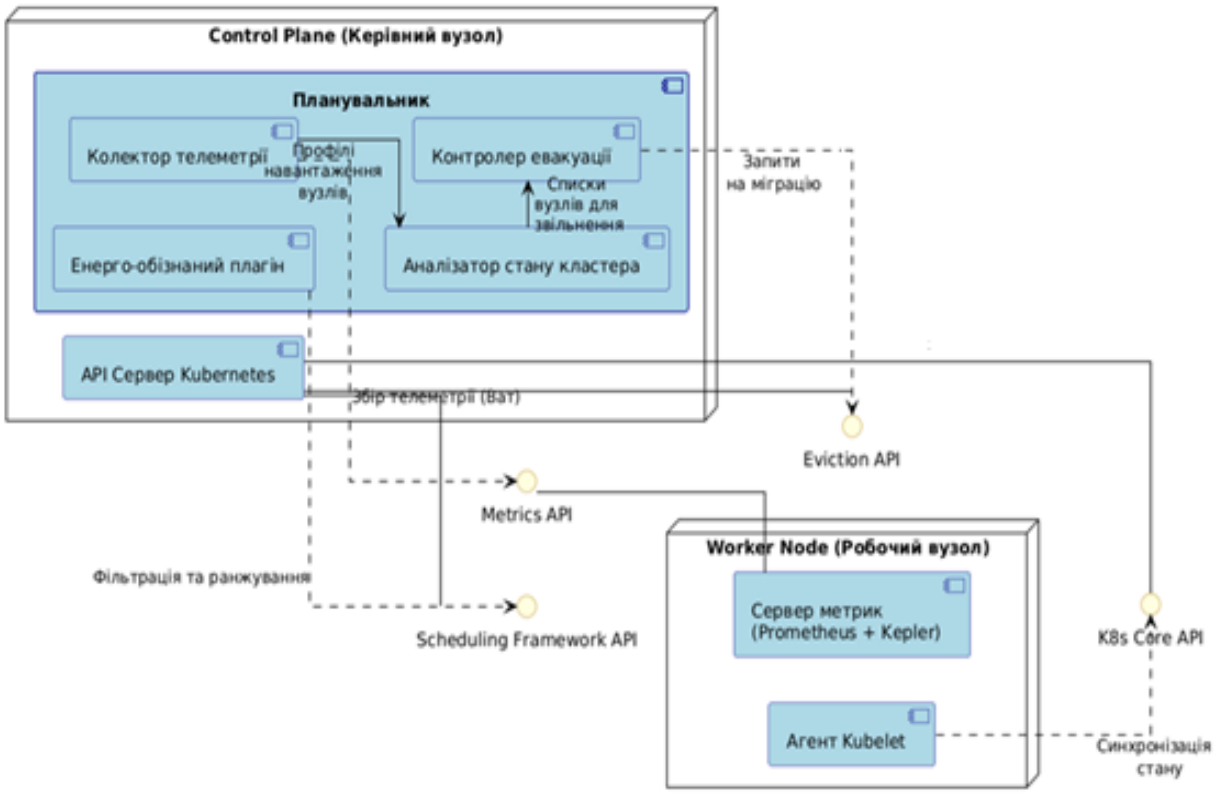

Рисунок 1 – Компонентна архітектура програмного модуля енергоефективногопланування

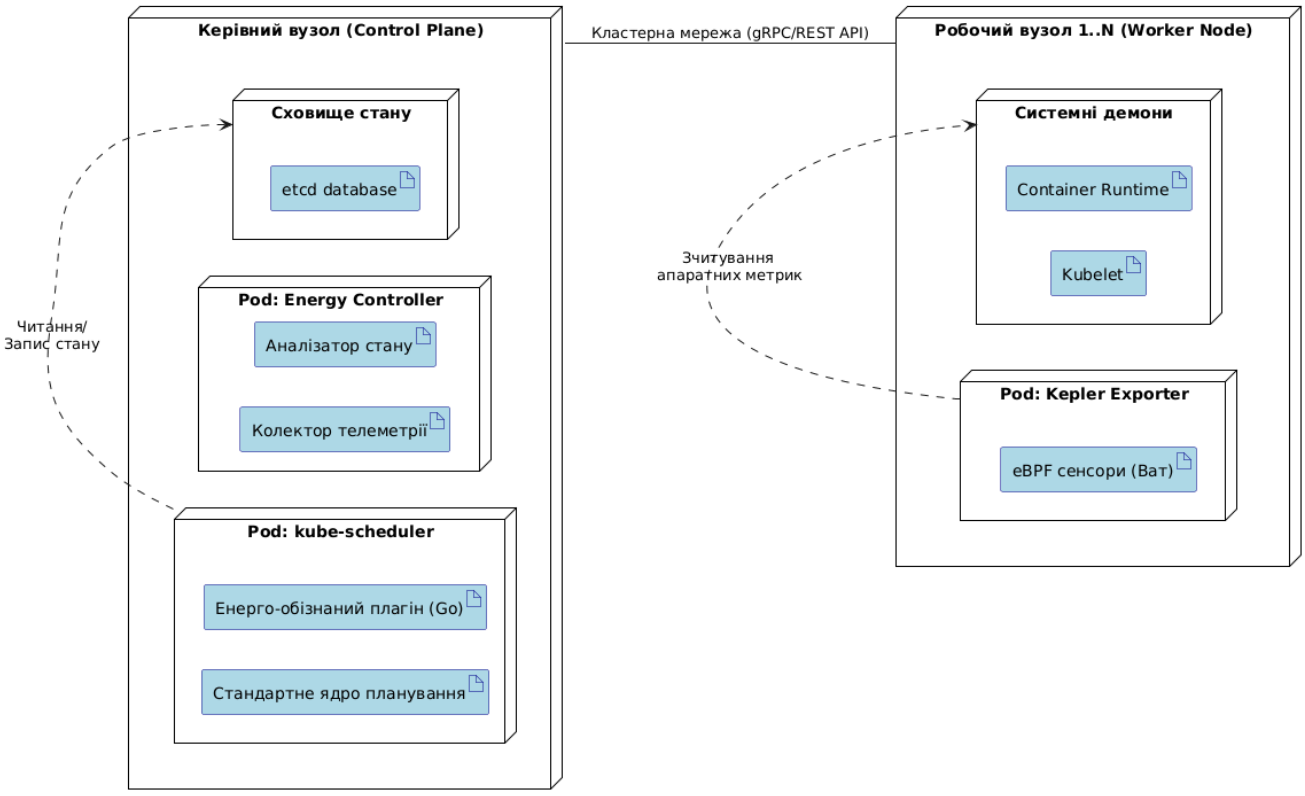

Рисунок 2 – UML діаграма розгортання

На робочих вузлах інфраструктури, крім стандартних системних компонентів (Kubelet та ContainerRuntime), як привілейований процес розгортається агент KeplerExporter. Він використовує технологію eBPF для зчитування апаратних лічильників процесора та мережевих інтерфейсів в обхід ядра ОС, що дозволяє отримувати високоточні показники споживаної потужності (у Ватах) без

створення додаткового навантаження на самі робочі вузли. Взаємодія між площинами здійснюється через захищену кластерну мережу за протоколами REST API та gRPC.

Для опису поведінки системи в часі та ілюстрації процесу прийняття рішень під час розгортання навантаження побудовано UML діаграму послідовності (див. рис.3). Вона демонструє повний життєвий цикл управління контейнером від моменту запиту до його фактичного запуску на оптимальному сервері.

Процес ініціюється користувачем або системою безперервної інтеграції (CI/CD), яка надсилає запит до API-сервера на розгортання нового мікросервісу (Pod). API-сервер реєструє об'єкт і ставить його у чергу нерозподілених завдань.Паралельно в системі функціонує асинхронний фоновий цикл: модуль моніторингу безперервно передає метрики до аналізатора стану. Якщо аналізатор фіксує перевантаження певного вузла, він превентивно генерує запити на евакуацію (Eviction API), додаючи існуючі контейнери до тієї ж черги планування.

Коли енерго-обізнаний планувальник бере контейнер із черги, він ініціює двохетапний процес: спочатку виконується фільтрація (відкидання вузлів з нестачею ресурсів), а потім – ранжування, де для кожного кандидата розраховується потенційний приріст енергоспоживання. Визначивши вузол із мінімальним приростом потужності, планувальник відправляє API-серверу команду прив'язки. Після цього Kubelet обраного робочого вузла отримує інструкцію, ініціалізує контейнерне середовище і переводить додаток у робочий стан.

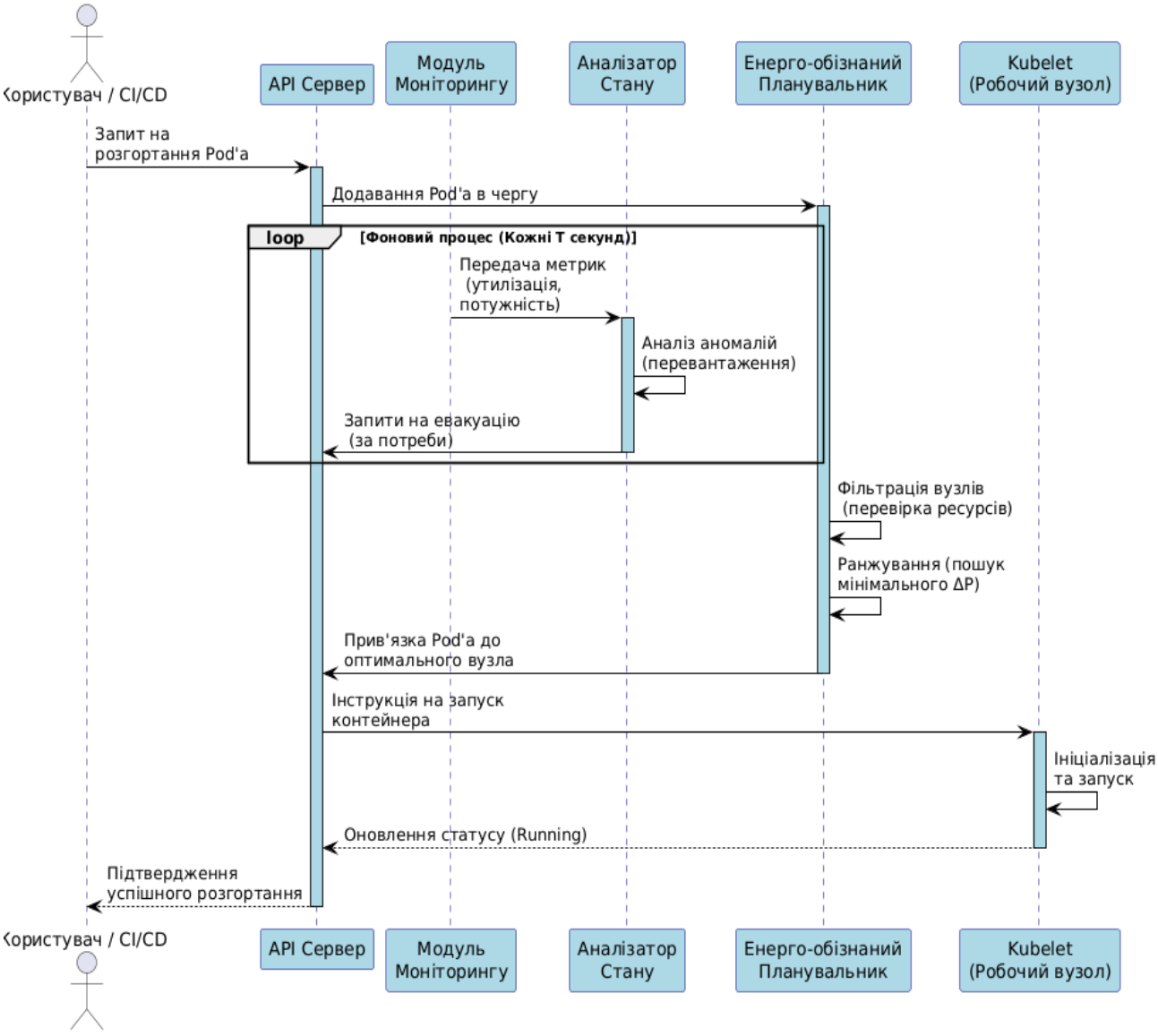

Рисунок 3 – UML діаграма послідовності

У модулі ми використовуємо спеціалізовані сховища, які вже є стандартом для екосистеми Kubernetes:

- etcd (Сховище ключ-значення). Це вбудована розподілена база даних самого Kubernetes. Наш планувальник не зберігає стан сам, він просто взаємодіє з API-сервером, який читає і пише об'єкти (стан вузлів, конфігурації подів) безпосередньо в etcd;
- TSDB (TimeSeries Database). Для збереження енергетичних метрик і телеметрії (Ватти, відсотки завантаження) класичні таблиці не підходять. Ми використовуємо вбудовану базу даних часових рядів системи Prometheus. Вона оптимізована для збереження мільйонів точок метрик у секунду;
- локальний In-Memory Кеш. Всі проміжні дані (наприклад, поточний розрахований рівень споживаної потужності для швидкого доступу під час фази ранжування) наш плагін зберігає просто в оперативній пам'яті (у структурах мови Go). Кеш синхронізується з API у фоновому режимі.

У цій архітектурі планувальник є Stateless (без стану), що означає, що він не зберігає жодних даних самостійно, а лише спирається на швидкі механізми Kubernetes (InformerCache та etcd) та систему моніторингу TSDB (див. рис.4). Така архітектура без зовнішньої бази даних робить модуль легким, швидким і відмовостійким.

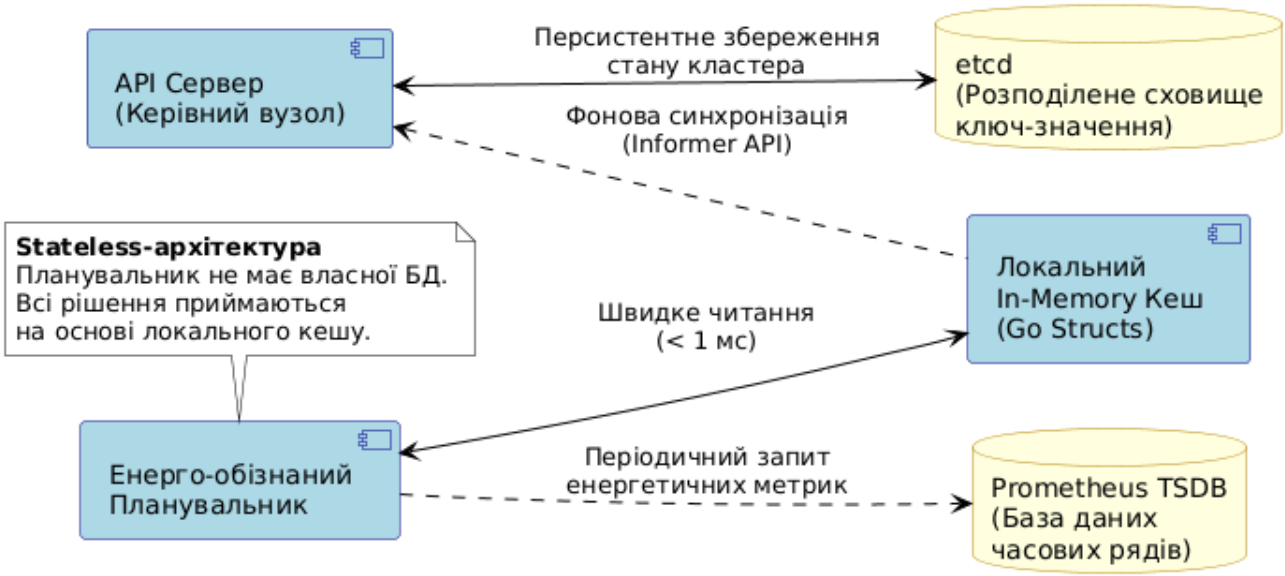

Рисунок 4 – UML діаграма архітектури доступу до даних за відсутності локальної бази даних

Висновок

Спроєктована архітектура програмного модуля на базі патернів Cloud-Native дозволила безшовно інтегрувати логіку енерго-обізнаного планування безпосередньо в керівну площину Kubernetes. Завдяки реалізації Stateless-підходу та використанню потокобезпечногоIn-Memoryкешування замість реляційних баз даних, система гарантує обробку запитів із мінімальними затримкамита оптимізацію енергоспоживання хмарних інфраструктур без порушення вимог щодо продуктивності.

Список використаних джерел

1. Kubernetes Scheduling Framework. Офіційна документація Kubernetes. URL: https://kubernetes.io/docs/concepts/scheduling-eviction/scheduling-framework/
2. Rossi F., Cardellini V., Presti F. L. Energy-aware deployment of microservices. 2020 IEEE International Conference on Cloud Engineering (IC2E). IEEE, 2020. – pp.115–125.
3. Beloglazov A., Buyya R. Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines. Concurrency and Computation: Practice and Experience, 2012. Vol.24, No.13. pp.1397–1420.
4. I.Spivak, S.Krepych, V.Horishni, “Організація Cloud-аріхтектури для систем забезпечення функціональної придатності статичних систем”, Advanced Information Systems, T.3, No.2, pp. 35-39
5. Kepler (Kubernetes-based Efficient Power Level Exporter). Офіційний репозиторій проекту. URL: https://github.com/sustainable-computing-io/kepler

УДК 004.415

МЕТОДИ ТА ПРОГРАМНА СИСТЕМА ГЕНЕРАЦІЇ UML-ДІАГРАМ КЛАСІВ З ВИКОРИСТАННЯМ АБСТРАКТНИХ СИНТАКСИЧНИХ ДЕРЕВ

Гордійчук С.А.
Західноукраїнський національний університет
магістр

I.Постановка проблеми

Розвиток інформаційних технологій та ускладнення архітектури програмних систем зумовлюють зростання вимог до засобів аналізу та документування програмного забезпечення. Сучасні програмні системи часто складаються з великої кількості модулів, класів та інтерфейсів, взаємодія між якими не завжди очевидна без додаткового архітектурного представлення. Ручне створення та підтримка UML-діаграм є трудомістким процесом, через що технічна документація часто швидко застаріває і виникає розрив між структурою коду та її представленням. Особливу складність становить автоматизована побудова UML-діаграм для програм, написаних мовами JavaScript та TypeScript. Історично JavaScript розвивався як прототипно-орієнтована мова з динамічною типізацією, що ускладнює статичне визначення класів та зв'язків. Однак поширення TypeScript додає до JavaScript систему статичної типізації, інтерфейси, абстрактні класи та узагальнення, що наближує розробку до класичної об'єктно-орієнтованої моделі. Завдяки цьому з'являється можливість більш точного статичного аналізу коду на основі абстрактного синтаксичного дерева й семантичної інформації.

II. Мета роботи

Метою роботи є розроблення методу та програмної системи автоматизованої генерації UML-діаграм класів [3] з вихідного коду JavaScript/TypeScript [1] з використанням абстрактних синтаксичних дерев і засобів семантичного аналізу TypeScript Compiler API, рисунок 1.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

1. Проаналізувати еволюцію об'єктної моделі JavaScript і TypeScript та визначити особливості їх відображення в UML-діаграмах класів.
2. Дослідити принципи побудови абстрактних синтаксичних дерев і можливості сучасних інструментів синтаксичного та семантичного аналізу.
3. Формалізувати задачу трансляції AST-вузлів у елементи UML-діаграми класів та розробити відповідну математичну модель відповідності.
4. Запропонувати алгоритми ідентифікації класів, інтерфейсів, атрибутів, методів, модифікаторів доступу та зв'язків між елементами програмної системи.
5. Розробити програмну систему, яка виконує аналіз TypeScript-проєкту, формує внутрішню UML-модель і генерує текстовий опис діаграми для подальшої візуалізації.
6. Провести тестування програмної системи на прикладах програмного коду різної складності та оцінити якість і повноту згенерованих UML-діаграм.

III. Основна частина

У даній роботі розв'язано актуальне науково-практичне завдання автоматизованої генерації UML-діаграм класів із вихідного коду JavaScript/TypeScript. Процес було формалізовано як задачу трансляції множини абстрактних синтаксичних дерев (AST) програмного проєкту у графову UML-модель. Проаналізовано еволюцію об'єктної моделі JavaScript від прототипно-орієнтованого підходу до сучасного класового синтаксису ECMAScript і строго типізованої моделі TypeScript. Визначено, що наявність явних класів, інтерфейсів, модифікаторів доступу та generic-типів у TypeScript створює необхідні передумови для високоточного статичного аналізу архітектури.

Для реалізації поставлених завдань було спроєктовано програмну систему з модульною архітектурою. Вона включає такі ключові компоненти: модуль завантаження проєкту (ProjectLoader), модуль побудови AST (ASTParser), модуль семантичного аналізу (SemanticAnalyzer), модуль формування внутрішньої UML-моделі (UMLModelBuilder), модуль визначення зв'язків (RelationResolver) та генератор текстового опису діаграми (DiagramGenerator). Основним інструментом аналізу обрано TypeScript Compiler API. Цей вибір обґрунтовано тим, що він забезпечує доступ не лише до синтаксичної структури програми, а й до семантичної інформації про типи, символи, імпорти та міжфайлові залежності через механізм TypeChecker.

Особливу увагу приділено розробці внутрішньої UML-моделі, яка слугує проміжним представленням між синтаксичним деревом і кінцевим форматом візуалізації (Mermaid або PlantUML). Ця абстракція дозволяє відокремити логіку аналізу від логіки рендерингу та значно спрощує тестування системи. У ній детально описуються сутності проєкту: класи, інтерфейси, атрибути з урахуванням модифікаторів видимості (public, private, protected) та статичності, методи з їхніми параметрами та типами повернення, а також модулі.

В основу системи покладено математичну модель відповідності між вузлами AST та елементами UML. Запропоновано комплексні алгоритми визначення зв'язків між елементами. Зокрема, зв'язки успадкування та реалізації визначаються надійно на основі синтаксичних конструкцій extends і implements. Асоціації ідентифікуються шляхом аналізу типів полів класу. Для розрізнення агрегації та композиції система аналізує спосіб ініціалізації об'єктів: створення екземпляра через оператор new всередині класу трактується як композиція, а передача об'єкта ззовні через конструктор – як агрегація. Залежності виявляються на основі типів параметрів методів, типів повернення та локального використання об'єктів у коді.

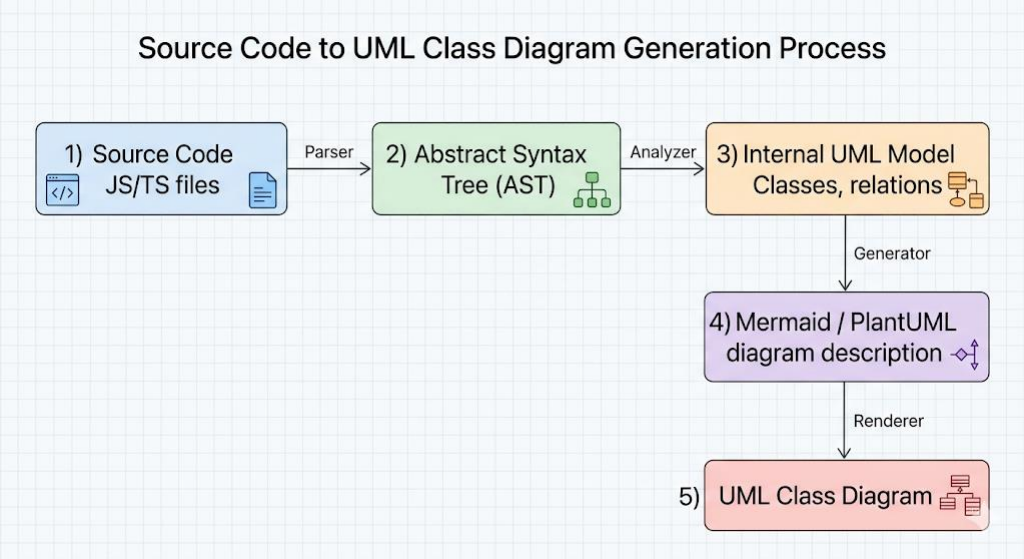

Рисунок 1 - Схема генерації UML-діаграми класів із TypeScript-коду

Додатково розроблено підхід до обробки специфічних та динамічних конструкцій TypeScript. Для представлення сутностей популярних фреймворків аналізуються декоратори, які автоматично транслюються у відповідні UML-стереотипи (наприклад, <<component>>, <<service>>, <<controller>>, <<entity>>, <<module>>). Це робить згенеровані діаграми значно інформативнішими та максимально наближеними до фактичної архітектури сучасних вебзастосунків.

Висновки

Для перевірки коректності розроблених методів проведено ґрунтовне тестування програмної системи на синтетичних, навчальних прикладах та фрагментах open-source-проєктів. Результати підтвердили високу ефективність підходу: для невеликих тестових проєктів досягнуто повне (100%) покриття виявлення класів і зв'язків. Для середніх навчальних проєктів значення F1-міри для визначення зв'язків становило 0.91, а для фрагментів реальних відкритих проєктів — 0.86. Водночас виявлено певні обмеження, пов'язані зі складністю аналізу функціонального стилю програмування (наприклад, React-компонентів у вигляді функцій), динамічними мутаціями об'єктів у JavaScript та потребою в додаткових механізмах фільтрації для візуалізації великих і перевантажених діаграм. Попри ці обмеження, розроблений інструмент довів свою високу практичну придатність для автоматизованого документування, реінжинірингу та архітектурного аналізу програмного забезпечення.

Список використаних джерел

1. TypeScript Documentation. Microsoft, 2025. URL: https://www.typescriptlang.org/docs/
2. TypeScript Compiler API Documentation. Microsoft, 2025. URL: https://github.com/microsoft/TypeScript/wiki/Using-the-Compiler-API
3. OMG Unified Modeling Language Version 2.5.1. Object Management Group, 2017. URL: https://www.omg.org/spec/UML/

УДК 004.023

КОМПЛЕКСНИЙ ПІДХІД ДО ПРОЄКТУВАННЯ ВЕБОРІЄНТОВАНИХ СИСТЕМ: ВІД КОМЕРЦІЙНИХ ПЛАТФОРМ ДО АНАЛІТИЧНИХ МОНІТОРІВ

Шпінталь М.Я.[1)], Бобрик А.А.[2)], Липівський Д.М.[3)], Паланюк В.Т.[4)], Цінцірук Л.Р.[5)]
Західноукраїнський національний університет
[1)] к.т.н.,доцент; [2-5)] бакалавр

І. Постановка проблеми

Сучасний етап розвитку інформаційного суспільства характеризується тотальною цифровізацією бізнес-процесів. Розробка веборієнтованих систем сьогодні вимагає не лише автоматизації окремих функцій (продажів, бронювання чи управління проєктами), а й створення інтегрованих рішень, здатних обробляти великі масиви даних у реальному часі. Проте при проєктуванні таких систем розробники часто стикаються з проблемою архітектурної розрізненості. Виникає необхідність узагальнення методологічних підходів до створення систем, які поєднують у собі складну бізнес-логіку e-commerce, гнучкість Agile-інструментів та потужність аналітичних моніторів.

ІІ. Мета роботи

Метою роботи є систематизація методів проєктування та програмної реалізації веборієнтованих інформаційних систем різного цільового спрямування. Дослідження базується на синтезі рішень для електронної комерції (зоотовари), систем бронювання квитків, платформ управління ІТ-проєктами та модулів real-time аналітики.

ІІІ. Основні результати дослідження

У ході комплексного дослідження було детально проаналізовано та програмно реалізовано чотири стратегічні вектори розробки сучасного ПЗ, кожен з яких вирішує специфічні задачі цифровізації:

Платформи електронної комерції (Бобрик А.А.). Досліджено архітектуру спеціалізованих інтернет-магазинів, де пріоритетом є адаптація інтерфейсу під специфіку товару (зоотовари). Ключовим результатом стало проєктування оптимального «шляху клієнта» (Customer Journey), що включає багатокритеріальні фільтри пошуку, систему управління замовленнями та інтеграцію автоматизованих сервісів доставки. Це забезпечує високу конверсію за рахунок мінімізації зусиль користувача при виборі та купівлі товарів.

Сервіси онлайн-бронювання (Липівський Д.М.). Визначено, що критичним аспектом таких систем є точність управління динамічними ресурсами в реальному часі. Реалізовано функціонал вибору місць, автоматичного резервування та генерації електронних квитків. Особливу увагу приділено безпечній інтеграції зовнішніх платіжних шлюзів (Stripe/LiqPay), що дозволяє автоматизувати фінансові транзакції та знизити навантаження на операційний відділ.

Корпоративні системи управління проєктами (Паланюк В.Т.). Узагальнено досвід впровадження гнучких методологій розробки ПЗ. Програмно реалізовано інструменти для Agile-менеджменту: інтерактивні дошки Scrum та Kanban, механізми планування спринтів та управління беклогом. Система забезпечує прозорість командної взаємодії через розподіл ролей (Owner, Scrum Master, Developer) та візуалізацію прогресу виконання задач.

Системи real-time моніторингу та аналітики (Цінцірук О.Р.). Розроблено архітектуру системи для безперервного збору та обробки потокових даних. Використання технології WebSockets дозволило реалізувати миттєву візуалізацію технічних і бізнес-метрик на динамічних дашбордах без необхідності перезавантаження сторінок. Це дає змогу адміністраторам оперативно виявляти аномалії в роботі серверної інфраструктури та аналізувати поведінку користувачів у момент їх активності на ресурсі.

Інтеграція цих векторів дозволила сформувати концепцію універсальної веборієнтованої системи, яка поєднує в собі комерційну ефективність, високу швидкість обробки даних та гнучкість управління процесами. Текстовий аналіз та об’єктно-орієнтоване моделювання підтвердили, що використання єдиного технологічного стеку (Laravel + Vue.js) є оптимальним для масштабування кожного з перелічених напрямків.

Спільним для всіх цих систем є використання об’єктно-орієнтованого моделювання. Для опису функціональних вимог та взаємодії акторів із системою найефективнішим інструментом визначено UML-діаграму варіантів використання (Use Case Diagram).

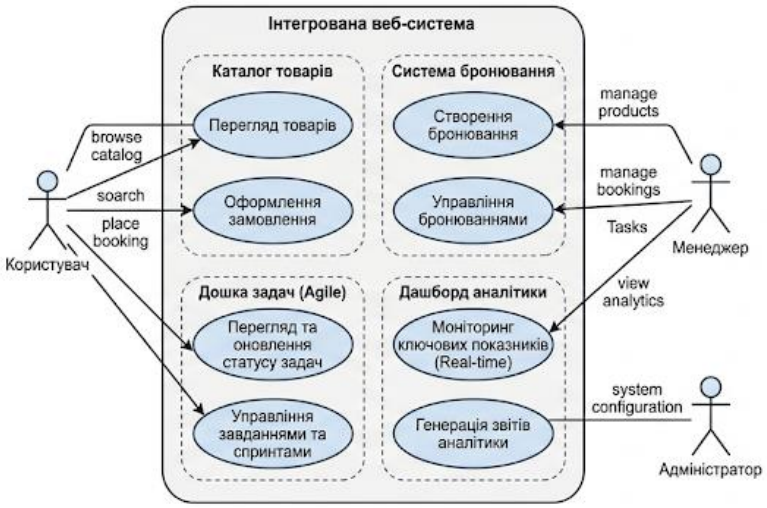

Рисунок 1 – UML-діаграма варіантів використання інтегрованої веб-системи

На діаграмі відображено взаємодію Користувача, Адміністратора та Менеджера з основними модулями: «Каталог товарів», «Система бронювання», «Дошка задач (Agile)» та «Дашборд аналітики»

Для внутрішньої логіки та структурування баз даних використано UML-діаграму класів (Class Diagram), яка описує зв’язки між основними сутностями: Користувачами, Проєктами, Замовленнями та Метриками.

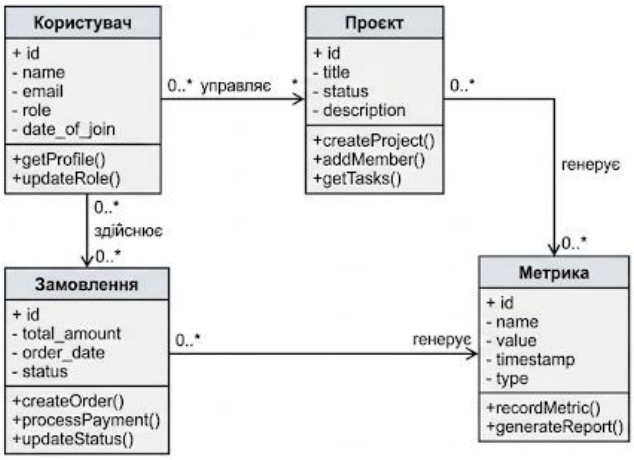

Рисунок 2 – UML-діаграма класів об’єктної моделі системи

Висновок

Практична реалізація розглянутих систем базується на сучасному стеку технологій: фреймворк Laravel (PHP) для серверної логіки, Vue.js (JavaScript) для створення динамічних інтерфейсів та СУБД MySQL для збереження даних. Таке поєднання дозволяє створювати системи з високою пропускною здатністю, що є критично важливим для аналітики в реальному часі та обробки пікових навантажень у комерційних сервісах.

Узагальнення досвіду розробки цих рішень дозволило сформувати єдиний підхід до побудови гнучких веб-систем, де аналітичний модуль стає «інтелектуальним ядром», що допомагає оперативно коригувати бізнес-стратегію на основі реальних даних про поведінку користувачів та стан серверної інфраструктури.

Список використаних джерел

1. Лодон К. С., Трейвер Ч. Г. Електронна комерція 2021: Бізнес, технології, суспільство. – Pearson, 2021. – 912 с.
2. Сазерленд Дж. Scrum. Навчись робити вдвічі більше за менший час. – Харків: КСД, 2016. – 288 с.
3. Laravel Documentation [Електронний ресурс]. – Режим доступу: https://laravel.com/docs

УДК 004.89:681.5:654.924

КОМП'ЮТЕРНО-ІНТЕГРОВАНА СИСТЕМА БЕЗПЕКИ ПРИВАТНОГО БУДИНКУ

Когут М.С.
Західноукраїнський національний університет
бакалавр

І. Постановка проблеми

Ефективне гарантування безпеки приватного сектору сьогодні неможливе без впровадження високотехнологічних охоронних комплексів. Такі системи [1] являють собою синергію апаратних модулів та програмного забезпечення, що спрямовані на запобігання крадіжкам, виявлення спроб проникнення та мінімізацію ризиків інших надзвичайних подій. Сучасна індустрія безпеки пропонує широку варіативність рішень: від закритих пропрієтарних екосистем відомих брендів до відкритих розробок на базі програмованих мікроконтролерів. Пропрієтарні комплекси, попри їхню високу експлуатаційну стабільність, створюють суттєвий фінансовий тиск на бюджет та обмежують користувача жорсткими рамками закритої архітектури. Водночас розробки на базі промислових мікроконтролерів та спеціалізованих мережевих модулів забезпечують максимальну адаптивність до специфічних запитів користувача при помірній собівартості. Це створює передумови для розробки автономних гібридних систем, які здатні інтегруватися в загальну мережу «розумного будинку» та забезпечувати локальне управління без залежності від зовнішніх хмарних серверів.

ІІ. Мета роботи

Метою розробки є проєктування комп'ютерно-інтегрованої системи керування безпекою заміського будинку дивись рисунок 1, що поєднує функції класичної охоронної сигналізації з елементами технології «розумний дім», а також обґрунтований вибір відповідних технічних засобів автоматизації (ТЗА).

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- Провести аналіз існуючих систем безпеки, орієнтованих на охорону приватних будинків, та дослідити технології «розумного будинку».
- Визначити функціональні та нефункціональні вимоги до архітектури системи безпеки.
- Розробити структурну схему комп'ютерно-інтегрованої системи безпеки заміського будинку.
- Здійснити аналітичний підбір та специфікацію технічних засобів автоматизації, мережевих контролерів та сенсорної периферії.
- Виконати математичний розрахунок параметрів системи автоматичного регулювання (ПІД- та ПІ2Д-регуляторів).
- Провести комп'ютерне моделювання для оцінки стійкості та показників якості регулювання розробленої системи.

ІІІ. Основна частина

У даній роботі було досліджено процес створення комп'ютерно-інтегрованої системи безпеки заміського будинку, побудованої на базі сучасних бездротових технологій Інтернету речей (IoT) та інноваційного обладнання смарт-безпеки. Розглянуто теоретичні аспекти побудови розподілених бездротових охоронних мереж, проведено аналіз ринку та визначено ключові вимоги до інтелектуального центрального керуючого модуля (хаба) та автономного периферійного обладнання.

Об'єктом управління є процес функціонування автоматизованої системи безпеки заміського будинку. Цей процес включає в себе такі ключові аспекти:

– Моніторинг периметра та внутрішнього простору: Включає фіксацію спроб подолання вікон або дверей за допомогою бездротових магнітоконтактних сповіщувачів із вбудованими акселерометрами (наприклад, Ajax DoorProtect Plus) та комбінованих оптико-електронних і акустичних сенсорів (Ajax CombiProtect), які безпомилково реагують на рух та специфічний звук розбиття скла. Окремо контролюється фізичний злом конструкцій (стін, сейфів, дверей) за рахунок аналізу вібрацій та зміни кута нахилу, а також забезпечується миттєва фотоверифікація тривог (Ajax MotionCam).

– Управління централізованою бездротовою екосистемою: Організація збору зашифрованих даних з усіх сенсорів через інтелектуальну централь (наприклад, Ajax Hub 2 Plus) за допомогою захищених радіопротоколів (Jeweller та Wings), що повністю замінили застарілі дротові лінії та

інтерфейс RS-485. Програмне забезпечення хаба та хмарна інфраструктура (Ajax Cloud) дозволяють логічно групувати датчики по віртуальних кімнатах і зонах, забезпечуючи гнучке налаштування політик безпеки.

– Інформування та оповіщення: Прийом цифрових пакетів даних від датчиків, їх мікропроцесорна обробка та багатоканальне відправлення тривожних сповіщень (Push-повідомлення, SMS, дзвінок) користувачу або безпосередньо на пульт централізованого спостереження. Цей процес реалізується завдяки вбудованим у централь високошвидкісним модулям зв'язку (Ethernet, Wi-Fi, 4G/LTE), що усуває потребу в додаткових зовнішніх телефонних комунікаторах чи дискретних GSM-модулях.

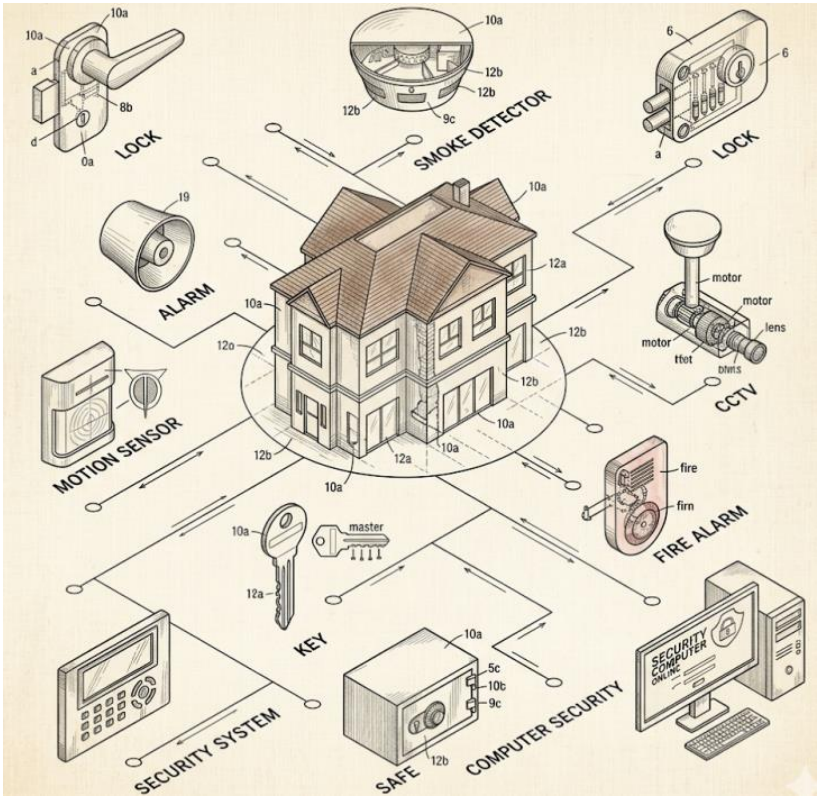

Рисунок 1 - Архітектура системи охорони для приватного будинку

Особливу увагу в роботі було приділено розробці та моделюванню контурів системи автоматичного регулювання. Застосування класичного ПІД-регулятора для об'єктів вищого порядку часто вимагає компромісу між швидкістю та стабільністю. Тому було розраховано параметри для модифікованого ПІ2Д-регулятора, утвореного додаванням каналу подвійного диференціювання. Для усунення проблеми підсилення високочастотних перешкод диференціальну складову було перенесено в канал зворотного зв'язку із застосуванням низькочастотного фільтра.

Висновок

Структурні схеми систем автоматичного регулювання дозволяють наочно представити проходження сигналів, що необхідно для моделювання поведінки системи. Розрахунок настроювальних параметрів здійснювався модальним методом синтезу. Результати комп'ютерного моделювання у середовищі MatLab продемонстрували, що динамічний процес у системі з ПІ2Д-регулятором ідеально збігається із бажаною перехідною характеристикою, тоді як класичний ПІД-регулятор дає перерегулювання. Крім того, удосконалена система довела високу інваріантність до зовнішніх збурень M(t) та нестаціонарності параметрів об'єкта управління в межах ±30%.

Реалізація комп’ютерно-інтегрованої системи безпеки згідно з розробленою структурою дозволить створити надійну, масштабовану та стійку до перешкод платформу для захисту заміського будинку, що сприятиме задоволенню потреб користувачів у безпеці. .

Список використаних джерел

1. Магауенов Р.Г. Системи охоронної сигналізації: основи теорії і принципи побудови. Навчальний посібник для вузів. 2-вид. М.: Гаряча лінія – Телеком. 2008. 496 с.
2. Лупенко С.А., Тиш Є.В. Прикладна теорія цифрових автоматів. Навчальний посібник. Тернопіль: ТНТУ ім. І. Пулюя. ДСТУ Б А.2.4-16:2008. Автоматизація технологічних процесів. Умовні графічні зображення приладів і засобів автоматизації в схемах. – К.: Мінрегіонбуд України, 2009.

УДК 681.3

РОЗРОБКА ВЕБ-СИСТЕМИ УПРАВЛІННЯ ПРОЄКТАМИ ДЛЯ КОМАНДНОЇ РОБОТИ

Співак І.Я.[1)], Лабик Б.В.[2)], Крепич С.Я.[3)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)]бакалавр; [3)]к.т.н., доцент

І. Постановка проблеми

У сучасних умовах розвитку ІТ-індустрії ефективне управління проєктами є критичним фактором успіху будь-якої розробки. Зростання складності програмних продуктів, необхідність координації територіально розподілених команд та стислі терміни виконання (deadlines) вимагають застосування спеціалізованих інструментів для структурування робочих процесів.

Таким чином, актуальною стає задача розробки легкої, швидкодіючої інформаційної системи на базі SPA-архітектури, яка б поєднувала в собі інтуїтивно зрозумілий інтерфейс, гнучкий інструментарій управління задачами та ефективні механізми синхронізації даних у реальному часі.

ІІ. Мета роботи

Метою дослідження є проєктування та програмна реалізація вебсистеми «TeamTrack», призначеної для автоматизації процесів управління ІТ-проєктами, координації роботи команд розробників та підвищення ефективності виконання задач у межах сучасних методологій управління проєктами.

Для досягнення поставленої мети необхідно вирішити такі основні завдання, як:

1) провести аналіз сучасних систем управління проєктами та визначити їх функціональні особливості;
2) спроєктувати архітектуру клієнт-серверної системи;
3) реалізувати модулі керування проєктами, задачами та користувачами;
4) забезпечити механізми авторизації та захисту даних;
5) реалізувати інтерактивний користувацький інтерфейс із підтримкою динамічного оновлення інформації;
6) виконати оцінювання продуктивності та швидкодії системи.

ІІІ. Особливості програмної реалізації системи «TeamTrack»

Програмна система «TeamTrack» реалізована на основі сучасної триланкової архітектури, яка включає клієнтський рівень (Frontend), серверний рівень (Backend) та рівень збереження даних (Database). Такий підхід забезпечує модульність, масштабованість та зручність подальшого супроводу програмного продукту.

В основі клієнтської частини використано бібліотеку React, що дозволяє реалізувати SPA-архітектуру (SinglePageApplication) із динамічним оновленням контенту без повного перезавантаження сторінки. Це суттєво покращує швидкість взаємодії користувача із системою та забезпечує високий рівень адаптивності інтерфейсу.

Для оптимізації управління станом застосунку використано бібліотеку Zustand, яка забезпечує швидке оновлення даних та мінімізує навантаження на клієнтську частину. Перевагою такого підходу є спрощення логіки синхронізації між компонентами інтерфейсу та підвищення продуктивності системи при роботі з великою кількістю задач.

Візуальне оформлення реалізоване за допомогою фреймворку Tailwind CSS, що забезпечує адаптивність вебінтерфейсу для різних типів пристроїв та дозволяє пришвидшити процес розробки інтерфейсних компонентів.

Серверна частина побудована на базі фреймворку NestJS, який підтримує модульний підхід до розробки та забезпечує високу надійність і структурованість програмного коду. Для організації взаємодії між клієнтською та серверною частинами використано REST API. Передача даних здійснюється у форматі JSON, що забезпечує універсальність та сумісність між різними компонентами системи.

Для забезпечення захисту користувацьких даних реалізовано механізм автентифікації та авторизації із застосуванням JWT-токенів (JSON WebTokens). Це дозволяє безпечно керувати сесіями користувачів та обмежувати доступ до функціональних можливостей системи відповідно до ролей.

Функціонал системи включає механізм Drag-and-Drop для візуального переміщення завдань, систему планування спринтів з дедлайнами та журнал активності, який дозволяє відстежувати хронологію змін у кожному проєкті.

База даних системи реалізована на основі СУБД MySQL, що забезпечує надійне збереження інформації про користувачів, проєкти, задачі та історію змін. Структура бази даних оптимізована для швидкого виконання операцій читання та запису.

Функціональні можливості системи включаютьстворення та адміністрування проєктів, управління задачами та підзадачами, механізм Drag-and-Drop для зміни статусу задач, планування спринтів і дедлайнів, ведення журналу активності користувачів, контроль виконання завдань у режимі реального часу та систему ролей та розмежування прав доступу.

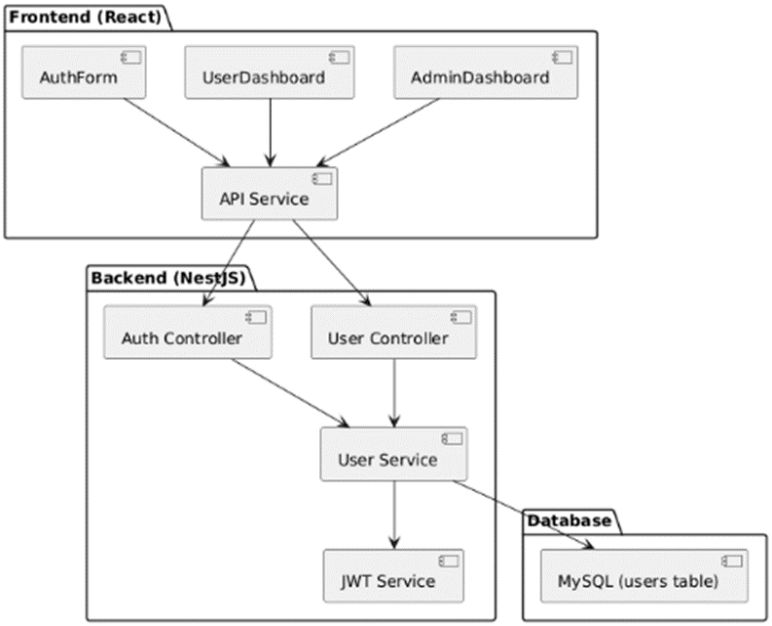

Рисунок 1 – Логічна структура програмних модулів системи

З метою підтвердження ефективності розробленої системи було проведено порівняльний аналіз швидкодії основних операцій (див. табл.1).

Таблиця 1

Характеристики продуктивності системи «TeamTrack»

Параметр	Значення	Примітка
Час завантаження SPA	< 1.5 с	При швидкості з'єднання 10 Мбіт/с
Час відгуку API (середній)	150-200 мс	Операції читання/запису
Час оновлення стану дошки	< 50 мс	Використання Virtual DOM та Zustand
Макс. кількість задач у проєкті	5000+	Без деградації продуктивності клієнта

Висновок

У результаті виконаної роботи розроблено вебсистему «TeamTrack», призначену для автоматизації процесів управління IT-проєктами та організації ефективної командної взаємодії. Реалізована система забезпечує створення та адміністрування проєктів, керування задачами, планування спринтів, контроль дедлайнів і ведення журналу активності користувачів.

Використання сучасного технологічного стеку React, NestJS, MySQL та TypeScript дозволило створити масштабовану клієнт-серверну систему з високою швидкодією та адаптивним інтерфейсом. Завдяки застосуванню SPA-архітектури, менеджера стану Zustand та REST API забезпечено швидке оновлення даних і комфортну взаємодію користувачів із системою в реальному часі.

Список використаних джерел

1. ReactDocumentation. Офіційна документація бібліотеки React. [Електронний ресурс] – Режим доступу: react.dev.
2. I.Spivak, S.Krepych, R.Krepych, “Construction of the criterion for the agree of expert groups estimates based on analysis of interval data”, International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). 2018. – pp.261-264
3. S.Krepych, I.Spivak, S.Spivak, “Approach to forecasting of utility costs using neural networks”, 15th International conference on Computer Sciences and Information Technologies (CSIT). 2020. – pp.387-391
4. ZustandDocumentation. Управління станом у React застосунках. [Електронний ресурс] – Режим доступу: pmnd.rs

УДК 004.056.55+004.738.5

АВТОМАТИЗОВАНА ЛЕГКОВАГОВА АГРЕГАЦІЯ КРИПТОПОТОКІВ IoT ЗА NIST SP 800-232 ТА ACE/OSCORE/EDHOC

Пащак С.А.[1)], Возна Н.Я.[2)]
Західноукраїнський національний університет
[1)] аспірант; [2)] д.т.н., професор

I. Постановка проблеми

Згідно з інформацією IoT Analytics у звіті «Стан IoT 2025», наприкінці 2024 року IoT-пристрої налічували 18,5 мільярда одиниць. До 21,1 мільярда мала би збільшитись кількість пристроїв на кінець 2025 року, що становитиме 114% від попереднього року. Серед типових кінцевих точок виокремимо RFID-мітки, медичні імпланти, сенсори промислової автоматизації та LPWAN-сенсори. Для них порушення безпеки телеметрії є деструктивним; в умовах обмежень на пам'ять, енергозбереження та пропускну здатність каналу ризик зростає. Стандартні AEAD-примітиви, як-от AES-GCM, і хеш-функції типу SHA-2 поки що надійні. Але потенціалу в теперішніх ROM/RAM буває недостатньо для систем, у яких використовуються 8- та 16-розрядні мікроконтролери та шлюзи. Дослідження Renner S., Pozzobon E., Mottok J. (AES-GCM, LNCS 13745, 2022) це підтверджує: AES-GCM на ESP32() виконано за 16 596 мікросекунд, Ascon – за 1 349 мікросекунд, тобто Ascon майже у 12 разів швидший, в однаковому програмному режимі.

У 2013 році Національний інститут стандартів і технологій (NIST) розпочав дослідження криптографії у середовищах з обмеженою обчислювальною потужністю; у 2018 році проведено відкритий конкурс, переможцем якого в лютому 2023 року визначено сімейство Ascon. Зрештою було опубліковано NIST SP 800-232 під назвою «Стандарти легковагової криптографії на основі Ascon для обмежених пристроїв», де визначено сімейство Ascon-AEAD128, Ascon-Hash256, Ascon-XOF128 та Ascon-CXOF128 як рекомендоване для пристроїв з обмеженими обчислювальними ресурсами. Паралельно IETF розробив протокол прикладної безпеки для середовища CoAP. Він містить RFC 8613 для захисту payload на рівні додатка; RFC 9528 для створення спільного секретного ключа Діффі-Гелмана; RFC 9200 для процесів, пов'язаних із авторизацією; та RFC 9594 для розподілу ключів у груповій комунікації через KDC.

Комбінація NIST SP 800-232 з RFC-стеком ACE/OSCORE/EDHOC дозволяє теоретично створити автоматизовану систему легковагової агрегації зашифрованого потоку даних. У цій схемі кожен сенсор-видавець видає блоки під захистом AEAD, а центральний KDC керує циркуляцією групових ключів. У науковій літературі не виявлено досліджень, які поєднують сервіс OPC UA PubSub Security Key Service (SKS, OPC 10000-14) із KDC ACE Group OSCORE (RFC 9594). Це становить наукову новизну дослідження. Фундаментом дослідження є дані про пристрої з розвідок стосовно розподілених комп'ютерних систем.

II. Мета та постановка задачі

Маючи набір IoT-сенсорів $N = \{s_1, \ldots, s_n\}$, кожен сенсор генерує зашифрований потік даних $d_i(t)$. На шлюзі (Aggregator) з'єднуємо потоки в $D(t) = (c_1(t), c_2(t), \ldots, c_n(t))$, де $c_i(t) = \text{Ascon-AEAD128.Enc}(K_i, N_i(t), d_i(t), AD_i)$. Підбираємо автоматизовану схему {E, KDC, Rekey} таку, що: (1) на вузлах-видавцях шифрування E реалізує Ascon-AEAD128 (NIST SP 800-232); (2) оркестрація ключів KDC відповідає RFC 9594 із транспортом OSCORE (RFC 8613) поверх CoAP; (3) генерація довгоживучих ключів через EDHOC (RFC 9528); (4) забезпечуються секретність у минуле/майбутнє при виході/вході учасників групи; (5) додаткові витрати $O(D, n) \rightarrow \min$ при дотриманні ліміту даних на ключ за NIST SP 800-232.

За потреби понад 100 потоків ($n>100$) чотири фактори обмежують Ascon-AEAD128: (a) Узгодження nonce – стандарт вимагає унікальності пари (K, N) в межах одного ключа; при використанні сенсорами одного ключа nonce ділиться між ними, що навантажує KDC. (b) Ascon-AEAD128 не має гомоморфної властивості та не має вбудованого оператора збирання і тому залежний від nonce; через це потрібно або повторно шифрувати на шлюзі (що може порушити цілісність), або мати груповий ключ. (c) Обмеження паралельного шифрування при спільному ключі – режим nonce-masking (NIST SP 800-232, опція з 256-бітним ключем $K = K_1 \parallel K_2$) частково знімає колізії nonce у багатоключових випадках, але збільшує внутрішній стан (320 біт в Ascon) на пристрої.

(d) При ротації виникають додаткові витрати на менеджмент ключів: досягнення ліміту 2^{54} байт/ключ або зміна складу групи спричиняють зміну ключа, вартість якої визначається топологією KDC.

III. Виклад основного матеріалу

Структуризація багатофункціональних потоків IoT-даних як основа агрегації

У класифікації [4] формалізовано поділ на чотири види потоків у розподіленій комп'ютеризованій системі: моніторингові, інтерактивні та діалогові, з відповідною інформаційною парністю компонентів. У контексті IoT-агрегації моніторингові потоки (давачі температури, тиску, струму) у більшості напрямлені в один бік від видавця до споживача або KDC; інтерактивні потоки діють за парним принципом «запит–відповідь»; діалогові потоки (адміністрування групи, приєднання/вихід) ініціюються адміністратором KDC. Розділення дозволяє зменшити обчислювальні витрати для кожного типу потоків та зменшити ентропійну ємність агрегованого блоку D(t), що ґрунтується на теоретико-числовій моделі кодування у розширених полях Галуа[3].

NIST SP 800-232: Ascon-AEAD128 та структурні обмеження для агрегації понад 100 потоків

Ascon-AEAD128 використовує 320-бітний внутрішній стан із SPN-перестановкою Ascon-p для ініціалізації і фіналізації та Ascon-p для обробки даних. Він забезпечує 128-бітну стійкість конфіденційності і цілісності у nonce-respecting сценарії. За NIST SP 800-232 ліміт даних на ключ становить:

$$D_{\max} \le 2^{54}\ {}^{байт}\!/_{ключ} \tag{1}$$

Вищезазначені обмеження (a)–(d) стають відчутними, коли n>100: простір, придатний для безконфліктної генерації nonce, стає базою, яка визначає ширину групи при точно визначеному розподілі nonce між учасниками, а час до досягнення ліміту (1) зворотно пропорційний n.

Таблиця 1

Порівняння легковагових AEAD-схем (фрагмент)

AEAD-схема	Стандарт	ESP32, мкс	ROM, кат.	RAM, кат.	Ліміт даних/ключ
Ascon-AEAD128	NIST SP 800-232 (13.08.2025)	1349	малий	малий	2^{54} байт
ДСТУ 7624 «Калина»	ДСТУ 7624:2014	-**	середній	середній	залежить від режиму
AES-GCM	NIST SP 800-38D	16596	середній	середній	2^{39} байт
Xoodyak	NIST LWC фіналіст	4677	малий	малий	див. специфікацію
GIFT-COFB	NIST LWC фіналіст	10471	малий	малий	див. специфікацію

** Час шифрування на ESP32 за: Renner S., Pozzobon E., Mottok J. «The Final Round: Benchmarking NIST LWC Ciphers on Microcontrollers»*
*** Даних для ДСТУ 7624 у відкритих публікаціях бенчмарків на ESP32 не виявлено.*

Автоматизована оркестрація ключів через стек RFC

Базові елементи поданої схеми: EDHOC – встановлює спільний OSCORE-контекст через обмін Діффі-Гелмана над COSE; OSCORE – використовує COSE/CBOR для end-to-end захисту CoAP-payload; ACE-OAuth – ґрунтована на OAuth 2.0 структура авторизації, адаптована під ресурсообмежене середовище, із CWT PoP-токенами; ACE Group Key Provisioning – розподіляє головний матеріал групи через KDC-інтерфейс; draft-ietf-ace-key-groupcomm-oscore-21 – профіль Group OSCORE для RFC 9594; draft-ietf-core-oscore-key-update-13 (KUDOS) – оновлює OSCORE-контекст між двома учасниками, уникаючи EDHOC, із можливістю дотримання forward secrecy.

Послідовність реалізації: (1) EDHOC між сенсором і AS створює спільний секрет; (2) AS видає PoP-токен; (3) сенсор пред'являє токен KDC і отримує груповий OSCORE-контекст; (4) Group OSCORE публікує дані; (5) при приєднанні або виході KDC виконує rekey за RFC 9594; (6) пари peer-to-peer оновлюють ключі через KUDOS.

Розриви у поєднанні між RFC-стеком ACE/OSCORE та IEC 62541-14:2026

Стандарт OPC UA PubSub (OPC 10000-14, IEC 62541-14) містить опис Security Key Service (SKS), у якому є функції GetSecurityKeys і SetSecurityKeys для SecurityGroup з параметрами KeyLifetime, MaxFutureKeyCount, MaxPastKeyCount. На ACE Group OSCORE KDC (RFC 9594) покладена майже така сама роль, але працює він інакше. Зауважимо п'ять розривів у поєднанні:

Транспортний розрив – SKS працює через протокол OPC UA Secure Channel (TCP, TLS або UA-сесія), а KDC – через стек CoAP/OSCORE (UDP). Можна спробувати перевести CoAP ↔ OPC UA, але це унеможливлює end-to-end з'єднання.

Розрив у rekey – робота SKS базується на часовій моделі, а KDC – на подієво-орієнтованій. SKS використовує KeyLifetime із масивом майбутніх ключів MaxFutureKeyCount, а KDC при приєднанні/виході або компрометації повторно ініціює rekey.

Аутентифікаційний розрив – SKS використовує OPC UA UserIdentityToken та X.509-сертифікати з відповідністю Roles ↔ SecurityGroups (OPC 10000-3), а KDC – OAuth 2.0 access tokens із proof-of-possession (PoP) у форматі CWT/COSE.

Криптографічний розрив – стандарт OPC UA не включає Ascon-AEAD128 для SecurityPolicyUri SKS; здебільшого профілі безпеки подані для AES-CBC/SHA-256 та AES-GCM/SHA-256. За переходу SKS↔KDC шлюз змушений здійснювати повторне шифрування.

Розрив у моделях груп – у OPC UA використовується SecurityGroupId з PubSubGroup (WriterGroup/ReaderGroup), а Group OSCORE поєднує групу через OSCORE Group Identifier. Відповідно, моделі груп різні, і це виключає тотожне зіставлення 1 до 1.

Теоретико-інформаційна модель ентропії агрегованого шифропотоку

Беручи до уваги математичний апарат [3], ентропія Шеннона агрегованого потоку $D(t)$ із символьним розподілом p_i визначається як:

$$H(X) = -\sum p_i \log_2 p_i \tag{2}$$

Ascon-AEAD128 для коректного шифрування потоку вимагає $H(X) \to \log_2(|\sum|)$: наближення до максимальної непередбачуваності символів алфавіту (ентропії алфавіту) є ознакою високої якості шифру та запобігає використанню гомоморфної агрегації без знання ключа. У Ascon-AEAD128 в багатоключовому випадку обмежується кількість пар $(N.M)$:

$$N \cdot M \le 2^{128} \tag{3}$$

де N – число повідомлень на ключ, M – число ключів; ця нерівність є вираженням резерву безпеки для невідповідного налаштування за [1, §2]. Оцінка складності ініціації rekey для LKH-схемою при $2\log_2 n + O(1)$ повідомлень та для основної P2P-схеми при $n-1$ повідомленнях:

$$O_{KDC}(n) = c_1 \cdot n + c_2 \cdot \log_2 n + c_3 \tag{4}$$

де $c_1 \cdot n$ відображає PoP-токенізовану розсилку та нотифікації для кожного учасника згідно з RFC 9594; $c_2 \cdot \log_2 n$ – застосування Logical Key Hierarchy (LKH) за схемою RFC 2627; c_3 – фіксовані витрати на генерацію групового контексту. Підбір коефіцієнтів c_1, c_2, c_3 для ESP32, STM32F7, nRF52 є предметом подальших досліджень.

Висновки

У дослідженні Renner S., Pozzobon E., Mottok J. показано, що в однаковому програмному режимі Ascon-AEAD128 у 12 разів швидший за AES-GCM на ESP32. Це є вагомим аргументом для розробки апаратних криптоакселераторів для AEAD128, і подальше використання їх у парі як основного примітиву для агрегації потоків в IoT. Застосування LKH (RFC 2627) над стеком RFC ACE/OSCORE/EDHOC забезпечує складність виконання rekey O(log n) порівняно з лінійним варіантом, який дає складність O(n); це відчутно, якщо кількість учасників n > 100. У спробі поєднати OPC UA PubSub SKS (OPC 10000-14) та ACE Group OSCORE KDC (RFC 9594) виявлено п’ять розривів у поєднанні: транспортний, у rekey, аутентифікаційний, криптографічний та у моделях груп; це створює фундамент для подальшої роботи з поєднання стандартів. Модель (4) дає можливість оцінити складність оркестрації груп розміром від десятка до тисячі сенсорів за умови можливості емпіричного підбору відповідних коефіцієнтів.

Список використаних джерел

1. Sönmez Turan M., McKay K., Chang D., Kang J., Kelsey J. Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions. NIST SP 800-232. Gaithersburg, MD : NIST, 2025. 65 p. DOI: 10.6028/NIST.SP.800-232.

2. Selander G., Preuß Mattsson J., Palombini F. Ephemeral Diffie-Hellman Over COSE (EDHOC). RFC 9528. IETF, March 2024. 82 p. DOI: 10.17487/RFC9528.

3. Шаряк В. В., Возна Н. Я., Николайчук Я. М. Кодування даних та організація розподілених баз даних у розширених полях Галуа : монографія. Тернопіль : ЗУНУ, 2023. 267 с.

4. Пітух І. Р., Возна Н. Я. Способи організації руху моніторингових, інтерактивних і діалогових даних у структурах розподілених комп’ютерних систем. Науковий вісник НЛТУ України. 2021. Т. 31, № 3. С. 101–108. DOI: 10.36930/40310316.

5. Palombini F., Tiloca M. Key Provisioning for Group Communication Using Authentication and Authorization for Constrained Environments (ACE). RFC 9594. IETF, September 2024. 97 p. DOI: 10.17487/RFC9594.

УДК 004.4

ПРОГРАМНА СИСТЕМА ДЛЯ ТУРИСТИЧНОЇ ПЛАТФОРМИ НА ОСНОВІ МІКРОСЕРВІСНОЇ АРХІТЕКТУРИ

Дудар А.О.[1)], Стасів І.С.[2)]
Західноукраїнський національний університет
[1)]магістрант; [2)]к.т.н., доцент

І. Постановка проблеми

Сучасні туристичні платформи повинні забезпечувати швидкий доступ до інформації про локації, маршрути й рекомендації, а також обробляти значні обсяги користувацьких даних. Зі зростанням навантаження підвищуються вимоги до масштабованості, відмовостійкості та обробки подій у режимі, близькому до реального часу.

Монолітний підхід у таких умовах ускладнює підтримку, масштабування та незалежне оновлення окремих компонентів. Тому для туристичної платформи доцільно використовувати мікросервісну архітектуру, яка дозволяє розділити систему на незалежні сервіси для роботи з користувачами, локаціями, маршрутами, рекомендаціями та подіями.

Додатково важливою є подієво-орієнтована взаємодія, за якої дії користувача передаються як події та обробляються асинхронно. Це зменшує зв'язаність між сервісами, підвищує гнучкість системи та дозволяє оновлювати рекомендації без блокування основних запитів користувача.

ІІ. Мета роботи

Метою роботи є розробка програмної системи туристичної платформи на основі мікросервісної архітектури з використанням подієво-орієнтованого підходу. Система повинна забезпечувати пошук туристичних локацій, формування маршрутів, обробку дій користувача та генерацію персоналізованих рекомендацій. Особлива увага приділяється побудові масштабованої архітектури, у якій окремі сервіси можуть взаємодіяти між собою через REST API та брокер повідомлень Apache Kafka.

ІІІ. Особливості програмної реалізації туристичної платформи

Розроблена програмна система реалізована як туристична платформа, що об'єднує клієнтський застосунок, API Gateway, набір мікросервісів, брокер повідомлень і базу даних. Клієнтська частина забезпечує взаємодію користувача із системою, відображення туристичних локацій, маршрутів і рекомендацій. API Gateway виконує роль єдиної точки входу для запитів користувача та перенаправляє їх до відповідних сервісів.

Основна бізнес-логіка системи розподілена між окремими мікросервісами. User Service відповідає за управління користувачами, їх профілями та діями в системі. Location Service забезпечує зберігання, пошук і фільтрацію туристичних локацій. Route Service реалізує формування маршрутів між туристичними об'єктами з урахуванням параметрів маршруту. Recommendation Service відповідає за генерацію персоналізованих рекомендацій на основі аналізу подій користувача. Така декомпозиція дозволяє спростити розробку, тестування та супровід системи, а також масштабувати лише ті компоненти, які мають найбільше навантаження.

Взаємодія між сервісами реалізується комбінованим способом. Для синхронних запитів, які потребують швидкої відповіді користувачу, використовується REST API. Наприклад, отримання списку локацій або побудова маршруту може виконуватися через прямий запит до відповідного сервісу. Для обробки дій користувача застосовується асинхронна взаємодія через Apache Kafka. Події, що виникають у процесі роботи користувача з платформою, передаються до брокера повідомлень і надалі обробляються recommendation-сервісом.

Подієво-орієнтований підхід дозволяє відокремити процес генерації подій від процесу їх обробки. Наприклад, після перегляду туристичної локації або створення маршруту система може одразу продовжити обслуговування користувача, а оновлення рекомендацій виконати асинхронно. Це зменшує навантаження на основні сервіси, підвищує швидкість відповіді системи та забезпечує можливість обробки великої кількості подій. Крім того, використання Kafka дозволяє зберігати послідовність подій і масштабувати обробники залежно від інтенсивності потоку повідомлень.

Для підвищення надійності системи застосовуються механізми відмовостійкості. Зокрема, Circuit Breaker дозволяє запобігати повторним зверненням до сервісу, який тимчасово недоступний,

Retry забезпечує повторну спробу виконання запиту, а Timeout обмежує час очікування відповіді. Використання таких механізмів мінімізує вплив збоїв окремих сервісів на роботу всієї платформи та дозволяє підтримувати стабільність системи навіть за умов часткової недоступності окремих компонентів.

Важливою складовою реалізації є контейнеризація сервісів. Кожен мікросервіс може розгортатися в окремому Docker-контейнері, що забезпечує ізоляцію середовища виконання, стабільність роботи та спрощення процесу розгортання. Контейнери містять програмний код, конфігурації, залежності та необхідне середовище виконання. Такий підхід полегшує перенесення системи між різними середовищами, зменшує ризик конфліктів залежностей і спрощує масштабування окремих компонентів.

Оркестрація контейнерів використовується для автоматизованого запуску, зупинки, масштабування та моніторингу сервісів. Це дозволяє забезпечити узгоджену взаємодію компонентів системи та підвищити її відмовостійкість. У разі збільшення навантаження окремі сервіси можуть масштабуватися незалежно, без необхідності змінювати або перевантажувати всю систему. Такий підхід є особливо корисним для туристичної платформи, у якій навантаження на сервіси рекомендацій, пошуку або маршрутів може змінюватися залежно від активності користувачів.

На рисунку 1 представлено загальну структуру взаємодії компонентів туристичної платформи.

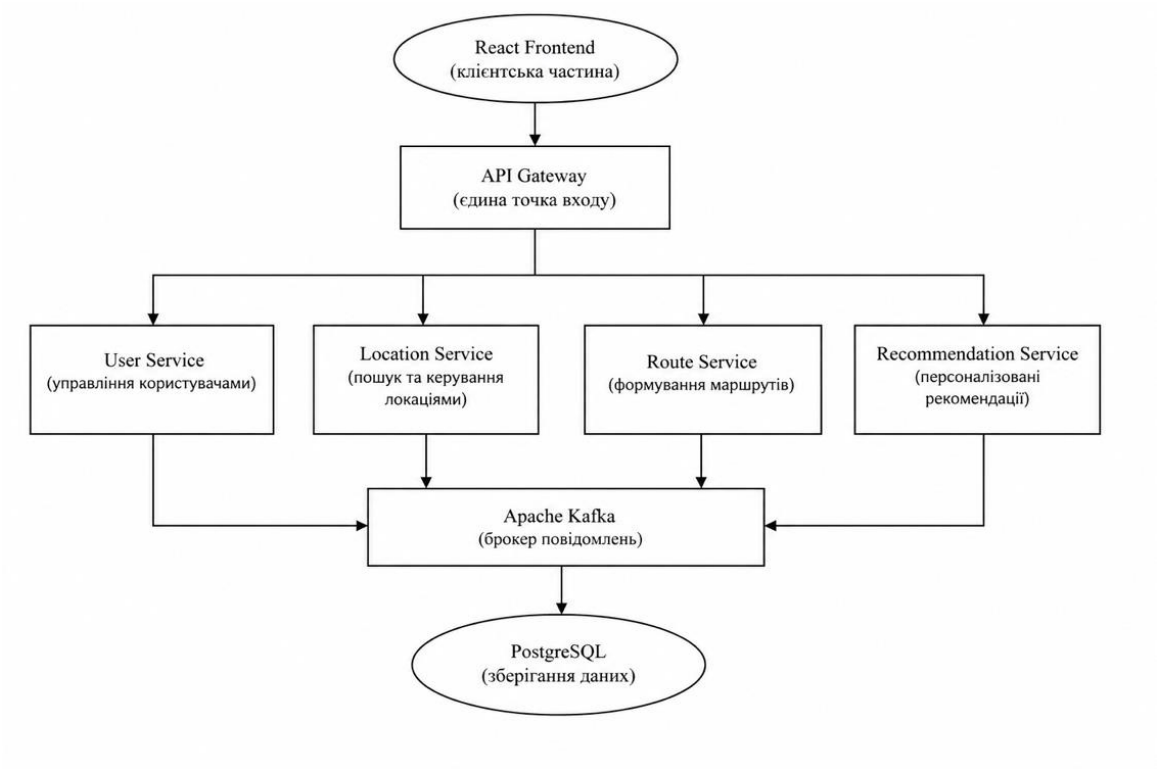

Рисунок 1 – Архітектура системи

Висновок

У роботі розглянуто особливості програмної реалізації туристичної платформи на основі мікросервісної архітектури. Запропонований підхід дозволяє розділити функціональність системи на незалежні сервіси, забезпечити гнучку взаємодію між компонентами та спростити подальше розширення платформи. Використання Apache Kafka забезпечує асинхронну обробку подій користувача та створює основу для формування персоналізованих рекомендацій у режимі, близькому до реального часу.

Список використаних джерел

1. Spring Boot Documentation [Електронний ресурс]. – Режим доступу: https://spring.io/projects/spring-boot Apache Kafka Documentation [Електронний ресурс]. – Режим доступу: https://kafka.apache.org/documentation/
2. Docker Documentation [Електронний ресурс]. – Режим доступу: https://docs.docker.com/
3. Microservices Architecture Guide [Електронний ресурс]. – Режим доступу: https://microservices.io/
4. Kubernetes Documentation [Електронний ресурс]. – Режим доступу: https://kubernetes.io/docs/

УДК 004.41

ГІБРИДНИЙ МЕТОД ARIMA-LSTM З МЕХАНІЗМОМ АДАПТИВНОГО ПЕРЕМИКАННЯ ВАГ КОМПОНЕНТ

Войтюк І.Ф.[1)], Коваль М.В.[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] магістрант

І. Постановка проблеми

Фондовий ринок є складною динамічною системою, у якій ціни акцій формуються під впливом численних різнорідних факторів – від макроекономічних показників до поведінки окремих учасників ринку. Сучасні платформи моніторингу біржових даних надають широкий спектр інструментів технічного аналізу, проте лише обмежена їх частина забезпечує прогнозування цінової динаміки засобами машинного навчання, і практично жодна не реалізує адаптивних підходів з урахуванням поточного режиму ринку. гКласична статистична модель ARIMA ефективно описує лінійну складову часового ряду, проте не здатна моделювати його нелінійну стохастичну компоненту. Рекурентні нейронні мережі типу LSTM успішно виявляють нелінійні залежності, однак потребують значних обсягів навчальних даних та схильні до перенавчання на коротких фінансових рядах. Гібридні моделі типу ARIMA-LSTM поєднують переваги обох підходів, проте традиційно використовують фіксовані ваги компонент, що знижує їх ефективність у різних режимах ринкової динаміки – трендовому, боковому та режимі підвищеної волатильності. Усунення цього недоліку шляхом адаптивного перемикання ваг визначає актуальність дослідження.

ІІ. Мета роботи

Метою роботи є розробка гібридного методу моніторингу та прогнозування цін акцій на фондовому ринку на основі декомпозиції часового ряду засобами ARIMA-LSTM з механізмом адаптивного перемикання ваг компонент відповідно до автоматично визначеного поточного ринкового режиму.

ІІІ. Гібридний метод прогнозування з адаптивним перемиканням

В основу запропонованого методу покладено припущення про адитивну декомпозицію ціни акції Pt на лінійну та нелінійну складові [3], що подається формулою (1):

$$P_t = L_t + N_t \tag{1}$$

де L_t – лінійна складова, яка моделюється процесом ARIMA, N_t – нелінійна стохастична складова, обумовлена нелінійними залежностями та зовнішніми факторами впливу на ринок.

Лінійна компонента описується моделлю ARIMA (p,d,q):

$$\Phi(B)(1-B)^d X_t = c + \Theta(B)\varepsilon_t \tag{2}$$

де B – оператор зсуву; $\Phi(B)$ та $\Theta(B)$ – поліноми авторегресії та ковзного середнього порядків p та q відповідно; d – ступінь інтегрування; ε_t – білий шум; c – константа моделі.

Нелінійна компонента N_t апроксимується LSTM-мережею, на вхід якої подаються залишки моделі ARIMA $e_t = P_t - P_t^{ARIMA}$, а також вектор технічних індикаторів. Стан мережі оновлюється через систему гейтів забування, входу та виходу, що дозволяє ефективно вловлювати довгострокові залежності у фінансовому ряді.

Ключовим елементом запропонованого методу є механізм адаптивного перемикання: ваги ARIMA- та LSTM-компонент визначаються динамічно залежно від поточного ринкового режиму. Підсумкова прогнозна оцінка ціни на горизонті h розраховується за формулою (3):

$$P_{t+h} = w_{ARIMA} \cdot P_{t+h}^{ARIMA} + w_{LSTM} \cdot N_{t+h}^{LSTM} \tag{3}$$

де $w_{ARIMA} + w_{LSTM} = 1$ – адаптивні вагові коефіцієнти, які залежать від поточного режиму ринку.

Класифікація ринкового режиму виконується за двома індикаторами технічного аналізу: ADX (Average Directional Index), що оцінює силу тренду, та нормалізованим показником волатильності ATR (Average True Range), обчисленим за формулою (4):

$$ATR_t^{norm} = ATR_t / P_t^{close} \quad (4)$$

Поточний режим ринку r_t визначається класифікаційною функцією $r_t = g(ADX_t, ATR_t^{norm})$ і належить до однієї з трьох категорій: трендовий, боковий (флет) або режим підвищеної волатильності. Відповідність ваг компонент режимам ринку наведено у таблиці 1.

Таблиця 1

Адаптивні ваги гібридного методу залежно від ринкового режиму

Режим ринку	Критерій класифікації	wARIMA	wLSTM
Трендовий	ADXt> 25	0.6	0.4
Боковий (флет)	ADXt ≤ 25, ATRnormt ≤ 0.02	0.5	0.5
Підвищена волатильність	ATRnormt> 0.02	0.2	0.8

Обґрунтування ваг базується на властивостях моделей: у трендовому режимі підвищується вага ARIMA, оскільки лінійна апроксимація ефективно описує спрямований стійкий рух ціни; у режимі підвищеної волатильності переважає LSTM, здатна фіксувати різкі нелінійні зміни. Для запобігання надмірно частим перемиканням у перехідних умовах застосовано згладжування: режим вважається зміненим лише за умови його стійкого спостереження протягом щонайменше трьох послідовних торгових сесій.

IV. Експериментальна верифікація методу

Запропонований метод реалізовано програмно з модульною архітектурою, що розділяє підсистеми отримання даних, обчислення технічних індикаторів, визначення ринкового режиму, навчання ARIMA- та LSTM-блоків і агрегації прогнозу. Експериментальне тестування виконано на історичних даних чотирьох акцій з різними переважними режимами: AAPL (трендовий), JPM (боковий), XOM (волатильний), SPY (трендовий) за період 2019–2024 рр. Точність прогнозу оцінювалась за метриками RMSE та DA (Directional Accuracy – точність визначення напряму руху ціни). Результати порівняння моделей на акції AAPL для горизонту прогнозу h = 1 день наведено у таблиці 2.

Таблиця 2

Порівняння точності моделей прогнозування (AAPL, h = 1 день)

Модель	RMSE	DA, %
ARIMA(2,1,2)	2.614	58.4
LSTM	1.873	63.7
Гібрид із фіксованими вагами 0.5/0.5	1.412	67.1
Запропонований метод (адаптивні ваги)	1.284	71.3

Запропонований метод знижує RMSE на 50.9 % порівняно з ізольованою моделлю ARIMA та на 31.4 % порівняно з ізольованою LSTM. Перевага над гібридом з фіксованими вагами становить 9.1% за RMSE та 4.2в.п. за DA, що підтверджує ефективність саме механізму адаптивного перемикання. На повному наборі тестових акцій точність визначення напряму руху ціни перебуває у діапазоні DA = 65.8–74.1% залежно від переважного ринкового режиму. Статистична значущість переваги підтверджена тестом Діболда–Маріано [5] ($p < 0.01$) для всіх парних порівнянь.

Висновок

Запропоновано гібридний метод моніторингу та прогнозування цін акцій на фондовому ринку, що поєднує статистичну модель ARIMA та рекурентну нейронну мережу LSTM з механізмом адаптивного перемикання ваг компонент за поточним режимом ринку. Експериментальна верифікація на історичних даних чотирьох акцій підтвердила перевагу запропонованого методу над ізольованими моделями та гібридом з фіксованими вагами як за метрикою RMSE, так і за точністю визначення напряму руху ціни. Подальші дослідження спрямовані на розширення набору ринкових режимів, інтеграцію аналізу тональності фінансових новин та адаптацію методу до внутрішньоденних таймфреймів.

Список використаних джерел

1. Box G. E. P., Jenkins G. M., Reinsel G. C., Ljung G. M. Time Series Analysis: Forecasting and Control. – 5th ed. – Hoboken : Wiley, 2015. – 712 p.
2. Zhang G. P. Time Series Forecasting Using a Hybrid ARIMA and Neural Network Model // Neurocomputing. – 2003. – Vol. 50. – P. 159–175.

УДК 004.4'2:004.65

МІКРОСЕРВІСНА АРХІТЕКТУРА СИСТЕМИ ЦИФРОВІЗАЦІЇ МЕДИЧНОЇ КАРТИ КОРИСТУВАЧА

Каськів А.В.
Західноукраїнський національний університет
магістрант

I. Постановка проблеми

В Україні цифровізація медичної документації реалізується через електронну систему охорони здоров'я (ЕСОЗ), яка станом на 2025 рік охопила понад 49 тисяч закладів та 475 тисяч медичних працівників, а в центральній базі даних накопичено понад 5 мільярдів медичних записів. Домінуюча сьогодні монолітна архітектура медичних інформаційних систем має суттєві обмеження: спільний процес сервера, бази даних і криптографічних операцій підвищує ризик компрометації майстер-ключа й утруднює застосування принципу мінімальних привілеїв. За даними ENISA, у 2024 році зафіксовано 487 інцидентів кібербезпеки в галузі охорони здоров'я, з яких 45% – атаки програм-вимагачів і 28% – витоки даних, що актуалізує задачу ізоляції критичних компонентів.

II. Мета роботи

Метою роботи є побудова чотири шарової мікросервісної архітектури системи цифровізації медичної карти користувача з ізоляцією криптографічних операцій у відокремленому процесі та організацією між компонентної взаємодії за принципом нульової довіри (NISTSP 800-207).

III. Архітектурне рішення

Запропонована система має чотиришарову структуру, узагальнену схему якої наведено на рисунку 1. Перший шар – клієнтський застосунок – реалізує інтерфейс і зберігає короткотривалі автентифікаційні токени. Другий шар – серверна компонента – обробляє запити, перевіряє права доступу та фіксує події у журналі аудиту. Третій шар – криптографічний сервіс – інкапсулює операції атрибутного шифрування CP-ABE та зберігає майстер-ключ у ізольованому процесі. Четвертий шар – сховище даних – складається з реляційної базиPostgreSQL з розширенням pgcrypto, кеш-сховища Redis та незмінного зовнішнього сховища для криптографічних якорів.

Рисунок 1 – Чотиришарова архітектура системи цифровізації медичної карти

Як видно з рисунку 1, виокремлення криптографічних операцій у окремий мікросервіс зумовлене ізоляцією майстер-ключа з обмеженим набором привілеїв, що відповідає принципу мінімальних привілеїв. Майстер-ключ зберігається у файлі з правами доступу 0600 всередині контейнера і доступнийсерверній компонентілише через REST-інтерфейс.

IV. Технологічний стек

Узагальнення технологічного стеку системи з розподілом за шарами архітектури та призначенням кожного компонента наведено в таблиці 1.

Таблиця 1

Технологічний стек системи цифровізації медичної карти

Шар	Технологія	Призначення
Клієнтський	Next.js, React, Tailwind	Інтерфейс для пацієнта, лікаря, адміністратора
Серверний	NestJS, TypeScript, Prisma	Контроль доступу, бізнес-логіка, журнал аудиту
Криптографічний	Python, Charm-Crypto, PBC	Атрибутнешифрування CP-ABE, ізоляціямайстер-ключа
Сховище	PostgreSQL 16, pgcrypto, Redis 7	Шифрування AES-256-GCM, кешатрибутів

Як показано у таблиці 1, серверну компоненту реалізовано на платформіNode.js з використанням NestJS і TypeScript. Доступ до бази даних організовано через Prisma, що генерує типи безпосередньо зі схеми даних і усуває ризик роз синхронізації моделі коду та структури сховища. Криптографічний сервіс реалізовано окремим мікросервісом на Python з бібліотекою Charm-Crypto, яка містить готову реалізацію канонічної схеми Bethencourt-Sahai-Waters з білінійною групою SS512.

V. Експериментальні характеристики

Середній час повного циклу обробки запитучитання поля медичної карти T обчислюється як сума часових витрат на основні етапи обробки і визначається формулою (1):

$$T = T_{auth} + T_{acc} + T_{crypto} + T_{audit} + T_{net} \quad (1)$$

де T_{auth}– час верифікації JWT-токена, T_{acc} – час прийняття рішення про доступ моделлю RBAC+ABAC, T_{crypto} – час розшифрування поля, T_{audit}– час запису у журнал аудиту, T_{net} – сумарна мережева затримка.

Згідно з формулою (1), експериментальні вимірювання дають середнє T ≈ 140 мс на сучасному серверному обладнанні, з яких половина припадає на криптографічне розшифрування. Включення кешу Redis скорочує T_{acc} з 34 до 6 мс. Навантажувальне тестування інструментом wrk демонструє пропускну спроможність до 147 запитів за секунду при 50 одночасних з'єднаннях; до рівня 20 з'єднань 99-й перцентиль часу відповіді залишається менше півсекунди, що відповідає галузевим нормам інтерактивної взаємодії.

Висновки

У роботі розроблено чотиришарову мікросервісну архітектуру системи цифровізації медичної карти користувача, що поєднує принципи нульової довіри (NIST SP 800-207) з ізоляцією критичних криптографічних операцій у відокремленому процесі з обмеженими привілеями. Виокремлення криптографічного сервісу у самостійний мікросервіс на Python з бібліотекою Charm-Crypto зумовлене відсутністю зрілих реалізацій атрибутного шифрування CP-ABE для платформи Node.js та необхідністю обмеження прямого доступу до майстер-ключа з боку серверної компоненти. Стандартизація між компонентної взаємодії через REST-інтерфейсіз JSON-серіалізацією повідомлень дозволяє незалежно розгортати, оновлювати та горизонтально масштабувати кожен шар системи без зміни прикладного коду решти компонентів. Експериментальнедослідження підтвердило середній час обробки запиту читання поля медичної карти близько 140 мс та пропускну спроможність до 147 запитів за секунду при 50 одночасних з'єднаннях, що відповідає галузевим нормам інтерактивної взаємодії з медичною інформаційною системою. Подальші дослідження спрямовані на горизонтальне масштабування криптографічногосервісу за балансувальником навантаження, інтеграцію з національними профілями стандарту HL7 FHIR та перехід на еліптичнікриві з рівнембезпеки 128 біт (BN, BLS12-381) для промислового впровадження.

Список використаних джерел

1. Rose S., Borchert O., Mitchell S., Connelly S. Zero Trust Architecture. NIST Special Publication 800-207. National Institute of Standards and Technology, 2020. doi: 10.6028/NIST.SP.800-207.
2. HL7 FHIR Release 5 Specification. HL7 International, March 2023. URL: https://hl7.org/fhir/R5/.
3. ISO/IEC 25010:2023. Systems and software engineering – SQuaRE – Product quality model. Geneva: ISO, 2023.
4. ENISA Threat Landscape: Health Sector 2024. European Union Agency for Cybersecurity, 2024.
5. Hu V.C., Kuhn D.R., Ferraiolo D.F. Attribute-based access control. IEEE Computer. 2015. Vol. 48, No. 2. P. 85–88. doi: 10.1109/MC.2015.33.

УДК 004.42

РОЗРОБКА КОМП'ЮТЕРНОГО ЗАСТОСУНКУ WEATHER HUB ДЛЯ ВІДОБРАЖЕННЯ МЕТЕОРОЛОГІЧНИХ ДАНИХ У РЕАЛЬНОМУ ЧАСІ

Плескун Б.Б.[1)], Крепич С.Я.[2)], Співак І.Я.[3)]
Західноукраїнський національний університет
[1)]студент; [2-3)]к.т.н., доцент

I. Постановка проблеми та актуальність

Найвідоміші на сьогодні метеорологічні застосунки – RainViewer, meteoblue та Weather Underground – попри широкий функціонал мають спільні недоліки: перевантаженість інтерфейсу велику кількість одночасно відображуваних параметрів, відсутність інтерактивності графіків (неможливість отримати точне значення у конкретний момент часу), а також незручний доступ до погодних даних за довільними географічними координатами. Ці обмеження знижують зручність використання продуктів для широкого кола користувачів, зокрема для тих, хто потребує швидкого й точного аналізу метеоінформації без зайвих складнощів.

II. Мета роботи

Метою роботи є підвищення зручності отримання метеорологічних даних шляхом розробки комп'ютерного застосунку Weather Hub на платформі .NET (C#, WPF) з інтерактивними графіками (OxyPlot), інтерактивною картою (GMap.NET) для перегляду погоди за координатами, відображенням якості повітря (AQI) та механізмом кешування запитів до OpenWeatherMap API.

III. Архітектура програмної системи

Застосунок Weather Hub побудовано на основі клієнт-серверної архітектури. Клієнтська частина (десктопний застосунок WPF) формує HTTP-запити до зовнішнього сервісу OpenWeatherMap API, отримує структуровані відповіді у форматі JSON та візуалізує їх через чотири функціональні модулі: головну панель (MainWindow), вікно аналізу погоди (WeatherWindow), вікно карти (MapWindow) та сервіс погоди (SimpleWeatherService). Кешування відповідей на стороні клієнта дозволяє скоротити кількість мережевих запитів і прискорити повторне відображення даних. Архітектурна схема системи наведена на рисунку 1.

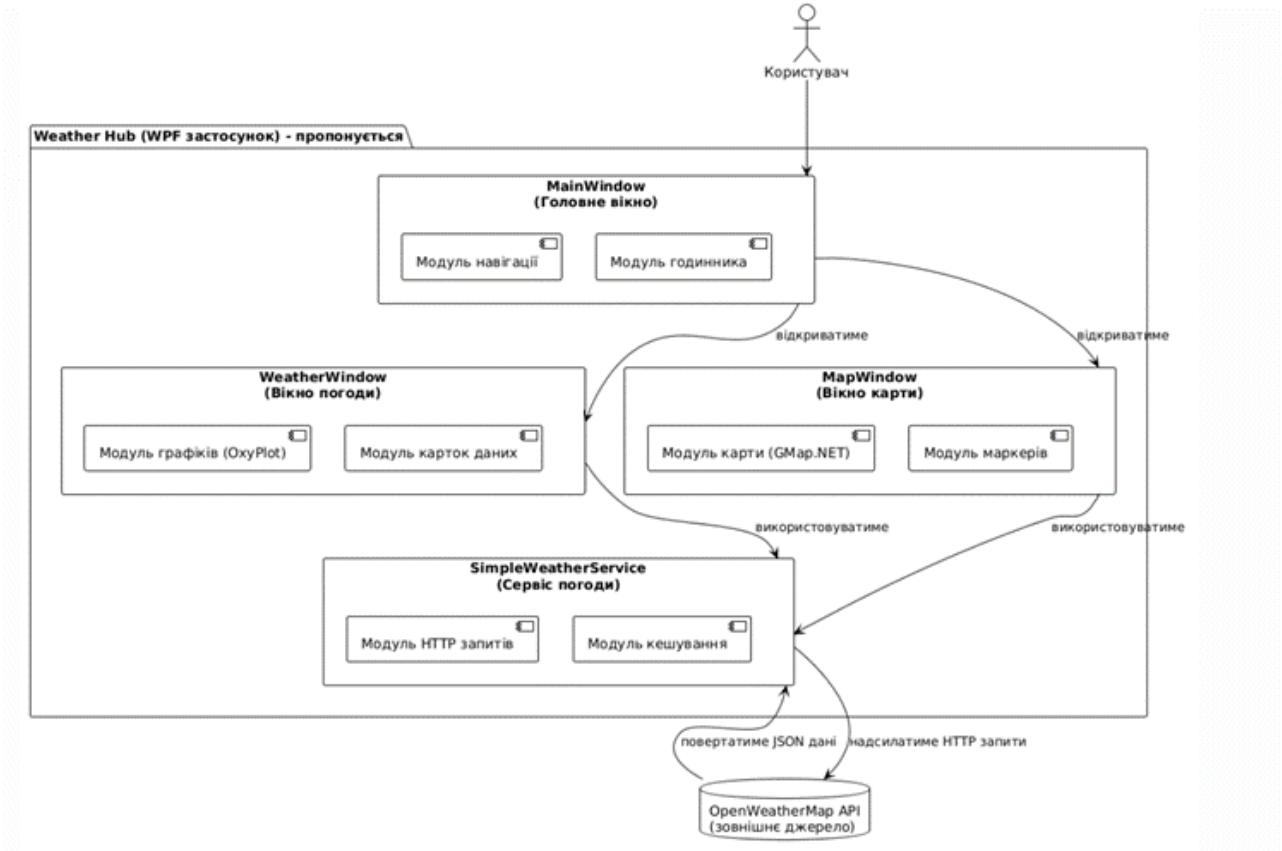

Рисунок 1 – Структурна схема програмної системи Weather Hub

Клас SimpleWeatherService виконує GET-запити до API з параметром назви міста або географічних координат, парсить JSON-відповідь та зберігає результат у словнику Dictionary<string, CacheEntry> з терміном дії 30 хвилин. Клас WeatherWindow взаємодіє з цим сервісом для отримання масиву температур і передає їх до методу UpdateChart(), який будує інтерактивний лінійний графік засобами OxyPlot. Клас MapWindow використовує бібліотеку GMap.NET для відображення карти OpenStreetMap та забезпечує отримання погоди за координатами при кліку на карту або ручному введенні широти і довготи. Ключовою функцією є побудова інтерактивного графіка температури за допомогою бібліотеки OxyPlot. Метод UpdateChart() формує лінійну серію точок на осі часу, де кожна точка відповідає прогнозованому значенню температури через кожні 3 години протягом 5 діб. При наведенні курсору на будь-яку точку графіка з'являється спливаюче вікно з точною датою, часом та значенням (див.рис. 2).

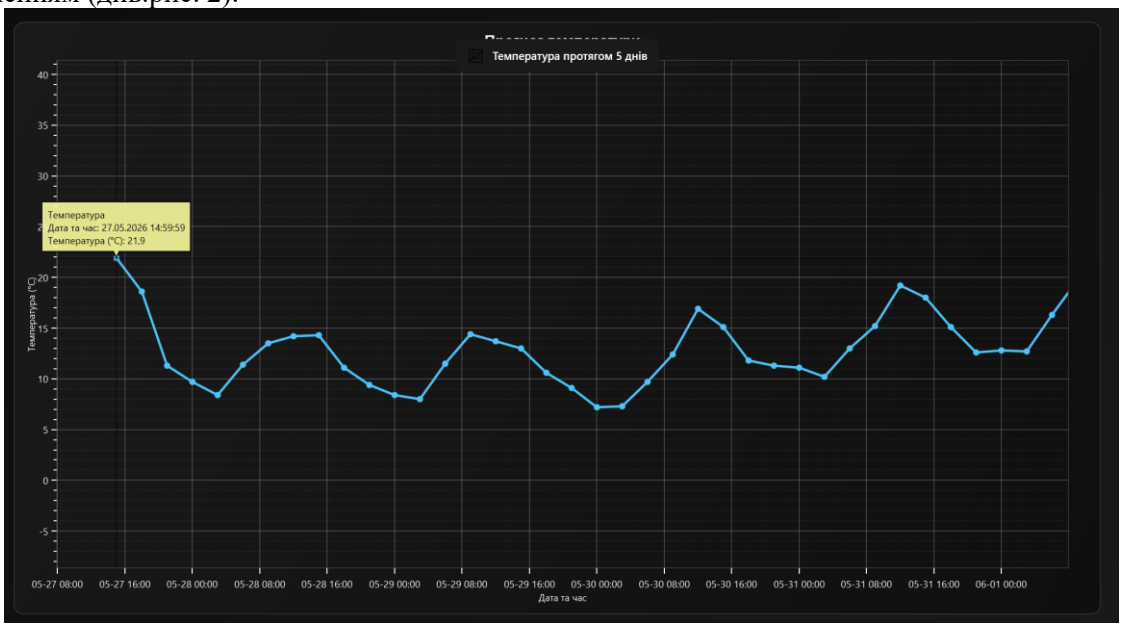

Рисунок 2 – Інтерактивний графік температури (OxyPlot)

Фрагмент коду реалізації:

```
private void UpdateChart(double[] temperatures)
{
    WeatherPlotModel.Series.Clear();
    var lineSeries = new LineSeries {
        Title = "Температура",
        Color = OxyColor.FromRgb(79, 195, 247),
        MarkerType = MarkerType.Circle,
        MarkerSize = 4, StrokeThickness = 3 };
    var startTime = DateTime.Now;
    for (int i = 0; i < temperatures.Length; i++) {
        var time = startTime.AddHours(i * 3);
        lineSeries.Points.Add(new DataPoint(
            DateTimeAxis.ToDouble(time), temperatures[i]));
    }
    WeatherPlotModel.Series.Add(lineSeries);
    WeatherPlotModel.InvalidatePlot(true);
}
```

Висновок

Розроблений застосунок Weather Hub усуває ключові недоліки існуючих аналогів: мінімалістичний інтерфейс із поділом даних на тематичні розділи спрощує сприйняття; інтерактивні графіки OxyPlot забезпечують точний перегляд значень у будь-який момент часу; модуль карти GMap.NET надає доступ до прогнозу для довільних географічних координат без обмеження населеними пунктами.

Список використаних джерел

1. OxyPlot Documentation [Електронний ресурс]. – Режим доступу: https://oxyplot.github.io/
2. I.Spivak, S.Krepych, M,Litvynchuk, S,Spivak, “Validation and data processing in JSON format”, IEEE EUROCON 19th International Conference on Smart Technologies, 2021. – pp.326-330
3. S.Krepych, P.Stakhiv, I.Spivak, “Analysis of the tolerance area parameters REC based on technological area scattering”, 12th International Conference on the Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), 2013. – pp.179-180
4. Microsoft Docs. Windows Presentation Foundation (WPF) [Електронний ресурс]. – Режим доступу: https://docs.microsoft.com/dotnet/desktop/wpf

УДК 004.891

АРХІТЕКТУРА ТА ЖИТТЄВИЙ ЦИКЛ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ УНІВЕРСАЛЬНОГО МАРКЕТПЛЕЙСУ ДЛЯ ОРЕНДИ ТОВАРІВ

Порплиця Н. П.[1)], Когут О.М.[2)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)]бакалавр

І. Постановка проблеми

Сучасний ринок економіки спільного споживання стрімко розвивається. Хоча на ньому домінують такі гіганти, як OLX, DOM.RIA чи AUTO.RIA, вони націлені переважно на продаж або на вузькі категорії (виключно щось одне), і функціонал оренди специфічних товарів там реалізований незручно або ж взагалі відсутній.

Водночас у повсякденному житті користувачів регулярно виникають ситуації, коли специфічний будівельний інструмент, вузькопрофільний прилад чи туристичне спорядження потрібні лише на короткий час або для одноразового використання, і купувати новий повноцінний прилад або спорядження у таких випадках є фінансово неграмотним.

Для вирішення цієї проблеми доцільно створити єдиний, універсальний маркетплейс, який забезпечував би можливість оренди найрізноманітніших речей: починаючи від інструментів і техніки, закінчуючи транспортними засобами та житлом. Такий підхід робить процес тимчасового користування речами набагато простішим ніж їх купівля, що й підтверджує актуальність розробки.

Окрім суто фінансової вигоди для користувачів, такий підхід значно сприяє розвитку екологічної складової та культури відповідального споживання. Зниження попиту на купівлю рідко використовуваних речей автоматично зменшує обсяги їх перевиробництва та мінімізує кількість електронних і побутових відходів. Це повністю відповідає сучасним глобальним тенденціям сталого розвитку, перетворюючи маркетплейс не лише на комерційний, а й на соціально значущий інструмент.

ІІ. Мета роботи

Метою роботи є проектування та програмна реалізація архітектури універсального маркетплейсу на базі технології ASP.NET, що дає автоматизацію повного циклу оренди різнотипних товарів у межах однієї цифрової платформи. Ключова увага падає на створення масштабованої системи з чітко визначеним конвеєром обробки даних, що дає ефективно керувати процесами пошуку, бронювання та взаємодії між користувачами.

ІІІ. Модель життєвого циклу маркетплейсу

На рисунку 1 проілюстровано загальну архітектуру та дворівневий життєвий цикл розробленого програмного забезпечення. Процес функціонування маркетплейсу логічно розділений на етап первинної підготовки та етап безперервної обробки замовлень.

Перший рівень, зображений у верхньому блоці, відповідає за безпечну ініціалізацію web-додатка. Цей процес розпочинається із запуску сервера та зчитування налаштувань конфігурації. Далі відбувається ініціалізація Entity Framework Core та здійснюється підключення до реляційної бази даних. На цьому етапі система виконує критичну перевірку: якщо підключення виявляється неуспішним, відбувається логування помилкиі кидання Error – процес підготовки зупиняється. У разі ж успішного з'єднання повертається код 200, який характеризує про готовність ситеми та перехід управління на наступний рівень.

Другий рівень, відображений у нижньому блоці діаграми, являє собою зациклений конвеєр обробки клієнтських запитів на оренду товарів. Знаходячись у стані очікування, сервер фіксує вхідний HTTP-запит та одразу виконує валідацію даних. Наступним ключовим кроком є перевірка доступності вибраного товару. Якщо річ наразі недоступна для бронювання, система подає відмову з помилкою та миттєво повертається до початкового стану очікування.

Якщо ж товар є вільним, запускається виконання основної бізнес-логіки: статус об'єкта змінюється на «Заброньовано», нова транзакція надійно зберігається у базі даних, після чого клієнту приходить успішна відповідь. Після закінчення операції система автоматично повертається в режим очікування нових HTTP-запитів, що й забезпечує безперервну та стабільну роботу нашого додатку.

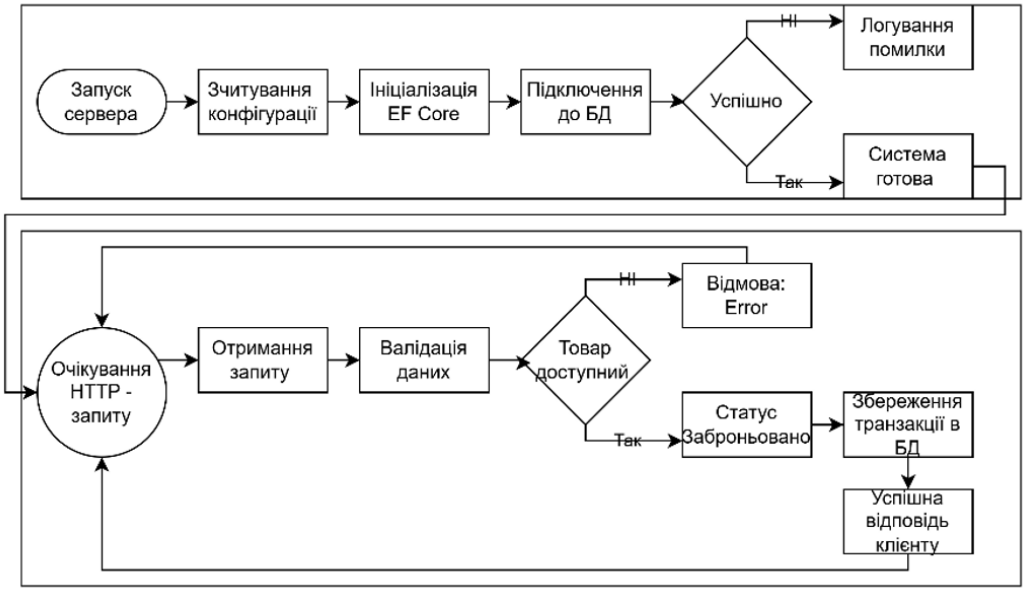

Рисунок 1 – Модель життєвого циклу маркетплейсу

IV. Архітектурна та програмна реалізація

Серверна частина маркетплейсу розроблена на базі мови C# та фреймворку ASP.NET Core. Це дозволяє нам створити сучасний додаток, який стабільно працює на різних операційних системах, таких як macOS та Linux.

Особливу увагу при розробці приділено кросплатформній оптимізації та зручності розгортання. Завдяки гнучкості .NET, процес локальної розробки та тестування серверної частини може ефективно здійснюватися на пристроях з ARM-архітектурою (зокрема, на базі процесорів серії AppleSilicon під управлінням macOS). Це забезпечує високу продуктивність під час написання коду та гарантує безшовну сумісність під час перенесення готових контейнерів через Docker у production-середовище на базі Linux-серверів або хмарних платформ.

Основна логіка побудована за принципами RESTful API, що дозволяє Backend ефективно взаємодіяти з web-інтерфейсом або мобільним додатком, а щоб сервер мав змогу обробляти одночасно багато запитів без затримок ми використовуємо асинхронне програмування. Для роботи з базою даних ми обрали EntityFramework Core. Він дозволяє значно простіше та безпечніше керувати даними та захищати систему від поширених загроз, таких як SQL-ін'єкції. Використання міграцій дає нам змогу легко оновлювати структуру БД при додаванні нових категорій товарів.

Не менш важливою частиною реалізації є контроль транзакцій: система гарантує, що один і той самий інструмент не буде заброньований двома різними людьми на однакові дати. Безпеку особистих кабінетів забезпечує стандарт JWT (JSON Web Tokens), який надійно захищає дані користувачів під час авторизації. Такий технічний стек робить маркетплейс надійним, швидким і зручним для кінцевого споживача.

Висновок

У ході роботи було розроблено та обгрунтовано архітектуру універсального маркетплейсу для оренди товарів. Аналіз ринку підтвердив, що створення такої платформи є актуальним, оскільки користувачі часто потребують інструменти чи техніку для одноразового використання, коли покупка є економічно невигідною. Запропонована дворівнева модель життєвого циклу програми довела свою ефективність, забезпечуючи стабільний запуск сервера та чіткий конвеєр обробки замовлень.

Використання технологій ASP.NET Core та EntityFramework Core дозволило реалізувати масштабовану систему, яка здатна витримувати високі навантаження та гарантувати безпеку транзакцій. Створений продукт пропонує зручну альтернативу традиційному продажу товарів, сприяючи раціональному споживанню ресурсів.

Список використаних джерел

1. ASP.NET Core documentation. https://learn.microsoft.com/en-us/aspnet/core/
2. Entity Framework Core documentation. https://learn.microsoft.com/en-us/ef/core/
3. JSON Web Token (JWT) introduction. https://jwt.io/introduction/
4. Botsman R., Rogers R. What's Mine Is Yours: The Rise of Collaborative Consumption. Harper Business, 2010. 304 p.
5. Docker Documentation. URL: https://docs.docker.com/

УДК 004.41:004.89

ІНТЕЛЕКТУАЛІЗОВАНИЙ ВЕБ-АСИСТЕНТ ПРОДАВЦЯ НЕРУХОМОСТІ З МОДУЛЕМ ОНЛАЙН КОНСУЛЬТАЦІЙ

Балицький Р.Р.[1)], Войтюк І.Ф.[2)]
Західноукраїнський національний університет
[1)] бакалавр; [2)] к.т.н., доцент

І. Постановка проблеми

Стрімка цифровізація ринку нерухомості спричинила стрімке зростання обсягів даних. Щодня генеруються тисячі нових оголошень про оренду та продаж об'єктів. Відповідно, веб-застосунки стали основним інструментом взаємодії між тими, хто шукає житло, та його власниками. Проте сучасні платформи стикаються з новою критичною проблемою: перевантаження продавців великою кількістю однотипних запитів від потенційних клієнтів. Існуючі рішення не забезпечують належної автоматизації первинної комунікації, що призводить до повільних відповідей та втрати зацікавленості покупців. Це зумовлює необхідність розробки принципово нового веб-застосунку на базі сучасних асинхронних архітектур, який міститиме інтелектуальний модуль для миттєвого консультування та автоматизації рутини продавця.

ІІ. Мета роботи

Мета роботи полягає у підвищенні ефективності та безпеки процесів пошуку, публікації та управління об'єктами нерухомості шляхом розробки спеціалізованого веб-застосунку. Застосунок має забезпечувати взаємодію між орендодавцями (продавцями) та клієнтами, поєднуючи високу швидкодію обробки запитів із впровадженням інтелектуалізованого модуля телефонії для цілодобових онлайн-консультацій.

ІІІ. Архітектура інтелектуалізованого веб-асистента

У рамках даної роботи було спроєктовано та розроблено повноцінний серверний веб-застосунок, орієнтований на автоматизацію роботи продавця нерухомості. Для досягнення цього було реалізовано класичну клієнт-серверну архітектуру. Основою системи є серверна частина, побудована з використанням сучасних технологій, зокрема фреймворку FastAPI. Цей інструмент дозволяє застосунку миттєво обробляти складні пошукові запити бази даних, а також забезпечує високу пропускну здатність для безперебійної взаємодії з модулем штучного інтелекту.

Особливу увагу при розробці приділено архітектурі модуля онлайн-консультацій на базі веб-сайту. Його побудовано на платформі ElevenLabs Conversational AI з увімкненим патерном RAG (Retrieval-Augmented Generation), що дозволяє генерувати відповіді, спираючись виключно на актуальні документи з підключеної бази знань компанії. Для побудови ланцюжків обробки діалогу, оркестрації промптів та отримання структурованих відповідей від LLM застосовано бібліотеку LangChain. Коли клієнт задає запитання на сайті, агент автоматично витягує релевантну інформацію зі сховища знань і синтезує природну відповідь у реальному часі через конвеєр STT, LLM, TTS. За результатами розмови система автоматично визначає рівень зацікавленості контакту (шкала cold/warm/hot, 0.0–1.0), генерує лист для подальшого зв'язку та надає рекомендації агенту щодо наступних кроків.

З метою забезпечення цілісності та достовірності даних, що надходять від зовнішніх сервісів, усі вхідні веб-хуки від ElevenLabs проходять обов'язкову перевірку HMAC-підпису на основі секретного ключа. Це унеможливлює підробку або несанкціоновану ініціацію обробки дзвінків. Доступ до зовнішніх API (ElevenLabs, OpenAI, Twilio) здійснюється виключно через ізольовані ключі, що зберігаються у захищених змінних середовища. Уся інформація, зібрана під час розмови (контактні дані, рівень інтересу, домовленості), зберігається у реляційній базі даних PostgreSQL з прив'язкою до унікального ідентифікатора розмови, що забезпечує повну аудитну трасу та контрольований доступ до результатів.

Для наочного відображення архітектури та послідовності взаємодії між компонентами системи розроблено діаграму послідовностей (Sequence Diagram) (рис. 1). Вона ілюструє повний цикл роботи голосового AI-агента: від налаштування агента ріелтором до ініціювання вихідного дзвінка, проведення розмови у реальному часі через конвеєр ElevenLabs ConvAI (STT, LLM, TTS) та автоматичного аналізу результатів після завершення дзвінка.

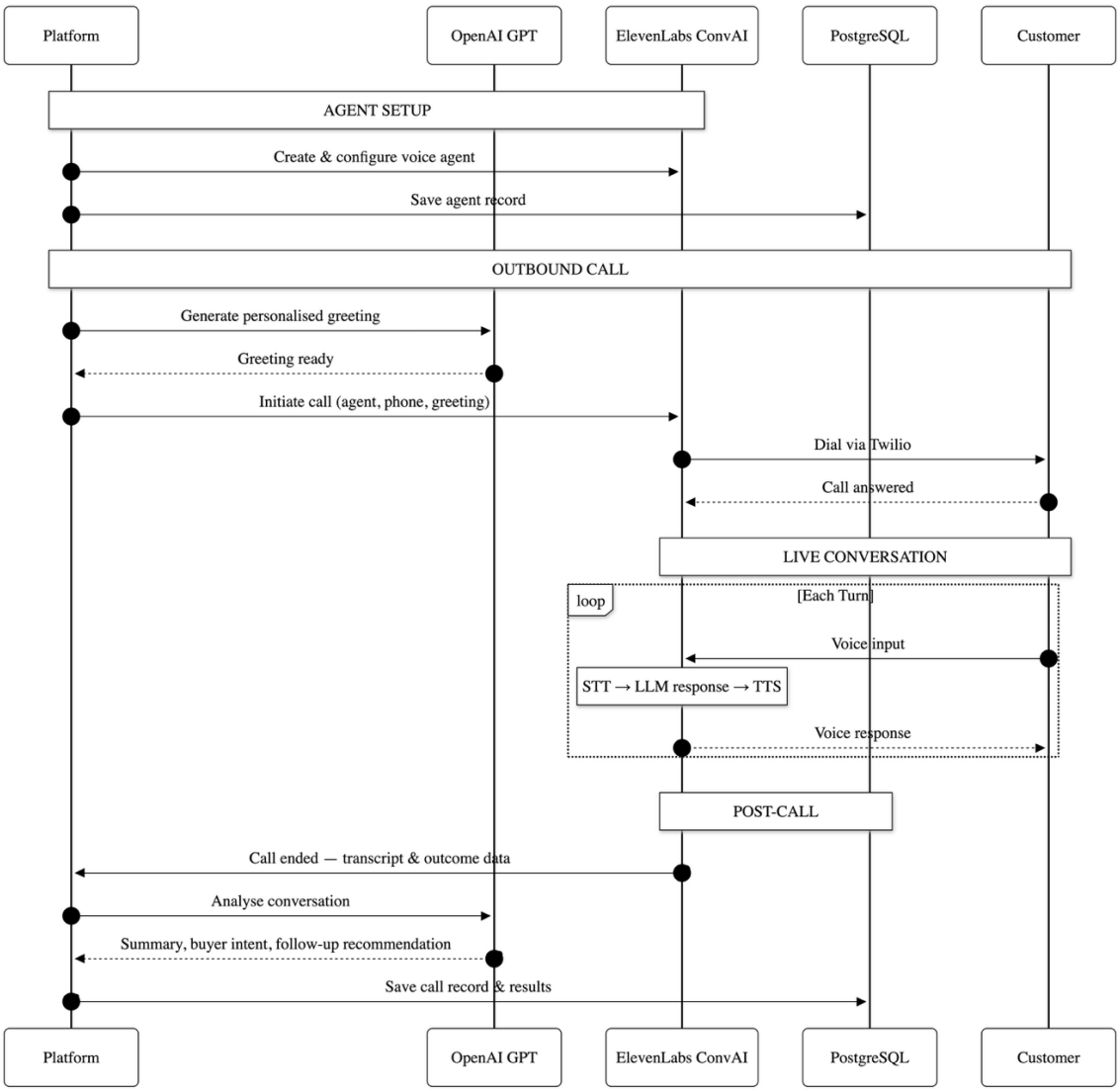

Рисунок 1 – Діаграма послідовностей взаємодії компонентів AI-платформи голосових агентів

Практична інтеграція для ріелторів реалізована у форматі вихідних автоматичних дзвінків потенційним покупцям: система самостійно генерує персоналізоване привітання на основі даних про об'єкт та клієнта, здійснює дзвінок через Twilio і проводить повноцінний діалог. Після кожної взаємодії платформа автоматично класифікує рівень зацікавленості контакту (cold / warm / hot), формує рекомендований лист для подальшого зв'язку та надає ріелтору аналітичні рекомендації щодо наступних кроків.

Висновок

Розроблено AI-платформу для автоматизації комунікацій на ринку нерухомості, яка поєднує вихідні голосові дзвінки та веб-консультації на основі технологій ElevenLabs ConvAI, OpenAI GPT та Twilio. Платформа самостійно веде переговори з потенційними покупцями, збирає структуровані дані, аналізує розмови та формує готові матеріали для подальшої роботи ріелтора – без залучення людини у процес первинної комунікації. Інтеграція RAG із базою знань забезпечує точність відповідей, а HMAC-верифікація веб-хуків та ізольоване зберігання ключів гарантують безпеку платформи. Система готова до розгортання як незалежне рішення для автоматизації продажів нерухомості.

Список використаних джерел

1. Lewis P., Perez E., Piktus A. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. Advances in Neural Information Processing Systems. 2020. Vol. 33. P. 9459-9474.
2. OpenAI. GPT-4 Technical Report. arXiv preprint. 2023. URL: https://arxiv.org/abs/2303.08774
3. LangChain Documentation. Introduction to LangChain for LLM application development. URL: https://python.langchain.com/
4. ElevenLabs. Conversational AI Platform Documentation. URL: https://elevenlabs.io/docs/conversational-ai/overview
5. Twilio Inc. Programmable Voice Documentation. URL: https://www.twilio.com/docs/voice

УДК 004.8

КЛІЄНТ-ОРІЄНТОВАНА АРХІТЕКТУРА ПЛАГІНА ДИНАМІЧНОЇ АДАПТАЦІЇ ІНТЕРФЕЙСІВ SAAS-ПЛАТФОРМ

Літинський О.Л.[1)], Папа О.А[2)], Вишневський Ю.Ю.[3)], Ковальський А.А.[4)]

Західноукраїнський національний університет

[1)]магістрант; [2)]д.філ., доцент; [3-4)] магістрант

Постановка проблеми

Стрімкий розвиток сучасних хмарних систем операційного управління (B2B SaaS) супроводжується значним ускладненням їх графічних інтерфейсів. Необхідність обробки великих масивів даних у високонавантажених корпоративних середовищах в умовах дефіциту часу створює надмірне когнітивне навантаження на користувача, викликаючи швидку втому та операційні помилки. Існуючі архітектурні рішення для адаптації зазвичай спираються на важкі серверні обчислення, що генерує критичні затримки, збільшує мережевий трафік та порушує приватність поведінкових даних користувача.

Відтак, в інженерії програмного забезпечення виникає потреба у проектуванні гнучкої, автономної архітектури на стороні клієнта, яка б забезпечувала проактивну зміну параметрів відображення веб-сторінки в реальному часі без залучення ресурсів серверного бекенду.

Мета роботи

Метою роботи є проектування та програмна реалізація клієнт-орієнтованої архітектури автономного плагіна для проактивної адаптації користувацьких веб-інтерфейсів SaaS-платформ на основі сучасних патернів проектування та технологій фронтенд-оптимізації.

Архітектурні рішення та особливості реалізації

Проектування архітектури адаптивного плагіна вимагає дотримання принципів модульності, низької зв'язності та високої продуктивності. Оскільки розроблене програмне забезпечення функціонує в середовищі клієнтського браузера паралельно з основною бізнес-логікою B2B SaaS-платформи, критично важливо уникнути конфліктів глобального простору імен та не допустити блокування головного потоку виконання.

Для досягнення цих цілей архітектуру плагіна побудовано на основі класичних об'єктно-орієнтованих патернів проектування.Побудована діаграма загальної архітектури (див.рис.1) наочно ілюструє концепцію ізольованого накладеного шару. Система чітко розмежована на три архітектурні контури:

1. контур операційної діяльності ПСП (SaaSHost), який включає модулі веб-додатка та DOM-дерево сторінки. Він функціонує за стандартною схемою, взаємодіючи з хмарним бекендом (API Gateway та базами даних) через асинхронні HTTPS REST-запити. Цей контур повністю відчужений від логіки адаптації, що забезпечує стабільність бізнес-процесів;
2. інтелектуальний контур діагностики (MathCore), який складається з компонентів TelemetrySensor та ViterbiDecoder. Він працює в пасивному фоновому режимі, перехоплюючи низькорівневі події та виконуючи логарифмічний інференс прихованих марковських ланцюгів на базі локального ковзного вікна.
3. контур проактивного управління (UI Adapter), який охоплює FuzzyController та UIManager. Він замикає гомеостатичний цикл: трансформує імовірнісні оцінки у чіткі коефіцієнти та виконує пряму інжекцію CSS-модифікаторів у DOM-структуру.

Головною перевагою такої архітектури є її повна автономність на стороні клієнта. Оскільки інтелектуальний плагін зациклений виключно всередині браузера і не здійснює мережевих запитів до сервера для прорахунку HMM чи Fuzzy-логіки, навантаження на хмарну інфраструктуру ПСП дорівнює нулю. Це гарантує максимальну масштабованість розробленого програмного забезпечення.

Внутрішня структура програмного модуля розділена на чотири ізольовані класи, які відповідають за конкретні етапи гібридної взаємодії:

– PluginCore (Головний контролер). Відповідає за життєвий цикл плагіна. У методі processTick() реалізовано основний таймер системи (наприклад, з інтервалом 1000 мс). Кожного такту ядро забирає дані з сенсора, прокидає їх через математичні класи і віддає команду UI-менеджеру на оновлення екрана;

– TelemetrySensor (Сенсорний модуль). Інкапсулює логіку роботи з подіями браузера. Містить методи onMouseMove та onClick, які обчислюють затримки та відхилення траєкторії курсору. На вимогу ядра повертає нормалізований вектор спостережень через публічний метод getObservationVector();
– ViterbiDecoder (HMM-декодер). Математичне серце системи. Зберігає матриці A та B (навчальні ваги моделі). Метод computeStateProbabilities() реалізує модифікований логарифмічний алгоритм Вітербі (через допоміжну функцію запобігання арифметичного заниження logSumExp) і повертає масив апостеріорних ймовірностей когнітивних станів користувача;
– FuzzyController (Нечіткий логік). Клас, що не має внутрішнього стану. Він приймає ймовірності станів від ViterbiDecoder, виконує операції фазифікації, застосовує продукційні правила з бази знань і повертає чіткі коефіцієнти дефазифікації за методом центру ваги (computeCentroid);
– UIManager (Виконавчий інтерфейсний модуль). Відповідає за безпосередню модифікацію DOM-дерева веб-сторінки. Метод injectCSSVariables() оновлює глобальні CSS-змінні (наприклад, plugin-base-font-size), що спричиняє миттєве, але плавне перемалювання елементів інтерфейсу самої SaaS-платформи засобами апаратного прискорення браузера.

Рисунок 1 – Діаграма загальної архітектури

Така чітка архітектурна декомпозиція забезпечує високу якість коду, легкість його тестування та дозволяє легко інтегрувати цей TypeScript-модуль у будь-яку веб-орієнтовану екосистему соціального підприємництва.

Висновок

Спроектована клієнт-орієнтована архітектура дозволила реалізувати автономний програмний модуль мовою TypeScript, повністю сумісний з будь-якими сучасними frontend-фреймворками (React, Angular, Vue). Перенесення обчислень на сторону клієнта гарантує час інференсу математичного ядра менше 0.5 мс за нульового додаткового навантаження на сервери SaaS-платформи. Результати експериментального впровадження розробленого архітектурного рішення підтвердили його ефективність, забезпечивши зниження середнього часу виконання операцій на 15.05% та скорочення рівня фрустрації користувачів на 73.80%.

Список використаних джерел

6. Chen, S., &Epps, J. (2021). Using Mouse Dynamics to Assess Cognitive Load. Proceedings of the ACM on Human-Computer Interaction, 5, pp.1-22.
7. Brdiczka, O., Yew, J., &Churchill, E. F. (2022). Adaptive User Interfaces: Challenges and Opportunities in the Era of AI. IEEE Pervasive Computing, 21(3), pp.34-42.
8. Pontones, C., Titzmann, A., Hübner, H., etal. (2023). ADABase: A Multimodal Dataset for Cognitive Load Estimation. Sensors, 23(1), 340p.
9. Li, Y., &Li, X. (2024). Real-timeWebEdge Computing for Adaptive Systems: Performance and Latency Optimization. Journal of WebEngineering, 23(2), 115-132.

УДК 004.77:004.42

ПРОГРАМНА РЕАЛІЗАЦІЯ ТА АРХІТЕКТУРА ВЕБ-СЕРВІСУ ДЛЯ ПІДТРИМКИ ПРОЦЕСІВ ОРЕНДИ ЖИТЛА

Балюх В.І.[1)], Крепич С.Я.[2)], Співак І.Я.[3)]
Західноукраїнський національний університет
[1)] студент; [2-3)] к.т.н., доцент

Вступ та актуальність розробки

Незважаючи на наявність на ринку великої кількості агрегаторів нерухомості (наприклад, OLX чи Flatfy), більшість із них функціонують виключно як вітрини оголошень. Вони не надають комплексних інструментів для супроводу всього життєвого циклу оренди. Традиційний процес пошуку супроводжується високим рівнем інформаційної асиметрії та комунікаційними бар'єрами (спам-дзвінки, використання сторонніх месенджерів). Метою розробки даного вебсервісу є створення єдиної платформи, яка автоматизує не лише пошук житла, але й процеси подачі формалізованих заявок, комунікації та управління об'єктами нерухомості.

Архітектура програмної системи

Практична реалізація розробленої системи базується на сучасній клієнт-серверній архітектурі з використанням стилю Representational State Transfer. Такий підхід дозволив розмежувати зони відповідальності:

1. Серверна частина реалізована мовою PHP з використанням фреймворку Laravel. Вона відповідає за бізнес-логіку, маршрутизацію, безпеку через токени Laravel Sanctum та взаємодію з реляційною базою даних PostgreSQL за допомогою ORM Eloquent.
2. Клієнтська частина розроблена за принципом Single Page Application на базі бібліотеки React та мови TypeScript. Збірка проєкту здійснюється за допомогою інструменту Vite, а стилізація інтерфейсу через Tailwind CSS. Обмін даними відбувається у форматі JSON за допомогою HTTP-клієнта Axios.

Розроблена система забезпечує виконання наступного ключового функціоналу:

1. Авторизація з розмежуванням ролей.
2. Керування об'єктами нерухомості, а саме, створення оголошень із мультизавантаженням фотографій, їх редагування, видалення та приховування.
3. Динамічна фільтрація каталогу за містом, ціною, кількістю кімнат та можливістю проживання з тваринами.
4. Подача та управління заявками на оренду, таких, як зміна статусів від “Очікує” до “Схвалено”/”Відхилено”.
5. Внутрішня комунікація користувачів через захищені чати.

Особливості реалізації. Автоматизація обробки заявок на оренду

Ключовою функціональною відмінністю розробленої платформи від звичайних дошок оголошень є наявність підсистеми формалізованих заявок. Замість неструктурованих телефонних дзвінків, орендар має можливість надіслати власнику офіційний запит із зазначенням бажаної дати заселення та супровідним повідомленням.

Загальну логіку взаємодії компонентів під час формування та обробки такої заявки зображено на діаграмі послідовності (див.рис.1).

Важливим архітектурним завданням на цьому етапі було забезпечення консистентності даних та захист орендодавців від спаму. Для цього на стороні сервера реалізовано жорстку бізнес-логіку: система блокує створення дублюючих запитів, якщо попередня заявка від цього ж користувача на даний об'єкт ще перебуває на розгляді. Програмну реалізацію серверної логіки валідації та створення заявки наведено у лістингу нижче.

```
publicfunctionstore(Request$request)
{
    $validated = $request->validate([
        'property_id' =>'required|exists:properties,id',
        'move_in_date' =>'required|date',
        'message' =>'nullable|string'
```

```
        ]);
        // Перевіряємо, чи це орендар
        if ($request->user()->role !== 'tenant') {
            returnresponse()->json(['message' =>'Тільки орендарі можуть подавати заявки'], 403);
        }
        // Перевіряємо, чи вже є активна заявка від цього орендаря на цю квартиру
        $existingRequest = RentalRequest::where('tenant_id', $request->user()->id)
            ->where('property_id', $validated['property_id'])
            ->whereIn('status', ['pending', 'approved'])
            ->first();
        if ($existingRequest) {
            returnresponse()->json(['message' =>'Ви вже подали заявку на це оголошення'], 422);
        }
        // Створюємо заявку
        $rentalRequest = RentalRequest::create([
            'property_id' =>$validated['property_id'],
            'tenant_id' =>$request->user()->id,
            'move_in_date' =>$validated['move_in_date'],
            'message' =>$validated['message'],
            'status' =>'pending'
        ]);
        returnresponse()->json($rentalRequest, 201);
    }
```

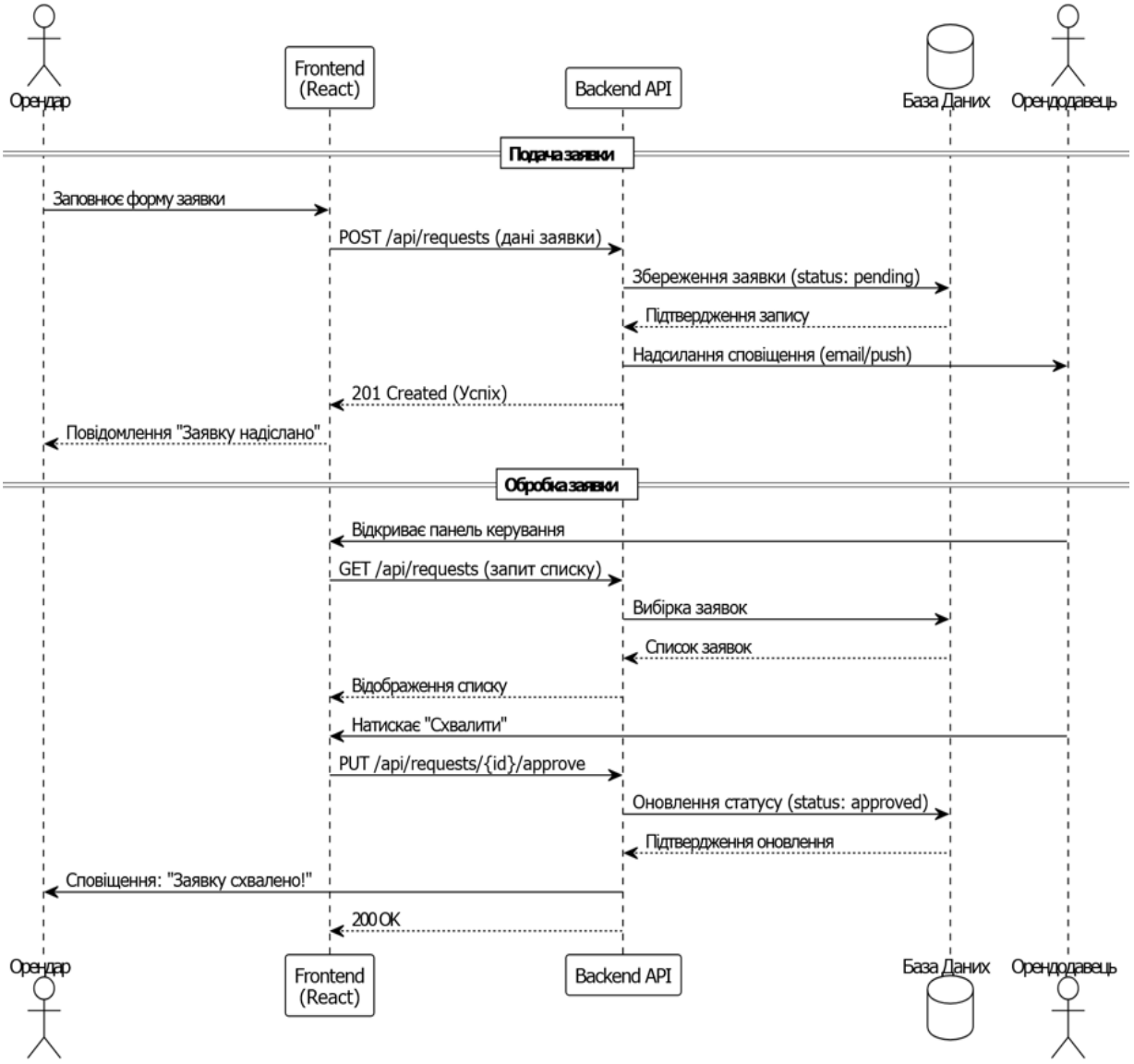

Рисунок 1 - Діаграма послідовності подачі та обробки заявки на оренду

На стороні клієнта HTTP-клієнт Axios перехоплює відповідь зі статусом 403 і виводить користувачеві відповідне попередження замість стандартної системної помилки. Успішно створені заявки миттєво відображаються в особистому кабінеті орендодавця, що на рисунку 2, де він може прийняти рішення щодо потенційного орендаря.

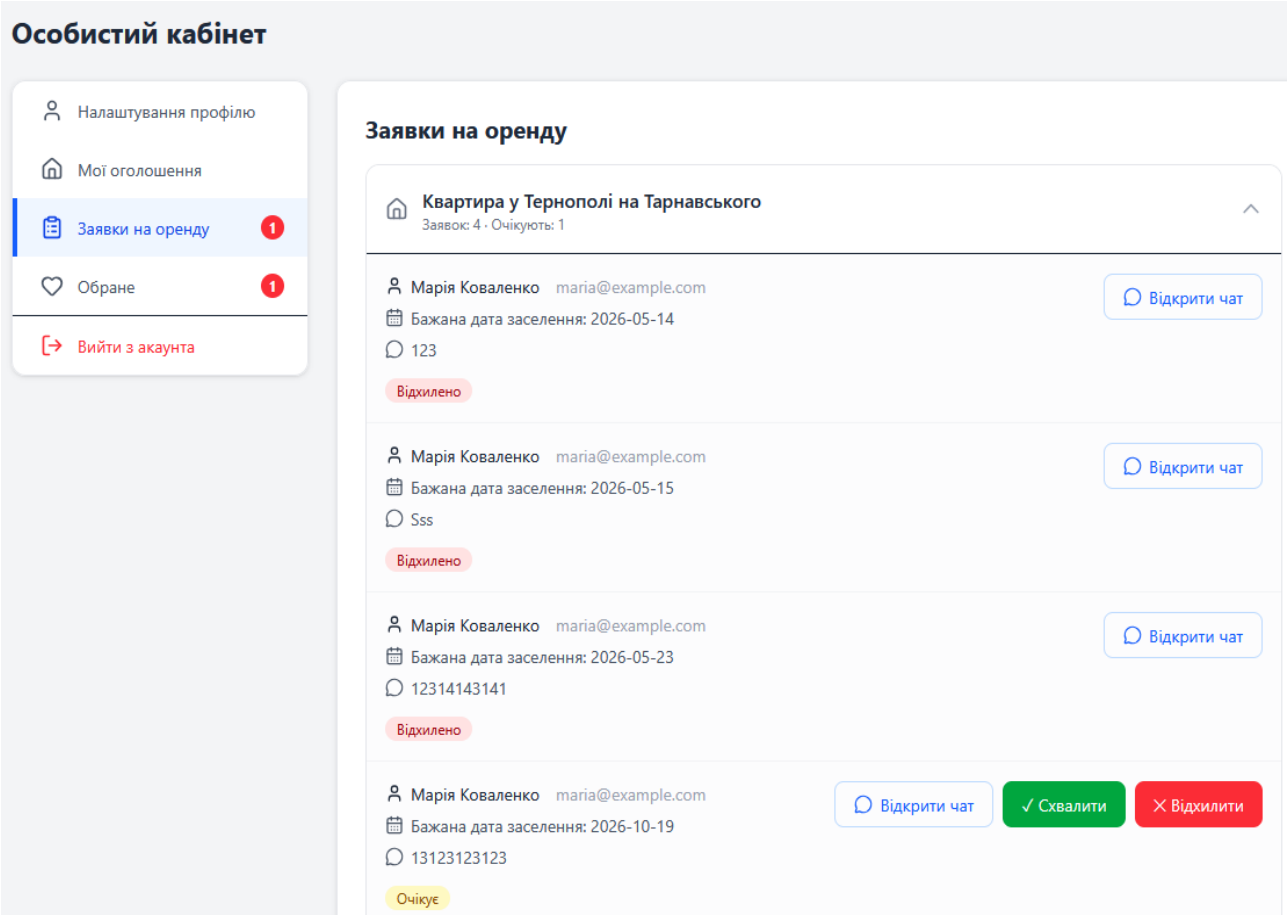

Рисунок 2.- Інтерфейс управління вхідними заявками в кабінеті орендодавця

Висновки

Розроблений вебсервіс є комплексним програмним рішенням, яке долає обмеження традиційних дошок оголошень. Використання стеку технологій React та Laravel дозволило створити швидкодіючий, безпечний та масштабований продукт. Впровадження фільтрації, системи структурованих заявок та внутрішніх чатів перетворює платформу на повноцінний інструмент підтримки та управління процесами оренди житлової нерухомості.

Список використаних джерел

1. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Prentice Hall, 2022. 432 p.
2. Крепич С.Я. Моделювання та забезпечення функціональної придатності статичних систем методами аналізу інтервальних даних/ С.Я.Крепич// дис.канд.тех.наук, НУ «Львівська політехніка», Львів, 2016. – 166с.
3. Stauffer M. Laravel: Up&Running: A Framework for Building Modern PHP Apps. 3rd Edition. O'ReillyMedia, 2022. 562 p.
4. React Documentation. URL: https://react.dev/
5. Співак І.Я., Крепич С.Я., Федоров О.А. Програмна система оцінювання ефективності праці в залежності від потреб. Матеріали школи-семінару молодих вчених і студентів «Комп'ютерні інформаційні технології» CIT'2019, 29 листопада 2019., Тернопіль, стр.34
6. Laravel Documentation. URL: https://laravel.com/docs/13.x/configuration
7. Laravel Sanctum. URL: https://laravel.com/docs/13.x/sanctum
8. Дивак.М.П., Крепич С.Я., Дивак Т.М., Манжула В.І., "Моделювання та забезпечення функціональної придатності технологічного обладнання лінії по виготовленню гіпсокартону в умовах змінних характеристик сировини". Вимірювальна та обчислювальна техніка в технологічних процесах. Вип.3. 2015. – с.186-192
9. Axios HTTP-Client. URL: https://devdocs.io/axios/api_intro
10. ORM Eloquent a begginers guide. URL: https://dev.to/icornea/mastering-eloquent-orm-a-beginners-guide-to-laravels-magic-2pj3

УДК 004.056.55:004.65

КЛАСИФІКАЦІЯ ПОЛІВ ЕЛЕКТРОННОЇ МЕДИЧНОЇ КАРТИ ЗА РІВНЕМ ЧУТЛИВОСТІ ТА ГІБРИДНЕ ШИФРУВАННЯ ДАНИХ

Каськів А.В.
Західноукраїнський національний університет
магістрант

I. Постановка проблеми

Медичні дані характеризуються різним ступенем чутливості: інформація про групу крові та інформація про ВІЛ-статус характеризуються принципово різним рівнем конфіденційності. У більшості наявних медичних інформаційних систем застосовується одноманітний підхід – або всі поля шифруються одним ключем, або не шифруються зовсім. За даними ENISA, у 2024 році в галузі охорони здоров'я зафіксовано 487 інцидентів кібербезпеки, з яких 28% – витоки даних, що актуалізує необхідність гранулярного підходу до захисту окремих груп полів залежно від ступеня їх чутливості.

II. Мета роботи

Метою роботи є розроблення моделі гранулярного захисту полів електронної медичної карти на основі чотирирівневої шкали чутливості та гібридної криптографічної схеми AES-256-GCM + CP-ABE.

III. Шкала чутливості полів медичної карти

Запропонована модель вводить чотирирівневу шкалу чутливості полів. Класифікація індукує природну партицію множини полів F на чотири попарно неперетинні підмножини. Шарову модель чутливості наведено на рисунку 1, а відповідні механізми криптографічного захисту для кожного рівня – у таблиці 1.

Рисунок 1 – Шарова модель чутливості полів медичної карти

Таблиця 1

Шкала чутливості полів та механізми захисту

Рівень	Приклади полів	Механізм захисту
L_1 – низький	Демографічні дані, контактна інформація, група крові	Відкрите зберігання; контроль на рівні застосунку
L_2 – середній	Історія звернень, призначення ліків, лабораторні результати	AES-256-GCM на рівні стовпців БД
L_3 – високий	Психоневрологічні дані, генетика, репродуктивний статус	Гібридна схема AES-256-GCM + CP-ABE
L_4 – критичний	ВІЛ-статус, дані про вживання наркотичних речовин	Гібридна схема + явна згода пацієнта

Як показано у таблиці 1, нижчі рівні чутливості захищаються лише на рівні застосунку, тоді як для високого та критичного рівнів обов'язковим є криптографічний захист з атрибутною політикою доступу.

IV. Гібридна схема шифрування

Безпосереднє шифрування медичних даних схемою CP-ABE є неефективним через обчислювальну складність білінійних спарювань. У моделі для кожного блоку полів генерується випадковий 256-бітний симетричний ключ K, яким шифрується блок даних M алгоритмом AES-256-GCM за формулою (1), а схема CP-ABE використовується лише для шифрування цього ключа з прив'язкою до дерева політики T за формулою (2):

$$C = Enc_{AES-GCM}(K, M, IV) \quad (1)$$

$$CT_K = Enc_{CP-ABE}(PK, K, T) \quad (2)$$

де C – шифротекст блоку даних, K – одноразовий симетричний ключ, IV – вектор ініціалізації, PK – відкритий параметр $CP-ABE$, T – деревовидна політика доступу. Згідно з формулами (1) і (2), користувач з атрибутами S, що задовольняє T, відновлює K через Decrypt $CP-ABE$ і розшифровує C. Аутентифікаційний тег GCM додатково забезпечує цілісність полів. Проблему динамічних контекстуальних атрибутів (декларація, згода) вирішено часовим маркером валідності та перевіркою актуального стану атрибута на рівні застосунку.

V. Експериментальні результати

Експериментальне дослідження продуктивності проведено на тестовому стенді з білінійною групою SS512 при різній кількості атрибутів у дереві політики. Час шифрування симетричного ключа схемою CP-ABE для типового сценарію медичної карти (5–10 листів у дереві політики, 3–5 використаних атрибутів) становить 80–140 мс, час розшифрування – 60–120 мс, шифрування 4 КБ блоку даних алгоритмом AES-256-GCM – менше 1 мс. Сумарний час обробки одного запиту читання поля третього рівня чутливості становить близько 140 мс. Відкликання доступу пацієнтом не потребує перешифрування полів і реалізується через зміну стану одного атрибута, що робить процедуру миттєвою та обчислювально дешевою.

VI. Порівняння з наявними рішеннями

У більшості сучасних систем класу EHR/EMR (Epic, Cerner/OracleHealth, OpenMRS, центральна база даних української ЕСОЗ) криптографічний захист реалізується як єдине шифрування на рівні стовпців бази даних, що не дозволяє розрізняти права доступу до окремих груп полів без додаткової перевірки на рівні застосунку. Запропонована модель усуває цю прогалину: навіть за умов несанкціонованого доступу до файлів сховища зловмисник отримує тільки шифротексти, для розшифрування яких потрібен набір атрибутів, що задовольняє відповідну політику.

Висновки

Розроблено модель гранулярного криптографічного захисту полів електронної медичної карти на основі чотирирівневої шкали чутливості та гібридної схеми AES-256-GCM + CP-ABE, у якій атрибутне шифрування виконується лише над коротким симетричним ключем, а блок даних шифрується значно швидшим AES-GCM. Така архітектура долає обчислювальну дорожнечу класичної CP-ABE та одночасно зберігає її гнучкий атрибутний контроль доступу. Проблему динамічних контекстуальних атрибутів (декларація, згода) розв'язано через часові маркери валідності та перевірку актуального стану на рівні застосунку, що забезпечує миттєве відкликання доступу без перешифрування полів. Експериментально підтверджено прийнятну для інтерактивної роботи продуктивність: 80–140 мс на шифрування ключа CP-ABE та менше 1 мс на блок даних AES-GCM. Подальші дослідження – перехід до сучасних еліптичних кривих з рівнем безпеки 128 біт (BLS12-381) та інтеграція з ресурсом Consent стандарту HL7 FHIR R5.

Список використаних джерел

1. National Institute of Standards and Technology. FIPS PUB 197: Advanced Encryption Standard (AES). November 2001 (оновлено 2023). doi: 10.6028/NIST.FIPS.197.
2. Dworkin M. Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC. NIST SP 800-38D, 2007. doi: 10.6028/NIST.SP.800-38D.
3. Bethencourt J., Sahai A., Waters B. Ciphertext-policy attribute-based encryption. Proc. IEEE Symposium on Security and Privacy. 2007. P. 321–334. doi: 10.1109/SP.2007.11.
4. Qiao J., Wang N., Fu J., Deng L., Wang J., Liu J. A lightweight CP-ABE scheme for EHR over cloud based on blockchain and secure multi-party computation. Trans. Emerging Telecom. Technologies. 2025. Vol. 36, No. 2. e70053.
5. HL7 FHIR Release 5 Specification. HL7 International, March 2023. URL: https://hl7.org/fhir/R5/

УДК 004.41

РОЗРОБКА СИСТЕМИ КЕРУВАННЯ СТУДЕНТСЬКИМИ ПРОЕКТНИМИ ГРУПАМИ

Крепич С.Я.[1)], Мороз С.В.[2)], Співак І.Я.[3)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] бакалавр; [3)] к.т.н., доцент

Постановка проблеми

У сучасних умовах цифровізації вищої освіти та переходу до проектно-орієнтованих методик навчання, здатність викладача ефективно координувати діяльність багатьох проектних груп стає критичним фактором успіху навчального процесу. Традиційні підходи до моніторингу, засновані на використанні неспеціалізованих засобів комунікації (електронна пошта, месенджери), виявляються недостатньо гнучкими та прозорими. Зокрема, за відсутності централізованого контролю виникає проблема втрати академічного інтелектуального капіталу: результати досліджень студентів не накопичуються в єдиній базі знань кафедри, що ускладнює моніторинг наступності тем та аналіз динаміки публікаційної активності протягом семестру. Окрім того, ручне формування звітів викладачем займає до 40% робочого часу, який міг би бути спрямований на наукове консультування. Таким чином, виникає гостра потреба у впровадженні інструментарію, який би автоматизував агрегацію даних та забезпечив верифіковане сховище для наукових доробків.

Мета роботи

Метою даної роботи є проектування та програмна реалізація інтерактивної веб-системи AcadRepo, призначеної для автоматизації процесів реєстрації проектних груп, централізованого зберігання наукових доробків (тез, статей, звітів) та генерації аналітичної звітності про активність студентських колективів.

Основна частина

У межах проведеного дослідження було спроектовано та реалізовано програмний комплекс, що базується на клієнт-серверній архітектурі та сучасних веб-технологіях.

Проектована система AcadRepo базується на ієрархічній моделі управління, де викладач виступає в ролі адміністратора (куратора) проектів, а студентські колективи — активними суб'єктами, що генерують звітний контент. Логіка системи побудована навколо життєвого циклу публікації, який включає стадії завантаження, первинної перевірки, верифікації та остаточного включення у зведений семестровий звіт. Такий підхід дозволяє перетворити звичайний файловий архів на інтерактивну аналітичну платформу.

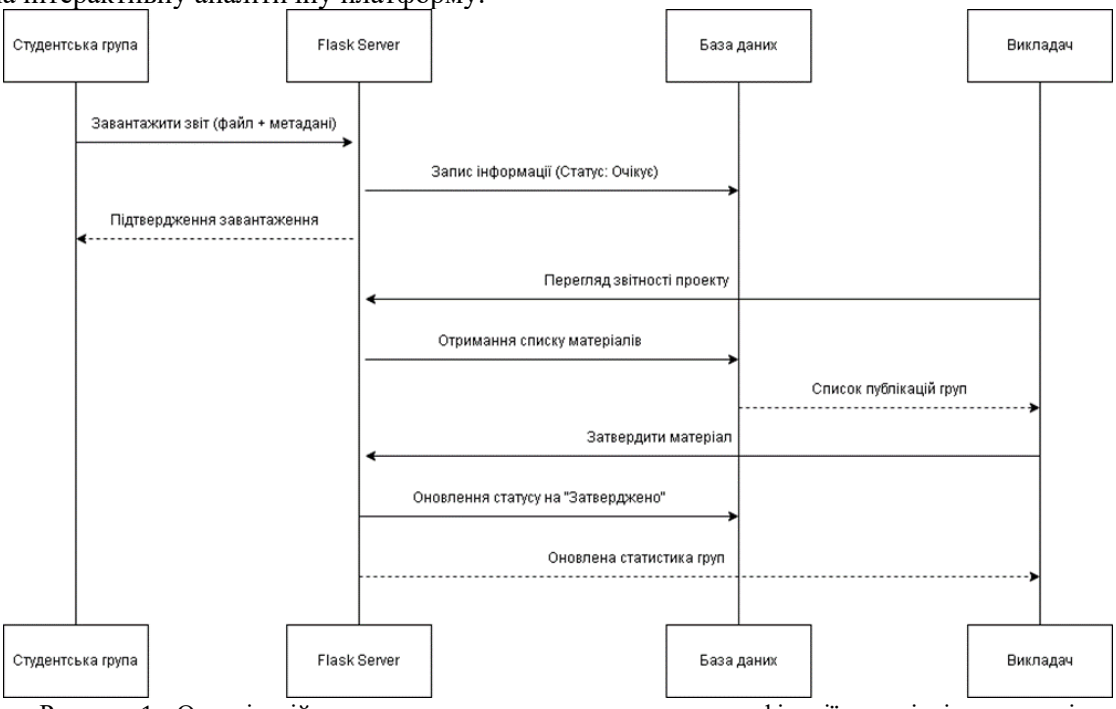

Рисунок 1 - Організаційна схема процесу завантаження та верифікації матеріалів у системі

Центральною ланкою системи є алгоритм автоматизованої звітності. Система не просто зберігає файли, а агрегує метадані кожної публікації для формування аналітичних розрізів успішності. На основі кількості затверджених матеріалів розраховується відсоток виконання плану кожної проектної групи. Візуалізація даних реалізована за допомогою компонентів Chart.js, що трансформують цифрові показники у зручні для сприйняття гістограми та кругові діаграми активності.

Апробація розробленого прототипу проводилася у межах кафедри комп'ютерних наук. Під час дослідної експлуатації було створено структуру колективів для дисципліни «Інженерія програмного забезпечення», завантажено понад 40 тестових публікацій (див.рис.2) та згенеровано зведені звіти за семестр. Результати випробувань показали високу швидкість обробки запитів (менше 200 мс) та стабільність роботи сховища при пікових навантаженнях.

Для реалізації взаємодії з репозиторієм на стороні клієнта розроблено компонент FileService, який використовує асинхронні запити для передачі бінарних даних.

```
const uploadPublication = async (fileData) => {
  const formData = newFormData();
formData.append('file', fileData.file);
formData.append('title', fileData.title);
formData.append('groupId', fileData.groupId);
try {
   const response = awaitaxios.post('/api/materials/upload', formData, {
headers: { 'Content-Type': 'multipart/form-data' }
   });
dispatch({ type: 'UPLOAD_SUCCESS', payload: response.data });
   return response.status;
  } catch (error) {
   console.error("Помилка завантаження матеріалу:", error);
  }
};
```

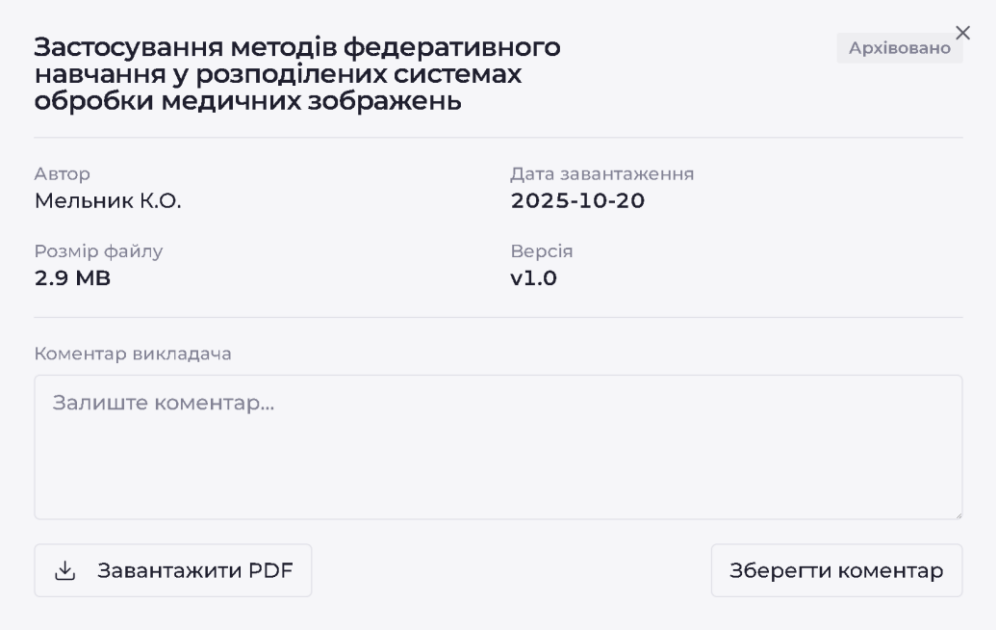

Рисунок.2 – Модальне вікно детальної інформації та верифікації публікації

Висновки

Розроблений прототип системи успішно демонструє життєздатність підходу до автоматизації академічного контролю через створення вузькоспеціалізованих репозиторіїв. Програмна реалізація на базі React забезпечила необхідну гнучкість інтерфейсу для роботи з великими реєстрами даних. Дослідна експлуатація підтвердила, що впровадження системи AcadRepo дозволяє централізувати інтелектуальні напрацювання кафедр та скоротити час на підготовку звітної документації на 30-40%.

Список використаних джерел

1. React Official Documentation. Components and Hooks. URL: https://react.dev/
2. Yu.Bobalo, M.Dyvak, S.Krepych, P.Stakhiv, “Evaluation of functional device suitability, with considering of random technological deviations of the parameters from the nominal and process of component aging”, Przegld Elektrotechniczny, Warszawa, Poland, Vol 2014, Nr 4, 2014. – P.224-228
3. Chart.js: Open source HTML5 Chartsdocumentation. URL: https://www.chartjs.org/

УДК 528.74:004.93

АЛГОРИТМ СЕГМЕНТАЦІЇ БУДІВЕЛЬ НА СУПУТНИКОВОМУ ЗНІМКУ МЕТОДОМ ГОЛОСУВАННЯ ТРЬОХ ПОРОГОВИХ МЕТОДІВ

Ткач М.М.
Західноукраїнський національний університет
студент

I. Постановка проблеми

Сегментація будівель на супутниковому знімку є початковим етапом задачі автоматизованого визначення висоти забудови за тінню та значною мірою визначає точність кінцевого результату. Завдання сегментації полягає у виділенні на знімку контурів будівель як суцільних областей, придатних для подальшого зіставлення з тінями та обчислення геометричних характеристик. Через нерівномірність освітлення, різну відбивну здатність дахів та наявність різнорідного фону жоден з відомих простих методів порогової обробки не забезпечує надійного результату на всіх типах знімків.

Класичні методи порогової обробки є прозорими, передбачуваними та піддаються налаштуванню обмеженим набором параметрів, проте кожен з них має власні обмеження: глобальний поріг чутливий до нерівномірного освітлення, адаптивний поріг реагує на яскраву текстуру поверхні, а контурний метод спрацьовує на будь-яких різких перепадах яскравості. Поєднання таких методів за принципом голосування підвищує надійність сегментації без істотного зростання обчислювальної складності.

II. Мета роботи

Метою роботи є розроблення алгоритму сегментації будівель на супутниковому знімку, що поєднує переваги кількох незалежних методів порогової обробки за рахунок процедури голосування за більшістю та забезпечує стійкість сегментації до хибних спрацювань окремих методів.

III. Основна частина

Перед сегментацією знімок переводиться у відтінки сірого зважуванням кольорових каналів. До отриманого зображення застосовується контрастно-обмежене адаптивне вирівнювання гістограми (CLAHE), що обробляє зображення локально та обмежує підсилення контрасту. На відміну від звичайного вирівнювання гістограми, такий підхід запобігає надмірному підсиленню шуму на однорідних ділянках і підвищує локальний контраст між будівлями, тінями та фоном. Після вирівнювання застосовується згладжування фільтром Гауса, що зменшує дрібнозернистий шум і стабілізує подальшу порогову обробку.

Перший метод використовує глобальний поріг яскравості за Отсу, що автоматично обирає поріг, який максимізує міжкласову дисперсію яскравості, тобто найкраще розділяє пікселі на дві групи – світлі дахи будівель та темніший фон. Метод найкраще працює за рівномірного освітлення та вираженого контрасту.

Другий метод застосовує адаптивний локальний поріг, що обчислюється у вікні фіксованого розміру навколо кожного пікселя як середньозважене значення яскравості сусідніх пікселів зі зміщенням на сталу величину. Завдяки локальному характеру метод нечутливий до плавних змін освітленості по полю знімка та коректно виділяє об’єкти, яскравіші за свій безпосередній окіл.

Третій метод ґрунтується на виявленні меж об’єктів детектором Кенні, який виконує згладжування зображення, обчислення градієнта яскравості, потоншення меж до товщини одного пікселя та порогову фільтрацію з відстеженням зв’язності. Виявлені тонкі контури не утворюють суцільних областей, тому до них застосовується розширення та закриття, що з’єднує близькі межі у замкнені контури. Замкнені контури заповнюються, формуючи суцільні маски будівель. Перевагою методу є здатність виділяти будівлі за слабкого контрасту дахів із фоном. Призначення, ознаки виявлення та обмеження трьох методів узагальнено в таблиці 1.

Таблиця 1

Методи виявлення будівель та їхні характеристики

Метод	Ознака виявлення	Обмеження
Глобальний поріг (Отсу)	Загальний контраст яскравих дахів із фоном	Чутливий до нерівномірного освітлення знімка
Адаптивний локальний поріг	Локальна яскравість об’єкта відносно околу	Реагує на яскраву текстуру поверхні
Детектор меж (Кенні)	Виражені краї будівель	Реагує на будь-які різкі перепади яскравості

Результати трьох методів об’єднуються процедурою голосування. Кожен метод формує двійкову маску, у якій одиниця відповідає пікселю, віднесеному до будівель, а нуль – пікселю фону. Піксель зараховується до підсумкової маски будівель, якщо щонайменше два з трьох методів класифікували його як такий. Формально умову віднесення пікселя до будівель записують у вигляді:

$$B(x, y) = 1, \textit{ якщо } M_1(x, y) + M_2(x, y) + M_3(x, y) \geq 2 \quad (1)$$

де $B(x, y)$ – результат класифікації пікселя з координатами x, y; M_1, M_2, M_3 – двійкові результати першого, другого та третього методів відповідно. Реалізація процедури зводиться до підсумовування трьох двійкових масок та порівняння суми з порогом, що дорівнює двом.

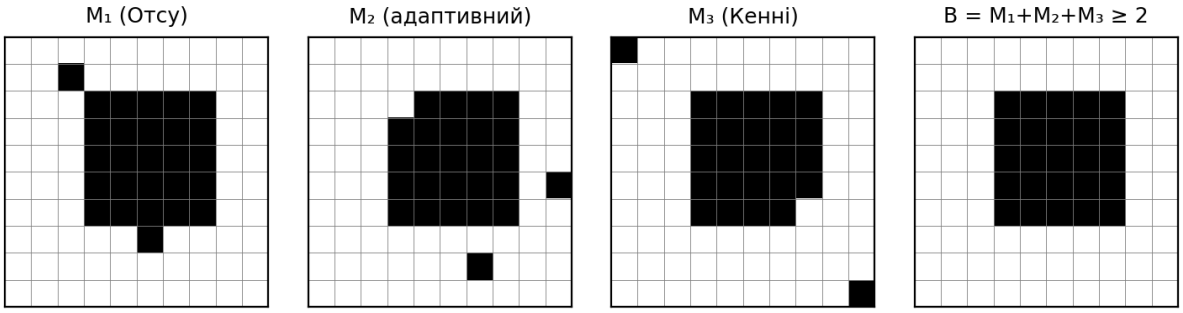

Рисунок 1 – Принцип формування підсумкової маски методом голосування

Принцип формування підсумкової маски методом голосування ілюструє рисунок 1. Доцільність правила більшості пояснюється тим, що кожен окремий метод дає хибні спрацювання за різних умов. Імовірність того, що один і той самий хибний об’єкт буде одночасно виявлений щонайменше двома методами, є істотно нижчою за ймовірність хибного спрацювання окремого методу. Водночас вимога згоди всіх трьох методів призводила б до пропуску значної частини справжніх будівель, виявлених лише двома методами через обмеження третього. Тому голосування за більшістю обрано як компроміс між повнотою та точністю виявлення. Отримана після голосування маска проходить морфологічну обробку операцією відкриття, що є послідовним застосуванням ерозії та нарощування і усуває дрібні поодинокі області шуму. На очищеній масці виділяються зовнішні контури об’єктів, кожен з яких перевіряється за двома критеріями: площа об’єкта має перевищувати задану мінімальну площу, а співвідношення сторін описаного прямокутника не повинно бути надто витягнутим. Перший критерій відсіває дрібний шум, другий – видовжені об’єкти на кшталт доріг та річок, які не є будівлями. Об’єкти, що задовольняють обидва критерії, формують підсумковий перелік будівель. Критерій компактності за описаним прямокутником записують у вигляді:

$$k = \min(w, h) / \max(w, h) \geq k_{\min} \quad (2)$$

де k – коефіцієнт компактності об’єкта; w, h – ширина та висота описаного навколо об’єкта прямокутника; $k_{\min}$ – нижня межа допустимих значень коефіцієнта компактності. Для компактних об’єктів коефіцієнт є помірним, а для видовжених – малим, що дозволяє відсіяти лінійні структури, зберігши будівлі.

Мінімальна допустима площа задається з урахуванням просторової роздільної здатності знімка, щоб виключити поодинокі пікселі та артефакти сегментації, але зберегти навіть невеликі за розміром будівлі. Поріг витягнутості підбирається експериментально як компроміс між відсіванням лінійних структур і збереженням видовжених у плані будівель, таких як виробничі корпуси чи складські комплекси.

Висновок

Розглянуто алгоритм сегментації будівель на супутниковому знімку, що поєднує три незалежні методи порогової обробки. Запропоновано виконувати попередню обробку знімка з контрастно-обмеженим адаптивним вирівнюванням гістограми та згладжуванням, а після голосування застосовувати морфологічне очищення та фільтрацію контурів за площею та співвідношенням сторін. Алгоритм є обчислювально простим, не потребує навчальних даних і придатний для оброблення знімків міської забудови без застосування спеціалізованих обчислювачів.

Список використаних джерел

1. Otsu N. A threshold selection method from gray-level histograms. IEEE Transactions on Systems, Man, and Cybernetics. 1979. Vol. 9, No. 1. P. 62–66. DOI: 10.1109/TSMC.1979.4310076.

УДК 004.41

РОЗРОБКА ВЕБ ЗАСТОСУНКУ ДЛЯ УПРАВЛІННЯ ОГОЛОШЕННЯМИ З ПРОДАЖУ ТА ОРЕНДИ НЕРУХОМОСТІ

Атаманчук В.В.[1)] , Юшко А.В.[2)]
Західноукраїнський національний університет
[1)]бакалавр; [2)]д.філ., старший викладач

І. Постановка проблеми

Стрімка цифровізація ринку нерухомості спричинила дуже велике зростання обсягів даних. Щодня генеруються тисячі нових оголошень про оренду та продаж об'єктів. Відповідно, веб-застосунки стали основним інструментом взаємодії безпосередньо між тими, хто шукає житло, та його власниками.

Проте незважаючи на велику кількість наявних платформ, ринок стикається з критичною проблемою: існуючі рішення не забезпечують належної прозорості та не захищають унікальність оригінальних публікацій. Це створює сприятливе середовище для недобросовісних посередників, які масово дублюють оголошення від реальних власників.

Отже, проблема полягає у суперечності між зростаючою потребою суспільства у прямій, безпечній та безбар'єрній взаємодії між власником і клієнтом та недосконалістю наявних програмних рішень. Сучасні платформи не лише не мають ефективних алгоритмів захисту від несанкціонованого копіювання контенту, а й страждають від проблем зі швидкодією, масштабованістю бекенду та ергономічністю користувацьких інтерфейсів.

Це зумовлює необхідність розробки принципово нового веб-застосунку на базі сучасних технологій, який гарантуватиме прямий зв'язок без посередників, з акцентом на оптимізований API, високі стандарти візуальної доступності й надійний захист авторських даних.

ІІ. Мета роботи

Мета роботи полягає у підвищенні ефективності та безпеки процесів пошуку, публікації та управління об'єктами нерухомості шляхом розробки спеціалізованого веб-застосунку, який забезпечує пряму взаємодію між орендодавцями/продавцями та клієнтами, поєднуючи високу швидкодію обробки запитів, покращений користувацький досвід (UX) та інтегровану систему протидії крадіжці контенту.

ІІІ. Основна частина

У рамках даної роботи було спроєктовано та розроблено повноцінний веб-застосунок, призначений для безпечного управління оголошеннями з продажу та оренди нерухомості. Головною метою створення цього програмного продукту було вирішення проблеми повільного пошуку, перевантажених інтерфейсів та вразливостей даних власників на ринку PropTech. Для досягнення цього було реалізовано класичну клієнт-серверну архітектуру, що дозволило незалежно оптимізувати обробку даних на сервері та відображення інформації у браузері користувача.

Основу системи становить серверна частина, реалізована із застосуванням технологічного стеку PERN, зокрема платформи Node.js та фреймворку Express.js. Вибір зазначених технологій обумовлений необхідністю забезпечення високої продуктивності при обробці складних пошукових запитів із багатопараметричною фільтрацією (за ціною, площею та локацією) в умовах паралельного виконання процесів.

Збереження даних реалізовано засобами реляційної СУБД PostgreSQL, структуру якої спроєктовано відповідно до принципів нормалізації з метою забезпечення цілісності даних та унеможливлення їх дублювання.

Для захисту інфраструктури та убезпечення учасників угод було створено комплексну підсистему безпеки. З метою недопущення масової реєстрації фейкових акаунтів третіми особами, впроваджено механізм суворої верифікації користувачів (наприклад, через інтеграцію з системами електронної ідентифікації рівня BankID). Звичайний гість може лише переглядати базу, тоді як авторизований та підтверджений власник отримує доступ до спеціальної панелі управління для публікації оголошень.

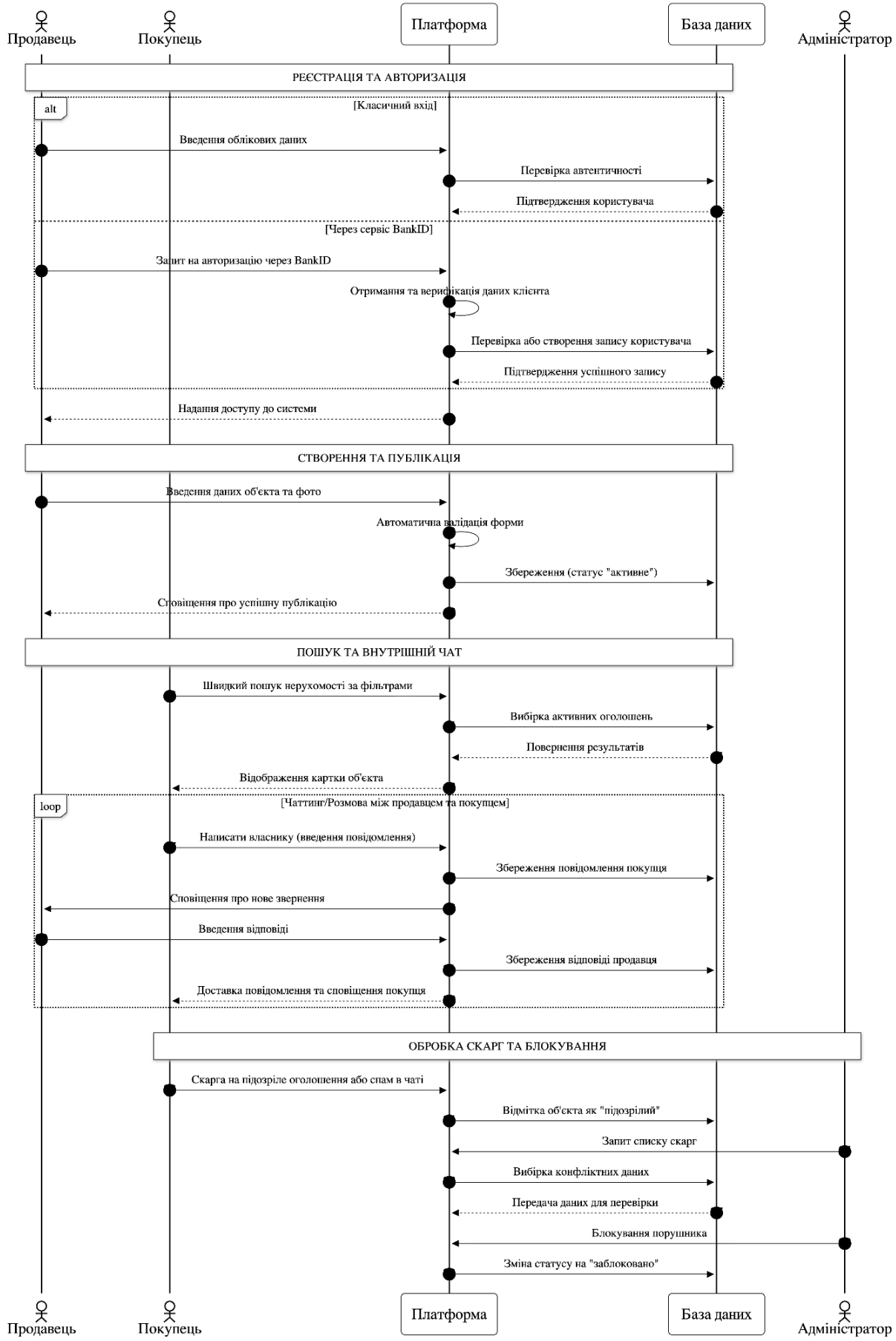

Рисунок 1 – Діаграма послідовностей (Sequence) взаємодії користувачів із веб-застосунком

Для наочного відображення функціональних вимог та взаємодії основних акторів із системою було розроблено діаграму послідовностей (Sequence) (див. рис.1). Вона деталізує архітектурну логіку та розподіл прав доступу: від базових можливостей багатопараметричної фільтрації та пошуку для неавторизованих клієнтів до розширеного функціоналу публікації з автоматичним захистом контенту для верифікованих власників.

Діаграма також ілюструє процеси безпечної внутрішньої комунікації (чат) та механізми модерації й обробки скарг, що підтверджує виконання ключової вимоги щодо забезпечення прямої взаємодії між сторонами угоди без залучення посередників.

Важливою нефункціональною вимогою до системи є забезпечення цілісності та унікальності даних. З цією метою розроблено програмний модуль динамічного маркування медіафайлів. У процесі публікації на кожну завантажену фотографію об'єкта система автоматично накладає алгоритмічний водяний знак (watermark), який містить верифікований контактний номер реального власника. Це робить несанкціоноване викрадення фотографій для використання на сторонніх ресурсах безкорисним та технічно унеможливлює приховування прямих контактів орендодавця недобросовісними агентами.

Клієнтська та візуальна частини застосунку розроблялися з урахуванням сучасних вимог до UI/UX-дизайну, з фокусом на максимальну ергономічність, зручність та високу сканованість (scannability) текстової інформації. Інтерфейс деталізованих карток нерухомості побудовано за принципом «перевернутої піраміди», де найважливіші параметри об'єкта (вартість, площа, локація) першочергово фокусують на собі увагу користувача, знижуючи когнітивне навантаження.

У результаті комплексної інтеграції цих модулів із серверною частиною було створено швидкий, безпечний та інтуїтивно зрозумілий інструмент. Реалізована система гарантує надійний зв'язок і захист даних на кожному етапі життєвого циклу оголошення: від моменту його публікації верифікованим власником до успішного знаходження та встановлення контакту кінцевим споживачем.

Окрему увагу під час проєктування системи було приділено питанням масштабування та стійкості до високих навантажень. Враховуючи специфіку сфери нерухомості (PropTech), де обсяги мультимедійних даних зростають експоненційно, архітектура застосунку передбачає чітке розмежування логіки обробки транзакцій та збереження статичного контенту.

Хоча основна бізнес-логіка та структуровані дані (користувачі, описи об'єктів, історія повідомлень) надійно обробляються у реляційній базі PostgreSQL, передбачено механізм делегування збереження медіафайлів із накладеними водяними знаками до спеціалізованих хмарних сховищ.

Крім того, асинхронна неблокуюча природа платформи Node.js дозволяє ефективно підтримувати велику кількість паралельних з'єднань, що є критично важливим фактором для забезпечення миттєвої доставки повідомлень у внутрішньому чаті системи. Такий комплексний архітектурний підхід не лише значно знижує навантаження на основний сервер, а й гарантує високу доступність сервісу (HighAvailability), закладаючи надійний технологічний фундамент для майбутньої інтеграції аналітичних модулів чи інструментів автоматизованої оцінки нерухомості.

Висновок

Створено сучасний і швидкий веб-застосунок для ринку нерухомості, орієнтований на пряму взаємодію без залучення посередників. Впроваджені алгоритми автоматичного накладання водяних знаків та сувора авторизація надійно захищають унікальність контенту. Інтерфейс, розроблений за принципом «перевернутої піраміди», гарантує високий рівень комфорту. Розроблена система готова до запуску як незалежна платформа для прямої та безпечної взаємодії.

Список використаних джерел

1. Moran K. The Inverted Pyramid: A Structure for Effective Web Writing. Nielsen Norman Group. URL:https://www.nngroup.com/articles/inverted-pyramid/https://www.nngroup.com/articles/inverted-pyramid/
2. Node.js Documentation. OpenJS Foundation. URL:https://nodejs.org/en/docs/https://nodejs.org/en/docs/
3. React: The library for web and nativ euser interfaces. MetaPlatforms, Inc. URL:https://react.dev/https://react.dev/
4. PostgreSQL: The World's Most Advanced Open Source Relational Database. Official Documentation. URL:https://www.postgresql.org/docs/https://www.postgresql.org/docs/
5. Система BankID Національного банку України. Офіційне інтернет-представництво НБУ. URL:https://id.bank.gov.ua/https://id.bank.gov.ua/
6. Закон України «Про захист персональних даних». Відомості Верховної Ради України. URL:https://zakon.rada.gov.ua/laws/show/2297-17https://zakon.rada.gov.ua/laws/show/2297-17

УДК 004.4

АРХІТЕКТУРНІ РІШЕННЯ СИСТЕМИ ДЛЯ ОРГАНІЗАЦІЇ КОМАНДНОЇ РОБОТИ

Крепич С.Я.[1)], **Гида І.Р.**[2)], **Співак І.Я.**[3)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] бакалавр; [3)]к.т.н., доцент

I. Постановка проблеми

Системи управління завданнями є ключовим інструментом організації командної роботи. Центральним об'єктом таких систем є дошка — робочий простір, що об'єднує колонки, картки завдань та учасників з різними правами доступу. Некоректна реалізація модуля створення дошок може призвести до порушення цілісності даних, зокрема до появи дошок без власника або некоректного розмежування прав доступу. Актуальним завданням є проєктування надійного модуля управління дошками з чіткою рольовою моделлю та атомарністю операцій.

II. Мета роботи

Метою роботи є проєктування структури сутностей модуля управління дошками системи «TaskTracker» та програмна реалізація функціоналу створення нової дошки з автоматичним призначенням ролі Owner засобами CleanArchitecture та патерну CQRS на платформі .NET 9.

III. Основна частина

Модуль управління дошками спроєктовано відповідно до принципів CleanArchitecture. Бізнес-логіка зосереджена у шарі Application, доменні сутності визначені у шарі Domain, а доступ до бази даних реалізовано через патернRepository у шарі Persistence. Для формалізації структури предметної області побудовано діаграму класів, що відображає ключові сутності модуля та зв'язки між ними.

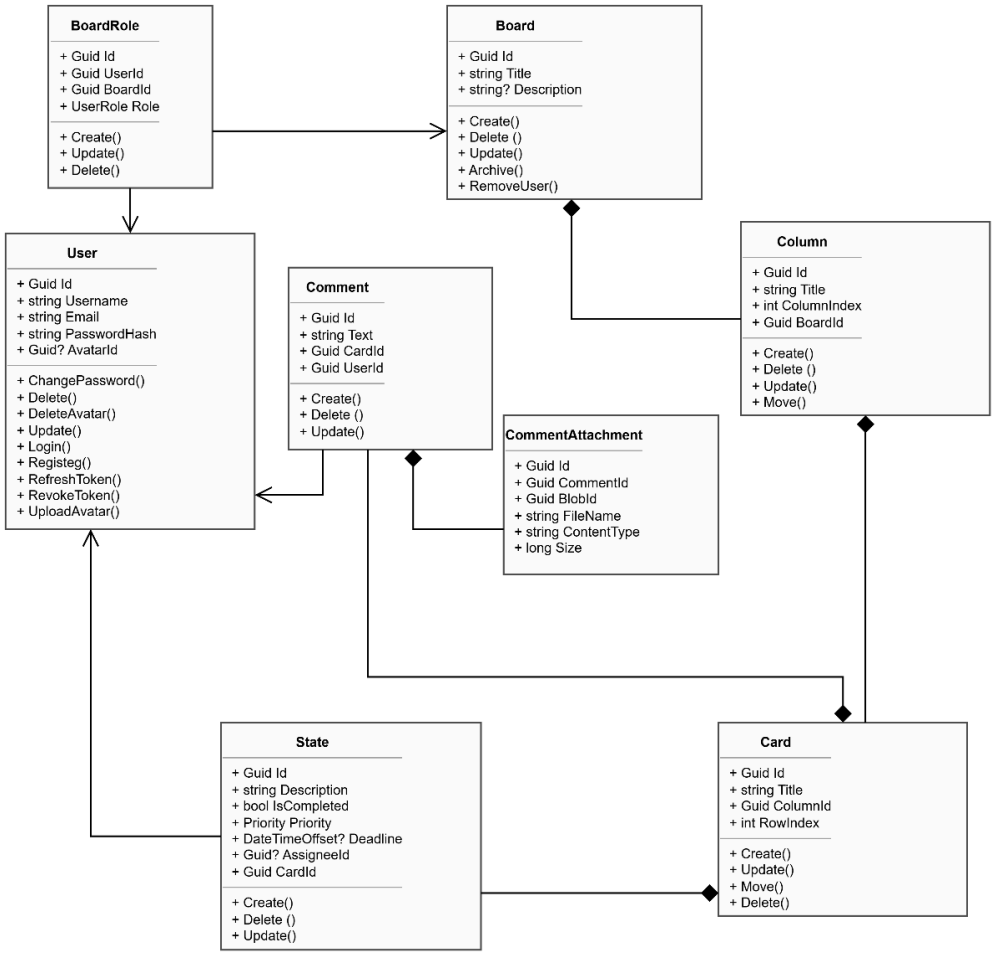

Рисунок 1 – Діаграма класів системи TaskTracker

Центральною сутністю модуля є клас Board, що містить атрибути Title, Description, CreatedBy, IsArchived та ArchivedAt. Зв'язок між користувачами та дошками реалізовано через проміжну сутність BoardRole, яка зберігає роль учасника (Owner, Admin, Member) для конкретної дошки. Така модель забезпечує гнучке розмежування прав доступу на рівні кожної окремої дошки.

Операція створення нової дошки реалізована у вигляді CQRS-команди CreateBoardCommand з відповідним обробником CreateBoardCommandHandler. Обробник виконує дві операції в межах однієї транзакції: створення запису дошки та автоматичне призначення ролі Owner користувачу, що ініціював створення.

Використання транзакції гарантує атомарність — неможливість появи дошки без власника у разі збою. Програмна реалізація наведена лістингом коду:

```
publicasyncTask<Guid>Handle(CreateBoardCommandrequest, CancellationTokencancellationToken)
{
usingvaruow = _unitOfWorkFactory.CreateUnitOfWork();
awaituow.BeginTransactionAsync();

try
    {
varboard = newDomain.Entities.Board
        {
Title = request.Title,
Description = request.Description,
CreatedBy = request.UserId.ToString()
        };

awaituow.Boards.CreateAsync(board, request.UserId, request.UserRole);
awaituow.SaveChangesAsync();

varboardRole = newDomain.Entities.BoardRole
        {
BoardId = board.Id,
UserId = request.UserId,
Role = request.UserRole
        };

awaituow.BoardRoles.AddAsync(boardRole);
awaituow.SaveChangesAsync();

awaituow.CommitTransactionAsync();
returnboard.Id;
    }
catch (Exceptionex)
    {
awaituow.RollbackTransactionAsync();
throw;
    }
}
```

Графічний інтерфейс модуля реалізований у компоненті BlazorWebAssembly (див.рис.2).

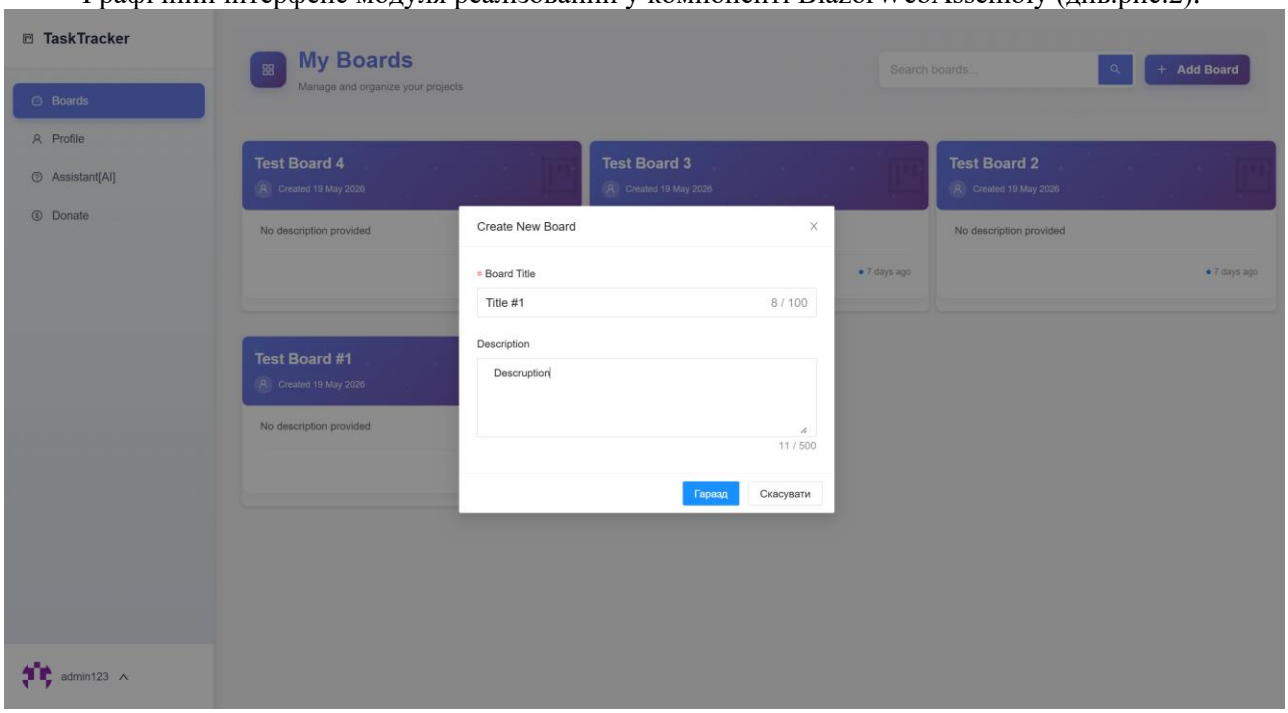

Рисунок 2 – Головна сторінка зі списком дошок та формою створення нової дошки

Висновок

У роботі спроєктовано структуру сутностей модуля управління дошками системи «TaskTracker» та реалізовано функціонал створення нової дошки. Застосування патерну CQRS дозволило чітко відокремити бізнес-логіку від рівня доступу до даних. Практична реалізація підтвердила відповідність обраної архітектури вимогам надійності та розширюваності системи.

Список використаних джерел

1. Документація AntDesignBlazor [Електронний ресурс]. – Режим доступу: https://antblazor.com/en-US/docs/introduce
2. Документація бібліотеки MediatR [Електронний ресурс]. – Режим доступу: https://github.com/jbogard/MediatR

УДК 005.8:004.4

ВЕБ-ОРІЄНТОВАНА СИСТЕМА УПРАВЛІННЯ ПРОЄКТАМИ ТА ЗАДАЧАМИ

Войтюк І.Ф.[1)], **Ярош Б.Я.**[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] бакалавр

І. Постановка проблеми та аналіз існуючих рішень

Сучасний етап розвитку IT-індустрії вимагає від інженерних команд високої оперативності та прозорості в управлінні процесами розробки програмного забезпечення. Ефективність реалізації проєктів безпосередньо залежить від злагодженості роботи команди, контролю витраченого часу та своєчасного виявлення технічних ризиків. Проте на багатьох підприємствах спостерігається проблема фрагментації інструментарію: використовуються розрізнені системи для постановки задач, ведення документації та відстеження робочого часу.

Аналіз існуючих рішень дозволяє виділити популярні системи управління проєктами, зокрема Jira від компанії Atlassian та Trello. Незважаючи на їхню потужність, вони мають певні недоліки для невеликих та середніх інженерних команд:

- Високий поріг входження: Jira є індустріальним стандартом, але її інтерфейс перевантажений специфічними налаштуваннями, які є надмірними для невеликих проєктів та уповільнюють щоденну оперативну роботу; крім того, вартість ліцензій робить її економічно недоцільною для стартапів та приватних підприємців (ФОП).

2. Обмеженість візуалізації планування: розширені функції типу «Таймлайн» (Timeline) у Trello доступні лише за платною підпискою, що ускладнює довгострокове стратегічне планування.

2. Слабка модель контролю бізнес-процесів: у Trello будь-який учасник може самостійно перевести завдання у статус «Готово» без попередньої перевірки коду чи тестування, що унеможливлює впровадження суворої рольової моделі доступу.

Актуальність даного дослідження зумовлена необхідністю розробки власної веб-орієнтованої системи управління проєктами та задачами, яка б поєднувала інтуїтивно зрозумілий інтерфейс із багаторівневою рольовою моделлю доступу (RBAC), вбудованим логуванням часу та модулем аналізу проєктних ризиків.

ІІ. Мета роботи

Метою роботи є підвищення ефективності управління процесами розробки ПЗ шляхом створення веб-орієнтованої системи управління проєктами та задачами з вбудованими механізмами моніторингу часу, візуалізації планування та модулем реєстру проєктних ризиків. Для досягнення мети вирішено такі завдання: обґрунтовано вибір стеку технологій (.NET, Blazor WebAssembly); спроєктовано реляційну базу даних; розроблено серверну та клієнтську частини із розмежуванням прав доступу; впроваджено інтерактивні модулі візуалізації (Kanban-дошка та Таймлайн).

ІІІ. Моделювання процесів управління проєктами

Для вирішення проблеми фрагментації даних розроблено систему комплексного життєвого циклу управління завданнями, що базується на концепції взаємопов'язаних модулів, що охоплюють усі етапи роботи інженерної команди:

1. Модуль безпеки та доступу (Role-Based Access Control). Система моделює п'ять ролей користувачів (Admin, Manager, Developer, Tester, CodeReview), гарантуючи, що переведення завдання у фінальний статус «Готово» (Done) виконується виключно роллю CodeReview після незалежної перевірки коду.
2. Модуль стратегічного планування (Timeline). Інтерактивний Таймлайн виконує функцію діаграми Ганта та візуалізує розподіл завдань у часі на основі часових міток StartDate та Deadline, що мінімізує ризики перетину критичних дедлайнів.
3. Модуль оперативного управління (Kanban). Процес трансформації стратегічних цілей у конкретні задачі (TaskItem), які проходять життєвий цикл із чотирьох статусів: ToDo, InProgress, InReview, Done. Зміна статусів реалізована через механізм Drag-and-Drop із миттєвою синхронізацією з базою даних через асинхронні HTTP-запити.
4. Процес логування робочого часу (Worklog). Окрема сутність, яка дозволяє кожному інженеру фіксувати фактично витрачений час на виконання конкретної задачі та формувати основу для аналізу використання ресурсів.

Для надійного збереження інформації спроєктовано реляційну базу даних, яка відповідає вимогам третьої нормальної форми (3NF). Взаємодія із СКБД реалізована засобами ORM-фреймворка Entity Framework Core за підходом Code-First, за якого структура таблиць автоматично генерується з класів-сутностей через механізм міграцій. Основні сутності системи наведено в Таблиці 1.

Таблиця 1

Основні сутності реляційної бази даних розробленої системи

Сутність	Призначення та ключові атрибути
Project	Центральний агрегатор робочої області; містить Id, Name, KeyWord та Description.
TaskItem	Атомарна одиниця роботи зі статусом та часовими мітками StartDate і Deadline.
User	Обліковий запис із полем Role, що визначає рівень доступу в системі.
Risk	Сутність модуля ризиків із оцінками Probability та Impact для конкретного проєкту.
Worklog	Журнал робочого часу (TimeSpent, LogDate), пов'язаний із завданням та користувачем.

IV. Архітектура та програмна реалізація

Для забезпечення високої швидкодії спроєктовано оптимізовану, слабко пов'язану клієнт-серверну архітектуру, що складається з чотирьох логічних проєктів: незалежного ядра, рівня доступу до даних, серверного інтерфейсу та клієнтського SPA-додатка.

Серверну частину побудовано на платформі ASP.NET Core Web API з використанням мови C#, а клієнтську – як Single Page Application на базі технології Blazor WebAssembly із бібліотекою компонентів MudBlazor. Безпека реалізована через безстановий (stateless) протокол авторизації на основі токенів JSON Web Token (JWT): після входу користувач отримує підписаний токен із його роллю, який додається до заголовків кожного запиту. На основі ролей захищено кінцеві точки Web API, а клієнтський інтерфейс динамічно адаптується за допомогою кастомного провайдера стану аутентифікації, приховуючи заборонені елементи керування від користувачів без відповідних привілеїв.

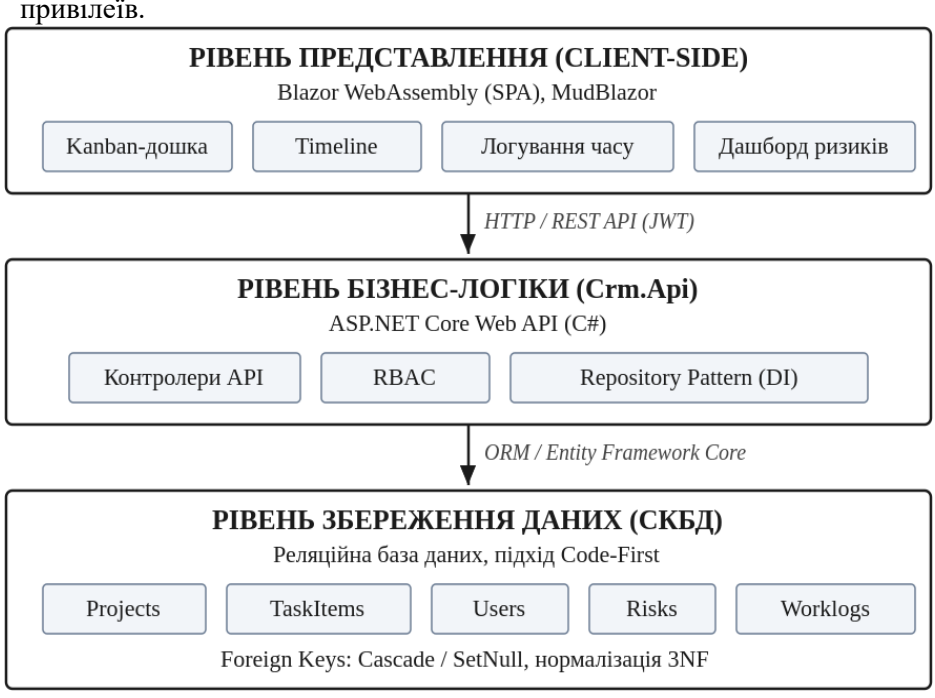

Рисунок 1 – Схема багаторівневої архітектури розробленої системи

Висновок

У роботі запропоновано та обґрунтовано підхід до створення інтегрованої веб-орієнтованої системи управління проєктами та задачами, адаптованої до потреб інженерних команд. Результат статті полягає в реалізації моделі управління проєктними процесами, яка поєднує інструменти оперативного управління задачами, планування через Таймлайн та вбудовані механізми логування робочого часу із розмежуванням прав доступу на основі ролей. Практична цінність полягає у розробці багаторівневої архітектури та повнофункціонального програмного продукту, готового до впровадження в реальних IT-командах та у приватних підприємців.

Список використаних джерел

1. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 7th ed. – Project Management Institute, 2021. – 250 p.
2. ISO 31000:2018. Risk management — Guidelines. International Organization for Standardization, 2018.
3.Jones M. JSON Web Token (JWT) / M. Jones, J. Bradley, N. Sakimura [Електронний ресурс] // Internet Engineering Task Force (IETF). – RFC 7519. – 2015. – Режим доступу до ресурсу: https://datatracker.ietf.org/doc/html/rfc7519

УДК 681.3

РОЗРОБКА ВЕБ-ДОДАТКУДЛЯ ФОРМУВАННЯ ЗБАЛАНСОВАНОГО МЕНЮ

Крепич С.Я.[1)], Клепас Д.В.[2)], Співак І.Я.[3)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] бакалавр; [3)] к.т.н., доцент

I. Постановка проблеми

Збалансоване харчування є важливою складовою підтримки здоров'я людини та профілактики багатьох захворювань. Проте процес складання раціону харчування потребує врахування значної кількості факторів, серед яких калорійність страв, вміст білків, жирів і вуглеводів, індивідуальні особливості користувача та його фінансові можливості. Більшість сучасних програмних рішень дозволяють лише вести облік спожитих продуктів або переглядати рецепти, але не забезпечують автоматизоване формування меню з урахуванням бюджету та необхідних поживних речовин.

Аналіз існуючих систем показав, що відсутні програмні продукти, які б одночасно враховували харчову цінність продуктів, вартість інгредієнтів та можливість автоматичного створення меню на визначений період часу. Це обумовлює актуальність розробки спеціалізованого веб-додатку для вирішення даної задачі.

II. Мета роботи

Метою роботи є дослідження сучасних підходів до автоматизованого планування раціонального харчування та розробка веб-додатку для формування збалансованого меню з урахуванням індивідуальних потреб користувача, фінансових обмежень і тривалості планування. Для досягнення поставленої мети необхідно виконати аналіз існуючих програмних рішень у сфері планування харчування, визначити функціональні та нефункціональні вимоги до системи, розробити структуру бази даних для зберігання інформації про продукти, рецепти та користувачів, а також реалізувати алгоритм автоматичного підбору страв на основі показників харчової цінності та вартості продуктів. Крім того, метою роботи є забезпечення зручного доступу користувачів до сформованого меню та супровідної інформації щодо необхідних продуктів для його приготування.

III. Особливості програмної реалізації системи

Структуру розробленого веб-додатку подано на рисунку 1. В основу програмної реалізації покладено компонентний підхід, який забезпечує розподіл функціональності між окремими модулями та спрощує подальший супровід і розширення системи.

Центральним елементом системи є модуль формування збалансованого меню, який реалізує алгоритм підбору страв відповідно до заданих користувачем параметрів. Вхідними даними для роботи алгоритму є інформація про користувача, обраний період планування, бюджетні обмеження, а також відомості про продукти та рецепти, що зберігаються в базі даних. Результатом роботи модуля є сформований набір страв на визначений період із розрахунком вартості та харчової цінності.

Компонент керування користувачами забезпечує реєстрацію, авторизацію та зберігання персональних даних. Додатково користувач може вказувати параметри, необхідні для формування меню, зокрема особливості харчування та інші індивідуальні характеристики.

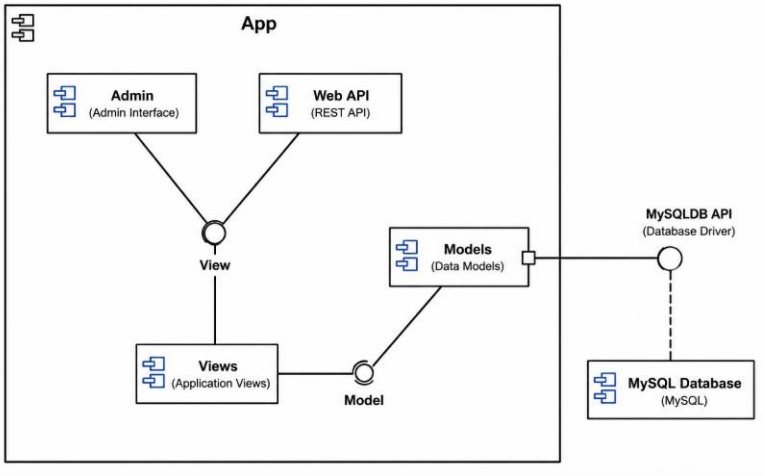

Рисунок 1 – Діаграма компонентів системи

Для зберігання інформації використовується база даних, яка містить відомості про продукти, рецепти, кухонне приладдя, магазини та ціни на товари. Така структура дозволяє виконувати швидкий пошук необхідної інформації та забезпечує цілісність даних під час роботи системи.

Окремий компонент призначений для керування каталогом рецептів. Він забезпечує додавання, редагування та перегляд рецептів, а також зберігання інформації про склад страв і необхідні інгредієнти. Взаємодія цього компонента з модулем формування меню дозволяє автоматично підбирати страви відповідно до встановлених критеріїв.

Для реалізації веб-застосунку використано архітектурний підхід MVT (Model–View–Template), який забезпечує розділення логіки обробки даних, користувацького інтерфейсу та механізмів доступу до бази даних. Такий підхід підвищує масштабованість програмного забезпечення та спрощує процес його модернізації.

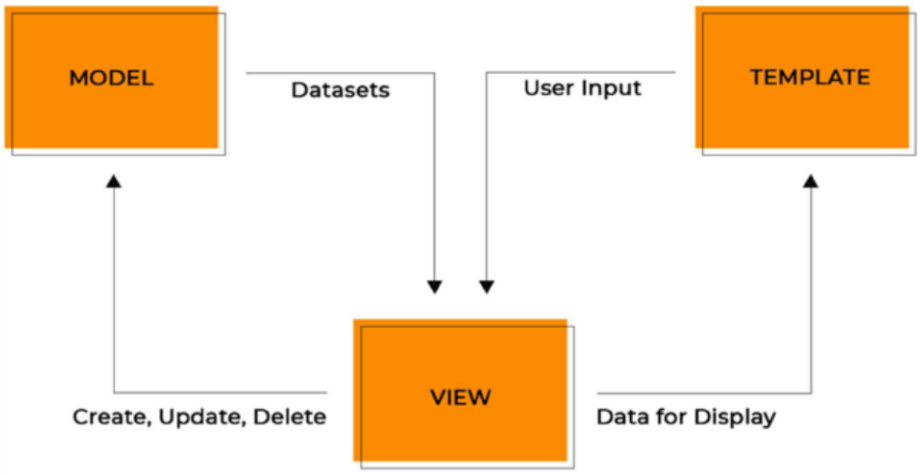

Рисунок 2 – Архітектура Model–View–Template

Компонент Model відповідає за роботу з даними та взаємодію з базою даних. У цьому модулі зберігається інформація про користувачів, продукти, рецепти, магазини та ціни на товари. Саме модель забезпечує створення, оновлення, видалення та отримання даних, необхідних для роботи системи.

Компонент View виконує роль посередника між користувацьким інтерфейсом та даними системи. Він приймає запити користувача, обробляє їх, звертається до моделей для отримання необхідної інформації та формує результати для подальшого відображення. Через даний компонент реалізовано формування збалансованого меню відповідно до заданих параметрів користувача.

Компонент Template забезпечує відображення інформації у веб-інтерфейсі. Він отримує підготовлені дані від View та формує сторінки застосунку, що відображають результати роботи алгоритму, інформацію про рецепти, продукти та сформоване меню.

Запропонована компонентна структура забезпечує модульність системи, можливість незалежного розвитку окремих підсистем та ефективну взаємодію між усіма складовими веб-додатку.

Висновок

У роботі розглянуто проблему автоматизованого формування збалансованого меню та проаналізовано існуючі програмні рішення у даній предметній області. Розроблено концепцію веб-додатку, який дозволяє формувати меню відповідно до заданих користувачем параметрів та бюджетних обмежень. Запропоноване рішення сприяє підвищенню ефективності планування харчування та може бути використане як основа для створення сучасних інформаційних систем підтримки здорового способу життя.

Список використаних джерел

1. Порівняння цін у супермаркетах України [Електронний ресурс] – Режим доступу до ресурсу: https://price.ua/ua
2. Офіційний сайт Python [Електронний ресурс] – Режим доступу до ресурсу: https://www.python.org/
3. Документація web-фреймворку Django [Електронний ресурс] – Режим доступу до ресурсу: https://docs.djangoproject.com/en/4.0
4. MVT архітектура Django [Електронний ресурс] – Режим доступу до ресурсу: https://www.onlinetutorialspoint.com/django/django-model-view-template-mvt-overview.html
5. S.Krepych, I.Spivaak, “Model of Human weight correction based on interval Data Analysis”, Int.J.Comput. T.19. 2020. – pp.128-136

УДК 004.415

ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ КОРПОРАТИВНОЇ HRM-СИСТЕМИ З АНАЛІТИЧНИМ МОДУЛЕМ

Шпінталь М.Я.[1)], Василів Д.Л.[2)]
Західноукраїнський національний університет
[1)]к.т.н.,доцент; [2)] бакалавр

I. Постановка проблеми

Сучасний етап розвитку інформаційного суспільства та корпоративного управління характеризується тотальною цифровізацією та переходом до концепції Data-Driven (управління на основі даних). Розробка сучасних систем управління людськими ресурсами (HRM) сьогодні вимагає не лише автоматизації рутинних адміністративних функцій (таких як облік робочого часу, бронювання відпусток чи ведення карток працівників), а й створення потужних, інтегрованих аналітичних рішень, здатних обробляти великі масиви даних у реальному часі. Проте при проєктуванні таких систем розробники та бізнес-аналітики часто стикаються з проблемою архітектурної розрізненості. На ринку існує безліч вузькоспеціалізованих рішень: окремо для рекрутингу, окремо для розрахунку заробітної плати та окремо для оцінки ефективності (Performance Review). Виникає критична необхідність узагальнення методологічних підходів до створення єдиної екосистеми, яка поєднує в собі складну бізнес-логіку управління кадрами, гнучкість користувацьких інтерфейсів та обчислювальну потужність аналітичних моніторів для вищого керівництва.

II. Мета роботи

Метою роботи є систематизація методів проєктування та безпосередня програмна реалізація веборієнтованої інформаційної HRM-системи, що містить глибоко інтегрований аналітичний модуль. Для досягнення цієї мети було поставлено та вирішено такі завдання:

Провести аналіз предметної області та визначити недоліки існуючих систем управління персоналом.

- Спроектувати архітектуру клієнт-серверного додатку, дивись рисунок, та розробити реляційну модель бази даних, що відповідає вимогам масштабованості.
- Здійснити візуальне об'єктно-орієнтоване моделювання бізнес-процесів за допомогою інструментарію UML.
- Реалізувати програмні модулі бекенду та фронтенду з використанням сучасного технологічного стеку, повністю уникаючи прямої залежності від монолітних структур.
- Впровадити алгоритми розрахунку аналітичних показників (KPI, плинність кадрів).

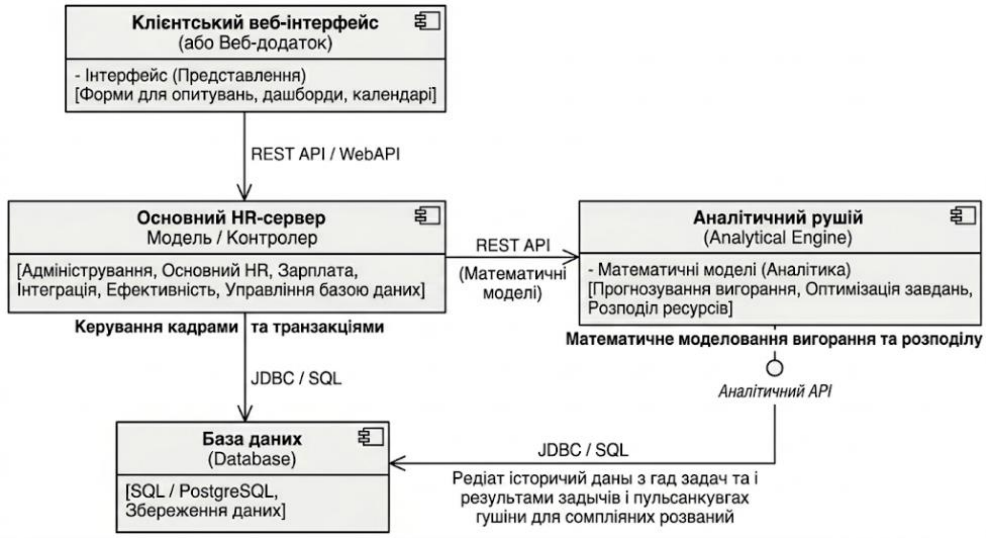

Рисунок 1 – Загальна архітектура системи

III. Основні результати дослідження

У ході комплексного дослідження та практичної розробки було детально спроектовано та програмно реалізовано стратегічні вектори роботи корпоративної системи, кожен з яких вирішує специфічні задачі цифровізації HR-процесів:

1. Модуль самообслуговування персоналу (Employee Portal).

Досліджено архітектуру інтерфейсів, орієнтованих на кінцевого співробітника. Ключовим результатом стало проєктування оптимального робочого простору (Dashboard), що включає доступ до особистої інформації, управління відсутністю (заяви на відпустки, лікарняні) та тренінгами. Логіка погодження відпусток реалізована за принципом скінченного автомата (State Machine), що автоматизує рух заявки від статусу «На розгляді» до «Затверджено», мінімізуючи бюрократичні затримки.

2. Модуль оцінки ефективності (Performance Review).

Визначено, що критичним аспектом таких систем є об'єктивність та прозорість критеріїв. Програмно реалізовано інструменти для постановки цілей (OKR) та збору результатів оцінювання (360-degree feedback). Алгоритм системи автоматично зважує отримані оцінки від керівника, самооцінку та результати проходження корпоративних курсів, формуючи єдиний стандартизований індекс ефективності (KPI) для кожного працівника.

3. Системи real-time моніторингу та аналітики (HR Analytics).

Розроблено архітектуру незалежного аналітичного модуля для збору та обробки даних з усіх підрозділів компанії. Реалізовано миттєву візуалізацію складних бізнес-метрик на динамічних дашбордах для керівництва. Модуль виконує агрегацію даних для розрахунку:

Коефіцієнта плинності кадрів (Turnover Rate) з розбивкою по відділах та місяцях.

Динаміки зміни загального фонду оплати праці у співвідношенні до зростання штату.

Кореляції між витратами на навчання персоналу та зростанням їхньої операційної продуктивності. Це дає змогу адміністраторам та HR-директорам оперативно виявляти аномалії в робочих процесах та аналізувати тенденції всередині колективу в реальному часі.

4. Об'єктно-орієнтоване та візуальне моделювання.

Унікальною особливістю розробки є строга відмова від використання лістингів вихідного коду для документування логіки на користь візуальних абстракцій. Спільним для всіх підсистем є використання об’єктно-орієнтованого моделювання.

Для опису функціональних вимог та прав доступу на основі ролей (RBAC) найефективнішим інструментом визначено UML-діаграму варіантів використання (Use Case Diagram), яка чітко розмежовує сценарії взаємодії рядового працівника, керівника відділу та системного адміністратора.

Внутрішня архітектура та структура бази даних спроектовані за допомогою UML-діаграми класів (Class Diagram) та ER-моделей, що описують зв’язки між основними сутностями: Користувачами, Відділами, Запитами на відпустку та Аналітичними метриками.

5. Технологічний стек та архітектурні рішення. Інтеграція вищезгаданих векторів дозволила сформувати концепцію надійної веборієнтованої системи, яка поєднує в собі продуктивність, безпеку та високу швидкість обробки даних. Практична реалізація розглянутих підсистем базується на сучасному та масштабованому стеку технологій:

Серверна частина (Back-end) побудована на платформі Node.js з використанням фреймворку NestJS та мови TypeScript, що забезпечує строгу типізацію та модульну архітектуру.

Клієнтська частина (Front-end) реалізована за допомогою бібліотеки React.js для створення динамічних, реактивних інтерфейсів.

База даних функціонує під управлінням СУБД PostgreSQL, яка є стандартом для збереження складних фінансових та корпоративних даних, гарантуючи транзакційну цілісність (ACID).

Висновок

Узагальнення досвіду розробки цих рішень дозволило сформувати єдиний, комплексний підхід до побудови гнучких корпоративних веб-систем. Розроблена HRM-система успішно вирішує проблему фрагментації кадрового програмного забезпечення. Впроваджений аналітичний модуль стає тим самим «інтелектуальним ядром», що допомагає оперативно коригувати бізнес-стратегію компанії на основі достовірних математичних даних про стан персоналу, ефективність їхньої роботи та розподіл бюджетів. Використання виключно графічних методів UML для проєктування системи довело свою ефективність у забезпеченні прозорості архітектурних рішень без необхідності прив'язки до конкретних фрагментів програмного коду. Створений продукт володіє високим потенціалом для подальшого масштабування та комерційного впровадження.

Список використаних джерел

1. Армстронг М. Практика управління людськими ресурсами. 14-те вид. Харків: Фабула, 2021. 864 с.
2. Соммервілл І. Інженерія програмного забезпечення. 10-те вид. Бостон: Pearson, 2018. 816 с.
3. Фаулер М. Архітектура корпоративних програмних додатків. Київ: ВД "Вільямс", 2020. 544 с.

УДК 004.41

АВТОМАТИЗАЦІЯ УПРАВЛІННЯ ВОЛОНТЕРСЬКОЮ ДІЯЛЬНІСТЮ В ГРОМАДСЬКИХ ОРГАНІЗАЦІЯХ ЗА ДОПОМОГОЮ CRM-СИСТЕМ

Задорожний М.Б.[1)], Крепич С.Я.[2)], Співак І.Я.[3)]
Західноукраїнський національний університет
[1)]бакалавр;[2-3)] к.т.н., доцент

Постановка проблеми

Громадські організації відіграють важливу роль у розвитку громадянського суспільства, реалізації соціальних, освітніх, благодійних та волонтерських проєктів. Вони сприяють вирішенню суспільно важливих проблем, залучають громадян до активної громадської діяльності, організовують соціальні ініціативи та забезпечують підтримку різних категорій населення.

Для ефективного виконання своїх завдань громадські організації взаємодіють із волонтерами, партнерами, донорами, навчальними закладами та іншими зацікавленими сторонами, що супроводжується постійним накопиченням значних обсягів інформації.

Одним із можливих шляхів вирішення зазначених проблем є впровадження спеціалізованої CRM-системи, яка забезпечить централізоване зберігання інформації, автоматизацію основних бізнес-процесів та зручний доступ до даних для всіх уповноважених користувачів. Використання такого програмного рішення дозволить підвищити ефективність управління проєктами, покращити взаємодію між учасниками організації та забезпечити більш якісний контроль над її діяльністю.

Мета

Метою роботи є розроблення універсальної CRM-системи для громадських організацій, яка забезпечить централізоване управління проєктами, завданнями, волонтерами, організаціями та взаємодіями із зацікавленими сторонами.

Система повинна автоматизувати основні процеси діяльності громадських організацій, спростити координацію роботи команд та підвищити ефективність управління інформацією.

Основний виклад

Запропоноване рішення являє собою веборієнтовану CRM-систему, побудовану на основі клієнт-серверної архітектури. Для реалізації клієнтської частини використано фреймворк Next.js та мову TypeScript, а серверна частина реалізована за допомогою платформи Supabase і системи керування базами даних PostgreSQL.

Основними функціональними модулями системи є:

- Реєстрація та авторизація користувачів. Система забезпечує створення облікових записів та автентифікацію користувачів із використанням електронної пошти та пароля.
- Управління проєктами. Користувачі можуть створювати проєкти, редагувати їх параметри, встановлювати статуси виконання та переглядати інформацію про поточну діяльність організації.
- Управління завданнями. Передбачено створення завдань, призначення відповідальних осіб, встановлення пріоритетів та контроль виконання робіт у межах проєктів.
- Управління волонтерами. Система дозволяє вести облік волонтерів, переглядати їх контактну інформацію та використовувати єдину базу даних учасників організації.
- Управління організаціями та учасниками організацій. Реалізовано можливість зберігання інформації про організації, їх учасників та ролі користувачів у межах організації.
- CRM-взаємодії. Передбачено механізм обліку контактів, партнерів та історії взаємодій із зовнішніми стейкхолдерами. Це дозволяє накопичувати інформацію про співпрацю з організаціями, освітніми установами, партнерами та потенційними донорами.
- Аналітика та звітність. Система забезпечує формування звітів щодо проєктної діяльності, виконання завдань та роботи з контактами.

На рисунку 1 наведено діаграму варіантів використання CRM-системи, яка відображає основні сценарії взаємодії користувачів із системою.

Розроблена система забезпечує єдине інформаційне середовище для громадських організацій та дозволяє автоматизувати ключові процеси управління проєктною діяльністю.

Рисунок 1 – Діаграма варіантів використання CRM-системи

Висновок

У роботі розроблено універсальну CRM-систему для громадських організацій, яка дозволяє автоматизувати процеси управління проєктами, завданнями, волонтерами, організаціями та CRM-взаємодіями. Використання сучасних вебтехнологій та хмарних сервісів забезпечує зручність використання системи, централізоване зберігання інформації та можливість подальшого розширення функціональних можливостей.

Практична цінність розробки полягає у підвищенні ефективності діяльності громадських організацій за рахунок автоматизації рутинних процесів, покращення контролю виконання завдань та спрощення взаємодії між учасниками організації. Розроблений програмний продукт може бути використаний як основа для подальшого розвитку інформаційних систем управління діяльністю громадських організацій та волонтерських об’єднань.

Список використаних джерел

1. ISO/IEC 25010:2023 Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE).
2. S.Krepych, I.Spivak, “Forecasting system of utilities service costs besed on neural network”, Advanced Information Systems. 2020. Vol.4. №4. pp/102-108
3. Крепич С.Я, Метод синтезу смугового фільтра для заданих обмежень на його модуль коефіцієнта передачі. Сучасні комп’ютерні інформаційні технології: Матеріали IV Всеукраїнської школи-семінару молодих вчених і студентів ACIT’2014. – Тернопіль: Економічна думка, 2014. – С.26-29

УДК 004.42

РОЗРОБКА АРХІТЕКТУРИ 2D-ГРИ ЖАНРУ ROGUELIKE НА БАЗІ ІГРОВОГО РУШІЯ UNITY

Стецюцяк О.А.[1)], **Стасів І.С.**[2)]
Західноукраїнський національний університет
[1)]студентка; [2)]к.т.н., доцент

І. Постановка проблеми

Сучасна індустрія розробки відеоігор характеризується високими вимогами до продуктивності та інтенсивністю використання обчислювальних ресурсів. Особливе місце посідають проєкти жанру Roguelike, де основою ігрового процесу є процедурна генерація контенту та безмежність ігрового світу. Згідно з аналізом розробки інді-проєктів, значна частка ігор стикається з проблемами низької частоти кадрів та надмірного споживання оперативної пам’яті через некоректне управління великою кількістю об’єктів у сцені. Ключовим фактором успішності такої гри є ефективна архітектура та впровадження механізмів динамічної оптимізації. Відсутність систем автоматизованого керування життєвим циклом об’єктів у реальному часі призводить до нестабільної роботи додатку на пристроях з обмеженими ресурсами. Актуальність дослідження зумовлена необхідністю створення гнучкої програмної системи на базі рушія Unity, яка здатна автоматично генерувати ігрове оточення у міфологічному сеттингу та забезпечувати високу продуктивність через алгоритми інтелектуального відсікання об’єктів на основі координат камери.

ІІ. Мета роботи

Метою роботи є розробка архітектури та програмна реалізація 2D-гри жанру Roguelike у міфологічному сеттингу, а також підвищення ефективності рендерингу та управління пам’яттю шляхом розробки методу динамічної процедурної генерації ігрового простору та алгоритмів автоматизованого відсікання невидимих об'єктів на базі ігрового рушія Unity.

ІІІ. Математична модель оцінювання ризиків

Для формалізації процесу управління ресурсами та стабілізації частоти кадрів пропонується використовувати модель динамічного оцінювання дистанції відсікання об'єктів. Відповідно до принципів комп'ютерної графіки, навантаження на систему визначається як функція від кількості активних ігрових об'єктів у сцені та їхньої віддаленості від камери [1, 2].

Інтегральний показник навантаження L_{total} розраховується за формулою (1), яка дозволяє агрегувати дані про активні чанки та декорації в єдиний індикатор стану продуктивності.

$$L_{total} = \sum_{i=1}^{n} w_i \cdot K_i \cdot N_i \qquad (1)$$

де n – загальна кількість активних модулів карти, K_i – нормалізований коефіцієнт навантаження i-го типу чанка (наприклад, щільність декорацій від 0 до 1), w_i – ваговий коефіцієнт важливості чанка (залежить від того, чи знаходиться він у полі зору), N_i – коефіцієнт складності об'єктів (кількість полігонів або активних скриптів).

Такий підхід дозволяє автоматизувати процес оптимізації. Система аналізує вхідні координати об'єктів, обчислює відстань і порівнює її з допустимими межами. Якщо розраховане значення перевищує встановлений поріг ($dist^2 > cullDistanceSqr$), система ініціює метод HandleChunkCulling, деактивуючи об'єкт.

Для формування профілю продуктивності виділено базовий набір метрик, що впливають на стабільність ігрового процесу. Опис базових показників наведено в Таблиці 1.

Таблиця 1

Базові показники ефективності системи процедурної генерації та оптимізації

Назва показника	Позначення	Опис та вплив на продуктивність
Відхилення позиції (MoveDelta)	K_1	Різниця між поточною та попередньою позицією камери. Визначає необхідність перерахунку сітки.
Щільність об'єктів (PropDensity)	K_2	Кількість декорацій на чанк. Високе значення підвищує навантаження на рендеринг.

Продовження таблиці 1

Дистанція відсікання (CullDistance)	K_3	Радіус видимості об'єктів. Збільшення підвищує навантаження, але покращує візуальний досвід.
Час кадру (FrameTime)	K_4	Час, витрачений на обробку одного кадру. Зростання вказує на необхідність агресивної оптимізації.

Оскільки вхідні дані мають різні одиниці виміру (метри, мілісекунди, штуки), необхідним етапом є їх нормалізація до єдиної шкали [0,1]. Розрахунок нормалізованого значення K_i здійснюється за формулою (2):

$$K_i = \begin{cases} 0, & якщо \quad x \le x_{\min} \\ \dfrac{x - x_{\min}}{x_{\max} - x_{\min}}, & якщо \quad x_{\min} < x < x_{\max} \\ 1, & якщо \quad x \ge x_{\max} \end{cases} \qquad (2)$$

де x – поточна дистанція до об'єкта, $x_{\min}$ – радіус зору камери (де об'єкт 100% активний), $x_{\max}$ – критична дистанція, при якій об'єкт повинен бути видалений через Destroy() для звільнення оперативної пам'яті.

IV. Архітектура та програмна реалізація

Для апробації запропонованого методу розроблено архітектуру ігрової системи, яка базується на трирівневій структурі: модулі процедурної генерації, аналітичного ядра оптимізації та інтерфейсу користувача. Обмін даними між компонентами реалізовано за допомогою внутрішньої системи подій Unity та механізму ScriptableObjects, що дозволяє гнучко налаштовувати параметри об’єктів оточення без зміни програмного коду. Дані про стан ігрового світу експортуються у форматі класів-контейнерів, після чого аналітичний модуль обробляє їх згідно з наведеним вище алгоритмом відсікання. Візуалізація результатів та поточної статистики (рівень, кількість зібраного досвіду, активні предмети) здійснюється через графічний інтерфейс, побудований на системі Unity UI.Важливим етапом впровадження методу є процедура калібрування вагових коефіцієнтів w_i. Для цього передбачено використання профілювання продуктивності на етапі тестування. Порівнюючи розраховані системою значення навантаження L_{total} із фактичною частотою кадрів та витратами пам’яті, можна оптимізувати ваги метрик (дистанція, кількість полігонів, активні скрипти). Це дозволяє адаптувати математичну модель під специфіку конкретних ігрових пристроїв, враховуючи їхні апаратні обмеження. Для оптимізації вагових коефіцієнтів застосовуються методи статистичного аналізу результатів тестування та ітераційне корегування параметрів об’єктів оточення. Практична цінність розробленого підходу полягає у мінімізації впливу людського фактора на процес наповнення рівнів. На відміну від ручного розміщення об'єктів, яке є трудомістким та суб’єктивним, запропонований підхід забезпечує безперервний моніторинг стану сцени та автоматичне керування ресурсами. Це дозволяє своєчасно виявляти та усуватизагрози продуктивності, такі як надмірне накопичення активних об’єктів поза межами видимості, що значно підвищує стабільність ігрового процесу та прогнозованість додатка.

Висновок

У роботі запропоновано метод динамічного управління ігровим середовищем та оптимізації продуктивності у 2D-іграх жанру Roguelike, що базується на аналізі просторових метрик та процедурній генерації контенту. Розроблена програмна архітектура на базі рушія Unity забезпечує можливість автоматизованого створення ігрових рівнів у міфологічному сеттингу та впровадження механізмів інтелектуального відсікання невидимих об'єктів у режимі реального часу. Це дозволяє суттєво зменшити навантаження на обчислювальні ресурси та підвищити стабільність частоти кадрів на пристроях з різними апаратними характеристиками.

Подальші дослідження спрямовані на вдосконалення алгоритмів процедурної генерації з використанням складніших математичних моделей для створення більш розгалужених лабіринтів, а також на розширення набору метрик для динамічного калібрування складності ігрового процесу залежно від дій гравця.

Список використаних джерел

1. UnityScripting API: officialdocumentation [Electronicresource]. – Access mode: https://docs.unity3d.com/ScriptReference/
2. Gregory J. GameEngineArchitecture. – 3rd ed. – BocaRaton : CRC Press, 2018. – 1162 p.

УДК 004.4

РОЗРОБКА ВЕБ-ЗАСТОСУНКУ ДЛЯ СПОРТИВНИХ ЛЮБИТЕЛІВ

Галайко Т.А.[1)], Крепич С.Я.[2)], Співак І.Я.[3)]
Західноукраїнський національний університет
[1)] бакалавр; [2-3)] к.т.н., доцент

I. Постановка проблеми та актуальність

У сучасних умовах цифровізації спортивний контент перетворився на один із найбільш динамічних сегментів інформаційного простору. Користувачі очікують не лише швидкого доступу до результатів матчів, а й можливості переглядати статистику команд і гравців, турнірні таблиці, історію зустрічей та аналітичні відомості в межах єдиного інтерфейсу. Наявні спортивні сервіси переважно або зосереджені на оперативному висвітленні подій, або на вузькоспеціалізованій статистиці, що ускладнює комплексне використання таких платформ спортивними любителями [1].

Додатковою проблемою є висока динамічність спортивних даних: результати матчів, статуси подій, показники команд і гравців постійно змінюються, тому інформаційний сервіс повинен забезпечувати швидке оновлення, структуроване збереження та коректне відображення актуальної інформації. Актуальність роботи зумовлена необхідністю створення веб-застосунку, який поєднує оперативність, зручний інтерфейс, базову аналітику та персоналізовані функції в межах одного програмного рішення.

II. Мета роботи

Метою роботи є розробка веб-застосунку для спортивних любителів, який забезпечує отримання, збереження, обробку та відображення спортивних даних у зручному для користувача вигляді. Для досягнення поставленої мети необхідно було проаналізувати предметну область, обґрунтувати вибір архітектури системи, спроєктувати структуру бази даних, реалізувати серверну і клієнтську частини, а також перевірити працездатність основних функціональних модулів застосунку.

III. Архітектура та програмна реалізація

У роботі обрано клієнт-серверну архітектуру, яка дозволяє розмежувати функції між рівнем представлення, серверною логікою та шаром зберігання даних. Серверну частину реалізовано засобами Java та SpringBoot, що забезпечує побудову RESTAPI, маршрутизацію запитів, організацію бізнес-логіки та взаємодію з базою даних. Клієнтську частину створено з використанням React і Vite, що дає змогу формувати динамічний інтерфейс користувача, підтримувати навігацію між сторінками без повного перезавантаження та швидко оновлювати вміст інтерфейсу [2, 3].

Узагальнену схему взаємодії основних компонентів розробленого рішення наведено на рисунку 1.

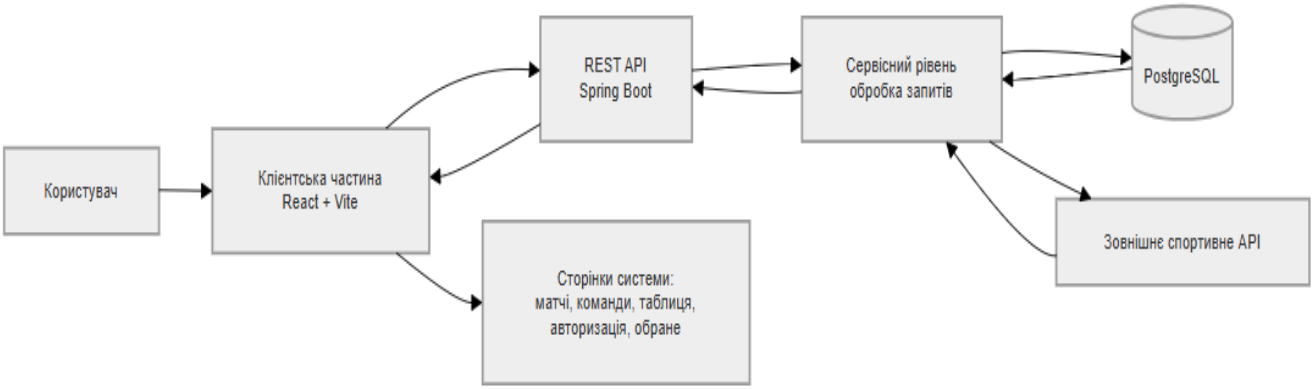

Рисунок 1 – Архітектура веб-застосунку для спортивних любителів

Для збереження спортивної й користувацької інформації використано PostgreSQL. У межах серверної частини реалізовано сутності, що відповідають ключовим об’єктам предметної області: матчам, командам, сезонам і турнірним таблицям. Доступ до даних організовано за допомогою JPA/Hibernate, що спрощує супровід програмного коду та інтеграцію схеми даних із серверною логікою[4].

IV. Ключові функції та візуальне підтвердження реалізації

У результаті реалізації створено веб-застосунок, який містить головну сторінку, сторінки матчів, команд, турнірної таблиці, детальної інформації про матч, а також модулі авторизації та обраного. На сторінці матчів користувач може переглядати події за статусами, використовувати пошук і переходити до деталізованої інформації. Сторінка турнірної таблиці підтримує вибір сезону, пошук команд та сортування статистичних показників. Реалізація персоналізованих функцій дає змогу працювати зі списком обраного, що підвищує зручність використання системи.

Важливою складовою є інтеграція із зовнішнім спортивним API, завдяки чому застосунок отримує актуальні дані про матчі, команди та турнірні таблиці. Отримана інформація проходить перевірку, перетворення до внутрішнього формату та надалі використовується у клієнтському інтерфейсі. Це дозволяє поєднати локальне збереження структури даних із можливістю автоматизованого оновлення відомостей про спортивні події.

Екранну форму реалізованого веб-застосунку наведено на рисунку 2.

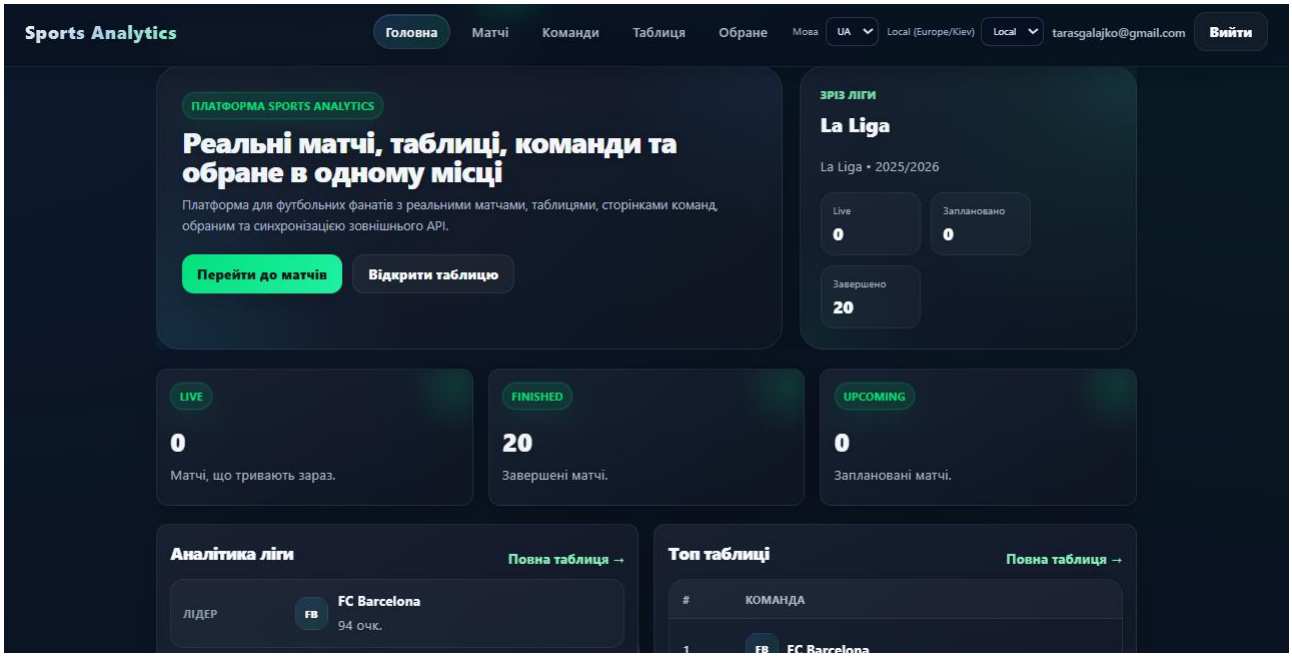

Рисунок 2 – Екранна форма головної сторінки веб-застосунку SportsAnalytics

Подана екранна форма ілюструє реалізований інтерфейс головної сторінки, який забезпечує швидкий доступ до матчів, турнірних таблиць та аналітичних блоків системи. Отримані результати свідчать, що запропонований підхід дає змогу поєднати в одному програмному продукті функції швидкого доступу до актуальної спортивної інформації та базового аналітичного опрацювання даних. На етапі тестування перевірено роботу API-запитів за допомогою Postman та коректність відображення сторінок у браузері, що підтвердило працездатність основних модулів і узгоджену взаємодію між клієнтською частиною, сервером і базою даних [5].

Висновок

У роботі розроблено веб-застосунок для спортивних любителів, який поєднує функції перегляду спортивних подій, відображення статистики, роботи з турнірними таблицями та персоналізованої взаємодії користувача із системою. Запропоноване рішення дає змогу забезпечити зручний доступ до актуальної спортивної інформації та створює передумови для подальшого розширення програмного продукту.

Подальші дослідження можуть бути пов’язані з розширенням аналітичних модулів, удосконаленням персоналізації, підтримкою додаткових видів спорту та впровадженням нових методів обробки спортивних даних.

Список використаних джерел

1. Spivak I., Krepych S., Litvynchuk M., Spivak S. Validation and data processing in JSON format. EUROCON2021 – 19th IEEE International Conference on Smart Technologies, 2021, pp.326-330
2. Крепич С.Я. Програмний комплекс оцінювання функціональної придатності пристроїв при заданих допустимих значеннях вихідних характеристик та допусків на параметри їх елементів. Сучасні комп’ютерні інформаційні технології: Матеріали Всеукраїнської школи-семінару молодих вчених і студентів АСІТ’2015 –Тернопіль: ТНЕУ, 2015. – С. 23-25

УДК 004.62

ПРЕДИКТИВНЕ СТИСНЕННЯ ТЕЛЕМЕТРИЧНИХ ДАНИХ У IoT-МЕРЕЖАХ НА ОСНОВІ ЛІНІЙНОГО ПРОГНОЗУВАННЯ НА КРАЙОВИХ ВУЗЛАХ

Бас В.В.[1)], Папа О.А.[2)]
Західноукраїнський національний університет
[1)] магістрант; [2)]д.філ., доцент

I. Постановка проблеми

Сенсорна телеметрія в IoT-мережах (MassiveIoT, LPWAN) переважно є повільно змінною та сильно автокорельованою: сусідні відліки відрізняються незначно й добре передбачаються за попередніми. Традиційна періодична передача повних відліків ігнорує цю передбачуваність, створюючи надлишковий трафік і прискорюючи розряд автономних вузлів. Подієві методи на кшталт Send-on-Delta частково усувають надлишковість, проте спираються на найпростіший прогноз — припущення про сталість сигналу, що неефективно для даних із вираженим трендом. Це зумовлює потребу в легких предиктивних методах стиснення, здатних враховувати динаміку сигналу безпосередньо на крайовому вузлі.

Наприклад, у мережах LoRaWAN корисне навантаження одного відліку часто становить лише кілька байтів, тоді як службові поля кадру (преамбула, адреси, заголовки, контрольна сума) додають десятки байтів. За періодичної передачі ця надлишковість багаторазово повторюється навіть тоді, коли вимірювана величина практично не змінюється.

II. Мета роботи

Метою роботи є зменшення обсягу переданої телеметрії та енергоспоживання IoT-вузлів шляхом розробки методу предиктивного стиснення, за якого крайовий пристрій передає лише похибку прогнозу, що перевищує заданий допуск, за гарантованого обмеження похибки відновлення сигналу.

III. Метод предиктивного стиснення

Запропонований метод базується на синхронній роботі однакових предикторів на крайовому вузлі та на сервері. На кожному кроці вузол прогнозує поточне значення сигналу за попередніми відліками. Для повільно змінних процесів достатньо легкого лінійного предиктора (екстраполяції першого порядку), що не потребує зберігання історії та складних обчислень:

$$\hat{x}_k = 2x_{k-1} - x_{k-2} \tag{1}$$

де $\hat{x}_k$ – прогнозоване значення на кроці k за двома попередніми відновленими відліками. Відхилення фактичного значення від прогнозу формує залишок:

$$e_k = x_k - \hat{x}_k \tag{2}$$

Залишок e_k передається на сервер лише тоді, коли його модуль перевищує допуск ε ($|e_k| > \varepsilon$); інакше відлік не передається, а сервер відновлює значення безпосередньо за прогнозом. Оскільки предиктор на сервері ідентичний, реконструкція виконується без накопичення похибки, а відхилення відновленого сигналу від фактичного гарантовано не перевищує ε. Параметр ε задає компроміс «стиснення – точність»: його збільшення зменшує кількість передач ціною підвищення допустимої похибки відновлення.

Алгоритм не потребує службового зворотного зв’язку для синхронізації: оскільки сервер відновлює пропущені відліки тим самим правилом прогнозу, що й вузол, обидві сторони завжди оперують ідентичною послідовністю відновлених значень. Обчислювальна складність методу становить O(1) на відлік, а потрібна пам’ять обмежується двома попередніми значеннями, що робить його придатним для найпростіших мікроконтролерів крайових вузлів.

Для запобігання довготривалому накопиченню неузгодженості предикторів (наприклад, унаслідок втрати пакета) періодично передається опорний (повний) відлік, який відновлює синхронізацію сторін та обмежує можливе розходження відновленого й фактичного сигналів у часі.

За потреби лінійний предиктор першого порядку узагальнюється до моделей вищого порядку (поліноміальна екстраполяція, авторегресія), що підвищує точність прогнозу для складніших сигналів ціною додаткових обчислень. Проте для типової повільно змінної телеметрії (температура, вологість,

тиск, рівень рідини) екстраполяція першого порядку забезпечує близький до оптимального компроміс між точністю та ресурсними витратами.

Роботу методу проілюстровано на рисунку 1: червоним позначено відліки, що реально передаються, а штрихова лінія — відновлений за прогнозом сигнал.

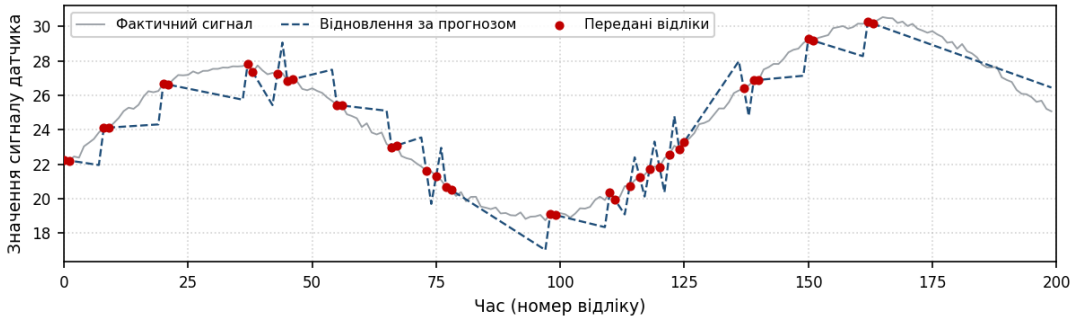

Рисунок 1 – Предиктивне стиснення сигналу телеметрії (ε = 2,0)

IV. Результати моделювання

Метод апробовано на синтетичному наборі телеметрії (200 відліків) із трендом, періодичною складовою та адитивним шумом. У Таблиці 1 наведено порівняння періодичної передачі, подієвого методу Send-on-Delta (предиктор «останнє значення») та запропонованого лінійного предиктора за однакового допуску ε.

Таблиця 1

Порівняння методів передачі телеметрії

Метод передачі	Передані відліки, %	Коефіцієнт стиснення	Похибка відновлення
Періодична (повна)	100	1,0	0
Send-on-Delta	37	2,7	≤ ε
Лінійний предиктор (запропонований)	18	5,6	≤ ε

Як видно з Таблиці 1, врахування тренду сигналу лінійним предиктором скорочує кількість переданих відліків до 18 % проти 37 % у Send-on-Delta, тобто забезпечує коефіцієнт стиснення близько 5,6× за тієї самої гарантованої точності відновлення. Менша кількість радіопередач пропорційно зменшує енергоспоживання вузла та подовжує час його автономної роботи.

Виграш методу безпосередньо залежить від передбачуваності сигналу: для гладких автокорельованих процесів коефіцієнт стиснення сягає 5–7×, тоді як для високодинамічних або сильно зашумлених даних він наближається до показників подієвих методів. Тому доцільним є адаптивне налаштування допуску ε та порядку предиктора під конкретний клас вимірюваної величини. У перерахунку на енергоспоживання скорочення кількості радіопередач зі 100 % до 18 % зменшує середню потужність, що витрачається на зв’язок, у кілька разів. Оскільки радіомодуль є основним споживачем енергії автономного вузла, це безпосередньо подовжує строк служби елемента живлення та збільшує інтервали між обслуговуванням мережі. Запропонований підхід орієнтований на системи екологічного моніторингу, промислової телеметрії та точного землеробства, де переважають повільно змінні величини, а ресурс автономних вузлів є критичним чинником експлуатації. Наукова новизна полягає у застосуванні легкого лінійного предиктора з обмеженою похибкою для подієвої передачі телеметрії, що, на відміну від класичного Send-on-Delta, враховує динаміку (тренд) сигналу й підвищує коефіцієнт стиснення за збереження гарантованої точності відновлення.

Висновки

Запропоновано метод предиктивного стиснення телеметрії, за якого крайовий вузол передає лише похибку прогнозу, що перевищує допуск, а сервер відновлює сигнал ідентичним предиктором з обмеженою похибкою. Моделювання підтвердило, що врахування динаміки сигналу підвищує коефіцієнт стиснення приблизно вдвічі порівняно з подієвим методом Send-on-Delta за тієї самої точності, зменшуючи трафік та енергоспоживання автономних вузлів. Перспективним напрямом є адаптивний вибір порядку предиктора залежно від характеру сигналу та поєднання предиктивного підходу з подієвою передачею для спеціалізованих систем моніторингу, зокрема Agro-IoT.

Список використаних джерел

1. Hassan M., Ali A. A SurveyonDataReductionTechniquesforIoTSensorNetworks. IEEE InternetofThingsJournal. 2023. Vol. 10, No. 5. P. 3452–3468.
2. Miśkowicz M. Send-on-deltaconcept: Anevent-baseddatareportingstrategy. Sensors. 2006. Vol. 6, No. 1. P. 49–63.

УДК 004.41

АРХІТЕКТУРНІ РІШЕННЯ ПРИ РОЗРОБЦІ ВЕБ-ЗАСТОСУНКУ ДЛЯ БРОНЮВАННЯ ГОТЕЛІВ НА ОСНОВІ КОМПОНЕНТНОГО ПІДХОДУ

Гончар Л.І.[1)], **Бодян.О.А.**[2)]
Західноукраїнський національний університет
[1)] к.е.н., доцент; [2)] бакалавр

І. Постановка проблеми

Сучасна готельна індустрія потребує надійних, масштабованих та ефективних веб-застосунків для зручного бронювання номерів. Зростання складності користувацького дизайну (UI) у таких системах призводить до того, що інтерфейс виявляється жорстко прив'язаним до внутрішньої логіки роботи програми.

Часто функції на кшталт перевірки доступності номерів, управління кошиком чи застосування фільтрів за зручностями реалізуються безпосередньо в коді відображення. Це робить складнішим у подальшому масштабування системи, її тестування та ефективну роботу над проєктом. Відсутність чіткої ізоляції бізнес-логіки від UI-компонентів стає серйозною проблемою, яка збільшує архітектурний борг і робить вже написаний код важким для швидкого повторного використання

ІІ. Мета роботи

Основною метою цієї роботи є створення архітектури вебзастосунку для бронювання готелів із чітким розділенням бізнес-логіки та інтерфейсу користувача. Особливий акцент зроблено на тому, щоб інтерфейс складався з модулів, придатних до повторного використання, а управління станом системи здійснювалося через спеціальні хуки (Custom Hooks).

Такий підхід забезпечує стабільність роботи застосунку при динамічних змінах даних (наприклад, актуалізація доступності номерів у реальному часі) та значно спрощує процес його підтримки й тестування.

ІІІ. Обґрунтування отриманих результатів

Для цього продукту використано клієнт-серверну архітектуру: клієнтська частина створена на основі бібліотеки React. Головним принципом стало чітке розділення бізнес-логіки та UI-компонентів, що дозволяє підтримувати чистоту коду й спрощує його обслуговування.Щоб реалізувати таке розділення, застосовано підхід із Custom Hooks. Уся логіка роботи з API, управління станами та необхідні розрахунки винесені в окремі модулі. Взаємодія між компонентами інтерфейсу та цією інкапсульованою логікою показана в таблиці 1.

Таблиці 1.

Архітектура розділення інтерфейсу користувача та інкапсульованої логіки

Назва UI-компонента	Візуальна відповідальність (Інтерфейс)	Custom Hook (Бізнес-логіка)
`HotelList`	Рендеринг сітки карток готелів, відображення стану завантаження (спінер) та пагінації.	`useHotelsFetch()` – виконання запитів до API, обробка помилок сервера, кешування результатів.
`BookingForm`	Інтерфейс вибору дат заїзду/виїзду (календар) та кількості гостей.	`useBookingValidation()` – перевірка доступності номерів на обрані дати, валідація введених даних.
`PriceCalculator`	Динамічне відображення підсумкової вартості, податків та застосованих промокодів.	`usePriceCalculation()` – виконання розрахунків за математичною моделлю, застосування знижок.
`AdminRoomGrid`	Візуальний конструктор та таблиця управління номерним фондом для персоналу.	`useAdminPanel()` – виконання CRUD-операцій (створення, оновлення, видалення) для номерів.

Хуки працюють із даними, що синхронізуються зі структурованою базою через REST API. Схему бази даних (див.рис.1) спроєктовано так, щоб забезпечити максимально високу точність і

надійність інформації про бронювання, платежі та відгуки користувачів, гарантуючи стабільність усієї системи.

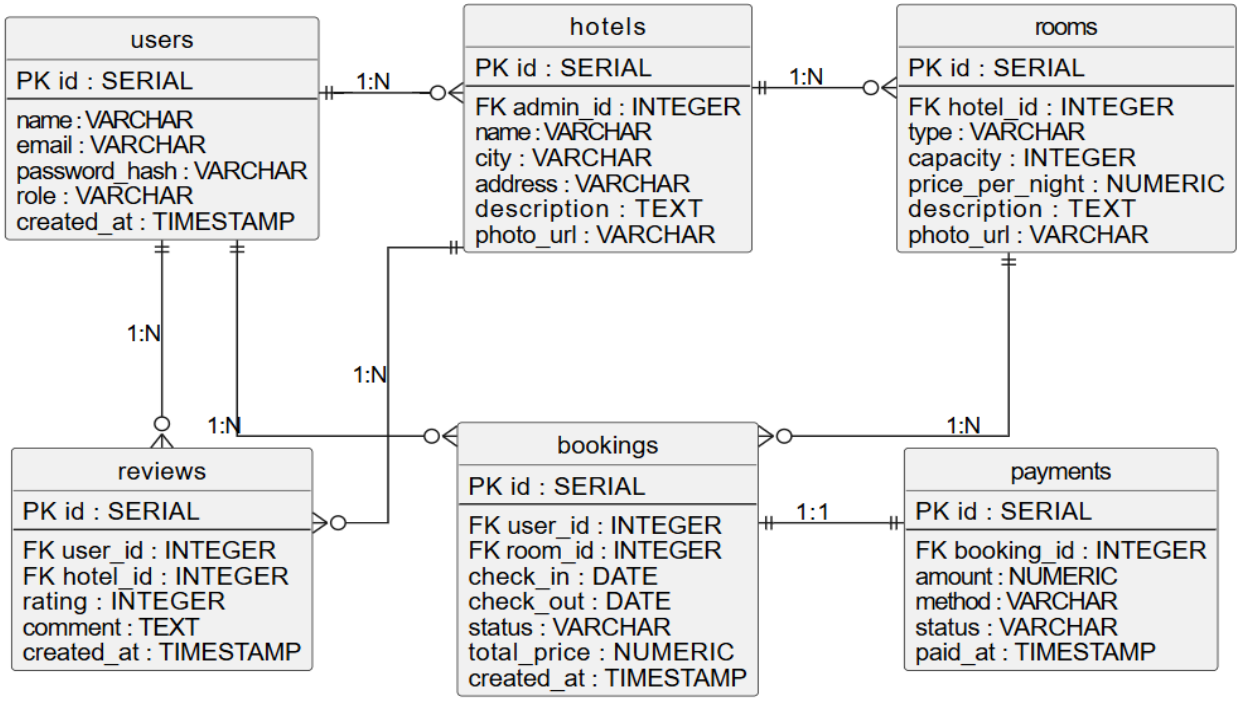

Рисунок 1 – Схема реляційної бази даних вебзастосунку бронювання готелів

Як показано на рисунку, ключовим елементом структури є сутність Bookings, яка встановлює зв'язок між користувачами (Users) та конкретними номерами готелів (Rooms). Така модель дозволяє зберігати повну історію транзакцій у таблиці Payments і забезпечує можливість отримання зворотного зв'язку через систему відгуків (Reviews). Розподіл даних на нормалізовані таблиці усуває дублювання інформації та створює умови для швидкого формування складних вибірок, наприклад, динамічного підрахунку рейтингу готелів у процесі роботи застосунку.

Висновки

У роботі представлено та аргументовано архітектурне рішення для вебзастосунку бронювання готелів, що базується на чіткому розділенні бізнес-логіки від інтерфейсу користувача. Використання Custom Hooks дало змогу повністю відокремити процес відображення та обробки даних, усунувши проблему надмірно пов'язаного коду. На практиці це значно спрощує розробку й підтримку проєкту, адже дозволяє безпечно оновлювати окремі модулі системи без ризику порушення роботи суміжних компонентів.

Збереження даних реалізовано через REST API у строго нормалізованій базі, що мінімізує ймовірність втрати інформації про клієнтів, їхні платежі чи доступність номерів. Загалом створена система є стабільною та готовою до подальшого масштабування. Наступним етапом розвитку проєкту стане оптимізація відображення ресурсомістких візуальних блоків і впровадження нових інструментів для управління готелем, а також удосконалення системи, що включає розширення функціоналу адміністративної панелі для персоналу та пришвидшення рендерингу складних UI-компонентів.

Список використаних джерел

1. A Guide to the Project Management Body of Knowledge (PMBOK Guide). 7th ed. – Project Management Institute, 2021. – 250 p.
2. S.Krepych, I.Spivak, Algorithm of Automatic Generation of Hotel Descriptions Using Templates Based on Markov Chains, 2018 International Scientific-Practical Conference Problems of Infocommunications Science and Technology Kharkiv, pp.257-260
3. Офіційна документація бібліотеки React. Настанови з використання компонентного підходу та React Hooks. URL: https://react.dev/
4. Літвинчук М.В., Крепич С.Я., Підсистема автоматичного генерування описів готелів для системи пошуку готелів «Hotelsbroker», Сучасні комп'ютерні інформаційні технології: Матеріали Всеукраїнської конференції з міжнародною участю ACIT'2017. – Тернопіль: ТНЕУ, 2017. – С.151-152
5. Гамма Е., Хелм Р., Джонсон Р., Влиссидес Д. Патерни проєктування об'єктно-орієнтованого програмного забезпечення. – 2020. – 366 с.
6. Крепич С.Я. Програмна система автоматичного генерування описів готелів із використанням шаблонів на основі ланцюгів Маркова. Вісник КрНУ імені Михайла Остроградського, Вип. 6(113), 2018. – с. 35-43

УДК 004.774

РЕКУРСИВНА КОНТЕЙНЕРНА МОДЕЛЬ ДЛЯ ВІЗУАЛЬНОГО КОНСТРУЮВАННЯ ВЕБ-СТОРІНОК

Прокіпчак В.І.[1)], Папа О.А.[2)]
Західноукраїнський національний університет
[1)] бакалавр;[2)] д.філ., доцент

I. Постановка проблеми

Візуальні конструктори веб-сторінок є ключовим компонентом сучасних систем управління контентом. Платформи Elementor (WordPress), Wix та Squarespace дозволяють створювати інтерфейси без програмування, однак їхня архітектура має принципове обмеження - фіксовану глибину вкладеності компонентів. Elementor оперує жорсткою трирівневою ієрархією «секція - колонка - віджет», Wix та Squarespace обмежуються двома рівнями. Це унеможливлює побудову складних адаптивних макетів із вкладеними сітками та рекурсивними композиціями компонентів, що є типовою вимогою сучасного веб-дизайну.

Додатковою проблемою є серіалізація: файлові JSON-об'єкти ускладнюють індексування, пошук та часткове оновлення структури на рівні СУБД. Актуальною є задача побудови моделі, що поєднує необмежену вкладеність із ефективним зберіганням у реляційному сховищі та виконанням структурних запитів безпосередньо на рівні бази даних.

II. Мета роботи

Метою роботи є розробка рекурсивної контейнерної моделі RecursoBlock для візуального конструктора веб-сторінок, яка забезпечує: 1) довільну глибину вкладеності компонентів; 2) ефективне управління деревом блоків зі складністю операцій O(n); 3) серіалізацію у формат JSONB бази даних PostgreSQL; 4) набір структурних пресетів для прискорення створення типових макетів.

III. Модель RecursoBlock

Кожен елемент сторінки подається як вузол дерева. Формально блок визначається кортежем:

$$B = \langle id, type, data, style, children \rangle \tag{1}$$

де $id \in UUID$ – унікальний ідентифікатор; $type \in T, |T| = 44$ – тип блоку (container, heading, text, image, gallery, hero тощо), $data : Map\langle String, Any\rangle$ – параметри контенту; $style : CSSProperties$ – візуальні властивості; $children : List\langle B\rangle$ – впорядкований список дочірніх блоків. Ключовою відмінністю є те, що блок типу *container* може включати довільні блоки, зокрема інші контейнери, формуючи рекурсивну деревоподібну структуру без обмеження глибини.

Контейнер параметризується вектором CSS Flexbox:

$$C = \langle direction, wrap, justify, align, gap \rangle \tag{2}$$

де $direction \in \{row, column, row-reverse, column-reverse\}$, $wrap \in \{nowrap, wrap\}$. Для типових макетів визначено множину пресетів $P = \{P_1, ..., P_6\}$, де кожен $P_k : \varnothing - B$ генерує попередньо налаштовану контейнерну ієрархію (1-4 колонки, сайдбар зліва/справа).

Операції над деревом визначаються рекурсивно через обхід у глибину (DFS). Функція пошуку:

$$find(blocks, id) = b, \quad \exists b \in flatten(blocks) : b.id = id \tag{3}$$

де $flatten$ – рекурсивне розгортання дерева методом пошуку в глибину (DFS). Аналогічно визначаються функції $update(blocks, id, f)$ для оновлення блоку за допомогою функції-трансформера f, $delete(blocks, id)$ для видалення вузла разом із його піддеревом та $addChild(blocks, parentId, child)$ для додавання нового дочірнього блоку до зазначеного контейнера.

Обчислювальна складність кожної з чотирьох операцій становить $O(n)$, де n – загальна кількість вузлів дерева. Для типових веб-сторінок, що містять від 20 до 200 блоків, емпірично виміряний час виконання операції пошуку не перевищує 1 мс у середовищі браузера Chrome 120, що забезпечує інтерактивну роботу візуального редактора без помітних затримок для користувача.

IV. Архітектура та програмна реалізація

Модель реалізовано у складі веб-застосунку електронної комерції на стеку Next.js 16, React 19 та Drizzle ORM. Дерево блоків серіалізується у JSONB-колонку таблиці pages бази PostgreSQL, забезпечуючи збереження ієрархії без JOIN-запитів та пошук за вмістом засобами операторів @> і ?. Систему апробовано на каталозі з 2816 товарами та 160 категоріями.

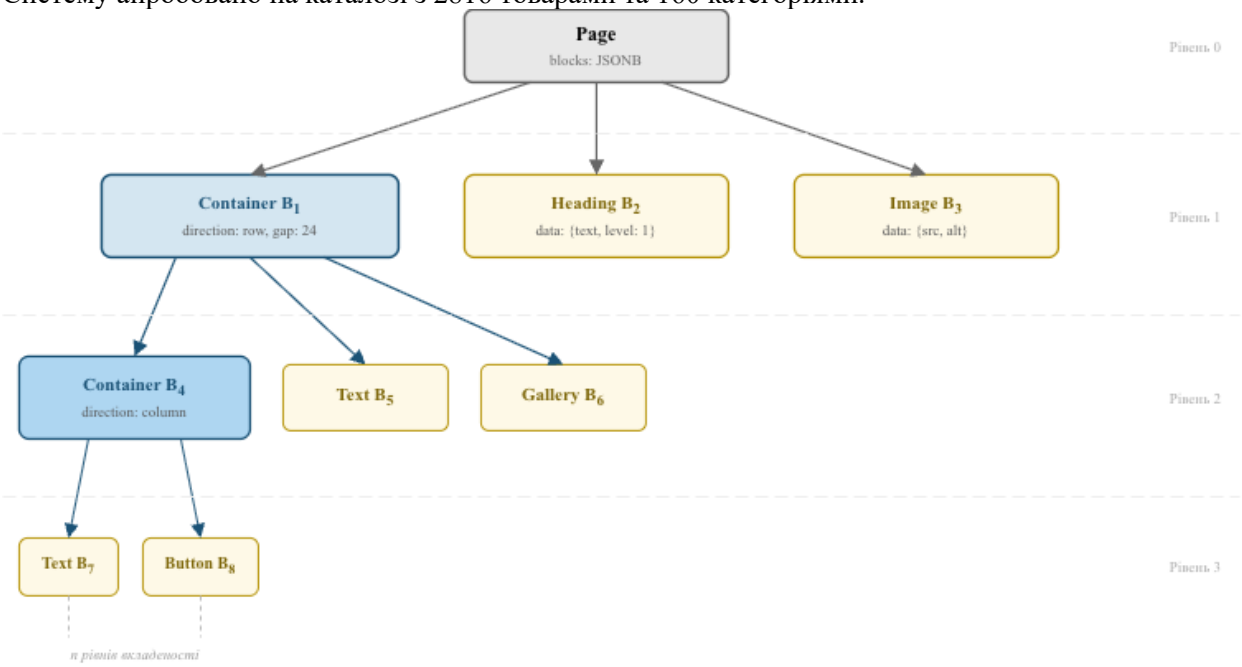

Рисунок 1 – Деревоподібна структура моделі RecursoBlock

Архітектура візуального редактора (див.рис.1) включає три взаємопов'язані модулі. Перший - LivePreview - відповідає за рекурсивний рендеринг дерева блоків у режимі реального часу з підтримкою інтерактивного виділення елементів, контекстного меню та переміщення блоків методом drag-and-drop. Другий модуль - BlockSettings - реалізує панель налаштувань обраного блоку за допомогою трьох вкладок: контент (текст, зображення, параметри), стиль (відступи, кольори, анімації) та розширені параметри (адаптивна видимість, трансформації). Третій модуль - NavigatorTree - надає ієрархічне відображення повної структури сторінки для швидкої навігації між вкладеними рівнями.

Рекурсивний React-компонент ContainerPreview забезпечує візуалізацію вкладених контейнерів на довільній глибині, підтримуючи виділення, додавання та видалення блоків на кожному рівні ієрархії.

Таблиця 1.

Порівняльна характеристика візуальних конструкторів

Характеристика	Elementor	Squarespace	Wix	RecursoBlock
Глибина вкладеності	3 рівні	2 рівні	2 рівні	Необмежена
Кількість типів |T|	~40	~25	~30	44
Серіалізація	JSON-файл	Хмарна	Хмарна	JSONB (БД)
Структурні пресети	Обмежені	Відсутні	Відсутні	6 типів
Відкритий код	Частково	Ні	Ні	Так

Висновок

Запропоновано рекурсивну контейнерну модель RecursoBlock для візуального конструювання веб-сторінок. На відміну від рішень із фіксованою ієрархією, модель забезпечує необмежену глибину вкладеності при складності операцій O(n). Серіалізація у PostgreSQL JSONB дозволяє зберігати складні макети без додаткових таблиць. Модель апробовано у веб-застосунку з каталогом 2816 товарів. Подальші дослідження спрямовано на колаборативне редагування та оптимізацію для дерев з $n > 10^3$ вузлів.

Список використаних джерел

1. Garrett J. J. The Elements of User Experience: User-Centered Design for the Web and Beyond.- 2nd ed.-New Riders, 2011.-192 p.
2. Krug S. Don’t Make Me Think, Revisited: A Common Sense Approach to Web Usability. - 3rd ed. - New Riders, 2014. - 216 p.

УДК 004.4

ПРОЕКТУВАННЯ ТА РОЗРОБЛЕННЯ СОЦІАЛЬНОЇ МЕРЕЖІ З АДАПТИВНИМ МЕХАНІЗМОМ ОБМЕЖЕННЯ СОЦІАЛЬНИХ ЗВ'ЯЗКІВ

Співак І.Я.[1)], Устиченко О.О.[2)], Крепич С.Я.[3)]
Західноукраїнський національний університет
[1)] к.т.н.,доцент; [2)]магістр; [3)] к.т.н., доцент

І. Постановка проблеми

Сучасні соціальні мережі зазвичай орієнтуються на збільшення кілткості соціальних звязків, наслідком того є інформаційне перевантаження користувачів та погіршення якості комунікацій. В підсумку взаємодія стає поверхневою, а ефективність комунікацій різко знижується.

Це породжує необхідність розроблення нових підходів для організації соціальних мереж, які регулюватимуть кількість контактів відовідно ло акиивностей кортстувача , тому що кількість не дорівнює якість. Одним із підходів є адаптивний механізм обмеження соціальних звязків.

ІІ.Мета роботи

Метою роботи є моделювання та проєктування соціальної мережі з адаптивним механізмом обмеження соціальних зав'язків шляхом розробленняархітектури програмної системи та структури бази даних.

ІІІ. Основна частина

На етапі проєктування визначено основну структуру. Система поділена на модулі с до яких належать :

- модуль управління користувачами
- модуль обміну повідомленнями,
- модуль системи квестів
- модуль адаптивного механізму керування соціальними зв'язками.

Якість соціальних зв'язків, їхня глибина та змістовність, є набагато важливішими за кількість контактів у соціальних мережах [1].

Основна ідея побудови проєкту є використанні клієнт-серверної архітектури. Принцип роботи полягає в тому, що сервер отримує запит, обробляє його і відповідає безпосередньо, без використання сторонніх ресурсів[2].

Клієнтська частина реалізована на C++, для реалізації користувацького інтерфейсу використовуватиметься QML, QMLпредставляє собою декларативною мовупрограмування.

Серверна частина логіки побудована на C++ та використанні сторонніх бібліотек таких як:

- Boost
- JWTTocket
- BCrypt

Модулі для зв'язку бази даних і сервера побудовані на бібліотеках:

- Mongocxx
- Bsoncxx

Для зберігання даних у системі спроєктовано структуру бази даних, що представлена у вигляді ER-діаграми. Вона включає основні сутності:

- Користувачі (Users)
- Контакти (Contacts)
- Повідомлення (Messages)
- активність користувачів (Activities);
- квести (Quests);
- виконані квести (UserQuests);
- сповіщення (Notifications).

Адаптивний механізм базується на аналізі активностікористувача, шо враховує кількість повідомлень, супутніх взаємодій та участь в квестах. На основі цихданих формується коефіцієнт активності, який впиває на кількість доступних зав'язків.

Загальна структура взаємодії компонентів подана на рисунку 1. Між сутностями визначено зв'язки, які забезпечують коректну взаємодію даних у системі та підтримку цілісності інформації.

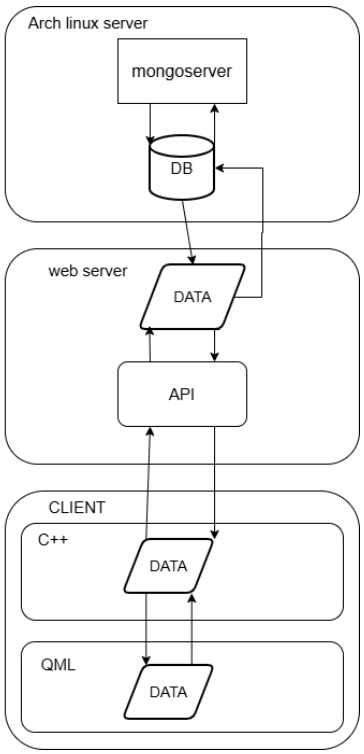

Рисунок 1 – Архітектура соціальної мережі з адаптивним механізмом обмеження соціальних зв’язків

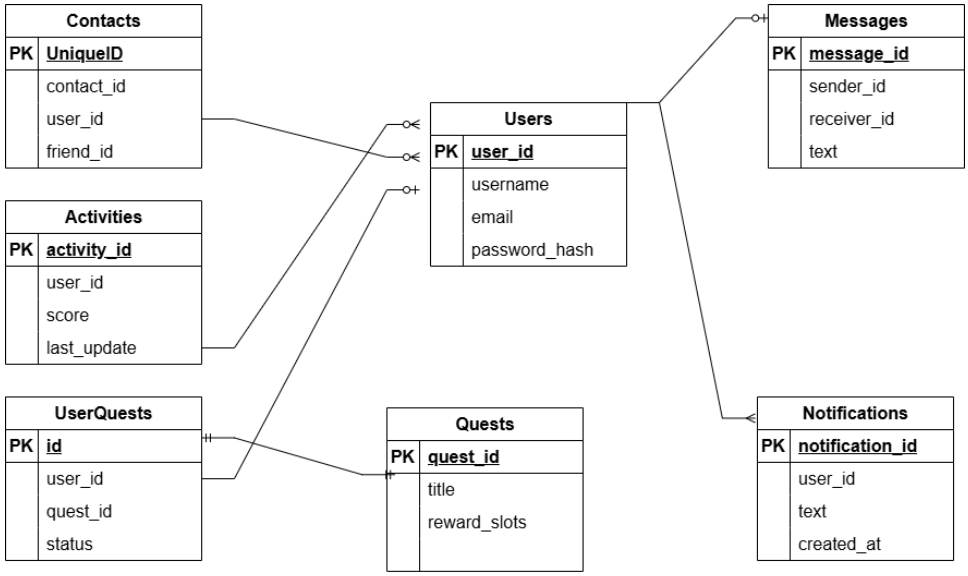

Рисунок 2 – ER-діаграма бази даних соціальної мережі

Висновок

У роботі виконано моделювання та проєктування соціальної мережі з адаптивним механізмом обмеження соціальних зв’язків. Розроблено архітектуру системи та структуру бази даних, що забезпечують реалізацію адаптивного керування соціальними зв’язками та створюють основу для подальшої програмної реалізації.

Список використаних джерел

1. Соціальні зв'язки та їхній вплив на наше здоров'я. Менталзон. URL: https://mentalzon.com/uk/post/2215/соціальні-звязки-та-їхній-вплив-на-наше-здоровя
2. Клієнт-серверна архітектура. Онлайн-курси від компанії QATestLab | Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/
3. S.Krepych, I.Spivak, Model of human wight correction based on interval data analysis. International Journal of Computing, Vol.19(1),2020, pp.128-136.
4. Співак І.Я., Крепич С.Я., Федоров О.А., Програмна система оцінювання ефективності праці в залежності від потреб. Матеріали школи-семінару молодих вчених і студентів «Комп’ютерні інформаційні технології» СІТ’2019, 29 листопада 2019р., Тернопіль, стр. 34

УДК 004.42

ІНТЕРАКТИВНА СИСТЕМА ОНЛАЙН-ЗАПИСУ ТА ОБСЛУГОВУВАННЯ КЛІЄНТІВ ДЛЯ ПСИХОЛОГІЧНОГО КОНСУЛЬТУВАННЯ

Дубецька В.В.[1)], Стасів І.С.[2)]
Західноукраїнський національний університет
[1)] бакалавр; [2)] к.т.н., доцент

І. Постановка проблеми

У сучасних умовах зростання потреби у дистанційних сервісах особливої актуальності набуває цифровізація психологічної допомоги.

Користувачі очікують можливості швидко знайти фахівця, ознайомитися з описом послуг, обрати зручний час консультації та отримати підтвердження запису без тривалого листування або телефонних дзвінків.

На практиці організація роботи психологічних сервісів часто здійснюється за допомогою окремих таблиць, месенджерів і ручного обліку звернень.

Такий підхід ускладнює ведення розкладу, підвищує ризик втрати даних, створює незручності для клієнтів і не забезпечує належного контролю за замовленнями та статусами консультацій.

Отже, виникає необхідність у створенні інтерактивної інформаційної системи, яка автоматизує онлайн-запис, спростить взаємодію між клієнтом і психологом, підвищить якість обслуговування та забезпечить надійне зберігання даних.

ІІ. Мета роботи

Метою роботи є розробка інтерактивної системи онлайн-запису та обслуговування клієнтів для психологічного консультування, яка забезпечить зручний запис на консультації, перегляд послуг і спеціалістів, обробку звернень користувачів та централізоване збереження інформації.

ІІІ. Особливості програмної реалізації інтерактивної системи

Розроблення системи доцільно здійснювати на основі веборієнтованої архітектури, що поєднує клієнтську частину, серверний рівень та реляційну базу даних.

Такий підхід відповідає результатам переддипломної практики, під час якої було обґрунтовано використання HTML, CSS, JavaScript, середовища Visual Studio Code та СУБД SQL Server для побудови сервісу.

Клієнтська частина забезпечує перегляд профілів психологів, доступних курсів і консультацій, реєстрацію, авторизацію та подання заявки через вебінтерфейс.

Серверний модуль виконує валідацію введених даних, обробляє HTTP-запити, взаємодіє з базою даних та формує відповідь для користувача.

У структурі бази даних доцільно передбачити сутності «Користувач», «Клієнт», «Психолог», «Курс», «Сеанс» і «Замовлення», що дає можливість організувати зберігання інформації про записи на консультації, придбання курсів та історію звернень.

Функціональні можливості системи мають охоплювати:

- реєстрацію користувачів;
- авторизацію;
- перегляд профілів психологів;
- вибір послуги;
- створення заявки на консультацію;
- підтвердження запису.

Для адміністратора або психолога повинні бути передбачені засоби редагування контенту, оновлення розкладу, перегляду звернень і контролю актуальності інформації на сайті.

Важливим аспектом реалізації є безпека персональних даних: перевірка коректності введеної інформації, обмеження доступу до окремих функцій, захист облікових записів та забезпечення цілісності даних.

Запровадження такої системи підвищує зручність взаємодії між клієнтом і психологом та зменшує кількість ручних операцій.

Таблиця 1

Основні функціональні можливості системи

Модуль	Призначення
Реєстрація та авторизація	Надання користувачам доступу до персоналізованих функцій системи
Профілі психологів	Перегляд інформації про спеціалістів, послуги та напрями консультування
Онлайн-запис	Формування заявки на консультацію та вибір зручного часу
Курси та послуги	Ознайомлення з доступними курсами й консультаційними програмами
База даних	Зберігання відомостей про клієнтів, психологів, сеанси та замовлення

Запропонований підхід дає змогу автоматизувати основні процеси психологічного консультування в онлайн-форматі, зменшити навантаження на спеціаліста та покращити якість сервісу. Крім того, система створює підґрунтя для подальшого розширення, зокрема впровадження сповіщень, статистичних звітів і персоналізованих рекомендацій для користувачів.

Перспективним напрямом розвитку системи є інтеграція механізмів сповіщення про підтвердження або зміну часу консультації, формування коротких статистичних звітів щодо кількості звернень і використання аналітичних даних для вдосконалення роботи сервісу. Такі можливості підвищують практичну цінність системи та розширюють її застосування в діяльності приватних фахівців і консультаційних центрів.

Окрему увагу слід приділити зручності користувацького інтерфейсу. Веб-застосунок має забезпечувати інтуїтивно зрозумілу навігацію, адаптивне відображення на різних пристроях, швидкий перехід між розділами та мінімальну кількість дій для оформлення запису. Це особливо важливо для користувачів, які звертаються по психологічну допомогу і потребують простого та комфортного способу взаємодії із сервісом.

Для підвищення ефективності роботи доцільно передбачити модуль обробки звернень, який забезпечує фіксацію дати і часу запису, контактних даних клієнта, виду обраної послуги та поточного статусу заявки. Це дає змогу уникнути дублювання записів, забезпечити впорядковане ведення історії звернень і спростити подальший аналіз навантаження на спеціалістів.

Практична реалізація системи повинна передбачати розмежування ролей користувачів. Клієнт отримує можливість переглядати інформацію про фахівців, обирати консультацію або курс, залишати заявку та відстежувати її статус. Психолог або адміністратор, у свою чергу, може редагувати відомості про послуги, керувати розкладом, переглядати звернення та підтверджувати записи.

Додатковою перевагою запропонованої системи є можливість централізованого накопичення даних про звернення, що створює підґрунтя для подальшого вдосконалення сервісу. На основі узагальненої інформації можна визначати найбільш затребувані послуги, аналізувати завантаженість спеціалістів, коригувати розклад консультацій та підвищувати якість організації психологічної допомоги.

Висновок

У роботі обґрунтовано доцільність створення інтерактивної системи онлайн-запису та обслуговування клієнтів для психологічного консультування.

Запропоноване рішення дозволяє впорядкувати взаємодію між клієнтом і психологом, автоматизувати обробку звернень та підвищити ефективність надання послуг.

Використання сучасних веб-технологій і засобів керування даними забезпечує зручність, надійність і перспективність подальшого розвитку системи.

Список використаних джерел

1. Пасічник В. В., Резніченко В. А. Організація баз даних та знань. Київ: Видавнича група BHV, 2016. 384 с.
2. Берко А. Ю., Верес О. М., Пасічник В. В. Системи баз даних та знань. Львів: Магнолія-2006, 2012. 584 с.
3. Шаров С. В., Осадчий В. В. Бази даних та інформаційні системи. Мелітополь: МДПУ ім. Б. Хмельницького, 2014. 352 с.
4. Крепич С.Я., Співак І.Я. Якість програмного забезпечення та тестування: базовий курс. Тернопіль: ФОП Паляниця В.А, 2020. – 478с.
5. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis: Wiley, 2011. 490 p.
6. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol: O'Reilly Media, 2020. 704 p.
7. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Boston: Pearson, 2014. 1440 p.

УДК 004.62

МАТЕМАТИЧНЕ ТА АЛГОРИТМІЧНЕ МОДЕЛЮВАННЯ АДАПТИВНОГО СТИСНЕННЯ ТЕЛЕМЕТРИЧНИХ ДАНИХ У СИСТЕМАХ AGRO-IoT

Бас В.В.[1)], Папа О.А.[2)]
Західноукраїнський національний університет
[1)] магістрант; [2)] д.філ., доцент

I. Вступ та постановка проблеми

Вирішення проблеми неефективного обміну потоковими телеметричними даними в системах аграрного Інтернету речей (Agro-IoT) вимагає переходу від теоретичної постановки до математичного та алгоритмічного моделювання. Основними вузькими місцями в таких мережах є низька пропускна здатність нестабільних каналів зв'язку (наприклад, LoRaWAN або NB-IoT) та накопичення надлишкових обсягів "сирих" даних. Традиційна поштучна передача кожного показника датчика створює значне навантаження через накладні витрати протоколу (overhead). Для оптимізації обміну даними доцільно впровадити алгоритм попередньої фільтрації та стиснення з динамічним підбором розміру пакету відправки (батчу).

II. Мета роботи

Метою роботи є підвищення ефективності обміну потоковими телеметричними даними в системах аграрного Інтернету речей (Agro-IoT) шляхом розробки математичної моделі оцінювання затримок передачі та алгоритму адаптивного стиснення з динамічним підбором розміру пакета (батчу), що мінімізує мережеві витрати за умов обмеженої пропускної здатності та нестабільних каналів зв'язку.

III. Математична модель адаптивного масштабування

Розглянемо неперервний потік телеметрії як часовий ряд; передавання даних здійснюється пакетами (батчами). Позначимо через B кількість записів у пакеті, через S – середній розмір одного стисненого запису (у байтах), а через H – розмір обов'язкового службового заголовка протоколу (overhead). Тоді повний розмір пакета L визначається співвідношенням (1):

$$L = B \cdot S + H \tag{1}$$

Очікуваний час доставки одного пакета $E[T(B)]$ з урахуванням ймовірності втрати та необхідності повторних відправлень (ретрансмісій) описується геометричним розподілом (2):

$$E[T(B)] = \frac{T_{tx}(B)}{1 - P_{loss}(B)} \tag{2}$$

де Ttx – базовий час передачі пакета, що дорівнює відношенню його розміру (1) до пропускної здатності каналу R; P_{loss} – ймовірність втрати пакета.

Якщо ймовірність помилки на рівні одного біта становить p, то ймовірність втрати пакета апроксимується лінійною функцією за малих значень p (3):

$$P_{loss}(B) \approx p \cdot (B \cdot \bar{S} + H) \tag{3}$$

Цільовою функцією оптимізації є середня затримка в розрахунку на один запис телеметрії, тобто відношення очікуваного часу доставки пакета (2) до кількості записів у ньому B. Підставляючи співвідношення (1) і (3) у формулу (2), отримуємо цільову функцію (4):

$$J(B) = \frac{B \cdot \bar{S} + H}{R \cdot B \cdot (1 - p(B \cdot \bar{S} + H))} \tag{4}$$

Для знаходження оптимального розміру батчу Bopt, що мінімізує затримку, візьмемо похідну функції $J(B)$ за змінною B та прирівняємо її до нуля. Аналітичний розв'язок цього рівняння (за умови незначного впливу заголовка на відсоток втрат) дає аналітичний вираз (5):

$$B_{opt} \approx \sqrt{\frac{H}{p \cdot \bar{S}^2}} \quad (5)$$

Формула (5) теоретично обґрунтовує баланс: що більші накладні витрати H, то більшими пакетами вигідно відправляти дані, проте зі зростанням ймовірності помилок каналу p розмір пакета B необхідно зменшувати для уникнення дорогих ретрансмісій.

IV. Алгоритмічне рішення на базі Edge-вузла

Для практичної реалізації на крайовому пристрої (Edge Node) розроблено алгоритм, що базується на зміненій концепції Send-on-Delta (SDT) у поєднанні з алгоритмами дельта-кодування (подібно до процедур у Gorilla TSDB).

Кроки алгоритму:

1. Контролер зчитує показання датчика і порівнює їх з останнім успішно відправленим значенням.

2. Якщо різниця між значеннями за модулем менша за заданий поріг чутливості δ , запис розцінюється як шум та ігнорується. Це автоматично фільтрує незначні коливання (шум сенсорів).

3. Якщо поріг перевищено, обчислюється різниця між поточним і попереднім значенням (дельта), яка стискається алгоритмом кодування довжин серій (RLE) і додається до буфера.

4. Як тільки кількість записів у буфері досягає динамічно розрахованого значення B_{opt} (5), пакет відправляється на сервер.

Запропонований алгоритм має сталу обчислювальну складність на кожен відлік і не потребує зберігання повної історії сигналу, що робить його придатним для виконання безпосередньо на мікроконтролерах крайових вузлів з обмеженим обсягом пам'яті. Поєднання порогової фільтрації з дельта-кодуванням одночасно зменшує і кількість службових заголовків, і корисне навантаження кожного пакета, що є критичним для низькошвидкісних каналів LoRaWAN та NB-IoT.

Архітектурно система складається з трьох рівнів: крайових вузлів збору даних із сенсорів, шлюзу агрегації та хмарного сервера зберігання й аналітики. Обмін даними між вузлами та шлюзом реалізовано за протоколом MQTT-SN, оптимізованим для сенсорних мереж із нестабільним зв'язком, тоді як стиснені пакети передаються на сервер для подальшої декомпресії та збереження у часовому ряді.

V. Експериментальне дослідження та оцінка результатів

Для перевірки ефективності запропонованого алгоритму було створено імітаційну модель потоку телеметрії Agro-IoT (вологість та температура ґрунту) обсягом 100 000 зразків. Параметри мережі імітували умови протоколу MQTT-SN у мережі з показником бітових помилок $p = 10^{-4}$.

Обчислимо цільові показники рівня обслуговування (SLO). У таблиці 1 наведено порівняння трьох підходів: стандартної передачі по одному повідомленню, передачі статичними батчами по 50 записів, та запропонованого адаптивного алгоритму $(SDT + B_{opt})$.

Таблиця 1

Результати моделювання передачі телеметрії

Метод передачі	Заощадження трафіку (%)	95-й перцентиль затримки (p95), мс	Втрата пакетів до перевідправки (%)
Базовий MQTT (B=1)	0%	1450	1.2%
Статичний батчинг (B=50)	18%	980	14.8%
Адаптивне стиснення (SDT + B_opt)	76.77%	340	3.5%

Як видно з результатів бенчмарку, використання SDT дозволяє відфільтрувати інформаційне сміття, що заощаджує 76.77% пропускної здатності. Водночас аналітично підібраний розмір пакета утримує ймовірність втрати на прийнятному рівні, зменшуючи цільовий показник затримки (p95) у понад 4 рази порівняно з неоптимізованим потоком.

Отримані значення узгоджуються з теоретичним висновком формули (5): надмірне збільшення розміру батчу підвищує ймовірність повторних передач і сумарну затримку, тоді як надто малі пакети збільшують частку службового трафіку. Динамічний підбір оптимального розміру пакета забезпечує компроміс між цими чинниками з урахуванням поточного стану каналу зв'язку.

Моделювання проводилося на синтетичному наборі даних, що імітує добовий цикл температурного режиму з додаванням адитивного білого гауссового шуму (AWGN) з відношенням сигнал-шум (SNR) 20 дБ.

На рисунку 1 видно, що алгоритм Send-on-Delta (SDT) ефективно фільтрує високочастотний шум, залишаючи лише значущі зміни сигналу (червоні точки).

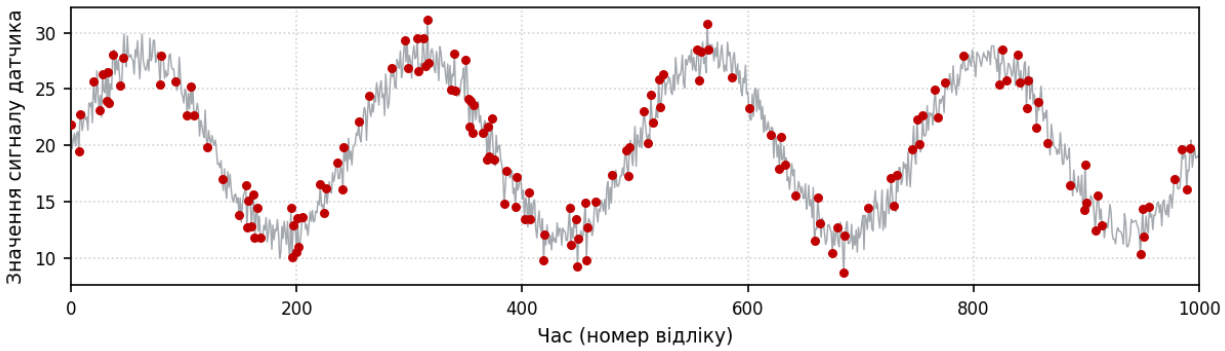

Рисунок 1 – Результати моделювання ефективності алгоритму

Як продемонстровано на рис. 2, при збільшенні порогу чутливості (threshold) економія зростає від 75% до 90%. Однак, при значенні порогу > 0.3 спостерігається "знегладжування" сигналів (втрата мікродинаміки). Тому обраний робочий діапазон порогу 0.05-0.10 є компромісним: він забезпечує економію трафіку в межах 76-78%, гарантуючи повну відповідність даних метрологічній точності датчика. Це підтверджує можливість застосування методу в реальних мережах Agro-IoT з обмеженою пропускною здатністю."

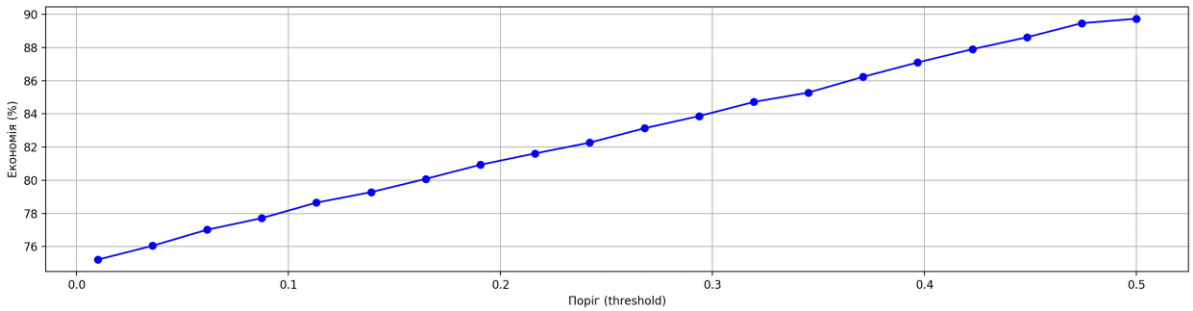

Рисунок 2 – Залежність економії трафіку від порогу чутливості алгоритму

Висновки

Запропонована математична модель оцінки затримок ілюструє нелінійну залежність ефективності передачі від розміру пакета даних. Аналітично знайдений оптимум функції, інтегрований з алгоритмом Send-on-Delta, був апробований на синтетичному наборі даних. Експерименти підтвердили, що адаптивне керування розміром батчу у поєднанні з попередньою фільтрацією шумів на крайовому вузлі здатне зменшити цільовий показник затримки (p95) понад у 4 рази та заощадити до 76–78 % мережевого трафіку в умовах нестабільних з'єднань Agro-IoT, зберігаючи відповідність даних метрологічній точності датчиків. Подальші дослідження спрямовані на адаптацію порогу чутливості в реальному часі залежно від статистичних характеристик сигналу та на розширення моделі для багатопараметричних потоків телеметрії.

Список використаних джерел

1. Pelkonen T., Franklin S., Teller J., Cavallaro P., Huang Q. Gorilla: A fast, scalable, in-memory time series database. Proceedings of the VLDB Endowment. 2015. Vol. 8, No. 12. P. 1816-1827.
2. Miskowicz M. Send-on-delta concept: An event-based data reporting strategy. Sensors. 2006. Vol. 6, No. 1. P. 49-63.
3. Burtscher M., Ratanaworabhan P. FPC: A high-speed compressor for double-precision floating-point data. IEEE Transactions on Computers. 2009. Vol. 58, No. 1. P. 18-31.
4. Stanford-Clark A., Truong H. L. MQTT for sensor networks (MQTT-SN) protocol specification. International Business Machines (IBM) Corporation. 2013. Version 1.2. P. 1-28.

УДК 004.42

ВЕБОРІЄНТОВАНА СИСТЕМА ОПТИМІЗАЦІЇ ЛОГІСТИЧНИХ МАРШРУТІВ ДОСТАВКИ ТЕХНІКИ

Варварич Л.В.[1)], **Войтюк І.Ф.**[2)]
Західноукраїнський національний університет
[1)]бакалавр; [2)] к.т.н., доцент

І. Постановка проблеми

Сучасні підприємства, що займаються продажем, обслуговуванням або доставкою техніки, щоденно виконують значну кількість логістичних операцій. До таких операцій належать приймання замовлень, формування заявок, визначення адрес доставки, планування руху транспортного засобу, контроль виконання замовлення та збереження інформації про результат доставки. У випадку доставки технічної продукції логістичний процес має особливе значення, оскільки техніка часто є вартісною, потребує обережного транспортування та чіткого дотримання строків доставки.

Однією з основних проблем у цій сфері є побудова раціонального маршруту. Зі збільшенням кількості адрес швидко зростає кількість можливих варіантів їх відвідування. У таких умовах ручне планування стає трудомістким, менш точним і залежним від досвіду відповідального працівника. Актуальність роботи зумовлена необхідністю автоматизації планування логістичних маршрутів доставки техніки за допомогою веборієнтованої системи. Використання вебзастосунку дозволяє централізовано зберігати дані про точки доставки, маршрути та замовлення, забезпечити доступ до системи через браузер і спростити взаємодію користувача з модулем оптимізації.

ІІ. Мета роботи

Метою роботи є підвищення ефективності планування доставки техніки шляхом розробки веборієнтованої системи оптимізації логістичних маршрутів на основі генетичного алгоритму.

ІІІ. Модель оптимізації логістичного маршруту

Для формалізації задачі доставки техніки маршрут можна подати у вигляді графа $G = (V, E)$, де V – множина вершин, що відповідають складу та точкам доставки, а E – множина ребер між ними. Кожному ребру відповідає вага, яка може визначати відстань, час руху або умовну вартість переміщення між двома точками. У межах розроблюваної системи основним критерієм оптимізації є мінімізація загальної довжини маршруту.Маршрут доставки можна подати як впорядковану послідовність точок:

$$M = \{v_0, v_1, v_2, ..., v_n\} \quad (1)$$

де v_0 – початкова точка маршруту, а $v_1, v_2, ..., v_n$ – точки доставки, які необхідно відвідати. Якість маршруту визначається сумарною довжиною переходів між сусідніми точками. Загальна довжина маршруту L(M) може бути обчислена за формулою:

$$L(M) = \sum_{i=0}^{n-1} d(v_i, v_{i+1}) \quad (2)$$

де $d(v_i, v_{i+1})$ – вартість найкоротшого переходу між двома сусідніми точками маршруту, обчислена за алгоритмом Флойда-Уоршелла.

Оскільки задача вибору найкращої послідовності точок належить до класу задач маршрутизації транспорту й має комбінаторний характер [1], для її розв'язання доцільно використати генетичний алгоритм [2]. У такому підході кожний можливий маршрут розглядається як окрема особина популяції, а послідовність точок доставки – як хромосома; стартова точка транспортного засобу фіксується на початку маршруту, а решта точок можуть змінювати своє положення під час роботи алгоритму.

Для оцінювання якості кожної особини використовується функція пристосованості. Оскільки потрібно мінімізувати довжину маршруту, значення пристосованості може бути обернено пропорційним до загальної довжини маршруту:

$$F(M) = \frac{1}{L(M)} \quad (3)$$

де $F(M)$ – значення функції пристосованості маршруту, $L(M)$ – загальна довжина маршруту. Чим коротший маршрут, тим більше значення функції пристосованості та тим вища ймовірність збереження цього варіанта під час наступних ітерацій алгоритму. Крім вартості проходження маршруту, у фітнес-функції також враховуються штрафи за неперевезений товар та неефективні зупинки.

Основними етапами генетичного алгоритму є формування початкової популяції, оцінювання пристосованості, відбір кращих особин, схрещування, мутація та формування нового покоління. Початкова популяція створюється шляхом генерації випадкових маршрутів, кожен з яких починається з фіксованої стартової точки транспортного засобу. Після цього для кожного маршруту обчислюється його вартість та значення функції пристосованості. На етапі відбору до наступного покоління переходять кращі 50% хромосом, тобто маршрути з найкращим значенням функції пристосованості. Процес повторюється протягом заданої кількості поколінь або до припинення покращення результату.

Для схрещування застосовано оператори OnePointCrossover та UniformCrossover: перший обмінює частини маршрутів двох батьківських хромосом після випадково обраної точки розділення, а другий формує нащадка, обираючи окремі елементи від різних батьків. При цьому контролюється, щоб кожна точка була включена в маршрут лише один раз. Мутація реалізована операторами Swap, Scramble, Insertion і Deletion, які відповідно переставляють дві зупинки, перемішують частину послідовності, переносять або вилучають окрему зупинку. Це дозволяє підтримувати різноманітність популяції та зменшувати ризик передчасного знаходження локально оптимального рішення.

Застосування генетичного алгоритму є доцільним для задачі доставки техніки, оскільки він дозволяє знаходити раціональний маршрут без повного перебору всіх можливих варіантів, що підтверджують і результати інших досліджень задач маршрутизації [3]. Такий підхід є гнучким і може бути розширений для врахування додаткових критеріїв, зокрема часу доставки, пріоритетності замовлень, кількості транспортних засобів або обмежень щодо вантажопідйомності.

IV. Програмна реалізація

Розроблювана веборієнтована система оптимізації логістичних маршрутів доставки техніки передбачає наявність клієнтської частини (Blazor-вебінтерфейсу), серверної частини, JSON-файлу з вхідними даними та модуля оптимізації маршруту. Клієнтська частина забезпечує взаємодію користувача із системою: введення точок доставки, перегляд списку замовлень, запуск оптимізації та відображення отриманого маршруту у вигляді списку, на графі та на карті. Серверна частина відповідає за зчитування та перевірку вхідних даних з JSON-файлу, виконання основної логіки роботи та виклик алгоритму оптимізації. Систему реалізовано на платформі ASP.NET Core за технологією Blazor мовою програмування C# [4]. Такий вибір пояснюється тим, що ASP.NET Core є сучасною платформою для створення вебзастосунків і вебсервісів, підтримує кросплатформність, роботу з API та побудову серверної логіки. Вхідні дані системи – точки маршруту, дороги між ними та параметри транспортних засобів – зберігаються у форматі JSON, тому використання окремої реляційної бази даних не передбачається. Модуль оптимізації маршруту приймає на вхід граф точок і доріг між ними; матриця найкоротших вартостей переміщення між вузлами обчислюється за алгоритмом Флойда–Уоршелла, після чого на її основі працює генетичний алгоритм. Після формування початкової популяції маршрутів виконується задана кількість поколінь генетичного пошуку, після чого система повертає користувачу маршрут із найкращим значенням функції пристосованості. Результатом роботи модуля є впорядкований список точок доставки та загальна довжина маршруту.

Висновок

У роботі запропоновано підхід до розробки веборієнтованої системи оптимізації логістичних маршрутів доставки техніки. Для розв’язання задачі побудови маршруту обґрунтовано використання генетичного алгоритму, який дозволяє подати маршрут як хромосому, оцінювати його якість за допомогою функції пристосованості та поступово покращувати результат за рахунок операцій відбору, схрещування та мутації. Запропонована система може бути використана для автоматизації процесу планування доставки техніки, скорочення часу формування маршруту та підвищення ефективності логістичних операцій.

Список використаних джерел

1. Dantzig G. B., Ramser J. H. The Truck Dispatching Problem. Management Science. 1959. Vol. 6, No. 1. P. 80–91.
2. Holland J. H. Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press, 1975. 183 p.
3. Ochelska-Mierzejewska J., Poniszewska-Maranda A., Maranda W. Selected Genetic Algorithms for Vehicle Routing Problem Solving. Electronics. 2021. Vol. 10, No. 24. Article 3147. DOI: 10.3390/electronics10243147.

УДК 004.8

АРХІТЕКТУРНІ ПРИНЦИПИ ТА ДОМЕН ТЕХНОЛОГІЙ РОЗРОБКИ СУЧАСНИХ КЛІЄНТ-СЕРВЕРНИХ ВЕБ-СИСТЕМ ДЛЯ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ

Гищук М.-М.Д.[1)], Манжула Н.В.[2)], Грицеляк М.І.[3)], Руденька Ю.І.[4)], Балайда Р.О.[5)], Кравченюк Х.Ю.[6)]

Західноукраїнський національний університет

[1-6)]бакалавр

Постановка проблеми

У сучасних умовах цифровізації більшість бізнес-процесів вимагають переходу від локальних десктопних програм до кросплатформних веб-орієнтованих систем. Розробка монолітних веб-додатків, де інтерфейс користувача та бізнес-логіка жорстко пов'язані на стороні сервера, призводить до значних труднощів при масштабуванні, підтримці коду та адаптації під мобільні пристрої. Відтак, актуальною задачею є проектування веб-систем на основі децентралізованої клієнт-серверної архітектури з використанням спеціалізованих фреймворків, що дозволяє розділити зони відповідальності розробників та забезпечити високу інтерактивність користувацького інтерфейсу.[1].

Мета роботи

Метою роботи є проектування загальної архітектури та обґрунтування домену технологічного стеку розробки веб-системи дляуправління бізнес-процесами, що базується на підході SinglePageApplication (SPA) та RESTful API.

Архітектурні рішення та особливості реалізації

Для забезпечення модульності та відмовостійкості, архітектуру систем подібного класу концептуально ділять на три ізольовані технологічні домени:

- ✓ клієнтську площину (Frontend),
- ✓ серверну площину (Backend),
- ✓ площину збереження даних (PersistenceLayer).

Для візуалізації взаємодії цих рівнів та відображення обраного стеку технологій розроблено UML-діаграму компонентів (див.рис.1).

Клієнтський домен (Frontend), як правило, реалізується у вигляді односторінкового додатка (SPA) з використанням сучасних JavaScript/TypeScript бібліотек (наприклад, React.js або Vue.js). Цей підхід переносить процес рендерингу HTML-сторінок із сервера на пристрій клієнта. Взаємодія з сервером зводиться виключно до асинхронного обміну даними (AJAX/Fetch) у форматі JSON, що кардинально зменшує обсяг мережевого трафіку та забезпечує реактивність інтерфейсу без перезавантаження сторінки браузера.

Серверний домен (Backend) зазвичайвиконує роль API-провайдера (на базі ASP.NET Core, Node.js або Python). Сервер проектується за принципом Stateless, де автентифікація кожного запиту відбувається за допомогою криптографічно захищених токенів (JWT). Архітектура бекенду поділяється на контролери (маршрутизація), шар сервісів (бізнес-правила) та шар доступу до даних. Для забезпечення масштабованості, незалежності компонентів та зручності тестування, логічну структуру серверної частини реалізовано на базі трирівневої архітектури (3-Tier Architecture). Програмний код розділено на три ізольовані шари:

- шар контролерів (Presentation / API Layer), якийвідповідає за прийом HTTP-запитів відклієнтськогододатка (абозовнішніхсистем), базовувалідаціювхіднихданих (через DTO) тамаршрутизацію. Контролеринемістятьскладноїлогіки і лише делегують завдання відповідним сервісам;
- шар бізнес-логіки (ServiceLayer), якийє ядром системи. Міститьключовіалгоритми, математичнірозрахункитаправилаобробкиінформації. Сервісинічогонезнаютьпро HTTP-контекстабоспецифікубазиданих, працюючи виключно з доменними моделями;
- шар доступу до даних (Data Access Layer), якийвідповідає за фізичну взаємодію з базою даних. Використання патерну Repository дозволяє інкапсулювати SQL-запити або виклики ORM, надаючи сервісам стандартизований набір методів для збереження та читання даних.

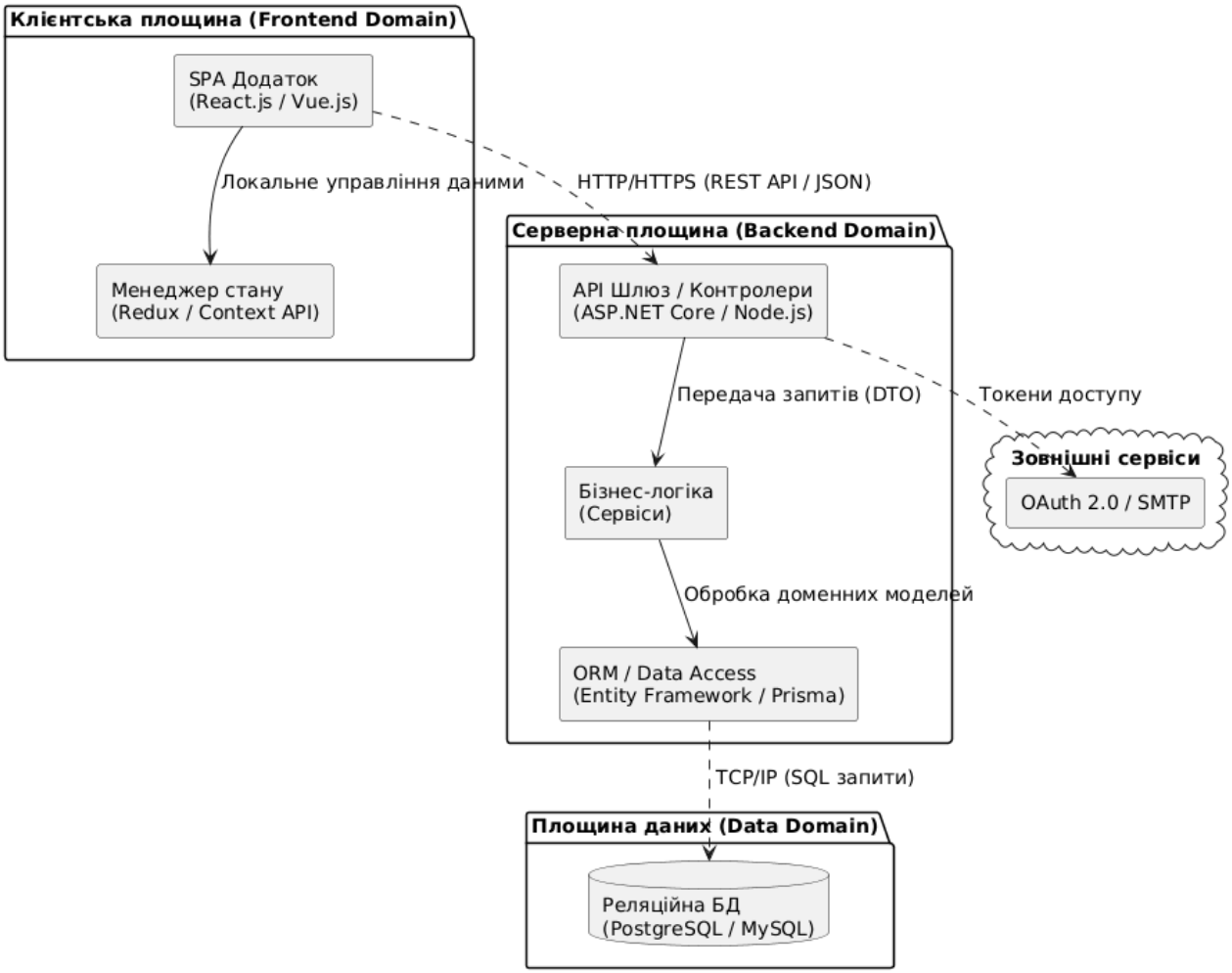

Рисунок 1 – Загальна компонентна архітектура та технологічний стек розробки веб-систем

Взаємодія між цими шарами відбувається за принципом інверсії залежностей (DependencyInversion з SOLID). Шари вищого рівня (контролери) залежать від абстракцій (інтерфейсів) шарів нижчого рівня, а не від їхніх конкретних реалізацій. Це дозволяє здійснювати незалежне модульне тестування бізнес-логіки та легко замінювати окремі компоненти системи без ризику пошкодження іншого коду.

Домен даних (Persistence), де для зберігання інформації використовується реляційна база даних (PostgreSQL або MySQL), гарантує цілісність транзакцій за вимогами ACID. Зв'язок між об'єктно-орієнтованим кодом бекенду та таблицями бази даних автоматизується за допомогою ORM-технологій (Object-RelationalMapping), що захищає систему від SQL-ін'єкцій та прискорює процес розробки.

Висновок

Застосування чітко структурованої клієнт-серверної архітектури з розподілом на технологічні домени є галузевим стандартом інженерії програмного забезпечення. Використання REST API як універсального контракту обміну даними дозволяє незалежно оновлювати клієнтську та серверну частини, а також створює фундамент для легкого масштабування системи або розробки додаткових мобільних клієнтів у майбутньому без переписування основної бізнес-логіки.

Список використаних джерел

1. Мартин Р. Чиста архітектура. Мистецтво розробки програмного забезпечення / пер. з англ. О. Лотоцької. Київ : Фабула, 2019. 368 с.
2. Lock A. ASP.NET Core in Action. 3rd ed. ShelterIsland : Manning Publications, 2023. 856 p.
3. Крепич С.Я., Співак І.Я. Якість програмного забезпечення та тестування: базовий курс. Тернопіль: ФОП Паляниця В.А, 2020. – 478с.
4. Banks A., Porcello E. Learning React: Modern Patterns for Developing React Apps. 2nd ed. O'ReillyMedia, 2020. 334p.
5. Newman S. Building Microservices: Designing Fine-Grained Systems. 2nd ed. O'ReillyMedia, 2021. 614 p.

УДК 004.93:629.7

АРХІТЕКТУРА СИСТЕМИ АВТОНОМНОГО СЛІДКУВАННЯ ЗА ОБ'ЄКТОМ ДЛЯ БПЛА НА ВБУДОВАНІЙ ПЛАТФОРМІ БЕЗ ГРАФІЧНОГО ПРИСКОРЮВАЧА

Крепич Р.В.
Західноукраїнський національний університет
аспірант

І. Актуальність дослідження

Безпілотні літальні апарати (БпЛА) дедалі ширше застосовуються для моніторингу протяжних інфраструктурних об'єктів – ліній електропередач, трубопроводів, промислових майданчиків, де потрібне автономне утримання рухомого об'єкта в полі зору бортової камери впродовж тривалого польоту. Традиційні рішення такої задачі спираються на супутникову навігацію (GPS) та обчислювально потужні графічні прискорювачі, що ускладнює їх застосування у середовищах з відсутнім або нестабільним GPS-сигналом і суперечить вимогам низької вартості та малої маси корисного навантаження. Перспективним напрямом є реалізація візуального слідкування безпосередньо на бортовому одноплатному комп'ютері без графічного прискорювача. Для таких умов ефективними є кореляційні методи трекінгу, що забезпечують високу швидкодію на центральному процесорі [1]. Започаткований фільтром MOSSE [2] клас кореляційних трекерів отримав розвиток у методі CSRT, який поєднує прийнятну точність зі здатністю працювати в реальному часі за обмежених обчислювальних ресурсів. Водночас побудова цілісної системи, що інтегрує бортове візуальне слідкування з контуром керування польотом на доступному апаратному забезпеченні (близько 100 доларів США), залишається актуальною інженерною задачею.

Окрему наукову проблему становить валідація таких систем: характеристики, отримані у програмній симуляції, не завжди відтворюються на реальному обладнанні. Дослідження виявляє двошаровий розрив між симуляцією та реальністю – на рівні сприйняття, де реальна оптика й шуми камери суттєво знижують стійкість окремих кореляційних трекерів, та на рівні оцінювання стану, де бортовий фільтр стану по-різному враховує візуальні вимірювання положення і швидкості. Урахування цих чинників є необхідною умовою коректного переходу від симуляційних експериментів до реальних польотів.

ІІ. Мета роботи

Метою роботи є розроблення архітектури системи автономного слідкування за рухомим об'єктом засобами БпЛА, яка повністю виконується на вбудованій обчислювальній платформі без графічного прискорювача та супутникової навігації, забезпечує утримання об'єкта в полі зору камери в реальному часі й придатна для моніторингу інфраструктурних об'єктів за обмежених вартості та енергоспоживання.

Для досягнення поставленої мети розв'язано такі задачі:

- обґрунтувати вибір кореляційного трекера для роботи на центральному процесорі без графічного прискорювача;
- розробити модульну архітектуру системи з інтеграцією бортового візуального слідкування та контуру керування польотом;
- реалізувати систему на доступному апаратному забезпеченні та провести її експериментальну апробацію.

ІІІ. Архітектурне рішення системи

Архітектуру системи побудовано за модульним принципом із чітким розділенням функцій сприйняття, обробки зображення, оцінювання стану та керування польотом. Система складається з п'яти основних блоків: камери, бортового обчислювача, каналу зв'язку, політного контролера та виконавчих механізмів. Загальну структуру системи наведено на рисунку 1.

Блок сприйняття реалізовано на основі камери Logitech C270, яка формує відеопотік із частотою близько 30 кадрів за секунду. Бортовий обчислювач – одноплатний комп'ютер Raspberry Pi5 – виконує захоплення кадрів і візуальне слідкування за об'єктом кореляційним трекером CSRT, реалізованим засобами бібліотеки OpenCV без використання графічного прискорювача. За координатами обмежувальної рамки (bounding box) обчислюється відхилення об'єкта від центра кадру, яке перетворюється на керуючі команди. Команди передаються до політного контролера Pixhawk 6C Pro з відкритою автопілотною платформою PX4 цифровим каналом MAVLink (USB, 921600 бод). У режимі OFFBOARD політний контролер виконує оцінювання стану апарата

розширеним фільтром Калмана (EKF2) та формує сигнали для виконавчих механізмів – регуляторів обертів і двигунів. У такий спосіб утворюється замкнений контур керування: зміщення об’єкта в кадрі коригує курс і швидкість БпЛА, повертаючи об’єкт до центра поля зору. Реалізація обчислювальної частини на платформі Raspberry Pi 5 забезпечує автономність системи за низьких вартості та маси.

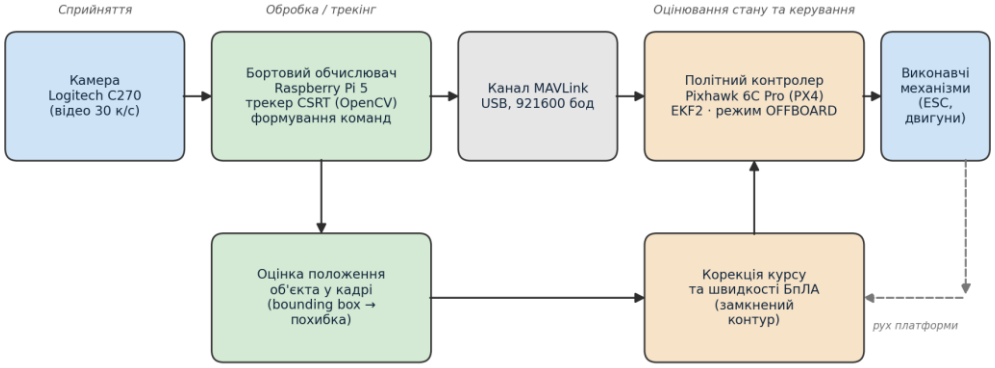

Рисунок 1 – Архітектура системи автономного слідкування за об’єктом на вбудованій платформі

IV. Експериментальна апробація

Працездатність запропонованої архітектури перевірено в напівнатурному середовищі та у послідовності керованих сценаріїв. Базовий сценарій передбачає ручне виведення апарата у режим зависання, перехід у режим OFFBOARD з автономним утриманням обраного об’єкта в кадрі та подальший вихід із режиму з безпечною стабілізацією. За попередніми експериментальними даними трекер CSRT забезпечує опрацювання відеопотоку в реальному часі із затримкою менше 100 мс та середнім відхиленням центра об’єкта від центра кадру в межах 10–15% його ширини; швидші, але менш стійкі трекери (зокрема MOSSE) за реальних оптичних умов втрачають точність у рази. Основні характеристики системи наведено в таблиці 1.

Таблиця 1

Основні характеристики системи

Характеристика	**Значення**
Обчислювальна платформа	Raspberry Pi 5 (CPU-only, без GPU)
Камера	Logitech C270, ~30 к/с
Метод візуального слідкування	CSRT (OpenCV)
Затримка опрацювання кадру	менше 100 мс
Середнє відхилення від центра кадру	10–15% ширини кадру
Польотний контролер	Pixhawk 6C Pro (PX4), OFFBOARD
Канал зв’язку	MAVLink (USB, 921600 бод)
Орієнтовна вартість апаратної частини	близько 100 USD

Висновок

У роботі запропоновано модульну архітектуру системи автономного слідкування за рухомим об’єктом засобами БпЛА, що повністю функціонує на вбудованій платформі Raspberry Pi 5 без графічного прискорювача та супутникової навігації. Візуальне слідкування реалізовано кореляційним трекером CSRT, а керування польотом – польотним контролером Pixhawk 6C Pro (PX4) у режимі OFFBOARD, поєднаним з бортовим обчислювачем каналом MAVLink. Експериментальна апробація підтвердила опрацювання відеопотоку в реальному часі та виявила потребу врахування двошарового розриву між симуляцією і реальністю на етапі переходу до натурних польотів. Модульність і використання доступного апаратного забезпечення роблять систему придатною для моніторингу інфраструктурних об’єктів за обмежених вартості й енергоспоживання та створюють основу для подальшої валідації в реальних польотних умовах.

Список використаних джерел

1. M. Dyvak, R. Krepych, S. Krepych, and I. Spivak, “Correlation-Based Methods for Tracking Moving Objects Using UAS Systems,” in Proc. IEEE Int. Conf. Advanced Computer Information Technologies (ACIT), 2025.
2. D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, “Visual Object Tracking using Adaptive Correlation Filters,” in Proc. IEEE CVPR, 2010, pp. 2544–2550. https://doi.org/10.1109/CVPR.2010.5539960.

УДК 004.415.2

РОЗРОБКА ВЕБ-ОРІЄНТОВАНОГО ЗАСТОСУНКУ ПРОГРАМНОГО РЕЄСТРАТОРА РОЗРАХУНКОВИХ ОПЕРАЦІЙ ДЛЯ АВТОМАТИЗАЦІЇ ОБЛІКУ ТА ФІСКАЛІЗАЦІЇ ТОРГОВЕЛЬНОЇ ДІЯЛЬНОСТІ

Небесьо Л.Й.[1], **Юшко А.В.**[2]
Західноукраїнський національний університет
[1]бакалавр; [2]д.філ., ст. викладач

I.Постановка проблеми

Сучасні підприємства роздрібної торгівлі потребують ефективних інструментів автоматизації обліку розрахункових операцій та забезпечення їх фіскалізації відповідно до вимог законодавства України. Існуючі програмні рішення часто є або надто складними у використанні, або дорогими для малого та середнього бізнесу.

Крім того, значна частина рішень має обмежену адаптивність, складний інтерфейс та недостатню інтеграцію з сучасними веб-технологіями. Це ускладнює процес обслуговування клієнтів та ведення обліку.

II. Мета роботи

Метою роботи є проєктування та розробка веб-орієнтованого застосунку програмного реєстратора розрахункових операцій, який забезпечує автоматизацію торговельного обліку, реєстрацію розрахункових операцій, формування електронних фіскальних чеків, ведення історії продажів, управління товарами та реалізацію базових функцій сучасної касової POS-системи

III. Архітектура та програмна реалізація

Архітектура системи побудована за клієнт-серверноюмоделлю, де frontend забезпечує взаємодію з користувачем, а backend реалізує бізнес-логіку та обробку запитів.

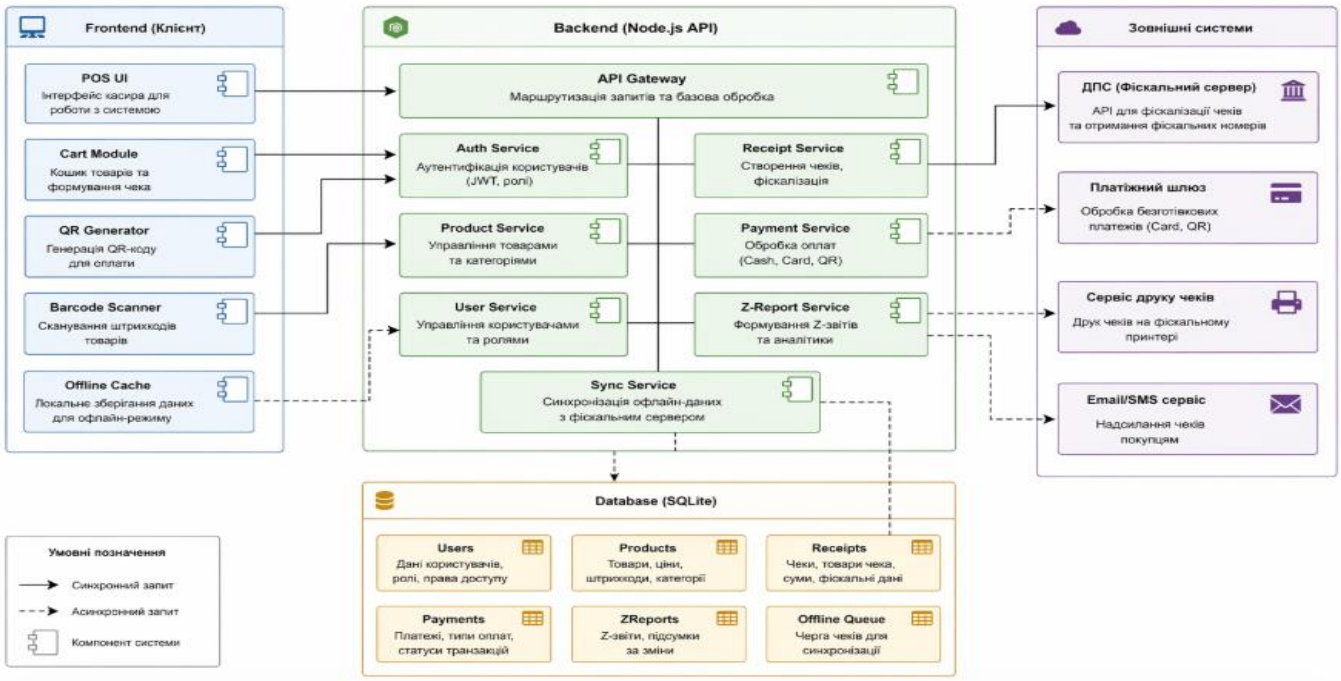

Рисунок 1 – Компонентна діаграма ПРРО системи

Компонентна діаграма, показана на рисунку 1, демонструє розподіл системи на окремі модулі, зокрема інтерфейс користувача, серверну частину, базу даних та зовнішні сервіси. Такий підхід забезпечує масштабованість, гнучкість та зручність підтримки програмного продукту.

Взаємодія компонентів системи під час здійснення продажу відображена на діаграмі послідовності (див.рис.2).

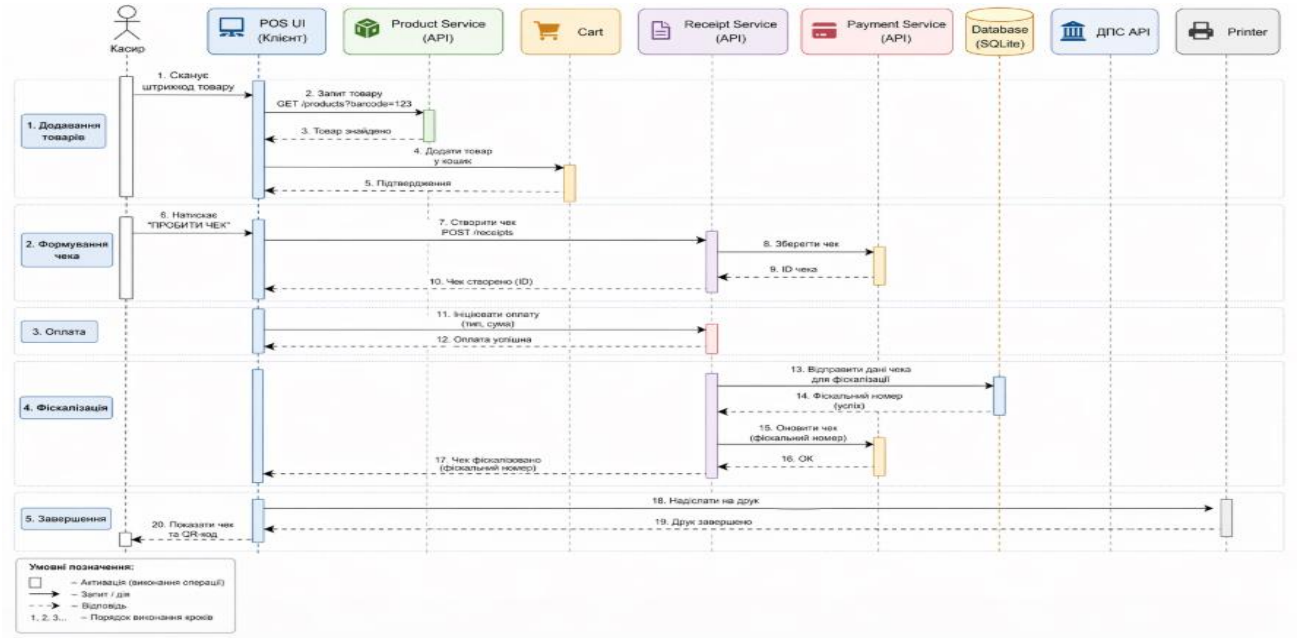

Рисунок 2 – Діаграма послідовності процесу продажу

Дана діаграма ілюструє послідовність дій від моменту сканування товару до формування та фіскалізації чека. Вона демонструє взаємодію між користувачем, клієнтським інтерфейсом, сервером та зовнішніми системами.

Для забезпечення безпеки доступу до системи використано механізм JWT-аутентифікації та рольову модель, які відображені на рисунку 3.

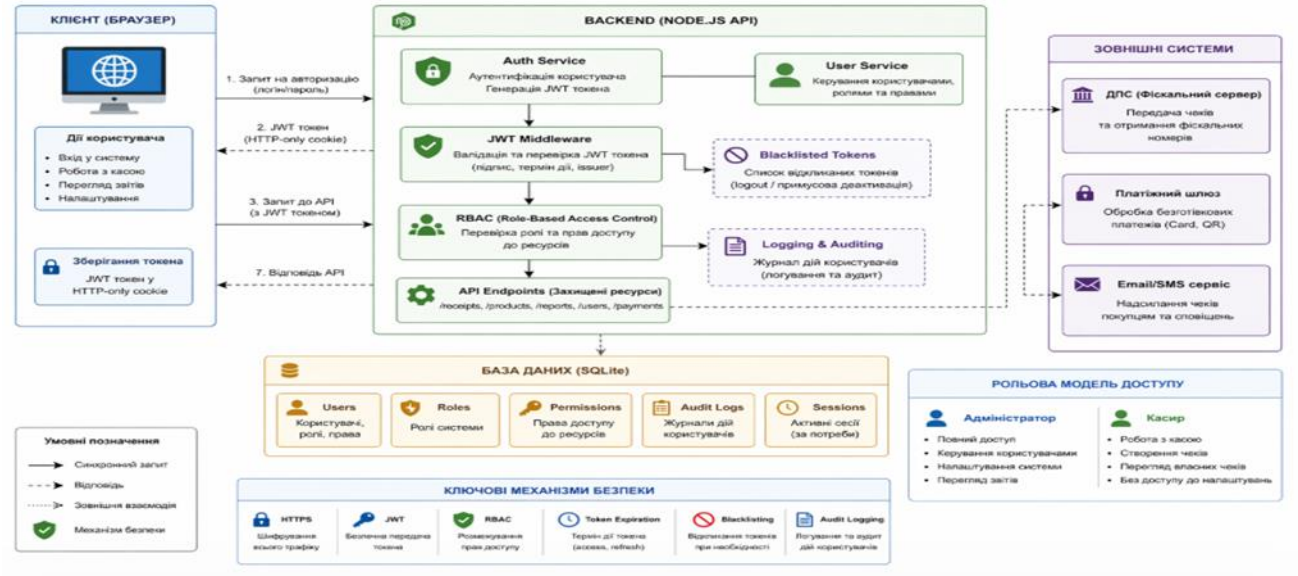

Рисунок 3 – Архітектура безпеки ПРРО системи

Застосування токенів дозволяє захистити API та забезпечити розмежування прав доступу між користувачами системи.

Висновки

У результаті виконання роботи було розроблено веб-орієнтований програмний реєстратор розрахункових операцій, який забезпечує автоматизацію обліку та фіскалізації торговельної діяльності.Запропоноване рішення характеризується: простотою використання; гнучкою архітектурою; можливістю масштабування; відповідністю сучасним вимогам до програмних систем.

Список літератури

1. Закон України «Про застосування реєстраторів розрахункових операцій у сфері торгівлі, громадського харчування та послуг» : Закон України від 06.07.1995 № 265/95-ВР. URL: https://zakon.rada.gov.ua/laws/show/265/95-вр.
2. Node.js Foundation. Node.js Documentation. 2026. URL: https://nodejs.org/en/docs.
3. SQLiteConsortium. SQLiteDocumentation. 2026. URL: https://www.sqlite.org/docs.html.

УДК 004.738.5:004.42:004.65.

ВЕБ-СЕРВІС ДЛЯ РЕЦЕНЗУВАННЯ МУЗИКИ

Берекета Т.М.[1)], **Куйдич О.О.**[2)], **Бодян О.А.**[3)]
Західноукраїнський національний університет
[1-3)]*бакалавр*

І. Постановка проблеми

Сучасні музичні сервіси надають користувачам доступ до мільйонів композицій, альбомів і виконавців, що, з одного боку, розширює можливості вибору, а з іншого — ускладнює процес пошуку якісного та релевантного контенту. У таких умовах особливого значення набувають системи рецензування та рейтингування музики, які дозволяють користувачам ділитися власними враженнями, оцінювати музичні твори та формувати колективну думку щодо їхньої якості. Наявні музичні платформи переважно орієнтовані на потокове відтворення аудіоконтенту та рекомендаційні механізми, тоді як функції створення змістовних рецензій, формування тематичних спільнот і накопичення експертних оцінок часто мають обмежений характер. Водночас активний розвиток соціальних веб-технологій створює передумови для розроблення спеціалізованих сервісів, у межах яких користувачі можуть не лише оцінювати музичні твори, а й брати участь у професійних та аматорських дискусіях щодо музичного контенту. Створення веб-сервісу для рецензування музики дозволяє поєднати можливості систем управління контентом, соціальних мереж та платформ колективного оцінювання. Такий підхід забезпечує централізоване зберігання рецензій, формування рейтингів музичних творів, альбомів і виконавців, підтримку комунікації між користувачами та підвищення ефективності пошуку нової музики на основі накопиченого досвіду спільноти[1,2].

ІІ. Мета роботи

Метою статті є розроблення веб-сервісу для рецензування музики, який забезпечує можливість публікації та аналізу користувацьких рецензій, оцінювання музичних творів і альбомів, формування рейтингів, організацію соціальної взаємодії між користувачами та підтримку процесів пошуку й рекомендації музичного контенту.

ІІІ. Проектування архітектури веб-сервісу

Архітектуру веб-сервісу для рецензування музики доцільно побудувати за клієнт-серверним принципом із виокремленням рівня користувацького інтерфейсу, серверної логіки, бази даних та зовнішніх сервісів. Такий підхід забезпечує масштабованість системи, зручність супроводу програмного коду та можливість подальшого розширення функціоналу.Клієнтська частина веб-сервісу відповідає за взаємодію користувача із системою. Через веб-інтерфейс користувач може реєструватися, авторизуватися, переглядати музичні альбоми й композиції, створювати рецензії, виставляти оцінки, залишати коментарі та переглядати рейтинги. Основна увага під час проєктування інтерфейсу приділяється зручності навігації, швидкому доступу до музичного контенту та простоті створення рецензій.Серверна частина реалізує основну бізнес-логіку веб-сервісу. Вона забезпечує обробку запитів користувачів, перевірку прав доступу, збереження рецензій, обчислення середніх рейтингів, формування списків популярних альбомів і виконавців, а також обробку коментарів та реакцій користувачів. Для взаємодії між клієнтською і серверною частинами доцільно використовувати REST API, що дозволяє уніфікувати обмін даними та спростити інтеграцію з іншими платформами.

База даних призначена для зберігання інформації про користувачів, музичні твори, альбоми, виконавців, рецензії, оцінки, коментарі та жанри. Окремо можуть зберігатися службові дані, пов’язані з історією дій користувачів, модерацією контенту та формуванням рекомендацій.Зовнішні сервіси можуть використовуватися для отримання додаткової інформації про музичний контент, зокрема назв альбомів, обкладинок, списків композицій, жанрової належності та даних про виконавців. Це дає змогу зменшити обсяг ручного введення інформації та підвищити повноту музичного каталогу.У загальному вигляді архітектура веб-сервісу передбачає взаємодію таких основних модулів: модуля автентифікації користувачів, модуля управління музичним каталогом, модуля рецензування, модуля рейтингування, модуля коментування, модуля пошуку та фільтрації, модуля модерації контенту та адміністративної панелі. Запропонована архітектура створює основу для реалізації функціонального, надійного та розширюваного веб-сервісу для рецензування музики, рисунок 1.

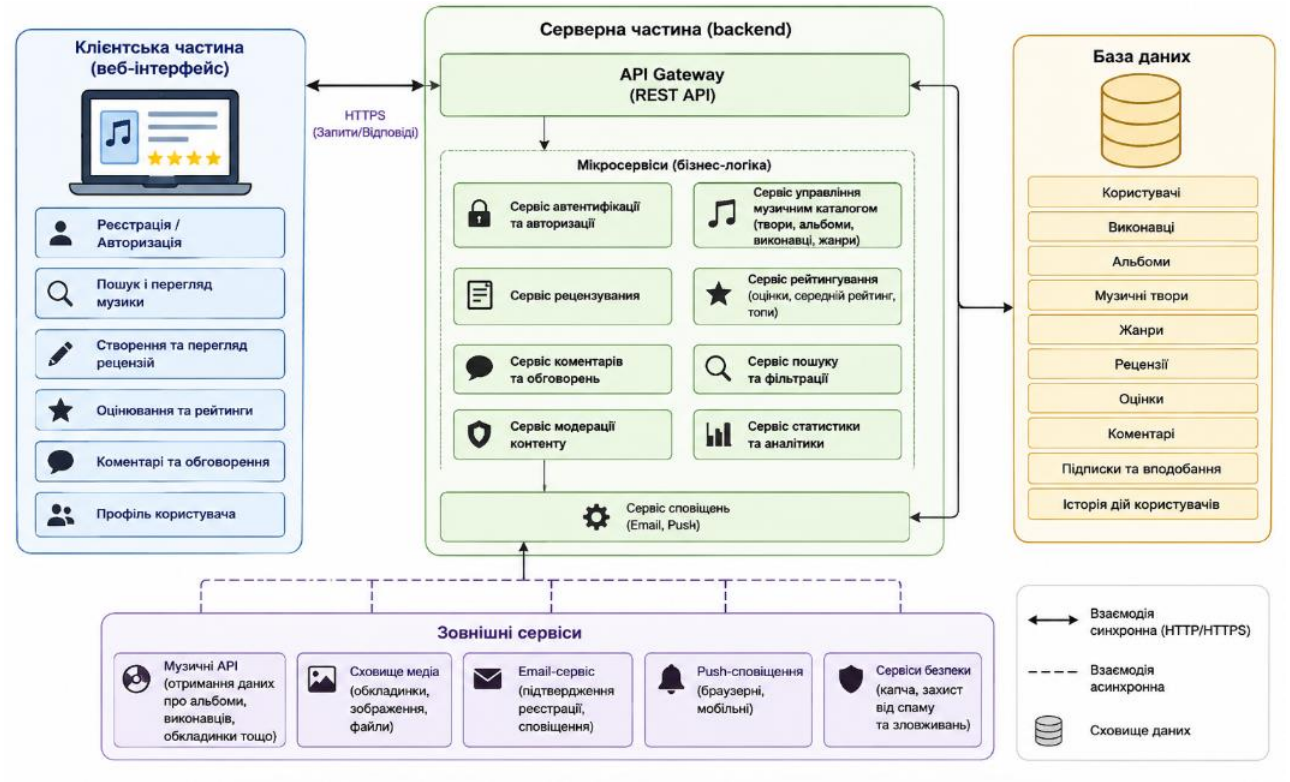

Рисунок 1 – Архітектура веб-сервісу рецензування музики

Програмна реалізація веб-сервісу для рецензування музики здійснюється на основі багаторівневої клієнт-серверної архітектури, яка забезпечує розподіл функціональності між інтерфейсом користувача, серверною частиною та системою зберігання даних. Такий підхід сприяє підвищенню продуктивності, спрощує супровід програмного забезпечення та забезпечує можливість подальшого масштабування системи.Клієнтська частина веб-сервісу реалізована із використанням сучасних вебтехнологій HTML5, CSS3 та JavaScript. Для побудови інтерактивного користувацького інтерфейсу доцільним є використання фреймворку React, який забезпечує компонентний підхід до розроблення та ефективне оновлення вмісту сторінок без повного їх перезавантаження. Інтерфейс користувача реалізує функції реєстрації та авторизації, перегляду музичного каталогу, створення та редагування рецензій, оцінювання музичних творів, роботи з коментарями та керування персональним профілем.Серверна частина системи реалізована з використанням платформи Node.js та фреймворку Express.js. Основним завданням сервера є обробка HTTP-запитів, виконання бізнес-логіки, управління доступом до даних та взаємодія із зовнішніми сервісами. Для організації взаємодії між клієнтом та сервером використовується архітектурний стиль REST API, який забезпечує стандартизований обмін даними у форматі JSON.Для забезпечення безпеки доступу до ресурсів системи використовується механізм автентифікації на основі JSON Web Token (JWT). Після успішної авторизації користувач отримує токен доступу, який використовується під час подальших запитів до серверної частини. Такий підхід дозволяє реалізувати безпечну систему керування сесіями та розмежування прав доступу.Запропонована програмна реалізація забезпечує високу швидкодію веб-сервісу, надійне зберігання даних, зручність користування та можливість подальшого розширення функціональних можливостей системи відповідно до потреб музичної спільноти.

Висновок

У статті розглянуто особливості проєктування веб-сервісу для рецензування музики, призначеного для накопичення, зберігання та опрацювання користувацьких оцінок і рецензій на музичний контент. Запропонована архітектура забезпечує ефективну взаємодію між користувачами та системою, а також створює передумови для подальшого розширення функціональних можливостей сервісу.

Список використаних джерел

1. Pachet, F.; Roy, P. Improving Recommendation Diversity with Playlist Generation. Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI) 2008, 25–32. https://doi.org/10.5555/1625275.1625280.
2. Schedl, M.; Gómez, E.; Urbano, J. Music Information Retrieval: Recent Developments and Applications. Foundations and Trends in Information Retrieval 2014, 8, 127–261. https://doi.org/10.1561/1500000042.

УДК 004.7:004.89

ЗНАННЯ-ОРІЄНТОВАНА ПРОГРАМНА СИСТЕМА УПРАВЛІННЯ КОРПОРАТИВНИМИ КОМП'ЮТЕРНИМИ МЕРЕЖАМИ

Попик Ю.І.
Західноукраїнський національний університет
аспірант

І. Постановка проблеми

Сучасні корпоративні комп'ютерні мережі являють собою складні розподілені інформаційні середовища, що об'єднують сервери, мережеве обладнання, програмні сервіси та велику кількість користувачів. Постійне зростання масштабів мережевої інфраструктури та обсягів інформаційних потоків ускладнює процеси адміністрування, моніторингу, розподілу ресурсів і забезпечення надійного функціонування мережі. Традиційні підходи до управління мережами переважно базуються на засобах моніторингу та наборі заздалегідь визначених правил, що обмежує можливості автоматизованого прийняття рішень в умовах динамічних змін[1].Одним із перспективних напрямів підвищення ефективності управління мережевою інфраструктурою є використання знання-орієнтованих технологій, які поєднують засоби подання знань, онтологічного моделювання та логічного виведення. Такий підхід дає змогу формалізувати структуру корпоративної мережі, описати взаємозв'язки між її компонентами та автоматизувати процес підтримки прийняття управлінських рішень[2].У статті запропоновано знання-орієнтовану програмну систему управління корпоративними комп'ютерними мережами, яка базується на онтологічному поданні знань і механізмах семантичного аналізу. Розроблений підхід забезпечує підвищення ефективності адміністрування мережевих ресурсів, своєчасне виявлення проблемних ситуацій та підтримку процесів прийняття рішень щодо управління корпоративною мережею.

ІІ. Мета роботи

Метою статті є розроблення знання-орієнтованої програмної системи управління корпоративними комп'ютерними мережами на основі онтологічного подання знань та механізмів семантичного виведення, яка забезпечує автоматизацію процесів моніторингу, аналізу стану мережевої інфраструктури, підтримки прийняття управлінських рішень і підвищення ефективності використання мережевих ресурсів.

ІІІ. Архітектура знання-орієнтованої програмної системи управління корпоративними комп'ютерними мережами

Архітектура запропонованої знання-орієнтованої програмної системи управління корпоративними комп'ютерними мережами розроблена відповідно до принципів модульності, масштабованості та семантичної інтеграції знань. Її основне призначення полягає у забезпеченні інтелектуалізованої підтримки процесів адміністрування мережевої інфраструктури шляхом поєднання засобів моніторингу, онтологічного подання знань та механізмів логічного виведення.На відміну від традиційних систем мережевого управління, які переважно реалізують функції збору та візуалізації даних, запропонована система забезпечує формалізацію знань про структуру корпоративної мережі, її ресурси, сервіси, користувачів та політики функціонування. Це дозволяє не лише накопичувати інформацію про поточний стан мережі, а й виконувати її семантичний аналіз для автоматизованого формування управлінських рішень.

Концептуально архітектура системи складається з декількох взаємопов'язаних рівнів. На нижньому рівні розташовано підсистему моніторингу, яка здійснює безперервний збір даних від серверів, комутаторів, маршрутизаторів, служб каталогів та інших компонентів корпоративної мережі. Отримана інформація проходить етап попереднього опрацювання, під час якого виконуються очищення, нормалізація та перетворення даних до формату, придатного для подальшого семантичного аналізу.Центральним елементом архітектури є онтологічне сховище знань, яке містить формалізований опис предметної області управління мережею. В онтології визначаються класи мережевих пристроїв, серверів, користувачів, сервісів, мережевих сегментів, подій та адміністративних політик, а також відношення між ними. Таке представлення забезпечує єдину семантичну модель корпоративної мережі та створює основу для застосування механізмів автоматизованого логічного виведення.На основі актуалізованої онтології функціонує механізм логічного виведення, який реалізує аналіз стану мережі шляхом використання набору продукційних правил та семантичних обмежень. Результатом роботи даного модуля є виявлення перевантажених

серверів, конфліктів конфігурацій, порушень політик безпеки, а також інших ситуацій, що потребують втручання адміністратора або автоматизованої реакції системи.Сформовані висновки передаються до модуля підтримки прийняття рішень, який здійснює генерацію рекомендацій або безпосереднє формування керувальних впливів. Залежно від характеру виявленої проблеми система може ініціювати процедури балансування навантаження між термінальними серверами, перенаправлення користувачів, зміну параметрів доступу до ресурсів або виконання інших адміністративних операцій.Завершальним елементом архітектури є підсистема взаємодії з адміністратором, яка забезпечує візуалізацію поточного стану мережі, результатів логічного аналізу та сформованих рекомендацій. Таким чином, реалізується замкнений цикл управління, у межах якого дані про стан мережі трансформуються у знання, а знання використовуються для підтримки процесів прийняття рішень.Структурну архітектуру системи наведено на рисунку 1.

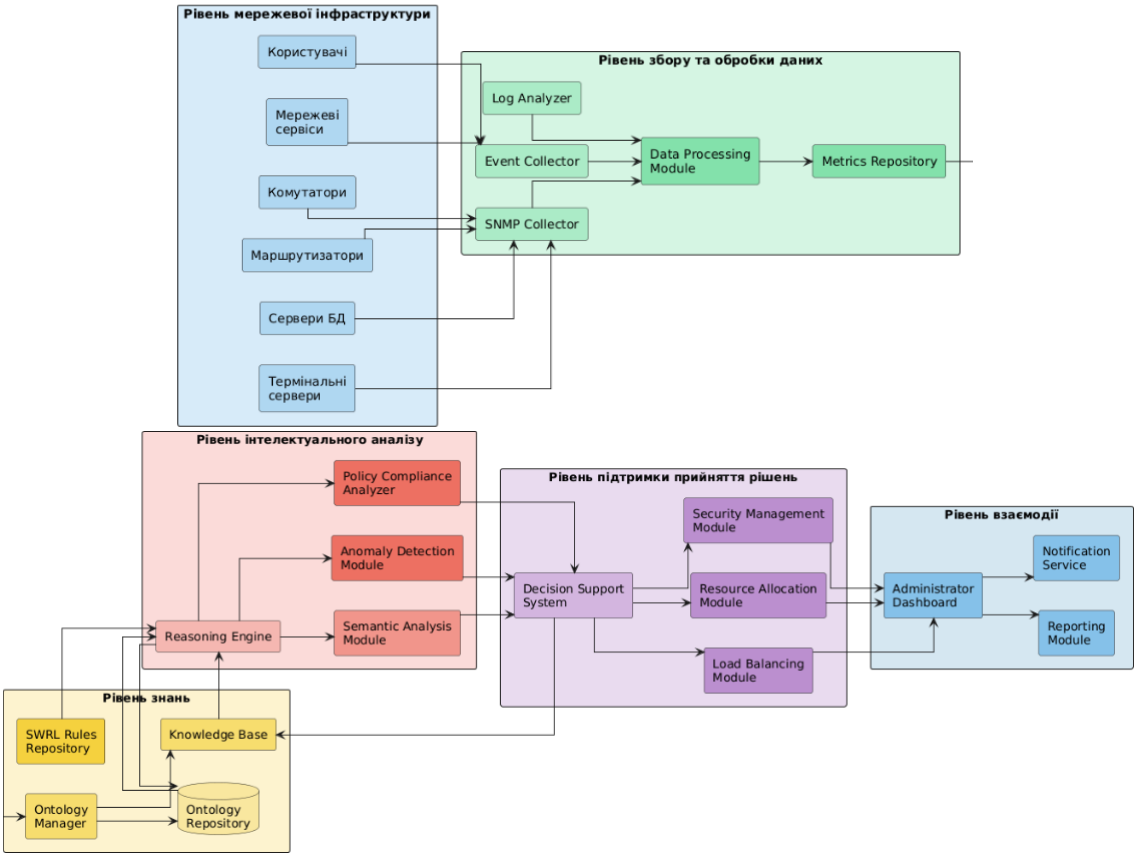

Рисунок 1 – Архітектура знання-орієнтованої системи

Розроблена архітектура забезпечує відокремлення процесів збору даних від процесів представлення знань та прийняття рішень, що сприяє підвищенню масштабованості системи та спрощує її адаптацію до корпоративних мереж різної структури. Крім того, використання онтологічного підходу створює передумови для інтеграції нових типів мережевих ресурсів, розширення бази знань та вдосконалення механізмів автоматизованого управління без суттєвої зміни загальної архітектури програмної системи.

Висновок

У статті запропоновано знання-орієнтовану програмну систему управління корпоративними комп'ютерними мережами, яка поєднує засоби моніторингу мережевої інфраструктури, онтологічного подання знань та механізми логічного виведення для підтримки процесів прийняття управлінських рішень. Розроблена архітектура забезпечує формалізацію знань про мережеві ресурси, користувачів, сервіси та політики функціонування корпоративної мережі в межах єдиного семантичного простору.

Список використаних джерел

1. Khan, M.A.; Shah, S.A.; Ahmed, A.; Alouini, M.-S. A Comprehensive Survey on Knowledge-Defined Networking. Telecom 2023, 4(3), 477–596. https://doi.org/10.3390/telecom4030025.
2. Apajalahti, K.; Hyvönen, E.; Niiranen, J.; Räisänen, V. Combining Ontological Modelling and Probabilistic Reasoning for Network Management. Applied Intelligence 2017, 46(4), 915–937. https://doi.org/10.3233/AIS-160419.

УДК 004.42:004.738.5:004.774

РОЗРОБКА TELEGRAM-БОТА-КОНСТРУКТОРА ДЛЯ ЗВОРОТНОГО ЗВ'ЯЗКУ

Гаврилишин В.В.[1)], **Яськов В.В.**[2)], **Галієвський Р.Р.**[3)]
Західноукраїнський національний університет
[1-3)]*бакалавр*

I. Постановка проблеми

У сучасних умовах цифровізації ефективний зворотний зв'язок між організаціями та користувачами є важливою складовою успішного функціонування інформаційних систем. Використання месенджерів як каналів комунікації дозволяє значно спростити процес обміну інформацією, забезпечити оперативне реагування на звернення користувачів та підвищити якість надання послуг.

Однією з найбільш популярних платформ для реалізації автоматизованої взаємодії є Telegram, який надає широкі можливості для створення чат-ботів. Проте розроблення ботів потребує спеціальних знань у галузі програмування, що обмежує можливість їх використання представниками малого бізнесу, освітніх установ та громадських організацій.Перспективним напрямом є створення Telegram-ботів-конструкторів, які дозволяють формувати функціональність ботів за допомогою графічного інтерфейсу без написання програмного коду. Такий підхід спрощує процес створення каналів зворотного зв'язку та забезпечує гнучке налаштування сценаріїв взаємодії з користувачами[1].

II. Мета роботи

Метою статті є розроблення Telegram-бота-конструктора для організації зворотного зв'язку, який забезпечує автоматизоване створення ботів без необхідності програмування, збір повідомлень користувачів, обробку звернень, налаштування форм взаємодії та централізоване управління каналами комунікації.

III. Особливості реалізації бота

Програмна реалізація Telegram-бота-конструктора для зворотного зв'язку базується на використанні клієнт-серверної архітектури та TelegramBot API, що забезпечує автоматизовану взаємодію між користувачами месенджера та програмною системою. Основною особливістю розробленого рішення є можливість створення та налаштування функціональності ботів без безпосереднього втручання у програмний код, що значно спрощує процес розгортання системи для кінцевих користувачів.Серверна частина реалізована з використанням платформи Node.js та фреймворку Express.js, які забезпечують обробку HTTP-запитів, взаємодію з TelegramBot API та реалізацію бізнес-логіки системи. Для організації обміну даними між клієнтською частиною та сервером використовується REST API, що дозволяє виконувати операції створення, редагування та видалення ботів, налаштування сценаріїв взаємодії та керування повідомленнями користувачів.

Одним із ключових компонентів системи є модуль керування ботами, який забезпечує реєстрацію Telegram-ботів за допомогою токенів доступу, отриманих через сервіс BotFather. Після підключення бота система автоматично створює необхідні конфігураційні параметри та забезпечує його інтеграцію з платформою конструктора.Для зберігання інформації використовується реляційна база даних MySQL. У базі даних зберігаються облікові записи адміністраторів, налаштування ботів, сценарії діалогів, повідомлення користувачів, категорії звернень, історія взаємодії та статистична інформація. Взаємодія з базою даних реалізується за допомогою ORM-технологій, що спрощують виконання операцій створення, читання, оновлння та видалення даних.

Модуль конструктора сценаріїв реалізовано у вигляді візуального редактора, який дозволяє створювати структуру діалогу шляхом додавання повідомлень, кнопок, умов переходу та форм введення інформації. Кожен сценарій зберігається у вигляді набору взаємопов'язаних елементів, що дає змогу динамічно формувати логіку роботи бота без перезапуску системи.Для приймання повідомлень від користувачів використовується механізм Webhook, який забезпечує оперативне отримання подій від Telegram-серверів. Після надходження повідомлення система виконує його аналіз, визначає поточний стан сценарію та формує відповідну реакцію згідно з налаштованою логікою. Такий підхід дозволяє забезпечити швидку обробку звернень та мінімізувати навантаження на сервер.

З метою підвищення безпеки реалізовано механізми автентифікації та авторизації користувачів адміністративної панелі на основі JSON WebToken (JWT). Передача даних між компонентами системи здійснюється через захищені канали зв'язку з використанням протоколу HTTPS. Додатково передбачено журналювання дій адміністраторів і резервне копіювання критично важливих даних.

Запропонована програмна реалізація забезпечує високу продуктивність, масштабованість та можливість одночасного обслуговування великої кількості ботів і користувачів. Модульна структура програмного забезпечення дозволяє легко розширювати функціональні можливості системи шляхом інтеграції нових сервісів, аналітичних модулів та засобів автоматизованої обробки повідомлень

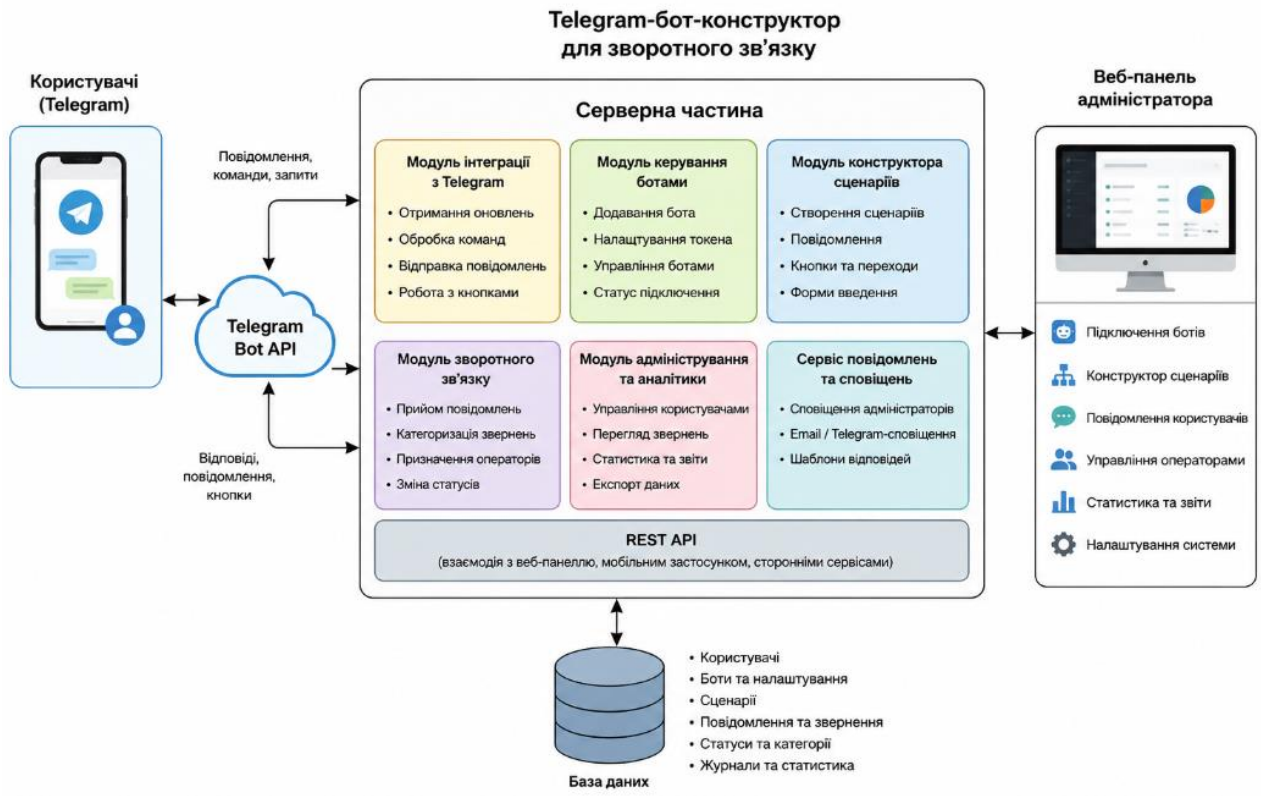

Рисунок 1 – Схема реалізації бота

Програмна реалізація Telegram-бота-конструктора виконана на основі платформи Node.js із використанням TelegramBot API для забезпечення взаємодії з користувачами месенджера. Серверна частина системи відповідає за обробку повідомлень, керування сценаріями роботи ботів, збереження налаштувань та маршрутизацію звернень.Для зберігання інформації про користувачів, ботів, сценарії діалогів і повідомлення використовується реляційна база даних MySQL. Взаємодія між компонентами системи реалізована через REST API, що забезпечує централізоване керування функціональністю конструктора. Для підвищення безпеки застосовано механізми автентифікації на основі JWT та захищений обмін даними через HTTPS.Розроблене програмне забезпечення підтримує створення та налаштування Telegram-ботів без програмування, забезпечує автоматизований збір звернень користувачів і може бути легко розширене додатковими модулями аналітики та інтелектуальної обробки повідомлень.

Висновок

У статті розглянуто особливості розроблення Telegram-бота-конструктора для організації зворотного зв'язку між користувачами та організаціями. Запропоновано архітектуру системи, яка забезпечує автоматизоване створення та налаштування ботів без необхідності програмування, а також централізоване керування сценаріями взаємодії та повідомленнями користувачів.У результаті дослідження реалізовано програмне рішення, що інтегрується з TelegramBot API та надає можливість створення гнучких сценаріїв обробки звернень, збору повідомлень і формування відповідей. Розроблена система забезпечує ефективну організацію комунікації, спрощує процес впровадження чат-ботів та зменшує витрати на їх підтримку.

Список використаних джерел

Fiadotau, M.; Rusanova, D. Managing Telegram Bots in Educational and Information Systems: Design Approaches and Implementation Features. Information 2023, 14(5), 271. https://doi.org/10.3390/info14050271.

УДК 004.738.5:004.42:004.65.

РОЗРОБЛЕННЯ МОБІЛЬНОГО СЕРВІСУ ДЛЯ АВТОМАТИЗАЦІЇ ПРОЦЕСІВ ДОГЛЯДУ ЗА СОБАКАМИ

Марценюк М.О.[1)], **Кульматицький М.Р.**[2)], **Бойко О.Ю.**[3)]
Західноукраїнський національний університет
[1-3)]*бакалавр*

І. Постановка проблеми

Домашні тварини є важливою складовою життя сучасного суспільства, а забезпечення належного догляду за ними потребує постійного контролю за станом здоров'я, харчуванням, фізичною активністю та виконанням профілактичних заходів. Власники собак стикаються з необхідністю своєчасного проведення вакцинації, ветеринарних оглядів, протипаразитарної обробки, дотримання режиму годування та фізичних навантажень. Невиконання цих процедур може негативно вплинути на здоров'я тварини та призвести до додаткових фінансових витрат на лікування. Розвиток мобільних технологій створює широкі можливості для автоматизації процесів догляду за домашніми тваринами. Сучасні мобільні сервіси дозволяють централізовано зберігати інформацію про тварину, планувати необхідні заходи та отримувати нагадування про важливі події. Водночас більшість існуючих рішень орієнтовані на виконання окремих функцій і не забезпечують комплексної підтримки процесів догляду за собакою в межах єдиного програмного середовища. У зв'язку з цим актуальним є розроблення мобільного сервісу, який поєднує функції обліку інформації про тварину, управління графіком процедур, контролю медичних показників та інформаційної підтримки власника. Використання такого сервісу дозволяє підвищити якість догляду за собакою, забезпечити своєчасне виконання профілактичних заходів та спростити процес взаємодії з ветеринарними фахівцями[1,2].

ІІ. Мета роботи

Метою статті є розроблення мобільного сервісу для автоматизації процесів догляду за собаками, який забезпечує планування повсякденних процедур, моніторинг стану тварини, ведення електронної картки здоров'я, формування нагадувань про вакцинацію та ветеринарні огляди, а також підтримку взаємодії власників із ветеринарними спеціалістами.

ІІІ. Проектування архітектури мобільного сервісу

Архітектуру мобільного сервісу для автоматизації процесів догляду за собаками доцільно побудувати за клієнт-серверним принципом, що забезпечує розподіл функцій між мобільним застосунком, серверною частиною, базою даних та зовнішніми сервісами. Такий підхід дозволяє централізовано зберігати інформацію про тварин, синхронізувати дані між пристроями користувача, формувати нагадування та забезпечувати доступ до історії догляду за собакою.

Клієнтська частина системи реалізується у вигляді мобільного застосунку, через який власник собаки може створювати профіль тварини, зазначати її породу, вік, вагу, стан здоров'я, графік годування, прогулянок, вакцинації та ветеринарних оглядів. Також мобільний застосунок забезпечує перегляд нагадувань, ведення щоденника догляду, збереження медичних записів і отримання рекомендацій щодо повсякденного догляду.Серверна частина відповідає за обробку запитів користувачів, автентифікацію, керування профілями тварин, збереження даних, формування розкладу подій і взаємодію з модулем сповіщень. Взаємодія між мобільним застосунком і сервером здійснюється через REST API, що забезпечує стандартизований обмін даними у форматі JSON.

База даних використовується для зберігання інформації про користувачів, собак, ветеринарні записи, графіки догляду, вакцинації, нагадування, історію годування, прогулянок і медичних процедур. Окремий модуль сповіщень забезпечує надсилання push-повідомлень про заплановані події, зокрема вакцинацію, прийом ліків, годування або візит до ветеринара.Додатково архітектура може передбачати інтеграцію із зовнішніми сервісами, зокрема картографічними сервісами для пошуку ветеринарних клінік і місць для вигулу собак, а також сервісами хмарного зберігання для резервного копіювання медичних документів.Запропонована архітектура забезпечує гнучкість, масштабованість і можливість подальшого розширення функціональності мобільного сервісу, зокрема додавання рекомендаційного модуля, інтеграції з ветеринарними кабінетами або підтримки декількох тварин в одному обліковому записі.

Програмна реалізація мобільного сервісу для автоматизації процесів догляду за собаками виконана на основі сучасних технологій розроблення мобільних та веборієнтованих програмних

систем. Основною метою реалізації є забезпечення зручного доступу користувачів до інформації про домашніх тварин, автоматизація планування процедур догляду та своєчасне інформування про заплановані події.Клієнтська частина системи реалізована за допомогою фреймворкуFlutter, який дозволяє створювати кросплатформні мобільні застосунки для операційних систем Android та iOS на основі єдиної кодової бази. Використання Flutter забезпечує високу продуктивність застосунку, швидке відображення інтерфейсу та можливість подальшого розширення функціональних можливостей системи. Інтерфейс користувача побудований відповідно до принципів адаптивного дизайну, що забезпечує коректне відображення елементів на мобільних пристроях з різними характеристиками екранів.

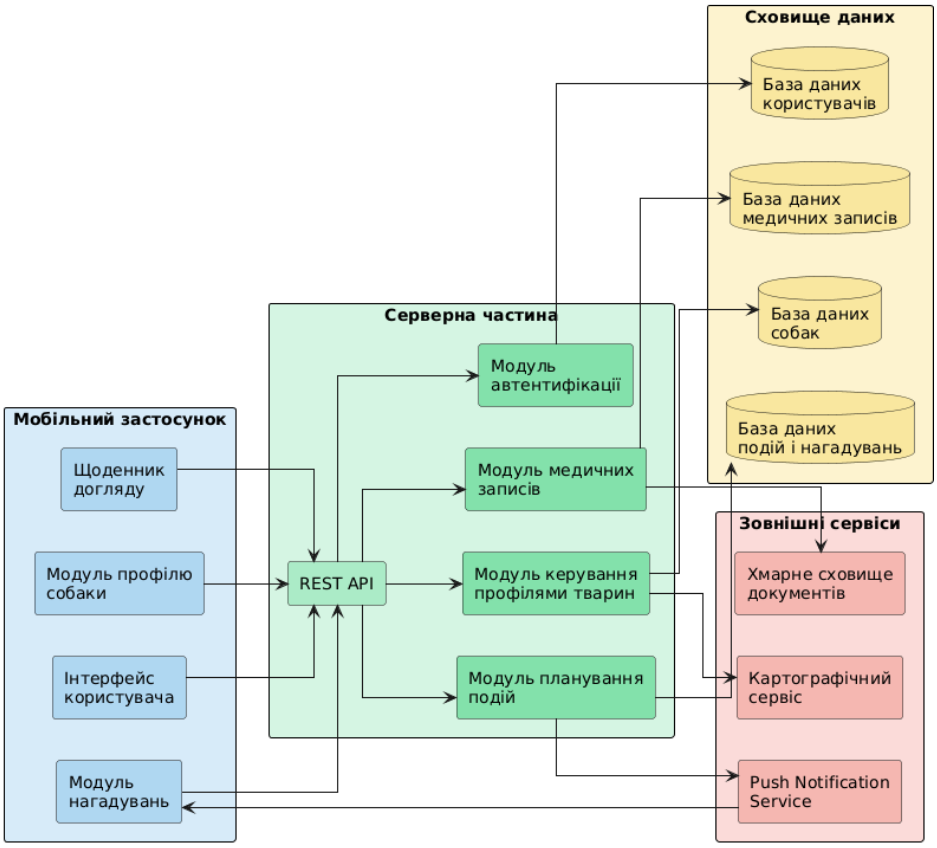

Рисунок 1 – Архітектура мобільного сервісу

Серверна частина мобільного сервісу реалізована з використанням платформи Node.js та фреймворку Express.js. Сервер забезпечує обробку запитів від клієнтського застосунку, керування бізнес-логікою, формування відповідей у форматі JSON та взаємодію з базою даних. Для обміну даними між клієнтською та серверною частинами використовується архітектурний стиль REST API, який забезпечує стандартизовану передачу інформації та спрощує інтеграцію з іншими програмними системами.

Висновок

У статті розглянуто особливості розроблення мобільного сервісу для автоматизації процесів догляду за собаками. Проведений аналіз предметної області підтвердив актуальність використання мобільних технологій для підтримки власників домашніх тварин під час планування та контролю процедур, пов’язаних із доглядом, моніторингом стану здоров’я та організацією профілактичних заходів.У результаті дослідження запропоновано архітектуру мобільного сервісу, побудовану за клієнт-серверним принципом, яка включає мобільний застосунок, серверну частину, базу даних та модулі інтеграції із зовнішніми сервісами. Розроблена архітектура забезпечує централізоване зберігання інформації про тварин, автоматичне формування нагадувань, ведення медичних записів та підтримку взаємодії користувача із системою в режимі реального часу.

Список використаних джерел

1. Zogaj, A.; Bretschneider, U.; Leimeister, J.M. Managing Crowd sourced Software Testing: A Case Study Based Insight on the Challenges of a Crowd sourcing Intermediary. Journal of Business Economics 2014, 84, 375–405. https://doi.org/10.1007/s11573-014-0721-9.
2. Birn-Hansen, A.; Grønli, T.-M.; Ghinea, G. A Survey and Taxonomy of Core Concepts and Research Challenges in Cross-Platform Mobile Development. ACM Computing Surveys 2018, 51, 1–34. https://doi.org/10.1145/3241739.

УДК 004.42:004.5:502.3

ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУНКУ МОНІТОРИНГУ РІВНЯ ЗАБРУДНЕННЯ АТМОСФЕРНОГО ПОВІТРЯ

Остафійчук Б.Б.[1)], **Новак С.В.**[2)], **Бойко С.В.**[3)]
Західноукраїнський національний університет
[1-3)]*бакалавр*

I. Постановка проблеми

Забруднення атмосферного повітря є однією з найважливіших екологічних проблем сучасності, яка безпосередньо впливає на здоров'я населення, стан екосистем та якість життя. Інтенсивний розвиток промисловості, збільшення кількості транспортних засобів і зростання урбанізації призводять до підвищення концентрації шкідливих речовин у повітрі, зокрема дрібнодисперсних частинок PM2.5 та PM10, оксидів азоту, діоксиду сірки, чадного газу та інших забруднювачів.

Сучасні системи екологічного моніторингу забезпечують накопичення значних обсягів даних про стан атмосферного повітря, проте доступ до такої інформації часто є ускладненим для пересічних користувачів. У зв'язку з цим особливої актуальності набуває розроблення мобільних застосунків, які дозволяють отримувати оперативну інформацію про якість повітря, відстежувати зміни екологічних показників та своєчасно реагувати на небезпечні рівні забруднення. Використання мобільних технологій створює можливість забезпечити користувачів актуальною екологічною інформацією незалежно від їх місцеперебування, а також реалізувати механізми автоматичного сповіщення про перевищення допустимих концентрацій забруднювальних речовин. Це сприяє підвищенню екологічної обізнаності населення та підтримує прийняття обґрунтованих рішень щодо планування повсякденної діяльності [1,2].

II. Мета роботи

Метою статті є проєктування та реалізація мобільного застосунку для моніторингу рівня забруднення атмосферного повітря, який забезпечує збір, обробку та візуалізацію екологічних даних у режимі реального часу, інформування користувачів про перевищення допустимих концентрацій забруднювальних речовин та підтримку прийняття рішень щодо мінімізації негативного впливу забрудненого повітря на здоров'я населення.

III. Особливості програмної реалізації

Програмна реалізація мобільного застосунку моніторингу рівня забруднення атмосферного повітря спрямована на забезпечення оперативного доступу користувачів до актуальної екологічної інформації, її обробки, аналізу та візуалізації у зручній формі. Під час розроблення програмного забезпечення особлива увага приділялася забезпеченню високої продуктивності, масштабованості, надійності та зручності використання мобільного застосунку в умовах постійного оновлення екологічних даних.Клієнтська частина системи реалізована за допомогою фреймворкуFlutter, що дозволяє створювати кросплатформний мобільний застосунок для операційних систем Android та iOS на основі єдиної програмної бази. Діаграму класів наведено на рисунку 1. Такий підхід суттєво зменшує витрати на розроблення та супровід програмного продукту, а також забезпечує однакову функціональність на різних мобільних платформах. Для побудови інтерфейсу користувача використано компонентний підхід, який дозволяє формувати окремі елементи відображення екологічної інформації у вигляді незалежних модулів.

Основою функціонування застосунку є механізм отримання даних про якість атмосферного повітря із зовнішніх інформаційних сервісів через REST API. Для взаємодії із серверними ресурсами використовуються HTTP-запити, а передача даних здійснюється у форматі JSON. Після отримання інформації виконується її автоматичне опрацювання, перевірка коректності та збереження в локальному сховищі мобільного пристрою для забезпечення швидкого доступу та можливості роботи в умовах нестабільного мережевого з'єднання.Важливою складовою програмної реалізації є модуль геолокації, який забезпечує автоматичне визначення поточного місцеположення користувача за допомогою GPS-технологій. Отримані координати використовуються для пошуку найближчих станцій моніторингу атмосферного повітря та відображення локальних показників якості повітря. Крім автоматичного визначення місцеположення передбачено можливість ручного вибору населеного пункту або регіону для перегляду відповідних екологічних показників.

Для візуалізації результатів моніторингу реалізовано модуль графічного представлення даних. Застосунок відображає поточні значення концентрацій основних забруднювальних речовин, індекс

якості повітря (AQI), історію змін екологічних показників та прогнозовані тенденції. Візуалізація реалізується за допомогою інтерактивних графіків, діаграм та кольорових індикаторів рівня небезпеки, що дозволяє користувачам швидко оцінювати поточний стан атмосферного повітря.Для забезпечення своєчасного інформування користувачів реалізовано підсистему сповіщень.

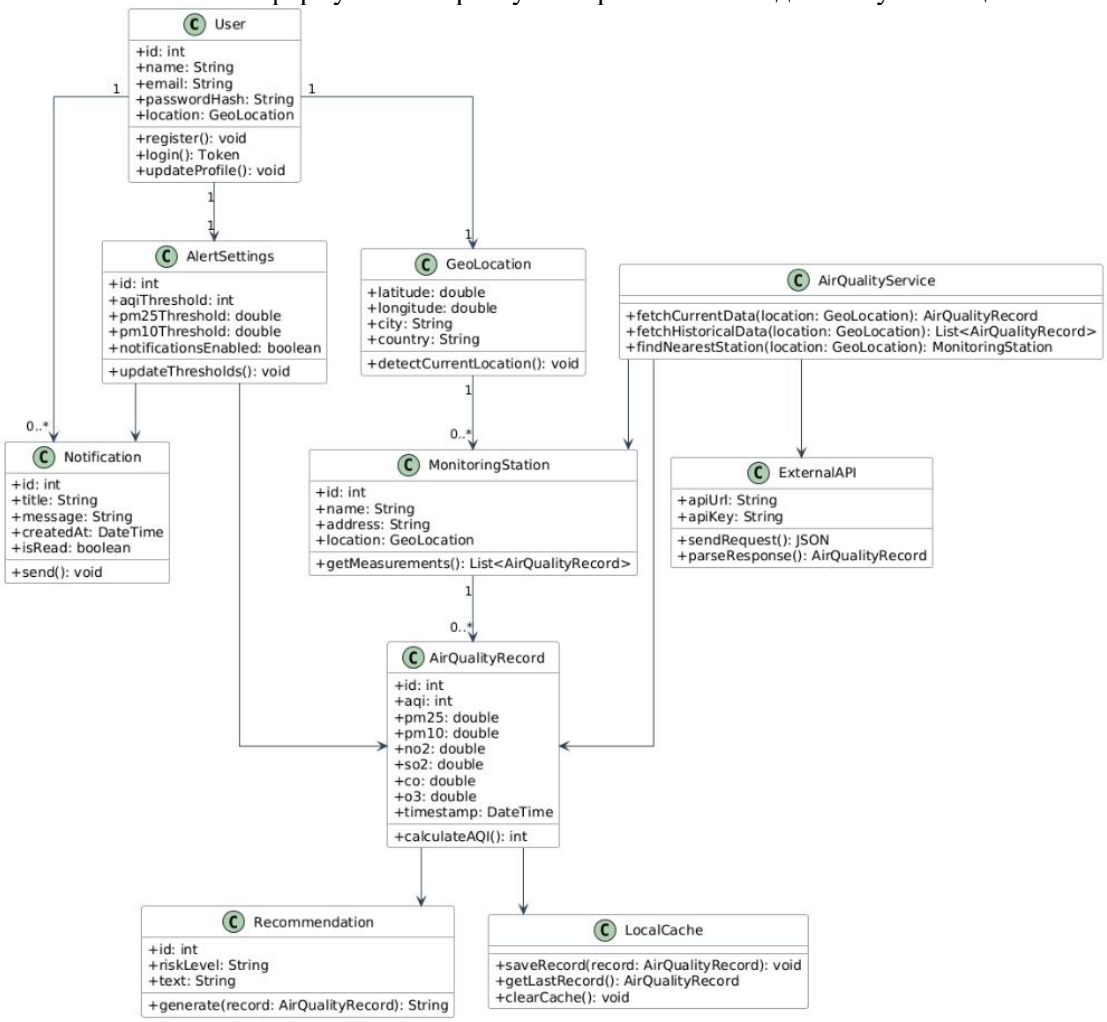

Рисунок 1 – Діаграма класів

З метою підвищення продуктивності застосунку реалізовано механізми кешування даних та асинхронного завантаження інформації. Це дозволяє мінімізувати навантаження на серверні ресурси та забезпечити швидке відображення екологічних показників навіть при значній кількості одночасних користувачів. Для захисту інформації використовується шифрування мережевого трафіку за протоколом HTTPS та механізми автентифікації користувачів.Розроблене програмне забезпечення характеризується модульною архітектурою, що забезпечує можливість подальшого розширення функціональності шляхом інтеграції нових джерел екологічних даних, використання методів прогнозування рівня забруднення атмосферного повітря та впровадження інтелектуальних механізмів підтримки прийняття рішень щодо екологічної безпеки населення.

Висновок

Реалізований застосунок забезпечує оперативне отримання даних про якість повітря, автоматичне визначення місцеположення користувача та інформування про небезпечні рівні забруднення. Отримані результати підтверджують доцільність використання мобільних технологій для підвищення доступності екологічної інформації та підтримки екологічно обґрунтованих рішень користувачів.

Список використаних джерел

1. Kumar, P.; Morawska, L.; Martani, C.; Biskos, G.; Neophytou, M.; DiSabatino, S.; Bell, M.; Norford, L.; Britter, R. The Rise of Low-Cost Sensing for Managing Air Pollutionin Cities. Environment International 2015, 75, 199–205. https://doi.org/10.1016/j.envint.2014.11.019.
2. Zheng, Y.; Liu, F.; Hsieh, H.-P. U-Air: When Urban Air Quality Inference Meets BigData. Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and DataMining 2013, 1436–1444. https://doi.org/10.1145/2487575.2488188.

УДК 004.42:004.65:004.738.5

РОЗРОБЛЕННЯ СЕРВІСУ ПЕРСОНАЛЬНОГО ЗБЕРІГАННЯ ДАНИХ ПРО ТУРИСТИЧНІ ПОЇЗДКИ

Гнитка М.І.[1)], **Каравацький В.І.**[2)], **Таратута О.Р.**[3)]
Західноукраїнський національний університет
[1-3)]*бакалавр*

І. Постановка проблеми

Сучасні туристичні поїздки супроводжуються значною кількістю цифрових даних, серед яких маршрути, квитки, бронювання житла, фінансові витрати, фотографії, нотатки та електронні документи. Зберігання такої інформації у різних застосунках і файлових сховищах ускладнює її пошук, систематизацію та подальше використання. У зв'язку з цим актуальним є розроблення спеціалізованого сервісу, який забезпечує персональне зберігання та впорядкування даних про туристичні поїздки. Персональний сервіс зберігання даних про подорожі дозволяє користувачеві формувати цифрову історію туристичної активності, об'єднувати різнорідні матеріали в межах окремих поїздок, здійснювати пошук за датами, країнами, містами чи категоріями даних, а також зберігати важливі документи у структурованому вигляді. Такий підхід підвищує зручність планування майбутніх подорожей і спрощує доступ до інформації про попередній туристичний досвід[1,2].

ІІ. Мета роботи

Метою статті є розроблення сервісу персонального зберігання даних про туристичні поїздки, який забезпечує централізоване накопичення, структурування, пошук і захищене зберігання інформації про маршрути, бронювання, витрати, документи, фотографії та інші матеріали, пов'язані з подорожами користувача.

ІІІ. Особливості проектування архітектури сервісу

Клієнтська частина сервісу відповідає за взаємодію користувача із системою. Через веб- або мобільний інтерфейс користувач може створювати записи про туристичні поїздки, додавати маршрути, дати подорожей, країни та міста, завантажувати квитки, бронювання, фотографії, нотатки й фінансові витрати. Особливу увагу під час проєктування інтерфейсу доцільно приділити простоті навігації, швидкому пошуку потрібної поїздки та зручному групуванню матеріалів за категоріями.Серверна частина реалізує основну бізнес-логіку сервісу. Вона забезпечує обробку запитів користувача, автентифікацію, керування обліковими записами, створення та редагування поїздок, завантаження файлів, пошук даних, формування статистики витрат і взаємодію з базою даних. Для обміну даними між клієнтською і серверною частинами доцільно використовувати REST API, що забезпечує стандартизовану передачу інформації у форматі JSON.

База даних призначена для зберігання структурованої інформації про користувачів, туристичні поїздки, маршрути, міста, країни, витрати, категорії документів і нотатки. Файлове сховище використовується для збереження неструктурованих даних, зокрема фотографій, PDF-документів, електронних квитків, підтверджень бронювання та інших супровідних матеріалів. Розділення структурованих даних і файлів підвищує продуктивність системи та спрощує масштабування сховища.

Важливою складовою архітектури є модуль безпеки, який забезпечує автентифікацію користувачів, авторизацію доступу до персональних даних, шифрування переданих файлів і захист конфіденційної інформації. Оскільки сервіс працює з персональними туристичними даними та документами, доцільним є використання токен-based автентифікації, протоколу HTTPS, резервного копіювання та механізмів контролю доступу.Додатково архітектура може передбачати модуль аналітики, який дозволяє користувачеві переглядати статистику подорожей за країнами, містами, періодами, витратами та типами активності. Такий модуль підвищує практичну цінність сервісу, оскільки перетворює накопичені дані не лише на архів подорожей, а й на інструмент аналізу особистого туристичного досвіду.

Запропонована архітектура забезпечує гнучкість, масштабованість і надійність сервісу персонального зберігання даних про туристичні поїздки. Вона створює основу для подальшого впровадження додаткових функцій, зокрема автоматичного розпізнавання даних із документів,

інтеграції з картографічними сервісами, побудови маршрутів та формування персональних рекомендацій щодо майбутніх подорожей.

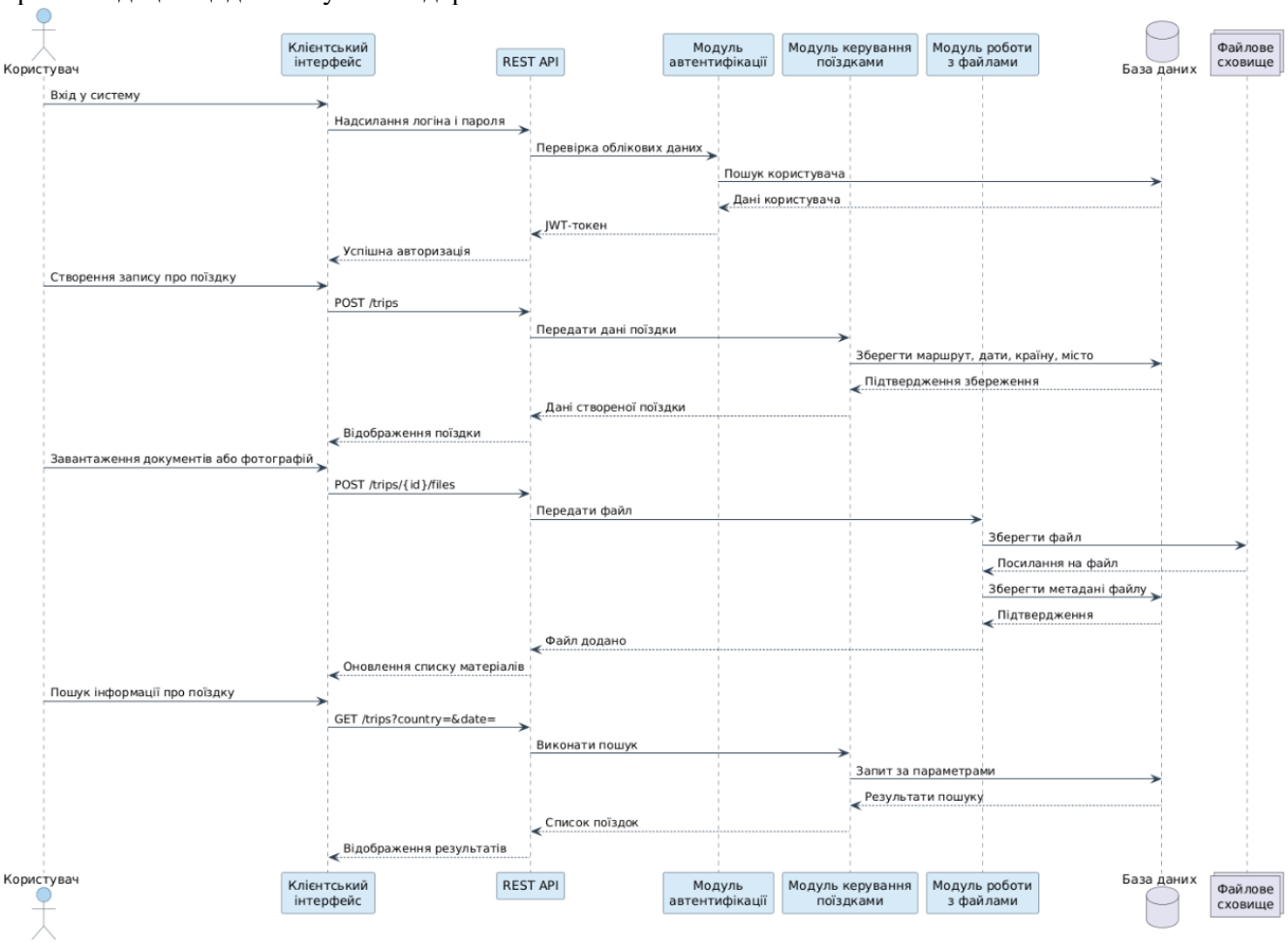

Рисунок 1 – Діаграма послідовності сервісу персонального зберігання даних про туристичні поїздки

Програмна реалізація сервісу виконана за клієнт-серверною архітектурою із використанням сучасних вебтехнологій. Клієнтська частина забезпечує взаємодію користувача із системою, створення та редагування інформації про туристичні поїздки, перегляд маршрутів, витрат і супровідних документів. Серверна частина реалізує бізнес-логіку, обробку запитів, автентифікацію користувачів та взаємодію з базою даних і файловим сховищем.Для зберігання структурованої інформації використовується реляційна база даних, яка містить відомості про користувачів, подорожі, маршрути, витрати та нотатки. Фотографії, квитки та інші документи зберігаються у файловому сховищі з прив’язкою до відповідних записів про туристичні поїздки. Такий підхід забезпечує надійне зберігання даних, швидкий пошук інформації та можливість масштабування сервісу.

Висновок

У статті розглянуто особливості проєктування та реалізації сервісу персонального зберігання даних про туристичні поїздки. Розроблено архітектуру системи, що забезпечує централізоване накопичення, структурування та зберігання інформації про подорожі, маршрути, витрати та супровідні документи. Запропонований сервіс дозволяє підвищити зручність управління туристичними даними, спростити їх пошук і забезпечити надійний доступ до інформації з різних пристроїв. Перспективою подальших досліджень є впровадження інтелектуальних механізмів аналізу подорожей, автоматичного формування рекомендацій та інтеграції з туристичними інформаційними системами.

Список використаних джерел

1. Sigala, M.; Gretzel, U. Advances in Social Media for Travel, Tourism and Hospitality: New Perspectives, Practice and Cases. Routledge, 2018. https://doi.org/10.4324/9781315565736.
2. Wang, D.; Xiang, Z.; Fesenmaier, D.R. Smartphone Use in Everyday Life and Travel. Journal of Travel Research 2016, 55(1), 52–63. https://doi.org/10.1177/0047287514535847.

УДК 004.65:004.8

ВЕБ-СЕРВІС ДЛЯ АНАЛІЗУ ТА ПЛАНУВАННЯ ВИРОБНИЦТВА ЕЛЕКТРИЧНОЇ ЕНЕРГІЇ СОНЯЧНИХ ЕЛЕКТРОСТАНЦІЙ

Дацків Т.І.[1)], **Борис В.В.**[2)], **ГлинськийТ.А.**[3)]
Західноукраїнський національний університет
[1-3)]*бакалавр*

І. Постановка проблеми

Зростання вартості традиційних енергоресурсів, необхідність зменшення викидів парникових газів та підвищення енергетичної незалежності держав сприяють активному впровадженню сонячних електростанцій. Водночас ефективне функціонування таких об'єктів значною мірою залежить від можливості прогнозування обсягів виробництва електроенергії, оскільки генерація безпосередньо пов'язана зі зміною погодних умов, рівнем сонячної радіації, температурою навколишнього середовища та іншими природними факторами[1].

Сучасні інформаційні технології та методи машинного навчання дозволяють створювати програмні засоби, здатні автоматично аналізувати великі обсяги історичних і метеорологічних даних та формувати прогнози виробництва електроенергії з достатньою точністю. Використання веб-технологій забезпечує зручний доступ до таких інструментів через мережу Інтернет, що робить можливим оперативний моніторинг стану сонячної електростанції, аналіз показників її роботи та підтримку процесів прийняття управлінських рішень [2,4].

Незважаючи на наявність великої кількості програмних продуктів для моніторингу та управління енергетичними об'єктами, актуальною залишається задача створення спеціалізованих веб-сервісів, які поєднують можливості збору даних, прогнозування генерації електроенергії та візуалізації результатів у єдиному інформаційному середовищі. Саме тому розроблення веб-сервісу для планування показників виробництва електроенергії сонячної електростанції є актуальним і практично значущим завданням.

ІІ. Мета роботи

Метою статті є розроблення веб-сервісу для планування показників виробництва електроенергії сонячної електростанції на основі аналізу історичних даних та метеорологічної інформації.

ІІІ. Проектування архітектури веб-сервісу

Проєктування архітектури є одним із ключових етапів розробки програмного забезпечення, оскільки визначає структуру системи, взаємодію її компонентів та принципи обробки інформації. Для веб-сервісу планування показників виробництва електроенергії сонячної електростанції архітектура повинна забезпечувати ефективний збір даних, їх зберігання, аналіз, прогнозування та відображення результатів користувачам через веб-інтерфейс.

З урахуванням сформованих вимог до системи доцільно використати багаторівневу клієнт-серверну архітектуру, яка забезпечує розподіл функціональності між окремими компонентами програмного забезпечення. Такий підхід дозволяє підвищити масштабованість, спростити супровід системи та забезпечити можливість подальшого розширення функціоналу.

Архітектура веб-сервісу складається з чотирьох основних рівнів: рівень представлення (Frontend); серверний рівень прикладної логіки (Backend); рівень прогнозування та аналітики; рівень зберігання даних.

Користувач взаємодіє із системою через веб-браузер, використовуючи графічний інтерфейс веб-сервісу. Запити від клієнтської частини передаються до серверного застосунку через REST API. Серверна частина виконує обробку бізнес-логіки, взаємодіє з базою даних, отримує інформацію із зовнішніх погодних сервісів та передає необхідні дані до модуля прогнозування.

Модуль прогнозування є окремим компонентом системи та реалізує алгоритми машинного навчання для оцінювання майбутніх показників генерації електроенергії. Для формування прогнозу використовуються історичні дані виробництва електроенергії та актуальні метеорологічні показники. Результати прогнозування зберігаються у базі даних і відображаються користувачеві через веб-інтерфейс.

Для отримання погодних параметрів система інтегрується із зовнішніми метеорологічними сервісами через API. Отримані дані проходять попередню обробку та використовуються як в

аналітичному модулі, так і під час побудови прогнозів. Зберігання інформації реалізується за допомогою реляційної бази даних, яка містить відомості про користувачів, сонячні електростанції, історію генерації електроенергії, метеорологічні показники та результати прогнозування. Загальну архітектуру веб-сервісу наведено на рисунку 1.

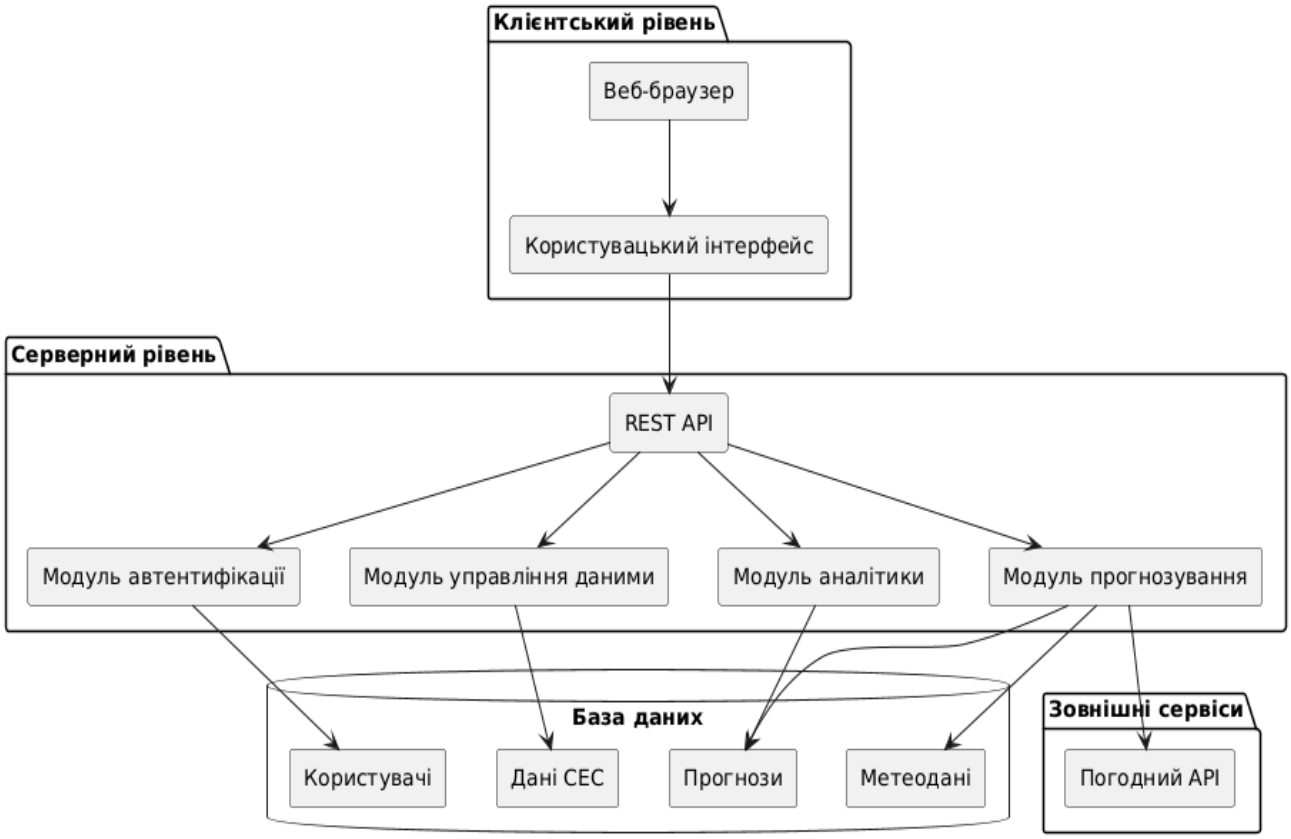

Рисунок 1 – Архітектура веб-сервісу планування показників виробництва електроенергії

Запропонована архітектура забезпечує модульність програмної системи, спрощує підтримку окремих компонентів та дозволяє інтегрувати нові алгоритми прогнозування без зміни основної структури веб-сервісу. Крім того, використання REST API та багаторівневої архітектури створює передумови для масштабування системи та її подальшого розвитку.

Висновок

У процесі виконання роботи проведено аналіз особливостей функціонування сонячних електростанцій, досліджено основні фактори, що впливають на виробництво електроенергії, а також розглянуто сучасні методи прогнозування генерації на основі метеорологічних даних.

Виконано огляд існуючих програмних рішень для моніторингу та планування роботи сонячних електростанцій, що дало змогу сформувати функціональні та нефункціональні вимоги до розроблюваної системи.

У межах роботи спроєктовано архітектуру веб-сервісу, структуру бази даних та інтерфейс користувача. Розроблена архітектура забезпечує модульність, масштабованість і можливість подальшого розвитку програмного забезпечення.

Список використаних джерел

1. DeWitt, David J., andJimGray. “Parallel Database Systems: The Future of High Performance Database Systems.” Communications of the ACM 35, no. 6 (1992): pp.85–98. https://doi.org/10.1145/129888.129894.
2. Graefe, Goetz. “Query Evaluation Techniques for Large Databases.” ACM Computing Surveys 25, no. 2 (1993): pp.73–169. https://doi.org/10.1145/152610.152611.
3. Крепич С.Я., Співак І.Я. Якість програмного забезпечення та тестування: базовий курс. Тернопіль: ФОП Паляниця В.А, 2020. – 478с.
4. Chaudhuri, Surajit, and Umeshwar Dayal. “An Overview of Data Warehousing and OLAP Technology.” ACM SIGMOD Record 26, no. 1 (1997): pp.65–74. https://doi.org/10.1145/248603.248616.
5. Співак І.Я., Крепич С.Я. Якість програмного забезпечення в контексті аналізу вимог: Підручник. Тернопіль: ВПЦ «Університетська думка», 2026. – 318с.

UDC 004.41

DEVELOPMENT OF A WEB-BASED STUDENT MANAGEMENT SYSTEM

Kamara J.[1)], Maslyiak Yu.[2)]
West Ukrainian National University
1)student, 2) PhD, Associate Professor

I. Problem statement

The rapid growth of information technology has transformed the management of educational data. Traditional methods of maintaining student records through paper documents and spreadsheets often lead to inefficiency, data duplication, and difficulties in retrieval information. This paper presents the design and implementation of a Web-Based Student Management System developed using PHP, MySQL, HTML, CSS, Bootstrap, and XAMPP. The system provides administrators with functionalities such as secure authentication, student registration, record management, search operations, photo uploads, and dashboard reporting. The proposed solution improves the efficiency of student data management and reduces administrative workload. Experimental testing confirmed that all major system functions operate successfully, providing a reliable and user-friendly platform for educational institutions.

II. Purpose of the work

Educational institutions manage large volumes of student information daily. Traditional methods of handling student records are often time-consuming and susceptible to human errors. As the number of students increases, the need for an automated and centralized information management system becomes more important.

The purpose of this research is to develop a web-based student management system that enables administrators to efficiently manage student records. The system allows authorized users to add, update, search, and delete student information through an intuitive web interface. The project was developed as a practical implementation of software engineering principles and modern web development technologies.

III. System Design and Implementation

The system follows a three-tier architecture consisting of the presentation layer, application layer, and database layer. The presentation layer was developed using HTML, CSS, and Bootstrap to provide a responsive and user-friendly interface. PHP was used to implement application logic, while MySQL served as the database management system.

The main functionalities of the system include:

- Administrator registration and login.
- Student registration.
- Student information management.
- Search functionality.
- Photo upload support.
- Dashboard statistics.
- Record editing and deletion.

The dashboard provides quick access to system statistics including total students, courses, and administrator accounts (view fig. 1).

The student registration module allows administrators to enter student details and upload profile images (view fig. 2).

Administrators can view, search, edit, and delete student records through a centralized interface (view fig.3).

The MySQL database stores administrator accounts and student records, ensuring efficient data retrieval and management (view fig. 4).

IV. Results and Discussion

System testing demonstrated that all implemented functionalities operated successfully. Administrator authentication, student registration, record editing, deletion, photo uploads, and search operations were tested and achieved expected outcomes. The system provides a reliable platform for managing educational records and significantly reduces the workload associated with manual data processing.

The use of PHP and MySQL allowed efficient communication between the application and database layers, while Bootstrap improved usability through a responsive user interface.

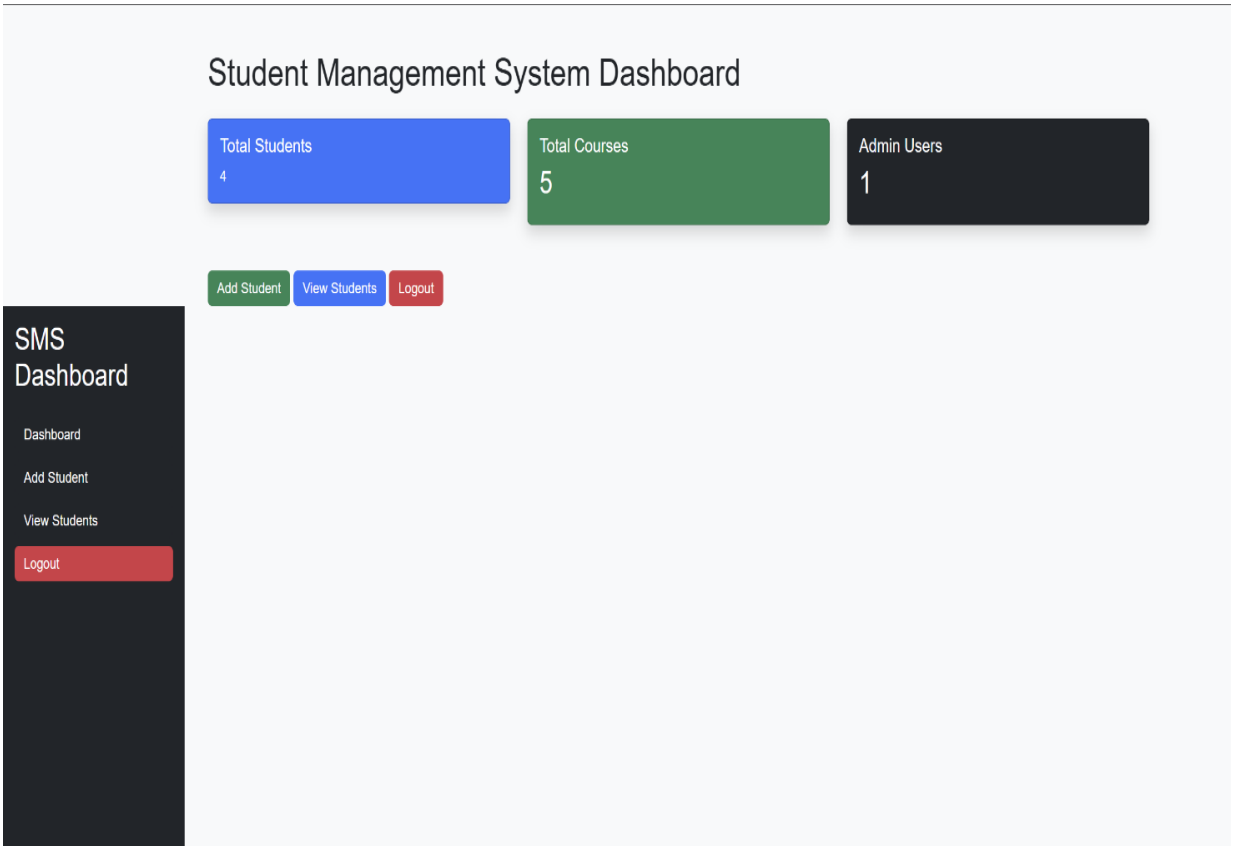

Figure 1 – Dashboard interface

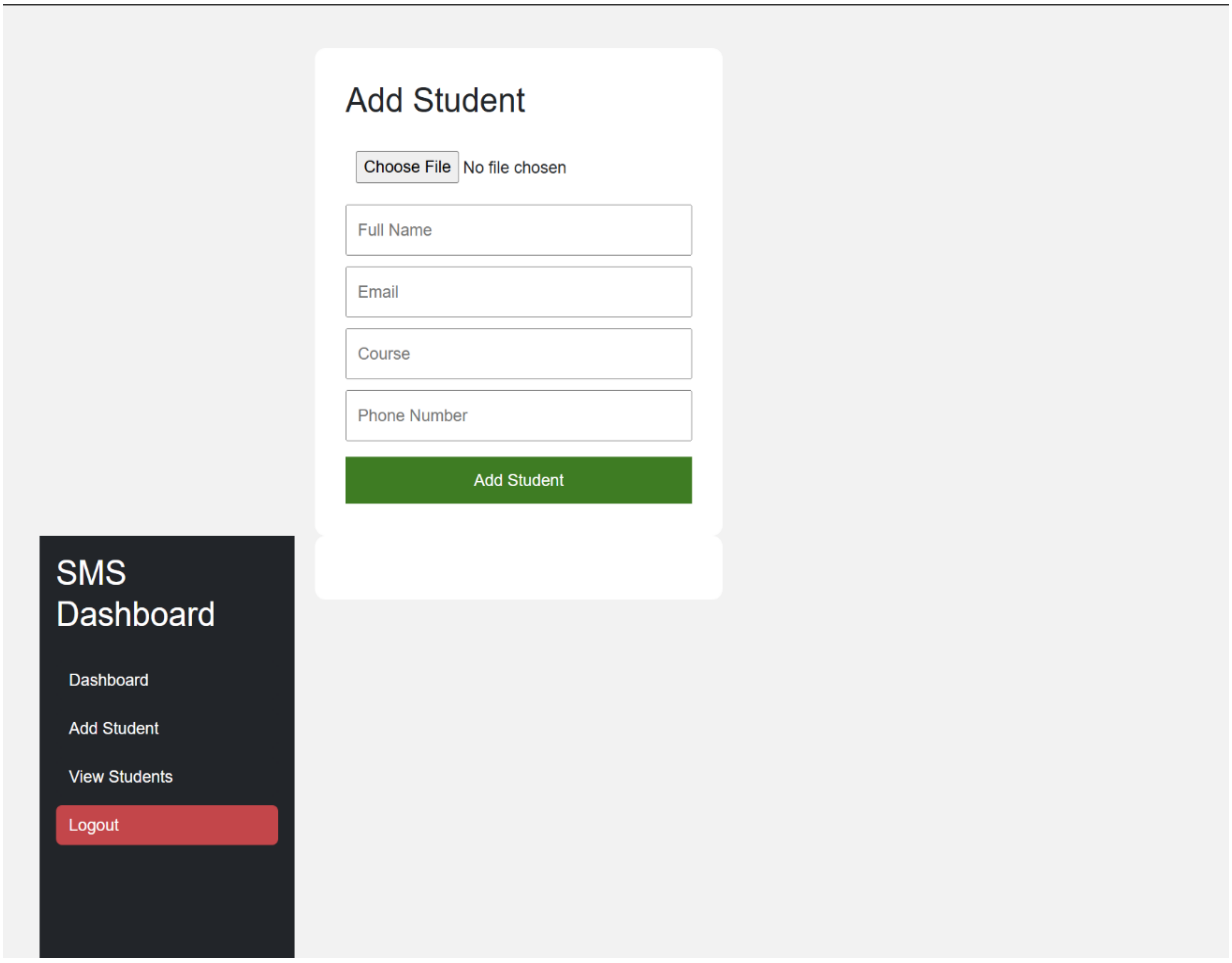

Figure 2 – Student registration form

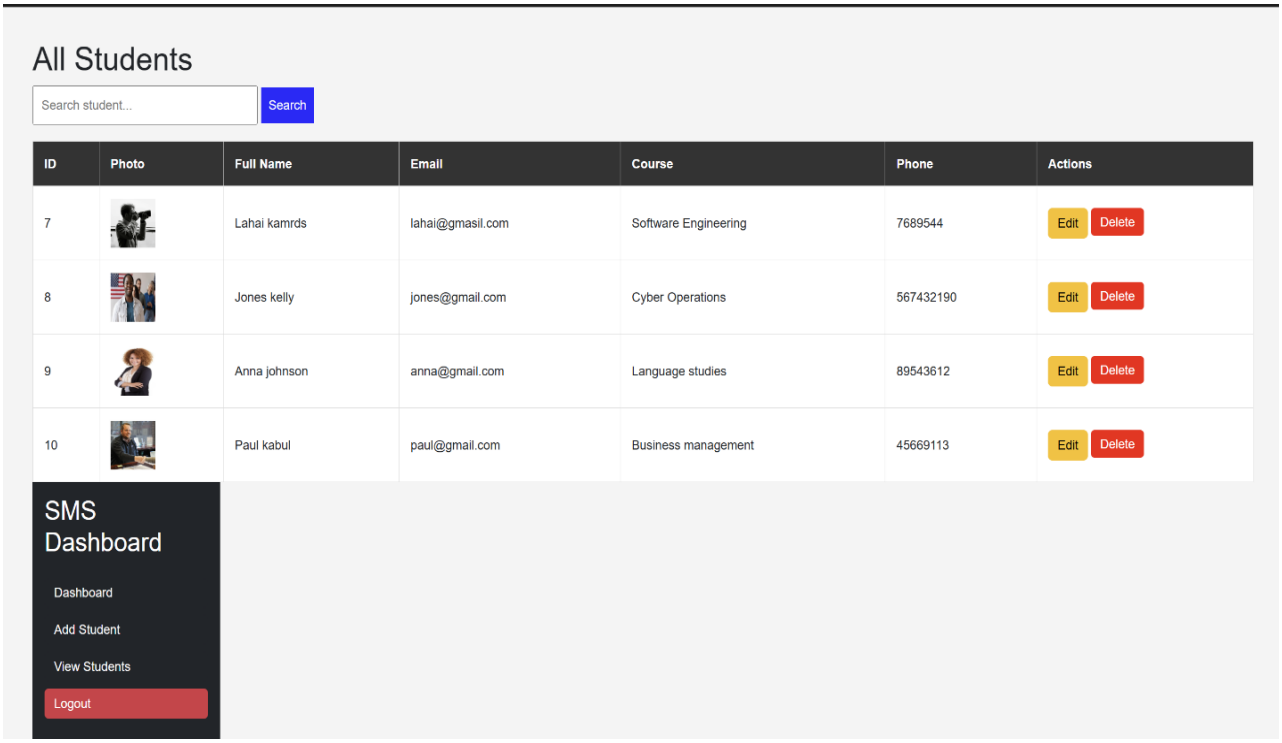

Figure 3 – Student records management

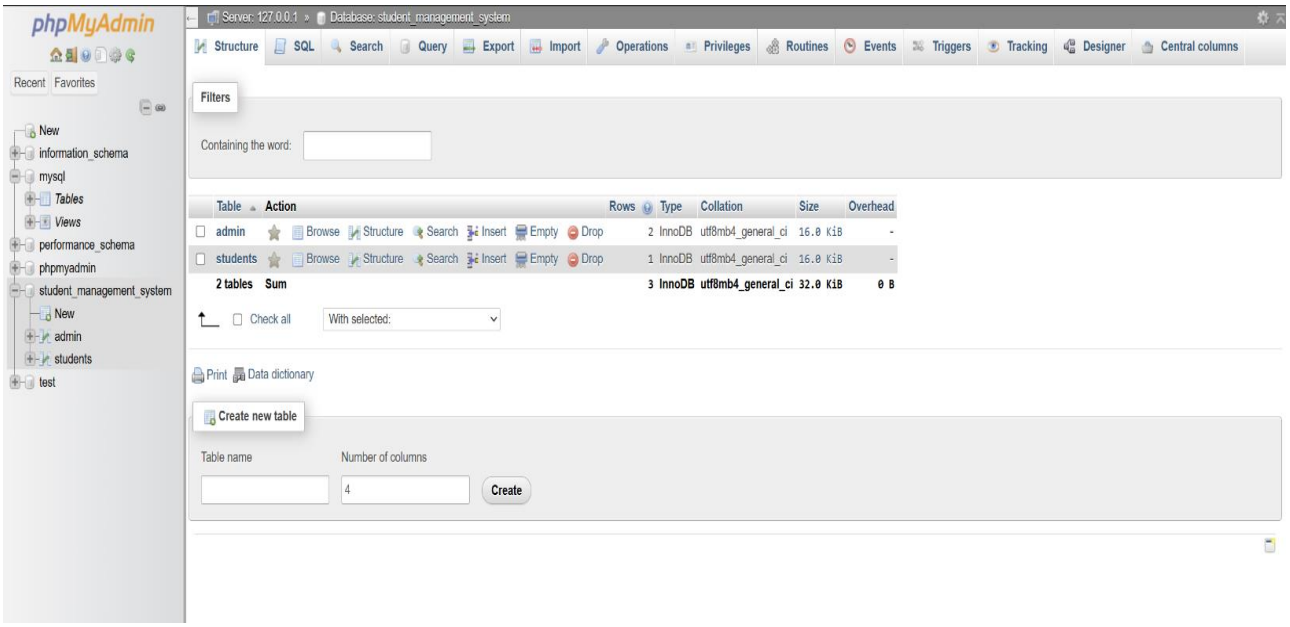

Figure 4 – Database Structure

Conclusion

The system follows a three-tier architecture consisting of the presentation layer, application layer, and database layer. The presentation layer was developed using HTML, CSS, and Bootstrap to provide a responsive and user-friendly interface. PHP was used to implement application logic, while MySQL served as the database management system.

References

1.PHP Documentation. https://www.php.net
2.MySQL Documentation. https://www.mysql.com
3.Bootstrap Documentation. https://getbootstrap.com
4.Mozilla Developer Network (MDN). https://developer.mozilla.org
5.Pressman, R. Software Engineering: A Practitioner's Approach, 2009.
6.Sommerville, I. Software Engineering, 2010.

УДК 681.3

ВЕБОРІЄНТОВАНА ПРОГРАМНА СИСТЕМА ОНЛАЙН-НАВЧАННЯ З ПІДТРИМКОЮ ПЕРСОНАЛІЗАЦІЇ НАВЧАЛЬНОГО КОНТЕНТУ

Вермій С.В.
Західноукраїнський національний університет, бакалавр

І. Постановка проблеми

Стрімкий розвиток інформаційних технологій та поширення дистанційної освіти зумовлюють необхідність створення сучасних програмних систем онлайн-навчання. Більшість існуючих освітніх платформ надають однаковий навчальний контент для всіх користувачів, не враховуючи індивідуальні особливості, рівень підготовки та інтереси студентів. Такий підхід може знижувати ефективність засвоєння матеріалу та мотивацію до навчання. Тому актуальним завданням є розроблення веб-орієнтованоїпрограмної системи, здатної адаптувати навчальний контент відповідно до характеристик конкретного користувача.

Важливим аспектом сучасних систем електронного навчання є забезпечення індивідуального підходу до кожного здобувача освіти. Використання персоналізованих рекомендацій дозволяє враховувати попередні результати навчання, темп засвоєння матеріалу та освітні потреби користувача. У зв'язку з цим виникає потреба у створенні програмних рішень, які поєднують функціональність систем управління навчанням із механізмами інтелектуального аналізу даних.

Сучасні тенденції цифровізації освіти передбачають використання адаптивних технологій навчання, які дозволяють формувати індивідуальні освітні траєкторії. Особливої актуальності це набуває в умовах змішаного та дистанційного навчання, де викладач не завжди має можливість оперативно враховувати особливості кожного студента. У таких умовах програмні системи, що автоматично аналізують поведінку користувачів та рекомендують відповідний контент, стають важливим інструментом підвищення якості освітнього процесу.

ІІ. Мета роботи

Метою роботи є розроблення веб-орієнтованої програмної системи онлайн-навчання з підтримкою персоналізації навчального контенту для підвищення ефективності освітнього процесу та покращення користувацького досвіду.

ІІІ. Архітектура та функціональні можливості системи

Запропонована система реалізована за клієнт-серверною архітектурою та складається з фронтенд- і бекенд-компонентів. Для реалізації клієнтської частини використано фреймворк Next.js, який забезпечує високу продуктивність та зручну взаємодію користувача із системою. Серверна частина розроблена на базі NestJS із використанням PostgreSQL для зберігання даних.

Рисунок 1 - Клієнт-серверна архітектура та функціональна схема системи онлайн-навчання

Взаємодія між клієнтською та серверною частинами здійснюється через RESTAPI, що забезпечує стандартизований обмін даними та спрощує інтеграцію із зовнішніми сервісами. Для оптимізації роботи застосовано механізми кешування та асинхронної обробки запитів, що дозволяє зменшити навантаження на сервер і підвищити швидкість відгуку системи. Такий підхід сприяє забезпеченню стабільної роботи навіть за умов значної кількості одночасно підключених користувачів.

Система підтримує реєстрацію та автентифікацію користувачів, керування курсами, формування навчальних програм, проходження навчальних модулів та контроль успішності. Особливістю розробленого рішення є механізм персоналізації, який аналізує активність користувача та формує рекомендації щодо навчального контенту.

Під час проєктування системи особливу увагу приділено масштабованості та безпеці. Використання сучасних вебтехнологій забезпечує можливість подальшого розширення функціоналу без суттєвих змін архітектури. Для захисту даних користувачів застосовано механізми авторизації на основі JWT-токенів, а також розмежування прав доступу залежно від ролі користувача в системі.

IV. Результати дослідження

У процесі реалізації проєкту було проведено аналіз предметної області, визначено основні вимоги до системи та сформовано модель даних для зберігання інформації про користувачів, курси та результати навчання. На основі проведеного аналізу розроблено структуру програмної системи, яка забезпечує ефективну взаємодію між усіма компонентами та підтримує можливість подальшого розширення функціоналу.

У результаті виконання роботи створено програмну систему, яка забезпечує управління освітнім контентом та підтримує персоналізоване формування рекомендацій. Реалізовано функціонал створення та проходження курсів, управління користувачами, відстеження навчального прогресу та автоматичного підбору матеріалів на основі аналізу взаємодії користувача із платформою.

Проведене тестування підтвердило коректність функціонування системи та можливість її використання в освітніх установах і комерційних онлайн-школах. Проведено серію функціональних та інтеграційних тестів, які підтвердили стабільність роботи програмної системи в умовах одночасної взаємодії декількох користувачів. Аналіз результатів тестування показав, що застосування персоналізованого підходу дозволяє скоротити час пошуку необхідних навчальних матеріалів та підвищити рівень активності користувачів під час проходження курсів. Використання механізмів персоналізації сприяє покращенню якості навчального процесу та підвищенню ефективності засвоєння навчального матеріалу.

Висновок

У роботі розроблено веборієнтовану програмну систему онлайн-навчання з підтримкою персоналізації навчального контенту, яка забезпечує адаптацію навчальних матеріалів до індивідуальних потреб користувачів, автоматизацію основних освітніх процесів та зручне управління навчальними курсами. У процесі розроблення спроєктовано клієнт-серверну архітектуру системи та реалізовано функціональні можливості реєстрації й автентифікації користувачів, створення курсів, проходження навчальних модулів, контролю успішності та відстеження навчального прогресу.

Особливу увагу приділено механізму персоналізації навчального контенту, який на основі аналізу результатів навчання, рівня підготовки та активності користувачів формує рекомендації щодо найбільш релевантних матеріалів. Проведене тестування підтвердило працездатність, стабільність і ефективність системи, а використання адаптивного підходу сприяє підвищенню залученості користувачів, покращенню засвоєння матеріалу та оптимізації процесу навчання.

Практична цінність розробки полягає у можливості використання системи в дистанційному та корпоративному навчанні, а також у внутрішніх освітніх платформах організацій. Завдяки модульній структурі система може бути адаптована до різних предметних областей, масштабована відповідно до потреб користувачів та доповнена сучасними методами штучного інтелекту для подальшого вдосконалення механізмів персоналізації й аналітики навчального процесу.

Список використаних джерел

1. Horton W. E-Learning by Design. – Hoboken: John Wiley & Sons, 2023. – 640 p.
2. Крепич С.Я., Співак І.Я. Якість програмного забезпечення та тестування: базовий курс. Тернопіль: ФОП Паляниця В.А, 2020. – 478с.
3. Співак І.Я., Крепич С.Я. Якість програмного забезпечення в контексті аналізу вимог: Підручник. Тернопіль: ВПЦ «Університетська думка», 2026. – 318с.
4. Pressman R., Maxim B. Software Engineering: A Practitioner's Approach. – 9th ed. – New York: McGraw-Hill Education, 2023. – 976 p.

Наукове видання

Комп’ютерні інформаційні технології

Матеріали
весняної школи-семінару молодих вчених і студентів СІТ’2026

Відповідальний за випуск:
Пукас А.В., д.т.н., професор,
завідувач кафедри комп’ютерних наук
Західноукраїнського національного університету

Підписано до друку 29.04.2026р.
Формат 60х84/16. Папір офсетний.
Друк офсетний. Зам. № 9-365
Тираж 25 прим.

Віддруковано ФО-П Шпак В. Б.
Свідоцтво про державну реєстрацію В02 № 924434 від 11.12.2006 р.
Свідоцтво платника податку: Серія Е № 897220
м. Тернопіль, вул. Просвіти, 6.
тел. 8 097 299 38 99
E-mail: tooums@ukr.net