CIT 2025

WORKSHOP

ШКОЛА-СЕМІНАР

молодих вчених і студентів

КОМП'ЮТЕРНІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

2 травня
2025 року

м. Тернопіль
вул. О. Теліги, 8

fcit.wunu.edu.ua

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

2 травня 2025 року

CIT’2025

Тернопіль
ЗУНУ
2025

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 СІТ'2025. – Тернопіль: ЗУНУ, 2025.

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 004.023

МАТЕМАТИЧНА МОДЕЛЬ РОЗПОДІЛУ РЕСУРСІВ У РОЗРОБЦІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З УРАХУВАННЯМ ПРОФЕСІЙНОГО ВИГОРАННЯ

Осадчук С.Б.[1)], Шпінталь М.Я.[2)]
Західноукраїнський національний університет
[1)] магістрант; [2)] к.т.н., доцент

I. Постановка проблеми

Ефективне управління командними ресурсами в процесі розробки програмного забезпечення є однією з ключових умов успішного завершення проєктів. В умовах зростаючої складності IT-продуктів та швидких змін вимог, все більш актуальним стає питання оптимального розподілу завдань між учасниками команди.

Особливу увагу необхідно приділяти факторам професійного вигорання, динамічного навантаження, дедлайнів, а також відповідності компетенцій виконавців до вимог конкретних задач.

Аналіз існуючих підходів показав, що більшість з них не враховують інтегровано як емоційно-психологічні аспекти роботи в команді, так і стохастичну природу змін у проектах (невизначеність складності, варіативність доступності ресурсів).

Існує потреба в розробці математичної моделі, яка б забезпечувала гнучкий і масштабований підхід до розподілу ресурсів з урахуванням багатьох факторів, у тому числі таких, як вигорання, ризики затримок та відповідність компетенцій.

II. Мета роботи

Метою дослідження є розробка математичної моделі, що дозволяє здійснювати розподіл завдань між членами команди розробників з урахуванням комплексу факторів, зокрема: відповідності компетенцій працівників до вимог задач, обмеженої доступності в робочому часі, пріоритетності та дедлайнів, індексу вигорання, а також рівня невизначеності в складності або тривалості виконання завдань.

III. Запропоноване рішення

Основу рішення становить модель цілочисельного лінійного програмування. В рамках цієї моделі кожне завдання може бути призначено лише одному виконавцю, який володіє необхідними компетенціями та має достатній резерв часу. Особливістю моделі є включення коефіцієнта вигорання до цільової функції, що дозволяє знижувати навантаження на працівників з підвищеним рівнем втоми. Індекс вигорання визначається на основі сукупності показників, що характеризують навантаження та активність працівників за певний період часу. Параметр невизначеності враховується шляхом застосування коефіцієнта варіації до оцінок трудомісткості задач, що формує адаптивний підхід до розподілу ресурсів в умовах стохастичних змін проекту.

Для задач, які не можуть бути призначені в рамках наявних ресурсів, передбачено введення штрафного коефіцієнта, що дозволяє системі зберігати стабільність у випадку дефіциту виконавців. Такий підхід також відкриває можливість для подальшого масштабування моделі в умовах змінного складу команди чи появи нових задач у процесі роботи.

Математична формалізація моделі може бути представлена наступним чином:

$$\min Z = \sum (T_{ij} \cdot x_{ij}) + \sum (B_i \cdot w_i) - \sum (C_{ij} \cdot x_{ij}) \quad (1)$$

де x_{ij} - бінарна змінна призначення завдання j виконавцю i, T_{ij} - оцінка часу виконання завдання, B_i - індекс вигорання, w_i - ваговий коефіцієнт, що регулює вплив фактору вигорання на оптимізацію, C_{ij} - показник відповідності компетенцій.

За наступних обмежень: $\sum j\ x_{ij} \cdot T_{ij} \le A_i$ для всіх i (обмеження доступного часу виконавця); $x_{ij} = 0$, якщо компетенція i не відповідає вимогам j (обмеження кваліфікації).

IV. Наукова новизна

Запропонована модель поєднує класичний апарат цілочисельного лінійного програмування з факторами, що рідко враховуються у подібних постановках задач. На відміну від традиційних

методів [1-3], цей підхід інтегрує психологічні аспекти роботи команди безпосередньо в цільову функцію.

Ключова новизна полягає у використанні індексу вигорання як вагового коефіцієнта цільової функції та уніфікованому підході до врахування невизначеності через масштабування трудомісткості задач. Це дозволяє точніше адаптувати розподіл ресурсів у динамічних умовах, що ілюструється на рисунку 1, де порівнюються ефективність запропонованого підходу та існуючих методів.

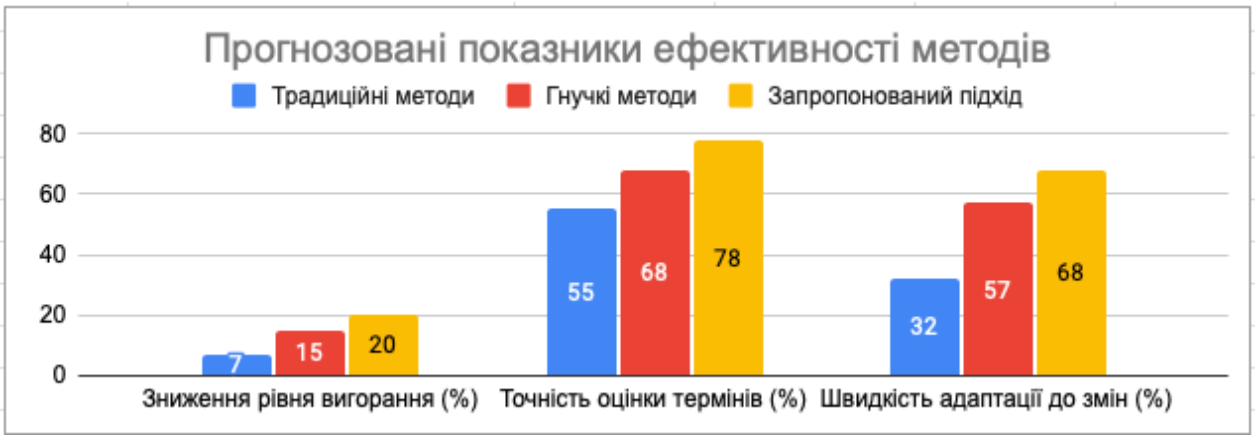


Рисунок 1 - Порівняння ефективності запропонованого підходу з традиційними та гнучкими методами

V. Очікувані результати

Практичне значення запропонованого рішення полягає в можливості ефективнішого розподілу завдань у реальних умовах командної роботи над IT-проєктами, що сприятиме покращенню продуктивності команди, зменшенню ризику вигорання співробітників та підвищенню якості виконуваних задач.

Очікується, що впровадження запропонованої моделі дозволить досягти більш рівномірного розподілу навантаження між працівниками, покращити відповідність задач виконавцям за їхніми компетенціями, а також зменшити ризики перевантаження та вигорання. Це, своєю чергою, має позитивно вплинути на стабільність команди, якість виконаних задач та своєчасність завершення проектів.

За результатами початкового моделювання очікується: зниження рівня професійного вигорання на 20-25%; підвищення точності оцінки термінів виконання задач на 15-20%; скорочення часу на перерозподіл задач при зміні умов проекту на 30-40%.

Верифікація ефективності запропонованої моделі планується у два етапи: спочатку на синтетичних даних для калібрування параметрів, а потім на реальних проєктних даних для оцінки практичної цінності підходу. Ключовими метриками оцінки будуть показники рівномірності розподілу навантаження та дотримання дедлайнів проєктів.

VI. Перспективи розвитку

У подальших дослідженнях передбачається доповнення моделі підтримкою багатопроєктного середовища, урахуванням фазової динаміки вигорання (наприклад, на основі історії відпусток або простоїв), а також інтеграція з зовнішніми системами управління (Trello, Jira) через їхні API, що дозволить застосовувати модель у реальних умовах роботи IT-команд.

Висновок

Таким чином, запропонована модель є логічним кроком до побудови практичного інструменту для оптимізації розподілу завдань у команді розробників. Формалізація факторів вигорання та невизначеності суттєво підвищує адаптивність управління проектами до динамічних змін. На наступному етапі передбачено програмну реалізацію моделі, її апробацію у реальних умовах IT-проєктів та аналіз результатів для подальшого вдосконалення.

Список використаних джерел

1. Yang S., Fu C. Critical chain scheduling in resource-constrained project management. // IEEE Transactions on Engineering Management. – 2022,Vol. 69, No. 3, pp. 858-871
2. Semchenko O.A. Формалізація задачі оптимального призначення виконавців у команді проекту на основі математичних моделей // Наукові записки УКУ. – 2021, Вип. 7,с. 92–98.
3. Процик О.В. Застосування методів лінійного програмування для оптимізації розподілу робочого часу в IT-проєктах // Вісник Хмельницького національного університету. – 2022, №2 (304), с. 124–129.

УДК 004.8

МАТЕМАТИЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ДЛЯ ВИЯВЛЕННЯ ПУХЛИН МОЗКУ ЗА ДОПОМОГОЮ ГЛИБОКОГО НАВЧАННЯ

Шпінталь М.Я.[1)], Мартинюк Ю.П.[2)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] магістрант

I. Постановка проблеми

Раннє виявлення пухлин мозку є важливою умовою успішного лікування та збільшення виживаності пацієнтів. Магнітно-резонансна томографія (МРТ) є основним методом візуалізації, що дозволяє виявити навіть невеликі утворення. Проте ручний аналіз знімків вимагає значного часу, спеціалізованих знань та часто супроводжується суб'єктивними помилками.

Тому актуальним стає створення автоматизованої системи, здатної на основі методів глибокого навчання і комп’ютерного зору визначати наявність пухлин та їх локалізацію з високою точністю.

II. Мета роботи

– Розробити концепцію та принципи функціонування автоматизованої системи для виявлення пухлин мозку на основі аналізу МРТ-зображень.
– Обґрунтувати вибір нейронної мережі та методів обробки зображень для досягнення найкращої точності.
– Визначити основні етапи створення програмного рішення.

III. Використовувані методи

– Глибоке навчання: побудова згорткових нейронних мереж (CNN) для автоматичного виділення ознак з медичних знімків.
– Комп’ютерний зір: обробка МРТ-зображень для підготовки вхідних даних та аналізу результатів.
– Аугментація даних: розширення навчального набору шляхом обертання, масштабування та інших змін зображень для покращення узагальнення.
– Оптимізаційні алгоритми: використання алгоритму Adam для стабільного й ефективного навчання мережі.
– Оцінка якості моделі: застосування метрик Accuracy, Sensitivity, Specificity та IoU для контролю якості класифікації та локалізації.

IV. Наукова новизна

– Запропонована архітектура моделі дозволяє одночасно виконувати два завдання: класифікацію типу пухлини та локалізацію її меж.
– Комбіноване використання категоріальної крос-ентропії та середньоквадратичної помилки в одній моделі є оптимальним рішенням для подібного типу задач.
– Удосконалено методику обробки медичних знімків для кращої роботи мережі на реальних даних.
– Сформовано підхід до розробки користувацького інтерфейсу з урахуванням потреб медичних працівників.

V. Реалізація системи

1. Підготовка та обробка даних

Формується датасет із різноманітних МРТ-зображень з пухлинами типу гліома, менінгіома, аденома гіпофіза та контрольними зразками без патологій.

Зображення масштабуються до розміру 216x216 пікселів, нормалізуються за яскравістю та контрастністю, що покращує стабільність навчання.

Додатково застосовується аугментація для підвищення загальної стійкості моделі до різних варіацій вхідних даних.

2. Архітектура моделі

Модель складається з послідовних згорткових блоків із шарами MaxPooling, які автоматично виділяють важливі ознаки з зображення.

На базі отриманих ознак модель розділяється на дві гілки:
– Класифікаційна — визначення типу пухлини серед заданих класів.

– Локалізаційна — прогнозування координат обмежувальної рамки пухлини (центр, ширина, висота).

3. Тренування моделі

Для навчання використовується комбінована функція втрат:

– крос-ентропія для класифікації,
– середньоквадратична помилка для локалізації.

Навчання моделі проводиться з використанням оптимізатора Adam, з моніторингом показників на тренувальній і валідаційній вибірках. Проводиться контроль перенавчання за допомогою методу EarlyStopping. На рисунку 1 зображенокриву тренування: точність класифікації та локалізація.

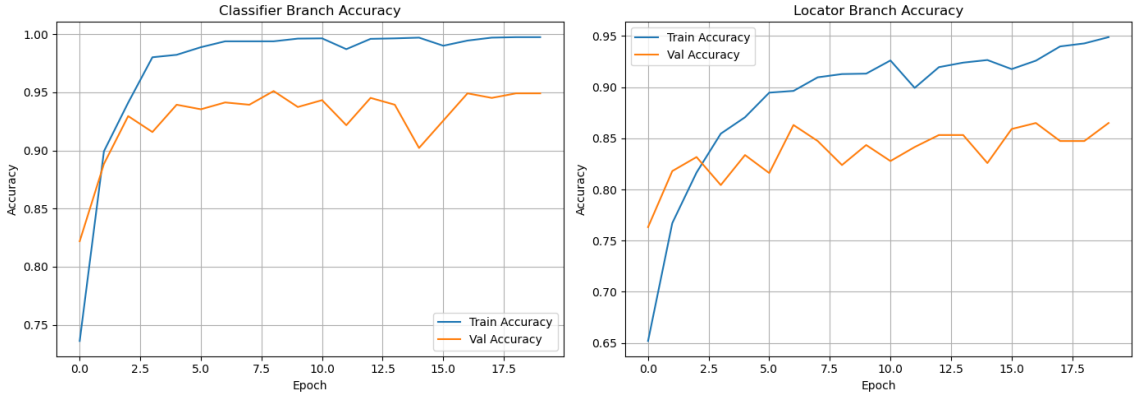


Рисунок 1 – Крива тренування: точність класифікації та локалізація протягом епох.

4. Розробка користувацького додатку

Функціонал програми включає:

– завантаження зображень у форматах JPEG/PNG;
– автоматичну обробку та аналіз;
– візуалізацію результатів із накладенням рамки та підписом типу пухлини;
– можливість збереження аналізованих зображень.

На рисунку 2 зображено блок-схему роботи користувача із системою.

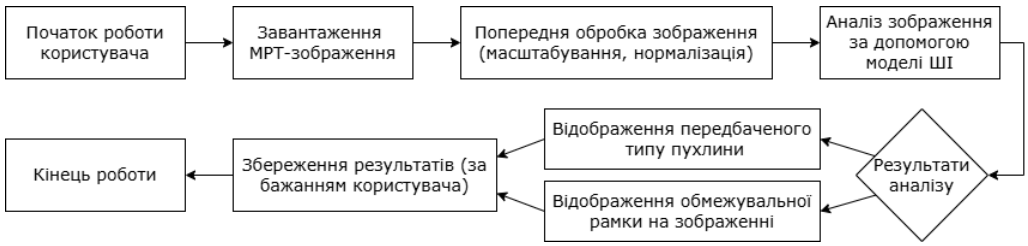


Рисунок 2 – Блок-схема роботи користувача із системою.

VI. Додаткові можливості (розширення в майбутньому)d

– Робота з об’ємними 3D-знімками (пакетна обробка томографічних даних).
– Мультикористувацький доступ для клінік (веб-версія).
– Інтеграція підтримки формату DICOM.
– Використання Explainable AI для пояснення рішень ШІ-моделі для лікарів.

Висновки

У результаті проведеної роботи створено теоретичну основу для розробки системи автоматичного виявлення пухлин мозку на основі методів глибокого навчання. Запропонована архітектура подвійного призначення дозволяє паралельно класифікувати тип пухлини та локалізувати її на МРТ-зображенні, що відкриває перспективи для інтеграції у клінічну практику.

Подальший розвиток системи передбачає розширення її функціональності за рахунок роботи з 3D-даними та підтримка різних форматів медичних зображень.

Список використаних джерел

1. Мінаєв Ю.М., Філімонова О.Ю. Розв’язання прикладних інженерних задач в нейронних мережах: навч.-метод. посібник. Київ: НАУ. – 2003, 75с.
2. Peleshchak R., Lytvyn V., Doroshenko M. Hechth–Nielsen theorem for a modified neural network with diagonal synaptic connections // *Mathematical Modeling and Computing*. – 2019. Vol. 6, No. 1, pp. 101–108

УДК 004.8

АРХІТЕКТУРА ДОДАТКУ ДЛЯ ОРГАНІЗАЦІЇ КОМАНДНОЇ РОБОТИ З ВИКОРИСТАННЯМ МАШИННОГО НАВЧАННЯ.

Крижанівський Р.В.[1)]
Західноукраїнський національний університет
[1)]*магістрант*

Постановка проблеми

Зі зростанням складності сучасних проектів у сфері інформаційних технологій та збільшенням кількості залучених учасників, питання ефективної організації взаємодії команди стає особливо актуальним. Важливими аспектами є оптимальне планування завдань, розподіл ресурсів, моніторинг прогресу та забезпечення прозорості комунікації. У зв'язку з цим виникає потреба у створенні програмних засобів, які б надавали менеджерам та членам команди можливість ефективно керувати проектною діяльністю. Особливий інтерес викликають системи, що інтегрують алгоритми машинного навчання для автоматизації аналізу та прийняття рішень.

Мета дослідження

Метою цієї роботи є розробка структури програмного забезпечення для підтримки командної роботи з використанням методів машинного навчання. Основна увага приділяється проектуванню багаторівневої архітектури, яка забезпечує модульність, масштабованість, інтеграцію інтелектуальних алгоритмів та зручність використання.

Структура системи

Проектована система має модульну багаторівневу архітектуру, побудовану з урахуванням вимог до масштабованості, продуктивності, гнучкості та підтримки інтелектуального аналізу даних. Система логічно поділена на п'ять основних структурних компонентів: клієнтський інтерфейс, серверна частина (Backend), підсистема машинного навчання, система управління даними та інфраструктура розгортання.

1. Клієнтська частина (Frontend) – це графічний інтерфейс користувача, через який здійснюється основна взаємодія з системою. Вона реалізована за допомогою фреймворку ReactJS[2], який забезпечує динамічну генерацію інтерфейсів та реактивне оновлення стану даних без перезавантаження сторінки. Для створення адаптивного дизайну, оптимізованого для різних пристроїв, використовується Tailwind CSS[3], що дозволяє ефективно реалізувати компоненти інтерфейсу користувача завдяки утилітарному підходу до стилізації.
2. Серверна частина (Backend API) реалізована за допомогою платформи Node.js, яка забезпечує обробку HTTP-запитів, реалізацію бізнес-логіки, перевірку даних та маршрутизацію. Серверна частина також виконує роль координатора між клієнтським інтерфейсом, підсистемою машинного навчання та базою даних.
3. Центральним інноваційним компонентом системи є модулі машинного навчання, реалізовані на основі TensorFlow[1] або подібних фреймворків. Ці модулі функціонують як окремі сервіси або мікросервіси, які отримують дані від серверної частини, виконують обчислення та повертають результати у форматі, придатному для інтерпретації. Підсистема машинного навчання включає такі модулі:
    a. Модуль кластеризації - групує користувачів або завдання за схожими характеристиками для оптимального розподілу навантаження.
    b. Модуль регресії – прогнозує час виконання завдань, кількість необхідних ресурсів або ризики затримки.
    c. Модуль рекомендацій – призначає завдання найбільш підходящим членам команди на основі історичних даних.
    d. Модуль аналізу поведінкових моделей – визначає індивідуальні стилі роботи, продуктивність у різних умовах, схильність до багатозадачності тощо.
    e. Модуль оцінки ефективності – генерує оцінки внеску кожного члена команди на основі кількісних та якісних показників.
4. Система управління базами даних використовується для зберігання інформації про користувачів, проекти, завдання, історію взаємодії та результати обчислень. Залежно від типу

даних та характеру запитів, можна використовувати як реляційні, так і документоорієнтовані СУБД, наприклад PostgreSQL[4] або MongoDB[5].

Для полегшення аналізу компонентів системи, на рисунку 1 зображено діаграму компонентів системи (UML Component Diagram), яка ілюструє основні компоненти системи та їх взаємодію.

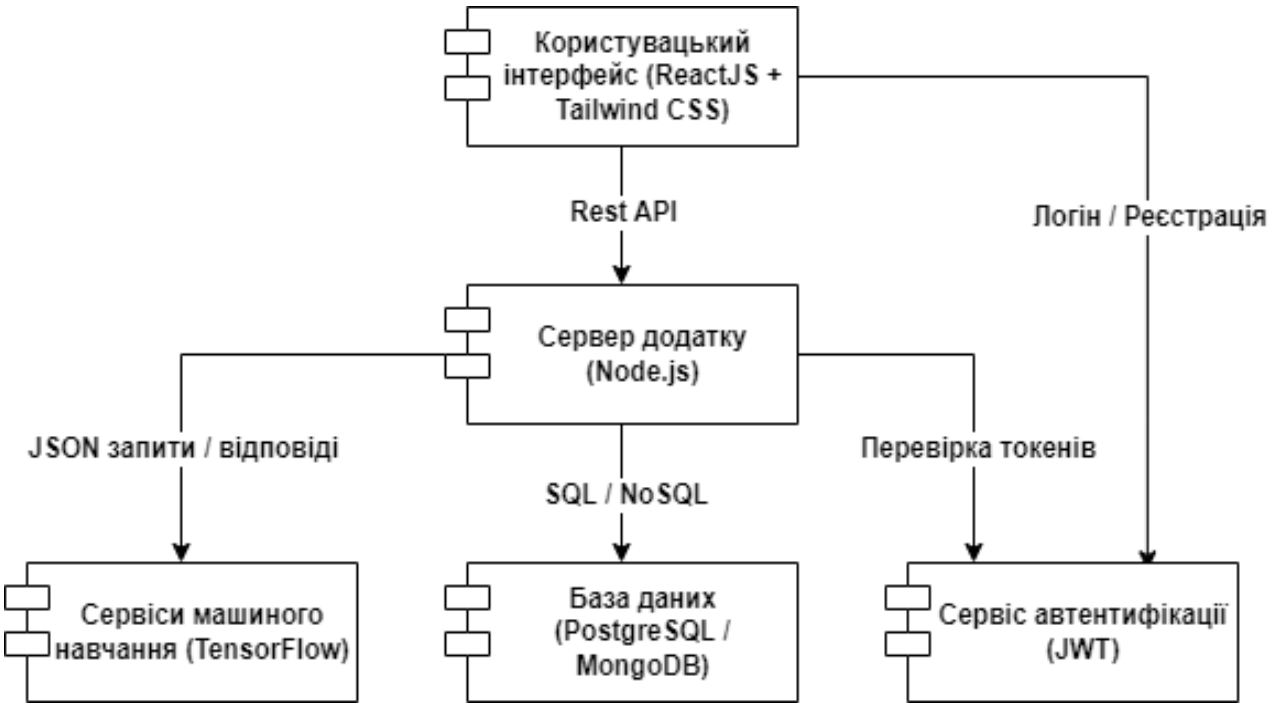


Рисунок 1 – Діаграма компонентів системи (UML Component Diagram).

Логіка взаємодії компонентів

Взаємодія між модулями системи реалізована за такою схемою: користувач взаємодіє з клієнтською частиною через веб-інтерфейс, який надсилає запити до серверної частини. Серверна логіка обробляє запити, звертається до бази даних та, за необхідності, ініціює обчислення з модулів машинного навчання. Результати повертаються користувачеві у візуалізованому вигляді, що забезпечує підтримку ефективного процесу управління в режимі реального часу.

Висновки

Запропонована архітектура програмного забезпечення демонструє системний підхід до вирішення проблеми організації командної роботи за допомогою методів машинного навчання. Вона забезпечує гнучкість, масштабованість та інтеграцію інтелектуальних інструментів, що дозволяє автоматизувати рутинні процеси та підвищити ефективність колективної діяльності. У майбутньому можливим напрямком розвитку цієї системи є інтеграція з популярними хмарними сервісами, такими як GoogleWorkspace або Microsoft Teams, що розширить функціональність платформи та спростить адаптацію користувачами. Крім того, перспективним є впровадження інструментів обробки природної мови (NLP) для автоматичного аналізу текстових комунікацій у команді, виявлення проблемних зон у взаємодії та підвищення якості управлінських рішень на основі комплексного аналізу даних.

Список використаних джерел

1. TensorFlow. An end-to-end open-source machine learning platform [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org.
2. ReactJS. A JavaScript library for building user interfaces [Електронний ресурс]. – Режим доступу: https://react.dev.
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. MongoDB. The developer data platform [Електронний ресурс]. – Режим доступу: https://www.mongodb.com.

УДК 004.4

СИСТЕМА МОНІТОРИНГУ СПОЖИТИХ РЕСУРСІВ

Маркевич Д.В.[1)], Крепич С.Я.[2)], Крепич Р.В.[3)]
[1-2)]Західноукраїнський національний університет
[3)] ВСП Кам'янець-Подільський фаховий коледж НРЗВО «Кам'янець-Подільський державний інститут»
[1)]бакалавр; [2)]к.т.н., доцент; [3)]викладач

I. Постановка проблеми

У світі спостерігається стрімке зростання споживання різних ресурсів – електроенергії, води, тепла, газу тощо. Зростаюче навантаження на інфраструктуру, підвищення вартості ресурсів, а також необхідність екологічного підходу до споживання потребують ефективних засобів контролю та управління [1-4]. У більшості випадків користувачі не мають уявлення про свої витрати ресурсів, що призводить до нераціонального споживання. У зв'язку з цим актуальним завданням є розробка програмного забезпечення, яке дозволяє здійснювати моніторинг споживаних ресурсів у реальному часі, фіксувати історію споживання, виявляти пікові навантаження, а також формувати рекомендації щодо оптимізації.

II. Мета роботи

Метою роботи є розробка системи моніторингу споживаних ресурсів, яка забезпечує збір, зберігання, обробку та візуалізацію даних. Передбачається створення інтуїтивного інтерфейсу для користувача, який дозволить переглядати інформацію про використання ресурсів, встановлювати обмеження, а також отримувати повідомлення у разі перевищення встановлених меж.

III. Основна частина

При розробці системи було здійснено детальний аналіз існуючих цифрових рішень у сфері контролю та оцінки вуглецевого сліду, а також загального споживання ресурсів. Зокрема, були розглянуті такі популярні мобільні додатки та сервіси, як GikiZero, CarbonFootprint та EarthHero. Кожен з них має певні переваги: наприклад, GikiZero пропонує базову оцінку вуглецевого сліду на основі анкетування, EarthHero дає загальні поради щодо зниження впливу на довкілля, а CarbonFootprint надає калькулятор для оцінки екологічного впливу побутової діяльності. Втім, було виявлено і низку істотних недоліків.

Найпоширенішими обмеженнями зазначених систем є відсутність можливості щоденного внесення даних про використання ресурсів, що унеможливлює точне відстеження динаміки змін споживання. Крім того, більшість платформ не підтримує гейміфікацію, яка могла б мотивувати користувачів повертатися до додатку, виконувати завдання, заробляти бали, досягати нових рівнів екологічної свідомості та отримувати нагороди. Ще одним важливим недоліком є відсутність інтерактивної взаємодії з екологічними консультантами, що могло б підвищити якість персоналізованих порад щодо скорочення вуглецевого сліду кожного окремого користувача.

З урахуванням зазначених проблем було розроблено власну систему моніторингу, яка має на меті забезпечити не лише функціональність збору даних, але й створення мотивуючого та освітнього середовища для користувача. Структура системи складається з кількох ключових модулів, кожен з яких виконує окрему роль:

- Модуль збору даних відповідає за прийом інформації від користувача у вигляді щоденних записів про споживання води, електроенергії, транспорту, продуктів харчування тощо.
- Модуль обробки здійснює аналіз отриманої інформації, нормалізацію, виявлення трендів і аномалій. Також він відповідає за розрахунок поточного рівня вуглецевого сліду користувача та генерує щомісячні звіти.
- База даних – централізоване сховище, реалізоване на основі PostgreSQL, у якому зберігаються всі дані користувача, історія змін, цілі, нагороди, коментарі, запити до еко спеціалістів тощо.
- Інтерфейс користувача – реалізований як сучасний веб-додаток з підтримкою адаптивної верстки. Він дозволяє користувачу взаємодіяти з системою максимально інтуїтивно: переглядати графіки динаміки споживання, поточний карбоновий слід, виконані завдання, отримані досягнення, таблицю лідерів, а також брати участь у тематичних екологічних челенджах.

На етапі проектування архітектури програмного забезпечення було створено діаграму класів (див.рис.1), що демонструє логіку взаємодії між основними сутностями системи: користувач, запис споживання, завдання, стаття, заявка на консультацію тощо.

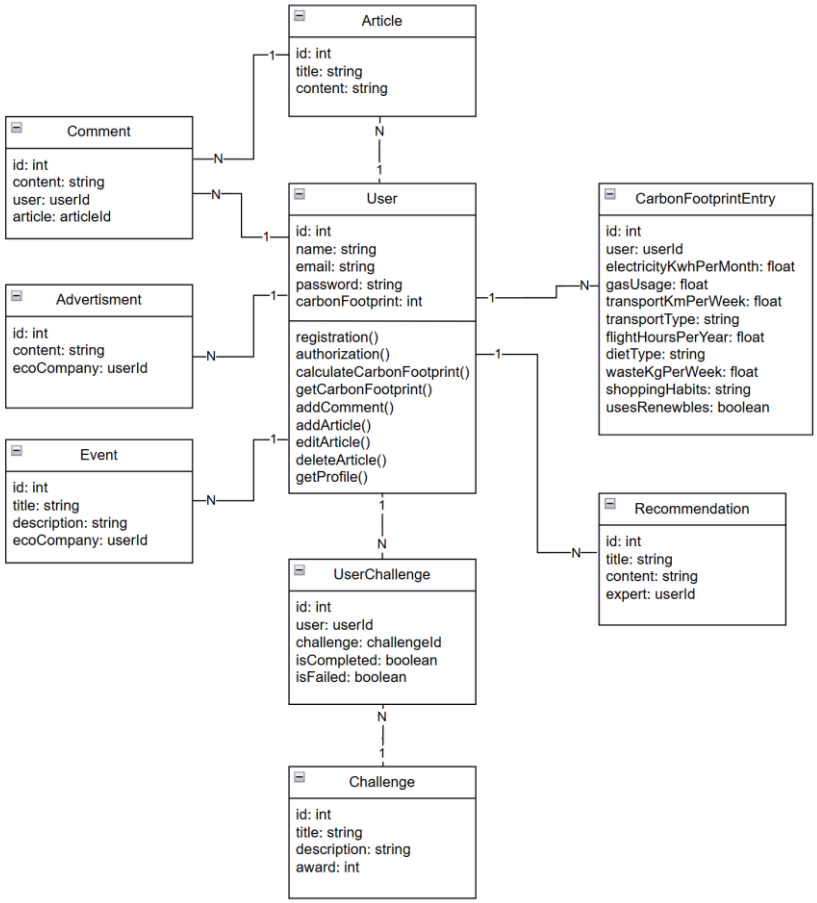


Рисунок 1 – Діаграма класів

Для реалізації серверної частини обрано технологічний стек NestJS,Prisma та PostgreSQL, що забезпечує гнучкість, модульність та високу продуктивність. Всі API розроблені у відповідності до REST-архітектури з урахуванням безпеки та масштабованості. Клієнтська частина створена з використанням React.js та NextJS, що дозволяє будувати швидкий, інтерактивний та SEO-дружній інтерфейс. Окрім основної інформаційної панелі, реалізовані додаткові секції:

- Блог із корисними статтями (див.рис.2) на тему сталого споживання, змін клімату, порад щодо зниження вуглецевого сліду. Користувачі можуть не лише читати публікації, а й залишати коментарі, ставити оцінки та ділитися думками.

- Система консультацій з еко-спеціалістами (див.рис.3) – інтегрована функція для створення заявки на персоналізовану онлайн-консультацію. Користувач заповнює форму, а фахівець у галузі екології зв'язується з ним у зручний час.
- Система гейміфікації (див.рис.4) – реалізовано щомісячні завдання з різними рівнями складності. За виконання користувачі отримують бали, відкривають нові рівні, підвищують свій статус в екосистемі та змагаються з іншими в таблиці лідерів.

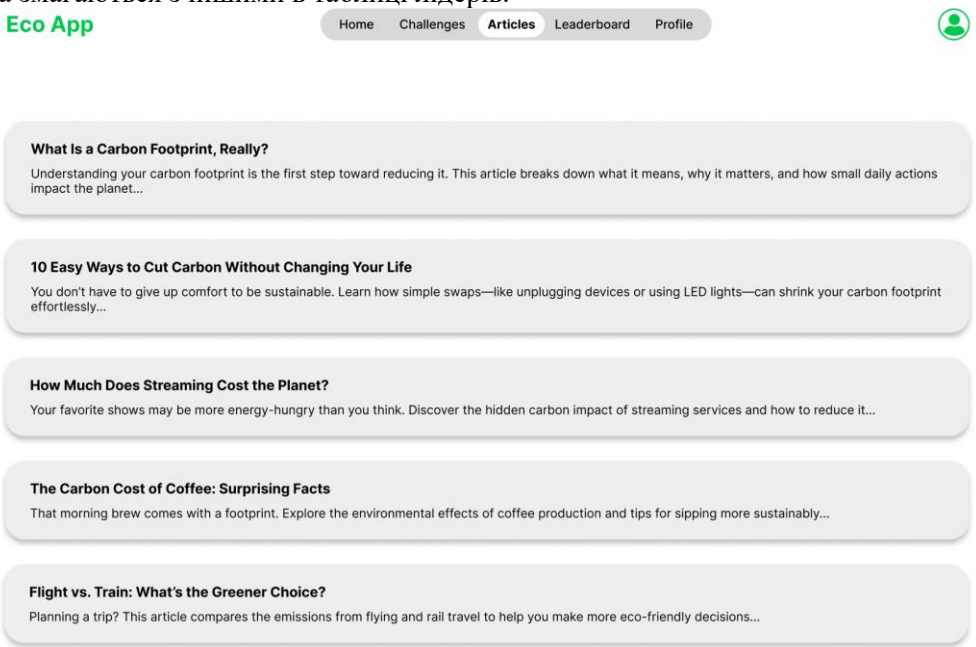


Рисунок 2 – Сторінка блогу

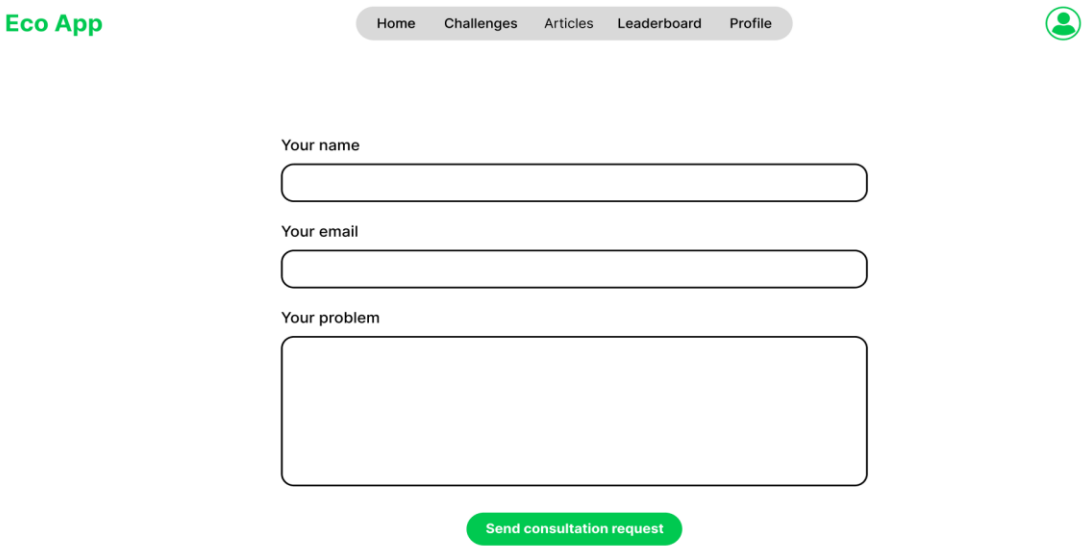
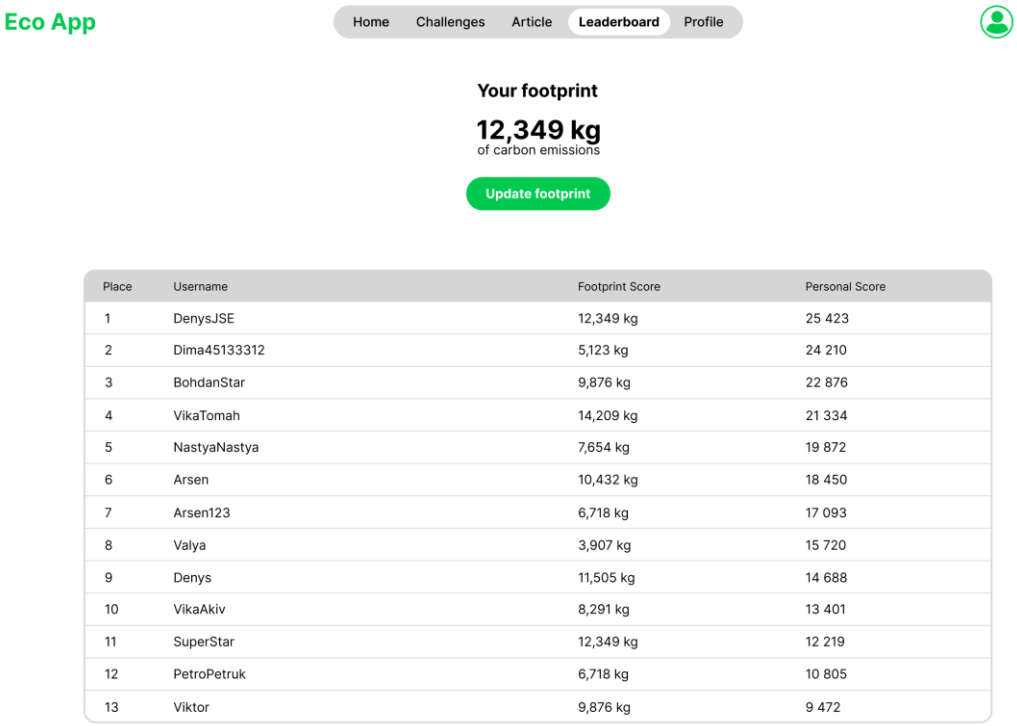


Рисунок 3 – Форма для залишення заявки



Place	Username	Footprint Score	Personal Score
1	DenysJSE	12,349 kg	25 423
2	Dima45133312	5,123 kg	24 210
3	BohdanStar	9,876 kg	22 876
4	VikaTomah	14,209 kg	21 334
5	NastyaNastya	7,654 kg	19 872
6	Arsen	10,432 kg	18 450
7	Arsen123	6,718 kg	17 093
8	Valya	3,907 kg	15 720
9	Denys	11,505 kg	14 688
10	VikaAkiv	8,291 kg	13 401
11	SuperStar	12,349 kg	12 219
12	PetroPetruk	6,718 kg	10 805
13	Viktor	9,876 kg	9 472

Рисунок 4 – Сторінка таблиці лідерів

Висновок

Розроблена система моніторингу споживаних ресурсів дозволяє користувачам ефективно контролювати витрати електроенергії, води, тепла тощо. Вона забезпечує гнучку інтеграцію з різноманітними пристроями, має зручний інтерфейс, підтримує аналітику та сповіщення. Використання такої системи сприятиме більш свідомому ставленню до споживання ресурсів, а також зниженню витрат для домогосподарств та підприємств.

Список використаних джерел

1. S. Krepych and I. Spivak, “Forecasting system of utilities service costs based on neural network”, *Advanced Information Systems*,Vol. 4, No. 4, pp.102–108

2. S. Krepych, I. Spivak and S. Spivak, “Approach to forecasting of utility costs using neural networks”, *15th International Conference on Computer Sciences and Information Technologies (CSIT)*, 2020, 1, pp.387-391

3. Крепич С.Я., Капуш М.В. та Співак І.Я., “Метод прогнозування витрат на комунальні послуги за допомогою нейронної мережі*”, CIT’2019*, 29 листопада 2019р., Тернопіль, стр. 14

4. Крепич С.Я., Капуш М.В. та Співак І.Я., “Система прогнозування витрат на комунальні послуги із використанням нейронних мереж”, *CSIT’2018*, 2 червня 2018р., Тернопіль, стр. 37-38

УДК 004.4

ВЕБ ЗАСТОСУНОК ДЛЯ ПРОВЕДЕННЯ ВОЛОНТЕРСЬКОЇ ДІЯЛЬНОСТІ

Співак І.Я.[1)], **Мартинюк М.В.**[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] бакалавр

І. Постановка проблеми

На сьогодні волонтерство відіграє надзвичайно важливу роль у суспільному житті, особливо в умовах воєнного часу, гуманітарних викликів та загальної соціальної нестабільності. Волонтери допомагають як на місцях, так і дистанційно — від евакуації людей і тварин до збору коштів і підтримки в освітніх чи медичних ініціативах. Проте сама система залучення людей до волонтерської діяльності все ще залишається хаотичною: інформація часто поширюється через соціальні мережі, в особистих чатах чи випадкових дописах, що значно ускладнює процес залучення нових учасників, особливо тих, хто не має зв'язків із певними організаціями.

У таких умовах виникла потреба створити цифрову платформу, яка б дозволила в єдиному онлайн-просторі об'єднувати волонтерів та організаторів проектів. Ця платформа мала б бути доступною, простою у користуванні та функціональною як для новачків, так і для досвідчених координаторів волонтерських ініціатив.

ІІ. Мета роботи

Метою моєї дипломної роботи стало проектування, розробка та впровадження сучасного веб-застосунку, який би виконував функцію онлайн-платформи для організації та координації волонтерської діяльності. На етапі визначення мети було враховано актуальність проблематики у сучасному українському суспільстві, зумовлену насамперед військовими діями, гуманітарною кризою та необхідністю оперативної взаємодії між волонтерами й організаторами допомоги.

У якості інструментарію для реалізації завдання обрано перевірені та сучасні технології:

- React.js — для розробки динамічного та інтуїтивно зрозумілого інтерфейсу користувача;
- Node.js із фреймворком Express — для створення надійної серверної логіки;
- MongoDB Atlas — як масштабоване хмарне сховище даних, що дозволяє зручно зберігати та обробляти великий обсяг інформації.

Такий стек технологій забезпечує високу швидкодію, кросплатформеність і гнучкість у подальшому розвитку системи.

Однією з ключових підцілей було створення простого у використанні та доступного інструменту як для користувачів без технічної підготовки (наприклад, волонтерів-початківців), так і для досвідчених організаторів, які координують велику кількість проектів. Додатково враховано перспективу розширення функціоналу, що дозволяє у майбутньому адаптувати систему під нові виклики або інтегрувати її з іншими платформами.

ІІІ. Реалізація системи

Платформа реалізована у вигляді односторінкового додатку з маршрутизацією на стороні клієнта. Я створив окремі сторінки для таких ключових функцій, як:

- Головна сторінка з короткою презентацією платформи;
- Реєстрація та авторизація користувача з токенами доступу (JWT);
- Пошук і перегляд волонтерських проектів за фільтрами (категорія, локація, дата);
- Особистий кабінет користувача (Your Dashboard), де волонтери бачать свої заявки, а організатори — створені ініціативи;
- Сторінка відгуків, де користувачі залишають враження після участі.

Серверна частина працює через Express, з чітко структурованим REST API. Усі дані зберігаються у MongoDB Atlas — це дало змогу уникнути локального хостингу і полегшило розгортання. База містить колекції користувачів, проектів, відгуків і заявок.

Для безпеки я реалізував хешування паролів (bcrypt), автентифікацію через JWT, обмеження доступу до захищених маршрутів, а також додав базовий захист від XSS-ін'єкцій та перевірку вхідних даних.

Рисунок 1 – Діаграма прецедентів

IV. Тестування та результати

Після розробки всі компоненти пройшли функціональне тестування. Перевірено десятки сценаріїв: від реєстрації користувача до створення та фільтрації проектів. Платформа також була перевірена на предмет сумісності з різними браузерами (Chrome, Firefox, Edge) і мобільними пристроями. В ході дослідної експлуатації система працювала стабільно навіть при одночасному доступі кількох користувачів.

Невеликі недоліки, які виникали в процесі тестування (наприклад, неповні повідомлення про помилки або проблеми в Safari), були усунені. Користувачі, які брали участь у дослідженні, позитивно оцінили інтерфейс, зрозумілу навігацію і загальну ідею платформи.

V. Висновок та перспективи

У рамках дипломної роботи було реалізовано веб-застосунок, покликаний вирішити проблему координації волонтерської діяльності в Україні. Під час розробки враховано ключові аспекти сучасної веб-архітектури: модульність, безпека, масштабованість і зручність для користувача. У результаті створено стабільний MVP-продукт, готовий до використання в реальному середовищі.

Система зручна як для волонтерів, які шукають можливість долучитися до ініціатив, так і для організаторів, що керують проектами. Інтуїтивний інтерфейс, гнучка архітектура та ефективна система фільтрів роблять платформу придатною для щоденного застосування.

У перспективі платформа може бути доповнена: інтеграцією з месенджерами (Telegram, Viber), багатомовністю, системою сповіщень, мобільним додатком та аналітикою для оцінки ефективності проектів.

Таким чином, створений застосунок не лише відповідає актуальним потребам користувачів, але й має потенціал до подальшого розвитку та масштабування.

Список використаних джерел

1. React documentation. [Електронний ресурс] – Режим доступу: https://react.dev
2. Express.js documentation. [Електронний ресурс] – Режим доступу: https://expressjs.com
3. MongoDB Atlas documentation. [Електронний ресурс] – Режим доступу: https://www.mongodb.com/atlas
4. bcrypt npm module. [Електронний ресурс] – Режим доступу: https://www.npmjs.com/package/bcrypt
5. JWT Introduction. [Електронний ресурс] – Режим доступу: https://jwt.io/introduction

УДК 004.4

АРХІТЕКТУРНІ ПАТЕРНИ ВЕБ-ДОДАТКУ ДЛЯ ТРЕКІНГУ ЩОДЕННИХ ЗВИЧОК НА БАЗІ ASP.NET

Крепич С.Я.[1)], **Салах О.М.**[2)]
Західноукраїнський національний університет
[1)] к.т.н. доцент; [2)] бакалавр

I. Постановка проблеми

Сучасні веб-додатки повинні відповідати високим вимогам до масштабованості, підтримуваності та безпеки. Прямолінійна реалізація без чіткого архітектурного розділення відповідальності призводить до зниження якості коду, ускладнює тестування, модифікацію функціоналу та інтеграцію нових компонентів. Окрім цього, відсутність структурованого підходу до організації внутрішньої логіки додатку ускладнює довготривалу підтримку проєкту. Саме тому доцільним є використання архітектурних патернів, які є перевіреними часом підходами для проектування застосунків.

II. Мета роботи

Метою цієї роботи є аналіз та обґрунтування використання архітектурних патернів у процесі розробки веб-додатку для трекінгу щоденних звичок на базі ASP.NET.

III. Основна частина

При розробці веб-додатку було застосовано трьохрівневу (3-tier) архітектуру, яка дозволяє логічно відокремити відповідальності між різними частинами системи, спростити підтримку та забезпечити масштабованість.

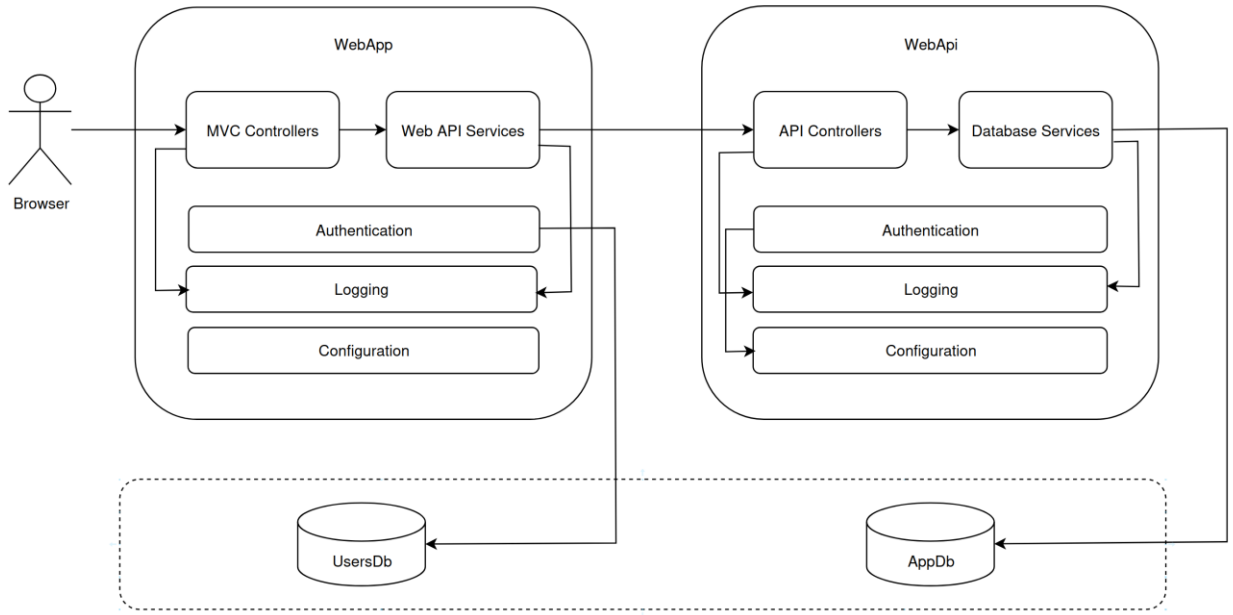


Рисунок 1 – Трьохрівнева архітектура додатку

Архітектура складається з трьох основних рівнів:

1. Рівень представлення (Presentation Tier)

Цей рівень реалізовано у вигляді ASP.NET Core MVC застосунку, що відповідає за взаємодію з кінцевим користувачем через браузер. Він дещо порушує принципи класичної багаторівевої архітектури, оскільки компонент аутентифікації напряму взаємодіє з рівнем даних (базою даних користувачів).

Основні компоненти:

- MVC контролери – обробляють запити браузера, формують відповіді у вигляді HTML.
- API сервіси – виконують HTTP запити до WebApi частини додатку.

Інші компоненти:

- Аутентифікація – генерація та перевірка токенів при доступі до методів контролерів.
- Логування.
- Конфігурація.

2. Рівень бізнес логіки (Application Tier)

Цей рівень реалізовано у вигляд ASP.NET Core WebApi застосунку. Від відповідає за обробку запитів з рівня представлення та реалізацію бізнес логіки.

Основні компоненти:

- REST API контролери – обробляють HTTP запити та делегують їхню обробку відповідним методам.
- Сервіси доступу бази даних – містять в собі як сервіси з бізнес логікою, так і репозиторії для доступу до даних.

Інші компоненти:

- Аутентифікація – перевірка вхідних токенів при доступі до захищеного ендпоінту.
- Логування.
- Конфігурація.

3. Рівень даних (Database Tier)

Цей рівень містить в собі дві бази даних:

- UsersDb – база для зберігання даних про користувачів.
- AppDb – база для зберігання усіх інших даних (звички, завдання, т.д.)

Головною перевагою цією архітектури є модульність, оскільки кожен можна змінювати та тестувати окремо.

На рівні презентації у проекті WebApp було використано Model-View-Controller (MVC) патерн, який є осноним способом повубови веб-застосунків в ASP.NET Core. Цей патерн дозволяє чітко розділити представлення, бізнес логіку та обробку запитів.

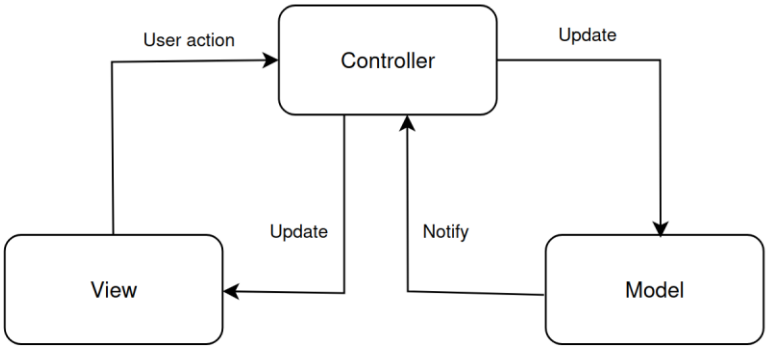


Рисунок 2 –Загальний вигляд патерну MVC

Проект WebApp використовує усі 3 головні компоненти MVC:

- Model – подаються у вигляді ViewModel-класів, які містять дані, що передаються між контролером і уявленням (View). Наприклад, модель для створення нової звички може включати поля назви, опису та дати початку.
- View – Razor-сторінки (.cshtml), що відповідають за візуальне представлення даних. Вони рендеряться на стороні сервера й повертаються користувачу у вигляді HTML-сторінок. Вони не містять бізнес-логіки, лише розмітку і базові умови/цикли.
- Controller – класи, що обробляють HTTP-запити, взаємодіють із Web API (через сервіси), обробляють результати, формують ViewModel і передають їх у відповідну View. Наприклад, TaskController може обробляти запит на створення нової звички (завдання), викликаючи відповідний метод API-клієнта і повертаючи результат у вигляді успішного HTTP коду 201.

Використання патерну MVC дає змогу чітко розділити відповідальність між різними компонентами, що у свою чергу забезпечує модульність та гнучкість додатку

У обох проектах використовується патерн впровадження залежностей (dependency injection) – один із базових принципів архітектури ASP.NET Core. Його суть полягає в тому, що об'єкти не створюють власні залежності самостійно, а отримують їх ззовні.

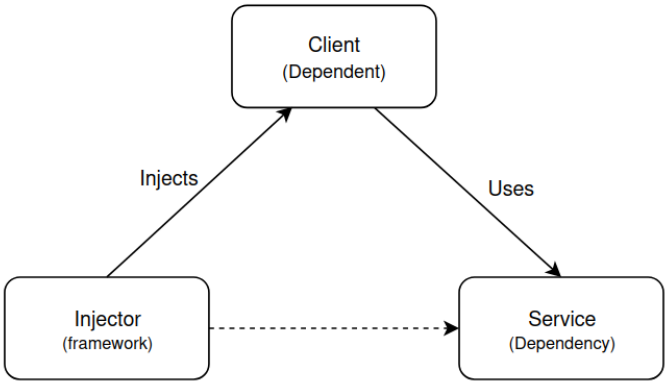


Рисунок 3 – Загальний вигляд патерну Dependency Injection

Наприклад, в проекті WebApp контролери не створюють Web API сервіси самостійно. Натомість сервіси реєстуютьсь при початку роботи програми і після цього впроваджуються через конструктор контролера (у даному випадку).

Патерн впровадження залежностей дає нам змогу контролювати життивий цикл об'єктів. Також він значно спрощує тестування додатку.

У проекті WebApi, тобто на рівні бізнес логіки, використовується патерн репозиторію (Repository), який абстрагує доступ до бази даних. Він виконує роль посередника між бізнес логікою та базою даних, тобто замість того, щоб безпосередньо взаємодіяти з Entity Framework у контролерах та сервісах, ці компоненти звертаються до репозиторіїв, які у свою чергу можуть бути отримані за допомогою патерну впровадження залежностей при попередній їх реєстрації у контейнері.

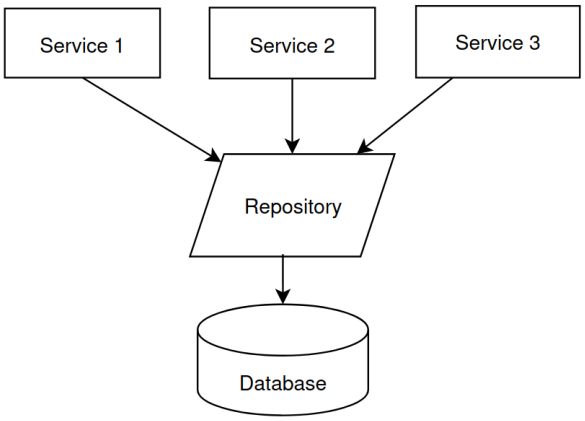


Рисунок 4 – Загальний вигляд патерну Repository

Патерн дозволяє досягти чистішої структури проекту, де обробка даних ізольована від бізнес логіки, що спрощує підтримута тестування системи.

Висновок

В роботі було проаналізовано та описано архітектурні рішення, використані при розробці веб-додатку для трекінгу щоденних звичок. Загалом, використання архітектурних і дизайнових патернів у проєкті підтвердило свою ефективність при створенні модульного веб-застосунку.

Список використаних джерел

1. ASP.NET Framework [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/introduction-to-aspnet-core?view=aspnetcore-9.0
2. What is the 3-Tier Architecture? [Електронний ресурс] – Режим доступу: https://www.tonymarston.net/php-mysql/3-tier-architecture.html
3. Overview of ASP.NET MVC -[Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-9.0
4. Крепич С.Я. Моделювання та забезпечення функціональної придатності статичних систем методами аналізу інтервальних даних. / С.Я. Крепич // дис. канд. техн. наук, НУ «Львівська політехніка», Львів, 2016. – 166с.

УДК 004.4

СИСТЕМА ДЛЯ ЛОГІСТИКИ ТА ЗБУТУ ТОВАРІВ МІСЦЕВОГО ВИРОБНИЦТВА

Папа О.А.[1)], **Антохі М.О.**[2)]
Західноукраїнський національний університет
[1)] д.філ.н., доцент; [2)] бакалавр

І. Постановка задачі

У сучасних умовах функціонування ринку малі та середні виробники продукції часто стикаються з рядом труднощів, пов'язаних із логістикою, контролем обліку, організацією доставки та ефективним управлінням замовленнями. Існуючі системи автоматизації або занадто дорогі, або не враховують особливості локальних підприємств. Крім того, багато інструментів орієнтовані виключно на онлайн-торгівлю, залишаючи поза увагою потреби в фізичній логістиці, плануванні маршрутів і роботі з клієнтами.

Завдання полягає в тому, щоб розробити програмне забезпечення, яке дозволить місцевим виробникам ефективно керувати базою товарів і клієнтів, формувати, обробляти та контролювати замовлення, оптимізувати маршрути доставки, автоматизувати облік оплат і контролювати фінансові потоки, а також здійснювати аналітичний моніторинг для прийняття управлінських рішень. Таке рішення повинно бути простим у використанні, недорогим у впровадженні та доступним як для технічно підготовлених, так і для пересічних користувачів.

ІІ. Мета роботи

Метою роботи є створення додатку для автоматизації процесів логістики та збуту товарів місцевого виробництва, який забезпечить повний цикл обробки інформації — від моменту надходження товару на склад до його доставки кінцевому споживачеві. Передбачається реалізація інструментів для ведення обліку товарів, управління замовленнями, контролю статусу оплат і доставок, аналітики продажів та взаємодії з клієнтами.

Особливу увагу приділено зручності використання, швидкодії інтерфейсу та можливості автономної роботи без постійного підключення до інтернету. Додаток орієнтований на локальні підприємства та повинен бути адаптований до реальних умов їх роботи: з обмеженими ресурсами, простим технічним забезпеченням і потребою в швидкому навчанні персоналу. Архітектура рішення має бути масштабованою, щоб у разі зростання бізнесу систему можна було легко адаптувати до нових вимог. Також важливим аспектом є забезпечення сумісності з майбутніми оновленнями програмного забезпечення та можливість інтеграції з іншими інформаційними системами. Очікується, що реалізований додаток стане ефективним інструментом підтримки управлінських рішень для малого та середнього бізнесу.

ІІІ. Проєктування системи

На основі аналізу предметної області та існуючих рішень було сформовано концепцію систему, яка охоплює ключові аспекти автоматизації логістичних та збутових процесів. До функціоналу системи входять можливості додавання, редагування та обліку товарів із категоризацією; управління замовленнями з переглядом статусу й обліком оплат; формування та оптимізація маршрутів доставки; ведення клієнтської бази з історією замовлень і контактною інформацією; а також аналітична звітність щодо продажів, залишків продукції на складі та популярності товарів.

Для реалізації відповідного функціоналу було спроєктовано реляційну базу даних, що включає п'ять основних сутностей:

- товар (ідентифікатор, назва, категорія, ціна, кількість на складі),
- клієнт (ідентифікатор, ім'я, адреса доставки, контактна інформація),
- замовлення (ідентифікатор, дата створення, статус, зв'язок з клієнтом),
- доставка (ідентифікатор, зв'язок із замовленням, адреса доставки, дата доставки)
- платежі (ідентифікатор, зв'язок із замовленням, сума, спосіб оплати).

Передбачені логічні зв'язки між таблицями:

- один товар може бути частиною кількох замовлень (зв'язок один до багатьох);
- кожне замовлення належить одному клієнту (один до одного);
- кожне замовлення має одну доставку (один до одного);

- кожне замовлення може бути пов’язане з кількома платежами (один до багатьох).

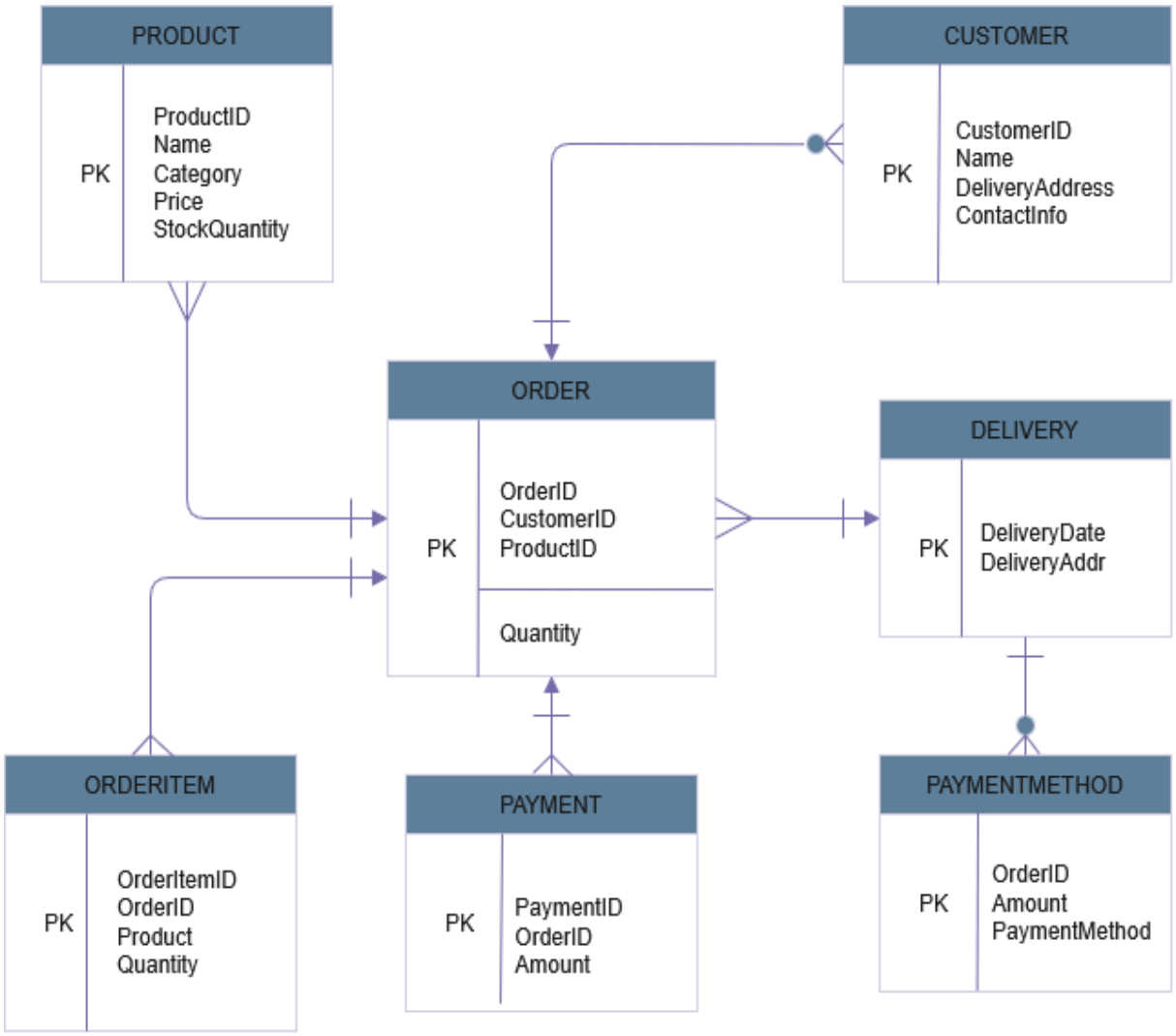


Рисунок 1 – UML діаграма

Для реалізації системи обрано стек технологій: мову програмування C#, фреймворк .NET Core[2] для серверної частини, базу даних SQLite[3] для зберігання даних, а також WPF для розробки інтерфейсу[4]. Уся розробка здійснювалася в середовищі Visual Studio, що забезпечило зручну інтеграцію з усіма компонентами проєкту. Завдяки застосуванню сучасних технологій було досягнуто високої продуктивності додатку та стабільності роботи при обробці великої кількості записів. Крім того, використання SQLite дало змогу реалізувати легку, портативну і водночас надійну систему збереження даних без потреби в окремому сервері БД. Завдяки використанню перевірених технологій і компонентів, система демонструє стабільну роботу навіть при зростанні кількості користувачів або збільшенні обсягу даних. Це дозволяє розгортати додаток як на окремих робочих станціях, так і в середовищі малого підприємства без суттєвих витрат на IT-інфраструктуру. Архітектура системи - класична клієнт-серверна, вона надає змогу використовувати механізми аутентифікації та авторизації користувачів.

IV. Реалізація системи

Результатом став повнофункціональний десктопний додаток з інтерфейсом, створеним на основі технології WPF. Програмна логіка побудована з використанням C# та .NET для обробки запитів, базу даних реалізовано на основі SQLite. Додаток забезпечує всі заявлені функції: управління товарами, замовленнями, клієнтами, доставками й платежами. Розроблена система має зручний інтерфейс, що дозволяє користувачам швидко виконувати повсякденні операції без необхідності спеціальної технічної підготовки. Основний акцент зроблено на інтуїтивну навігацію, простоту візуального оформлення та логічну послідовність дій користувача.

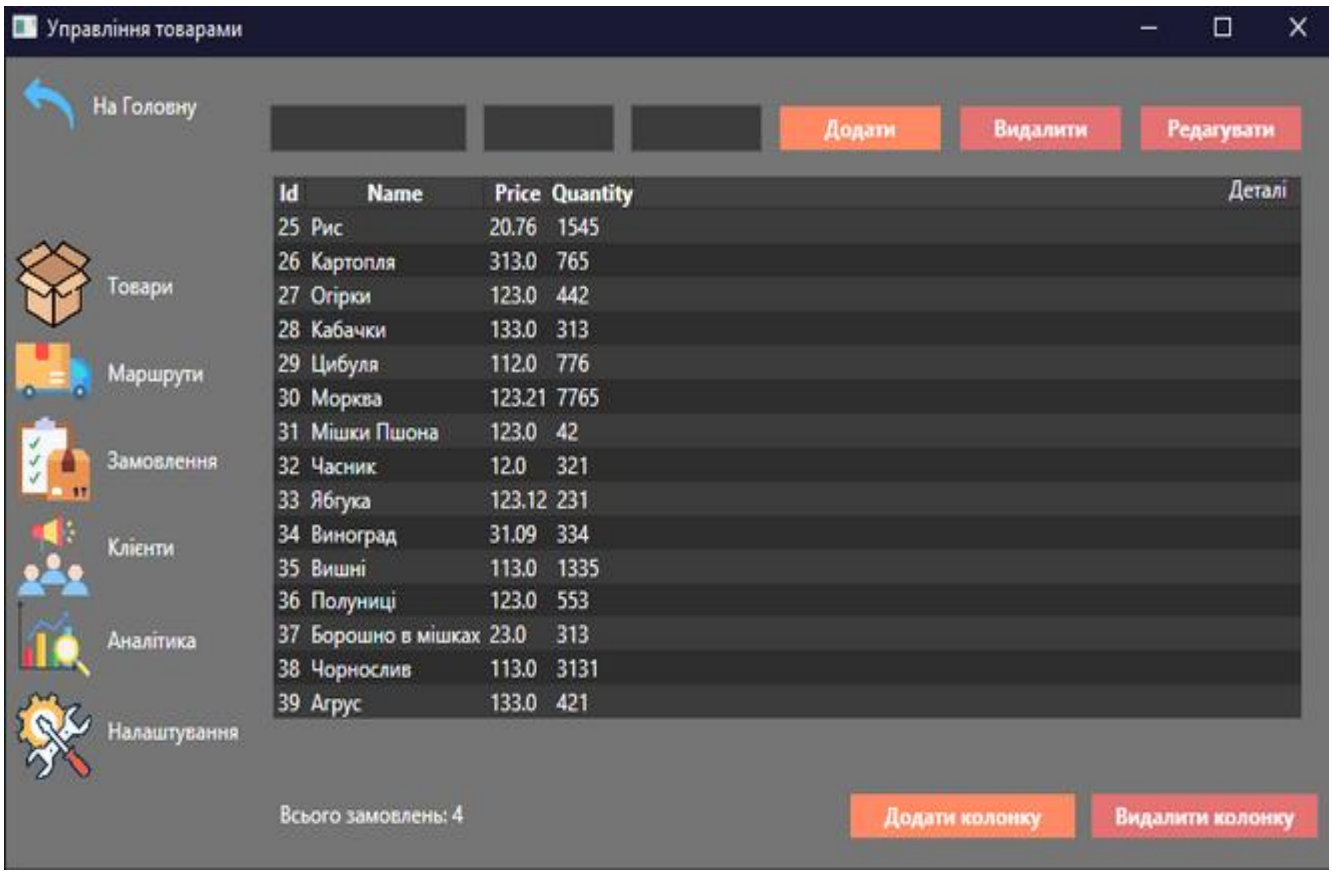


Рисунок 2 – Вікно управління товарами в додатку

З рисунка 2 можна помітити вікно управління товарами, можна сказати основне вікно додатку, воно дозволяє додавати, редагувати та видаляти товари крім того саму таблицю можна розширювати додаючи нові колонки або відповідно видаляючи не потрібні наприклад при необхідності додати колонку категорії, це уже залежить від потреб користувача, сам додаток з коробки пропонує осноні три категорії тобто Ім'я, ціна та кількість товару. Вміст вікна відповідає елементу PRODUCT діаграми з рисунка 1 тобто елементи з відси формуються в замовлення яке дал оплачується та доставляється.

V. Висновок

У результаті роботи створено додаток для логістики та збуту товарів місцевого виробництва, який відповідає вимогам простоти, ефективності та доступності для локальних виробників. Він дозволяє створювати таблиці з товарами, клієнтами, таблиці замовлень та маршрутів, а також проводити різноманітні маніпуляції з ними: редагування, сортування, фільтрацію та пошук. Крім того, додаток надає звіт у PDF-форматі, який включає в себе широкий спектр інформації, такої як кількість проданих товарів, дата замовлень, статуси доставок та фінансові дані. Інтерфейс системи є інтуїтивно зрозумілим, що дозволяє мінімізувати час на навчання персоналу. Система сприятиме підвищенню ефективності обліку продукції, планування доставки та взаємодії з клієнтами, що, у свою чергу, позитивно вплине на розвиток місцевого бізнесу та зміцнення конкурентоспроможності малих підприємств. Важливо зазначити, що структура розробленого рішення дозволяє легко адаптувати його під потреби конкретного підприємства завдяки гнучкій архітектурі та модульному підходу. Це відкриває можливості для подальшого вдосконалення системи — зокрема, розширення її функціональності, інтеграції з іншими сервісами або переходу до хмарних рішень у майбутньому.

Список використаних джерел

1. Freeman A., Sanderson A. *Pro WPF in C# 2022.* Apress, 2022.
2.Офіційна документація ASP.NET Core [Електронний ресурс] – Режим доступуhttps://learn.microsoft.com//aspnet/core
3."Що таке SQLite?" [Електронний ресурс] – Режим доступу: https://www.sqlite.org/
4. Korman B. *Beginning WPF with C#.* Apress, 2021.

УДК 004.4

АНАЛІЗ .NET MAUI ДЛЯ СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДОСТАВКИ ЇЖІ

Городиський Н.І.
Західноукраїнський національний університет
бакалавр

I. Постановка проблеми

Ринок доставки їжі в Україні стрімко зростає: у 2023 році його обсяг досяг $1.5 млрд із прогнозованим зростанням на 10–15% щороку [1,3]. Молодь (18–35 років, 65% користувачів) віддає перевагу мобільним застосункам через зручність і швидкість доставки, а середній чек зріс на 15% порівняно з 2022 роком через економічні зміни [1]. Конкуренція між сервісами, такими як Glovo (частка ринку 60%) та Raketa, вимагає кросплатформних рішень із високою продуктивністю та мінімальними витратами на розробку, адже 70% замовлень припадає на міста з населенням понад 500 тис. осіб [1]. Вибір оптимального фреймворку для таких застосунків є актуальною задачею [2].

У таких умовах зростає потреба у швидкому створенні стабільних та ефективних мобільних застосунків, які підтримують як Android, так і iOS, оскільки 90% користувачів в Україні використовують ці платформи [3].

Розробники стикаються з вибором фреймворку, який дозволяє створювати додатки з високою продуктивністю, мінімальними витратами та адаптивним дизайном. Оскільки користувачі очікують швидкого відгуку (до 30 хвилин доставки), стабільної роботи та зручного інтерфейсу, ключовими критеріями стають продуктивність, швидкість розробки, підтримка платформ і здатність інтегрувати сучасні функції, як-от геолокація [1–3].

II. Мета роботи

Метою роботи є аналіз фреймворку .NET MAUI як інструменту для створення кросплатформного мобільного застосунку доставки їжі з підтримкою геолокації та push-повідомлень, порівняння його з Flutter та ReactNative за продуктивністю, розміром APK і часом розробки, а також обґрунтування вибору .NET MAUI для реалізації такого проєкту на українському ринку [2,5–7].

III. Основна частина

Розробка кросплатформних мобільних застосунків доставки їжі потребує фреймворку, який забезпечує високу продуктивність, зручність розробки та підтримку сучасних платформ, адже користувачі очікують швидкої доставки (до 30 хвилин) і стабільної роботи на різних пристроях [1,3]. .NET MAUI, Flutter і ReactNative є провідними технологіями для створення таких застосунків, але їхні характеристики різняться залежно від потреб проєкту, таких як інтеграція геолокації, обробка великих обсягів даних і забезпечення адаптивного дизайну [8].

Для обґрунтованого вибору фреймворку було проаналізовано .NET MAUI, Flutter та ReactNative за низкою параметрів [2,5–7]: мова програмування, інтерфейс користувача, продуктивність, інструменти розробки, підтримка платформ, рівень підтримки спільноти, а також легкість у навчанні. Ці критерії є ключовими для оцінки, оскільки вони впливають на швидкість розробки, якість кінцевого продукту та можливість масштабування в умовах зростаючого попиту на ринку доставки їжі, де конкуренція між сервісами, такими як Glovo, вимагає швидкого виведення продукту на ринок [1,3].

Результати аналізу подано у вигляді порівняльної таблиці, яка дає змогу наочно оцінити переваги та недоліки кожного фреймворку для конкретного сценарію використання.

Таблиця №1

Порівняння .NET MAUI, Flutter і ReactNative

Параметр	.NET MAUI	Flutter	ReactNative
Мова та екосистема	.NET MAUI використовує C# та екосистему .NET — потужну мову з широким набором бібліотек і тісною інтеграцією з VisualStudio.	Flutter працює на Dart — менш популярній, але простій у вивченні мові з лаконічним синтаксисом та хорошими інструментами (FlutterDevTools).	ReactNative базується на JavaScript — доступній мові з величезною екосистемою, гарячим перезавантаженням і швидким циклом розробки.

Продовження таблиці №1

Інтерфейс користувача	.NET MAUI використовує нативні UI-компоненти кожної платформи, що забезпечує природний вигляд додатків.	Flutter рендерить інтерфейс самостійно через Skia, що дозволяє створювати гнучкі, кросплатформні й візуально якісні UI.	ReactNative поєднує JavaScript з нативними компонентами через міст, що може викликати затримки, але підтримується великою кількістю готових UI-бібліотек.
Продуктивність	.NET MAUI використовує нативнийрендеринг і виграє від AOT-компіляції та оптимізацій .NET 6, що дає високу продуктивність.	Flutter забезпечує стабільні 60 FPS завдяки мові Dart і рушію Skia.	ReactNative покладається на міст між JS і нативним кодом, що може впливати на продуктивність, але її можна покращити оптимізацією та нативними модулями.
Інструменти розробки	.NET MAUI має підтримку VisualStudio, екосистему Xamarin.Forms і NuGet. Microsoft забезпечує хорошу документацію й інструменти.	Flutter пропонує якісну документацію, багато плагінів через pub.dev і зручну інтеграцію з IDE.	ReactNative має велику спільноту, багато бібліотек, хорошу документацію від Facebook і зручні інструменти, як Expo.
Підтримка платформ	.NET MAUI охоплює Android, iOS, macOS і Windows, підтримує різні форм-фактори — від телефонів до ПК.	Flutter працює на Android, iOS, вебі, macOS і Windows, зокрема підтримує PWA.	ReactNative зосереджений на Android і iOS, але спільнота розширює підтримку інших платформ. Фреймворк залишається мобіле-орієнтованим.
Підтримка спільноти	.NET MAUI підтримується Microsoft, яка відома довгостроковою підтримкою своїх технологій — це позитивний сигнал для майбутнього фреймворку.	Flutter стрімко набрав популярність завдяки Google та активній спільноті, що свідчить про його перспективність.	ReactNative має велику спільноту та відкритий розвиток, хоча участь Facebook зменшилася.
Крива навчання	.NET MAUI має нестрімку криву навчання для тих, хто вже знає C# та .NET. Новачкам у C# знадобиться час на освоєння мови й фреймворку.	Flutter зі своїм підходом на віджетах зручний для тих, хто знайомий з декларативним UI, але може бути складним для прихильників імперативного стилю.	ReactNative з JavaScript та React легко освоїти тим, хто вже має досвід з цими технологіями.

Як видно з таблиці, кожен із розглянутих фреймворків має свої сильні та слабкі сторони, що впливають на їхню придатність для створення застосунків доставки їжі. .NET MAUI вирізняється нативністю завдяки використанню UI-компонентів платформ, що забезпечує природний вигляд і швидкий доступ до функцій, таких як геолокація, а також інтеграцією з Microsoft Azure для обробки даних у реальному часі, що важливо для відстеження замовлень [5,7]. Flutter підходить для швидкого створення візуально привабливих інтерфейсів завдяки багатій бібліотеці віджетів і функції hotreload, що дозволяє оперативно тестувати дизайн каталогу страв [4,7]. ReactNative, своєю чергою, забезпечує доступність для широкого кола розробників через використання JavaScript і підтримку бібліотек, таких як Expo, що спрощує інтеграцію з платіжними системами [6]. Проте для ринку доставки їжі, де користувачі цінують швидкість доставки (до 30 хвилин) і безперебійну роботу, .NET MAUI є кращим

вибором завдяки своїй здатності забезпечувати стабільність, підтримку Microsoft і сумісність із широким спектром пристроїв, що допомагає конкурувати з лідерами, такими як Glovo [1, 2, 6, 7].

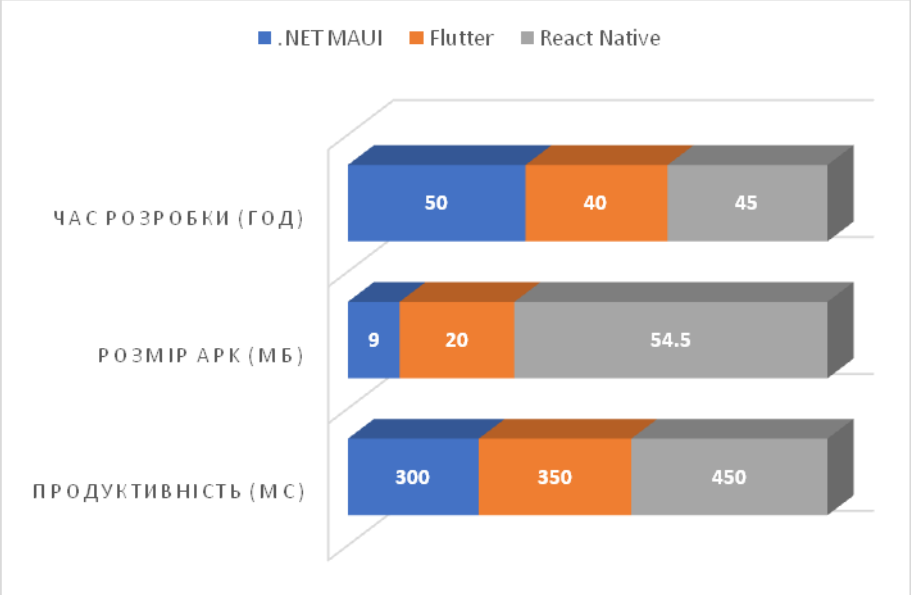


Рисунок 1 – Порівняння продуктивності .NET MAUI, Flutter і ReactNative.

.NET MAUI демонструє високу продуктивність (час запуску ~300 мс) і компактний розмір застосунку (~9 МБ), що робить його ефективним рішенням для створення швидких і стабільних мобільних додатків, здатних працювати на різних пристроях [2,5]. Хоча час розробки трохи вищий (~50 годин), він компенсується зручністю для розробників, які вже знайомі з C# та .NET, а також можливістю використання нативних API для геолокації [7,8]. Це робить .NET MAUI перспективним вибором для реалізації застосунку доставки їжі, особливо в умовах вимог до продуктивності, масштабованості та інтеграції з локальними сервісами [2,6,7].

Висновок

Аналіз показав, що .NET MAUI є доцільним вибором для розробки кросплатформних застосунків доставки їжі. Фреймворк забезпечує високу продуктивність (час запуску ~300 мс), невеликий розмір APK (~9 МБ) і стабільну підтримку Microsoft [2,5]. Це важливо для українського ринку, де швидкість, ефективність на різних пристроях і конкуренція з лідерами, такими як Glovo, є ключовими [1,3]. Подальші дослідження можуть стосуватися інтеграції штучного інтелекту для персоналізації замовлень, а також оптимізації енергоспоживання застосунку для підвищення його енергоефективності [4]. Крім того, варто розглянути можливість використання .NET MAUI для створення веб-версії застосунку, що розширить доступність сервісу для користувачів, які віддають перевагу браузеру [2,7]. Такий підхід може додатково зміцнити позиції на ринку, враховуючи зростання популярності веб-доступу до сервісів доставки [3].

Список використаних джерел

1. Сухорукова Г. Попит на доставку їжі та напоїв в Україні. KyivstarBusinessHub – корпоративний блог для бізнесу. URL: https://hub.kyivstar.ua/articles/stan-rinku-fud-dostavki-v-ukrayini-yak-zrostaye-popit-na-dostavku-yizhi-ta-napoyiv.
2. Muruganandham. .NET MAUI vs. Fluttervs. ReactNative: A Comprehensive Comparison. KANINI. URL: https://kanini.com/blog/net-maui-vs-flutter-vs-react-native/.
3. Genesis Dynamics GD. Europeon-demand food delivery platforms market size, industry trends&forecast. LinkedIn: LogInorSignUp. URL: https://www.linkedin.com/pulse/europe-on-demand-food-delivery-platforms-market-size-qvbmf/.
4. Devs E. A. 8 Best Practices for Flutter Developers in 2025. Medium. URL: https://medium.com/@expertappdevs/8-best-practices-for-flutter-developers-in-2025-d26f7780b9d2.
5. Longe T. NET MAUI vs ReactNative – Comparison of App Frameworks 2025. UXCamBlog. URL: https://uxcam.com/blog/net-maui-vs-react-native.
6. Infosoft D. React nativevs fluttervs .NET MAUI: which framework to choose for your next mobile app development? - DP infosoft. DP Infosoft. URL: https://dpinfosoft.com/react-native-vs-flutter-vs-dotnet-maui/.
7. Maksym M., Muzyka R. .NET MAUI vs. Flutter: Head-to-Head Comparison and UseCases. Leobit. URL: https://leobit.com/blog/net-maui-vs-flutter-head-to-head-comparison-and-use-cases-for-the-most-popular-cross-platform-development-frameworks/.
8. Sorathiya N. Exploring the power of the flutter geolocator. DhiWise – Build Sites&Apps from YourIdeas, Instantly. URL:https://www.dhiwise.com/post/maximizing-user-experience-integrating-flutter-geolocator.

УДК 004.4

ПІДВИЩЕННЯ БЕЗПЕКИ ДАНИХ У ХМАРНИХ СХОВИЩАХ ЗА ДОПОМОГОЮ ПРОАКТИВНИХ МЕХАНІЗМІВ СПОВІЩЕНЬ

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

Постановка проблеми

У сучасному цифровому світі хмарні сховища перетворилися на фундаментальний елемент IT-інфраструктури для організацій та індивідуальних користувачів. Вони пропонують безпрецедентну гнучкість, масштабованість та доступність даних. Однак, разом із перевагами, централізація великих обсягів інформації в хмарі створює значні виклики у сфері безпеки. Забезпечення конфіденційності, цілісності та доступності даних, що зберігаються та обробляються у хмарних середовищах, залишається критично важливим завданням.

Традиційні підходи до захисту даних, такі як механізми контролю доступу на основі ролей, політики паролів та шифрування даних (як при передачі, так і при зберіганні), безумовно, є необхідною основою безпеки. Проте, вони часто виявляються реактивними за своєю природою. Ці методи ефективні проти відомих типів атак, але можуть бути недостатніми для протидії новим, складним та цілеспрямованим загрозам, таким як атаки нульового дня, інсайдерські загрози, фішингові атаки, що ведуть до компрометації облікових записів, або складні атаки програм-вимагачів. Сучасний ландшафт кіберзагроз вимагає переходу від суто реактивних до більш проактивних стратегій захисту.

Мета роботи

Мета роботи полягає у дослідженні та розробці моделі системи проактивних сповіщень для підвищення рівня безпеки у хмарних сховищах даних. Завдання включають: аналіз типових загроз безпеці для хмарних сховищ; ідентифікацію подій та патернів активності, що можуть свідчити про порушення безпеки; розробку правил та алгоритмів для виявлення аномалій на основі аналізу логів дій користувачів (авторизації, операції з файлами); проектування механізму генерації та доставки сповіщень відповідним користувачам або адміністраторам системи.

Основна частина

Пропонується розробити систему, що базується на аналізі логів активності користувачів у хмарному сховищі. Система буде відстежувати такі події, як спроби входу в систему з незвичних геолокацій або IP-адрес, багаторазові невдалі спроби авторизації, масове видалення або завантаження файлів, спроби доступу до файлів іншими користувачами без відповідних прав. На основі визначених правил будуть генеруватися сповіщення різного рівня критичності (інформаційні, попереджувальні, критичні), які надсилатимуться користувачам або адміністраторам через обрані канали (електронна пошта, сповіщення в інтерфейсі системи).

Програмна реалізація

Для реалізації системи обрано сучасний та надійний стек веб-технологій. Серверна частина (backend) реалізована на базі фреймворку NestJS (Node.js), що забезпечує високу продуктивність, модульність і підтримку TypeScript. Клієнтська частина (frontend) створена з використанням бібліотеки React, яка дозволяє реалізовувати динамічні та інтерактивні інтерфейси для користувачів і адміністраторів.

Для зберігання логів активності використано реляційну СУБД PostgreSQL, яка вирізняється надійністю, масштабованістю та ефективною підтримкою складних запитів. Для оптимізації пошуку та фільтрації даних реалізовано індексацію за ключовими параметрами: час події, ідентифікатор користувача, тип події.

На поточному етапі реалізовано базовий функціонал – модуль перегляду логів активності користувачів, що забезпечує централізований збір, надійне зберігання та візуалізацію основних дій у системі (авторизації, операції з файлами). Інструмент призначено для адміністраторів системи з метою аудиту безпеки, аналізу поведінки користувачів та виявлення підозрілої активності.

Інтерфейс модуля реалізовано у вигляді інтерактивних таблиць (див. рисунки 1, 2) з можливістю сортування та застосування фільтрів за параметрами: діапазон дат, користувач, тип події (наприклад, login, upload, delete, share), IP-адреса тощо.

id serial	ipAddress v...	browser...	platform va...	deviceTy...	location varchar	userAgent varchar	loginTime timestamp	userId...
1	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-02-27 00:15:44.668	1
2	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-02-28 03:12:12.275	1
4	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-04-30 01:36:12.488	NULL
5	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-04-30 01:42:09.203	NULL
6	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-04-30 01:42:13.171	NULL
7	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-04-30 01:43:42.242	NULL
8	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-04-30 01:44:17.927	NULL
9	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-04-30 01:44:57.776	NULL
10	5.58.95.197	Chrome	Windows	Desktop	Ukraine, Ternopil	Mozilla/5.0 (Win…	2025-04-30 01:45:41.47	NULL

Рисунок 1 – Сторінка перегляду логів авторизації

id serial	action var...	createdOn timestamp	userId...	fileId in...	shared...	user	file	shared_file
1	create	2025-02-26 22:17:16.50…	2	1	NULL	user	file	shared_file
2	delete	2025-02-26 22:17:25.56…	2	1	NULL	user	file	shared_file
3	create	2025-02-26 23:45:48.54…	2	2	NULL	user	file	shared_file
4	download	2025-02-28 00:05:36.54…	2	2	NULL	user	file	shared_file
5	download	2025-02-28 00:07:00.78…	2	2	NULL	user	file	shared_file
6	download	2025-02-28 00:23:56.61…	2	2	NULL	user	file	shared_file
7	download	2025-02-28 01:14:25.43…	2	2	NULL	user	file	shared_file
8	create	2025-02-28 01:14:49.86…	2	3	NULL	user	file	shared_file
9	download	2025-02-28 01:14:58.50…	2	3	NULL	user	file	shared_file

Рисунок 2 – Сторінка перегляду логів дій

Висновок

У роботі запропоновано концепцію та архітектурну модель системи проактивних сповіщень, спрямовану на суттєве посилення безпеки даних у хмарних сховищах. На відміну від традиційних реактивних підходів, запропонована система фокусується на ранньому виявленні потенційно небезпечних дій та аномалій через аналіз логів активності. Впровадження такої системи дозволить значно підвищити обізнаність користувачів та адміністраторів щодо підозрілої активності, пов'язаної з їхніми обліковими записами або системою загалом. Це, в свою чергу, забезпечить можливість швидкого реагування на критичні сповіщення, наприклад, шляхом зміни пароля чи блокування підозрілих сеансів, що критично важливо для мінімізації ризиків несанкціонованого доступу, витоку даних чи інших інцидентів безпеки. Реалізований на даний момент модуль перегляду логів є важливим фундаментом для подальшого розвитку. Майбутня робота концентруватиметься на імплементації алгоритмів виявлення аномалій та розробці гнучкого механізму генерації й доставки сповіщень. Запропонований проактивний підхід до моніторингу безпеки може бути інтегрований в існуючі платформи хмарних сховищ або реалізований як окреме рішення, що дозволить значно покращити їх загальну захищеність та стійкість до сучасних кіберзагроз.

Список використаних джерел

1. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – [Б. м.] : Prentice Hall, 2017. – 432 с.
2. Clean Code: A Handbook of Agile Software Craftsmanship. – [Б. м.] : Prentice Hall, 2008. – 431 с.
3. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 624 с.
4. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. – [Б. м.] : O'Reilly Media, 2020. – 330 с.
5. Code complete: [a practical handbook of software construction]. 2-ге вид. Redmond, Wash : Microsoft Press, 2004. 914 с.
6. Documentation | NestJS - A progressive Node.js framework [Електронний ресурс] – Режим доступу: https://docs.nestjs.com/
7. PostgreSQL: Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/

УДК 004.8

РОЗРОБКА ІНТЕГРОВАНОЇ РОБОТИЗОВАНОЇ СИСТЕМИ-НАГАДУВАННЯ ДЛЯ ЛЮДЕЙ ІЗ ДЕМЕНЦІЄЮ

Ваверчак Я.В.[1)], **Папа О.А.**[2)]
Західноукраїнський національний університет
[1)]бакалавр; [2)]д.філ.н., доцент

І. Постановка проблеми

У сучасному суспільстві зростає потреба в інноваційних рішеннях для підтримки людей із когнітивними порушеннями, зокрема деменцією. Все більше людей похилого віку залишаються на самоті або перебувають під доглядом близьких, які змушені поєднувати турботу з роботою. Це створює додаткове емоційне та фізичне навантаження на доглядальників.

Саме пацієнти, які страждають на деменцію, часто не здатні самостійно дбати про себе, забувають про важливі щоденні дії — як-от прийом ліків чи гігієнічні процедури. Багато існуючих цифрових рішень, таких як нагадування на телефоні чи голосові помічники, не сприймаються або ігноруються користувачами похилого віку.

До того ж, психологічна ізоляція та тривожність значно посилюються в умовах відсутності стабільного зовнішнього контролю чи нагадувань. Тому виникає потреба у більш «живій» та доступній формі підтримки — системі, яка не просто повідомляє, а взаємодіє з людиною.

ІІ. Мета роботи

Метою є створення зручної в користуванні, адаптивної та емоційно приємної системи-нагадування у вигляді робота, який може ефективно нагадувати про необхідні дії. Система повинна допомогти зберігати рутину користувача та зменшити необхідність у постійному втручанні сторонніх осіб.

Додатково ця система має слугувати засобом комунікації для опікунів: надавати їм оперативну інформацію про стан підопічного, дії, які він виконав, або ж ситуації, коли втручання необхідне. Таким чином, вона виступає як сполучна ланка між людиною з деменцією та її доглядальником.

ІІІ. Особливості реалізації та соціальне значення системи

Розроблений прототип базується на ідеї інтерактивного помічника у формі невеликого робота, що імітує поведінку домашнього улюбленця. Такий формат викликає позитивні емоції та більшу довіру з боку користувача. Основні можливості:

1. **Рух та комунікація**: робот здатен самостійно наближатися до людини, подавати сигнали та звертатися з простими повідомленнями, щоб нагадати про дію. Це особливо важливо для людей, які швидко втрачають увагу або не реагують на візуальні підказки на екрані. Він також може використовувати звукові та світлові ефекти, щоб привертати увагу, адаптуючи інтенсивність залежно від часу доби.
2. **Зв'язок із користувачем**: система враховує ситуацію — час, поведінку користувача в минулому, особливості середовища — і підбирає оптимальний спосіб звернення. Наприклад, у вечірній час робот може говорити спокійніше, а вранці — більш енергійно. Крім того, він може виконувати короткі діалоги, які підтримують когнітивну активність користувача.
3. **Фізичні підтвердження дій**: через прості сенсори або взаємодію з предметами (наприклад, відкриття дверцят шафи чи рух дозатора) система фіксує, що дія була виконана. Це дозволяє уникнути помилкових повторень або збоїв у графіку нагадувань. Якщо дія не зафіксована — робот може повторно звернутися або надіслати сповіщення опікуну.
4. **Контроль для доглядальника**: спеціальний застосунок дозволяє задавати нові нагадування, отримувати сповіщення та відслідковувати загальну активність користувача. Він також дозволяє налаштовувати гнучкі сценарії реагування та синхронізувати інформацію з іншими членами родини або медичним персоналом. Опікун має змогу бачити графік, оцінювати виконання завдань і навіть ініціювати віддалену взаємодію через робота.

У процесі реалізації передбачено кілька сценаріїв використання: ранкове пробудження, нагадування про прийом ліків, запрошення на обід, рекомендація відпочинку. Робот не лише інформує, а й допомагає дійти до необхідного місця або ввімкнути відповідний пристрій, наприклад, чайник чи телевізор. За бажанням, система може супроводжувати аудіо-нагадуванням улюблену пісню чи звуки природи для створення комфортної атмосфери.

На етапі прототипування було враховано інтерфейсні аспекти: для людей з погіршеним зором розроблено збільшені шрифти, контрастні візуальні підказки, озвучення кнопок. Робот має базові функції автономної навігації, які дозволяють уникати перешкод та слідувати за користувачем у разі потреби.

Крім функцій нагадування, робот може використовуватись як освітній або розважальний пристрій. Він може транслювати аудіокниги, проводити нескладні вправи для тренування пам'яті, відтворювати улюблені відео або проводити короткі вікторини. Це робить його багатофункціональним помічником, який може позитивно впливати на емоційний стан користувача.

У майбутньому система може бути доукомплектована блоком машинного навчання, який дозволить адаптувати поведінку робота до індивідуальних звичок та ритму життя конкретної людини. Таким чином, нагадування не будуть стандартними, а навпаки — максимально персоналізованими. Наприклад, якщо користувач зазвичай ігнорує перше повідомлення — система навчиться повторювати його за певний час або в інший спосіб.

Також є потенціал для співпраці з медичними інформаційними системами, що дозволить інтегрувати дані про стан здоров'я користувача, рекомендовані препарати чи навіть медичні графіки. У такому випадку робот стане частиною цілісної системи моніторингу та супроводу літніх людей у домашніх умовах.

Рішення може бути масштабованим — як для використання вдома, так і для впровадження у спеціалізованих установах. Потенційна інтеграція з розумним будинком або хмарними сервісами дає змогу розширити функціональність і зробити систему більш адаптивною до потреб конкретного користувача. Передбачена також можливість розпізнавання емоцій (у майбутніх версіях) для ще більш точного реагування на стан людини.

Система має важливе значення з точки зору підтримки людей із деменцією, оскільки допомагає їм залишатися активними та незалежними. Водночас вона зменшує стрес для родичів або соціальних працівників, які опікуються такими людьми. Подальші дослідження можуть включати покращення поведінкових алгоритмів робота, розширення сенсорної бази та інтеграцію з іншими технологіями «розумного дому».

Одним із напрямів розвитку такої системи є її використання в мультикористувацькому середовищі. Наприклад, якщо в будинку проживають кілька осіб похилого віку, робот може ідентифікувати кожного користувача та персоналізувати взаємодію залежно від конкретних потреб, графіка або медичних показань. Це відкриває нові горизонти для колективного догляду та підтримки.

Висновок

Запропонована концепція та реалізований прототип демонструють перспективність використання роботизованих нагадувальних систем для покращення життя людей із когнітивними порушеннями. Вона має потенціал стати частиною сучасних медико-соціальних підходів до догляду та підтримки таких осіб.

Окрім функціонального застосування, система також виконує важливу роль у формуванні емоційного комфорту, підвищенні рівня самостійності та зниженні соціальної ізоляції користувачів. Залучення таких технологій може значно покращити якість життя не лише самих пацієнтів, а й їхніх родичів, створюючи нову парадигму у сфері турботи та цифрового догляду.

Загалом роботизована система-нагадування може бути важливим доповненням до існуючих моделей підтримки в соціальній сфері та охороні здоров'я. Її впровадження сприятиме модернізації підходів до догляду за людьми з когнітивними розладами та формуванню інклюзивнішого суспільства, де технології дійсно служать людині. Поширення подібних рішень сприятиме розвитку сфери доглядових технологій в Україні та інтеграції до європейського ринку інноваційних соціальних сервісів.

Впровадження таких рішень також може надихнути українських розробників працювати над соціально орієнтованими технологіями, створюючи продукти, що відповідають не лише вимогам ринку, а й реальним потребам людей похилого віку. Це, у свою чергу, сприятиме зростанню рівня довіри до технологій серед вразливих категорій населення та популяризації гуманного підходу до інженерії.

Список використаних джерел

1. Prince, M., Wimo, A., Guerchet, M., Ali, G. C., Wu, Y. T., & Prina, A. (2015). World Alzheimer Report 2015 - *The Global Prevalence of Dementia: An update since 2009*. Alzheimer's Disease International.
2. Broadbent, E., Stafford, R., & MacDonald, B. A. (2009). Acceptance of healthcare robots for the elderly. *International Journal of Social Robotics*, *1*(4), 319-330.

УДК 004.4

МАТЕМАТИЧНА МОДЕЛЬ ОЦІНКИ ПРОФЕСІЙНОСТІ ЕКСПЕРТА

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

І. Постановка проблеми

У сучасному інформаційному суспільстві експертна оцінка відіграє ключову роль у прийнятті рішень, зокрема у сферах контролю якості програмного забезпечення, відбору кадрів та проєктного менеджменту. Висновки експертів часто мають критичне значення для успішної реалізації технологічних рішень, однак відсутність чітко визначених критеріїв професійної компетентності експертів знижує довіру до таких оцінок. У більшості випадків оцінювання експертів здійснюється інтуїтивно або базується на суб'єктивних судженнях, що ускладнює верифікацію результатів та впровадження об'єктивних систем підтримки прийняття рішень. У зв'язку з цим виникає необхідність у формалізації процесу визначення рівня професійності експерта на основі кількісно вимірюваних показників. Особливо актуальним є використання відкритих джерел даних, таких як професійна мережа LinkedIn, що дає змогу об'єктивно отримувати інформацію про досвід, навички, активність та інші характеристики спеціалістів. Для впровадження подібного підходу необхідно розробити математичні моделі, які дозволяють визначити вагу кожного показника та забезпечити цілісну й обґрунтовану оцінку.

ІІ. Мета роботи

Метою дослідження є побудова формалізованої математичної моделі для оцінювання професійної компетентності експертів у галузі інформаційних технологій на основі відкритих даних з платформи LinkedIn. У межах роботи передбачено виокремлення найбільш інформативних кількісних показників, що характеризують експерта, а також визначення коефіцієнтів їхньої важливості за допомогою інтервального методу. Завданням дослідження є розробка як повної моделі, що враховує усі доступні показники, так і спрощеного варіанту моделі для випадків із обмеженим обсягом даних. Окрема увага приділяється перевірці точності та стабільності запропонованих моделей на основі реальної вибірки експертів із відомим рівнем професійної підготовки.

ІІІ. Основна частина

Актуальність проблеми оцінювання експертної компетентності підтверджується зростанням інтересу наукової спільноти до побудови об'єктивних систем аналізу кваліфікації спеціалістів. Сучасні професійні мережі, зокрема LinkedIn, містять значний обсяг структурованої інформації про кар'єрний шлях фахівців, їхній досвід, освіту, професійні досягнення та зв'язки в галузі. Ці дані становлять цінну емпіричну базу для розробки об'єктивних методів оцінювання експертизи. У публікації [1] було виділено низку кількісних показників із платформи LinkedIn, які можуть бути використані для побудови формалізованої моделі визначення рівня професійної компетентності експерта, зокрема такі як:

E – досвід роботи (стаж робота);
C – кількість компаній;
Ed – освіта;
P – кількість проектів;
S – кількість навичок експерта;
CL – кількість сертифікатів;
L – кількість мов;
RR –отримані рекомендації;
GR – надані рекомендації тощо.

Визначення відповідних вагових коефіцієнтів становить ключове методологічне завдання у розробці системи оцінювання експертів. Замість того, щоб покладатися на суб'єктивне призначення ваг (коефіцієнтів), було застосовано строгий математичний підхід, що базується на інтервальному численні [2-7], для отримання емпірично обґрунтованих коефіцієнтів.

Доцільно побудувати математичну модель, що включає всі дванадцять критеріїв експертності, для відображення повного спектру професійних кваліфікацій у галузі розробки програмного забезпечення. Оберемо лінійну структуру для моделі наступного виду:

$$R_i = k_1 \cdot E_i + k_2 \cdot C_i + k_3 \cdot Ed_i + k_4 \cdot P_i + k_5 \cdot S_i + k_6 \cdot CL_i + k_7 \cdot L_i + k_8 \cdot RR_i + \\ + k_9 \cdot GR_i + k_{10} \cdot Sk_i + k_{11} \cdot SkE_i + k_{12} \cdot Ec_i + k_{13} \quad (1)$$

де $k_j, j = 1,2,3,..,13$ – невідомі коефіцієнти, значення яких необхідно вирахувати на основі аналізу зібраних даних групи експертів, з якими дослідницька група має особистий та професійний досвід взаємодії; i – змінна величина в моделі, залежить від кількості експериментальних даних.

Відповідно, складемо систему лінійних алгебраїчних рівнянь вигляді (2):

$$\begin{cases} R_1 = k_1 \cdot E_1 + k_2 \cdot C_1 + k_3 \cdot Ed_1 + k_4 \cdot P_1 + k_5 \cdot S_1 + k_6 \cdot CL_1 + k_7 \cdot L_1 + k_8 \cdot RR_1 + \\ \quad + k_9 \cdot GR_1 + k_{10} \cdot Sk_1 + k_{11} \cdot SkE_1 + k_{12} \cdot Ec_1 + k_{13} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ R_i = k_1 \cdot E_i + k_2 \cdot C_i + k_3 \cdot Ed_i + k_4 \cdot P_i + k_5 \cdot S_i + k_6 \cdot CL_i + k_7 \cdot L_i + k_8 \cdot RR_i + \\ \quad + k_9 \cdot GR_i + k_{10} \cdot Sk_i + k_{11} \cdot SkE_i + k_{12} \cdot Ec_i + k_{13} \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ R_{24} = k_1 \cdot E_{24} + k_2 \cdot C_{24} + k_3 \cdot Ed_{24} + k_4 \cdot P_{24} + k_5 \cdot S_{24} + k_6 \cdot CL_{24} + k_7 \cdot L_{24} + k_8 \cdot RR_{24} + \\ \quad + k_9 \cdot GR_{24} + k_{10} \cdot Sk_{24} + k_{11} \cdot SkE_{24} + k_{12} \cdot Ec_{24} + k_{13} \end{cases} \quad (2)$$

Розв’язком системи є область коефіцієнтів моделі. Використавши метод найменших квадратів для знаходження оцінок коефіцієнтів моделі СЛАР (2), отримаємо таку модель:

$$R_i = 0.019 \cdot E_i + 0.057 \cdot C_i + 0.09 \cdot Ed_i + 0.163 \cdot P_i + 0.068 \cdot S_i + 0.984 \cdot CL_i + 0.147 \cdot L_i + \\ + 0.219 \cdot RR_i - 0.6 \cdot GR_i - 0.029 \cdot Sk_i + 0.066 \cdot SkE_i - 0.001 \cdot Ec_i + 3.289 \quad (3)$$

Нижче в таблиці № 1 представлені критерії та відповідні розраховані коефіцієнти.

Таблиця №1

Коефіцієнти критеріїв для комплексної моделі

Критерій	**Коефіцієнт**
Досвід (E)	0,0196575442390424
Компанії (C)	0,0569782832521855
Освіта (Ed)	0,0901619452246176
Проєкти (P)	0,162695920923237
Пропоновані сервіси (S)	0,068472027848529
Сертифікати (CL)	0,984301916333666
Мови (L)	0,147480317293241
Отримані рекомендації (RR)	0,219047874182111
Надані рекомендації (GR)	-0,599862676337443
Навики (Sk)	-0,0286572646337195
Підтверджені навики (SkE)	0,0656870040622226
Кількість підтверджень (Ec)	-0,0009915060875

Для розрахунку коефіцієнтів було використано дані вибірки експертів, з якими дослідницька група має особистий та професійний досвід взаємодії. Це дозволило забезпечити надійну основу для формування референтних оцінок компетентності, що стали еталоном у побудові моделей. Як зазначалося вище, рейтинг (Rating) відображає узагальнену індивідуальну оцінку професійної компетентності, надійності та кваліфікації експерта у сфері оцінювання програмного забезпечення. Побудована модель була перевірена на точність: результати оцінювання, згенеровані на її основі, не виходять за межі 15% похибки порівняно з реальними експертними оцінками. На Рисунку 2 подана діаграма, яка показує наскільки точними є результати, отримані через обрахунок за допомогою комплексної моделі в порівнянні до критерію Rating:

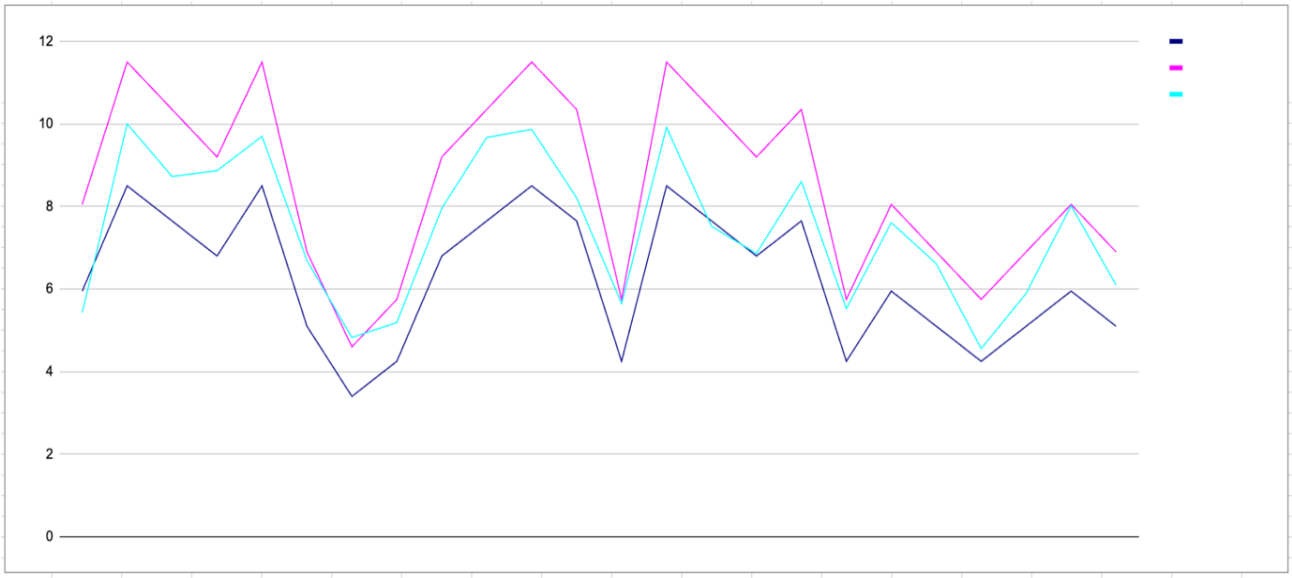


Рисунок 1 - Діаграма адекватності побудованої математиної моделі

Висновки

У даному дослідженні розроблено математичну модель оцінювання кваліфікації експертів з програмної інженерії. Було створено математичний апарат для об'єктивного перетворення професійних показників LinkedIn у комплексну оцінку експертизи, науково обґрунтовано вагові коефіцієнти для кожного критерію за допомогою інтервального числення на основі калібрувальної вибірки. Також формалізовано систему з 12 кількісних критеріїв, що охоплюють технічні компетенції та професійне визнання, та розроблено і протестовано математичну модель, яка демонструє високу точність в межах 15% допустимого відхилення. Запропоноване рішення має наукову новизну через емпіричний підхід до визначення вагових коефіцієнтів та практичну цінність для процесів підбору персоналу й формування проектних команд у галузі програмної інженерії.

Список використаних джерел

1. I.Spivak, S.Krepych, S.Stadnyk, R.Krepych “Identification the criterion for expert professional experience to determine his ‘quality’”, Комп’ютерні інформаційні технології: Матеріали зимової школи-семінару молодих вчених і студентів CIT’2024. – Тернопіль: ЗУНУ, 2024. с.58-60
2. I.Spivak, S.Krepych and S. Budenchuk, “Methods and Means of Expert Evaluation of Software Systems on the Basis of Interval Data Analysis”, 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering 2018, pp.164-167
3. I. Spivak, S.Krepych and R.Krepych, “Construction of the Criterion for the Agree of Expert Groups Estimates Based on Analysis of Interval Data”, 2018 International Scientific-Practical Conference Problems of Infocommunications Science and Technology, Kharkiv, pp.261-264
4. I.Spivak and S.Krepych. “Interval calculations: theory and practice”, Publisher: LAP LAMBERT Academic Publishing ISBN: 978-613-9-94967-0, 2019.
5. I.Spivak and S.Krepych. “Expert evaluation of software systems based on interval data analysis”, Publisher: LAP LAMBERT Academic Publishing ISBN: 978-613-9-96108-5, 2018.
6. I.Spivak, S.Krepych, R.Krepych and A.Bayurskii, “Construction of a Criterion for Assessing the Level of Objectivity of Experts Based on a Modified Interval Expert Appraisal Method”, 2019 IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 – Proceedings, Kyiv, pp.311-314
7. I.Spivak, S.Krepych, A.Bayurskii and S.Spivak, “Criterion for evaluation the level of experts competence during the evaluation of a software system based on the modified interval method of expert evaluation”, 2021 11th International Conference on Advanced Computer Information Technologies, ACIT 2021 – Proceedings, 2021, pp.582-586

УДК 004.932:004.5:794

АНАЛІЗ МЕТОДІВ ГЛИБОКОГО НАВЧАННЯ ДЛЯ РОЗПІЗНАВАННЯ ЖЕСТІВ У БЕЗКОНТАКТНОМУ КЕРУВАННІ ІГРОВИМИ ДОДАТКАМИ

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

I. Постановка проблеми

Безконтактне керування відеоіграми є актуальним напрямом у розробці інтерактивних додатків, зокрема в контексті віртуальної та доповненої реальності. Основна ідея полягає у заміні традиційних засобів керування (джойстиків, миші, клавіатури) системами, що реагують на жести рук або рухи тіла користувача. Однією з найефективніших технологій для реалізації таких систем є глибоке навчання, зокрема згорткові нейронні мережі (CNN), які дозволяють розпізнавати візуальні шаблони з високою точністю. Проблемою є вибір оптимальної архітектури моделі, забезпечення достатньої кількості навчальних даних та досягнення стабільної точності в реальних умовах.

II. Мета роботи

Метою роботи є аналіз ефективності застосування згорткових нейронних мереж для класифікації жестів, а також розробка повноцінного програмного модуля для розпізнавання команд у системі безконтактного керування відеоіграми.

III. Методика дослідження

Для аналізу було створено власний датасет жестів, що включає 5 класів: відкрита долоня, кулак, жест "peace", великий палець вгору та "okay". Збір даних здійснювався через вебкамеру з подальшою аугментацією.

Навчання здійснювалося за допомогою моделі CNN у середовищі TensorFlow/Keras. Архітектура включала два згорткових блоки з шарами MaxPooling та повнозв'язні шари, останній з яких — softmax для багатокласової класифікації. Модель навчалась упродовж 10 епох на зображеннях розміром 64×64 пікселі.

Для забезпечення коректного навчання застосовано генератор зображень ImageDataGenerator з розділенням на навчальну та валідаційну вибірки у співвідношенні 80/20. Навчання проводилось із використанням оптимізатора Adam та функції втрат categorical crossentropy.

IV. Результати навчання

Модель продемонструвала поступове і стабільне покращення показників класифікації протягом усього процесу навчання. З кожною епохою спостерігалося зростання точності як на тренувальній, так і на валідаційній вибірках, що свідчить про ефективне навчання мережі без суттєвих ознак перенавчання. Після завершення 10 епох було досягнуто таких результатів:

- Точність на тренувальній вибірці (training accuracy): 97%, що вказує на високу здатність моделі до відтворення шаблонів, присутніх у навчальних даних.
- Точність на валідаційній вибірці (validation accuracy): 85%, що підтверджує здатність моделі узагальнювати знання на нові, раніше не бачені зображення.

На рисунку 1 подано графік динаміки точності моделі протягом епох.

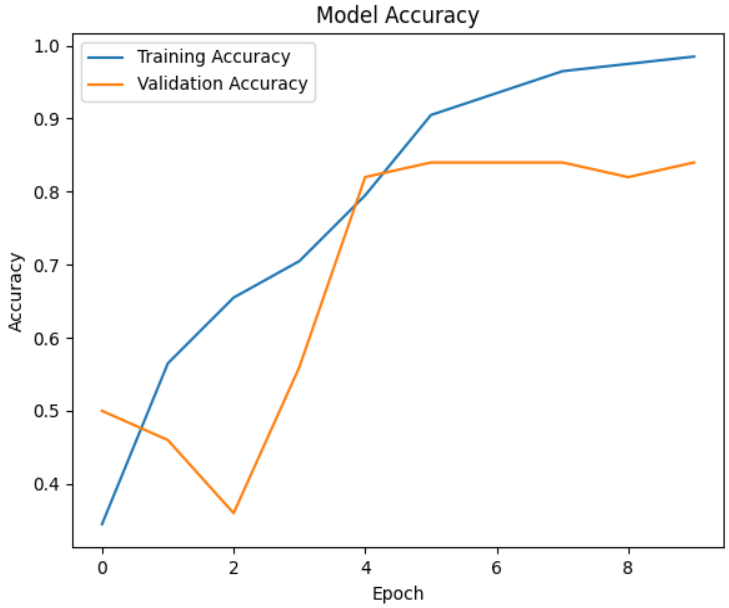


Рисунок 1 – Залежність точності моделі від кількості епох

З графіка видно, що тренувальна і валідаційна точність зростають синхронно, що підтверджує належний епох баланс між складністю архітектури та обсягом навчальних даних.

На рисунку 2 зображено зміну значень функції втрат (loss) на тренувальній та валідаційній вибірках упродовж 10 епох навчання.

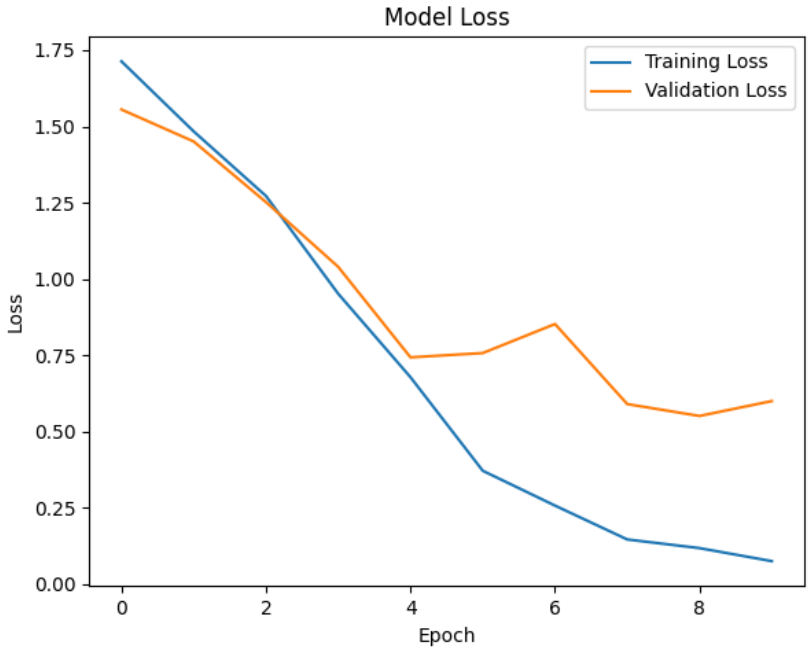


Рисунок 2 – Зміна значень функції втрат

V. Інтеграція з ігровим середовищем

Розроблена модель була збережена у форматі .h5 та інтегрована в систему керування на основі бібліотек OpenCV та pyautogui. Модель класифікує кадр з руки в реальному часі та викликає відповідну дію в грі (рух, стрибок, атака тощо).

У подальшій реалізації передбачено підключення моделі до Unity через WebSocket або локальний сервер Python для інтерактивного керування 3D-персонажем за допомогою жестів

VI. Інтеграція з Unity через WebSocket / REST API

Для реалізації повноцінної інтерактивної взаємодії між моделлю розпізнавання жестів та ігровим середовищем Unity було запропоновано використання WebSocket або REST API як методу передачі даних.

Система розпізнає жест у Python-середовищі, після чого за допомогою WebSocket клієнта або HTTP-запиту передає назву жесту до Unity, де обробляється і викликає відповідну дію у грі.

Такий підхід забезпечує низьку затримку, кросплатформенність і легку адаптацію до будь-якого типу ігрового рушія, зокрема для VR/AR додатків. Unity виступає як клієнт або сервер, що слухає події та виконує команди, наприклад: зміна анімації, переміщення персонажа або виконання дії.

Додатково можлива реалізація буфера кадрів для зменшення спрацьовувань у разі шуму або нестабільного відеопотоку.

У перспективі можливе використання локального Flask-сервера з Python, який буде одночасно обробляти відео з камери, передбачати жести та надсилати команди до Unity у форматі JSON.

Висновки

Застосування загорткових нейронних мереж дозволяє ефективно реалізувати систему розпізнавання жестів для відеоігор. Навіть базова архітектура CNN демонструє високу точність, простоту інтеграції та низьку затримку при класифікації. Результати навчання свідчать про перспективність цього підходу для масового впровадження в ігрову індустрію, освіту та реабілітаційні технології.

Список використаних джерел

1. TensorFlow Keras API documentation [Електронний ресурс] – Режим доступу :https://www.tensorflow.org
2. OpenCV Python Tutorials [Електронний ресурс] – Режим доступу: https://docs.opencv.org
3. MediaPipe Hands [Електронний ресурс] – Режим доступу: https://google.github.io/mediapipe
4. Zhan F. “Hand Gesture Recognition with Convolution Neural Networks”, 20th International Conference on Information Reuse and Integration for Data Science (IRI), 2019. pp.295-298
5. Gesture Recognition Datasets [Електронний ресурс] – Режим доступу: https://www.kaggle.com
6. Шпінталь М.Я., Красько М.Г. “Математичне та програмне забезпечення для безконтактного керування відеоіграми за допомогою комп’ютерного зору”, Комп’ютерні інформаційні технології: Матеріали зимової школи-семінару молодих вчених та студентів CIT’2024. Тернопіль. с.44-45
7. PyAutoGUI Documentation [Електронний ресурс] – Режим доступу: https://pyautogui.readthedocs.io

УДК 004.4

АРХІТЕКТУРНІ ТА ТЕХНОЛОГІЧНІ ПІДХОДИ ДО СТВОРЕННЯ КЛІЄНТ-СЕРВЕРНОЇ СИСТЕМИ УПРАВЛІННЯ ДАНИМИ ТУРИСТИЧНОГО БІЗНЕСУ

Папа О.А.[1)], **Привроцький Д.А.**[2)]
Західноукраїнський національний університет
[1)] д.філ.н., доцент;,[2)]бакалавр

І. Постановка проблеми

Сучасний туристичний бізнес функціонує в умовах жорсткої конкуренції та високої динаміки змін споживчого попиту. Це зумовлює необхідність оперативної обробки великих обсягів інформації, що включає дані про тури, локації, бронювання, платежі, відгуки, документацію, партнерські зв'язки тощо. Ручна обробка таких даних не лише уповільнює бізнес-процеси, але й підвищує ризик помилок, що може призвести до втрати клієнтів або доходів. У зв’язку з цим постає завдання розробки сучасної веб-орієнтованої клієнт-серверної системи управління даними туристичного оператора, яка б забезпечувала централізований доступ до інформації, автоматизацію рутинних процесів, підвищення ефективності роботи персоналу та покращення якості обслуговування клієнтів.

Ключовими вимогами до такої системи є:масштабованість (можливість розширення функціональності без радикальних змін структури),безпека даних (контроль доступу, шифрування, автентифікація),продуктивність (швидке завантаження сторінок, оптимізована обробка запитів),кросплатформність і адаптивність інтерфейсу (зручна робота як з ПК, так і з мобільних пристроїв).Забезпечення цих характеристик можливе лише за умови застосування принципів інженерії програмного забезпечення, що включає всебічний аналіз вимог, моделювання архітектури, вибір відповідного технологічного стеку, впровадження засобів тестування та документації.

ІІ. Мета роботи

Метою даної роботи є дослідження та обґрунтування архітектурних і технологічних рішень, прийнятих при розробці веб-орієнтованої клієнт-серверної системи для управління даними туристичного оператора. Акцент зроблено на виборі сучасного технологічного стеку, який забезпечує надійність, масштабованість, високу продуктивність, безпеку та зручність для кінцевих користувачів.

ІІІ. Основна частина

Для системи було обрано клієнт-серверну архітектуру, в якій фронтенд і бекенд чітко розділені.Клієнтська частина взаємодіє з сервером через REST API, отримуючи доступ до бази даних і бізнес-логіки. Така модель забезпечує масштабованість, спрощує розгортання та підтримку, а також дозволяє незалежно оновлювати окремі компоненти системи.Було обрано наступний стек:

- необхідності писати окремий CSS-код, що значно пришвидшує розробку та спрощує підтримку Next.js та TypeScript–розробка інтерфейсу користувача та логіки взаємодії з бекендом реалізована за допомогою Next.js, що є фреймворком для React із підтримкою серверного рендерингу та статичної генерації сторінок. Завдяки використанню TypeScript досягається типобезпечність, зменшується кількість помилок на етапі розробки, а також підвищується якість коду за рахунок автодоповнення, перевірки типів та кращої документації.
- Supabase –використовується як бекенд-інфраструктура. Це відкрите рішення, побудоване на основі PostgreSQL, яке забезпечує набір функцій, необхідних для сучасного веб-застосунку: реєстрацію та автентифікацію користувачів, керування базою даних у режимі реального часу, автоматичне створення RESTful API, а також можливість запуску серверної логіки за допомогою edge-функцій. Supabase також підтримує систему підписок, що дозволяє оновлювати дані на клієнті в реальному часі без необхідності ручного оновлення сторінки.
- Tailwind CSS – для створення адаптивного, сучасного та зручного інтерфейсу користувача було обрано утилітарний CSS-фреймворк. Tailwind дозволяє швидко створювати кастомізовані UI-компоненти без стилів. Комбінація з класами Tailwind дозволяє уникати дублювання стилів і забезпечує консистентність у візуальному оформленні.

Supabase підтримує JWT-аутентифікацію, контроль доступу до таблиць на рівні записів (Row-Level Security, RLS), що відповідає сучасним стандартам захисту персональних даних та забезпечує

гнучке управління правами доступу для різних типів користувачів. Всі з'єднання з API реалізовані через захищений протокол HTTPS, що гарантує конфіденційність переданих даних.

Використання хмарної архітектури забезпечує високу доступність і відмовостійкість системи. Завдяки можливості горизонтального масштабування ресурси можуть динамічно збільшуватись відповідно до навантаження. Також інтеграція з CDN (Content Delivery Network) дозволяє зменшити час завантаження сторінок і покращити користувацький досвід. Підтримка серверного рендерингу (SSR) у Next.js додатково сприяє покращенню SEO-оптимізації та прискорює початкове відображення вмісту на сторінках.

Обраний стек технологій дозволяє ефективно реалізувати функціональність системи управління даними для туристичного бізнесу, забезпечуючи високу продуктивність, безпеку, масштабованість та зручність як для кінцевих користувачів, так і для адміністративного персоналу.

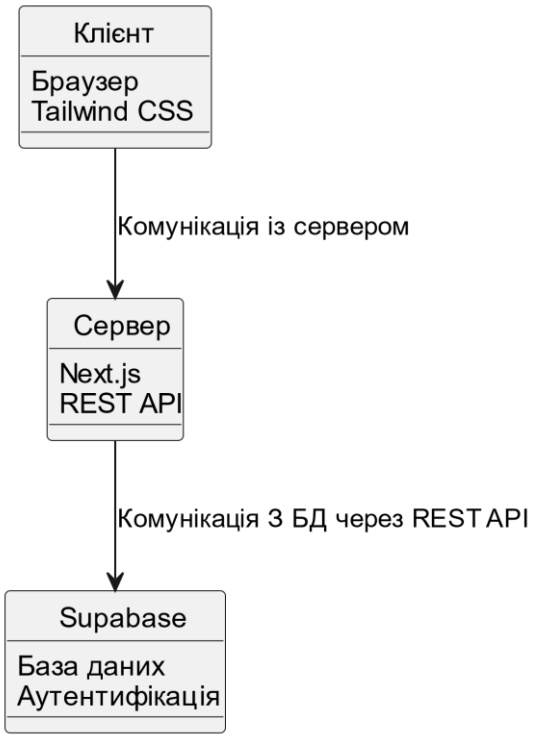


Рисунок 1 – Вигляд клієнт-серверної моделі

Висновок

Розробка клієнт-серверної веб-системи для управління даними туристичного бізнесу потребує комплексного підходу до вибору архітектури та технологічного стеку. У межах цієї роботи було обґрунтовано застосування Next.js, Supabase, Tailwind CSS і TypeScript як взаємодоповнюючих рішень, що забезпечують гнучкість, безпеку, швидкість розробки та зручність подальшої підтримки. Обраний стек технологій і архітектурні підходи повністю відповідають вимогам до сучасних веб-рішень, орієнтованих на автоматизацію бізнес-процесів у сфері туризму. Результатом є система, що легко масштабовується, безпечно обробляє дані користувачів, має високу продуктивність і забезпечує позитивний користувацький досвід. Такий підхід демонструє ефективність поєднання відкритих інструментів і сучасних принципів розробки для вирішення прикладних задач у галузі.

Список використаних джерел

1. Supermicro. What Is Client-Server Architecture? [Електронний ресурс] – Режим доступу:https://www.supermicro.com/en/glossary/client-server-architecture
2. Supabase. Use Supabase with Next.js [Електронний ресурс] – Режим доступу: https://supabase.com/docs/guides/getting-started/quickstarts/nextjs
3. S.Krepych, I.Spivak. “Algorithm of automatic generation of hotel descriptions using templates based on Markov chains”, International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). 2018. pp.257-260
4. Restack. Supabase Next.js Starter Guide [Електронний ресурс] – Режим доступу: https://www.restack.io/docs/supabase-knowledge-supabase-nextjs-starter
5. Tailwind CSS. Rapidly build modern websites without ever leaving your HTML [Електронний ресурс] – Режим доступу: https://tailwindcss.com

УДК 004.4

ПРОГРАМНИЙ КОМПЛЕКС ПЕРЕВІРКИ ПРАКТИЧНИХ РОБІТ СТУДЕНТІВ НА УНІКАЛЬНІСТЬ

Глухов О.В.[1)], Крепич С.Я.[2)], Співак І.Я.[3)]
[1-3)]Західноукраїнський національний університет
[1)]магістр; [2-3)] к.т.н., доцент

Постановка задачі

У роботі розглянуто побудову системи виявлення плагіату в академічних текстах на прикладі студентських робіт. Пропонована система використовує метод n-грам, що дозволяє аналізувати схожість текстів шляхом розбиття їх на послідовності з n слів або символів. Рішення орієнтоване на автоматичну перевірку PDF-документів і забезпечує інтеграцію з внутрішньою базою робіт навчального закладу, включно з архівами минулих років.

Мета дослідження

Метою дослідження є розробка програмної системи для виявлення плагіату в PDF-документах, яка здатна здійснювати порівняння текстів на основі методу n-грам як між поточними роботами, так і з роботами, збереженими у внутрішній базі навчального закладу за попередні роки.

Програмна реалізація

Основні сценарії функціонування програми:

1. процес вилучення та попередньої обробки тексту з PDF-файлів із попередньо обробкою (очищення, токенізація);
2. процес вибору кількості n-грам для подальшої передачі на метод розрахунку схожості текстів (чим менше число n обране (мінімально 2), тим точніше буде реалізовуватись перевірка);
3. процес візуалізації результатів перевірки з підсвічуванням ідентичних фрагментів для 2 файлів;
4. процес табличної візуалізації перевірки набору файлів один з одним.

На рисунку 1 зображено головне (початкове) вікно користувацького інтерфейсу програми для перевірки текстів на плагіат. Інтерфейс містить основні елементи керування: кнопки для завантаження PDF-документів, вибору папки для порівняння, запуску аналізу, а також поле для перегляду результатів. Дизайн орієнтований на простоту використання, щоб забезпечити швидкий доступ до основних функцій системи.

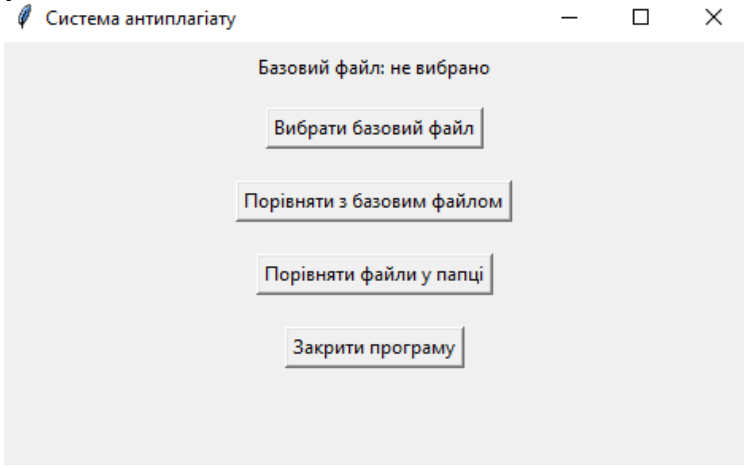


Рисунок 1 – Початкове вікно програми

На рисунку 2 зображено інтерфейс локального вибору файлів або директорій для перевірки на плагіат. Користувач може вручну вибирати один або кілька PDF-документів з локального сховища або цілу директорію для подальшого аналізу. Такий вибір дозволяє зручніше працювати з великими обсягами даних, коли потрібно перевірити декілька документів або цілу колекцію файлів на наявність плагіату. Після вибору користувач може ініціювати перевірку, і система починає обробку вибраних документів.

Інтерфейс розроблений з урахуванням зручності користувача, забезпечуючи простий і зрозумілий процес вибору файлів чи папок.

Кнопки управління дозволяють здійснювати вибір файлів і запускати процес перевірки на плагіат, що підвищує ефективність роботи. Локальний вибір є важливим етапом, оскільки дає користувачу можливість обрати саме ті документи, які потребують перевірки, і допомагає організувати роботу з файлами з урахуванням індивідуальних потреб користувача.

Рисунок 3 демонструє інтерфейс, який відображає структуру результатів перевірки текстів на плагіат після завершення процесу порівняння. В інтерфейсі відображається список документів, які підлягали перевірці, із зазначенням рівня схожості кожного документа з іншими роботами, що зберігаються в базі даних. Система підсвічує фрагменти тексту, які містять потенційні випадки плагіату, що дозволяє користувачеві наочно оцінити схожість і визначити місця, де може бути виявлений плагіат. Кожен фрагмент, що підлягає перевірці, містить інформацію про відсоток схожості, що дозволяє точно оцінити ступінь ймовірного плагіату. Інтерфейс дозволяє зручно переглядати результати перевірки, надаючи користувачеві можливість детально вивчити кожен потенційно підозрілий фрагмент. Також є можливість порівнювати отримані результати з іншими роботами в базі даних, що полегшує процес виявлення можливих випадків академічного шахрайства. Використання цього інтерфейсу забезпечує зручність роботи та максимальну ефективність виявлення плагіату, що є важливим інструментом для підтримки академічної доброчесності.

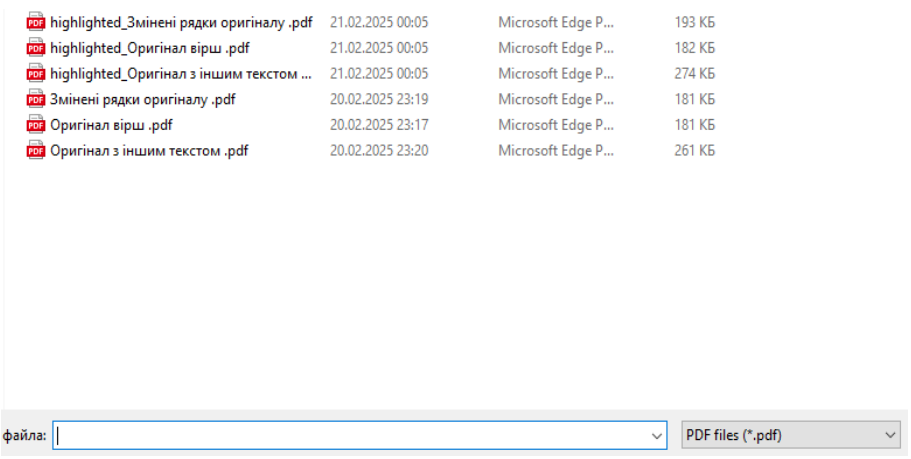


Рисунок 2 – Локальний вибір

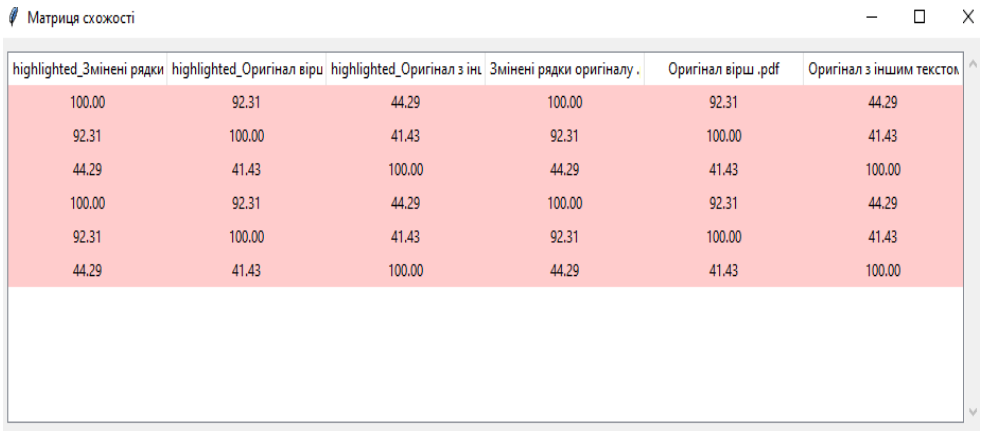


Рисунок 3 – Структура результатів перевірки

Висновок

У роботі було розроблено програмну систему для виявлення плагіату в академічних текстах, яка базується на методі n-грам. Система дозволяє автоматично аналізувати PDF-документи, виділяти підозрілі фрагменти та порівнювати їх із внутрішньою базою робіт навчального закладу, включно з архівами попередніх років. Запропоноване рішення сприяє підвищенню академічної доброчесності та автоматизує процес перевірки на плагіат.

Список використаних джерел

1. Крепич С.Я., Глухов О.В., Співак І.Я. Підхід до аналізу унікальності практичних робіт студентів. *Комп'ютерні інформаційні технології: Матеріали школи-семінару молодих вчених і студентів CIT'2024*. – Тернопіль: ЗУНУ, 2024. С.26-27

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 V Всеукраїнської школи-семінару молодих вчених і студентів ACIT'2015*. – Тернопіль: ТНЕУ, 2015. – С. 23-25.

3. A.Bayurskii, S.Krepych. Intelligent System Analyzing Quality of Land Plots. *International Conference "Advanced Computer Information Technologies" ACIT-2018*. pp.166-169

3. Крепич С.Я., Глухов О.В., Співак І.Я. Дослідження ефективності застосування методу перевірки студентських робіт на унікальність. *Комп'ютерні інформаційні технології: Матеріали зимової школи-семінару молодих вчених і студентів CIT'2024*. – Тернопіль: ЗУНУ, 2024. С.26-27

УДК 004.4

ПОДОЛАННЯ НЕЕФЕКТИВНОСТІ ДОКУМЕНТООБІГУ ШЛЯХОМ АНАЛІЗУ КОРИСТУВАЦЬКОГО ДОСВІДУ

Бас В.В.
Західноукраїнський національний університет
бакалавр

I. Постановка проблеми

У багатьох підприємствах, що здійснюють зовнішньоекономічну діяльність, процес документообігу залишається фрагментованим, частково ручним та схильним до помилок. Особливо це стосується оформлення логістичних декларацій, де кожна неточність може призвести до затримок, фінансових втрат і ускладнень у співпраці з міжнародними партнерами.

Аналіз користувацького досвіду (UX) є ефективним інструментом у виявленні «вузьких місць» цифрової взаємодії з такими системами. Саме через глибоке вивчення сценаріїв використання, потреб і поведінки користувачів можна розробити рішення, яке не лише автоматизує документообіг, але й зробить його дійсно зручним і ефективним. [2][5]

II. Мета роботи

Метою цієї роботи є подолання неефективності логістичного документообігу шляхом створення веб-орієнтованого застосунку, в основі якого лежить системний UX-аналіз. Кінцевою метою є зменшення кількості помилок, прискорення процесів подання декларацій та підвищення задоволеності користувачів.

III. Основна частина

У межах роботи було виконано:дослідження існуючих підходів до подання логістичних декларацій;аналіз сценаріїв користувача та побудова userflow;проєктування структури інтерфейсу та створення макетів у Figma;розробка фронтенд-рішення на основі отриманих UX-висновків;тестування користувацького шляху та оцінка зручності інтерфейсу.

Аналіз користувацького досвіду

У результаті дослідження були виявлені основні проблеми, з якими стикаються користувачі:надмірна кількість полів і складна структура форм[6];відсутність зручної навігації та попередніх шаблоні[4];труднощі з редагуванням та перевіркою даних;неінтуїтивне візуальне оформлення[2].

На основі цих висновків була побудована діаграма користувацького потоку (див.рис.1), яка охоплює процес автентифікації, створення, редагування, збереження чернеток і подання декларацій.

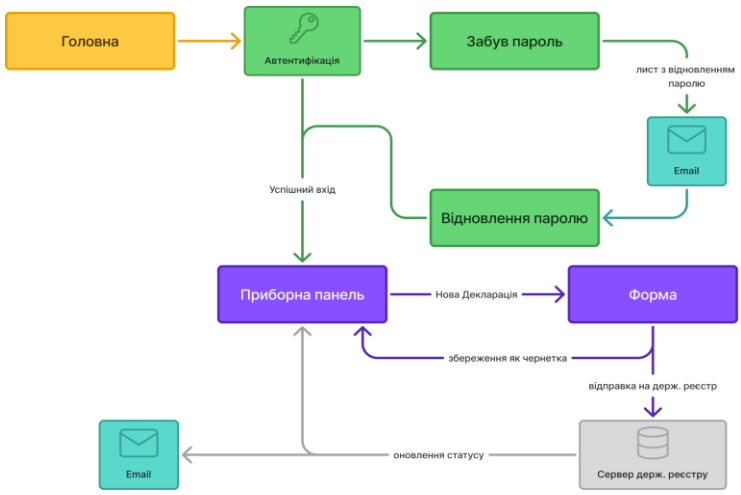


Рисунок1 - Діаграма користувацького потоку від автентифікації до отримання результату подання декларації

Ця діаграма зображає основний потік користувача зі створенням та заповненням форми з успішним створенням декларації, також дає змогу побачити побічні шляхи користувача, такі як збереження як чернетка та шлях відновлення пароля.

Розробка інтерфейсу

Основний акцент було зроблено на адаптивності та зручності:

Інтерактивна таблиця декларацій (див.рис.2): дозволяє швидко орієнтуватись у поточному статусі документів, фільтрувати та сортувати дані, а також завантажувати друковані версії[5].

Форма подання декларації: побудована на принципах модульності та гнучкості, з автозаповненням, валідацією та підказками[4].

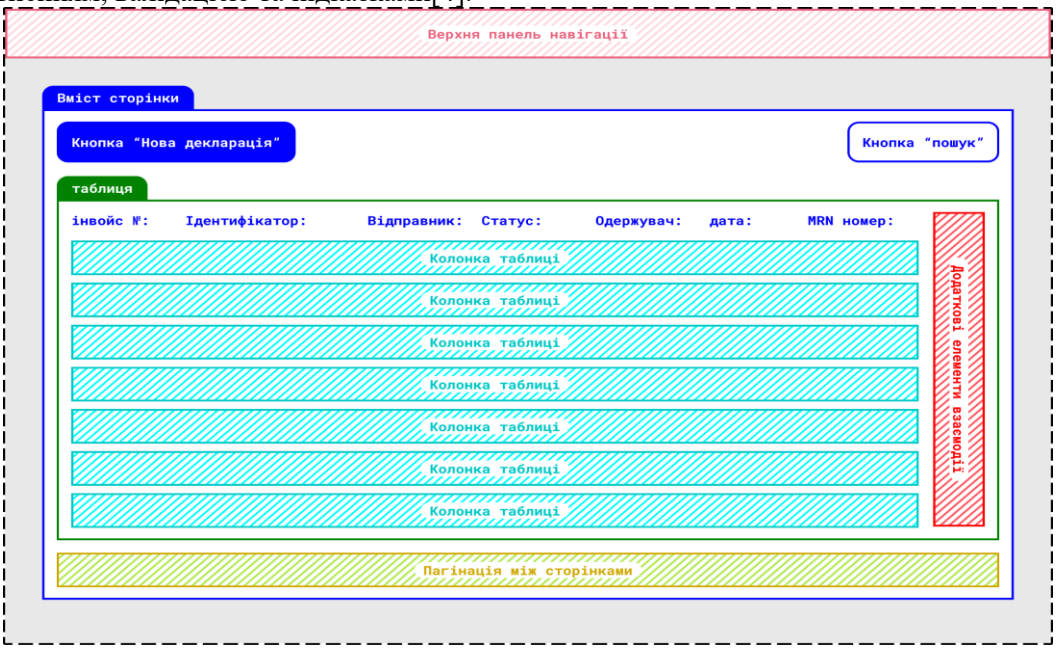


Рисунок 2 - Макет таблиці створено в figma

Динамічні елементи інтерфейсу: реалізовано розгортання секцій форми залежно від введених даних (наприклад, секція «Посередник»). Схема Форм для заповнення (див.рис.3): внутрішня форма таблиці була розбита на кілька секцій для зручного візуального групування елементів і відповідних полей для воду тексту чи вибору даних.

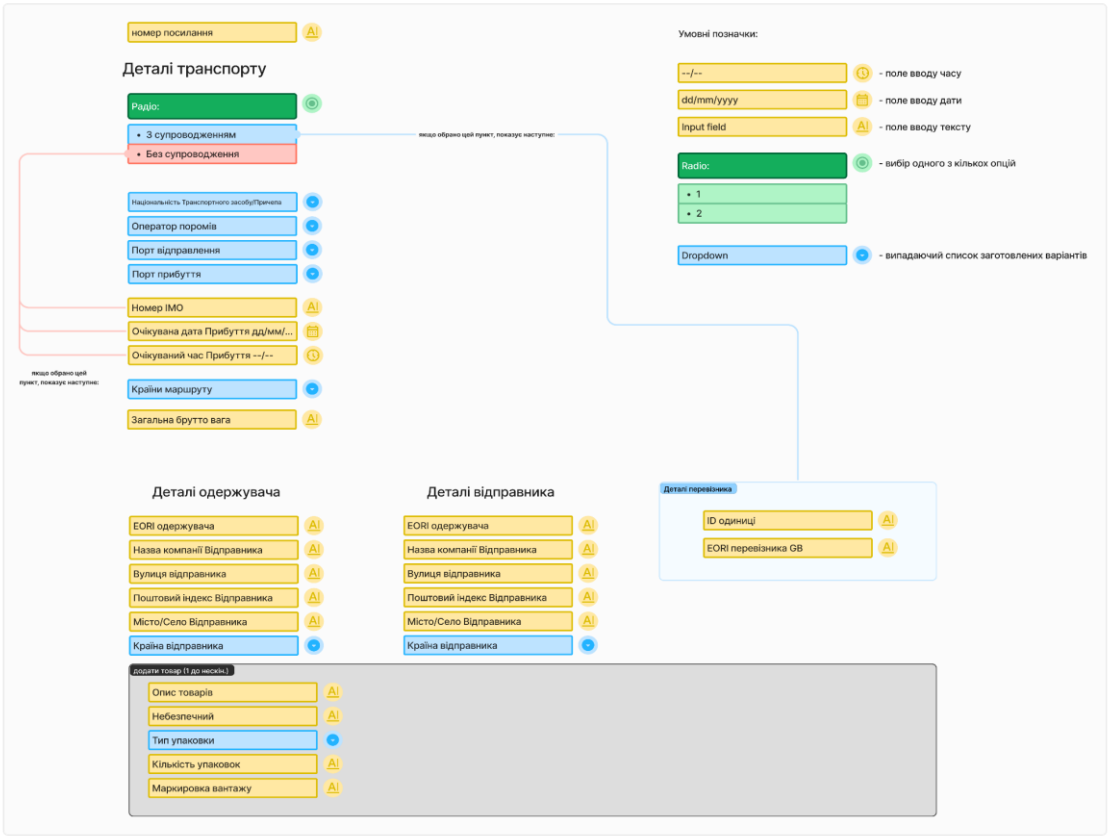


Рисунок 3 - Схема структури форми подання декларації

Технічна реалізація

Фронтенд частину веб-додатку реалізовано із застосуванням сучасного стеку веб-технологій: HTML5 — для семантичної розмітки сторінок, CSS3 — для стилізації інтерфейсів із використанням адаптивного підходу (Flexbox і Grid), а також TypeScript — для забезпечення типобезпеки під час розробки. Основою для побудови інтерфейсу стала бібліотека React, що дозволила організувати компоненти у гнучку та повторно використовувану структуру.

Розробка інтерфейсу здійснювалася на основі попередньо спроєктованої дизайн-системи у середовищі Figma. У процесі створення дизайн-системи було використано змінні (FigmaVariables), типові кольорові палітри, стандартизовану типографіку та набір UI-компонентів, зібраних за принципом AtomicDesign. Експорт стилів та змінних для розробників здійснювався через інструмент DevMode, що дозволяло швидко інтегрувати дизайн у код без втрати відповідності до макетів.

Для тестування збереження та обробки даних декларацій використано хмарну NoSQL-базу даних MongoDBAtlas. У межах практичного етапу було створено базу даних declarationsDB, яка забезпечувала виконання базових CRUD-операцій (створення, читання, оновлення та видалення записів).

Таким чином, технічнареалізаціябазуєтьсянапоєднаннізручногокористувацькогодосвіду, відповідності до сучасних принципів UI/UX-дизайну та застосування ефективних інструментів веб-розробки, що забезпечує як естетичну привабливість, так і високу функціональність продукту.

Завдяки такому підходу, система не лише відповідає очікуванням користувачів, а й є технологічно стійкою, масштабованою та готовою до інтеграції з іншими рішеннями.

Висновки

UX-орієнтований підхід дозволив не лише створити сучасний веб-додаток, але й вирішити конкретну прикладну проблему — неефективність логістичного документообігу. Завдяки глибокому розумінню потреб користувачів, було розроблено рішення, що поєднує адаптивний інтерфейс, інтуїтивно зрозумілу структуру форм і потужні механізми автоматизованої обробки даних. Це дозволило суттєво зменшити навантаження на персонал, мінімізувати ризик помилок під час заповнення документів та підвищити загальну точність переданих даних.

Користувачі отримали можливість швидше й ефективніше виконувати свої завдання, що призвело до помітного зростання продуктивності. Окрім того, покращення якості взаємодії із системою позитивно вплинуло на загальний рівень обслуговування клієнтів і внутрішніх процесів.

Цей досвід наочно демонструє, що глибокий аналіз користувацького досвіду є критично важливим етапом у розробці ефективних цифрових рішень. Він забезпечує не лише досягнення поточних бізнес-цілей, а й створює підґрунтя для довгострокового розвитку, адаптації до нових викликів і масштабування системи відповідно до зростаючих потреб логістичних компаній.

Список використаних джерел

1. Shipmate UX CaseStudy: Redesigning Logistics for Efficiency [Електронний ресурс] – Режим доступу: https://medium.com/design-bootcamp/shipmate-ux-case-study-redesigning-logistics-for-efficiency-13e0f73e9442
2. UX/UI Design for Logistics Management Software [Електронний ресурс] – Режим доступу: https://turumburum.com/cases/logistic
3. How to Develop a Document Management System for Logistics [Електронний ресурс] – Режим доступу: https://acropolium.com/blog/how-to-develop-a-document-management-system-for-logistics
4. UX Design for Custom Transport Management Software: Case Study [Електронний ресурс] – Режим доступу: https://brightinventions.pl/blog/ux-design-transport-management-software-case-study
5. UX/UI Design for Logistics [Електронний ресурс] – Режим доступу: https://cieden.com/ux-ui-design-for-logistics
6. UX Case Study for Logistics App [Електронний ресурс] – Режим доступу: https://pranalishinde.medium.com/ux-case-study-for-logistics-app-463b5da5c8b8
7. Крепич С.Я. Моделювання та забезпечення функціональної придатності статичних систем методами аналізу інтервальних даних. / С.Я. Крепич // дис. канд. техн. наук, НУ «Львівська політехніка», Львів, 2016. – 166с.
8. User Experience and Logistics [Електронний ресурс] – Режим доступу: https://keremcandemirtas.medium.com/user-experience-and-logistics-1c772b3c35fc
9. An Insight In to The Applications Of UX Design In Logistics Processes [Електронний ресурс] – Режим доступу: https://medium.com/%40uaxe/an-insight-into-the-applications-of-ux-design-in-logistics-processes-c30a68c670a8
10. Game-Changing Document Management Practices for Logistics Professionals [Електронний ресурс] – Режим доступу: https://kanboapp.com/en/documents-and-paperless/7-game-changing-document-management-practices-for-logistics-professionals
11. Document Management in Shipping and Logistics [Електронний ресурс] – Режим доступу: https://www.pairsoft.com/blog/leveraging-ap-automation-for-effective-document-management-in-shipping-and-logistics
12. A. Hendrian, M.L. Hamzah, Zarnelly, Anofrizen, T.K.Ansyar. “Evaluation of User Experience in Mobile Applications Using User Experience Questionnaire and User Centered Design Methods”, International Conference on Circuit, Systems and Communication (ICCSC), 2024. pp.1-6

УДК 004.4

РОЗРОБКА МОБІЛЬНОГО ДОДАТКА ДЛЯ СТВОРЕННЯ ТА КЕРУВАННЯ НАВЧАЛЬНИМИ РОЗКЛАДАМИ

Фанта В.В.
Західноукраїнський національний університет
бакалавр

I. Постановка проблеми

У сучасних навчальних закладах ефективна організація розкладу занять є критично важливою складовою навчального процесу. Через велику кількість викладачів, предметів і навчальних груп складність створення розкладу значно зростає. Ручне складання розкладу призводить до численних помилок, дублювання занять або перетинів у навантаженні викладачів. Щоб вирішити ці проблеми, було розроблено мобільний додаток для Android, який дозволяє адмініструвати навчальні розклади з урахуванням усіх обмежень і перевірок [1].

II. Мета роботи

Метою дипломного проєкту є створення повноцінного мобільного застосунку, який надає користувачу інтуїтивно зрозумілий інтерфейс для керування викладачами, предметами, часом занять та побудовою щотижневого розкладу для декількох навчальних груп, а також реалізація зручного використання розкладів та отримання з них необхідної інформації.

III. Основна частина

Для реалізації застосунку використано мову програмування Java, фреймворк Android SDK, а також локальну базу даних SQLite для зберігання всієї інформації [1,2]. Розробка здійснювалася у середовищі AndroidStudio. Для обміну даними між компонентами додатка використовуються Intent, Bundle, а також збереження стану через базу.Застосунок реалізовано на основі багаторівневої архітектури, де логіка взаємодії з даними повністю відокремлена від інтерфейсу користувача. Це забезпечує можливість масштабування та розширення функціоналу без порушення основної структури. На нижньому рівні знаходиться локальна база даних SQLite, у якій зберігаються таблиці: викладачі, предмети, часи занять, розклад для кожної групи. Всі запити до бази реалізовані в окремому класі DBHelper, що містить методи для додавання, оновлення, видалення та отримання даних. Це дозволяє легко змінювати схему бази або переносити логіку до зовнішнього API в майбутньому.

На рівні взаємодії з користувачем використано компоненти Activity, RecyclerView, CardView, TableLayout, ScrollView, PopupMenu, AlertDialog та ImageButton. Всі списки викладачів, предметів та груп відображаються за допомогою RecyclerView, що дозволяє динамічно оновлювати контент без перезавантаження сторінки. Елементи розкладу створюються як таблиці з можливістю взаємодії з кожною клітинкою, що дозволяє зручно змінювати дані без виходу на окрему сторінку редагування.

Керівник навчального процесу створює розклад для груп на семестр, зберігає дані в локальній базі та синхронізує їх з іншими пристроями за допомогою функціоналу програми.Викладач або студент може переглядати свій персональний розклад, який за необхідності можна дублювати та редагувати, додавати нові дані та увімкнути сповіщення для інформування про початок занять в групі.

Основні функції застосунку:

- Додавання та редагування базових даних: користувач може додавати викладачів, навчальні предмети та часи проведення занять. Кожен із цих елементів зберігається у локальній базі даних (SQLite), а для кожного типу даних створено окрему сторінку з можливістю перегляду, додавання та видалення записів.
- Формування розкладу: у додатку реалізована окрема сторінка для створення розкладу занять для кожної окремої групи. Користувач обирає день тижня, час, предмет та викладача для кожної пари. При цьому система автоматично перевіряє, чи не зайнятий обраний викладач у цей час в іншій групі, і в разі конфлікту підсвічує його червоним кольором. Таким чином забезпечується уникнення помилок при складанні розкладу.
- Можливість редагування існуючого розкладу: реалізовано режим редагування (EditMode), який активує можливість змінювати дані у вже сформованій таблиці.

- Push-сповіщення: у додатку реалізована система сповіщень, яка надсилає повідомлення про зміну розкладу або видалення занять.
- Можливість переглядати особистий розклад викладачів на окремій сторінці. Система шукає заняття підписані обраним викладачем, групує їх та створює розклад на їхній основі окремий розклад, за допомогою якого можна дізнатись яке заняття у певного викладача в той чи інший час (див.рис. 1).
- Інтерактивність: у таблицях розкладу можна натискати на будь-яку комірку, щоб змінити значення через спливаюче меню. Також реалізовано динамічну перевірку доступності викладача у заданий час на основі всіх наявних розкладів [3].

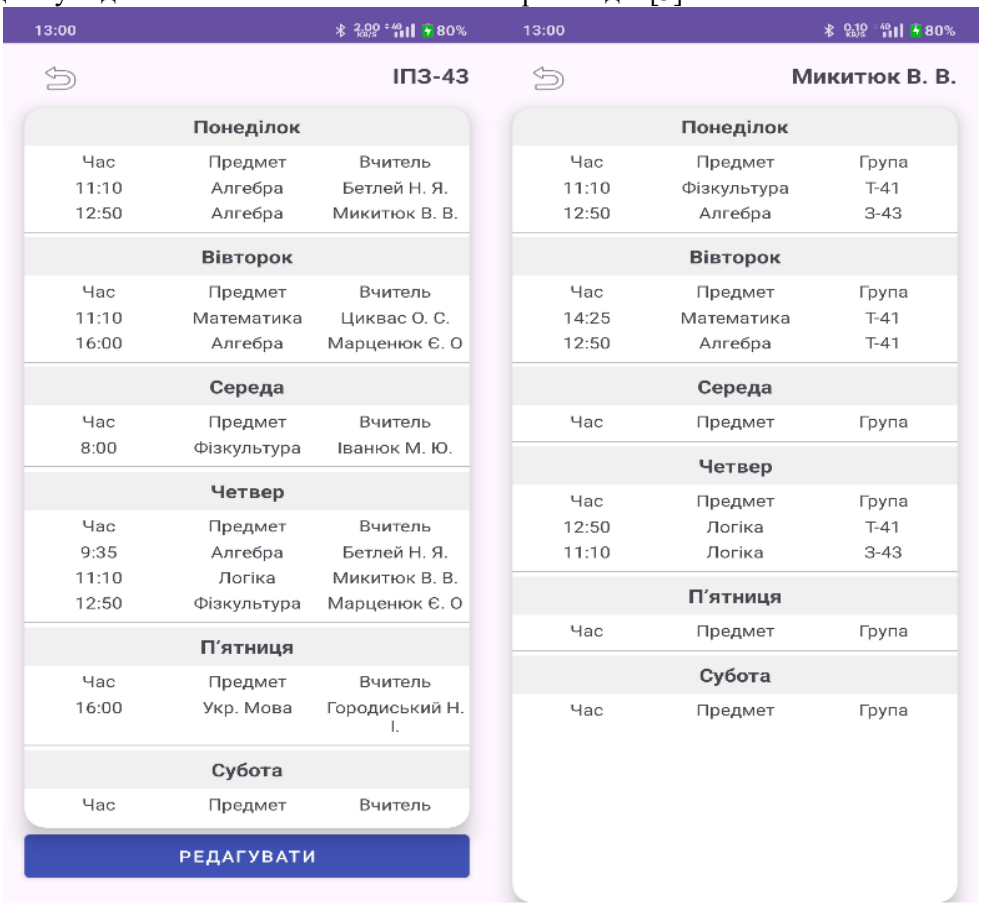


Рисунок 1 – Порівняння розкладу групи та викладача

Основними перевагами реалізованого рішення є офлайн-робота, швидкодія, гнучкість тапростота у користуванні. Додаток повністю працює без інтернет-з'єднання, що є важливим для навчальних закладів із нестабільним доступом до мережі,використання локальної бази даних дозволяє працювати із записами миттєво навіть на пристроях середньої продуктивності [2]. Інтерфейс розроблено таким чином, щоб ним міг скористатися навіть користувач без спеціальної комп'ютерної підготовки, проєкт можна легко використовувати до потреб різних закладів освіти, включаючи школи, університети, курси тощо, або адаптувати його для певних специфічних варіантів використання.

Висновок

Створений мобільний застосунок є зручним і потужним інструментом для адміністрації та використання розкладу занять у навчальному закладі. Його гнучкість, інтуїтивність, функціональність і можливість редагування безпосередньо з телефону дозволяє ефективно організувати навчальний процес, зменшити навантаження на адміністративний персонал та знизити ймовірність помилок у плануванні. У перспективі система може бути легко інтегрована з хмарними сервісами, веб-версією або синхронізована з електронними журналами.

Список використаних джерел

1. Meier, R. *Professional Android*. Wiley, 2018. — 816с.
2. Burnette, E. *Hello, Android: Introducing Google's Mobile Development Platform*. Pragmatic Bookshelf, 2010. — 284с.
3. Nielsen, J. *Usability Engineering*. Morgan Kaufmann, 1994. — 362с.

УДК 004.4

РОЗРОБКА ПРОТОТИПУ СИСТЕМИ ПРОГНОЗУВАННЯ ПОПИТУ З ВИКОРИСТАННЯМ ЧАСОВИХ РЯДІВ ТА API

Озійчук В.О.[1)], Крепич С.Я.[2)], Крепич Р.В.[3)]
[1-2)] Західноукраїнський національний університет;
[3)] ВСП Кам'янець-Подільський фаховий коледж НРЗВО «Кам'янець-Подільський державний інститут»
[1)] бакалавр; [2)]к.т.н., доцент; [3)]викладач

Постановка проблеми

В умовах сучасного динамічного ринку, що характеризується швидкими змінами споживчих преференцій та зростаючою глобальною конкуренцією, здатність підприємств адекватно реагувати на коливання попиту набуває критичного значення. Точне прогнозування попиту на продукцію чи послуги перетворюється з бажаної аналітичної функції на ключовий фактор забезпечення операційної ефективності та стратегічної стійкості бізнесу. Неточність прогнозів неминуче призводить до цілої низки негативних наслідків: від накопичення надлишкових та неліквідних складських запасів, що заморожують обігові кошти, до дефіциту ходових позицій, який спричиняє втрачені можливості продажу, зниження лояльності клієнтів та, як підсумок, зменшення загальної рентабельності діяльності [1].

Традиційні підходи до прогнозування, часто засновані на спрощених статистичних методах або експертних оцінках [2-3], все частіше виявляються недостатньо гнучкими та точними. Вони погано справляються з обробкою великих обсягів різнорідних даних ("BigData"), які генеруються сучасними бізнес-процесами, та не завжди здатні адекватно враховувати складні нелінійні залежності, множинні сезонні коливання (тижневі, річні), а також вплив зовнішніх факторів (маркетингові акції, економічні зміни), що є характерними для сучасних ринкових умов.

У зв'язку з цим виникає обґрунтована потреба в розробці та впровадженні автоматизованих програмних рішень нового покоління. Такі системи мають використовувати сучасні методи аналізу даних, зокрема алгоритми машинного навчання та моделювання часових рядів, для побудови більш адаптивних та точних прогнозних моделей. Водночас, важливим аспектом при розробці подібних рішень є забезпечення їхньої легкої інтеграції в існуючу IT-інфраструктуру підприємства. Реалізація доступу до прогнозів через стандартизовані програмні інтерфейси (API) дозволяє оперативно передавати результати моделювання іншим корпоративним системам (ERP, SCM, CRM) для своєчасного прийняття обґрунтованих операційних та тактичних рішень [5].

Мета роботи

Метою даної роботи є розробка та апробація архітектури функціонального прототипу системи для автоматизованого прогнозування попиту. Основна задача – продемонструвати повний цикл роботи подібної системи: від отримання вхідних даних з імітованого джерела до надання кінцевого прогнозу через API, використовуючи моделі аналізу часових рядів.

Основна частина

У межах проведеної роботи було спроєктовано та реалізовано прототип системи, що складається з кількох логічно відокремлених модулів, які взаємодіють між собою:

Компонент симуляції даних: Для демонстрації роботи системи було створено програмний модуль, що імітує джерело історичних даних про продажі. Цей модуль надає тестові часові ряди через спеціально розроблений API-ендпоінт, що дозволяє імітувати отримання даних з реальної бази даних або зовнішнього сервісу.

Модуль навчання моделі: Розроблено програмний компонент, відповідальний за процес навчання прогнозної моделі. Він отримує історичні дані від компонента симуляції, виконує необхідну попередню обробку та підготовку часового ряду, після чого навчає модель прогнозування. В прототипі використано алгоритми, здатні ідентифікувати та моделювати тренди і сезонні патерни в даних [4]. Результатом роботи модуля є збережений артефакт – навчена модель.

Сервіс прогнозування (API): Створено окремий сервіс у вигляді веб-API, який інкапсулює логіку використання навченої моделі. При старті сервіс завантажує збережений артефакт моделі. У відповідь на зовнішні запити (що містять, наприклад, бажаний горизонт прогнозування) сервіс використовує модель для генерації прогнозу попиту на вказаний майбутній період і повертає результат у структурованому форматі (наприклад, JSON).

Механізм взаємодії та візуалізації: Продемонстровано взаємодію між компонентами за допомогою скриптів оркестрації та клієнтських запитів до API прогнозування. Результати прогнозу візуалізуються у вигляді графіка часового ряду для наочної інтерпретації.

Запропонована архітектура (див. рис.1) дозволяє гнучко замінювати компоненти (наприклад, джерело даних або алгоритм моделювання) та забезпечує слабку зв'язність між модулями навчання та прогнозування.

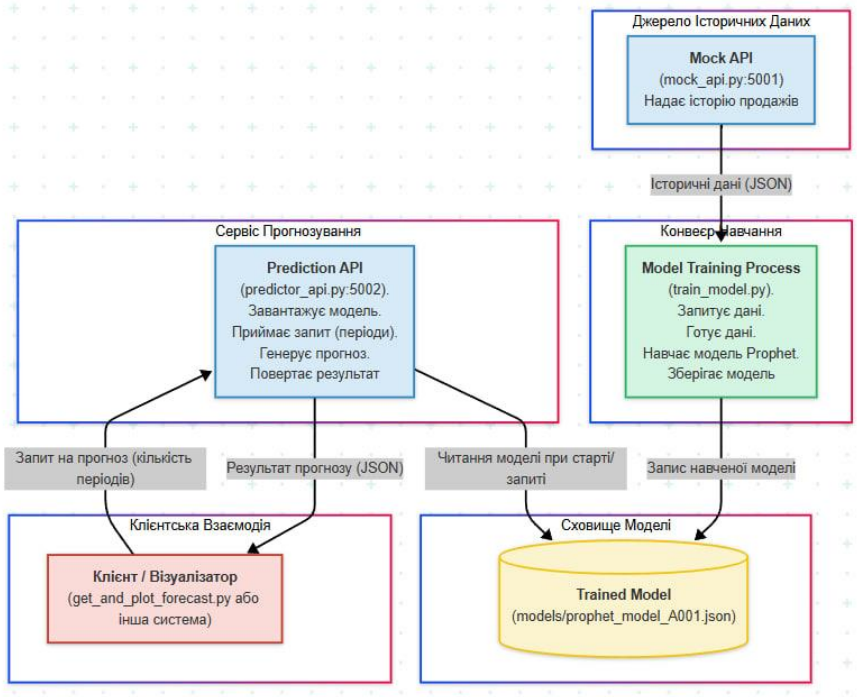


Рисунок 1 - Загальна архітектура прототипу системи прогнозування

Технічна реалізація

Функціональний прототип системи було реалізовано з використанням мови програмування Python, обраної завдяки її багатій екосистемі бібліотек, що ідеально підходять для завдань обробки даних, машинного навчання та веб-розробки. Це дозволило створити цілісне рішення в рамках єдиного технологічного стеку.

Для реалізації API – як для імітаційного джерела даних, так і для сервісу прогнозування – було застосовано легковаговий веб-фреймворк, наприклад, Flask. Такий вибір забезпечив швидке розгортання необхідних HTTP-ендпоінтів, що відповідають за надання історичних даних та прийом запитів на прогноз відповідно. Flask добре підходить для створення мікросервісів, дозволяючи легко обробляти вхідні запити та повертати результати у стандартному форматі JSON.

Робота з даними, зокрема їх завантаження, очищення, трансформація та аналіз, здійснювалася за допомогою потужної бібліотеки Pandas. Вона надає зручні структури даних (DataFrame) та широкий набір функцій для ефективної маніпуляції табличними та часовими даними, що є невід'ємною частиною підготовки даних для моделювання часових рядів.

Ключовим елементом системи є використання спеціалізованої бібліотеки для аналізу та прогнозування часових рядів, такої як Prophet, розробленої Facebook (Meta). Ця бібліотека була обрана завдяки її здатності автоматично виявляти та моделювати складні патерни в даних, включаючи довгострокові тренди, різні види сезонності (наприклад, тижневу, річну) та опціонально враховувати вплив свят чи інших зовнішніх подій. Це дозволяє створювати досить точні прогнози навіть без глибокого ручного налаштування параметрів моделі.

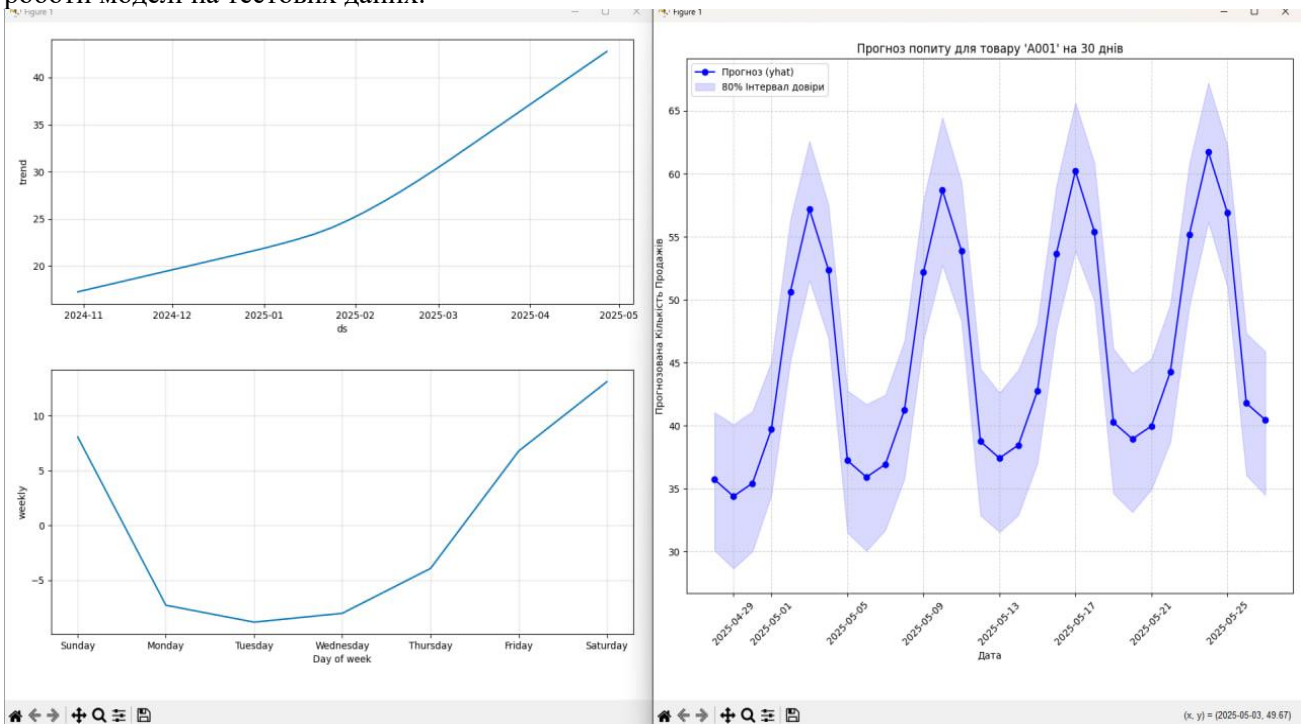
Після завершення процесу навчання на історичних даних, навчена модель серіалізувалася та зберігалася у вигляді файлу на диску. Цей крок є критично важливим, оскільки дозволяє відокремити ресурсоємний етап навчання від швидкого етапу генерації прогнозів. Сервіс прогнозування при старті або першому запиті завантажує цей збережений артефакт моделі, що унеможливлює необхідність повторного навчання при кожному запиті. Для моделей Prophet рекомендується використовувати вбудовані механізми серіалізації у формат JSON для забезпечення сумісності.

Для візуалізації результатів – як діагностичних графіків компонентів моделі (тренд, сезонності), отриманих під час навчання, так і фінального графіка прогнозу попиту з інтервалами довіри – використовувалася бібліотека Matplotlib. Вона надає гнучкі інструменти для створення різноманітних статичних графіків, що допомагають інтерпретувати результати моделювання.

Взаємодія між окремими компонентами системи (зокрема, отримання даних модулем навчання від імітаційного API та надсилання запитів клієнтом до сервісу прогнозування) була реалізована за допомогою стандартних HTTP-запитів, для виконання яких використовувалася бібліотека Requests. Це забезпечує стандартний та універсальний спосіб комунікації між сервісами.

Оркестрація запуску всіх компонентів у правильній послідовності (спочатку Mock API, потім навчання, потім Prediction API) здійснювалася за допомогою допоміжного скрипта (run.py), який керував запуском та, за потреби, зупинкою окремих процесів.

Результати роботи системи, включаючи візуалізацію розкладених компонентів часового ряду моделлю (отриманих під час аналізу навчальних даних) та графік фінального прогнозу попиту на майбутній період (згенерований сервісом прогнозування), представлені на рисунку 2. Ці візуалізації слугують підтвердженням працездатності розробленого прототипу та дозволяють оцінити якість роботи моделі на тестових даних.



а) б)

Рисунок 2 - Результати роботи системи: а) компоненти моделі (тренд, сезонність); б) графік прогнозу попиту з інтервалами довіри

Такий підхід відповідає сучасним практикам розробки систем машинного навчання (MLOps), де етапи підготовки даних, навчання моделі та її використання в продакшені (через API) є відокремленими процесами.

Висновки

Розроблений прототип системи успішно демонструє життєздатність підходу до автоматизації прогнозування попиту з використанням аналізу часових рядів та сервіс-орієнтованої архітектури. Реалізація окремих модулів для симуляції даних, навчання моделі та надання прогнозів через API дозволяє створити гнучке та масштабоване рішення.

Автоматизація процесу навчання та можливість отримання прогнозів через програмний інтерфейс суттєво знижують трудовитрати порівняно з ручними методами та відкривають можливості для інтеграції прогнозів у системи управління запасами, логістикою чи маркетингом. Хоча прототип працює на імітованих даних, він закладає міцну основу для побудови повноцінної системи прогнозування при використанні реальних історичних даних підприємства.

Список використаних джерел

1. Hyndman, R.J. & Athanasopoulos, G. Forecasting: principles and practice (2nd ed.). OTexts. 2018. 384p.
2. I.Spivak, S.Krepych and S. Budenchuk, “Methods and means of expert evaluation of software systems on the basis of interval data analysis”, 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering 2018, pp.164-167
3. I.Spivak, S.Krepych and R.Krepych, “Construction Of The Criterion For The Agree Of Expert Groups Estimates Based On Analysis Of Interval Data”, 2018 International Scientific-Practical Conference Problems of Infocommunications Science and Technology, Kharkiv, pp.261-264
4. Прогнозування часових рядів [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Прогнозування_часових_рядів
5. Taylor SJ, Letham B. “Forecasting at scale. The American Statistician”. 2018. Vol.72(1), pp.37-45.

УДК 004.41

ПЕРСОНАЛІЗОВАНА ВЕБ-ПЛАТФОРМА ДЛЯ ПЕРЕГЛЯДУ РОЗКЛАДІВ ТА БРОНЮВАННЯ КВИТКІВ У КІНОТЕАТРІ

Шпінталь М.Я.[1)], Тринька Д.В.[2)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] бакалавр

Постановка проблеми

У сучасну цифрову епоху веб-портали стали невід'ємною частиною взаємодії користувачів із різними сервісами, зокрема у сфері розваг. Однією з найпопулярніших форм дозвілля залишаються перегляди фільмів у кінотеатрах. У зв'язку з цим зростає попит на зручні онлайн-сервіси, які дозволяють швидко отримати інформацію про фільми, розклад сеансів, здійснювати бронювання квитків, а також отримувати персоналізовані рекомендації на основі індивідуальних вподобань.

Інтеграція рекомендаційних систем у веб-портали кінотеатрів забезпечує не лише підвищення користувацького досвіду, а й сприяє комерційній ефективності сервісу, оскільки збільшує кількість переглядів і залучає нову аудиторію. Такі системи допомагають користувачам орієнтуватися у великому обсязі контенту, зменшують час на пошук і формують лояльність до платформи.

Таким чином, розробка веб-порталу кінотеатру з інтегрованою рекомендаційною системою є актуальним завданням, що поєднує інструменти веб-розробки, аналізу даних і штучного інтелекту, спрямовані на покращення взаємодії з кіноглядачами та розвиток цифрових сервісів у галузі кіноіндустрії.

Мета роботи

Метою роботи є провести аналіз існуючих веб-порталів аналогів, виділити їх основні переваги та недоліки, та на основі проведеного аналізу спроектувати новий програмний продукт – веб-портал кінотеатру, який реалізуватиме можливість бронювання квитків на сеанси та реалізуватиме систему рекомендацій на основі попередніх бронювань та в подальшому його програмно реалізувати.

Дослідження рекомендаційних систем та можливостей їх впровадження у веб-порталі кінотеатру

Зараз, все більшої популярності набувають рекомендаційні системи, що базуються на штучному інтелекті, такі системи інтегрують у найрізноманітніші сфери. Такі системи аналізують попередній досвід та поведінку користувачів, та наоснові цього пропонують їм нові продукти чи послуги.

Аналіз існуючих аналогів показав, що у сфері кіноконтенту, вже є реалізовані рекомендаційні системи зокрема у таких сервісах: Netflix, IMDb. Хоча це відомі стримінгові сервіси, все ж вони містять потужні рекомендаційні системи кіноконтенту, які є популярними у всьому світі. Аналіз ж існуючих веб-порталів кінотеатрів в Україні показав, що зараз не снує таких систем, які реалізують потужні рекомендаційні системи кіноконтенту, як у стримінгових сервісів, але їх притаманна певна персоналізація через реалізацію функції персональних облікових записів користувачів. Це, безумовно, актуалізує тему цієї роботи.

Безумовно, така система буде доречною і для веб-порталу кінотеатру, адже її наявність буде підвищувати якість обслуговування глядачів та міститиме для них цілий ряд переваг як для глядачів, так і для самого закладу:

- персоналізуватиме досвід поточного користувача ресурсу, тобто буде формувати персональні пропозиції контенту для перегляду, базуючись на попередньоиу досвіді взаємодій клієнта з системою;
- сприятиме підвищенню лояльності глядачів, адже користувач веб-порталу буде частіше заходити на сайт, щоб переглянути сформовані пропозиції та зекономити час на пошук необхідного контенту;
- сприятиме збільшенню обсягів проданих квитків, адже сформованісписки рекомендованого контенту однозначно стимулюють для покупки чи бронювання квитків на кіносеанси, які користувач міг пропустити;
- сприятиме оптимізації маркетингових стратегій кінотеатру,адже на основі зібраної аналітики система зможе формувати рекомендації певного контенту для конкретних сегментів аудиторії;

- сприятиме підвищенню конкурентоспроможності конкретного кінотеатру чи їх мережі, тобто інтерактивна персоналізація веб-порталу робитиме веб-платформу унікальною, яка буде суттєво відрізнятися від типових веб-сайтів із звичайним розкладом кіносеансів.

Аналіз вимог

Для якісної програмної реалізації будь-якого проєкту дуже важливим є етап аналізу вимог. У межах цієї праці на цьому етапі було розроблено діаграму варіантів використання системи веб-порталу кінотеатру, яка подана на рисунку 1. Вона відображає функціонал майбутньої програмної системи та можливості взаємодії з ним користувачів.

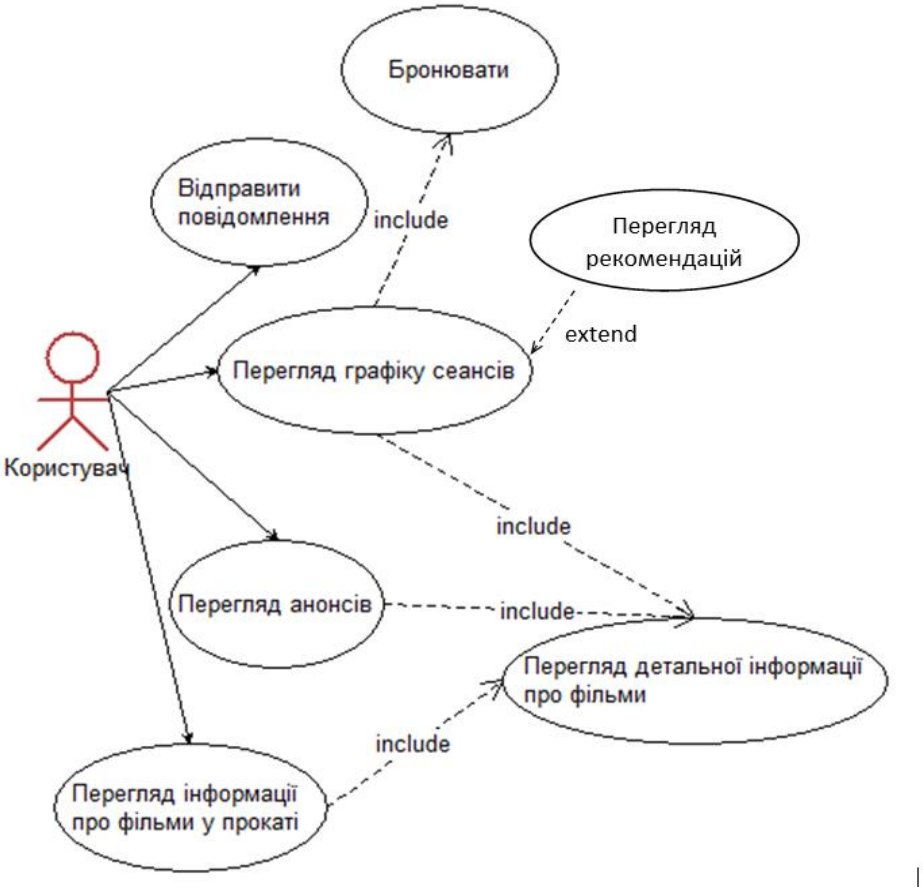


Рисунок 1 – Діаграма варіантів використання

Як видно з рисунку 1, користувач веб-порталу зможе переглядати анонси фільмів, графіки сеансів, детальну інформацію про фільми, а також матиме доступ до персональних рекомендацій на основі попередніх бронювань, після авторизації. Реалізація запропонованого функціоналу, дозолить персоналізувати пересічний веб-портал кінотеатру та популяризувати його серед користувачі мережі.

Висновок

У праці було досліджено необхідний функціонал для сучасного веб-порталу кінотеатру. Показано, що важливою складовою частиною такого функціоналу має бути потужна рекомендаційна система, яка повинна базуватися на історії переглядів та бронювань зареєстрованих користувачів. Проведено аналіз вимог до розроблюваного веб-порталу кінотеатру та побудовано діаграму варіантів використання системи.

Список використаних джерел

1. Delimayanti, M. K., Laya, M., Warsuta, B., Faydhurrahman, M. B., Khairuddin, M. A., Ghoyati, H., Naryanto, R. F. Web-Based Movie Recommendation System using Content-Based Filtering and KNN Algorithm. Inproc. 9th International Conference on Information Technology, Computer, and Electrical Engineering, 2022, pp. 314-318.
2. Wibisono, C. H., Purwanti, E., & Effendy, F. A systematic literature review of movie recommender systems for movie streaming service. In AIP Conference Proceedings, 2023, Vol. 2554, No. 1.
3. Movie Lens: Collaborative Filtering for Movie Recommendations [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/MovieLens.
4. Çano, E., & Morisio, M. Hybrid recommender systems: A systematic literature review. Intelligent data analysis, 2017, 21(6), pp.1487-1524.

УДК 004.4

ПРОГРАМНА СИСТЕМА ДЛЯ МОНІТОРИНГУ ТА УПРАВЛІННЯ АГРОВИРОБНИЦТВОМ

Порплиця Н.П.[1)], Корман М.Б.[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] студент

Постановка проблеми

Сучасне агровиробництво переживає значні виклики: глобальні зміни клімату, виснаження родючості ґрунтів, коливання цін і перебої в постачанні мінеральних добрив. Традиційні підходи до планування сівозмін і норм добрив часто базуються на емпіричних даних і не враховують динамічні характеристики ґрунтового середовища, що призводить до надмірного внесення поживних речовин, деградації ґрунтів та зниження довгострокової продуктивності посівів. Водночас для України, одного з провідних експортерів зернових культур, недостатня адаптація агротехнологій під сучасні умови створює економічні і екологічні ризики та збільшує залежність від імпорту добрив.

Необхідність комплексного підходу до моніторингу стану ґрунтів і планування сівозміни стає критичною для забезпечення стабільної врожайності при мінімальних ресурсних затратах. Таким чином постає завдання створення програмної системи, що на основі агрохімічного аналізу ґрунту та історії вирощування культур формуватиме оптимальні схеми сівозміни й розраховуватиме дозування добрив з урахуванням екологічних та економічних параметрів.

Мета і завдання роботи

Метою даного курсового проєкту є створення програмної системи для моніторингу та управління агровиробництвом, яка б на основі поточних даних про стан ґрунтів і попередні культури забезпечувала адаптивне планування сівозміни, розрахунок рекомендованих норм внесення добрив та автоматизовану генерацію звітів і візуалізацій для оцінки ефективності агротехнологічних процесів.

Для досягнення цієї мети передбачається всебічний аналіз існуючих агрохімічних методів дослідження ґрунтів та програмних підходів до обробки таких даних, розробка алгоритму інтерпретації хімічного складу ґрунту й результатів попередніх врожаїв, інтеграція моделей оптимізації сівозміни з урахуванням підтримки родючості, а також створення модуля розрахунку норм мінеральних добрив і зручного користувацького інтерфейсу для введення даних і формування звітів.

Спосіб розрахунку кількості добрив для використання

Для коректної та безперебійної роботи програми потрібно підібрати правильну форму розрахунку. Вибір правильної формули для обчислень урожаю та використання добрив, є критично важливим для коректної роботи програми.

У цій роботі буде використано балансовий спосіб для визначення необхідної кількості того чи іншого елемента.

Балансовий метод розрахунку норми добрив передбачає визначення різниці між потребою культури у поживних речовинах та їх наявними запасами в ґрунті з урахуванням коефіцієнтів засвоєння. Спочатку визначають запланований урожай (ц/га) та винос елементів живлення на одиницю продукції (кг/ц).Потім агрохімічним аналізом обраховують запас поживних речовин у ґрунті (кг/га) і застосовують коефіцієнт використання ґрунтових запасів (30–50 %) та добрив (50–80 %). Норма добрив (кг/га діючої речовини) обчислюється за формулою: (урожай×винос – запас×коефіцієнт ґрунту) / коефіцієнт добрив. При низьких показниках агрохімічного аналізу норма добрив зростає, а при високих запасах – зменшується. Це одночасно забезпечує збалансоване оптимальне живлення рослин, ефективне використання ресурсів та суттєву економію коштів.

Кожна сільськогосподарська культура буде мати свої потреби в поживних речовинах, а саме азот. калій, фосфор, В кожному типу грунту будуть стандартні значення поживних речовин з можливістю їх зміни.

Цільова функція в даному методі буде оцінювати, скільки саме потрібно добрив кожного типу кг/га, за наявності площо поля автоматично перераховувати загальну вагу добрив.

Програмна реалізація

Для виконання програмної реалізації я використав мову програмування C# і Фреймворк: .NET (WPF – Windows Presentation Foundation) для створення графічного інтерфейсу.

Використання C# у поєднанні з .NET дає змогу швидко та ефективно розробляти десктопні застосунки завдяки потужній стандартній бібліотеці класів і тісній інтеграції з середовищем розробки Visual Studio. Завдяки сильній статичній типізації та вбудованому механізму збирання сміття.

Фреймворк WPF, що працює поверх .NET, дозволяє будувати сучасний і гнучкий графічний інтерфейс із чітким розділенням логіки. Активна підтримка Microsoft і велика спільнота розробників гарантують довгострокову підтримку, регулярні оновлення та багатий вибір готових рішень для будь-яких сценаріїв.

Розробка проекту велась у середовищі Visual Studio Community 2022, яке забезпечує зручні інструменти для написання, налагодження та тестування коду. У якості основи обрано мову програмування C# та фреймворк .NET з використанням WPF (Windows Presentation Foundation) для створення сучасного графічного інтерфейсу.

На рисунку 1 показано приклад використання програми з вже проведеним розрахунком добрив.

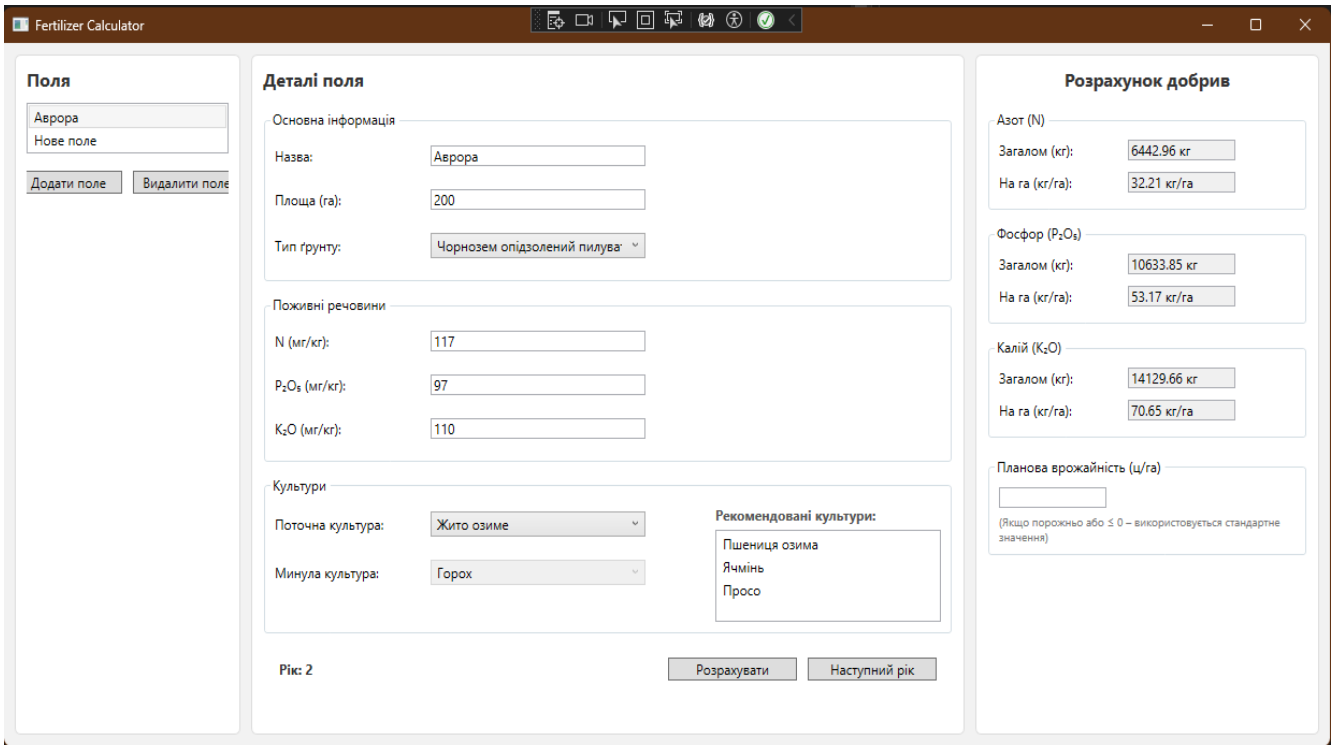


Рисунок 1 – Основна форма розробленого програмного модуля

Висновок

У роботі було роботі досліджено систему для моніторингу та управління агровиробництвом, яка на основі агрохімічного аналізу ґрунту та історії вирощування культур забезпечує адаптивне планування сівозмін і розрахунок рекомендованих норм внесення добрив. Використання балансового методу для обчислення потреби в поживних речовинах дозволяє враховувати як винос елементів із ґрунту, так і їх надходження з добрив, що зменшує ризики надлишкового чи недостатнього внесення та оптимізує витрати ресурсів.

Список використаних джерел

1. C# language documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/csharp/
2. Git Documentation [Електронний ресурс] – Режим доступу: https://git-scm.com/doc
3. GitHub Docs [Електронний ресурс] – Режим доступу: https://docs.github.com/en
4. Git in Visual Studio [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/version-control/git-with-visual-studio?view=vs-2022
5. Introducing JSON [Електронний ресурс] – Режим доступу: https://www.json.org/json-en.html
6. Newtonsoft json documentation [Електронний ресурс] – Режим доступу: https://www.newtonsoft.com/json/help/html/introduction.htm
7. Visual Studio documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/visualstudio/windows/?view=vs-2022
8. Visual Studio product family documentation [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/uk-ua/visualstudio/?view=vs-2022
9. XAML overview (WPF .NET) [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/dotnet/desktop/wpf/xaml/?view=netdesktop-9.0

УДК 004.4

ВЕБ-ОРІЄНТОВАНА СИСТЕМА УПРАВЛІННЯ ФАРМАЦЕВТИЧНИМИ ОПЕРАЦІЙНИМИ ПРОЦЕСАМИ

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

I. Постановка проблеми

Сучасні фармацевтичні установи стикаються з необхідністю цифровізації своєї діяльності. Традиційне ведення обліку медикаментів, клієнтів і продажів потребує значних ресурсів та схильне до людських помилок. Водночас, стрімкий розвиток інформаційних технологій дозволяє створювати інтерактивні системи, що спрощують операційну діяльність. Потреба у впровадженні веб-орієнтованих систем, здатних працювати в реальному часі, зростає разом із вимогами до зручності, точності та безпеки обліку.

II. Мета роботи

Розробити веб-орієнтовану систему для ефективного управління фармацевтичними операційними процесами. Система повинна дозволяти керування даними про медикаменти, клієнтів, рецепти, користувачів і продажі. Особлива увага приділяється розмежуванню доступу за ролями, зручності інтерфейсу та захисту інформації.

III. Основна частина

Розроблена система побудована за архітектурою “клієнт-сервер”[1](див.рис.1).Такий підхід передбачає поділ додатку на два основних компоненти:

- Клієнт (frontend) – відповідає за інтерфейс користувача, реалізований на основі бібліотеки React[2], що забезпечує динамічне оновлення даних без перезавантаження сторінки
- Сервер (backend) – побудований з використанням Node.js[4] + Express, забезпечує обробку HTTP-запитів, аутентифікацію, валідацію та бізнес-логіку.

Ця модель була обрана завдяки її гнучкості, масштабованості та розділенню відповідальностей, що дозволяє незалежно розвивати frontend і backend частини проєкту.

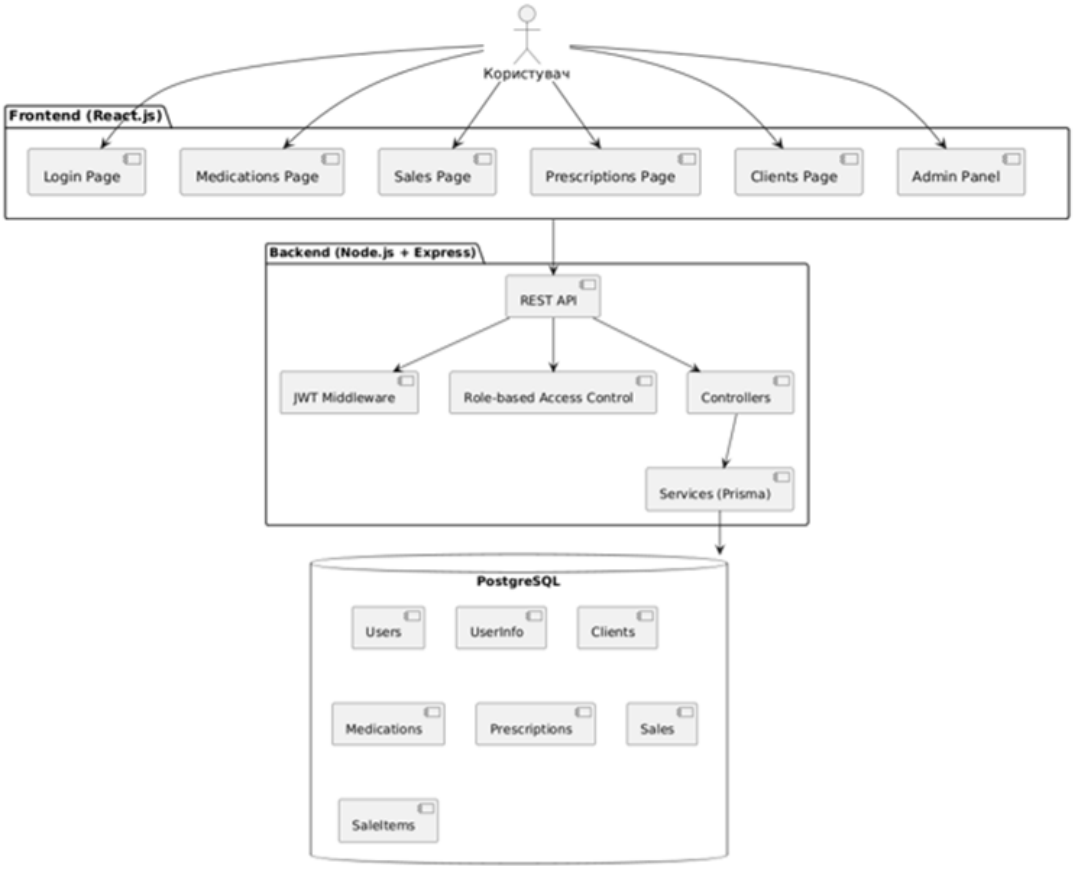


Рисунок 1 – Вигляд архітектури системи управління

Розробка веб-орієнтованої системи управління фармацевтичними операційними процесами здійснювалася із застосуванням сучасного стеку технологій та інструментів, орієнтованих на забезпечення продуктивності, зручності та безпеки. Фронтенд реалізовано з використанням React — однієї з найпопулярніших JavaScript-бібліотек для створення інтерфейсів користувача. Компонентний підхід React дозволив побудувати гнучку структуру UI, де кожен блок (таблиця, форма, модальне вікно) реалізовано як окремий незалежний компонент, що спрощує тестування та повторне використання. Для маршрутизації використано ReactRouter, а для обробки форм — ReactHookForm. Візуальне оформлення інтерфейсу реалізоване за допомогою Tailwind CSS[3] — утилітарного фреймворку, що забезпечує гнучке налаштування стилів без потреби в написанні кастомного CSS. Для зв'язку з серверною частиною використано Axios — клієнт HTTP-запитів, через який реалізовано комунікацію з REST API backend-серверу. Доступ до маршрутів у фронтенді обмежено через компонент PrivateRoute, який перевіряє JWT-токен користувача перед наданням доступу до певних сторінок.Бекенд реалізовано на Node.js із застосуванням Express — гнучкого веб-фреймворку для створення RESTful API. Вся бізнес-логіка, перевірка прав доступу, валідація вхідних даних та обробка помилок організовані у вигляді окремих middleware. Для аутентифікації користувачів використовується JWT (JSON WebToken) [7,8]. При кожному запиті клієнта сервер перевіряє валідність токена й роль користувача. Для зберігання даних використовується реляційна база даних PostgreSQL[6]. Архітектура БД побудована на основі принципів нормалізації, забезпечуючи чітке розділення інформації про медикаменти, користувачів, рецепти, продажі та клієнтів,на рисунку 2 зображено ERD діаграму БД.

Рисунок 2 – Структура бази даних

Моделювання взаємозв'язків та генерація SQL-запитів реалізовано через Prisma ORM[5], що дозволяє працювати з базою даних за допомогою об'єктів і типів. Prisma також забезпечує

автоматичну генерацію схем, зручне створення міграцій та перевірку типів на рівні коду. Проєкт підтримує рольову модель доступу (RBAC) — окремі функції та API-доступи дозволені або заборонені на основі ролі користувача (USER, PHARMACIST, ADMIN, DOCTOR), на рисунку 3 зображено діаграму варіантів використання. Усі запити до API проходять авторизаційну перевірку.

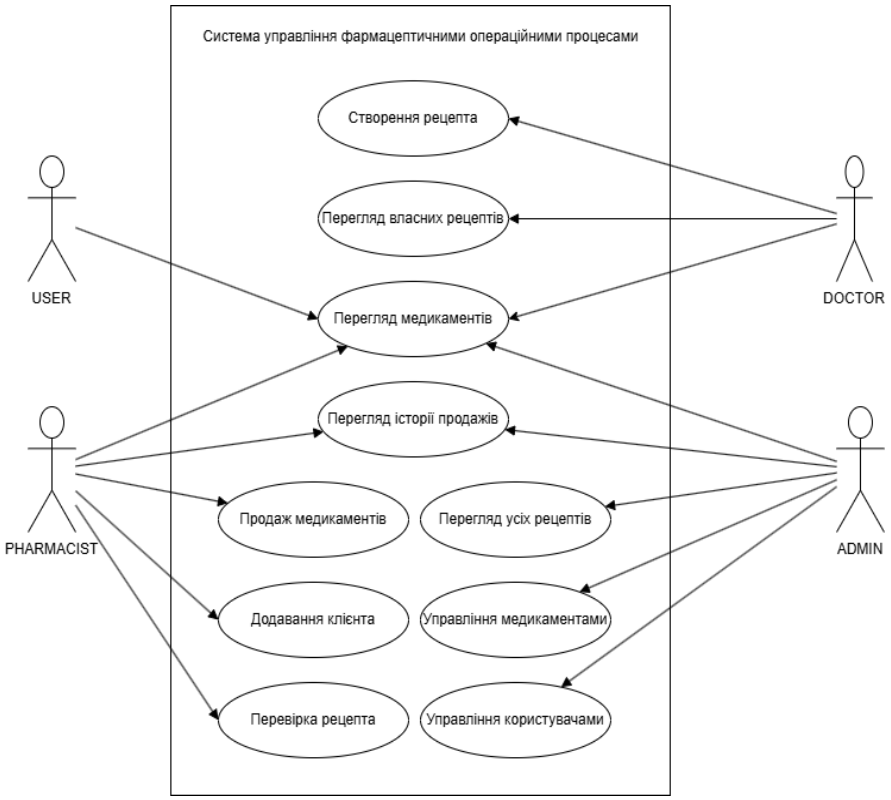


Рисунок 3 – Діаграма варіантів використання

Таким чином, реалізація базується на поєднанні сучасних підходів до UI/UX-дизайну, ефективної серверної логіки, безпечної авторизації, надійного збереження даних і зручного розширення в майбутньому[9,10].

Висновок

Розробка веб-системи управління фармацевтичними операційними процесами потребувала комплексного підходу до вибору бази даних, архітектури і технологічного стеку. У межах реалізації було обґрунтовано доцільність застосування архітектури «клієнт-сервер» із розподіленням відповідальностей між frontend і backend частинами, що забезпечило масштабованість, гнучкість і зручність підтримки.

Використання React, Tailwind CSS та ReactRouter дозволило створити динамічний, адаптивний і зручний користувацький інтерфейс. Серверна частина на Node.js з Express і Prisma ORM забезпечує надійну взаємодію з реляційною базою даних PostgreSQL, підтримує складні зв’язки між сутностями та гарантує збереження й цілісність даних. Реалізована система автентифікації на основі JWT та рольової моделі доступу (RBAC) забезпечує захист даних і контроль дій користувачів на всіх рівнях взаємодії. Результатом є стабільна, розширювана та безпечна веб-система, що автоматизує ключові процеси у сфері фармації — від роботи з клієнтами й продажу медикаментів до управління персоналом і рецептами.

Обраний стек технологій і архітектурні рішення повністю відповідають вимогам до сучасних веб-додатків у галузі охорони здоров’я. Такий підхід доводить ефективність застосування відкритих інструментів і сучасних принципів розробки для цифрової трансформації фармацевтичної діяльності.

Список використаних джерел

1. QATestLab. Client-server architecture [Електронний ресурс] – Режим доступу: https://en.training.qatestlab.com/blog/technical-articles/client-server-architecture/
2. React – A JavaScript library for building user interfaces [Електронний ресурс] – Режим доступу: https://reactjs.org
3. Tailwind CSS. Rapidly build modern websites without ever leaving your HTML [Електронний ресурс] – Режим доступу: https://tailwindcss.com
4. Node.js. JavaScript runtime built on Chrome's V8 JavaScript engine [Електронний ресурс] – Режим доступу: https://nodejs.org
5. Prisma. Next-generation Node.js and TypeScript ORM [Електронний ресурс] – Режим доступу: https://www.prisma.io
6. PostgreSQL: The World’s Most Advanced Open Source Relational Database [Електронний ресурс] – Режим доступу: https://www.postgresql.org
7. JWT.IO. JSON WebTokens Introduction [Електронний ресурс] – Режим доступу: https://jwt.io/introduction
8. Spivak, I., Krepych, S., Litvynchuk, M. and Spivak, S. “Validation and data processing in JSON format”, *IEEE EUROCON 19th International Conference on Smart Technologies*, 2021. pp.326–330
9. I. Spivak, S. Krepych and S. Budenchuk, “Methods and means of expert evaluation of software systems on the basis of interval data analysis”, 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering 2018, pp.164-167
10. Poliarush O., Krepych S., Spivak I., “Hybrid approach for data filtering and machine learning inside content management system”, Advanced Information Systems , 2023, 7(4), pp.70-74

УДК 004.4

РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ БРОНЮВАННЯ КВИТКІВ

Іванюк М.Ю.
Західноукраїнський національний університет
бакалавр

I. Постановка проблеми

У сучасному світі значна частина операцій переміщується в цифровий простір, і процес бронювання квитків на заходи, транспорт чи кіносеанси не є винятком. Незручність традиційного бронювання, пов'язана з необхідністю фізичної присутності, чергами або обмеженим часом роботи кас, потребує вирішення. Мобільні застосунки дають змогу швидко та зручно здійснювати бронювання незалежно від місцезнаходження користувача. Проте багато наявних рішень не враховують інтерфейсну зручність, багатомовність або офлайн-доступ. Тому виникла потреба у створенні зручного, швидкого та функціонального мобільного додатка, який дозволить здійснювати бронювання квитків через Android-пристрої. [1].

II. Мета роботи

Метою роботи є розробка мобільного додатка для Android, що дозволяє користувачам переглядати афіші подій, здійснювати бронювання квитків, зберігати їх у локальній базі та отримувати сповіщення про майбутні події. Додаток повинен мати зручний інтерфейс, працювати без підключення до інтернету (у режимі офлайн) і забезпечувати збереження та захист даних користувача.

III. Основна частина

Застосунок розроблений мовою Java із використанням фреймворку Android SDK у середовищі AndroidStudio [1]. Для локального збереження даних, зокрема історії замовлень, профілю користувача та списку збережених квитків, застосовується база даних SQLite. Робота з базою даних реалізована через клас DBHelper, який містить усі необхідні методи для читання, запису, оновлення та видалення записів [3].

Для реалізації взаємодії між компонентами застосовуються елементи Intent, Bundle, а також системи життєвого циклу Android-додатків. Архітектура побудована за багаторівневим принципом, де логіка бізнес-процесів повністю відокремлена від користувацького інтерфейсу, що дозволяє легко масштабувати додаток [2].

Інтерфейс додатка створено з використанням елементів RecyclerView, CardView, ScrollView, AlertDialog, BottomNavigationView, що забезпечує зручну навігацію та швидкий доступ до основного функціоналу. Весь контент динамічно оновлюється без потреби перезапуску сторінки, що покращує користувацький досвід.

Основні функції застосунку:

- Пошук подій або рейсів: реалізовано зручну систему пошуку з фільтрацією за датою, категорією, місцем проведення.
- Бронювання квитків: користувач може переглянути доступні місця, вибрати потрібні та забронювати їх.
- Оплата: інтеграція з платіжними сервісами (наприклад, GooglePay) забезпечує безпечну та швидку оплату квитків [2].
- Зберігання квитків: після бронювання квиток зберігається в локальній базі та може бути відкритий у будь-який момент без доступу до Інтернету.
- Push-сповіщення: реалізована система повідомлень на основі Firebase, яка інформує про наближення події або зміни у замовленні.
- Перегляд історії: користувач має доступ до власної історії замовлень, може повторно переглянути або скасувати бронювання.
- Збереження у «обране»: події можна додавати до «обраного» для швидкого доступу.

Крім основного функціоналу, в додатку реалізована система QR-кодів для перевірки квитків на вході. Після бронювання генерується унікальний код, який можна відсканувати при перевірці, що значно пришвидшує вхід учасників на заходи. Це особливо актуально для великих івентів, де важлива швидкість і точність ідентифікації. Застосунок також підтримує адаптивну верстку, що дозволяє комфортно користуватись сервісом на смартфонах з різним розміром екрана. У

майбутньому планується реалізація хмарної синхронізації замовлень, створення веб-версії для адміністраторів подій, а також додавання модулів аналітики, що дозволить власникам заходів відстежувати кількість бронювань, заповненість залів та популярність подій у реальному часі.

Інтерфейс додатку побудовано на основі адаптивної верстки з використанням компонентів MaterialDesign. Основні екрани додатку: сторінка подій, пошук, екран з деталями події, бронювання, особистий кабінет користувача та історія замовлень. Для ефективної взаємодії користувача з додатком реалізовано спливаючі повідомлення, діалогові вікна, інтерактивні списки та QR-коди для кожного замовлення.

Розробка додатку проходила кілька ключових етапів: аналіз вимог, проєктування інтерфейсу, реалізація функціоналу, тестування та оптимізація. На першому етапі було визначено цільову аудиторію, її потреби та типові сценарії використання: замовлення квитків на транспорт (автобуси, потяги), на культурні події (концерти, театри), а також на кіносеанси. Наступним кроком стало створення прототипу в системі Figma, який дозволив сформувати логіку переходів між екранами. Після затвердження прототипу розпочалась реалізація інтерфейсу та основних функцій у Flutter.

Тестування проводилось у кількох середовищах: на фізичних Android-пристроях із різними розмірами екранів, а також на емуляторах для моделювання повільного з'єднання. Було виконано ручне та автоматизоване тестування основних сценаріїв — бронювання, скасування, перегляд квитків, а також перевірку валідації введених даних. У результаті було досягнуто стабільності, швидкої роботи та низької ймовірності збоїв.

Усі чутливі дані зберігаються у зашифрованому вигляді, а платіжна інформація обробляється виключно на стороні захищених сервісів GooglePay або Stripe. Це відповідає сучасним вимогам безпеки та захисту персональних даних.

Продуктивність застосунку оптимізовано завдяки використанню кешування та мінімізації запитів до бази даних, що забезпечує швидку роботу навіть на пристроях із обмеженими ресурсами.

Також закладено потенціал до масштабування: у майбутньому можливе додавання функцій аналітики, персоналізованих рекомендацій, інтеграції з соціальними мережами або створення веб-версії сервісу.

Крім основної функції бронювання, додаток може відігравати роль цифрового помічника для організаторів подій. Завдяки впровадженню аналітичних інструментів у майбутніх версіях можна відстежувати статистику продажу квитків, пік завантаження системи, популярність подій за категоріями, містами чи віковими групами користувачів. Такі дані сприятимуть ухваленню обґрунтованих рішень щодо розміщення реклами, вибору часу події або ціноутворення.

Інтерфейс додатку спроектовано таким чином, щоб забезпечити адаптивність до різних розмірів екранів — як смартфонів, так і планшетів. Дизайн реалізовано з урахуванням принципів доступності: висока контрастність кольорів, чітка типографіка та можливість масштабування шрифтів роблять додаток зручним навіть для людей з порушеннями зору. У майбутньому заплановано впровадження голосових підказок та підтримку навігації за допомогою зовнішніх пристроїв для людей з інвалідністю.

Окрему увагу було приділено реалізації багатомовного інтерфейсу. На поточному етапі додаток підтримує українську та англійську мови, проте архітектура дозволяє легко додавати інші мовні пакети. Такий підхід відкриває перспективу використання додатку на міжнародному рівні, зокрема для туристів, які відвідують події або користуються транспортними послугами в Україні. Завдяки цьому розробка має потенціал стати частиною ширшої цифрової екосистеми у сфері мобільних сервісів.

Висновок

Розроблений мобільний застосунок для бронювання квитків є сучасним, зручним і функціональним інструментом, який дозволяє автоматизувати процес купівлі квитків на події чи транспорт. Його основними перевагами є кросплатформність, адаптивність до різних пристроїв, інтуїтивний інтерфейс, офлайн-роботата можливість масштабування. У перспективі додаток може бути інтегрований з іншими сервісами, такими як електронні квитки, календарі, системи лояльності та платіжні системи. Це робить його актуальним продуктом для широкого кола користувачів, а також потенційно привабливим для комерційного використання.

Список використаних джерел

1. Android Developers. [Електронний ресурс]. – Режим доступу: https://developer.android.com.
2. Жуковський С. М., Коваль А. І. Розробка мобільних додатків: архітектура, інтерфейси, інтеграція. – Львів: ЛНУ, 2021. – 212 с.
3. Васильєв А. Н. Програмування під Android. Професійний підхід. – К.: Діалектика, 2020. – 448 с.

УДК 004.4

ВЕБ-ОРІЄНТОВАНИЙ ДОДАТОК ПЛАНЕР

Мочурад Б.Р.
Західноукраїнський національний університет
бакалавр

I. Постановка проблеми

Розвиток цифрових технологій значно змінив підхід до організації повсякденної діяльності, особливо в контексті управління часом, завданнями та ресурсами. З кожним роком зростає попит на ефективні інструменти планування, які дозволяють користувачам швидко та зручно організовувати особисті справи, робочі процеси або навчальні завдання. В умовах динамічного ритму життя планування набуває все більшого значення як для окремих користувачів, так і для організацій різного масштабу. Воно допомагає підвищити продуктивність, забезпечити ефективне використання часу, мінімізувати ризики та досягати поставлених цілей.

Традиційні засоби планування, як-от паперові щоденники, електронні таблиці або десктопні програми, мають ряд недоліків: відсутність мобільності, складність в інтеграції з іншими сервісами, обмежена автоматизація та низький рівень адаптивності до змін. Зі збільшенням кількості користувачів мобільних пристроїв, особливо смартфонів, все більше уваги приділяється розробці веб-орієнтованих додатків, які можна використовувати незалежно від платформи або фізичного розташування. Такі додатки забезпечують високу доступність, дозволяють зберігати та синхронізувати дані в хмарі, а також можуть бути легко масштабовані під потреби різних груп користувачів.

Таким чином, проблема полягає у відсутності доступного, зручного та функціонального веб-додатку, який би дозволяв користувачам ефективно планувати свої справи, синхронізувати дані між пристроями, мати можливість авторизації та управління завданнями без необхідності складної інсталяції або налаштування. Це й визначає актуальність розробки сучасного веб-орієнтованого додатку "Планер".

II. Мета роботи

Метою даної дипломної роботи є розробка сучасного веб-орієнтованого додатку "Планер", який забезпечить користувачам зручний інструмент для ефективного планування особистих і робочих завдань. Додаток повинен бути доступним з будь-якого пристрою, мати інтуїтивно зрозумілий інтерфейс, підтримку авторизації, зберігання даних у хмарі та функціонал керування завданнями.

Для досягнення цієї мети необхідно:

- дослідити предметну область та виявити основні потреби користувачів у плануванні;
- провести аналіз існуючих веб-орієнтованих рішень та визначити їх переваги й недоліки;
- обґрунтувати вибір технологій, зокрема використання платформи Firebase як бекенд-рішення;
- спроєктувати архітектуру додатку, включно з базою даних, інтерфейсом та функціональністю;
- реалізувати клієнтську частину з урахуванням принципів UI/UX-дизайну;
- протестувати роботу додатку та перевірити його зручність, стабільність і функціональність.

Очікуваним результатом є створення повноцінного веб-додатку, що полегшує процес планування, забезпечує швидкий доступ до інформації та відповідає сучасним вимогам користувачів.

III. Основна частина

Для реалізації фронтенду використано популярну бібліотеку React, яка забезпечує високу продуктивність, модульність і зручність у розробці. React дозволяє створювати окремі компоненти для кожного елемента інтерфейсу, що значно полегшує підтримку та розширення додатку. Завдяки компонентному підходу, кожен елемент інтерфейсу можна модифікувати незалежно, що підвищує гнучкість та зменшує час розробки. Крім того, React підтримує реактивність, що означає автоматичне оновлення інтерфейсу при зміні даних без необхідності перезавантажувати всю сторінку. Інтерфейс додатку адаптивний, що дозволяє зручно працювати як на великих екранах, так і на мобільних пристроях, завдяки підтримці принципів responsivedesign.

Для серверної частини використано Node.js, що є асинхронним середовищем для виконання JavaScript на сервері. Node.js відомий своєю високою продуктивністю завдяки поділу задач на невеликі частини, що дозволяє обробляти великі обсяги одночасних запитів, не блокуючи процеси. Для спрощення роботи з HTTP-запитами та маршрутизацією використано фреймворк Express. Express дозволяє легко організувати обробку запитів від клієнта, визначати маршрути для різних дій (створення, редагування, видалення завдань), а також забезпечує підтримку різних методів HTTP (GET, POST, PUT, DELETE). Завдяки цим інструментам, серверна логіка є гнучкою, масштабованою та ефективною[1].

Для зберігання даних використано MongoDB, документно-орієнтовану базу даних, яка відома своєю гнучкістю та можливістю роботи з великими обсягами структурованих даних. MongoDB дозволяє зберігати дані у вигляді JSON-документів, що зручно для додатків, які часто змінюють структуру своїх даних або потребують горизонтального масштабування. У рамках "Планера" дані користувачів, завдань і категорій зберігаються в окремих колекціях, що дає змогу ефективно організувати взаємодію між ними. Завдяки MongoDB, можна швидко виконувати запити на вставку, оновлення, видалення та пошук даних, що суттєво підвищує ефективність роботи додатку[2].

Нижче зображено структуру колекції завдання ,яка описує основні поля документів:

Таблиця 1

Поля та їх опис колекції завдання

Поле	Тип даних	Опис
_id	ObjectId	Унікальний ідентифікатор завдання (генерується автоматично MongoDB).
taskName	String	Коротка назва завдання, яка відображає його суть.
description	String	Детальний опис завдання, що пояснює його суть і вимоги.
endTime	Date	Дата та час, до якого завдання має бути виконане.
timeToSpend	Number	Орієнтовний час, необхідний для виконання завдання (у годинах).
owner	ObjectId	Унікальний ідентифікатор користувача, який створив це завдання (посилання на _id з колекції користувачі).
done	Boolean	Статус виконання завдання (true – виконано, false – очікує виконання).
createdAt	Date	Дата та час створення завдання (генерується автоматично).

Основними функціональними вимогами до веб-додатку "Планер" є забезпечення повного циклу взаємодії користувача із системою планування, що включає низку ключових можливостей для ефективного управління завданнями та безпеки даних. Користувач повинен мати можливість створити обліковий запис, вказавши унікальний email та пароль. Для забезпечення безпеки, дані пароля мають зберігатися в зашифрованому вигляді, використовуючи сучасні алгоритми хешування, такі як bcrypt, що захищає користувача від несанкціонованого доступу до його облікового запису. Після реєстрації користувач може авторизуватися в системі через форму входу, вказавши свій email та пароль. Після успішної авторизації, користувач отримує доступ до персональних даних за допомогою JWT-токена (JSON WebToken). JWT є компактним і безпечним способом передачі даних між клієнтом і сервером у вигляді підписаного JSON-об'єкта, що дозволяє підтвердити особу користувача та його права доступу [3].

Крім того, передбачена можливість зміни паролю, що вимагає введення поточного паролю для підтвердження прав користувача. Це забезпечує додатковий рівень захисту облікового запису. Основна функціональність додатку зосереджена на управлінні завданнями. Користувач може створювати нові завдання, вказуючи їх назву, опис, дедлайн, пріоритет і орієнтовний час виконання. Кожне завдання автоматично прив'язується до облікового запису користувача, що забезпечує індивідуальне планування.Також передбачено редагування вже створених завдань, що дозволяє змінювати будь-які параметри, такі як дата дедлайну, пріоритет або опис. Користувач може видаляти непотрібні завдання, що допомагає підтримувати порядок в системі та уникати перевантаження. Важливою функцією є можливість відстежувати час, витрачений на виконання завдань. Це дозволяє користувачу моніторити продуктивність і ефективно планувати свій час, а також дає змогу оцінити,

чи були завдання виконані в межах встановлених термінів. На рисунку 1 зображено як саме користувач додає завдання.

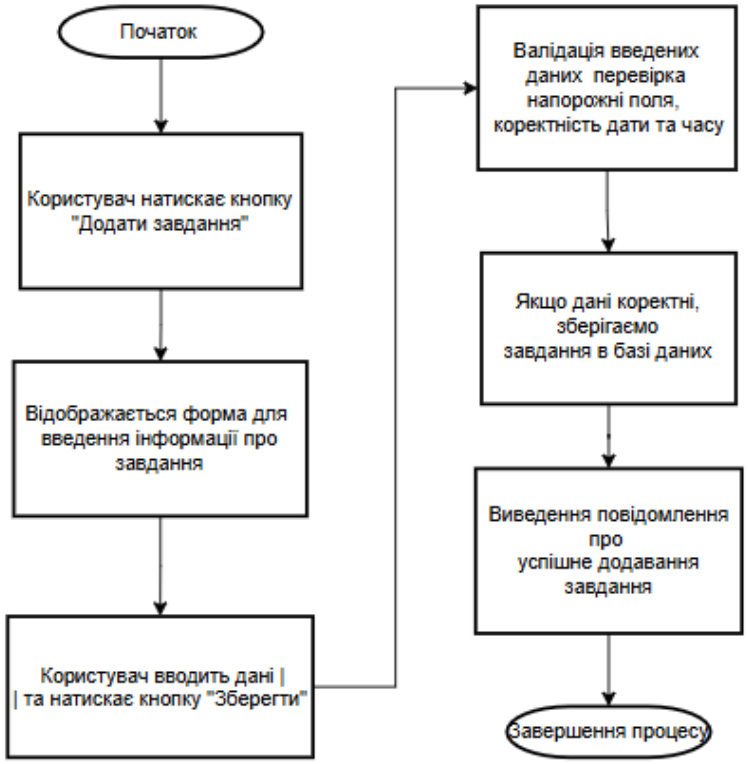


Рисунок 1 – Блок-схема додавання нового завдання

Інтерфейс користувача спроєктовано з дотриманням сучасних принципів UI/UX-дизайну: мінімалізм, зручна навігація, адаптивність до мобільних пристроїв. Завдяки компонентному підходу React, інтерфейс легко масштабувати та підтримувати.

Завершальний етап роботи включав тестування додатку, під час якого перевірялась коректність роботи API, обробка помилок, а також зручність інтерфейсу. Тестування проводилось як вручну, так і з використанням автоматизованих інструментів. У результаті усунуто ряд недоліків, покращено логіку авторизації та перевірки введених даних.

Результатом роботи є сучасний веб-додаток "Планер", який дозволяє ефективно організовувати завдання та керувати часом, забезпечуючи зручність, безпеку та швидкодію при роботі на будь-якому пристрої.

Висновок

У межах роботи було спроєктовано та реалізовано веб-орієнтований додаток "Планер", який дозволяє користувачам ефективно планувати особисті та робочі завдання. Проведено аналіз предметної області, вивчено існуючі рішення, що дало змогу визначити основні вимоги до функціоналу системи. У результаті дослідження було створено додаток, що складається з клієнтської частини, реалізованої на React, серверної логіки на Node.js з використанням Express, а також бази даних MongoDB для збереження інформації про користувачів і завдання.

Забезпечено повний набір функцій: реєстрацію та авторизацію користувачів, створення, редагування, видалення завдань, встановлення дедлайнів, визначення часу виконання та відстеження прогресу. Інтерфейс спроєктовано відповідно до сучасних принципів UI/UX-дизайну, що забезпечує зручність використання на різних пристроях.

Проведене тестування підтвердило стабільність роботи додатку та відповідність поставленим вимогам. Отриманий результат може бути використаний як основа для подальшого розвитку системи, зокрема, додавання нових функцій, таких як сповіщення, календарна візуалізація та синхронізація з іншими сервісами.

Таким чином, поставлену мету роботи досягнуто — створено сучасний, функціональний та масштабований веб-додаток для планування, який вирішує актуальні завдання організації особистого часу.

Список використаних джерел

1. “Рішення Node.js для високопродуктивних додатків: надання потужності майбутньому веб-розробки” [Електронний ресурс] – Режим доступу: https://javascript.org.ua/rishennya-node-js-dlya-visokoproduktivnih-dodatkiv-nadannya-potuzhnosti-majbutnomu-veb-rozrobki/
2. “Посібник для початківців по запитах до MongoDB за допомогою Compass: оволодіння запитами MongoDB та розуміння відмінностей із SQL” [Електронний ресурс] – Режим доступу: https://javascript.org.ua/posibnik-dlya-pochatkivcziv-po-zapitah-do-mongodb-za-dopomogoyu-compass-ovolodinnya-zapitami-mongodb-ta-rozuminnya-vidminnostej-iz-sql/
3. “Як використовувати JSON WebTokens (JWT) для аутентифікації” [Електронний ресурс] – Режим доступу: https://devzone.org.ua/post/iak-vykorystovuvaty-json-web-tokens-jwt-dlia-avtentyfikatsiyi

УДК 004.4

ПРОГРАМНА РЕАЛІЗАЦІЯ ПЛАТФОРМИ ДЛЯ ПРОДАЖУ МОТУЗКОВИХ ВИРОБІВ

Білотіл В.Е.
Західноукраїнський національний університет
бакалавр

Постановка проблеми

Ринок мотузкових виробів в Україні має значний потенціал, але часто виробники або продавці не мають ефективного онлайн-інструменту для презентації свого асортименту, залучення клієнтів та здійснення продажів. Існуючі торговельні майданчики можуть бути перевантажені іншими категоріями товарів, що ускладнює пошук специфічної продукції. Відсутність спеціалізованої онлайн-платформи, присвяченої виключно мотузковим виробам, обмежує можливості як для продавців, так і для потенційних покупців, які шукають конкретні типи мотузок, шнурів, канатів тощо. Розробка спеціалізованого вебсайту дозволить оптимізувати процес купівлі-продажу, розширити ринок збуту для виробників/продавців та спростити процес вибору та придбання для споживачів.

Мета роботи

Мета роботи полягає у аналізі поточного стану онлайн-ринку мотузкових виробів, виявленні потреб потенційних користувачів (продавців та покупців), аналізі функціоналу існуючих e-commerce рішень та розробці, на основі отриманих даних, повноцінного веб сайту для продажу мотузкових виробів, який би відповідав сучасним вимогам електронної комерції та максимально задовольняв потреби цільової аудиторії.

Програмна реалізація системи

Під час аналізу було розглянуто такі системи-аналоги, що спеціалізуються на продажу мотузкових виробів. Було виявлено спільні проблеми, які повторюються на більшості з них. Насамперед привертає увагу обмежений асортимент товарів. Більшість сайтів пропонують досить вузький вибір продукції, що ускладнює підбір виробів для різних цілей — від декоративного використання до технічного чи спортивного.

Крім цього, помітною є тенденція до використання неякісного візуального контенту. Фотографії часто не передають структуру матеріалу, деталі плетіння або реальний вигляд виробу при практичному застосуванні. Через це потенційний покупець не має повного уявлення про товар і може сумніватися в доцільності покупки.

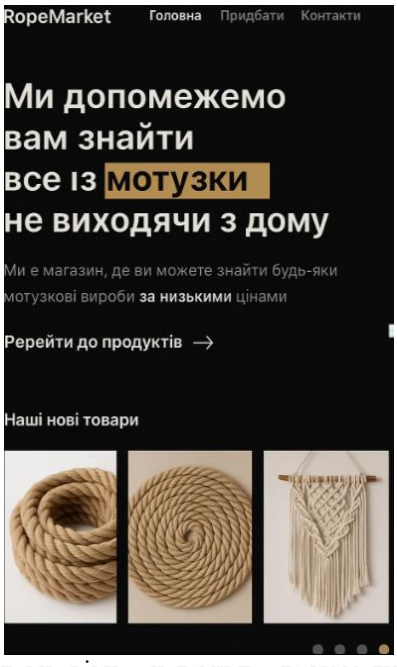


Рисунок 1 – Головне вікно системи продажу мотузкових виробів

Ще однією спільною проблемою є незручний інтерфейс користувача. Деякі сайти мають застарілий або надто складний дизайн, що ускладнює навігацію, пошук за параметрами і процес оформлення замовлення. Це безпосередньо впливає на показники відмов та знижує конверсію.

Також спостерігається брак детальної та актуальної інформації про товари. Часто не вказується діаметр мотузки, тип волокна, особливості плетіння чи допустиме навантаження. Це створює бар'єри для прийняття рішення про покупку, особливо у випадках, коли виріб використовується для відповідальних цілей — наприклад, у спорті чи будівництві.

Аналіз систем-аналогів показав, що розроблювана платформа має забезпечити базовий функціонал для користувачів і адміністраторів. Клієнти повинні мати змогу реєструватися, авторизуватись, переглядати каталог, оформлювати замовлення та залишати відгуки. Адміністрація повинна мати доступ до управління товарами, категоріями, замовленнями й відгуками. Система має включати генерацію звітів, базову аналітику, захист даних та контроль прав доступу.

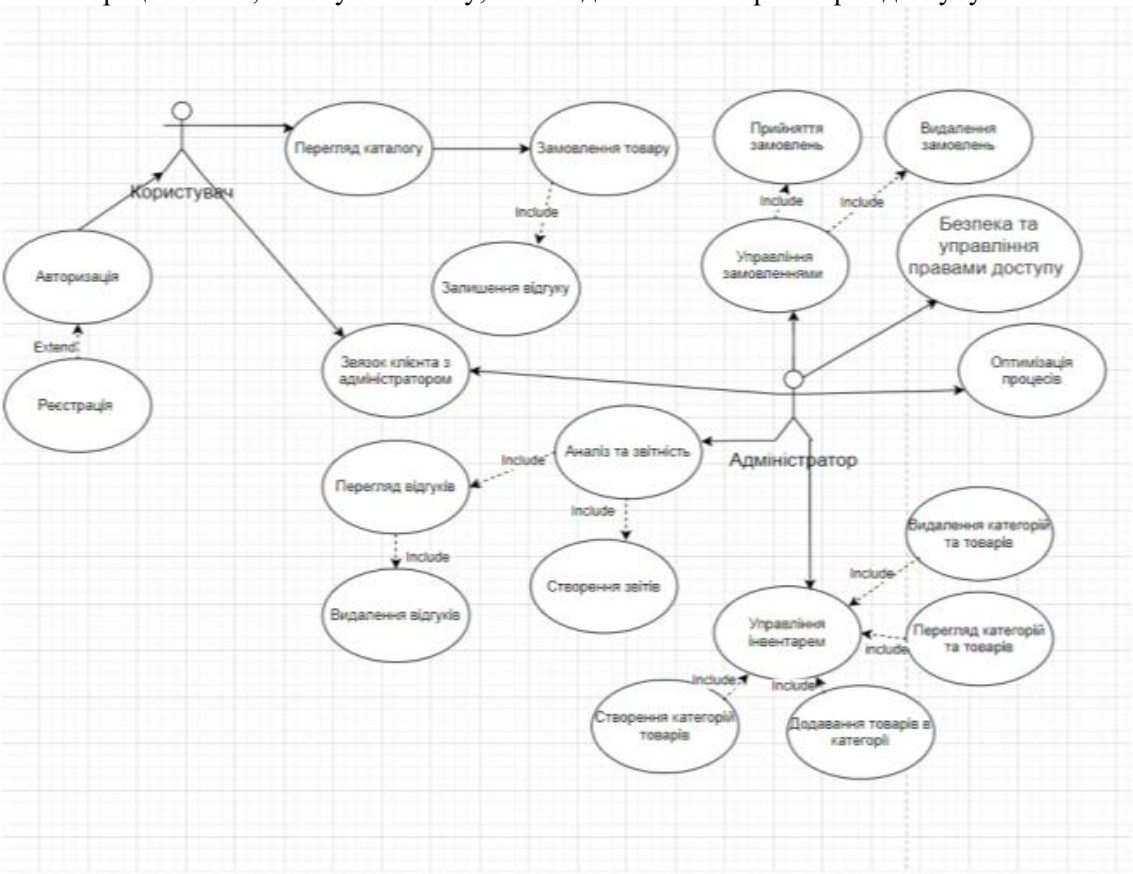


Рисунок 2 – Діаграма варіантів використання системи

Як показано на рисунку 2, користувач взаємодіє із системою через базові функції інтернет-магазину. Основними діями є перегляд каталогу товарів, оформлення замовлення та залишення відгуків після покупки. Для доступу до цих функцій користувач повинен пройти процедуру авторизації, яка, у свою чергу, передбачає можливість попередньої реєстрації. Також реалізована можливість зв'язку з адміністрацією сайту у разі виникнення питань або потреби в підтримці.

Висновок

У роботі було вирішено завдання зі створення спеціалізованого вебсайту для продажу мотузкових виробів, що стало відповіддю на відсутність ефективної онлайн-платформи для цієї ніші ринку. Основною метою була розробка функціонального інтернет-магазину. В ході роботи було реалізовано ключові можливості для клієнтів, включаючи реєстрацію, перегляд каталогу, оформлення замовлень та залишення відгуків. Паралельно створено адміністративну панель для повного управління товарами, замовленнями, користувачами та контентом сайту. В результаті розробки отримано готовий до використання вебсайт, який може слугувати ефективним інструментом електронної комерції для продажу мотузкових виробів.

Список використаних джерел

1. "Сучасні тенденції в e-commerce: Як створити успішний інтернет-магазин" [Електронний ресурс] – Режим доступу: https://ecommerce-experts.com.ua/trends

УДК 004.4

ІНТЕРНЕТ-ПЛАТФОРМА ТА ФОРУМ СОЦІАЛЬНИХ НОВИН ТА ДИСКУСІЙ

Крепич С.Я.[1)], Дяків Б.І.[2)], Крепич Р.В.[3)]
[1-2)]Західноукраїнський національний університет
[3)] ВСП Кам'янець-Подільський фаховий коледж НРЗВО «Кам'янець-Подільський державний інститут»
[1)] к.т.н., доцент; [2)]бакалавр; [3)] викладач

Постановка проблеми

Сучасні інтернет-форуми часто не відповідають вимогам інтерактивності, безпеки та зручності через застарілий дизайн, відсутність автоматизації модерації та обмежену інтеграцію з сучасними технологіями. Фрагментація контенту, недостатня персоналізація та низька ефективність пошуку обмежують залучення користувачів. Існує потреба у створенні інноваційної платформи, яка поєднає зручний інтерфейс, AI-генерацію контенту, гнучку систему сортування та надійний захист даних.

Мета роботи

Мета роботи – розробити та впровадити веб-форум із інтуїтивним інтерфейсом, автоматизованою генерацією постів за допомогою штучного інтелекту, системою сортування за популярністю та підвищеним рівнем безпеки. Платформа має забезпечити ефективну комунікацію, зменшити навантаження на модераторів та залучити користувачів через персоналізований функціонал.

Основна частина

Система побудована за клієнт-серверною моделлю, де:

- фронтенд (клієнтська частина) – відповідає за інтерфейс користувача (HTML, CSS, TypeScript, Angular).
- бекенд (серверна частина) – обробляє логіку, зберігає дані в БД та надає API (.NET, C#).
- REST API – використовується для комунікації між фронтендом і бекендом через HTTP-запити (GET, POST, PUT, DELETE).

Система сортування враховує кількість реакцій, коментарів та переглядів, що забезпечує швидкий доступ до актуальних тем. Безпека реалізована через хешування паролів, розмежування прав доступу та шифрування даних.

На рисунку 1 зображено діаграму класів розроблюваного форуму соціальних новин та дискусій.

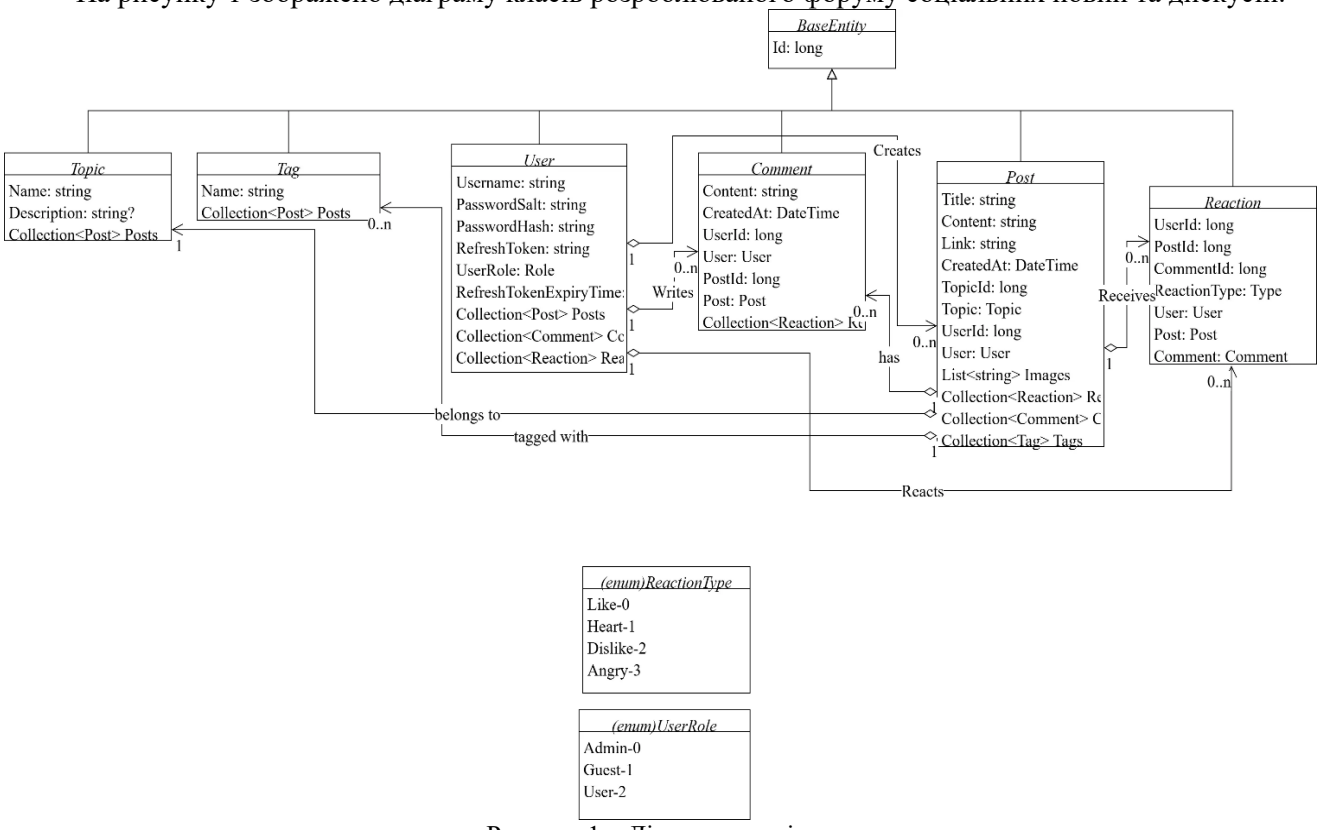


Рисунок 1 – Діаграма класів

Система автоматично створює пости на основі заголовків, використовуючи модель GPT-4. Реалізація складається з таких ключових компонентів:

OpenAiSettings

Визначає шаблон запиту до AI, включаючи роль ("Content-maker") та список тем (спорт, технології тощо). Запит формується у вигляді JSON-об'єкта для стандартизації відповіді.

Приклад коду:

```
public class OpenAiSettings : IOpenAiSettings
{
   public string GetPostPrompt(string title)
   {
      var role = "Content-maker";
      var topics =
         "TopicId 1 = Sport" +
         "TopicId 2 = Technology" +
         "TopicId 3 = Self-Development" +
         "TopicId 4 = Health & Wellness" +
         "TopicId 5 = Finance" +
         "TopicId 6 = Education" +
         "TopicId 7 = Travel" +
         "TopicId 8 = Productivity" +
         "TopicId 9 = Books & Literature" +
         "TopicId 10 = Entertainment";
      return $@"You are {role}. Your goal is to write content for the post based on its title: '{title}'.
            Select the most appropriate topic from the following list (by TopicId) and generate tags for it:
            {topics}.
            Provide the result in the form of a JSON object. An example of JSON object:
            {{
               ""Content"": ""Sample content based on the title"",
               ""TopicId"": 2,
               ""Tags"": [""tag1"", ""tag2""]
            }}";
   }
}
```

PostGenerator

Відповідає за виклик AI та перетворення відповіді в DTO для створення поста. Використовує AutoMapper для мапування даних.

Приклад коду:

```
public async Task<CreatePostDto> GeneratePost(CreatePostOnlyTitleRequestDto createPostOnlyTitleRequestDto)
{
   ValidatePostForGenerateParametres(createPostOnlyTitleRequestDto);
   string prompt = this.openAiSettings.GetPostPrompt(createPostOnlyTitleRequestDto.Title);
   var response = await this.GenerateTextResponse(prompt);
   var postResponse = JsonConvert.DeserializeObject<OpenAiGenerateResponseDto>(response);
   var createPostDto = this.mapper.Map<CreatePostDto>(postResponse);
   createPostDto.Title = createPostOnlyTitleRequestDto.Title;
   return createPostDto;
}
```

CompletionService

Безпосередньо взаємодіє з OpenAI API, використовуючи модель GPT-4. Обробляє помилки та логує невдалі спроби.

Приклад коду:

```
public async Task<string> CreateCompletionAsync(string prompt)
{
   ValidatePrompt(prompt);
   var chat = this.openAiApi.Chat.CreateConversation();
   chat.Model = Model.GPT4;
   chat.AppendUserInput(prompt);
   string? response = await chat.GetResponseFromChatbotAsync();
```

```
            if (response == null)
            {
                this.logger.LogError("Chatbot response is null");
                throw new InvalidOperationException("Chatbot response is null");
            }
            return response;
        }
```

Переваги реалізації

- Стандартизація: Дані повертаються у JSON-форматі, що спрощує їх обробку.
- Гнучкість: Можливість легко змінити модель AI (наприклад, на GPT-3.5) або додати нові теми.
- Безпека: валідація вхідних даних запобігає помилкам (наприклад, пустий заголовок).

Діаграма взаємодії компонентів подана на рисунку 2

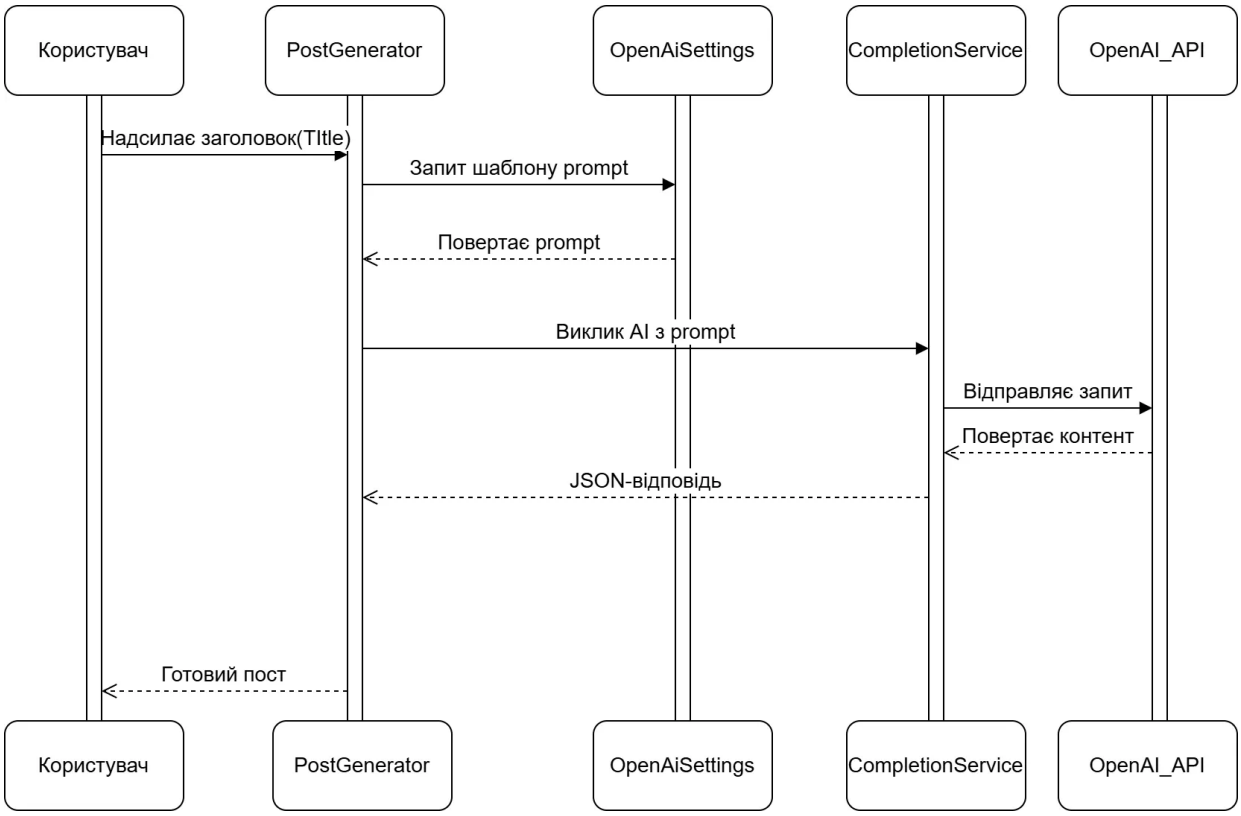


Рисунок2- Діаграма взаємодії компонентів

Висновок

Розроблений форум інтегрує сучасні технології для підвищення якості комунікації та зручності користувачів. Впровадження AI-генерації контенту та гнучкого сортування сприяє активізації спільноти. Подальші дослідження спрямовані на вдосконалення алгоритмів штучного інтелекту та розширення мультимедійних функцій.

Список використаних джерел

1. “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding” [Електронний ресурс] – Режим доступу: https://arxiv.org/abs/1810.04805
2. Крепич С.Я. Моделювання та забезпечення функціональної придатності статичних систем методами аналізу інтервальних даних. / С.Я. Крепич // дис. канд. техн. наук, НУ «Львівська політехніка», Львів, 2016. – 166с.
3. Smith J., “Modern Web Development with .NET and TypeScript”, Tech Innovations Journal, 2023. – с. 78-85.
4. OpenAI, “GPT-4 Technical Report” [Електронний ресурс] – Режим доступу: https://cdn.openai.com/papers/gpt-4.pdf
5. Microsoft, “REST API Design Guidelines” [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design
6. Spivak, I., Krepych, S., Litvynchuk, M. and Spivak, S. “Validation and data processing in JSON format”, *IEEE EUROCON 19th International Conference on Smart Technologies*, 2021. pp.326–330
7. Microsoft, "ASP.NET Core документація" [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/uk-ua/aspnet/core/
8. Poliarush O., Krepych S., Spivak I., “Hybrid approach for data filtering and machine learning inside content management system”, Advanced Information Systems , 2023, 7(4), pp.70-74

УДК 004.4

ПРОГРАМНА СИСТЕМА ДЛЯ КАУЧСЕРФІНГУ

Співак І.Я.[1)] , Устиченко О.О.[2)]
Західноукраїнський національний університет
[1] к.т.н., доцент; [2)] бакалавр

І. Постановка проблеми

Пошук місця ночівлі є важливою та невід'ємною частиною міграції, подорожі чи поїздки людини чи групи людей. Особливо це питання актуальне для молоді, яка прагне подорожувати з найменшими витратами. Крім того каучсерфінг– це спосіб знаходження безкоштовного житла у мешканців міста.

На сьогодні існують окремі платформи, які надають такі послуги, але чи то з обмеженою функціональністю, чи за платний доступ, або без підтримки користувача. Також деякі з них не надають розкіш швидкості комунікації між гостем та хостом.

Отже, наявність простої та зручної програмної системи є важливою потребою для населення. Можливість швидкого пошуку місць для ночівлі та додавання інформації про доступне житло стає особливо актуальною в умовах подорожей, переїздів або тимчасового проживання.

ІІ.Мета роботи

Метою роботи є створення програмної системи каучсерфінгу, яка дозволяє користувачам обирати чи пропонувати місце для ночівлі, реєструватися в системі, переглядати профіль. Також головною метою є створення дуже зручного та простого інтерфейсу для того, щоб будь-який користувач мав можливість швидко зрозуміти як користуватися додатком.

ІІІ. Основна частина

Каучсерфінг базується на гостьовій взаємодопомозі. Це мережа гостинності, вкотрій люди безкоштовно дають тимчасовий прихисток для ночівлі. Головна ціль каучсерфінгу полягає в тому, що один користувач пропонує тимчасове вільне місце для ночівлі іншим користувачам, а інший користувач шукає місце для ночівлі під час міграції, подорожі чи поїздки. Цей процес відбувається за власним бажанням та добровільною згодою обох сторін, зазвичай це не обходиться без культурного обміну.

Потреба в такій системі зростає, оскільки багато людей подорожують з обмеженим бюджетом, а типові місця для ночівлі, такі як готелі, мотелі чи хостели, переважно є дорогими.

Формування вимог до системи. Програмна система для каучсерфінгу володіє таким функціональними:

- Реєстрація користувача
- Авторизація користувача
- Пошук місць ночівлі
- Пропонування місць ночівля

та нефункціональними вимогами:

- Зрозумілий інтерфейс
- Стабільна робота системи

Дана програмна система має трирівневу клієнт-серверну архітектуру, яку проілюстровано на рисунку 1. Принцип роботи полягає в тому, що декілька серверів обробляють запит клієнта. Розподіл операцій знижує навантаження на сервер[1].

Клієнтська частина застосунку побудована за допомогою мови програмування QML. QML(*QtMeta Language* або *QtModeling Language*) - декларативна мова програмування, заснована наJavaScripti призначена для розробкизастосунків, які роблять основний наголос накористувацький інтерфейс[2]. Серверна частина побудована за допомогою мови програмування C++ та фреймворку QtCreator. Її зміст полягає в взаємодії з сервером бази даних MongoDB, яка зберігатиме всі дані. Сервер обробляє запити від клієнта, виконує певні операції та надсилає у відповідь отриманий результат. Для обміну даними використовується формат JSON. JSON (JavaScript Object Notation) – це текстовий формат, призначений для зберігання структурованих даних[3]. Клієнтська частина має змогу працювати разом із сервером як на одному комп'ютері, так і віддалено, що дозволяє розгорнути систему на кількох пристроях.

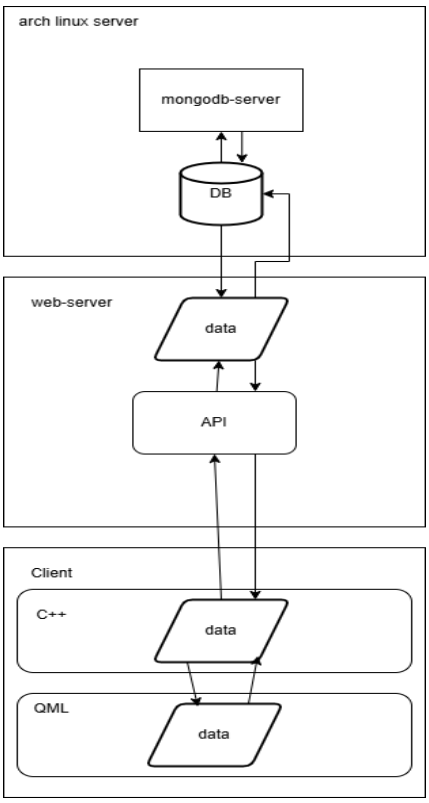


Рисунок 1 – Архітектура системи

Структуру бази даних зображено на рисунку 2 у вигляді ER – діаграми.

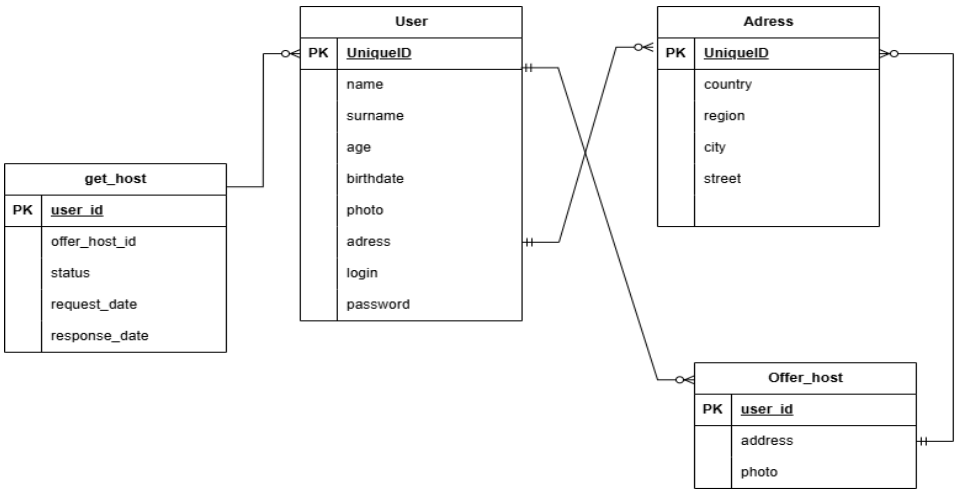


Рисунок 2 –ER – діаграма

Згідно спроектованої архітектури, було розроблено основні модулі:

- Модуль реєстрації: виконує реєстрацію нового користувача.
- Модуль авторизації: виконує вхід користувача в систему.
- Модуль пошуку: виконує пошук за містом.
- Модуль створення місця ночівлі: виконує створення нового місця ночівлі.
- Модуль отримання місця ночівлі: виконує пропонування місця ночівлі.

Висновок

В результаті роботи було досліджено підхід для створення крос платформного застосунку для пошуку місця ночівлі. Було здійснено повний цикл створення програмного продукту: починаючи від аналізу предметної області до проектування архітектури та структури бази даних і до програмної реалізації. Таким чином поставлені завдання були успішно виконані та розроблена система відповідає технічним вимогам.

Список використаних джерел

1. Клієнт-серверна архітектура. Онлайн-курси від компанії QATestLab [Електронний ресурс] – Режим доступу: https://training.qatestlab.com/blog/technical-articles/client-server-architecture/
2. Учасники проектів Вікімедіа. QML – вікіпедія. [Електронний ресурс] – Режим доступу:https://uk.wikipedia.org/wiki/QML
3. Що таке JSON | Навіщо потрібен цей формат. Apix-Drive. [Електронний ресурс] – Режим доступу: https://apix-drive.com/ua/blog/useful/scho-take-json

УДК 004.4

АЛГОРИТМІЧНА ІНФРАСТРУКТУРА СИСТЕМИ ДЛЯ РЕКОМЕНДАЦІЇ ВАКАНСІЙ НА ОСНОВІ АНАЛІЗУ ТЕКСТОВОЇ ІНФОРМАЦІЇ

Співак І.Я.[1)], Матковський М.О.[2)], Крепич С.Я.[3)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] магістрант; [3)]к.т.н., доцент

I. Постановка проблеми

Зростаючий обсяг інформації на ринку праці вимагає нових підходів до автоматизованого пошуку відповідних вакансій. Одним із таких підходів є застосування системи рекомендацій, яка дозволяє не лише зіставити резюме з наявними вакансіями, але й пояснити вибір системи. Така система повинна враховувати численні фактори: від семантичного контексту до статистичних зв'язків між кандидатами і роботодавцями.

II. Мета роботи

Метою роботи є побудова багаторівневої алгоритмічної інфраструктури, що дозволяє здійснювати інтелектуальний аналіз текстів вакансій і резюме з подальшим формуванням точних рекомендацій. Основна увага приділяється створенню модулів, які можуть бути інтегровані в існуючі системи і забезпечувати лінгвістичну обробку, семантичне зіставлення та фільтрацію на основі заданих параметрів. Система має забезпечити високий рівень персоналізації, швидку реакцію на зміни у вхідних даних та надання детальних звітів про рівень релевантності між кандидатами та вакансіями і навпаки.

III. Архітектурні рішення та інтеграція

Розроблена система має модульну архітектуру, що дозволяє легко масштабувати та адаптувати її під потреби різних цифрових платформ. Кожен компонент виконує чітко визначену функцію і може розвиватись незалежно, що значно спрощує підтримку та оновлення.

Архітектура передбачає використання принципів розділення відповідальності (Separation of Concerns), що дає змогу ізолювати лінгвістичну, обчислювальну та інтерфейсну частини. Комунікація між компонентами реалізована через стандартизовані інтерфейси, що підтримують передачу структурованих даних у форматі JSON або XML.

З метою забезпечення гнучкості та повторного використання, ключові обчислювальні блоки побудовані як бібліотеки з можливістю розгортання в окремому середовищі або включення до складу існуючого сервісу. Передбачено можливість масштабування через контейнеризацію (Docker) та автоматизацію розгортання у хмарному середовищі.

Окремий модуль відповідає за інтеграцію із зовнішніми сервісами: він підтримує обмін даними, а також адаптує інформацію до внутрішніх форматів системи. Це дозволяє системі ефективно взаємодіяти із зовнішніми базами даних, порталами пошуку роботи, аналітичними платформами та іншими цифровими ресурсами.

IV. Опис компонентів

В основі запропонованої системи лежить ідея багатокрокової обробки вхідних даних, у якій кожен компонент виконує окрему функцію, водночас формуючи єдиний аналітичний цикл. Виходячи з необхідності високоточної відповідності між вакансіями та резюме, було запропоновано логічне поділення задачі на етапи: від лінгвістичного аналізу текстів до побудови метрик подібності та фінального формування персоналізованих результатів. Такий підхід забезпечує як модульність, так і аналітичну глибину, дозволяючи масштабувати систему під різні потреби.

Для досягнення цілей, поставлених у межах даного дослідження, система була структурована у вигляді набору взаємопов'язаних функціональних компонентів. Кожен з них виконує специфічну роль у загальному процесі аналізу та рекомендації, забезпечуючи послідовну обробку вхідних даних — від їх попередньої лінгвістичної трансформації до генерації звітів про релевантність. Нижче наведено короткий опис основних логічних модулів системи:

- Компонент лінгвістичної обробки проводить глибоку морфологічну та лексичну обробку текстів вакансій та резюме. Особливу роль відіграє лематизація — приведення слів до початкової форми, що дозволяє враховувати варіативність мови, уникати дублювань і підвищувати якість пошуку.

- Компонент семантичного зіставлення формує векторні представлення текстів за допомогою методів TF-IDF, Word2Vec або BERT і порівнює їх на основі косинусної подібності. Це дозволяє визначити не лише явні збіги, а й приховану семантичну близькість між вакансією та профілем кандидата.

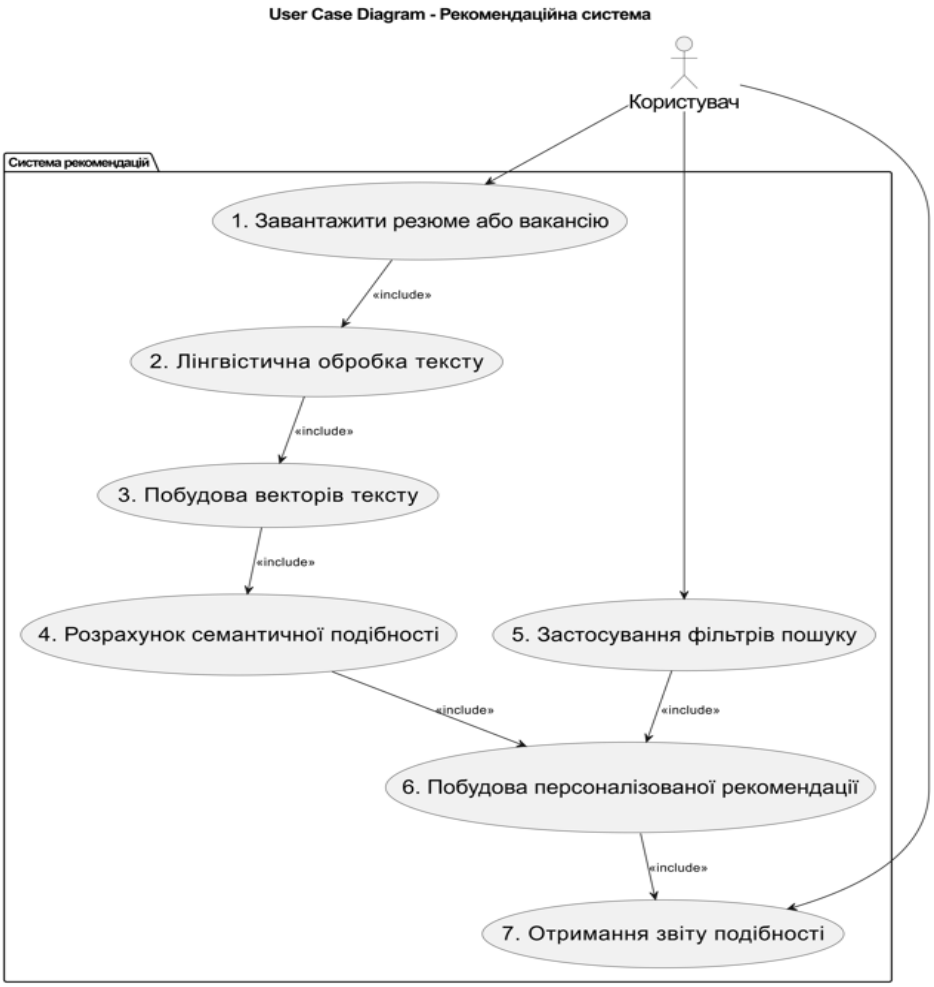


Рисунок 1 – User Case Diagram системи рекомендацій

- Компонент фільтрації та персоналізації дає змогу користувачу налаштовувати запити за фільтрами (локація, рівень заробітної плати, тип зайнятості) і враховує історію його активності для точнішої побудови рекомендацій.
- Компонент інтеграції слугує посередником між системою та зовнішнім середовищем. Завдяки адаптованим інтерфейсам, він забезпечує передачу даних у стандартизованому вигляді, у тому числі у форматах JSON, XML або через RESTful API.
- Компонент звітності генерує текстові та графічні звіти, які демонструють рівень подібності між вакансією та резюме. Такі звіти містять пояснення, які саме параметри збіглися: досвід, навички, ключові слова, тип компанії тощо.

Висновок

Запропонована система рекомендацій — це масштабований і адаптивний інструмент, який поєднує в собі алгоритмічну гнучкість, семантичну точність та практичну інтегровність. Її використання дозволяє значно підвищити якість взаємодії між кандидатами та роботодавцями, зменшити час на пошук, а також покращити прозорість і контроль у процесі відбору кадрів.

Список використаних джерел

1. Spivak I., Krepych S., "Recursive method of forming an optimal set of tasks according to the criterion of maximizing earnings", Advanced Information Systems, 2024. 8(3). pp.72-76
2. Krepych S., Spivak I., "Improvement of SVD algorithm to increase the efficiency of recommendation systems", Advanced Information Systems, 2021. 5(4). pp.55-59
3. Джерард К., "AI and Machine Learning for Coders", O'Reilly Media, 2020.
4. Петренко О.М., "Штучний інтелект і нейромережі", Київ: Наукова книга, 2019.
5. Poliarush O., Krepych S., Krepych I., "Hybrid approach for data filtering and machine learning inside content management system", Advanced Information Systems, 2023. 7(4). pp.70-74
6. Krepych S., Spivak I., "Forecasting system of utilities service costs based on neural network", Advanced Information Systems, 2020. 4(4). pp.102-108
7. Spivak I., Krepych S., Porplytsya N., Spivak S. "Method of estimation the level of influence of motivational factors on labor efficiency", *2020 IEEE 15th International Scientific and Technical Conference on Computer Sciences and Information Technologies*, CSIT 2020 – Proceedings, 2020, 1, pp.159-162
8. Spivak I.Ya., Krepych S.Ya. Horishniy V.I. "Organization of CLOUD architecture for systems ensuring the functional suitability of static systems". Scientific journal "Modern Information Systems", Kharkiv, Volume 3, No. 2, 2019. - pp. 35-39
9. Співак І.Я., Крепич С.Я. "Оцінювання рівня впливу мотивації праці на ефективність діяльності", Інформаційні технології та комп'ютерна інженерія, №3, 2020, С. 22-29

УДК 004.4

СОЦІАЛЬНА ПЛАТФОРМА МІКРОБЛОГІВ З ФУНКЦІЄЮ АВТОМАТИЧНОГО ВИДАЛЕННЯ КОМЕНТАРІВ

Порплиця Н.П.[1)], Сімашко І.В.[2)]
Західноукраїнський національний унівеситет
[1)]к.т.н, доцент; [2)]студент

І. Постановка проблеми

Соціальні мережі є важливою складовою сучасного суспільства, оскільки вони об'єднують людей, сприяють обміну ідеями та створюють простір для самовираження. Однак разом із цим вони стикаються з численними викликами, серед яких ключове місце займає управління контентом. Однією з найактуальніших проблем є модерація коментарів, які можуть включати небажані висловлювання, спам або нецензурний контент.

Під терміном «модерація» розуміють процес оцінювання та управління вмістом, який створюється і публікується користувачами, з метою забезпечення його відповідності встановленим правилам та стандартам платформи. Цей процес є критично важливим для підтримки здорового середовища у спільнотах та запобігання поширенню образливих, неприпустимих або шкідливих матеріалів.

Сьогодні існує багато підходів до вирішення проблеми модерації коментарів. До них належить ручна модерація, яка передбачає перевірку контенту модераторами, та автоматичне фільтрування за ключовими словами. Проте ці підходи мають свої недоліки: ручна модерація є трудомісткою та залежить від суб'єктивності модератора, тоді як автоматичні фільтри можуть допускати помилкові блокування.

Крім того, спостерігається попит на інструменти, які надають користувачам більше контролю над своїм контентом і дозволяють їм встановлювати власні правила модерації. Це може включати функції налаштування критеріїв фільтрації, автоматичного приховування певних типів коментарів, або навіть введення правил для автоматичного видалення коментарів через заданий період часу.

Впровадження таких інструментів не лише допоможе користувачам почуватися більш комфортно та впевнено на платформі, але й сприятиме створенню більш персоналізованого досвіду. Крім того, це дозволить платформам забезпечити більш ефективне управління контентом, відповідно до потреб кожного користувача. Таким чином, підвищення зручності використання та зміцнення контролю над контентом стане вагомим кроком у напрямку розвитку соціальних мереж.

ІІ. Мета роботи

Метою даного дослідження є розробка прототипу соціальної платформи мікроблогів, яка надає користувачам можливість автоматичного видалення коментарів після заданого ними часу. Такий функціонал спрямований на забезпечення більшої гнучкості в управлінні контентом, що дозволить користувачам самостійно контролювати тривалість доступності своїх коментарів.

Цей підхід не лише покращує зручність і простоту використання платформи, але й сприяє підвищенню відповідальності за публікації, даючи користувачам більше інструментів для керування своїм контентом. Крім того, впровадження цієї функції допоможе зменшити навантаження на модераторів платформи, оскільки частина управління контентом буде передана самим користувачам, що оптимізує процес модерації загалом.

ІІІ. Основна частина

Функція автоматичного видалення коментарів реалізується через інтеграцію таймерів, які користувач може налаштувати при створенні або редагуванні коментаря. Після завершення встановленого часу коментар автоматично видаляється з бази даних. На відміну від підходів, які передбачають аналіз тексту чи виявлення ключових слів, запропонована функція є максимально простою та прозорою для користувача.

Розроблений прототип включає наступні ключові компоненти:

- Модуль користувацького інтерфейсу для налаштування таймера видалення коментарів. Інтерфейс створено таким чином, щоб він був інтуїтивно зрозумілим і доступним для користувачів.

- Серверну логіку для обробки запитів на видалення після завершення вказаного часу. Таймери виконуються у фоновому режимі, що забезпечує безперебійну роботу платформи.
- Базу даних для збереження інформації про коментарі та їхні таймери. Використання PostgreSQL дозволило досягти високої швидкості роботи із запитами та надійності збереження даних.

Технології, використані для розробки прототипу, включають мову програмування Ruby, фреймворкRubyonRails, а також PostgreSQL для збереження даних. Особливу увагу було приділено тестуванню та оптимізації, щоб мінімізувати затримки у видаленні коментарів та забезпечити високу продуктивність системи.

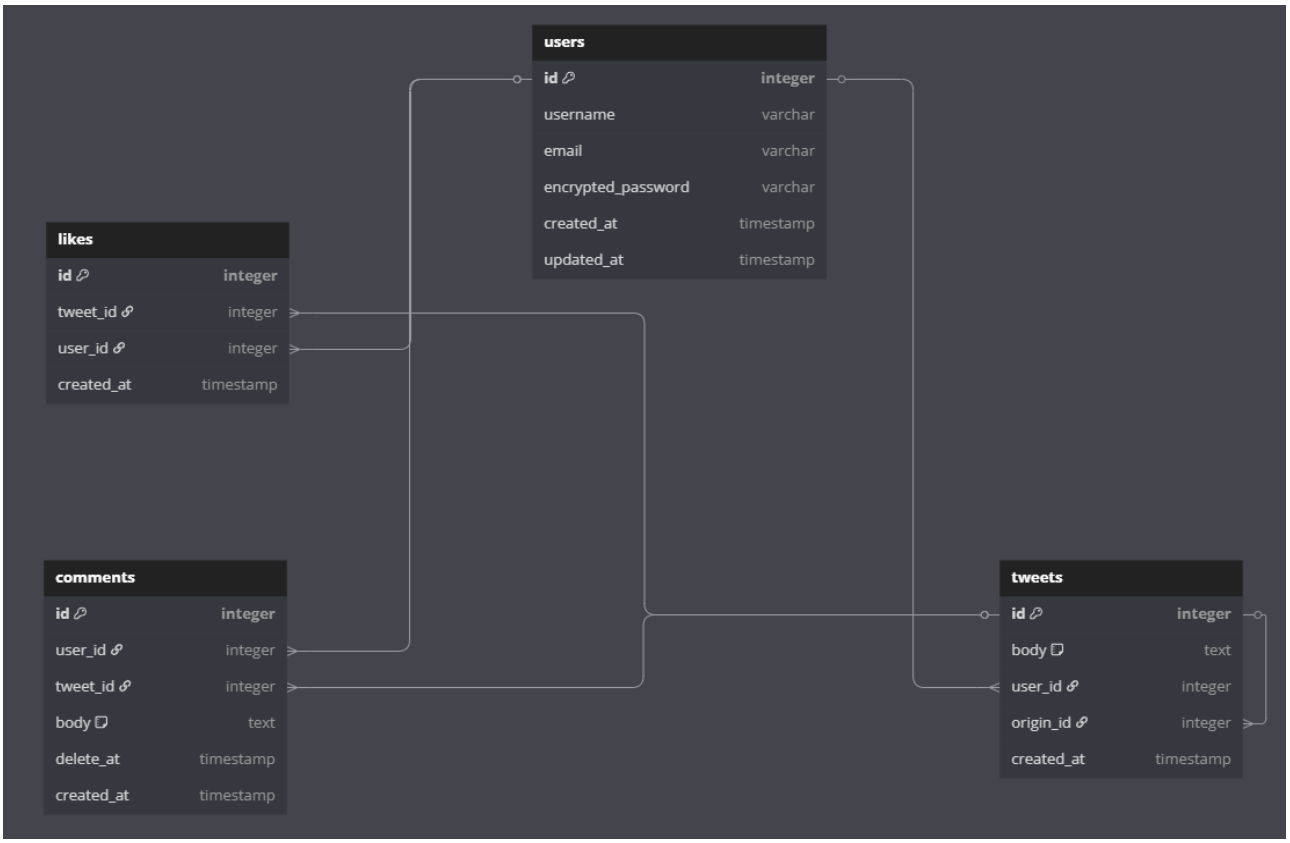


Рисунок 1 – Схема бази даних

Під час тестування прототипу були використані сценарії, що імітують різні способи взаємодії користувачів із платформою. Наприклад, створення коментарів із різними налаштуваннями часу видалення, зміна цих налаштувань перед видаленням і перевірка їх виконання у встановлений час. Результати тестування підтвердили коректність роботи прототипу, його стійкість до помилок і швидкість виконання запитів.

IV. Висновок

У результаті роботи створено прототип соціальної платформи мікроблогів, що дозволяє користувачам самостійно визначати час видалення коментарів. Це підвищує гнучкість управління контентом та сприяє покращенню користувацького досвіду. Запропоноване рішення є простим в реалізації та не потребує складних алгоритмів для аналізу тексту, що робить його доступним для впровадження навіть на платформах із обмеженими ресурсами. У майбутньому функціонал може бути розширено, наприклад, інтеграцією додаткових опцій для більш детального керування контентом.

Список використаних джерел

1. Ruby on Rails Guides [Електронний ресурс] – Режим доступу: https://guides.rubyonrails.org/
2. Postgre SQL Documentation [Електронний ресурс] – Режим доступу: https://www.postgresql.org/docs/
3. "Дослідження впливу соціальних мереж на наше життя та мислення" [Електронний ресурс] – Режим доступу: https://drukarnia.com.ua/articles/doslidzhennya-vplivu-socialnikh-merezh-na-nashe-zhittya-ta-mislennya-013_6

УДК 004.4

МОДУЛЬ УПРАВЛІННЯ ДАНИМИ МАГАЗИНУ-ВИСТАВКИ

Стасів І.С.[1)], Атаманчук Ю.В.[2)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] бакалавр

І. Постановка проблеми

У сучасному бізнес-середовищі рівень автоматизації внутрішніх процесів є важливим чинником для забезпечення конкурентоспроможності підприємств роздрібної торгівлі. Системи, що підтримують ефективне управління інформаційними ресурсами, дозволяють значно знизити витрати, підвищити продуктивність та зменшити ймовірність помилок. Особливо актуальна ця проблема для спеціалізованих торгових точок, таких як магазини-виставки, де важливо не тільки забезпечити широкий асортимент товарів, але й швидко і точно управляти даними, що стосуються продукції: від залишків товару до інформації про поставки і продажі.

Магазини-виставки чайної продукції мають специфічні вимоги: величезний асортимент сортів чаю, різноманітні постачальники, сезонні оновлення товарів, а також необхідність проведення консультацій з клієнтами і демонстрації зразків продукції. Це вимагає наявності оперативного доступу до актуальної і точної інформації. В умовах, коли облік ведеться вручну або з використанням загальних інструментів, таких як Excel, виникає безліч обмежень: неможливість забезпечити багатокористувацький доступ, проблеми з масштабованістю, збереженням даних та їх безпекою.

Тому, є потреба у розробці спеціалізованого програмного забезпечення, яке дозволить централізовано і ефективно управляти даними магазину, мати простий інтерфейс для персоналу та забезпечити можливість розвитку та розширення функціональності в майбутньому.

ІІ. Мета роботи

Метою роботи є розробка модуля управління даними магазину-виставки чайної продукції, який дозволить здійснювати повноцінний електронний облік усіх ключових аспектів торговельної діяльності. Зокрема, система повинна забезпечувати ефективне управління асортиментом товарів, збереження інформації про постачальників, реєстрацію клієнтів, фіксацію продажів, а також контроль залишків на складі.

Одним із головних завдань є створення веб-застосунку, який буде доступний через браузер і дозволить користувачам взаємодіяти із системою у зручному графічному інтерфейсі. Інтерфейс має бути інтуїтивно зрозумілим, адаптивним до різних пристроїв (настільні ПК, планшети, ноутбуки) та підтримувати основні операції — перегляд, додавання, редагування і видалення даних. Загалом, мета роботи полягає у створенні програмного продукту, який відповідає вимогам сучасної торговельної практики, є масштабованим, надійним, зручним у користуванні та легко інтегрується у поточну організаційну структуру магазину.

ІІІ. Основна частина

Система управління даними магазину-виставки реалізована на основі клієнт-серверної архітектури. Такий підхід дозволяє розділити функціональність між клієнтським (frontend) та серверним (backend) компонентами, що забезпечує масштабованість і зручність для подальшого розвитку системи. Можливість використання цієї архітектури також дає змогу інтегрувати мобільні додатки або підключати сторонні сервіси, що значно розширює функціональність системи.

Клієнтська частина розроблена на базі JavaScript із застосуванням HTML5 та CSS3, що дозволяє зручно взаємодіяти з користувачем через інтуїтивно зрозумілий графічний інтерфейс у браузері. Це дає змогу створювати адаптивний дизайн, що забезпечує коректне відображення на різних пристроях, від настільних ПК до мобільних планшетів.

Серверна частина реалізована на платформі Node.js із використанням популярного фреймворку Express.js. Вона відповідає за обробку запитів від клієнта, взаємодію з базою даних і надсилання результатів у форматі JSON. Для зберігання даних обрано MongoDB – документно-орієнтовану NoSQL базу даних, яка дає можливість гнучко працювати з великою кількістю інформації. Усі основні елементи системи, такі як товари, постачальники, клієнти та продажі, зберігаються у вигляді колекцій документів, що дозволяє швидко додавати нові елементи або змінювати існуючі.

Особливу увагу було приділено проектуванню даних і моделюванню предметної області. Основними сутностями в системі є товари, категорії, замовлення та клієнти, для кожної з яких створено відповідну модель з характеристиками та атрибутами. Для кращого розуміння взаємозв'язків між класами та об’єктами було розроблено діаграму класів у стилі UML, що чітко показує структуру системи і зв’язки між її компонентами. Це дозволяє ефективно проектувати і реалізовувати систему, забезпечуючи її належну працездатність та можливість масштабування.

Такий підхід дозволяє легко розширювати систему, зменшувати навантаження на сервер за допомогою вертикального і горизонтального масштабування, а також забезпечувати можливість оновлення окремих компонентів без зупинки роботи всієї системи. Розгортання клієнтської та серверної частини можна здійснити на різних хостингах або через хмарні сервіси, що додає гнучкості та знижує витрати на підтримку інфраструктури.

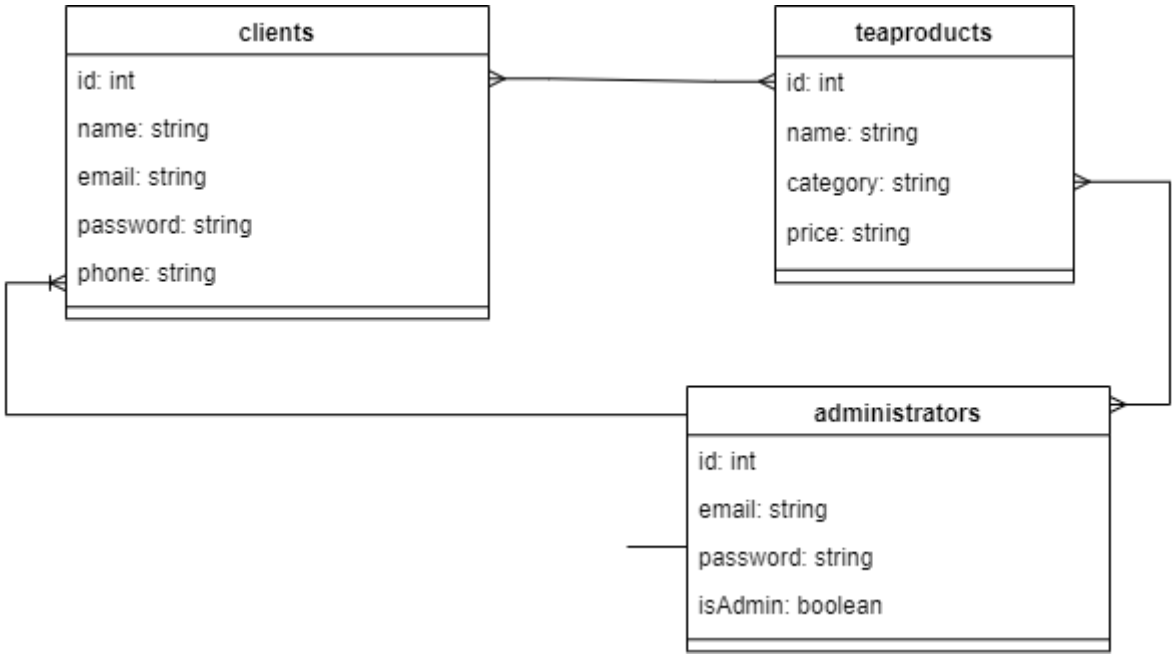


Рисунок 1 – Діаграма класів

Висновок

Розроблений модуль управління даними магазину-виставки чайної продукції успішно виконує поставлені завдання щодо автоматизації ключових бізнес-процесів, що виникають у щоденній діяльності торговельної точки. Завдяки впровадженню цієї системи вдалося досягти суттєвих покращень у сфері ведення обліку, контролю товарних залишків, управління замовленнями, а також взаємодії з постачальниками та клієнтами. Зокрема, точність обліку значно зросла завдяки автоматизованому введенню та збереженню даних про кожну товарну одиницю, постачання, продаж або повернення. Всі зміни фіксуються в системі в режимі реального часу, що дозволяє уникати помилок, пов’язаних з ручною обробкою або дублюванням інформації. Крім того, система сприяє підвищенню загальної ефективності роботи магазину. Користувачі отримують доступ до зрозумілого та функціонального інтерфейсу, що полегшує навчання нових працівників та зменшує кількість помилок у процесі роботи. Менеджмент має змогу своєчасно реагувати на зміни в попиті, проводити аналіз продажів, оптимізувати запаси та приймати обґрунтовані управлінські рішення.

Таким чином, впровадження модуля управління даними забезпечує інформаційну підтримку управління магазином, покращує координацію дій між працівниками та дозволяє зосередити увагу на стратегічному розвитку бізнесу, а не на рутинних адміністративних процесах.

Список використаних джерел

1. Братко О.І. Проєктування інформаційних систем. – К.: Видавництво Ліра-К, 2020.
2. Сидоренко В.Ю. Основи розробки баз даних. – Харків: ХНУРЕ, 2019.
3. Левченко І.А. Автоматизація обліку товарів у сфері торгівлі // Науковий вісник КНУТД. – 2022. – №4. – С. 34–38.
4. Грищенко С.О. Технології створення веб-застосунків. – К.: НАУ, 2021.
5. Freeman E., Robson E. HeadFirstDesignPatterns. — O'ReillyMedia, 2021.
6. Express.js Guide. [Електронний ресурс] - Режим доступу: https://expressjs.com/https://expressjs.com
7. BootstrapDocumentation. [Електронний ресурс] - Режим доступу:https://getbootstrap.com

УДК 004.4

ЗАСТОСУНОК ДЛЯ ДОГЛЯДУ ЗА РОСЛИНАМИ

Співак І.Я.[1)], Сворінь Д.В.[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)]бакалавр

Постановка проблеми

У повсякденному житті догляд за рослинами може відійти на другий план. Внаслідок цього виникає ризик того, що людина забуде про важливі процедури догляду, необхідні для здоров'я та життя рослин. Підтвердженням цього є популярність мобільних застосунків, які допомагають користувачам своєчасно доглядати за рослинами. Наприклад застосунок Plant Parent, який має більше десяти мільйонів завантажень та середню оцінку 4,5 з 5 зірок при більш ніж 78 тисяч відгуків [1].

Оптимальним рішенням у такій ситуації є розробка мобільного застосунку, який буде своєчасно надсилати нагадування користувачу. Застосунок функціонує на мобільній платформі Android, яка є найпоширенішою операційною системою серед користувачів смартфонів. Станом на квітень 2025 року, Android має 72,23% частки ринку мобільних операційних систем, що значно перевищує частку iOS у 27,39% [2].

Android розпочався у 2003 році як проєкт американської технологічної компанії Android Inc. з розробки операційної системи для цифрових камер. У 2004 році проєкт змінився на операційну систему для смартфонів. Android Inc. була придбана американською компанією-розробником пошукових систем Google Inc. у 2005 році. У Google команда Android вирішила базувати свій проєкт на Linux, операційній системі з відкритим кодом для персональних комп'ютерів [3].

Мета роботи

Метою роботи є аналіз розробленого мобільного застосунку на базі операційної системи Android, а саме архітектури, візуальної частини та функціоналу програми.

Основна частина

На ринку сучасних Android-застосунків існує велика кількість програм які пов'язані із рослинами. На даний момент особливою популярністю користуються програми із використанням штучного інтелекту, які можуть розпізнати рослину за фотографією. Але через їх велику кількість, буває складно знайти саме ту програму, яка допомагає саме із обліком та сповіщеннями про догляд за рослинами.

Серед популярних програм, які реалізують потрібний функціонал, можна виділити кілька наступних: Plant Parent: Plant Care Guide, Planta та Plantora – Plant Identify, Care. Усі ці програми є досить великими та багатофункціональними застосунками із постійною підтримкою з боку розробників та великою кількістю постійних користувачів.

Проте головною проблемою цих програм для догляду за рослинами є відсутність можливості працювати в режимі Offline, тобто без доступу до інтернету. Крім того, під час розробки власного застосунку, варто взяти приклад із популярних програм та звернути увагу на дотримання сучасним вимогам дизайну, оскільки це покращує досвід користування застосунком та допомагає краще взаємодіяти з програмою.

Під час проектування системи, було вирішено обрати сучасний набір інструментів та технологій. При цьому важливо використовувати саме офіційні джерела та рекомендації, оскільки такий вибір гарантує довгий період підтримки цих інструментів та бібліотек. Програма була написана в середовищі Android Studio із використанням інструментів Android SDK. Для написання коду використовувалась мова програмування Kotlin, яка призначена для розробки Android-застосунків.

Як основний архітектурний патерн було обрано MVVM. MVVM розшифровується як Model, View, ViewModel.

- Model: Зберігає дані програми. Model не може безпосередньо взаємодіяти з View. Зазвичай рекомендується надавати дані ViewModel через Observables.
- View: Представляє інтерфейс користувача програми без будь-якої логіки програми. Вона спостерігає за ViewModel.
- ViewModel: Діє як зв'язок між Model та View. Цей клас відповідає за перетворення даних з Model. Він надає потоки даних View. Він також використовує перехоплювачі або зворотні виклики для оновлення View. ViewModel запитує дані з Model [4].

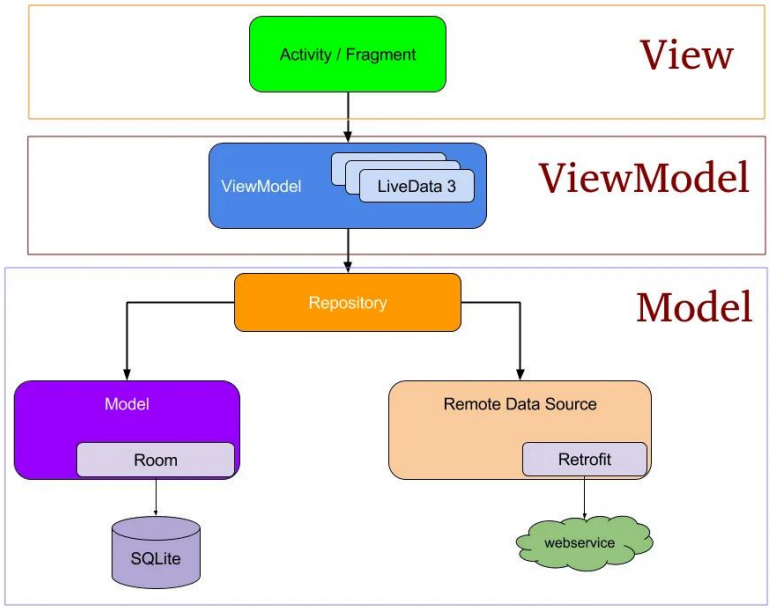


Рисунок 1 – Архітектура системи

Як видно з рисунку 1, архітектура MVVM забезпечує чітке розділення відповідальностей між усіма шарами застосунку. Кожен компонент виконує власну функцію. Завдяки такому підходу до побудови архітектури забезпечується висока модульність коду, що дозволяє легко тестувати кожен шар незалежно від інших.

Особливої уваги заслуговує те, як репозиторій (Repository), який є частиною шару Model, взаємодіє з джерелами даних. Репозиторій абстрагує логіку отримання даних, а отже немає різниці звідки саме надходять дані — з мережі, локальної бази даних чи іншого сервісу. Це забезпечує гнучкість та масштабованість архітектури: при потребі можна легко додати нове джерело даних або реалізувати кешування без впливу на інші шари, зокрема на ViewModel та View.

Крім того, оскільки ViewModel виступає посередником між View та Model, він ізольовує логіку обробки даних та стану інтерфейсу від самої View. Це значно спрощує підтримку інтерфейсу користувача.

Таким чином, зображена на рисунку архітектура MVVM є не лише зручною для розробки, але й ефективною для розширення функціоналу програми, тестування, та довгострокової підтримки.

Розглянемо функціонал системи. Основна властивість системи випливає з аналізу інших застосунків: програма повинна працювати і без доступу до Інтернету. Таким чином було зосереджено в першу чергу на локальному збереженні даних та правильному функціонуванні застосунку в режимі Offline. При розширенні функціоналу у майбутньому можна використати хмарні сервіси для зберігання даних.

Окрему увагу було приділено сповіщенням. Адже важливо показувати сповіщення вчасно, але без додаткового навантаження на ресурси та акумулятор. Для планування сповіщень було обрано метод setInexactRepeating, що перекладається як “встановити неточне повторення”. Цей метод використовується для встановлення повторюваного таймера, що не вимагає абсолютної точності спрацювання. Це означає, що система може трохи зсунути час виконання, наприклад, об'єднати кілька таймерів різних застосунків, щоб економити заряд акумулятора. Оскільки сповіщення для такого типу застосунку не потребують моментальної швидкості спрацювання, як наприклад у месенджерах, їх відхилення на кілька хвилин не вплине на якість користування застосунком, але покращить час роботи мобільного пристрою від акумулятора.

Дизайн застосунку розроблявся із використанням сучасних рекомендацій та підходів Material - офіційної дизайн-практики від Google. Інтерфейс користувача розроблявся із використанням фреймворку Jetpack Compose. Було використано стандартні елементи із позиціонуванням, де користувач їх звик бачити - пошук та сортування зверху екрану, кнопка додавання та редагування рослини спроектована у системну кнопку FAB (Floating Action Button), яка розташована внизу екрану. Це забезпечує стандартизованість та зрозумілість інтерфейсу користувача.

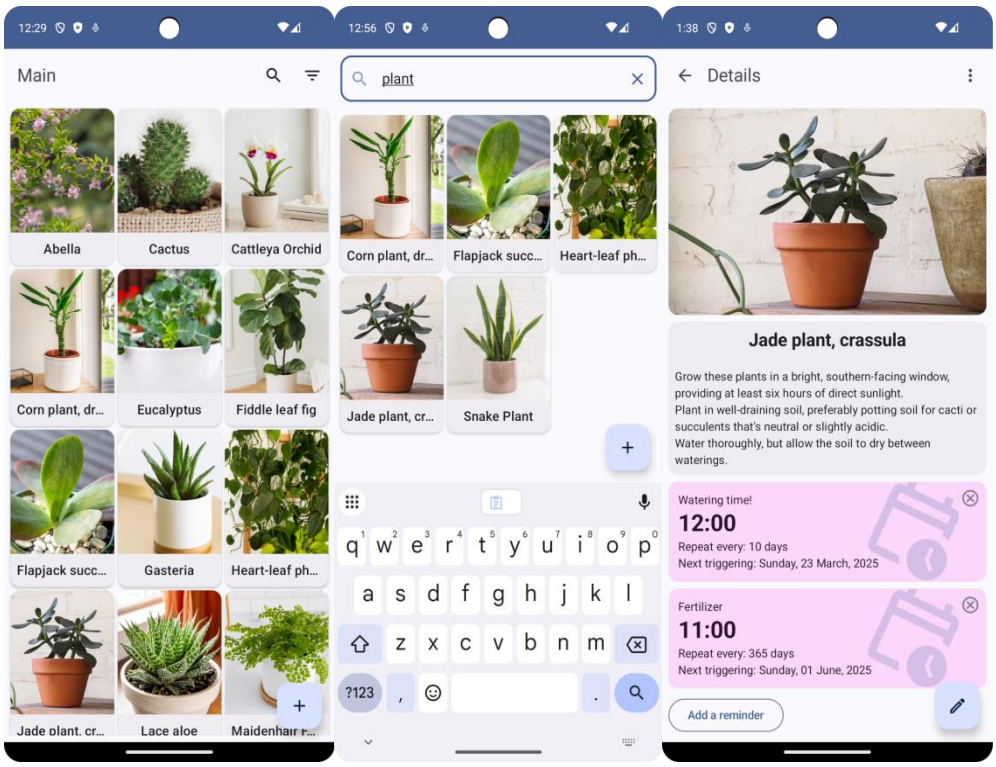


Рисунок 2 – Приклад дизайн застосунку

Отже, використання офіційних інструментів, таких як Android Studio та Android SDK, сприяє створенню надійного та ефективного застосунку. Дотримання рекомендованих практик розробки забезпечує чистоту коду, безпеку обробки даних та стабільність роботи програми. Це дозволяє легко впроваджувати оновлення, що полегшує подальшу підтримку продукту. Крім того, регулярне використання актуальних версій бібліотек допомагає уникнути відомих вразливостей і забезпечує сумісність із новими версіями платформи Android [5]. Такі підходи, що відповідають сучасним стандартам розробки, сприяють досягненню високої стабільності та надійності застосунку [6].

Висновок

У роботі було здійснено аналіз розробленого мобільного застосунку для платформи Android, що охоплював архітектурні рішення, інтерфейс користувача та реалізований функціонал. Було наведено основні інструменти, за допомогою яких розроблялася програма. Зокрема було аргументовано використання методів планування сповіщень, що можуть не зовсім точно спрацьовувати, при цьому значною мірою економити заряд пристрою.

Візуальна частина програми відповідає сучасним принципам дизайну, а функціональні можливості успішно реалізують поставлені задачі.

У результаті роботи було підтверджено доцільність використання сучасних архітектурних підходів, що забезпечують масштабованість та зручність у підтримці застосунку. Таким чином, застосунок є повноцінним інструментом, здатним ефективно виконувати функції з нагадування щодо догляду за рослинами.

Список використаних джерел

1. Planta: Plant & Garden Care. [Електронний ресурс] – Режим доступу: https://play.google.com/store/apps/details?id=com.stromming.planta&hl=uk
2. Mobile Operating System Market Share Worldwide. [Електронний ресурс] – Режим доступу: https://gs.statcounter.com/os-market-share/mobile/worldwide.
3. Android operating system. [Електронний ресурс] – Режим доступу: https://www.britannica.com/technology/smartphone
4. Priyank Kumar. Understanding MVVM Architecture in Android. Medium. [Електронний ресурс] – Режим доступу: https://medium.com/swlh/understanding-mvvm-architecture-in-android-aa66f7e1a70b.
5. Meet Android Studio. Get started with Android Studio. [Електронний ресурс] – Режим доступу: https://developer.android.com/studio/intro.
6. Ronak Patel. Android App Development Best Practices. [Електронний ресурс] – Режим доступу: https://aglowiditsolutions.com/blog/android-app-development-best-practices/.

УДК 004.023

ІНТЕРНЕТ-МАГАЗИН З ПРОДАЖУ КЕРАМІЧНИХ ВИРОБІВ

Циквас О.Р.
Західноукраїнський національний університет
бакалавр

І. Постановка проблеми

Сучасний розвиток інформаційних технологій та зростання популярності онлайн-торгівлі створюють сприятливі умови для розвитку електронної комерції в різних галузях. Ринок керамічного посуду не є винятком, і все більше споживачів віддають перевагу зручності та широкому вибору, які пропонують інтернет-магазини. Однак, існуючі онлайн-платформи часто не повною мірою враховують специфічні потреби покупців керамічної продукції, такі як можливість детального розгляду текстури, кольору та унікальних особливостей кожного виробу. Це створює перешкоди для прийняття обґрунтованого рішення про покупку та знижує рівень задоволеності клієнтів.

ІІ. Мета роботи

Метою розробки є покращення процесу онлайн-продажу керамічного посуду шляхом створення функціонального та зручного інтернет-магазину, який враховує специфіку даної категорії товарів та забезпечує високий рівень задоволеності клієнтів.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- Провести аналіз існуючих інтернет-магазинів, що пропонують керамічний посуд, та визначити їх сильні та слабкі сторони.
- Визначити вимоги до функціональності та дизайну майбутнього інтернет-магазину.
- Розробити архітектуру програмного забезпечення та структуру бази даних інтернет-магазину.
- Реалізувати основні функціональні модулі інтернет-магазину, включаючи каталог товарів, кошик, оформлення замовлення, особистий кабінет користувача та адміністративну панель.
- Провести тестування та налагодження розробленого програмного забезпечення.
- Оцінити економічну ефективність та практичну значущість розробленого інтернет-магазину.

ІІІ. Основна частина

У даній роботі було досліджено процес створення інтернет-магазину з продажу керамічних виробів. Розглянуто теоретичні аспекти електронної комерції, проведено аналіз ринку та визначено вимоги до майбутнього інтернет-магазину.Було спроектовано структуру бази даних, яка забезпечує ефективне зберігання та управління даними про користувачів, товари, замовлення та інші сутності, необхідні для функціонування інтернет-магазину. Розроблено детальний план програмної реалізації проекту, включаючи вибір технологій, структуру коду, розробку серверної та клієнтської частин, а також стратегію тестування. Особливу увагу було приділено розробці інтерфейсу з користувачем, який є критично важливим для забезпечення позитивного досвіду покупців. Було визначено принципи проектування UI, описано основні елементи інтерфейсу та запропоновано візуальний стиль, що відповідає тематиці керамічних виробів. Розроблені прототипи сторінок інтернет-магазину дозволяють наочно представити структуру та функціональність інтерфейсу. Реалізація інтернет-магазину згідно з розробленим планом дозволить створити зручну, привабливу та ефективну платформу для продажу керамічних виробів, що сприятиме задоволенню потреб покупців та успішному розвитку бізнесу.

Об'єктом управління є процес функціонування інтернет-магазину, що спеціалізується на продажу керамічного посуду. Цей процес включає в себе такі ключові аспекти:

Управління асортиментом продукції: Включає в себе формування каталогу товарів, їх класифікацію, опис характеристик (матеріали, розміри, колір, стиль), оновлення інформації про наявність та ціни. Особлива увага приділяється представленню унікальних характеристик керамічних виробів, таких як текстура, ручна робота, авторський дизайн.

Забезпечення зручного користувацького досвіду: Організація інтуїтивно зрозумілої навігації по сайту, розробка ефективної системи пошуку та фільтрації товарів, забезпечення якісних фотографій та, можливо, 3D-візуалізації продукції.

Управління замовленнями: Прийом, обробка, підтвердження та відстеження замовлень. Включає в себе взаємодію з клієнтами, обробку платежів, організацію доставки.

Взаємодія з клієнтами: Надання консультацій, обробка запитів та скарг, збір відгуків, реалізація програм лояльності.

Маркетинг та просування: Розробка та реалізація стратегій просування інтернет-магазину, включаючи SEO-оптимізацію, рекламу в соціальних мережах, email-маркетинг.

Аналітика та звітність: Збір та аналіз даних про відвідування сайту, продажі, поведінку користувачів для прийняття обґрунтованих управлінських рішень. Ефективне управління цими аспектами є критично важливим для забезпечення успішної діяльності інтернет-магазину, задоволення потреб клієнтів та досягнення поставлених бізнес-цілей.

Діаграми варіантів використання (Use Case Diagrams) - це один з типів діаграм мови UML (Unified Modeling Language), яка використовується для моделювання поведінки системи. Вони допомагають описати, що система повинна робити з точки зору користувача, не вдаючись у деталі того, як саме це буде реалізовано.

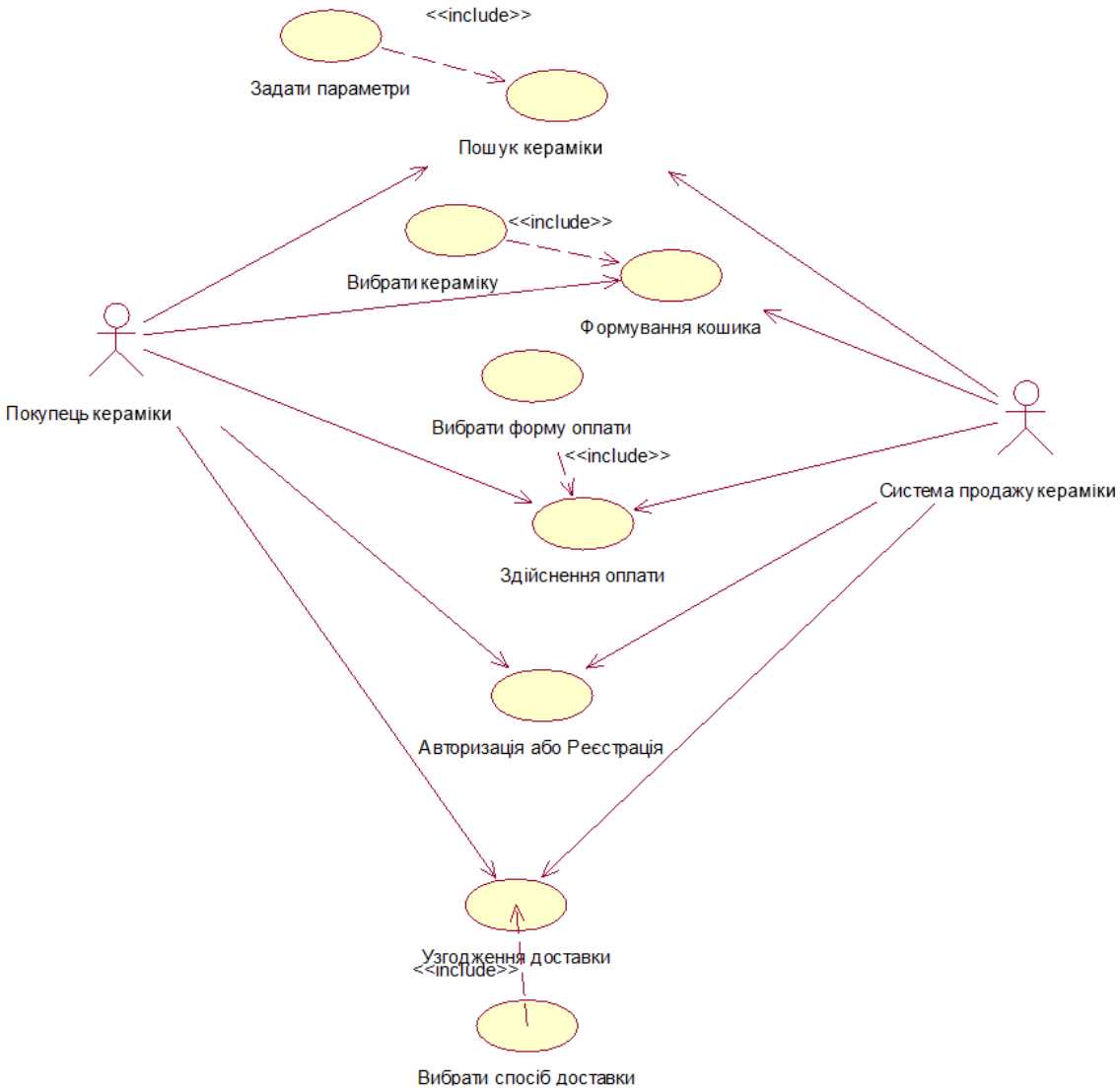


Рисунок 1 – Діаграма «Варіантів використання» інтернет магазину керамічних виробів

Список використаних джерел

1. Лодон К. С., Трейвер Ч. Г. Електронна комерція 2021: Бізнес, технології, суспільство. E-commerce 2021: Business, Technology, Society. — Pearson, 2021. 912 с.
2. Дакетт Д. HTML і CSS: Проєктування та створення вебсайтів (HTML and CSS: Design and Build Websites). Wiley, 2011.512с.
3. Соммервіл І. Інженерія програмного забезпечення (Software Engineering). Pearson Education, 2016. 816 с.
4. Прессман Р. С. Інженерія програмного забезпечення: Практичний підхід (Software Engineering: A Practitioner’s Approach). McGraw-Hill, 2014. 928 с.

УДК 004.4

РЕАЛІЗАЦІЯ МОДУЛЯ КЕРУВАННЯ ДАНИМИ В СИСТЕМІ УПРАВЛІННЯ РЕСУРСАМИ ПІДПРИЄМСТВА

Капустинський М.С.[1)], Стасів І.С.[2)]
Західноукраїнськийнаціональнийуніверситет
[1)] бакалавр;[2)] к.т.н., доцент

I. Постановка проблеми

У сучасних умовах цифрової трансформації підприємств ефективне управління людськими ресурсами набуває особливої важливості. Ручне ведення обліку працівників, використання паперових носіїв та електронних таблиць є неефективними, оскільки призводять до втрати часу, зростання кількості помилок і складнощів в аналізі даних. Для розв'язання цих проблем необхідно впроваджувати спеціалізовані програмні рішення, які дозволяють автоматизувати облік персоналу, спростити адміністративні процедури та забезпечити оперативний доступ до актуальноїі нформації.

II. Мета роботи

Метою роботи є розробка модуля керування даними в системі управління ресурсами підприємства, який дозволяє виконувати основні операції з обліку працівників: додавання, редагування, видалення та перегляд даних про співробітників. Система також повинна забезпечувати безпеку даних, інтуїтивно зрозумілий інтерфейс та можливість масштабування.

III. Сценарії використання системи

Система управління ресурсами підприємства має широкий спектр сценаріїв використання, які охоплюють основні потреби у роботі з персоналом. Основнісценарії включають:

- додавання працівника – забезпечує можливість створення нового профілю працівника із заповненням особистої інформації, контактних даних, посади та дати прийняття на роботу.
- редагування інформації про працівника – дозволяє вносити зміни до вже існуючих профілів працівників, включаючи оновлення контактної інформації, посади, відділу або додавання нових сертифікатів і досягнень.
- видалення працівника – дає можливість видалення профілю працівника у випадку звільнення або завершення співпраці з підприємством. Процедура включає підтвердження дії для запобігання випадковому видаленню даних.

Перегляд інформації про працівника – надає зручний інтерфейс для перегляду детальної інформації про кожного працівника, включаючи його персональні дані, кваліфікацію, поточну посаду та історію змін.

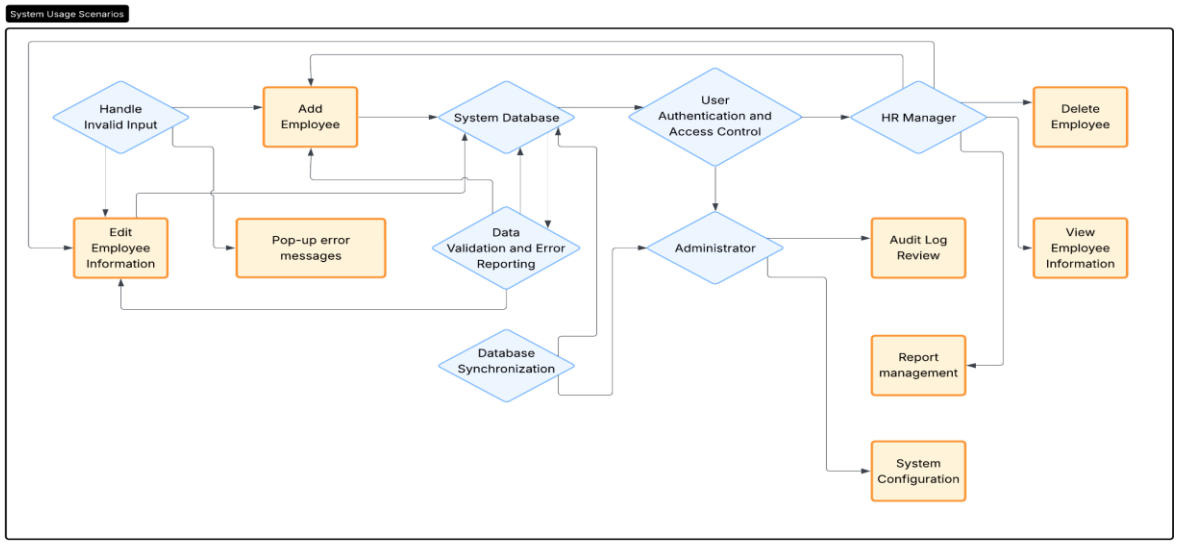


Рисунок 1 – Діаграма варіантів використання системи (Use case)

Ці сценарії спрямовані на підвищення ефективності процесів управління персоналом, спрощення адміністративних процедур і забезпечення прозорості кадрових операцій. Реалізація

даних функцій дозволяє організаціям оптимізувати роботу HR-відділів, забезпечити актуальність і точність даних про працівників, а також підтримувати високий рівень кібербезпеки.

IV. Технологічні основи системи

Проєктована система є веб-застосунком, реалізованим на основі сучасних технологій, що забезпечують високий рівень продуктивності, безпеки та масштабованості. Архітектура побудована за принципом клієнт-серверної моделі.

Клієнтська частина реалізована з використанням бібліотеки React та інструменту збірки Vite, що дозволяє створювати продуктивні, адаптивні та інтуїтивно зрозумілі інтерфейси. Для типізації застосовано TypeScript, що підвищує стабільність коду.

Ключові особливості:

- модульна архітектура з багаторазовим використанням компонентів.
- інтуїтивний дизайн,згідно UX/UI, використання кас томних рішень та Material UI.
- ефективна робота з даними через Axios і React Query.
- реалізація динамічного пошуку та фільтрації персоналу.

Серверна частина побудована на базі Node.js з використанням Express.js, що забезпечує високу продуктивність та масштабованість.

Основні функції:

- реалізація RESTful API для CRUD-операцій.
- централізоване легування та обробка помилок.
- гнучка система конфігурацій для різних середовищ розгортання.

База даних використовується MongoDB, що дозволяє ефективно працювати з великими обсягами динамічної інформації.

Переваги:

- індексація основних полів для швидкого пошуку.
- гнучка схема колекції employees з основними атрибутами: firstname, lastname, position, salary, bonuses, email, phoneNumber, address, hireDate, tasks.
- масштабованість через кластеризацію та використання MongoDB Atlas.
- підтримка цілісності даних за допомогою Mongoose.

V. Функціональні можливості системи

Система управління ресурсами підприємства охоплює всі основні потреби HR-підрозділів та керівництва підприємств.

Ключові функції:

- керування профілями працівників - додавання, редагування, видалення та перегляд інформації про персонал.
- формування звітності – спрощення та оптимізація процесу аналітики ресурсів підприємства.
- безпека даних – сучасні методи шифрування та захисту від атак.

Типові сценарії використання системи:

- додавання працівника – створення нового профілю з повним заповненням особистої та професійної інформації.
- редагування даних – оновлення контактної інформації, посади, кваліфікації, досягнень.
- видалення працівника - з можливістю підтвердження для уникнення випадкового видалення.
- перегляд інформації – зручний доступ до повної історії змін і актуальних даних про працівника.

Висновок

Розроблений модуль керування даними значно спрощує процес обліку персоналу та підвищує ефективність роботи HR-підрозділів підприємств. Використання сучасних веб-технологій забезпечує високу продуктивність системи, її безпеку та зручність використання. Система має великий потенціал для подальшого розвитку та застосування у різних сферах господарської діяльності, включаючи малі та середні підприємства, освітні заклади та державні установи.

Список використаних джерел

1. Sommerville I. Software Engineering (10th Edition). Pearson Education, 2016.
2. Usability.de. UX Design Services and Best Practices. [Електроннийресурс] - Режимдоступу: https://www.usability.de/leistungen/ux-design.html
3. Figma. Difference Between UI and UX Design. [Електроннийресурс] - Режимдоступу: https://www.figma.com/resource-library/difference-between-ui-and-ux/
4. Leapsome. HR Software and Solutions for Human Resource Management. [Електроннийресурс] - Режимдоступу: https://www.leapsome.com/hris-discovery/hr-software
5. React. A JavaScript Library for Building User Interfaces. [Електроннийресурс] — Режимдоступу: https://react.dev

УДК 004.4

ОПТИМІЗОВАНА ПАРАДИГМА ДЛЯ ЗБЕРЕЖЕННЯ ДАНИХ У НАТИВНИХ ДОДАТКАХ ДЛЯ IOS

Сюрпіта О.Я.
Західноукраїнський національний університет
магістрант

І. Постановка проблеми

Безперервна еволюція мобільних технологій, зокрема в екосистемі iOS, зумовила зсув парадигм програмування у бік декларативних підходів, які забезпечують більшу прозорість, адаптивність та узгодженість логіки представлення даних і бізнес-логіки. У відповідь на ці трансформації компанія Apple запропонувала низку новітніх інструментів, з-поміж яких вирізняються SwiftUI - декларативний фреймворк для створення користувацьких інтерфейсів - та SwiftData, який забезпечує сучасне управління персистентними даними.

Інтеграція SwiftData у програмні архітектури застосунків на основі SwiftUI засвідчує тенденцію до уніфікації процесів зберігання й відображення інформації. Завдяки використанню нативних типів Swift для моделювання даних та зручній системі анотацій, SwiftData фактично нівелює розрив між логікою даних та їх представленням, сприяючи спрощенню проєктування моделей. Водночас, процес впровадження SwiftData є багаторівневим і включає послідовні етапи: побудову моделей, позначення зв'язків між сутностями, конфігурацію контейнера моделей і взаємодію з контекстом збереження.

З метою всебічного аналізу можливостей цього фреймворку, у дослідженні розглядаються приклади реалізації у вигляді трьох демонстраційних проєктів: застосунок для ведення бібліотеки книг (із реалізацією зв'язку «один до одного»), месенджер (зв'язок «один до багатьох») та електронний навчальний щоденник (зв'язки «багато до багатьох»). Кожен з них ілюструє ключові аспекти використання SwiftData у контексті CRUD-операцій, а також методів фільтрації, пошуку, сортування і міграції даних.

Варто зауважити, що попри природну інтеграцію SwiftData у SwiftUI та зручність його застосування, цей інструмент все ще перебуває у фазі становлення: зокрема, він обмежений до iOS 17 і має звужений програмний інтерфейс порівняно з Core Data. Тим не менш, динаміка його розвитку свідчить про потужний потенціал щодо майбутнього розширення функціональних можливостей та адаптації під ширші сценарії.

У цьому контексті дослідження фокусується на поглибленому аналізі SwiftData як компонента оптимізованої парадигми збереження даних у нативних iOS-застосунках, з урахуванням як технічних, так і концептуальних аспектів його інтеграції.

ІІ. Мета роботи

Метою дослідження є аналіз та вдосконалення підходів до збереження даних у нативних iOS-додатках шляхом дослідження можливостей фреймворку SwiftData у контексті декларативного програмування на базі SwiftUI.

ІІІ. Метод моделювання інформаційних процесів в соціальних мережах

Об'єктом моделювання є архітектура управління персистентними даними в застосунках, розроблених з використанням SwiftUI. У цьому контексті застосування SwiftData розглядається як структурний елемент, що забезпечує зв'язок між логікою збереження та відображення даних у декларативному середовищі.

Модель охоплює сутності, представлені у вигляді класів Swift з анотаціями, які описують атрибути та зв'язки типів «один до одного», «один до багатьох» і «багато до багатьох». Застосунок у цьому випадку виступає як система з інтегрованим контейнером моделей, що підтримує взаємодію з контекстом збереження даних (ModelContext).

Особливу увагу приділено динаміці CRUD-операцій, впливу структурних змін (міграцій) та сценаріям фільтрації, сортування й пошуку. Для відтворення реалістичних сценаріїв використано експериментальні модулі - демонстраційні додатки з різними рівнями складності структури даних.
Модельна система є частиною декларативного підходу, де логіка керування станом даних і взаємодія з інтерфейсом користувача відбувається в рамках єдиного середовища SwiftUI з використанням інструментів.

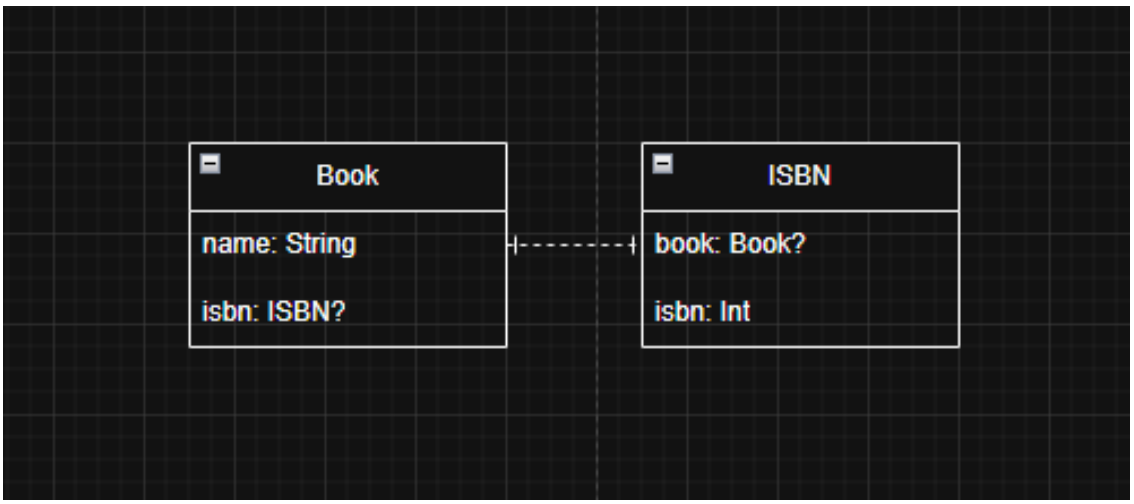


Рисунок 1 –Діаграма зв’язку "Один до одного" між сутностями Book та ISBN

У цьому випадку модель відображає зв’язок типу «один до одного» між сутностями Book та ISBN, що демонструє, як кожна книга в базі даних має унікальний ISBN-код. Така реалізація дозволяє здійснювати точний ідентифікаційний зв’язок між книгами та їх унікальними ідентифікаторами, спрощуючи управління даними та підвищуючи ефективність пошуку та фільтрації. Зв’язок між сутностями підтверджує принцип декларативного програмування, де зв’язки чітко вказуються і відображаються через анотації в коді.

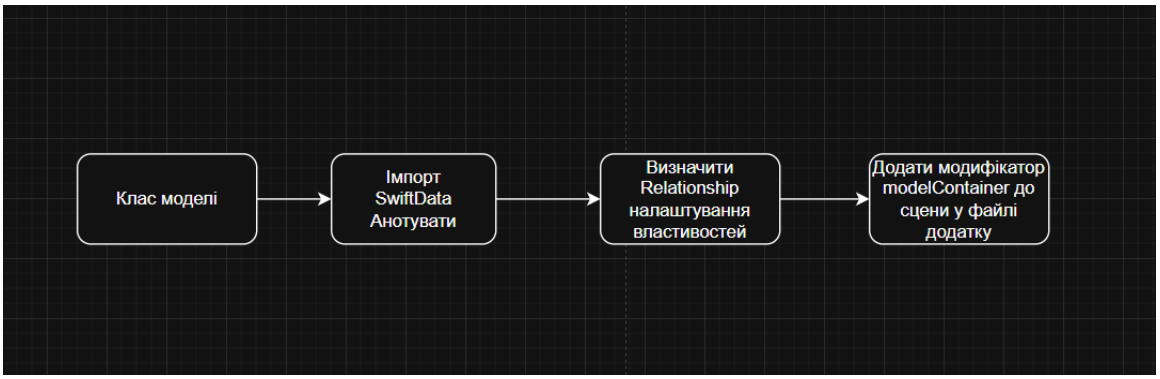


Рисунок 2 – Чотири основні кроки впровадження SwiftData в iOS-додаток

Для ефективної реалізації SwiftData у нативному застосунку на базі SwiftUI необхідно дотримуватись чіткої послідовності дій, що охоплюють усі етапи створення й налаштування моделей даних. Нижче наведено основні кроки, які забезпечують коректне підключення та використання SwiftData у проєкті дивитись рисунок 2.

Висновок

Розроблено метод багатомасштабного моделювання динаміки поширення інформації в соціальних мережах за допомогою агентних моделей, що дає змогу ефективно прогнозувати та аналізувати взаємодії в онлайн-спільнотах. Використання цих моделей дозволяє оптимізувати процеси збору і аналізу даних, а також сприяє створенню адаптованих стратегій управління інформаційними потоками. Це відкриває нові можливості для дослідження та вдосконалення взаємодії користувачів в цифрових мережах.

Список використаних джерел

1. Apple Inc. SwiftUI Documentation. SwiftUI Framework for Building User Interfaces. Apple Developer, 2023. Retrieved from https://developer.apple.com/documentation/swiftui
2. Smith, J.; Johnson, L. The Evolution of iOS Development: From UIKit to SwiftUI. Journal of Mobile Software Engineering, 2021, №45(3), 98-112.
3. Evans, P.; Miller, J. Exploring SwiftData: Data Management for iOS Applications. Journal of Modern iOS Development, 2022, 15(4), 123-139.
4. Morrison, A.; Green, S. Data Persistence in iOS: A Comparative Analysis of SwiftData and Core Data. International Journal of Mobile Development, 2021, 12(1), 45-62.

УДК 004.4

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ЗАСТОСУВАННЯ МЕТОДІВ АНАЛІЗУ ТЕКСТУ І МАШИННОГО НАВЧАННЯ ДЛЯ ВИЯВЛЕННЯ ФЕЙКОВИХ НОВИН

Крепич С.Я.[1)], Драгуца Р.А.[2)], Співак І.Я.[3)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] магістрант; [3)]к.т.н., доцент

Постановка проблеми

У сучасному інформаційному середовищі масове поширення фейкових новин становить серйозну загрозу суспільству, політиці, безпеці та довірі до медіа. Швидкість розповсюдження неправдивих матеріалів через соціальні мережі значно перевищує здатність людини верифікувати кожне повідомлення вручну. Через це виникає потреба у створенні автоматизованих систем для аналізу текстової інформації, які здатні виявляти фейкові новини з високою точністю.

Мета роботи

Розробити концепцію функціонування інтелектуальної системи, що дозволяє автоматично визначати правдивість текстів новин шляхом обробки природної мови (NLP) та застосування методів машинного навчання та обґрунтувати вибір архітектури нейронної мережі, методів попередньої обробки тексту та векторизації даних.

Реалізація системи

1. Підготовка та обробка даних

Дані для навчання взято з відкритих датасетів: Fake.csv та True.csv, що містять фейкові та правдиві новини відповідно. На рисунку 1 зображено вміст файлу True.csv. На рисунку 2 зображено вміст файлу Fake.csv.

title	text	subject	date
As U.S. budget fight looms, Republicans flip their fiscal script	WASHINGTON (Reuters) - The head of a conservative Republican faction in the U.S. Congress, who voted...	politicsNews	December 31, 2017
U.S. military to accept transgender recruits on Monday: Pentagon	WASHINGTON (Reuters) - Transgender people will be allowed for the first time to enlist in the U.S. m...	politicsNews	December 29, 2017
Senior U.S. Republican senator: 'Let Mr. Mueller do his job'	WASHINGTON (Reuters) - The special counsel investigation of links between Russia and President Trump...	politicsNews	December 31, 2017

Рисунок 1 – Вміст файлу True.csv

title	text	subject	date
Donald Trump Sends Out Embarrassing New Year's Eve Message; This is Disturbing	Donald Trump just couldn t wish all Americans a Happy New Year and leave it at that. Instead, he had...	News	December 31, 2017
Drunk Bragging Trump Staffer Started Russian Collusion Investigation	House Intelligence Committee Chairman Devin Nunes is going to have a bad day. He s been under the as...	News	December 31, 2017
Sheriff David Clarke Becomes An Internet Joke For Threatening To Poke People 'In The Eye'	On Friday, it was revealed that former Milwaukee Sheriff David Clarke, who was being considered for ...	News	December 30, 2017

Рисунок 2 –Вміст файлу Fake.csv

Виконано злиття наборів даних, очищення текстів та випадкове перемішування для зменшення упередженості. Для попередньої обробки тексту використано функцію wordopt(), яка прибирає зайві символи, знаки пунктуації та числа.

Лістинг коду:

```
def wordopt(text):
    text = text.lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub("\\W", " ", text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
```

2. Архітектура моделі

- Вхідні вектори TF-IDF передаються у вхідний шар.
- Побудовано три приховані шари: 128, 64 та 32 нейрони відповідно, з функцією активації ReLU.
- Вихідний шар з одним нейроном активується функцією sigmoid, що дозволяє класифікувати тексти як фейкові (0) або правдиві (1).
- Модель реалізована за допомогою фреймворку TensorFlow та Keras.

3. Тренування моделі

Використано train_test_split для розділення даних (75% — навчання, 25% — тестування).
Лістинг коду:

```
x = df["text"]
y = df["class"]
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25)
```

Навчання тривало 5 епох з пакетом у 64 елементи, з валідацією на тестовій вибірці. Модель збережено у форматі .keras, а векторизатор TF-IDF — у форматі .pkl. На рисунку 3 зображено точність моделі під час навчання.

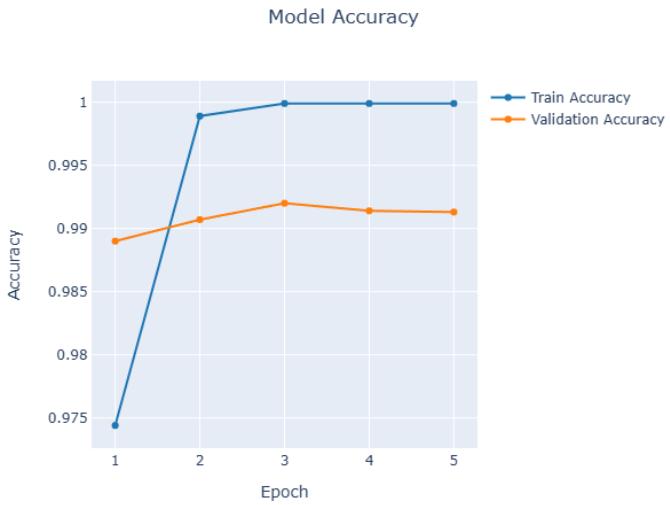


Рисунок 3 – Криві навчання моделі: точність по епохах

4. Розробка модуля ручного тестування

Створено скрипт manual_testing.py, який дозволяє вводити текст вручну, обробляти його та отримувати миттєвий результат класифікації.

Назва	**Текст**(урізано)	**Тема**	**Дата**	**Клас**
Seven Iranians freed in the prisoner swap have...	21st Century Wire says This week...	Middle-east	January 20, 2016	0
#Hashtag Hell&TheFakeLeft	By Dady Chery and Gilbert Mercier...	Middle-east	January 19, 2016	0
Astroturfing: Journalist Reveals Brainwashing ...	Vic Bishop Waking TimesOur reality...	Middle-east	January 19, 2016	0

Векторизація нового тексту здійснюється на основі збереженого TF-IDF-моделя, а класифікація — з використанням збереженої нейромережі. Користувачу відображається відповідь: "Fake" або "True" залежно від результату моделі.

Висновки

У ході роботи реалізовано інтелектуальну систему автоматичного виявлення фейкових новин на основі текстового аналізу. Застосування методів машинного навчання та класичних підходів до векторизації забезпечило високі показники точності, що підтверджує ефективність і доцільність використання таких рішень у сферах медіаконтролю та інформаційної безпеки.

Список використаних джерел

1. Литвин В.В., Шевчук О.С., Бродюк О.О. Методи та засоби аналізу текстової інформації у системах виявлення фейкових новин, 2021.
2. Жук Ю.О., Паламарчук О.В. Застосування алгоритмів машинного навчання для класифікації фейкових новин, 2020.
3. Головач Т.С., Чеканова Г.В. Застосування штучного інтелекту у боротьбі з дезінформацією, 2022.

УДК 004.4

ОНТОЛОГІЧНИЙ ПІДХІД ДО МОДЕЛЮВАННЯ ЗНАНЬ В ІТ-КОНСАЛТИНГУ ДЛЯ СППР

Смаль О.І.[1)], Дзиґа Ю.В.[2)]
Західноукраїнський національний університет
[1)]магістр; [2)] аспірант

І. Постановка проблеми

Управління складними та різноманітними знаннями є одним з найважливіших викликів у сфері ІТ-консалтингу. Для ефективної роботи систем підтримки прийняття рішень (СППР) необхідна чітка, структурована та машинозчитувана репрезентація знань про технології, бізнес-процеси, клієнтські потреби та консалтингові методології. Одним із перспективних підходів до вирішення цієї задачі є застосування онтологічного моделювання. Онтології дозволяють створити єдину, несуперечливу та розширювану модель знань, що враховує технічні, бізнесові, методологічні аспекти, динаміку ринку та складність взаємозв'язків між елементами предметної області.

Застосування онтологій в ІТ-консалтингу є особливо актуальним, оскільки ця галузь характеризується швидким розвитком технологій, постійною зміною ринкових умов та унікальністю кожного клієнтського запиту. Традиційні підходи до управління знаннями, що базуються на неструктурованих документах або реляційних базах даних, часто не можуть забезпечити необхідну гнучкість та глибину семантичного аналізу. Онтології, навпаки, надають формальний апарат для представлення семантичних взаємозв'язків між різними сутностями, що дозволяє системам підтримки прийняття рішень не просто обробляти дані, а й розуміти їхнє значення та контекст. Це є ключовим для формування якісних та релевантних рекомендацій в умовах невизначеності та великих обсягів інформації.

ІІ. Мета роботи

Для моделювання знань у ІТ-консалтингу пропонується побудова предметно-орієнтованої онтології, яка включає такі концепти, як типи ІТ-проєктів (наприклад, впровадження ERP, розробка ПЗ), технології (мови програмування, фреймворки, бази даних, хмарні платформи), клієнти (галузь, розмір, бізнес-процеси, проблеми), консультанти (спеціалізація, кваліфікація, досвід), а також типи вимог, рішень та метрик успіху. Для опису онтології доцільно використовувати мову OWL, яка підтримує формальну логіку та автоматичне виведення знань.

ІІІ. Основний матеріал

Онтологія виконує низку критичних функцій у складі СППР: вона є єдиною семантичною основою для всіх компонентів системи, забезпечує логічне узгодження термінів та концепцій, підтримує семантичний пошук і дозволяє проводити логічне виведення на основі сформульованих правил. Наприклад, система може автоматично вивести, що складний проєкт потребує команди з високим рівнем кваліфікації. Важливою перевагою онтологій також є підтримка контекстуалізації рішень, що підвищує релевантність рекомендацій та спрощує інтеграцію з іншими інформаційними джерелами — CRM-системами, системами управління проєктами тощо.

Детальніше розглядаючи функції онтологій, варто зазначити їхню здатність до семантичної інтероперабельності. У складних ІТ-середовищах дані часто зберігаються в різних форматах і системах. Онтологія виступає як спільний семантичний рівень, що дозволяє об'єднувати інформацію з різнорідних джерел, надаючи їй єдиного, узгодженого значення. Це значно спрощує аналіз даних з різних департаментів або інструментів, що є типовим для ІТ-консалтингу. Крім того, онтології дозволяють реалізувати інтелектуальний пошук, де система розуміє не тільки ключові слова запиту, але й їхнє семантичне значення та зв'язки. Це дає змогу знаходити релевантнішу інформацію, наприклад, "всі проекти, які включали міграцію на хмарні сервіси для фінансового сектору", навіть якщо ці терміни не були явно вказані у первинному описі проекту.

Важливою частиною онтологічного підходу є логічне виведення (reasoning). Використання онтологічних міркувальників (reasoners) дозволяє системі не просто зберігати знання, а й генерувати нові факти з існуючих, а також перевіряти їх на несуперечність. Наприклад, якщо в онтології визначено, що "Microsoft Azure" є підкласом "Хмарних платформ", то система автоматично виведе, що будь-який проект, який використовує Azure, також використовує хмарні платформи. Це дозволяє

формулювати більш складні запити та виявляти приховані взаємозв'язки, що є критичним для глибокого аналізу ситуації та прийняття обґрунтованих рішень. Більше того, reasoners можуть виявляти неявні протиріччя у знаннях, що допомагає підтримувати високу якість онтології.

Програмна реалізація СППР, заснованої на онтологіях, передбачає наявність модулів взаємодії з онтологією, модуля логічного виведення, інтерфейсу користувача, а також інтеграцію з інструментами побудови й редагування онтологій (наприклад, Protégé). Для обробки знань використовуються запити SPARQL, reasoner-и для виведення та графові алгоритми для аналізу структур зв'язків. Актуальним завданням є напівавтоматичне наповнення онтології на основі аналізу даних із зовнішніх джерел.

Розробка програмного забезпечення такої системи вимагає не лише реалізації згаданих модулів, але й вибору відповідної архітектури, яка забезпечить масштабованість та ефективність обробки знань. Це може бути сервіс-орієнтована архітектура, де онтологічний сервіс надає функціонал для роботи зі знаннями іншим компонентам СППР.

Математичне забезпечення включає не тільки формальну логіку, що лежить в основі OWL та reasoner-ів, але й алгоритми обробки графів для аналізу складних взаємозв'язків між концептами та екземплярами онтології. Наприклад, для визначення оптимального шляху розгортання IT-інфраструктури або ідентифікації потенційних конфліктів у технологічних стеках. Важливим етапом є також варіфікація та валідація онтології та системи в цілому, щоб переконатися в коректності представлених знань та точності висновків. Це може включати експертну оцінку, порівняння з реальними кейсами та тестування на репрезентативних даних.

Крім того, в процесі розробки необхідно враховувати процес еволюції онтології. Знання в IT-консалтингу не є статичними: нові технології з'являються, бізнес-процеси змінюються, а вимоги клієнтів еволюціонують. Тому СППР має бути здатною адаптуватися до змін у предметній області. Це передбачає розробку механізмів для оновлення та розширення онтології, включаючи додавання нових концептів, властивостей або взаємозв'язків.

Цей процес може бути як ручним (з участю експертів), так і напівавтоматичним, з використанням методів машинного навчання для виявлення нових знань з неструктурованих текстів або даних. Такий динамічний підхід забезпечує довгострокову релевантність та ефективність онтологічно-орієнтованих СППР в IT-консалтингу.

Онтологічний підхід є фундаментальним для створення інтелектуальних систем підтримки прийняття рішень в IT-консалтингу, оскільки він дозволяє формалізувати, інтегрувати та ефективно використовувати знання предметної області. Це сприяє підвищенню якості аналізу, обґрунтованості рекомендацій та загальної ефективності консалтингових послуг. Перспективними напрямами подальших досліджень є інтеграція з методом прецедентів (CBR), розширення онтологій за рахунок нечітких знань та використання методів машинного навчання для автоматичного оновлення бази знань.

Висновки

Онтологічне моделювання знань виступає ключовим елементом у створенні гнучких, масштабованих і інтелектуальних систем підтримки прийняття рішень. Воно дозволяє структуровано подати знання предметної області, логічно їх аналізувати та поєднувати з іншими методами — такими як CBR і нечітка логіка. У результаті системи стають здатними не лише накопичувати досвід, але й логічно адаптуватися до нових ситуацій.

Подальші дослідження передбачають автоматизацію побудови та збагачення онтологій, а також створення гібридних платформ для гнучкої підтримки IT-рішень. Застосування таких систем дозволить IT-консалтинговим компаніям значно підвищити свою конкурентоспроможність, надаючи клієнтам більш точні, обґрунтовані та індивідуалізовані рішення, а також оптимізуючи внутрішні процеси управління знаннями та експертизою. Це відкриває шлях до створення нового покоління СППР, які стануть незамінним інструментом для сучасного IT-консультанта.

Список використаних джерел

1. Безкоровайний О. Р. Автоматизація процесів менеджменту та інтелектуальної діяльності : монографія. Тернопіль : ТНЕУ, 2010. – 256 с.
2. Ситник В. Ф. Системи підтримки прийняття рішень : навч. посіб. / В. Ф. Ситник. – К. : КНЕУ, 2009. – 614 с.
3. Zhang, Y., Li, X., & Wang, J. (2016). Research on Knowledge Representation in Expert System. International Journal of Computer Science and Information Security (IJCSIS), 14(9), 1–6. Режим доступу: https://www.researchgate.net/publication/300617400_Research_on_Knowledge_Representation_in_Expert_System
4. Басюк Т. М., Литвин В. В. Мови опису онтологій : навч. посіб. / Видавництво Львівської політехніки, 2020. – 276 с.

УДК 004.4

ДОСЛІДЖЕННЯ МЕТОДІВ ПОБУДОВИ СИСТЕМ ТЕКСТОВОЇ КЛАСИФІКАЦІЇ НА ОСНОВІ ЄДИНОЇ ПЛАТФОРМИ

Сеник Д.С.[1)], Онищук А.З.[2)]
Західноукраїнський національний університет
[1)]магістр; [2)] аспірант

I. Проблема

Сучасні системи автоматичної обробки тексту дедалі частіше стикаються із завданням класифікації документів. Однак порівняння ефективності різних методів класифікації є непростою задачею через складність і багатогранність обробки природної мови. Алгоритми можуть показувати різні результати залежно від особливостей вхідних даних, структури тексту, предметної області, якості попередньої обробки тощо. Окрім цього, на ефективність класифікації значно впливають такі фактори, як вибір ознак, метод індексації, модель векторного подання тексту, а також алгоритм навчання. Через велику кількість змінних важко оцінити реальну якість окремого методу або зробити об'єктивне порівняння кількох підходів без стандартизованої процедури.

Ще однією важливою проблемою є висока розмірність простору ознак, що призводить до зростання обчислювальних витрат і ризику перенавчання моделей. Коли класифікатор «запам’ятовує» тренувальні дані замість того, щоб узагальнювати інформацію, він втрачає здатність правильно працювати з новими, раніше не баченими прикладами. Тому виникає потреба у зменшенні кількості ознак без значної втрати інформації, що також вимагає вибору відповідного методу скорочення простору. Сумарно це створює багатоаспектну проблему, яку можливо вирішити лише шляхом детального аналізу всіх етапів процесу класифікації.

II. Мета завдання

Головна мета дослідження полягає в побудові об'єктивного та уніфікованого підходу до порівняння методів класифікації текстових документів. Для цього необхідно реалізувати кілька популярних алгоритмів обробки тексту, протестувати їх у однакових умовах та визначити, які з них забезпечують найкращу продуктивність на заданому наборі даних. Особлива увага має бути приділена однаковості вхідних умов: ті самі тексти, той самий підхід до попередньої обробки, однакові методи індексації та скорочення розмірності. Це дозволить виключити зовнішні фактори впливу та зосередитися безпосередньо на якості алгоритмів.

Також важливо оцінити, як різні моделі індексації (наприклад, "мішок слів", Word2vec, n-грам тощо) впливають на результат класифікації, і який вплив має вибір підходу до зменшення кількості ознак. Застосування технік, таких як TF-IDF, латентно-семантичний аналіз, поточкова взаємна інформація, дозволяє по-різному інтерпретувати важливість термінів, тому аналіз їхньої ефективності в реальних умовах має велике практичне значення. Результати мають не лише вказати на найкращі методи, але й продемонструвати залежність точності класифікації від обраної конфігурації системи.

III. Основна частина

Завдання дослідження охоплює повний цикл обробки тексту для класифікації документів.

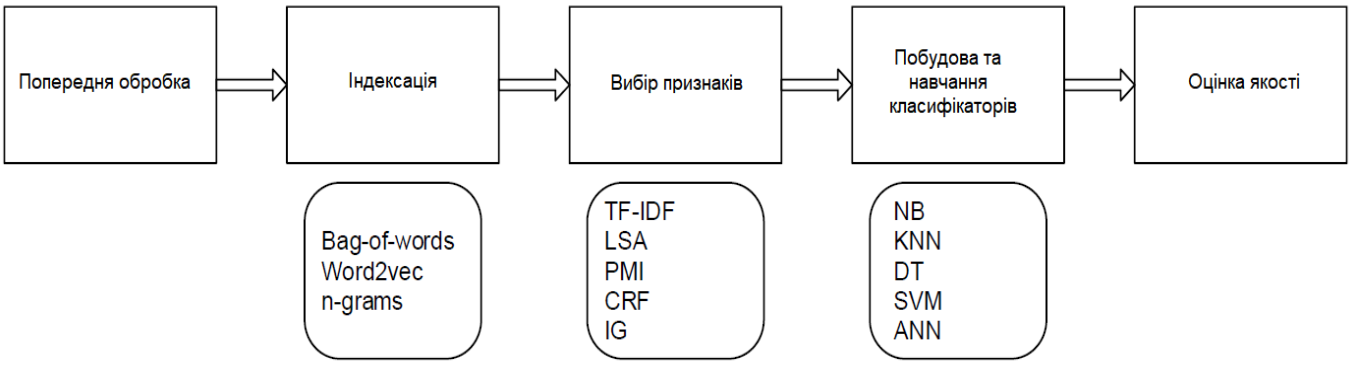


Рисунок 1 – Етапи процесу автоматичної класифікації текстів

Перший етап — попередня обробка тексту, яка включає в себе токенізацію (розбиття тексту на слова або інші одиниці), видалення функціональних слів (наприклад, прийменників і сполучників), морфологічний аналіз із визначенням частин мови та стематизацію (зведення слів до базових форм). Це дозволяє усунути надлишкові елементи з тексту і зменшити обсяг оброблюваної інформації.

Другий етап — індексація документів. Тут застосовуються різні моделі: «мішок слів» перетворює текст у вектор, де кожне слово має свою вагу, Word2vec створює вектори на основі контекстного значення слова, а n-грамна модель працює з послідовностями символів чи слів. Кожен документ у підсумку перетворюється на числову модель, яка зручно обробляється класифікатором. Важливо, щоб для всіх документів (тренувальних і тестових) використовувався один і той самий метод індексації.

Третій етап — зменшення розмірності ознак, оскільки висока кількість термінів значно ускладнює навчання моделі та підвищує ризик перенавчання. Скорочення простору ознак можливе за допомогою технік, таких як TF-IDF, LSA, PMI, CRF або використання статистичних критеріїв, наприклад, коефіцієнтів інформаційного посилення. Ці методи дозволяють зберегти найінформативніші ознаки, зменшуючи при цьому шум та обсяг вхідних даних. Завданням є підібрати такі методи, які забезпечать високу точність при мінімальних витратах обчислювальних ресурсів.

Четвертий етап — полягає в тому, щоб побудувати математичну модель, здатну розпізнавати закономірності в текстових даних. Після перетворення текстів у числові вектори (на попередньому етапі), ці вектори разом із відповідними мітками категорій подаються на вхід алгоритму машинного навчання. Популярні моделі включають наївний баєсівський класифікатор, логістичну регресію, метод опорних векторів (SVM) або нейронні мережі. Під час навчання модель «вчиться» пов’язувати особливості тексту з певною категорією, щоб потім робити передбачення на нових даних.

П’ятий етап — включає перевірку якості роботи моделі. Це робиться на спеціальному тестовому наборі даних, який не використовувався під час навчання. Результати класифікації порівнюються з реальними мітками, і обчислюються метрики якості: точність (accuracy), повнота (recall), точність класифікації (precision), F1-міра тощо. Якщо модель показує добрі результати, її можна використовувати для класифікації нових, невідомих текстів. У разі низької якості модель можна покращити, змінивши параметри або використавши інші алгоритми.

Висновки

Детальний і послідовний аналіз кожного з етапів процесу класифікації текстових даних є надзвичайно важливим для повного усвідомлення того, як саме змінюється ефективність застосовуваних алгоритмів залежно від вибраних стратегій обробки інформації.

Здійснене дослідження наочно демонструє, що на точність класифікації значний вплив мають не тільки використовувані моделі машинного навчання, але й такі етапи, як попередня обробка текстів, способи їх індексації, а також техніки зменшення розмірності простору ознак. Запровадження уніфікованого середовища, в якому можливо випробувати різноманітні підходи в однакових умовах, сприяє більш об’єктивному аналізу отриманих результатів і надає змогу обґрунтовано оцінити, які саме рішення є найбільш ефективними для конкретних типів задач.

У результаті можна зробити висновок, що досягнення високої точності класифікації текстів вимагає системного та комплексного підходу до побудови всієї архітектури обробки даних. Це охоплює всі кроки — починаючи від очищення тексту від шумів, його лематизації, токенізації та нормалізації, і завершуючи вибором відповідних способів подання текстів у числовій формі, обранням найдоцільніших моделей, а також застосуванням ефективних методів скорочення розмірності. Така всебічна методологія дозволяє не тільки досягати стабільно високих результатів у класифікації, але й створювати більш гнучкі, масштабовані та адаптивні системи, які можуть успішно використовуватись у широкому спектрі предметних галузей, де обробка текстових даних відіграє ключову роль.

Список використаних джерел

1. Катющенко Д. Дослідження методів попередньої обробки та векторизації текстових документів [Текст]: / Д. Катющенко, Ю. Олійник //Матеріали наукової конференції студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка» – ІОТ-2018, 24– м. Київ: С. 193-195

2. Ханько Г. В. Огляд алгоритмів текстового аналізу даних[Текст]: / Г.В. Ханько // “Інформатика та обчислювальна техніка” – ІОТ-2017– Київ: 2017. С. 17 – 20.

УДК 004.4

ДОДАТОК ДЛЯ ЕКСПОРТУ ТА ІМПОРТУ МУЗИКИ МІЖ РІЗНИМИ ПЛАТФОРМАМИ

Тимчишин В.С.[1)], Макогон О.А.[2)]
Західноукраїнський національний університет
[1)]д.філ., ст. викладач; [2)] бакалавр

І. Постановка проблеми

У сучасному світі існує велика кількість музичних потокових сервісів, таких як Spotify, Apple Music, YouTube Music, Deezer та інші. Користувачі часто мігрують між платформами або користуються кількома одночасно, що призводить до потреби переносити плейлисти та бібліотеки між різними сервісами. Цей процес зазвичай вимагає значних зусиль та часу, оскільки кожна платформа використовує власні формати даних та API. Відсутність універсального рішення для синхронізації музичних колекцій між різними сервісами створює незручності для користувачів та обмежує їх можливості у виборі платформи, що найкраще відповідає їх потребам.

ІІ. Мета роботи

Мета роботи – розробити кросплатформний додаток, який забезпечить безперешкодний імпорт та експорт плейлистів, альбомів та улюблених треків між різними музичними сервісами, спростить міграцію бібліотек та дозволить синхронізувати музичні колекції користувачів. Додаток покликаний подолати бар'єри між екосистемами стрімінгових платформ та надати користувачам повну свободу у використанні їхніх музичних бібліотек.

ІІІ. Основна частина

Розроблений додаток базується на архітектурі клієнт-сервер, де клієнтська частина реалізована з використанням React Native для забезпечення кросплатформності, а серверна частина побудована на Node.js. Для взаємодії з музичними сервісами використовуються їхні публічні API, а також методи веб-скрапінгу для платформ з обмеженим доступом до API.

Архітектура системи включає такі основні модулі:

модуль аутентифікації, що забезпечує безпечний доступ до акаунтів користувачів на різних платформах за допомогою OAuth, використовує токенізацію для збереження сеансу користувача без необхідності повторного входу;

модуль аналізу та перетворення даних, який відповідає за обробку різних форматів даних та пошук відповідностей між треками на різних платформах;

модуль синхронізації, що виконує основні операції імпорту та експорту;

модуль кешування, який зберігає історію операцій та метадані для оптимізації майбутніх перенесень.

Для вирішення проблеми ідентифікації однакових треків на різних платформах розроблено алгоритм нечіткого пошуку, який використовує метадані треків (назва, виконавець, альбом, тривалість) для знаходження найбільш ймовірних відповідностей. У випадках неоднозначності користувачу пропонується обрати правильний трек із списку кандидатів.

Важливою особливістю додатку є підтримка офлайн-режиму, який дозволяє користувачам підготувати перенесення даних без активного інтернет-з'єднання та виконати їх пізніше. Для цього використовується локальна база даних SQLite.

Додаток також передбачає систему повідомлень, яка інформує користувача про статус перенесення треків та можливі помилки під час процесу. Інтерфейс користувача адаптований для різних розмірів екранів і забезпечує зручну навігацію між основними функціями програми. Завдяки модульній структурі архітектури додаток легко розширюється для інтеграції нових музичних сервісів у майбутньому.

Програма також підтримує гнучку систему налаштувань, яка дозволяє користувачам обирати сервіси, з якими вони хочуть працювати. Завдяки використанню REST API забезпечується швидка і надійна взаємодія між клієнтською та серверною частинами. Інтегрований механізм логування дозволяє виявляти та аналізувати потенційні проблеми під час роботи додатку. Крім того, реалізовано функцію резервного копіювання, що гарантує збереження важливих даних користувача у випадку збоїв або переустановлення програми. У перспективі планується додати функцію аналітики для відстеження музичних вподобань користувачів та надання персоналізованих рекомендацій.

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

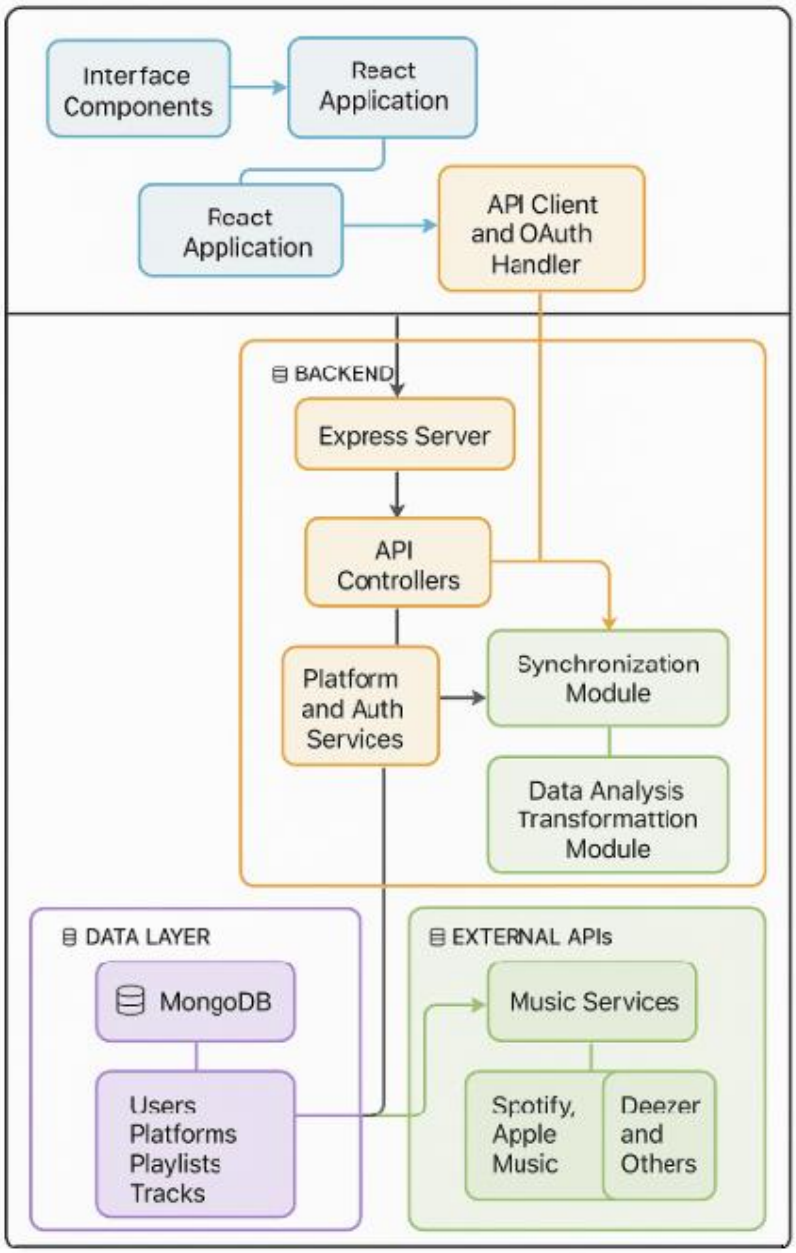


Рисунок 1 – Архітектура системи

Висновок

У роботі досліджено проблему переносу музичних колекцій між різними стрімінговими платформами та розроблено кросплатформний додаток, який забезпечує ефективний імпорт та експорт музики між сервісами. Застосування алгоритму нечіткого пошуку дозволило підвищити точність співставлення треків на різних платформах. Розроблений додаток може бути розширений для підтримки нових музичних платформ та інтеграції з голосовими помічниками для покращення користувацького досвіду.

Список використаних джерел

1. Сенів М.М., Пелещишин А.М. "Методи інтеграції даних у гетерогенних інформаційних системах", Науковий вісник НЛТУ України, 2021, вип. 31(5), с. 88-94.
2. Chomicki P., Jedrzejowicz J. "Methods for Mapping Entities Between Music Streaming Services", Applied Sciences, 2020, Vol. 10(2), pp. 638-652.
3. Wang X., Chen Y., Yang J. "Cross-platform Music Recommendation System Using Deep Learning Approach", IEEE Transactions on Multimedia, 2022, Vol. 24, pp. 1783-1795.
4. Spotify Web API Documentation [Електронний ресурс] – Режим доступу: https://developer.spotify.com/documentation/web-api/
5. Deezer Developers Documentation [Електронний ресурс] – Режим доступу: https://developers.deezer.com/api

УДК 004.9

ПРОГРАМНИЙ ДОДАТОК ДЛЯ МОБІЛЬНОГО МОНІТОРИНГУ СЕРЦЕВОГО РИТМУ НА ПЛАТФОРМІ iOS

Боднар О.Л.[1)], Манжула В.І.[2)]
Західноукраїнський національний університет
[1)]бакалавр, [2)]д.т.н., професор

Постановка проблеми

У сучасному світі спостерігається стрімке зростання кількості серцево-судинних захворювань, що становить серйозну загрозу для здоров'я населення. Така ситуація потребує розробки нових ефективних підходів до їхнього моніторингу, діагностики та профілактики на ранніх етапах. З огляду на широке розповсюдження смартфонів і пристроїв, які можна носити, особливої актуальності набуває використання мобільних технологій у сфері охорони здоров'я.

Сучасні мобільні пристрої, вже сьогодні, дозволяють здійснювати базові медичні вимірювання, зокрема, частоти серцевих скорочень (ЧСС), без необхідності у придбанні складного чи дорогого обладнання. Одним із перспективних рішень є застосування камери смартфона для фіксації ЧСС шляхом аналізу змін кольору шкіри під час пульсації. Такий метод дозволяє користувачам самостійно, без медичного втручання, здійснювати регулярний контроль стану серцево-судинної системи. Це відкриває нові можливості для своєчасного виявлення проблем та профілактики ускладнень.

Однак, незважаючи на потенціал таких технологій, існує низка викликів: зокрема, питання точності вимірювань, зручності використання у повсякденному житті та доступності інтерпретації отриманих даних для пересічного користувача залишаються відкритими та потребують подальшого дослідження.

Мета дослідження

Метою дослідження є розробка програмного додатку для iOS, який забезпечуватиме безперервний моніторинг серцевого ритму користувача, зберігатиме дані для подальшого аналізу та може надавати попередження про можливі аномалії, спираючись на алгоритми обробки сигналів, а також фіксувати рівень кисню та артеріального тиску вручну.

Структура системи

Розроблений застосунок реалізовано як нативний iOS-додаток із використанням сучасних інструментів та фреймворків, що входять до екосистеми Apple. Для обробки відеопотоку з камери застосовується фреймворк AVFoundation, який дає змогу працювати з вхідним відеосигналом у режимі реального часу, а також забезпечує необхідну продуктивність для зчитування пульсації методом фотоплетизмографії (PPG). Для надсилання локальних сповіщень користувачеві про зміни у стані здоров'я або необхідність вимірювань використовується UserNotifications. Архітектура додатку побудована за принципами MVVM для забезпечення масштабованості та підтримуваності.

До основних компонентів системи входять (див. рис. 1):

Інтерфейс користувача – створений із використанням SwiftUI, що забезпечує реактивність, простоту оновлення стану інтерфейсу та легкість адаптації під різні пристрої. Підтримується темний режим, адаптивне компонування для різних розмірів екранів. Інтерфейс включає в себе графіки частоти серцевих скорочень (ЧСС), зручну статистику за періодами та історію вимірювань, що охоплює також рівень кисню в крові та артеріальний тиск. Усі візуальні елементи розроблено з урахуванням принципів доступності та ергономіки.

Модуль збору даних – відповідає за зчитування сигналу пульсації з використанням задньої камери (rear camera) та спалаху (flash). Методика базується на аналізі фотоплетизмографічного сигналу, що виникає при освітленні пальця користувача. Такий підхід не потребує зовнішніх сенсорів і дозволяє проводити вимірювання швидко та зручно.

Модуль збереження – використовує Core Data як основне рішення для локального зберігання інформації. Користувач має можливість переглядати дані за різні часові періоди: день, тиждень, місяць, що забезпечує глибокий аналіз змін у стані здоров'я. Усі додаткові параметри – такі як рівень кисню та артеріального тиску – вводяться вручну, після чого зберігаються у спільному форматі даних для аналізу.

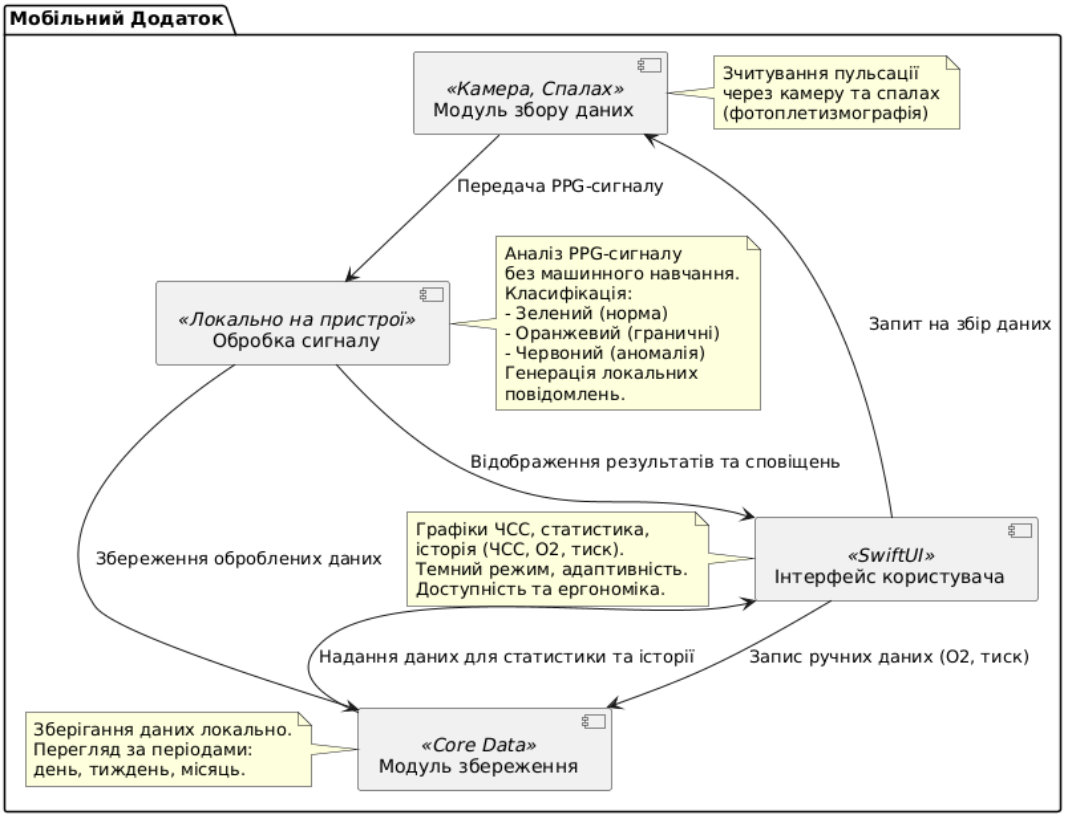


Рисунок 1 – Діаграма компонентів

Обробка сигналу – виконує аналіз зібраного PPG-сигналу локально на пристрої, без використання алгоритмів машинного навчання. Оброблені результати класифікуються за трирівневою шкалою безпеки: зелений – норма, оранжевий – граничні значення, червоний – виявлена аномалія. У випадку виявлення критичних змін система автоматично генерує локальні повідомлення, щоб повідомити користувача про потенційну загрозу або необхідність повторного вимірювання.

Логіка взаємодії компонентів

Взаємодія користувача з додатком побудована по інтуїтивно зрозумілому сценарію (див.рис.2), що дозволяє швидко і без додаткових інструкцій розпочати процес вимірювання.

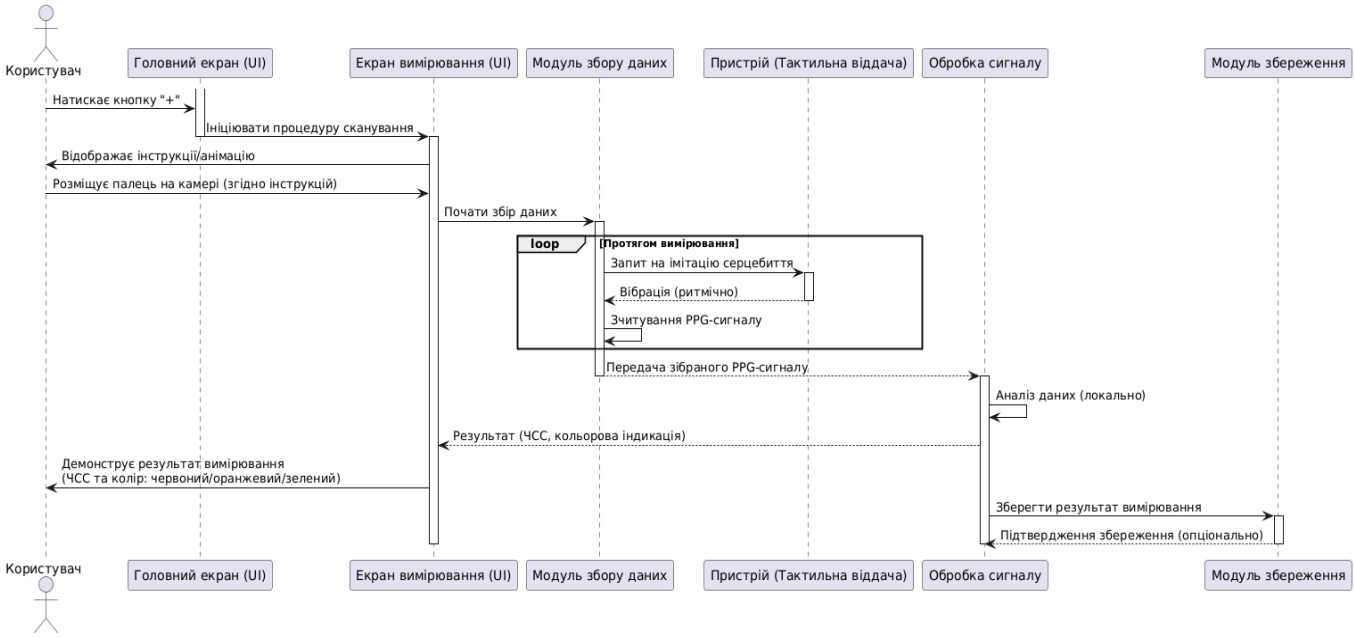


Рисунок 2 – Діаграма послідовності процесу вимірювання ЧСС

Першим кроком є натискання кнопки “+” на головному екрані застосунку, що ініціює запуск процедури сканування. Після цього відкривається спеціалізований екран, на якому користувач бачить інструкції або анімацію, що демонструє правильне розміщення пальця на камері зі спалахом. У процесі вимірювання пристрій використовує тактильну віддачу (вібрацію), яка ритмічно імітує серцебиття, створюючи більш реалістичне та емоційно залучене відчуття процесу. Зібрані дані аналізуються локально. Після завершення короткої фази обробки, яка триває лише кілька секунд, користувачеві одразу демонструється результат вимірювання. ЧСС виводиться як числове значення, а також у вигляді кольорової індикації, що дозволяє швидко зорієнтуватися у стані здоров’я: червоний колір сигналізує про критичні відхилення і потенційну загрозу, оранжевий – про граничні значення, які вимагають уваги, зелений – про нормальні показники. Такий підхід забезпечує як інформативність, так і візуальну доступність навіть для недосвідчених користувачів.

Окремий розділ додатку передбачає перегляд історії вимірювань (див.рис. 3) – як графічно (лінійні графіки), так і списком за датою, що дає змогу користувачеві простежити динаміку зміни ЧСС, а також вручну введених показників (кисень, тиск). Історія представлена зручним фільтром за періодами (день, тиждень, місяць). Наразі функціонал експорту даних або їхнього обміну з іншими сервісами чи пристроями не реалізовано, що обумовлено як потребою зберегти конфіденційність, так і фокусом на автономному використанні додатку.

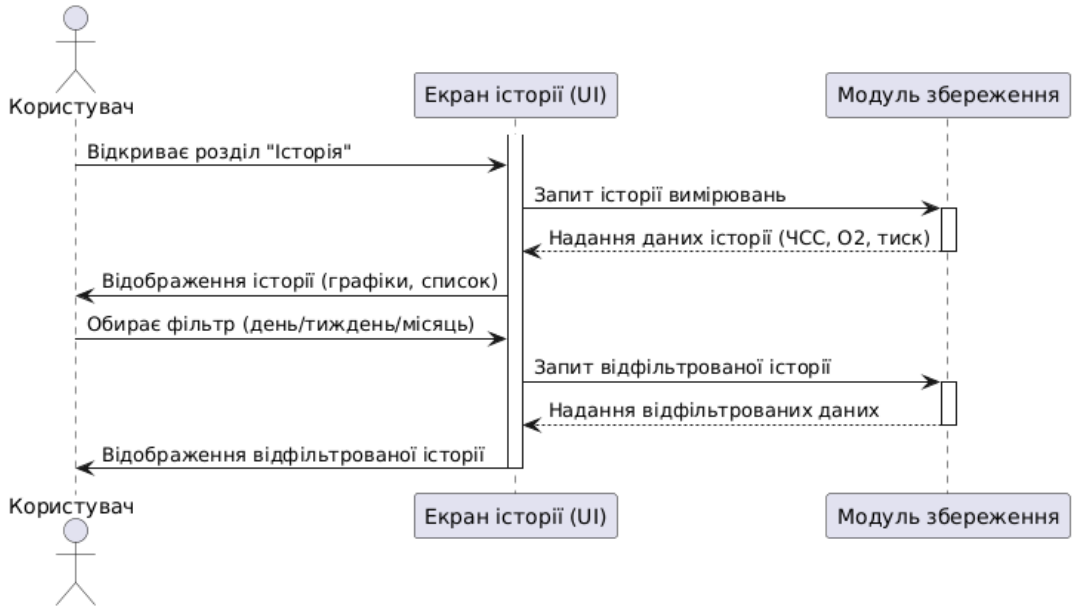


Рисунок 3 – Діаграма послідовності перегляду історії вимірювань

Висновки

Розроблений додаток забезпечує зручний та доступний спосіб самостійного моніторингу ЧСС без потреби у додатковому обладнанні, а також демонструє ефективність використання сучасних засобів iOS для мобільного моніторингу життєвих показників. Перевагами даного додатку є офлайн-робота, простота інтерфейсу та інтерактивна візуалізація результатів. Використання SwiftUI, CoreData та HealthKit дозволило створити надійне рішення з високим рівнем інтеграції в екосистему Apple. Подальші дослідження можуть включати розширення функціональності за допомогою машинного навчання для прогнозування ризиків.

Список використаних джерел

1. Apple HealthKit Documentation [Електронний ресурс]. – Режим доступу: https://developer.apple.com/documentation/healthkit.
2. SwiftUI. Declarative Framework for Building UI [Електронний ресурс]. – Режим доступу: https://developer.apple.com/xcode/swiftui/.
3. AVFoundation Framework [Електронний ресурс]. – Режим доступу: https://developer.apple.com/av-foundation/.
4.UserNotifications Framework [Електронний ресурс]. – Режим доступу: https://developer.apple.com/documentation/usernotifications.
5. Core Data Programming Guide [Електронний ресурс]. – Режим доступу: https://developer.apple.com/documentation/coredata .
6. Photoplethysmography: Technology, Signal Processing, and Applications [Електронний ресурс]. – Режим доступу: https://doi.org/10.1016/B978-0-12-819456-4.00002-2.
7. WHO. Cardiovascular diseases (CVDs) [Електронний ресурс]. – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/cardiovascular-diseases-(cvds).

УДК 004.8

ERP-СИСТЕМА ДЛЯ ВНУТРІШНІХ БІЗНЕС-ПРОЦЕСІВ КОМПАНІЇ

Геремій І.С.[1)], Писар Б.Р.[2)]
Західноукраїнський національний університет
[1)] викладач; [2)] студент

І. Постановка проблеми

У сучасних умовах цифрової трансформації компанії стикаються з необхідністю автоматизації внутрішніх процесів для підвищення ефективності та контролю. Одним з найактуальніших завдань є організація зручної, прозорої та контрольованої системи обробки звернень працівників - довідок, запитів, заявок тощо. У більшості компаній такі процеси досі виконуються вручну, через електронну пошту або месенджери, що унеможливлює системний облік, аналітику та контроль виконання.

Іншою проблемою є обмежена доступність єдиного середовища, в якому співробітники можуть взаємодіяти з адміністративними, бухгалтерськими чи HR-службами, не витрачаючи час на особисті звернення чи неструктуровану комунікацію.

ІІ. Мета роботи

Метою роботи є розробка та впровадження внутрішньої ERP-системи для автоматизації бізнес-процесів компанії з акцентом на управління зверненнями (Help-Desk), створення та облік довідок, забезпечення єдиного каналу комунікації між працівниками та адміністрацією компанії.

ІІІ. Основна частина

ERP-система розробляється з орієнтацією на внутрішні потреби середньої компанії. В її основу покладено два ключові модулі: Help-Desk для обробки внутрішніх запитів працівників та довідковий модуль для автоматизації процесу створення та видачі довідок (про зарплату, зайнятість тощо).

Help-Desk модуль реалізує наступні функції:

- Створення заявок працівниками з вибором категорії проблеми;
- Призначення відповідального менеджера;
- Контроль термінів виконання;
- Зворотній зв’язок через коментарі;
- Статистика обробки запитів.

Схематично функції модуля представлені на рисунку1.

Модуль довідок включає:

- Шаблонізацію типових документів;
- Автоматичне підвантаження даних працівника (прізвище та ім’я, посада);
- Відстеження статусу запиту на довідку;
- Можливість завантажити або роздрукувати готовий документ.

ERP-система складається з множини взаємопов’язаних сервісів, кожен з яких відповідає за окрему бізнес-функцію. Для забезпечення ефективної взаємодії між frontend-частиною та мікросервісною архітектурою застосовується Gateway - центральний маршрутизатор та координаційний компонент системи.

IdentityServer-один із ключових мікросервісів системи, що відповідає за управління обліковими даними користувачів, їх груповою належністю та загальною структурою компанії. Цей сервіс забезпечує централізоване зберігання інформації про працівників, що використовується більшістю інших мікросервісів ERP-системи.

Notification-мікросервіс відповідальний за створення, збереження та доставку сповіщень у режимі реального часу. Сервіс забезпечує інформування користувачів про події, що відбуваються в системі, за допомогою WebSocket-з'єднання, а також надає API для взаємодії зі сповіщеннями.

Tasks-мікросервіс відповідальний за управління завданнями в ERP-системі. Він забезпечує створення, редагування, видалення, сортування та фільтрацію завдань, а також підтримує взаємодію з іншими сервісами, зокрема Projects, до якого належить розподіл завдань у межах проектів.

Business Processes-мікросервіс, який відповідає за агрегацію всіх бізнес-процесів у єдину централізовану систему.

Certificate Request - мікросервіс, що відповідає за створення службового документа для працівника на основі поданої заявки.

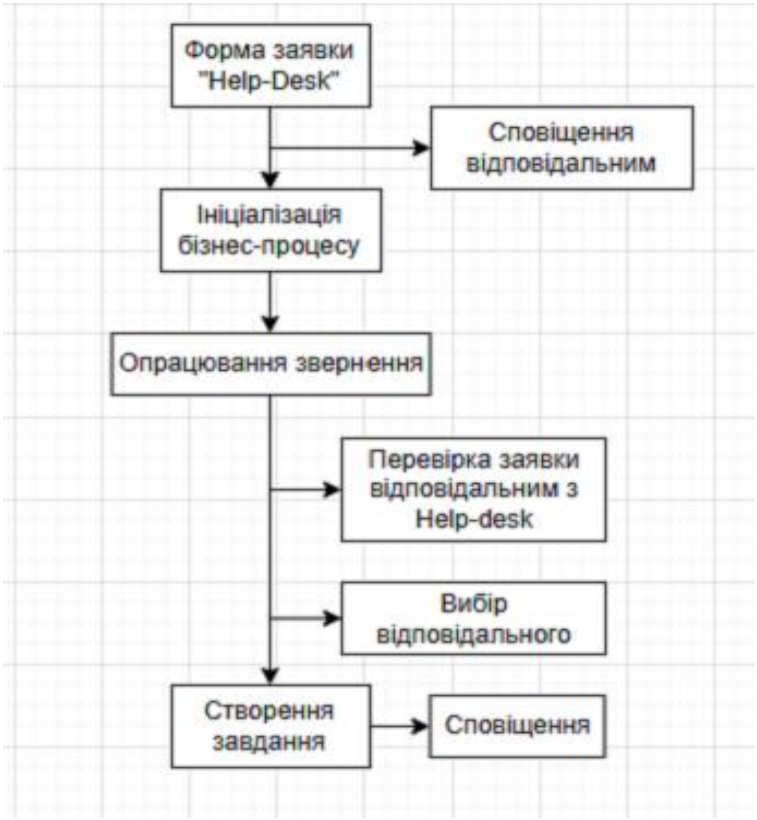


Рисунок 1 – Діаграма функцій процесу “Help-Desk”

Окрім основних, система містить також допоміжні мікросервіси, які розширюють функціональні можливості ERP та забезпечують підтримку внутрішніх процесів. Нижче наведено короткий опис таких сервісів:

- Absence - мікросервіс для обліку відсутності працівників. Підтримує різні типи відсутності: відпустка, відрядження, лікарняний, віддалена робота тощо.

- Calendar - мікросервіс, що відповідає за бронювання конференц-залів та організацію зустрічей. Також взаємодіє з мікросервісами Notification (для внутрішніх сповіщень) та Scheduler (для нагадувань про події).

- News - мікросервіс для управління новинами та внутрішніми оголошеннями в системі ERP. Забезпечує інформування працівників про важливі події, оновлення та зміни.

- Chat - мікросервіс для внутрішньої комунікації між працівниками. Дозволяє здійснювати обмін повідомленнями в межах ERP-системи.

- Dynamic Lists - мікросервіс для створення динамічних списків, які можуть використовуватись для ведення обліку, формування документів або збору певних типів інформації.

Завдяки впровадженню ERP-системи значно зменшується навантаження на адміністративні служби, підвищено швидкість обробки звернень та прозорість виконання завдань. Користувачі мають можливість бачити статус своїх запитів в реальному часі, а керівництво — отримувати статистику та звіти.

Висновок

Розроблена ERP-система дозволяє ефективно вирішувати ключові внутрішні завдання компанії, зменшуючи адміністративне навантаження та підвищуючи прозорість бізнес-процесів. Вона особливо корисна для оптимізації Help-Desk процесів і автоматизації видачі довідок. Подальший розвиток системи може включати інтеграцію з зовнішніми сервісами (наприклад, e-mail, календарі), розширення звітності та впровадження мобільної версії.

Список використаних джерел

1. Investopedia — "Enterprise Resource Planning (ERP)" [Електронний ресурс]. – Режим доступу: https://www.investopedia.com/terms/e/erp.asp
2. IBM – What is ERP (Enterprise Resource Planning) [Електронний ресурс]. – Режим доступу: https://www.ibm.com/topics/enterprise-resource-planning
3. Oracle – What is ERP [Електронний ресурс]. – Режим доступу: https://www.oracle.com/erp/what-is-erp/
4.SAP SE. Офіційний сайт SAP ERP [Електронний ресурс]. – Режим доступу:. https://www.sap.com/ukraine/products/erp.html

УДК 004.4

ВЕБ-ДОДАТОК ДЛЯ ПІДБОРУ ТА БРОНЮВАННЯ ПОДОРОЖЕЙ

Гриб В.В.
Західноукраїнський національний університет
бакалавр

Постановка проблеми

Кожен з нас стикався з потребою планувати подорож – для себе, родини чи друзів, на відпустку чи вікенд. Часто важко обрати: популярний курорт чи маловідоме місце, літак чи автівка, готель чи будиночок у природі. Ми переглядаємо десятки сайтів, читаємо відгуки, порівнюємо ціни, шукаємо найкращі варіанти. Це забирає чимало часу, який можна було б зекономити, якби всі інструменти для планування й бронювання були на одній платформі.

Хоч інтернет-платформи пропонують багато варіантів подорожей, більшість із них не дає змоги обрати тур за заданими параметрами – типом відпочинку, бюджетом чи тривалістю. Це ускладнює планування і вимагає більше часу. Автоматичний підбір турів за введеними критеріями міг би значно полегшити пошук і допомогти швидше знайти ідеальний варіант.

Мета роботи

Метою роботи є створення веб-застосунку, за допомогою якого можна буде обрати подорож на різноманітні випадки життя та для будь-якого бюджету. Користувач матиме можливість забронювати тур, вибрати зручні варіанти, транспорту та житла відповідно до власних уподобань, заповнити анкету для кращого підбору подорожі з урахуванням інтересів, побажань та попереднього досвіду. Крім того є можливість запропонувати власні маршрути для майбутніх поїздок, що сприятиме розвитку спільноти мандрівників.

Основна частина

У рамках розробки системи було здійснено детальний аналіз існуючих цифрових рішень у сфері туристичних онлайн-сервісів, орієнтованих на підбір, бронювання та персоналізацію подорожей. Зокрема, були розглянуті такі сервіси, як Farvater та Viki-tour. Кожен із них має певні переваги: наприклад, Farvater забезпечує широкий вибір турів та житла з інтеграцією бронювання, а Viki-tour дозволяє швидко організувати маршрути на основі інтересів користувача. Водночас було виявлено й низку істотних недоліків. Найпоширенішими обмеженнями вказаних систем є відсутність гнучкої адаптації турів під унікальні потреби користувача та обмежені можливості щодо подальшої модифікації заброньованих турів. Крім того, в більшості сервісів не передбачена взаємодія з персональними менеджерами або автоматизованими порадами щодо вибору подорожі, заснованими на анкетуванні або попередній історії бронювань.

З урахуванням зазначених проблем було розроблено власну систему для підбору та бронювання подорожей, що має на меті забезпечити не лише зручний і швидкий пошук турів, а й персоналізацію, гнучке керування обраними турами, інтерактивність та зворотній зв’язок. Структура системи складається з кількох ключових модулів, кожен з яких виконує окрему роль:

Модуль перегляду турів відповідає за формування списку доступних пропозицій з можливістю сортування та фільтрації за ціною, тривалістю, місцем розташування тощо. Передбачена також опція пошуку за ключовими словами.

Модуль бронювання дозволяє користувачеві здійснити бронювання обраного туру, вказати кількість осіб, зберегти замовлення в особистому кабінеті та отримати підтвердження елеткронною поштою.

Модуль обраних турів забезпечує збереження подорожей, які зацікавили користувача, з можливістю швидкого доступу до них, порівняння иа подальшого бронювання.

Модуль історії бронювань містить усі попередні замовлення користувача із зазначенням статусу, дати, кількості осіб і вартості.

Модуль відгуків дозволяє користувачам залишати коментарі до подорожей, оцінювати тури за 5-бальною шкалою, редагувати або видаляти свої відгуки, а також переглядати думки інших.

База даних, реалізована на основі MariaDB, зберігає всі дані про користувачів, тури, бронювання, історію дій, відгуки, обрані тури, а також логіку ролей (звичайний користувач, адміністратор).

Клієнтська частина – це сучасний веб-застосунок із адаптивною верстkою, що дозволяє користувачу інтуїтивно взаємодіяти з платформою: переглядати деталі туру, керувати бронюваннями,

залишати відгуки, редагувати профіль, отримувати сповіщення про знижки, брати участь у тематичних подорожах

На етапі проєктування архітектури було створено діаграму варіантів використання (рисунок 1), яка демонструє логіку взаємодії між основними сутностями системи: користувач, тур, обране, бронювання, відгук тощо.

Завдяки комплексному аналізу існуючих рішень та врахуванню виявлених недоліків, розроблена система поєднує в собі найкращі практики сучасних туристичних онлайн-сервісів з інноваційними підходами до персоналізації. Особлива увага приділена покращенню користувацького досвіду. Подальший розвиток системи передбачає інтеграцію штучного інтелекту для аналізу поведінкових даних та покращення рекомендаційного механізму, що відкриває нові перспективи для підвищення ефективності сервісу.

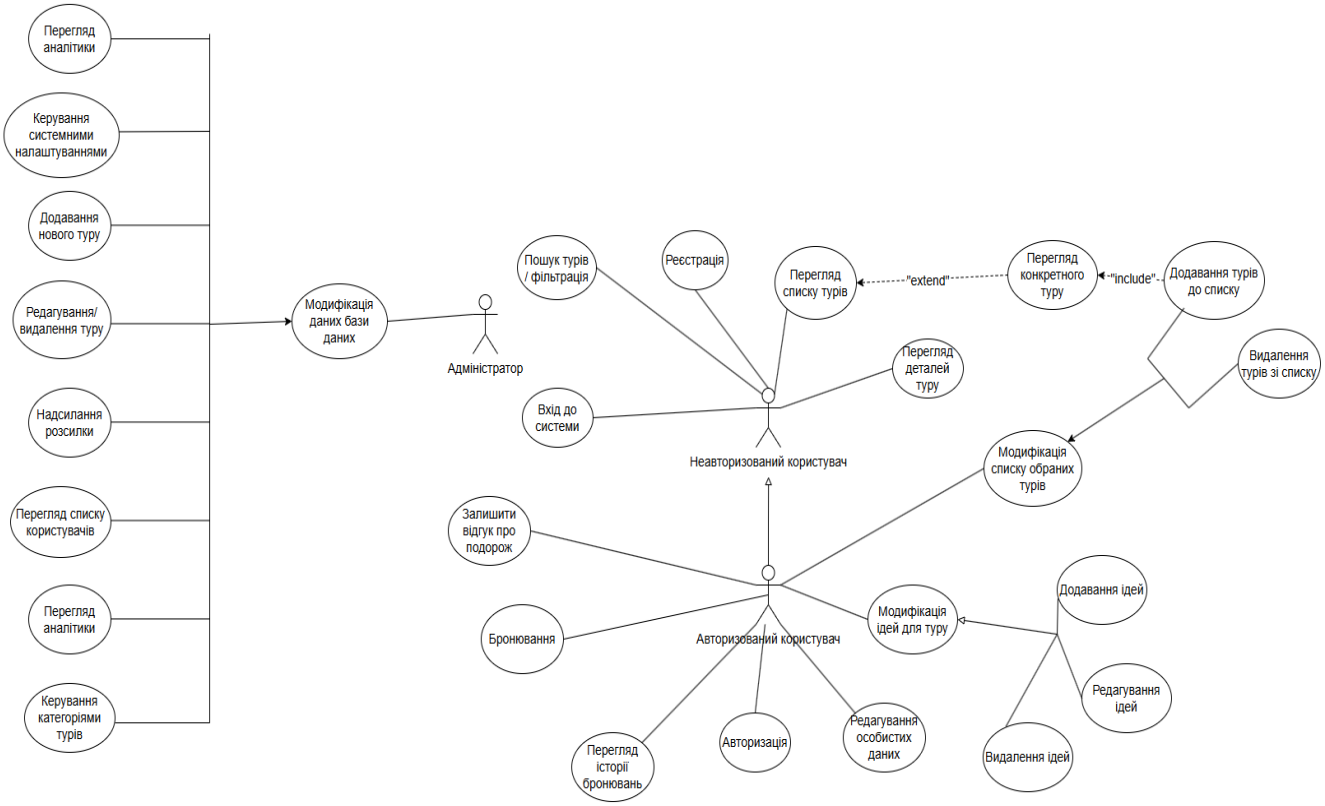


Рисунок 1 – Діаграма варіантів використання

Для реалізації серверної частини системи було обрано мову програмування PHP, яка дозволяє створювати надійні та масштабовані веб-сервіси. Обробка запитів користувача реалізована через контролери, що взаємодіють із реляційною базою даних для збереження даних про тури,бронювання, відгуки та профілі користувачів.

Клієнтська частина реалізована з використанням HTML, CSS та JavaScript, що дозволяє створити адаптивний, швидкий та інтуїтивно зрозумілий інтерфейс. Основна інформаційна панель містить можливість пошуку подорожей, перегляду деталей туру, додавання до обраного а також здійснення онлайн-бронювання.

Висновок

Розроблена система підбору та бронювання турів дозволяє користувачам ефективно планувати свої подорожі, обираючи з широкого переліку доступних напрямків відповідно до уподобань і бюджету. Система забезпечує зручний інтерфейс для перегляду турів, фільтрацію за параметрами, бронювання, ведення історії поїздок та залишення відгуків. Її використання є зручною для мандрівників, сприяє економії часу при виборі туру та загальному покращенню взаємодії між клієнтами і туристичними операторами.

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: навчальнийпосібник, 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.

УДК 004.4

ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ СТВОРЕННЯ ПЕРСОНАЛІЗОВАНИХ ВІДЕО ДЛЯ БІЗНЕСУ

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

І. Постановка проблеми

У сучасному бізнес-середовищі надзвичайно важливо швидко привертати увагу клієнтів і виділятися серед конкурентів. Особливо це стосується процесів холодного аутрічу, де перше враження має вирішальне значення. Традиційні способи комунікації, як-от електронна пошта або текстові повідомлення, часто не досягають бажаного ефекту. Персоналізовані відео, створені за допомогою штучного інтелекту, відкривають нові можливості для бізнесу у сфері комунікацій.
Використовуючи аватари на базі Microsoft Azure та алгоритми генерації відео, можна створити систему, яка дозволяє автоматизовано генерувати унікальні відеозвернення до клієнтів. Це забезпечує як персоналізацію, так і високу якість контенту, що критично важливо для встановлення довіри та збільшення конверсій.

ІІ. Мета роботи

Метою даної роботи є дослідження методів створення персоналізованого відеоконтенту за допомогою штучного інтелекту, аналіз існуючих рішень у сфері відеогенерації для бізнесу, розробка архітектури програмного забезпечення для формування відеозвернень, а також реалізація системи, що дозволяє компаніям створювати персоналізовані відео з використанням Azure AI Avatars.

ІІІ. Архітектура системи та основні принципи генерації відео

Робота системи базується на поетапній побудові відеоконтенту з використанням готових елементів та інтеграції сервісів штучного інтелекту. Архітектура передбачає взаємодію кількох основних компонентів: модуля вибору аватара, інтерфейсу налаштування фону, генератора тексту, модуля озвучення та сервісу об’єднання всіх елементів у єдиний відеофайл.

Користувач у взаємодії з інтерфейсом формує вхідні дані: задає бажаний фон, обирає візуальний образ віртуального персонажа та голос для озвучення. Система також надає можливість згенерувати текст за допомогою мовної моделі, після чого він адаптується під обраний голос. Після цього всі компоненти автоматично обробляються та компілюються у відео.

Побудова відео здійснюється як єдиний процес на основі попередньо обраних параметрів, що забезпечує узгодженість між усіма елементами — фоном, аватаром, голосом і текстом. Такий підхід дозволяє досягти стабільної якості результату та уникнути технічних ускладнень, пов'язаних із редагуванням окремих частин відео. Завдяки цьому система залишається простою у використанні, забезпечуючи водночас достатню гнучкість для створення варіативного контенту шляхом зміни початкових налаштувань перед генерацією.

ІV. Програмна реалізація

Реалізація системи базується на поєднанні кількох сучасних технологій, кожна з яких виконує окрему функцію в процесі створення персоналізованого відео. Основною частиною є модуль генерації відео аватара, у якому використовується сервіс Azure AI Avatars. Він дозволяє формувати відеофрагменти з віртуальним персонажем, що озвучує заданий текст у синхронізації з мімікою.

Для генерації текстового сценарію передбачена інтеграція з мовною моделлю від OpenAI, яка дозволяє користувачеві отримувати згенеровані тексти для відеозвернень на основі короткого запиту чи опису. Згенерований текст передається у подальший ланцюг обробки.

Після вибору всіх параметрів (аватар, фон, голос, текст), відео збирається за допомогою бібліотеки Remotion, яка дозволяє програмно керувати відео рендерингом у середовищі JavaScript. Саме ця бібліотека відповідає за об’єднання аватара з вибраним фоном і аудіо озвученням у фінальний відеофайл.

Фронтенд частина реалізована з використанням React, що забезпечує динамічну взаємодію користувача з інтерфейсом. Для побудови серверної частини застосовується Fastify, легкий і продуктивний фреймворк для Node.js. Робота з базою даних організована через бібліотеки Objection.js та Knex.js, а самі дані зберігаються у PostgreSQL. Така архітектура дозволяє ефективно керувати користувацькими сесіями та параметрами відео.

На рисунку 1 зображена сторінка відеоредактора для налаштування параметрів відео.

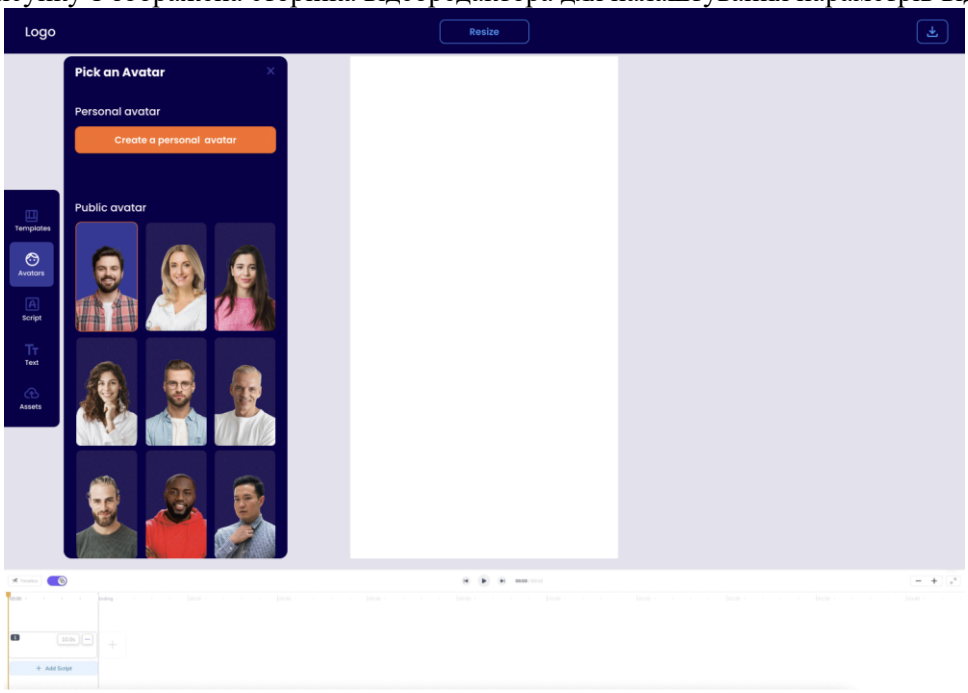


Рисунок 1 – Сторінка відеоредактора

На рисунку 2 зображено головну сторінку з усіма відео які створив користувач.

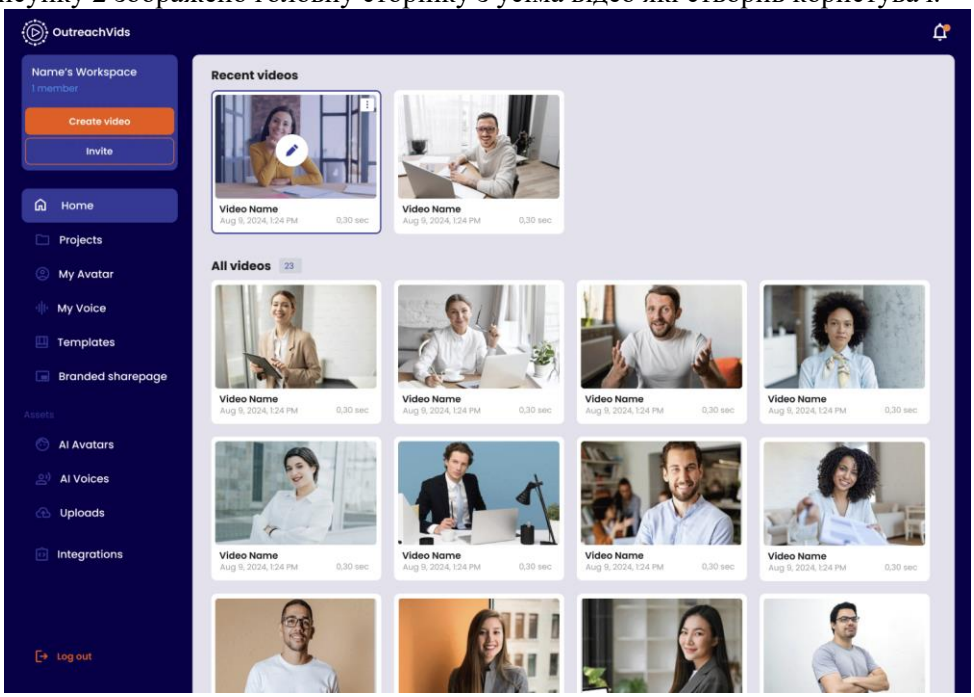


Рисунок 2 – Головна сторінка з створеними відео

Висновок

Розроблена система дозволяє значно покращити ефективність бізнес-комунікацій завдяки високому рівню персоналізації та залученості. Впровадження такого інструменту дає змогу підприємствам виділятися серед конкурентів, покращити якість взаємодії з потенційними клієнтами та підвищити конверсію холодного аутрічу.

Список використаних джерел

1. OpenAI API Reference [Електронний ресурс]. Режим доступу:https://platform.openai.com/docs/api-reference/introductionhttps://platform.openai.com/docs/api-reference/introduction
2. Microsoft. Text-to-SpeechAvataroverview: Azure AI Services. [Електронний ресурс]. - Режим доступу: https://learn.microsoft.com/azure/ai-services/speech-service/text-to-speech-avatar/what-is-text-to-speech-avatar
3. Remotion AG. MakevideosprogrammaticallywithReact [Електронний ресурс]. Режим доступу: https://www.remotion.dev/docs/

УДК 681.3

РОЗРОБКА ВЕБ-СИСТЕМИ ДЛЯ ПІДТРИМКИ НАВЧАННЯ ПРОГРАМУВАННЮ

Парій А.С.
Західноукраїнський національний університет
бакалавр

І. Постановка проблеми

У сучасному освітньому процесі навчананя програмуванню є однією з ключових складових підготовки фахівців ІТ-сфери. Однак існуючі методи часто не забезпечують достатньої інтерактивності, зручності та ефективного зворотного зв'язку між студентами та викладачами. В умовах обмежених ресурсів у закладах освіти – відсутності інтерактивних дошок, сучасних лабораторій, нестабільного доступу до програмних середовищ – особливо актуальним є створення інструментів, які б дозволяли навчати програмування у віддаленому або гібридному форматі.

Існуючі онлайн-редактори коду (наприклад, Collabedit, JSFiddle, Codeanywhere) мають низку обмежень: відсутність структури для організації навчального процесу, недостатня підтримка контролю версій, неможливість оцінювання виконаних завдань, обмежена система ролей та доступу. Ці сервіси здебільшого орієнтовані на розробників, а не на освітній процес. Враховуючи вищевикладене, виникає необхідність у створенні власної вебсистеми, адаптованої до потреб викладачів і студентів у процесі вивчення мов програмування.

ІІ. Мета роботи

Метою роботи є розробка вебдодатку, який дозволить забезпечити ефективну взаємодію між викладачем та студентом під час навчання програмуванню. Основні функціональні можливості, що реалізуються в системі: створення та редагування коду в реальному часі, формування та призначення завдань, контроль за їх виконанням, вбудований чат для комунікації, формування звітів та рейтингових таблиць.

Технічною метою є реалізація системи за допомогою сучасних вебтехнологій, зокрема Node.js та MongoDB. Система повинна бути зручною, адаптивною до різних типів пристроїв, а також такою, що не потребує встановлення додаткового ПЗ на клієнтському боці, що робить її придатною для широкого кола освітніх установ.

ІІІ. Методика реалізації

Система розроблена з використанням класичної клієнт-серверної архітектури. Серверна частина реалізована з використанням середовища Node.js та фреймворку Express.js, що забезпечує асинхронну обробку запитів і масштабованість. Для зберігання даних використано MongoDB – документоорієнтовану СУБД, яка ідеально підходить для гнучкого моделювання складних об'єктів та зв'язків між ними. Роботу з базою реалізовано за допомогою бібліотеки Mongoose, що дозволяє чітко описати структури сутностей та зв'язки між ними на рівні схем.

Основними об'єктами системи є: «Користувач», «Задача», «Сесія редагування», «Роль», «Група». Кожен об'єкт має визначені поля, типи даних і асоціації. Користувачі можуть мати різні ролі, що регламентують їхні можливості у системі: створення задач, їх призначення, перегляд або редагування рішень.

На рисунку 1 зображено логічну схему взаємозв'язків між ключовими об'єктами веб-системи, зокрема між студентами, задачами, ролями та сесіями редагування. Така структура бази даних забезпечує простоту доступу до необхідної інформації, дозволяє ефективно зберігати і структурувати дані, а також забезпечує масштабованість системи.

Окрім логічної моделі, для наочного уявлення внутрішньої реалізації додатку, розроблено схему компонентів та потоків обробки даних. На рис. 2 представлено загальну архітектуру програмної реалізації системи, яка ілюструє, як користувач через інтерфейс взаємодіє з сервером і базою даних, а також які компоненти відповідають за обробку запитів.

Реалізовано зручний інтерфейс, який підтримує перегляд і редагування коду з підсвічуванням синтаксису, перегляд змін в історії правок, а також коментування фрагментів коду. Для комунікації реалізовано інтегрований чат з підтримкою групових обговорень. Також передбачено систему аналітики – формування індивідуальних звітів про успішність студентів, рейтингів і списків завдань з оцінками. Розгортання системи може здійснюватися як локально, так і в хмарному середовищі.

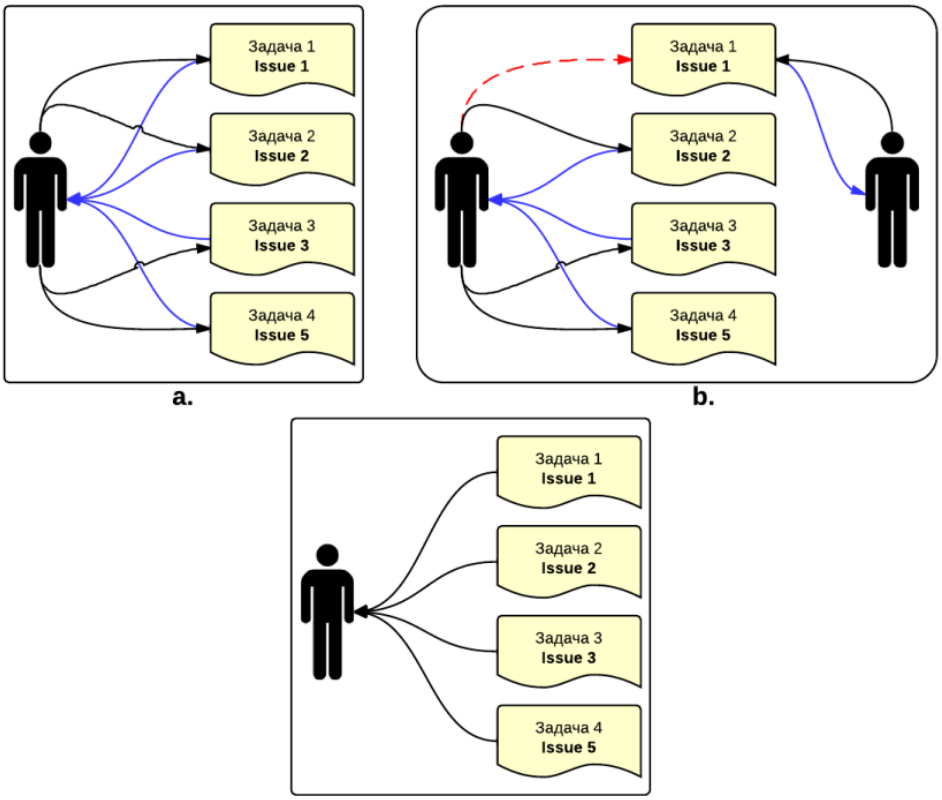


Рисунок 1 – Структура зв'язків між об'єктами в базі даних

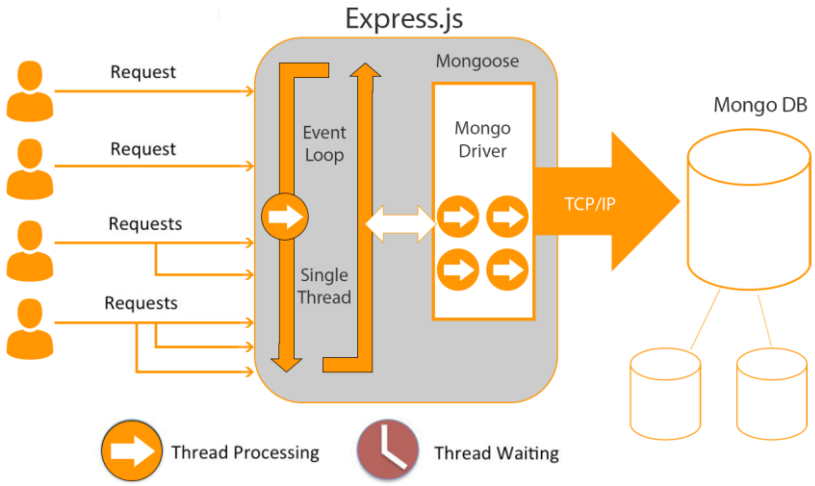


Рисунок 2 – Загальна схема програмної реалізації веб-додатку

Висновок

Розроблена вебсистема є інноваційним рішенням для організації навчання програмуванню у сучасному освітньому середовищі. Вона дозволяє не лише замінити або доповнити традиційні форми навчання, а й покращити зворотний зв'язок, забезпечити гнучкість і прозорість навчального процесу. Створений інструмент може бути адаптований до різних навчальних дисциплін, масштабований та інтегрований у вже існуючі системи дистанційного навчання. Система дозволяє викладачам контролювати якість виконання завдань, а студентам — здобувати практичні навички у зручному онлайн-середовищі. Надалі її можна доповнити автоперевіркою коду, інтеграцією з LMS (наприклад, Moodle), а також впровадити інструменти оцінювання за допомогою тестів або модульного контролю..

Список використаних джерел

1. Палієнко В.М. Основи об'єктно-орієнтованого програмування. – К.: Вища школа, 2020. – 256 с.
2. Звіт про розвиток дистанційного навчання в Україні, 2023. – [Електронний ресурс]. – Режим доступу: https://mon.gov.ua
3. MongoDB Documentation – [Електронний ресурс]. – https://www.mongodb.com/docs/
4. Node.js Documentation – [Електронний ресурс]. – https://nodejs.org

УДК 004.4

АНАЛІТИЧНА СИСТЕМА ДЛЯ АНАЛІЗУ КІБЕРСПОРТИВНОЇ ГРИ DOTA 2

Порплиця Н.П.[1)], Буркацький О.А.[2)] , Стасів І.С.[3)]
Західноукраїнський національний університет
[1)]к.т.н., доцент; [2)] бакалавр; [3)]к.т.н., доцент

I. Постановка проблеми

У світі сучасного кіберспорту надзвичайно важливим є вміння швидко реагувати на зміни ігрової мети, розуміти, які герої перебувають у топі, а які втрачають актуальність, та приймати рішення, спираючись на перевірені статистичні дані. Особливо це стосується гравців з високим рівнем MMR, для яких одна неправильна дія навіть під час вибору героя може коштувати перемоги. Попри наявність різноманітних аналітичних платформ, багато з них перевантажені зайвою інформацією або мають заплутаний інтерфейс, що створює зайві бар'єри для швидкого аналізу.

Розроблена система аналітики Dota 2 орієнтована на те, щоб надати користувачам доступ до чітко структурованих, візуально зручних і персоналізованих даних про героїв, статистику матчів і тренди мети. Завдяки інтеграції з відкритими API, зокрема OpenDota, а також впровадженню власного алгоритму оцінки ефективності героїв (метрика d2pt), система формує таблиці, рейтинги й графіки, які допомагають швидко ухвалювати зважені рішення. Це рішення стане корисним як для гравців, так і для тренерів, аналітиків чи стримерів, яким важливо залишатися в курсі останніх змін на професійній сцені та пристосовуватись до нових умов гри.

II. Мета роботи

Основною метою проєкту є дослідження способів збору, обробки та подання аналітичної інформації для гри Dota 2. У межах дослідження розглянуто існуючі сервіси, проаналізовано їхні переваги й недоліки, а також розроблено програмне рішення, що дає змогу ефективно відслідковувати зміни в меті, оцінювати героїв та візуалізувати дані в зручному вебінтерфейсі.

III. Архітектура системи та основні принципи генерації відео

Функціонування системи базується на покроковому зборі й обробці відкритої статистики з API OpenDota. Архітектура складається з кількох основних блоків. Модуль збору даних який надсилає запити до OpenDota, отримує "сирі" дані про матчі, героїв, пік-рейти тощо. Далі серверна частина обробляє запити, виконує агрегацію статистики, кешує результати й формує відповіді для клієнта. База даних використовується для зберігання мета-статистики, історії матчів та рейтингів (реалізована на MongoDB). Реалізація клієнтської частини відповідає за візуалізацію: графіки, таблиці, інтерфейс користувача.

Користувач може застосовувати фільтри: за героями, ролями, частотою вибору, переможністю тощо. Вбудовані інструменти дозволяють швидко побачити зміни в меті, визначити найефективніших персонажів, а також зрозуміти, як ті чи інші герої виступають у професійних матчах.

Важливою складовою є власна метрика d2pt, яка враховує не лише winrate чи частоту пікання, а й стабільність результатів у різних рейтингових діапазонах. Це дозволяє створити більш об'єктивну оцінку ефективності героїв.

IV. Програмна реалізація

Реалізація системи базується на поєднанні кількох сучасних технологій, кожна з яких виконує окрему функцію в процесі збору, обробки та візуалізації аналітичних даних з гри Dota 2. Основною частиною є серверна платформа на базі Node.js, яка реалізує RESTful API через фреймворк Express. Саме вона відповідає за обробку запитів, комунікацію з API OpenDota [1] та передачу обробленої інформації на клієнтську частину.

Для зберігання даних використовується MongoDB, що дозволяє ефективно кешувати мета-статистику, історію матчів та рейтинг героїв. Запити до бази реалізовані з використанням Mongoose, що забезпечує просту й масштабовану роботу з колекціями даних.

Фронтенд реалізовано за допомогою React [3], що забезпечує швидку взаємодію користувача з інтерфейсом, а також гнучке оновлення компонентів при зміні фільтрів чи параметрів відображення. Для побудови динамічних графіків, таких як winrate, matches, d2pt рейтинг чи contestrate, використовується бібліотека Recharts. Окрім цього, в UI реалізована система пагінації, пошуку та адаптивної фільтрації, що значно покращує зручність користування.

Завдяки чітко розділеній архітектурі, система залишається гнучкою, легко масштабується і готова до подальшого розширення — наприклад, додавання аналітики за турнірами, нових метрик або авторських рейтингів героїв.

На рисунку 1 представлено приклад сторінки героя з динамічними графіками, що дозволяють відстежити зміни його ефективності у матчах високого рівня

Рисунок 1 – Скриншот сторінки окремого героя з графіками

На рисунку 2 зображено таблицю з аналітикою героїв за ключовими метриками за останні 30 днів.

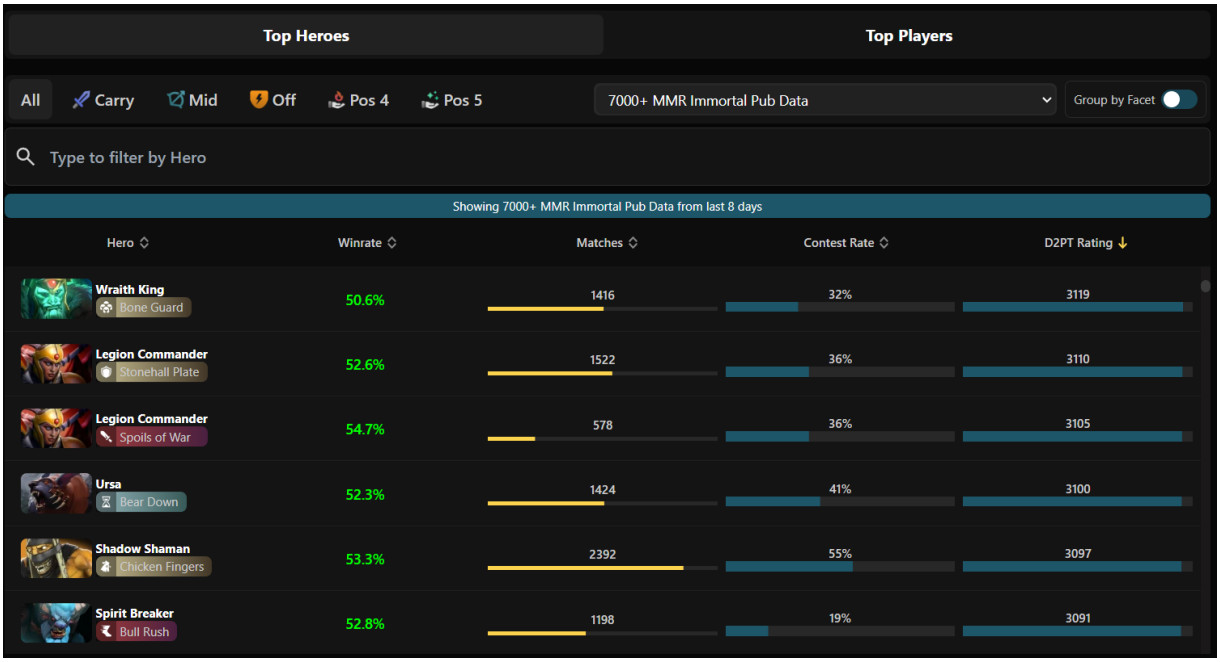


Рисунок 2 – Скриншот загального дашборду з таблицею героїв

Висновок

Розроблене рішення демонструє, як за допомогою публічних API та сучасних вебтехнологій можна створити зручний інструмент для аналізу мети в Dota 2. Воно дозволяє гравцям адаптуватися до змін, приймати обґрунтовані рішення при виборі героїв, а також відстежувати динаміку професійної сцени. Гнучка архітектура платформи забезпечує можливість її подальшого розширення як у напрямку глибшої турнірної аналітики, так і для впровадження нових алгоритмів прогнозування чи машинного навчання.

Список використаних джерел

1. OpenDota API Documentation [Електронний ресурс]. – Режим доступу:https://platform.openai.com/docs/api-reference/introductionhttps://docs.opendota.com
2. ValveCorporation. Dota 2 Web API Overwiew [Електроний ресурс]. – Режим доступу: https://developer.valvesoftware.com/wiki/Steam_Web_API
3. Meta. React – A Java Script library for building user interfaces [Електронний ресурс]. – Режим доступу: https://react.dev/docs/getting-started

УДК 004.4

ПРОГРАМНИЙ МОДУЛЬ АНАЛІЗУ ВІДЕОПОТОКІВ З ВИКОРИСТАННЯМ LLM МЕРЕЖ

Петровський Ю.М.[1)], Пукас А.В.[2)]
Західноукраїнський національний університет
[1)] бакалавр, [2)] д.т.н., професор

I. Постановка проблеми

У сучасному світі спостерігається стрімке зростання споживання медіа-контенту через Over-The-Top (OTT) платформи. Зростаюче навантаження на програмну інфраструктуру моніторингу, підвищення вимог до якості відеопотоків, а також необхідність автоматизації процесів контролю потребують ефективних програмних рішень для управління медіа-контентом. У багатьох випадках розробники програмного забезпечення стикаються з відсутністю готових інструментів для створення систем моніторингу якості відеопотоків з інтеграцією штучного інтелекту.

Однією з ключових проблем сучасної індустрії цифрових медіа є забезпечення стабільної якості відеопотоків в умовах постійно зростаючого обсягу контенту та різноманітності технічних параметрів передачі. Традиційні методи контролю якості відео часто вимагають значних людських ресурсів та не завжди здатні оперативно виявляти критичні проблеми, що може призводити до погіршення користувацького досвіду та фінансових втрат для провайдерів контенту.

Сучасні системи потокового відео працюють з величезними обсягами даних, що передаються через різноманітні мережеві протоколи та канали зв'язку з різними характеристиками пропускної здатності та стабільності. Це створює додаткові виклики для забезпечення консистентної якості медіа-контенту, оскільки параметри передачі можуть динамічно змінюватися залежно від мережевих умов, завантаженості серверів та технічних характеристик кінцевих пристроїв користувачів.

Крім того, існуючі рішення для моніторингу якості відео зазвичай надають лише технічні метрики без контекстуального аналізу та практичних рекомендацій щодо усунення виявлених проблем. Це вимагає від операторів глибоких технічних знань для інтерпретації результатів та прийняття обґрунтованих рішень щодо оптимізації параметрів системи.

У зв'язку з цим актуальним завданням є розробка програмного модуля, який дозволяє здійснювати автоматизований аналіз OTT-потоків у реальному часі, виявляти проблеми якості медіа-контенту, використовувати можливості штучного інтелекту для контекстуального аналізу та формувати інтелектуальні рекомендації щодо їх усунення.

II. Мета роботи

Метою роботи є розробка програмного модуля аналізу відеопотоків з інтеграцією LLM технологій, який забезпечує збір, зберігання, обробку та інтелектуальний аналіз даних про якість медіа-потоків. Передбачається створення модульної програмної архітектури з графічним інтерфейсом користувача, який дозволить операторам моніторити якість відеопотоків, налаштовувати параметри аналізу та отримувати автоматичні звіти з рекомендаціями щодо усунення виявлених проблем.

III. Основна частина

У рамках розробки програмного модуля було здійснено детальний аналіз існуючих програмних рішень у сфері моніторингу медіа-потоків. Зокрема, були розглянуті такі технології та бібліотеки, як FFmpeg для обробки мультимедіа, OpenCV для аналізу відео та різні Python frameworks для створення GUI додатків. Також проаналізовано комерційні рішення, включаючи Video Quality Monitor від Elecard, PEVQ від Opticom та інші спеціалізовані інструменти для оцінки якості відео.

Аналіз показав, що існуючі рішення мають ряд суттєвих обмежень. По-перше, більшість систем зосереджена на вимірюванні окремих технічних параметрів (PSNR, SSIM, VMAF) без комплексного підходу до оцінки якості користувацького досвіду. По-друге, традиційні інструменти не забезпечують інтеграцію з сучасними технологіями штучного інтелекту для автоматичного генерування пояснень та рекомендацій. По-третє, існуючі рішення часто вимагають складного налаштування та значних обчислювальних ресурсів, що робить їх недоступними для малих та середніх організацій.

Втім, було виявлено відсутність готових рішень, що поєднували б функціональність аналізу відеопотоків з можливостями великих мовних моделей для генерування інтелектуальних звітів. З урахуванням зазначених проблем було розроблено власний програмний модуль, який забезпечує

комплексний аналіз якості відеопотоків з використанням сучасних технологій машинного навчання. Архітектура програмного модуля складається з кількох ключових компонентів, кожен з яких реалізує окрему функціональність.

На етапі проєктування програмної архітектури було створено діаграму компонентів модуля (див.рис.1), що демонструє логіку взаємодії між основними програмними компонентами: модулями аналізу потоків, системою управління конфігурацією, інтерфейсом користувача та інтеграцією з LLM.

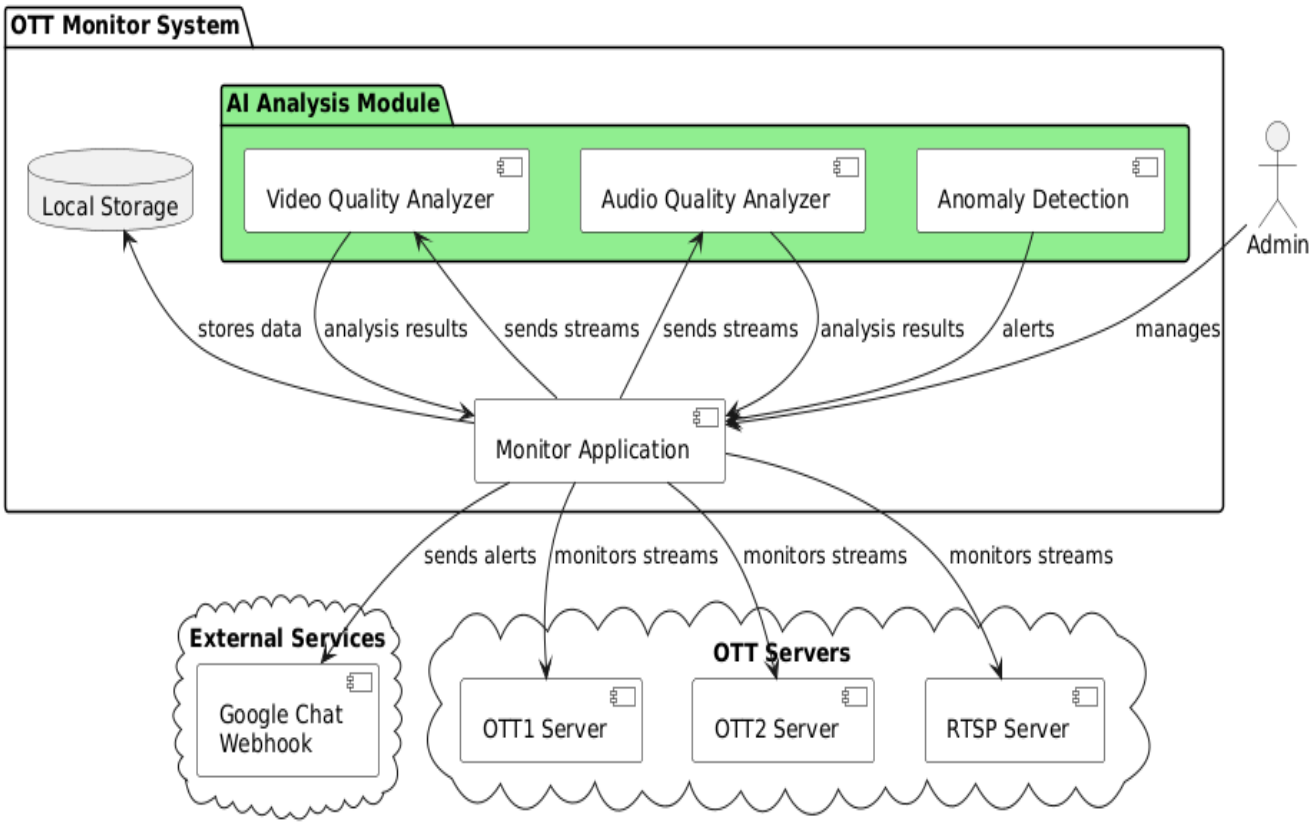


Рисунок 1 – Діаграма компонентів програмного модуля

Для реалізації програмного модуля обрано технологічний стек Python з використанням бібліотек asyncio для асинхронних операцій, tkinter для створення графічного інтерфейсу користувача, OpenCV для аналізу відеопотоків та інтеграція з API великих мовних моделей. Програмна архітектура спроектована з урахуванням принципів модульності та розширюваності для забезпечення легкої підтримки та розвитку системи.

Програмний модуль включає в себе декілька основних функціональних блоків. Перший блок відповідає за захоплення та попередню обробку відеопотоків з різних джерел, включаючи локальні файли, IP-камери та потокові сервіси. Для цього використовуються можливості OpenCV та FFmpeg, що дозволяє працювати з широким спектром відеоформатів та протоколів передачі даних. Другий блок реалізує алгоритми аналізу якості відео, включаючи виявлення артефактів стиснення, оцінку чіткості зображення, аналіз кольорової гами та детекцію втрачених кадрів.

Особливу увагу приділено інтеграції з великими мовними моделями через API OpenAI. Цей компонент дозволяє формувати детальні текстові звіти про стан відеопотоків, надавати рекомендації щодо покращення якості та пояснювати виявлені проблеми зрозумілою для користувача мовою. LLM аналізує численні метрики якості відео та генерує структуровані висновки з практичними порадами щодо оптимізації параметрів кодування та передачі.

Графічний інтерфейс користувача, реалізований за допомогою tkinter, забезпечує інтуїтивне управління всіма функціями програмного модуля. Інтерфейс включає панель налаштувань для конфігурації параметрів аналізу, область візуалізації відеопотоків з накладеними метриками якості, секцію відображення результатів аналізу та панель генерованих LLM звітів. Користувач може в реальному часі спостерігати за процесом аналізу, налаштовувати пороги якості та експортувати результати в різні формати. Система логування та збереження результатів забезпечує можливість

ретроспективного аналізу даних про якість відеопотоків. Всі метрики зберігаються в структурованому форматі з часовими мітками, що дозволяє відстежувати тенденції зміни якості протягом тривалих періодів. Програмний модуль також підтримує налаштування автоматичних сповіщень про критичні зниження якості відеопотоків.

Важливою особливістю розробленого програмного модуля є його здатність адаптуватися до різних типів відеоконтенту. Алгоритми аналізу враховують специфіку різних жанрів медіа-контенту, включаючи спортивні трансляції з високою динамікою зображення, кінофільми з різноманітними сценами освітлення та корпоративні відеоконференції з статичним фоном. Для кожного типу контенту система автоматично налаштовує параметри аналізу та пороги виявлення проблем якості.

Модуль інтеграції з LLM реалізований з використанням REST API та підтримує роботу з різними моделями штучного інтелекту, включаючи GPT-4, Claude та локально розгорнуті моделі. Система формує структуровані запити до LLM, що містять технічні метрики якості відео, контекстуальну інформацію про тип контенту та поточні параметри кодування. У відповідь LLM генерує детальні пояснення виявлених проблем, рекомендації щодо коригування параметрів кодування та прогнози щодо подальших тенденцій зміни якості.

Архітектура програмного модуля побудована на принципах мікросервісів, що забезпечує високу масштабованість та можливість горизонтального розширення системи. Кожен функціональний компонент може працювати незалежно, що дозволяє розподіляти обчислювальне навантаження між різними серверами та забезпечувати відмовостійкість системи. Комунікація між компонентами здійснюється через асинхронні повідомлення з використанням черг повідомлень.

Для забезпечення високої продуктивності аналізу відеопотоків використовуються сучасні алгоритми паралельної обробки даних. Програмний модуль підтримує багатопотокову обробку кадрів відео, що дозволяє одночасно аналізувати декілька потоків або різні сегменти одного потоку. Це особливо важливо при роботі з відеопотоками високої роздільної здатності (4K, 8K), де обсяг даних для обробки значно перевищує можливості однопотокових алгоритмів.

Система моніторингу якості включає в себе алгоритми виявлення різних типів артефактів, включаючи блокування (blockiness), розмивання (blurriness), шум (noise), артефакти стиснення та втрату кольорової інформації. Для кожного типу артефактів розроблено спеціалізовані метрики, що враховують особливості людського сприйняття відеозображення та корелюють з суб'єктивними оцінками якості.

Додатково програмний модуль забезпечує можливість інтеграції з зовнішніми системами моніторингу та управління через стандартизовані API. Це дозволяє включати модуль аналізу відеопотоків до існуючих екосистем управління медіа-контентом та автоматизувати процеси прийняття рішень щодо оптимізації параметрів передачі відео.

Результати тестування програмного модуля показали його ефективність у виявленні проблем якості відео в режимі реального часу. Система здатна обробляти до 50 одночасних відеопотоків з роздільною здатністю Full HD при використанні стандартного серверного обладнання. Точність виявлення критичних проблем якості складає 94%, що перевищує показники більшості існуючих комерційних рішень.

Висновок

Розроблений програмний модуль аналізу відеопотоків з використанням LLM технологій дозволяє ефективно контролювати якість медіа-контенту в реальному часі. Програмне рішення забезпечує модульну архітектуру, зручний графічний інтерфейс користувача, інтеграцію з сучасними технологіями машинного навчання та автоматизовані звіти з рекомендаціями. Використання такого програмного модуля сприятиме підвищенню ефективності моніторингу медіа-потоків та автоматизації процесів контролю якості в OTT індустрії.

Список використаних джерел

1. Van Rossum, G. Python Programming Language [Електронний ресурс]. – Режим доступу: https://www.python.org/
2. Bradski, G. OpenCV Library Documentation [Електронний ресурс]. – Режим доступу: https://opencv.org/
3. FFmpeg Development Team. FFmpeg Documentation [Електронний ресурс]. – Режим доступу: https://ffmpeg.org/documentation.html
4. Python Software Foundation. Tkinter GUI Programming [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/tkinter.html
5. AsyncIO Documentation for Asynchronous Programming [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/library/asyncio.html
6. OpenAI API Documentation [Електронний ресурс]. – Режим доступу: https://platform.openai.com/docs

УДК 004.4

ПЛАТФОРМА ДЛЯ СПІЛЬНОТИ КНИГОЛЮБІВ

Папа О.А.[1)], **Птиць Д.Р.**[2)]
Західноукраїнський національний університет
[1)] *д. філ.., доцент;* [2)] *бакалавр*

I. Постановка задачі

Багато користувачів шукають прості платформи для ведення списку прочитаних книг, збереження улюблених видань або формування списку бажаного для читання. У більшості випадків реалізація таких функцій пов'язана зі складними системами з базами даних або реєстрацією.

Метою розробки є створення веб-застосунку, орієнтованого на користувачів, які цікавляться книгами. Основне завдання полягає у розробці інтуїтивно зрозумілого та функціонального інтерфейсу, що дозволяє користувачам реєструватися, переглядати, додавати, зберігати книги, а також формувати власну віртуальну бібліотеку.

II. Мета роботи

Метою роботи є розробка функціонального, безпечного та інтуїтивно зрозумілого веб-застосунку для користувачів, які цікавляться літературою. Застосунок повинен забезпечити користувачам можливість входу до системи, управління власним обліковим записом, формування віртуальної бібліотеки, взаємодії з книжковим контентом та налаштування особистих даних у захищеному середовищі.

Особливу увагу в розробці застосунку було приділено зручності користування, швидкодії інтерфейсу та можливості роботи без потреби у складному серверному середовищі. Вебзастосунок орієнтований на широку аудиторію користувачів — від активних читачів до невеликих книжкових спільнот, які потребують простого й доступного інструменту для ведення особистої бібліотеки, формування добірок книг та взаємодії у тематичних групах.

III. Проектування системи

На основі аналізу предметної області — онлайн-платформ для книголюбів — сформовано концепцію сервісу, що поєднує каталог особистих бібліотек, соціальну взаємодію та біржу обміну / продажу книг.

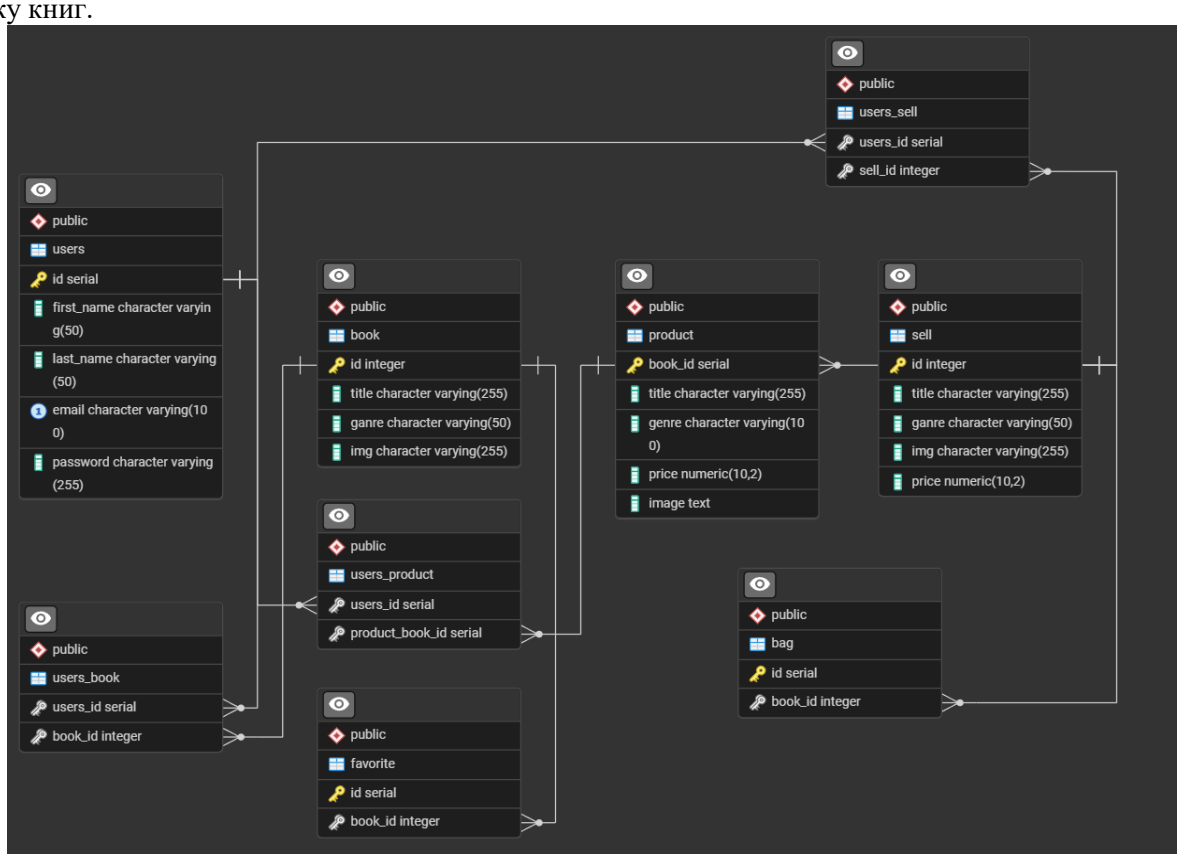


Рисунок 1 – ER Діаграма

Ключовий функціонал охоплює: книги (Books) – додавання, редагування, класифікація за жанрами, станом і мовою; завантаження обкладинки, бібліотека користувача (UserLibrary) – приватний перелік прочитаних / бажаних книг, фільтри та статистика, уподобання (Favorites) – швидке додавання книжок у «обране», кошик / Замовлення (Cart → Order) – формування заявки на купівлю або обмін, розрахунок суми, вибір способу отримання, маркетплейс (Sell / Trade) – розміщення оголошень про продаж чи обмін, керування активними лотами, користувачі (Users) – профіль, рейтинг, історія угод, налаштування сповіщень, аналітика – топ-книги, популярні жанри, активність спільноти.

IV. Реалізація системи

Результатом став повнофункціональний вебзастосунок, побудований на основі сучасних вебтехнологій. Клієнтська частина реалізована з використанням React.js та Tailwind CSS, що забезпечує адаптивний та зручний інтерфейс для користувача. Серверна логіка розроблена за допомогою Node.js з фреймворком Express, що відповідає за обробку запитів, а також забезпечує роботу з JWT-автентифікацією та обробку файлів.

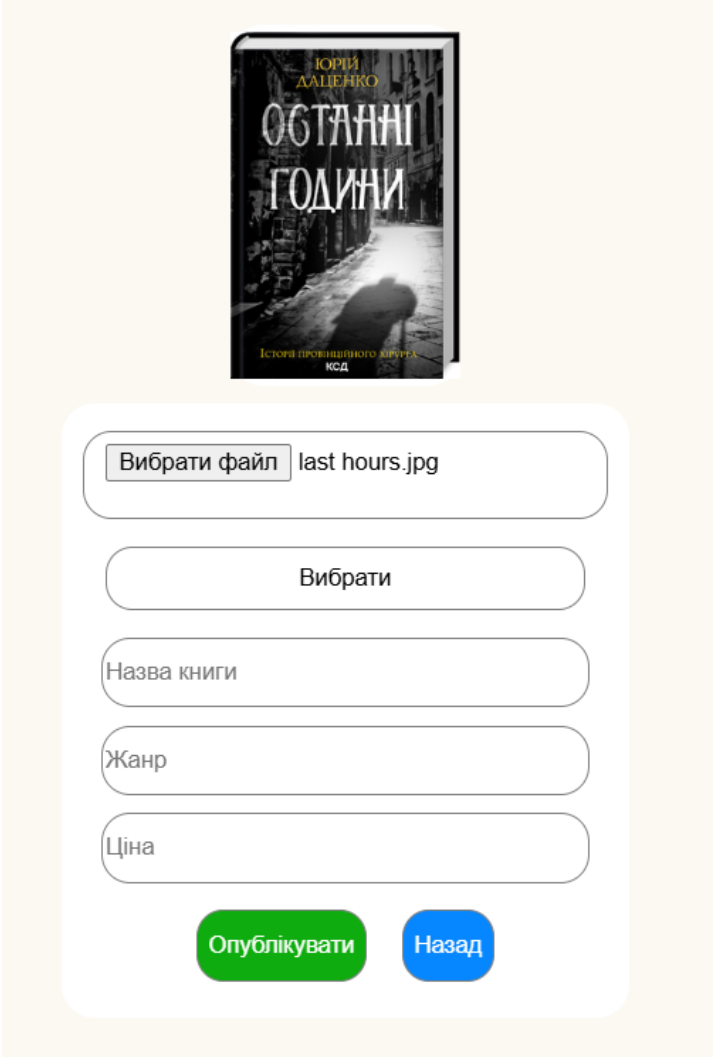


Рисунок 2 – Вікно виставлення книги в продаж

На рисунку зображено вікно для виставлення книги на продаж, яке є одним з ключових елементів платформи. Дане вікно дозволяє користувачеві додати нову книгу до розділу «Продаж», заповнивши основні параметри: зображення книги, назву, жанр та ціну.

Інтерфейс реалізовано з урахуванням зручності користувача — всі поля чітко структуровані, інтуїтивно зрозумілі, а завантажене зображення книги одразу візуалізується на сторінці. Окремими кнопками передбачено дії "Опублікувати" та "Назад".

Даний функціонал відповідає сутності SELL у базі даних, де зберігаються всі оголошення, розміщені користувачами. Після підтвердження система обробляє інформацію та додає її до відповідної таблиці в базі даних, після чого книга стає доступною іншим користувачам для перегляду, обміну або придбання.

Висновок

У результаті реалізовано фронтенд-застосунок на основі React, що дозволяє користувачу зручно керувати своєю віртуальною книжковою бібліотекою та профілем без необхідності у серверній частині або базі даних. Структура програми побудована за принципами компонентного підходу, що забезпечує гнучкість, розширюваність і легку підтримку. Архітектура рішення дозволяє у майбутньому підключити серверну частину або базу даних для зберігання інформації в реальному часі, не змінюючи основної логіки.

Список використаних джерел

1. React documentation [Електронний ресурс]. – Режим доступу: https://react.dev/
2. React Router [Електронний ресурс]. – Режим доступу: https://reactrouter.com
3. JavaScript Guide [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
4. Meta. React – A Java Script library for building user interfaces [Електронний ресурс]. – Режим доступу: https://react.dev/docs/getting-started

УДК 004.4

АВТОМАТИЗОВАНА СИСТЕМА ОБЛІКУ ТА ФОРМУВАННЯ ДОРОЖНІХ ЛИСТІВ «CALCULATIONS»

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

I. Постановка задачі

Транспортно-логістична галузь є однією з ключових артерій економіки України. Значну її частку складають малі підприємства та приватні перевізники (ФОП), які забезпечують гнучкість і доступність послуг на регіональному та національному рівнях. Однак саме цей сегмент бізнесу стикається з серйозними операційними викликами. Через обмежені фінансові та кадрові ресурси більшість перевізників змушена покладатися на застарілі методи управління: паперові журнали, таблиці в Microsoft Excel або ручні розрахунки.

Такий підхід неминуче призводить до низки проблем:

- **Низька точність розрахунків:** Помилки при визначенні кілометражу, розрахунку витрат пального та вартості перевезення ведуть до прямих фінансових збитків.
- **Високі часові витрати:** Ручне заповнення дорожніх листів, звітів та іншої документації є монотонним процесом, що забирає час, який можна було б витратити на пошук нових замовлень чи оптимізацію діяльності.
- **Ризик людського фактора:** Неуважність, втома або недостатня кваліфікація персоналу підвищують імовірність помилок, що негативно впливає на фінансову стійкість і репутацію перевізника.
- **Відсутність аналітики:** Паперові чи табличні дані складно аналізувати для прийняття стратегічних рішень, наприклад, для оцінки рентабельності окремих маршрутів чи транспортних засобів.

Існуючі на ринку програмні продукти, такі як **Route4Me** чи **OptimoRoute**, орієнтовані переважно на великі логістичні компанії. Вони пропонують потужний, але надлишковий функціонал, є дорогими у впровадженні та підтримці, і не враховують специфічні потреби малого бізнесу, як-от динамічне завантаження транспорту чи необхідність швидкої персоналізації звітної документації. Це створює гостру потребу в розробці доступного, гнучкого та інтуїтивно зрозумілого інструменту, який би став надійним помічником для малих перевізників.

II. Мета роботи

Метою є розробка сучасного веб-додатку «Calculations» для автоматизації ключових транспортно-логістичних розрахунків, що враховує потреби малих логістичних компаній і приватних перевізників. Запропонована система повинна забезпечувати інтеграцію з картографічними сервісами для точного визначення відстаней, реалізовувати декілька методів обліку витрат пального з урахуванням завантаження транспортного засобу, а також дозволяти персоналізувати маршрути й структуру звітної документації.

III. Проєктування системи

Аналіз сучасних програмних рішень для транспортної логістики показав, що більшість із них – наприклад, Route4Me чи OptimoRoute — орієнтовані переважно на середній та великий бізнес. Такі системи мають високий поріг входу, складну структуру налаштувань і інтеграцій, а також не дозволяють гнучко адаптувати логіку розрахунків і звітності під потреби малого підприємства чи приватного перевізника. Водночас простіші онлайн-сервіси часто обмежені функціональністю, не враховують динамічне завантаження транспортних засобів або не дозволяють персоналізувати документацію, що знижує їхню практичну цінність для цільової аудиторії.

З урахуванням цих викликів для реалізації системи було обрано клієнт-серверну архітектуру, яка дозволяє чітко розділити інтерфейс, бізнес-логіку та роботу з даними на окремі рівні. Користувацький інтерфейс створено з використанням React, Radix UI та Tailwind CSS, що гарантує інтуїтивність, швидкість і адаптивність роботи на різних пристроях. Серверна частина побудована на базі Next.js, а бізнес-логіка організована через окремий сервісний шар, що координує всі ключові процеси між клієнтом, базою даних і зовнішніми API-сервісами. На схемі архітектури системи (рис.

1) відображено послідовність обробки запитів – від дій користувача у браузері до збереження та обробки інформації у базі даних, а також взаємодію з сервісами автентифікації та геолокації.

Розробка системи «Calculations» була спрямована на створення інтуїтивно зрозумілого, зручного та адаптивного інтерфейсу для кінцевого користувача. Усі основні операції – вибір транспортного засобу, планування маршруту, облік витрат пального та генерація звітів – виконуються через сучасну веб-форму, доступну з будь-якого пристрою.

Користувач проходить автентифікацію за допомогою сервісу Clerk, що забезпечує безпечний доступ до персональних даних і підтримує багатофакторний захист. Система інтегрована з TomTom API для точного розрахунку відстаней між пунктами маршруту та використовує кешування для прискорення обробки даних. Кожен маршрут та всі розрахунки зберігаються в історії, з можливістю фільтрації та перегляду попередніх операцій. Особливу увагу приділено автоматизації документообігу — користувач може швидко згенерувати PDF-звіт за власними налаштуваннями.

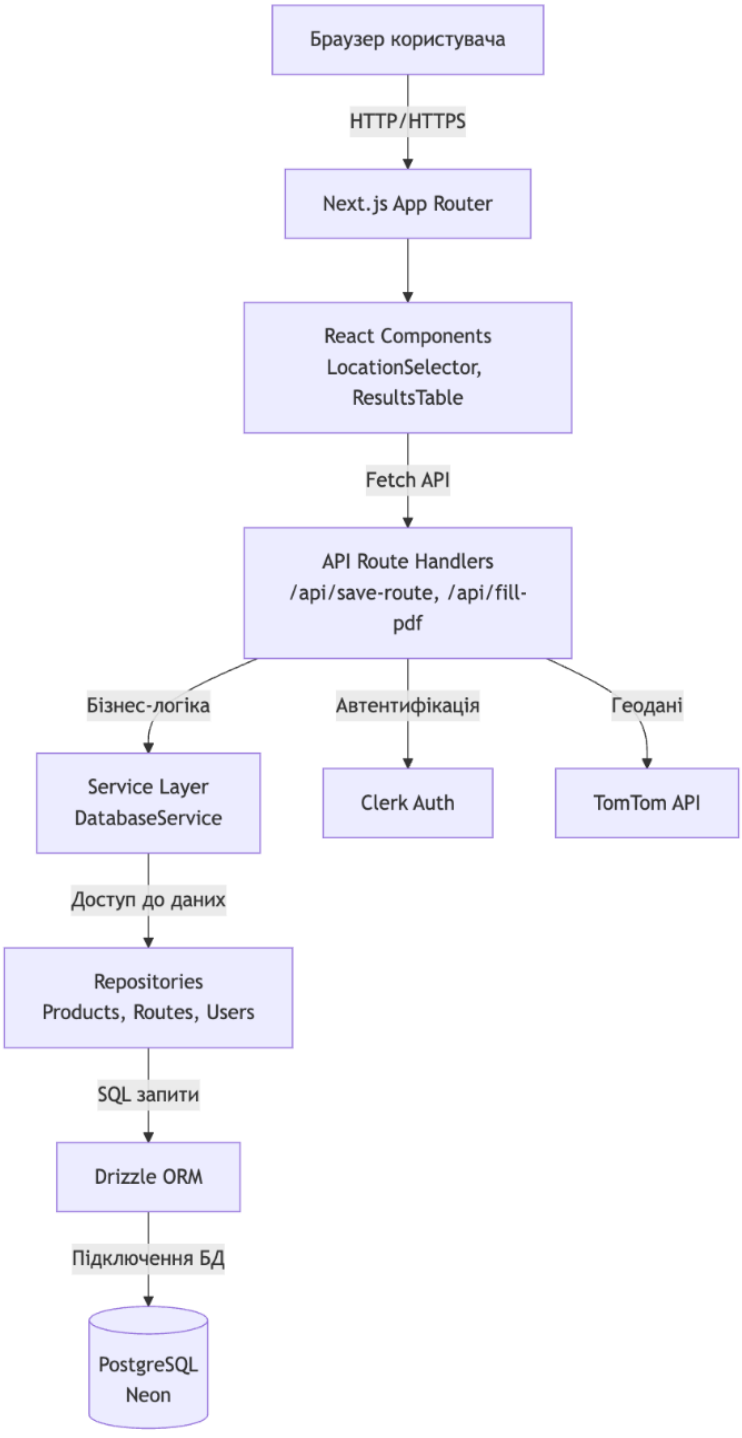


Рисунок 1 – Клієнт-серверна архітектура системи 'Calculations'

На рисунку 2 представлено головний інтерфейс системи, де показано процес вибору транспортного засобу та введення пунктів маршруту для автоматичного розрахунку відстані. Компоненти інтерфейсу, реалізовані на React, забезпечують динамічну взаємодію без перезавантаження сторінки. Форма складається з селектора для вибору транспортного засобу та полів для маршруту з функцією автозаповнення. Дії користувача ініціюють асинхронні запити до серверної частини, яка обробляє дані та звертається до TomTom API. Отримана відстань та візуалізація маршруту на інтегрованій карті негайно відображаються на клієнті, що демонструє ефективність обраної архітектури та забезпечує комфортну роботу.

Окрім кілометражу, система автоматично розраховує орієнтовні витрати пального, враховуючи параметри автомобіля та вантажу. Ці дані стають основою для миттєвої генерації персоналізованого дорожнього листа у форматі PDF. Усі розрахунки зберігаються в історії користувача для аналізу, швидкого відтворення маршрутів та спрощення майбутньої звітності.

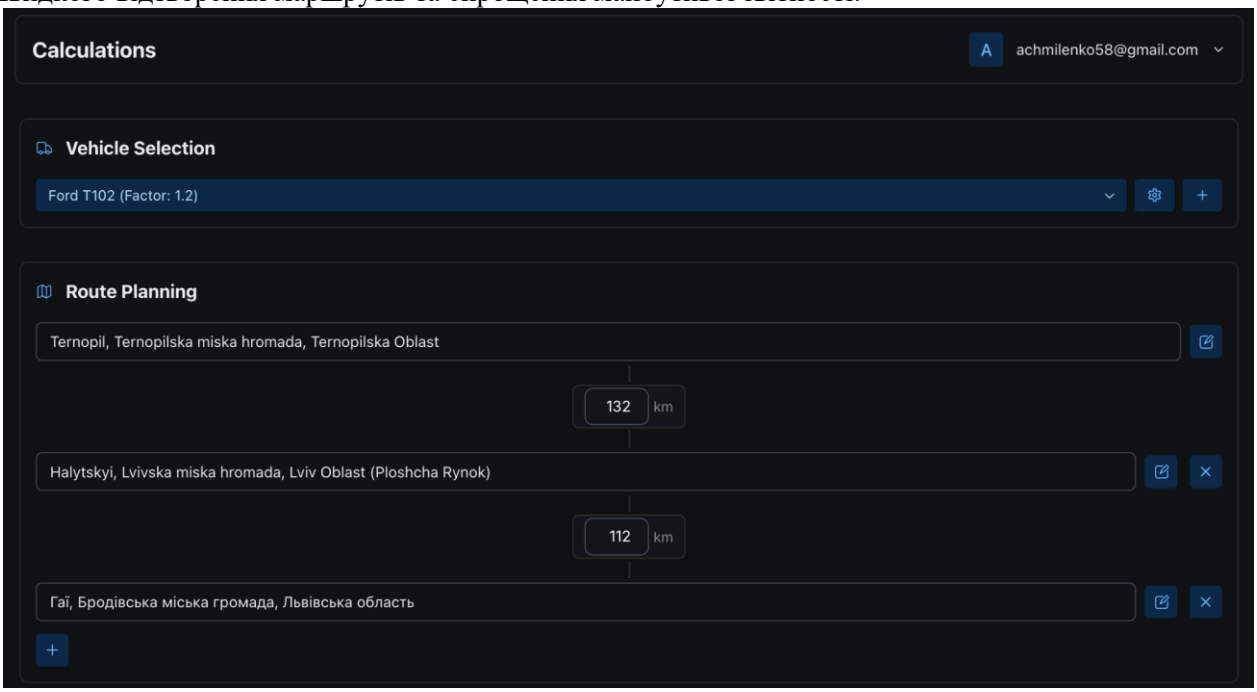


Рисунок 2 – Головний інтерфейс планування маршруту в системі “Calculations”

Висновок

Розроблена система «Calculations» повністю враховує потреби малого бізнесу та приватних перевізників у транспортній логістиці. Запропоноване рішення забезпечує автоматизацію планування маршрутів, точний облік витрат пального й персоналізовану генерацію звітів, що дозволяє суттєво підвищити ефективність та прозорість роботи. Завдяки використанню сучасних технологій і клієнт-серверної архітектури система легко масштабується й підтримує подальший розвиток.

Перспективи вдосконалення проєкту полягають у розширенні функціоналу — впровадженні мобільної версії, поглибленій аналітиці, а також інтеграції з іншими бізнес-системами, такими як ERP чи CRM, для ще більш комплексного покриття бізнес-процесів користувачів.

Список використаних джерел

1. Про затвердження Правил перевезень вантажів автомобільним транспортом в Україні : Наказ Міністерства транспорту України від 14.10.1997 № 363. URL: https://zakon.rada.gov.ua/laws/show/z0128-98#Text .
2. Левченко О. П., Ковальська Л. А. Цифровізація логістичних процесів як фактор підвищення конкурентоспроможності підприємств. *Економіка та суспільство*. 2023. № 54. URL: https://economyandsociety.in.ua/index.php/journal/article/view/3045.
3. Next.js by Vercel - The React Framework for the Web : official documentation [Електронний ресурс] – Режим доступу https://nextjs.org/docs.
4. Supabase. Use Supabase with Next.js [Електронний ресурс] – Режим доступу: https://supabase.com/docs/guides/getting-started/quickstarts/nextjs
5. S.Krepych, I.Spivak. “Algorithm of automatic generation of hotel descriptions using templates based on Markov chains”, International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T). 2018. pp.257-260
6. TomTom Maps APIs : official documentation [Електронний ресурс] – Режим доступу: https://developer.tomtom.com/products/maps-api
7. .Державна служба статистики України. Статистика транспорту [Електронний ресурс]. – Режим доступу: https://ukrstat.gov.ua/operativ/operativ2023/tr/tr_u/tr_u_2023.htm

УДК 004.4

ВЕБ-ОРІЄНТОВАНА СИСТЕМА ДЛЯ ПРОДАЖУ БУДІВЕЛЬНИХ МАТЕРІАЛІВ

Марчак Н.О.
Західноукраїнський національний університет
бакалавр

І. Постановка задачі

У сучасних умовах будівельний ринок характеризується зростанням попиту на онлайн-продаж будівельних матеріалів, зокрема газоблоків, піноблоків, цементу та супутніх товарів. Кінцеві споживачі стикаються з труднощами при пошуку потрібних матеріалів через велику кількість розрізнених сайтів, обмежену фільтрацію за параметрами, відсутність актуальної інформації про наявність, а також неінтуїтивний інтерфейс багатьох онлайн-магазинів. Це ускладнює прийняття зваженого рішення щодо купівлі, особливо у випадках, коли важливими є технічні характеристики, ціна, умови доставки та оперативність оформлення замовлення.

Виникає потреба у створенні сучасної веб-системи, яка б дозволяла користувачам швидко і зручно обирати будівельні матеріали, переглядати їх детальні характеристики, фільтрувати за основними параметрами, перевіряти залишки на складі та оформлювати замовлення без зайвих кроків і складної реєстрації.

ІІ. Мета роботи

Метою роботи є розробка веб-орієнтованої системи для продажу будівельних матеріалів, яка покликана вирішити проблему ускладненого вибору для кінцевого споживача. Головне завдання — створити єдиний, інтуїтивно зрозумілий простір, де користувач може зручно знаходити, фільтрувати та порівнювати товари за ключовими параметрами.

Система має забезпечувати не лише швидкий доступ до детальної інформації про продукцію, а й максимально спрощувати процес оформлення замовлення. Для цього проєкт передбачає створення адаптивного веб-інтерфейсу, що коректно працює на різних пристроях.

Важливим аспектом є закладення надійної архітектури, яка б гарантувала захист персональних даних, була готова до інтеграції з інструментами доставки та легко масштабувалася для майбутнього розширення асортименту й функціоналу, створюючи таким чином ефективний та конкурентоспроможний онлайн-інструмент.

ІІІ. Проєктування системи

Під час проєктування веб-орієнтованої системи для продажу будівельних матеріалів основна увага приділялася зручності користувача, масштабованості, швидкодії та надійності зберігання даних. Враховуючи вимоги сучасного ринку, архітектуру було розроблено на основі стеку MERN (MongoDB, Express, React, Node.js), що забезпечує як гнучкість розробки, так і ефективну взаємодію між клієнтською та серверною частинами.

Логічна модель даних системи побудована на виділенні ключових сутностей: категорії товарів, самі товари, замовлення, позиції замовлень та відгуки користувачів. Кожна сутність відображає окрему колекцію в базі даних MongoDB, а зв’язки між ними реалізовані через унікальні ідентифікатори (category_id, product_id, order_id). Це дозволяє ефективно організувати зберігання великої кількості даних, забезпечити швидкий пошук і фільтрацію, а також спростити масштабування системи при розширенні асортименту чи впровадженні нових функцій.

На рисунку 1 наведено концептуальну модель даних, яка ілюструє взаємозв’язки між основними колекціями: кожен товар належить певній категорії, замовлення складається з декількох позицій, а користувачі можуть залишати відгуки про товари. Такий підхід забезпечує гнучкість у роботі з інформацією та дає змогу швидко адаптувати систему до змін ринку чи запитів користувачів.

Веб-система для продажу будівельних матеріалів реалізована на основі MERN-стеку, що забезпечує продуктивність, адаптивність та простоту подальшого розширення функціоналу. Серверна частина на Node.js та Express ефективно обробляє запити, а гнучка структура MongoDB дозволяє надійно зберігати складні дані про товари та замовлення.

Основну увагу приділено розробці сучасного каталогу товарів з гнучкою фільтрацією, який дає змогу користувачам швидко знаходити потрібну продукцію серед широкого асортименту.

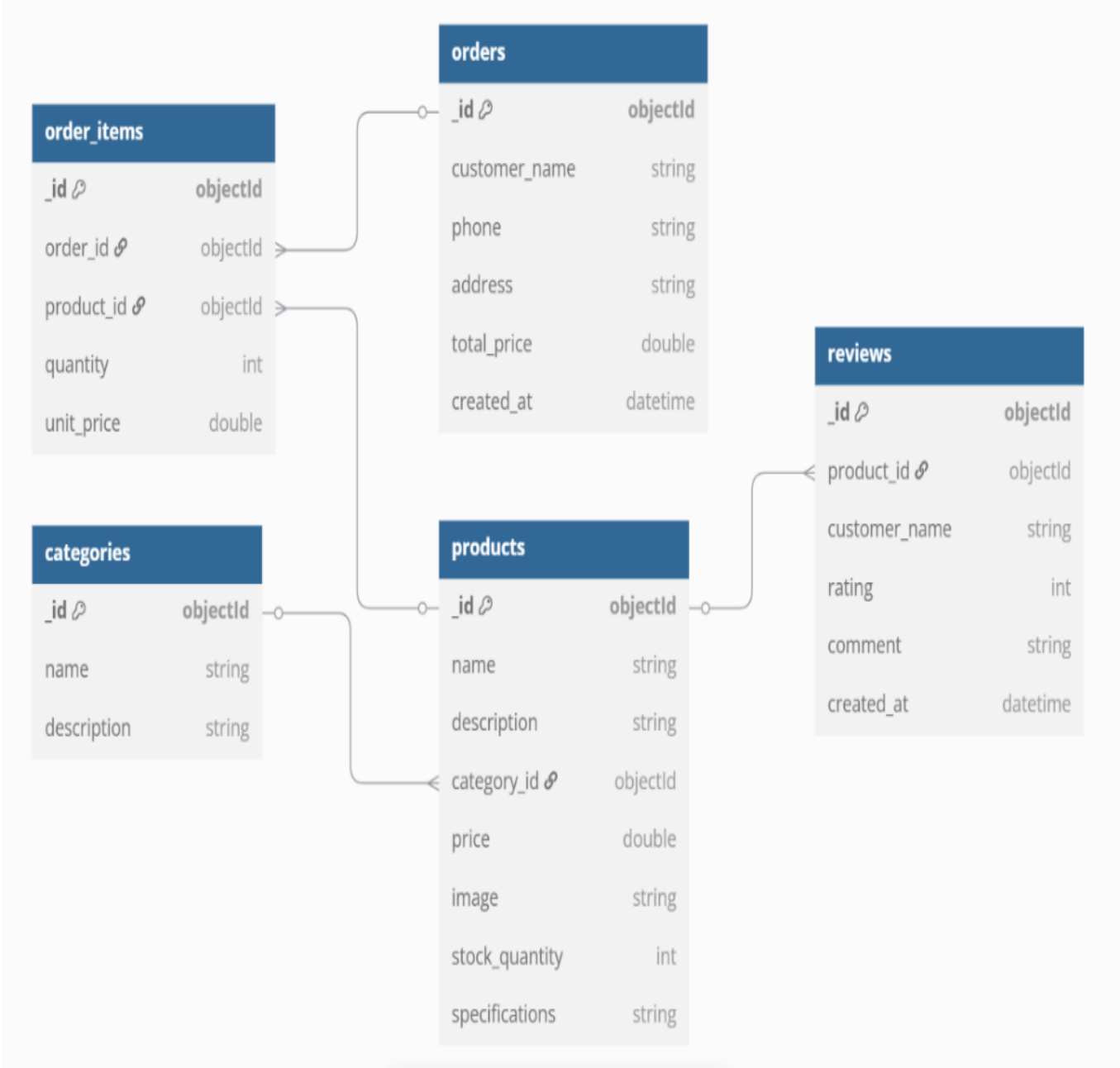


Рисунок 1 – Концептуальна модель даних

Інтерфейс каталогу організовано за категоріями та підкатегоріями (див.рис.2), що спрощує навігацію й дозволяє ефективно працювати з різними групами товарів — газоблоками, піноблоками, супутніми матеріалами тощо. Кожна товарна позиція містить фотографію, ключові характеристики, ціну, а також можливість додавання до обраного чи в кошик.

Для зручності користувача реалізовано сортування, вибір кількості товарів, перегляд різних форматів відображення.

Оформлення замовлень здійснюється через інтерактивний кошик з можливістю редагування списку товарів та введення контактних даних. Такий підхід мінімізує кількість кроків для користувача, що знижує ймовірність незавершеної покупки.

Адаптивний дизайн гарантує коректну роботу системи як на десктопах, так і на мобільних пристроях. Для адміністраторів реалізовано окремий функціонал управління асортиментом і обробки замовлень.

Таким чином, адміністративна частина замикає повний цикл взаємодії, забезпечуючи ефективне виконання замовлень, розміщених клієнтами.

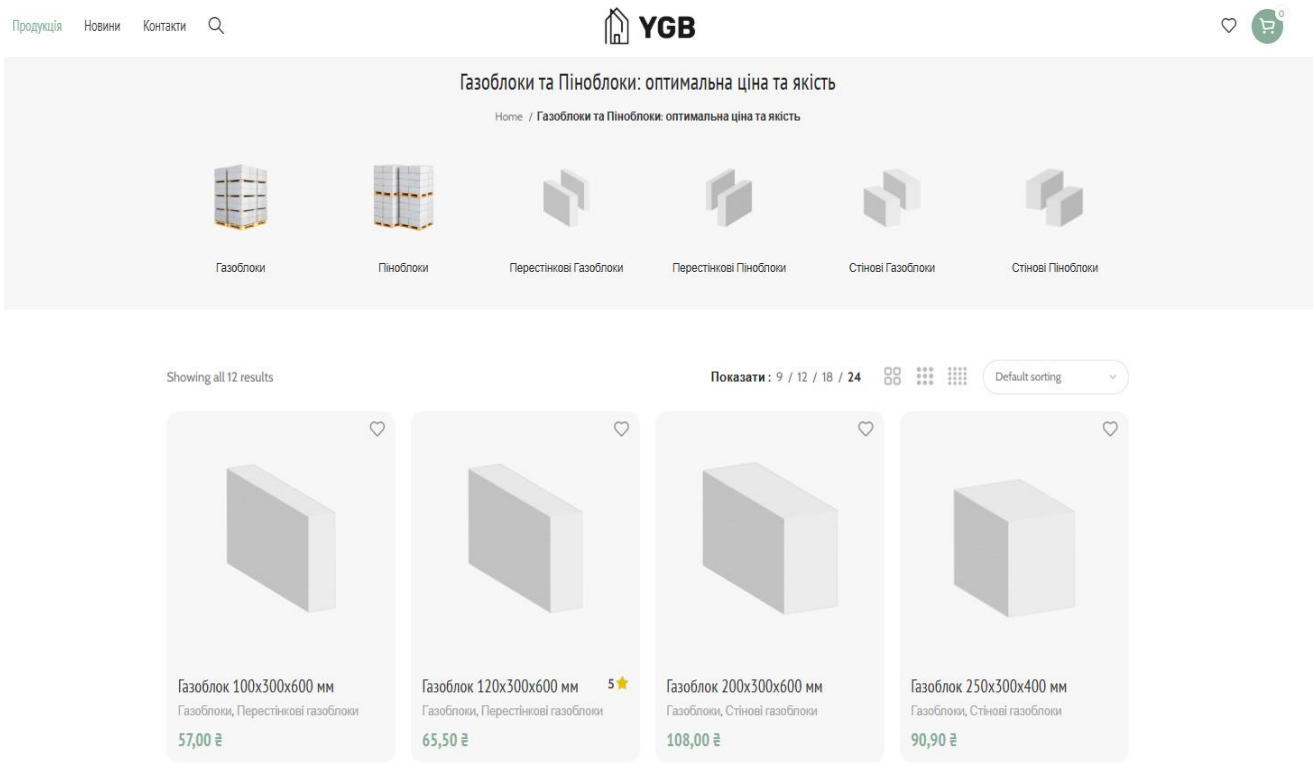


Рисунок 2 – Інтерфейс каталогу товарів у веб-системі для продажу будівельних матеріалів

Висновок

Розроблена веб-орієнтована система для продажу будівельних матеріалів забезпечує зручний пошук, детальний перегляд характеристик і швидке оформлення замовлень для кінцевих користувачів. Завдяки впровадженню сучасної архітектури, функціонального каталогу й адаптивного інтерфейсу вдалося підвищити ефективність роботи онлайн-магазину та задовольнити актуальні потреби ринку.

Система є масштабованою, підтримує гнучке розширення функціоналу та може інтегруватися з додатковими сервісами для автоматизації бізнес-процесів у сфері торгівлі будматеріалами.

Окремо варто відзначити, що при розробці було враховано питання інформаційної безпеки, захисту персональних даних і надійності обробки замовлень.

Запропонована структура бази даних дає змогу ефективно працювати з великими обсягами товарних позицій та історією замовлень. У перспективі передбачено впровадження нових функцій — онлайн-оплати, розширеної аналітики для адміністраторів, автоматизованої логістики та інтеграції з зовнішніми платформами. Це дозволить ще більше підвищити якість сервісу та оптимізувати бізнес-процеси для власників і користувачів веб-системи.

Список використаних джерел

1. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd ed. O'Reilly Media, 2019. 444 p.
2. Articles on E-commerce UX [Електронний ресурс] – Режим доступу:/ Nielsen Norman Group. URL: https://www.nngroup.com/articles/tags/e-commerce/.
3. Data Modeling Introduction [Електронний ресурс] – Режим доступу:// MongoDB Manual. MongoDB, 2024. URL: https://www.mongodb.com/docs/manual/core/data-modeling-introduction/.
4. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. O'Reilly Media, 2017. 616 p.
5. OWASP Top 10 [Електронний ресурс] – Режим доступу:/ OWASP Foundation. URL: https://owasp.org/www-project-top-ten/
6. S. Krepych, I.Spivak, Improvement of SVD algorithm to increase the efficiency of recommendation systems. Advanced Information Systems, 5(4), 2021, pp.55-59
7. MongoDB. The developer data platform [Електронний ресурс]. – Режим доступу: https://www.mongodb.com.
8. Крепи С.Я., Співак І.Я. Якість програмного забезпечення та тестування: базовий курс. Тернопіль: ФОП Паляниця В.А., 2020. – 478с
9. "Посібник для початківців по запитах до MongoDB за допомогою Compass: оволодіння запитами MongoDB та розуміння відмінностей із SQL" [Електронний ресурс] – Режим доступу: https://javascript.org.ua/posibnik-dlya-pochatkivcziv-po-zapitah-do-mongodb-za-dopomogoyu-compass-ovolodinnya-zapitami-mongodb-ta-rozuminnya-vidminnostej-iz-sql/
10. Node.js. JavaScript runtime built on Chrome's V8 JavaScript engine [Електронний ресурс] – Режим доступу: https://nodejs.org

УДК 004.8

МЕТОДИ ТА ПРОГРАМНІ ЗАСОБИ ДЛЯ ПІДВИЩЕННЯ ЯКОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІЗ ЗАСТОСУВАННЯМ МАШИННОГО НАВЧАННЯ

Літинський О.Л.
Західноукраїнський національний університет
бакалавр

I. Постановка задачі

З розвитком інформаційних технологій постала проблема ефективного управління якістю програмного забезпечення, зокрема в умовах зростання масштабів і складності проєктів. Ручний аналіз коду є трудомістким, схильним до помилок і потребує значного досвіду від розробників.

З цієї причини виникає нагальна потреба в автоматизованих інструментах, здатних виконувати якісний аналіз коду та надавати обґрунтовані рекомендації щодо його вдосконалення. Застосування машинного навчання в цьому напрямі відкриває нові можливості для адаптивного аналізу, що враховує контекст і стиль написання коду.

II. Мета роботи

Метою дослідження є розробка програмної системи, яка за допомогою методів машинного навчання дозволить автоматизувати процес аналізу програмного коду та підвищити його якість. Основні завдання системи включають виявлення поширених помилок, дублювання, складних ділянок, неефективних шаблонів і загальне покращення структури коду відповідно до сучасних стандартів розробки.

III. Структура системи

Проєктована система для автоматизованого аналізу програмного коду має модульну багаторівневу архітектуру, орієнтовану на масштабованість, продуктивність та адаптивність до різних стилів програмування. Система поділена на п’ять ключових компонентів[3]: клієнтський інтерфейс, серверну частину (Backend), підсистему машинного навчання, систему управління даними та інфраструктуру розгортання.

Клієнтська частина (Frontend) - забезпечує інтерактивну взаємодію користувача з системою. Реалізована з використанням ReactJS, що дає змогу динамічно оновлювати контент без перезавантаження сторінки. Користувачі можуть подавати код на аналіз, переглядати результати, отримувати рекомендації та залишати зворотний зв’язок.

Серверна частина (Backend API) - реалізована на Node.js і виконує роль центрального логічного вузла. Вона забезпечує маршрутизацію запитів, валідацію вхідних даних, а також взаємодіє з підсистемою машинного навчання та базою даних.

Підсистема машинного навчання - серцевина інтелектуальної логіки системи, яка виконує обробку та аналіз коду. Вона включає кілька спеціалізованих модулів, реалізованих на базі TensorFlow[2] або PyTorch:

- Модуль класифікації - ідентифікує типові помилки та антипатерни в коді.
- Модуль кластеризації - виявляє повторювані структури та дублікати коду.
- Модуль адаптації до стилів програмування - аналізує особливості стилю для формування контекстно-орієнтованих рекомендацій.
- Модуль навчання на прикладах - постійно оновлює моделі на основі нових даних з відкритих репозиторіїв.
- Модуль генерації рекомендацій - формує підказки щодо оптимізації коду (спрощення, рефакторинг, покращення читабельності).

Система управління даними - відповідає за зберігання, організацію та доступ до даних, що використовуються і генеруються всіма компонентами системи. Для роботи з різними типами даних застосовуються спеціалізовані сховища. PostgreSQL[4] використовується для зберігання структурованої інформації - таких як облікові записи користувачів, сесії, журнали запитів, метадані про результати аналізу та версії моделей. MongoDB застосовується для зберігання напівструктурованих та неструктурованих даних, зокрема:

- ✓ вихідного коду, поданого на аналіз у форматі JSON;
- ✓ проміжних представлень коду, сформованих у процесі обробки.

Інфраструктура розгортання - система контейнеризована за допомогою Docker, що спрощує процес масштабування, тестування та оновлення. Для хостингу та автоматичного масштабування використовується AWS. Таке рішення забезпечує високу доступність, безперервність обслуговування та ефективне управління ресурсами.

З метою спрощення аналізу компонент системи, на рисунку 1 представлено діаграму компонентів (UML Component Diagram), що демонструє ключові компоненти системи та взаємозв'язки між ними.

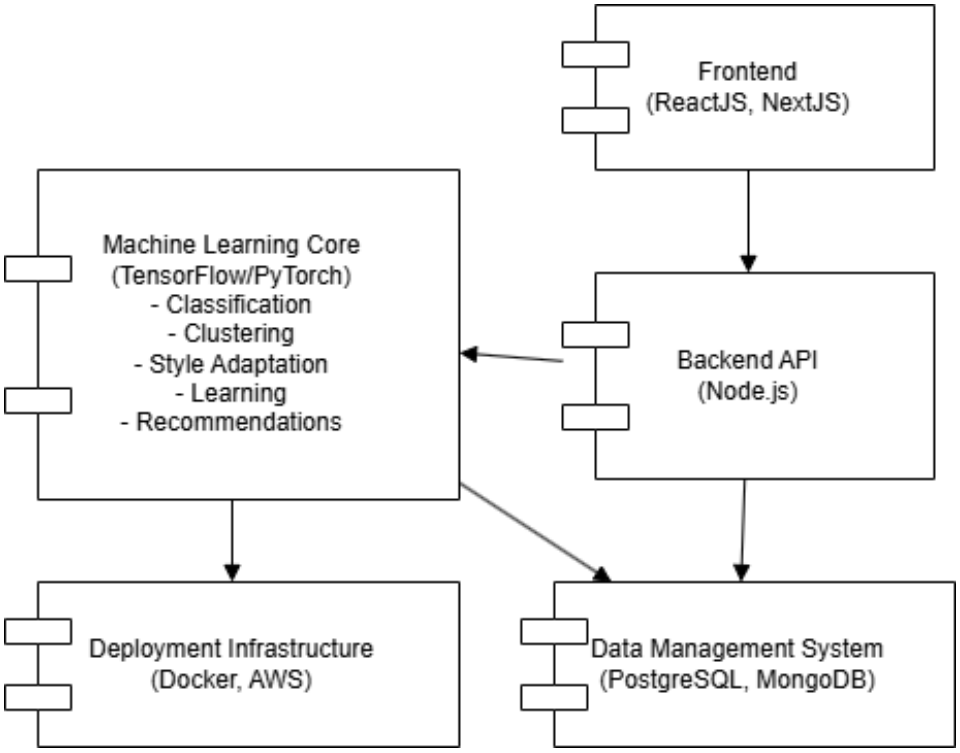


Рисунок 1 – Діаграма компонентів системи (UML Component Diagram)

III. Логіка взаємодії компонентів

Користувач надає програмний код через клієнтський інтерфейс. Система надсилає код до серверної частини, де модуль попередньої обробки підготовлює дані. Далі модуль інтелектуального аналізу застосовує моделі машинного навчання, навчені на реальних наборах коду, для виявлення недоліків. Результати передаються до модуля візуалізації, де відображаються рекомендації. Користувач може взаємодіяти із системою через модуль зворотного зв'язку, впливаючи на подальше вдосконалення моделей. Усі модулі працюють у взаємозв'язку за допомогою API, а хостинг забезпечується через хмарні сервіси з використанням Docker і AWS.

Висновок

У межах дослідження запропоновано систему, здатну підвищувати якість програмного забезпечення завдяки автоматичному аналізу коду із застосуванням машинного навчання. Використання модульної структури, адаптації до стилів програмування та зворотного зв'язку забезпечує гнучкість і масштабованість рішення. Така система може бути цінним інструментом для розробників, покращуючи якість програмного коду та скорочуючи витрати часу на його перевірку.

Список використаних джерел

1. Goodflellow, I., Bengio, Y., & Courville, A (2016). Deep Learning.
2. TensorFlow. An end-to-end open-source machine learning platform [Електронний ресурс]. - Режим доступу: https://www.tensorflow.org.
3. Martin, R. C. (2009). Clean Code: A Handbook of Agile Software Craftsmanship. Prentice Hall.
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. Spivak, I., S. Krepych, Krepych, R., Bayurskii, A. Construction of a criterion for assessing the level of objectivity of experts based on a modified interval expert appraisal method. 2019 IEEE International Scientific-Practical Conference: Problems of Infocommunications Science and Technology, PIC S and T 2019 – Proceedings, Kyiv, pp.311-314
6. PostgreSQL. The world's most advanced open source relational database [Електронний ресурс]. - Режим доступу: https://www.postgresql.org.

УДК 004.4

КІБЕРВАРТОВІ БАЗ ДАНИХ: РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ МОНІТОРИНГУ ТА ПРОАКТИВНОГО РЕАГУВАННЯ НА СУЧАСНІ КІБЕРЗАГРОЗИ

Гончар Л.І.[1)], Сенківський Д.В.[2)], Полюхович П.Г.[3)]
Західноукраїнський національний університет
[1)]к.е.н., доцент; [2-3)] магістрант

І. Постановка проблеми

У сучасному цифровому світі, де обсяги та значущість даних зростають експоненціально, бази даних перетворилися на найпривабливішу мішень для кіберзлочинців. Загрози безпеці, від несанкціонованого доступу та ін'єкцій коду до хитромудрих внутрішніх зловживань, вимагають не просто захисту, а проактивних та інтелектуальних механізмів виявлення та запобігання.

Традиційні, статичні методи захисту, на жаль, часто виявляються безсилими перед обличчям нових, еволюціонуючих типів кібератак, не дозволяючи вчасно реагувати на них. Це створює критичну потребу у розробці принципово нового інтелектуального програмного забезпечення.

ІІ. Мета роботи

Метою даного дослідження є розробка інноваційного програмного забезпечення, що функціонує як невидимий, але пильний щит для баз даних. Це програмне забезпечення покликане здійснювати безперервний, глибокий моніторинг кожної дії у базах даних. Його ключова місія – ефективно виявляти загрози безпеці шляхом всебічного аналізу поведінки користувачів та їхніх запитів, а також забезпечувати блискавичне та своєчасне реагування на будь-які інциденти, використовуючи потужність вбудованих алгоритмів машинного навчання та сучасних систем оповіщення.

ІІІ. Особливості реалізації системи

1. Архітектура та структура: Модульна міць для адаптивного захисту

Розроблене програмне забезпечення є втіленням передової, багаторівневої модульної архітектури. Ця гнучка структура дозволяє системі ефективно реагувати на найрізноманітніші типи загроз, адаптуючись до унікальних потреб кожної організації. Система складається з таких життєво важливих компонентів, що працюють у синергії:

Модуль моніторингу активності: Це серце системи, що здійснює безперервний, детальний запис абсолютно всіх запитів до бази даних. Він фіксує не лише час та IP-адресу, але й дані про користувача, тип запиту та його обсяг. Ці критично важливі дані оперативно передаються у форматі журналів для подальшого, глибинного аналізу. Особливою перевагою цього модуля є його виняткова здатність обробляти колосальні потоки запитів у режимі реального часу, абсолютно без зниження продуктивності системи.

Модуль виявлення аномалій: Справжній "мозок" системи, реалізований на основі передового машинного навчання. Він використовує потужні моделі класифікації та кластеризації, такі як K-Means, IsolationForest та RandomForest, для розпізнавання прихованих загроз. Цей модуль постійно вивчає та запам'ятовує типові шаблони поведінки користувачів, безперервно порівнюючи нові дії з накопиченими історичними даними.

Модуль перевірки правил: Цей модуль діє як надійний фільтр, що містить всеосяжний набір статичних і динамічних правил. Його призначення – безпомилкове виявлення вже відомих та поширених атак, таких як SQL-ін'єкції, XSS та спроби підвищення привілеїв (privilege escalation). Надзвичайно важливо, що в ньому реалізовано автоматичне оновлення сигнатур, що живляться з найактуальніших загальнодоступних баз даних, таких як OWASP та CVE, забезпечуючи постійну актуальність захисту.

Система оповіщення та реакції: Це система швидкого реагування, яка оперативно генерує детальні повідомлення про інциденти. Ці сповіщення миттєво доставляються через різноманітні канали – email, SMS або інтегруються з популярними месенджерами, такими як Telegram та Slack. При виявленні загрози високої критичності система підтримує автоматизовані, рішучі дії: миттєве блокування користувачів, скасування активних сесій або тимчасове обмеження доступу, мінімізуючи потенційну шкоду.

Модуль журналювання та аудиту: Цей модуль є незамінним архівом, що забезпечує незмінне збереження повної історії всіх запитів, кожної зміни у структурі бази даних, прав доступу та системних налаштувань. Дані зберігаються в абсолютно захищеному вигляді, з цифровим підписом та надійним хешуванням SHA-512. Це життєво важливо для проведення глибокого форензик-аналізу (криміналістичного дослідження) у разі будь-яких інцидентів.

Модуль резервного копіювання та відновлення: Наша система гарантує безперервність роботи та цілісність даних завдяки підтримці створення інкрементальних, повних та хронологічних бекапів. Дані зберігаються у високонадійному зашифрованому вигляді (використовуючи AES-256 та RSA) як на захищених хмарних серверах, так і на локальних сховищах. Особлива увага приділяється періодичному тестуванню відновлення, що є критично важливим для верифікації цілісності та гарантії доступності копій у будь-який момент.

Інтерфейс адміністратора: Реалізований як інтуїтивно зрозумілий та потужний веб-додаток з гнучким розподіленим рівнем доступу. Він надає адміністраторам повний контроль та візуалізацію: інтерактивні графіки навантаження, інформативні дашборди загроз, повний контроль над діями користувачів та всебічну аналітику безпеки. Система також надає зручну можливість експорту звітів у популярних форматах (PDF, CSV, JSON) для подальшого аналізу та звітності.

На рисунку 1 представлена архітектура інтелектуальної системи виявлення аномалій у веб-трафіку.

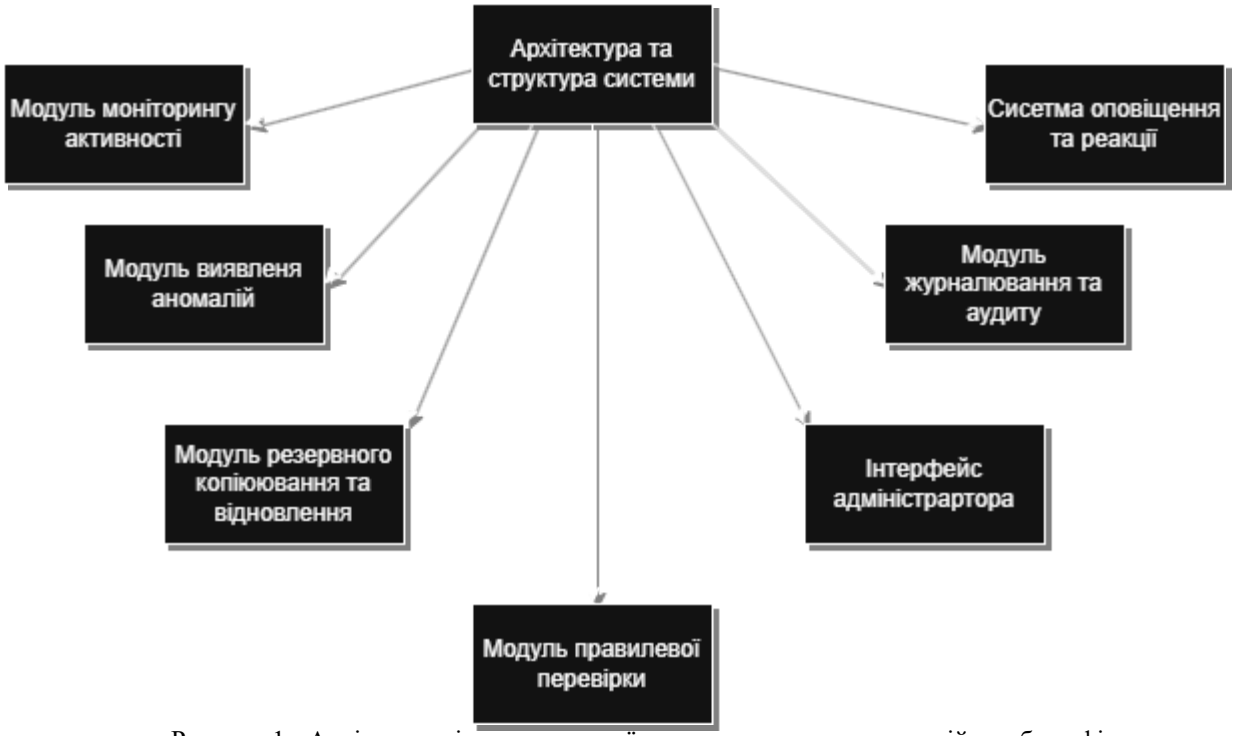


Рисунок 1 - Архітектура інтелектуальної системи виявлення аномалій у веб-трафіку

2. Використані технології та підходи: Фундамент передової безпеки

Розробка цього програмного забезпечення базувалась на міцних, сучасних інженерних та наукових принципах забезпечення безпеки даних, що гарантує його надійність та ефективність:

Алгоритми машинного навчання: Серце інтелектуального захисту.

IsolationForest — для надточного виявлення аномальних точок, без потреби у попередній розмітці класів.

DecisionTree / RandomForest — для високоефективної класифікації запитів, чітко розрізняючи легітимні та загрозливі дії.

Clustering (K-Means) — для розумної сегментації користувачів за їхньою поведінкою, виявляючи групи з подібними патернами.

Дані для навчання цих алгоритмів були ретельно отримані з симульованих сценаріїв реальних атак, включаючи insider threat, brute-force, SQL-injection, що забезпечує високу ефективність виявлення.

Шифрування та передача даних: Безкомпромісна конфіденційність та цілісність.

Стандарт AES-256 для захисту даних у стані спокою, гарантуючи їхню недоступність для сторонніх.

TLS 1.3 для криптографічно захищеного обміну даними між клієнтом та сервером, запобігаючи перехопленню та підробці.

Сертифікати X.509 для надійної автентифікації серверів, підтверджуючи їхню справжність.

Управління доступом: Багаторівневий контроль для максимальної безпеки.

RBAC (Role-Based Access Control) — чіткий поділ прав доступу за ролями користувачів (адміністратор, аналітик, аудитор).

ABAC (Attribute-Based Access Control) — врахування контексту доступу (час, геолокація, тип пристрою) для динамічного прийняття рішень щодо доступу.

Передбачено гнучку можливість інтеграції з існуючими корпоративними системами, такими як LDAP/Active Directory.

Сумісність і стандарти: Відповідність найвищим світовим вимогам.

Система повністю відповідає строгим вимогам міжнародних стандартів: GDPR, ISO/IEC 27001, NIST SP 800-53, PCI DSS, що підтверджує її надійність та відповідність нормативним актам.

Наявність JSON-інтерфейсу забезпечує безшовну інтеграцію з провідними SIEM-системами (Security Information and Event Management), такими як Splunk, ELK Stack та ArcSight, для централізованого управління безпекою.

3. Результати впровадження та тестування: Доведені ефективність та надійність

Для підтвердження ефективності та надійності системи було проведено серію ретельних експериментів у віртуалізованому середовищі, що імітувало реальні умови з високим навантаженням:

Обсяг запитів: Система продемонструвала бездоганну роботу, успішно обробляючи понад мільйон запитів на день.

Середній час виявлення загрози: Блискавичне реагування, що становило менше 0,7 секунди, дозволяє мінімізувати потенційну шкоду.

Точність класифікації аномалій: Вражаюча точність у діапазоні 96–98% (залежно від сценарію), підкреслює надійність алгоритмів машинного навчання.

Фальшиві позитивні спрацьовування: Надзвичайно низький показник – менше 3%, що свідчить про високу якість виявлення без зайвих сповіщень.

Вплив на продуктивність системи: Мінімальний – до 5% на типовому сервері з 8 ядрами та 16 ГБ оперативної пам'яті, що забезпечує стабільну роботу основних систем.

Рівень адаптації системи користувачами: Отримано виключно позитивні оцінки, завдяки прозорій інтеграції з існуючими процесами. Система не вимагала значних змін у поточній інфраструктурі, що забезпечило швидке та легке впровадження.

Висновки

Запропоноване програмне забезпечення є значним кроком вперед у забезпеченні кібербезпеки, гарантуючи ефективне виявлення та моніторинг загроз у базах даних. Воно майстерно поєднує перевірені часом традиційні методи безпеки з революційними, інноваційними підходами машинного навчання.

Його продумана модульна структура, вражаюча масштабованість та бездоганна відповідність найвищим міжнародним стандартам роблять цю систему ідеальним рішенням для застосування у найвимогливіших сферах: державних структурах, фінансовому секторі, охороні здоров’я та великому корпоративному середовищі.

Надійне виявлення аномалій, мінімальний вплив на продуктивність і інтуїтивно зрозумілий інтерфейс дозволяють значно зменшити ризики витоків даних, підвищити загальну довіру до інформаційної системи та впевнено відповідати на всі сучасні виклики кібербезпеки.

Список використаних джерел

1. ISO/IEC 27001:2022 — Міжнародний стандарт систем управління інформаційною безпекою. URL: https://www.iso.org
2. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST). URL: https://csrc.nist.gov
3. Sandhu R., Coyne E.J., Feinstein H.L., Youman C.E. Role-Based Access Control Models. ACM Computing Surveys, 1996.
4. Sujata S., Kamlesh R. SQL Injection Detection and Prevention Techniques: A Survey. IEEE, 2021.
5. Bertino E., Sandhu R. Database Security: Concepts, Approaches, and Challenges. IEEE Transactions on Dependable and Secure Computing, 2005.
6. GDPR — General Data Protection Regulation. URL: https://gdpr-info.eu
7. PCI DSS — Payment Card Industry Data Security Standard. URL: https://www.pcisecuritystandards.org
8. Kumar A., Singh R., Gupta S. A Comprehensive Guide to Database Anomalies Detection. arXiv. URL: https://arxiv.org
9. TensorFlow Security Guide — Інструменти та рекомендації з безпеки ML-систем. URL: https://www.tensorflow.org/security

УДК 004.4

СИСТЕМА АВТОМАТИЗОВАНОГО ПОСТИНГУ В ГРУПАХ FACEBOOK

Войтюк І.Ф.[1)], Бойко А.А.[2)]
Західноукраїнський національний університет
[1)] к.т.н., доцент; [2)] бакалавр

І. Постановка проблеми

Публікація інформації в Facebook-групах є важливою складовою цифрового маркетингу та просування послуг. Водночас ручне розміщення контенту у десятках або сотнях груп потребує значних часових витрат, координації акаунтів та постійного контролю. Це створює потребу у програмному забезпеченні, яке дозволить автоматизувати процес постингу, уникнути рутинних дій та зменшити ризик блокування облікових записів. Важливим є також забезпечення можливості інтеграції з CRM-системами, з яких береться вихідна інформація для постів.

ІІ. Мета роботи

Метою роботи є дослідження процесів автоматизації публікацій у Facebook-групах та створення програмного продукту, що дозволяє масове розміщення контенту з різних акаунтів і джерел, із мінімальним втручанням користувача. Система повинна автоматизувати процес авторизації, чергового постингу, заповнення форм доступу та інтеграцію з CRM для витягування контенту.

ІІІ. Основна частина

У межах роботи було створено консольну програму мовою C# з використанням платформи .NET 7.0. Основна реалізація базується на принципах модульності, об’єктно-орієнтованого програмування та шаблонного підходу до генерації текстів. У проєкті застосовано реляційну базу даних PostgreSQL для збереження облікових записів, логів, CRM-записів та черг публікацій. Структура бази відповідає 3-й нормальній формі та включає сутності users, groups, posts, crm_records та logs. Ключовим компонентом системи є механізм формування черги постів для кожного акаунта. Черга реалізована за принципом FIFO і забезпечує ізольовану обробку для кожного облікового запису. Це дозволяє уникнути конфліктів при одночасному постингу від декількох користувачів. Кожен пост містить шаблонізований контент, у який динамічно підставляються дані з CRM-систем (наприклад, назва міста, вид послуги, опис вакансії).

Реалізовано повноцінний retry-механізм, який дозволяє у випадку помилки (мережевий збій, недійсний токен, помилка Facebook API) виконати повторну публікацію із затримкою. Система підтримує гнучке логування з розділенням по акаунтах, а також фіксує статус кожної публікації. Антиспам-механізм включає випадкові затримки між постами та унікалізацію текстів за допомогою шаблонів, що дозволяє зменшити ймовірність блокування.

Для підключення зовнішніх CRM використовується універсальний підхід – дані імпортуються у вигляді JSON у таблицю crm_records, після чого постобробка та вставка у шаблони виконується автоматично. Це дозволяє легко адаптувати систему до змін у структурі вхідних даних.

Програму протестовано із використанням юніт-тестів (фреймворк xUnit, бібліотека Moq), а також у рамках дослідної експлуатації на кількох акаунтах. Під час тестування перевірялась робота механізмів постингу, retry, шаблонізації, взаємодії з CRM та логування.

Загальна архітектура системи побудована за модульним принципом, де кожен функціональний блок (завантаження акаунтів, обробка шаблонів, взаємодія з CRM, публікація, логування) виконує чітко визначене завдання та взаємодіє з іншими через контрольований інтерфейс. Такий підхід забезпечує легку підтримку та можливість масштабування проєкту. Наприклад, за потреби можна реалізувати нові модулі для інтеграції з іншими соціальними мережами (Telegram, Instagram) або зовнішніми джерелами даних без зміни основної логіки системи.

Вихідні шаблони публікацій розміщуються у текстових файлах (.txt), які зчитуються динамічно при формуванні черги. Це дозволяє змінювати зміст постів без перекомпіляції програми. Крім того, система дозволяє підключати довільну кількість акаунтів — кожен з них обробляється окремим контекстом, що виключає перетин потоків або дублювання.

Процес публікації організований у вигляді черги, яка обробляється в циклі з урахуванням контрольованих затримок. У разі успішної публікації відповідний запис оновлюється зі статусом SENT, а у разі помилки фіксується в лог-файл з причиною. Якщо помилка тимчасова (наприклад, помилка мережі), активується механізм повторної спроби з лімітованою кількістю циклів.

Окремим підрозділом реалізована система логування, яка створює окремі лог-файли по кожному акаунту у форматі log_email.txt. Це дозволяє локалізовано відстежити помилки або події публікації, не змішуючи дані між обліковими записами.

На рисунку 1 наведено загальну архітектуру системи, де відображено основні програмні модулі, зв'язки між ними та зовнішні сервіси.

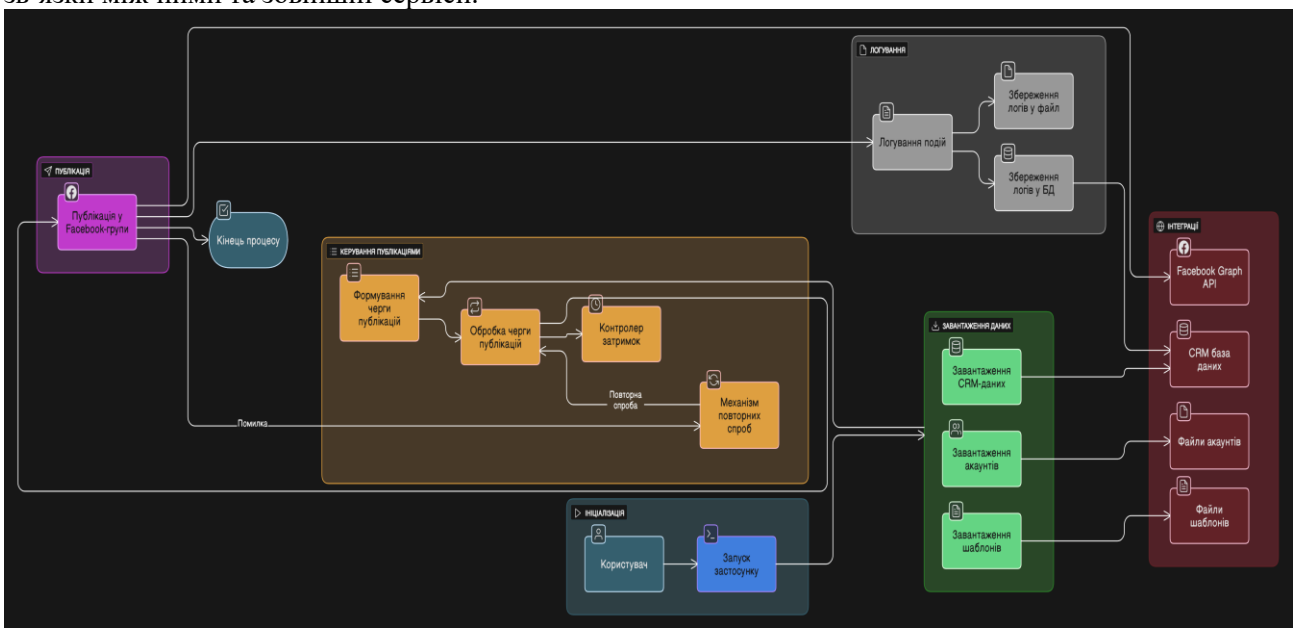


Рисунок 1 – Діаграма загальної архітектури системи

Попри ефективну реалізацію основних функцій, застосування створеної системи у реальних умовах стикається з низкою обмежень, пов'язаних із політикою доступу до Facebook API. Згідно з офіційною документацією Meta, можливість програмного постингу в групах доступна лише за умови, що акаунт є адміністратором відповідної групи і надав додатку відповідні дозволи через механізм перевірки прав (App Review). Це означає, що у випадках, коли система мала б працювати із загальнодоступними сторонніми групами, функціональність публікації буде заблокована на рівні API або спричинить помилки доступу.

Ще одним практичним обмеженням є заборона на автоматизоване приєднання до груп, заповнення форм при вступі та проходження CAPTCHA. Жоден із цих процесів не підтримується через офіційні інструменти Facebook, а емуляція поведінки користувача (через headless-браузери чи скрипти) є порушенням умов використання платформи. Такі дії несуть ризик блокування акаунтів, додатків, або обмеження на IP-рівні, що унеможливлює масштабне використання системи в умовах комерційного розгортання.

З огляду на зазначене, система доцільна для застосування в обмеженому середовищі — наприклад, для роботи з власними Facebook-групами, у яких користувач має адміністративні права. Крім того, архітектура проєкту дозволяє трансформувати її під роботу з іншими платформами, такими як Telegram, Discord або Slack, які мають відкриті API для автоматизації без жорстких обмежень. Таким чином, проєкт зберігає інженерну цінність та може бути адаптований до умов, які не суперечать правилам використання цільових платформ.

Висновок

У результаті виконання дипломної роботи було створено систему автоматизованого постингу, яка дозволяє суттєво скоротити час і ресурси, необхідні для масового розміщення контенту в соціальних мережах, зокрема у Facebook-групах. Система є масштабованою, підтримує інтеграцію з CRM-системами та відповідає сучасним вимогам до програмної інженерії. Отримані результати можуть бути використані як основа для розширення функціоналу (інтерфейс адміністратора, вебверсія, інтеграція з Telegram та іншими платформами).

Список використаних джерел

1. "Що таке Facebook Graph API?" [Електронний ресурс] – Режим доступу: https://developers.facebook.com/docs/graph-api
2. "Що таке .NET 7.0?" [Електронний ресурс] – Режим доступу: https://learn.microsoft.com/dotnet
3. Бончук Л. І., Главацький І. В. "Інженерія програмного забезпечення: навч. посіб." – Львів: Львівська політехніка, 2020. – 356 с.
4. Чекмарьов С. Г. "Тестування програмного забезпечення: навчальний посібник." – Харків: ХНУРЕ, 2018. – 214 с.

УДК 004.4

СИСТЕМА АВТОМАТИЗАЦІЇ ОБЛІКУ КЛІЄНТІВ І ПРОДАЖІВ ДЛЯ МАЛОГО БІЗНЕСУ

Братців К.І.
Західноукраїнський національний університет
бакалавр

І. Постановка проблеми

У сучасних умовах стрімкого розвитку інформаційних технологій та глобальної цифрової трансформації бізнес-процесів питання автоматизації управління клієнтськими даними та продажами набуває особливого значення, особливо для сегменту малого бізнесу. Малі підприємства, які зазвичай мають обмежені фінансові, технічні та людські ресурси, стикаються з численними викликами під час впровадження складних і дорогих CRM (Customer Relationship Management) або ERP (Enterprise Resource Planning) систем.

Такі системи часто є надмірно складними, потребують значних інвестицій у навчання персоналу, інтеграцію та підтримку, що робить їх недоступними для багатьох малих компаній. Через ці обмеження значна частина малих підприємств в Україні та світі продовжує покладатися на ручні методи обліку клієнтів і продажів.

Використання електронних таблиць, паперових журналів або інших неавтоматизованих інструментів призводить до низки проблем, таких як помилки в даних, втрата важливої інформації про клієнтів, дублювання записів, а також ускладнення аналізу бізнес-показників. Усе це знижує операційну ефективність, ускладнює прийняття обґрунтованих управлінських рішень і обмежує можливості для масштабування бізнесу. Крім того, сучасний ринок характеризується високою конкуренцією, що змушує малі підприємства шукати способи оптимізації своїх процесів для підвищення продуктивності та залучення клієнтів.

У цьому контексті автоматизація стає не просто конкурентною перевагою, а необхідною умовою для виживання та розвитку. Однак більшість доступних на ринку програмних рішень не враховують специфіку малого бізнесу, зокрема локального контексту, що включає обмежений бюджет, невелику кількість працівників і потребу в простих, інтуїтивно зрозумілих інструментах. Це формує запит на розробку доступних, гнучких і адаптованих рішень, які могли б забезпечити систематизацію інформації про клієнтів, замовлення, історію взаємодії та ключові бізнеспоказники без надмірних витрат

ІІ. Мета роботи

Метою даної роботи є створення простої, доступної та функціональної інформаційної системи, призначеної для автоматизації процесів обліку клієнтів і продажів у малому бізнесі. Розробка спрямована на задоволення специфічних потреб малих підприємств, зокрема тих, що працюють у локальному контексті, шляхом забезпечення зручного інструменту для управління клієнтськими даними, замовленнями та аналітикою продажів. Система має сприяти підвищенню операційної ефективності, зменшенню кількості помилок, пов’язаних із ручним обліком, а також спрощенню управлінських процесів.

Досягнення поставленої мети передбачає вирішення таких завдань: аналіз потреб малого бізнесу, розробка модульної архітектури системи, вибір сучасного технологічного стеку, створення інтуїтивно зрозумілого інтерфейсу для користувачів без технічної підготовки, а також забезпечення можливості швидкого впровадження та масштабування системи в майбутньому. У результаті система має стати доступним рішенням, яке допоможе малим підприємствам оптимізувати свої бізнес-процеси та підвищити конкурентоспроможність на ринку.

ІІІ. Основна частина

Для реалізації поставленої мети було проведено ґрунтовний аналіз потреб малого бізнесу в Україні. Дослідження показало, що переважна більшість малих підприємств потребує базових інструментів для управління клієнтськими даними, створення та відстеження замовлень, ведення історії покупок, формування звітності, а також роботи з програмами лояльності. Водночас підприємства прагнуть уникати складних і дорогих рішень, які потребують тривалого навчання чи залучення спеціалістів для впровадження.

На основі отриманих даних було розроблено інформаційну систему з модульною архітектурою, яка забезпечує гнучкість і можливість подальшого розширення функціоналу. Для реалізації серверної частини системи обрано платформу ASP.NET Core, яка відзначається високою продуктивністю, кросплатформною сумісністю та підтримкою сучасних стандартів розробки. Для створення інтерфейсу користувача використано бібліотеку React, що дозволяє будувати динамічні та адаптивні вебінтерфейси. Як систему управління базами даних обрано SQLite – легку вбудовану СУБД, яка не потребує складного налаштування та ідеально підходить для малого бізнесу з обмеженими ресурсами. Інтерфейс системи розроблено з урахуванням потреб користувачів без технічної підготовки. Він має інтуїтивно зрозумілу навігацію, мінімалістичний дизайн і оптимізований процес внесення даних, що дозволяє швидко освоїти систему навіть непідготовленим працівникам. Завдяки цьому підприємства можуть самостійно впроваджувати систему без залучення додаткових спеціалістів, що знижує витрати на її використання.

Система реалізує такі ключові функціональні модулі:

- Облік клієнтів: модуль дозволяє зберігати деталізовану інформацію про клієнтів, включаючи сегмент (наприклад, роздрібні чи оптові покупці), контактні дані, історію взаємодії та покупки. Це забезпечує персоналізований підхід до кожного клієнта.
- Управління замовленнями: модуль підтримує створення замовлень, відстеження їх статусів та формування аналітичних звітів для оцінки ефективності продажів.
- Програми лояльності: система дозволяє створювати та адмініструвати програми лояльності, а також вести облік відгуків клієнтів, що сприяє підвищенню їхньої задоволеності.
- Аналітика продажів: модуль забезпечує гнучку фільтрацію даних за клієнтами, періодами, типами замовлень і іншими параметрами, що допомагає власникам бізнесу приймати обґрунтовані рішення.

Особливу увагу при розробці приділено адаптації системи до реальних сценаріїв роботи малого бізнесу. Наприклад, передбачено швидке внесення даних із мінімальною кількістю обов'язкових полів, інтеграцію з електронною поштою для автоматичних нагадувань клієнтам, а також збереження повної історії взаємодії для подальшого аналізу. Усе це робить систему зручною та ефективною для повсякденного використання.

Висновок

Розроблена інформаційна система для автоматизації обліку клієнтів і продажів є ефективним рішенням для малого бізнесу, яке відповідає ключовим вимогам сучасного програмного забезпечення: простоті, доступності, гнучкості та високій продуктивності. Завдяки використанню сучасного технологічного стеку – ASP.NET Core, React і SQLite – система забезпечує стабільну роботу, швидке розгортання та можливість масштабування в разі зростання потреб підприємства. Запропоноване рішення сприяє цифровізації малого бізнесу, що є важливим кроком до підвищення його конкурентоспроможності та економічного розвитку.

Система дозволяє підприємствам оптимізувати управління клієнтськими даними, зменшити кількість помилок, пов'язаних із ручним обліком, і покращити якість прийняття управлінських рішень завдяки аналітичним інструментам. У перспективі система може бути розширена шляхом інтеграції з платіжними сервісами, хмарними сховищами, інструментами маркетингової аналітики чи іншими зовнішніми платформами. Це відкриває нові можливості для її використання в різних галузях і сегментах ринку.

Впровадження таких розширень дозволить системі залишатися актуальною в умовах швидкої еволюції технологій і потреб бізнесу, забезпечуючи малим підприємствам інструменти для ефективного розвитку та конкуренції на ринку.

Список використаних джерел

1. Боднар О.Л., Манжула В.І. ERP-СИСТЕМА ДЛЯ ВНУТРІШНІХ БІЗНЕС-ПРОЦЕСІВ КОМПАНІЇ. Комп'ютерні інформаційні технології: Матеріали весняної школи-семінару молодих вчених і студентів СІТ'2024. – Тернопіль: ЗУНУ, 2025, 110 с.
2. Бутенко Н.В. Впровадження концепції CRM на промисловому ринку. Економіка та держава. 2011. № 3. С. 40–42.
3. Freeman A., Smith J. Pro ASP.NET Core 3. Apress, 2020. 78-102 с.
4. Overview of ASP.NET Core MVC | Microsoft Learn. [Електронний ресурс]. – Режим доступу: https://learn.microsoft.com/en-us/aspnet/core/mvc/overview?view=aspnetcore-9.0
5. Типи баз даних: особливості, відмінності та приклади | DOU. [Електронний ресурс]. – Режим доступу: https://dou.ua/lenta/articles/types-of-databases/
6. Next.JS: сучасна технологія для веб-розробки | Wezom. [Електронний ресурс]. – Режим доступу: https://wezom.com.ua/ua/blog/rozbirajemo-freymvork-nextjs-viznachennya-priznachennya-ta-perevagi

УДК 004.896.3

МОБІЛЬНИЙ ДОДАТОК ДЛЯ КОНТРОЛЮ ХАРЧУВАННЯ

Малайна О.І.[1)], **Вовчук Н.В.**[2)], **Денис Н.В.**[3)], **Беч Д.Р.**[3)], **Думка І.М.**[4)]
Західноукраїнський національний університет
[1-4)] *студент*

І. Постановка проблеми

У ритмі сучасного життя все менше часу залишається на повноцінне планування раціону, контроль калорійності, режиму харчування та балансу нутрієнтів. Переїдання, пропущені прийоми їжі, перекуси з високим вмістом цукру чи жиру — типові проблеми, з якими стикаються мільйони людей. Здорове харчування перетворюється з рекомендації дієтолога на щоденний виклик. У таких умовах мобільний додаток для контролю харчування стає не просто корисним інструментом — він формує нову культуру харчової поведінки, підтриману даними, аналітикою та індивідуальним підходом [1,2].

ІІ. Мета роботи

Мобільний додаток контролю харчування - це програмне забезпечення для смартфонів, яке дозволяє користувачам:

- фіксувати прийоми їжі;
- розраховувати калорійність та склад страв;
- слідкувати за вмістом білків, жирів, вуглеводів, води;
- будувати персоналізований раціон згідно з метою (схуднення, набір ваги, підтримка форми);
- отримувати поради, нагадування й аналітику в зручному інтерфейсі.

Найсучасніші додатки використовують штучний інтелект, розпізнавання зображень страв, інтеграцію з фітнес-браслетами, а також соціальні функції (наприклад, виклики між друзями на тему «хто сьогодні харчувався краще»).

ІІІ. Проектування архітектури сервісу

Архітектура мобільного додатку спроєктована з дотриманням принципів розділення відповідальностей (SoC), модульності, інверсії залежностей та відмовостійкості. Базовою парадигмою обрано MVVM (Model–View–ViewModel), яка забезпечує зручне тестування, масштабованість і відокремлення логіки від представлення даних.Узагальнено, архітектура додатку охоплює:

- клієнтську частину (мобільний застосунок),
- серверну частину (REST API),
- систему керування даними (база даних),
- зовнішні сервіси (аналітика, розпізнавання, push-сповіщення).

Клієнтська частина (Frontend). Мобільний застосунок розроблено на платформі Android з використанням Kotlin та бібліотек Jetpack. Компоненти:

1. View (UI) – реалізовано з використанням JetpackCompose, з адаптивною підтримкою темної/світлої теми, доступності (accessibility) і анімацій.
2. ViewModel – логіка бізнес-процесів, обробка даних, управління станами (StateFlow).
3. Model – опис сутностей: User, Meal, Product, TrackingEntry, Goal.
4. RoomDatabase – зберігання локальних даних для офлайн-доступу (ORM поверх SQLite).
5. Retrofit – клієнт HTTP-запитів до серверного API.
6. FirebaseCloudMessaging – отримання сповіщень про прийоми їжі та зміну рекомендацій.
7. GoogleFit API – синхронізація з фітнес-даними (кроки, калорії, пульс).

Особливості. Кешування даних з автоматичним оновленням при з’єднанні.Підтримка офлайн-режиму.Локалізація інтерфейсу.

Серверна частина (Backend). Реалізована на базі Node.js (Express) або альтернативно Laravel (PHP). Основні функціональні модулі:

1. Контролери (API layer) — маршрутизація запитів;
2. Сервісна логіка (Servicelayer) — обробка запитів, перевірка цілей, формування рекомендацій;
3. Модуль розпізнавання страв — звернення до зовнішнього ML-сервісу;

4. JWT Middleware — захист маршрутів, перевірка авторизації;
5. Email/Push-модуль — надсилання повідомлень;
6. Сервіс рекомендацій — базова логіка або ML-модель.

REST API реалізовано з урахуванням принципів RESTful-дизайну, передбачено валідацію даних (Joi або LaravelFormRequests), рисунок 1.

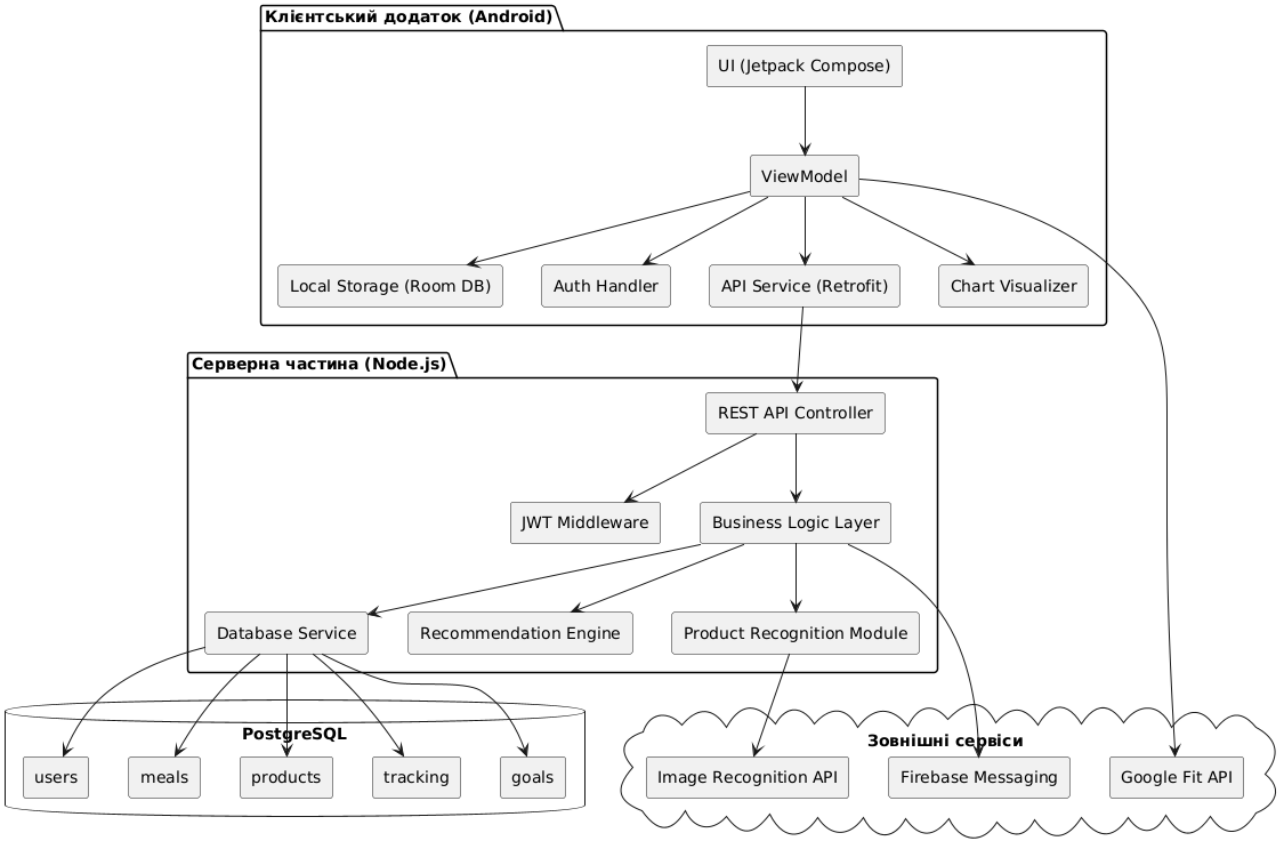


Рисунок 1 – Архітектура сервісу

Зважаючи на поширеність операційної системи Android серед цільової аудиторії, було обрано платформу Android SDK з використанням мови Kotlin. Як архітектурну модель було застосовано MVVM (Model–View–ViewModel), яка забезпечує модульність, легку підтримку й тестування компонентів.

Серверна частина реалізована з використанням Node.js та бази даних PostgreSQL, що дозволяє масштабувати проєкт при збільшенні кількості користувачів. Для взаємодії між клієнтом і сервером реалізовано REST API із захистом через JWT-токени.

Моніторинг води та ваги. Щодня користувач отримує сповіщення щодо споживання води, введення ваги, рівня активності. Дані вносяться вручну або синхронізуються з GoogleFit.

Візуалізація даних. За допомогою бібліотеки MPAndroidChart реалізовано побудову графіків калорійності, співвідношення БЖВ, зміни маси тіла.Алгоритм персоналізованих підказок базується на простій логіці:

- якщо калорійність спожитого перевищує денну норму >20%, з'являється рекомендація знизити споживання;

- якщо не досягнуто мети по білках, додаток радить продукти з високим вмістом білка.

Висновок

Мобільні додатки для контролю харчування — це приклад ефективної взаємодії технологій і щоденних практик. Вони стають особистими дієтологами, які не лише рахують калорії, а й мотивують, підказують і формують здорові звички. Завдяки інтелектуальним алгоритмам, аналітиці та зручному інтерфейсу, такі додатки допомагають перетворити турботу про харчування з рутини на інструмент самопізнання та покращення якості життя.

Список використаних джерел

1. Freedman, M. R. (2011). CalorieConfusion: TeachingtheConsumerto "EatSmart". NutritionToday, 46(4), 166–170.
2. Whelan, K. &Lang, R. (2020). Nutrition: A VeryShortIntroduction. OxfordUniversityPress.

УДК 004.896.3

ПРОГРАМНА СИСТЕМА ДЛЯ МАСОВИХ РОЗСИЛОК

Толуб'як М.С.[1)], **Степовенко Д.С.**[2)], **Сімчук Ю.О.**[3)]
Західноукраїнський національний університет
[1-3)]*студент*

І. Постановка проблеми

У XXI столітті електронна комунікація стала невід'ємною частиною діяльності будь-якої організації — незалежно від галузі, масштабів чи географічного розташування. В умовах цифровізації зростає потреба у швидкому, гнучкому та контрольованому обміні інформацією з великою кількістю отримувачів. Саме тому системи масових розсилок (bulkmailingsystems) набули широкого поширення як ефективний інструмент організації інформаційної взаємодії з цільовими аудиторіями [1,2].

Системи масових розсилок дозволяють централізовано формувати, керувати та доставляти повідомлення великому числу адресатів через електронну пошту, SMS, push-нотифікації або інші цифрові канали. Вони знайшли застосування у багатьох сферах: від маркетингу, електронної комерції та новинних служб — до внутрішніх комунікацій в освітніх закладах, органах державної влади, банківських структурах і великих корпоративних мережах.

Проте, попри значну популярність подібних систем, існує низка викликів, пов'язаних з їх розробкою, впровадженням і масштабуванням. До таких викликів належать:

- персоналізація повідомлень з урахуванням атрибутів отримувачів;
- оптимізація часу доставки та обробка великої кількості запитів;
- надійність і стійкість до помилок, зокрема відсутність дублювань або втрати повідомлень;
- інтеграція з іншими інформаційними системами;
- аналітика результатів (відсоток доставлених/відкритих/прочитаних повідомлень);
- забезпечення інформаційної безпеки (аутентифікація, авторизація, контроль доступу, захист від спаму та атак).

Зважаючи на вищевказані потреби, особливу актуальність набуває створення гнучкої, масштабованої, модульної системи масових розсилок, яка може бути адаптована під різні задачі й умови експлуатації. Така система повинна забезпечувати простоту керування через зручний інтерфейс, підтримувати складні сценарії фільтрації цільової аудиторії, мати розвинену інфраструктуру логування та моніторингу, а також бути сумісною з корпоративними стандартами безпеки.

ІІ. Мета роботи

Метою роботи є охарактеризувати архітектуру, функціональність, технічну реалізацію та практичні аспекти застосування системи масових розсилок у сучасних IT-інфраструктурах

ІІІ. Проектування архітектури системи

Проєктування архітектури системи масових розсилок базується на низці ключових принципів програмної інженерії, що забезпечують гнучкість, масштабованість, розширюваність і стійкість до відмов:

Модульність — функціональні компоненти системи (керування розсилками, шаблони, доставка, аналітика) реалізуються у вигляді окремих модулів із чіткими інтерфейсами;

Мікросервісна архітектура — дозволяє ізолювати компоненти для незалежної розробки, тестування та масштабування;

Розділення відповідальності — реалізовано поділ на презентаційний, логічний і інфраструктурний рівні;

Підтримка асинхронних процесів — застосування черг повідомлень (наприклад, RabbitMQ), які знижують навантаження на основний сервіс під час масових відправлень;

Безпечна обробка даних — забезпечується за рахунок контролю доступу, авторизації, шифрування та ведення журналу подій.

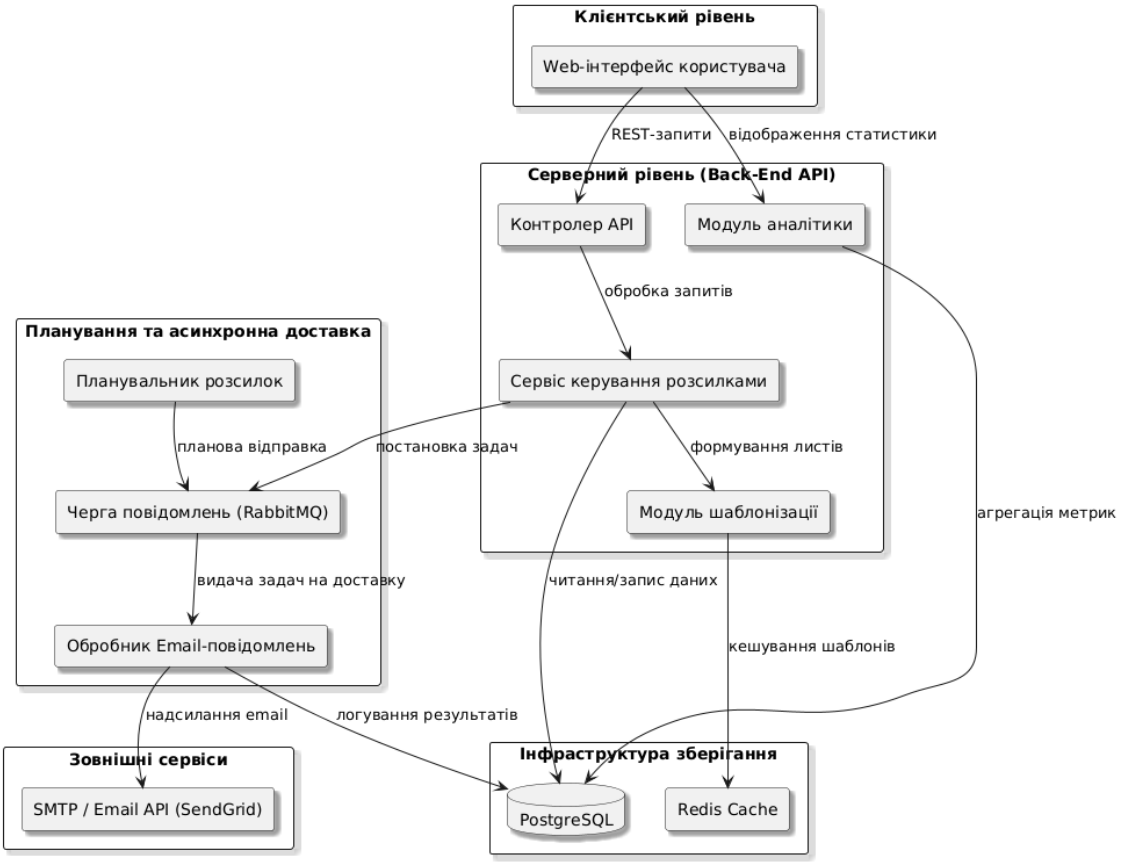


Рисунок 1–Архітектура системи масових розсилок

На логічному рівні система розподіляється на наступні функціональні компоненти:

1. Інтерфейс користувача (UI/UX) — реалізований у вигляді вебінтерфейсу, через який адміністратор або уповноважений користувач керує розсилками, фільтрує аудиторії, працює з шаблонами повідомлень;

2. Серверна логіка (API-сервер) — приймає запити з інтерфейсу, обробляє їх, здійснює валідацію та передає завдання у відповідні модулі;

3. Модуль шаблонізації — дозволяє створювати динамічні повідомлення з використанням шаблонів і змінних (ім’я користувача, дата події тощо);

4. Сервіс доставки — відповідає за фізичну відправку листів або повідомлень через SMTP, Email API або інші канали зв’язку;

5. Модуль аналітики — формує статистику успішності розсилок, відстежує статуси, помилки, показники відкриття (openrate, clickrate);

6.База даних — зберігає інформацію про користувачів, шаблони, історію розсилок, результати та логи.

Для реалізації системи обрано такий стек технологій:Back-end: Java (SpringBoot), або PHP (Laravel);Front-end: React.js, Tailwind CSS;База даних: PostgreSQL, Redis (для кешування шаблонів);Черга повідомлень: RabbitMQ;Поштова інтеграція: SMTP (через JavaMail або SwiftMailer), або сторонні сервіси (SendGrid, Mailgun);Контейнеризація: Docker для розгортання сервісів;CI/CD: GitHubActions або Jenkins для автоматизації деплою;Моніторинг: Prometheus, Grafana.

Висновок

Проєктування архітектури системи масових розсилок передбачає використання перевірених технологій, модульний підхід та підтримку масштабованості. Такий підхід дозволяє забезпечити ефективне функціонування системи в умовах великої кількості користувачів, високої частоти запитів і потреби у безперервній роботі. У подальшому можлива реалізація багатоканальної розсилки, розпаралелення задач на основі мікросервісів та підтримка мобільних платформ.

Список використаних джерел

1. Sommerville, I. SoftwareEngineering. 10th ed. Boston: Pearson, 2016. 792 p.
2. Richardson, C. (2018). MicroservicesPatterns: WithexamplesinJava. ManningPublications.

УДК 004.896.3

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ ОБМІНУ ВАЛЮТ

Дячок Б.А.[1)], **Дериш А.Р.**[2)], **Ганяк Р.П.**[3)], **Арапов В.В.**[4)], **Савчук В.В.**[5)]
Західноукраїнський національний університет
[1-5)]*студент;*

І. Постановка проблеми

У сучасних умовах цифрової трансформації фінансового сектору спостерігається стрімке зростання попиту на автоматизовані засоби обміну валют, зокрема у формі веб-орієнтованих додатків. Це обумовлено потребою в оперативному, зручному та безпечному доступі до фінансових операцій незалежно від часу та місця перебування користувача. Розвиток електронної комерції, дистанційної роботи, а також активізація міжнародної діяльності як фізичних осіб, так і суб'єктів господарювання зумовлюють необхідність мати ефективні засоби для швидкого обміну валюти з мінімальними часовими та транзакційними витратами [1].

Класичні методи валютного обміну, які передбачають особисте відвідування банківських відділень або пунктів обміну валют, все частіше замінюються цифровими сервісами, які дозволяють здійснювати такі операції в онлайн-режимі. Проте більшість наявних рішень орієнтовані на великі фінансові організації або обмежуються функціоналом перегляду курсів валют без повноцінної інтеграції з банківськими API, системами користувацької ідентифікації, веденням історії транзакцій та формуванням аналітичних звітів [2].

ІІ. Мета роботи

Метою дослідження є створення веб-орієнтованого програмного забезпечення, яке дозволяє автоматизувати процес обміну валют з урахуванням безпеки, точності й масштабованості.

ІІІ. Проектування архітектури системи

Процес проєктування веб-додатку для обміну валют охоплює етапи аналізу вимог, вибору технологічного стеку, побудови загальної архітектури, моделювання логіки взаємодії компонентів та розробки структури бази даних. Ретельне проєктування забезпечує масштабованість, надійність, безпеку та зручність у подальшій підтримці та модернізації системи.

Визначення функціональних та нефункціональних вимог. На етапі формування вимог до системи були виділені такі основні функціональні компоненти:

- перегляд курсів валют у режимі реального часу;
- виконання операцій з обміну валют (вибір пар, суми, підтвердження транзакції);
- калькулятор обміну з урахуванням комісій;
- авторизація та реєстрація користувачів;
- історія операцій користувача;
- інтерфейс адміністратора для перегляду звітності;
- підтримка мультивалютності.

Серед нефункціональних вимог основними були:

- висока доступність (24/7);
- захищений обмін даними (HTTPS, SSL, токенізація);
- адаптивність інтерфейсу для мобільних пристроїв;
- обробка до 10 000 запитів щодня з низькою латентністю;
- інтеграція з банківськими API (НБУ, ПриватБанк, Monobank).

Система має багаторівневу архітектуру з чітким розподілом відповідальностей. Клієнтський рівень (PresentationLayer):забезпечує взаємодію користувача з системою, динамічну генерацію інтерфейсів.Серверний рівень (ApplicationLayer):обробляє бізнес-логіку, керує сесіями користувача, обробляє API-запити.Рівень даних (DataLayer):відповідає за зберігання інформації в СУБД, виконує запити через ORM.

Програмна реалізація веб-додатку для обміну валют базується на розробленій архітектурі з чітким розмежуванням клієнтської, серверної та інфраструктурної частин. Основними принципами при реалізації були: модульність, масштабованість, захищеність та підтримка RESTful API.

Реалізація клієнтської частини (Frontend). Клієнтська частина розроблена на основі Vue.js як односторінковий додаток (SPA). Застосовується компонентний підхід. Всі компоненти логічно

згруповані:Exchange.vue — обмін валют;Login.vue, Register.vue — аутентифікація;Dashboard.vue — персональний кабінет;AdminPanel.vue — панель адміністратора.

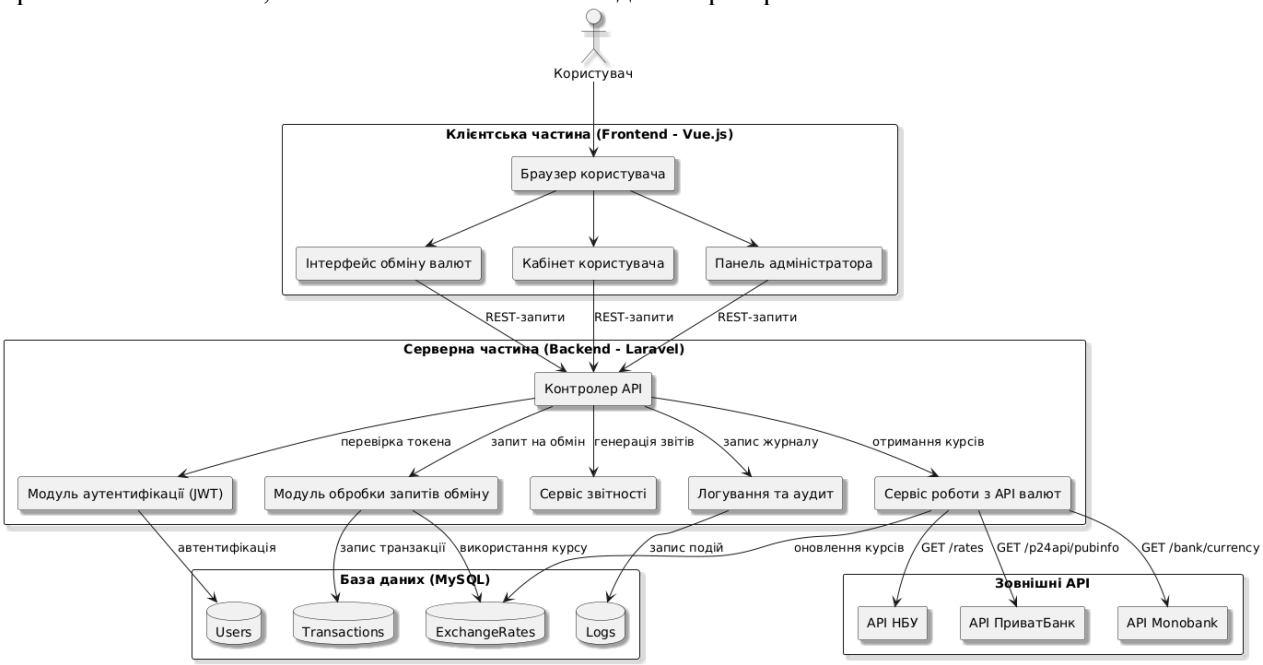


Рисунок 1–Архітектура системи

Реалізація серверної частини (Backend). Сервер реалізовано з використанням Laravel (версія 10), що забезпечує безпечну маршрутизацію, ORM-доступ до БД та підтримку REST API.Для автоматичного оновлення курсів валют щоденно виконується CRON-скрипт, який викликає API НБУ, ПриватБанку та Monobank. Реалізовано це через LaravelScheduler.Реалізовані механізми захисту:реєстрація з валідацією та шифруванням паролів (bcrypt);токенізована авторизація через LaravelSanctum;захист API через middlewareauth:sanctum;HTTPS-з'єднання (у продакшн-середовищі);захист від SQL-ін'єкцій та CSRF.

Інтерфейс користувача (UI) є ключовим компонентом взаємодії кінцевого користувача із веб-додатком для обміну валют. Його якість визначає загальний досвід використання системи, рівень інтуїтивності, зручність навігації, адаптивність до різних пристроїв та доступність. При проєктуванні та реалізації інтерфейсу було враховано принципи UX-дизайну, а також специфіку роботи з фінансовими транзакціями.

Інтерфейс реалізовано у вигляді односторінкового додатку (SPA) з використанням фреймворку Vue.js. Це дозволило організувати логіку інтерфейсу у вигляді незалежних компонентів, що динамічно оновлюються без повного перезавантаження сторінки.Основні компоненти інтерфейсу:HomePage.vue — головна сторінка з курсами валют;ExchangeForm.vue — форма обміну валют;Login.vue, Register.vue — вхід та реєстрація;Dashboard.vue — особистий кабінет користувача;AdminPanel.vue — панель адміністратора;TransactionHistory.vue — історія обмінів;RateGraph.vue — графік динаміки курсів.

Висновок

Реалізація веб-додатку була виконана з урахуванням сучасних стандартів розробки веб-систем. Розділення архітектури на компоненти frontend, backend і інтеграційний шар забезпечує гнучкість, простоту тестування і масштабування. Laravel дозволив швидко впровадити ключову бізнес-логіку, тоді як Vue.js забезпечив швидкий рендеринг і сучасний користувацький досвід. Структура бази даних є достатньо гнучкою для подальшого розширення системи, наприклад, через додавання мультивалютних гаманець, графіків або API для мобільних клієнтів.

Список використаних джерел

1. Kho, Kimberly Mae M., Renz Paul T. Fababeir, and Julianne Dominique J. Torres. 2024. "An Analysis of the Impact on Modern Web Application Development: Basis for PHP Framework Efficiency Model." Polytechnic University of the Philippines, February 2024.
2. Runsewe, Oluwayemisi, Olajide Soji Osundare, Samuel Folorunsho, and Lucy Akwawa. 2024. "Optimizing User Interface and User Experience in Financial Applications: A Review of Techniques and Technologies." World Journal of Advanced Research and Reviews 23, no. 3 (September): 934–942.

УДК 004.896.3

ВЕБ СЕРВІС ДЛЯ АНАЛІЗУ РИНКУ ПРАЦІ

Коваль А.В.[1)], **Олексійовець Я.А.**[2)], **Машталєр О.А.**[3)], **Мясковський О.О.**[4)]
Західноукраїнський національний університет
[1-4)]*студент*

І. Постановка проблеми

Сучасний ринок праці динамічно трансформується під впливом як зовнішніх, так і внутрішніх чинників. Натомість система професійної освіти залишається здебільшого академізованою, що ускладнює її адаптацію до потреб економіки. Це, у свою чергу, спричиняє низький рівень затребуваності випускників серед роботодавців. В умовах постійної зміни запитів ринку критично важливо забезпечити узгодження між освітніми програмами та реальною потребою в конкретних знаннях і навичках [1-2]. Одним із найбільш доступних джерел актуальної інформації про стан ринку праці сьогодні виступає інтернет-простір. Спеціалізовані вебресурси з пошуку роботи містять публікації, у яких вказуються вимоги до кандидатів, необхідні компетенції та функціональні обов'язки. З огляду на високу частоту оновлення та щоденну появу нових вакансій і резюме, цей інформаційний масив має великий потенціал для системного аналізу.

Незважаючи на це, випускники часто стикаються з труднощами у працевлаштуванні через невідповідність своїх знань ринковим вимогам, а компанії, у свою чергу, відчувають гостру нестачу кваліфікованих кадрів. Завдання закладів вищої освіти полягає у своєчасному виявленні потреб ринку праці та формуванні у студентів відповідних професійних компетентностей. Реалізувати це можливо завдяки застосуванню сучасних методів аналізу великих даних, зокрема шляхом обробки текстів вакансій, які роботодавці регулярно публікують з докладним описом очікувань.

ІІ. Мета роботи

Метою дослідження є розробка сервісу для аналізу ринку праці.

ІІІ. Проектування архітектури вебсервісу

На початковому етапі проєктування було ухвалено рішення відмовитися від використання мікросервісної архітектури на користь монолітного підходу, що зумовлено обмеженістю ресурсів, зокрема часу та кількості залучених розробників. Як результат, для створення вебзастосунку обрано традиційну клієнт-серверну архітектуру, яка передбачає поділ проєкту на три основні компоненти:

Клієнт — елемент, що безпосередньо взаємодіє з користувачем: формує запити до сервера, очікує на відповіді та здійснює їхню візуалізацію або обробку на інтерфейсному рівні;

Сервер — центральний логічний компонент, що приймає запити від клієнта, обробляє їх (зокрема, виконує операції з даними, взаємодіє з базами даних або сторонніми сервісами) та повертає сформовану відповідь;

База даних (БД) — компонент для зберігання, організації та управління структурованими даними, що використовуються в рамках функціонування застосунку.

Архітектурна схема цієї взаємодії подана на рисунку 1 — «Схема клієнт-серверної архітектури».

Принципи взаємодії компонентів. Взаємодія між компонентами системи реалізується за наступною послідовністю:

1. Запит клієнта до сервера – ініціюється користувачем для отримання, створення, оновлення або видалення даних;

2. Запит сервера до бази даних – сервер обробляє запит клієнта та звертається до бази даних із відповідною операцією;

3. Обробка та відповідь – після отримання даних від бази, сервер обробляє їх, формує відповідь і надсилає її клієнтові.

Під час розробки проєкту виконавцю було надано вільний вибір технологічного стеку та програмного забезпечення майже в усіх аспектах, за винятком двох обов'язкових компонентів, вже визначених замовником:

ClickHouse – колоночна система управління базами даних, яка використовується для зберігання агрегованих аналітичних даних про вакансії, зібраних із сайтів-агрегаторів;

Minio S3 – масштабоване об’єктне файлове сховище, яке вже впроваджене на стороні компанії і використовується для зберігання неструктурованих даних у межах проєкту.

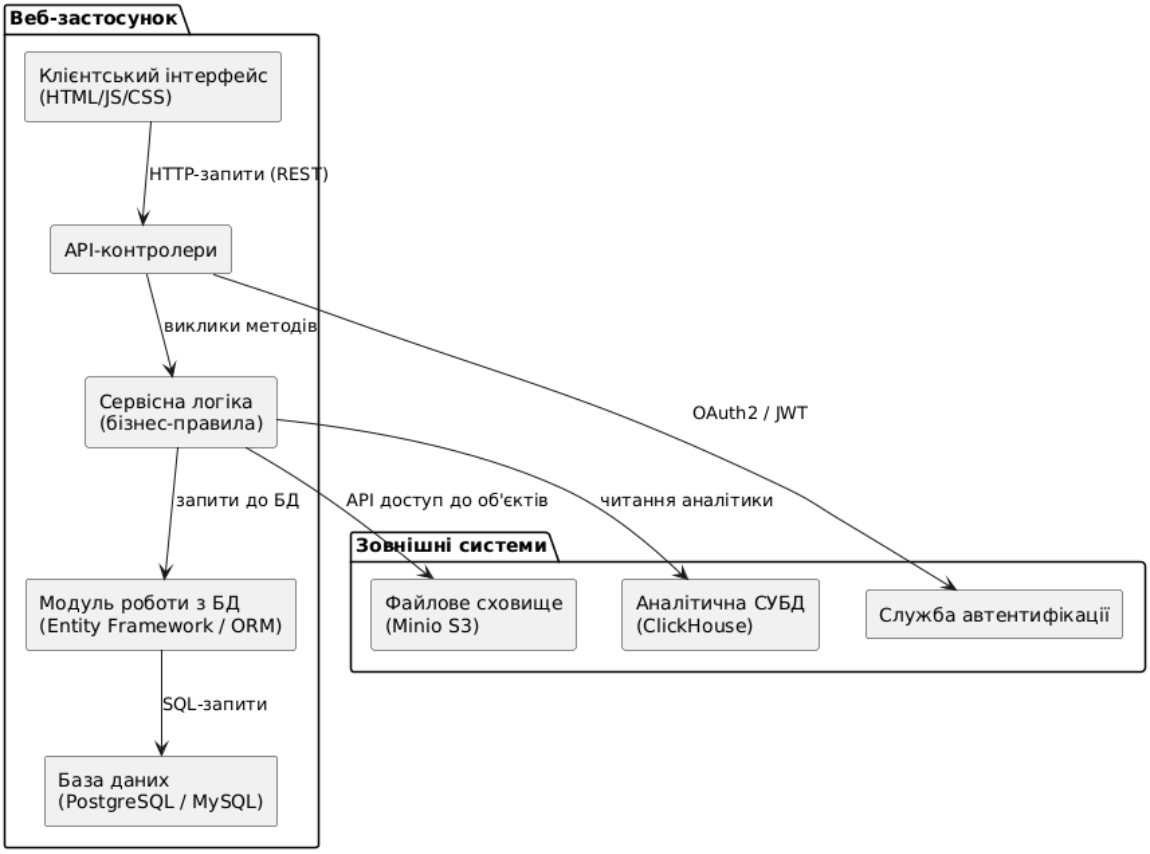


Рисунок 1–Архітектура сервісу

Використання ClickHouse. ClickHouse є повнофункціональною стовпцевою СУБД, яка організовує дані у вигляді стовпців і обробляє їх пакетно (у вигляді масивів, або chunks). Такий підхід, відомий як векторизоване виконання запитів, дозволяє знизити витрати на обробку та суттєво підвищити продуктивність при роботі з великими обсягами аналітичної інформації.

Схема збирання даних організована наступним чином, парсери збирають дані з агрегаторів вакансій і зберігають їх у чотирьох окремих базах PostgreSQL (по одній на кожен джерело);

Далі, за допомогою спеціальних Python-скриптів, ці дані обробляються, агрегуються та записуються в таблицю ClickHouse, яка містить 58 аналітичних стовпців.Таким чином, на рівні зберігання інформації система використовує чотири взаємодоповнюючі компоненти:

1. PostgreSQL — для транзакційної логіки та облікових даних;
2. ClickHouse — для зберігання агрегованих аналітичних масивів;
3. Minio S3 — для зберігання великих об’єктів (файлів);
4. Redis — для кешування часто запитуваних результатів з PostgreSQL і ClickHouse.

Серверна частина застосунку. Для реалізації серверної логіки було обрано мову програмування Go (Golang), що забезпечує високу продуктивність та масштабованість застосунку.

Висновок

Таким чином, розроблений сервіс відповідає сучасним вимогам до гнучких, безпечних та масштабованих веб-рішень і має значний потенціал для подальшої еволюції в рамках цифрової трансформації бізнес-процесів.

Список використаних джерел

1. Wolf, F. A., Arquint, L., Clochard, M., Oortwijn, W., Pereira, J. C., &Müller, P. (2021). Gobra: Modular specification and verification of Goprograms. CAV’21 Conference Proceedings.
2. Emmanni, P. S. (2021). The role of TypeScript in enhancing development with modern JavaScript frameworks. International Journal of Science and Research (IJSR), 10(2).

УДК 004.896.3

МОБІЛЬНИЙ СЕРВІС ДЛЯ УПРАВЛІННЯ ПРОЦЕСАМИ ОРЕНДИ ТЕХНІКИ

Вітюнова А.В.[1)],**Вівчар М.І.**[2)],**Мазур Ю.О.**[3)]
Західноукраїнський національний університет
[1-3)]*студент*

І. Постановка проблеми

Сучасні ІТ-рішення в галузі логістики, сервісного обслуговування та управління технічними ресурсами дедалі частіше орієнтуються на мобільні платформи, що зумовлено широким розповсюдженням смартфонів, підвищенням продуктивності мобільних пристроїв і загальною цифровізацією бізнес-процесів. В умовах активного використання техніки в таких ключових галузях, як будівництво, аграрний сектор, промисловість, енергетика та ІТ-інфраструктура, виникає об'єктивна потреба в створенні гнучких, масштабованих і зручних у користуванні інформаційних систем, які забезпечують прозоре, оперативне та ефективне управління процесами оренди обладнання[1].

Такі системи повинні надавати змогу клієнтам швидко отримувати актуальну інформацію про наявність техніки, технічні характеристики, умови оренди, а також створювати й відстежувати заявки у режимі реального часу. З іншого боку, адміністративний персонал повинен мати змогу централізовано контролювати життєвий цикл оренди, переглядати фінансову аналітику, обробляти документи та підтримувати зворотний зв’язок із користувачами. Саме тому мобільні сервіси, як складова частина цифрових платформ управління ресурсами, стають невід’ємним інструментом оптимізації логістичних і сервісних операцій на сучасному етапі розвитку економіки[2].

ІІ. Мета роботи

Метою створення мобільного сервісу є забезпечення автоматизованої підтримки життєвого циклу процесу оренди техніки: від моменту реєстрації користувача до завершення дії договору. Реалізація такого сервісу дозволяє підвищити прозорість процедур, зменшити час обробки заявок, оптимізувати контроль за наявністю техніки та борговими зобов’язаннями орендарів.

Для досягнення цієї мети було розроблено клієнтську частину застосунку для мобільної платформи Android, реалізовано REST API для взаємодії з серверною частиною, а також спроєктовано структуру бази даних для зберігання основних сутностей предметної області.

ІІІ. Проектування архітектури сервісу

Розроблений сервіс включає такі основні модулі:

1. Модуль каталогу техніки — забезпечує перегляд доступної техніки, її параметрів, фотографій та умов оренди;

2. Модуль формування заявки — дозволяє користувачеві створити заявку на оренду з вибором дати, тривалості та типу обладнання;

3. Модуль керування документами — реалізує завантаження та зберігання сканованих копій паспортів, водійських посвідчень, договорів тощо;

4. Модуль контролю оренди — відображає поточні оренди, статус платежів, строки завершення договорів;

5. Модуль адміністратора — передбачає зміну статусу заявки, підтвердження оплати, внесення нових одиниць техніки до системи.

Застосунок реалізовано відповідно до архітектурної моделі MVVM (Model–View–ViewModel). У межах цієї моделі:

- View (UI) — реалізовано за допомогою JetpackCompose;
- ViewModel — містить логіку взаємодії з даними;
- Model — включає об'єкти предметної області та сервіси для роботи з API/БД.

Для обміну даними із сервером використовується бібліотека Retrofit, а для збереження даних локально — RoomDatabase.На стороні сервера реалізовано Laravel REST API з використанням JWT-аутентифікації та MySQL як основної бази даних. Структура взаємодії клієнта й сервера подана на рисунку 1.

Зберігання даних у клієнтському середовищі здійснюється за допомогою RoomDatabase (ORM над SQLite), що забезпечує кешування даних та офлайн-доступ. Для аутентифікації застосовується інтеграція з FirebaseAuth або власне API з JWT-токенами.

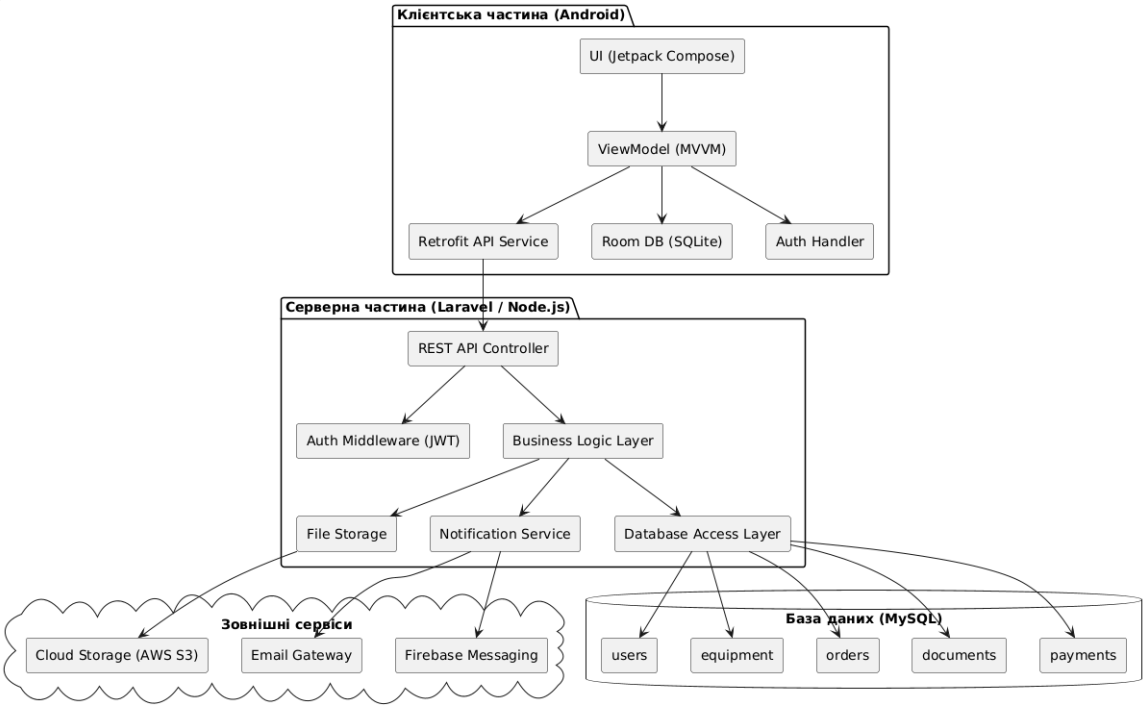


Рисунок 1–Архітектура сервісу

Серверна частина реалізована у вигляді REST API на платформі Laravel (PHP) з використанням бази даних MySQL. Ключові маршрути забезпечують:

- отримання списку техніки;
- створення, підтвердження і скасування заявок;
- обробку файлів (через файлове сховище або S3);
- обробку платежів та формування звітів.

Контроль доступу реалізовано через middleware з перевіркою автентифікаційнихтокенів, а також ролей користувачів (звичайний користувач, адміністратор). Для надсилання push-сповіщень використовується FirebaseCloudMessaging, а для email-повідомлень — SMTP-шлюз Laravel.

Мобільний клієнт надсилає HTTP-запити до серверного API через Retrofit. Усі запити супроводжуються JWT-токенами авторизації. У разі відсутності інтернет-з'єднання застосунок переходить до читання з локальної бази. Обробка відповідей та помилок виконується у ViewModel.У разі змін статусу заявки (наприклад, підтвердження оренди адміністратором) сервер автоматично генерує push-сповіщення, яке доставляється клієнту через Firebase.

У результаті реалізовано стабільний та масштабований мобільний застосунок, що підтримує повний цикл взаємодії клієнта з орендодавцем. Застосунок протестовано на пристроях з Android 10–13. Час завантаження даних становить у середньому 250–400 мс, рівень стабільності (CrashFree) під час тестування склав 99,8 %.

Висновок

Реалізований мобільний сервіс забезпечує повноцінну підтримку процесів оренди техніки, охоплюючи всі ключові аспекти — від обробки заявки до контролю платежів. Вибір платформи Android, спільне використання сучасних засобів Jetpack, Retrofit та Laravel API дозволили досягти високої продуктивності, розширюваності та зручності інтерфейсу. Розробка має практичну цінність для підприємств, які надають техніку в оренду, та може бути адаптована для інших сфер (лізинг транспорту, прокат спорядження тощо).

Список використаних джерел

1. Mednieks, Z., Dornin, L., Meike, G., & Nakamura, M. Programming Android: Java Programming for the New Generation of Mobile Devices. 2nd ed. Sebastopol: O'ReillyMedia, 2012. 516 p.
2. Molen, J. van. Laravel: Up and Running. A Framework for Building Modern PHP Apps. 2nd ed. Sebastopol: O'ReillyMedia, 2019. 350 p.

УДК 004.896.3

ІНФОРМАЦІЙНА СИСТЕМА ДЛЯ АНАЛІЗУ ТА ОБЛІКУ БУДІВЕЛЬНИХ ВИТРАТ

Варчук О.Д.[1)], **Лужецький В.М.**[2)], **Сохан В.В.**[3)]
Західноукраїнський національний університет
[1-3)]*студент;*

І. Постановка проблеми

Будівельна галузь є однією з найресурсоємніших сфер економіки, в якій ефективне управління фінансами має ключове значення. Через велику кількість задіяних сторін, масштабність проєктів і значну частку ручної праці в обліку витрат, часто виникають труднощі з точністю, своєчасністю та прозорістю фінансової інформації. Типовими проблемами є дублювання даних, складність аналізу відхилень від бюджету, неузгодженість звітності та складність контролю витрат в режимі реального час[1,2].

Застосування автоматизованої інформаційної системи дозволяє вирішити вищезгадані проблеми шляхом централізованого зберігання інформації, спрощення процесів введення даних, автоматичного формування звітів та реалізації алгоритмів аналітики. Запропонована система надає користувачам інструменти для ведення обліку витрат, планування бюджету, контролю перевитрат, аналізу фінансових показників і прийняття обґрунтованих управлінських рішень[3].

Відтак, виникає необхідність у розробці комплексної інформаційної системи для аналізу та обліку витрат, що забезпечить достовірність даних, вчасність аналізу та прозорість управлінських рішень.

ІІ. Мета роботи

Метою створення такої системи є забезпечення прозорого обліку будівельних витрат, моніторингу фінансового стану проєктів, формування аналітичних звітів та прогнозування відхилень від бюджету.

ІІІ. Проектування архітектури системи

Система побудована за клієнт-серверною архітектурою. Клієнтська частина — веб-додаток, реалізований за допомогою фреймворків Vue.js або React.js. Серверна частина — Laravel або Node.js. База даних — PostgreSQL або MySQL.Додатково інтегровано зовнішні сервіси: Chart.js (візуалізація), XLS/PDF експорт, API інтеграції з бухгалтерським ПЗ (BAS ERP).

Розроблена інформаційна система передбачає реалізацію широкого спектра функціональних можливостей, які охоплюють весь цикл управління фінансами у будівельних проєктах. Основні функції включають:

1. Ведення детального обліку витрат за категоріями (будівельні матеріали, заробітна плата, оренда техніки, логістика, адміністративні витрати);
2. Побудова бюджету на основі проєктної документації із вказанням запланованих витрат на кожному етапі;
3. Формування внутрішніх управлінських і зовнішніх фінансових звітів для контролю діяльності;
4. Візуалізація витрат у вигляді графіків, таблиць і діаграм, що дозволяє відстежувати тенденції, виявляти відхилення та приймати рішення в оперативному режимі;
5. Організація зберігання документації, що підтверджує витрати (акти, рахунки, накладні) у цифровому вигляді з можливістю їх подальшого перегляду й аналізу;
6. Реалізація багаторівневої системи доступу з розмежуванням прав користувачів (менеджери, бухгалтери, інженери, керівники);
7. Реєстрація змін у даних з веденням журналу аудиту для забезпечення підзвітності та контролю.

Архітектура інформаційної системи побудована за принципом розділення логіки між клієнтською, серверною частиною та базою даних. Вибрано модель "тонкий клієнт — потужний сервер". Веб-інтерфейс забезпечує користувачам зручний доступ до функціоналу з будь-якого пристрою через браузер. Серверна частина відповідає за бізнес-логіку, валідацію, обчислення та взаємодію з базою даних.

Компоненти системи, рисунок 1. Клієнтська частина (Frontend): реалізована за допомогою фреймворків Vue.js або React.js, забезпечує відображення даних, форми введення, аналітичні

панелі.Серверна частина (Backend): реалізована у середовищі Laravel або Node.js. Обробляє запити, перевіряє права доступу, виконує запити до бази даних.База даних: PostgreSQL або MySQL. Включає всі сутності, пов’язані з проєктами, витратами, користувачами, документами тощо.Зовнішні сервіси: API для експорту у формати Excel, PDF; модулі візуалізації (Chart.js, ApexCharts); інтеграція з бухгалтерськими системами.Завдяки модульній архітектурі, система може бути масштабована, доповнена новими модулями (наприклад, прогнозування витрат, машинне навчання) та адаптована під специфіку конкретного підприємства.

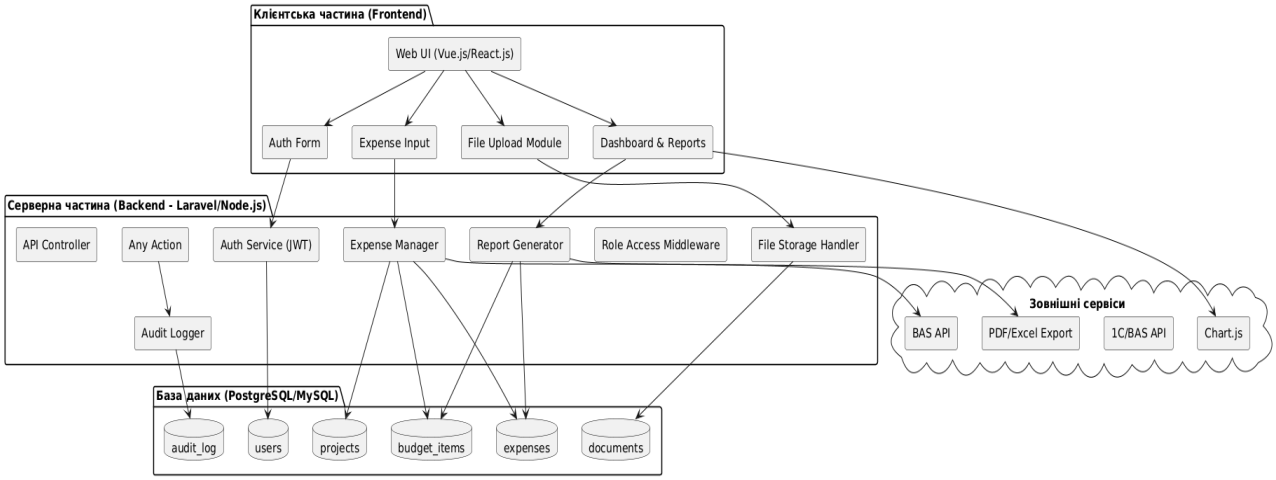


Рисунок 1–Архітектура системи

Аналітична підсистема системи забезпечує гнучку візуалізацію фінансових показників. Основні компоненти:

1. Діаграма витрат за категоріями — кругові графіки дозволяють побачити структуру витрат;
2. Динаміка витрат у часі — лінійні графіки для виявлення сезонних чи проектних піків;
3. Порівняння план/факт — стовпчикові діаграми та таблиці для оперативного реагування на відхилення;
4. Модуль експорту — можливість формування звітів у форматах PDF, Excel для внутрішнього чи зовнішнього використання.

Реалізація обліку витрат.Користувач, що має відповідні повноваження, додає новий запис про витрати через інтерфейс: обирає категорію, вказує суму, дату, прикріплює підтверджуючий документ. Система автоматично перевіряє перевищення встановленого бюджету та повідомляє про відхилення. Усі дані зберігаються в базі та одразу доступні для аналізу й формування звітності. Завдяки цьому формується єдиний центр контролю витрат, що знижує ймовірність фінансових зловживань і дублювання інформації.

Система реалізує комплексний підхід до безпеки:аутентифікація користувачів здійснюється з використанням JWT-токенів; доступ до функціоналу обмежено ролями (RBAC);Всі критичні дії логуються в окрему таблицю audit_log, передбачено шифрування персональних файлів;Підтримується багатофакторна автентифікація (опційно), система веде контроль цілісності збережених документів.

Висновок

Розроблена інформаційна система для аналізу та обліку будівельних витрат забезпечує повний цикл управління фінансовими потоками в рамках будівельного проєкту — від планування бюджету до формування фінансової звітності. Система підтримує інтеграцію з бухгалтерським обліком, дозволяє виявляти критичні перевитрати та оптимізувати фінансові ресурси. Завдяки модульній архітектурі система може бути адаптована до потреб будівельних компаній різного масштабу.

Список використаних джерел

1. Семенченко, В. І. Інформаційні системи в економіці будівництва: навчальний посібник. Київ: КНЕУ, 2020. 278 с.
2. Sommerville, I. SoftwareEngineering. 10th ed. Boston: Pearson, 2016. 792 p.
3. Череп, А. В., Ткаченко, І. В. Системи управління ресурсами підприємств у будівництві (ERP): методи впровадження та оптимізація. // Економіка та держава. – 2022. – №12. – С. 53–58.

УДК 004.896.3

ІНФОРМАЦІЙНА ВЕБСИСТЕМА ДЛЯ ОНЛАЙН-ЗНАЙОМСТВА СТУДЕНТІВ

Дементьєв Р.В.[1)], **Сулейманов Р.Т.**[2)], **Костак В.В.**[3)]
Західноукраїнський національний університет
[1-3)]*студент;*

І. Постановка проблеми

Сучасні університети дедалі більше інтегрують цифрові технології в навчальний процес, адміністративне управління та соціально-культурне середовище. Проте, попри стрімкий розвиток інформаційних систем у сфері освіти, питання ефективної комунікації та соціальної взаємодії студентів залишається недостатньо вирішеним. Особливо це стосується новоприбулих студентів, які стикаються з труднощами адаптації, соціалізації та пошуку однодумців у новому середовищі.

Водночас досвід функціонування масових платформ знайомств (Tinder, Bumble, FacebookDating тощо) засвідчує високий попит на сервіси для пошуку контактів. Однак жодна з наявних платформ не враховує специфіку академічного середовища, не гарантує безпеки взаємодії в межах одного університету та не забезпечує фільтрації за релевантними критеріями — факультетом, курсом, напрямом навчання, участю у студентських ініціативах.

У цьому контексті особливої актуальності набуває створення спеціалізованої інформаційної системи, орієнтованої на потреби студентства. Така система має сприяти розвитку горизонтальних зв'язків, формуванню спільнот за інтересами, підвищенню академічної інтеграції та особистісному розвитку.

ІІ. Мета роботи

Мета дослідження. Розробити веборієнтовану інформаційну систему онлайн-знайомств, призначену для студентів одного університету, яка забезпечує:

-реєстрацію та автентифікацію на основі університетських облікових записів;

- створення та налаштування профілю користувача;
- інтелектуальний пошук за інтересами, курсом, факультетом;
- можливість взаємодії (вподобання, чат, участь у подіях);
- підтримку безпечного середовища спілкування через модерацію та механізми блокування.

Об'єкт дослідження. Процеси цифрової соціалізації студентів в умовах вищої освіти.

Предмет дослідження. Архітектура, моделі та інтерфейси веборієнтованої системи для знайомства та комунікації студентів.

ІІІ. Проектування та програмна реалізація

Проєктування архітектури є ключовим етапом у створенні сучасного вебзастосунку, оскільки воно визначає логіку взаємодії всіх компонентів системи, обсяг їх відповідальності, спосіб обміну даними та забезпечення масштабованості, безпеки й підтримки.

Вибір архітектурного підходу.

З огляду на вимоги до інтерфейсної зручності, швидкості реагування, адаптивності та відокремлення клієнтської й серверної частин, було обрано архітектурний підхід SPA (SinglePageApplication). У такій архітектурі весь фронтенд реалізується як єдина сторінка, що динамічно оновлюється без повного перезавантаження. Це забезпечує високу швидкодію, плавний користувацький досвід і гнучкість при майбутньому масштабуванні.

Серверна частина системи представлена як RESTfulWeb API, що працює з базою даних і надає доступ до функціональності через HTTP-запити у форматі JSON.

Загальна структура системи. Система складається з трьох основних рівнів:

Клієнтський рівень (Frontend) – реалізовано за допомогою фреймворкуAngular 8 з використанням TypeScript, SCSS та бібліотеки AngularMaterial.

Серверний рівень (Backend) – реалізовано на основі ASP.NET CoreWeb API 3.1 з використанням архітектурної моделі MVC без View, ORM EntityFrameworkCore, DI та JWT.

Рівень даних – база даних, створена на Microsoft SQL Server, а також зовнішнє хмарне сховище Cloudinary для зберігання зображень.

Взаємодія компонентів. Зв'язок між клієнтом і сервером забезпечується через HTTP-запити: Angular-інтерфейс надсилає запити до Web API, отримує у відповідь JSON-дані й оновлює інтерфейс

без перезавантаження сторінки. Всі критично важливі дії (реєстрація, логін, надсилання повідомлень) захищені за допомогою токенів доступу.

Хмарне сховище. Для уникнення перевантаження бази даних файлами мультимедіа обрано Cloudinary як зовнішнє сховище для зображень. Зображення не зберігаються безпосередньо в БД, а лише їхні посилання.

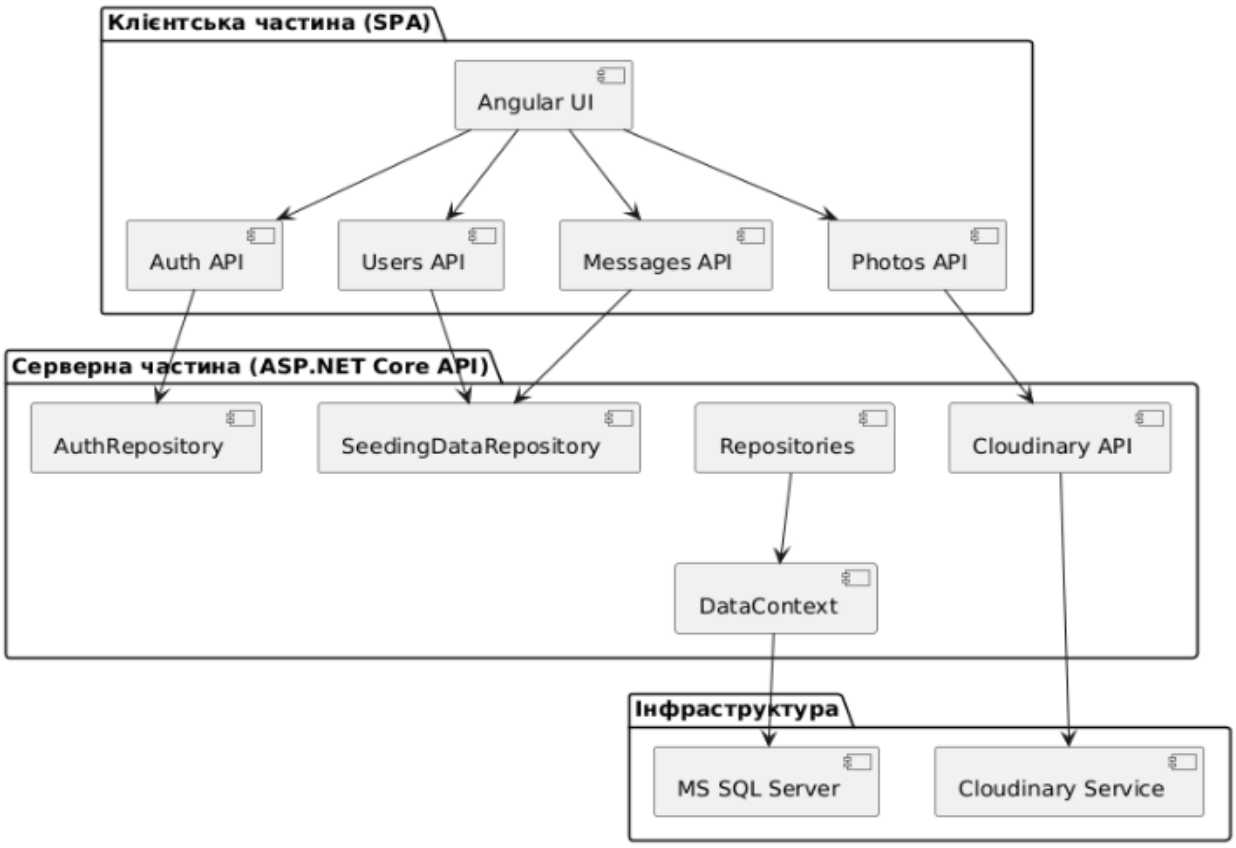


Рисунок 1–Архітектура системи

Розроблено архітектуру системи на основі парадигми SPA (SinglePageApplication), у якій клієнтська частина реалізована з використанням Angular 8, а серверна частина — з використанням ASP.NET CoreWeb API. Застосовано підхід розділення обов'язків та впроваджено шаблони проєктування MVC, DI, Repository.

Спроєктовано та реалізовано базу даних на основі Microsoft SQL Server 2019. Всі дані користувачів, повідомлення та події зберігаються в структурованому вигляді. Зображення профілів зберігаються у хмарному сервісі Cloudinary, що забезпечує масштабованість і оптимізацію роботи з медіафайлами.

Висновок

Реалізовано безпечну систему автентифікації з використанням JWT-токенів та хешування паролів із унікальною «сіллю» на основі алгоритму SHA-512, що дозволяє ефективно протистояти атакам типу «радужні таблиці» та словниковим атакам.

Проведено багаторівневе тестування системи, що охоплює модульне, інтеграційне, функціональне, UI/UX та навантажувальне тестування. За результатами випробувань встановлено стабільність роботи, відповідність функціональності очікуванням, а також високий рівень безпеки та зручності користування.

Таким чином, розроблена система UniConnect повною мірою задовольняє поставлені вимоги та може бути рекомендована до використання як сучасний інструмент для підвищення рівня студентської взаємодії в межах навчального закладу.

Список використаних джерел

1. Watson, J. (2019). Beginning C# and .NET: FromNoviceto Professional (8th ed.). Apress.
2. Troelsen, A., &Japikse, P. (2021). Pro C# 10 with .NET 6: FoundationalPrinciplesandPracticesinProgramming (11th ed.). Apress.

УДК 004.896.3

ІНФОРМАЦІЙНА ВЕБСИСТЕМА ДЛЯ УПРАВЛІННЯ ДОКУМЕНТОПОТОКАМИ ФІРМИ НА ОСНОВІ НЕРЕЛЯЦІЙНОЇ СУБД

Арапов В.В.[1)], **Підзирайло Х.Я.**[2)], **Коломієць С.П.**[3)]
Західноукраїнський національний університет
[1-3)]*студент*

І. Постановка проблеми

У сучасних умовах цифрової трансформації бізнесу однією з ключових проблем ефективної організації внутрішніх процесів у компаніях є управління документообігом. Зростання обсягів інформації, що циркулює між відділами, підрозділами та зовнішніми контрагентами, вимагає створення таких систем, які забезпечують зберігання, структуризацію, доступ, фільтрацію та захист документів у зручному та автоматизованому вигляді [1].

Традиційні рішення управління документообігом часто реалізовані на основі реляційних СУБД, які передбачають жорстко фіксовану схему даних. Такий підхід обмежує можливості гнучкої адаптації до нових типів документів, особливо у випадках, коли структура метаданих є змінною, або коли в системі одночасно працюють різні типи користувачів із різним рівнем доступу. У свою чергу, документоорієнтовані бази даних (наприклад, MongoDB) надають можливість зберігати напівструктуровані або неструктуровані дані, що дозволяє динамічно змінювати схему зберігання без втрати узгодженості та продуктивності.

З іншого боку, зростає популярність веборієнтованих інформаційних систем, які надають доступ до функціоналу управління документами через браузер. Це дозволяє забезпечити кросплатформеність, доступність у будь-який час, централізоване адміністрування та підтримку спільної роботи над документами [2].

Особливу актуальність така система має для малих та середніх підприємств, які прагнуть оптимізувати внутрішні бізнес-процеси без залучення складних та дорогих корпоративних ECM (EnterpriseContentManagement) рішень. Створення простої, адаптивної та ефективної вебсистеми для управління документообігом на базі NoSQL-технологій дозволяє забезпечити баланс між функціональністю, швидкістю розгортання та масштабованістю.

ІІ. Мета роботи

Метою є створення веборієнтованої інформаційної системи управління документопотоками, що використовує потенціал NoSQL-СУБД для ефективного керування цифровими документами фірми, включаючи створення, категоризацію, пошук і контроль доступу.

У результаті реалізації такої системи компанія отримує гнучкий інструмент, який сприяє зменшенню часу обробки документів, підвищенню прозорості процесів, поліпшенню керованості та відповідності вимогам інформаційної безпеки.

ІІІ. Проектування та програмна реалізація

Проєктування інформаційної системи управління документообігом базується на визначенні цілей системи, функціональних та нефункціональних вимог, виборі технологічного стеку, розробці архітектури програмної системи, структури даних і моделей взаємодії між компонентами. Враховуючи специфіку документообігу, важливими чинниками проєктування стали: підтримка напівструктурованих даних, контроль доступу, швидкий пошук, масштабованість і простота розгортання.

До ключових функціональних вимог системи належать:завантаження документів різних форматів (.pdf, .docx, .xlsx);додавання метаданих (тема, тип, дата, відповідальний, теги);категоризація документів за відділами, проєктами, тегами;повнотекстовий пошук і фільтрація;розмежування прав доступу за ролями (читання, редагування, завантаження);зберігання історії змін і журналізація;резервне копіювання та відновлення.

Система спроєктована на основі трирівневої клієнт-серверної архітектури, де:Frontend (Vue.js) — інтерфейс користувача, що працює як SPA;Backend (Node.js + Express) — серверна логіка, авторизація, API, логіка доступу;СУБД (MongoDB) — зберігання документів, метаданих, прав доступу.

Архітектура передбачає використання REST API для взаємодії між клієнтом і сервером. Завдяки використанню NoSQL-сховища MongoDB забезпечується висока продуктивність при роботі з динамічними структурами документів, рисунок 1.

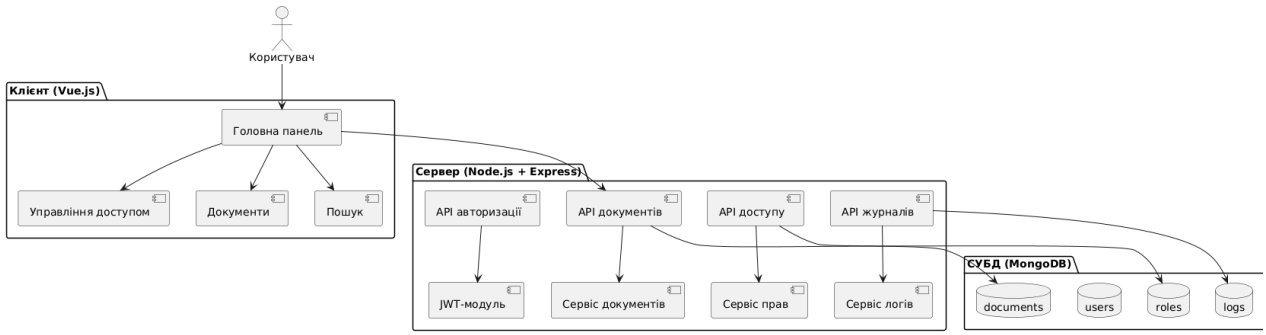


Рисунок 1–Архітектура системи

Реалізація інформаційної системи управління документообігом здійснювалася на основі трирівневої архітектури з чітким розділенням клієнтської частини - рисунок 2, серверної логіки та рівня зберігання даних. Такий підхід дозволив забезпечити модульність, масштабованість, спрощення тестування, гнучкість у підтримці та можливість подальшого розширення функціоналу.

Клієнтська частина (Vue.js). Інтерфейс користувача реалізований як односторінковий додаток (SPA) за допомогою фреймворку Vue.js 3. Ключові особливості: адаптивність (TailwindCSS); компонентна структура; використання VueRouter для маршрутизації;Axios для асинхронної взаємодії з API;JWT-авторизація.

Основні компоненти:Login.vue, Register.vue — сторінки входу та реєстрації;DocumentList.vue — таблиця документів з фільтрацією та тегами;UploadForm.vue — завантаження нового документа;AccessManager.vue — керування правами доступу;Profile.vue — особисті дані користувача.

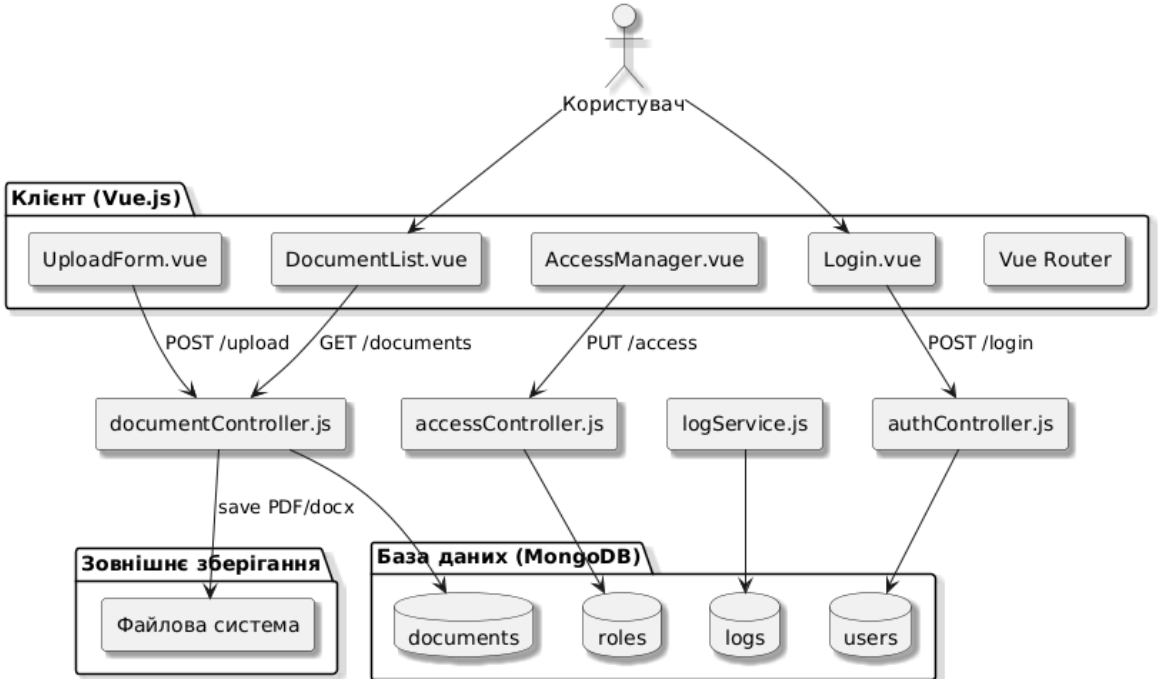


Рисунок 2–Реалізація системи

Висновок

Програмна реалізація системи забезпечує виконання всіх заявлених функцій із високим ступенем гнучкості та розширюваності. Поєднання Vue.js, Express.js і MongoDB дозволило реалізувати сучасну та продуктивну платформу для зберігання, обробки й контролю доступу до документів. Модульна структура коду спрощує обслуговування та тестування, а реалізація ролей і журналу дій забезпечує відповідність стандартам інформаційної безпеки.

Список використаних джерел

1. Sommerville, I. Software Engineering. 10th ed. Boston: Pearson, 2016. 792 p.
2. Carvalho, I., Sá, F., &Bernardino, J. (2023). Performance Evaluation of NoSQL Document Databases: Couchbase, CouchDB, and MongoDB. Algorithms, 16(2), 78. https://doi.org/10.3390/a16020078

УДК 004.896.3

СИСТЕМА ДЛЯ ОРГАНІЗАЦІЇ КОМУНІКАЦІЙ В AR-СИСТЕМАХ

Леськів Д.О.[1)], **Олійник А.П.**[2)], **Ференс З.Р.**[3)], **Барановський С.В.**[4)]
Західноукраїнський національний університет
[1)]студент;[2)]аспірант;[3-4)]студент

І. Постановка проблеми

У статті розглянуто архітектуру та особливості реалізації системи для організації обміну повідомленнями в середовищі доповненої реальності (AR). Проаналізовано вимоги до взаємодії користувачів у багатокористувацьких AR-додатках, запропоновано модель обміну повідомленнями, що підтримує синхронну й асинхронну взаємодію. Представлено реалізацію прототипу системи з використанням Unity, PhotonEngine та WebSocket API [1].Сучасні AR-системи потребують не просто передачі тексту — вони потребують живого, контекстного та швидкого обміну. І проєктування таких систем — це не тільки технічний виклик, а й спроба створити нову мову взаємодії у змішаному цифрово-реальному світі. Можливо, вже дуже скоро ми будемо листуватися не в чатах, а в просторі навколо нас, де кожне повідомлення має свою координату, значення і візуальну присутність.

З розвитком технологій доповненої реальності (AugmentedReality, AR) зростає потреба у створенні ефективних систем комунікації, що забезпечують інтерактивну взаємодію користувачів у реальному часі. Однією з ключових функціональних складових сучасних AR-додатків є модуль обміну повідомленнями, який має враховувати особливості візуалізації, просторової прив'язки та продуктивності пристроїв[2].

ІІ. Мета роботи

Метою дослідження є розробка програмного модуля для обміну повідомленнями в системі доповненої реальності.

ІІІ. Проектування архітектури модуля

Доповнена реальність (AR) давно перестала бути лише цікавинкою для геймерів чи інженерів. Вона стає повсякденною технологією — в освіті, охороні здоров’я, бізнесі та культурі. Але коли кілька користувачів одночасно перебувають в AR-просторі, виникає питання: як вони можуть ефективно взаємодіяти одне з одним? І тут на перший план виходить система для обміну повідомленнями.

Уявіть собі, що студенти, перебуваючи в аудиторії з AR-окулярами, бачать не лише викладача, а й цифрові коментарі своїх колег, які з'являються у просторі біля відповідних об’єктів. Або технік, який обслуговує складну установку, отримує підказки від колег у режимі реального часу — просто в полі зору. Все це потребує точного і надійного обміну повідомленнями, який враховує специфіку доповненої реальності.

Існуючі AR-системи, як-от Microsoft HoloLens, MagicLeap або мобільні додатки на основі ARKit/ARCore, часто використовують стандартні платформи для передачі повідомлень, зокрема REST API або MQTT. Проте ці рішення недостатньо адаптовані для сценаріїв з високою динамікою, коли необхідно обмінюватися даними з низькою затримкою й з урахуванням просторової контекстності.

Запропонована система складається з таких основних компонентів, рисунок 1:

- Модуль ініціалізації сесії: встановлення з’єднання між клієнтами через сервер;
- Сервер обміну повідомленнями: реалізовано на базі Photon Server або WebSocket-сервера;
- Клієнтська частина: реалізована в Unity з використанням AR Foundation, інтегрує AR-фреймворки та мережевий стек;
- Система просторового контексту: прив'язує повідомлення до 3D-простору або до об'єктів реального середовища;
- Модуль черги повідомлень: підтримує асинхронний обмін та буферизацію.

З урахуванням поставлених вимог до інтерактивної взаємодії користувачів у середовищі доповненої реальності, програмна реалізація системи обміну повідомленнями була здійснена на основі ігрового рушія Unity із застосуванням модуля AR Foundation для кросплатформової підтримки ARKit (iOS) та ARCore (Android). Для забезпечення обміну повідомленнями в реальному часі використано два альтернативні технологічні підходи:

Photon PUN 2 (PhotonUnityNetworking) — високорівнева платформа для розробки багатокористувацьких додатків, яка надає серверну інфраструктуру у хмарі;

WebSocket (реалізований засобами бібліотеки WebSocketSharp для Unity) — низькорівневий протокол з підтримкою двосторонньої передачі даних із мінімальною затримкою.

Програмна структура клієнта передбачає реалізацію кількох функціональних модулів, що відповідають заобробку вхідних та вихідних повідомлень, просторову прив’язку повідомлень до 3D-об’єктів у AR-сцені, взаємодію з інтерфейсом користувача.

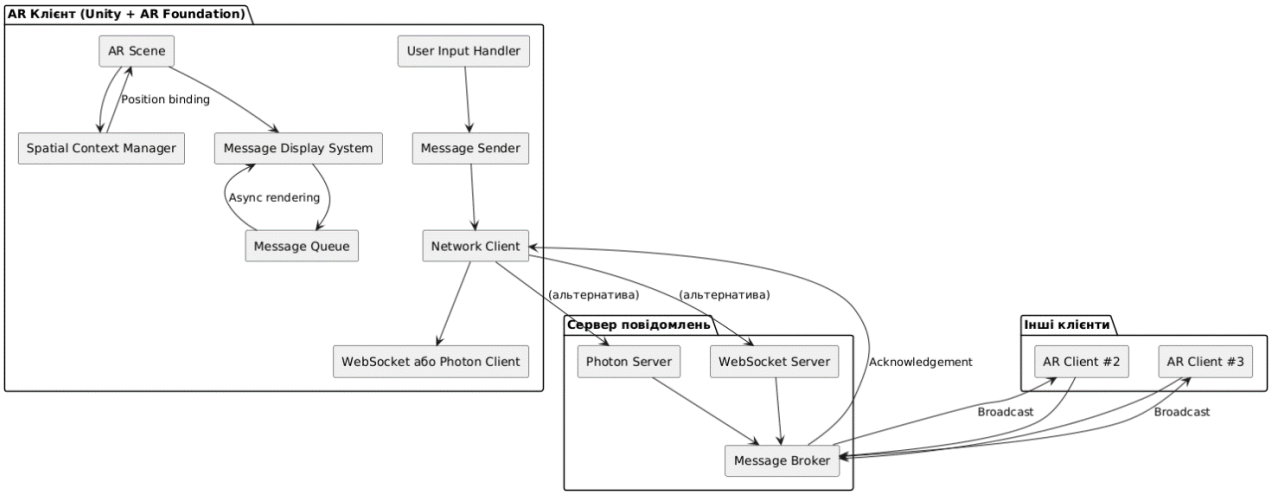


Рисунок 1–Архітектура програмного модуля

Розглянемо основні компоненти:MessageSender — формування та передача повідомлення у мережу;MessageReceiver — прийом, парсинг і візуалізація повідомлення;MessageData — структура даних для опису повідомлення;ARAnchorManager — модуль AR Foundation для створення «якорів» у просторі доповненої реальності;MessageQueue — буфер обробки повідомлень із підтримкою життєвого циклу.

Алгоритм передавання повідомлень. Передавання повідомлення в системі передбачає такі кроки:

1. Формування об'єкта повідомлення (включаючи текст, ідентифікатор користувача, координати розміщення в сцені, часову мітку).
2. Серіалізація об'єкта у формат JSON.
3. Виклик методу передачі через PhotonNetwork.RaiseEvent(...) або WebSocket.Send(...).
4. На стороні отримувача — десеріалізація, побудова об'єкта та візуалізація у відповідній позиції сцени.

Однією з ключових відмінностей реалізованої системи є прив’язка повідомлень до фізичного або семантичного простору сцени. Для цього застосовано механізм AR Anchor, що дозволяє зберегти позицію повідомлення незалежно від рухів користувача:

Кожне повідомлення обробляється на стороні клієнта, розміщується в 3D-просторі на віртуальній панелі в полі зору користувача, з прив'язкою до певної точки реального простору (наприклад, GPS-координати або imageanchor). Було проведено тестування прототипу на трьох пристроях (Android з ARCore, iOS з ARKit, HoloLens 2). Середній час доставки повідомлення — 85 мс при використанні Photon, 140 мс — WebSocket. Система забезпечує збереження контексту при втраті з'єднання та повторну синхронізацію після відновлення.

Висновок

Реалізована система обміну повідомленнями в AR забезпечує контекстну, просторово обґрунтовану, низькозатратну взаємодію користувачів у режимі реального часу. Програмна архітектура побудована на модульних принципах, що дає змогу легко адаптувати систему до різних типів пристроїв та сценаріїв використання — від мобільних освітніх додатків до промислових AR-рішень.

Список використаних джерел

1. Azuma, R. T. (1997). A Survey of Augmented Reality. Presence: Teleoperators and Virtual Environments, 6(4), 355–385.
2. Billinghurst, M., Clark, ., &Lee, G. (2015). A Survey of Augmented Reality. Foundations and Trends in Human–Computer Interaction, 8(2–3), 73–272.

Наукове видання

Комп’ютерні інформаційні технології

Матеріали
весняної школи-семінару молодих вчених і студентів СІТ’2025

Відповідальний за випуск:
Пукас А.В., д.т.н., професор,
завідувач кафедри комп’ютерних наук
Західноукраїнського національного університету

Підписано до друку 29.04.2025р.
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