МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп’ютерної інженерії

Дудас Владислав Сергійович

Програмна система оповіщення студентів кафедри комп’ютерної інженерії на основі технології Push API / Notification system for students of the Department of Computer Engineering based on Push API technology

спеціальність: 123 – Комп’ютерна інженерія
освітньо-професійна програма – Комп’ютерна інженерія

Кваліфікаційна робота

Виконав: студент групи КІ-41
Дудас Владислав Сергійович

Науковий Керівник
к.т.н., Піцун О. Й.

ТЕРНОПІЛЬ-2023

РЕЗЮМЕ

Кваліфікаційна робота на тему «Програмна система оповіщення студентів кафедри комп'ютерної інженерії на основі технології Push API» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня «бакалавр» містить 104 сторінки пояснюючої записки, 21 рисунок, 2 таблиці, 9 додатків. Обсяг графічного матеріалу 2 аркуші формату А3.

Метою кваліфікаційної роботи є розробка програмного засобу для оповіщення студентів кафедри комп'ютерної інженерії на основі технології Push API.

У роботі проведено аналіз сучасних технологій оповіщення та вибрано технологію Push API як найбільш відповідну для потреб системи. Запропонована система розроблена на основі монолітної архітектури та розділена на серверну та клієнтську частини. Серверна частина відповідає за збереження та керування повідомленнями, а клієнтська частина надає інтерфейс для студентів, де вони можуть переглядати стан повітряної тривоги та провести тестування системи сповіщень.

Розроблена система дозволяє студентам отримувати миттєві повідомлення про початок повітряної тривоги у регіоні навчального закладу. Це спрощує процес комунікації та забезпечує швидку інформаційну взаємодію між кафедрою та студентами. Важливою особливістю системи є можливість увімкнення тестової повітряної тривоги та тестування роботи системи оповіщень.

У результаті тестування системи було підтверджено її ефективність та функціональність. Система дозволяє збільшити ефективність комунікації між кафедрою та студентами, спрощує поширення важливої інформації та покращує загальний досвід та безпеку під час навчання.

Ключові слова: ПРОГРАМНА СИСТЕМА ОПОВІЩЕНЬ, ПОВІТРЯНА ТРИВОГА, ОПОВІЩЕННЯ ПРО ПОВІТРЯНУ ТРИВОГУ, ТЕХНОЛОГІЯ PUSH API.

RESUME

The qualification work on the topic "Notification system for students of the Department of Computer Engineering based on Push API technology" from specialty 123 "Computer engineering" of the bachelor's degree contains 104 pages of an explanatory note, 21 images, 2 tables, 9 appendices. The amount of graphic material is 2 sheets of A3 format.

The purpose of the qualification work is to develop a software tool for notifying students of the Department of Computer Engineering based on the Push API technology.

The paper analyzes modern notification technologies and selects the Push API technology as the most suitable for the needs of the system. The proposed system is developed on the basis of a monolithic architecture and is divided into server and client parts. The server side is responsible for storing and managing messages, while the client side provides an interface for students to view the air alert status and test the notification system.

The developed system allows students to receive instant messages about the beginning of an air alarm in the region of the educational institution. This simplifies the communication process and ensures fast information interaction between the department and students. An important feature of the system is the possibility of turning on a test air alarm and testing the operation of the warning system.

As a result of system testing, its efficiency and functionality were confirmed. The system makes it possible to increase the efficiency of communication between the department and students, simplifies the distribution of important information and improves the overall experience and safety during education.

Keywords: ALERTING SOFTWARE, NOTIFICATION SYSTEM, AIR ALERTS, PUSH API TECHNOLOGY.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

API	–	Application Programming Interface
ДСНС	–	Державна служба з надзвичайних ситуацій
УКМЦ	–	Український кризовий медіацентр
UICDS	–	Unified Incident Command and Decision Support
FEMA	–	Federal Emergency Management Agency
IRIS	–	International Resources Inventory System
SoKNOS	–	Service-Oriented ArchiteCtures Supporting Networks of Public Security
SMART-C	–	Social media alert and response to threats to citizens
ЗМІ	–	Засоби масової інформації
EDXL-RM	–	Emergency data exchange language resource messaging
HTML	–	HyperText Markup Language
CSS	–	Cascading Style Sheets
PHP	–	Personal Home Page Tools
SQL	–	Structured query language
AJAX	–	Asynchronous JavaScript And XML
DOM	–	Document Object Model
BP	–	Base Prototypes
ASP	–	Active Server Pages
JSP	–	Jakarta Server Pages
XML	–	Extensible Markup Language
JSF	–	JavaServer Faces
JSP	–	JavaServer Pages
JPA	–	Java Persistence API

HTTP	–	HyperText Transfer Protocol
HTTPS	–	HyperText Transfer Protocol Secure
IDEA	–	Integrated Development Environment Application
JVM	–	Java Virtual Machine
ОС	–	Операційна система
SMS	–	Short Message Service
JSON	–	JavaScript Object Notation
LL	–	Left-to-left
LR	–	Left-to-right
CSV	–	Comma-separated values
REST	–	Representational State Transfer
IoC	–	Inversion of Control
MVC	–	Model-View-Controller
MVVM	–	Model-View-ViewModel
URL	–	Uniform Resource Locator
HDFS	–	Hadoop Distributed File System
JDBC	–	Java Database Connectivity
S3	–	Simple Storage Service
UML	–	Unified Modeling Language
PNG	–	Portable Network Graphics
JPEG	–	Joint Photographic Experts Group
POM	–	Project Object Model
ЦП	–	Центральний процесор
ОЗП	–	Оперативна пам'ять
JDK	–	Java Development Kit
ПЗ	–	Програмне забезпечення
ПК	–	Персональний комп’ютер

ВСТУП

В XXI столітті, інтернет став невід'ємною частиною життя людей. Щодня з'являється все більше та більше інформації про світ, буденне життя та нові професії. Веб-програмування стала одною з таких.

Коли з'явились перші персональні комп'ютери, вони були дуже складні та масивні. Не кожна людина могла собі дозволити таку розкіш. Проте, з плином часу та завдяки геніям, які впевнено рухали технологічний прогрес, стало можливо зменшити розмір і саму концепцію персонального комп'ютера. Раніше це були масивні та досить одно направлені машини, а зараз це багатофункціональні робочі станції з високою швидкодією та продуктивністю для будь якої задачі, на яку вже не впливає розмір самого гаджета. Саме це вплинуло на лояльність молодих спеціалістів до нових технологій, завдяки яким у майбутньому технологічний прогрес буде і далі йти тільки вперед і вгору.

Саме в цей період зайнятості з'явилась розробники, адже необхідно було створювати нові програми для нових і різноманітних систем. На даний час, ця професія має дуже великий попит, а талановиті програмісти в даний момент працюють у кращих компаніях світу. А завдяки зручним інструментам розробки, створювати все нові і нові програми тепер не є складною задачею.

Розвиток технологій не стоїть на місці і з кожним днем будь-яка сфера піддається автоматизації та спрощенню. Не є винятком і модернізація та спрощення роботи системи веб-сповіщень.

Програмістам доволі важко знайти місце роботи без досвіду та необхідних навичок. Саме для цього потрібна кваліфікаційна робота, яка буде слугувати молодому спеціалісту в якості портфоліо, таким чином роботодавець зможе побачити рівень та навички програміста.

Практична цінність полягає у розробці програмного засобу, який зміг би оповіщати студентів кафедри комп'ютерної інженерії про повітряну тривогу у

регіоні навчального закладу та збільшити за рахунок цього безпеку роботи студентів та викладачів кафедри та факультету.

Робота є актуальною, оскільки існує потреба в ефективній та швидкій системі оповіщення студентів про важливі події на кафедрі комп'ютерної інженерії. Використання технології Push API дозволить надсилати сповіщення безпосередньо на пристрої, з якими працюють студенти, що забезпечить їм швидкий доступ до актуальної, критично важливої інформації.

Метою даної кваліфікаційної роботи є розробка програмної системи оповіщення студентів кафедри комп'ютерної інженерії на основі технології Push API. Головним завданням є створення функціонального та надійного рішення, яке надасть студентам необхідну інформацію та сповіщатиме їх про зміни станів повітряної тривоги.

Таким чином, розробка програмної системи оповіщення студентів на основі технології Push API має практичну цінність для кафедри комп'ютерної інженерії, забезпечуючи швидке та зручне сповіщення студентів про події та новини.

Завдання, що потрібно виконати для досягнення мети розробки, включають:

- Аналіз потреб та вимог студентів та кафедри комп'ютерної інженерії щодо системи оповіщення.
- Аналіз алгоритмів парсингу;
- Аналіз алгоритмів сповіщень;
- Аналіз архітектури системи;
- Проектування архітектури системи;
- Розробка серверної частини;
- Розробка клієнтської(веб) частини;
- Аналіз та налаштування системи Push API;
- Тестування та валідація;
- Документування;

– Впровадження та підтримка.

Виконання цих завдань дозволить досягти мети розробки та створити програмну систему оповіщення студентів кафедри комп'ютерної інженерії, яка сприятиме покращенню комунікації та оперативно попереджуватиме студентів про зміну стану повітряної тривоги у області.

За результатами роботи опубліковано тези доповіді на VII науково-практичній конференції «Інтелектуальні комп’ютерні системи та мережі» [1]. Копії публікації наведено у додатку К.

АНАЛІЗ СИСТЕМИ ОНЛАЙН СПОВІЩЕННЯ ПРО ПОДІЇ

1.1 Алгоритми оповіщення про події

Аналіз здійснюється на основі вимог, що пред'являються до створюваної системи та з урахуванням особливостей предметної області. Результати повинні лягти в основу обґрунтування вибору платформи, технологій та середовища розробки програмного забезпечення. Для обґрунтування вибору застосовують методи прийняття рішень.

Існує безліч рішень та аналогів веб-ресурсів з даними про ракетну тривогу в Україні і за кордоном. Ці рішення мають на меті забезпечити актуальну та точну інформацію громадськості про ракетні тривоги та інші надзвичайні ситуації.

Один з найвідоміших веб-ресурсів з інформацією про ракетну тривогу в Україні - це сайт Державної служби з надзвичайних ситуацій України (ДСНС). Сайт ДСНС надає інформацію про різні типи надзвичайних ситуацій, в тому числі про ракетні тривоги, а також оновлення щодо поточних надзвичайних ситуацій. Сайт також надає контактну інформацію для ДСНС та інших служб надзвичайних ситуацій, а також поради щодо підготовки до надзвичайних ситуацій.

Іншим веб-ресурсом, який надає інформацію про ракетні тривоги в Україні, є сайт Українського кризового медіацентру (УКМЦ). Сайт УКМЦ надає новини та оновлення про різні аспекти конфлікту в Україні, включаючи ракетні тривоги та іншу військову діяльність. Сайт також надає аналіз та коментарі щодо конфлікту від експертів та журналістів.

Серед закордонних практик існують деякі сервісно-орієнтовані платформи та системи для сумісного обміну інформацією для управління надзвичайними ситуаціями та планування реагування, зокрема XchangeCore11, раніше відома як Уніфікована система підтримки прийняття рішень щодо інцидентів (UICDS) [17], [18], Система інвентаризації ресурсів

FEMA (IRIS) [19], Сервісно-орієнтована архітектура підтримки мереж громадської безпеки (SoKNOS) [20] та соціальні оповіщення ЗМІ та реагування на загрози громадянам (Smart-C) [21].

Ще одним цікавим прикладом є, стандарт EDXL для управління ресурсами (EDXL-RM) [12], якщо запит на певний ресурс уже було розміщено, але в міру розвитку інциденту ми розуміємо, що запитуваний ресурс більше не потрібен і потрібен інший ресурс , попередній запит можна скасувати, надіславши повідомлення про відкликання запиту, яке стає доступним через API служби.

Окрім цих офіційних веб-ресурсів, також існують кілька соціальних медіа-аккаунтів та онлайн-спільнот, які надають інформацію про ракетні тривоги в Україні. Одна з таких спільнот - "Український військовий портал" на Facebook, який надає новини та оновлення про військову діяльність в Україні, включаючи ракетні тривоги. Також є кілька облікових записів у Twitter, які надають оновлення про ракетні тривоги та інші надзвичайні ситуації.

Було досліджено кілька головних українських веб ресурсів, але з різним візуальним поданням, дизайном та структурою, а саме:

- https://vadimklimenko.com/map/
- https://alerts.in.ua
- https://map.ukrainealarm.com

Одним з основних переваг веб-ресурсів з інформацією про ракетну тривогу в Україні є їх швидкість та доступність. Інформація на цих ресурсах оновлюється зазвичай в режимі реального часу, що дозволяє громадськості дізнаватися про ракетну тривогу або іншу надзвичайну ситуацію майже миттєво. Крім того, ці ресурси є безкоштовними та доступними для використання всім, хто має доступ до Інтернету.

Проте, існують також деякі недоліки у веб-ресурсах з інформацією про ракетну тривогу в Україні. Одним з найбільших недоліків є можлива

неактуальність інформації на деяких ресурсах, оскільки оновлення на них можуть бути затримані або не відображати повну картину ситуації. Крім того, немає гарантії, що інформація, яка надається на цих веб-ресурсах, є повною та точною. Також інформація на цих ресурсах може бути складною для розуміння для тих, хто не має достатньої освіти або знань у цій галузі.

1.2 Технології розробки веб-додатків

Веб-додатки (web applications) є програмними застосунками, які використовують веб-браузер як інтерфейс для взаємодії з користувачем. Вони доступні через Інтернет і працюють на веб-сервері. Веб-додатки дозволяють користувачам виконувати різноманітні завдання та функції, такі як робота з базою даних, обробка форм, відображення інформації, спілкування з іншими користувачами і багато іншого.

Веб-додатки зазвичай розробляються з використанням веб-технологій, таких як HTML, CSS, JavaScript та серверних мов програмування, таких як Java, Python, PHP, Ruby тощо. Вони можуть бути статичними, коли контент на сторінках залишається незмінним, або динамічними, коли вони взаємодіють з користувачем і можуть змінювати вміст на основі його дій.

Розробка веб-додатків є процесом створення програмного забезпечення, яке працює через веб-браузер. Цей тип додатків надає можливість користувачам взаємодіяти з інформацією, використовуючи веб-інтерфейс.

Основні складові розробки веб-додатків включають:

1. Фронтенд (клієнтська частина): Розробка фронтенду займається створенням користувацького інтерфейсу, з яким взаємодіє користувач. Вона включає в себе розробку HTML, CSS і JavaScript коду, а також використання фреймворків та бібліотек для полегшення розробки і покращення користувацького досвіду.

2. Бекенд (серверна частина): Розробка бекенду фокусується на реалізації логіки додатку, обробці запитів користувачів та взаємодії з базою

даних. Для цього використовуються мови програмування, такі як Java, Python, PHP, Ruby, C# та інші. Також використовуються фреймворки та інструменти для полегшення розробки, роботи з базами даних та забезпечення безпеки додатку.

3. База даних: Веб-додатки часто використовують бази даних для зберігання і організації інформації. Розробка веб-додатків включає створення та керування базами даних, проектування таблиць, створення запитів та забезпечення ефективного зберігання та отримання даних.

4. Тестування: Важливим етапом в розробці веб-додатків є тестування. Воно включає написання тестів для перевірки правильності роботи додатку, виявлення та виправлення помилок та забезпечення якості програмного забезпечення.

5. Розгортання та підтримка: Після успішного розроблення додатку його необхідно розгорнути на сервері, щоб зробити його доступним для користувачів. Крім того, після розгортання потрібно забезпечити підтримку та обслуговування додатку, виявляти та виправляти помилки, вдосконалювати функціональність тощо.

Розробка веб-додатків вимагає знань мов програмування, фреймворків, баз даних, принципів веб-розробки та досвіду роботи з різними інструментами. Також важливо бути в курсі сучасних тенденцій і нововведень в цій сфері. Розробка веб-додатків включає постійне вдосконалення і пристосування до змінних вимог та технологій, що забезпечує постійний розвиток і еволюцію цієї галузі.

Існує безліч технологій, які використовуються для розробки веб-додатків. Основні з них:

- HTML: мова розмітки, яка використовується для створення структури веб-сторінки.
- CSS: мова опису стилів, яка використовується для стилізації веб-сторінок.

- Java: універсальна об'єктно-орієнтована мова програмування, яка використовується для розробки веб-додатків та багатьох інших видів програмного забезпечення.
- JavaScript: мова програмування, яка використовується для створення динамічних елементів на веб-сторінках, таких як анімація, перехід на іншу сторінку без перезавантаження сторінки та багато іншого.
- PHP: мова програмування на серверному боці, яка використовується для створення скриптів та обробки даних на сервері.
- SQL: мова запитів до баз даних, яка використовується для зберігання та керування даними, що використовуються в веб-додатках.
- AJAX: технологія, яка дозволяє здійснювати асинхронні запити до сервера без перезавантаження сторінки.
- jQuery: бібліотека JavaScript, яка дозволяє забезпечувати більш зручну роботу з DOM-деревом та виконувати інші завдання.
- Bootstrap: фреймворк CSS, який надає готові компоненти та стилі для швидкої розробки веб-сторінок.

Кожна з цих технологій має свою особливість, позитивну та негативну сторони. Крім того, існуючі підходи до адаптивної композиції процесу [14], [15], [16] обмежуються структурованими BP, які використовують міркування на основі правила події-умови-дії для адаптації процесу.

Веб-додатки використовуються в різних сферах, включаючи електронну комерцію, соціальні мережі, онлайн-банкінг, освіту, здоров'я, подорожі, розваги та багато інших. Вони надають зручність користувачам, оскільки їх можна відкрити на будь-якому пристрої з доступом до Інтернету та використовувати безпосередньо в браузері. Розглянемо їх схематичне порівняння детально наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння мов програмування для розробки веб-додатків

Мова програмування	**Швидкодія**	**Простота в освоєнні**	**Оптимізація**	**Універсальність**	**Популярність**
Java	*Висока*	*Середня*	*Висока*	*Висока*	*Дуже популярна*
Python	*Середня*	*Висока*	*Середня*	*Висока*	*Дуже популярна*
JavaScript	*Середня*	*Висока*	*Середня*	*Висока*	*Дуже популярна*
PHP	*Середня*	*Висока*	*Середня*	*Висока*	*Дуже популярна*
Ruby	*Середня*	*Середня*	*Середня*	*Середня*	*Популярна*
TypeScript	*Висока*	*Середня*	*Висока*	*Висока*	*Популярна*

Ця таблиця надає загальний порівняльний огляд кількох мов програмування веб-додатків за вказаними критеріями. Зверніть увагу, що оцінки можуть бути умовними, оскільки кожна мова має свої особливості та відповідає різним вимогам проектів. Рішення щодо вибору мови

програмування повинно ґрунтуватися на конкретних потребах, вміннях команди та інших факторах, що впливають на проект.

Розробка та еволюція веб-додатків мають глибокі корені, що починаються зі становлення Інтернету та появи перших веб-сторінок. Ось короткий огляд історії розробки та еволюції веб-додатків:

- Перший веб-сервер і веб-сторінка: У 1989 році Тім Бернерс-Лі, британський фізик і інформаційний науковець, створив перший веб-сервер та написав першу веб-сторінку. Це був початок Інтернету та веб-розробки.
- Статичні веб-сторінки: На ранній стадії веб-розробки, веб-сторінки були в основному статичними, тобто їх вміст не змінювався після завантаження. Розробники використовували HTML для створення та форматування сторінок.
- Введення скриптових мов: Впровадження скриптових мов, таких як JavaScript, в кінці 1990-х років, дозволило розробникам додавати динамічний функціонал до веб-сторінок. Це відкрило шлях до інтерактивних елементів, валідації форм, асинхронних запитів та інших динамічних можливостей.
- Розробка серверних скриптових мов: З'явилися серверні скриптові мови, такі як PHP, ASP, JSP, що дозволяли генерувати веб-сторінки на сервері перед їх відправленням на клієнтську сторону. Це дозволило розробникам створювати складні динамічні веб-додатки.
- Розробка веб-фреймворків: З'явлення веб-фреймворків, таких як Ruby on Rails, Django, Laravel і Node.js, спростило розробку веб-додатків. Вони надали розробникам готову інфраструктуру та набір інструментів для розробки веб-додатків швидше та ефективніше.
- Розширення можливостей веб-додатків: З популярністю AJAX (асинхронний JavaScript і XML), веб-додатки стали більш динамічними та реактивними без необхідності повного перезавантаження сторінки.

Впровадження HTML5 та CSS3 додало нові можливості для відео, аудіо, графіки, анімації та багато іншого.

- Розвиток мобільних додатків: З появою смартфонів та планшетів з'явилася потреба в мобільних веб-додатках. Розробники почали створювати адаптивні та мобільні версії своїх веб-додатків або використовувати фреймворки для гібридної або нативної розробки мобільних додатків.
- Сучасні тенденції: Сьогодні веб-додатки стають все більш складними та інтерактивними. Використання розподілених систем, хмарних технологій, мікросервісної архітектури, розширених можливостей JavaScript-фреймворків, швидкості та безпеки є актуальними тенденціями.

Це лише загальний огляд історії розробки та еволюції веб-додатків. З розвитком технологій та зміною потреб користувачів, веб-додатки продовжують розвиватися, надаючи більше функціональних можливостей та поліпшуючи користувацький досвід.

Java є однією з найпопулярніших мов програмування для розробки веб-додатків, але не можна однозначно визначити "найкращу" мову програмування для цієї цілі, оскільки вибір залежить від конкретних потреб та вимог проекту. Однак, можемо порівняти можливості Java з деякими іншими популярними мовами програмування для веб-розробки:

1. Java vs. Python: Обидві мови популярні для веб-розробки. Java має широке сприйняття в корпоративному середовищі, має потужну екосистему та розширені можливості для розробки масштабних додатків. Python, з іншого боку, відомий своєю простотою, читабельністю та швидким розвитком. Python також має багатий стек фреймворків для веб-розробки, таких як Django та Flask.

2. Java vs. JavaScript: Java та JavaScript є двома різними мовами програмування. Java використовується для серверної розробки та розробки веб-додатків з використанням фреймворків, таких як Spring і JavaServer Faces (JSF). JavaScript використовується для клієнтської розробки та має сильну

підтримку для веб-інтерфейсів, маніпуляції DOM та взаємодії з сервером за допомогою AJAX.

3. Java vs. PHP: Java та PHP є популярними мовами для серверної розробки веб-додатків. Java має сильну типізацію, багатий набір функцій та хорошу масштабованість, що робить її відповідною для великих та складних проектів. PHP відомий своєю простотою та легкістю в освоєнні, є популярним в середовищі веб-розробки та має широку підтримку готових фреймворків, таких як Laravel та Symfony.

Вибір мови програмування для розробки веб-додатків залежить від багатьох факторів, таких як особисті вподобання, потреби проекту, наявність ресурсів та специфікації проекту. Кожна мова має свої переваги та недоліки, і кращим варіантом буде той, який найкраще відповідає потребам конкретного проекту.

Проаналізувавши існуючі технології створення веб-додатків було обрано фреймворк Spark для мови програмування Java. Як було зазначено вище, Java - це об'єктно-орієнтована мова програмування, яка використовується для розробки веб-додатків та багатьох інших видів програмного забезпечення.

Однією з переваг Java є те, що вона є платформо-незалежною мовою, тобто програми, написані на Java, можуть працювати на будь-якій операційній системі, яка має відповідний інтерпретатор Java.

Для розробки веб-додатків на Java використовуються такі технології, як Servlets, JSP (Java Server Pages), JPA (Java Persistence API) та Spring Framework. Вони дозволяють розробляти веб-додатки з високою продуктивністю, забезпечуючи зручний API для роботи з HTTP запитами та базами даних.

Java також має широкий спектр інструментів для розробки додатків, таких як Eclipse, NetBeans та IntelliJ IDEA.

IntelliJ IDEA , для прикладу, є інтегрованим середовищем розробки (Integrated Development Environment або IDE) для програмування,

розробленою компанією JetBrains. Вона призначена для підтримки різних мов програмування, зокрема Java, Kotlin, Scala, Groovy та багатьох інших.

IntelliJ IDEA надає розширений набір інструментів для розробки програмного забезпечення, включаючи редактор коду з підсвічуванням синтаксису, автодоповненням, рефакторингом, аналізом коду, налалагодженням і багатьма іншими корисними функціями.

Це популярне середовище розробки серед професіоналів програмування, яке надає зручний та продуктивний спосіб розробки програмного забезпечення. IntelliJ IDEA також підтримує розширення плагінами, що дозволяє розширити його функціональність для задоволення конкретних потреб розробника[10].

Крім того, IntelliJ IDEA пропонує інтеграцію з різними засобами контролю версій, системами збирання проектів та іншими інструментами розробки, що сприяє покращенню ефективності роботи розробника.

Сама ж Java - це потужна та популярна об'єктно-орієнтована мова програмування, розроблена компанією Sun Microsystems (зараз є власністю компанії Oracle[9]). Вона була випущена в 1995 році і з тих пір стала однією з найпоширеніших мов програмування у світі.

Основні особливості мови Java:

1. Об'єктно-орієнтованість: Java побудована на концепції об'єктно-орієнтованого програмування, що дозволяє створювати модульні, універсальні та підтримувані програми.

2. Платформонезалежність: Java працює на основі віртуальної машини Java (Java Virtual Machine, JVM), що дозволяє виконувати Java-програми на різних операційних системах без необхідності перекомпіляції. Це робить Java платформонезалежною.

3. Безпека: Java має вбудовану систему безпеки, що дозволяє контролювати доступ до ресурсів, управляти пам'яттю та запобігати вразливостям.

4. Велика бібліотека: Java поставляється з великою стандартною бібліотекою, яка містить готові класи та методи для різних завдань, таких як робота з рядками, мережеве програмування, обробка введення-виведення, робота з базами даних та багато іншого. Це спрощує розробку програм і зменшує кількість написаного коду.

5. Багатопоточність: Java надає потужні засоби для роботи з багатопоточними програмами, дозволяючи одночасно виконувати різні частини програми.

6. Розширюваність: Java підтримує розширення функціональності через використання плагінів та бібліотек сторонніх розробників.

Java знайшла застосування в багатьох сферах програмування, таких як розробка веб-додатків, мобільних додатків, вбудованих систем, наукових досліджень, фінансових програм, ігор та багато іншого. Велика спільнота розробників та багато доступних ресурсів роблять Java популярним вибором для багатьох проектів. Зараз Java має велику активну спільноту розробників, що дозволяє отримувати широку підтримку, документацію та розвиток мови та її інструментів, фреймворків та плагінів[11].

1.3 Аналіз архітектури веб-додатків

Побудова архітектури додатку — це комплекс заходів, який спрямований на те, щоб чітко визначити, як буде побудована система.

Важливо розуміти те, що немає ідеального рішення при визначенні принципу проектування архітектури, ми не можемо сказати, що оцей архітектурний патерн підходить під будь-яку задачу та використовувати слід лише його. Все залежить від конкретного проєкту та його завдань. Але ви можете побудувати усе, що завгодно, на будь-якій архітектурі, але чи буде це виправданим? Навряд чи.

Отже, що нам потрібно для визначення повної архітектури проєкту:

- Технічне завдання

- Архітектор, або людина з багатим досвідом розробки
- Розуміння дедлайнів
- Розуміння життєвого циклу проєкту
- Розуміння навантаження

Як бачимо, це дуже багато інформації, яку треба проаналізувати для того, щоб визначитись з повною архітектурою проєкту. Але для верхньорівневої архітектури іноді нам вистачить лише ідеї проєкту.

Загалом відокремлюють три типи верхньорівневої архітектури веб додатків, це:

- Моноліт
- Мікросервіси
- Серверлес

Моноліт

Моноліт — це архітектурне рішення, у якому усі компоненти та модулі тісно пов’язані між собою, та залежать один від одного, докладніше проілюстровано на рисунку 1.1.

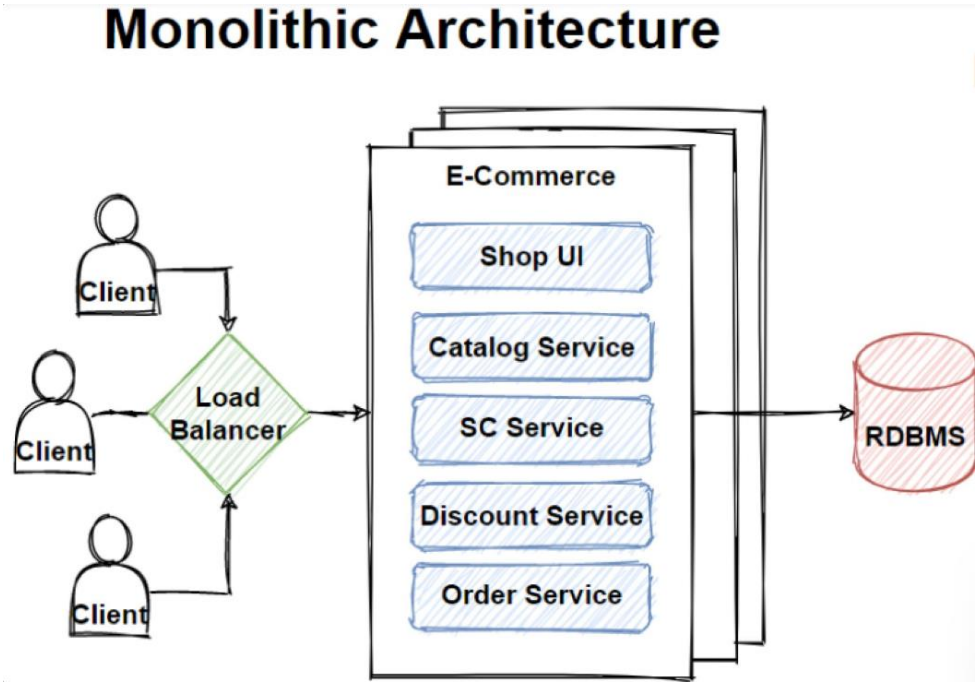

Рисунок 1.1 – Монолітна архітектура

Загалом вважається, що це застаріле архітектурне рішення, але я з цим не погоджуюсь, бо у такого рішення є свої переваги, і воно дуже гарно підходить при побудові деяких проєктів.

Переваги:

- Простота розгортання. Моноліти дуже швидко і відносно просто розгортати, завдяки тому що у моноліту, зазвичай, єдина точка входу
- Розробка. Розробка моноліту зазвичай дуже швидка, бо усі компоненти та модулі знаходяться в одній кодовій базі та завжди під рукою
- Налагодження. Налагодження моноліту дуже спрощена за рахунок того, що усе поряд, і є можливість відстежити усю ланку виконання коду

Недоліки:

- Масштабування. Моноліти масштабуються лише повністю, тобто якщо навантаження зростає лише на один модуль, ми не можемо масштабувати лише цей модуль, ми маємо масштабувати весь моноліт
- Надійність. Якщо моноліт виходить з ладу, то виходить весь цілком
- Зміна та оновлення технологій. У моноліті це майже неможливо
- Недостатня гнучкість. Моноліти негнучкі, тобто зміна одного модуля у моноліті майже завжди впливатиме на інший модуль

Мікросервіси

Мікросервіси — це архітектурне рішення, яке базується на розподілі модулів на окремі системи, які спілкуються між собою за допомогою повідомлень.

Уся система — це набір маленьких систем, які пов’язані між собою, докладніше проілюстровано на зображенні 1.2.

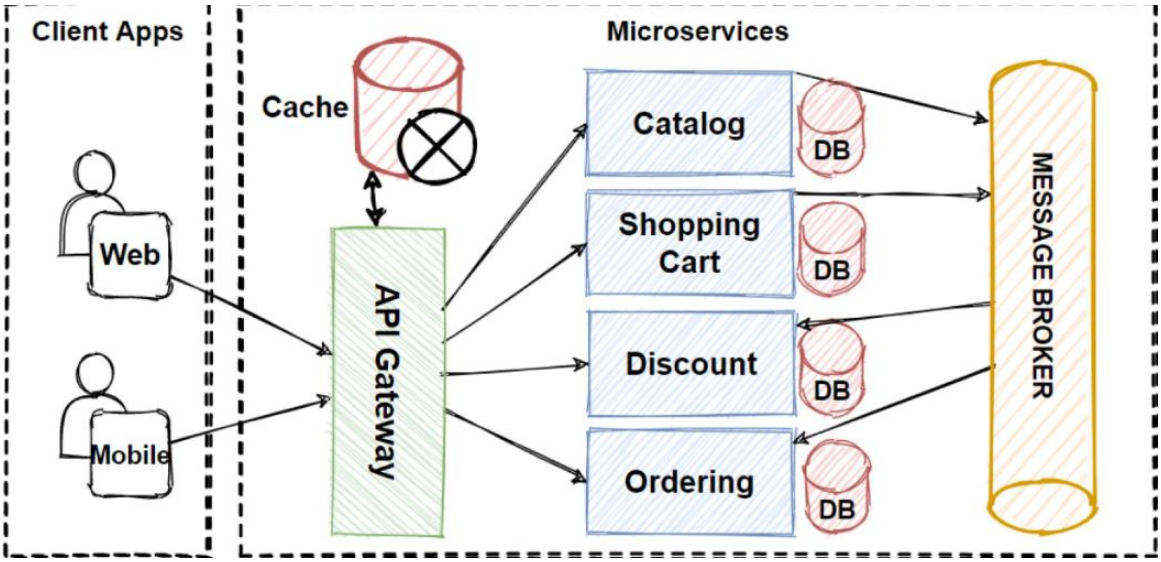

Рисунок 1.2 – Мікросервісна архітектура

Наразі це дуже модна і популярна архітектура, але у неї є свої недоліки та, на мій погляд, вона зовсім не підходить для маленьких та середніх проєктів.

Переваги:

- Гнучкість. Мікросервіси дуже гнучкі за рахунок того, що кожен сервіс є самостійною системою, та тому зміни у ній можуть вплинути лише на неї
- Масштабування. На відміну від моноліту, ми можемо масштабувати лише певну частину системи, тому що вона самостійна
- Гнучкість технологій. У мікросервісах кожна з підсистем може бути реалізована будь-якою мовою програмування та за допомогою будь-яких технологій
- Надійність. Зазвичай вихід із ладу однієї з підсистем не ламає усю систему загалом

Недоліки:

- Процес розробки. Мікросервіси дуже непросто розробляти, тому що треба робити декілька підсистем та налагодити взаємодію між ними

- Комунікація команд. Дуже часто буває так, що одна команда розробляє одну підсистему, тож треба налагодити комунікації між командами, щоб налагодити взаємодію між підсистемами
- Налагодження. Мікросервіси дуже складно налагоджувати, тому що треба знайти, який сервіс зламався, та чому
- Розгортання. Початкове розгортання дуже непросте, та додавання нових сервісів потребують налагодження ключових частин проєкту

Серверлесс

Серверлесс — це архітектурне рішення, яке фокусується на розробці, а не на розгортанні та взаємодії між сервісами.

Серверлесс — це альтернатива мікросервісами, яка автоматизує усе розгортання завдяки хмарним технологіям. Докладніше проілюстровано на зображенні 1.3. З назви можна подумати, що тут відсутня серверна частина, але це не так, тут відсутня робота з сервером як з середовищем.

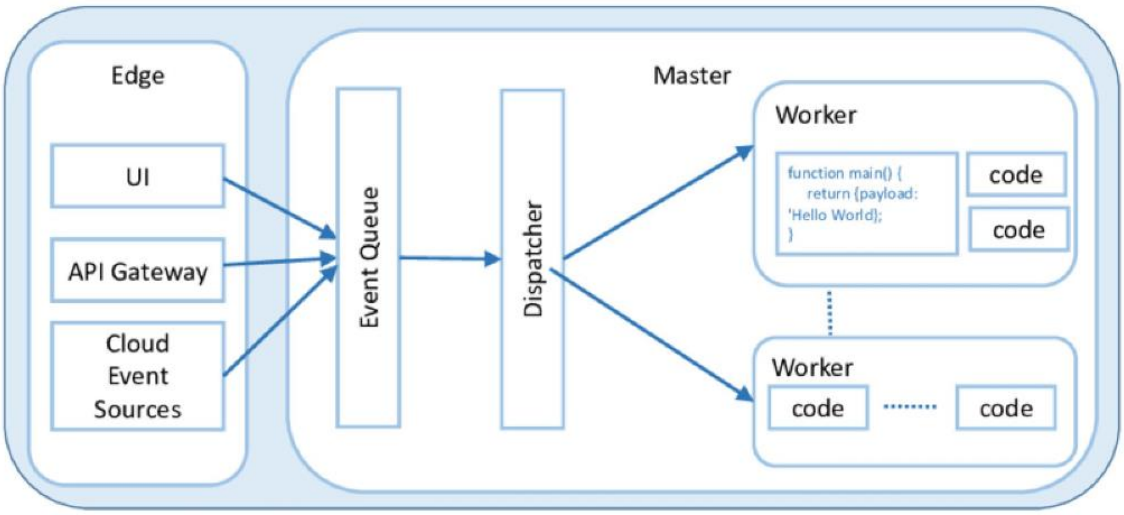

Рисунок 1.3 – Серверлесс архітектура

Переваги:

- Гнучкість. Серверлесс дуже гнучкий за рахунок відокремленості одного модулю від іншого
- Абстракція від ОС. Хмара сама вирішить, яка ОС потрібна, та як її треба налаштувати
- Легкий поріг входу. Зазвичай серверлесс це дуже простий, відокремлений код, тож розібратися з проєктом нескладно
- Надійність. При правильній побудові проєкту надійність також сама як і у мікросервісів

Недоліки:

- Гнучкість. Так, це також мінус, бо у розробника обмежений вплив на масштабування, усе вирішує хмара
- Налагодження. Так само як у мікросервісах налагодження ускладнена взаємодією між компонентами
- Vendor Lock. Кожна хмара працює за власними правилами, тому переїзд з однієї хмари на іншу майже неможливий
- Cascade Failur. При неправильній побудові проєкту компоненти можуть дуже сильно впливати на інщі компоненти, що призводить до падіння усього проєкту

1.4 Висновки та постановка задачі

В ході дослідження аналогічних веб-сервісів, аналізу їх сильних та слабких сторін стало можливим оцінити, які саме проблеми необхідно подолати в нашій системі.

Необхідно створити сервер з бекенд-частиною, на якому буде базуватись сайт з фронтенд частиною.

Згідно з розробленими функціональними та нефункціональними вимогами встановимо комплекс завдань, що має вирішити розроблюваний програмний модуль:

- надати можливість користувачу вільно переглядати дані про повітряні тривоги;
- надати можливість користувачу запустити тестове сповіщення для перевірки дієздатності веб-додатку;
- надати можливість користувачу зробити пробний запуск повітряної тривоги;
- надати користувачу застосунок який автоматизовано буде повідомляти про початок та закінчення повітряної тривоги.

Отже, розумівши усі переваги та недоліки тої чи іншої архітектури і технології розробки веб додатків, ми сміливо можемо обрати саме те поєднання, що відповідає темі поставленої задачі.

Ще раз варто наголосити, що будь-який проєкт можливо виконати з будь-якою архітектурою, але це може бути неефективним або у плані розробки, або у плані експлуатації проєкту.

2 АЛГОРИТМИ РОБОТИ ТА МОДУЛІ САЙТУ

2.1 Алгоритми оповіщення

Сповіщення (також відомі як повідомлення або повідомлення) є способом передачі інформації або повідомлення від одного джерела до одного або багатьох отримувачів.

Системи оповіщення мають довгу історію розвитку, починаючи з простих методів передачі повідомлень до сучасних технологій з використанням інтернету та мобільних пристроїв. Ось загальний огляд історії розвитку систем оповіщення:

1. Традиційні засоби оповіщення: Стародавні системи оповіщення включали в себе використання гучних сигналів, дзвонів, світлових сигналів, тривог та фізичного сповіщення. Ці методи використовувалися для передачі сигналів і повідомлень у разі надзвичайних ситуацій або важливих подій.

2. Телефонна система оповіщення: У 20-му столітті поширення телефонних мереж дало змогу використовувати телефони для широкого оповіщення. Системи автоматичних дзвінків та розсилки голосових повідомлень були розроблені для передачі масових повідомлень великій кількості людей.

3. Електронна пошта: З появою електронної пошти у 1970-х роках, стала можливою передача повідомлень на велику відстань швидко та зручно. Електронна пошта стала популярним засобом масової комунікації та оповіщення.

4. SMS-повідомлення: Зі зростанням мобільних телефонів у 1990-х роках, SMS-повідомлення стали популярним засобом миттєвого сповіщення. Вони дозволяють відправляти короткі текстові повідомлення на мобільні телефони.

5. Push-сповіщення: З'явлення смартфонів та розробка мобільних додатків призвело до використання технологій push-сповіщень. Це дозволяє

додаткам відправляти повідомлення користувачам навіть тоді, коли додаток неактивний. Push-сповіщення стали ефективним способом залучення уваги користувачів та надання їм актуальної інформації.

6. Веб-сповіщення: Останнім розвитком є веб-сповіщення, які використовують технологію PushAPI для надсилання сповіщень через веб-браузери. Вони дозволяють веб-додаткам надсилати повідомлення користувачам навіть під час відсутності веб-сторінки. Це стало потужним інструментом для взаємодії з користувачами в реальному часі.

Розвиток систем оповіщення продовжується, з постійними інноваціями та вдосконаленнями. Сучасні технології, такі як миттєві повідомлення, соціальні мережі, та розширені можливості мобільних пристроїв, надають більше можливостей для швидкого та ефективного оповіщення користувачів.

В контексті веб-додатків, сповіщення використовуються для інформування користувачів про події, оновлення або важливу інформацію, що стосується їх активності в додатку. Сповіщення можуть бути відображені у різних формах, таких як текстові повідомлення, вікна сповіщень, поп-апи, звукові сигнали, електронні листи тощо. Вони можуть бути надіслані негайно або за певним запланованим часом.

Сповіщення веб-додатків можуть бути згенеровані автоматично, на основі певних умов або дій користувача, або вони можуть бути відправлені вручну адміністраторами або іншими користувачами. Вони можуть містити текстову інформацію, посилання, зображення або будь-який інший контент, який є необхідним для повідомлення.

Сповіщення є важливим елементом взаємодії користувача з веб-додатком, оскільки вони допомагають інформувати, зацікавити та взаємодіяти з користувачами, забезпечуючи їм актуальну інформацію та зручний спосіб спілкування з додатком.

Алгоритми оповіщення - це процеси, що дозволяють надсилати повідомлення або сповіщення користувачам в різних системах та програмах.

В залежності від мети та типу оповіщення, існують різні алгоритми, наприклад:

- Оповіщення на основі подій (Event-based notifications):

Система надсилає повідомлення користувачу в режимі реального часу при настанні певної події, наприклад, отримання нового електронного листа або повідомлення в чаті.

- Оповіщення на основі запитів (Request-based notifications):

Користувач отримує сповіщення, коли він або інший користувач запитується про певну інформацію або виконує певну дію.

- Оповіщення на основі змін (Change-based notifications):

Користувач отримує сповіщення, коли відбувається зміна в даних або системі. Наприклад, користувач може отримати сповіщення про зміну ціни на товар в Інтернет-магазині.

- Оповіщення на основі часу (Time-based notifications):

Користувач отримує сповіщення у певний час або з інтервалом, наприклад, про подію, що відбудеться через деякий час.

- Оповіщення на основі місцезнаходження (Location-based notifications): Користувач отримує сповіщення на основі його місцезнаходження, наприклад, про наближення до магазину або ресторану.

- Оповіщення на основі поведінки (Behavior-based notifications): Користувач отримує сповіщення, коли його дії в системі вказують на певні потреби або інтереси. Наприклад, користувач може отримати сповіщення про новий товар, який відповідає його попереднім покупкам.

Ці алгоритми можуть бути використані в різних цілях, таких як електронні листи, повідомлення в соціальних мережах, сповіщення в додатках месенджерів і т. д.

Аналіз алгоритмів оповіщення важливий для визначення ефективності і надійності систем оповіщення. Основні критерії, за якими можна провести аналіз алгоритмів оповіщення, такі:

– Швидкість оповіщення: час, необхідний для оповіщення користувача про подію, повинен бути мінімальним.

– Надійність: система оповіщення повинна бути надійною і гарантувати доставку повідомлень до користувача.

– Масштабованість: система оповіщення повинна бути здатною працювати з великою кількістю користувачів і об'єктів.

– Функціональність: система оповіщення повинна бути здатною до відправлення повідомлень різного типу, наприклад, текстових, зображень, відео тощо.

– Відмовостійкість: система оповіщення повинна бути здатною працювати при збоях обладнання або програмного забезпечення.

При аналізі алгоритмів оповіщення важливо враховувати конкретний контекст їх використання. Наприклад, для оповіщення користувачів про події на веб-сайті можна використовувати алгоритм пуш-сповіщень, а для оповіщення про надходження електронної пошти можна використовувати електронну пошту або повідомлення на мобільний пристрій.

Крім того, важливим елементом аналізу алгоритмів оповіщення є їх вартість. Вартість може бути визначена розглядом таких факторів, як вартість обладнання, програмного забезпечення, трафіку мережі та інші.

Ще одним чинником, який може вплинути на вибір алгоритму оповіщення, є тип додатку, який використовується. Наприклад, якщо це мобільний додаток, то можна використовувати пуш-сповіщення, які повідомляють користувачів про нові повідомлення або події навіть тоді, коли додаток не запущено. Якщо це веб-додаток, то можна використовувати веб-сповіщення, які можуть відображати повідомлення навіть у відсутності відкритої сторінки додатку.

Також важливим чинником є спосіб оповіщення. Наприклад, якщо потрібно оповістити користувачів про дзвінок або звукове повідомлення, то

використовують звукові сигнали. Якщо ж це візуальне повідомлення, то можна використовувати спливаючі вікна, іконки або текстові повідомлення.

З огляду на усі перераховані фактори, було розроблено підходящий по умовах, швидкості та надійності алгоритм(див рисунок 2.1), який і буде використовуватись у подальших етапах розробки програмного продукту.

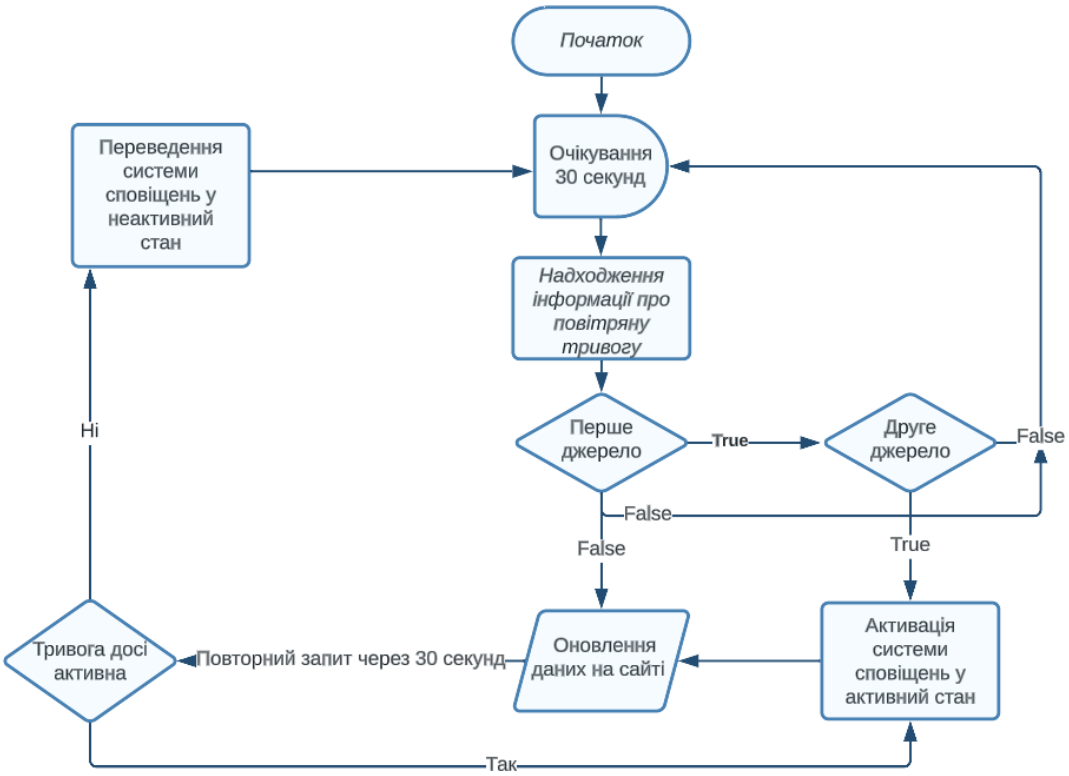

Рисунок 2.1 – Блок-схема логіки алгоритму

Отже, при виборі алгоритму оповіщення важливо враховувати тип додатку, цільову аудиторію, тип повідомлення та спосіб оповіщення, разом із серверною частиною та обсягом даних, який буде передавати сам алгоритм.

2.2 Принципи та алгоритми парсингу

Парсинг (англ. parsing) - це процес аналізу та обробки даних з файлів або джерел, які містять інформацію в певному форматі, щоб отримати корисну

інформацію. У контексті програмування, парсинг зазвичай означає аналіз текстових даних (таких як HTML-сторінки, JSON або XML файли), з метою отримання певних значень або структури даних.

Парсинг (аналіз) відноситься до процесу розбору структурованої інформації, такої як текст, HTML-сторінки або документи, для отримання корисної інформації з них. При парсингу застосовуються принципи та алгоритми для ефективного розпізнавання та обробки даних.

Ось декілька ключових принципів та алгоритмів, які часто використовуються при парсингу:

1. Регулярні вирази: Регулярні вирази є потужним інструментом для виявлення та витягування даних з текстових рядків. Вони базуються на шаблонах і дозволяють швидко знаходити та збирати потрібну інформацію.

2. Дерево розбору (parse tree): Дерево розбору представляє структуру розбору документа, де кожен вузол відповідає окремому елементу або токену. Це дозволяє аналізувати інтерпретувати структуру та відносини між елементами документа.

3. Синтаксичний аналіз (parsing): Синтаксичний аналіз використовується для перетворення послідовності символів на структуроване представлення, яке можна легко обробити. Це може включати використання граматик або формальних специфікацій для визначення правил розпізнавання мови.

4. DOM-парсер (Document Object Model): DOM-парсер використовується для парсингу HTML- або XML-документів і створює дерево DOM, яке представляє структуру документа. Це дозволяє легко навігувати по документу та отримувати доступ до його елементів та властивостей.

5. XPath та CSS-селектори: XPath та CSS-селектори є мовами запитів, які дозволяють вибирати елементи зі структурованих даних, таких як XML або HTML. Вони забезпечують мощні засоби для точного вибору інформації на основі шляхів, класів, ідентифікаторів та інших атрибутів елементів.

Ці принципи та алгоритми складають основу парсингу та допомагають ефективно витягувати корисну інформацію зі структурованих документів. Застосовуючи відповідні методи та інструменти парсингу, можна автоматизувати обробку та аналіз різноманітних даних з високою точністю та ефективністю.

Принципи парсингу визначають правила, за якими здійснюється процес видобування корисної інформації зі структурованого тексту. Основні принципи парсингу можна описати наступним чином:

– Розпізнавання структури: спочатку необхідно визначити структуру джерела даних. Це може бути документ HTML, XML або JSON, база даних або будь-який інший формат, що містить корисну інформацію.

– Розпізнавання тегів: після визначення структури джерела даних необхідно розпізнати теги, які містять корисну інформацію. Теги можуть мати атрибути, які містять додаткову інформацію.

– Розпізнавання тексту: після розпізнавання тегів необхідно видобути корисну інформацію з тексту. Це можуть бути дані, які містяться між тегами, або атрибути тегів.

– Обробка даних: після видобування корисної інформації необхідно обробити її, щоб отримати більш структуровані дані або виконати інші дії. Наприклад, можна перетворити дані у формат, який підходить для зберігання в базі даних.

– Обробка помилок: під час парсингу можуть виникати різноманітні помилки, такі як некоректний формат даних або невідомий тег. Потрібно передбачити обробку цих помилок і повідомляти користувача про проблеми з парсингом.

Взагалі кажучи, принципи парсингу можуть відрізнятися в залежності від формату джерела даних та вимог до видобування корисної інформації.

Існує безліч алгоритмів парсингу залежно від формату даних, який потрібно розібрати, та інструментів, які використовуються. Одним з найпоширеніших алгоритмів є рекурсивний спуск.

Рекурсивний спуск - це алгоритм, який використовується для парсингу контекстно-вільних граматик, таких як мови програмування або мови розмітки. Алгоритм працює шляхом рекурсивного спуску в глибину синтаксичного дерева, розбираючи кожен вузол і переходячи до наступного вузла, поки не будуть розібрані всі елементи.

Інший поширений алгоритм парсингу - це алгоритм LL(1), який використовується для парсингу мов програмування, таких як Java або C++. Цей алгоритм працює шляхом вибору правила граматики, яке відповідає наступному вхідному символу, та аналізує вхідну послідовність зліва направо.

Інші алгоритми парсингу включають Bottom-Up або LR-алгоритми, які також використовуються для парсингу мов програмування, а також алгоритми парсингу для конкретних форматів даних, таких як XML, JSON або CSV.

У мові програмування Java є кілька популярних інструментів та фреймворків для парсингу, що допомагають зчитувати та обробляти структуровані дані:

1. Jsoup: Це потужна бібліотека для парсингу HTML-документів. Вона надає зручні методи для отримання, зміни та видалення елементів, а також для навігації по структурі HTML.

2. Jackson: Це бібліотека для роботи з форматом JSON. Вона дозволяє зчитувати та записувати JSON-дані в об'єкти Java, а також працювати з ними.

3. Gson: Це ще одна бібліотека для роботи з JSON. Вона надає можливості зчитування та запису JSON-даних в об'єкти Java, а також підтримку серіалізації та десеріалізації.

4. JAXB: Це стандартна бібліотека Java для роботи з XML. Вона дозволяє перетворювати об'єкти Java в XML-документи та навпаки.

5. Apache POI: Це бібліотека для роботи з форматами документів Microsoft Office, такими як Excel, Word і PowerPoint. Вона дозволяє зчитувати, записувати та редагувати дані в цих форматах.

6. jSoup: Це бібліотека для парсингу та обробки HTML і XML. Вона має простий та зрозумілий API для навігації по структурі документа, отримання елементів та їх атрибутів.

Це лише кілька прикладів інструментів та фреймворків, які використовуються для парсингу в мові програмування Java. Залежно від ваших потреб, можуть бути інші варіанти, які також можуть бути корисними.

Gson - це бібліотека для роботи з форматом JSON (JavaScript Object Notation) в мові програмування Java. Вона дозволяє зчитувати JSON-дані та перетворювати їх у об'єкти Java (десеріалізація), а також здійснювати зворотню операцію - перетворювати об'єкти Java в JSON-формат (серіалізація).

Основні переваги Gson включають:

1. Простота використання: Gson має простий і зрозумілий API, що дозволяє легко взаємодіяти з JSON-даними.

2. Гнучкість: Вона надає різні способи налаштування процесу серіалізації та десеріалізації, включаючи виключення або включення певних полів, роботу зі специфічними типами даних тощо.

3. Підтримка анотацій: Gson підтримує використання анотацій для зазначення специфічних правил серіалізації та десеріалізації для полів або класів.

4. Підтримка вкладених об'єктів та колекцій: Gson може обробляти складні структури даних, такі як вкладені об'єкти та колекції, автоматично перетворюючи їх на JSON-формат і навпаки.

5. Висока продуктивність: Gson є швидким і ефективним, що дозволяє обробляти великі об'єми даних без значних затримок.

Gson також підтримує додаткові функції, такі як серіалізація до XML-формату, обробка специфічних форматів дати та часу, обробка спеціальних символів та інше.

Взагалі, Gson є потужним інструментом для роботи з JSON-даними в мові програмування Java. Вона широко використовується в розробці веб-додатків, мобільних додатків та інших проектів, де потрібна обробка та обмін даними у форматі JSON.

JSON (JavaScript Object Notation) - це легкий відкритий формат обміну даними, що базується на синтаксисі об'єктів JavaScript. Він використовується для передачі структурованих даних між різними системами.

Основні риси формату JSON:

- Простота читання та написання: JSON використовує зрозумілі для людей синтаксичні правила, що дозволяє легко створювати та аналізувати дані.
- Структура об'єктів: JSON використовує набір пар ключ-значення, де ключі є рядками, а значення можуть бути примітивами (числа, рядки, булеві значення), об'єктами, масивами або іншими вкладеними об'єктами.
- Масиви: JSON підтримує використання масивів, які можуть містити послідовність значень.
- Незалежність від мови програмування: JSON може бути використаний в багатьох мовах програмування, оскільки його синтаксис є загальним і не пов'язаним з конкретною мовою.
- Підтримка вкладених структур: JSON дозволяє вкладати об'єкти один в одного, що дозволяє створювати складні структури даних.
- Підтримка простих типів даних: JSON підтримує такі прості типи даних, як числа, рядки, булеві значення та спеціальні значення, такі як null.
- Розширюваність: JSON може бути розширений для підтримки додаткових типів даних за допомогою механізму серіалізації та десеріалізації.

JSON широко використовується для передачі даних веб-додатками,

використання API, зберігання конфігураційних файлів та обміну даними між сервером та клієнтом. Він є популярним форматом серіалізації та десеріалізації даних в багатьох мовах програмування, включаючи Java.

Дані з джерела інформації про повітряну тривогу, продуктивно та просто можна отримати скориставшись принципом Синтаксичний аналіз (parsing). Маючи посилання на безпосереднє джерело, яке являє собою звичайний json файл, можна скористатись вбудованою функцією мови програмування, а саме класом *InputStreamReader*, який має метод *openStream()*. Таким чином, завдячуючи структурі файлу, можна, використовуючи інструменти мови, перетворити структура файлу у стрічку, та розділити її, отримавши таким чином конкретно ту інформацію, яка необхідна для визначення стану повітряної тривоги. Опираючись на інформацію вище, складаю блок-схему алгоритму парсингу даних з джерела повітряної тривоги(див рис 2.2).

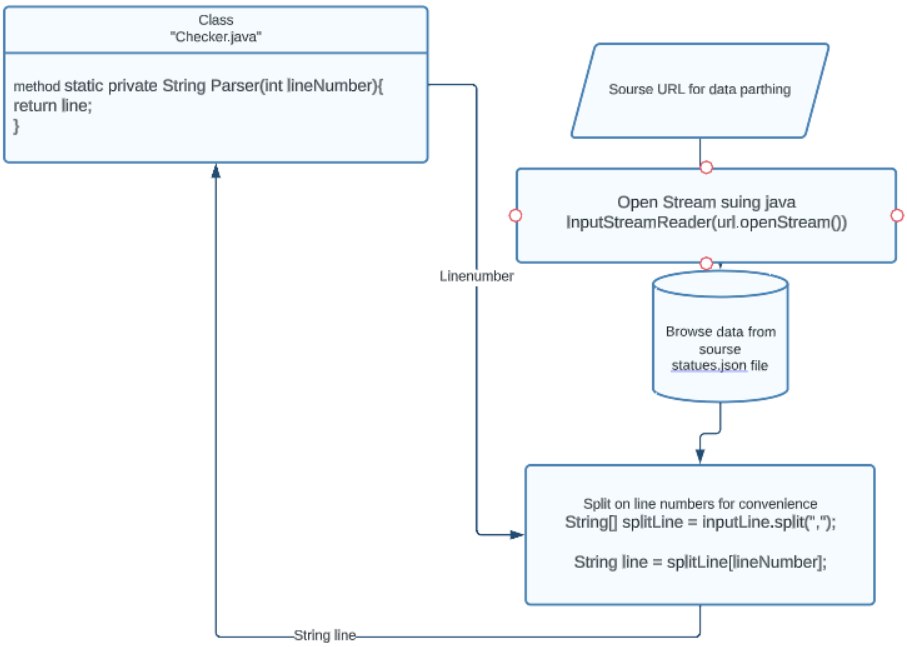

Рисунок 2.2 – Схематична блок-схема роботи парсингу.

Для другого ж джерела є простіший метод отримання інформації, за рахунок наявного та простого для розуміння і універсального API[22].

Параметр */v1/iot/active_air_raid_alerts_by_oblast.json* з методом Get, Повертає стан повітряних тривог в областях. Компактне API для використання в IoT пристроях. Результат повертається у вигляді JSON, що містить рядок:

"ANNNNNNNNNNNANNNNNNNNNNNNNN", де:

A - повітряна тривога активна в усій області,

P - часткова тривога в районах чи громадах,

N - немає інформації про повітряну тривогу.

Для кожної букви рядка є своя область в наступному порядку:

["Автономна Республіка Крим", "Волинська область", "Вінницька область", "Дніпропетровська область", "Донецька область", "Житомирська область", "Закарпатська область", "Запорізька область", "Івано-Франківська область", "м. Київ", "Київська область", "Кіровоградська область", "Луганська область", "Львівська область", "Миколаївська область", "Одеська область", "Полтавська область", "Рівненська область", "м. Севастополь", "Сумська область", "Тернопільська область", "Харківська область", "Херсонська область", "Хмельницька область", "Черкаська область", "Чернівецька область", "Чернігівська область"]

Тобто перша буква в рядку - статус повітряної тривоги в Автономній Республіці Крим, друга - в Волинській області, третя - в Вінницькій області і т.д.

Єдиним нюансом є те, що ля доступу до API необхідно використовувати персональний API токен. Отримати його можна відправивши запит на електронну адресу api@alerts.in.ua [22].

Сам токен являє собою секретний ключ, який дозволить отримати доступ до API використовуючи REST API.

Отже, серед розглянутих систем, алгоритмів та принципів парсингу, а також API, та бібліотек, які надають можливіть отримувати дані з веб-ресурсів за допомогою парсингу, було досліджено потужні та зручні інструменти, які широко й заслужено використовуються у різноманітних проєктах та додатках. По при API, яке підходить для розробки проєкту по функціоналу та тематиці,

було обрано бібліотеку Gson, завдяки її численним перевагам та простоті використання.

2.3 Архітектура фреймворку Spark

Фреймворки - це набір певних структур, шаблонів та інструментів, які використовуються для розробки програмного забезпечення. Архітектура фреймворка визначає структуру, організацію та спосіб взаємодії компонентів. Основні принципи та концепції архітектури фреймворків можуть варіюватися в залежності від специфіки конкретного фреймворка.

Основні компоненти архітектури включають:

1. Інверсія керування (Inversion of Control, IoC): Фреймворки часто використовують принцип IoC, де керування життєвим циклом об'єктів здійснюється самим фреймворком, а не розробником. Це означає, що фреймворк бере на себе відповідальність за створення, управління та взаємодію об'єктів.
2. Модульність: Фреймворки надають можливість розбити програмне забезпечення на модулі, що дозволяє більш ефективно управляти, розширювати та підтримувати додаток.
3. Шаблони проектування: Фреймворки часто використовують популярні шаблони проектування, такі як MVC (Model-View-Controller) або MVVM (Model-View-ViewModel), для організації логіки та взаємодії компонентів додатку.
4. Компонентна архітектура: Фреймворки надають можливість побудови додатків з використанням компонентів, які можуть бути повторно використовані та легко змінювати свою функціональність.
5. Розширюваність: Фреймворки часто надають механізми для розширення функціональності за допомогою розширень, плагінів або модулів.

6. Безпека: Фреймворки надають засоби для забезпечення безпеки додатків, включаючи механізми автентифікації, авторизації та захисту від атак.

7. Підтримка базової функціональності: Фреймворки часто надають готові рішення для загальних задач, таких як робота з базами даних, обробка HTTP-запитів, кешування даних тощо.

Розвиток архітектури фреймворків проходить через постійні зміни та вдосконалення, спрямовані на поліпшення продуктивності, розширення функціональності та полегшення розробки веб-додатків. Сучасні фреймворки надають розробникам широкі можливості для швидкої та ефективної розробки веб-додатків з високою якістю та масштабованістю.

Spark - це легкий фреймворк для розробки веб-додатків на мові Java. Хоча він не пов'язаний з Apache Spark, він пропонує простий та зручний спосіб створення веб-додатків з використанням вбудованого веб-сервера Jetty.

Він був створений Мартіном Гренхагеном (Martin Grennberg) в 2011 році як простий та ефективний інструмент для швидкого розробки веб-додатків.

Початково фреймворк називався "Sinatra for Java", оскільки він брав натхнення з популярного фреймворку Ruby - Sinatra. Захоплений простотою та елегантністю Sinatra, Мартін вирішив створити аналогічний фреймворк для мови Java, який би забезпечував легкість використання та розробки.

У 2012 році фреймворк отримав нове ім'я - Spark. Ім'я відображало основну філософію фреймворку: бути швидким, простим і займати мало місця. Spark вибрав шлях мінімалізму, зосередившись на основних функціях розробки веб-додатків і надавши розробникам велику свободу у виборі додаткових бібліотек та інструментів.

Згодом Spark набув популярності серед розробників Java-середовища, завдяки своїй простоті, легкості в освоєнні та зручному способу розробки веб-додатків. Цей фреймворк продовжує активно розвиватись і підтримуватись спільнотою розробників.

Сьогодні Spark є одним з популярних виборів для розробки мікросервісних архітектур, RESTful API та веб-додатків на мові Java. Він продовжує покращуватись та вдосконалюватись, дозволяючи розробникам швидко створювати потужні та ефективні веб-додатки.

Основні можливості фреймворку Spark включають:

1. Маршрутизація: Spark надає простий спосіб визначення маршрутів, що відповідають URL-шляхам. Ви можете легко визначити, який код виконуватиметься для кожного маршруту.

2. Обробка HTTP-запитів: Ви можете легко обробляти різні типи HTTP-запитів, такі як GET, POST, PUT, DELETE та інші. Spark надає зручні методи для отримання параметрів запиту, роботи з заголовками, обробки файлів тощо.

3. Шаблони: Spark підтримує використання шаблонів для генерації веб-сторінок. Ви можете використовувати різні шаблонні мови, такі як Freemarker, Mustache, Velocity та інші, для відображення даних на стороні сервера.

4. Фільтри: Ви можете використовувати фільтри для перехоплення запитів і відповідей перед тим, як вони потрапляють до маршрутів. Це дозволяє вам здійснювати певну логіку перед обробкою запитів або після них, таку як аутентифікація, авторизація, логування тощо.

5. Інтеграція з іншими фреймворками: Spark добре поєднується з іншими фреймворками та бібліотеками Java. Ви можете використовувати його разом з JPA (Java Persistence API), Hibernate, Spring та багатьма іншими для створення повноцінних веб-додатків.

6. Вбудований веб-сервер: Spark поставляється з вбудованим веб-сервером Jetty, що дозволяє вам запускати ваші веб-додатки без необхідності встановлення додаткових серверів.

Основними особливостями фреймворку Spark є:

– Швидкість обробки даних: Spark забезпечує високу швидкість обробки даних завдяки розподіленій обробці, оптимізації операцій та можливості кешування проміжних результатів обробки.

– Різноманітність джерел даних: Spark підтримує різноманітні джерела даних, такі як HDFS, Cassandra, HBase, S3, JDBC, Elasticsearch та багато інших.

– Розширюваність: Spark має гнучку архітектуру та API, що дозволяє розширювати функціональність та підтримувати різні мови програмування (Java, Scala, Python, R).

– Можливості машинного навчання: Spark має підтримку бібліотеки машинного навчання MLlib, що надає можливості для створення та навчання моделей машинного навчання.

– Можливості обробки потокових даних: Spark має підтримку бібліотеки стіків Spark Streaming, що дозволяє обробляти потокові дані в режимі реального часу.

– Можливості графічної обробки: Spark має підтримку бібліотеки графічної обробки GraphX, що дозволяє працювати з графічними структурами даних.

Окрім цього, Spark має багато інструментів для моніторингу та управління обчислювальними кластерами, таких як інтерфейси візуалізації, системи моніторингу метрик, інструменти для керування пам'яттю та ресурсами.

Одним з таких інструментів є Velocity. Velocity є шаблонним двигуном (template engine), який використовується в фреймворку Spark для генерації динамічного контенту у веб-додатках. Velocity надає простий та елегантний спосіб розділення логіки бізнес-логіки та представлення веб-сторінок.

За допомогою Velocity, розробники можуть створювати шаблони, в яких вказують статичний контент та вставляють динамічні дані за допомогою змінних, виразів та директив. Velocity надає можливість виконувати умови,

цикли, обробляти колекції даних та використовувати розширені функції форматування.

Використання Velocity у фреймворку Spark дозволяє розробникам легко створювати веб-сторінки з динамічним контентом, забезпечувати персоналізацію, форматування та генерацію даних на основі шаблонів. Velocity інтегрується з Spark і дозволяє розробникам створювати візуально привабливі та функціональні веб-сторінки з меншими зусиллями.

Одним із ключових переваг використання Velocity є його простота та легкість в освоєнні. Він має зрозумілий синтаксис, хорошу документацію та активну спільноту, що допомагає розробникам швидко впроваджувати шаблони та виконувати потрібні динамічні операції.

Загалом, Velocity є потужним інструментом для створення динамічного контенту у фреймворку Spark. Він допомагає забезпечити відокремлення логіки веб-додатка від представлення, полегшує розробку та підтримку веб-сторінок і сприяє швидкому впровадженню змін у веб-інтерфейсі.

Загалом, Spark є простим у використанні та має невеликий розмір, але при цьому надає потужні можливості для розробки веб-додатків на мові Java. Він є популярним вибором серед розробників, які шукають легкий та ефективний фреймворк для своїх проектів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ САЙТУ

3.1 Розробка серверної частини сайту

Створення серверу - це процес розгортання програмного забезпечення, яке буде обслуговувати запити від клієнтських програм, таких як веб-браузери, мобільні додатки, інтерактивні телевізори та інше. Для створення серверу потрібно вибрати платформу, мову програмування, базу даних та фреймворки, які будуть використовуватися для розробки та запуску програмного забезпечення. Після цього розробники створюють програмний код та налаштовують середовище, в якому сервер буде працювати. Зазвичай це включає установку необхідних залежностей, конфігурування веб-сервера, встановлення сертифікатів безпеки та інші дії, які дозволяють запустити сервер та забезпечити його безперебійну роботу. Після запуску сервера він готовий приймати запити від клієнтів та обробляти їх відповідно до логіки програми, що була написана розробниками.

Користуючись менеджером пакетів Maven, встановлюю необхідний мені фреймворк написавши залежність у файл pom.xml, що показано на рисунку 3.1.

```
<dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-core</artifactId>
    <version>2.9.4</version>
</dependency>
```

Рисунок 3.1 – Залежність для фреймворку Spark

Після цього, менеджер пакетів завантажить усе необхідне.

Отже, використовуючи інструменти, надані фреймворком Spark, створюю клас *ServerCore* з методом *Server* який і буде відповідальний за запуск серверу.

Для роботи серверу можна вказати його порт написавши:

```
port(8080);
```

Це не є обов'язковим кроком тому, що у серверу є порт по замовчуванню - 4567.

Для налаштування ресурсів серверу також потрібно вписати розташування зображень та інших файлів, необхідних для роботи серверу. Приклад на ведено на рисунку 3.2

```
String projectDir = System.getProperty("user.dir");
String staticDir = "/src/main/resources/public";
staticFiles.externalLocation( externalFolder: projectDir + staticDir);
```

Рисунок 3.2 – Налаштування для розташування ресурсів серверу

Тепер сервер налаштований та готовий до роботи.

Викликаючи метод stop(), сервер зупиняється, а всі маршрути очищаються. Зачекайте, а як щодо запуску сервера?

Сервер запускається автоматично, коли ви робите щось, що вимагає запуску сервера (тобто оголошення маршруту або встановлення порту).

Також є можливість вручну запустити сервер, викликавши метод init().

Основним будівельним блоком програм у Spark є набір маршрутів. Маршрут складається з трьох простих частин:

- Дія (get, post, put, delete, head, trace, connect, options)
- Шлях (/hello, /users/:name)
- Зворотна дія (request, response) -> { }

Маршрути зіставляються в порядку їх визначення. Викликається перший маршрут, який відповідає запиту.

Отже, розробка серверної частини є важливим та невід'ємним процесом для кожного незалежного та перспективного веб-додатка. Завдяки менеджеру пакетів Maven було завантажено необхідні інструменти, які дозволять відносно легко створити та налаштувати сервер, такі як Spark, Gson та Unirest.

Завдяки функціоналу мови програмування та фреймворку та бібліотек розробка проєкту стає набагато простішою.

3.2 Розробка структури сайту

Процес розробки структури сайту з використанням HTML, CSS та java фреймворку Spark складається з декількох етапів:

1. Планування структури сайту. На цьому етапі визначається, які сторінки будуть присутні на сайті, як вони будуть пов'язані між собою, яким буде розташування елементів на сторінках тощо. Це можна зробити за допомогою діаграми структури сайту.
2. Розробка HTML-коду. HTML - це мова розмітки, яка використовується для створення структури сторінок. На цьому етапі розробляються HTML-шаблони для кожної сторінки сайту. У шаблонах вказується розташування елементів, наприклад, меню, заголовки, текстові блоки, кнопки тощо.
3. Стилізація сайту з використанням CSS. CSS - це мова стилізації, яка дозволяє змінювати вигляд елементів на сторінках. На цьому етапі розробляється CSS-код для кожного HTML-шаблону. У CSS-коді вказується, які кольори, розміри, шрифти тощо використовувати для елементів сторінки.
4. Розробка логіки з використанням java фреймворку Spark. Spark - це фреймворк для розробки веб-додатків на мові програмування Java. На цьому етапі розробляється логіка взаємодії користувачів з сайтом, наприклад, обробка форм, валідація даних, збереження даних в базі даних тощо.
5. Розгортання сайту. Після успішного тестування та внесення всіх необхідних змін, сайт можна розгорнути на сервері. Для цього необхідно завантажити всі необхідні файли на сервер та налаштувати його таким чином, щоб сайт був доступний в Інтернеті.

1. Планування структури сайту.

З точки зору оптимізації завантаження критично важливої інформації, сайт буде містити лише 1 сторінку, на якій буде розміщена вся необхідна інформація. Структуру програмної частини розроблено сайту наведено на рисунку 3.1.

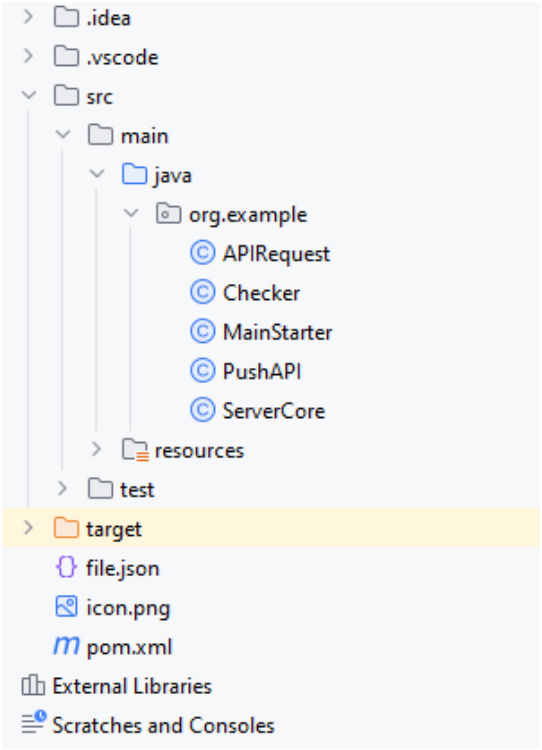

Рисунок 3.3 Структура програмної частини розроблено сайту

У директорії «java», пакету «org.example», заходяться головні файли та класи, директорія є кореневою.

Директорія «resources» вміщує файли з html – верстkою, css – файлами стилів. У директорії «public» зберігається файл стилів «style.css» та зображення у форматах «.png», «.jpeg», «.ico».

Універсальна мова моделювання (Unified Modelling Language або UML) — це мова позначень або побудови діаграм, призначена для визначення, візуалізації і документування моделей зорієнтованих на об’єкти систем програмного забезпечення [6]. UML не є методом розробки, у конструкціях

цієї мови не повідомляється про те, що робити першим, а що останнім, і не надається інструкцій щодо побудови вашої системи, але ця мова допомагає вам наочно переглядати компонування системи і полегшує співпрацю з іншими її розробниками. Конструкції UML створюються з багатьох модельних елементів, які позначають різні частини системи програмного забезпечення [7]. Елементи UML використовуються для побудови діаграм, які відповідають певній частині системи.

Існують такі типи діаграм:

- діаграма випадків використання;
- діаграми класів;
- діаграми послідовності;
- діаграми стану;
- діаграми компонентів та інші.

У дипломній роботі я використовую діаграми класів [8] . На діаграмах класів зображуються різноманітні класи, які утворюють систему і їх взаємозв’язки. Діаграми класів називають «статичними діаграмами», оскільки на них показано класи разом з методами і атрибутами, а також статичний взаємозв’язок між ними: те, яким класам відомо про існування яких класів, і те, які класи є частиною інших класів, — але не показано методи, які при цьому викликаються.

Є декілька способів побудови діаграм в залежності від того, яким чином вони будуть використовуватися:

- концептуальна. В даному випадку діаграма класів UML здійснює опис моделі певної предметної області, і в ній передбачаються тільки класи прикладних об'єктів;

- специфічна. Діаграма використовується в процесі проектування різних інформаційних систем;

- реалізаційна. Діаграма класів включає в себе класи, які використовуються у програмному коді.

UML діаграму класів наведено у графічній частині ДП.КІ.9500020.00.00.000 ДК.

Файл «pom.xml» - це файл конфігурації для Apache Maven, який використовується для збірки проекту та керування залежностями. У ньому міститься інформація про проект, таку як ім'я, версія, залежності та плагіни, необхідні для збірки проекту. POM (Project Object Model) може бути налаштований для автоматичного завантаження залежностей з репозиторіїв, що дозволяє автоматизувати процес збірки та розгортання проекту. POM є стандартом для Maven-проектів та дозволяє забезпечувати їхню однакову структуру та налаштування.

У таблиці 3.1 наведено перелік системних вимог до сервера.

Таблиця 3.1 – Системні вимоги до сервера

Найменування	Параметри
ЦП	Intel Celeron G4900 3.1 ГГц
ОЗП	2 Гб
Об’єм жорсткого диску	64 Гб
Операційна система	Windows XP/2000/2003 Linux/FreeBSD
Фреймворк	Spark
Обов’язкове ПЗ	OpenJDK-17, Intellij IDEA, Веб-браузер

2. Розробка HTML-коду.

У процесі розробки структури сайту з використанням HTML, CSS та java фреймворку Spark важливо керуватися принципами доброго дизайну та

ергономіки користування, щоб сайт був зрозумілим та зручним для використання для користувачів. Крім того, необхідно пам'ятати про безпеку сайту та захист від можливих кібератак, тому належне захист сайту від вразливостей є важливим елементом в розробці веб-додатків.

Файл «main.vm», який і є, по суті, основною сторінкою проєкту, має розширення .vm, яке використовується у фреймворку Spark для реалізації шаблонів сторінок. Це означає, що у веб-додатку, побудованому на основі Spark, використовується шаблонний підхід до створення веб-сторінок. Файли з розширенням .vm містять шаблон HTML-сторінок зі спеціальними тегами, які вказують на те, які дані потрібно вставити в HTML-код, кожен тег починається з символу «$»(див рис 3.3),. Ці теги можуть містити як звичайні текстові значення, так і програмний код на мові програмування Java. Spark використовує бібліотеку Velocity для парсингу файлів .vm та генерації HTML-коду з вставленими даними. В результаті, збірка проекту з такими файлами приведе до генерації статичних HTML-сторінок на основі шаблонів з динамічно підставленими даними.

```
<h4>    Перше джерело: $firstSourceStatus </h4>
<h4>    Друге джерело: $secondSourceStatus</h4>
```

Рисунок 3.3 Приклад тегу

Для підключення бібліотеки Velocity необхідно у файл «pom.xml» прописати залежність, яка наведена на рисунку 3.4.

```
<dependency>
    <groupId>com.sparkjava</groupId>
    <artifactId>spark-template-velocity</artifactId>
    <version>2.7.1</version>
</dependency>
```

Рисунок 3.4 – Залежність для бібліотеки Velocity

Такий підхід дає змогу ефективно розділити логіку сторінок та розмітку, забезпечуючи зручний і підтримуваний спосіб роботи з веб-сторінками у проекті на фреймворку Spark.

2. Стилізація сайту з використанням CSS.

Завдяки можливостям CSS вдалось досягти стриманого та строгого дизайну сторінки, яка перекликається з темою проєкту (див рис 3.4). З правої частини екрану розташована інтерактивна мапа, яка міняє свій колір в залежності від стану повітряної тривоги на даний момент часу (див рис 3.5). Такого результату можна досягти за допомогою можливостей фреймворку Spark, та його унікальних можливостей з використанням тегів.

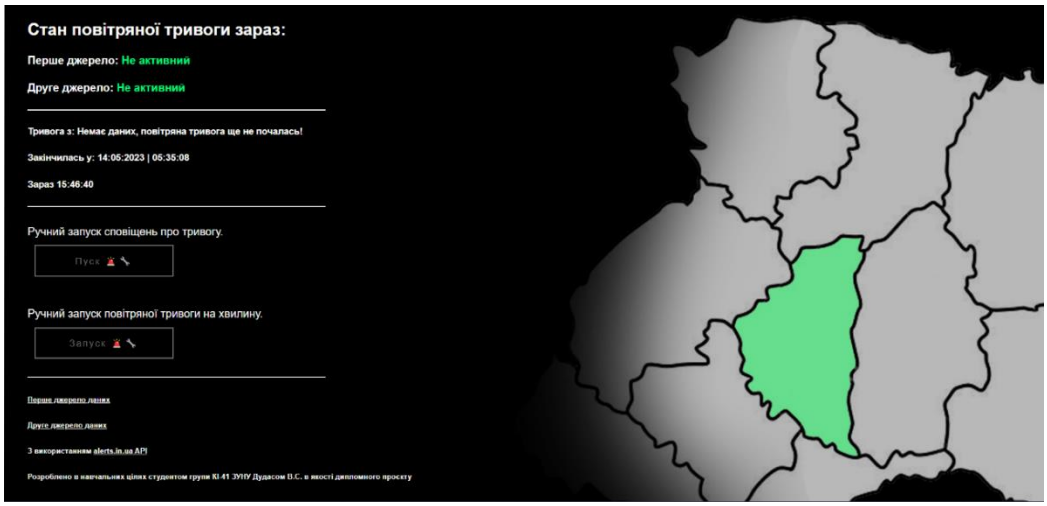

Рисунок 3.4 - Зображення сторінки сайту

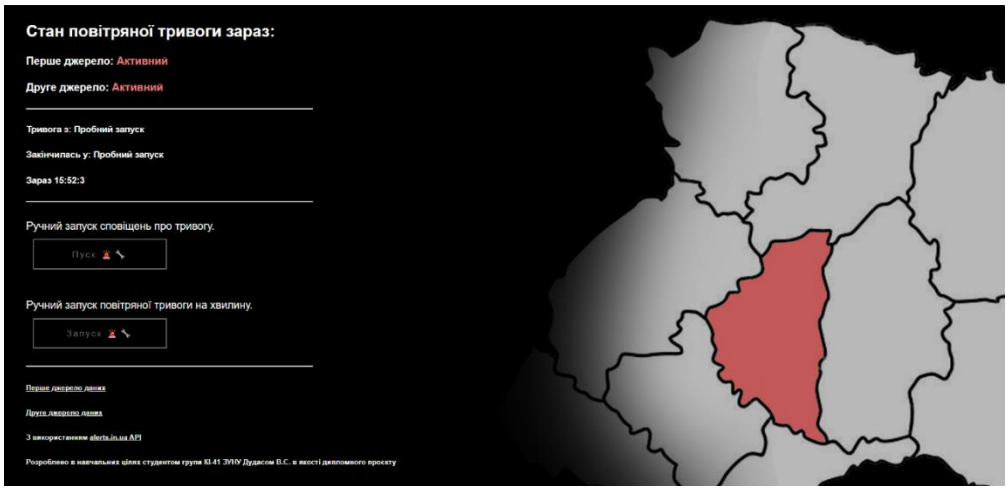

Рисунок 3.5 – Демонстрація сайту під час повітряної тривоги

3. Розробка логіки з використанням java фреймворку Spark.

Розробка логіки з використанням Java фреймворку Spark передбачає написання коду на мові програмування Java, який використовує функціонал фреймворку для обробки даних. Файл «ServerCore» в головній мірі відповідає за логіку сторінок серверу. Для прикладу логіка маршруту «/m» наведена на рисунку 3.5.

```
get( path: "/m", (request, response) -> {
    Map<String, Object> model = new HashMap<>();

    if (firstSourceStatus && secondSourceStatus) {
        response.redirect( location: "/on");
    } else {
        response.redirect( location: "/off");
    }

    model.put("pathToBgr", "/bgr.png");
    model.put("start_at", Checker.StartTime());
    model.put("end_at", Checker.EndTime());

    //*.vm file located in resources directory
    return new VelocityTemplateEngine().render(
            new ModelAndView(model,  viewName: "main.vm")
    );
});
```

Рисунок 3.5 - Логіка маршруту «/m»

Отже, методи firstSourceStatus та secondSourceStatus отримують значення з класів «Checker» та «APIRequest», який використовуючи інструменти парсингу та API з alerts.in.ua беручи дані з перевірених ресурсів. Завдяки активності двох джерел, надійність інформації на сайті в рази більша, ніж коли використовувалось лише одне джерело. В залежності від стану повітряної тривоги алгоритм переадресовує маршрут на «/on»(див. рис. 3.6) або «/off»(див. рис. 3.7).

```
get( path: "/on", (request, response) -> {
    Map<String, Object> model = new HashMap<>();

    model.put("firstSourceStatus", "<a style=\"color:#f08080\";>Активний</a>");
    model.put("secondSourceStatus", "<a style=\"color:#f08080\";>Активний</a>");

    model.put("pathToBgr", "/bgr_red.png");

    model.put("start_at", Checker.StartTime());
    model.put("end_at", Checker.EndTime());

    response.header( header: "Refresh", value: "30; url=/m");

    return new VelocityTemplateEngine().render(
            new ModelAndView(model, viewName: "main.vm")
    );
});
```

Рисунок 3.6 – Код алгоритму «/on»

```
get( path: "/off", (request, response) -> {
    Map<String, Object> model = new HashMap<>();

    model.put("firstSourceStatus", "<a style=\"color:#00ff7f\";>Не активний</a>");
    model.put("secondSourceStatus", "<a style=\"color:#00ff7f\";>Не активний</a>");

    model.put("pathToBgr", "/bgr_green.png");

    model.put("start_at", Checker.StartTime());
    model.put("end_at", Checker.EndTime());

    response.header( header: "Refresh", value: "30; url=/m");

    return new VelocityTemplateEngine().render(
            new ModelAndView(model, viewName: "main.vm")
    );
});
```

Рисунок 3.7 – Код алгоритму «/off»

У кінці кожного алгоритму є хедер, котрий перезавантажує сторінку кожні 30 секунд.

На сайті також наявна кнопка, яка може запустити тестове сповіщення, кнопка реалізується кодом наведеним на рисунку 3.8:

```
post( path: "/t", (request, response) -> {
    Map<String, Object> model = new HashMap<>();
    String firstSourceStatusNow;
    APIRequest.SecondSource();            if (firstSourceStatus && secondSourceStatus) {
        firstSourceStatusNow = "Активний\n";
    } else {
        firstSourceStatusNow = "Не активний\n";
    }
    PushAPI.display( title: "Увага!\uD83D\uDEA8", text: "Перевірка ручного режиму сповіщень!\n" +
            "Стан повітряної тривоги у Тернополі: \n" + firstSourceStatusNow +
            "Зараз " + LocalTime.now().format(DateTimeFormatter.ofPattern("HH:mm:ss")) + "🕛\n");

    response.redirect( location: "/m");


    return new VelocityTemplateEngine().render(
            new ModelAndView(model,  viewName: "main.vm")
    );
});
```

Рисунок 3.8 – Код кнопки для тестового сповіщення

Сповіщення працюють за допомогою методу `display`, класу `PushAPI`, який приймає два значення, заголовок та текст сповіщення.

```
public static void display(String title, String text) throws AWTException {
    //Obtain only one instance of the SystemTray object
    SystemTray tray = SystemTray.getSystemTray();
    //If the icon is a file
    Image image = Toolkit.getDefaultToolkit().createImage( filename: "icon.png");
    TrayIcon trayIcon = new TrayIcon(image,  tooltip: "null");
    //Let the system resize the image if needed
    trayIcon.setImageAutoSize(true);
    //Set tooltip text for the tray icon
    trayIcon.setToolTip("Система сповіщень повітряних тривог");
    tray.add(trayIcon);

    trayIcon.displayMessage(title, text, MessageType.WARNING);
}
```

Рисунок 3.9 – Код алгоритму класу PushAPI

Повна блок схема роботи програми зображена на ДП.КІ.9500020.00.00.000 БС.

4. Розгортання сайту.

Ngrok - це сервіс, який дозволяє локальному серверу зробити свій ресурс доступним глобально через інтернет. Він дозволяє тунелювати (перенаправити) з'єднання з глобальної мережі до локальної машини, де працює сервер. При цьому не потрібно налаштовувати відкриття портів на роутері або файрволі, що робить процес налаштування значно простішим.

Ngrok має безкоштовний план, який дозволяє використовувати основні можливості сервісу. Для цього потрібно завантажити ngrok-клієнт на локальний комп'ютер та запустити його з параметрами, які вказують на порт, на якому запущений веб-сервер. Після цього ngrok виділяє глобальну адресу, за якою можна звернутися до сайту.

Ngrok - це інструмент, який дозволяє тимчасово публікувати локальний веб-сервер у глобальній мережі Інтернет. Це особливо корисно для тестування і розробки веб-сайтів або додатків на локальному комп'ютері.

Щоб розгорнути локальний сайт з доступом до Інтернету за допомогою ngrok, потрібно виконати такі кроки:

1. Завантажити і встановити ngrok.
2. Запустити локальний веб-сервер.
3. Відкрити командний рядок (термінал) та ввести наступну команду, де [port] - це порт, на якому запущений локальний веб-сервер:

```
ngrok http [port]
```

4. Ngrok створити тунель до вашого локального веб-сервера та надасть тимчасовий публічний URL-адрес, який можна використовувати для доступу до сайту з будь-якого Інтернет-пристрою(див рис 3.10).

5. Щоб зупинити тунель та вимкнути ngrok, просто закрийте командний рядок.

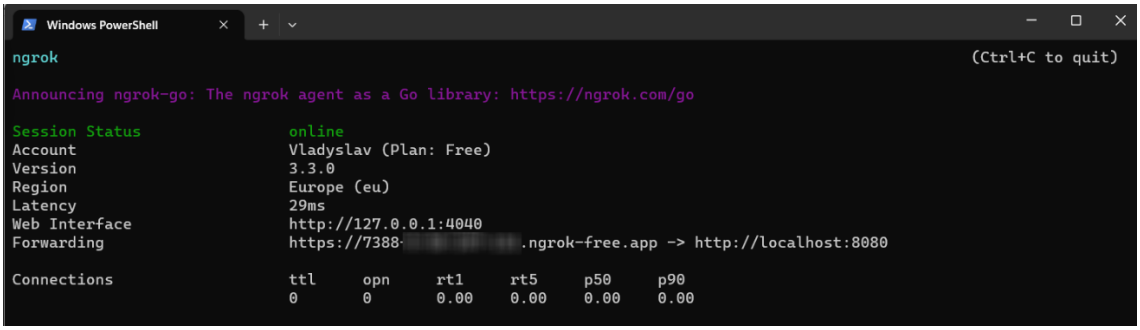

Рисунок 3.10 – Термінал ngrok

Важливо зауважити, що ngrok створює тимчасовий тунель, тому публічний URL-адрес буде працювати лише на час, поки тунель активний. Для постійного публікування веб-сайту необхідно буде скористатися іншими інструментами, наприклад, Cloudflare Tunnel, який було описано раніше.

3.3 Порівняльний аналіз та тестування

Тестування

У цьому розділі будуть перевірятись на працездатність алгоритми, які відповідають за кнопки, та алгоритми отримання даних з сайтів, які відповідають за інформацію про повітряні тривоги.

Кожні 30 секунд програма звіряється з файлом, у якому є записи про повітряну тривогу, як тільки вона починається, за для надійності інформації, викликається метод, котрий робить запит до другого джерела. Якщо тривога підтверджується другим джерелом, користувач отримує сповіщення про це, а задній фон та статус на сайті зміняться(див рис 3.11). Так буде до того моменту, допоки повітряна тривога не закінчиться. Після закінчення тривоги, інформація оновиться, та дані про завершення тривоги будуть доступні.

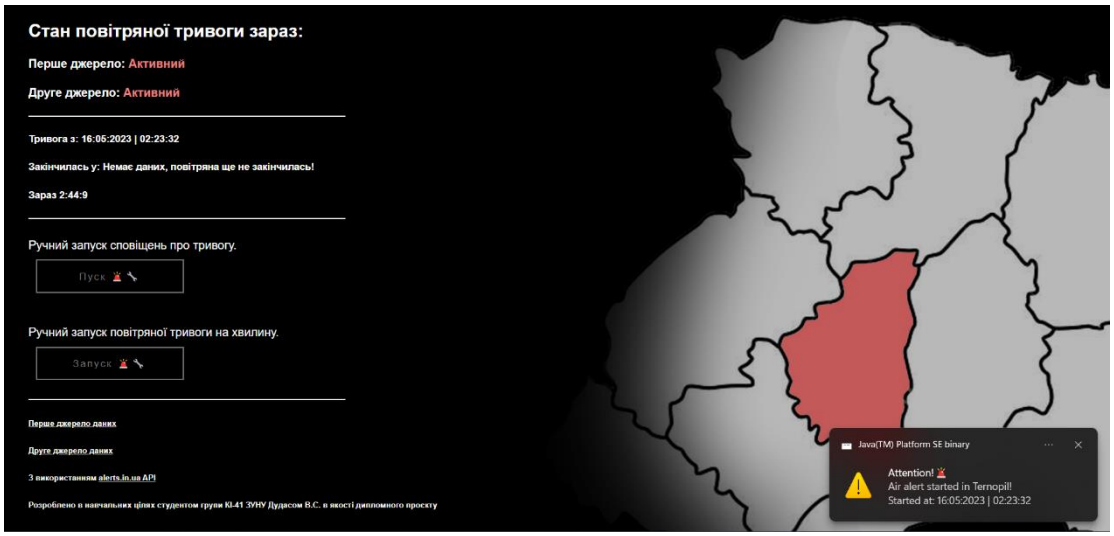

Рисунок 3.11 – Демонстрація повітряної тривоги

Також, на сайті, у окремо виділеному розділі, наявні дві кнопки(див рис 3.12), для перевірки системи сповіщень, та для запуску тестової повітряної тривоги на одну хвилину, для демонстрацій працездатності програми, за її відсутності.

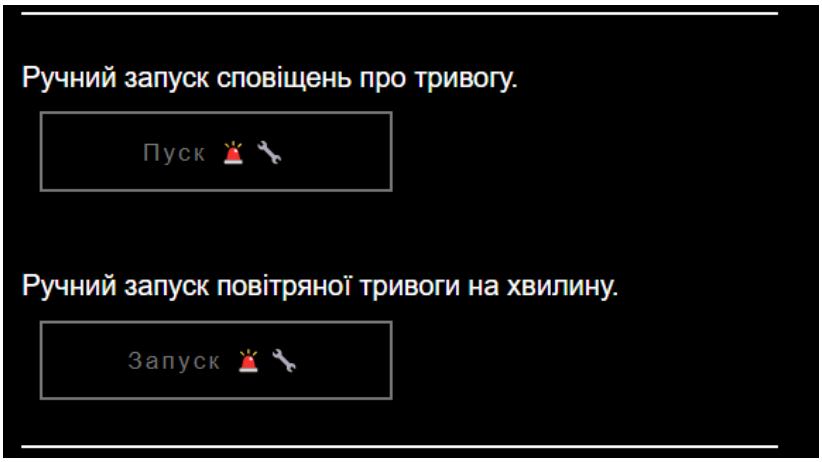

Рисунок 3.12 - Блок з кнопками

Якщо натиснути кнопку «Ручний запуск сповіщень про тривогу.», на пристрій буде надіслано сповіщення, яке інформує користувача про актуальний стан повітряної тривоги та час на даний момент(див рис 3.13).

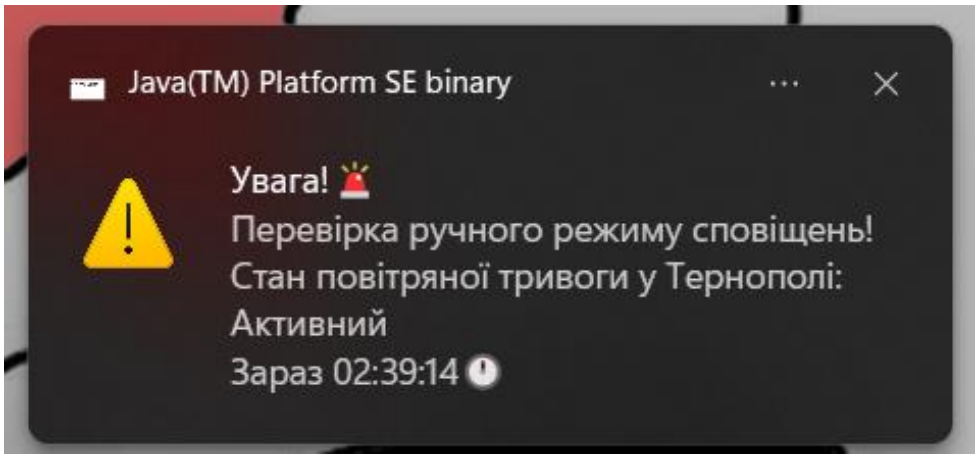

Рисунок 3.13 – Запущене вручну сповіщенняє

Також, звернувши увагу на зображення терміналу із запущеним ngrok (див рис 3.14), можна дійти висновку, що все працює справно.

Запити з статусом 200 OK є вдалими та активними, інші ж у свою чергу, котрі мають статус 302 Found, були розпізнані системою, але не мали ніякої графічної частини тому, що вони відповідальні за переадресацію користувача по необхідному шляху.

```
HTTP Requests
-------------

GET /m                          302 Found
GET /r                          302 Found
GET /                           302 Found
GET /on                         200 OK
GET /bgr_red.png                200 OK
GET /style.css                  200 OK
GET /bgr_red.png                200 OK
GET /style.css                  200 OK
GET /m                          302 Found
GET /on                         200 OK
```

Рисунок 3.14 - Термінал ngrok

Аналіз

Отже, провівши порівняльний аналіз наявних веб-додатків для інформування населення про повітряну тривогу можна зробити такі висновки. Хоча і всі, або більшість наявних ресурсів мають приємне візуальне оформлення та спрямовані не на одну область, у них, на відміну від розробленого проєкту немає можливості миттєво відправляти сповіщення, що може бути критично важливо у регіонах, наближених до лінії фронту. А за рахунок мінімалізму, та відсутності великої карти зі всіма областями, а показ лише одної області, гарантовано буде давати вплив на швидкодію завантаження веб-ресурсу, оскільки інформації для завантаження буде набагато менше, що також є дуже важливим критерієм для інструменту такого типу.

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Розрахунок витрат на виконання проєктного рішення

Нормування праці в процесі створення ПЗ істотно ускладнено в силу творчого характеру праці програміста. Тому трудомісткість розробки ПЗ може бути розрахована на основі системи моделей з різною точністю оцінки.

Трудомісткість розробки ПЗ розраховують за формулою:

$$t = t_o + t_u + t_n + t_{\text{налаг}} + t_{\text{д}} \quad (4.1)$$

де $t_{\text{о}}$ - витрати праці на підготовку й опис поставленої задачі;

t_u - витрати праці на дослідження алгоритму рішення задачі;

t_a - витрати праці на розробку блок-схеми алгоритму;

t_n - витрати праці на програмування по готовій блок-схемі;

$t_{\text{налаг}}$- витрати праці на налагодження програми на ПК;

$t_{\text{д}}$ - витрати праці на підготовку документації.

$$t = 136.5 + 420 + 420 + 2\,100 + 980 = 4\,056{,}7 (\text{люд/год})$$

4.1.1. Розрахунок витрат на оплату праці розробників

Розрахуємо середнього динну оплату програміста. Для цього необхідно спочатку визначити його річний фонд грошового забезпечення. Це можна зробити, знаючи місячне грошове забезпечення програміста. Воно складає приблизно 15000,00 гривень. Таким чином, річний фонд грошового забезпечення 180000 гривень.

Кількість робочих годин у році розраховуємо за формулою:

$$N_p = (N - N_n - N_b) * 8 \quad (4.2)$$

де N – загальна кількість днів у році,

N_n – кількість святкових днів у році,

N_b – кількість вихідних днів у році.

$$N_p = (365 - 11 - 107) \times 8 = 1\,976$$

Приймається, що кількість святкових днів у році – 14, а вихідних – 104. Середньо годинна оплата праці програміста визначається за формулою:

$$C_n = \frac{\Phi_p}{N_p} \qquad (4.3)$$

де Φ_p – річний фонд грошового забезпечення.

$$C_n = 180\,000 / 1\,976 = 91{,}1$$

Витрати на оплату праці розробників програми складають:

$$\mathrm{B_{оп}} = \mathrm{C}_n * \mathrm{T_з} \qquad (4.4)$$

де $\mathrm{T_з}$- загальна кількість годин роботи програміста над проектом.

$$\mathrm{B_{оп}} = 91{,}1 \times 3\,295 = 300\,174{,}5$$

4.2 Визначення експлуатаційних витрат

4.2.1**.** Витрати, пов'язані з розробкою програми на ПК розраховуються:

$$\mathrm{B_{ПК}} = \mathrm{T_{ПК}} * t_c \qquad (4.5)$$

де $\mathrm{T_{ПК}}$ - час використання ПК для розробки програми,

$C_{ПК}$ - собівартість машинного часу обчислювальної техніки (розраховує бухгалтерія підприємства).

$$В_{ПК} = 3\,295 \times 0{,}09 = 296{,}55$$

Собівартість однієї години роботи ПК дорівнює:

$$C_{ПК} = \frac{В_e}{Ф_{ПК}} \qquad (4.6)$$

де $В_e$- річні поточні витрати на експлуатацію ПК,

$Ф_{ПК}$ - річний фонд часу корисної роботи ПК.

$$C_{ПК} = 1\,450 / 1\,976 = 0{,}73$$

Розрахуємо річний фонд часу роботи ПК. Визначивши дійсний річний фонд часу ПК у годинах, отримаємо можливість оцінити собівартість годин машинного часу.

Дійсний річний фонд часу ПК дорівнює:

$$Ф_д = N_P - (Ф_m + Ф_{річ}) \qquad (4.7)$$

де $Ф_m$ - місячний фонд часу на профілактику і ремонт ПК (час профілактики щомісячно – 5 годин),

$Ф_{річ}$ - річний фонд часу на профілактику і ремонт ПК (час профілактики щорічно – 6 діб)

$$Ф_д = 1\,976 - (5 + 144) = 1\,797 \text{ (год на рік)}$$

4.2.2. Річні поточні витрати на експлуатацію програмного забезпечення визначаються за формулою:

$$B_{\text{ПК}} = B_E + B_{\text{А}_\text{р}} + B_{\text{РЕМ}_\text{р}} + B_{\text{ДК}} + \text{В}_\text{І} \quad (4.8)$$

де B_E – річні витрати на електроенергію для ПК,

$\text{В}_{\text{А}_\text{р}}$ – річні відрахування на амортизацію,

$B_{\text{РЕМ}_\text{р}}$ – річні витрати на ремонт ПК,

$B_{\text{ДК}}$ – річні витрати на додаткові комплектуючі ПК,

$\text{В}_\text{І}$ – інші витрати.

$$B_{\text{ПК}} = 268{,}7 + 6\,300 + 2\,520 + 840 + 2100 = 12\,028{,}7$$

Суму річних амортизаційних відрахувань визначаємо за такою формулою:

$$\text{В}_{A_\text{р}} = \text{Ц}_{\text{ПК}} * \text{Н}_\text{А} \quad (4.9)$$

$$\text{В}_{A_\text{р}} = 42\,000 * 0{,}15 = 6\,300$$

де $\text{Ц}_{\text{ПК}}$ – балансова вартість ПК,

$\text{Н}_\text{А}$– норма амортизаційних відрахувань (дорівнює 15% у квартал).

Балансову вартість ПК розраховуємо за формулою:

$$\text{Ц}_{\text{ПК}} = \text{Ц}_\text{Р} * (1 + \text{К}_{\text{УН}}) \quad (4.10)$$

$$\text{Ц}_{\text{ПК}} = 42\,000 * (1 + 0{,}12) = 47\,040$$

де $\text{Ц}_\text{Р}$ – ринкова вартість ПК,

$\text{К}_{\text{УН}}$ – коефіцієнт, що враховує витрати на установку й налагодження ПК (приймається рівним 12%).

4.2.3. У ході розробки програмного комплексу ПК використовується на таких етапах програмування:

- написання програми за готовою схемою алгоритму;
- налагодження програми на ПК;
- підготовки документації по задачі.

Таким чином, витрати машинного часу склали ($t_{маш}$):

$$t_{маш} = t_n + t^k_{налаг} + t_д, \quad (4.11)$$

$$t_{маш} = 420 + 3\,150 + 980 = 4\,550$$

Витрати на оплату машинного часу розраховуємо за формулою:

$$В_{маш} = t_{маш} * С_{ПК}. \quad (4.12)$$

$$В_{маш} = 4\,550 * 0{,}73 = 3\,321{,}5$$

Загальні витрати на розробку програмного комплексу складають:

$$В_{заг} = В_{оп} + В_{маш}. \quad (4.13)$$

$$В_{заг} = 300\,174{,}5 + 3\,321{,}5 = 303\,496$$

4.3 Розрахунок ціни споживання

Витрати на електроенергію, що споживає ПК, визначаємо за формулою:

$$B_E = Р_{ПК} * Ф_{ПК} * Ц_Е * К_{ІВ} \quad (4.14)$$

$$B_E = 0{,}17 кВт/год \times 1\,976 \times 1{,}68 \times 0{,}8 = 268{,}7$$

де $Р_{ПК}$ – паспортна потужність ПК,

$Ф_{ПК}$– річний фонд корисного часу роботи ПК,

$Ц_{Е}$– вартість 1 кВт/год електроенергії,

$К_{ІВ}$ – коефіцієнт інтенсивного використання ПК (0,7 - 1).

Таким чином, розрахункове значення витрат на електроенергію, що споживає ПК, складає:

- витрати на поточний і профілактичний ремонт (приймаються рівними 6% від вартості ПК):

$$B_{\text{РЕМ}_\text{р}} = \text{Ц}_{\text{ПК}} * 0{,}06, \quad (4.15)$$

$$B_{\text{РЕМ}_\text{р}} = 42\,000 * 0{,}06 = 2\,520$$

- витрати на додаткові комплектуючі – витрати необхідні для забезпечення експлуатації ПК (приймаються рівними 2% від вартості ПК):

$$B_{\text{ДК}_\text{р}} = \text{Ц}_{\text{ПК}} * 0{,}02, \quad (4.16)$$

$$B_{\text{ДК}_\text{р}} = 42\,000 * 0{,}02 = 840$$

- інші витрати, тобто непрямі витрати пов'язані з експлуатацією ПК (приймаються рівними 5-10% від вартості ПК):

$$B_{\text{І}_\text{р}} = \text{Ц}_{\text{ПК}} * 0{,}05, \quad (4.17)$$

$$B_{\text{І}_\text{р}} = 42\,000 * 0{,}05 = 2100$$

4.4 Визначення економічної ефективності

За міжнародним стандартам для оцінки ефективності розробки ПЗ застосовують такі показники:

- внутрішня норма дохідності;
- чистий приведений дохід;
- рентабельність;
- термін окупності.

Показник внутрішньої дохідності характеризує величину чистого прибутку (чистого валового доходу), що припадає на одиницю інвестиційних вкладень у кожному часовому інтервалі життєвого циклу проекту. Розрахунок цього показника виконується за такою формулою:

$$\sum_{i=0}^{T} \frac{D_i}{(1+q)^i} - \sum_{i=0}^{T} \frac{K_i}{(1+q)^i} = 0 \qquad (4.18)$$

де D_i - дохід (прибуток) у і-му періоді;

K_i - інвестиційні вкладення в і-му періоді з урахуванням інфляційних процесів;

i - періоди виконання і впровадження проекту;

T - загальний період (тривалість) життєвого циклу проекту;

q - показник внутрішньої норми дохідності.

Показник інвестиційних вкладень з урахуванням інфляційних процесів обчислюємо за формулою:

$$K_i = \varphi_i + R_i. \qquad (4.19)$$

$$K_i = 0{,}15$$

де φ_i - коефіцієнт інфляції на поточний період;

R_i- інвестиційні платежі в і-му періоді (капітальні вкладення).

Дохід від розробки ПЗ у і-му періоді розраховуємо за формулою:

$$D_i = J_i(B_i - C_i). \quad (4.20)$$

$$D_i = 1(200\,000 - 97\,483{,}5) = 102\,516{,}5$$

де B_i - ціна продажу програмного продукту в і-му періоді;

C_i - собівартість програмного продукту (фактично дорівнює сумі витрат на розробку ПЗ);

J_i - кількість ПЗ.

Вартість продажу розробленого продукту розраховують за формулою:

$$\mathrm{B_i} = B_{\text{заг}}\left(1 + \frac{p}{100}\right). \quad (4.21)$$

$$\mathrm{B_i} = 303\,496\left(1 + \frac{0{,}15}{100}\right) = 303\,951{,}2$$

де p - середній рівень рентабельності на поточний період.

Показник рентабельності інвестицій. У практиці середнього бізнесу для визначення ефективності проектних рішень широко використовується.

$$p = \frac{\sum_{i=0}^{\mathrm{T}} \frac{D_i}{(1+q)^i}}{\sum_{i=0}^{\mathrm{T}} \frac{K_i}{(1+q)^i}} - 1 > 0. \quad (4.22)$$

У ринкових умовах при ціновій політиці, що змінюється, показник терміну окупності є одним з головних для підприємств. Він визначається на основі величини капітальних витрат по періодах розробки програмного продукту та величини фактичних чи прогнозних доходів:

$$\sum_{i=0}^{\mathrm{T}} K_i = \sum_{i=0}^{\mathrm{T}} D_i \qquad (4.23)$$

де T - термін окупності,

D_i - дохід (прибуток) у поточному періоді,

K_i - капітальні витрати у поточному періоді.

Економічна ефективність полягає у відношенні результату від розробленого програмного продукту до затрачених ресурсів:

$$\mathrm{E} = \frac{D_i}{\mathrm{B}_{\text{заг}}} \qquad (4.24)$$

$$E = \frac{102\ 516{,}5}{303\ 496} = 0{,}33$$

Тоді термін окупності можна розрахувати за такою формулою:

$$\mathrm{T} = \frac{1}{\mathrm{E}} \qquad (4.25)$$

$$T = \frac{1}{0{,}33} = 3{,}03 \qquad (4.26)$$

ВИСНОВКИ

1.У процесі аналізу існуючих рішень та аналогів було виявлено, що на ринку існує широкий спектр веб-ресурсів та додатків, які надають можливість отримувати оповіщення про події. Було також проаналізовано технології розробки веб-додатків, зокрема мову програмування Java та фреймворк Spark, як потенційно вигідні варіанти для реалізації проекту. Крім того, проведений аналіз архітектури веб-додатків, зокрема монолітної, мікросерверної та серверлес архітектур, які виявилися важливими для забезпечення масштабованості та надійності системи.

2.У процесі розробки були створені алгоритми оповіщення про повітряні тривоги, які дозволяють ефективно виявляти та надсилати сповіщення користувачам про стан повітряної тривоги у заданому регіоні. Також було проведено аналіз та розроблено алгоритми парсингу сайтів, що дозволяють отримувати необхідну інформацію з веб-ресурсів та використовувати її для подальшої обробки. Також, для отримання даних було використано надійне та перевірене API. Архітектура фреймворку Spark також була розглянута та використана для розробки функціоналу сайту.

3.У результаті розробки була створена серверна частина сайту, яка забезпечує зберігання та обробку даних про події, а також надсилання сповіщень користувачам. Була розроблена структура сайту, яка включає різні модулі та функціональність для зручного використання користувачами. Також було проведено порівняльний аналіз та тестування реалізованого функціоналу, що дозволило оцінити його продуктивність, швидкість відгуку та функціональність.

4.Було виконано розрахунок витрат на виконання проєктного рішення, включаючи розробку, інфраструктуру та інші складові. Визначена експлуатаційна вартість системи, яка включає витрати на розробку та

різноманітні витрати. Встановлена економічна ефективність проєкту, порівнявши витрати та отримані користі.

5.В цілому, виконаний проєкт програмної системи оповіщення студентів кафедри комп’ютерної інженерії на основі технології Push API дозволяє ефективно виявляти, обробляти та оповіщати користувачів про актуальний на конкретний відрізок часу стан повітряної тривоги. Він базується на сучасних технологіях розробки веб-додатків, таких як мова програмування Java та фреймворку Spark. Проєкт пройшов тестування та порівняльний аналіз, що підтвердило його продуктивність та функціональність. Техніко-економічне обґрунтування показує його економічну вигідність та рентабельність.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Hillel Blog : вебсайт URL: https://blog.ithillel.ua/articles/web-application-architecture
2. Spark Framework: вебсайт URL: http://Spark.com/
3. Електронна енциклопедія Вікіпедія: Веб-сайт URL: http://uk.wikipedia.org/wiki/Веб-сайт.
4. Електронна енциклопедія Вікіпедія: Веб-додаток URL: http://uk.wikipedia.org/wiki/Веб-додаток.
5. GitHub фреймворку Spark: вебсайт URL: https://github.com/perwendel/spark
6. Розділ 2. Основи UML – діаграм: URL: https://docs.kde.org/trunk4/uk/kdesdk/umbrello/uml-basics.html
7. Мюллер Р.Дж. Бази даних і UML. Проєктування / переклад з англ. Е. Молодцова. – М.: Видавництво “Лори”,2002 – 432с.: ил.
8. Острей О.Р. Діаграми класів UML як засіб моделювання інформаційної системи моніторингу освіти / М.: - 2008. № 2. – С.85-89.
9. About Oracle Company : вебсайт URL: https://www.oracle.com/uk/corporate/
10. IntelliJ IDEA overview :вебсайт URL: https://www.jetbrains.com/help/idea/discover-intellij-idea.html
11. What is Java technology and why do I need it? : вебсайт URL: https://www.java.com/en/download/help/whatis_java.html
12. "Emergency data exchange language resource messaging (EDXL-RM) 1.0 Errata", *OASIS Emergency Management Technical Committee*.
13. "National incident management system guideline for mutual aid", Aug. 2020, [online] Available: https://www.fema.gov/emergency-managers/nims/components.\

14. J. Yu, Q. Z. Sheng, J. K. Swee, J. Han, C. Liu and T. H. Noor, "Model-driven development of adaptive web service processes with aspects and rules", *J. Comput. Syst. Sci.*, vol. 81, no. 3, pp. 533-552, 2015.

15. W. Song and H.-A. Jacobsen, "Static and dynamic process change", *IEEE Trans. Services Comput.*, vol. 11, no. 1, pp. 215-231, 2016.

16. L. Sabatucci and M. Cossentino, "Supporting dynamic workflows with automatic extraction of goals from BPMN", *ACM Trans. Auton. Adaptive Syst.*, vol. 14, no. 2, pp. 1-38, 2019.

17. J. W. Morentz, "Unified incident command and decision support (UICDS): A Department of Homeland Security initiative in information sharing", *Proc. IEEE Conf. Technol. Homeland Secur.*, pp. 321-326, 2008.

18. B. Shafiq, S. Ae Chun, V. Atluri, J. Vaidya and G. Nabi, "Resource sharing using UICDS framework for incident management", *Transforming Government: People Process Policy*, vol. 6, no. 1, pp. 41-61, 2012.

19. "Incident resource inventory system", Apr. 2018, [online] Available: https://nimstools.preptoolkit.org/.

20. S. Döweling, F. Probst, T. Ziegert and K. Manske, "SoKNOS: An interactive visual emergency management framework", *Proc. GeoSpatial Vis. Analytics*, pp. 251-262, 2009.

21. N. Adam, J. Eledath, S. Mehrotra and N. Venkatasubramanian, "Social media alert and response to threats to citizens (SMART-C)", *Proc. 8th Int. Conf. Collaborative Comput. Netw. Appl. Worksharing*, pp. 181-189, 2012.

22. Alerts.in.ua API:вебсайт URL https://devs.alerts.in.ua/

23. T. L. Hinrichs, N. S. Gude, M. Casado, J. C. Mitchell and S. Shenker, "Practical declarative network management", Proc. 1st ACM Workshop Res. Enterp. Netw. (WREN '09), pp. 1-10, 2009.

24. A. Voellmy, H. Kim and N. Feamster, "Procera: A language for high-level reactive network control", Proc. HotSDN’12, pp. 43-48, 2012.

25. J. Robie, R. Cavicchio, R. Sinnema and E. Wilde, "RESTful service description language (RSDL) describing RESTful services without tight coupling", Proc. Balisage: The Markup Conf., 2013.

26. Шрифт цифрової держави : вебсайт URL: https://thedigital.gov.ua/fonts.

27. MVN repository: вебсайт URL: https://mvnrepository.com/.

28. MVN repository Spark Project Core: вебсайт URL: https://mvnrepository.com/artifact/org.apache.spark/spark-core.

29. L. Li and W. Chou, "Design patterns for RESTful communication web services", Proc. IEEE Int. Conf. Web Serv. (ICWS’10), pp. 512-519, Jul. 2010.

30. Stack Overflow: вебсайт URL:https://stackoverflow.com/.

31. Lucid – створення діаграм : вебсайт URL: https://lucid.app/documents#/documents?folder_id=recent