Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем та програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістра
(назва освітнього ступеня)

на тему: Проєктування та розробка системи моніторингу за керованими рухомими об'єктами на карті з використанням технологій Java, Spring та протоколу GTFS

Виконав: студент 6 курсу, групи СПм-61
спеціальності 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

Корба Дмитро Ігорович
(підпис) (прізвище та ініціали)

Керівник Мудрик І.Я.
(підпис) (прізвище та ініціали)

Нормоконтроль Стоянов Ю.М.
(підпис) (прізвище та ініціали)

Завідувач кафедри Петрик М.Р.
(підпис) (прізвище та ініціали)

Рецензент Осухівська Г.М
(підпис) (прізвище та ініціали)

Тернопіль 2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет _____
(повна назва факультету)

Кафедра _____
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)
« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістра
(назва освітнього ступеня)

за спеціальністю 121 «Інженерія програмного забезпечення»
(шифр і назва спеціальності)

студенту Корбі Дмитру Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Проектування та розробка системи моніторингу за керованими рухомими об'єктами на карті з використанням технологій Java, Spring та протоколу GTFS»

Керівник роботи ст. викл. каф. ПІ Мудрик І.Я.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 20___ року № _____

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст роботи (перелік питань, які потрібно розробити)

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка

Студент _____ Корба Д.І.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Мудрик І.Я.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота магістра за спеціальністю «Інженерія програмного забезпечення» на тему «Проєктування та розробка системи моніторингу за керованими рухомими об'єктами на карті з використанням технологій Java, Spring та протоколу GTFS». Пояснювальна записка до випускної роботи містить 74 сторінок, 27 ілюстрацій, 16 бібліографій, 1 таблиця та 4 додатки.

Розробка системи моніторингу за рухом транспорту є важливим завданням в контексті сучасного розвитку міст та транспортної інфраструктури. Використання технологій Java, Spring та протоколу GTFS дозволить створити ефективний інструмент для відстеження та оптимізації руху транспортних засобів.

Мета роботи: розробити систему моніторингу, яка використовує технології Java та Spring для відстеження руху керованих об'єктів на карті.

Очікувані результати: Створення функціональної та ефективної системи моніторингу руху транспорту на карті з використанням технологій Java та Spring, яка інтегрує протокол GTFS. Система повинна надавати точну та актуальну інформацію про місцезнаходження керованих об'єктів.

Ключові слова: відстеження, управління статусом, оптимізація ефективності, логістика, GIS (географічні інформаційні системи) , GTFS, Java Spring

SUMMARY

Master's qualification work in the specialty "Software Engineering" on the topic "Design and development of a monitoring system for controlled moving objects on the map using Java, Spring and GTFS protocol technologies." The explanatory note to the final thesis contains 74 pages, 27 illustrations, 16 bibliographies, 1 table and 4 appendices.

The development of a traffic monitoring system is an important task in the context of the modern development of cities and transport infrastructure. The use of Java, Spring and the GTFS protocol will allow to create an effective tool for tracking and optimizing the movement of vehicles.

The goal of the work: to develop a monitoring system that uses Java and Spring technologies to track the movement of controlled objects on the map.

Expected results: Creation of a functional and effective traffic monitoring system on the map using Java and Spring technologies, which integrates the GTFS protocol. The system should provide accurate and up-to-date information about the location of controlled objects.

Keywords: tracking, status management, efficiency optimization, logistics, GIS (geographic information systems), GTFS, Java Spring

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Протокол GTFS – General Transit Feed Specification це відкритий формат даних, розроблений для обміну інформацією про громадський транспорт.

UML (Unified Modeling Language) – уніфікована мова моделювання яка використовується при розробці ПЗ.

Software architecture – архітектура програмного продукту. Це структура програми або обчислювальної системи, яка включає програмні компоненти, а також відносини між ними.

Проектування ПЗ – це процес визначення архітектури, набору компонентів, їх інтерфейсів, інших характеристик системи і кінцевого складу програмного продукту.

Вимоги – це умови, можливості, які повинні задовольнятися системою, компонентом системи, або якими система/компонент системи повинна володіти для забезпечення умов контракту, стандартів, специфікацій чи інших регулюючих документів.

ЗМІСТ

ВСТУП

Сучасні технології дозволяють ефективно використовувати інформацію для покращення якості життя громади та оптимізації громадського транспорту. Розробка системи моніторингу контрольованих рухомих об'єктів на карті є важливим етапом у вдосконаленні транспортної інфраструктури. Цей проект спрямований на створення ефективної та орієнтованої на користувача системи відстеження транспортних маршрутів та їх переміщення за допомогою технологій Java, Spring та GTFS.

Тема дослідження та розробки з системи моніторингу за керованими рухомими об'єктів на карті з використанням сучасних технологій Java є цікавим напрямком для розробки програмного забезпечення, спрямованого на відстеження транспортних засобів (і не тільки їх) та їх руху за допомогою вказаних технологій Spring та протоколу роботи з геоданими GTFS. Давайте розглянемо основні аспекти, які можна включити в цей проект:

Потребою користувачів цього проекту є проєктування та розробка системи моніторингу, яка дозволить відслідковувати рух транспортних засобів на карті в режимі реального часу, отримувати інформацію про розклад руху, затримки та зміни маршруту. Основні завдання включають:

- Надання користувачам зручного веб-інтерфейсу;
- отримання та оновлення даних від транспортних агентств за протоколом GTFS;
- зберігання та ведення інформації про транспортні маршрути та їх рух у базі даних;
- інтеграція з картографічними сервісами для відображення руху транспортних засобів на карті;

Ця тематика пропонує широкий спектр можливостей для реалізації та розвитку. Дана розробка об'єднує етапи робот з проектування та розробки архітектури, написання та підготовку документації проєкту, використання баз

даних для збереження великих масивів даних, веб-розробкою та реалізацією клієнтського додатку, використанням картографічних сервісів та обробкою даних громадського транспорту.

Для досягнення цих цілей важливо: визначення функціональних вимог, основні функції та можливості програми. Які дані відстежувати? Які функції має включати програма?; вибрати стек технологій для розробки, мову програмування, фреймворк та інші інструменти, які відповідають потребам; спроектувати бази даних для зберігання інформації про транспортні засоби, їх рух та інші важливі дані; здійснити інтеграцію з GPS, функціонал збору даних від GPS-пристроїв транспортних засобів; розробити користувацький інтерфейс, де вони можуть переглядати карту з відображенням руху транспортних засобів у можливість відслідковування руху в реальному часі з використанням технології для швидкої обробки та оновлення даних; механізми безпеки даних: для забезпечення конфіденційності зібраних даних про рух транспортних засобів; реалізація звітності: функції звітності, щоб користувачі могли отримувати інформацію про маршрути, швидкість та інші характеристики руху; провести інтенсивне тестування програми для виявлення та виправлення можливих помилок та недоліків та відлагодження; забезпечити етапи впровадження та підтримки: запустити програму в роботу, а потім її підтримку та обслуговування. Варто врахувати зміни в технологічному середовищі та потребах користувачів.

Загальна мета системи моніторингу контрольованих рухомих об’єктів на карті полягає в забезпеченні відстеження та керування розташуванням, статусом і поведінкою цих об’єктів у реальному часі, що дозволяє оптимізувати логістику, безпеку та ефективність роботи в різних галузях і секторах. . Це полегшує прийняття обґрунтованих рішень і покращує загальну обізнаність про ситуацію для покращеного відстеження активів, управління автопарком і реагування на надзвичайні ситуації.

1 АНАЛІТИЧНА ЧАСТИНА

1.1. Актуальність теми та опис об’єкта проектування

Система моніторингу керованих рухомих об'єктів на карті - це комплексне рішення, призначене для відстеження та аналізу реального часу та історичних переміщень об'єктів у визначеній географічній зоні. Система використовує архітектуру клієнт-сервер із використанням передових технологій, щоб надати користувачам надійну та зручну платформу для моніторингу та керування рухомими об’єктами.

Такі предмети отримують все більше розповсюдження в різних сферах людської діяльності. Наприклад, в агропромисловості вони використовуються для точного, ефективного та своєчасного внесення добрив. Також, використання керованих рухомих об'єктів може бути в пригоді для нагляду за безпекою на великих територіях із значущими інфраструктурними об'єктами.

У сучасному світі громадський транспорт стає неодмінною складовою структури міського життя. Відстеження руху та оптимізація маршрутів транспортних засобів стають критичними завданнями для забезпечення швидкого та ефективного пересування населення. Протокол GTFS надає стандартизовану форму представлення даних про громадський транспорт, що спрощує їх обробку та використання.

Збір інформації з транспортних агентств передбачає розроблення системи для отримання та оновлення даних від транспортних агентств, що використовують протокол GTFS (General Transit Feed Specification) [2]. Використання цього механізму заданого протоколу дозволяє стандартизувати дані про громадський транспорт та спосіб їх отримання, доставки на сервер чи до клієнтів. Використання бази даних існує для зберігання інформації про транспортні маршрути, зупинки, розклади руху тощо. Варто використовувати технології, такі як MySQL або PostgreSQL. Проектування та розробка веб-інтерфейсу передбачає створення веб-інтерфейсу для користувачів, де вони

можуть відслідковувати рух транспортних засобів на карті. Використання Java та Spring для розробки бізнес-логіки та обробки запитів з використанням клієнт-серверних архітектурних рішень

Інтеграція з картографічними сервісами, використання сервісів, таких як Google Maps або OpenStreetMap, для відображення маршрутів та руху транспортних засобів на карті. Реалізація механізму відстеження руху транспортних засобів в реальному часі, використовуючи дані, отримані від транспортних посередників чи пристроїв.

Сповіщення користувачів та можливість надсилати сповіщення користувачам про затримки, зміни маршрутів або інші події, що стосуються громадського транспорту.

Захист інформації забезпечує безпеку та конфіденційності інформації, особливо при обробці особистих даних користувачів та транспортних фірм, підприємств.

Процес моніторингу та відладки включає механізми виявлення та виправлення можливих проблем у роботі системи. А також інтеграція з іншими сервісами надає можливість розширення функціоналу за допомогою інших сервісів чи додатків. Створення докладної документації, яка описує архітектуру, використані технології, інструкції щодо розгортання та інше.

Метою такої розробки системи моніторингу контрольованих рухомих об’єктів на карті є надання комплексного та ефективного рішення для відстеження, аналізу та керування переміщеннями об’єктів у визначеній географічній зоні на карті. Система проектується та досліджується, аби задовольнити різноманітні потреби в різних галузях промисловості та IT-індустрії, та спрямована на покращення ситуаційної обізнаності, розробки нових API, підвищити ефективність роботи та сприяти прийняттю кращих рішень в майбутніх застосунках. Основні цілі системи моніторингу включають:

1. Відстеження в реальному часі – допомагає контролювати поточне місцезнаходження керованих рухомих об'єктів у реальному часі. Забезпечує миттєву видимість руху активів, транспортних засобів або осіб.

2. Аналітика та історія переміщень – дозволяє зберігти та керувати історичними даними про рух для перегляду та аналізу. Сприяє визначенню моделей, тенденцій та історичних траєкторій.
3. Геозонування та оповіщення. Цей аспект визначає географічні межі (геозонування), щоб створити віртуальні периметри на карті. Дозволяє створювати сповіщення, коли об’єкти входять або виходять із попередньо визначених геозонованих зон, підвищуючи безпеку та операційний контроль.
4. Операційна ефективність сприяє покращеню в безпеці бізнесу та транспортних перевезень сприяє заходам безпеки, контролюючи та керуючи розташуванням активів. Негайне сповіщення про несанкціоновані переміщення або відхилення від заздалегідь визначених маршрутів, логістику та ефективність ланцюга поставок, відстежуючи рух товарів.
5. Підтримка прийняття рішень допомагає особам, які приймають рішення, даними в реальному часі та історичними даними для прийняття обґрунтованих рішень. Покращує загальну обізнаність про ситуацію, забезпечуючи повний огляд рухомих об’єктів.
6. Автентифікація користувача та контроль доступу - лише авторизовані користувачі мають доступ до конфіденційної інформації. Дозволяє керувати різними рівнями доступу на основі ролей користувачів, підтримуючи безпеку даних.
7. Відкрита та оперативна звітність – завдяки такого роду системам надаються точні та перевірені дані про рух. Система звітності існує для аналізу, перевірки відповідності та оцінки ефективності.
8. Допомога в сценаріях реагування на надзвичайні ситуації, надаючи інформацію в реальному часі про місцезнаходження активів або осіб. Дозволяє швидко приймати рішення під час кризових або несподіваних подій.
9. Настроюваний інтерфейс користувача дозволяє виокремити зручний інтерфейс із настроюваними інформаційними панелями та картами.

Дозволяє користувачам налаштовувати систему відповідно до їхніх конкретних потреб моніторингу.

1.2. Огляд існуючих рішень

В даний час існує безліч систем моніторингу керованих рухомих об'єктів на карті з використанням різних технологій. Ось декілька прикладів систем, які забезпечують такий тип моніторингу.

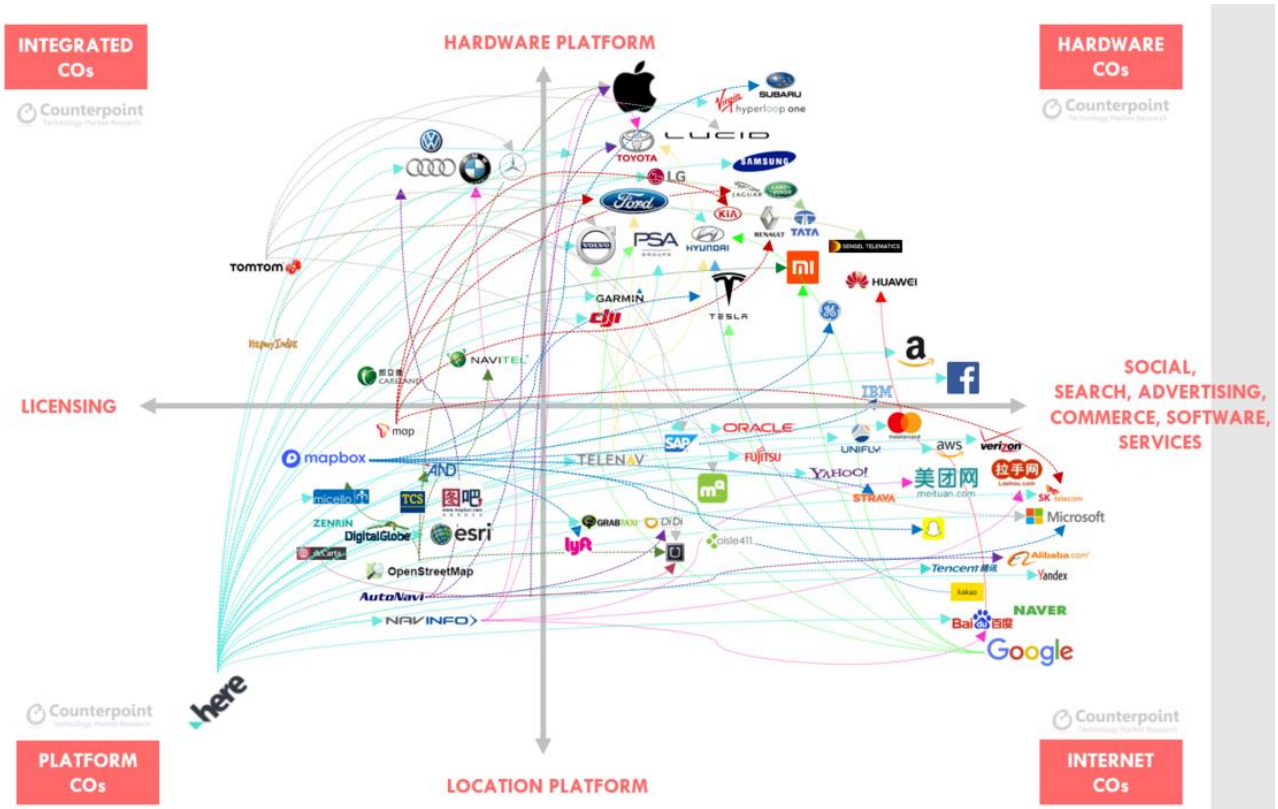

Рис. 1.1 – Схема зв’язків систем керування за рухомих об'єктів на карті з використаняням програмно-апаратних рішень світових компаній

Платформа Google Maps від корпорації Google надає API для Google Maps, який можна використовувати для створення систем моніторингу з переміщенням об’єктів у реальному часі та за історичними даними.

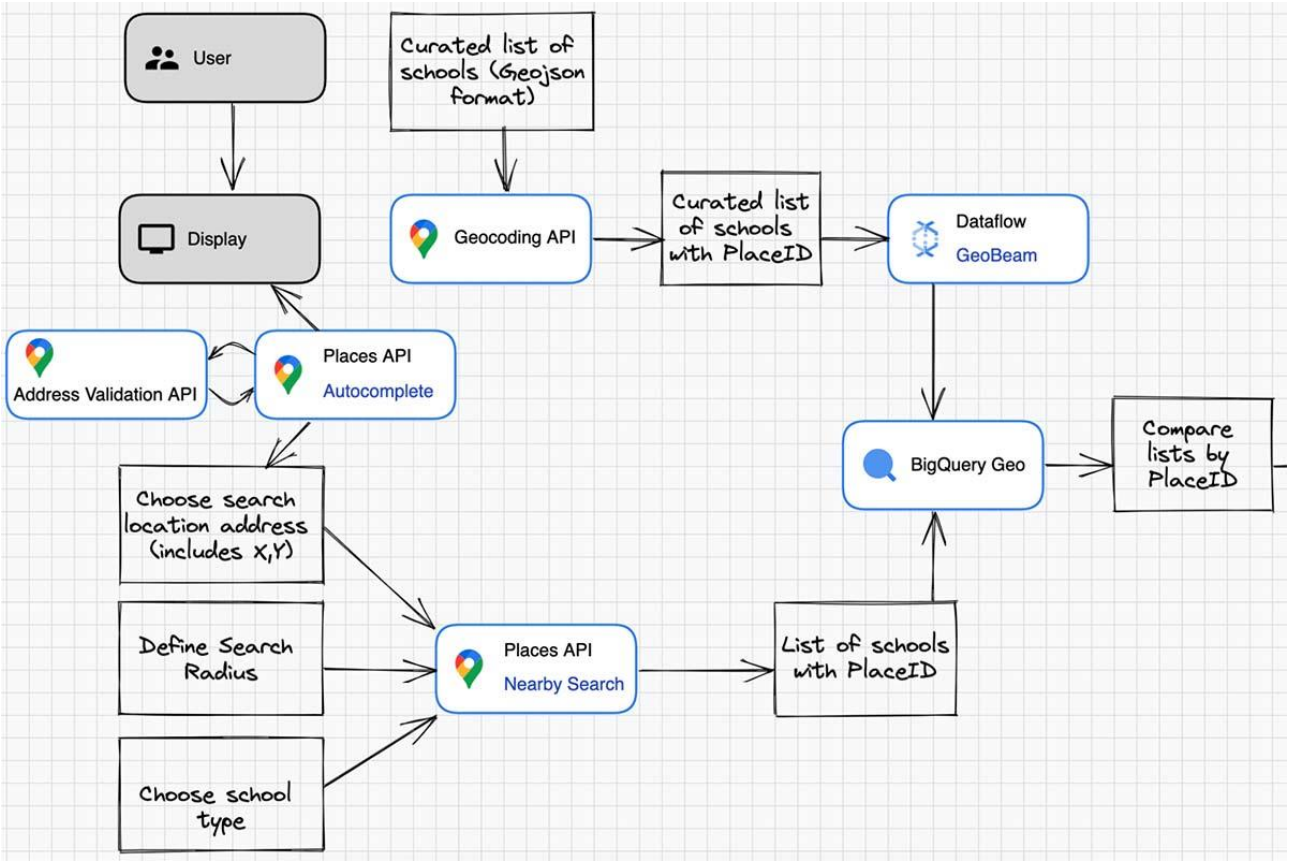

Рис. 1.1 – Схема способу використання платформи Google Maps та її сервісів для роботи з мітками шкіл в американській сфері безпеки дітей

Mapbox також надає потужний інструментарій для створення власних карт і використання їх у системах моніторингу.

Leaflet — це легкий інтерактивний інтерфейс JavaScript для відображення карт. Його можна використовувати для створення систем моніторингу.

ArcGIS від Esri — це геопросторова платформа, яка дозволяє створювати передові системи моніторингу за допомогою геопросторової аналітики.

OpenStreetMap (OSM) надає безкоштовно доступні картографічні дані, які можна використовувати для створення систем моніторингу.

Географічна вкладка: Geotab пропонує рішення для моніторингу транспортних засобів і водіїв за допомогою GPS та інших технологій.

Wialon має функціональність для моніторингу руху транспортних засобів, включаючи геозонування та відстеження маршруту.

Navixy — це платформа моніторингу транспортних засобів і об’єктів, яка пропонує різноманітні функції відстеження та аналізу руху.

Traccar — це програмне забезпечення GPS-відстеження з відкритим кодом, яке можна використовувати для створення систем моніторингу.

Ці системи використовуються в різних галузях, таких як транспорт і логістика, охорона, служби реагування на надзвичайні ситуації та інші, для відстеження переміщення об’єктів на карті з метою оптимізації процесів і прийняття обґрунтованих рішень.

Рис. 1.3 – Схема функціонування системи керування за рухомих об'єктів на карті з використаняням програмно-апаратних рішень

Упродовж першого півріччя 2023 року Google і TomTom залишаються провідними у сфері ефективності екосистеми розташування. Ринок стрімко розвивається, і ключові компанії використовують аналітику даних та моделі відкритого партнерства для забезпечення реального часу геолокаційної інтелектуальної розвідки. Компанії, такі як Mapbox та Telenav, швидко висувають конкурентні пропозиції, викликаючи традиційні платформи.

Платформи розташування стають високомодульними, оскільки картографічні компанії використовують хмарну інфраструктуру, передові інструменти розробки та моделі відкритого партнерства. Галузь переживає

значні зміни, оскільки постачальники карт переходять до платформенного підходу, фокусуючись не лише на продажу карт і навігаційних рішень для автомобільних виробників і пристроїв. Нові сервіси та можливості виникають завдяки інтеграції геолокаційних даних із наявними картографічними даними, а також використанню аналітичних механізмів для обробки потоків геолокаційних даних у реальному часі. Платформи визначення місцезнаходження нового покоління формуються шляхом об'єднання модулів, що містять багато данних про картографію, інформацію про місцезнаходження, аналітику та різноманітні сервіси.

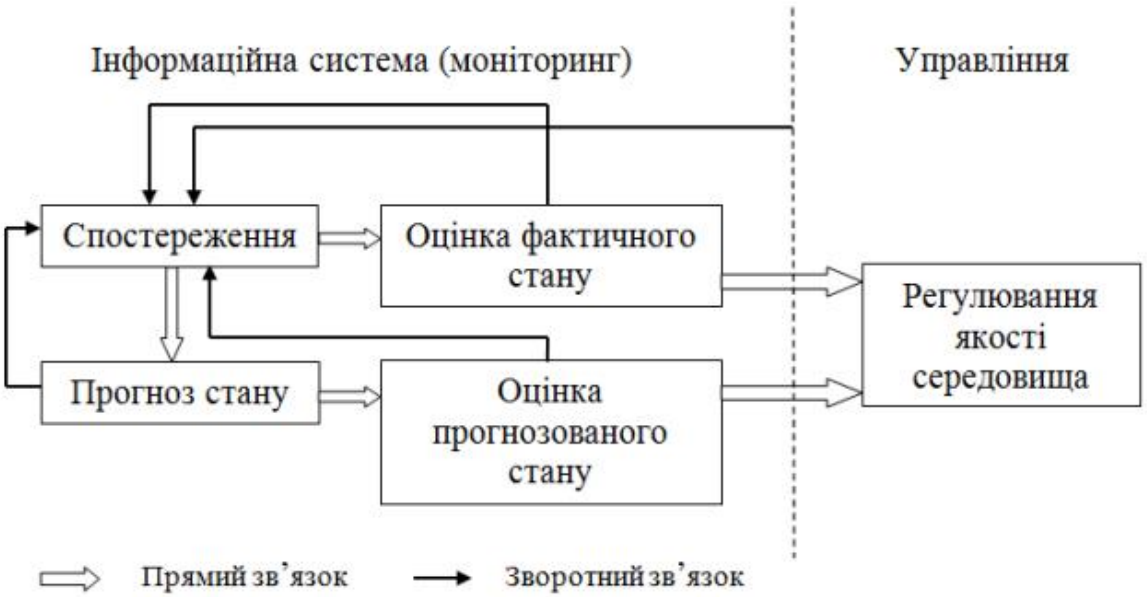

Рис. 1.4 – Блок-схема функціонування процесів в системах моніторингу

Моніторинг за керованими рухомими об'єктами на карті, реалізований з використанням технологій Java, може мати різні цілі, в залежності від конкретного контексту і вимог застосування. Ось кілька потенційних цілей для такого моніторингу:

Реальний час: забезпечення миттєвого відображення рухомих об'єктів на карті. Можливість відслідковувати зміни місцезнаходження в реальному часі.

Історичний аналіз: збір і зберігання історичних даних про рух об'єктів для подальшого аналізу. Вивчення маршрутів, зупинок, інтервалів руху та інших паттернів.

Геофенсинг та Алерти: встановлення віртуальних периметрів (геофенсів) на карті. Сповіщення адміністраторів або користувачів при входженні або виходженні об'єктів з геофенсів.

Управління маршрутами: відстеження відповідності руху об'єктів заданим маршрутам. Сповіщення про відхилення від маршруту.

Оптимізація: планування оптимальних маршрутів для зменшення витрат пального та збільшення продуктивності. Моніторинг часу простою і робочих годин об'єктів.

Безпека та захист: виявлення несанкціонованого руху об'єктів або випадків крадіжок, негайне сповіщення про підозрілі події.

Аналітичні можливості: забезпечення користувачів засобами аналізу даних, зокрема, статистичними звітами і візуалізацією.

Інтеграція з Іншими Системами: можливість інтеграції з іншими корпоративними системами, такими як системи управління флотом, CRM, ERP тощо.

Масштабованість: забезпечення системою моніторингу масштабованості для високоефективної роботи при збільшенні обсягу даних та користувачів.

Ці цілі можуть слугувати основою для розробки та вдосконалення системи моніторингу за керованими рухомими об'єктами на карті, реалізованої з використанням Java-технологій..

1.3. Спосіб роботи протоколу GTFS

GTFS (General Transit Feed Specification) - це відкритий формат даних, розроблений для обміну інформацією про громадський транспорт. Цей стандарт дозволяє транспортним агентствам оприлюднювати розклади громадського транспорту та пов'язані дані, такі як маршрути, зупинки та географічні

координати. GTFS сприяє розвитку та впровадженню сервісів маршрутних планерів, додатків для мобільних пристроїв та інших інструментів для користувачів громадського транспорту.

Основні компоненти GTFS включають:

- agency.txt: Інформація про транспортні агентства, які надають послуги.
- stops.txt: Інформація про зупинки громадського транспорту.
- routes.txt: Інформація про маршрути, які обслуговуються транспортним агентством.
- trips.txt: Інформація про конкретні поїздки, які виконуються на певних маршрутах.
- stop_times.txt: Розклади прибуття та відправлення на конкретних зупинках для конкретних поїздок.
- calendar.txt і calendar_dates.txt: Інформація про розклади руху в залежності від календаря та виняткових дат.
- fare_attributes.txt і fare_rules.txt: Інформація про тарифи та правила їх застосування.

GTFS є важливим інструментом для розробників та сервісів, які працюють з громадським транспортом, оскільки він дозволяє стандартизувати та спрощувати обмін даними між різними системами.

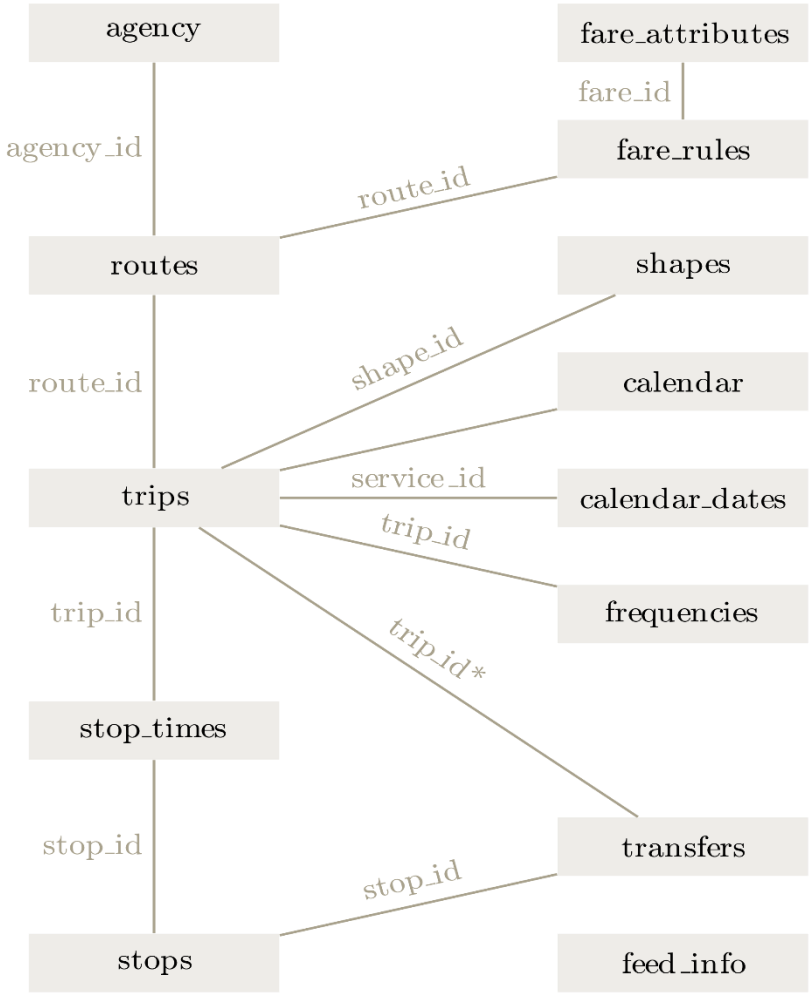

Рис. 1.5 – Блок-схема функціонування сервісів в системах моніторингу з використанням протколу GTFS

Система на основі мобільного пристрою для класифікації виду транспорту під час подорожі включає мобільний пристрій, включаючи систему визначення місцезнаходження та акселерометр. Мобільний пристрій налаштований на збір даних про місцезнаходження та прискорення під час подорожі. Система також включає в себе блок обробки даних, налаштований на отримання даних про місцезнаходження від мобільного пристрою, отримання контекстних даних, пов’язаних з безліччю транспортних систем, і обробки даних про місцезнаходження та щонайменше частини контекстних даних за допомогою першого класифікатора режиму транспортування. пов'язаний з першою з безлічі транспортних систем і щонайменше другою частиною контекстних даних з

використанням другого класифікатора режиму транспортування, пов'язаного з другою із безлічі транспортних систем. Блок обробки даних також налаштований на класифікацію виду транспорту під час поїздки та оновлення класифікаторів на основі введення користувача.

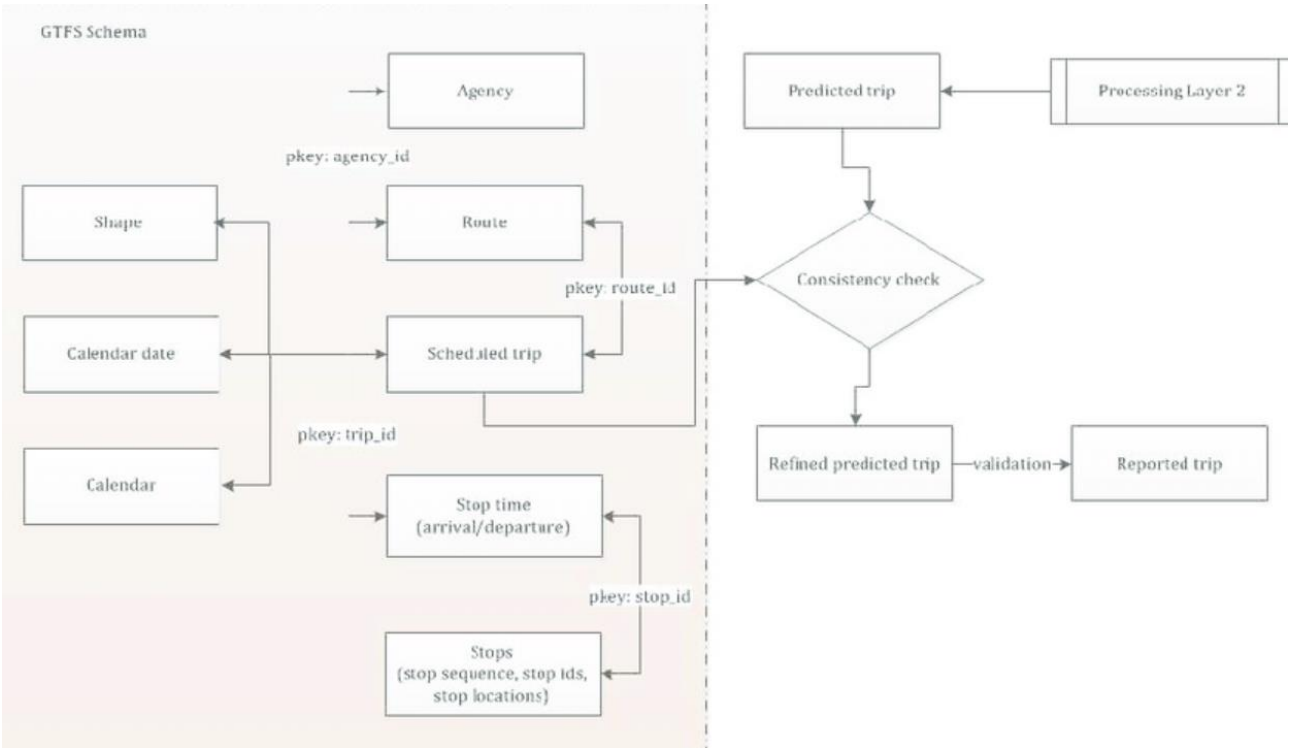

Рис. 1.5 – Потоки подій в протоколі GTFS

1.4. Значення та важливість системи керування та спосіб розподілу частот транзитних послуг у ній

Метод динамічного розподілу налаштувань частоти транзитної послуги включає використання AVL/APC для визначення часу подорожі та коливань попиту протягом дня. На основі цього формуються кластери періодів часу, і день розбивається. Для кожного періоду часу, для якого буде призначено нове налаштування частоти, розраховуються діапазони розподілу частоти, у межах

яких час очікування на зупинках для мультимодальних пересадок скорочується, і розподіл частоти вибирається за критеріями, включаючи покриття попиту пасажирів і скорочення експлуатаційних витрат. Множину рішень налаштування частоти обчислюють за допомогою підходу розгалуження та зв’язку з послідовним квадратичним програмуванням (SQP) або послідовного генетичного алгоритму з штрафом зовнішньої точки. Чутливість рішень налаштування частоти перевіряється, щоб визначити найбільш надійне рішення налаштування частоти для нового налаштування, і розклад транзитної служби оновлюється відповідно.

1.7. Постановка задачі

Задачі для тематики та розробка системи моніторингу за керованими рухомими об'єктами на карті включають аналіз вимог до системи, проектування бази даних для ефективного відстеження рухомих об'єктів, розробку API для взаємодії із системою, створення інтерфейсу користувача для зручного взаємодії, впровадження механізмів візуалізації даних на мапі, реалізацію геофенсінгу для встановлення зон моніторингу, налагодження системи сповіщень, розробку аналітичних інструментів для обробки даних та можливість інтеграції з іншими системами, ураховуючи аспекти безпеки, тестування та оптимізацію для забезпечення ефективності системи.

2 ПРОЕКТУВАННЯ АРХІТЕКТУРИ ТА МОДЕЛЮВАННЯ ПРОГРАМНОГО РІШЕННЯ

2.1. Аналіз предметної області та виявлення вимог

2.1.1. Аналіз вимог до реалізації та процесу розробки

Функціональні вимоги до розробки системи моніторингу за керованими рухомими об'єктами на карті можна розглядати в контексті основних функціональних модулів, які визначають її можливості для користувачів та підсистеми, з якими вона інтегрується. Ось огляд проектованих функціональних вимог:

Функціональні вимоги до розробки системи моніторингу рухомих об'єктів на карті визначають набір ключових функціональностей, необхідних для ефективної роботи системи. Ось кілька основних функціональних вимог:

1. Реєстрація та Авторизація: можливість користувачам реєструватися в системі. Авторизація за допомогою унікальних ідентифікаторів та паролів.

2. Додавання та Видалення Об'єктів. Можливість додавати нові рухомі об'єкти для моніторингу. Функціонал для вилучення об'єктів з системи.

3. Реальний Час та Історія Руху. Відображення поточного руху об'єктів на карті в реальному часі. Можливість переглядати історію руху для аналізу.

4. Геофенсинг та Сповіщення. Встановлення географічних областей для моніторингу. Автоматичні сповіщення при подіях, таких як виходження об'єктів за межі встановлених зон.

5. Інтерактивна Карта та Візуалізація - відображення об'єктів на інтерактивній карті. Графічна візуалізація шляху руху об'єктів та їх характеристик.

6. Статистичні Звіти та Аналітика. Автоматична генерація статистичних звітів на основі руху об'єктів та можливість проведення аналітичних операцій для отримання детальної інформації.

7. Керування Доступом та Авторизація. Налаштування прав доступу для адміністраторів та звичайних користувачів. Обмеження доступу до конфіденційної інформації відповідно до рівнів доступу.

8. Автоматизована Система Повідомлень, а саме: надсилання автоматичних повідомлень користувачам або адміністраторам, інтеграція з різними каналами сповіщень.

9. Інтеграція з Іншими Системами. можливість взаємодії з іншими системами, такими як системи управління флотом чи корпоративні бази даних.

Додатково, корисними будуть Інструменти: аналітичні:

- Надання інструментів для аналізу та вивчення зібраних даних.

- Підтримка порівняння різних об'єктів та їх характеристик.

Ці функціональні вимоги визначають ключові можливості системи моніторингу, які спрямовані на забезпечення надійного та ефективного відстеження рухомих об'єктів на карті.

До нефункціональних вимоги віднесено:

Надійність:

- Забезпечення зручного та адаптивного інтерфейсу для користувачів на різних пристроях.
- Забезпечення доступності системи протягом робочого дня та 24/7 для адміністраторів.

Продуктивність:

- Здатність підтримувати не менше 30 одночасно працюючих користувачів.
- Максимальний час відгуку системи: не більше 0.5 секунд для типових завдань та 1.5 секунд для більш складних.

Придатність до експлуатації:

- Масштабованість системи для підтримки зростання кількості користувачів.
- Забезпечення автоматизованого оновлення версій програми через систему контролю версій.

2.1.2. Вибір та обгрунтування архітектурної моделі системи

За основу архітектурногї моделі предсталвення системи взято відому та поширену в наш час багатошарову архітектуру типу «Клієнт-Сервер».

Архітектура клієнт-сервер реалізує режим роботи, спрямований на багато користувачів, і є розподіленою, коли клієнти та сервери розташовані на відмінних вузлах локальної чи глобальної обчислювальної мережі. На малюнку 1 наведено приклад локальної мережі із єдиним сервером.

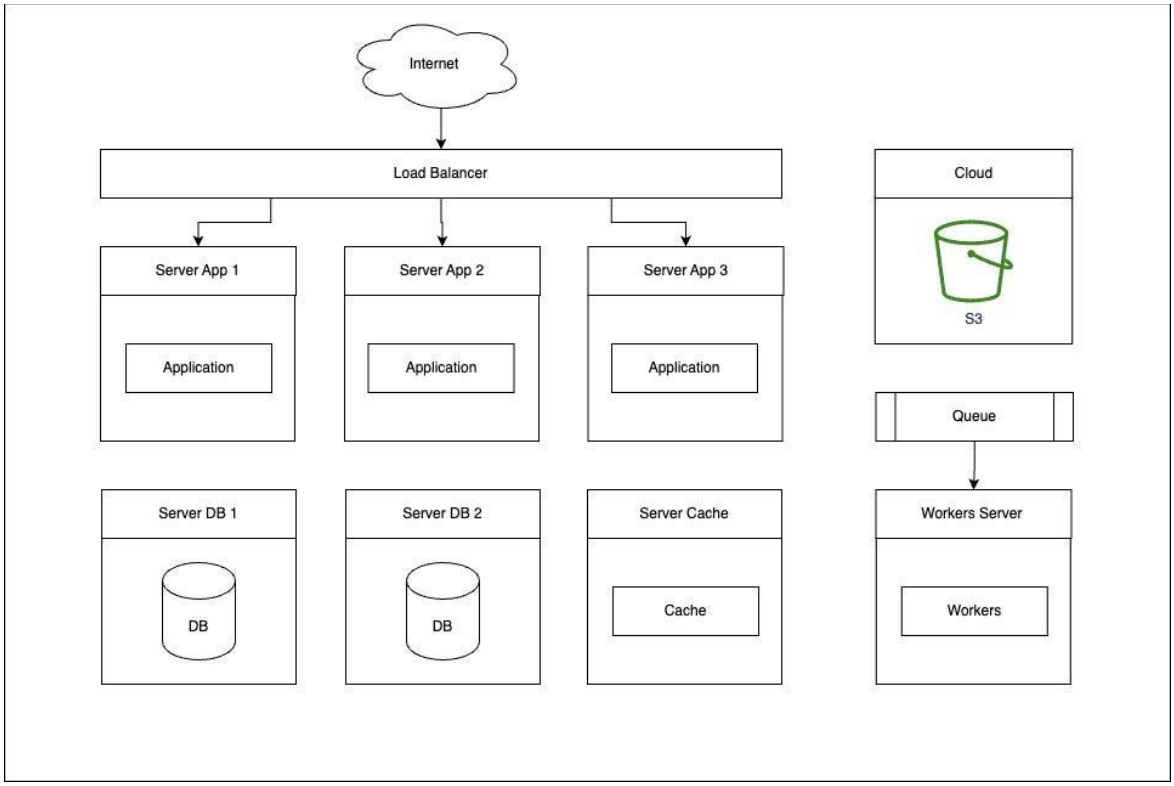

Рис. 2.1 – Представлення багаторівневої архітектури «Клієнт-сервер»

Структура системи, організованої за модульним принципом, є необхідною умовою для втілення принципів відкритості та розподіленості. Ця організація забезпечує характеристики розподілених даних, процесів та управління. Концепція ґрунтується на поняттях об'єкту та об'єктного простору, які знаходять своє відображення в архітектурі КЕІС-платформи. Загалом, схема клієнт-серверної архітектури включає три рівні представлення.

Перевага локальної мережі перед централізованою обчислювальною системою полягає в можливості відкритого доступу та використання обчислювальних ресурсів через єдине передавальне середовище, без обмежень, пов'язаних з раніше визначеними принципами взаємодії обчислювального обладнання. Іншими словами, це означає легкість масштабування системи КЕІС (розподіленої комплексної системи, де об'єкти розташовані у вигляді окремих модулів на різних вузлах мережі).

Системи моніторингу з використаням клієнт-серверною архітектурою використовуються для збору, аналізу та відображення інформації про стан різних систем, мереж і процесів. Такі системи можуть бути застосовані в різних сферах, включаючи IT-інфраструктуру, виробничі процеси, мережі зв'язку, електроенергетику та багато інших областей.

Основні елементи клієнт-серверної архітектури для систем моніторингу включають:

- Серверна частина (backend): Сервер відповідає за збір, обробку та збереження даних, які надходять від різних джерел моніторингу. Він також може виконувати аналіз даних для виявлення аномалій або попередження можливих проблем.
- Клієнтська частина (frontend): Клієнтська частина представляє інтерфейс для користувача. Це може бути веб-інтерфейс, десктопний додаток або мобільний додаток, який дозволяє користувачам взаємодіяти з системою моніторингу, переглядати дані та отримувати повідомлення.

- Протоколи комунікації: Сервер та клієнт взаємодіють за допомогою певних протоколів комунікації, таких як HTTP або WebSocket. Це забезпечує передачу даних між клієнтом та сервером в реальному часі.
- База даних: Система моніторингу може використовувати базу даних для збереження та організації отриманих даних. Це дозволяє ефективно здійснювати пошук і аналіз історичних даних.
- Модулі моніторингу: Це компоненти, які забезпечують збір конкретних видів даних, таких як використання CPU, стан мережі, робота серверів тощо.
- Механізми сповіщення: Система може мати вбудовані механізми сповіщення, які попереджають адміністраторів про можливі проблеми або аномалії за допомогою електронних листів, SMS або інших каналів.

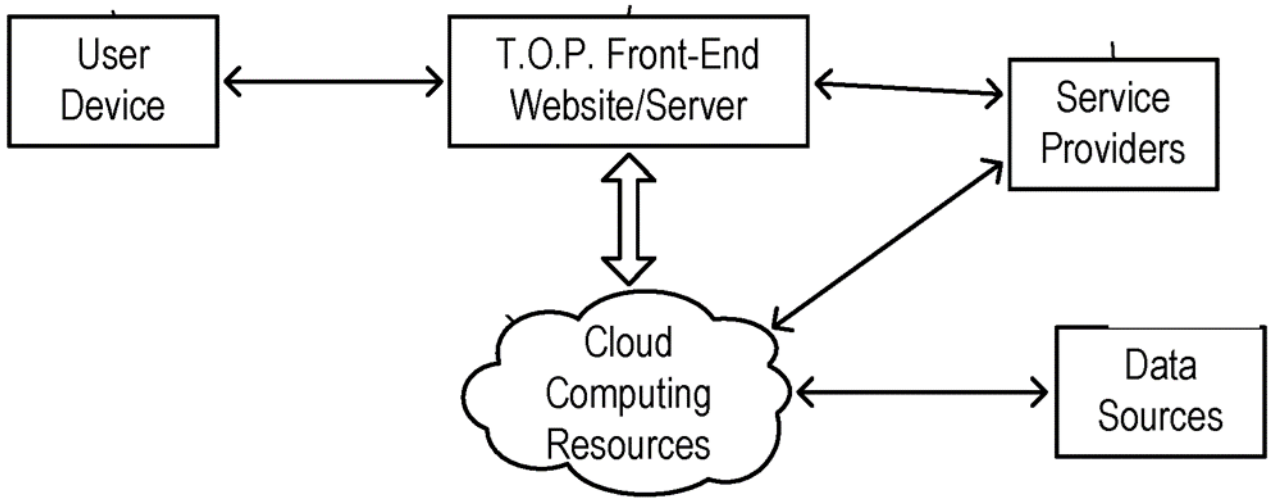

Рис. 2.2 – Представлення архітектури системи з огляду сервісів

В загальному, схема клієнт-серверної архітектури включає три рівні представлення: рівень, на якому дані взаємодіють з користувачем; рівень обробки даних додатком; і рівень взаємодії з базою даних. За цією схемою користувач (клієнт) може вводити дані, які після контролю і перетворення застосуванням, потрапляють в базу даних. В інших випадках користувач може

запитувати обробку даних від додатку, який отримує необхідні дані з бази даних. Після отримання цих даних, сервер обробляє їх та результати можуть бути збережені в базі даних або повернуті користувачеві у зручному для нього форматі, такому як текстовий документ, електронна таблиця, графік тощо.

Реалізація клієнт-серверної архітектури в обчислювальній мережі може бути різноманітною. Вибір конкретної схеми залежить від різних факторів, таких як територіальний розподіл видалених підрозділів підприємства, вимоги до експлуатаційної надійності, швидкодії та простоти обслуговування.

Використання клієнт-серверної архітектури дозволяє забезпечити ефективний та розподілений збір та аналіз даних моніторингу, забезпечуючи швидкий доступ та відображення результатів для користувачів.

2.1.3. Обґрунтування вибору медотології для підходу виконання проекту

Проектування процесу розробки згілно методології - це опис процесу створення та підтримки системи на протязі їхнього життєвого циклу, розглядаючи послідовність етапів та відповідні процеси, які виконуються на кожному з них. Для кожного етапу визначається склад та порядок виконання робіт, остаточні результати, методи та засоби, необхідні для їхнього виконання, а також ролі та відповідальність учасників. Опис життєвого циклу інформаційної системи надає змогу спланувати та організувати процес колективної розробки та забезпечити ефективне управління ним.

Життєвий цикл системи можна розглядати як послідовність подій, що відбуваються під час її створення та використання. Відображаючи різні стани системи, він розпочинається з моменту, коли виникає ідея необхідності конкретної розробки і завершується припиненням використання цієї системи.

Існують різні методології управління життєвим циклом систем, які визначають послідовність та спосіб виконання етапів у процесі розробки. Один

із значущих підходів до управління життєвим циклом інформаційних систем — це Rational Unified Process (RUP), що означає Раціональний Об'єднаний Процес. Вибір саме цієї методології для даного проекту є обґрунтованим, оскільки вона використовує ітеративну модель розробки.

Методологія RUP визначає абстрактний загальний процес, який служить основою для того, щоб організація чи проектна команда могли розробити спеціалізований процес, відповідний своїм потребам.

Однією з особливостей цієї методології є акцент на розробці та документуванні вимог. Для опису вимог у RUP використовуються прецеденти використання (use cases). Повний набір таких прецедентів використання системи разом із логічними відносинами між ними утворює модель прецедентів використання.

Кожен прецедент використання представляє собою опис сценаріїв, які показують, як користувач взаємодіє з системою для виконання конкретного завдання. Відповідно до методології RUP, всі функціональні вимоги повинні бути висвітлені у формі прецедентів використання.

RUP визначається як методологія, орієнтована на архітектуру. Зазначається, що реалізація та тестування архітектури системи повинні розпочинатися на ранніх етапах проекту. Методологія використовує термін "виконувана архітектура" (executable architecture) – це основа програми, яка дозволяє втілити архітектурно значущі прецеденти використання. Зазначається, що ці основи виконуваної архітектури мають бути реалізовані якнайшвидше. Це дозволяє оцінити адекватність прийнятих архітектурних рішень та внести необхідні корективи ще на ранніх етапах проекту.

Життєвий цикл проекту RUP складається з чотирьох фаз (див. Рис. 1.2):

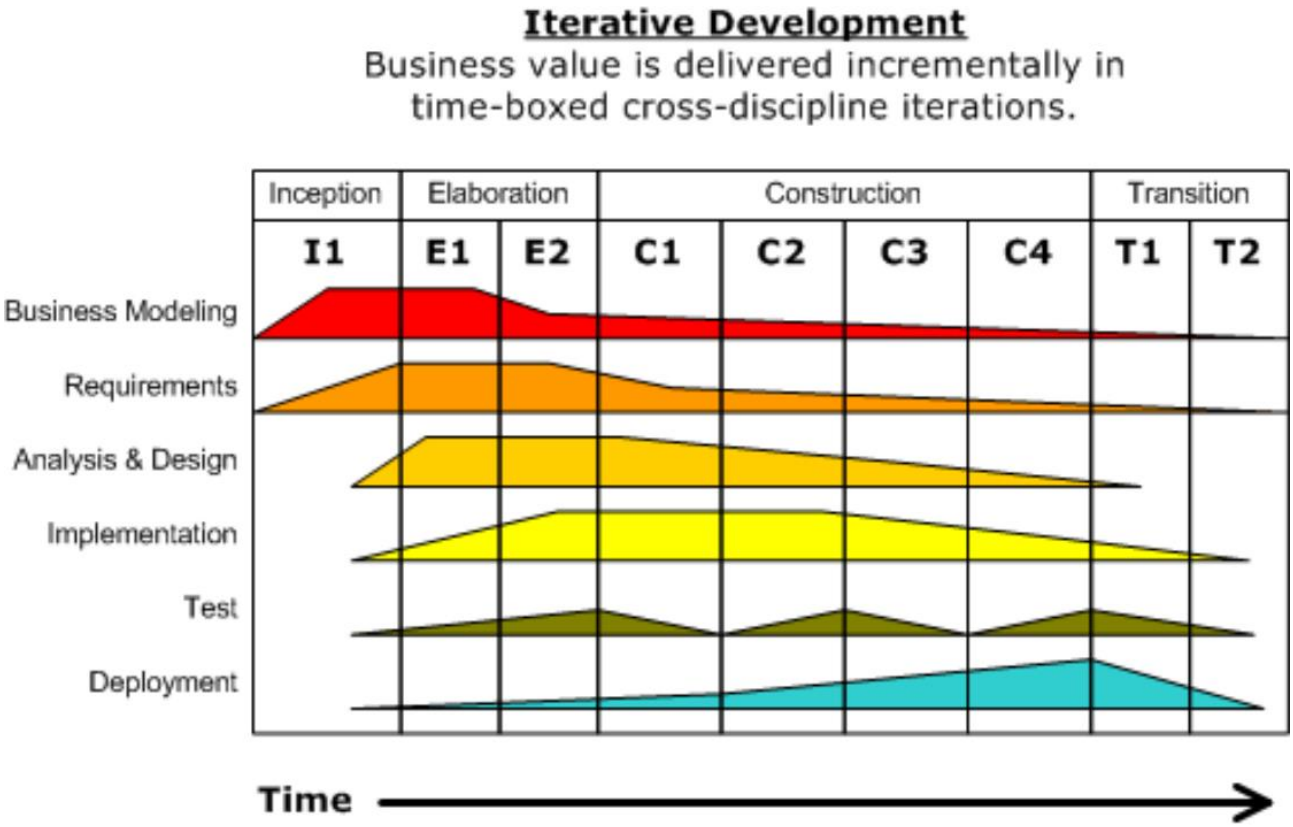

Рис. 1.2 – Фази та етапи проектування проєкту згідно RUP

Початок (Inception): під час цієї стадії важливо визначити бачення та межі проекту, розробити економічне обґрунтування, ідентифікувати більшість прецедентів використання та провести оцінку бюджету, графіку та ризиків проекту.

Проектування (Elaboration): на цьому етапі, на основі вимог та ризиків проекту, створюється основа архітектури системи. Головними завданнями є детальний опис більшості прецедентів використання, створення протестованої базової архітектури (з використанням архітектурно значущих прецедентів використання) та зниження ключових ризиків.

Побудова (Construction): на цьому етапі відбувається розробка кінцевого продукту. Проводиться створення основної частини вихідного коду системи, а також випуск проміжних демонстраційних прототипів.

Впровадження (Transition): ця остання фаза включає проведення бета-тестування та усунення виявлених дефектів, при необхідності – міграцію даних. Крім того, на цьому етапі виконуються завдання, необхідні для маркетингу та

продажу. Після завершення фази проводиться аналіз результатів виконання всього проекту.

2.2. Обгрунтування вибору середовища розробки програмної системи та мови програмування

Створення веб-додатків стало необхідністю для підприємств різних розмірів та спрямованостей. Незалежно від ціннісних пропозицій підприємства, вони виявляють потребу у власному веб-присутності. В сучасному світі існує кілька підходів до розробки веб-додатків, проте одним з найпопулярніших є використання фреймворків.

Java, яка є мовою програмування загального призначення, може підтримувати розробку різноманітних програм. У цьому контексті Spring можна охарактеризувати як екосистему продуктів, яка створена для допомоги розробникам Java. Ця екосистема представляє собою обширний контейнер, що включає ключові компоненти, такі як Spring Framework, Spring Data JPA, Spring Security, Spring JDBC, Spring Boot та інші. Використання Spring дозволяє розробникам спростити та поліпшити процес створення веб-додатків на мові Java, забезпечуючи високий рівень функціональності та продуктивності.

Spring був створений для спрощення початкового етапу розробки та економії часу. Цей інструмент має значні переваги, включаючи:

- Зменшення часу, необхідного для розробки, і підвищення ефективності розробницької команди.

- Вбудовані HTTP-сервери, такі як Jetty і Tomcat, в комплекті для тестування веб-додатків.

- Автоматична настройка всіх компонентів готової програми Spring.

- Налаштування за замовчуванням для модульних та інтеграційних тестів, що спрощує розробку та тестування Java-програм.

- Усунення потреби у трудомісткому написанні шаблонного коду, анотацій та обширних налаштувань XML.

- Легка інтеграція Spring Boot з іншими елементами екосистеми Spring.

- Розширений вибір плагінів для співпраці з вбудованими базами даних і базами даних у пам'яті.

- Проста і зручна інтеграція з різними службами баз даних і чергами, такими як Oracle, MySQL, MongoDB і ActiveMQ.

- Адміністративна можливість керування програмою за допомогою віддаленого доступу.

- Швидкий доступ до інтерфейсу командного рядка для прискорення тестування та розробки програм Spring Boot.

Щодо середовища розробки, я віддаю перевагу використанню Intellij Idea. Це середовище, яке я використовую протягом кількох років, і для мене воно є дуже зручним. Просто кажучи, IDE є зручним інструментом програмного забезпечення, який поєднує у собі текстовий редактор, налагоджувач і компілятор [10]. Деякі IDE також дозволяють встановлювати різноманітні плагіни, розширюючи його можливості.

2.3. Проектування програмної системи

2.3.1. Пошук акторів та варіантів використання

Для розробки системи обліку інвентарю фітнес-центру за методологією RUP ми використовуватимемо інструменти проектування, зокрема універсальну мову моделювання UML.

UML є мовою графічного опису, призначеною для об'єктного моделювання у галузі розробки програмного забезпечення, детального моделювання бізнес-процесів, системного проектування та відображення організаційних структур. В основі цієї технології формується єдина інформаційна модель. UML забезпечує підтримку етапів життєвого циклу інформаційної системи та для цього пропонує графічні інструменти – діаграми.

У стадії створення концептуальної моделі для опису діяльності використовується діаграма прецедентів.

Перед початком безпосередньої роботи на програмним забезпеченням необхідно визначити акторів та варіанти використання системи. Для демонстрації даних сценаріїв можна скористатися UML діаграмами. UML діаграма – це графічний інструмент для моделювання систем та їх компонентів, який надає структуроване представлення взаємодій між різними частинами системи.

Визначившись із основними акторами та варіантами використання варто зазначити потік роботи основних варіантів використання нашої системи. Всі основні та альтернативні варіанти використання описані далі.

При аналізі предметної області я визначив таких акторів:

- Звичайний користувач;
- Адміністратор системи;
- Менеджер компанії (експерт області логістики).

Враховуючи раніше описані функціональні вимоги до системи, вартовкзаати початкову діаграму прецедентів системи моніторингу на рисунку.

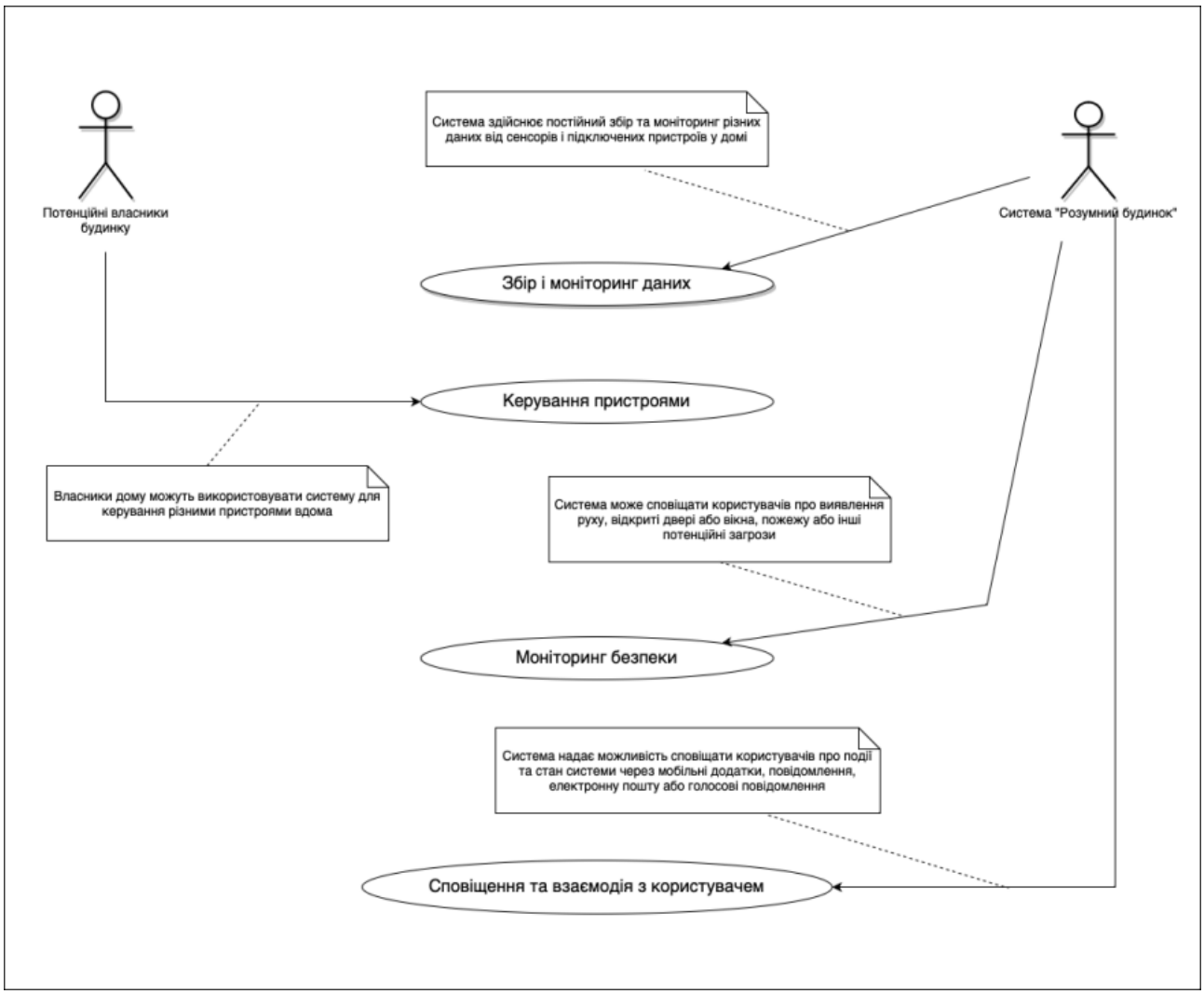

Рис. 2.2 – UML-діаграма варіанти використання системи

Unified Modeling Language надає стандартний набір графічних символів та правил для створення діаграм, що дозволяє спростити розуміння та спілкування щодо архітектури та функціональності системи.

2.3 Побудова архітектури класів та підсистем

З використанням усієї раніше дослідженої та представленої інформації, ми можемо визначити класи, якими буде володіти наша система обліку інвентарю і побудувати діаграму класів.

Діаграми класів UML надають комплексне візуальне уявлення статичної структури системи, визначаючи компоненти класів, їх атрибути та методи, а також складну мережу взаємозв'язків. У цьому контексті клас візуально представлений прямокутником, розділеним на три секції, які включають назву класу, атрибути та методи.

Атрибути класу є важливими компонентами, які інкапсулюють внутрішні дані або поля, що характеризують екземпляри цього класу. Ці атрибути відображаються в другій секції прямокутника класу, де їх назви та відповідні типи даних надають уявлення про основні властивості, що визначають стан класу.

Узагальнено можна сказати, що діаграма класів UML виступає потужним інструментом для розробників та учасників проекту, надаючи чітку та стандартизовану візуальну мову для розуміння, проектування та передачі статичної структури системи. Однак, для створення такої діаграми слід визначити класи, які входять до складу системи.

Клас Equipment визначає представлення обладнання у системі, з яким працюю трекінг. У кожному екземплярі цього класу міститься унікальний ідентифікатор (id), назва (name) та кількість одиниць (quantity) обладнання. Стан кожного елемента обладнання (condition) може бути визначений як "робочий," "потребує ремонту" або "відсутній." Для кожного обладнання також зберігається інформація про дату його додавання до системи (dateAdded).

Клас User представляє користувачів у системі обліку підключеного обладнання. У кожного користувача є унікальний ідентифікатор (id), ім'я (name),

роль (role), яка визначає його функціональні можливості, і логічне значення accessSystem, що вказує на його доступ до системи. Методи цього класу User надають засоби взаємодії та управління системою користувачів у трекінг-системі.

У системі зв’язності використовується клас "Category" для класифікації та групування обладнання за певними критеріями чи категоріями. Цей клас сприяє ефективному структуруванню інвентарю з метою зручного управління та проведення аналізу.

Після детального опису всіх класів системи та їхніх взаємозв'язків на рівні екземплярів, можна створити діаграму класів, яка буде використовуватися в подальшій розробці.

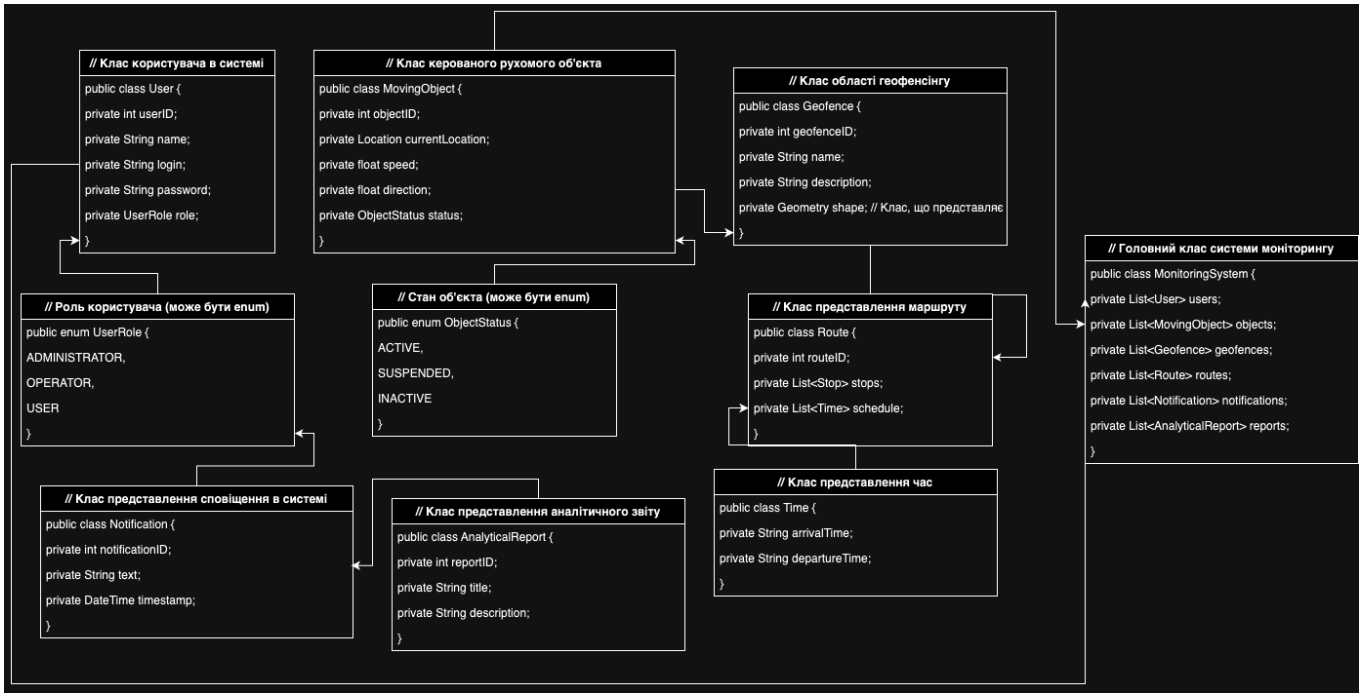

Рис. 2.5 – Діаграма класів проектованої системи клієнтського доступу до геоданих

Ці класи представляють основні сутності та їх взаємозв'язки в системі моніторингу за керованими рухомими об'єктами на карті. У реальному проекті може бути необхідно додати додаткові атрибути та методи для вирішення конкретних вимог вашої системи.

2.4 Використання GTFS (Загальна специфікація транзитного каналу)

Щоб реалізувати використання GTFS (Загальна специфікація транзитного каналу) у програмі Java Spring, виконайте ці загальні дії. GTFS — це стандарт для представлення даних громадського транспорту, і включення його в програму Java Spring може надати цінну інформацію для відстеження та моніторингу систем громадського транспорту.

1) Залежності. Підєдняння бібліотеки відбувається шляхом підклбючення у свій файл pom.xml необхідні залежності для роботи з даними GTFS. Для цього підходить бібліотека, яка може аналізувати та обробляти канали GTFS. Однією з таких бібліотек є onebusaway-gtfs, яку ми включаємо у проект Maven наступним чином:

Лістинг 2.1. Підєдняння бібліотеки:

```
@Service
<dependency>
     <groupId>org.onebusaway</groupId>
          <artifactId>onebusaway-gtfs</artifactId>
     <version>1.3.4</version>
</dependency>more methods for accessing other GTFS data as
needed
}
```

2) Конфігурація. Здійснюється налаштування програми на Spring для обробки даних GTFS. Визначаємо клас конфігурації, де ви можете налаштувати необхідні компоненти.

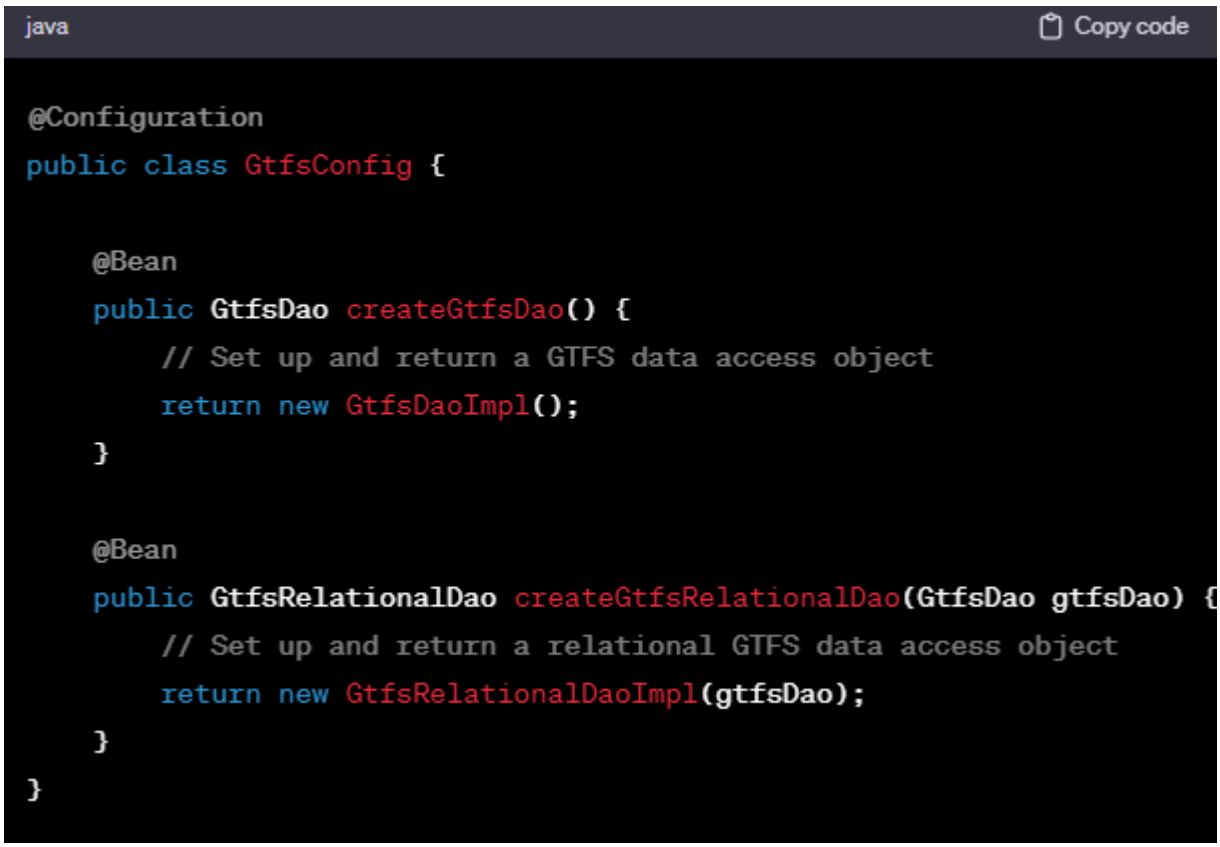

Рис. 2.3 – Під'єднання бібліотеки для Spring-проєкту

3) Сервіс для операцій GTFS. Створюємо клас обслуговування для обробки операцій, пов'язаних із GTFS, наприклад читання даних із каналів GTFS.

Лістинг 2.2. Сервіс під'єднання операцій:

```
@Service
public class GtfsService {
    @Autowired
    private GtfsRelationalDao gtfsRelationalDao;

    public List<Agency> getAllAgencies() {
        // Retrieve a list of all agencies from the GTFS feed
        return gtfsRelationalDao.getAllAgencies();
    }     // Add more methods for accessing other GTFS data as
needed
}
```

4) Контролер для веб-взаємодій. Розробляємо контролер для надання функцій, пов’язаних із GTFS, задля веб-взаємодії.

Лістинг 2.3. Контролер GTFS для веб-взаємодій

```
@RestController
@RequestMapping("/gtfs")
public class GtfsController {

    @Autowired
    private GtfsService gtfsService;

    @GetMapping("/agencies")
    public ResponseEntity<List<Agency>> getAllAgencies() {
        // Return a list of all agencies as JSON
        List<Agency> agencies = gtfsService.getAllAgencies();
        return new ResponseEntity<>(agencies, HttpStatus.OK);
    }

    // Add more endpoints for other GTFS-related operations
}
```

5. Інтеграція з Картами та Frontend. Інтеграція відбувається з даними GTFS із картографічними службами (наприклад, Google Maps) і опісля розробляються інтерфейсні компоненти для візуалізації та взаємодії з даними.

6. Оновлення. Якщо потрібні оновлення в режимі реального часу, використовуються інтеграції механізму отримання й обробки даних у режимі реального часу від логістичних агентств та датчиків.

7. Обробка та журналювання помилок. Використовуємо належну обробку помилок і журналювання, щоб забезпечити надійність і зручність обслуговування нашого модуля програми.

Виконуючи ці кроки, ми можемо включити дані GTFS у свою програму Java Spring, дозволяючи створювати функції, пов’язані з моніторингом і відстеженням громадського транспорту. Також, варто налаштувати реалізацію відповідно до конкретного випадку використання та вимог.

2.5 Система підтримки прийняття рішень

Громадські транспортні агентства є справжніми виробниками великого обсягу даних, завдяки високій частоті та швидкості їхньої діяльності. Одним із наборів даних, що належить до цього виду і є відкритим для громадськості, є Загальна специфікація транзитного каналу (GTFS). Ці дані GTFS включають в себе інформацію про заплановані транспортні послуги, такі як місця зупинок, маршрути та розклад.

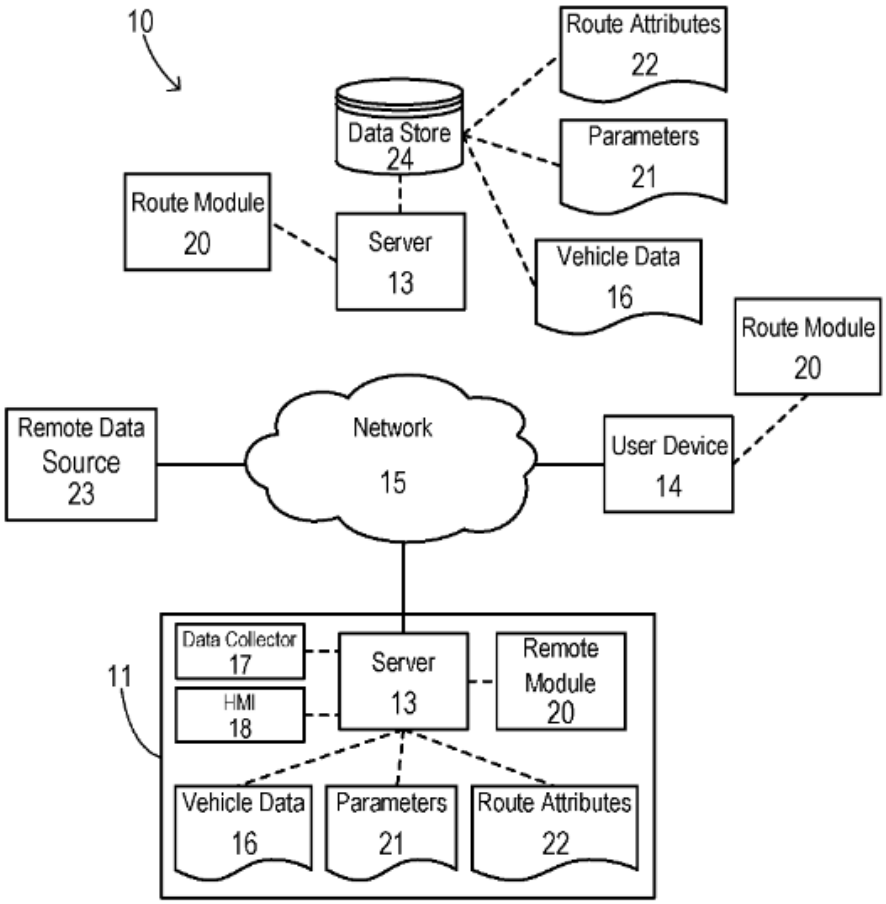

Рис. 2.4 – Під’єднання бібліотеки для Spring-проєкту

Ця розробка призначена для висвітлення потенціалу даних GTFS, зокрема, ставить за мету розгляд розробки інструменту візуалізації даних GTFS. Цей інструмент призначений для відображення просторових та часових моделей

транзитних послуг. Він надає можливість отримати якісну інформацію та зрозуміти характер транспортного обслуговування.

Проте, існує необхідність у створенні динамічного та інтерактивного інструменту візуалізації, який здатний вимірювати та відображати функціонування транспортної системи як у географічному, так і в статистичному аспектах. На базі попередніх досліджень це дослідження розробляє новий інструмент візуалізації для системи громадського транспорту. Інструмент має шість модулів візуалізації, що відображають різні робочі характеристики громадської транспортної системи: мобільність, швидкість, потік, щільність, прогрес та аналіз.

Користувач має можливість порівнювати два модулі одночасно для здійснення аналізу. Модуль аналізу забезпечує значущу статистичну інформацію та вимірювання, також враховуючи подібність та результати кластеризації, що базуються на характеристиках транзитної діяльності.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ТА ТЕСТУВАННЯ

3.1 Створення класів

Для точного відстеження положення транспортного засобу і збору інформації, створюємо Java програму з використанням фреймворку Spring, яка буде опрацьовувати дані з GPS трекерів і формувати GTFS дані.

3.1.1 Формування сутності GPS даних

GPS трекери надсилають дані на сервер, використовуючи протокол передачі даних - TCP/IP, ці дані, структоровані у мапу параметр:значення і виглядають так:

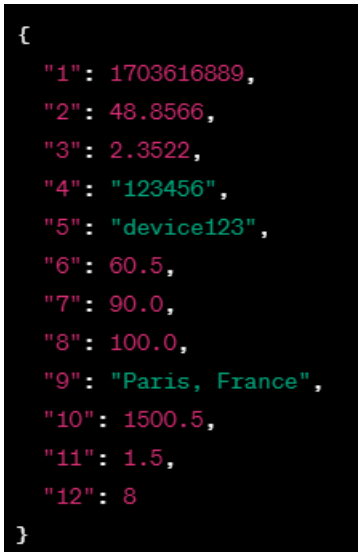

Рисунок 3.1 – Дані з трекера у форматі JSON

У цьому прикладі:

- "1": 1703616889 вказує на timestamp.
- "2": 48.8566 та "3": 2.3522 вказують на широту та довготу відповідно.
- "4": "123456" та "5": "device123" представляють accountID та deviceID.
- "6": 60.5 вказує на швидкість в км/год.
- "7": 90.0 вказує на напрямок руху (heading).

- "8": 100.0 вказує на висоту.
- "9": "Paris, France" представляє адресу.
- "10": 1500.5 вказує на пройдену відстань (одометр) в кілометрах.
- "11": 1.5 вказує на HDOP
- "12": 8 вказує на кількість супутників.

Розуміючи структуру даних які надсилає трекер, створюємо сутність яка буде відображати ці дані і зберігати їх в базу даних.

```java
@Entity
@Table(name = "gps_data")
@Data
public class GPSData {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @Column(name = "timestamp")
    private long timestamp;
    @Column(name = "latitude")
    private double latitude;
    @Column(name = "longitude")
    private double longitude;
    @Column(name = "account_id")
    private String accountID;
    @Column(name = "device_id")
    private String deviceID;
    @Column(name = "speed_kph")
    private double speedKPH;
    @Column(name = "heading")
    private double heading;
    @Column(name = "altitude")
    private double altitude;
    @Column(name = "address")
    private String address;
    @Column(name = "odometer_km")
    private double odometerKM;
    @Column(name = "hdop")
    private double HDOP;
    @Column(name = "satellite_count")
    private int satelliteCount;
```

Рисунок 3.2 – Сутність GPSData

Після отримання даних на сервері, вони піддаються обробці за допомогою парсера який зчитує сирі дані і заповнює сутність `gps_data` яка потім зберігається в БД.

3.1.2 Створення класів GTFS даних

Наступним кроком буде створення класів які відображатимуть різні сутності з GTFS даних, ось деякі з них:

Клас AgencyPojo представляє об'єкт, який відображає агентство яке формує дані GTFS.

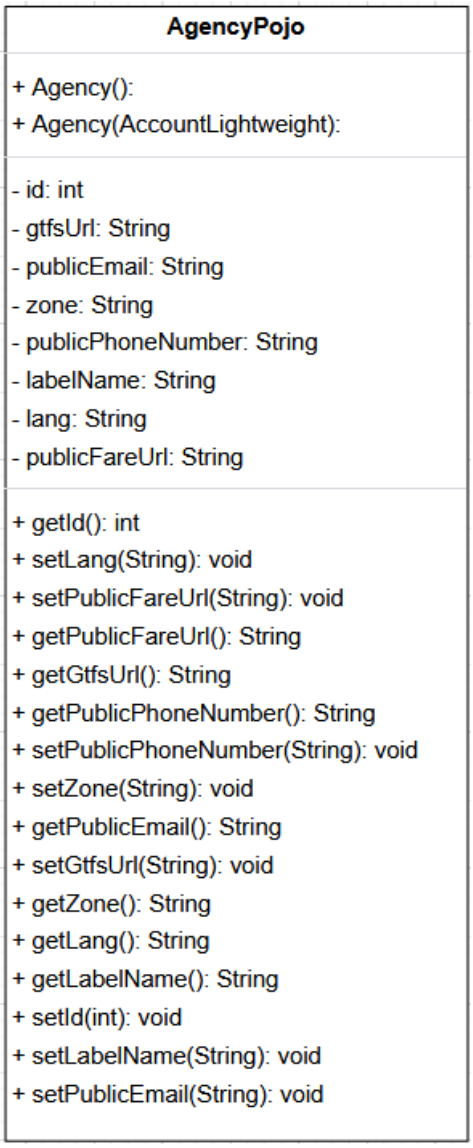

Рисунок 3.3 – Клас AgencyPojo

Він містить поля, які відображають різні атрибути агентства, такі як ідентифікатор, назва, URL для отримання GTFS-даних, зона обслуговування, мова, телефон для громадськості, URL для тарифів, та електронна адреса.

Клас AgencyStepConfig конфігурує крок в Spring Batch для обробки облікових записів та створення файлу GTFS для агентств.

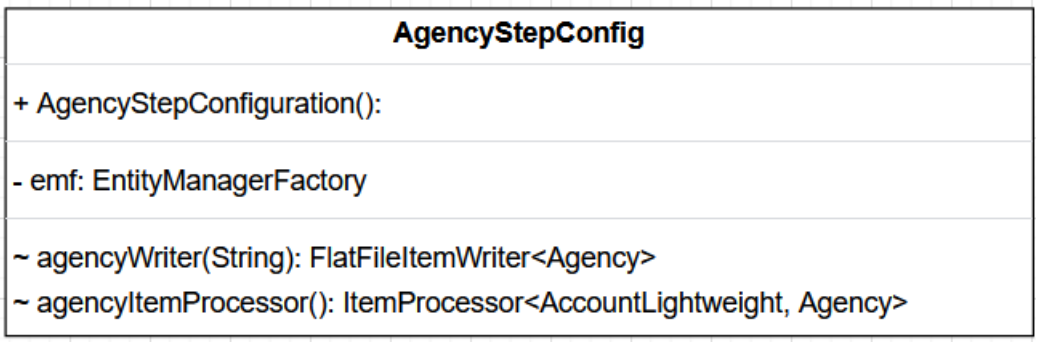

Рисунок 3.4 – Клас AgencyStepConfig

Він використовує JpaPagingItemReader для читання облікових записів з бази даних, вказуючи SQL-запит для вибору відповідних даних. Далі використовується ItemProcessor, який використовує конструктор Agency для перетворення об'єкта AccountLightweight в об'єкт Agency. Нарешті, клас використовує FlatFileItemWriter для запису об'єктів Agency у текстовий файл у форматі GTFS, зазначаючи шлях для збереження файлу та вказуючи формат рядка та властивості екстрактора полів.

Клас `CalendarPojo` представляє об'єкт, який відображає календарні події у форматі GTFS. Він містить поля, такі як ідентифікатор служби, дні тижня та дати початку та завершення подій.

Рисунок 3.5 – Клас CalendarPojo

Клас `CalendarStepConfig` містить конфігурацію для обробки календарних подій та створення файлу `calendar.txt` у фораматі GTFS.

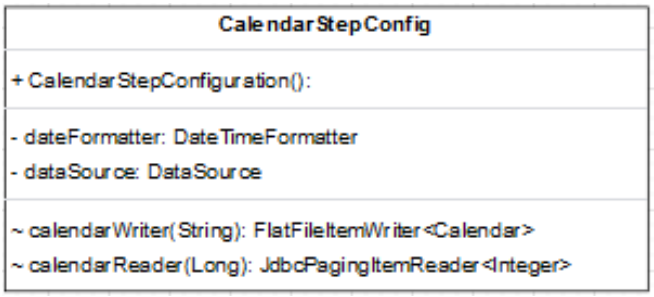

Рисунок 3.6 – Клас CalendarStepConfig

Для кожного типу GTFS даних існують свої класи-біни, разом з конфігураційним класами які заповняють дані і являються кроками, виконанням яких займається клас `StaticGtfsJobConfig`, що має конфігураційні методи для кожного кроку в процесі створення GTFS-фіда, такі як читання інформації про обліковий запис, дати календаря, агентства, маршрути, календарі, зупинки, часи зупинки, подорожі та форми. Кожен крок має визначений читач, процесор та письменник для обробки і збереження даних.

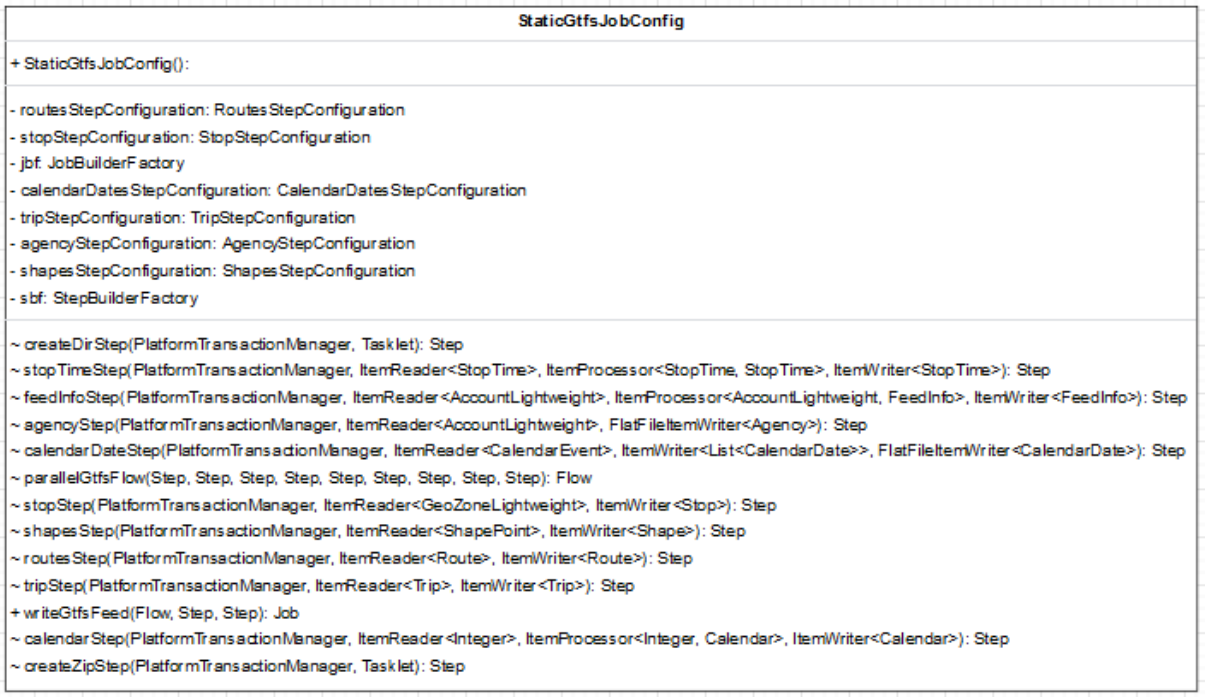

Рисунок 3.7 – Клас StaticGtfsJobConfig

Основний Job `writeGtfsFeed` складається з кроків, які виконуються послідовно: створення директорії, паралельне виконання важливих кроків GTFS (через `split`), та створення ZIP-архіву.

Загальна структура об'єднаного потоку `parallelGtfsFlow` використовує функціональність `Flow` для об'єднання різних кроків у паралельному виконанні, що дозволяє оптимізувати час виконання задачі.

Цей конфігураційний клас використовує Spring Batch для визначення кроків, читачів, процесорів та письменників, що складають задачу створення GTFS-фіда з джерела даних.

В результаті виконання кроків створення GTFS ми отримаєм набір файлів які будуть містити всю необхідну інформацію.

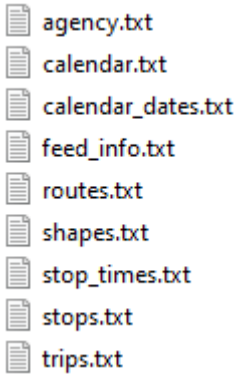

Рисунок 3.8 – Файли GTFS

В першому рядку кожного файлу можна побачити опис інформації яка міститься в кожній колонці. Для прикладу розглянемо файл stops.txt.

```
stop_id,stop_code,stop_name,stop_desc,stop_lat,stop_lon
4711,0320,ТРЦ Кінг Кросс (320),"вул. Стрийська, 30",49.774093,24.012718
```

Рисунок 3.9 – Приклад файлу stops.txt

Даний файл містить інформацію про Id зупинки, код зупинки, назву зупинки, кориткий опис, координати зупинки(широта, довгота).

3.2 Тестування системи

Тестування є важливою частиною розробки програмного забезпечення, оскільки воно дозволяє переконатися, що код працює так, як очікується. У цьому розділі ми розглянемо тестування класу на прикладі StopStepConfig, який відповідає за конфігурацію кроку імпорту зупинок у форматі GTFS. Решту класів тестуються по такому ж принципу.

Інструменти та технології:

JUnit 5: Фреймворк для тестування Java-програм, який надає анотації для визначення методів тесту та їхніх властивостей.

Mockito: Бібліотека для створення підроблених об'єктів (mock objects) у юніт-тестах.

Цілі тестування:

- Перевірка налаштування stopStep на правильність створення кроку для імпорту зупинок.
- Переконання, що параметри кроку встановлюються вірно і відповідають очікуванням.
- Перевірка, що внутрішні залежності (наприклад, geozonePoiReader та stopItemProcessor) вірно інтегровані.

Основні етапи тесту:

- Підготовка середовища тестування:

Використано аннотації JUnit, такі як @Test та @ExtendWith(MockitoExtension.class), для вказівки на те, що цей клас є тестом та для використання розширення Mockito для ініціалізації мок-об'єктів.

- Ініціалізація мок-об'єктів:

Створено мок-об'єкти для EntityManagerFactory, DataSource, та StepBuilderFactory.

Створено об'єкт StopStepConfig та ініціалізовано його за допомогою аннотації @InjectMocks, щоб автоматично вставити мок-об'єкти відповідних полів.

- Налаштування мок-об'єктів:

Створено мок-об'єкти для читача (geozonePoiReader), обробника (stopItemProcessor), та письменника (stopWriter) даних.

Викликано метод stopStep об'єкту StopStepConfig з передачею мок-об'єктів та інших параметрів.

- Перевірка коректності конфігурації:

Здійснено ряд перевірок, використовуючи Mockito-функції verify, щоб переконатися, що методи об'єктів були викликані з правильними параметрами та в правильному порядку.

Наприклад, перевіряється, чи викликано методи для налаштування читача (geozonePoiReader) та перевіряється, чи викликано методи для налаштування обробника (stopItemProcessor) та письменника (stopWriter).

```
@ExtendWith(MockitoExtension.class)
class StopStepConfigurationTest {

    @Mock
    private EntityManagerFactory entityManagerFactory;

    @Mock
    private DataSource dataSource;

    @Mock
    private StepBuilderFactory stepBuilderFactory;

    @InjectMocks
    private StopStepConfiguration stopStepConfiguration;

    @Test
    void testStopStepConfiguration() throws Exception {
        // Arrange
        ItemReader<GeoZoneLightweight> geozonePoiReader = mock(JpaPagingItemReader.class);
        ItemProcessor<GeoZoneLightweight, Stop> stopItemProcessor = mock(ItemProcessor.class);
        ItemWriter<Stop> stopWriter = mock(ItemWriter.class);

        // Act
        stopStepConfiguration.stopStep(entityManagerFactory, geozonePoiReader, stopItemProcessor, stopWriter);

        // Assert
        verify(stepBuilderFactory).get("stopStep");
        verify(geozonePoiReader).setName("geozonePoiReader");
        verify(geozonePoiReader).setEntityManagerFactory(entityManagerFactory);
        verify(geozonePoiReader).setQueryProvider(any());
        verify(geozonePoiReader).setParameterValues(any());
        verify(geozonePoiReader).afterPropertiesSet();

        verify(stopItemProcessor).setName("stopItemProcessor");

        verify(stopWriter).setName("stopWriter");
        verify(stopWriter).setResource(any());
        verify(stopWriter).setHeaderCallback(any());
        verify(stopWriter).setLineAggregator(any());
    }
}
```

Рисунок 3.10 – Тест для класу StopStepConfig

Цей тест допомагає гарантувати, що конфігурація стопового кроку відбувається правильно та очікувано при виклику відповідного методу у класі StopStepConfig.

3.3 Використання системи через користувацький інтерфейс

Наступним кроком буде відображення інформації з GTFS файлів і даних GPS у зрозумілий інтерфейс користувачу.

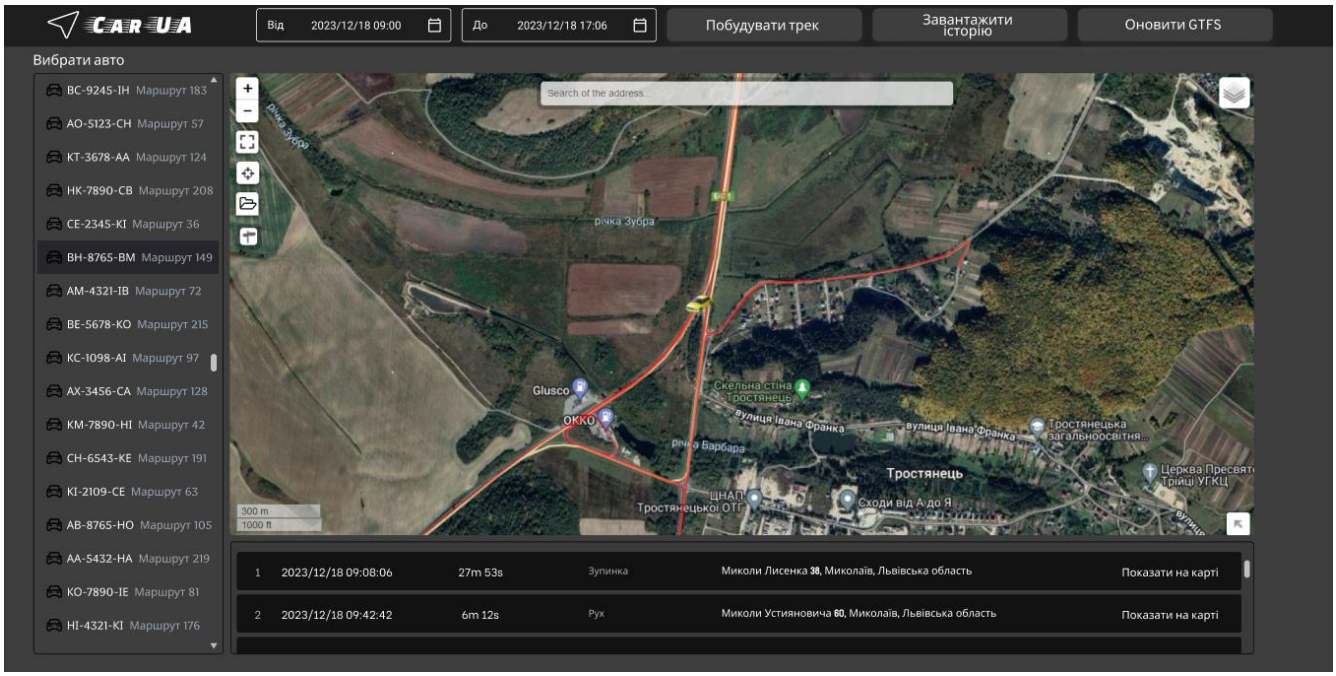

Рисунок 3.11 – Головна сторінка додатку

Інтерфейс виконаний у форматі веб-додатку і містить такі функціональні складові:

- список доступних транспортних засобів
- селектор дати
- кнопки навігації
- вікно карти
- список який відображає історію руху

Вибираючи ТЗ зі списку ми зразу можем спостерігати його останнє місцезнаходження на карті у вигляді іконки автомобіля. У випадку якщо нам треба візуально переглянути рух автомобіля за певний час – вводимо потрібну

дату у селекторі і натискаємо кнопку «Побудувати трек», це дозволить нам побачити як ТЗ рухався у вказаному періоді.

Рисунок 3.12 – Селектор дати і кнопка побудови треку

Також під час побудови треку за вказаний час, знизу під картою відображається історія руху ТЗ, де можна побачити детальніші дані про те коли він рухався чи зупинявся і скільки часу ця дія тривала. Якщо треба переглянути де ця дія відбулася на карті, можна натиснути кнопку «Показати на карті».

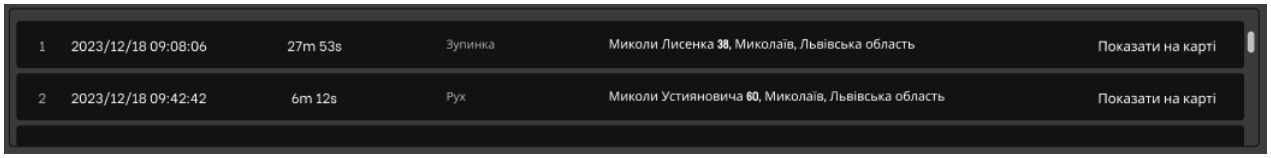

Рисунок 3.13 – Вікно історії руху ТЗ

Історію також можна завантажити у форматі Excel, натиснувши кнопку «Завантажити історію».

У випадку якщо були оновленні дані про маршрут, зупинку чи ТЗ, можна натиснути на кнопку «Оновити GTFS» для того щоб програма формувала актуальну статистику.

Рисунок 3.14 – Навігаційні кнопки для завантаження історії і оновлення GTFS

Загалом даний інтерфейс достатньо інтуїтивний і дозволяє користувачу легко користуватися доступним функціоналом.

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці. Вимог охорони праці з ціллю збереження здоров'я при роботі за комп’ютерною технікою. Планування робочого місця

У наш час інформаційної технології комп'ютерна робота стала неовід'ємною частиною повсякденного життя, і для багатьох людей вона є основним видом занять. Однак зручність та продуктивність використання комп'ютерної техніки повинні бути взаємопов’язані з безпекою умов праці. Особливо актуальним стає питання безпеки та збереження здоров'я при роботі за комп'ютером працівників сфери інформаційних технологій.

При написанні кваліфікаційної роботи та розробці системи моніторингу за керованими рухомими об'єктами на карті використовувався персональний комп’ютер та обладнання, таке як електронно-цифрові датчики, GPS-трекери. Тому слід зазначити, що при роботі за комп’ютерною технікою необхідно дотримуватися вимог охорони праці з ціллю збереження здоров'я.

Мета дослідження в розділі — висвітлити ключові аспекти планування робочого місця для забезпечення не лише ефективності, але й безпеки та збереження здоров'я працівників, які використовують комп'ютерну техніку у своїй роботі. В процесі розглянуті ергономічні аспекти, освітлення, обладнання робочого місця, а також важливість регулярних перерв та фізичної активності. Додатково, досліджено аспекти психосоціального середовища та методи контролю за дотриманням вимог безпеки на робочому місці.

Для того, щоб виявити та проаналізувати шкідливі і небезпечні виробничі фактори необхідно почати з аналізу дотримання вимог, встановлених санітарними правилами і нормами [ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин»] для виробничих приміщень та робочих місць.

При роботі з комп’ютером відбувається значний вплив на нервово-емоційний стан операторів. Ця діяльність характеризується великим

навантаженням на м'язи рук при використанні клавіатури, інтенсивною зоровою роботою та значним розумовим перенапруженням.

Отже, раціональне організування робочого місця повинно сприяти зменшенню втоми працівників та підвищенню продуктивності праці, а також уникненню загального дискомфорту та оптимальному розташуванню інструментів та предметів праці.

На виробництві проводиться сертифікація робочих місць для об'єктивної оцінки умов праці. Використовується "Гігієнічна класифікація праці" за показниками небезпечності та шкідливості факторів виробничого середовища, напруженості та тяжкості трудового процесу. За принципами гігієнічної класифікації умови праці поділяються на чотири класи.

- Перший клас визначає оптимальні умови праці, які створюють передумови для підтримки високого рівня працездатності працівників.
- Другий клас охоплює допустимі умови праці, характеризовані рівнями факторів виробничого середовища та трудового процесу, що не перевищують встановлені нормативи.
- Третій клас описує шкідливі умови праці, де рівні шкідливих виробничих факторів перевищують нормативи, що може призвести до негативного впливу на здоров'я працівника та/або його нащадків.
- Четвертий клас – небезпечні/екстремальні умови праці.

На основі сертифікації робочого місця необхідно провести оцінку інтенсивності робіт у наступних аспектах:

1. Відповідність обладнання нормативно-технічним вимогам.
2. Аналіз документації та характеру виконуваних робіт, врахування обсягу робіт.
3. Визначення відповідності площі та обсягу робочого місця чинним нормам.
4. Перевірка спеціалізованого обладнання робочого місця, включаючи засоби захисту та їх технічний стан.

Врахування відповідності технологічного процесу, інструментів, устаткування та засобів контролю вимогам стандартів безпеки та норм охорони праці.

Обладнання та організація робочого місця ВДТ ЕОМ повинні відповідати конструкції всіх його елементів та взаємному розташуванню, враховуючи ергономічні вимоги, що визначені [ДСТУ 7299:2013 «Дизайн і ергономіка. Робоче місце оператора. Взаємне розташування елементів робочого місця. Загальні вимоги ергономіки»]. Раціональне планування робочого місця повинно забезпечити оптимальне розміщення інструментів та предметів праці, зменшення втоми працівників та підвищення продуктивності праці, а також уникнення загального дискомфорту.

Для усунення ризику ураження електричним струмом слід правильно розміщувати обладнання та електричні кабелі. Інші заходи щодо забезпечення електробезпеки повинні відповідати загальним заходам пожежної та електробезпеки.

Для забезпечення пожежної безпеки рекомендується використовувати приховану електромережу, використовувати надійні розетки з пожежобезпечних матеріалів, регулярно чистити внутрішні частини комп'ютерів від пилу та розміщувати їх на окремих столах. Також важливо рідше вставляти та виймати вилки з розеток, а робочі місця повинні бути розташовані на відстані від вікон та інших стін відповідно до норм.

Конструкція робочої поверхні ВДТ повинна підтримувати оптимальне робоче положення, а стільці - мати форму спинки, що повторює форму спини працівника. Важливо враховувати висоту стільця, щоб уникнути тиску на стегна чи куприк, та обладнати його підлокітниками.

Щодо розташування робочих місць, їх слід розташовувати враховуючи світлові умови, забезпечуючи оптимальну відстань до вікон та використовуючи спеціалізовані фільтри для екранів ВДТ. Екрани ВДТ повинні бути розташовані на оптимальній відстані від очей користувача, а параметри лазерного

випромінювання принтерів повинні відповідати вимогам [ДСанПІН 3.3.2.007-98].

Розробники програмних рішень, ІТ-фахівці постійно працюють з обладнанням, яке підключено до електропостачання. При організації робочого місця потрібно ретельно перевіряти безпечність підключення до електромереж.

Нормативна база, що регулює умови праці за комп'ютерною технікою, визначає вимоги, які мають забезпечувати оптимальні умови працівників. Ця робота присвячена детальному аналізу вимог охорони праці з метою збереження здоров'я при роботі за комп'ютером, зокрема у контексті планування робочого місця в рамках розробки проєкту системи моніторингу за керованими рухомими об'єктами на карті з використанням новітніх технологій.

Наразі, в Україні проводиться розробка та вдосконалення національних нормативних документів, спрямованих на охорону праці користувачів ПК. Найбільш повним нормативним документом щодо забезпечення охорони праці користувачів ПК є «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами (ВДТ) електронно-обчислювальних машин» (ДСПіН 3.3.2.007–98.) [12].

Врахування всіх цих заходів забезпечить комфортні умови праці, підвищить продуктивність та сприятиме збереженню здоров'я при роботі за персональним комп'ютером.

4.2. Безпека в надзвичайних ситуаціях. Пожежна безпека при улаштуванні електроустановок

У процесі написання кваліфікаційної роботи та розробки data-системи керованими рухомими об'єктами на карті використовувався персональний комп’ютер та цифрове обладнання. При роботі з персональним комп’ютером серед різних видів небезпек варто виділити ризик пожежі внаслідок несправності

електрообладнання, недотримання або порушення правил пожежної безпеки обслуговуючим персоналом, що може призвести до пожежі та неправильні дії персоналу в надзвичайних ситуаціях.

Відповідно до «Правил улаштування електроустановок» реалізовані такі групи заходів електробезпеки: конструктивні заходи забезпечують захист від випадкового контакту з струмопровідними частинами за допомогою їх ізоляції та захисних оболонок. Оскільки згідно з [НПАОП 40.1-1.32-01 "Правила встановлення електроустановок. Електричне обладнання спеціальних установок"] офісні приміщення в основному відносяться до класу пожежонебезпечної зони П-IIа (приміщення, в яких розташовані тверді горючі речовини), тому ступінь забезпечується захист ізоляції обладнання IP44.

Приміщення, в яких працюють розробники програмного забезпечення, класифікуються як приміщення без підвищеного ризику ураження електричним струмом. Обладнання, яке використовується в цих кімнатах, є споживачем електроенергії, що живиться від мережі змінного струму 220 В від мережі із заземленою нейтраллю, і належить до електричних установок до 1000 В закритої конструкції.

Іншим важливим завданням безпеки в надзвичайних ситуаціях є створення заходів пожежної безпеки. Закон України «Про пожежну безпеку» визначає загальні правові, соціальні і економічні основи забезпечення пожежної безпеки на території України, регулює відносини між державними органами, юридичними та фізичними особами у цій галузі, незалежно від виду їх діяльності та форм права власності.

Пожежна безпека - це стан об’єкта, при якому можливість виникнення та розвитку пожежі та вплив її небезпечних факторів на людей виключається з регульованою ймовірністю, а також забезпечується захист матеріальних цінностей. Для забезпечення пожежної безпеки в закладах проводиться протипожежна профілактика, що включає комплекс організаційно-технічних заходів, спрямованих на забезпечення безпеки людей, запобігання пожежі,

обмеження її поширення, а також створення умов для успішного подолання пожежі.

Для ліквідації пожежі на початковій стадії її розвитку персонал об’єктів повинен використовувати первинне обладнання для пожежогасіння. До них відносяться:

- вогнегасники;
- протипожежне обладнання (покривала з негорючих теплоізоляційних тканин, пожежні відра, ящики з піском, лопати, ломи, сокири і тп);
- автоматичні системи пожежогасіння.

Первинне обладнання пожежогасіння, може розташовуватися як окремо, так і у складі протипожежних щитів. Це залежить від агрегатного стану та особливостей горіння різних горючих речовин та матеріалів пожежі [ДБН В. 1.1.7-2016 «Пожежна безпека об’єктів будівництва. Загальні вимоги»] поділяються на відповідні класи. Офісні приміщення містять дерев'яні меблі, електронне обладнання і паперові матеріали. Клас пожежі в офісному приміщенні [ДБН В.1.1-7:2016 «Пожежна безпека об`єктів будівництва»] - пожежі твердих речовин, переважно органічного походження, горіння яких супроводжується тлінням (деревина, пластик, папір) і визначається як клас А.

Категорія приміщень [ДСТУ Б В.1.1-36:2016 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою»] - визначається як категорія D. Визначення типу та розрахунок кількості первинних засобів пожежогасіння обладнання [ДБН В.1.1-7:2016 «Пожежна безпека об`єктів будівництва. Загальні вимоги»]. Крім того, адміністративні приміщення повинні бути оснащені протипожежними датчиками, які реагують на підвищення температури, на дим та полум’я. До прикладу, такі моделі датчиків, як ДТЛ, ІТМ.

Для запобігання пожежі надзвичайно важливо правильно оцінити пожежонебезпеку будівлі, виявити небезпеку та обґрунтувати методи та засоби протипожежної охорони та захисту.

Однією з умов забезпечення пожежної безпеки є усунення можливих джерел займання. Несправності електропроводки, електричних розеток та вимикачів можуть стати причиною запалення у лабораторії. Необхідно регулярно здійснювати плановий огляд – виявляти та усувати існуючі несправності, щоб уникнути виникнення пожежі, з цих причин.

Іншою причиною запалення у лабораторних приміщеннях може бути несправність електроприладів. Щоб усунути виникнення пожежі, із цих причин, необхідно своєчасно проводити ремонт електроприладів та не використовувати несправні електроприлади.

При обігріванні приміщень електронагрівальними приладами з відкритими нагрівальними елементами може призвести до запалення у лабораторії. З огляду на те, що в приміщенні є папір, який є легкозаймистим матеріалом - відкриті нагрівальні поверхні можуть спричинити пожежу. В лабораторії рекомендується не використовувати відкриті нагрівальні прилади задля запобігання пожежі.

Коротке замикання в електропроводці може призвести до пожежі. Електропроводка повинна бути прихованою, щоб зменшити ймовірність пожежі внаслідок короткого замикання.

Ще однією з причин запалення є влучення блискавки у будівлю. Влітку під час грози може потрапити блискавка в будинок, що призведе до можливої пожежі. Щоб уникнути цього, рекомендується встановити громовідвід на даху будинку.

Недотримання заходів пожежної безпеки та куріння в приміщенні також можуть спричинити пожежу. Для ліквідації пожежі в результаті паління в лабораторії необхідно категорично заборонити куріння і дозволяти це лише в суворо відведеному місці.

Перед початком роботи, з метою запобігання пожежі, необхідно провести інструктаж з протипожежних заходів з працівниками, що працюють у лабораторії, де необхідно ознайомити працівників з правилами пожежної безпеки, а також навчити їх користуватися первинним обладнанням для гасіння пожежі.

Перше, що необхідно зробити під час виникнення пожежі це відключити електропостачання, зателефонувати до пожежної охорони, евакуювати людей з приміщень відповідно до плана евакуації, та розпочати гасіння полум’я вогнегасниками. Якщо є незначне джерело полум’я, тоді потрібно, використавши підручні засоби, зупинити доступ повітря до джерела займання.

Під час роботи комп’ютерів забороняється ремонтувати їх на робочому місці, не допускається працювати на пошкодженому обладнанні або захаращувати робочі місця матеріалами, які не використовуються для поточних робіт. Ремонт, будь-яке технічне обслуговування та налагодження комп'ютера, а також інші операції з цього приводу слід проводити лише тоді, коли живлення повністю вимкнено. У тих випадках, коли ремонт та інші процедури неможливо виконати при відключенні електроживлення, необхідно, щоб обладнання та периферійні пристрої були заземлені, ремонт виконували двоє або більше робітників з використанням інструментів з ізольованими ручками, а також килимки-діелектрики на підлогу.

Режим праці та відпочинку працівників електронно-обчислювальної техніки визначається [ДСанНіП 3.3.2-007-98]. Під час робочого дня кожні 40-50 хвилин роботи потрібно робити перерви для відпочинку, які тривають близько 3-5-хвилин. Загальна тривалість роботи на день не повинна перевищувати 4 години, а на тиждень - 20 годин.

Таким чином, при дотриманні правил безпеки в надзвичайних ситуаціях, можна запобігти виникненню критичних результатів. Програмний продукт та процес виконання, що описаний у кваліфікаційній роботі, розроблений з урахуванням вимог техніки безпеки та пожежної безпеки.

ВИСНОВКИ

Дослідження є важливим етапом, оскільки воно дозволяє зрозуміти контекст та вимоги проекту, обрати оптимальні технології та розробити стратегію реалізації системи моніторингу.

Перший етап дослідження визначає основні вимоги до системи. Це включає вивчення потреб користувачів, визначення функціональних та нефункціональних вимог, а також уточнення особливостей використання протоколу GTFS в контексті моніторингу транспортного руху.

Проведення технічного дослідження включає в себе огляд технологій Java та Spring для вибору найбільш підходящих інструментів та фреймворків для реалізації системи. Також важливо вивчити специфікації протоколу GTFS та розуміти його можливості та обмеження. Вивчення існуючих систем моніторингу транспортного руху для зрозуміння їхніх переваг та недоліків. Це допомагає уникнути повторення помилок та використовувати кращі практики.

Дослідження також повинно враховувати можливості подальшого розвитку системи, включаючи нові функції, інтеграцію з іншими платформами, апгрейд технічних компонентів та інше.

У сучасному світі, де рух транспорту великих міст стає все більш інтенсивним, розробка системи моніторингу за рухомими об'єктами з використанням сучасних підходів у поєднанні з протоколом GTFS (General Transit Feed Specification) дозволить створити ефективну систему.

В цілому, розробка системи моніторингу за керованими рухомими об'єктами є важливим кроком у вдосконаленні транспортної інфраструктури та забезпеченні більш ефективного управління рухом транспорту в містах. Використання технологій Java, Spring та протоколу GTFS дозволяє створити ефективний інструмент для відстеження та оптимізації руху транспортних засобів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. GTFS - General Transit Feed Specification" (Офіційна документація) – [Електронний ресурс] – Режим доступу до ресурсу: https://gtfs.org/en/realtime/changes/
2. GTFS Static Overview - [Електронний ресурс]: - Режим доступу до ресурсу: https://developers.google.com/transit/gtfs
3. Roush, Wade (2012). "Welcome to Google transit: How (and why) the search giant is remapping public transportation" [Електронний ресурс] – Режим доступу до ресурсу: hhttps://ctaa.org/wp-content/uploads/2018/10/Spring_12_DigitalCT.pdf
4. Clean Code: A Handbook of Agile Software Craftsmanship, Robert C. 2008, с. 464 ISBN – 978-0132350884
5. Effective Java: A Programming Language Guide J. Bloch. Addison-Wesley Longman, Amsterdam, 2nd Revised edition (REV). edition, (2008)
6. Craig Larman (2004). Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. New York: Pearson.
7. Spring Framework – [Електронний ресурс] – Режим доступу до ресурсу: http://spring-projects.eu/projects/spring-framework/.0
8. M. Kleppmann, "Designing Data-Intensive Applications", O'Reilly Media, 2017. R. C. Martin, "Clean Architecture: A Craftsman's Guide to Software Structure and Design", 2017.
9. J. Wagner, "Web Performance in Action", Manning, 2017.
10. Philippe Kruchten (2003). The Rational Unified Process: An Introduction (3rd Edition). Boston: Addison-Wesley Professional.
11. Rational Unified Process. Вікіпедія: веб-сайт. URL: https://uk.wikipedia.org/wiki/Rational_Unified_Process (дата звернення: 10.12.2023).

12. Martin Fowler. "UML Distilled: A Brief Guide to the Standard Object Modeling Language." - Boston: Addison-Wesley, 2003. - 192 с.
13. ДСанПІН 3.3.2.007-98 [Електронний ресурс]. – 1998. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98#Text.
14. ДСТУ Б В.1.1-36:2016 «Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою»
15. Жидецький В.Ц. Охорона праці користувачів комп'ютерів. – Львів: Афіша, 2000.– 176с.
16. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – Інженерія програмного забезпечення (Освітньо-професійна програма - «Програмне забезпечення систем», Освітньо-наукова програма - «Інженерія програмного забезпечення») для студентів усіх форм навчання / Упор.: М.Р. Петрик, Д.М. Михалик, О.Ю. Петрик, Г.Б. Цуприк - Тернопіль: ТНТУ, 2020- 51с.

ДОДАТКИ

ДОДАТОК А

Публікація в науковому виданні

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

13-14 грудня 2023 року

ТЕРНОПІЛЬ
2023

УДК 004.4
Корба Д., Мудрик І.
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ПРОЕКТУВАННЯ ТА РОЗРОБКА СИСТЕМИ МОНІТОРИНГУ РУХОМИХ ОБ'ЄКТІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЙ JAVA, SPRING ТА ПРОТОКОЛУ GTFS

Korba D., Mudryk I.

DESIGN AND DEVELOPMENT OF MOBILE OBJECTS MONITORING SYSTEM USING JAVA, SPRING AND GTFS PROTOCOL TECHNOLOGIES

Цей проект має на меті створення інтегрованої системи моніторингу руху транспортних засобів, об'єднуючи в собі різноманітні технології та методи стеження. Одним із ключових завдань є дослідження та вибір оптимальних технологій, таких як GPS-технології, RFID та системи візуального спостереження, з метою забезпечення найвищої точності та ефективності системи. Центральним елементом проекту є інтеграція технології GTFS для оптимізації обробки даних про розклади громадського транспорту. Це дозволяє стандартизувати формат обміну інформацією, спрощуючи взаємодію з різними системами та додатками, а також покращуючи точність визначення маршрутів та графіків руху транспортних засобів.

Технічний стек включав в себе використання Java та Spring для розробки модулю моніторингу, а також використання протоколу GTFS для стандартизованого обміну інформацією про громадський транспорт.

Забезпечення безпеки та конфіденційності системи було досягнуте за допомогою механізмів безпеки, таких як JWT, для захисту доступу та збереження конфіденційності даних. Процес виявлення аномалій в системі включав розробку функціоналу та системи сповіщень для небезпечних ситуацій. Практика тестування з використанням JUnit 5 та оптимізація коду забезпечували високу якість та масштабованість системи.

У висновку, проєкт визначається як важливий крок у вдосконаленні сучасних систем управління та контролю транспортним рухом. Впровадження різноманітних технологій стеження, спрямованих на забезпечення високої точності та ефективності, разом із використанням стандартів, таких як GTFS, дозволяє створити інтегровану систему, що відповідає сучасним стандартам безпеки та зручності використання. У майбутньому розвитку цієї системи, важливо розглядати можливість впровадження додаткових функціональностей, таких як прогнозування трафіку та оптимізація маршрутів. Додаткові модулі для взаємодії з іншими "розумними" системами транспортної інфраструктури можуть покращити загальну ефективність та покращити управління дорожнім рухом.

Література

1. GTFS Static Overview - [Електронний ресурс]: - https://developers.google.com/transit/gtfs
2. GTFS - [Електронний ресурс]:- https://en.wikipedia.org/wiki/GTFS
3. Roush, Wade (2012). "Welcome to Google transit: How (and why) the search giant is remapping public transportation" [Електронний ресурс] – Режим доступу до ресурсу: hhttps://ctaa.org/wp-content/uploads/2018/10/Spring_12_DigitalCT.pdf
4. Harrelson, Chris. "Happy Trails with Google Transit" - [Електронний ресурс] - https://googleblog.blogspot.com/2006/09/happy-trails-with-google-transit.html.
5. GTFS: Making Public Transit Data Universally Accessible - [Електронний ресурс]: - https://gtfs.org/
6. General Transit Feed Specification - [Електронний ресурс]: - https://www.transitwiki.org/TransitWiki/index.php/General_Transit_Feed_Specification

ДОДАТОК Б

Лістинг коду:

```
public abstract class GTFS {

   private static final Logger LOG = LoggerFactory.getLogger(GTFS.class);

   private static final String DEFAULT_DATABASE_URL = "jdbc:postgresql://localhost/gtfs";

   public static FeedLoadResult load (String filePath, DataSource dataSource) {
      JdbcGtfsLoader loader = new JdbcGtfsLoader(filePath, dataSource);
      FeedLoadResult result = loader.loadTables();
      return result;
   }

   public static ValidationResult validate (String feedId, DataSource dataSource) {
      Feed feed = new Feed(dataSource, feedId);
      ValidationResult result = feed.validate();
      return result;
   }

   public static DataSource createDataSource (String url, String username, String password) {
      String characterEncoding = Charset.defaultCharset().toString();
      LOG.debug("Default character encoding: {}", characterEncoding);
      if (!Charset.defaultCharset().equals(StandardCharsets.UTF_8)) {
         // Character encoding must be set to UTF-8 in order for the database connection to work
without error.
         // To override default encoding at runtime, run application jar with encoding environment
variable set to
         // UTF-8 (or update IDE settings). TODO we should also check that JDBC and the database
know to use UTF-8.
         throw new RuntimeException("Your system's default encoding (" + characterEncoding + ")
is not supported. Please set it to UTF-8. Example: java -Dfile.encoding=UTF-8 application.jar");
      }
      // ConnectionFactory can handle null username and password (for local host-based
authentication)
```

```
        ConnectionFactory connectionFactory = new DriverManagerConnectionFactory(url,
username, password);
        PoolableConnectionFactory poolableConnectionFactory = new
PoolableConnectionFactory(connectionFactory, null);
        GenericObjectPool connectionPool = new GenericObjectPool(poolableConnectionFactory);

    }

    public static void main (String[] args) {

        Options options = getOptions();
        CommandLine cmd = null;
        try {
            cmd = new DefaultParser().parse(options, args);
        } catch (ParseException e) {
            LOG.error("Error parsing command line: " + e.getMessage());
            printHelp(options);
            return;
        }

        if (cmd.hasOption("help")) {
            printHelp(options);
            return;
        }

        if (!cmd.getArgList().isEmpty()) {
            LOG.error("Extraneous arguments present: {}", cmd.getArgs());
            printHelp(options);
            return;
        }

        if (!(cmd.hasOption("load") || cmd.hasOption("validate") || cmd.hasOption("graphql"))) {
            LOG.error("Must specify one of 'load', 'validate', or 'graphql'.");
            printHelp(options);
            return;
```

```
}

String databaseUrl = cmd.getOptionValue("database", DEFAULT_DATABASE_URL);
String databaseUser = cmd.getOptionValue("user");
String databasePassword = cmd.getOptionValue("password");
LOG.info("Connecting to {} as user {}", databaseUrl, databaseUser);


DataSource dataSource = createDataSource(databaseUrl, databaseUser, databasePassword);
FeedLoadResult loadResult = null;
if (cmd.hasOption("load")) {
   String filePath = cmd.getOptionValue("load");
   loadResult = load(filePath, dataSource);
   LOG.info("The unique identifier for this feed is: {}", loadResult.uniqueIdentifier);
}

if (cmd.hasOption("validate")) {
   String feedToValidate = cmd.getOptionValue("validate");
   if (feedToValidate != null && loadResult != null) {
      LOG.warn("Validating the specified feed {} instead of {} (just loaded)",
            feedToValidate, loadResult.uniqueIdentifier);
   }
   if (feedToValidate == null && loadResult != null) {
      feedToValidate = loadResult.uniqueIdentifier;
   }
   if (feedToValidate != null) {
      LOG.info("Validating feed with unique identifier {}", feedToValidate);
      ValidationResult validationResult = validate (feedToValidate, dataSource);
      LOG.info("Done validating.");
   } else {
      LOG.error("No feed to validate. Specify one, or load a feed in the same command.");
   }
}

if (cmd.hasOption("graphql")) {
```

```
            Integer port = Integer.parseInt(cmd.getOptionValue("graphql"));
            LOG.info("Starting GraphQL server on port {}", port);
            throw new UnsupportedOperationException();
        }

    }

    private static Options getOptions () {
        Options options = new Options();
        options.addOption(Option.builder("h").longOpt("help").desc("print this message").build());
        options.addOption(Option.builder().longOpt("load").hasArg()
                .argName("file").desc("load GTFS data from the given file").build());

options.addOption(Option.builder().longOpt("validate").hasArg().optionalArg(true).argName("feed
")
                .desc("validate the specified feed. defaults to the feed loaded with the --load
option").build());
        options.addOption(Option.builder("d").longOpt("database")
                .hasArg().argName("url").desc("JDBC URL for the database. Defaults to " +
DEFAULT_DATABASE_URL).build());
        options.addOption(Option.builder("u").longOpt("user")
                .hasArg().argName("username").desc("database username").build());
        options.addOption(Option.builder("p").longOpt("password")
                .hasArg().argName("password").desc("database password").build());
        options.addOption(Option.builder().longOpt("graphql")
                .desc("start a GraphQL API on the given port").optionalArg(true).build());
        options.addOption(Option.builder().longOpt("json").desc("optionally store in
result.json").build());
        return options;
    }

    private static void printHelp(Options options) {
        final String HELP = String.join("\n",
                "java -cp gtfs-lib-shaded.jar com.conveyal.gtfs.GTFS [options]",
```

```
            "",
            ""
        );
        HelpFormatter formatter = new HelpFormatter();
        System.out.println(); // blank line for legibility
        formatter.printHelp(HELP, options);
        System.out.println(); // blank line for legibility
    }


}
public class GTFSMain {

    private static final Logger LOG = LoggerFactory.getLogger(GTFSMain.class);

    public static void main (String[] args) throws Exception {
        Options options = getOptions();
        CommandLineParser parser = new DefaultParser();
        CommandLine cmd = parser.parse( options, args);
        String[] arguments = cmd.getArgs();
        if (cmd.hasOption("help")) {
            printHelp(options);
            return;
        }
        if (arguments.length < 1) {
            System.out.println("Please specify a GTFS feed to load.");
            System.exit(1);
        }
        File tempFile = File.createTempFile("gtfs", ".db");

        GTFSFeed feed = new GTFSFeed(tempFile.getAbsolutePath());
        feed.loadFromFile(new ZipFile(arguments[0]));

        if(cmd.hasOption("validate")) {
            feed.validate();
            JsonManager<ValidationResult> json = new JsonManager(ValidationResult.class);
            ValidationResult result = new ValidationResult(arguments[0], feed, new FeedStats(feed));
            String resultString = json.writePretty(result);
```

```
        File resultFile;
        if (arguments.length >= 2) {
            resultFile = new File(arguments[1]);
            FileUtils.writeStringToFile(resultFile, resultString);
            LOG.info("Storing validation result at: {}", resultFile.getAbsolutePath());
        } else {
            LOG.info("Printing validation result for {}", feed.feedId);

            System.out.print(resultString);
        }
    }
    feed.close();

    LOG.info("reopening feed");

    // re-open
    GTFSFeed reconnected = new GTFSFeed(tempFile.getAbsolutePath());

    LOG.info("Connected to already loaded feed");

    LOG.info("  {} routes", reconnected.routes.size());
    LOG.info("  {} trips", reconnected.trips.size());
    LOG.info("  {} stop times", reconnected.stop_times.size());
    LOG.info("  Feed ID: {}", reconnected.feedId);
}

private static void printHelp(Options options) {
    final String HELP = String.join("\n",
            "java -jar gtfs-lib-shaded.jar [options] INPUT.zip [result.json]",
            "Load input GTFS feed into storage-backed MapDB for use in Java-based",
            "applications. Optionally, write feed to output location. For more, see",
            "https://github.com/conveyal/gtfs-lib#gtfs-lib",
            "", // blank lines for legibility
            ""
    );
    HelpFormatter formatter = new HelpFormatter();
    System.out.println(); // blank line for legibility
    formatter.printHelp( HELP, options );
    System.out.println(); // blank line for legibility
```

```
    }

    private static Options getOptions () {
        Options options = new Options();
        Option help = new Option("help", false, "print this message");
        Option validate = new Option("validate", false, "run full validation suite on input GTFS feed (optionally
store at [result.json])");
        options.addOption(help);
        options.addOption(validate);
        return options;
    }
    public class GTFSCache extends BaseGTFSCache<GTFSFeed> {
    public GTFSCache(String bucket, File cacheDir) {
        super(bucket, cacheDir);
    }

    public GTFSCache(String bucket, String bucketFolder, File cacheDir) {
        super(bucket, bucketFolder, cacheDir);
    }

    @Override
    protected GTFSFeed processFeed(GTFSFeed feed) {
        return feed;
    }

    @Override public GTFSFeed getFeed (String id) {
        return this.get(id);
    }

public List<String> getOrderedStopListForTrip (String trip_id) {
        Iterable<StopTime> orderedStopTimes = getOrderedStopTimesForTrip(trip_id);
        List<String> stops = Lists.newArrayList();
        // In-order traversal of StopTimes within this trip. The 2-tuple keys determine ordering.
        for (StopTime stopTime : orderedStopTimes) {
            stops.add(stopTime.stop_id);
        }
```

```
        return stops;
    }

    public LineString getStraightLineForStops(String trip_id) {
        CoordinateList coordinates = new CoordinateList();
        LineString ls = null;
        Trip trip = trips.get(trip_id);

        Iterable<StopTime> stopTimes;
        stopTimes = getOrderedStopTimesForTrip(trip.trip_id);
        if (Iterables.size(stopTimes) > 1) {
            for (StopTime stopTime : stopTimes) {
                Stop stop = stops.get(stopTime.stop_id);
                Double lat = stop.stop_lat;
                Double lon = stop.stop_lon;
                coordinates.add(new Coordinate(lon, lat));
            }
            ls = gf.createLineString(coordinates.toCoordinateArray());
        }
        // set ls equal to null if there is only one stopTime to avoid an exception when creating
linestring
        else{
            ls = null;
        }
        return ls;
    }

public Geometry getMergedBuffers() {
        if (this.mergedBuffers == null) {
//            synchronized (this) {
                Collection<Geometry> polygons = new ArrayList<>();
                for (Stop stop : this.stops.values()) {
                    if (getStopTimesForStop(stop.stop_id).isEmpty()) {
                        continue;
                    }
```

```
                if (stop.stop_lat > -1 && stop.stop_lat < 1 || stop.stop_lon > -1 && stop.stop_lon < 1) {
                    continue;
                }
                Point stopPoint = gf.createPoint(new Coordinate(stop.stop_lon, stop.stop_lat));
                Polygon stopBuffer = (Polygon) stopPoint.buffer(.01);
                polygons.add(stopBuffer);
            }
            Geometry multiGeometry = gf.buildGeometry(polygons);
            this.mergedBuffers = multiGeometry.union();
            if (polygons.size() > 100) {
                this.mergedBuffers = DouglasPeuckerSimplifier.simplify(this.mergedBuffers, .001);
            }
//          }
        }
        return this.mergedBuffers;
    }

    public Polygon getConvexHull() {
        if (this.convexHull == null) {
            synchronized (this) {
                List<Coordinate> coordinates = this.stops.values().stream().map(
                        stop -> new Coordinate(stop.stop_lon, stop.stop_lat)
                ).collect(Collectors.toList());
                Coordinate[] coords = coordinates.toArray(new Coordinate[coordinates.size()]);
                ConvexHull convexHull = new ConvexHull(coords, gf);
                this.convexHull = (Polygon) convexHull.getConvexHull();
            }
        }
        return this.convexHull;
    }
```

ДОДАТОК В

Підключення бібліотек:

```
dependencies {
    implementation 'org.springframework:spring-core:4.3.17.RELEASE'
    implementation 'org.springframework:spring-web:4.3.17.RELEASE'
    implementation 'org.springframework:spring-beans:4.3.17.RELEASE'
    implementation 'org.springframework:spring-context:4.3.17.RELEASE'
    implementation 'org.springframework:spring-aspects:4.3.17.RELEASE'
    implementation 'org.springframework:spring-context:4.3.17.RELEASE'
    implementation 'org.springframework:spring-webmvc:4.3.17.RELEASE'
    implementation 'org.springframework:spring-websocket:4.3.17.RELEASE'
    implementation 'org.springframework:spring-context-support:4.3.17.RELEASE'
    implementation 'org.springframework:spring-tx:4.3.17.RELEASE'
    implementation 'org.springframework:spring-orm:4.3.17.RELEASE'
    implementation 'org.springframework:spring-oxm:4.3.17.RELEASE'
    implementation 'org.springframework.data:spring-data-jpa:1.11.12.RELEASE'
    implementation 'org.springframework.batch:spring-batch-core:3.0.9.RELEASE'

    implementation 'org.apache.logging.log4j:log4j-slf4j-impl:2.17.1'
    implementation 'com.goldmansachs:gs-collections:5.1.0'
    implementation 'javax.inject:javax.inject:1'
    implementation 'org.apache.commons:commons-lang3:3.5'
    implementation 'org.apache.velocity:velocity:1.7'
    implementation 'org.apache.velocity:velocity-tools:2.0'
    implementation 'org.hibernate:hibernate-core:5.2.12.Final'
    implementation 'org.hibernate:hibernate-ehcache:5.2.12.Final'
    implementation 'com.fasterxml.jackson.core:jackson-databind:2.9.5'
    implementation 'com.fasterxml.jackson.dataformat:jackson-dataformat-
csv:2.9.5'
    implementation 'org.apache.commons:commons-math3:3.6.1'
    implementation 'javax.activation:activation:1.1.1'
    implementation 'com.sun.mail:mailapi:1.6.2'
    implementation 'com.sun.mail:javax.mail:1.6.2'
    implementation 'org.aspectj:aspectjrt:1.8.13'
    implementation 'com.google.guava:guava:20.0'
    implementation 'com.google.code.gson:gson:2.8.5'
    implementation 'com.google.protobuf:protobuf-java:3.5.1'
    implementation 'org.olap4j:olap4j:1.2.0'
    implementation 'org.apache.poi:poi:3.17'
    implementation 'org.apache.poi:poi-ooxml:3.17'
    implementation 'joda-time:joda-time:2.9.9'
    implementation 'org.jfree:jcommon:1.0.24'
    implementation 'org.jfree:jfreechart:1.0.19'
    implementation 'net.sf.jasperreports:jasperreports:6.4.1'
    implementation 'com.lowagie:itext:2.1.7'
    implementation 'com.vividsolutions:jts-core:1.14.0'
    implementation 'com.squareup.okhttp:okhttp:2.7.5'
    implementation 'javax.annotation:javax.annotation-api:1.3.1'
    implementation 'javax.validation:validation-api:2.0.1.Final'
    implementation 'com.fasterxml.jackson.core:jackson-annotations:2.9.0'
    implementation 'com.goldmansachs:gs-collections-api:5.1.0'
    implementation 'javax.validation:validation-api:2.0.1.Final'
    implementation 'org.aspectj:aspectjweaver:1.8.9'
    implementation 'org.hibernate.javax.persistence:hibernate-jpa-2.1-
api:1.0.0.Final'
    implementation 'org.jboss.spec.javax.transaction:jboss-transaction-
api_1.2_spec:1.0.1.Final'
    implementation 'com.googlecode.protobuf-java-format:protobuf-java-
format:1.2'
    implementation group: 'com.graphhopper', name: 'directions-api-client',
```

```
version: '0.10.1-4'
    implementation group: 'com.corundumstudio.socketio', name: 'netty-socketio',
version: '1.7.17'
    implementation 'io.jsonwebtoken:jjwt:0.6.0'
    implementation group: 'org.mindrot', name: 'jbcrypt', version: '0.3m'
    implementation 'org.hashids:hashids:1.0.3'
    implementation group: 'com.jcraft', name: 'jsch', version: '0.1.44-1'
    implementation group: 'com.nordstrom.tools', name: 'remote-session',
version: '1.2.1'
```

ДОДАТОК Г