Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

“Інформаційна веб-система ідентифікації місць для паркування за даними сервісу OpenStreetMap”

Виконав: студент 2 курсу, групи 2ІСТ-21м
спеціальності 126 – «Інформаційні системи та технології»
_____ Гусак С. В.

Керівник: к.т.н., доц. каф. САІТ
_____ Крижановський Є. М.
«01» ____12____ 2022 р.

Опонент: доц. каф. АІІТ
_____ Богач І. В.
«15» ____12____ 2022 р.

Допущено до захисту

Завідувач кафедри САІТ
_____ д.т.н., проф. Мокін В. Б.
«05» ____12____ 2022 р.

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра системного аналізу та інформаційних технологій
Рівень вищої освіти – ІІ-й (магістерський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 126 Інформаційні системи та технології
Освітньо-професійна програма – Інформаційні технології аналізу даних та зображень

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Мокін В. Б.
«16» 09 2022 р.

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Гусаку Сергію Вікторовичу

1. Тема роботи: "Інформаційна веб-система ідентифікації місць для паркування за даними сервісу OpenStreetMap",
керівник роботи: Крижановський Є.М., к.т.н., доц. каф. САІТ,
затверджені наказом закладу вищої освіти від «14» 09 2022 року №203
2. Строк подання студентом роботи «01» 12 2022 року
3. Вихідні дані до роботи:

Набір даних по місцезнаходженню місць для паркування отриманих з OpenStreetMap.

4. Зміст текстової частини:
   - аналіз проблем ідентифікації місць паркування;
   - вибір оптимальних інформаційних технологій;
   - збирання даних та адаптація даних з OpenStreetMap;
   - розроблення веб-системи ідентифікації місць для паркування.
5. Перелік ілюстративного матеріалу:
   - REST-API веб-системи;
   - запит на OpenStreetMap;
   - зображення головної сторінки веб-системи;
   - заборонені місця для паркування;
   - місця для паркування;
   - модель роботи веб-сервісу;

6. Консультанти розділів МКР

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
5	Бурєннікова Н. В., д.е.н.. професор каф. ЕПВМ	10.11.22р	15.11.22р

7. Дата видачі завдання «16» 09 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів МКР	Строк виконання етапів роботи	Приміт
1	Аналіз проблем ідентифікації місць паркування	09.2022	
2	Основні етапи виконання роботи та огляд набору вхідних даних	09.2022	
3	Вибір оптимальних інформаційних технологій	10.2022	
4	Розробка веб-системи	10.2022	
5	Економічна частина	11.2022	
6	Оформлення матеріалів до захисту МКР	11.2022	

Студент Гусак С.В.

Керівник роботи Крижановський Є.

АНОТАЦІЯ

УДК 004.9+ 303.732.4

Гусак С. В. Інформаційна веб-система ідентифікації місць для паркування за даними OpenStreetMap. Магістерська кваліфікаційна робота зі спеціальності 126 – інформаційні системи та технології, освітньо-професійна програма – інформаційні технології аналізу даних та зображень. Вінниця: ВНТУ, 2022. 115с.

На укр. мові. Бібліогр.: 20 назв; рис.: 44; табл.: 5

В магістерській кваліфікаційній роботі проаналізовано можливість знаходження місця для паркування та відображення місць на карті у м. Вінниці за допомогою відкритих джерел та OpenStreetMap на Java та запропоновано можливість відображення частин проїжджої частини, де категорично заборонено зупинятись та паркуватись. Розроблено веб-систему, яка вирішує задачу ідентифікації місць для паркування.

Ілюстративна частина складається з 6 плакатів.

У розділі економічної частини розглянуто питання доцільності розробки та впровадження інформаційної технології визначення місць для паркування.

Ключові слова: паркування, автомобіль, геостатистичний аналіз, моніторинг, дорога, геоінформаційна система.

ABSTRACT

Husak S. V. Information web system for identification of parking places based on OpenStreetMap data. Master's qualification thesis on specialty 126 - information systems and technologies, educational and professional program - information technologies of data and image analysis. Vinnytsia: VNTU, 2022. 115p.

In Ukrainian speech Bibliography: 20 titles; Fig.: 44; tab.: 5

The master's thesis analyzed the possibility of finding a parking place and displaying places on the map in Vinnytsia using open sources and OpenStreetMap in Java, and proposed the possibility of displaying parts of the roadway where it is strictly forbidden to stop and park. A web system has been developed, which solves the problem of identifying parking spaces.

The illustrative part consists of 6 posteri.

In the section of the economic part, the issue of the feasibility of developing and implementing information technology for determining parking spaces is considered.

Keywords: parking, car, geostatistical analysis, road, geoinformation system.

ЗМІСТ

ВСТУП

Актуальність теми. Очевидно, що збільшення кількості автомобілів на дорогах, паркування транспортних засобів, потребує збільшення паркувальних місць. Але є такі частини міст в яких немає можливості збільшити кількість паркувальних місць що в підсумку спричиняє перевантаження цієї частини міста. Також не налагоджений механізм відстеження коштів за паркування та притягнення порушників до відповідальності, перешкоди та обмеження руху пасажирів та водіїв, створення корупційних схем. Іншими словами, кошти які мали б наповнювати бюджет міста та витрачатись на вже існуючих та створення нових майданчиків для паркування, не використовуються за призначенням. Також, постає питання неефективності чинного регулювання, яке має негативний вплив на громаду, бізнес та владу.

Мета і завдання роботи. Метою даної роботи є розробка веб-системи для підвищення точності ідентифікації місць для паркування на основі даних відкритих джерел та OpenStreetMap у місті Вінниця.

Процес розробки інформаційної веб-системи ідентифікації місць для паркування передбачає виконання таких задач:

– збирання вхідних даних по парковках для відображення їх на карті;

– формування програмних вимог та вибір оптимальних інформаційних технологій;

– створення бази даних системи;

– розробка інформаційної веб-системи ідентифікації місць для паркування;

– перевірка інформаційної веб-системи на практиці.

Об’єктом дослідження магістерської кваліфікаційної роботи є процес ідентифікації місць для паркування на основі даних відкритих джерел та OpenStreetMap.

Предметом дослідження магістерської кваліфікаційної роботи є технології та засоби ідентифікації та відображення паркувальних місць на карті.

Новизна одержаних результатів. Дістала подальший розвиток інформаційна технологія ідентифікації місць для паркування для міста Вінниця, яка використовує актуальні дані сервісу OpenStreetMap, які постійно оновлюються користувачами з усього світу.

Практичне значення роботи полягає у можливості використання розробленої інформаційної системи для ідентифікації місць паркування, а також місць, де паркування заборонене.

Апробація результатів магістерської кваліфікаційної роботи. Результати роботи доповідались на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2022-2023 рр.).

Публікації результатів магістерської кваліфікаційної роботи. Опубліковано тези на Всеукраїнській науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2022-2023 рр.) [1].

1 АНАЛІЗ ПРОБЛЕМ ІДЕНТИФІКАЦІЇ МІСЦЬ ПАРКУВАННЯ

1.1 Суть технічної проблеми

Розумна парковка – це спеціалізоване паркувальне місце, створене за допомогою датчиків і сучасних технологій для швидкого та легкого пошуку паркувальних місць, забезпечення безпеки та автоматизації процесу паркування автомобіля на стоянці. Створення спеціальних парковок почалося майже одночасно з появою перших автомобілів. Кількість автомобілів стрімко зростає і впроваджуються сучасні технології, що дозволяють вирішити проблему обмежених місць для паркування.

Паркування – основна проблема наших міст. Адже у наш час дуже важко знайти місце щоб припаркувати авто і Вінниця не стала винятком. Як повідомляє Вінницька міська рада, сьогоднішнім рішенням виконкому затверджено проект рішення міської ради «Про затвердження зональної схеми ВМТГ для встановлення ставок збору за місця для паркування транспортних засобів». Загалом визначено п’ять зон – враховувались трафіку на дорогах та концентрація ділової активності на територіях. Ще зовсім недавно для багатьох з нас автомобіль вважався розкішшю, а не засобом пересування. Час змінив наше ставлення до цього. Але хаотичне розміщення автівок стало великою проблемою для усіх міст, і Хмельницький не став винятком. Адже більшість багатоповерхівок в місті збудовані за радянських часів, коли проектанти, як правило, передбачали у дворах стійкі для вибивання килимів, а в кращому випадку – декілька гойдалок та пісочниць. До формули “одна сім’я – один автомобіль”, не кажучи вже про два автомобілі на сім’ю, ніхто з архітекторів серйозно не ставився [2]. Відтак виконати її зараз на практиці складно. У нових житлових комплексах забудовники намагаються принаймні виконувати передбачену державними будівельними нормами квоту парко місць, яка змінюється залежно від року затвердження проекту Втім, якщо підрахувати, то автостоянка в дворі має займати вдвічі більше

землі, ніж будинок, оскільки зараз багато людей мають авто. У Хмельницькому під час капремонтів прибудинкових територій обов’язково передбачені місця для пар кування автомобілів, але все одно для тимчасового паркування таких місць бракує (рис. 1.1).

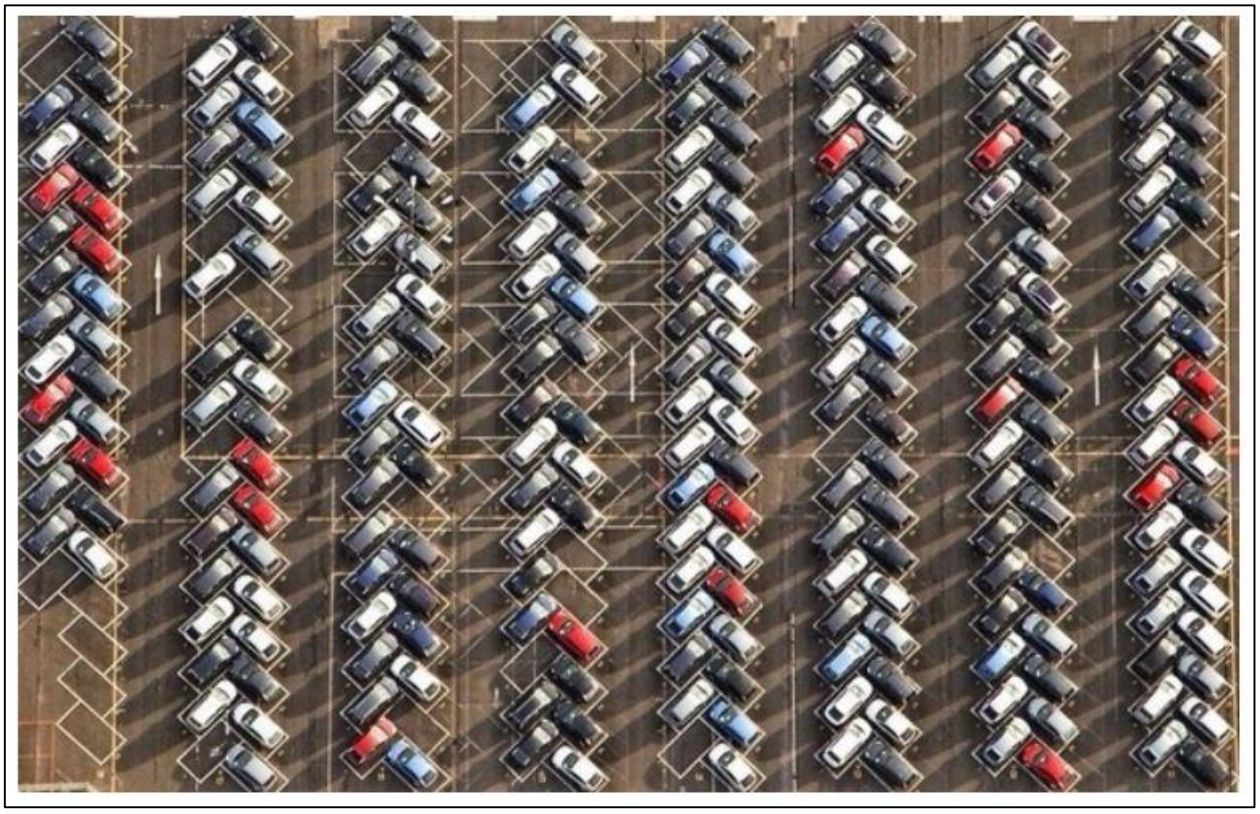


Рисунок 1.1 – Приклад паркувальної зони

Ще більш гостро ця проблема відчувається в центрі міста, адже автомобілів, які потрібно десь прилаштувати на час відсутності власника, так само більшає. Лише торік в Україні зареєстрували 32,8 тис. вживаних легкових авто, які ввезли з-за кордону. Це на 35% більше, ніж в серпні 2019-го. Серед власників “євроблях” чимало й хмельничан. Відтак безлад з паркуванням машин у недозволених місцях справді зашкалює. “Майстри парковок” нерідко залишають автомобілі і на газонах, і на дитмайданчиках, і на “зебрі”, і на тротуарах [2]. А дехто, прибортовуючи своє авто, взагалі перекриває смугу для руху іншим водіям, і щоб його об’їхати, доводиться перетинати подвійну

суцільну смугу. Все це створює не лише незручності для водіїв та пішоходів, а й аварійні ситуації.

Пандемія ще більше загострила цю проблему, оскільки у громадському транспорті почали діяти карантинні обмеження. Дозволяється перевозити пасажирів лише відповідно до кількості сидячих місць. Відтак мешканці міста почали більше користуватися власним транспортом, який потрібно десь припаркувати.

Майже два роки тому в Україні набрали чинності нові правила паркування. Наприклад, тепер можна не чекати власника авто для того, щоб вручити штраф. Досить сфотографувати порушення, а квитанцію надіслати поштою. Але правила так і не почали діяти, адже система збору інформації та фотофіксації досі на 100 відсотків не запрацювала. Тому інфраструктура для збору та аналізу даних почне отримувати відомості не лише з камер, але й від інспекторів та поліцейських.

У Хмельницькому додатковим інструментом контролю стане відділ інспекції з паркування. Його створять при управлінні транспорту та зв'язку Хмельницької міської ради. За словами заступника міського голови Миколи Ваврищука, орієнтовно він запрацює всередині квітня. Інспектори матимуть повноваження притягувати до відповідальності автомобілістів, які залишили свій транспортний засіб у непередбаченому для цього місці. Втім, Микола Ваврищук вважає, що інспекція глобально не вирішить проблему хаотичного паркування, але її створення важливе для впровадження інших комплексних заходів.

"Серед запланованих нами заходів є будівництво нових паркінгів, зокрема багаторівневих. А в окремих місцях створюватимемо платні парковки, адже багато водіїв залишають свої автомобілі на вулицях центральної частини міста. Через це інші жителі міста не можуть припаркуватись на короткий час", – розповів Микола Ваврищук [3].

Аби дізнатися, що думають містяни з цього приводу, 16 березня, стартувало онлайн-опитування, яке проводила міськрада. Тож хмельничани

мали можливість висловити свої думки та запропонувати своє бачення щодо розв’язання цієї проблеми.

Серед найголовнішого:

– Проблеми з паркуванням авто відчувають багато водіїв;

– Більшість підтримує ідею початку роботи інспекції з паркування;

Більше 73% людей підтримують ідею платного паркування в центрі та 56% – біля ринків. Загалом від містян надійшло майже 700 пропозицій. Після ретельного аналізу усі побажання містян мають лягти у стратегію розв’язання проблеми паркування.

За словами Миколи Ваврищука, такі опитування будуть проводитись й надалі, аби окреслити проблему в кожному мікрорайоні міста

Міста є ключовими економіко-географічними та адміністративними одиницями сучасного світу. Людство в історії вже колись проходило стадію міст-держав – самодостатніх політичних та економічних гравців на континентальних картах. Історія, як відомо, повторюється по спіралі, і сучасні мегаполіси багато в чому знову стають такими гравцями.

Найбільші міста світу за чисельністю населення перевищують, іноді в рази, цілі країни і стають аренами неймовірних перетворень. Стрімка урбанізація та темпи розширення міст змінюють підхід до планування та розвитку їх інфраструктури. Як невеликий приклад, за даними Міжнародної організації з міграції (МОМ), 3 мільйони людей мігрують до міст щотижня – це еквівалент нового Мадрида чи Буенос-Айреса. Розвиток будівельних технологій дозволив перейти до вертикального просторового розширення. Значно щільніше стало транспортне сполучення, зросло інформаційне навантаження, зменшилася доступна житлова площа на одного мешканця. При цьому рівень потреб мешканців не лише виріс, а й якісно змінився та розширився [3].

Як наслідок, міста та агломерації стикаються з новими викликами, а мешканці мають нові очікування та нові потреби. Мегаполіси вчаться

надавати додаткові послуги своїм жителям, намагаючись зробити їхнє життя легшим, безпечнішим і комфортнішим.

Таке поєднання факторів призвело до необхідності пошуку нових рішень, нових парадигм відносин між містом і його населенням. Міста змушені переходити від формату знеособленої «території для виживання та задоволення базових потреб жителів» до формату самоідентифікації міста як «живої істоти», визнаючи мешканців суб'єктами, а не об'єктами урбаністики. життя – і, як наслідок, до інтерактивної комунікації між містом і такими суб'єктами. як колективний диференційований розум. Іншими словами, перетворити на «розумне» місто, на Smart City.

Ця трансформація, у міру появи нових технологічних рішень, найчастіше відбувається органічно, в локальному масштабі, в результаті випадкового збігу фінансових можливостей міста чи окремих його адміністративних одиниць з бажаннями та потребами різних зацікавлених сторін у конкретному адміністративна одиниця в певний момент часу [4].

Водночас сила трансформації безсумнівна, але вона також таїть у собі високі ризики – ризики дисгармонії та розбіжності між окремими технологічними рішеннями в загальноміському масштабі. Це безпекові ризики, тупики в логістичному управлінні міськими потоками, дисбаланс між комфортом мешканців і виконанням критичних міських сервісних завдань. Для мінімізації цих ризиків процес перетворення на «розумне» місто має відбуватися відповідно до середньо- та довгострокового плану, узгодженого всіма зацікавленими сторонами, а безпосередня реалізація – на основі єдиної технологічної платформи. Давайте трохи глибше розглянемо теорію розумних міст і існуючі технологічні рішення. «Розумне» місто – це місто, яке «слухає» та пристосовується до потреб своїх мешканців.

Smart City або «розумне місто», яке дає мешканцям більший комфорт, безпеку, екологічність, порівняно з сучасним мегаполісом, вже не просто винахід мрійників-фантастів. Це реальні концепції, окремі елементи яких реалізуються в Нью-Йорку та Сінгапурі, Токіо та Берліні, Чикаго та Києві.

Південна Корея будує перше розумне місто, спроектоване з нуля. І незабаром Сонгдо відкриє нову еру міського розвитку – еру «smartpolis».

Основний напрямок розвитку – «розумні» парктроніки. Такі датчики вбудовуються в проїжджу частину на паркувальних місцях і контролюють зайняте або вільне місце над ними, передаючи дані в загальну систему. За допомогою мережі таких датчиків створюється карта паркування, статус якої передається користувачам на вулицях за допомогою спеціальних екранів або мобільного додатку. Є багато можливостей збільшити потенціал розумних датчиків паркування. Однією з нових концепцій закритих паркінгів є навісні датчики, які додатково виконують функцію охорони та моніторингу автомобіля. Ще одним напрямком розумного паркування є розробка та впровадження автоматизованих парковок (найчастіше багаторівневих), на яких дії водіїв зведені до мінімуму. Водій заходить на спеціальну зону платформу і виходить з автомобіля. Далі платформа сама переносить автомобіль у спеціально відведене, відведене або вільне місце та повідомляє водієві його номер. Щоб отримати транспортний засіб, водієві необхідно авторизуватися та ввести цей номер на спеціальному табло чи панелі керування, після чого платформа також автоматично опустить автомобіль на майданчик.

Міста завжди були платформами для реалізації нових можливостей. Останні оцінки світової економіки свідчать, що 80% світового ВВП виробляється в містах. Сьогоднішні мегатренди швидкої урбанізації, на фоні глобальних змін клімату та виснаження ресурсів, набувають особливої актуальності для сучасних міст [4].

Міста починають вирішувати нові категорії задач, стаючи площадками величезних інновацій та полігонами для впровадження новітніх технологій, відкриваючи нові можливості для тестування та впровадження винаходів. Міста відкривають широкий спектр можливостей для постачальників і споживачів інтелектуальних технологій, котрі можуть допомогти

оптимізувати споживання ресурсів і поліпшення якості послуг за рахунок більш ефективного управління попитом і пропозицією.

«Smart city» («Розумне місто») – місто, в якому для підвищення якості життя міської спільноти активно використовуються сучасні інформаційні технології. Інформаційні технології «Розумного міста» «вбудовуються» у виробничі процеси міських структур з метою якіснішого надання послуг, зменшення вартостей споживаних ресурсів, поліпшення комунікації та взаємодії влади з містянами, розширення спектру послуг надаваних туристам.

Проте деякі з ідей «Розумного міста» виявились недостатньо ефективними за рахунок використання занадто дорогого спеціалізованого обладнання, а не дешевших рішень на основі мережі Інтернет. Часто застосунки «Розумного міста» демонструють технологічно цікаві ідеї, не реагуючи на реальні потреби громадян та пропонують завищені обіцянки, які не можуть бути ефективно реалізовані.

В країнах європейського союзу зазначена системна концепція включає наступні напрямки: мобільність інфраструктури, управління навколишнім середовищем, управління міською спільнотою, управління міським господарством, розвиток та підвищення ефективності економіки.

В переважній більшості відомих на даний час проектів «Smart city» чітко не окреслено вичерпний перелік та профільність використання інформаційних технологій. Діючі проекти в основному спрямовані на вирішення проблем енергоефективності, оптимізації трафіку міського транспорту, прикладних застосунках на базі геоінформаційних технологій і т. ін. Це зумовлено, зокрема, необхідністю вирішенням біжучих проблем функціонування міста як складної багатопрофільної та багатокомпонентної системи. Інформаційні технології при цьому покликані зіграти роль інтегруючих ланок між різними інфраструктурними рівнями та підсистемами і забезпечувати надійний та якісний обмін інформацією між ними[5].

В подібній ситуації, яка потребує суттєвого поліпшення, знаходиться переважна більшість українських міст, які декларують необхідність реалізації

перспективних проектів «Розумних міст». Розробляються окремі елементи та ланки інформаційних технологій «Розумного міста».

В обласних центрах та великих містах реально забезпечується вільний доступ до WiFi, реалізується моніторинг руху громадського транспорту, функціонують комплекси мереж «розумних» світлофорів, підтримуються on-line сервіси з оплати комунальних послуг і т. ін.

На даний час вичерпно не опрацьовані системні комплексні підходи до формування цілісного інформаційно-технологічного базису, необхідного для якісної реалізації проектів «Розумного міста».

Для кваліфікованого вирішення проблем, пов’язаних з формуванням класів ІТ для реалізації проектів «Розумне місто», зокрема, до розробки методологічних засад та способів застосування сучасних інформаційних технологій, що дозволяло б враховувати як загальнодержавні, так і місцеві інтереси, забезпечувати раціональне використання людських, технологічних, природних, матеріальних та фінансових ресурсів, без сумніву слід застосовувати комплексний системний підхід [6].

Дослідження таких складних комплексів інформаційних технологій пов’язане з об’єктивними складнощами, які зумовлені зокрема наступними причинами:

– класи інформаційних технологій, що використовуються у сучасних містах, належать до класу складних систем, які характеризуються великими множинами кількісних і якісних параметрів;

– між зазначеними технологіями існують різнотипові зв’язки різної природи та інтенсивності, котрі суттєво можуть змінюватись в часі;

– комплексні системи управління сучасних міст мають слабку організаційну структурованість, є доволі нестійкими та достатньо хаотичними; явища та параметри, які характеризують системи, у багатьох випадках не мають стійких імовірнісних розподілів;

– складна система, якою є сучасне місто, визначається параметрами, що динамічно змінюються і відображають рівень знання щодо таких систем,

за якого неможливо однозначне визначення їхніх морфологічних і функціональних особливостей.

Стан комплексної адміністративно-господарської та інформаційно-технологічної складної системи, якою є система класу «Розумне місто», визначається її функціональним наповненням, характеристиками внутрішніх процесів, процедурами її взаємодії із зовнішнім середовищем, державними, муніципальними та громадськими чинниками .

Можна виділити базові напрямки використання інформаційних технологій в проектах «Розумне місто»:

– використання великих масивів даних та аналітичного їх опрацювання з метою підвищення якості інформаційних послуг;

– мобільне збирання даних та оперативне реагування на проблемні ситуації;

– віртуальна взаємодія між різнорідними установами, структурами та мешканцями міста;

– активне використання соціокомунікаційних технологій, що покращують взаємодію місцевих органів влади з громадянами.

Для підготовки та прийняття управлінських рішень, які будуть сприяти досягненню мети, інформаційні технології в проектах «Розумного міста» повинні реалізовувати сучасні інтелектуальні методи та підходи до розв’язання широкого спектру задач, опиратись на гіпервеликі бази даних. Одним з базових напрямків подолання проблем інтеграції комплексу різнопланових інформаційних технологій у проектах «Розумне місто» є використання програмно-алгоритмічних застосунків, які реалізують процеси побудови відповідних онтологій, інтерпретують методи вирішення задач пошуку інформації, системи формування профільних рекомендацій, адаптивні процедури вибору моделей і критеріїв оцінки управлінських рішень. Такий підхід дає змогу формувати повний та якісний інтегрований простір даних та знань «Розумного міста» [7].

1.2 Огляд існуючих методів вирішення технічної проблеми

Використання локальних систем позиціонування для різних об'єктів є однією з гарячих тем у різних сферах суспільства. У різних ситуаціях потрібні знання того, як визначити місцезнаходження в певний момент часу (в реальному часі). Для цього проміжок часу між вимірюваннями повинен бути таким, щоб об'єкт, рухаючись з характерною для нього швидкістю, встигав подолати відстань не більше ніж у подвоєну точність позиціонування. Наприклад, щоб забезпечити позиціонування (в реальному часі) з точністю до 1 м для людини, яка рухається зі швидкістю 5,4 км/год або (1,5 м/с), вимірювання слід проводити з частотою принаймні приблизно один раз на 1,3 секунди [6].

Муніципальна система моніторингу трафіку також може бути інтегрована в середовище «розумного» міста для відслідковування заторів та ситуації на дорогах в місті. Хоча системи моніторингу руху на основі відеокамер вже доступні та розгорнуті у багатьох містах, широко поширене спілкування з низьким рівнем споживаної потужності може забезпечити щільний інформаційний потік. Моніторинг руху може бути реалізований за допомогою сенсорних засобів та GPS, встановлених на сучасних транспортних засобах, в комплексі з давачами якості повітря та акустичними давачами, розміщеними вздовж однієї і тої ж самої дороги. Ця інформація є надзвичайно важливою для муніципальної влади та громадян з метою дисциплінування трафіку й опреативного спрямування дорожніх офіцерів в проблемні зони та попереднього планування маршрутів.

Муніципальна система моніторингу енергоспоживання: Разом із службою моніторингу якості повітря, може надавати послугу моніторингу енергоспоживання всього міста, що дає змогу органам влади та громадянам отримати чітке та детальне уявлення про обсяг енергії, необхідний для різних пос луг (громадс ьке ос вітлення, транс порт, с вітлофори, контрольні відеокамери, опалення/охолодження громадських будівель тощо). У свою

чергу, це дозволить визначити основні джерела енергоспоживання та встановити пріоритети для оптимізації їх поведінки. Це співпадає з вектором сформованим Європейською директивою щодо підвищення енергоефективності в найближчі роки. Для реалізації такої послуги, пристрої моніторингу спожитої електричної потужності повинні бути інтегровані з енергосистемою міста. Крім того, буде також можливе посилення цієї служби за допомогою активних функцій для контролю місцевих енергетичних виробничих структур (наприклад, фотоелектричних панелей).

Сервіс «Розумні парковки» містить дорожні давачі та інтелектуальні дисплеї, які спрямовують водіїв підказуючи найкоротший шлях до найближчого паркування у потрібній локації. Переваги, отримані від цієї послуги, різноманітні: скорочення часу пошуку паркування, зменшення

викидів CO від автомобіля, зменшення заторів і кількості дорожніх пригод. Служба «розумного паркування» може бути інтегрована в міську інфраструктуру, оскільки багато компаній в Європі пропонують продукти для її реалізації. Крім того, за допомогою технологій короткотривалої комунікації, таких як ідентифікатори радіочастот («RFID») або приповерхневої комунікації («NFC»), можна реалізувати електронну системи контролю доступу для зареєстрованих жителів чи інвалідів, що забезпечує кращий сервіс для громадян, які можуть легітимно використовувати цільові слоти і ефективний інструмент для швидкого виявлення порушень [7].

«Розумне муніципальне освітлення»: для підтримки директиви 20-20- 20 оптимізація ефективності вуличного освітлення є важливою особливістю. Зокрема, ця послуга може оптимізувати інтенсивність вуличного освітлення відповідно до часу доби, погодних умов та присутності людей. Для того, щоб належним чином працювати, така послуга повинна включати вуличні ліхтарі, інтеровані в інфраструктуру «розумного міста». Також можливо використати збільшену кількість підключених точок, для забезпечення повсюдного підключення «WiFi». Крім того, система виявлення несправностей може бути легко реалізована поверх контролерів вуличного освітлення.

Контролюючи зазначені параметри, можливо підвищити рівень комфорту осіб, які знаходяться у цих середовищах, що також може мати позитив з точки зору їх продуктивності праці, при одночасному зменшенні витрати на опалення чи кондиціонування.

Платформа «Citrix XenMobile» пропонує власний набір програмних продуктів і рішень для проектів класу «розумне місто». Компоненти дозволяють вирішувати завдання управління мобільними пристроями, застосунками і мобільним контентом, забезпечують безпеку роботи пристроїв і захист даних. Платформа забезпечує підключення до безпечного мережевого шлюзу для підключення до муніципальних сервісів. Управління муніципальними сервісами здійснюється централізовано одним застосунком для забезпечення мобільності в умовах «розумного міста». Платформа має власне сховище мобільних застосунків. Для розробки мобільних застосунків платформа пропонує пакет «Citix MDX Toolkit». Даний «SDK» скорочує час розроблення міських мобільних застосунків, пропонуючи набір готових функцій для розв'язання задачі їх обгортки, що працюють на пристроях «Android/IOS», в безпечну оболонку та інтеграції із загальною муніципальною системою.

Програмні продукти та технології «MobileIron» дозволяють провести інтеграцію будь-якої мобільної платформи з існуючими муніципальними ресурсами. При цьому основний внесок технологія вносить на захист переданих даних на муніципальні пристрої, можливість відслідковування мобільного пристрою і зниження ризиків втрати конфіденційної інформації. Безпека переданих муніципальних даних забезпечується за допомогою інтелектуального шлюзу. «Mobile Iron» може забезпечувати управління мобільністю підприємства не тільки для власних ресурсів, але і за допомогою хмарних технологій. Для розробників власних муніципальних застосунків [8].

«MobileIron» пропонує пакет «Mobile AppConnectSDK», призначений для упаковки мобільних застосунків в контейнери, які підтримують захисту

конфіденційної інформації, безпечну передачу даних і можливість зміни застосунків.

Платформа «Symantec Mobility Suite» пропонує с тандартний набір програмних продуктів для вирішення завдань бізнесу: управління мобільними пристроями, застосунками і контентом, а також забезпечує захист від робо інформаційної безпеки. Платформа дозволяє призначати політики безпеки і доступу до мобільних пристроїв і муніципальних даних. Платформа пропонує власний «SDK» для розроблення безпечних контейнерів мобільних застосунків і їх упакування [8].

Платформа «SOTI» дозволяє управляти мобільними пристроями віддалено і пропонує комплексну конфігурацію роботи застосунків в поєднанні з налаштуванням правил користування пристроями. Платформа забезпечує багатофакторну авторизацію з інформаційній системі «розумного міста». «SOTI» пропонує потужний «SDK» у формі «Android for Work» для розроблення мобільних застосунків. Недоліком можна вважати відсутність можливості упаковки застосунків в безпечну оболонку.

Платформа «MaaS 360 FiberLink» пропонує набір різних програмних продуктів для управління мобільними пристроями, мобільними застосунками, а також забезпечує безпеку і захист даних на муніципальних сервісах. Платформа надає хмарне середовище для роботи застосунків і зберігання даних. Для роботи мобільних пристроїв платформа пропонує безпечний браузер і пошту, весь контент знаходиться в хмарі.

Платформа пропонує для розробників власний пакет для розроблення як нативних, так і гібридних мобільних застосунків. Для розробки нативних застосунків платформа підтримує «Android» та «IOS» пристрої. Розробка гібридних мобільних застосунків побудована на використанні фреймворка

«Apache Cordova».

Платформа «AirWatc h» забезпечує управління мобільними

середовищами, зручний доступ до муніципальних застосунків і захист даних. Платформа пропонує різні сценарії роботи з мобільним пристроєм в

рамках одного рішення. Управління організовано від моменту розробки програми і до моменту її використання. Платформа також пропонує власний «SDK» для розроблення мобільних застосунків під свою платформу. Готові функції, що поставляються «SDK» – це управління мобільними пристроями, застосунками і даними.

Платформа BlackBerry Enterprise Mobility Suite пропонує повний набір компонентів для забезпечення управлінням мобільними пристроями, застосунками і контентом, авторизації користувачів і інструментів для розробки контейнерів безпеки мобільних застосунків.

Платформа «Good» містить стандартні інструменти для забезпечення управління муніципальною мобільністю. Основна перевага платформи в тому, що вона застосунково дає можливість відокремити призначені для користувача дані і застосунки від муніципальних на мобільних пристроях. Виконання концепції «BYOD» досягається за рахунок використання спеціальних контейнерів для обгортки мобільних застосунків.

Програмно-алгоритмічна платформа «Enterprise Mobility Manager» подана складається з наступних модулів:

– Модуль управління мобільними пристроями призначений для конфігурації мобільних пристроїв, призначення прав доступу до функцій пристрою і ролей кожному користувачеві міської системи класу «розумне місто» і забезпечення моніторингу дії користувача;

– Модуль управління мобільними застосунками зберігає і публікує застосунки для муніципальних пристроїв. Завантаження і установка програмного забезпечення може здійснюватися віддалено;

– Модулі для мобільних платформ надають доступ до функцій пристрою;

– Консоль «EMM» надає користувачам веб-інтерфейс для роботи [9]. Зареєстрований користувач має певний набір прав для налаштування необхідних конфігурацій залежно від призначеної йому ролі: адміністратор, звичайний користувач, тощо;

– Мобільний агент «WSO2 Android Agent» встановлюється на мобільний пристрій. При запуску агента у користувача з'являється можливість вибору адреси для підключення і авторизації в системі «Enterprise Mobility Manager».

Передача команд і даних між встановленим агентом і платформою «WSO2 EMM» здійснюється за допомогою шлюзу мобільних пристроїв. Даний шлюз також здійснює авторизацію і аутентифікацію мобільного пристрою по протоколу «OAuth 2.0». Завантаження та оновлення застосунків відбуваються за допомогою «Push Notification Connector».

Використання RTLS (систем визначення місця розташування в реальному часі) для систем розумного паркування залежить від завдань і цілей.

Smart Parking - Додаток безкоштовний, але послуга паркування платна. Додаток «Smart Parking» доступний для завантаження як для Android, так і для iOS, і містить безліч функцій, призначених для того, щоб допомогти водіям направляти автомобілі на доступні місця для паркування, повідомляти про умови та тарифи на автостоянці, а також використовувати безконтактні платежі (рис. 1.2).

Рисунок 1.2 – Логотип Smart Parking

4Park Smart Parking App - Доступна для Android та iOS та дозволяє взаємодіяти з усіма системати Smart Parking [9]. Це дозволяє виконувати операції – оплатити та забронювати паркувальне місце, отримати спеціальні дозволи (ділові, туристичні автобуси, для інвалідів), паркування автомобіля, проїзд до безкоштовної стоянки, віртуалізація карток та оплата за паркування (рис. 1.3).

Рисунок 1.3 –Логотип 4Park Smart Parking App

GCC Smart Parking – GCC Smart Parking розроблено з кількома функціями технології автоматизованого паркування, опцію живого чату для розміщення запитів, пов’язаних з платежами, історією бронювання, політикою скасування. Оплату за бронювання можна здійснювати з їхніх телефонів за допомогою інтернет-банкінгу. У разі скасування, сума автоматично повертається на банківський рахунок користувача [9]. Доступний для Android та iOS (рис. 1.4).

Рисунок 1.4 – Логотип GCC Smart Parking

Найбільшою групою, що поділяється на кілька підгруп, є радіолокаційні технології. Ті ж, у свою чергу, діляться на стандартні технології передачі даних (Wi-Fi, Bluetooth, ZigBee), так чи інакше призначені для вимірювання відстаней, і ті, які, виходячи з фізичних властивостей модуляції, найбільш придатні для вимірювання відстані (UWB, NFER та інші) [6].

Технологія UWB (Ultra Wideband) використовує короткі імпульси з максимальною пропускною здатністю на мінімальній центральній частоті. Для

більшості виробників центральна частота становить кілька гігагерц, а відносна смуга пропускання становить 25-100%. Технологія використовується у зв’язку, радіолокації, вимірюванні відстані та позиціонуванні [8].

Виробники пропонують різні варіанти технології UWB. Існують різні види імпульсів. В одних випадках використовуються відносно потужні одиничні імпульси, в інших – сотні мільйонів імпульсів малої потужності в секунду. Використовується як когерентна (послідовна) обробка сигналу, так і некогерентна обробка. Усе це призводить до значної невідповідності характеристик СШП систем різних виробників.

Фундаментальні обмеження технології UWB [9]:

– важко побудувати систему зі значною потужністю передачі (типова потужність передавача 50 мкВт, дуже потужна 10 мВт);

– обмеження з боку частотних регуляторів (у зв'язку з цим системи зазвичай використовуються в приміщеннях, де їх малопотужний шумоподібний сигнал не впливає на інші системи і навіть не проявляється).

Переваги:

– високий рівень завадостійкості;

– висока безпека;

– чим вища частота, тим вища точність (але менший діапазон).

Недоліки:

– малий радіус дії (до 10 м);

– складна інфраструктура.

Методи позиціонування: AoA, ToF, ToA, TDoA у Wi-Fi. Wi-Fi – це технологія передачі даних середньої дальності, яка зазвичай охоплює десятки метрів і використовує неліцензовані діапазони частот для забезпечення доступу до мережі. Оскільки спочатку Wi-Fi не передбачалося використовувати як технологію локального позиціонування, стандартна мережа надає інформацію з точністю лише до точки доступу, тому RSSI або, з деякими модифікаціями, інші спеціалізовані методи (наприклад, TDoA) використовуються для підвищити точність локації [6–7].

Переваги:

– широке застосування;

– низька вартість обладнання.

Недоліки:

– для підвищення точності потрібне збільшення щільності базових станцій;

– перевантаження ефірного Wi-Fi;

– недостатня точність позиціонування для ряду завдань;

– навіть при використанні спеціальних подовжувачів Wi-Fi (в ідеальних умовах 3-5 метрів, реально 10-15 метрів).

Методи визначення позиції: RSSI на основі TDoA у ZigBee. ZigBee – це стандарт для набору протоколів зв’язку високого рівня з використанням малих малопотужних цифрових трансиверів на основі стандарту IEEE 802.15.4 для бездротових персональних мереж.

Технологія ZigBee дозволяє створювати бездротові мережі з автоматичною ретрансляцією самоорганізованих і самовідтворюваних повідомлень. ZigBee розроблений для мобільних пристроїв, які потребують гарантованого захисту

передача даних на відносно низьких швидкостях і можливість тривалої роботи мережевих пристроїв від автономних джерел живлення (батарей) [7-8].

Стандарт ZigBee передбачає використання частотних каналів в діапазонах 868 МГц, 915 МГц і 2,4 ГГц. Найвища швидкість передачі даних і високий опір досягаються в діапазоні 2,4 ГГц. Тому більшість виробників мікросхем випускають приймачі саме для цього діапазону, який передбачає 16 частотних каналів з кроком 5 МГц.

Швидкість передачі даних разом із службовою інформацією в ефірі становить 250 Кбіт/с. При цьому середня пропускна здатність вузла для корисних даних залежно від завантаженості мережі та кількості повторних передач може бути в межах 5...40 кбіт/с [9].

Відстані між вузлами мережі складають десятки метрів при роботі в приміщенні і сотні метрів на відкритому повітрі. Завдяки ретрансляції зона покриття мережі може значно збільшитися [9].

Переваги:

– підтримує як прості топології мережі («точка-точка», «дерево» і «зірка»), так і mesh-топологію з ретрансляцією і маршрутизацією повідомлень;

– містить можливість вибору алгоритму маршрутизації залежно від вимог програми та стану мережі;

– простота розгортання, обслуговування та модернізації;

– здатність до самоорганізації та самовідтворення;

– низьке енергоспоживання.

Недоліки:

– низька швидкість передачі даних.

NanoLOC – це технологія від Nanotron Technologies, яка багато в чому схожа на старішу версію NanoNET. Крім швидкості передачі інформації в 1 Мбіт за секунду на відстані в кілька сотень метрів, ця технологія дозволяє визначати відстань між приймачами. Похибка визначення відстані становить 2 метри, що дозволяє визначити, де знаходиться трансивер по відношенню до інших подібних трансиверів. Якщо потрібне визначення в тривимірній системі координат, знадобляться чотири (або більше) передавача NanoLOC, координати розташування яких уже відомі [10].

Переваги:

– можливість роботи в неліцензійних діапазонах частот з потужністю до 100 мВт;

– методи визначення місця розташування, що забезпечують можливість локалізації об'єктів за межами периметра зони обслуговування зі зниженням точності;

– великий вибір готового програмного забезпечення (з відкритими кодами);

– автокореляційні властивості сигналу роблять технологію стійкою до зовнішнього шуму.

Недоліки:

– обмеження на кількість пристроїв у сегменті;

– власна розробка.

Bluetooth – це специфікація бездротових персональних мереж малого радіусу дії, що працюють у діапазоні частот 2,4–2,4835 ГГц. У Bluetooth несуча частота сигналу змінюється 1600 разів на секунду псевдовипадковим чином, це дозволяє

Основною метою проекту Аксіома ГК "Розумна парковка" є впровадження програмно-апаратної системи, що дозволяє в режимі реального часу відстежувати стан кожного паркувального місця. Рішення базується на технології компанії Nedap AVI, яка, у свою чергу, розробила рішення SENSIT. Суть рішення полягає в тому, щоб у кожне паркувальне місце ввести датчик, який за допомогою електромагнітних та інфрачервоних датчиків визначає наявність або відсутність транспортного засобу. Далі по радіоканалу датчики передають інформацію на базові станції, які накопичують інформацію про групу місць і передають її на сервер управління. Програмне забезпечення, розроблене Axioma Group, дозволяє систематизувати інформацію, отриману від датчиків, будувати аналітичні звіти про зайнятість, оборотність паркувальних місць, передавати інформацію в навігаційні системи, а також на LED-дисплеї, розташовані в місті. Все обладнання, яке використовується на відкритому повітрі, має сертифікат принаймні IP65 і повністю адаптоване до кліматичних умов.

Parkeon представлений у понад 4000 містах і столичних районах у 60 країнах світу, пропонуючи інноваційні інтелектуальні транспортні засоби та рішення для паркування [11].

Запис:

– автомобіль зупиняється перед входом, прямо перед шлагбаумом;

– Датчик розпізнає наявність автомобілів, а відповідна камера автоматично розпізнає та записує номер автомобіля, дату та час в'їзду;

– Система дистанційного керування отримує дані, надані цією камерою, і додає їх до бази даних. Якщо клієнт уже зареєстрований, також додається інформація про членство та дата резервування;

– Шлагбаум на в'їзді піднімається, дозволяючи водієві заїхати на стоянку. Шлагбаум автоматично опускається, коли автомобіль проїжджає через датчик безпеки.

Вихід:

– автомобіль зупиняється на виїзді, прямо перед шлагбаумом;

– датчик розпізнає наявність автомобіля, а камера перевіряє стан. номер для відповідності інформації про оплату або набір автомобілів в базі даних;

– система контролю підтверджує правильний платіж або дозвіл на паркування без оплати, щоб дозволити виїзд;

– піднімаємо шлагбаум на виїзді, дозволяючи водієві залишити стоянку;

– шлагбаум автоматично опускається, як тільки автомобіль проїжджає через датчик безпеки.

Принцип роботи Mikkom AS101 ProPark полягає в точному визначенні розташування вільних і зайнятих місць і / або підрахунку кількості автомобілів, які в'їхали і виїхали. Світлова індикація кожного паркувального місця та інформаційні табло вказують водієві вільні місця та оптимальний маршрут до них згідно з рисунком 2. Система забезпечує оперативний та постійний контроль завантаженості з наданням всієї інформації персоналу, контроль за світлофори, дисплеї та шлагбауми [12]. Система створена на базі охоронного комплексу AS101 Pro і має притаманні комплексу високу надійність, зручність і простоту експлуатації.

В рамках цього проекту створено низку нових пристроїв (ультразвукові датчики присутності транспортних засобів згідно з рисунком 3, дистанційні

індикатори зайнятості місць, інформаційні табло, датчики входу/виїзду, суматори транспортних засобів, що проїжджають) та програмного забезпечення.

1.3 Висновки

У даному розділі було проведено аналіз предметної галузі, охарактеризовано, що таке пошук місць для паркування, визначено, що для виконання даного завдання необхідно працювати із географічними даними. Здійснено огляд відомих аналогів і більшість з них працюють чудово та мають можливість оплати паркування. Проаналізовано, актуальність даної системи у порівняні з існуючими аналогами та опубліковано тези до даної магістерської роботи.

2 ВИБІР ОПТИМАЛЬНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

2.1 Вибір оптимальних інформаційних технологій

Мова програмування – це штучна мова, створена для передачі команд машинам, зокрема комп'ютерам. Мови програмування використовуються для створення програм, які контролюють поведінку машин, та для запису алгоритмів.

Суворіше визначення: мова програмування – це система позначень для опису алгоритмів і структур даних, певна штучна формальна система, засобами якої можна виражати алгоритми. Мову програмування визначає набір лексичних, синтаксичних і семантичних правил, що задають зовнішній вигляд програми та дії, які виконує виконавець (комп'ютер) під її управлінням.

З часу створення перших програмованих машин було створено понад дві з половиною тисячі мов програмування. Щороку до них додаються нові. Деякими мовами вміє користуватись тільки невелике число їхніх розробників, інші стають відомі мільйонам людей. Професійні програмісти зазвичай застосовують у своїй роботі декілька мов програмування.

Мови програмування високого рівня оперують сутностями ближчими людині, такими як об'єкти, змінні, функції. Мови програмування нижчого рівня оперують сутностями ближчими машині: байти, адреси, інструкції. Текст програми мовою високого рівня зазвичай набагато коротший ніж текст такої самої програми мовою низького рівня, проте програма має більший розмір [11].

Універсальні та спеціалізовані. Спеціалізовані мови теж бувають Тюрінг-повні, та все ж їх ділянка застосування обмежена, як, наприклад, у мови shell.

Підтримувані парадигми програмування об'єктно-орієнтовані, логічні, функційні, структурні та інші.

Імперативні мови ґрунтуються на ідеї змінної, значення якої змінюється присвоєнням. Вони називаються імперативними оскільки складаються із послідовностей команд, які звичайно містять присвоєння змінних <code> <назва_змінної> = <[[вираз]]> </code>, де вираз може посилатися на значення змінних присвоєних попередніми командами (рис. 2.1).

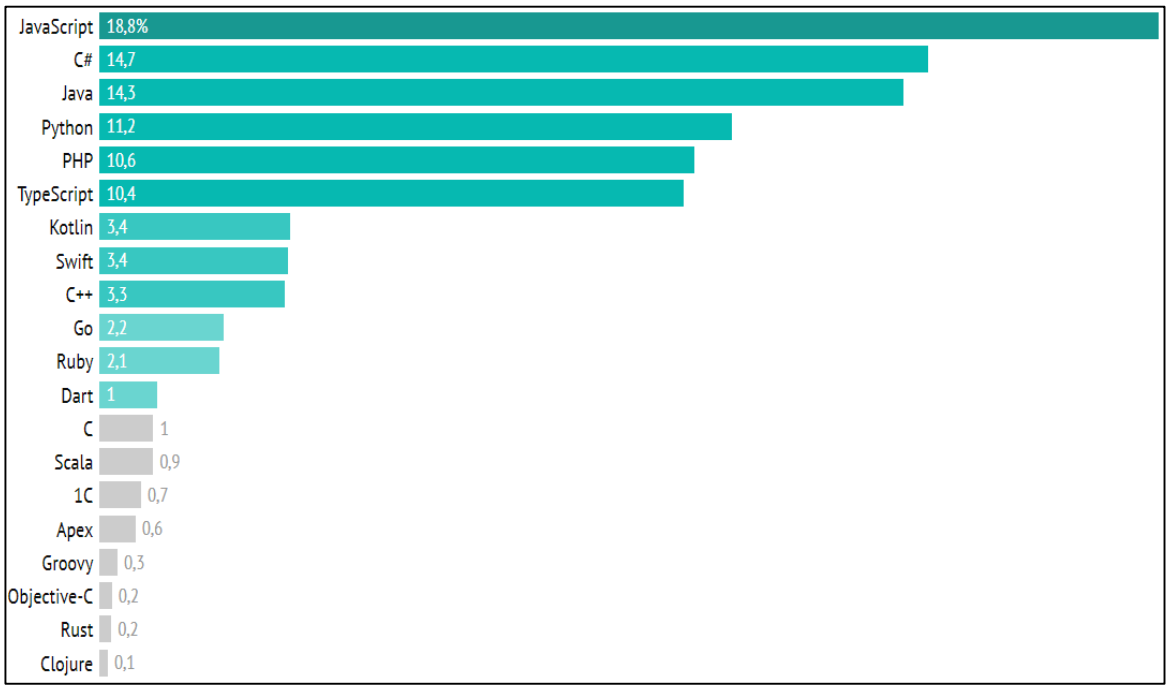


Рисунок 2.1 – Рейтинг мов програмування.

Об'єктно-орієнтоване програмування(ООП, іноді об'єктно-орієнтоване програмування, раніше об'єктно-орієнтоване програмування Object-oriented programming, OOP) – одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція Однією з переваг ООП є краща модульність програмного забезпечення (тисячу функцій процедурної мови, в ООП можна замінити кількома десятками класів із своїми методами) [12]. Попри те, що ця парадигма з'явилась в 1960-х роках, вона не мала широкого застосування до 1990-х, коли розвиток комп'ютерів та комп'ютерних мереж дав змогу писати надзвичайно об'ємне і складне програмне забезпечення, що змусило

переглянути підходи до написання програм [11]. Сьогодні багато мов програмування або підтримують ООП (PHP, Lua) або ж є цілком об’єкт орієнтованими (зокрема, Java, C#, C++, Python, Ruby і Objective-C, ActionScript 3, Swift, Vala).

Об'єктно-орієнтоване програмування сягає своїм корінням до створення мови програмування Стимулом в 1960-х роках, одночасно з посиленням дискусій про кризу програмного забезпечення. Через ускладнення апаратного та програмного забезпечення було дуже важко зберегти якість програм. Об'єктно-орієнтоване програмування частково розв'язує цю проблему шляхом наголошення на модульності програми.

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм, або як перелік інструкцій комп'ютеру, ООП-програми можна вважати сукупністю об'єктів. Відповідно до парадигми об'єктно-орієнтованого програмування, кожен об'єкт здатний отримувати повідомлення, обробляти дані, та надсилати повідомлення іншим об'єктам. Кожен об'єкт – своєрідний незалежний автомат з окремим призначенням та відповідальністю [12].

Для клієнтської частини розглядалося декілька фреймворків React Native та AngularJS та обраний AngularJS побудоване на основі JavaScript. Головним фактором у виборі технологій стала швидкість розробки та мультиплатформенність.

JavaScript – високорівнева мова програмування, яка підтримує імперативний, функціональний, подієво-орієнтований підходи. Вона має динамічну типізацію та застосовується для запису послідовних операцій «сценаріїв» чи «скриптів». Такі послідовності зазвичай інтерпретуються, а не компілюються, а тому не потребують додаткових програм чи інструментів перетворення в інший рівень кодування.

Кожен веб-застосунок чи сайт побудований з використанням трьох технологій – HTML, CSS та JavaScript. Остання виступає «мозком» розробки й відповідає за інтерактивність й взаємодію з користувачем.

Варто зазначити, що все частіше компанії не обмежуються роботою із JavaScript лише в браузері, а й користуються платформою Node.js, середовищем виконання JavaScript-коду, для написання серверних застосунків. Яскравими прикладами проектів за такої комбінації є Netflix та PayPal.

Angular.js уже знайшов значну кількість своїх прихильників. Це фреймворк, головне призначення якого – створення одно сторінкових додатків. Завдяки патерну MVC (Model-view-controller) додатки на Angular є більш структурованими, полегшує процеси тестування та розробки (рис. 2.2).

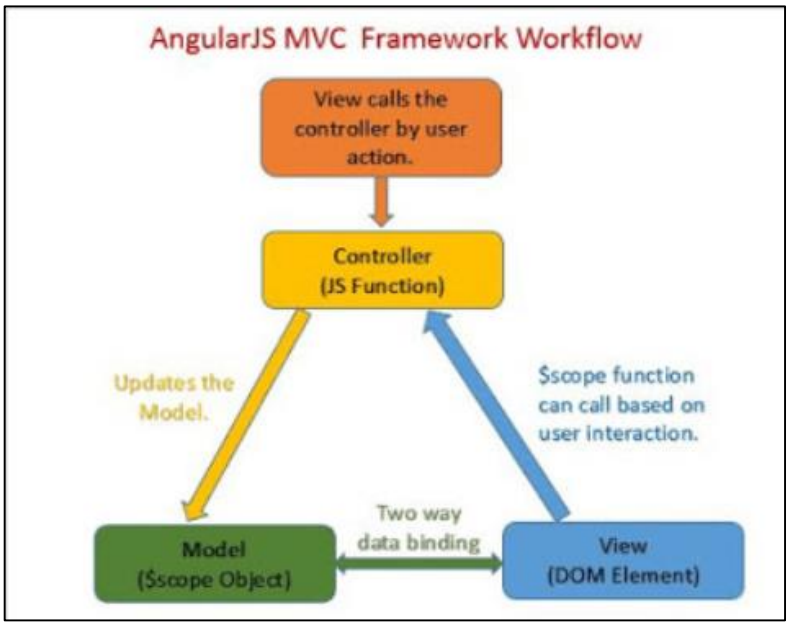


Рисунок 2.2 – Модель MVC для AngularJS

Освоєння азів фреймворку не вимагає великої трати часу, тому, при бажанні, всього за декілька годин, можна навчитися писати невеличкі додатки. Натомість, вивчення фічей може зайняти декілька місяців [13].

Сильними сторонами Angular.js є:

– підтримка Google та Microsoft;

– двостороннє зв’язування даних;

– широке коло ком’юніті, в число якого входять як розробники, так і ті, хто хочуть бути корисними в розвитку та покращенні фреймворку;

– чудова взаємодія з іншими бібліотеками;

– відображення шаблонів як HTML-тегів, що позбавляє сервер від додаткової роботи;

– декларативне програмування полегшує підтримку та читання коду;

– можливість розробки web-додатків, які будуть завантажуватись, як web-сторінки, але, водночас, забезпечувати додаткові функціональні можливості.

Однією вагомою особливістю фреймворку є використання двостороннього зв'язування даних. Це свідчить про те, що зміни, які відбуваються, відразу ж можна переглянути на об'єктах додатку (рис. 2.3).

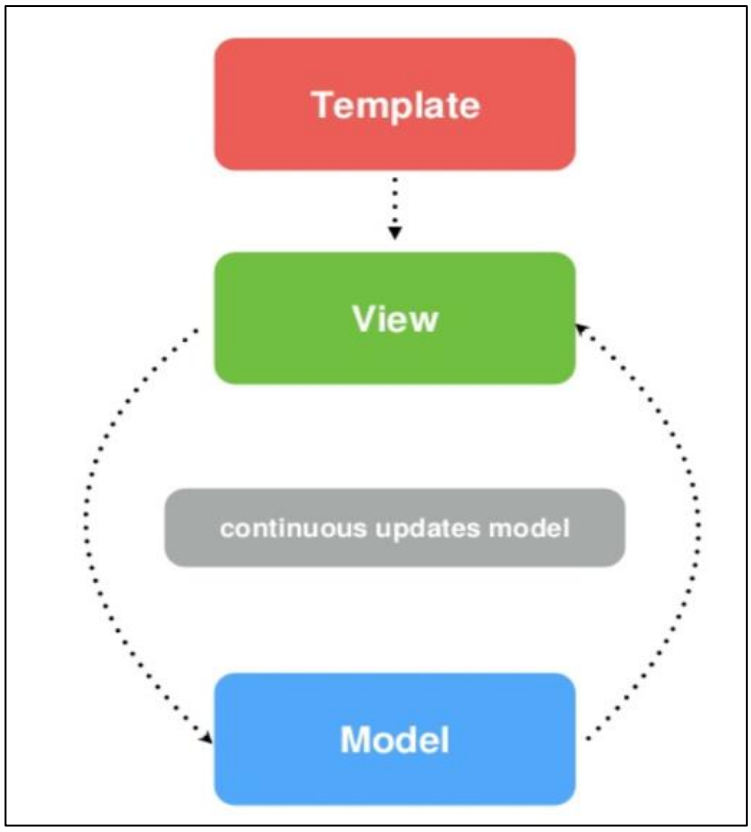


Рисунок 2.3 – Структура використання AngularJs

Two-way binding формується з допомогою компілятора та, безпосередньо, завдяки $compile. Тоді Angular.js генерує DOM-елементи, що використовуються в подальшому. $compile перетворює HTML-код у функцію, яка викликається для генерації контенту [13].

Архітектура Angular:

– Module – структури, у яких директиви, сервіси, компоненти об’єднані якоюсь логікою;

– Component – клас і з шаблоном, який відповідає за застосування логіки на сторінці. Частина цього коду може використовуватись у всьому додатку;

– Template – частина HTML-коду із синтаксисом;

– Service – клас typescript, який призначений для зберігання, отримання і обробки даних;

– Router – відповідає за відображення контенту при переході між екранами;

– Pipe – форматує дані в HTML-шаблоні. Pipe, які найчастіше використовуються: date (форматування дати), number (форматування числа), uppercase (переведення рядка у верхній регістр), slice (обрізання рядка), decimal (“диктує” формат числа);

– Directives – зміна зовнішнього вигляду або поведінки DOM-елемента (Document Object Model) [14].

Охарактеризуємо основні директиви Angular.js:

– ng-app – забезпечує зміну поведінки, використовуючи спеціальні теги;

– ng-bind – виконує прив’язку до властивості innerText HTML-елемента;

– ng-init – відбувається ініціалізація змінних додатку;

– ng-model – вказує на необхідність двостороннього зв’язування даних;

– ng-class – виконує визначення наборів класу, що застосовуватимуться до елементу;

– ng-controller – забезпечує приєднання контролерів до HTML DOM;

– ng-repeat – для кожного елементу колекції створюється декілька екземплярів;

– ng-form - директива створює об’єкт FormGroup і відповідає за його подальше прив’язування його до форми;

– ng-if - видалення/додавання елемента;

– ng-for - перебирає в певному шаблоні елементи масиву.

В якості хмарного провайдера було обрано Heroku, головна перевага цього провайдера полягає в тому, що він підтримує найпопулярніші мови програмування, а також надає безкоштовний обліковий запис і базу даних. Для датчика було обрано інфрачервоний датчик Sharp, який підключається до апаратної обчислювальної платформи Arduino, а для з’єднання датчика та сервера використовувався міні-модуль Ethernet. В якості бази даних було обрано PostgreSQL [14].

Під час вибору мов програмування я зупинився на Java для backend частини веб-сервісу та для отримання даних з OSM тому що. Java мають доступ до великої колекції бібліотек з відкритим кодом

Найкращі програмісти світу розробили шаблони, які роблять процедуру розробки більш простою. Підтримка з боку таких гігантів, як Google і Apache, дає зрозуміти, що ця мова програмування ще довго буде використовуватися в проектах. Java має гарно пророблений API, великий вибір інструментарію, велику кількість фреймворків.

Java може використовуватися для створення будь-якого програмного забезпечення: корпоративного програми, новинного сайту, настільної гри, пошукової системи. З джавою працюють популярні веб-ресурси, такі як Amazon.com, eBay.com, linkedin.com, aliexpress.com, і т. д.

Також за допомогою цієї мови програмування розробка виконується швидше, реалізація його обходиться набагато дешевше і в результаті виходить надійне додаток, що працює з усіма операційними системами, на будь-якому гаджеті [13].

Spring Boot – це фреймворк на основі Java з відкритим кодом, який використовується для створення мікросервісу. Він розроблений Pivotal Team і використовується для створення автономних і готових до виробництва Spring

додатків. У цьому розділі ви познайомитесь із Spring Boot і познайомитесь із його основними поняттями.

Micro Service – це архітектура, яка дозволяє розробникам самостійно розробляти та розгортати сервіси. Кожна запущена служба має власний процес, і це забезпечує полегшену модель для підтримки бізнес-додатків.

Переваги:

– легке розгортання;
– проста масштабованість;
– сумісний з контейнерами;
– мінімальна комплектація;
– менший час виготовлення.

Spring Boot надає хорошу платформу для Java-розробників для розробки автономної версії Spring програми продуктивного рівня, яку можна просто запустити . Ви можете почати роботу з мінімальними конфігураціями без необхідності повного налаштування конфігурації Spring [14].

Переваги:

– прості для розуміння та розробки програми Spring;
– підвищує продуктивність;
– скорочує час розробки.

Цілі:

– уникнути складної конфігурації XML у Spring;
– спростити розробку готових додатків Spring;
– скоротити час розробки та запустити додаток самостійно;
– запропонувати простіший спосіб почати роботу з програмою.

Функції та переваги, які Spring Boot пропонує:

– забезпечує гнучкий спосіб налаштування Java Beans, конфігурацій XML і транзакцій бази даних;

– забезпечує потужну пакетну обробку та керує кінцевими точками REST;

– автоматично налаштовано, не потрібні ручні налаштування;
– він пропонує додаток Spring на основі анотацій;
– ролегшує керування залежностями;
– він включає вбудований контейнер сервлетів [13].

Spring Boot автоматично налаштовує вашу програму на основі залежностей, які ви додали до проекту за допомогою анотації @EnableAutoConfiguration. Наприклад, якщо база даних MySQL знаходиться у вашому шляху до класів, але ви не налаштували жодного з'єднання з базою даних, тоді Spring Boot автоматично налаштує базу даних у пам'яті.

Точкою входу програми для завантаження Spring є клас, що містить анотацію @SpringBootApplication і основний метод.

Spring Boot автоматично сканує всі компоненти, включені в проект, використовуючи анотацію @ComponentScan.

Керування залежностями є складним завданням для великих проектів. Spring Boot вирішує цю проблему, надаючи набір залежностей для зручності розробників.

Наприклад, якщо ви хочете використовувати Spring і JPA для доступу до бази даних, достатньо включити у свій проект залежність spring-boot-starter-data-jpa.

Зауважте, що всі запускачі Spring Boot дотримуються однакового шаблону іменування spring-boot-starter-*, де * вказує на те, що це тип програми.

Приклади для кращого розуміння перегляньте наведені нижче початкові елементи Spring Boot [14].

Залежність Spring Boot Starter Security використовується для Spring Security (рис. 2.4).

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Рисунок 2.4 – Підключення залежності для Spring Security

Веб-залежність Spring Boot Starter використовується для написання REST-API (рис. 2.5).

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

Рисунок 2.5 – Підключення залежності для написання REST-API

Залежність Spring Boot Starter Thyme Leaf використовується для створення веб-програми (рис. 2.6).

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

Рисунок 2.6 – Підключення залежності для створення веб-програми

Залежність Spring Boot Starter Test використовується для написання тестів(рис. 2.7).

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-test</artifactId>
</dependency>
```

Рисунок 2.7 – Підключення залежності для написання тестів

Коротко про Spring Boot Spring широко застосовується для створення масштабованих застосувань. Для веб-застосунків Spring пропонує Spring MVC, який є широко застосовуваним модулем Spring фреймворку, який використовується для створення масштабованих веб-застосунків. Але основним недоліком проектів на основі цього фреймворку є їх конфігурація, яка може займати багато часу та бути занадто складною для початківців. Вирішенням цієї проблеми став Spring Boot [15].

Spring Boot (рис 2.8) – це open-source фреймворк, який розробляється і підтримується компанією Pivotal. Spring Boot побудований на верхівці Spring фреймворку і містить в собі всі його можливості. З його допомогою розробники можуть швидко розпочати роботу не витрачаючи багато часу на налаштування конфігурацій для запуску свого веб-застосунку

Рисунок 2.8 – Структура Spring Boot

Отже, Spring Boot вирішує наступні проблеми:

– робить легшим процес початку роботи зі Spring framework;

– мінімізує кількість створення конфігурацій застосунку вручну;

– здійснює автоконфігурацію на основі файлів property і JAR classpath;

– допомагає вирішити конфлікти з залежностями для Maven чи Gradle;

– пропонує вбудований Tomcat сервер;

– пропонує легке створення та підтримку для REST end points;

– процес деплою застосунку дуже простий. War або jar файл може бути легко задеплоєний на tomcat сервер;

– мікро сервісна архітектура;

Точкою входу в програму є клас відмічений анотацією @SpringBootApplication.

Якщо ви додали анотацію @SpringBootApplication до класу, вам не потрібно додавати анотації @EnableAutoConfiguration, @ComponentScan і @SpringBootConfiguration. Анотація @SpringBootApplication включає всі інші анотації.

Зазвичай Spring Boot Application реалізує архітектуру MVC. Шаблон проектування MVC передбачає поділ даних програми, інтерфейсу користувача та керуючої логіки на три окремих компоненти: Модель, Подання та Контролер – таким чином, що модифікація кожного компонента може здійснюватися незалежно [15].

Архітектура Spring Boot застосунку Spring Boot має 4 основних рівні:

– Presentation Layer – обробляє http запити, переводить JSON в Java об’єкти і автентифікує запит, передаючи його до бізнес рівню;

– Business Layer – обробляє всю бізнес логіку застосунку. Він складається з класів сервісів та використовує методи надані рівнем доступу до даних. Також цей рівень виконує валідацію та авторизацію;

– Persistence Layer – містить всю логіку роботи з БД, а також конвертує Java об’єкти з рядків БД та назад в ці рядки БД;

– Database Layer – На цьому рівні виконуються всі CRUD операції(рис. 2.9-2.10).

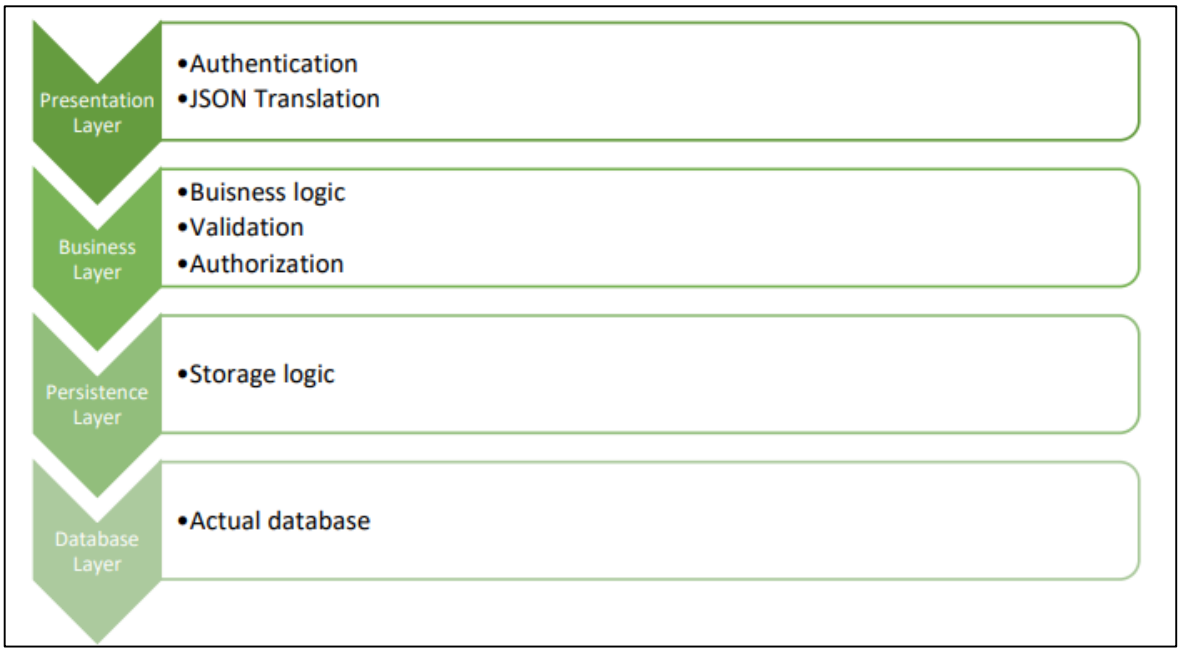


Рисунок 2.9 – Рівні архітектури Spring Boot

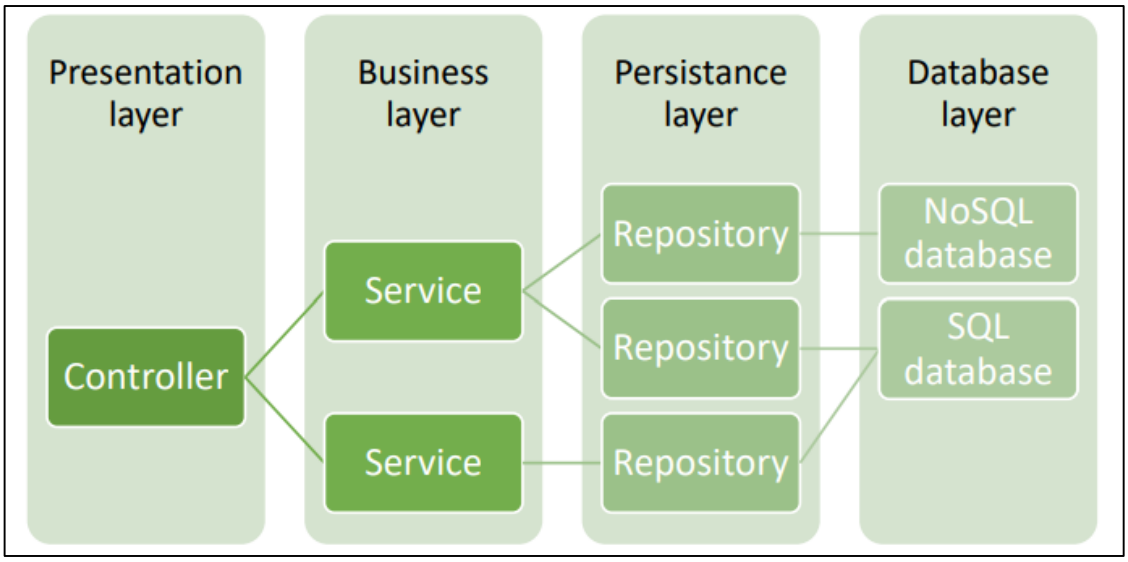


Рисунок 2.10 – Рівні архітектури Spring Boot застосунку

Spring boot flow (рис. 2.11):

– клієнт виконує http запит;

– запит йде до контролера, контролер знаходить метод, який може обробити даний запит і обробляє його. Після цього він викликає відповідний сервіс, якщо це необхідно. Presentation Layer •Authentication •JSON Translation Business Layer •Buisness logic •Validation •Authorization Persistence Layer •Storage logic Database Layer •Actual database Database layer Persistance layer

Business layer Presentation layer Controller Service Repository NoSQL database Repository SQL database Service Repository;

– на рівні сервісів виконується вся необхідна бізнес логіка. Цей рівень виконує операції над даними, які відображені в JPA за допомогою класів моделей;

– повертається відповідь клієнту у вигляді View або JSON даних.

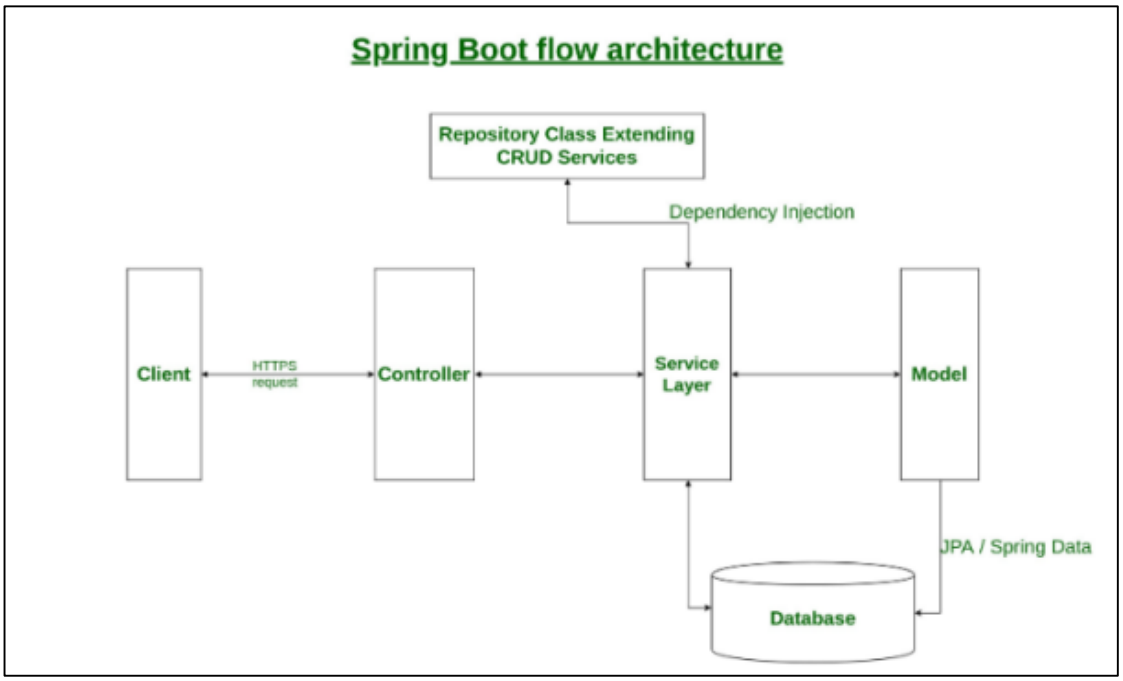


Рисунок 2.11 – Spring boot flow

Дані зберігаються і передаються між рівнями та клієнту у вигляді DTO та моделей. Модель представляє формальні базові конструкції даних. DTO (data transfer object) створений для того, щоб транспортувати дані від сервера до клієнта і навпаки[16].

Spring boot модель містить дані програми. Зазвичай включає в себе POJO-класи (Plain Old Java Objects) - прості старі Java-об'єкти або по-іншому - біни.

Подання або вигляд. Використовується для візуалізації та відображення даних програми за допомогою інтерфейсу користувача. Відповідає за те, як виглядатимуть дані моделі у браузері користувача.

Контролер потрібен для обробки запитів користувача і виклику серверних служб. Він структурує запит, створює відповідну модель для її подальшого відображення в браузері.

Як ми вже з вами розібралися, Spring MVC Framework логічно побудований на основі фронт-контролера DispatcherServlet, що обробляє всі HTTP-запити та відповіді. Щоб краще зрозуміти як він працює, подивимося, як усе влаштовано зсередини(рис. 2.12).

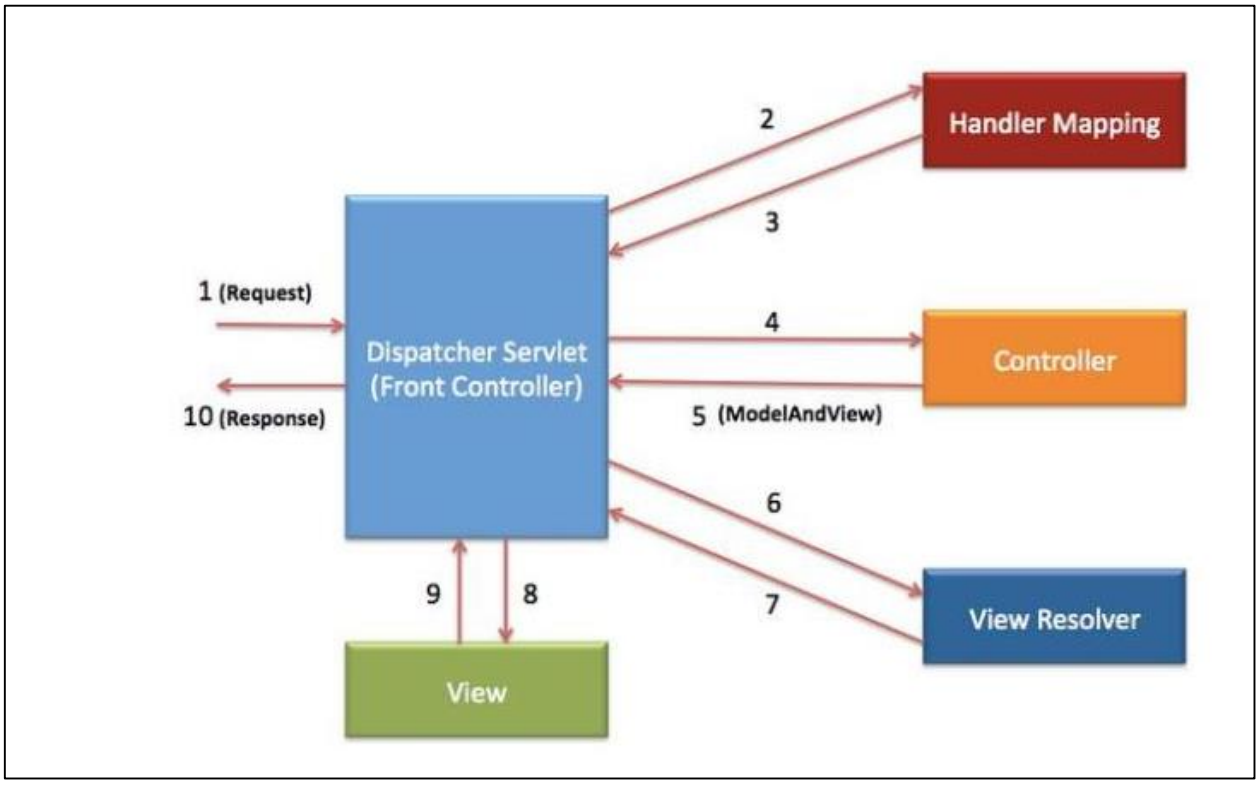


Рисунок 2.12 – Концепція роботи MVC

PostgreSQL – це об’єктно-реляційна система керування базами даних (СУБД). PostgreSQL підтримує такі типи індексів: B-дерево, хеш, R-дерево, GiST, GIN. При необхідності можна зробити найновіші типи індексів. У цій роботі, крім звичайних індексів, для швидкого пошуку найближчих точок використовувався індекс GiST. GiST (англ. Generalized Search Tree, Generalized search tree) – індексна структура, яка є узагальненою версією R-дерева і надає стандартні методи навігації по дереву та його оновлення (розбиття та видалення вузлів) [16]. GiST – це збалансоване (за висотою) дерево, листові

вузли якого містять пари (key, rid), де key – це ключ, а rid – вказівник на відповідний запис на сторінці даних. Внутрішні вузли містять пари (p, ptr), де p – деякий предикат (використовується як ключ пошуку), який буде виконано на всіх успадкованих вузлах, а ptr – вказівник на інший вузол у дереві. Це дерево визначає основні методи SEARCH, INSERT, DELETE та інтерфейс для написання спеціальних методів, які можуть контролювати роботу цих (базових) методів.

Метод SEARCH контролюється функцією Consistent, яка повертає «true», якщо вузол задовольняє предикат, метод INSERT контролюється функціями penalty, picksplit і union, які дозволяють нам оцінити складність операції вставки у вузол, розділіть вузол при переповненні та перебудуйте дерево, якщо необхідно, метод DELETE знаходить літерне дерево, що містить ключ, видаляє пару (key, rid) і, якщо необхідно, перебудовує батьківські вузли за допомогою функції об’єднання [16].

GiST – це прямий індекс, який використовується для повнотекстового пошуку PostgreSQL. Це означає, що для кожного вектора, який описує всі маркери в документі, створюється описовий підпис, який включає маркери в цьому векторі. Принцип роботи схожий з розрядними індексами, але є відмінності. Продемонструємо на прикладі: нехай токен w1 асоціюється з сигнатурою 001000,

маркер w2 дорівнює 000010. Тоді документ, що містить лише маркери w1 і w2, приєднається до підпису 001010 (001000 | 000010). На відміну від бітових індексів, відображення токенів у сигнатурі є неоднозначним, тобто маркер w3 із сигнатурою 001000 може існувати. Це дозволяє значно заощадити на розмірі індексу, але вимагає вторинної перевірки на повну відповідність між маркерами запиту та документа. Слід також зазначити, що якщо маркер запиту має підпис, наприклад, 000001, то документ із підписом 001010 не містить його однозначно, незважаючи на неоднозначність відображення маркера.

Перевагою GiST є швидкість створення та розмір індексу порівняно з GiN (у 3 рази), тому він кращий для динамічно постійно оновлюваних даних. Для статичних або рідко оновлюваних даних краще підійде індекс GiN (він шукає в 3 рази швидше).

Mapbox – це онлайн карти для веб-сервісів та користувачів яка може підключатись як до них через власні бібліотеки Mapbox GL JS для веб-сервісів та Maps SDKs для Android та IOS. Mapbox надає потужні механізми маршрутизації, точний час подорожі з урахуванням заторів та інтуїтивно зрозумілі покрокові вказівки, які допоможуть вам створити привабливу навігацію.

Mapbox.js є асинхронним – коли ви створюєте такий шар, як, шар не відразу знає, які плитки завантажити та інформацію про його присвоєння. Натомість він завантажує інформацію у виклик API.

Для більшості речей, які ви напишете, це не проблема, оскільки Mapbox.js добре справляється з цими оновленнями на льоту. Якщо ви пишете код, який повинен знати, коли шари та інші динамічно завантажуванні об’єкти готові, ви можете використовувати подію, щоб прослухати їх стан готовності [17].

2.2 Огляд можливостей ресурсу OpenStreetMap

OpenStreetMap – це відкритий проект, спрямований на збирання, збереження та розповсюдження загальнодоступних геопросторових даних, створення інструментів для роботи з ними силами спільноти волонтерів.

Готові для використання карти доступні онлайн на самому сайті OpenStreetMap і через безліч інших сайтів, що мають додаткові функції. За допомогою функції «Експорт» можна отримати HTML-код для вставки на будь-який сайт з можливістю нанесення маркера (але є і потужніші інструменти, такі як Easymap Архівна копія від 13 лютого 2011 року на Wayback Machine, за допомогою яких, наприклад, можна будувати

маршрути). Карти також доступні для скачування у вихідному форматі Архівна копія від 10 березня 2020 року на Wayback Machine та у форматах для різних автомобільних та інших навігаторів.Доступні також спеціальні програмидля мобільних пристроїв (Java, Apple iOS, Windows Mobile, Android та ін.) Архівна копія від 8 серпня 2020 року на Wayback Machine і для комп'ютерів Архівна копія від 22 квітня 2020 року на Wayback Machine [15].

Фонд вільного програмного забезпечення закликає всіх робити свій внесок у проект OpenStreetMap з метою створення вільної альтернативи пропріє тарній Google Earth, завдання заміни якої знаходиться у списку високо пріоритетних проектів Фонду. OpenStreetMap описується як відповідь Google та комерційно обмеженим геоданим від світу вільного програмного забезпечення. Карти OpenStreetMap використовують такі сайти та організації як Організація Об'єднаних Націй, Вікіпедія, Microsoft Bing Maps, MapQuest, Вікімапія, Оксфордський університет, сайт президента США, французька газета «Libération», американські рятувальники, а також Космознімки, Monopoly City Streets та інші. Офіційно на можливість використання карток у своїх продуктах вказують навігаційні компанії Garmin, Автосупутник та PocketGIS. За даними Alexa.com, більшість відвідувачів OpenStreetMap приходять з Німеччини та США, а найвищі місця в рейтингах відвідуваних сайтів OpenStreetMap займає в Австрії та Німеччині.

OpenStreetMap(OSM), по суті, не є мапою у звичному розумінні, це база геопросторових даних. Вона містить географічні координати окремих точок та інформацію про об'єкти вищого порядку – лінії, що з'єднують точки, зв'язки, які можуть включати точки й лінії, а також атрибути всіх зазначених об'єктів. Тому на основі одних і тих самих даних OSM можуть бути побудовані різноманітні сервіси, що відрізняються як способом відображення, так і функціональністю [6]. Мапу, пошук об'єктів та сервіс прокладання маршрутів на головній сторінці OpenStreetMap слід розглядати лише як один із прикладів використання даних бази OSM (рис. 2.13).

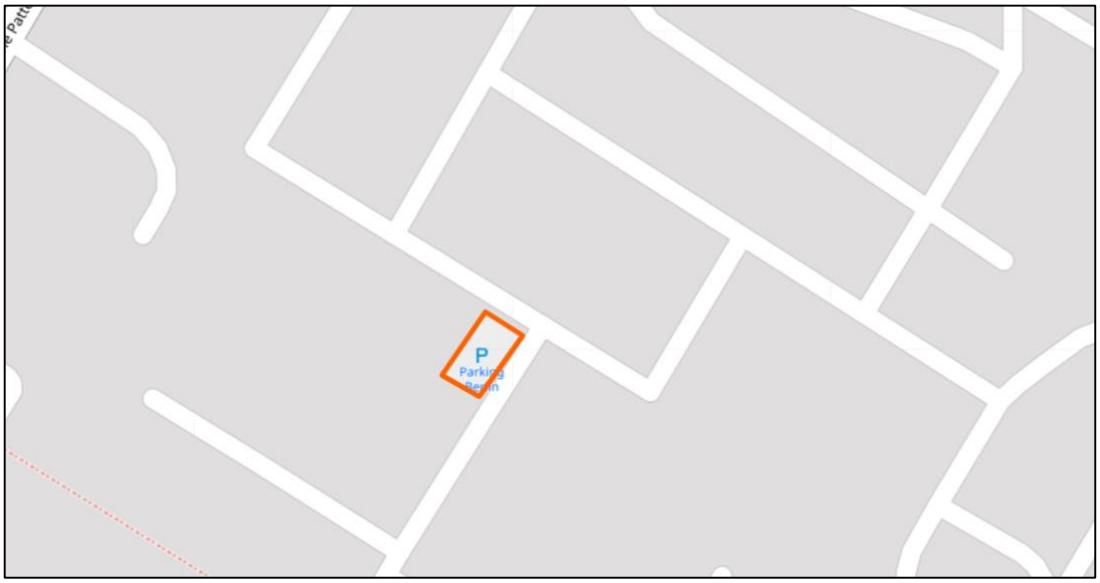


Рисунок 2.13 – Зображення паркувальної зони на карті OSM.

Мапи двовимірні, без зазначення висот над рівнем моря, ізоліній. Хоча також набуває поширення позначення висотних характеристик окремих об'єктів та розвиваються проекти з їх рендерингу.

Дані мап, як всієї Землі, так і окремих її ділянок, можуть бути як завантажені у форматі OSM, так і перетворені в інші формати для роботи з ними в геоінформаційних системах, для створення друкованих мап або перетворення в формати для GPS-навігаторів.

Дані OpenStreetMap можуть бути стилізовані у потрібний користувачу спосіб. На головному веб-сайті OpenStreetMap присутні чотири різні стилі (шари) мап, для показу яких використовується вільна JavaScript-бібліотека leaflet.js. Але ви можете не обмежуватись однією бібліотекою та обрати інші, як вільні так і патентовані аналоги [6].

2.3 Висновки

В даному розділі було проаналізовано передові технологій, які б задовільнять усі потреби, який був описаний у завданні. Дані технології чітко поєднуються між собою, створюючи потужний та швидкий веб-додаток, який

забезпечує користувача всіма необхідними ресурсами. Також у даному розділі виконано формування програмних вимог для веб-сервісу та було побудована концепція кінцевого продукту. Було проведено відбір оптимальних інформаційних технологій для серверної частини було обрано мову програмування Java та фреймворк для створення веб серверу із використанням патерну MVC що являє собою під собою умовний поділ додатку на три частини. Для користувацької частини додатку було використано мову програмування JavaScript та бібліотеку AngularJs, яка використовується для створення одно сторінкових додатків за допомогою патерну MVC. OpenStreetMap розглянуто для отримання даних по місцезнаходженню місця для паркування та їх зображення на карті.

3 ЗБИРАННЯ ТА АДАПТАЦІЯ ДАНИХ МІСЦЬ ПАРКУВАННЯ З OPENSTREETMAP

3.1 Збирання та адаптація даних

Для отримання даних будемо використовувати дані які нам пропонує OSM. Для їх отримання будемо використовувати Overpass API. Overpass API (раніше відомий як OSM Server Side Scripting або OSM3S до 2011 року) – це API лише для читання, який обслуговує спеціальні вибрані частини картографічних даних OSM. Він діє як база даних у мережі: клієнт надсилає запит до API та отримує назад набір даних, який відповідає запиту.

Процес розроблення одного застосунку для різних інформаційно-технологічних платформ вимагає написання коду під кожну платформу на рідній мові, що є досить трудомістким завданням. Використання кросплатформних засобів розроблення дозволяє значно зменшити час розробки під кожну платформу. Для підтримання платформ фреймворку використовується технологія «Apache Cordova». Розроблення прототипу фреймворку продемонстровано на прикладі платформи «Android». Все більше розробників вибирають створення гібридних застосунків. Це пов'язано з тим, що для їх розробки можна залучати наявних фахівців з «HTML/CSS/JavaScript», а також використовувати готові напрацювання, створені в ході розроблення веб-застосунків для муніципальних програмно-алгоритмічних комплексів та супутнього ПЗ [14].

Для отримання даних будемо використовувати дані які нам пропонує OSM. Для їх отримання будемо використовувати Overpass API. Overpass API (раніше відомий як OSM Server Side Scripting або OSM3S до 2011 року) – це API лише для читання, який обслуговує спеціальні вибрані частини картографічних даних OSM. Він діє як база даних у мережі: клієнт надсилає запит до API та отримує назад набір даних, який відповідає запиту.

На відміну від основного API, який оптимізовано для редагування, Overpass API оптимізовано для споживачів даних, яким потрібно кілька елементів за один раз або до приблизно 10 мільйонів елементів за кілька хвилин, обидва вибрані за критеріями пошуку, як-от розташування, тип об’єктів, властивості тегів, близькість або їх комбінації. Він діє як сервер бази даних для різних служб [17].

Крім того, існує довідник Overpass QL/мовна довідка. Настійно рекомендуємо ознайомитися з різними функціями за допомогою Overpass Turbo, інтерактивного веб-інтерфейсу. Для застарілих програм також існує рівень сумісності, який забезпечує плавний перехід від XAPI (рис. 3.1).

Рисунок 3.1 – Інтерфейс Overpass turbo

На даному малюнку зображено інтерфейс Overpass turbo на якому зображено приклад запиту для отримання парковок у заданому місці на карті та повернення даних у форматі json. Overpass API має можливість повертати дані в різних форматах, таких як json, xml, csv.

```
{
  "type": "way",
  "id": 834681322,
  "bounds": {
    "minlat": 52.4807606,
    "minlon": 13.3387922,
    "maxlat": 52.4810250,
    "maxlon": 13.3393207
  },
  "nodes": [
    3406961568,
    7791676788,
    7791650681,
    7791650682,
    7791676787,
    3406961568
  ],
  "geometry": [
    { "lat": 52.4810250, "lon": 13.3392765 },
    { "lat": 52.4809905, "lon": 13.3393207 },
    { "lat": 52.4807606, "lon": 13.3388365 },
    { "lat": 52.4807952, "lon": 13.3387922 },
    { "lat": 52.4808614, "lon": 13.3389317 },
    { "lat": 52.4810250, "lon": 13.3392765 }
  ],
  "tags": {
    "amenity": "parking_space",
    "capacity": "18"
  }
},
```


Рисунок 3.2 – Приклад даних у вигляді JSON.

На рисунку 3.2 зображено BoundingBox парковки, координати її меж та загальна кількість місць на парковці [10].

Оскільки кількість автомобілів швидко зросла протягом останніх кількох років (рис. 3.3), збільшується потреба у місцях для паркування та пошуку. Якщо припустити, що кожен день середній водій витрачає на пошуки такого місця 20 хвилин, то це приблизно 120 годин на рік, які можна витратити на щось більш корисне [4].

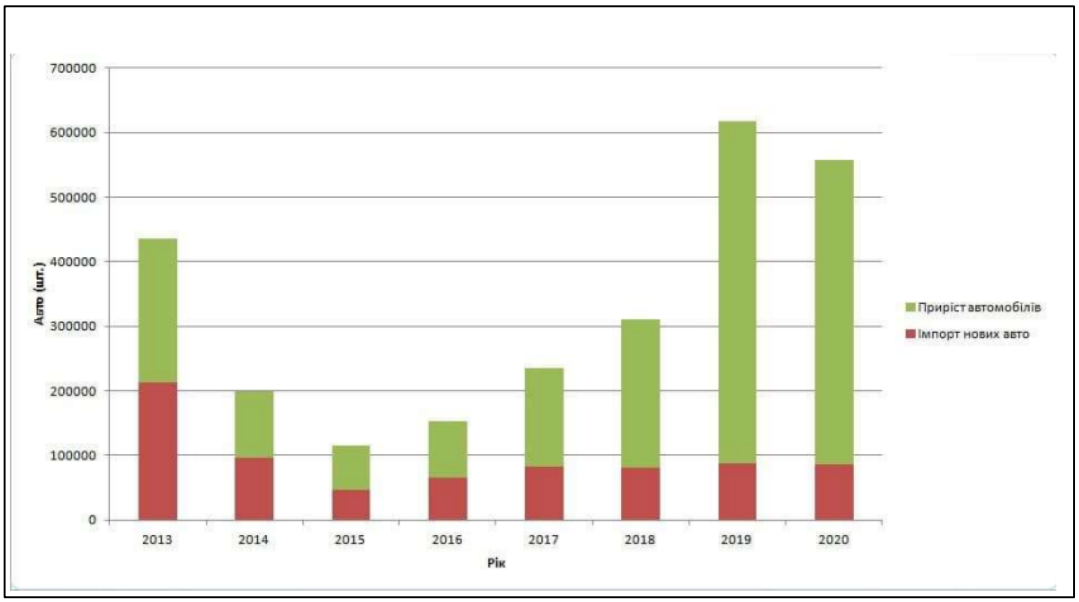


Рисунок 3.3 – Приріст кількості транспортних засобів в Україні

Як видно з рисунка 3.3, загальне збільшення кількості транспортних засобів включає імпорт нових автомобілів (червона частина колонки) та вживаних автомобілів, ввезених з-за кордону (зелена частина колонки). При цьому помітно, що з 2016 по 2020 рік імпорт нових автомобілів залишається приблизно на тому ж рівні, але частка імпорту вживаних автомобілів поступово збільшується. Це пов’язано зі змінами в законодавстві щодо розмитнення автомобілів, що ввозяться з-за кордону, а саме 25 листопада 2018 року прийнято закон про спрощення процедури розмитнення ввезених з-за кордону вживаних автомобілів [5].

Щоб розробити наш прототип розумного паркування, першим кроком є дослідження необхідних функцій і вимог, які є важливими для цього типу системи. Дослідження цих функцій і вимог дозволить зрозуміти процес розробки всієї системи. Для встановлення камери в загальнодоступній зоні потрібен дозвіл на камеру відеоспостереження. Важливим фактором, який слід враховувати, є проблема конфіденційності, яка слідує за рішеннями на основі бачення. Надсилання зображень на сервер або хмарний сервіс через Інтернет також супроводжується проблемами безпеки. Щоб вирішити цю проблему безпеки, можна використовувати певний процес шифрування. Усі ці

обмеження можуть негативно вплинути на зручність використання системи. Виконання завдань з обробки зображень у місці, де зображення фіксуються без зберігання чи надсилання будь-яких даних зображення через Інтернет, може допомогти уникнути цих проблем, згаданих вище. Отже, аспект конфіденційності та безпеки слід розглядати як вимогу інтелектуального рішення для паркування. Щоб побудувати динамічну розумну систему паркування, місця для паркування мають визначатися автоматично. Це завдання можна виконати за допомогою алгоритму виявлення об'єктів. Вибраний підхід для виявлення паркувальних місць – виявлення транспортних засобів за допомогою технології глибокого навчання. Вибраний підхід заснований на його простоті виявлення паркувальних місць на стоянці. Інший підхід, який можна використовувати для виявлення місць для паркування, - це визначення лінії. Однак у цього підходу є деякі недоліки, тобто лінії можуть з часом стиратися або розмиватися або просто покриватися снігом. Метою передачі даних в системі є передача даних на сервер, де можна зареєструвати інформацію про паркування. Передані дані мають на меті надати інформацію про стан паркінгу, тобто про те, скільки паркувальних місць доступно і скільки з цих місць для паркування незайнятих [18].

Перед тим, як сідати за кермо у Європі, уважно вивчіть правила дорожнього руху країн на своєму шляху. Основні правила і дорожні знаки збігаються з українськими. Але кожна країна має свої особливості, у тому числі різні заходи покарань і розмірів штрафів за правопорушення. Штрафи в Європі дуже високі: від кількадесят до кількасот євро. Причому, «відмазатися» і «домовитися» з дорожньою поліцією навряд чи вийде, вас можуть притягнути до відповідальності за хабар.

Крім інформації про правила і штрафи, запишіть телефони дорожньої поліції і страхової компанії (представника у кожній країні). Ці контактні дані стануть у нагоді при ДТП чи несправностях.

Правила дорожнього руху в Україні хоча й важко назвати ідеальними, все ж мало кого збентежать. Але є країни, які дивують несподіваними, неймовірними та навіть абсурдними законами.

Ми вирішили дослідити, які ПДР інших країн викликають в іноземців подив та чим пояснюються незвичайні заборони та дозволи.

Вирушаючи в Європу в поїздку, ви їздитимете дуже багато, тому бажано встановити газ на авто в Сервіс Газ Vip. Газобалонне обладнання дозволить долати далекі відстані без необхідності дозаправитися. Також це суттєво заощадить гроші на паливі, а заощаджені гроші можна буде витратити на приємні дрібниці.

Часто водії, які приїхали до країн Європи, ризикують потрапити під штраф, а через незнання правил паркування в країнах Європи.

Основні правила паркування у Європі запам'ятати не складно, оскільки розмітка використовується однакова.

Жовта зигзагоподібна лінія – паркуватися заборонено.

Біла - можна безкоштовно залишити автомобіль з 18:00 до 8:00, а в решту годин доби доведеться сплатити стоянку автомобіля.

На дорожніх знаках є вся необхідна інформація - в які дні та час паркуватись можна.

У Європі поширені підземні стоянки, а наявність вільних місць повідомляє табло над в'їздом.

У центрі великих міст є багато безкоштовних парковок з обмеженим часом перебування. Вони, в основному, розташовані неподалік головних визначних пам'яток, а також великих торгових центрів.

Знайти паркування будь-якого типу у більшості великих міст вам допоможуть спеціалізовані сайти на кшталт car-parking.eu.

Ну і, звичайно ж, не забувайте, що у разі неправильного паркування водія очікує штраф і досить високий. В основному сума у вашій квитанції коливатиметься в межах від 40 до 80 євро [19]. Однак, якщо ви залишили авто

на місці, призначеному для інвалідів, сума штрафу буде від 200 до 400 євро в залежності від країни, в якій ви знаходитесь.

Інспекція з паркування в Києві перейшла з попереджуючого режиму роботи до штрафування власників автомобілів за порушення правил парковки.

Про це написав начальник інспекції з парковки Дмитро Рахматуллін на своїй сторінці в Facebook.

За його словами, інспектори досі не мають сертифікованих технічних засобів фотофіксації. Повідомлення про штраф розміщують на лобовому склі автомобіля.

Також інспектори почали користуватися велосипедами сервісу прокату Nextbike з метою підвищення результативності.

Луцькі парковки – камінь спотикання як для місцевої влади, так і для водіїв та пішоходів. Загалом, не дивлячись на привернення уваги до цієї проблеми місцевими активістами, як то кажуть, віз і нині там.

У Луцьку 80 % порушень ПДР пов'язані саме із паркуванням у недозволених для цього місцях. Тому й виходить парадокс: заборонених для паркування місць вдосталь, а облаштованих під парковку майданчиків немає.

Пішоходи нарікають на недолугих водіїв, які не вміють паркуватися, водії ж, своєю чергою, бідкаються, що у місті не вистачає парковок, а влада нічого не робить для цього. Владці ж вкотре нагадують лучанам: бюджет не гумовий, тому усіх проблем (зокрема й з парковками) не вирішить [8].

На відміну від основного API, який оптимізовано для редагування, Overpass API оптимізовано для споживачів даних, яким потрібно кілька елементів за один раз або до приблизно 10 мільйонів елементів за кілька хвилин, обидва вибрані за критеріями пошуку, як-от розташування, тип об’єктів, властивості тегів, близькість або їх комбінації. Він діє як сервер бази даних для різних служб.

Також в подальшому можна буде використовувати інші ресурси для збору даних про наявність місць для паркування.

Блок-схема системи призначена для відображення загальної структури системи, а саме основних блоків, підсистем і основних зв'язків між ними.

Система розділена на дві підсистеми, а саме підсистема автомобіля та підсистема паркування.

У свою чергу підсистема автомобіля ділиться на чотири групи:

– блок датчиків та інформаційних пристроїв (ІЧ);
– блок виконавчих пристроїв автомобіля;
– блок бездротових модулів автомобіля;
– пристрої обробки інформації (бортовий комп'ютер та інші блоки).

До першої групи належать камери, лідер(и), ультразвукові датчики, радари, датчики тиску в гальмівній системі, датчик кута повороту керма. Вхідні дані передаються на подальшу обробку.

Пристрої обробки інформації, включаючи бортовий комп'ютер і блоки управління виконавчими пристроями. Результатом роботи бортового комп'ютера є обробка даних від датчиків, камер і відправка потоку даних в паркувальну підсистему через блок бездротових модулів, а також отримання команд управління від системи для обробки і передачі в систему управління. блок відповідних виконавчих механізмів [19].

Підсистема паркування поділяється на чотири групи:

– блок парктроніків та інформаційних підсистем (ППБ, ПВС);
– блок виконавчих пристроїв для паркування;
– блок бездротових паркувальних модулів;
– пристрої обробки інформації (блоки керування, сервери).

Дані з автомобіля потрапляють через комунікаційні модулі (бездротові модулі, Ethernet) на сервер даних. При цьому на сервер сервісу передаються дані з датчиків паркування: датчиків руху, датчиків освітлення при необхідності, але в основному дані про зайняті місця і дані про локалізацію від таких виконавчих пристроїв, як блок маячків (міток). Особливістю цієї структури є те, що самі мітки не є елементами інфраструктури. Тому вони не

беруть участь у формуванні мережі, маршрутизації тощо. Тобто вони звільнені від усіх завдань, крім вимірювання відстані до мобільних об’єктів (автомобілів) та передачі результатів на сервер через транспортну інфраструктуру. Це дозволяє істотно спростити теги і, відповідно, знизити енергоспоживання і вартість міток. Мітки взаємодіють з інфраструктурою через двонаправлений радіоінтерфейс. Інтерфейс визначає частотний діапазон, форму радіохвилі, методи модуляції та кодування, формати пакетів, а також команди та звіти, за якими обмінюються мітками та інфраструктурою.

Діаграму можна умовно розділити на складові елементи:

– авто (встановлюється певний додаток на бортовий комп’ютер автомобіля) через комунікаційні модулі створює запит на підключення автомобіля до системи (передача певного ідентифікатора для авторизації користувача);

– встановлення каналу зв'язку з системою, перевірка клієнта в базі даних (дозвіл на резервування медіаресурсу);

– запит на виділення медіаресурсу (повернути конкретний IP), на який в подальшому будуть надсилатися медіадані з автомобіля;

– збирання даних з датчиків і відеокамер автомобіля, генерація медіа-потоку і відправка його на спеціальний ресурс;

– одночасно збираються дані з датчиків паркування та передаються разом з даними автомобіля на сервер сервісу для розрахунку контрольних точок, маршруту та паркувального маневру;

– після чого всі обчислення використовуються сервером додатків для генерації керуючих команд;

– блок управління рухом автомобіля (БКР) передає команди на бортовий комп'ютер автомобіля;

– команди попередньо обробляються та надсилаються на відповідні блоки керування потрібними виконавчими механізмами [17].

Для кращого розуміння системи розглянемо – основні етапи функціонування системи. Оскільки для простоти впровадження цієї системи

створення бази цифрових відбитків пальців і динамічне уникнення перешкод виключено, а модель середовища надається у вигляді карти середовища системою транспортного засобу для інфраструктури V2X або камер паркування. Тобто система використовує статичну карту паркування та передбачає точну само локалізацію автомобіля.

Після підключення автомобіля до системи на карті визначається початкова точка (позиція) автомобіля для подальшого планування маршруту. Маршрут створюється за допомогою карти та статичного глобального плану маршруту, який по суті відображає пункти призначення через певні координати на карті. Таким чином, запит (перевірка) місця розташування триватиме, поки транспортний засіб не досягне кінцевого пункту призначення.

Після аналізу та перегляду функціональної діаграми, етапів роботи системи та діаграми послідовності, давайте перейдемо до діаграми варіантів використання, показаної в, щоб зрозуміти, як рухається автомобіль.

Після знаходження перехідних позицій (цілей) із запланованого шляху та маневру паркування на сервісному сервері дані зберігаються або оновлюються на сервері даних, після чого використовуються на сервері додатків для формування команд керування. Блок управління дорожнім рухом (БКР) передає їх в автомобіль через модулі зв'язку. Після цього здійснюється попередня обробка команд управління бортовим комп'ютером автомобіля і передача на відповідні блоки управління виконавчими механізмами. Наприклад, профіль швидкості, команди прискорення та уповільнення обробляються блоком електромотора (ЕМ) і блоком приводу гальм.

(BGP), щоб скинути швидкість при зміні напрямку або зупинці на кінцевій траєкторії.

Система управління базами даних - сукупність програмних і лінгвістичних засобів загального або спеціального призначення, що забезпечують управління створенням та використанням баз даних [16].

Основними функціями СУБД є:

– керування даними, що знаходяться в зовнішній пам'яті (на дисках);

– керування даними, що знаходяться в оперативній пам'яті з використанням дискового кеша;

– запис змін до журналу, резервне копіювання і відновлення бази даних після збоїв;

– підтримка мов БД (мова визначення даних, мова маніпулювання даними).

Зазвичай сучасна СУБД містить наступні компоненти:

– ядро, яке відповідає за управління даними у зовнішній і оперативній пам'яті, та журналізацію;

– процесор мови бази даних, що забезпечує оптимізацію запитів на вилучення та зміну даних і створення, як правило, машинно-незалежного виконуваного внутрішнього коду;

– підсистему підтримки часу виконання, яка інтерпретує програми маніпуляції даними, створюють користувальницький інтерфейс з СУБД;

– також сервісні програми (зовнішні утиліти), що забезпечують ряд додаткових можливостей по обслуговуванню інформаційної системи [19].

Для збереження даних будемо використовувати базу даних PostgreSQL.

PostgreSQL – широко розповсюджена система керування базами даних з відкритим сирцевим кодом. Прототип був розроблений в Каліфорнійському університеті Берклі в 1987 році під назвою POSTGRES, після чого активно розвивався і доповнювався. В червні 1990 року з'явилась друга версія із переробленою системою правил маніпулювання та роботи з таблицями, у 1991 році – третя версія, із доданою підтримкою одночасної роботи кількох менеджерів збереження, покращеним механізмом запитів і доповненою системою внутрішніх правил. В цей час POSTGRES використовувався для реалізації великих систем, таких як: система аналізу фінансових даних, пакет моніторингу функціональності потоків, база даних відстеження астероїдів, система медичної інформації, кілька географічних систем. POSTGRES також використовувався як навчальний інструмент в кількох університетах. 1992 року POSTGRES став головною СКБД наукового комп'ютерного проєкту

Sequoia 2000. 1993 року кількість користувачів подвоїлась. Стало зрозуміло, що для підтримки й подальшого розвитку необхідні великі витрати часу на дослідження баз даних, тому офіційно проект Берклі було зупинено на версії 4.2. 1994 року додали інтерпретатор мови SQL, вдосконалили сирцевий код і виклали в Інтернеті свою реалізацію під назвою Postgres95. 1996 року програмний продукт було перейменовано на PostgreSQL із початковою версією 6.0. Подальшою підтримкою й розробкою займається група спеціалістів у галузі баз даних, які добровільно приєднались до цього проєкту.

PostgreSQL є однією з найпопулярніших систем управління базами даних. Сам проект postgresql еволюціонував з іншого проекту, який називався Ingres. Формально розвиток postgresql почався ще 1986 року. Тоді він називався POSTGRES. А в 1996 році проект був перейменований на PostgreSQL, що відображало більший акцент на SQL. І 8 липня 1996 року відбувся перший реліз товару.

З того часу вийшло безліч версій postgresql. Поточною версією є версія 14. Однак, регулярно також виходять підверсії.

PostgreSQL підтримується всім основних операційних систем - Windows, Linux, MacOS [18].

Для збереження гео-даних використовується бібліотека PostGIS.

PostGIS - це просторовий розширювач бази даних для об’єктно реляційної бази даних PostgreSQL. Він додає підтримку географічних об'єктів, що дозволяє виконувати запити про розташування в SQL. Також PostGIS пропонує багато можливостей, які рідко можна знайти в інших конкуруючих просторових базах даних, таких як Oracle Locator/Spatial та SQL Server.

Liquibase – це система управління міграціями бази даних. Як і багато інструментів, які служать для полегшення життя розробників програмного забезпечення, Liquibase має «зворотний бік медалі», з яким доводиться рано чи пізно зіткнутися. Завжди можна легко сказати, які зміни схеми даних відбулися під час переходу з версії на версію програми. Підтримується акуратність у папках, що не дозволяє розслаблятися

Для роботи з базами даних потрібно jOOQ об’єктно-орієнтовані запити, широко відомий як jOOQ, - це легке відображення бази даних бібліотека програмного забезпечення в Java що реалізує шаблон активного запису. Його мета - бути обома реляційний і об'єктно-орієнтований шляхом надання а мова домену для побудови запитів з класи, генеровані від а схема бази даних.

jOOQ стверджує, що SQL має бути на першому місці при будь-якій інтеграції баз даних. Таким чином, він не вводить нового текстового мова запитів, але швидше дозволяє побудувати рівнину SQL з об’єктів jOOQ та коду, сформованого зі схеми бази даних. jOOQ використовує JDBC викликати базові запити SQL.

Хоча це забезпечує абстракція на додаток до JDBC, jOOQ не має такої ж функціональності та складності, як стандарт об’єктно-реляційне відображення бібліотеки, такі як EclipseLink або Зимовий сон.

Близькість jOOQ до SQL має переваги перед типовими об'єктно-реляційними бібліотеками зіставлення. SQL має багато функцій, які не можна використовувати в об'єктно-орієнтований парадигма програмування; цей набір відмінностей позначається як невідповідність об’єктно-реляційного імпедансу. Знаходячись близько до SQL, jOOQ допомагає запобігти синтаксичні помилки і проблеми зіставлення типів. Крім того, подбає про змінне зв’язування. Також у jOOQ можна створювати дуже складні запити, які включають псевдоніми, профспілки, вкладений виділяє та складні приєднання. jOOQ також підтримує специфічні для бази даних функції, такі як UDT, типи переліку, збережені процедури і рідні функції.

JOOQ – гарний вибір у програмі Java, де важливі SQL та конкретна реляційна база даних. Це альтернатива, коли JPA / Hibernate занадто багато абстрагує, JDBC занадто мало. Він показує, як сучасна предметно-орієнтована мова може значно підвищити продуктивність праці розробників, проваджуючи SQL у Java [20].

Окрім підключення до бази даних JOOQ створює власні файли які він використовую для написання запитів(рис 3.4). Конфігурація, яку ми створили під час цього повідомлення у блозі, забезпечує створення наступних класів:

Класи, створені в пакеті com.example.test.jooq.tables, містять метадані бази даних. jOOQ називає ці класи «глобальними» артефактами .

Класи у пакеті com.example.test.jooq.tables є класами таблиці, які описують структури таблиць бази даних. Можемо використовувати цей клас для запису запитів бази даних до даних, що зберігаються в таблиці бази даних todos [17].

Класи у пакеті com.example.test.jooq.tables.records класами записів, які містять інформацію про рядки таблиці. Запити до бази даних, які отримують дані з таблиці бази даних todos, повертають об'єкти Record (якщо ми вирішимо це зробити) (рис. 3.4).

```
<plugin>
    <groupId>org.jooq</groupId>
    <artifactId>jooq-codegen-maven</artifactId>
    <version>3.15.3</version>
    <executions>
        <execution>
            <phase>generate-sources</phase>
            <goals>
                <goal>generate</goal>
            </goals>
            <configuration>
                <jdbc>
                    <driver>org.postgresql.Driver</driver>
                    <url>jdbc:postgresql://localhost:5432/OSMgis</url>
                    <!--suppress UnresolvedMavenProperty -->
                    <user>${DB_USERNAME}</user>
                    <!--suppress UnresolvedMavenProperty -->
                    <password>${DB_PASSWORD}</password>
                </jdbc>
                <generator>
                    <database>
                        <includes>city|users|segments|ways|parking_ways|restriction_time</includes>
                    </database>
                    <target>
                        <packageName>com.example.test.jooq</packageName>
                        <directory>src/main/java</directory>
                    </target>
                </generator>
            </configuration>
        </execution>
```

Рисунок 3.4- Конфігурація для генерування файлів.

3.2 Висновки

У даному розділі було розглянуто можливі види отримання даних з OSM, розглянуто мову QL для написання запитів на Overpass Turbo написано запит для отримання потрібних даних по місцезнаходженню паркувальних місць. Вибрано СУБД для зберігання даних та розроблено базу даних веб-системи ідентифікації місць паркування, розгляну бібліотеки які можна використати для роботи з базою даних та створено базу даних за допомогою бібліотеки.

4 РОЗРОБЛЕННЯ ВЕБ-СИСТЕМИ ІДЕНТИФІКАЦІЇ МІСЦЬ ДЛЯ ПАРКУВАННЯ ЗА ДАНИМИ СЕРВІСУ OPENSTREETMAP

4.1 Архітектура програмного забезпечення веб-системи

Архітектура нашого сервісу базується на типовій моделі MVC.Наш рівень клієнта (подання) буде написаний на Java, Javascript, HTML. Цей рівень архітектури – це те, з чим користувач буде взаємодіяти, щоб отримати доступ до функцій нашого додатку. Перш за все, потрібно створити чітку структуру папок, файлів, імпортів та експортів. На рисунку 4.1 зображено файлову структуру яку використовує веб-сервіс.

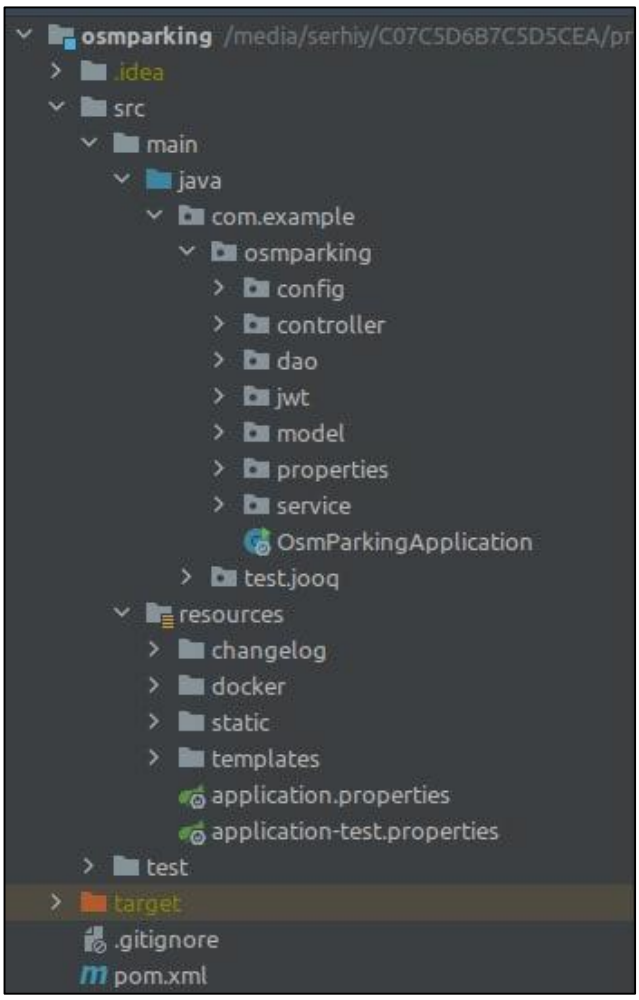


Рисунок 4.1 – Файлова структура клієнтської частини веб-сервісу

На рисунку 4.2 зображена модель веб-сервісу, за якою буде працювати веб-сервіс.

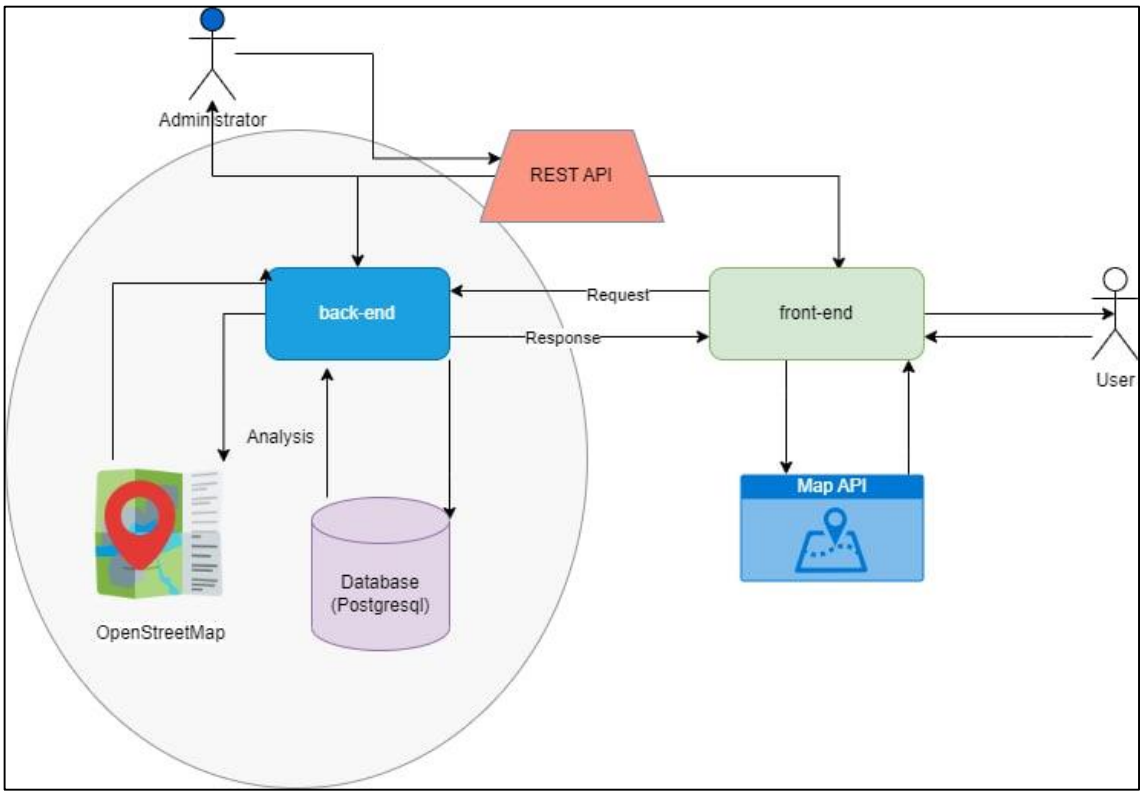


Рисунок 4.2 – Модель роботи веб-сервісу

Отже, після встановлення всіх залежностей веб-сервісу та налаштування середовища розробки, подальший прогрес над веб-сервісом не повинен вимагати занадто багато витрачених ресурсів.

4.2 Розробка бази даних та компонентів взаємодії з нею

Задля реалізації веб-сервісу ідентифікації місць для паркування, потрібно здійснити порівняльний аналіз існуючих технологій, для забезпечення дійсно швидко працюючого інструменту.

Для того щоб створити REST контролер (рис. 4.3) необхідно створити клас і позначити його анотацією @RestController та @RequestMapping, в параметрах якої вказується шлях до контролера. В самому класі визначаються

методи ендпоінти, кожний з яких позначається однією з анотацій @GetMapping, @PostMapping, @PutMapping, 21 @DeleteMapping. Ці методи приймають параметри, або DTO об’єкти з даними. Повертають методи дані у вигляді DTO об’єктів.

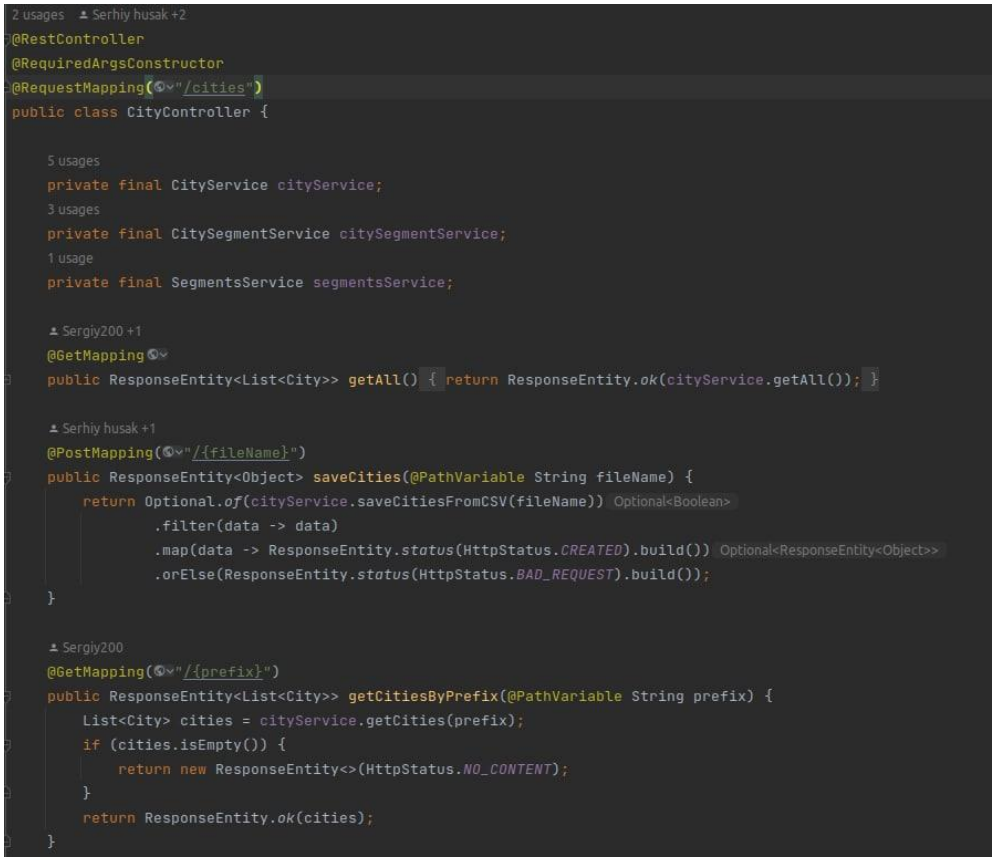
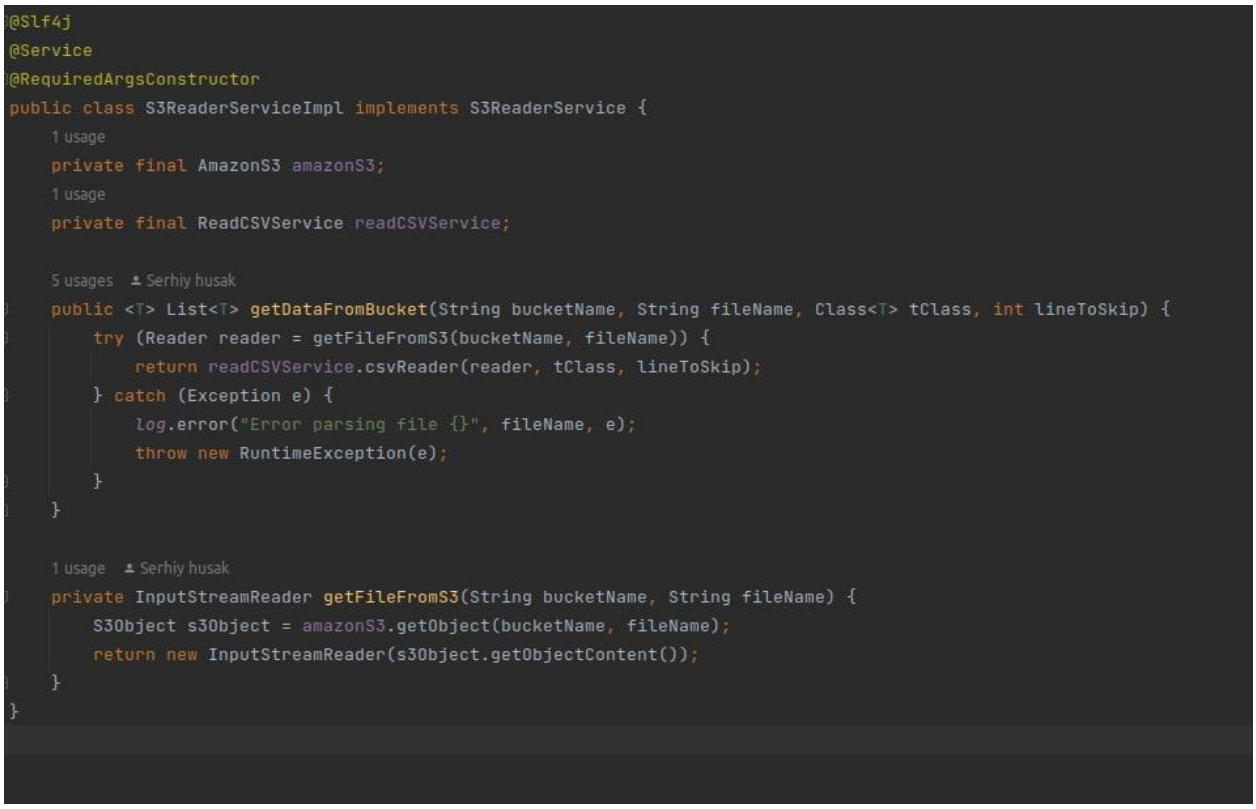


Рисунок 4.3 – Використання Rest контролерів для отримання даних

Amazon Simple Storage Service (Amazon S3) – це сервіс зберігання об'єктів, що пропонує найкращі в галузі показники продуктивності, масштабованості, доступності та безпеки даних. Клієнти будь-якої величини та з будь-якої промислової галузі можуть зберігати та захищати необхідний обсяг даних для практично будь-якого прикладу використання. Наприклад, для озер даних, хмарних додатків та мобільних додатків [18]. Вигідні класи сховища та прості у використанні інструменти адміністрування дозволяють оптимізувати витрати, організувати дані та точно налаштувати обмеження доступу

відповідно до потреб бізнесу чи законодавчих вимог. Та було додано логіку для отримання файлів із Amazon S3 (рис. 4.4)



```
@Slf4j
@Service
@RequiredArgsConstructor
public class S3ReaderServiceImpl implements S3ReaderService {
    1 usage
    private final AmazonS3 amazonS3;
    1 usage
    private final ReadCSVService readCSVService;

    5 usages  Serhiy husak
    public <T> List<T> getDataFromBucket(String bucketName, String fileName, Class<T> tClass, int lineToSkip) {
        try (Reader reader = getFileFromS3(bucketName, fileName)) {
            return readCSVService.csvReader(reader, tClass, lineToSkip);
        } catch (Exception e) {
            log.error("Error parsing file {}", fileName, e);
            throw new RuntimeException(e);
        }
    }

    1 usage  Serhiy husak
    private InputStreamReader getFileFromS3(String bucketName, String fileName) {
        S3Object s3Object = amazonS3.getObject(bucketName, fileName);
        return new InputStreamReader(s3Object.getObjectContent());
    }
}
```

Рисунок 4.4 – Зчитування файлів із Amazon S3

Отримавши файли із Amazon S3 потрібно отримати дані із CSV файлу та для цього було додано парсер який витягує дані із CSV у довільний тип даних. CSV – текстовий формат, призначений для представлення табличних даних. Рядок таблиці відповідає рядку тексту, який містить одне або кілька полів, розділених комами [17].

Формат CSV стандартизовано не повністю. Ідея використовувати коми для поділу полів очевидна, але при такому підході виникають проблеми, якщо вихідні табличні дані містять коми або переклади рядків. Можливим вирішенням проблеми ком і переносів рядків є укладання даних у лапки, однак вихідні дані можуть містити лапки. Крім цього терміном CSV можуть позначатися схожі формати, в яких роздільником є символ табуляції (TSV) або

точка з комою. Багато програм, які працюють із форматом CSV, дозволяють вибирати символ роздільника та символ лапок (рис. 4.5).

```
@Service
@RequiredArgsConstructor
public class ReadCSVServiceImpl implements ReadCSVService {

    1 usage  Serhiy husak
    @Override
    public <T> List<T> csvReader(Reader reader, Class<T> clazz, int lineToSkip) {
        return new CsvToBeanBuilder<T>(reader)
                .withType(clazz) CsvToBeanBuilder<T>
                .withSkipLines(lineToSkip)
                .build() CsvToBean<T>
                .parse();
    }
}
```

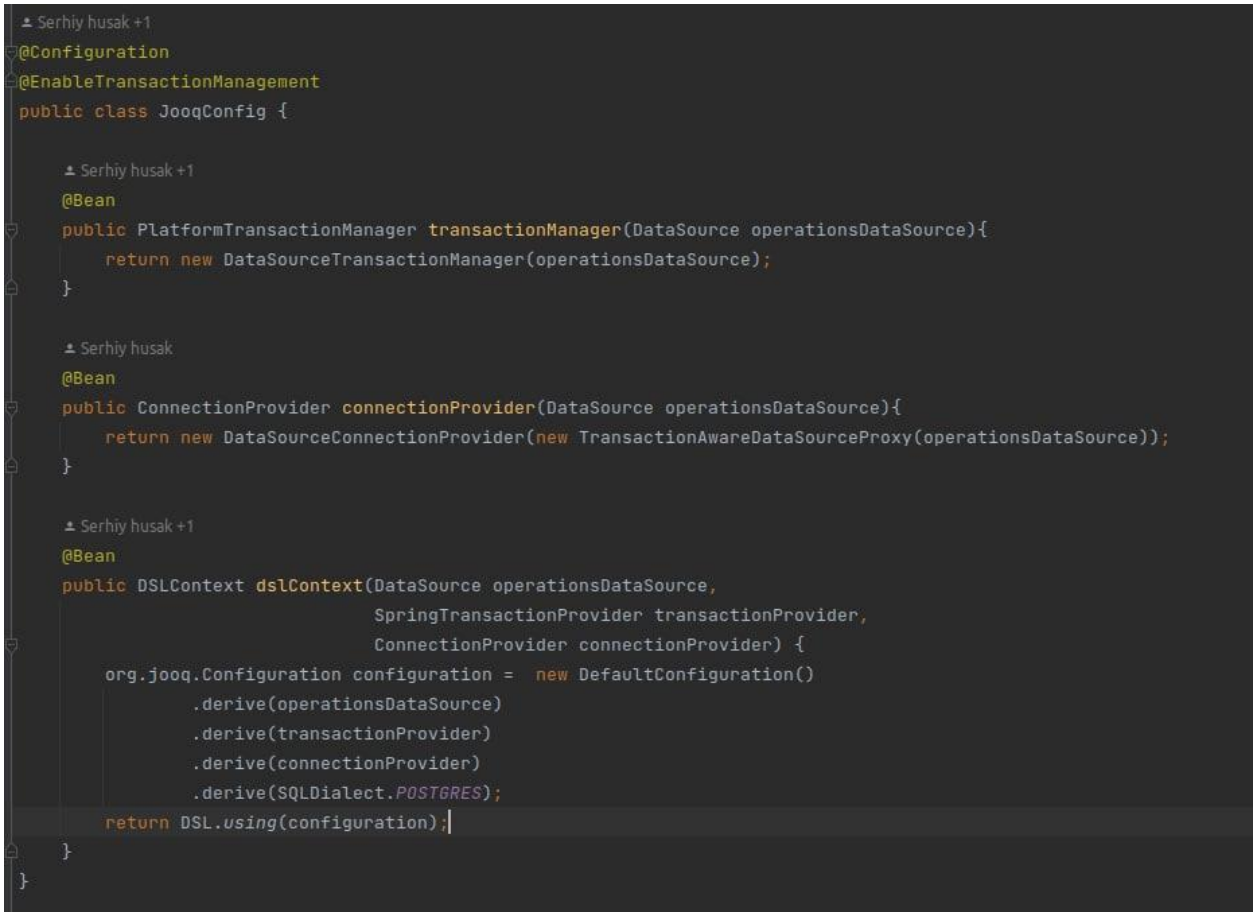
Рисунок 4.5 – Зчитування даних з CSV файлу

Також дані для обробки можна отримати із OpemStreetMap за допомогою Overpass API, що дуже зручно адже це відкриті дані (рис. 4.6).

```
8 usages  Serhiy husak
@Override
public <T> List<T> getElements(BoundingBox boundingBox, String queryPattern, Class<T> tClass) {
    String query = createQuery(boundingBox, queryPattern);
    try {
        return mapper
                .readerForListOf(tClass)
                .readValue(getContent(query));
    } catch (Exception e) {
        log.error("Error parsing request with BoundingBox {}", boundingBox, e);
        throw new RuntimeException(e);
    }
}
```

Рисунок 4.6 – Отримання даних з Overpass API.

Jooq має як спростили простоту та безпеку експлуатації традиційних даних OPM, і зберігає гнучкість рідного SQL, який більше схожий на проміжний шар ORMS та JDBC. Для коду ферми, які люблять писати SQL, Jooq може повністю задовольнити свій контроль, може бути використаний для запису SQL з кодом Java (рис. 4.7) [18].



```
Serhiy husak +1
@Configuration
@EnableTransactionManagement
public class JooqConfig {

    Serhiy husak +1
    @Bean
    public PlatformTransactionManager transactionManager(DataSource operationsDataSource){
        return new DataSourceTransactionManager(operationsDataSource);
    }

    Serhiy husak
    @Bean
    public ConnectionProvider connectionProvider(DataSource operationsDataSource){
        return new DataSourceConnectionProvider(new TransactionAwareDataSourceProxy(operationsDataSource));
    }

    Serhiy husak +1
    @Bean
    public DSLContext dslContext(DataSource operationsDataSource,
                                 SpringTransactionProvider transactionProvider,
                                 ConnectionProvider connectionProvider) {
        org.jooq.Configuration configuration =  new DefaultConfiguration()
                .derive(operationsDataSource)
                .derive(transactionProvider)
                .derive(connectionProvider)
                .derive(SQLDialect.POSTGRES);
        return DSL.using(configuration);
    }
}
```

Рисунок 4.7 – Конфігурація jooq для роботи з базою даних.

Також для користувачів було додано авторизацію яка була виконана за допомогою Spring Security. Зазвичай включення будь-якого стартера в POM-файл нічого не дає: щоб щось запрограмувати, треба написати додатковий код. У випадку Spring Security генерує пароль:

Так, Spring Security створив якогось користувача за умовчанням. Ім'я його user , а пароль генерується автоматично під час запуску програми.

Отже, що відбувається при одному тільки додаванні spring-boot-starter-security в POM-файл:

Spring Security створює користувача з ім'ям user та автоматично з генерованим паролем, який можна переглянути в консолі.

Створюється сторінка з формою для введення імені та пароля -маємо Form-based автентифікацію [18].

Ім'я та пароль реально перевіряються.

Всі url виявляються недоступними, поки ми не «залогінимся» під цим користувачем.

І ще створюється сторінка, де можна «розломитися». Вона знаходиться за адресою logout.

Автентифікацію виконує AuthenticationManager, але визначати цей бін явно нам не треба. Натомість треба перевизначити метод configure (AuthenticationManagerBuilder auth) класу WebSecurityConfigurerAdapter – так ми отримаємо доступ до білдера AuthenticationManagerBuilder, а через нього налаштуємо потрібний нам AuthenticationManager [17].

Далі йдуть специфічні налаштування вибраного AuthenticationManager. Вони уточнюється, як AuthenticationManager витягує збереженого користувача, щоб потім порівняти його з введеним (рис. 4.8).

```
@Configuration
@EnableWebSecurity
@RequiredArgsConstructor
public class WebSecurityConfig extends WebSecurityConfigurerAdapter {

    1 usage
    private final JwtFilter jwtFilter;
    2 usages
    private final WebSecurityProperties webSecurityProperties;

    Sergiy200 +1
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .csrf().disable() HttpSecurity
                .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS) SessionManagementConfigurer<HttpSecu
                .and() HttpSecurity
                .authorizeRequests() ExpressionUrlAuthorizationConfigurer<...>.ExpressionInterceptUrlRegistry
                .antMatchers(webSecurityProperties.getAdminPaths()).hasAuthority("ADMIN")
                .antMatchers(webSecurityProperties.getPermitAllPaths()).permitAll()
                .and() HttpSecurity
                .addFilterBefore(jwtFilter, UsernamePasswordAuthenticationFilter.class);
    }

    Sergiy200
    @Bean
    public PasswordEncoder passwordEncoder() { return new BCryptPasswordEncoder(); }
}
```

Рисунок 4.8 – Налаштування автентифікації для користувача.

Розглянемо структуру бази даних яку буде використовувати наш веб-сервіс (рис. 4.9)

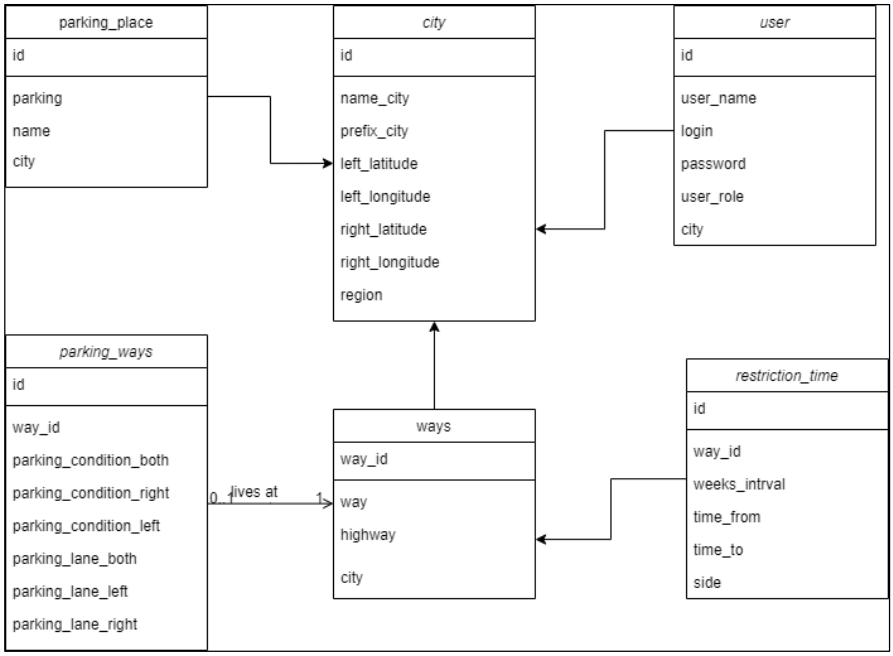


Рисунок 4.9 – Структура таблиць бази даних.

Для зручнішої роботи з базою даних було вирішено також підключити Liquibase (рис. 4.10).

```
<changeSet author="serhiy" id="users">
    <createTable tableName="users">
        <column autoIncrement="true" name="id" type="INTEGER">
            <constraints nullable="false" primaryKey="true" primaryKeyName="users_pk"/>
        </column>
        <column name="user_name" type="VARCHAR"/>
        <column name="login" type="VARCHAR"/>
        <column name="password" type="VARCHAR"/>
        <column name="user_role" type="VARCHAR"/>
    </createTable>
</changeSet>
<changeSet author="serhiy" id="city">
    <createTable tableName="city">
        <column autoIncrement="true" name="id" type="INTEGER">
            <constraints nullable="false" primaryKey="true" primaryKeyName="city_pk"/>
        </column>
        <column name="name_city" type="VARCHAR"/>
        <column name="prefix_city" type="VARCHAR"/>
        <column name="left_latitude" type="FLOAT8"/>
        <column name="left_longitude" type="FLOAT8"/>
        <column name="right_latitude" type="FLOAT8"/>
        <column name="right_longitude" type="FLOAT8"/>
        <column name="region" type="GEOMETRY"/>
    </createTable>
</changeSet>
<changeSet id="setUnique" author="serhiy">
    <addUniqueConstraint tableName="users" columnNames="login"/>
</changeSet>
```

Рисунок 4.10 – Приклад скрипта для Liquibase

Наведено додавання нового леєра на карту за допомогою MapBox для відображення даних на карті отриманих з серверної частини та оброблення респонсу на JavaScript (рис. 4.11).

```
function getWays($http, linesId) {
    $http.get('/segments/way', {
        params: {
            leftLatitude: firstPoint.lat,
            leftLongitude: firstPoint.lng,
            rightLatitude: secondPoint.lat,
            rightLongitude: secondPoint.lng
        }
    }).then(
        function successCallback(response) {
            let paint = {
                'line-width': 3,
                'line-color': '#25831c',
            };
            let lineLayer = new Layer(linesId, type: 'line', linesId, paint);
            drawLines(lineLayer, response.data);
        },
        function errorCallback(response) {
            console.log(response);
        });
}
```

Рисунок 4.11 – Приклад додавання нового леєра на карту

Точкою входу Spring Boot Application є клас, що містить анотацію @SpringBootApplication. Цей клас повинен мати основний метод для запуску програми Spring Boot [16]. Анотація @SpringBootApplication включає автоконфігурацію, сканування компонентів і конфігурацію завантаження Spring (рис. 4.12).

```
import ...

1 usage  Serhiy200 +2
@SpringBootApplication
public class OsmParkingApplication {

    Serhiy200 +1
    public static void main(String[] args) {
        SpringApplication.run(OsmParkingApplication.class, args);
    }
}
```

Рисунок 4.12 – Точка входу програми.

4.3 Програмна реалізація та застосування веб-системи

Ознайомлення та аналіз з даною предметною областю здійснювалось поетапно – від огляду набору даних до програмної реалізації використовуючи дані з відкритих джерел та протестована на Вінницькій області.

На всіх етапах підготовки програмного модуля до реалізації було проведено аналіз з метою зменшення можливості виникнення помилок та незапланованих сценаріїв роботи в майбутньому. Застосовано дані з OpenStreetMap для ідентифікації місць паркування для Вінницької області.

– розроблено авторизацію у веб систему для надання певним користувачам доступ для оновлення карти (рис. 4.13)

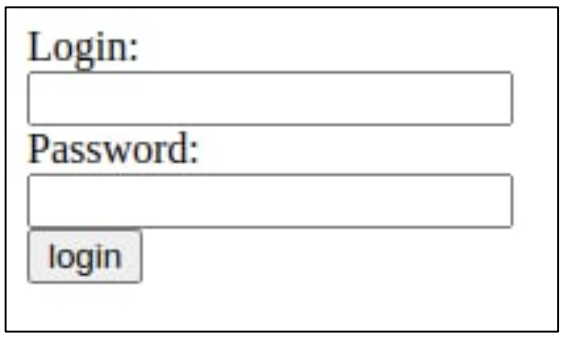


Рисунок 4.13 – Форма яка використовується для авторизації користувача

– відображено інтерактивну карту з бібліотеки MapBox, яку можна приховати при необхідності (рис.4.14).

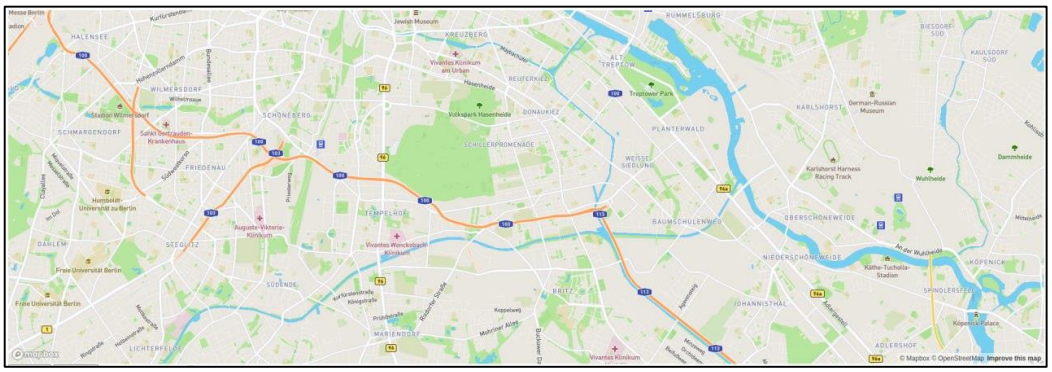


Рисунок 4.14 – Інтерактивна карта

– додано вибір області для відображення даних на карті (рис. 4.15).

Рисунок 4.15 – Виділена область для відображення даних.

– додано відображення актуальних місць для паркування у Вінницькій області за даними сервісу OpenStreetMap (рис 4.16).

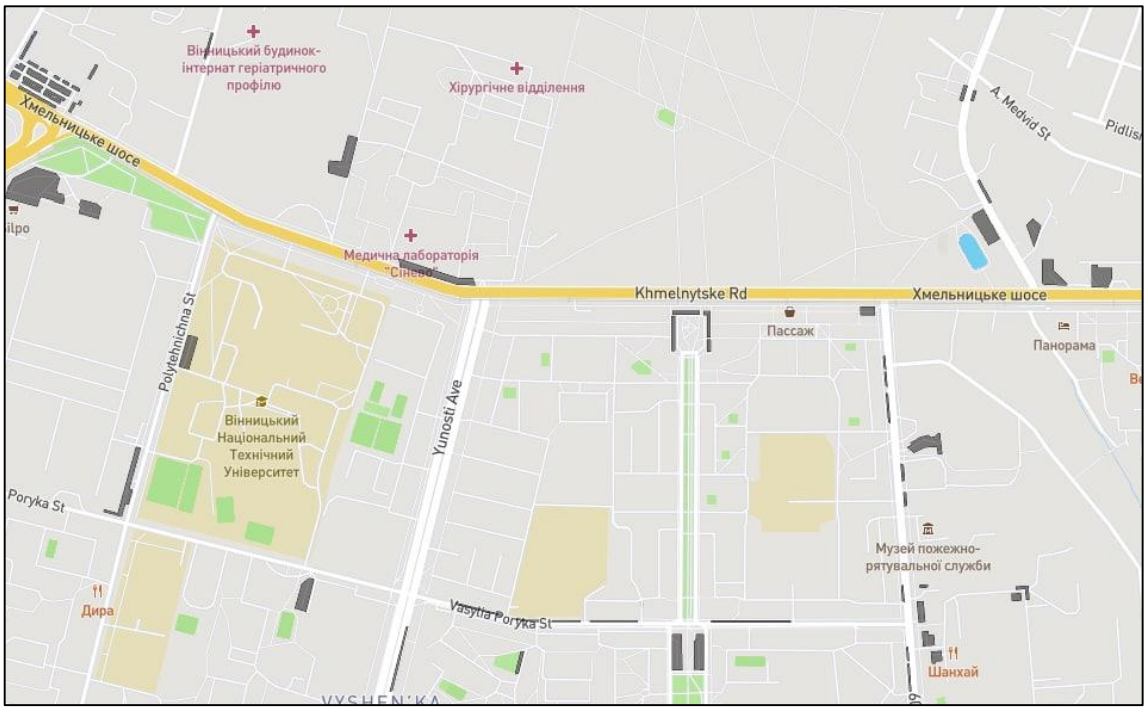


Рисунок 4.16 – Відображення місць для паркування у Вінниці.

– додано відображення всіх доріг на карті (рис. 4.17).

Рисунок 4.17 – Відображення всіх доріг у вибраній області

– додано відображення червоним кольором частин дороги де не можна паркуватись через забороняючий знак (рис. 4.18).

Рисунок 4.18 – Відображення доріг на карті із забороненим місцем паркування.

Усі запити із клієнтської частини обробляються за допомогою такого концепту як роутинг та концепту передачі даних REST API.

REST API – це прикладний програмний інтерфейс (API), який використовує HTTP-запити для отримання, вилучення, розміщення і видалення даних. Абревіатура REST в контексті API розшифровується як «передача стану уявлення» (Representational State Transfer).

```
@PostMapping("/city")
public ResponseEntity<Object> addCity(@RequestBody CityDto cityDto) {
    return Optional.of(cityService.addCity(cityDto)) Optional<Boolean>
            .filter(data -> data)
            .map(data -> ResponseEntity.status(HttpStatus.CREATED).build()) Optional<ResponseEntity<Object>>
            .orElse(ResponseEntity.status(HttpStatus.BAD_REQUEST).build());
}
```

Рисунок 4.19 – Фрагмент обробки POST-запиту веб-системи

Основна ідеї REST API полягає в поділі різних при зверненні до одного і того ж URL за допомогою HTTP методів, основні з яких:

– GET – використовується для отримання даних;

– POST – використовується для створення нового запису;

– PUT – використовується для оновлення вже існуючої записи;

– PATCH – використовується для оновлення, але тільки тоді, коли змінюється ідентифікатор запису;

– DELETE – використовується для видалення запису [9].

На рисунках 4.19 та 4.20 зображено обробку запиту із клієнтської частини та його зворотня відповідь.


```
"type": "FeatureCollection",
"features": [
    {
        "geometry": {
            "type": "LineString",
            "coordinates": [
                [
                    28.4260128,
                    49.2277116
                ],
                [
                    28.4266778,
                    49.2277435
                ],
                [
                    28.4267109,
                    49.2274339
                ],
                [
                    28.427681,
                    49.2274753
                ],
                [
                    28.427768,
                    49.2267228
                ],
                [
                    28.4271892,
                    49.226695
                ],
                [
                    28.4271495,
                    49.227048
                ],
```


Рисунок 4.20 – Відповідь обробки POST-запиту

Дані у даному форматі використовуються для відображенн їх накарті. Після того як, серверна частина була реалізована, щоб запустити у безперебійну роботу. Результат роботи сервера зображено на рисунку 4.21.

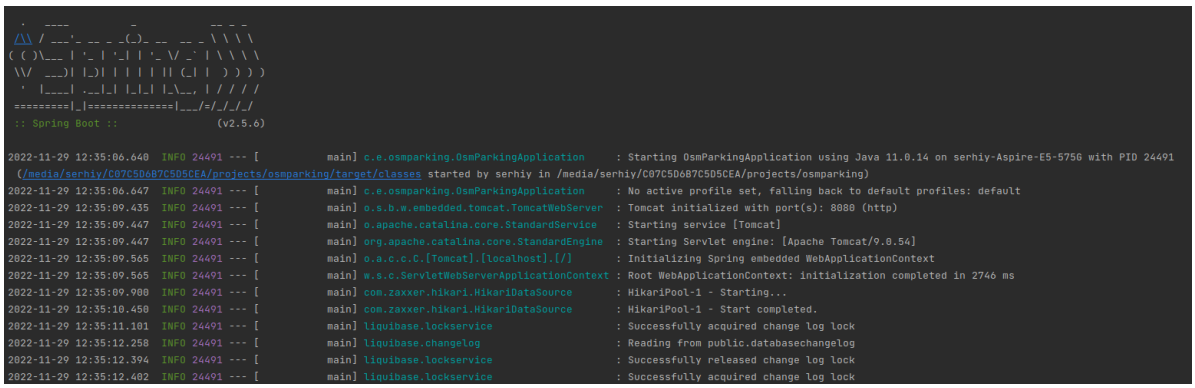


Рисунок 4.21 – Запуск сервера системи

Після того, як серверна частина була реалізована, саме час приступити до розробки інтерфейсу користувача. Як раніше було вже згадано в цьому, допоможе бібліотека javascript AngularJs.

Перш за все, щоб згенерувати базові залежності проекту потрібно скористатися інструментом AngularJs виклику даних з серверної частини [18].

На рисунку 4.22 зображени приклад виклику даних з серверної частини.

```
function getParkings($http, polygonId) {
    $http.get('/parking', {
        params: {
            leftLatitude: firstPoint.lat,
            leftLongitude: firstPoint.lng,
            rightLatitude: secondPoint.lat,
            rightLongitude: secondPoint.lng
        }
    }).then(
        function successCallback(response) {
            console.log("succes");
            let paint = {
                'fill-color': '#0e0e0e',
                'fill-opacity': 0.5
            };
            let lineLayer = new Layer(polygonId, type: 'fill', polygonId, paint);
            drawLines(lineLayer, response.data);
        },
        function errorCallback(response) {
            console.log("error");
            console.log(response);
        });
}
```

Рисунок 4.22 – Виклик даних з серверної частини

AngularJs Router – одна з найпопулярніших платформ маршрутизації. Бібліотека розроблена з інтуїтивно зрозумілими компонентами, що дозволяє створити декларативну систему маршрутизації для вашого додатка. Це

означає, що ви можете точно заявити, який із ваших компонентів має певний маршрут. За допомогою декларативної маршрутизації ви можете створювати інтуїтивно зрозумілі маршрути, зручні для читання, полегшуючи управління архітектурою вашого додатка [17].

На рисунку 4.23 відображено налаштування роутингу між компонентами інтерфейсу.

```
wayManagement.factory('loginInterceptor', ['$q', '$window', function ($q, $window) {
    return {
        responseError: function (response) {
            console.log('Failed with', response.status, 'status');
            if (response.status === 403 || response.status === 401) {
                $window.location.replace('#!login');
            }
            return $q.reject(response);
        }
    }
}])

wayManagement.config(function ($httpProvider) {
    $httpProvider.interceptors.push('loginInterceptor')
})

wayManagement.config(function ($routeProvider) {
    $routeProvider.when("/", {
        templateUrl: 'views/mapbox.html',
        controller: 'way-controller',
    })
    $routeProvider.when("/login", {
        templateUrl: 'views/login.html',
        controller: 'way-controller',
    })
})
```

Рисунок 4.23 – Налаштування роутингу веб-системи

4.4 Висновки

Було розроблено архітектуру програмного забезпечення, де було описано всі найважливіші аспекти розробки веб-застосунку. Розроблено веб-систему ідентифікації місць для паркування та протестовану у місті Вінниця де веб-система успішно знайшла та показала на карті місця для паркування

біля Вінницького національного технічного університету. Дані з OpenStreetMap дійсно підвищило точність ідентифікації місць паркування на карті. Розроблено можливість виділити на карті область у вигляді BoundingBox та вибрати які дані відобразити у даній області, це можуть бути місця для паркування частини дороги де паркування заборонене через забороняючий знак, також місця де паркування заборонене у певний період часу, наприклад з 7 до 9 годин ранку.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу інформаційної веб-системи ідентифікації місць паркування за даними сервісу OpenStreetMap.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету кафедри системного аналізу та інформаційних технолгій: к.т.н., доц. Козачко О.М., к.т.н., доц. Крижановський Є. М., к.т.н., доц. Варчук І. В. Для проведення технологічного аудиту було використано таблицю 5.1 в якій за п’ятибальною шкалою використовуючи 12 критеріїв здійснено оцінку комерційного потенціалу.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї

Продовження таблиці 5.1

Кри-терій	0	1	2	3	4
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документі	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
12	в на виробництво та реалізацію продукту	значних коштів та часу			

В таблиці 5.2 наведено рівні потенціалу розробки за допомогою якої після проведення оцінювання можна зробити висновки чи варто займатися розробкою даної веб-системи

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

В таблиці 5.3 наведено результати оцінювання експертами комерційного потенціалу розробки.

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	Козачко О.М.	Крижановський Є.М.	Варчук І.В.
	Бали, виставлені експертами:		
1	3	1	2
2	4	5	1
3	2	1	2
4	1	3	3
5	3	2	2
6	2	3	1
7	1	2	3

Продовження таблиці 5.3

Критерії	Прізвище, ініціали, посада експерта		
	Козачко О.М.	Крижановський Є.М.	Варчук І.В.
	Бали, виставлені експертами:		
8	4	1	4
9	2	4	3
10	3	3	2
11	5	3	4
12	2	3	3
Сума балів	$СБ_1$=32	$СБ_2$=31	$СБ_3$=30
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{32+31+30}{3} = 31$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 31 бали, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Інформаційної веб-система аналізу динаміки географічної структури зовнішньої торгівлі товарами України, а саме програма, яка допомагає проводити моніторинг, аналізувати та порівнювати дані. Дана програма буде цікава різним організаціям та компаніям в яких цільовий ринок направлений закордон.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи групуються за такими статтями: витрати на оплату праці, витрати на соціальні заходи, матеріали, паливо та енергія для науково-виробничих цілей, витрати на службові відрядження, програмне забезпечення для наукових робіт, інші витрати, накладні витрати.

Основна заробітна плата кожного із дослідників $З_о$, якщо вони працюють в наукових установах бюджетної сфери визначається за формулою:

$$З_0 = \frac{М}{Т_р} * t \text{ (грн)}, \quad (5.1)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

T_P – число робочих днів в місяці; приблизно $T_P \approx 21...23$ дні;

t – число робочих днів роботи дослідника.

Для розробки програмні засоби необхідно залучити програміста з посадовим окладом 17000 грн. Кількість робочих днів у місяці складає 40, а кількість робочих днів програміста складає 22. Зведемо сумарні розрахунки до таблиця 5.4.

Таблиця 5.4 – Заробітна плата дослідника в науковій установі бюджетної сфери

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Керівник	23000	1095,2	5	5476
Програмний інженер	17000	809,5	40	32380
Всього				37856

Розрахунок додаткової заробітної плати робітників. Додаткова заробітна плата $З_д$ всіх розробників та робітників, які приймали устають в розробці нового технічного рішення розраховується як 10 - 12 % від основної заробітної плати робітників.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати розраховується за формулою:

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (5.2)$$

$$З_д = 0{,}1 * 37856 = 3785 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{ЗП}$ дослідників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$Н_{ЗП} = (З_о + З_д) * \frac{\beta}{100}, \quad (5.3)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

β – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, % .

Дана діяльність відноситься до бюджетної сфери, тому ставка єдиного внеску на загальнообов'язкове державне соціальне страхування буде складати 22%, тоді**:**

$$Н_{ЗП} = (37856 + 3785\ ) * \frac{22}{100} = 9161 \text{ (грн)}.$$

Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_і * Ц_і * К_і, \quad (4.5)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Також в таблиці 5.5 розглянемо комплектуючі які необхідні для розробки інформаційної системи ідентифікації місць для паркування

Таблиця 5.5 – Комплектуючі, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Папір	150	1	150
Ручка	20	1	20
CD-диск	30	1	30
Флешка	150	1	150
Всього			350
З врахуванням коефіцієнта транспортування			400

Програмне забезпечення для наукової роботи включає витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення необхідного для проведення дослідження.

Для написання магістерської роботи використовувалися редактор Intellij idea та база даних PostgreSQL.

Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо розраховують за формулою:

$$A = \frac{Ц * Т}{Т_{кор} * 12}, \quad (5.6)$$

де Ц – балансова вартість даного виду обладнання (приміщень), грн.;

$Т_{кор}$ – час користування;

Т – термін використання обладнання (приміщень), цілі місяці.

Згідно пункта 137.3.3 Податкового кодекса амортизація нараховується на основні засоби вартістю понад 2500 грн. В нашому випадку для написання магістерської роботи використовувався персональний комп'ютер вартістю 20000 грн.

$$А = \frac{20000 \cdot 1}{2 \cdot 12} = 833 \text{ (грн)}.$$

До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень розраховують за формулою:

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i}, \quad (5.7)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

Для написання магістерської роботи використовується персональний комп'ютер для якого розрахуємо витрати на електроенергію.

$$В_е = \frac{0{,}3 \cdot 200 \cdot 1{,}68 \cdot 0{,}5}{0{,}8} = 63 \text{ (грн)}.$$

Витрати на службові відрядження, витрати на роботи, які виконують сторонні підприємства, установи, організації та інші витрати в нашому досліджені не враховуються оскільки їх не було.

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як (100…150)% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.8)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 37856 \cdot \frac{100}{100\%} = 37856(грн).$$

Сума всіх попередніх статей витрат дає витрати, які безпосередньо стосуються даного розділу МКНР

$$В = 37856+3785+9161+400+833+63+37856=89954 \text{ (грн)}.$$

Прогнозування загальних втрат ЗВ на виконання та впровадження результатів виконаної МКНР здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.9)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії науково-дослідних робіт, то коефіцієнт β = 0,9.

Звідси:

$$ЗВ = \frac{89954}{0,9} = 99948,9(грн).$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової

роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою:

$$\Delta П_i = \sum_1^n (\Delta Ц_o * N * Ц_o * \Delta N)_i * \lambda * \rho * \left(1 - \frac{\upsilon}{100}\right), \quad (5.10)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

ρ – коефіцієнт, який враховує рентабельність продукту. ρ = 0,25;

υ – ставка податку на прибуток. У 2022 році – 18%.

Припустимо, що ціна за програмний продукт зросте на 600 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року на 45 шт., протягом другого року – на 35 шт., протягом третього року на 25 шт. Реалізація продукції до впровадження розробки складала 1 шт., а її ціна до складає 10000 грн. Розрахуємо прибуток, яке отримає підприємство протягом трьох років.

$$\Delta П_1 = [600 \cdot 1 + (10000 + 600) \cdot 45] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 117815.59 \text{ (грн)}.$$

$$\Delta П_2 = [600 \cdot 1 + (10000 + 600) \cdot (45 + 35)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 208983.28 \text{ (грн)}.$$

$$\Delta П_3 = [600 \cdot 1 + (10000 + 600) \cdot (45 + 35 + 25)] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 274103.05(\text{грн}).$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки за формулою:

$$PV = k_{\text{інв}} \cdot \text{ЗВ}, \qquad (5.11)$$

де $k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = 2…5).

$$PV = 2 \cdot 84586{,}7 = 169173.4(\text{грн}).$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \qquad (5.12)$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t}\ , \qquad (5.13)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{117815.59}{(1+0{,}2)^1} + \frac{208983.28}{(1+0{,}2)^2} + \frac{274103.05}{(1+0{,}2)^3} = 401931.37 \text{ (грн)}.$$

$$Е_{абс} = (401931.37 - 169173.4) = 232757.97 \text{ (грн)}.$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього користуються формулою:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1\ , \qquad (5.14)$$

де $Т_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[3]{1 + \frac{232757.97}{169173.4}} - 1 = 0{,}26 = 26\%.$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f\ , \tag{5.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина $f = (0{,}05\ldots0{,}1)$.

$$\tau_{min} = 0{,}16 + 0{,}05 = 0{,}21.$$

Так як $E_в > \tau_{min}$ то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$\mathrm{T_{ок}} = \frac{1}{\mathrm{E_в}}\,. \tag{5.16}$$

$$\mathrm{T_{ок}} = \frac{1}{0{,}26} = 3{,}8 \text{ (роки)}.$$

Так як $\mathrm{T_{ок}} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

5.5 Висновки

Проведено оцінку комерційного потенціалу інформаційної веб-системи інформаційної веб-системи ідентифікації місць для паркування за даними

сервісу OpenStreetMap, а саме програма отримує дані з OpenStreetMap та відображує їх на карті.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 89954 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 84586,7 грн.

Вкладені інвестиції в даний проект окупляться через 3,8 роки, приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки склала 401931,37 грн.

ВИСНОВКИ

Проведено аналіз предметної галузі, охарактеризовано, пошук місць для паркування, визначено, що для виконання даного завдання необхідно працювати із географічними даними. Здійснено огляд відомих аналогів і більшість з них працюють чудово та мають можливість оплати паркування.

Відбулося формування програмних вимог для веб-сервісу. Було проведено відбір оптимальних інформаційних технологій для серверної частини було обрано мову програмування. Розглянуто сервіси для отримання даних по місцезнаходженню місця для паркування та їх зображення на карті.

Розглянуто можливі види отримання даних з OSM, розглянуто мову QL для написання запитів для отримання та обробки даних з джерела OpenStreetMap. Розглянуто технології для роботи з базами даних та розроблено схему бази даних веб-системи ідентифікації місць паркування. Та проведено збір даних по місцезнаходженню місць для паркування та зберігання їх у базу даних для подальшого відображення їх на карті.

Проаналізовано передові технологій, які б задовольнили весь функціонал. Дані технології чітко поєднуються між собою, створюючи потужний та швидкий веб-додаток, який забезпечує користувача всіма необхідними ресурсами та покращує ідентифікацію місць для паркування.

Розроблено архітектуру програмного забезпечення, де було описано всі найважливіші аспекти розробки веб-застосунку. Розроблено веб-систему ідентифікації місць для паркування, яка надає користувачеві можливість:

- авторизуватись на сайті;
- відкрити карту;
- знаходження місць паркування за BoundingBox;
- завантажити дані з OSM;
- завантаження даних по дорогах з Amazon S3 bucket;
- розрахунок заборонених місць для паркування на дорозі.

Веб-система ідентифікації місць для паркування була перевірена у Вінницькій області, де веб-система успішно знайшла та показала на карті місця для паркування біля Вінницького національного технічного університету. Дані з OpenStreetMap підвищили точність ідентифікації місць для паркування.

Проведено оцінку комерційного потенціалу інформаційної веб-системи ідентифікації місць для паркування за даними сервісу OpenStreetMap, а саме програма отримує дані з OpenStreetMap та відображує їх на карті.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 89954 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 84586,7 грн. Вкладені інвестиції в даний проект окупляться через 3,8 роки.

Магістерська кваліфікаційна робота виконана у повному обсязі, розроблено веб-систему для підвищення точності ідентифікації місць паркування за даними сервісу OpenStreetMap у місті Вінниця.

За результатами даної роботи були написані тези доповіді, які в підсумку, були опубліковані в збірнику матеріалів Всеукраїнської науково-практичної інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи».

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Крижановський Є. М., Гусак С.В. Інформаційна веб-система ідентифікації місць для паркування за даними сервісу OpenStreetMap. *Всеукраїнська науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2022-2023 рр.)*. URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/viewFile/16821/14039

2. OpenStreetMap. URL: https://uk.wikipedia.org/wiki/OpenStreetMap

3. Mobile-parking. URL: https://ktps.kyiv.ua/services/pay/mobile-parking

4. Написання запитів для отримання даних з OpenStreetMap. URL: https://coderlessons.com/articles/java/ispolzovanie-jooq-s-spring-generatsiia-koda

5. Spring security. URL: https://sysout.uk/dobavlenie-spring-security-v-proekt-nastrojki-po-umolchaniyu/

6. Overpass API. URL: https://wiki.openstreetmap.org/wiki/Overpass_API

7. Jooq spring. URL: https://coderlessons.com/articles/java/ispolzovanie-jooq-s-spring-generatsiia-koda

8. Штрафи за паркування. URL: https://konkurent.ua/publication/4792/ak-za-kordonom-virishili-problemu-z-parkovkami/

9. Web-орієнтована система моніторингу автостоянок. URL: http://dspace.wunu.edu.ua/bitstream/316497/18094/1/%D0%92%D1%96%D0%B2%D1%87%D0%B0%D1%80.pdf

10. Web-орієнтована розумна парковка. URL: http://elar.khmnu.edu.ua/jspui/bitstream/123456789/11951/1/%D0%94%D0%A0_%D0%9A%D0%BE%D0%B2%D0%B0%D0%BB%D0%B5%D0%BD%D0%BA%D0%BE%20%D0%92.%D0%92._%D0%9A%D0%862%D0%BC-20-1%20%281%29.pdf

11. Мова програмування. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

12. Об’єктно-орієнтоване програмування. URL: https://uk.wikipedia.org/wiki/%D0%9E%D0%B1%27%D1%94%D0%BA%D1%82%D0%BD%D0%BE%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

13. Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Upper Saddle River, NJ, 2019. 464 p.

14. Brett D. McLaughlin, Head First Object-Oriented Analysis and Design, O'Reilly Media, California, 2006. 236 c.

15. Josh Long Cloud Native Java: Designing Resilient Systems with Spring Boot, Spring Cloud, and Cloud Foundry 1st Edition, O'Reilly Media, California, 2020. 624 p.

16. Joshua Bloch, Effective Java 3rd Edition Addison-Wesley Professional, O'Reilly Media, California, 2020. 392 p.

17. Mark Heckler, Spring Boot: Up and Running. 1st, , O'Reilly Media, USA, 2021 300 p.

18. Роберт С. Мартін, Чистий код, Фабула, Харків, 2019. 416 с.

19. Роберт С. Мартін, Чиста архітектура, Фабула, Харків, 2019. 416 с.

20. Henrietta Dombrovska, Boris Novikov, Anna Bailliekova, PostgreSQL Query Optimization. 1st, Apress, NY, 2021. 319 p.

Додаток А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації

ЗАТВЕРДЖУЮ
Завідувач кафедри САІТ
_____ д.т.н., проф. Мокін В. Б.
«_19_» _____09_____ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на магістерську кваліфікаційну роботу

«Інформаційна веб-система ідентифікації місць для паркування за даними сервісу OpenStreetMap»

08-53.МКР.002.02.000.ТЗ

Керівник: к.т.н., доц. каф. САІТ
_____ Крижановський Є.М.
«_19_» _____09_____ 2022 р.

Розробив: студент гр. 2ІСТ-21м
_____ Гусак С.В.
«_19_» _____09_____ 2022 р.

Вінниця 2022

1. Підстава для проведення робіт

Підставою для виконання роботи є наказ № 203 по ВНТУ від «14» _09_ 2022 р., та індивідуальне завдання на МКР, затверджене протоколом № 3 засідання кафедри САІТ від «14» __09___ 2022 р.

2. Джерела розробки:

− Крижановський Є. М., Мокін В.Б., Яцолт А.Р., Скорина Л.М.. Системний аналіз та проектування ГІС. – Електронний навчальний посібник Вінниця : ВНТУ, 2015. – 127 с.

− Spring security. URL: https://sysout.uk/dobavlenie-spring-security-v-proekt-nastrojki-po-umolchaniyu/

− Robert C. Martin, Clean Code: A Handbook of Agile Software Craftsmanship, Upper Saddle River, NJ, 2019, 464 p.

3. Мета і призначення роботи:

Метою є розробка системи ідентифікації місць для паркування України, яка на основі вибірок із бази даних буде проводити аналіз паркувальних місць.

4.Вихідні дані для проведення робіт:

Дані отримані по паркувальних місцях з OpenStreetMap.

5.Методи дослідження:

Методи системного аналізу даних. Класифікація об’єкту дослідження

6. Етапи роботи і терміни їх виконання:

1. Аналіз проблем ідентифікації місць паркування..... 19.09 – 25.09
2. Огляд та обробка вхідних даних ..... 26.09 – 30.09
3. Вибір оптимальних інформаційних технологій..... 05.10 – 13.10
4. Розробка веб-системи для аналізу даних..... 15.10 – 30.10
5. Економічна частина ..... 12.11 – 23.11
6. Оформлення пояснювальної записки. ..... 24.11 – 29.11

7. Очікувані результати та порядок реалізації:

Отримання повнофункціональної системи ідентифікації місць для паркування.

8. Вимоги до розробленої документації

Пояснювальна записка оформлена у відповідності до вимог «Методичних вказівок до виконання та оформлення магістерських кваліфікаційних робіт для студентів спеціальності 126 – «Інформаційні системи та технології» денної форми навчання».

9. Порядок приймання роботи

Публічний захист ..... «_19_» _____12___ 2022 р.

Початок розробки..... «_19_» __ ____09____ 2022 р.

Граничні терміни виконання МКР ..... «_30_» _____11____ 2022 р.

Розробив студент групи 2ІСТ-21м _____ Гусак С.В.

Додаток Б

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: «Інформаційна веб-система ідентифікації місць для паркування за даними сервісу OpenStreetMap»
Тип роботи: магістерська кваліфікаційна робота
Підрозділ: кафедра САІТ
Науковий керівник: Крижановський Є.М. к.т.н., доц. каф. САІТ

Показники звіту подібності Unicheck

Оригінальність	86,3 %
Схожість	14,7 %

Аналіз звіту подібності (відмітити потрібне):

☑ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Опис прийнятого рішення:

Робота допускається до захисту

Особа, відповідальна за перевірку Жуков С. О.

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи Гусак С.В.

Керівник роботи Крижановський Є.М.

Додаток В

Лістинг програми

```
package com.example.osmparking;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class OsmParkingApplication {

    public static void main(String[] args) {
        SpringApplication.run(OsmParkingApplication.class, args);
    }
}
package com.example.osmparking.controller;

import com.example.osmparking.model.BoundingBox;
import com.example.osmparking.service.ParkingService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequiredArgsConstructor
@RequestMapping("parking")
public class ParkingPlacesController {
    private final ParkingService;
```

```
    @PostMapping("osm")
    public ResponseEntity<Object> findParking(BoundingBox boundingBox)
{
        return ResponseEntity.ok(parkingService.loadParking(boundingBox));
    }

    @GetMapping
    public ResponseEntity<Object> getParking(BoundingBox boundingBox) {
        return ResponseEntity.ok(parkingService.getParking(boundingBox));
    }
}
package com.example.osmparking.dao.impl;

import com.example.osmparking.dao.ParkingPlaceDao;
import com.example.osmparking.model.BoundingBox;
import com.example.osmparking.model.GeoJson;
import com.example.osmparking.model.ParkingPlace;
import com.fasterxml.jackson.databind.ObjectMapper;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.jooq.DSLContext;
import org.jooq.Query;
import org.jooq.Record1;
import org.springframework.stereotype.Repository;

import java.util.List;
import java.util.stream.Collectors;

import static com.example.osmparking.dao.fields.Fields.*;
```

```
import static com.example.test.jooq.public_.Tables.PARKING_PLACES;
import static com.example.test.jooq.public_.tables.Ways.WAYS;
import static java.text.MessageFormat.format;

@Slf4j
@Repository
@RequiredArgsConstructor
public class ParkingPlacesDaoImpl implements ParkingPlaceDao {

   private static final String PARKING_PLACE_GEO = "parking_place";
   private final DSLContext;
   private final ObjectMapper;

   @Override
   public void upsert(List<ParkingPlace> parkingPlaces) {
      List<Query> queries = buildQueries(parkingPlaces);
      dslContext.batch(queries).execute();
   }

   @Override
   public List<GeoJson> get(BoundingBox boundingBox) {
      return dslContext

.select(stGeoJson((PARKING_PLACES.PARKING)).as(PARKING_PLACE_GE
O))
            .from(PARKING_PLACES)
            .where(stContains(getBboxField(boundingBox),
PARKING_PLACES.PARKING))
            .fetch(this::getGeoJson);
   }
```

```
        private GeoJson getGeoJson(Record1<String> geoJson) {
            try {
                return
objectMapper.readValue(geoJson.get(PARKING_PLACE_GEO,       String.class),
GeoJson.class);
            } catch (Exception e) {
                log.error("Error parsing result {}", geoJson);
                throw new RuntimeException(format("Error parsing result {0} {1}",
geoJson, e));
            }
        }

        private List<Query> buildQueries(List<ParkingPlace> parkingPlaces) {
            return parkingPlaces.stream()
                    .map(parkingPlace -> dslContext.insertInto(PARKING_PLACES,
PARKING_PLACES.ID,                                PARKING_PLACES.NAME,
PARKING_PLACES.PARKING)
                            .values(parkingPlace.getId(),          parkingPlace.getName(),
parkingPlace.getGeometry())
                            .onConflict(PARKING_PLACES.ID)
                            .doUpdate()
                            .set(PARKING_PLACES.NAME, parkingPlace.getName())
                            .set(PARKING_PLACES.PARKING,
parkingPlace.getGeometry()))
                    .collect(Collectors.toUnmodifiableList());
        }
    }
    package com.example.osmparking.dao.fields;
```

```
import com.example.osmparking.model.BoundingBox;
import lombok.AccessLevel;
import lombok.NoArgsConstructor;
import org.jooq.Field;
import org.jooq.QueryPart;

import static org.jooq.impl.DSL.field;

@NoArgsConstructor(access = AccessLevel.PRIVATE)
public class Fields {
    public static Field<?> stAsText(QueryPart... parts) {
        return field("st_asText({0})", parts);
    }

    public static Object setToGeometry(Object... bindings) {
        return field("{0}::geometry", bindings);
    }

    public static Field<Boolean> stIntersects(QueryPart... parts) {
        return field("ST_Intersects({0}, {1})", boolean.class, parts);
    }

    public static Field<Object> getBboxField(BoundingBox boundingBox) {
        return field("'BOX(" +
                boundingBox.getLeftLongitude() + " " +
boundingBox.getLeftLatitude() + "," +
                boundingBox.getRightLongitude() + " " +
boundingBox.getRightLatitude() + ")'::box2d"
        );
    }
```

```
    public static Field<Boolean> stContains(QueryPart... parts) {
        return field("ST_Contains({0}, {1})", boolean.class, parts);
    }

    public static Field<String> stGeoJson(Object... geometry) {
        return field("ST_asGeoJson({0})", String.class, geometry);
    }

    public static Field<Object> getGeometrySRID(String geometry) {
        return field("st_geometryFromText({0},4326)", geometry);
    }

}
```

Додаток Г

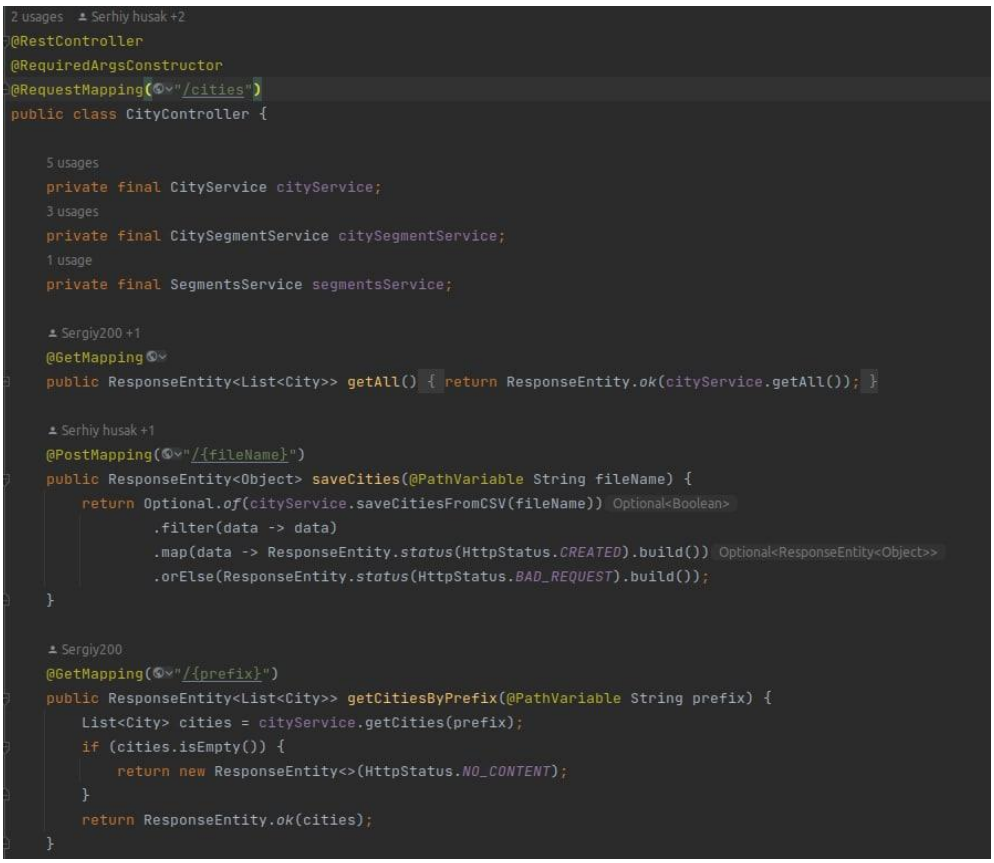
ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ВЕБ-СИСТЕМА ІДЕНТИФІКАЦІЇ МІСЦЬ ДЛЯ ПАРКУВАННЯ ЗА ДАНИМИ СЕРВІСУ OPENSTREETMAP

Виконав: студент гр. 2ІСТ-21м
_____ Гусак С.В.
«_01» _____12_____ 2022 р.

Керівник: к.т.н., доц. каф. САІТ
_____ Крижановський Є.М.
«_02_» _____12_____ 2022 р.

Нормоконтроль: к.т.н., доцент
_____ Жуков С. О.
«_02_» _____12_____ 2022 р.



```
2 usages  Serhiy husak +2
@RestController
@RequiredArgsConstructor
@RequestMapping("/cities")
public class CityController {

    5 usages
    private final CityService cityService;
    3 usages
    private final CitySegmentService citySegmentService;
    1 usage
    private final SegmentsService segmentsService;

    Sergiy200 +1
    @GetMapping
    public ResponseEntity<List<City>> getAll() { return ResponseEntity.ok(cityService.getAll()); }

    Serhiy husak +1
    @PostMapping("/{fileName}")
    public ResponseEntity<Object> saveCities(@PathVariable String fileName) {
        return Optional.of(cityService.saveCitiesFromCSV(fileName)) Optional<Boolean>
                .filter(data -> data)
                .map(data -> ResponseEntity.status(HttpStatus.CREATED).build()) Optional<ResponseEntity<Object>>
                .orElse(ResponseEntity.status(HttpStatus.BAD_REQUEST).build());
    }

    Sergiy200
    @GetMapping("/{prefix}")
    public ResponseEntity<List<City>> getCitiesByPrefix(@PathVariable String prefix) {
        List<City> cities = cityService.getCities(prefix);
        if (cities.isEmpty()) {
            return new ResponseEntity<>(HttpStatus.NO_CONTENT);
        }
        return ResponseEntity.ok(cities);
    }
```

Рисунок Г.1 – REST-API веб-сервісу

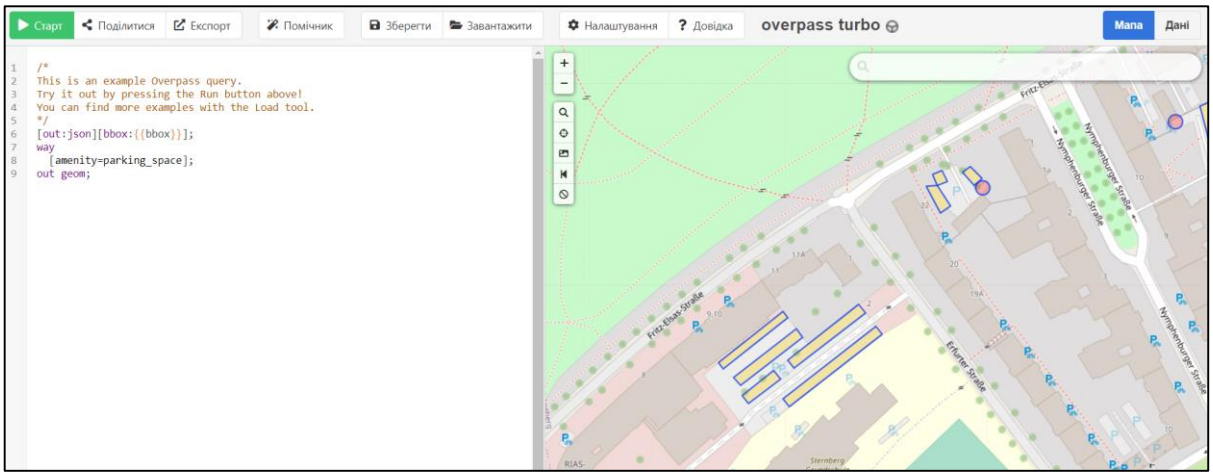


Рисунок Г.2 – Запит на OpenStreetMap

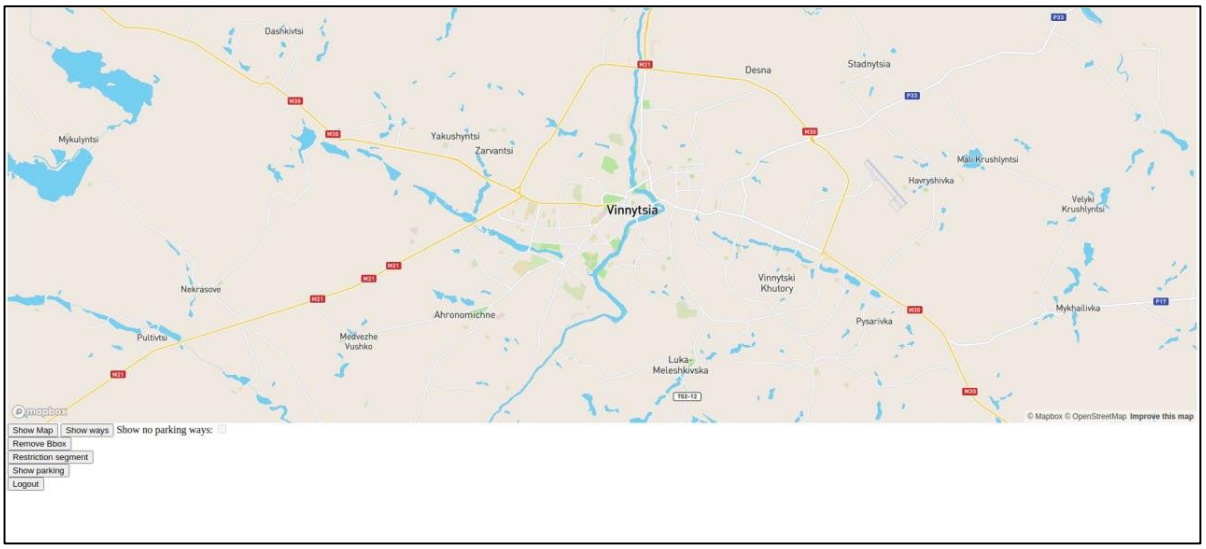


Рисунок Г.3 – Головна сторінка веб-системи

Рисунок Г.4 – Заборонені місця для паркування

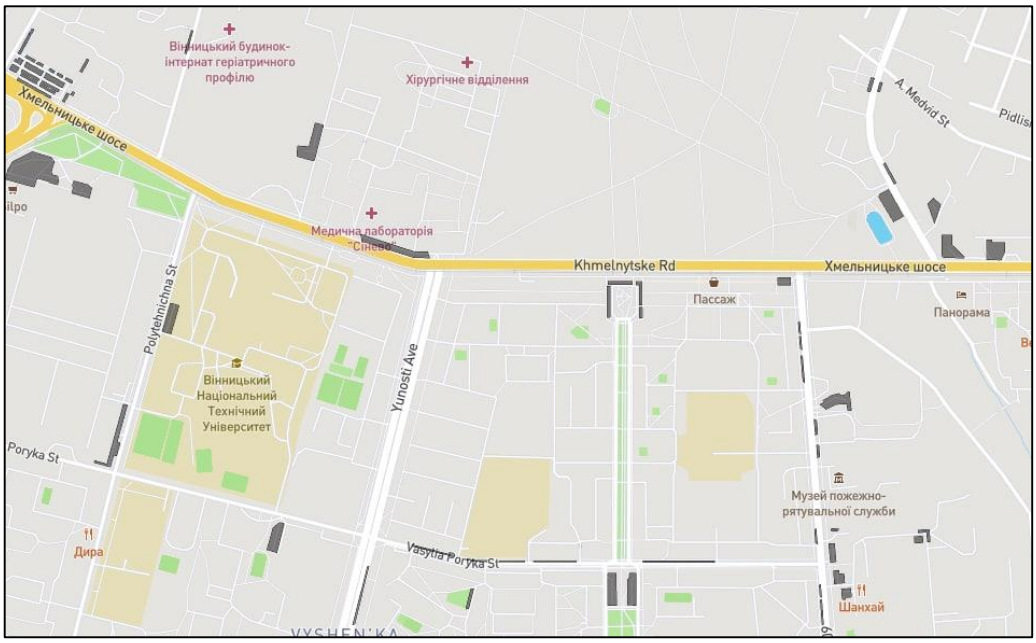


Рисунок Г.5 – Паркувальні місця

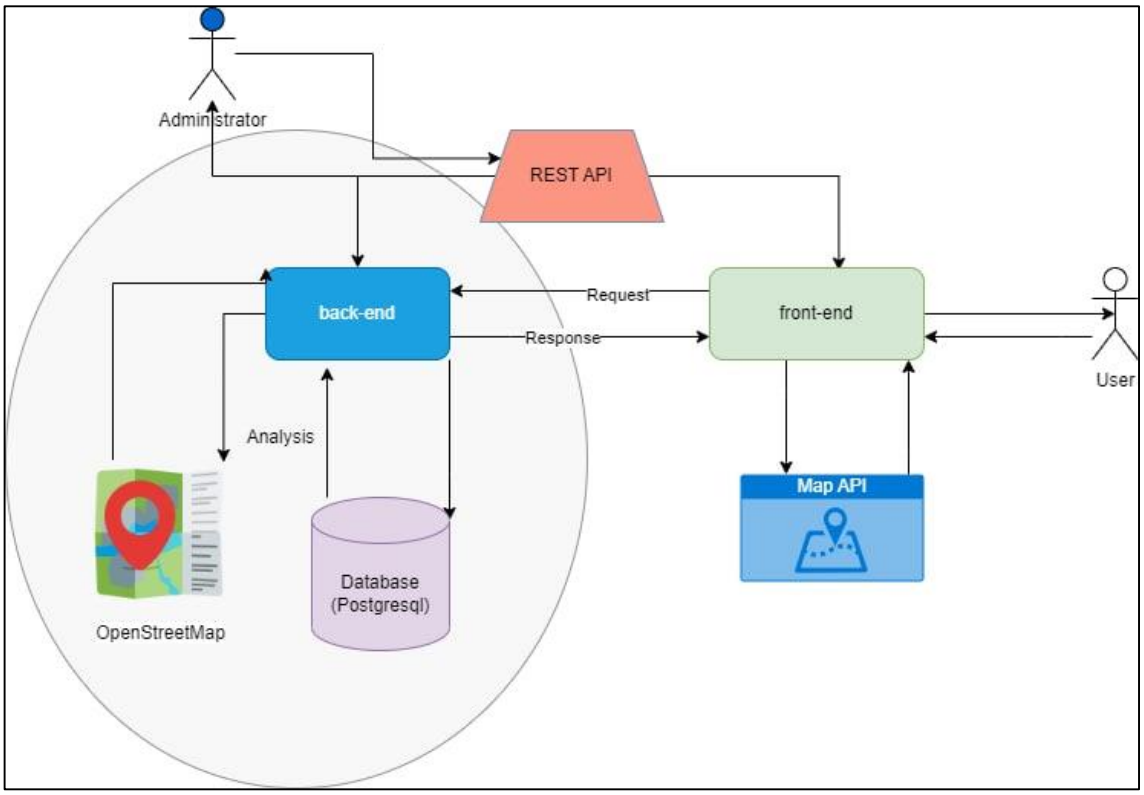


Рисунок Г.6 – Модель роботи веб-системи