НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

"На правах рукопису"
УДК 004.4

«До захисту допущено»
В.о. зав.кафедри
_____ Олександр КОВАЛЬ
“___” _____2022 р.

Магістерська дисертація

За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення
на тему: Інструментальні засоби формування енергоефективних маршрутів для електромобілів

Виконав: студент 2 курсу, групи ТВ-12мп
Бортнічук Нікіта Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент, к.е.н., Гусєва І.І. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2022

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
в.о. зав. кафедри
_____ Олександр КОВАЛЬ
(підпис)
«____» _____ 2022 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Бортнічук Нікіта Олександрович
(прізвище, ім'я, по батькові)

1. Тема дисертації: Інструментальні засоби формування енергоефективних маршрутів для електромобілів
Науковий керівник Гусєва Ірина Ігорівна, к.е.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "07" листопада 2022 року №4067-с
2. Строк подання студентом дисертації 05 грудня 2022 року
3. Вихідні дані до роботи Операційна система – MacOS , мова програмування – Kotlin, середовище розробки – Intellij IDEA, Android Studio. Теоретичне та практичне описання процесу створення системи формування енергоефективних маршрутів для електромобілів.
4. Перелік питань, які потрібно розробити
   - Проаналізувати існуючі рішення і дослідження визначення енерговитратності електромобіля;
   - Провести аналіз вхідних даних та визначити алгоритм обчислення витраченої енергії за заданим маршрутом
   - Розробити програмне забезпечення для застосування алгоритму і побудови відповідного маршруту.
5. Орієнтований перелік ілюстративного матеріалу архітектура системи, діаграма класів, результати роботи системи, порівняння з аналогами
6. Дата видачі завдання «01» жовтня 2021 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.10.2021	виконано
2	Дослідження предметної області	01.10.2021 - 01.11.2021	виконано
3	Постановка вимог до проєктування системи	01.11.2021 - 15.11.2021	виконано
4	Дослідження існуючих рішень	15.11.2021 - 01.12.2021	виконано
6	Розробка програмного продукту	05.03.2022 - 07.10.2022	виконано
7	Тестування	07.10.2022 - 11.11.2022	виконано
8	Захист програмного продукту	17.10-2022 – 21.10.2022	виконано
9	Підготовка магістерської дисертації	22.10.2022 – 20.11.2022	виконано
10	Передзахист	21.11.2022 - 25.11.2022	виконано
11	Захист	19.12.2022 – 23.12.2022	виконано

Студент _____ __Бортнічук Н. О.__
(підпис) (прізвище та ініціали)

Науковий керівник _____ ___Гусєва І. І._____
(підпис) (прізвище та ініціали)

Реферат

Актуальність. Показники забрудненості повітря у великих мегаполісах досягають критичних значень. Абсолютна більшість шкідливих викидів припадає на парникові гази, що спричиняють автомобілі з двигуном внутрішнього згорання. Одним із рішенням цієї проблеми може бути перехід на електричні транспортні засоби, які не спричиняють забруднення атмосфери при своїй роботі. Але вважати електромобілі абсолютно екологічним транспортом не можна. Адже для їх пересування використовується електроенергія, яка, в свою чергу, часто виробляється з невідновлюваних джерел, тим самим зумовлюючи викиди отруйних речовин. Тож вкрай необхідно розробити шляхи енергоефективного використання електричних транспортних засобів. Одним з таких способів є знаходження такого маршруту, на подолання якого витратиться найменша кількість електроенергії.

Метою роботи є розробка програмного забезпечення, яке за допомогою проаналізованих даних та застосування їх до алгоритму обчислення витраченої енергії формує найбільш енергоефективний маршрут та відображає його користувачу через мобільний застосунок.

Для досягнення мети необхідно виконати наступні задачі:

- Проаналізувати існуючі рішення і дослідження визначення енерговитратності електромобіля;
- Провести аналіз вхідних даних;
- Визначити алгоритм обчислення витраченої енергії за заданим маршрутом;
- Розробити програмне забезпечення для застосування алгоритму і побудови відповідного маршруту.

Об’єктом дослідження є визначення витрат енергії електромобілем.

Предметом дослідження є програмне забезпечення для формування енергоефективних маршрутів для електромобілів.

Методи дослідження. На етапі збору необхідних даних використовувалися емпіричні методи дослідження, а саме: спостереження, порівняння та вимірювання. При побудові алгоритму обчислення спочатку був проведений аналіз компонентів алгоритму, а потім синтез цих компонентів в ціле.

Практичне значення одержаних результатів полягає в відображенні розрахованого найбільш енергоефективного маршруту користувачу через мобільний застосунок, що дає можливість економити енерговитрати електромобіля.

Особистий внесок магістранта. При визначенні алгоритму обчислення енерговитратності електромобіля існуючі підходи і методи були застосовані для обчислення саме кількості електроенергії, яка необхідна для подолання заданого маршруту. Був здійснений пошук та аналіз даних і спроектовано програмне забезпечення, що реалізує даний алгоритм. Також розроблений мобільний застосунок для відображення результатів роботи системи.

Структура та обсяг дисертації. Магістерська дисертація складається зі вступу, шести розділів, висновків та 3 додатків. Робота містить посилання на 25 джерел, 36 ілюстрацій та 26 таблиць. Основна частина роботи викладена на 85 сторінках.

Ключові слова: маршрут, енергоефективність, квазі-статична модель, щільність повітря, коефіцієнт аеродинамічного опору, коефіцієнт опору кочення, координати, сервер, мобільний застосунок, геолокація.

Abstract

Relevance. Air pollution indicators in large megacities reach critical values. Most harmful emissions are greenhouse gases caused by cars with an internal combustion engine. One of the solutions to this problem can be the transition to electric vehicles that do not cause air pollution during their operation. But electric cars cannot be considered as an absolutely ecological transport. After all, electricity is used for their movement, which, in turn, is often produced from non-renewable sources, thereby causing the emission of toxic substances. Therefore, it is extremely necessary to develop ways of energy-efficient use of electric vehicles. One of these methods is to find a route that will consume the least amount of electricity.

The aim of the work is to develop software that, using the analyzed data and applying it to the algorithm for calculating the spent energy, forms the most energy-efficient route and displays it to the user through a mobile application.

To achieve the goal, it is necessary to complete the following points:

- Analyze existing solutions and research on determining the energy consumption of an electric vehicle;
- Analyze input data;
- Determine the algorithm for calculating the spent energy according to the given route;
- Develop software for applying the algorithm and building the appropriate route.

The object of the research is the construction of the most energy-efficient route for an electric car.

The subject of the research is to determine the energy consumption of an electric vehicle for a given route.

Research methods. At the stage of collecting the necessary data, empirical research methods were used, namely observation, comparison, and measurement. When building the calculation algorithm, the analysis of the algorithm components was first carried out, and then the synthesis of these components.

The practical value of the obtained results consists in displaying the calculated most energy-efficient route to the user through the mobile application, which makes it possible to save the energy consumption of the electric vehicle.

Personal contribution of the master's student. When determining the algorithm for determining the energy consumption of an electric vehicle, existing approaches and methods were adapted to calculate the exact amount of electricity required to cover a given route. The search and analysis of data was carried out, and the software implementing this algorithm was built. A mobile application has also been developed to display the results of the system.

The structure and scope of the dissertation. The master's thesis consists of an introduction, six chapters, conclusions and 2 appendices. The work contains references to 25 sources, 36 illustrations and 26 tables. The main part of the work is laid out on 85 pages.

Keywords. Route, energy efficiency, quasi-static model, air density, drag coefficient, rolling resistance coefficient, coordinates, server, mobile application, geolocation.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ПЗ – Програмне забезпечення;

КПП – Коробка перемикання передач;

API – Application Programming Interface, набір визначень підпрограм, протоколів взаємодії та засобів для створення програмного забезпечення;

KMM – Kotlin Multiplatform Mobile, інструментарій для побудови мобільних кросплатформенних застосунків;

NoSQL – No Structured Query Language, неструктурована база даних;

GPS – Global Positioning System, глобальна система позиціонування;

JSON – JavaScript Object Notation, текстовий формат обміну даними між комп'ютерами;

CSV – Comma-Separated Values, значення, розділені комою.

ВСТУП

Сучасний розвиток суспільства характеризується великою чисельністю населення, а отже, і зростанням економічних, соціальних та енергетичних потреб. Їх забезпечення зумовлює зростання кількості шкідливих викидів у атмосферу, виникнення парникового ефекту та зміни клімату в цілому. Показники забрудненості повітря у великих мегаполісах досягають критичних значень. Це особливо згубно впливає на здоров'я мешканців міст, а також на інших живих істот, які змушені мігрувати в пошуках чистішого середовища існування, що викликає розбалансованість екосистем. Одним з основних типів цих забруднень є вихлопні гази автомобілів з двигуном внутрішнього згоряння. За даними Управління екології та природних ресурсів Київської міської державної адміністрації, 78% забруднення повітря міста спричиняє автотранспорт [1].

Однією з альтернатив такому шкідливому виду транспорту є використання електромобілів, що останнім часом починають займати значну частку в автоіндустрії. Світові продажі електричного автотранспорту збільшились удвічі у 2021 році в порівнянні з попереднім роком і становили 8.3% від усіх продажів легковиків [2]. Таке зростання популярності пояснюється значними перевагами електромобілів над «паливними» авто, а саме:

- Екологічність. Відсутність вихлопів дозволяє ефективно знизити негативний вплив на навколишнє середовище і природу;
- Економічність. Електрика коштує дешевше інших видів палива, а відсутність складних механізмів, витратних матеріалів та запчастин виключає часті поломки і необхідність в заміні;
- Високий коефіцієнт корисної дії;
- Функція регенерації енергії під час зниження швидкості;
- Довговічність і простота в обслуговуванні.

Проте, незважаючи на всі переваги, електромобілі не є повністю екологічним транспортом, адже вони потребують досить великої кількості електроенергії, яка часто виробляється за допомогою невідновлювальних джерел, що супроводжується шкідливими викидами в атмосферу. Таким чином, масове використання електромобілів не вирішить глобальну проблему забруднення навколишнього середовища. Тому вкрай необхідно розробити шляхи енергоефективного використання електричних транспортних засобів, які допоможуть зменшити кількість використання енергії, тим самим уповільнять процес погіршення стану довкілля. Одним із способів економії є енергоефективне подолання маршруту від початкової точки до пункту призначення.

Для визначення найбільш енергоефективного маршруту, достатньо знайти той, на подолання якого витратиться найменша кількість електроенергії. Для цього у роботі досліджуються способи обчислення витрат енергії для прогнозованого маршруту. Основний аналіз проводиться на основі існуючих досліджень знаходження відстані, що залишилося подолати до моменту розрядки акумулятора (RDR – remaining driving range). Однією з представлених математичних моделей є квазі-статична модель, за допомогою якої визначається потужність електричного мотору в залежності від зовнішніх факторів, які можуть вплинути на опір руху електромобіля. При цьому враховується коефіцієнт корисної дії мотору, а також враховується процес регенерації енергії при зниженні швидкості.

Розроблене програмне забезпечення є комплексною системою, яка складається з серверної частини та мобільного застосунку. На сервері відбуваються основні обчислення та звернення до зовнішніх джерел даних (API) та локальної бази даних. Мобільний застосунок, в свою чергу, взаємодіє із сервером, надсилаючи та отримуючи необхідні дані, і відображає результати роботи системи користувачу. Більшість існуючих аналогічних систем фокусуються на подоланні маршруту якнайшвидшим шляхом, що часто не є найбільш енергоефективною опцією. Натомість, побудована система будує маршрут, в першу чергу враховуючи енерговитратність електромобіля.

Магістерська дисертація складається з шести розділів і висновків.

У першому розділі визначається мета дослідження, задачі, які необхідно виконати для її досягнення, а також призначення розробленої системи.

У другому розділі проводиться аналіз існуючих систем, визначаються їх переваги та недоліки.

У третьому розділі визначається апарат вирішення поставленої задачі, а саме описується алгоритм визначення енерговитратності електромобіля на визначеному маршруті.

У четвертому розділі описуються програмна реалізація розробленої системи, її архітектура, а також засоби розробки ПЗ.

У п’ятому розділі проводиться апробація результатів роботи системи, отримані результати порівнюються з існуючими засобами.

У шостому розділі описується стратегія реалізації стартап-проєкту для розробленої системи

У висновках зазначається ступінь розв’язання мети і задач роботи, а також викладаються результати роботи системи.

1 ПОСТАНОВКА ЗАДАЧІ ФОРМУВАННЯ ЕНЕРГОЕФЕКТИВНИХ МАРШРУТІВ ДЛЯ ЕЛЕКТРОМОБІЛІВ

1.1 Мета роботи та формулювання задачі

Мета: Створити систему, що за допомогою аналізу вхідних даних та застосування їх до алгоритму обчислення витраченої енергії, формує маршрут, який є найбільш енергоефективним та відображає його користувачу через мобільний застосунок.

Задачі, які потрібно розв’язати для досягнення мети:

- Провести аналіз існуючих рішень побудови маршрутів для автомобілів;
- Проаналізувати існуючі дослідження визначення енерговитратності електромобіля;
- Визначити алгоритм побудови енергоефективного маршруту;
- Проаналізувати дані специфікацій автомобілів, динамічні дані про погоду, затори, дані про автомобільні дороги (покриття, світлофори) та застосувати отримані дані в побудований алгоритм;
- Розробити систему передачі даних про маршрут на мобільний застосунок;
- Розробити застосунок, що отримує та відображає отримані дані.

1.2 Призначення системи

Розроблена система призначена для визначення найбільш енергоефективного маршруту від заданої точки початку шляху до пункту призначення для електромобілів за допомогою формули обчислення потужності мотору автомобіля та витраченої енергії із застосування карти ефективності електричного двигуна для

розгону та вільного руху відповідно. В результаті система визначає кількість витраченої енергії на можливих маршрутах, визначає найменш енерговитратний і виділяє його на карті місцевості.

Додатковим призначенням системи є визначення місцезнаходження зарядних станцій для електромобілів у разі, якщо кількість передбаченої витраченої електроенергії для найбільш енергоефективного маршруту перевищує поточний стан заряду акумулятора.

Користувачами даної системи можуть бути водії легкових електричних транспортних засобів або пасажири, що асистують водію.

1.3 Формулювання критеріїв набору даних

Для реалізації алгоритму визначення енерговитратності автомобіля необхідні наступні дані:

- Специфікації автомобіля;
- Дані про погоду, а саме: температура повітря, вологість, тиск;
- Дані про затори в реальному часі;
- Тип покриття автомобільної дороги (асфальт, бруківка тощо);
- Місцеположення світлофорів;
- Географічні координати маршруту;
- Початкове географічне положення користувача та кінцева точка прибуття;

База даних електромобілів повинна бути достатньо обширною і оновленою для того, щоб максимальна кількість користувачів могла використовувати дану систему.

Для визначення місцеположення зарядних станцій, необхідно отримати від користувача поточний стан заряду акумулятора і набір координат, що використовуються для побудови маршруту.

1.4 Програмна реалізація системи

Система повинна складатись з мобільного застосунку та серверної частини.

Мобільний застосунок клієнта (водія електричного транспортного засобу) – має зберігати дані про автомобіль користувача; визначати його місцезнаходження та кінцеву точку прибуття; відображати маршрут, що був побудований системою. При цьому необхідно розробити такий застосунок, який можна було б запускати на різних мобільних операційних системах для покриття більшої кількості користувачів.

Серверна частина – повинна бути такою, яка має можливість отримувати дані з відповідних джерел, зберігати інформацію про транспортний засіб користувача у базі даних, отримувати дані місцезнаходження користувача та кінцевої точки прибуття, визначати енерговитратність кожного з отриманих маршрутів, формувати найбільш енергоефективний маршрут та надсилати відповідні дані на мобільний застосунок.

Висновки до розділу 1

В результаті дослідження проблеми формування енергоефективних маршрутів для електромобілів визначено мету та задачі.

Призначенням системи визначено формування енергоефективного маршруту.

Визначено набір даних, необхідний для визначення енерговитратності електромобіля та визначення місцезнаходження зарядних станцій впродовж маршруту.

Описано загальну архітектуру системи, що необхідно розробити для вирішення поставлених задач. Обрано клієнт-серверний підхід у побудові архітектури. У якості клієнта виступає мобільний застосунок для системи Android, при цьому передбачено подальшу розробку для інших мобільних систем.

2 АНАЛІЗ ІСНУЮЧИХ ДОСЛІДЖЕНЬ І СИСТЕМ

2.1 Опис алгоритму визначення спожитої енергії електромобіля

В основу побудови алгоритму вкладено використання квазістатичної математичної моделі симуляції роботи шасі, трансмісії та електричного двигуна (рис. 2.1) [3]. Квазістатичний підхід є обчислювально ефективним, оскільки припускає, що транспортний засіб рухається точно з передбаченою швидкістю. Це припущення зручне, тому що не потрібно розв’язувати диференціальні рівняння, а вимоги до потужності можна легко обчислити, розв’язуючи алгебраїчні рівняння [4].

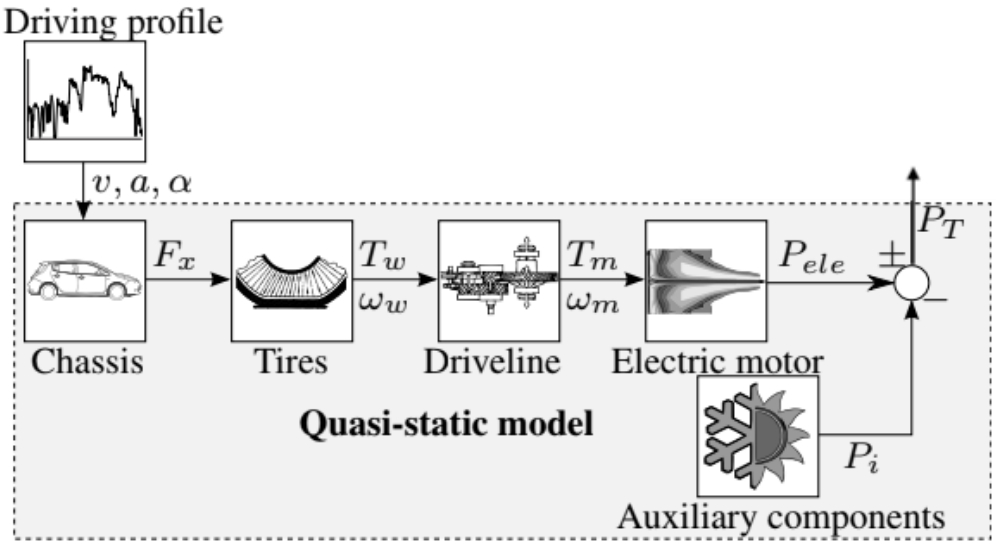


Рисунок 2.1 – Квазістатична модель руху електромобіля

Як видно з рисунку 2.1, квазістатична модель являє собою поступовий алгоритм визначення загальної потужності руху електромобіля. Спочатку обчислюється сила, що діє на автомобіль при його русі, потім враховуються параметри коліс і трансмісії для визначення електричної потужності мотору і застосовується карта ефективності електродвигуна. Також враховуються витрати на додаткові компоненти, що необхідні для роботи автомобіля. В результаті ми

отримуємо електричну потужність, що необхідна для визначення енерговитратності електромобіля при його русі.

Визначення загальної сили, необхідної для руху автомобіля

Електромобіль складається з багатьох компонентів, рух яких, для спрощення, можна вважати рівномірним. Таким чином, електричний транспортний засіб можна представити як одну зосереджену масу. Тобто, для того, щоб обчислити загальну силу, необхідну для руху транспортного засобу, достатньо підсумувати всі сили, що діють на об'єкт, в даному випадку автомобіль (рис. 2.2).

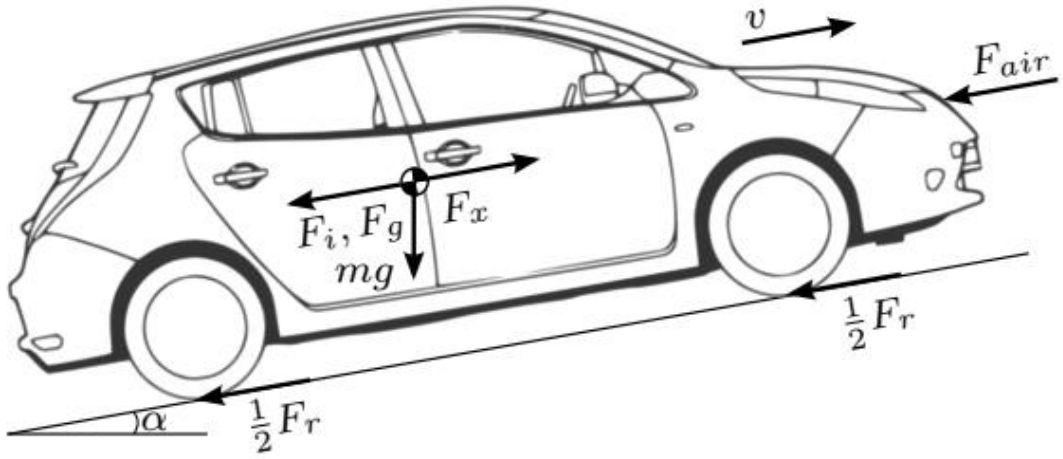


Рисунок 2.2 – Сили, що діють на електромобіль під час руху

Варто зауважити, що в такому разі всі сили є векторними величинами. Тоді необхідна для руху транспортного засобу сила F_x задається формулою:

$$F_x = F_{air} + F_g + F_r + F_i \tag{2.1}$$

де F_{air} – аеродинамічна сила лобового опору;

F_g – сила підйому нагору;

F_r – сила опору кочення;

F_i – це сила, необхідна для прискорення/уповільнення автомобіля.

У механіці рідин та газів аеродинамічна сила лобового опору – це сила, яка діє на тіло занурене у повітря (або інший газ), і виникає внаслідок відносного руху між тілом і газом [5]. Аеродинамічна сила F_{air} обчислюється за формулою:

$$F_{air} = \frac{1}{2}\rho_{air} c_w A v^2 \tag{2.2}$$

де ρ_{air} – густина повітря, кг/м^3;

c_w – коефіцієнт лобового опору;

А – площа лобової частини автомобіля, м^2;

v – швидкість руху, м/с.

В свою чергу густина повітря обчислювалась з урахуванням його вологості за законом Дальтона для парціальних тисків, що говорить: «Тиск газової суміші дорівнює сумі парціальних тисків кожного газу». В результаті можна розширити закон Дальтона на густину газових сумішей, таких як суміш повітря та пари (вологе повітря):

$$\rho_{air} = \frac{P_d}{R_d T} + \frac{P_v}{R_v T} \tag{2.3}$$

де P_d – парціальний тиск сухого повітря, Па;

P_v – парціальний тиск водяної пари, Па;

R_d – питома газова стала сухого повітря (287,05 $\frac{\text{Дж}}{\text{кг·К}}$);

R_v – питома газова константа водяної пари (461,495 $\frac{\text{Дж}}{\text{кг·К}}$);

Т – температура повітря, К.

Для підрахунку парціального тиску водяної пари було використано наступний поліном [6]:

$$Pv = RH \cdot E_s(T) \tag{2.4}$$

де RH – відносна вологість повітря, %;

$E_s(T)$ – тиск насиченої пари, мБар.

$$Es(T) = \frac{e_{s0}}{[p(T)]^8} \tag{2.5}$$

$$p(T) = c_0 + T(c_1 + T(c_2 + T(c_3 + T(c_4 + T(c_5 + T(c_6 + T(c_7 + T(c_8 + Tc_9)))))))))$$

де

e_{s0} = 6.1078

c_0 = 0.99999683

c_1 = -0.90826951 × 10-2

c_2 = 0.78736169 × 10-4

c_3 = -0.61117958 × 10-6

c_4 = 0.43884187E-8 × 10-8

c_5 = -0.29883885 × 10-10

c_6 = 0.21874425 × 10-12

c_7 = -0.17892321 × 10-14

c_8 = 0.11112018 × 10-16

c_9 = -0.30994571 × 10-19

Сила підйому нагору F_g обчислюється за наступною формулою:

$$F_g = mgsin(\alpha) \tag{2.6}$$

де m – маса автомобіля, кг;

g – прискорення вільного падіння, м/с^2;

$sin(\alpha)$ – синус кута нахилу дороги.

Як відомо, синус кута – це відношення протилежної сторони прямокутного трикутника до гіпотенузи. Відповідно, для того, щоб змоделювати цей трикутник необхідно для визначених точок локації знайти їх висоту над рівнем моря. Оскільки відстань між двома точками незначна, можна прийняти, що відстань між цими двома точками є прямою лінією (рис. 2.3).

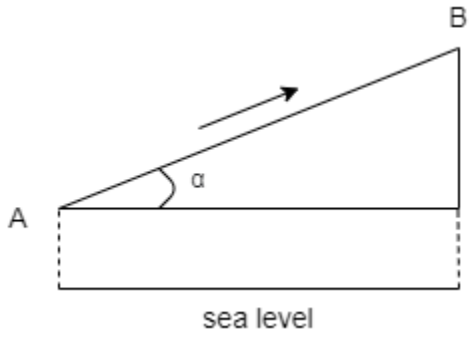


Рисунок 2.3 – ілюстрація руху вгору

Тому, синус кута нахилу дороги визначаємо, за формулою:

$$sin(\alpha) = \frac{E_B - E_A}{d} \tag{2.7}$$

де

E_B, E_A – висота над рівнем моря відповідних точок, м;

d – відстань між точками A і B, м.

Силу опору кочення знаходять за формулою:

$$F_r = mgK_r \tag{3.8}$$

де K_r – коефіцієнт опору кочення.

Коефіцієнт опору кочення, можна визначити з урахуванням типу покриття дороги (табл. 2.1) [7]:

Таблиця 2.1 — Коефіцієнт опору кочення

Тип покриття дороги	Коефіцієнт опору кочення
Асфальт	0,01 – 0,02
Гравій	0,02 – 0,025
Бруківка	0,025 – 0,05
Грунт	0,025 – 0,035
Пісок	0,1 – 0,3

Сила, необхідна для прискорення/уповільнення автомобіля, обчислюється за формулою:

$$F_i = ma \tag{2.9}$$

де

a – прискорення автомобіля, м/с^2.

Визначення потужності

Знайшовши загальну силу, можна знайти необхідну для руху транспортного засобу механічну потужність. Вона вираховується домноживши силу на швидкість електромобіля:

$$P_{mec} = F_x v \tag{2.10}$$

Отже загальна формула представлена у наступному вигляді:

$$P_{mec} = \frac{1}{2}\rho_{air} c_w A v^3 + mgsin(\alpha)v + mgK_r v + mav \tag{2.11}$$

Ця модель досить точно розраховує механічну потужність транспортного засобу з доволі низькою обчислювальною вартістю.

Як було запропоновано [8], взаємозв'язок між механічною потужністю і потужністю електричного мотору може бути визначений з певним ступенем точності, використовуючи стаціонарну карту ефективності електродвигуна (рис. 2.4) (Додаток Б) як функцію швидкості обертання ротора і крутного моменту:

$$P_{ele} = \frac{P_{mec}}{\eta_m(\omega_m, T_m)}, P_{mec} > 0 \quad (2.12)$$

де

η_m – ефективність електричного мотору;

ω_m – швидкість обертання ротора, $с^{-1}$;

T_m – крутний момент, Н・м;

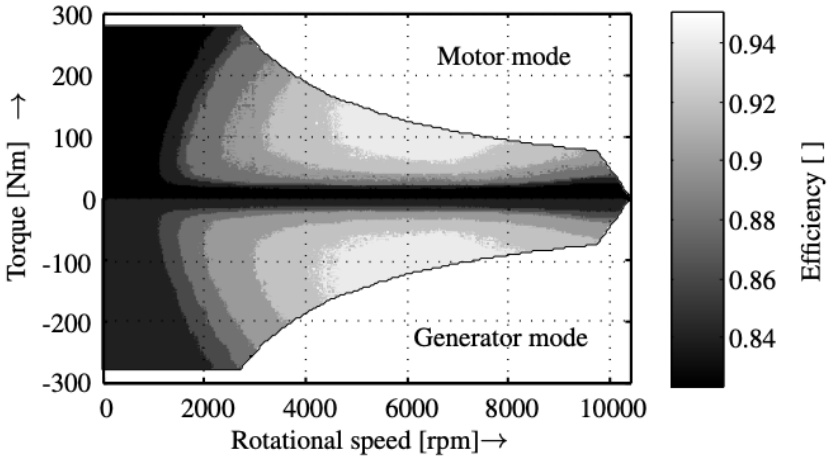


Рисунок 2.4 – Карта ефективності електродвигуна

Швидкість обертання ротора обчислюється за формулою:

$$\omega_m = \frac{v i_d}{r_{tire}} \quad (2.13)$$

В свою чергу крутний момент можна визначити за формулою:

$$T_m = \frac{F_x r_{tire}}{i_d} \quad (2.14)$$

де

i_d - передавальне число передачі;

r_{tire} – радіус колеса, м.

Оскільки в роботі в основному розглядаються легкові автомобілі з найбільш поширеною 5-ступінчатою КПП, для визначення передавального числа передачі була використана наступна таблиця (табл. 2.2) [9]:

Таблиця 2.2 – Діапазони передавального числа для 5-ступінчатої КПП

Передача	Діапазон передавального числа
1	3.0 – 4.0
2	2.0 – 2.9
3	1.2 – 1.9
4	0.9 – 1.2
5	0.7 – 0.9
Задня	3.0 – 4.0

Для спрощення, в роботі вважалось, що загалом для розгону потрібні перші три передачі, відповідно для вільного руху – четверта та п’ята. Оскільки досить важко передбачити, яка саме передача використовувалась в конкретний момент, були взяті середні значення передавального руху для відповідних типів руху. Отже, при розрахунках використовувалась наступна таблиця (табл. 2.3):

Таблиця 2.3 – Передавальне число для різних типів руху

Тип руху	Передавальне число
Розгін	2.1
Вільний рух	0.9

Важливою особливістю сучасних електромобілів є те, що певна кількість кінетичної та потенційної енергії можна відновити засобами системи рекуперативного гальмування. Під час гальмування при маневрах електродвигун працює як генератор, забезпечуючи таким чином додатковий гальмівний момент для коліс. Відновлену енергію потім можна використовувати для постачання електроенергії до трансмісії або допоміжних аксесуарів. Кількість гальмівного моменту залежить від стратегії роботи гальмівної системи. Стратегія роботи оптимізує розподіл гальмівного моменту між механічними та рекуперативними гальмами таким чином, щоб була згенерована максимальна електрична потужність [10].

Згенерована електроенергія обчислюється за допомогою формули:

$$P_{ele} = P_{mec}\eta_m(\omega_m, -T_m)k_{v_x}, P_{mec} > 0 \quad (2.15)$$

де

k_{v_x} – параметр обмеження використання двигуна.

Оскільки генерована потужність залежить від ω_m, було б складним завданням подавати живлення на шину живлення на низьких швидкостях. Через це параметр k_{v_x} використовується для обмеження використання електродвигуна в генераторному режимі за формулою (2.16) так, що механічні гальма застосовуються на дуже низьких швидкостях, а на високих швидкостях транспортний засіб гальмується здебільшого електричним двигуном:

$$k_{v_x} = \begin{cases} 0 & , \ v_x \leq 3.5\ m/s \\ \frac{v_x - 3.5}{5}, & 3.5 < v_x < 8\ m/s \\ 0.9 & , \ v_x \geq 8\ m/s \end{cases} \quad (2.16)$$

Карта ефективності η_m зазвичай добре визначена лише для рухового режиму (верхній квадрант рисунку 2.4). Щоб розширити карту до режиму генератора, втрати потужності визначаються як:

$$\eta_m(\omega_m, -T_m) = 2 - \frac{1}{\eta_m(\omega_m, T_m)} \quad (2.17)$$

Визначення енерговитратності на маршруті

Для визначення енерговитратності на заданому маршруті, потрібно знайдену електричну потужність помножити на час, на якій вона була застосована і просумувати ці значення:

$$A_{route} = \sum_{i=1}^{n} P_{ele_i} * t_i \quad (2.18)$$

де A_{route} – загальна енерговитратність на маршруті;

n – кількість ділянок на дорозі;

P_{ele_i} – потужність необхідна для руху на конкретній ділянці;

t_i – час, за який ділянка може бути подолана

Враховуючи специфіку електричного мотору, а саме можливість переходити в режим регенерації енергії при зниженні швидкості, за формулою 3.18 визначаємо також кількість згенерованої енергії.

В результаті загальна формула для визначення кількості енергії на маршруті виглядає наступним чином:

$$A = A_{route} - A_{gen} \tag{2.19}$$

де

A_{gen} – кількість згенерованої енергії

2.2 Мобільний застосунок Google Maps

Google Maps – це веб-картографічна платформа, яку пропонує корпорація Google. Вона надає супутникові зображення, аерофотозйомку, карти вулиць, інтерактивні панорамні види вулиць (Street View), стан дорожнього руху в реальному часі та планування маршруту для подорожей пішки, на автомобілі, велосипеді та громадським транспортом. Карти Google Maps працюють як на комп'ютерах, так і на мобільних пристроях (телефонах і планшетах), але саме на смартфонах її максимальна продуктивність досягається завдяки простоті пошуку точки пересадки та можливості спостерігати за місцем розташування самого користувача в режимі реального часу.

Однією з головних особливостей Google Maps є можливість побудови маршруту від початкової точки до пункту призначення. Початок маршруту може бути визначений декількома способами, серед них:

- GPS-положення пристрою користувача
- Проставленням мітки на карті;

- Введенням конкретної адреси в спеціальне поле для вводу.

Користувач має можливість обрати вид транспорту та за наданими початковою та кінцевою точкою побудувати найбільш оптимальний маршрут разом з декількома альтернативними варіантами.

Google Maps розраховує маршрут на основі багатьох факторів для конкретної дороги. Алгоритм враховує обмеження швидкості, тип дороги (основна/другорядна дорога, місцева дорога, шосе/автостради), кількість світлофорів, тип дорожнього покриття (добре чи погане). Це скорочує час для відносно «хороших» доріг із великими обмеженнями швидкості і меншою кількістю світлофорів; і збільшує час для «поганих» доріг з більшою кількістю світлофорів і меншими обмеженнями швидкості. Він використовує датчики тепла, а також інші сенсорні пристрої, включаючи телефони, які мають пасажири на певній дорозі в певний час, щоб показати дорожній рух і затори. За кожен із цих факторів нараховуються штрафні бали. (Більша кількість розподілених штрафних балів = Більша тривалість) [11].

В результаті будується маршрут, що є найшвидшим. Він виділяється кольором. Крім цього для даного маршруту прораховується час, який передбачений на проходження маршруту, а також відстань у кілометрах. Додатково визначаються два альтернативних маршрути, які користувач може обрати за бажанням (рис. 2.5). При цьому упродовж маршруту фігурують різні кольори. Вони означають поточну ситуацію заторів на дорогах:

- Синій – дорога вільна від заторів;
- Помаранчевий – середня рівень затримки у заторі;
- Червоний – суттєві затримки у заторі. Чим темніше червоний, тим нижча швидкість руху на дорозі.

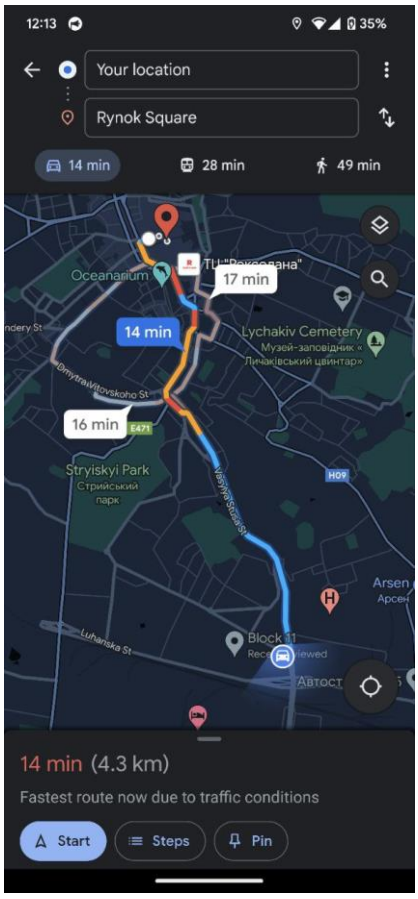


Рисунок 2.5 – Приклад роботи мобільного застосунку Google Maps

Переваги мобільного застосунку Google Maps у контексті планування маршрутів:

- Визначення точок початку та кінця маршруту за допомогою геолокації, введення даних локації, визначення локації на мапі;
- Можливість додавання проміжних точок маршруту;
- Визначення альтернативних шляхів;
- Позначення ділянок з уповільненим рухом (затори);
- Голосові та візуальні вказівки водію під час подолання маршруту;
- Визначення найшвидшого способу проходження маршруту;

2.3 Мобільний застосунок Waze

Waze є дочірньою компанією Google, яка надає супутникове навігаційне програмне забезпечення для смартфонів та інших комп'ютерів, які підтримують глобальну систему позиціонування (GPS). На додаток до навігації від початкової точки до пункту призначення, він включає інформацію про час у дорозі та маршрут, надані користувачем, під час завантаження інформації, що залежить від місця розташування, через стільникову мережу. Waze описує свою програму як таку, що розвивається спільнотою, яку можна безкоштовно завантажити та використовувати [12]. Waze на відміну від Google Maps має кілька додаткових корисних функцій, таких як:

- Відкладене планування маршруту на основі попередніх даних про завантаженість доріг у даний час;
- Можливість інтерактивної взаємодії з іншими користувачами під час поїздки;
- Попередження про об'єкти контролю дотримання правил дорожнього руху, таких як: пост поліції, камери контролю швидкості тощо.
- Можливість додавання міток дорожньо-транспортних пригод, автомобілів на узбіччі тощо.

Проте, Waze не визначає ділянки дороги, на яких є затримки через затори (рис. 2.6).

Натомість Waze є такою собі соціальною мережею для водіїв у реальному часі. Водії можуть бачити місцезнаходження один одного, якщо вони знаходяться поблизу, а також взаємодіяти, надсилаючи повідомлення і реакції.

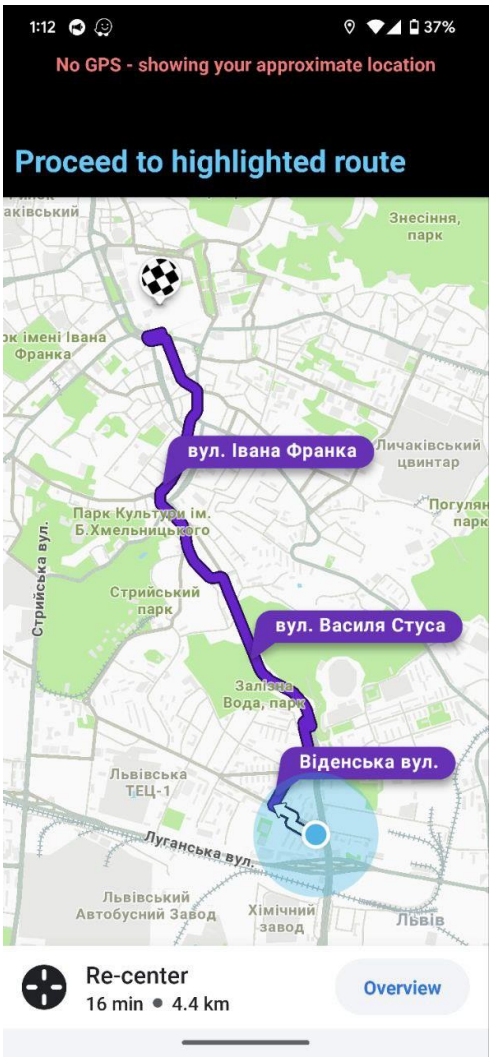


Рисунок 2.6 – Приклад роботи мобільного застосунку Waze

2.4 Недоліки існуючих системи

Вищезазначені застосунки є дійсно комплексними повноцінними системами з безліччю корисних особливостей та функцій. Проте, в контексті прокладання маршрутів від точки А до точки Б є деякі недоліки.

По-перше, вираховуючи оптимальний маршрут, вони спираються в першу чергу на час, що потрібен на подолання маршруту. Це, безперечно, критично важливий параметр для сучасної людини. Проте, з точки зору економії енергії, часто найшвидший маршрут не є найбільш енергоефективним. Цей фактор ніяк не сприяє збереженню довкілля і зниженню його забруднення. Навпаки, масовість

застосування цих застосунків у контексті побудови маршруту саме за таким чинником може спричинити погіршення стану навколишнього середовища. І наразі за допомогою згаданих систем неможливо визначити найбільш енергоефективний маршрут.

По-друге, стрімке зростання попиту на електромобілі не відображено в вищезгаданих застосунках. Ці види транспорту, безперечно є більш екологічними, а отже, необхідно популяризувати їх використання.

Виходячи з цього, виникає необхідність розробити систему, що може надати відсутні функції, а саме побудова маршруту, виходячи з енергоефективності. Для цього потрібно побудувати алгоритм, що обчислює витрати електроенергії упродовж проходження маршруту.

Використання електромобілів є більш екологічним способом пересування, ніж інші авто, отже, система повинна враховувати специфіку електричного мотору.

Реалізація алгоритму повинна бути представлена у вигляді системи, що має мобільний застосунок у якості взаємодії з користувачем.

Висновки до розділу 2

Проаналізовано моделі та алгоритми для обчислення та визначення найбільш енергоефективного маршруту. Даний алгоритм є доволі точним з низькою обчислювальною вартістю.

Окремо враховані специфічні функції електричного мотору, а саме регенерація енергії при зниженні швидкості. Це є однією з найбільш важливих можливостей електричного мотору в контексті економії витрат.

Проаналізовано існуючі системи, що вирішують проблему побудови маршрутів. Виявлено, що існують досить зручні і багатофункціональні застосунки, що не тільки мають можливість будувати маршрут, а й цілий комплекс інших функцій.

Виявлено недоліки у критеріях визначення найбільш оптимального маршруту, а саме відсутність формування шляхів за кількістю витраченої енергії задля її економії.

Вирішено розробити систему, що має необхідний функціонал для усунення виявлених недоліків. При цьому, необхідно зберегти зручність користування системою, аби заохотити користувачів допомагати запобіганню забруднення навколишнього середовища.

3 ОПИС ЗАСОБІВ РОЗРОБКИ

3.1 Серверна частина

Серверна частина розроблена мовою програмування Kotlin за допомогою набору фреймворків для побудови веб застосунків Spring. В якості бази даних була використана NoSQL база даних MongoDB.

Мова програмування Kotlin

Kotlin — це статично типізована мова програмування, яка працює за допомогою віртуальної машини Java, а також може бути скомпільована у вихідний код JavaScript. Автори ставили за мету створити мову більш лаконічний і типобезпечніший, ніж Java, і простіший, ніж Scala. Наслідком спрощення порівняно зі Scala стали також швидша компіляція та краща підтримка мови у IDE. Мова повністю сумісна з Java, що дозволяє Java-розробникам поступово перейти до її використання [13].

Синтаксис мови переважно комбінує спадок із двох мовних гілок: C/C++/Java та ML (за словами творців, через Scala). З найбільш характерних елементів, від першої гілки успадковано блоки коду, обрамлені фігурними дужками; а від другої – постфіксна вказівка типів змінних і параметрів (спочатку ідентифікатор, потім роздільник – двокрапка, а потім тип) і ключові слова «fun» і «val». Крапка з комою як роздільник операторів необов'язкова (як у Scala, Groovy та JavaScript); у більшості випадків переведення рядка достатньо, щоб компілятор зрозумів, що вираз закінчився. Крім об'єктно-орієнтованого підходу Kotlin також підтримує процедурний стиль з використанням функцій. Як і C, C++ і D, точка входу в програму — функція main, що приймає масив параметрів командного рядка. Програми Kotlin також підтримують perl- і shell-стиль інтерполяції рядків (змінні, включені в рядок, замінюються на свій вміст). Також підтримується виведення типів.

Веб-фреймворк Spring

Spring Framework забезпечує комплексну модель програмування та конфігурації для сучасних корпоративних програм на основі Java на будь-якій платформі розгортання [14].

Ключовим елементом Spring є інфраструктурна підтримка на рівні додатків: Spring зосереджується на «сантехніці» корпоративних додатків, щоб команди могли зосередитися на бізнес-логіці рівня застосунків без непотрібних зв'язків із певними середовищами розгортання.

В розробленій програмі використовувалися фреймворки Spring Boot, а також утиліти Spring для бази даних MongoDB, а також для веб-застосунку.

Spring Boot дозволяє легко створювати автономні застосунки на основі Spring продуктивного класу, які можна «просто запускати». Основні переваги:

- Створення автономні програми Spring
- Використання Tomcat, Jetty або Undertow безпосередньо (не потрібно розгортати WAR-файли)
- Надання впевнених «початкових» залежностей, щоб спростити конфігурацію збірки
- Автоматичне налаштовування Spring і бібліотеки сторонніх розробників, коли це можливо
- Представлення готових до виробництва функції, такі як показники, перевірки працездатності та зовнішня конфігурація
- Абсолютна відсутність генерації коду та жодних вимог щодо конфігурації XML

Spring Boot дотримується багаторівневої архітектури, у якій кожен рівень взаємодіє з рівнем, що знаходиться безпосередньо під ним або над ним (ієрархічна структура) (рис. 3.1).

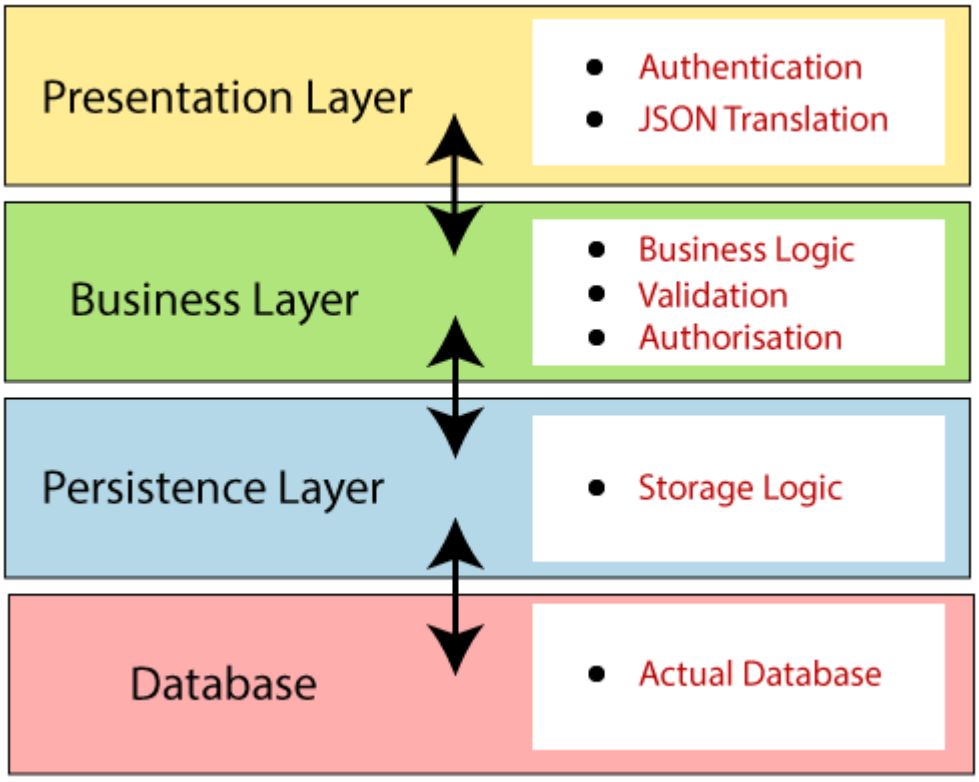


Рисунок 3.1 – Архітектура Spring Boot застосунку

Як видно з рисунку 3.1, у Spring Boot застосунку є чотири рівні:

- Презентаційний: презентаційний рівень обробляє HTTP-запити, перетворює параметр JSON в об’єкт, а також автентифікує запит і передає його на бізнес-рівень.
- Бізнес-рівень: бізнес-рівень обробляє всю бізнес-логіку. Він складається з класів обслуговування та використовує послуги, що надаються рівнями доступу до даних. Він також виконує авторизацію та перевірку.
- Рівень збереження: рівень збереження містить усю логіку зберігання та перекладає бізнес-об’єкти з рядків бази даних у них.
- Рівень бази даних: на рівні бази даних виконуються операції CRUD (створення, отримання, оновлення, видалення).

База даних MongoDB

MongoDB — це кросплатформна документоорієнтована програма бази даних із доступним вихідним кодом. Класифікована як NoSQL база даних, MongoDB

використовує JSON-подібні документи з додатковими схемами. MongoDB розроблено компанією MongoDB Inc. і ліцензовано згідно з загальнодоступною ліцензією на стороні сервера (SSPL), яка в кількох дистрибутивах вважається не безкоштовною.

MongoDB підтримує пошук за полями, діапазонами та регулярними виразами. Запити можуть повертати певні поля документів, а також містити визначені користувачем функції JavaScript. Запити також можна налаштувати для повернення випадкової вибірки результатів заданого розміру. Поля в документі MongoDB можна індексувати за допомогою первинних і вторинних індексів або індексу.

MongoDB забезпечує високу доступність із наборами реплік. Набір реплік складається з двох або більше копій даних. Кожен член набору реплік може виступати в ролі первинної або вторинної репліки в будь-який час. За замовчуванням усі записи та читання виконуються на основній репліці. Вторинні репліки зберігають копію даних первинної за допомогою вбудованої реплікації. Коли основна репліка виходить з ладу, набір реплік автоматично проводить процес вибору, щоб визначити, яка вторинна має стати основною. Вторинні можуть за бажанням обслуговувати операції читання, але ці дані є узгодженими лише за умовчанням.

Якщо репліковане розгортання MongoDB має лише одного вторинного члена, до набору необхідно додати окремий демон під назвою арбітр. Він несе єдину відповідальність, яка полягає в тому, щоб вирішити питання про вибори нових передвиборів. Як наслідок, ідеалізоване розподілене розгортання MongoDB потребує принаймні трьох окремих серверів, навіть у випадку лише одного основного та одного вторинного.

MongoDB масштабується горизонтально за допомогою сегментування. Користувач вибирає шард-ключ, який визначає, як будуть розподілятися дані в колекції. Дані розбиваються на діапазони (на основі ключа сегмента) і розподіляються між кількома сегментами. (Шард — це майстер з однією або

декількома репліками.) Крім того, ключ сегмента можна хешувати, щоб зіставити з сегментом, що забезпечує рівномірний розподіл даних. MongoDB може працювати на кількох серверах, балансуючи навантаження або дублюючи дані, щоб підтримувати роботу системи в разі апаратного збою [15].

3.2 Мобільний застосунок

Мобільний застосунок був побудований для операційної системи Android, проте була застосована платформа Kotlin Multiplatform Mobile, що дозволяє будувати нативні застосунки під операційні системи не тільки Android, а IOS та інші. Частина, що відповідає за користувацький інтерфейс Android застосунку побудована за допомогою бібліотеки ComposeUI.

Kotlin Multiplatform Mobile

Kotlin Multiplatform Mobile — це SDK для розробки програм для iOS та Android. Він пропонує всі сукупні переваги створення кросплатформних і нативних програм.

Його виробництво довіряють багато провідних світових компаній, зокрема Philips, Netflix, Leroy Merlin і VMWare.

За допомогою Kotlin Multiplatform можна створювати різні мультиплатформенні проекти для багатьох платформ, включаючи веб, робочий стіл та інші власні платформи.

Програми Kotlin працюватимуть на різних операційних системах, таких як macOS, Windows, Linux, Android, iOS, watchOS та інших [16].

Kotlin Multiplatform – це не кросплатформний фреймворк, а скоріше додатковий інструмент спільного використання коду, який можна використовувати як у додатках, створених із нуля, так і в існуючих власних додатках. Це дає розробникам безпрецедентну гнучкість щодо того, чим ділитися, а чим ні.

Рекомендується розділяти бізнес-логіку мобільних додатків, включаючи мережеву логіку, базу даних, бізнес-логіку та презентаційну логіку, яка не залежить від конкретної платформи та може бути легко повторно використана. З іншої сторони, не рекомендується ділитися компонентами інтерфейсу користувача, які значною мірою залежать від можливостей нативних платформ (рис. 3.2).

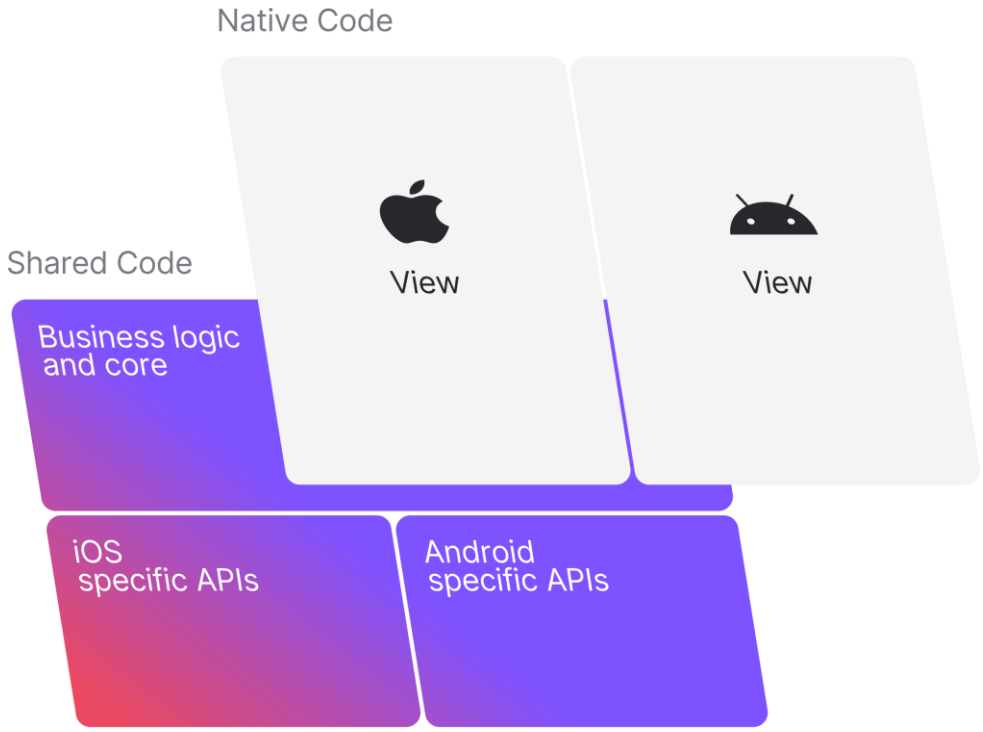


Рисунок 3.2 – Архітектура застосунку побудований на KMM

Окрім KMM існує декілька інших підходів для побудови кросплатформенних застосунків, найвідоміші з них це: Xamarin, React Native та Flutter.

Проте, Koltin Mulitplatform Mobile має значну кількість переваг, порівняно з конкурентами (рис. 3.3).

Головним недоліком KMM є те, що цей інструмент досі знаходиться в розробці і доступний тільки в бета-версії. Проте, вже зараз можна у повній мірі ефективно використовувати представлений інструментарій.

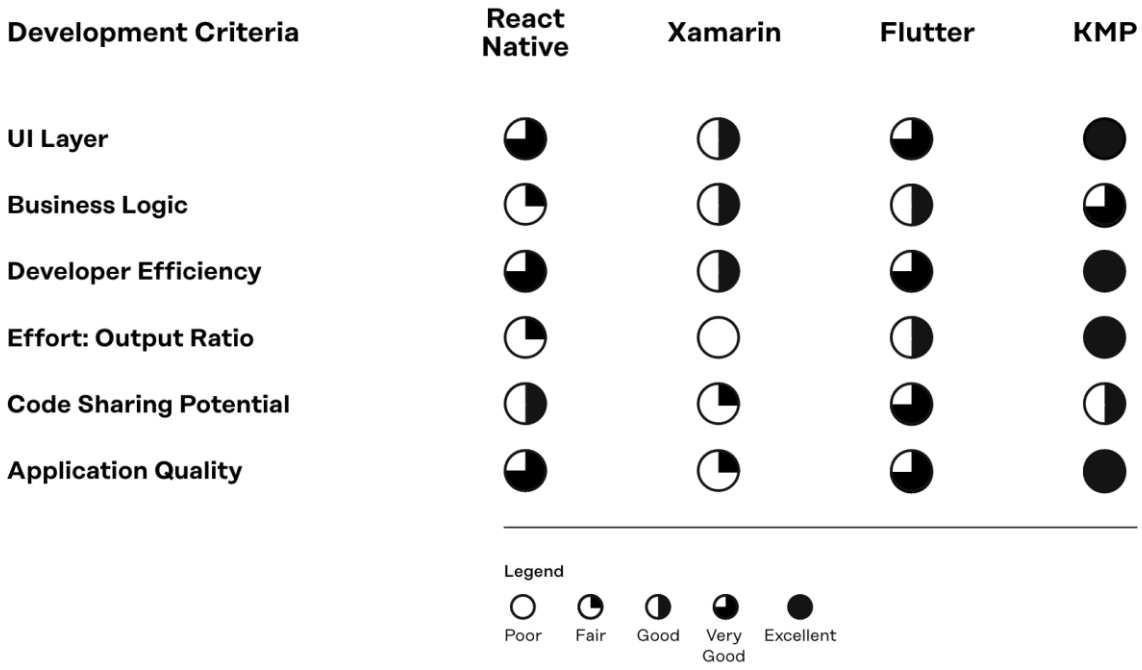


Рисунок 3.3 – Порівняння KMM з іншими інструментами

Android

Для побудови MVP (minimum valuable product) продукту мобільний застосунок був розроблений для системи Android.

Android – це операційна система з відкритим кодом на базі Linux для мобільних пристроїв, таких як смартфони та планшетні комп'ютери. Android був розроблений Open Handset Alliance під керівництвом Google та іншими компаніями.

Android пропонує уніфікований підхід до розробки застосунків для мобільних пристроїв, що означає, що розробникам потрібно розробляти лише для Android, а їхні застосунки повинні мати можливість запускатися на різних пристроях під керуванням Android.

Перша бета-версія Android Software Development Kit (SDK) була випущена Google у 2007 році, а перша комерційна версія Android 1.0 була випущена у вересні 2008 року[17].

На рисунку 3.4 представлені переваги операційної системи Android.

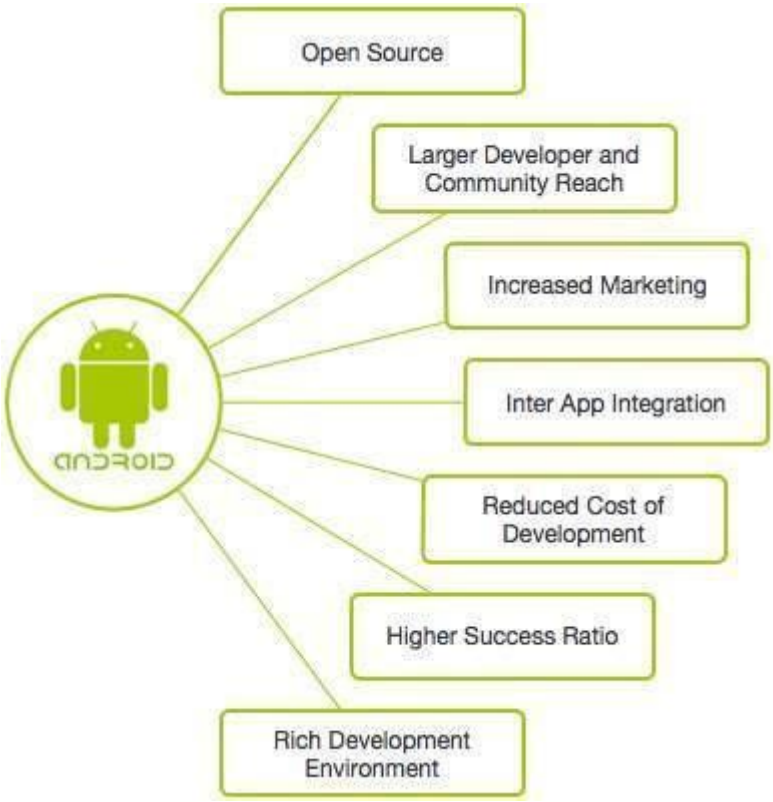


Рисунок 3.4 – Переваги операційної системи Android

Бібліотека ComposeUI

Для побудови презентаційної частини користувацького інтерфейсу була використана бібліотека Jetapck Compose UI.

Jetpack Compose — рекомендований сучасний інструментарій Android для створення нативного інтерфейсу користувача. Це спрощує та прискорює розробку інтерфейсу користувача на Android. Дана бібліотека дозволяє швидко втілювати програму в життя з меншою кількістю коду, потужними інструментами та інтуїтивно зрозумілими Kotlin APIs.

Jetpack Compose має наступні переваги:

- Менше коду: Робіть більше з меншою кількістю коду та уникайте цілих класів помилок, тому код простий і легкий в обслуговуванні;
- Інтуїтивно зрозумілий: Просто опишіть свій інтерфейс користувача, а Compose подбає про інше. Коли стан програми змінюється, інтерфейс користувача автоматично оновлюється;

- Прискорення розвитку: Сумісний з усім наявним кодом, тому ви можете застосовувати його, коли забажаєте. Швидке ітерування з попереднім переглядом і повною підтримкою Android Studio.
- Потужний: Створюйте красиві програми з прямим доступом до API платформи Android і вбудованою підтримкою матеріального дизайну, темної теми, анімації тощо [18].

Висновки до розділу 3

Проведено аналіз і опис технологій, необхідних для реалізації програмної системи.

Для розробки серверної частини використовувались такі технології, як мова програмування Kotlin та фреймворк Spring Boot. У якості бази даних використовувалася об’єктна база даних MongoDB. Реалізація клієнтської частини, тобто мобільного застосунку, відбувалась за допомогою таких інструментів як Kotlin Multiplatform Mobile, бібліотека Jetpack Compose.

Перераховано переваги і недоліки обраних технологій, порівняно з аналогічними інструментами. Визначено, що технології є актуальними і зручними для розробки відповідних компонентів ПЗ.

4 ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

4.1 Загальна архітектура системи

У якості загальної архітектури системи був обраний клієнт-серверний підхід побудови програмного забезпечення(рис. 4.1).

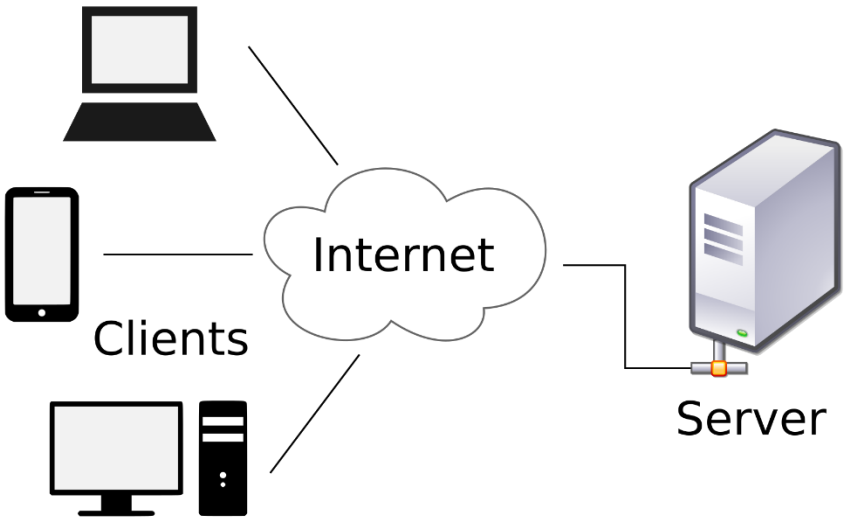


Рисунок 4.1 – Модель клієнт-серверної архітектури

У якості клієнта в даному випадку виступає мобільний застосунок, який по суті є репрезентативною частиною роботи серверу для взаємодії з користувачем.

Обмін даними між клієнтом та сервером відбувається за допомогою REST підходу побудови веб-застосунків. REST – це архітектурний стиль програмного забезпечення, який визначає набір правил, які використовуватимуться для створення веб-сервісів. Веб-сервіси, які відповідають архітектурному стилю REST, відомі як RESTful веб-сервіси. Це дозволяє системам, що роблять запити, отримувати доступ до веб-ресурсів і маніпулювати ними за допомогою єдиного попередньо визначеного набору правил [19].

Сервер, в свою чергу, відповідає за отримання даних із зовнішніх джерел даних, а також з локальної бази даних. В результаті відповідних дій користувача сервер обробляє необхідні дані і надсилає їх клієнту.

Загальну архітектуру системи представлена на рисунку 4.2.

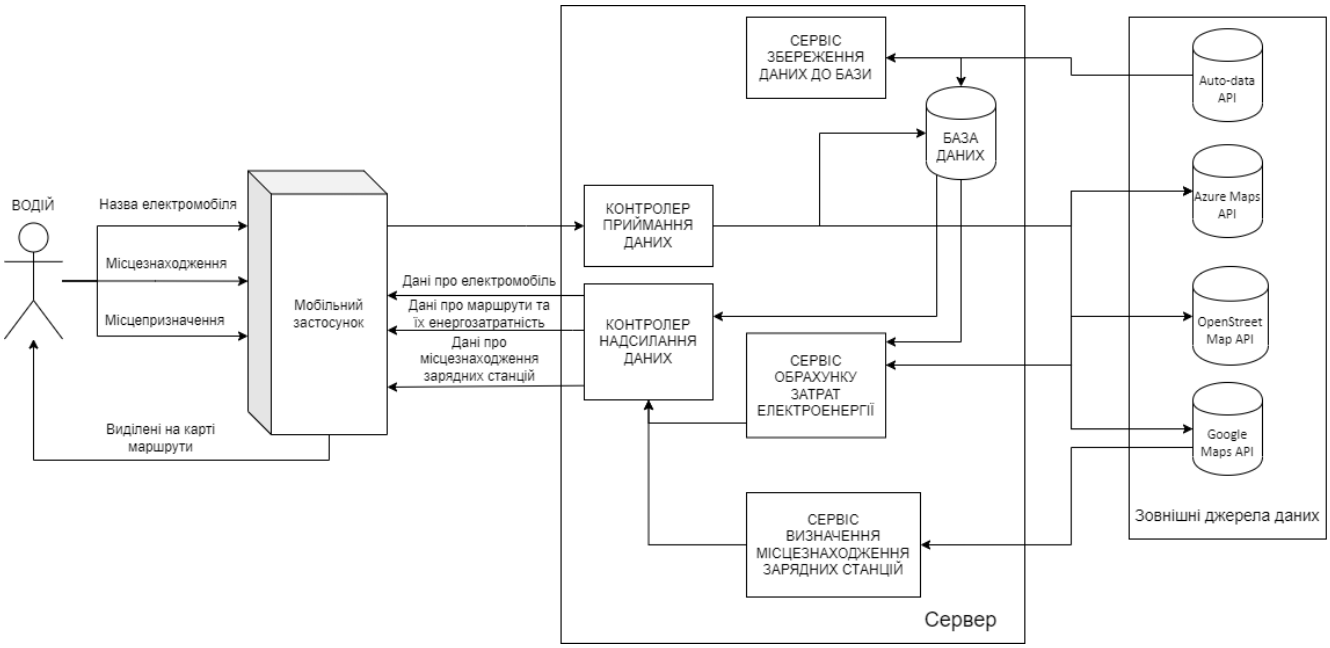


Рисунок 4.2 – Загальна архітектура системи

Загалом весь процес роботи програмного забезпечення можна описати наступним чином:

1. Одразу після запуску серверу надсилається запит на оновлення даних про електромобілів. Після цього дані зберігаються до бази даних;
2. Користувач на мобільному застосунку вводить назву автомобіля. Дані надсилається на сервер. Відбувається запит на отримання даних про відповідний електромобіль. Ці дані надсилаються на мобільний застосунок і відображаються користувачу.
3. Визначається місцезнаходження користувача за GPS, користувач обирає пункт призначання. Дані надсилаються на сервер.
4. Сервер робить запити до бази даних і до зовнішніх джерел, обчислює витрати енергії на маршруті і визначає найбільш енергоефективний.
5. Побудовані маршрути разом з найбільш енергоефективним надсилаються на мобільний застосунок і відображаються користувачу.
6. Користувач може ввести дані про поточний стан батареї. Тоді система порівняє цей показник з кількістю енергії, витраченої на найбільш енергоефективному маршруті, і в разі якщо друге перевищить, зробить

відповідний запит до зовнішнього джерела. В результаті дані будуть отримані і відображені на клієнті.

На рисунку 4.3 представлено діаграму розгортання системи.

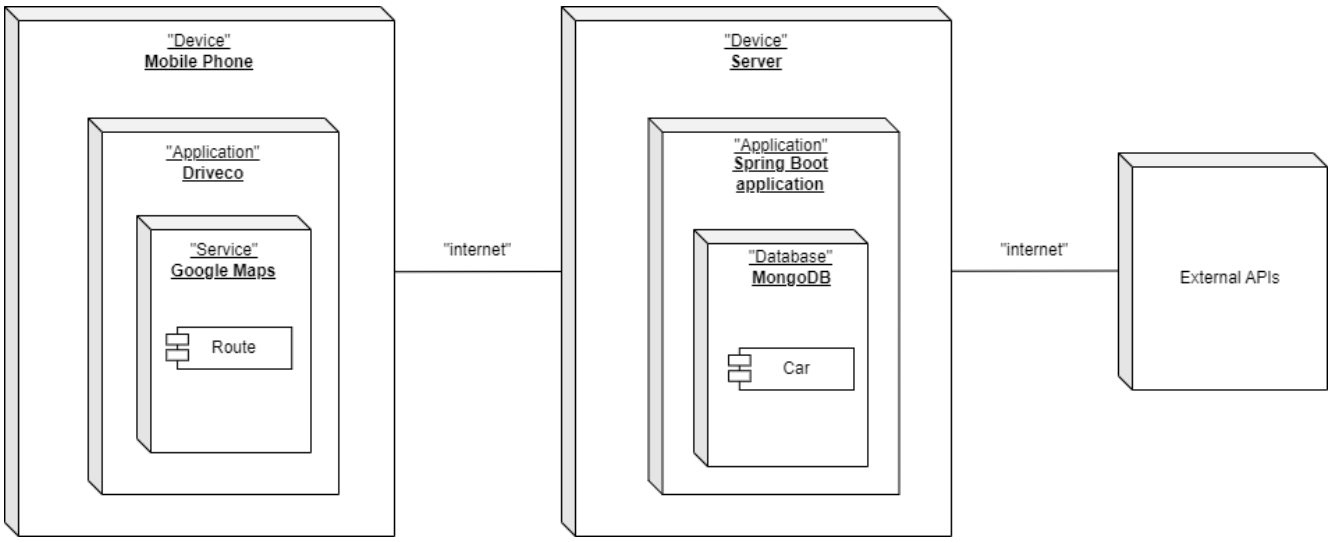


Рисунок 4.3 – Діаграма розгортання системи

На даній діаграмі зображені три основні сутності, з яких розгортається система, а саме

- Мобільний телефон;
- Сервер
- Зовнішні джерела

На мобільному телефоні встановлюється застосунок Driveco, який за допомогою даних про місцезнаходження користувача, його автомобіля та динамічних даних про стан дороги, визначає найбільш енергоефективний маршрут. В застосунку використовується сервіс Google Maps, який будує карту, на якій прокладаються маршрути.

На сервері запускається Spring Boot застосунок, де відбувається всі обчислення енерговитратності маршрутів. Також на сервері піднімається сутність MongoDB бази даних, яка зберігає дані автомобілів за таблицею Car.

Зовнішні джерела даних представляють собою в основному динамічні дані, необхідні для відповідних обчислень.

Взаємозв’язок, а саме передача даних між сутностями відбувається за допомогою мережі Інтернет.

На рисунку 4.4 представлена діаграма прецедентів взаємодії користувачів ПЗ із системою.

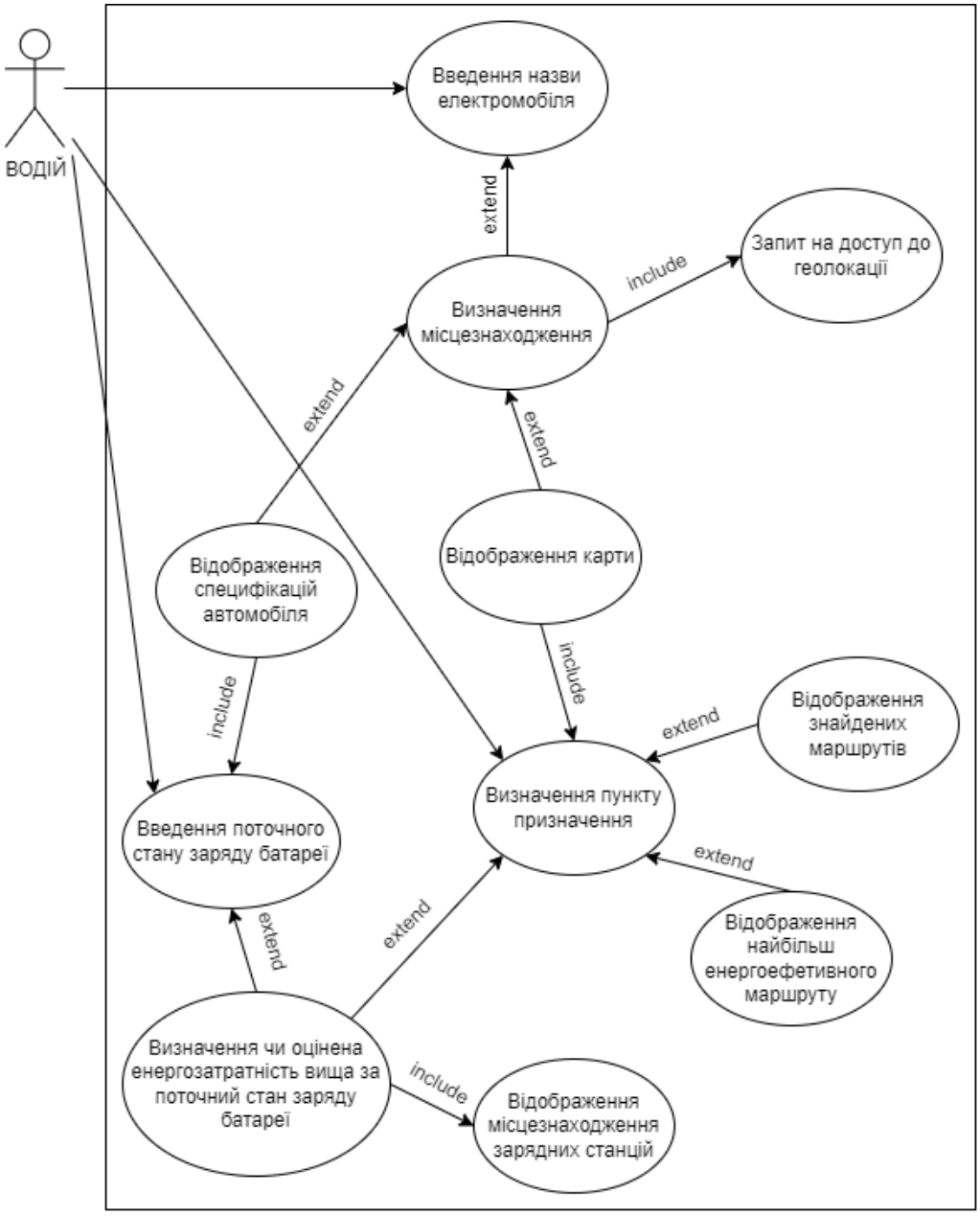


Рисунок 4.4 – Діаграма прецедентів

Для роботи даної системи потрібен лише один тип користувачів – водій. Він може шукати автомобіль за його назвою, визначати пункт призначення маршруту, а також додатково вводити поточний стан акумулятора. При цьому система визначає місцезнаходження користувача, якщо є дозвіл на надання геолокація.

Після цього користувачу відображається карта місцевості, а також специфікації електромобіля. Після того як користувач визначив пункт призначення, відбувається обчислення, та визначення маршрутів за їх енерговитратністю. Якщо користувач надав дані про поточний стан заряду акумулятора і він менший ніж кількість витраченої енергії на визначеному маршруті, відображаються місцезнаходження зарядних станцій близьких до маршруту.

4.2 Опис необхідного набору даних

Для застосування алгоритму визначення кількості енергії, що потрібна для подолання обраного маршруту необхідні наступні дані:

- Дані про погоду у реальному часі, а саме температура повітря, атмосферний тиск, вологість. Ці дані використовувались для підрахунку щільності повітря. Джерелом даних є відкритий, безкоштовний API “openweathermap.org”;
- Специфікації електромобіля, а саме маса авто, його ширина, а також коефіцієнт лобового опору та радіус колеса. Дані були отримані з API “api.auto-data.net”. У виборці присутні приблизно 560 електромобілів;
- Для визначення ефективності електричного мотору була застосована карта ефективності електродвигуна (рис. 2.4) з відповідним набором даних;
- Дані маршрутів були отримані у вигляді наборів точок локації, що в свою чергу являє собою пару значень географічної широти та висоти за допомогою Microsoft Azure Maps API;
- Для того, щоб визначити моменти зниження швидкості були використані дані маневрів на маршруті, а також місцезнаходження світлофорів і завантаженість доріг. Дані отримувались з Google Maps Routes API а також з openstreetmap API;

- Дані про рельєф та тип покриття автомобільної дороги також отримувались з вищезгаданих джерел.

Всі дані отримувались у форматі JSON, за винятком карти ефективності електродвигуна. Ці дані оброблялись у форматі CSV.

4.3 Архітектура серверної частини

Архітектура серверної частини побудована за допомогою шаблону програмування MVC (model – view – controller). Цей шаблон є найбільш поширеним в контексті побудови програмного забезпечення сучасних серверних застосунків. Шаблон MVC пропонує розділити код на 3 компоненти (рис. 4.6) [20].

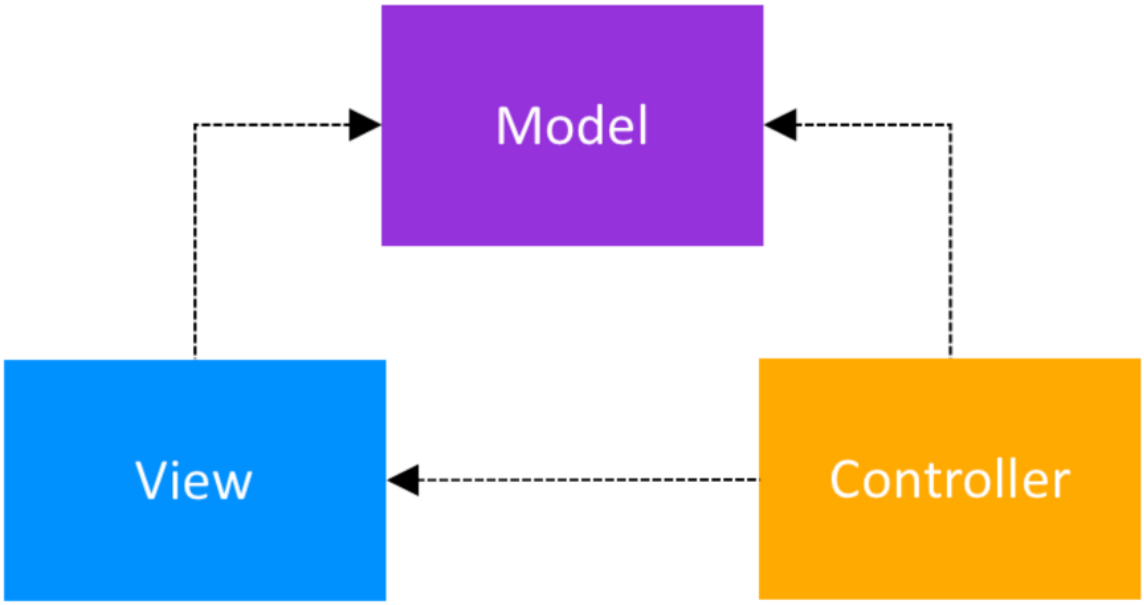


Рисунок 4.6 – Схема шаблону MVC

Під час створення класу/файлу програми розробник повинен класифікувати її за одним із наступних трьох рівнів:

- Model(модель): цей компонент зберігає дані програми. Він не знає про інтерфейс. Модель відповідає за обробку логіки домену (справжні бізнес-правила) і зв’язок із базою даних і мережевими рівнями.

- View(перегляд): це рівень інтерфейсу користувача, який містить компоненти, видимі на екрані. Крім того, він забезпечує візуалізацію даних, що зберігаються в моделі, і пропонує взаємодію з користувачем.
- Controller(контролер): цей компонент встановлює зв’язок між представленням і моделлю. Він містить основну логіку програми, отримує інформацію про поведінку користувача та оновлює модель відповідно до потреб.

В розробленій системі серверний застосунок відповідає за рівень моделі та контролеру, в той час як за рівень перегляду відповідає мобільний застосунок.

В якості контролер рівня в системі є класи-контролери, що відповідають за отримання даних з моделі і надсилання даних на мобільний застосунок, серед них:

- CarController – дані про електромобіль;
- WeatherController – погодні дані;
- RoutesController – дані про маршрути;
- ChargingStationsController – дані про зарядні станції;

Крім цього допоміжний сервіс клас PowerCalculatorService, в якому відбуваються всі обчислення.

За рівень моделі відповідають класи-репозиторії, а також API-клієнти, що отримують інформацію з бази даних та із зовнішніх джерел відповідно.

На рисунку 4.7 зображена діаграма класів серверного застосунку, який реалізує шаблон MVC.

Більшість даних система отримує із зовнішніх джерел і не зберігає їх в базі даних. Це актуально для динамічних даних, таких як дані про погоду або точки маршруту. Зберігання в базі даних в такому разі було б неефективно і достатньо дорого. Проте дані про специфікації електромобілів оновлюються не дуже часто, а тому є сенс зберігати їх в базі даних і оновлювати при необхідності. Оскільки в такому разі нам потрібна лише одна таблиця, було вирішено використати об’єктну базу даних MongoDB.

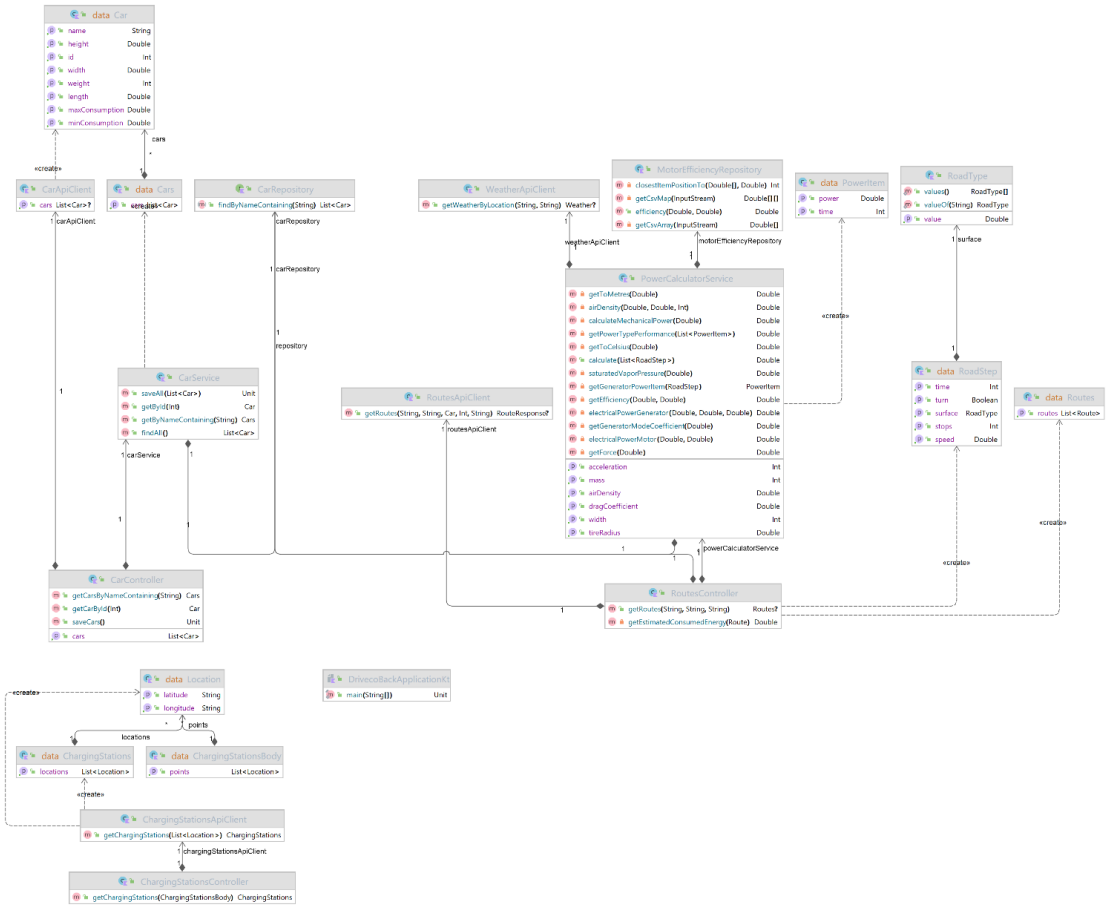


Рисунок 4.7 – Діаграма класів серверного застосунку

Діаграма бази даних виглядає наступним чином (рис. 4.8):

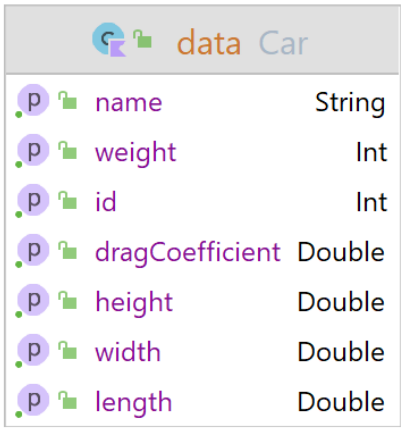


Рисунок 4.8 – Діаграма бази даних системи

Як видно з рисунку 4.8 Діаграма бази даних складається з однієї таблиці Car і має наступні поля: name (тип String), weight (тип Int), id (тип Int), dragCoeffiecient (тип Double), height (тип Double), width (тип Double), length (тип Double).

4.3 Архітектура мобільного застосунку.

Хоча мобільний застосунок був розроблений тільки для системи Android, було передбачено розробку і під інші мобільні операційні системи, наприклад IOS. Це стало можливим завдяки Kotlin Multiplatform Mobile інструменту, що був описаний вище. Отже загалом структура застосунку має наступні компоненти:

- Спільний модуль (Shared) – в основному відповідає за запити до серверу та операції з даними;
- Android app – відповідає за побудову специфічних (нативних) компонентів Android застосунку. В основному це елементи користувацького інтерфейсу.
- IOS app – відповідає за побудову специфічних (нативних) компонентів IOS застосунку. В основному це елементи користувацького інтерфейсу.

На рисунку 4.9 представлена загальна структура системи мобільних застосунків.

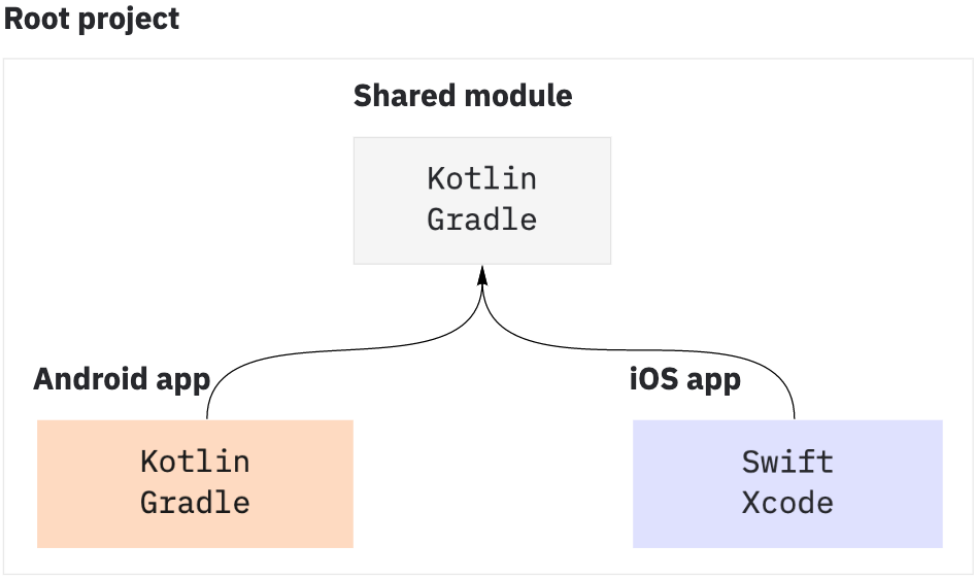


Рисунок 4.9 – Структура розробленої системи для мобільних застосунків

Архітектура MVVM.

В свою чергу архітектура мобільного застосунку Android розроблений за допомогою шаблону MVVM.

Model–View–ViewModel(MVVM) — це визнаний галуззю шаблон архітектури програмного забезпечення, який усуває всі недоліки шаблонів проектування MVP і MVC. MVVM пропонує відокремити логіку представлення даних (Views або UI) від основної частини бізнес-логіки програми [21].

Окремі рівні коду MVVM:

- Model: цей рівень відповідає за абстракцію джерел даних. Модель і ViewModel працюють разом, щоб отримати та зберегти дані.
- View: мета цього шару — повідомити ViewModel про дії користувача. Цей рівень спостерігає за ViewModel і не містить жодної логіки програми.
- ViewModel: відкриває ті потоки даних, які мають відношення до View. Крім того, він служить сполучною ланкою між Model та View (рис. 4.10).

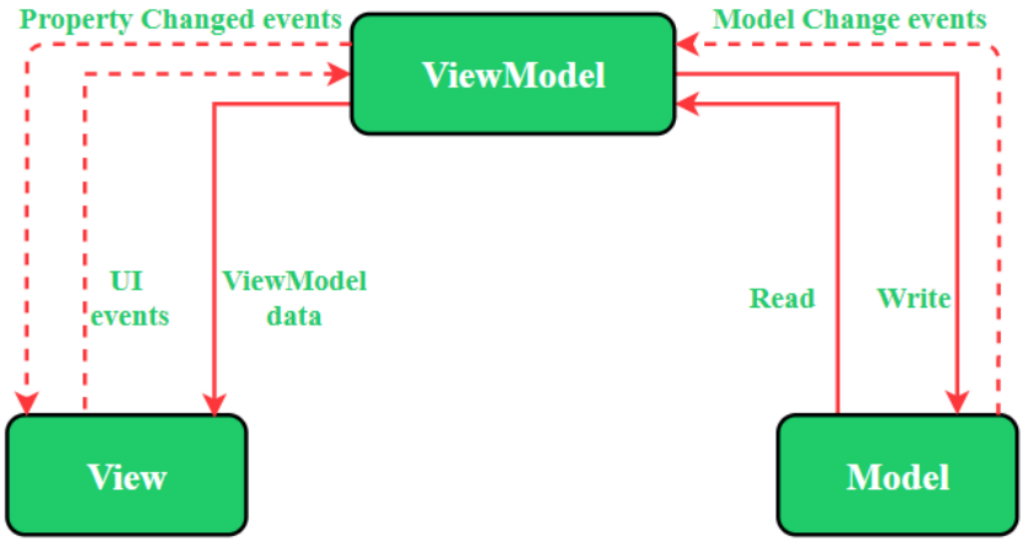


Рисунок 4.10 – Шаблон архітектури MVVM (Model View ViewModel) в Android

Шаблон MVVM має певну схожість із шаблоном проектування MVP (Model-View-Presenter), оскільки роль Presenter виконує ViewModel. Однак недоліки шаблону MVP були вирішені MVVM наступними способами:

- ViewModel не містить жодних посилань на View.
- Між View і ViewModel існують зв’язки «багато-до-одного».

- Немає методів запуску для оновлення View.

В розробленій системі рівень Model повністю винесений в спільний модуль KMM, рівні ViewModel та Model реалізуються для кожної з платформ окремо.

Тоді розроблену архітектуру можна представити за наступною діаграмою (рис. 4.11):

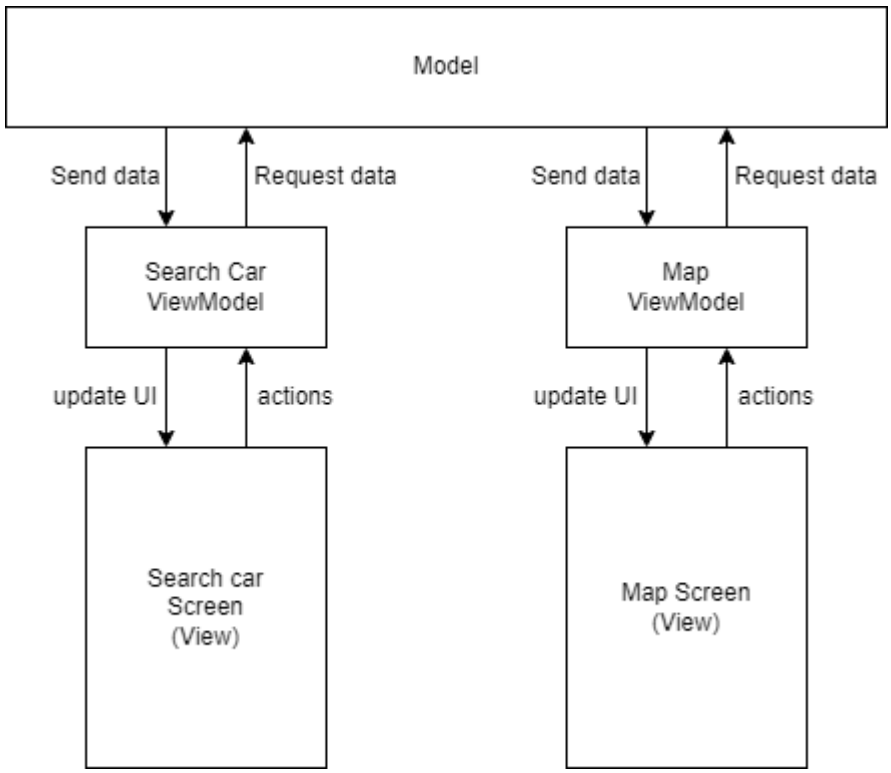


Рисунок 4.11 – архітектура застосунку за шаблоном MVVM

На діаграмі представлено два екрани які представляють рівень View, а саме:

- Search car Screen – являє собою екран пошуку електромобіля;
- Map Screen – являє собою карту, на якій відображаються маршрути та інші умовні позначки, такі як місцезнаходження користувача, пункт призначення тощо.

У кожного з екранів є свій ViewModel. Коли на екрані відбувається якась подія, View взаємодіє з відповідною ViewModel. Той в свою чергу взаємодіє з Model, запитуючи необхідні дані. В результаті Model надсилає відповідні дані до ViewModel, а ViewModel оновлює представлення інтерфейсу, тобто View.

Даний підхід є доволі зручним при побудові мобільних застосунків, що робить розробку більш швидкою і якісною. До того ж шаблон MVVM наразі є рекомендованим компанією Google для розробки Android застосунків.

Висновки до розділу 4

В четвертому розділі магістерської дисертації було розглянуто детальний опис програмної реалізації всієї розробленої системи. Основним методом розробки ПЗ є клієнт-серверний підхід, де клієнтом є мобільний застосунок.

Були представлені загальна архітектура ПЗ, діаграма розгортання, діаграма прецедентів.

Проаналізовано дані, що необхідні для застосування алгоритму знаходження енерговитратності електромобіля на заданому маршруті. В основному, джерелами є відкриті для користування та безкоштовні API, з яких можна отримувати дані у форматі JSON. Була завантажена база даних електромобілів з понад 560 одиниць.

Для серверної частини визначена діаграма класів, як відображення побудови застосунку за шаблоном MVC. Архітектура мобільного застосунку розроблена за шаблоном MVVM, що є рекомендованим для розробки мобільного ПЗ.

5 ОПИС ОДЕРЖАНИХ РЕЗУЛЬТАТІВ

5.1 Робота користувача з системою

Основним способом взаємодії користувача з системою є мобільний застосунок на операційній системі Android. Для використання системи користувач має встановити мобільний застосунок смартфон з операційною системою Android.

При запуску користувач потрапляє на початкову сторінку застосунку, де представлений короткий опис застосунку, а також кнопка для пошуку електромобіля (рис. 5.1).

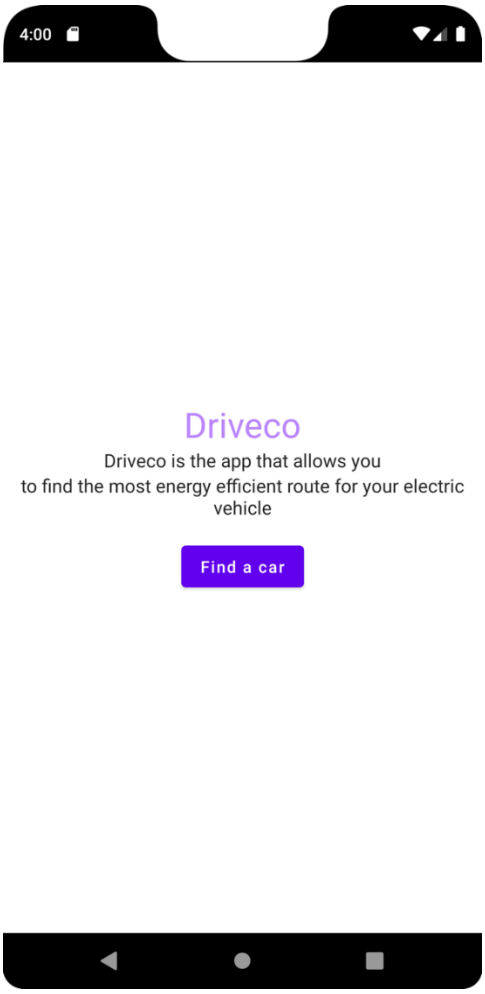


Рис 5.1 – Початковий екран

Після натискання кнопки “Find a car” користувач потрапляє на екран пошуку електромобіля (рис. 5.2).

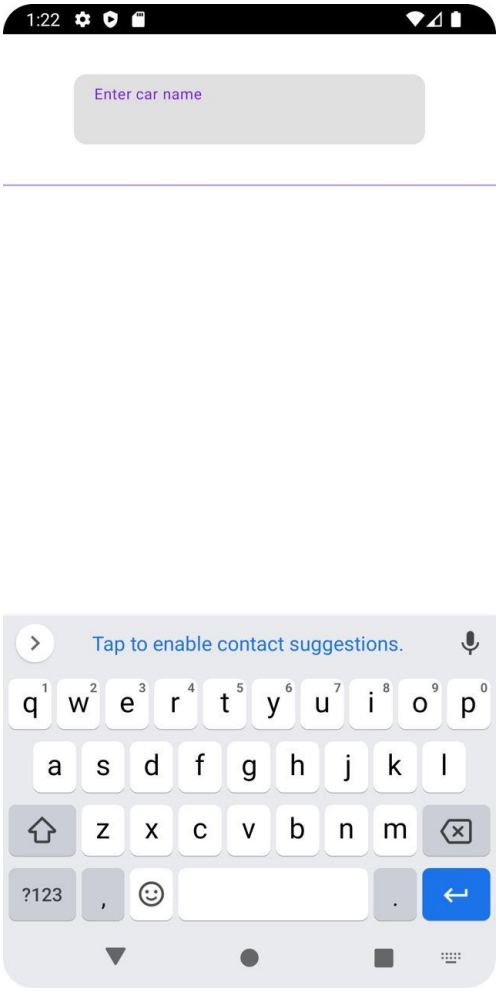


Рисунок 5.2 – Екран пошуку електромобіля

При введенні хоча б 3-х символів відбувається запит на сервер для пошуку електромобіля. Сервер, в свою чергу, звертається до локальної бази даних, щоб отримати набір електромобілів, назва яких задовольняє умову, що введений текст міститься в даній назві. Після успішного виконання запиту, користувачу відображається список знайдених машин. Надалі, при кожному введенні нового символу знову буде відбуватись відповідний запит до серверу. Таким чином користувачу не потрібно кожен раз натискати «Пошук» для власне пошуку авто (рис. 5.3).

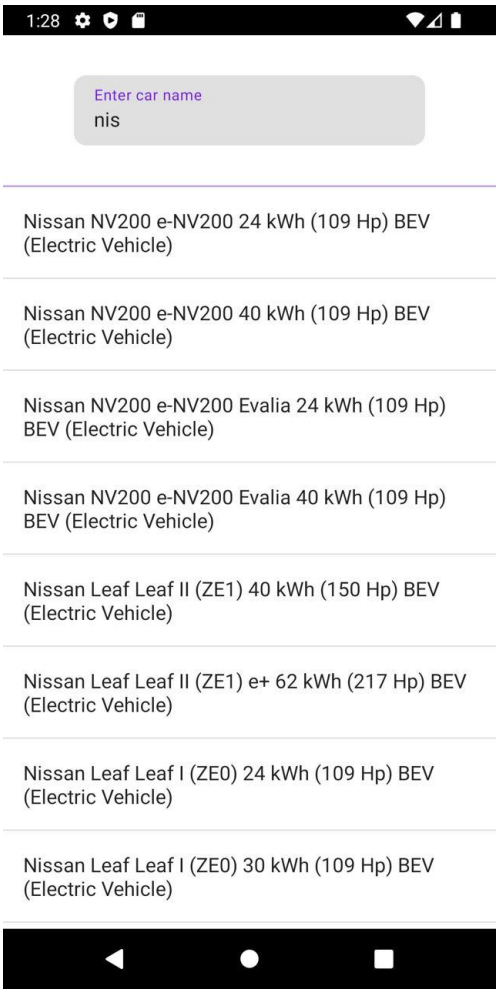


Рисунок 5.3 – Приклад успішного пошуку електромобілів

Тепер користувач може ввести більш точну назву, тим самим звузивши пошук або одразу обрати будь-який електромобіль, що відповідає його власному. Для цього, йому потрібно натиснути на секцію з відповідним авто. В результаті відповідна секція виділиться більш темним кольором, а внизу з’явиться кнопка «Save car» (рис. 5.4). Користувач може вільно відміняти вибір або обрати інший автомобіль при помилковому виборі.

Натиснувши кнопку «Save car» користувач потрапляє на екран, де йому відображається географічна карта. При цьому, якщо користувач не надав дозвіл на отримання місцезнаходження, система запитує про надання такого дозволу (рис. 5.5).

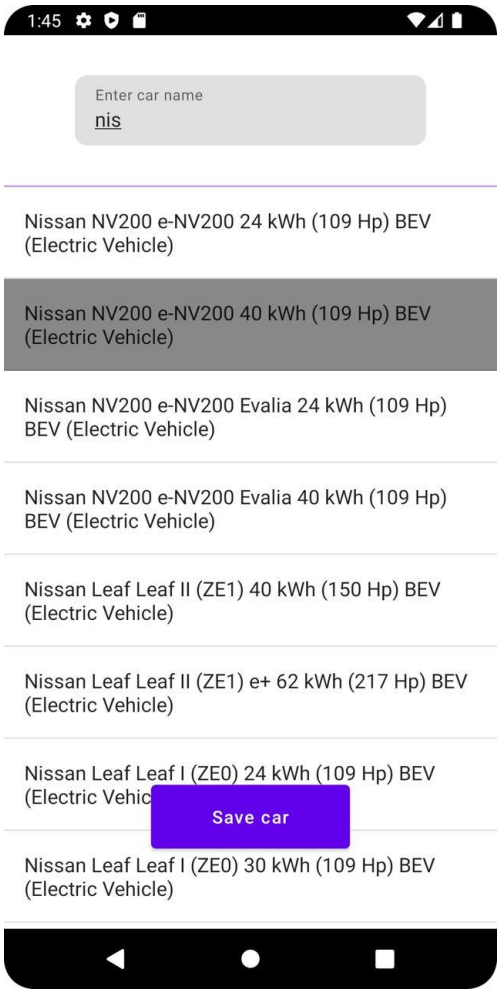


Рисунок 5.4 – Приклад вибору електромобіля користувачем

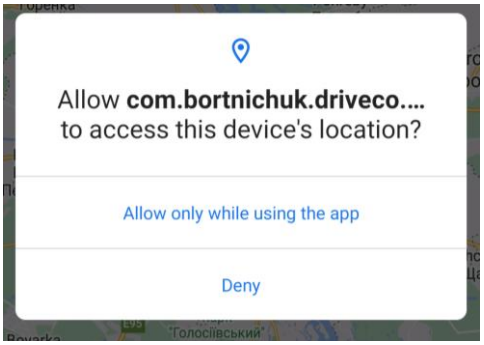


Рисунок 5.5 – Запит на надання доступу до місцезнаходження пристрою

Після того, як користувач зрештою надав доступ до локації, відкривається географічна карта (рис. 5.6).

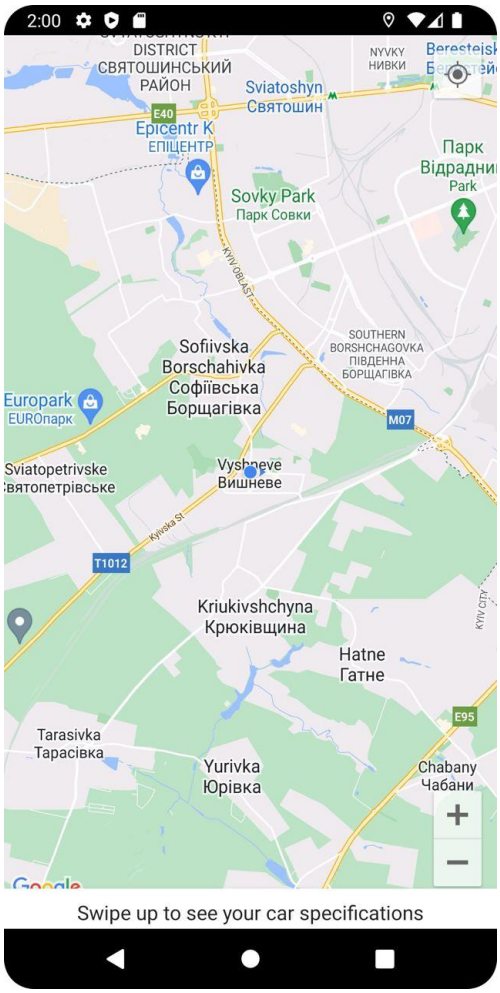


Рисунок 5.6 – Географічна карта

На рисунку 5.6 видно, що для пристрою було визначено місцезнаходження. Воно позначається синьою точкою. Дана карта дозволяє збільшувати або зменшувати маршрут за допомогою кнопок «+» та «-» відповідно, що розташовані справа від екрану. В правому верхньому куті знаходиться кнопка, натискання якої спричиняє відцентрування та масштабування точки локації пристрою.

В нижній частині екрану можна побачити компонент на білому фоні. Якщо здійснити проведення пальцем цього компоненту вверх відкриється модальне вікно з специфікаціями електромобіля, а саме: його повна назва, маса, довжина, висота та ширина, а також поточний стан акумулятора, який користувач може опціонально ввести сам (рис. 5.7).

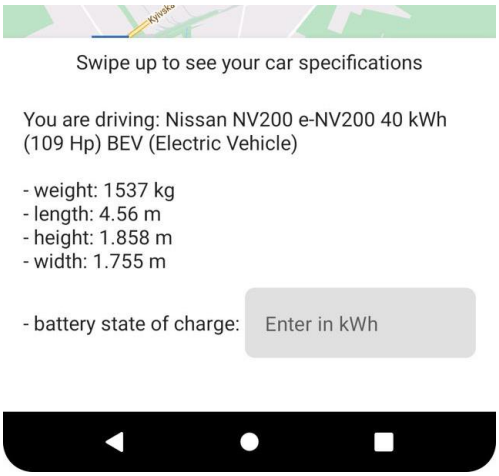


Рисунок 5.7 – Модальне вікно з специфікаціями електромобіля

Для того щоб почати будувати маршрут не вистачає лише пункту призначення. Користувач може обрати його, натиснувши і трохи потримавши на відповідній точці на карті.

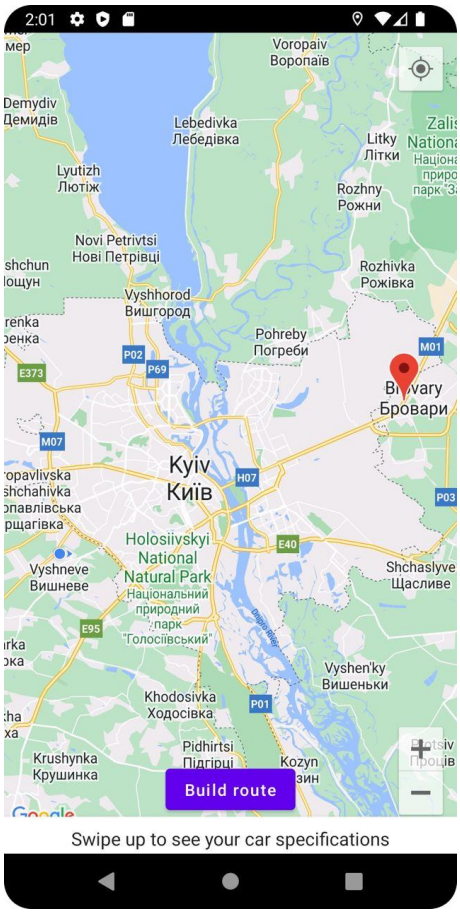


Рисунок 5.8 – Визначена точка призначення маршруту

В результаті на екрані з’явиться червоний маркер і кнопка «Build route».

При натисканні кнопки «Build route» відбудеться запит на сервер з відповідними даними, а саме початкова і кінцеві точки маршруту, а також ідентифікатор електромобіля. На сервері відбудуться запити до зовнішніх джерел для отримання динамічних даних та до бази даних для отримання специфікацій машини, а також всі необхідні обчислення. Для цього необхідно значний проміжок часу, тому на час проведення всіх операцій відображається екран завантаження (рис. 5.9).

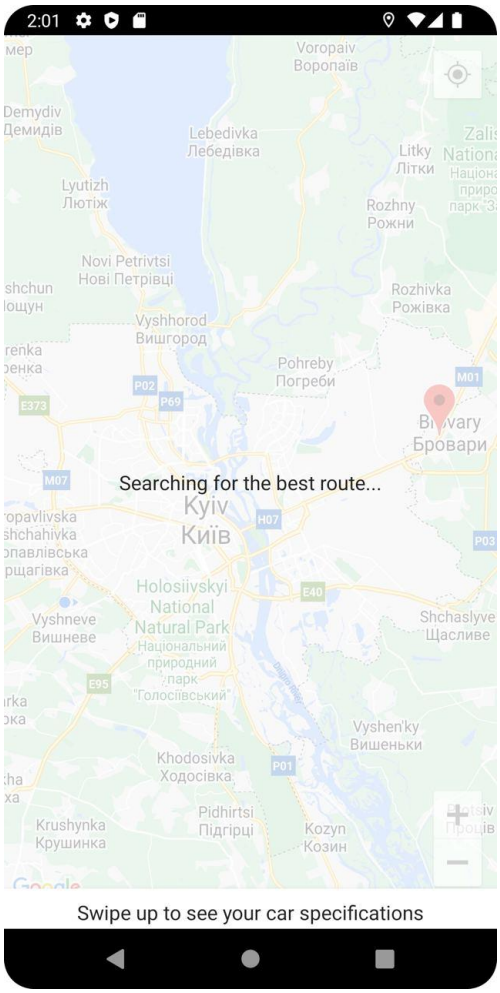


Рисунок 5.9 – Екран завантаження маршруту

Після недовгого очікування (зазвичай не більше 3-х секунд), користувачу відображаються побудовані маршрути (рис. 5.10).

Маршрут, на подолання якого витратиться найменша кількість електроенергії, виділяється яскраво фіолетовим кольором, таким чином користувач

може його легко визначити. Всі інші можливі розглянуті маршрути позначені сірим кольором.

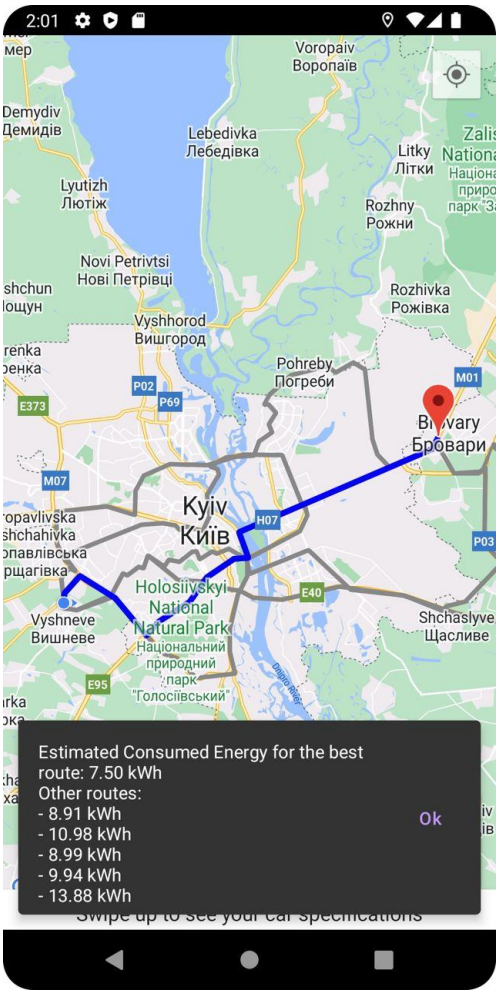


Рисунок 5.10 – Приклад побудови найбільш енергоефективного маршруту

На рисунку 5.10 побудований найбільш енергоефективний маршрут з м. Вишневе до м. Бровари.

Крім виділення на карті система визначає кількість енергії, витраченої на кожному з маршрутів. Видно, що на подолання найбільш економного маршрут потрібно витратити 7.5 кВт*г.

Як було зазначено вище, користувач може ввести значення поточного стану акумулятора електромобіля в модальному вікні специфікацій машини (рис. 5.7). На рисунку 5.11 зображено приклад введення такого значення.

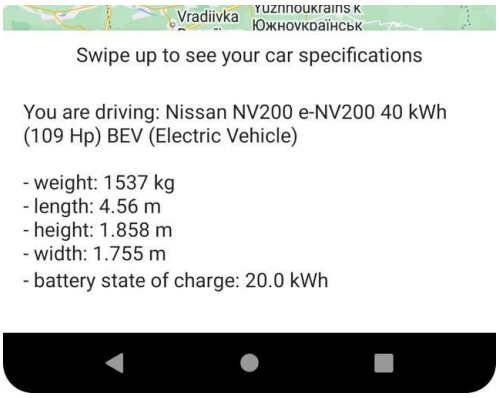


Рисунок 5.11 – Приклад введеного значення стану акумулятора

Тоді, якщо витрати для визначеного найбільш енергоефективного маршруту перевищують введене значення, відбудеться запит до серверу для отримання найближчих до маршруту зарядних станцій. В результаті на карті будуть позначені їх місцезнаходження (рис. 5.12).

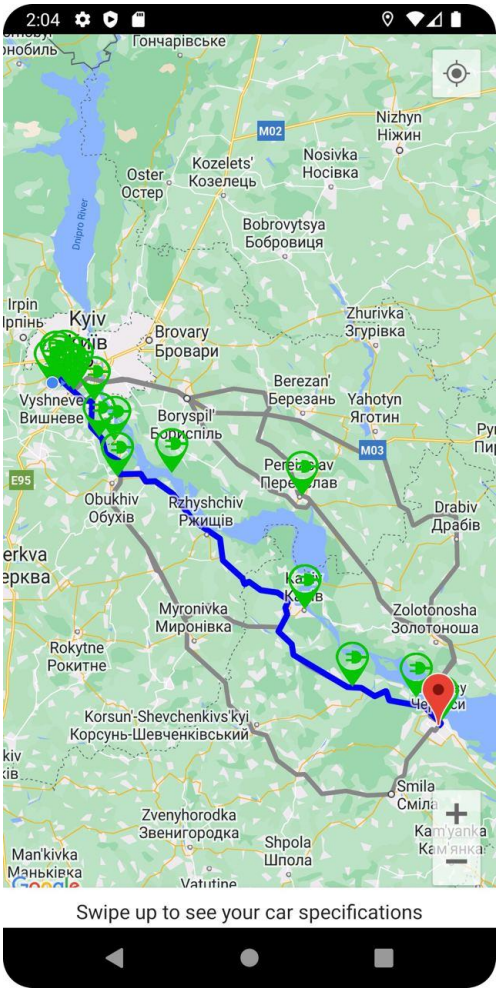


Рисунок 5.12 – Зарядні станції на маршруті

Місцезнаходження зарядних станції позначені зеленими маркерами.

5.2 Порівняння отриманих результатів з існуючими системами

Для порівняння отриманих результатів з існуючи системами, а саме Google Maps та Waze використаємо побудову маршруту від центра міста Вишневе (50.39203,30.36821) до мікрорайону в південно-східній частині міста Бортничі (50.37686,30.69341).

Отже, розроблена система визначила маршрут, що веде через Солом’янський район, далі через міст Патона виїжджає на лівий берег міста Київ і через Дарницький район доїжджає до пункту призначення. Загалом даний маршрут займає 34277 метрів. Оцінені витрати складають 5.9 кВт*г (рис. 5.13).

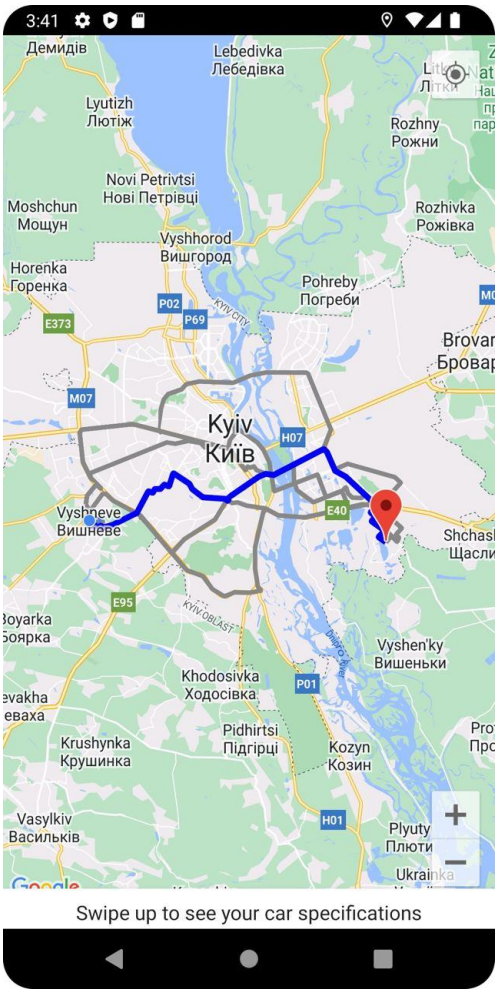


Рисунок 5.13 – Приклад прокладення маршруту для прикладних точок

Далі розглянемо побудову маршрутів для тих самих точок з іншими системами.

На відміну від розробленої системи Google Maps побудував маршрут через Голосіївський район за напрямком метро, але при цьому виїжджаючи на лівий берег також через міст Патона, далі маршрути ідентичні. При цьому вказано, що дистанція даного маршруту складає близько 35 км і займає 46 хвилин (рис. 5.14). Порівнявши з аналогічним маршрутом, побудованим розробленою системою визначаємо, що на його подолання витратися 6.14 кВт*г.

Застосунок Waze визначив маршрут що веде через Велику Окружну дорогу і через Дарницький міст, що займає 42км і 52 хвилини (рис. 5.15). На аналогічний маршрут, визначений системою, витратиться 7.58 кВт*г.

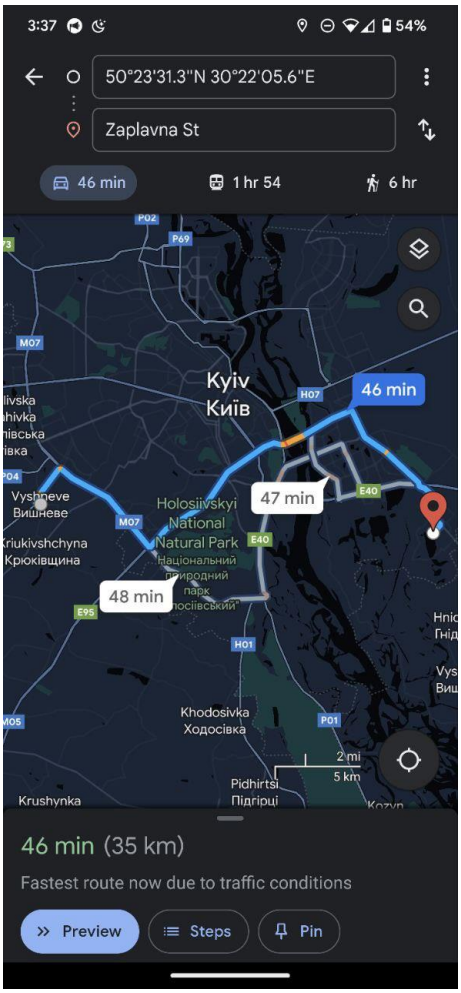


Рисунок 5.14 – Google Maps

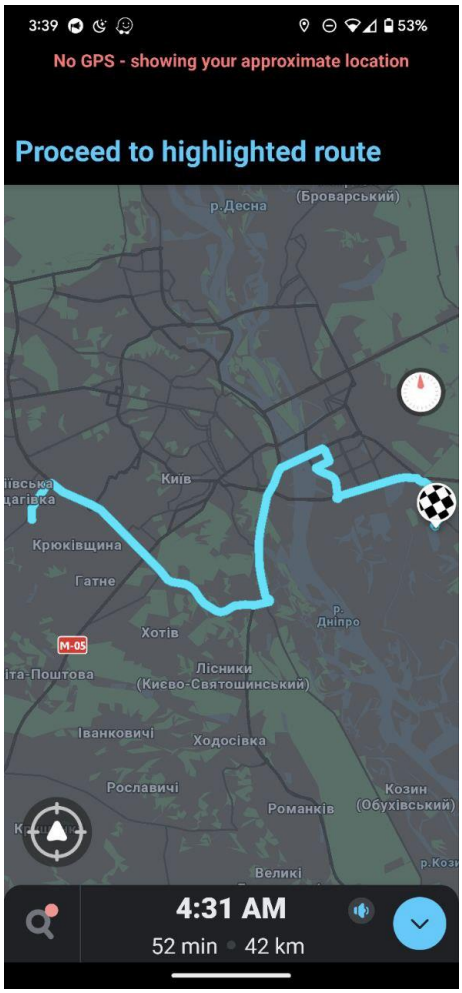


Рисунок 5.15 – Waze

Таким чином, можна побачити, що існуючі системи визначили менш енергоефективні маршрути, ніж розроблена система. Для більшої наочності

експеримент проводився і для інших маршрутів. Результати представлено на наступній таблиці (табл. 5.1).

Таблиця 5.1 – Результати порівняння результатів з існуючими системами

Маршрут		Витрати (кВт*г)		
Початок	Кінець	Driveco	Google Maps	Waze
50.39203,30.36821	50.53461,30.45793	3.66	3.67	4.18
50.44966,30.46576	49.74304,31.44520	25.49	25.49	25.49
50.44966,30.46576	50.51585,30.60148	3.43	3.45	3.45
50.44966,30.46576	51.48435,31.29616	31.26	32.21	31.48

В таблиці 5.1 розроблена система представлена під назвою «Driveco».

Висновки до розділу 5

В п’ятому розділі роботи було представлено роботу користувача з системою. Інтерфейс дозволяє користувачу з достатньою повнотою використовувати розроблені функції програмного забезпечення.

Визначено, що система відповідає поставленому завданню, тобто будує найбільш енергоефективний маршрут серед можливих.

Додатково надана можливість дізнатись місцезнаходження зарядних станцій, якщо поточний стан заряду акумулятора нижчий за визначений для маршруту.

Отримані результати порівняно з існуючими системами. Визначено, що альтернативні системи часто пропонують не найбільш енергоефективні маршрути, на відміну від розробленого ПЗ, що ніяк не допомагає зменшити забруднення навколишнього середовища.

6 РОЗРОБКА СТАРТАПУ ПРОЄКТУ

6.1 Опис ідеї проєкту

Загальна ідея проєкту полягає в тому, щоб розробити таке програмне забезпечення, яке визначало б найбільш енергоефективний маршрут для електромобілів через мобільний застосунок. Для формування цілісного уявлення про ідею проєкту необхідно сформувати зміст ідеї, можливі напрямки застосування, вигоди для користувача, а також основні відмінності від існуючих аналогів [22]. Перші три пункти представлено в таблиці 6.1.

Таблиця 6.1 – Опис ідеї стартап-проєкту

Зміст ідеї	Напрямки застосування	Вигода для користувача
Створення програмного забезпечення, яке визначає найбільш енергоефективний маршрут для електромобілів	1. Використання системи водіями для вибору найменш енерговитратного маршруту	Економія коштів; Зменшення витрат електроенергії, як результат – внесок у збереження планети
	2. Прокладання маршруту, навігація	Краще орієнтування на дорозі
	3. Пошук найближчих зарядних станцій для електромобілів	Дозволяє швидко знаходити зарядну станцію або заздалегідь планувати маршрут з підзарядкою

У якості аналогічних систем були розглянуті мобільний застосунок Google Maps та мобільний застосунок Waze, а саме функціонал побудови маршрутів для автомобілів. Основні відмінності між розробленою системою і аналогами , а також порівняльний аналіз показників цих систем представлений на таблиці 6.2.

Таблиця 6.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проєкту

№ п/п	Техніко-економічні характеристики ідеї	(потенційні) товари/ концепції конкурентів			Слабкі (W), нейтральні (N) та сильні (S) сторони		
		Driveco	Google Maps	Waze	W	N	S
1.	Доступність	Всюди, де є інтернет і доступ до геолокації	Всюди, де є телефон(через можливість офлайн карт)	Всюди, де є інтернет		+	
2.	Використання GPS	Обов’язкове	Не обов’язкове	Не обов’язкове	+		
3.	Врахування погодних даних (температура, тиск, вологість)	Так	Ні	Ні			+
4.	Побудова маршруту	За витратами енергії	За швидкістю	За швидкістю			+
5.	Тип транспортного засобу	Електромобіль	Будь-який транспортний засіб	Будь-який транспортний засіб	+		
6.	Складність використання	Просто	Середня складність	Середня складність			+

6.2 Технологічний аудит проєкту

В ході планування і розробки програмного забезпечення ціллю було використовувати найбільш сучасні технології, що роблять сам процес розробки простим, а головне таким, що може розширюватись у майбутньому. На таблиці 6.3 наведений аудит технологій, за допомогою яких можна реалізувати ідею проекту.

Таблиця 6.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проекту	Технології і реалізації	Наявність технологій	Доступність технологій
1.	Створення серверного застосунку	Фреймворк для веб-застосунків Spring Boot	Наявна	Доступна
		Бібліотека Node.js	Наявна	Доступна
		Фреймворк для веб-застосунків Python Django	Наявна	Доступна
2.	Створення кросплатформеного мобільного застосунку	React Native	Наявна	Доступна
		Xamarin	Наявна	Доступна
		Flutter	Наявна	Доступна
		Kotlin Multiplatform	Наявна	Доступна
Обрані технології реалізації ідеї проекту: Для розробки серверного застосунку був обраний фреймворк Spring; для створення кросплатформеного застосунку був обраний інструмент Kotlin Multiplatform				

6.3 Аналіз ринкових можливостей стартап-проєкту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів.

Спочатку необхідно провести аналіз попиту (табл. 6.4)

Таблиця 6.4 – Попередня характеристика потенційного ринку стартап-проєкту

№ п/п	Показники стану ринку (найменування)	Характеристика
1.	Кількість головних гравців, од	3
2.	Загальний обсяг продаж, грн/ум.од	10 млн.
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5.	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6.	Середня норма рентабельності в галузі (або по ринку), %	21

Згідно з отриманими результатами, ринок є привабливим для входження за попереднім оцінюванням.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 6.5).

Таблиця 6.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності в поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1.	Економічна їзда на електромобілі	Користувачі електричних транспортних засобів	Використання програмного забезпечення прокладання найбільш енергоефективних маршрутів	Користуватись смартфоном, що може надавати локацію за GPS
2.	Зменшення викидів в атмосферу	Користувачі електричних транспортних засобів	Усвідомлення участі в збереженні навколишнього середовища	Слідувати визначеним маршрутом

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проєкту, та факторів, що йому перешкоджають (табл. 6.6, 6.7). Фактори в таблиці подаються в порядку зменшення значущості.

Таблиця 6.6 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1.	Висока якість продукту	Вихід на міжнародний ринок	Розширення підтримки програмного продукту в країнах світу

Продовження таблиці 6.6

2.	Техноло-гічний розвиток	Розвиток смартфонів та використовувани х технологій	Впровадження нових технологій у застосунок; Покращення алгоритму визначення енерговитратності електромобіля на заданому маршруті

Таблиця 6.7 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1.	Конкуренція	Поява конкурентів зі схожим функціоналом	Введення нового функціоналу. Проведення маркетингової кампанії.
2.	Неоптимізована робота продукту	Використання застосунком більшої кількості даних із зовнішніх джерел	Удосконалення додатку шляхом оптимізації запитів до зовнішніх джерел.

Після цього необхідно провести аналіз пропозиції: визначити загальні риси конкуренції на ринку (табл. 6.8).

Таблиця 6.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	У чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоби бути конкурентоспроможною)
1. Тип конкуренції: чиста	На ринку присутні декілька компаній, але в основному вони прив’язані до визначення маршруту найшвидшим шляхом.	Підтримка високої якості продукту та впровадження нових функцій.

Продовження таблиці 6.8

2. Рівень конкурентної боротьби: - глобальний	Компанії-конкуренти з різних країн.	Розвиток в українській IT сфері з подальшим виходом на міжнародний ринок
3. Галузева ознака: - міжгалузева	Використання продукту для побудови маршрутів, що є найбільш економними у витратах.	Вдосконалення та покращення алгоритму обчислення.
4. Конкуренція за видами товарів: - товарно-видова	Конкуренція між програмним забезпеченням одного виду.	Випуск якісніших версій програмного забезпечення, взаємодія з іншими сервісами.
5. Характер конкурент-них переваг: - нецінова	Функціональні можливості програмного забезпечення.	Розширення функціональних можливостей
6. Інтенсив-ність: - не марочна	Важливим є якість продукту, а не бренд.	Розвиток продукту

На таблиці 6.9 наводиться більш детальний аналіз умов конкуренції в галузі (за моделлю 5 сил М. Портера).

Таблиця 6.9 – Аналіз конкуренції в галузі за М. Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Google Maps, Waze	Сильний бар'єр входження, адже застосунку розроблені великими компаніями	Не сильний фактор постачальників, адже використовуються власні API	Досить сильний фактор споживачів, продукт можна замінити аналогами	Відсутні. Наявні лише конкуренти з аналогічними розробками
Висновки	Досить інтенсивна конкурентна боротьба, необхідно працювати над розвитком застосунку	Входи в ринок досить сильні, а це означає, що строки виходу конкурентів на ринок складають від 1 року	Залежності від постачальників майже немає	Споживачі не завжди мають прямий вплив на застосунок, але їх думка щодо покращень враховується	Постійно додавати нові функції для збереження місця на ринку

За таблицею 6.9 визначено, що конкуренти є великими гравцями ринку, а отже виходити на ринок потрібно з якісним, повним можливостей продуктом. Сильними сторонами можуть бути впровадження нових корисних функцій, яких немає у конкурентів.

На основі аналізу конкуренції, проведеного в таблиці 6.9, а також із урахуванням характеристик ідеї проєкту, наведених в таблиці 6.2, вимог

споживачів до товару, як показано в таблиці 6.5, та факторів маркетингового середовища, продемонстрованих в таблицях 6.6 і 6.7, визначається та обґрунтовується перелік факторів конкурентоспроможності. Аналіз оформлено в таблиці 6.10.

Таблиця 6.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1.	Потреби споживачів	Потреби споживачів зумовлюють необхідність розробки проекту
2.	Оригінальна особливість побудови маршруту за енергоефективністю	На ринку немає поширених застосунків, що реалізують дану особливість
3.	Допомога в збереженні довкілля	Зменшення витрат електроенергії зумовлює зменшення шкідливих викидів в атмосферу
4.	Ціна продукту	Безкоштовна, як і в конкурентів
5.	Технічне обслуговування	Випуск нових версій продукту, постійне оновлення

За визначеними факторами конкурентоспроможності проведено аналіз сильних та слабких сторін стартап-проєкту (табл. 6.11).

Таблиця 6.11 – Порівняльний аналіз сильних та слабких сторін

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним проектом						
			-3	-2	-1	0	+1	+2	+3
1.	Потреби споживачів	15					+		
2.	Оригінальна особливість побудови маршруту за енергоефективністю	20	+						

Продовження таблиці 6.11

3.	Допомога в збереженні довкілля	18	+						
4.	Ціна продукту	20							+
5.	Технічне обслуговування	11						+	

Для закінчення ринкового аналізу можливостей впровадження проєкту складений SWOT-аналіз (матриці аналізу сильних (Strength) та слабких (Weak) сторін, загроз (Troubles) та можливостей (Opportunities) (табл. 6.12).

Таблиця 6.12 – SWOT-аналіз стартап-проєкту

Сильні сторони (S): – Розумна цінова політика – Простота користування – Підтримка програмного продукту – Інструкція по експлуатації – Якість продукту – Оригінальність ідеї на ринку – Користь	Слабкі сторони (W): – Сумнівна точність алгоритму – Невелика доля ринку, через невелику кількість електромобілів
Можливості (O): – Вихід на міжнародний ринок – Оптимізація роботи продукту Додавання нових особливостей	Загрози (T): – Витрата лімітів зовнішніх сервісів Менша кількість користувачів, ніж прогнозовано

На основі SWOT-аналізу розроблено альтернативи ринкової поведінки (перелік заходів) для виведення стартап-проєкту на ринок та орієнтовний оптимальний час їх ринкової реалізації з огляду на потенційні проєкти конкурентів, що можуть бути виведені на ринок

Визначено альтернативи, які аналізуються з точки зору строків та ймовірності отримання ресурсів (табл. 6.13).

Таблиця 6.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1.	Проведення демо-конференції для міжнародного ринку	30%	2-3 місяці
2.	Безкоштовне розповсюдження створеного продукту	80%	6 місяців
3.	Розповсюдження продукту за підпискою на окремі особливості	30%	1 рік

В результаті аналізу було обрано безкоштовне розповсюдження створеного продукту як альтернативу, адже ймовірність отримання ресурсів в такому разі дуже висока. Витрати при цьому планується покривати впровадженням реклами.

6.4 Розроблення ринкової стратегії програмного продукту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів [23]. На таблиці 6.14 представлені цільові групи потенційних споживачів, а також обрана основна.

Таблиця 6.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1.	Користувачі електромобілів	готові	середній	мала	легко

Продовження таблиці 6.14

2.	Користувачі інших транспортних засобів	Не готові	низький	велика	важко
Які цільові групи обрано: користувачі електричних транспортних засобів					

За результатами аналізу потенційних груп споживачів (сегментів) обрано стратегію концентрованого маркетингу, тобто зосередження на одному сегменті.

На таблиці 6.15 визначається базова стратегія розвитку.

Таблиця 6.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проєкту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1.	Розробка застосунку для побудови найбільш енергоефективного маршруту для електромобілів	Вибірковий розподіл	Цінова політика, Оригінальність функціоналу, його розширення	Стратегія спеціалізації

Наступним кроком було обрано стратегію конкурентної поведінки (табл. 6.16).

Таблиця 6.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1.	Так	Так, буде шукати нових і частково забирати існуючих	Так, деякі особливості, що полегшують роботу з системою користувачу	Стратегія заняття конкурентної ніші.

На таблиці 6.17 показана розробка стратегії позиціонування, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 6.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкуренто-спроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1.	Правильне передбачення витраченої енергії на маршруті	Стратегія спеціалізації	Покращення алгоритму обчислення енерговитратності	Маршрут, енергоефективність, економія

6.5 Розроблення маркетингової програми стартап-проєкту

На першому етапі створення маркетингової програми була сформована маркетингова концепція товару (табл. 6.18) [24].

Таблиця 6.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1.	Формування енергоефективних маршрутів	Економія витрат електроенергії	Зменшення кількості витрат енергії, щоб дістатись від точки А до точки Б.
2.	Відображення місцезнаходження зарядних станцій	Є важливим фактором планування довгих маршрутів	Відображає зарядні станції близькі до маршруту

Після цього розроблена трирівнева маркетингова модель товару, що включає в себе: уточнення ідеї продукту та/або послуги, його фізичні складові, особливості процесу його надання (табл. 6.19).

Таблиця 6.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Мобільний застосунок для формування енергоефективних маршрутів для електромобілів		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Визначення найбільш енергоефективного маршруту	М	Тх
	2. Відображення місцезнаходження зарядних станцій впродовж маршруту.	М	Тх

Продовження таблиці 6.19

	Якість: безпечний, простий інтерфейс
	Пакування: мобільний застосунок, доступний з Play Market
	Марка: Driveco
III. Товар із підкріпленням	До продажу: не потребує особливих вимог
	Після продажу: не потребує особливих вимог
За рахунок чого потенційний товар буде захищено від копіювання: спеціальними засоби шифрування вихідного коду	

Наступним кроком стало визначення цінових меж (табл. 6.20).

Таблиця 6.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1.	Відсутні	1200 грн/місяць	Від 13 тис. грн	0-160 грн/місяць

Наступним кроком визначено оптимальну систему збуту, в межах якої приймається рішення (табл. 6.21) [25].

Таблиця 6.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Закупівля відбувається через інтернет	- пошук користувачів - статистика - реклама	Канал нульового рівня	Через інтернет

Останньою для складової маркетингової програми розроблено концепцію маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 6.22).

Таблиця 6.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1.	Клієнти дізнаються про новий функціонал та удосконаленням завдяки рекламі	Соціальні мережі, відгуки в продуктових маркетах	Інтернет маркетинг	Заохотити користувача через повідомлення про захист довкілля	Допомогти збереженню довкіллю

Висновки до розділу 6

В результаті проведення аналізу розробки стартап-проєкту було визначено головну ідею, а саме створення мобільного застосунку для формування енергоефективних маршрутів.

Визначено технології Spring Boot і Kotlin Multiplatform для реалізації ідеї та зазначено сприятливі умови для ринкової комерціалізації проекту.

У результаті визначення базової стратегії вибрано стратегію спеціалізації. Стратегія охоплення ринку – заняття конкурентної ніші, яка є унікальною на ринку.

Цільовою аудиторією визначено користувачів електромобілів.

В результаті, визначено, що продукт має високий потенціал для успішного стартапу, а тому вартує подальшої імплементації.

ВИСНОВКИ

В результаті виконання роботи створено систему, що за допомогою аналізу вхідних даних та застосування їх до алгоритму обчислення витраченої енергії, формує маршрут, який є найбільш енергоефективним та відображає його користувачу через мобільний застосунок.

Проведений аналіз існуючих рішень побудови маршрутів. Виявлено, що наразі розглянуті аналогічні системи використовують час, як основний параметр для формування оптимального маршруту. В той час як цінність розробленої системи полягає в використання параметру енерговитратності електромобіля на заданому маршруті.

Проаналізовано існуючі дослідження визначення енерговитратності електромобіля, за основу було взято квазі-статичну модель руху машини. В результаті розроблений алгоритм знаходження найбільш енергоефективного маршруту. Обчислення є достатньо точними з невисокою обчислювальною вартістю.

Проаналізовано дані, необхідні для обчислень, а саме: дані специфікацій автомобілів, динамічні дані про погоду, затори, дані про автомобільні дороги (покриття, світлофори).

Розроблено систему передачі результатів роботи системи на мобільний застосунок. Система побудована за допомогою сучасних і надійних технологій.

Розроблено мобільний застосунок для зручної взаємодії користувача з системою. Мобільний застосунок складається з мінімальної кількості екранів, що полегшує його використання. Мобільний застосунок розроблений на системі Android, проте врахована подальша розробка для інших систем, таких як IOS.

Результати роботи системи є таким, що дозволяють економити енерговитрати на маршруті від точки А до точки Б. В результаті порівняння з існуючими системами, визначено, що енергоефективний маршрут не завжди є найшвидшим, проте він дозволяє проїхати його з найменшою кількістю витрат.

Таким чином, користувач може економити на заряджанні електричного транспортного засобу, скорочуючи свої витрати. Більш того, це допомагає зменшити кількість необхідної видобувної електроенергії, тим самим зменшити кількість шкідливих викидів в атмосферу.

Розроблено стратегію реалізації і комерціалізації стартап-проєкту розробленої системи. Визначено, що умови для цього є сприятливими, а отже необхідно продовжувати імплементацію програмного забезпечення. Першим необхідним кроком має стати розробка застосунку під платформу IOS для заохочення більшої кількості користувачів, а також важливо шукати покращення алгоритму обчислення енерговитратності.

Програмний продукт створено в рамках міжнародного проекту Інструментальні засоби підвищення енергоефективності керування транспортними засобами (спільно з Університетом Малаги, Королівство Іспанія)

Список використаних джерел

1. Основні показники охорони навколишнього природного середовища м. Києва / Головне управління статистики у м. Києві. Київ: 2016.
2. Євгеній Скрибка. Електромобілі 2021: знову подвоїлись [Електронний ресурс]. 2022. URL: https://www.nefterynok.info/stati/elektromobl-2021-znovu-podvolis
3. M. Daigle, K. Goebel. A model-based prognostics approach applied to pneumatic valves // *International Journal of Prognostics and Health Management*, ISSN 2153-2648, 2011.
4. Javier A. Oliva, Christoph Weihrauch, Torsten Bertram. Model-Based Remaining Driving Range Prediction in Electric Vehicles by using Particle Filtering and Markov Chains // *World Electric Vehicle Journal* Vol. 6 - ISSN 2032-6653 - © 2013. p.0204
5. Вакуленко М. О. Тлумачний словник із фізики: [6644 статті] / М. О. Вакуленко, О. В. Вакуленко. К. : Видавничо-поліграфічний центр «Київський університет», 2008. 767 с.
6. Morten Lybech Thøgersen, M.Sc. Modelling of the Variation of Air Density with Altitude through Pressure, Humidity and Temperature // *EMD International A/S*, 2005.
7. Агейкин Я.С. Проходимость автомобилей. – М.: Машиностроение, 1981. 232 с.
8. Guzzella, L., Sciarretta, A. Vehicle propulsion systems: Introduction to modeling and optimization // Springer Verlag, Heildelberg. 2005.
9. Marshall Brain, Cherise Threewitt. How Manual Transmissions Work [Електронний ресурс]. 2021. URL: https://auto.howstuffworks.com/transmission.htm
10. Javier A. Oliva, Torsten Bertram. Adaptive Driving Situation Characterization for Predicting the Driving Load of Electric Vehicles in

Uncertain Environments. European conference of the prognostics and health management society, 2014.

11. [Електронний ресурс]. URL: https://www.google.com/maps/about/
12. [Електронний ресурс]. URL: https://support.google.com/waze/answer/11942300?hl=en-GB
13. Андрей Бреслав. Язык программирования Kotlin // Открытые системы. 2011. № 09.
14. Кларенс Хо, Роб Харроп. Spring 3 для профессионалов. — М.: «Вильямс», 2012. 880 с. ISBN 978-5-8459-1803-1.
15. Banker, Kyle (March 28, 2011), MongoDB in Action (1st ed.), Manning, pp. 375, ISBN 978-1-935182-87-0
16. “Kotlin for cross-platform mobile development”. JetBrains: Developer Tools for Professionals and Teams. Retrieved 20 August 2020.
17. Dawn Griffiths. Headfirst Android Development. 1st edition, O'Reilly, 2015
18. [Електронний ресурс]. URL: https://developer.android.com/jetpack/compose
19. Erik Wilde, Cesare Pautasso. REST: From Research to Practice. — Springer Science & Business Media, 2011. 528 p. ISBN 978-1-4419-8303-9.
20. Davis, Ian. “What Are The Benefits of MVC?”. Internet Alchemy. 2016.
21. Pete Weissbrod. "Model–View–ViewModel Pattern for WPF: Yet another approach". 2008.
22. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. Київ : НТУУ «КПІ», 2016. 28 с.
23. Бланк, С. Стартап. Настольная книга основателя / С. Бланк, Б. Дорф ; пер. с англ. Т. Гутман, И. Окунькова, Е. Бакушева. 2-е изд. Москва: АльпинаПаблишер, 2014. 614 с.

24. Рогова Е.М., Ткаченко Е.А., Фияксель Э.А. Венчурный менеджмент. М.:Издательство: Высшая Школа Экономики, 2011 г. 500 с. / ISBN: 978-5-7598-0746-9.
25. Antikarov V., Copeland T. Real Options: A Practitioner's Guide. Texere LLC Publishing, New York, 2001.

ДОДАТОК А

Програмний код

Інструментальні засоби формування енергоефективних маршрутів для електромобілів

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_НН ІАТЕ_ІПЗЕ_ ТВ-12мп

Аркушів 15

2022

PowerCalculatorService.kt

```
package com.bortnichuk.drivecoback.model.service.calculator

import com.bortnichuk.drivecoback.model.entity.db.Car
import com.bortnichuk.drivecoback.model.entity.domain.Location
import com.bortnichuk.drivecoback.model.entity.response.weather.Weather
import com.bortnichuk.drivecoback.model.repository.CarRepository
import com.bortnichuk.drivecoback.model.repository.MotorEfficiencyRepository
import com.bortnichuk.drivecoback.model.service.api.WeatherApiClient
import org.springframework.beans.factory.annotation.Autowired
import org.springframework.data.repository.findByIdOrNull
import org.springframework.stereotype.Service
import kotlin.math.pow
import kotlin.math.sin

@Service
class PowerCalculatorService constructor(
        private val motorEfficiencyRepository: MotorEfficiencyRepository,
        private val carRepository: CarRepository,
        private val weatherApiClient: WeatherApiClient
) {

    private companion object {
        const val KELVIN = 273.15
        const val FREE_FALL_ACC = 9.81

        const val INCH = 2.54

        const val GEAR_RATIO_ACC = 2.1
        const val GEAR_RATIO = 0.9

        const val averageTurnSpeed = 5.5
        const val stopTime = 5
    }
```

```
fun calculate(start: Location, carId: Int, roadStepList: List<RoadStep>): Double {

    val weather = weatherApiClient.getWeatherByLocation(start)
    val airDensity = weather?.let {
        airDensity(it.main.pressure, it.main.tempKelvin.toCelsius, it.main.humidity)
    } ?: 0.0

    val car = carRepository.findByIdOrNull(carId)

    val motorList = roadStepList.map {
        val mecPower = calculateMechanicalPower(it.speed, airDensity, car)
        PowerItem(
            electricalPowerMotor(
                mecPower,
                getEfficiency(
                    getForce(it.speed, airDensity, car),
                    it.speed,
                    car?.tireRadius ?: 0.0
                )
            ),
            it.time
        )
    }

    val generatorList = roadStepList.flatMap {

        val list = mutableListOf<PowerItem>()
        if (it.turn) {
            list.add(
                getGeneratorPowerItem(it, airDensity, car)
            )
        }
        for (i in 0..it.stops) {
```

```
                list.add(
                        getGeneratorPowerItem(it, airDensity, car)
                )
            }
            list
        }

        return motorList.powerTypePerformance - generatorList.powerTypePerformance
    }

    private fun getGeneratorPowerItem(roadStep: RoadStep, airDensity: Double, car: Car?): PowerItem
{
        val speed = (roadStep.speed - averageTurnSpeed) / 2
        return PowerItem(
                electricalPowerGenerator(
                        calculateMechanicalPower(speed, airDensity, car),
                        getEfficiency(getForce(speed, airDensity, car), speed, car?.tireRadius ?: 0.0),
                        speed
                ),
                stopTime
        )
    }

    private val List<PowerItem>.powerTypePerformance: Double
        get() = sumOf { it.power } * sumOf { it.time }

    private fun calculateMechanicalPower(speed: Double, airDensity: Double, car: Car?): Double {
        return getForce(speed, airDensity, car) * speed
    }

    private fun electricalPowerMotor(mecPower: Double, efficiency: Double) = mecPower / efficiency

    private fun electricalPowerGenerator(mecPower: Double, efficiency: Double, speed: Double) =
            mecPower * (2 - 1 / efficiency) * speed.generatorModeCoefficient
```

```
    private fun getEfficiency(force: Double, speed: Double, tireRadius: Double): Double {
        val torque = force * tireRadius / GEAR_RATIO
        val rotationalSpeed = speed * GEAR_RATIO / tireRadius

        return motorEfficiencyRepository.efficiency(torque, rotationalSpeed)
    }

    private fun getForce(speed: Double, airDensity: Double, car: Car?, acceleration: Double = 0.0):
Double {
        car ?: return 0.0
        val slope = 15 * Math.PI / 180
        val rollingResistanceCoefficient = RoadType.ASPHALT.value

        return 0.5 * airDensity * car.dragCoefficient * car.width * speed.pow(2) +
                car.weight * FREE_FALL_ACC * sin(slope) +
                car.weight * FREE_FALL_ACC * rollingResistanceCoefficient +
                car.weight * acceleration
    }

    private fun airDensity(pressure: Double, temperature: Double, humidity: Double): Double {

        val temperatureCelsius = temperature.toCelsius

        val dryAirGasConstant = 287.05
        val waterVaporGasConstant = 461.495

        val waterVaporPressure = humidity * saturatedVaporPressure(temperatureCelsius)
        val dryPressure = pressure - waterVaporPressure

        return (dryPressure / (dryAirGasConstant * temperature)) +
                waterVaporPressure / (waterVaporGasConstant * temperatureCelsius)
    }
```

```
private fun saturatedVaporPressure(temperature: Double): Double {
    val c0 = 0.99999683
    val c1 = -0.90826951 * 10.0.pow(-2)
    val c2 = 0.78736169 * 10.0.pow(-4)
    val c3 = -0.61117958 * 10.0.pow(-6)
    val c4 = 0.43884187 * 10.0.pow(-8)
    val c5 = -0.29883885 * 10.0.pow(-10)
    val c6 = 0.21874425 * 10.0.pow(-12)
    val c7 = -0.17892321 * 10.0.pow(-14)
    val c8 = 0.11112018 * 10.0.pow(-16)
    val c9 = -0.30994571 * 10.0.pow(-19)
    val p = c0 + temperature * (c1 +
            temperature * (c2 +
            temperature * (c3 +
            temperature * (c4 +
            temperature * (c5 +
            temperature * (c6 +
            temperature * (c7 +
            temperature * (c8 +
            temperature * c9))))))))
    return 6.1078 / p.pow(8)
}

private val Double.toCelsius
    get() = this - KELVIN

private val Double.toMetres
    get() = this * INCH / 10

private val Double.generatorModeCoefficient: Double
    get() = if (this <= 3.5) {
        0.0
    } else if (this > 3.5 && this < 8) {
```

```
            (this - 3.5) / 5
        } else {
            0.9
        }
}
```

MapScreen.kt

```
package com.bortnichuk.driveco.android.ui.map

import android.Manifest.permission.ACCESS_COARSE_LOCATION
import android.Manifest.permission.ACCESS_FINE_LOCATION
import android.content.Context
import android.content.pm.PackageManager
import android.graphics.Bitmap
import android.graphics.BitmapFactory
import android.graphics.drawable.BitmapDrawable
import androidx.activity.compose.ManagedActivityResultLauncher
import androidx.activity.compose.rememberLauncherForActivityResult
import androidx.activity.result.contract.ActivityResultContracts
import androidx.compose.foundation.background
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardActions
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.material.*
import androidx.compose.runtime.*
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.platform.LocalContext
import androidx.compose.ui.platform.LocalFocusManager
import androidx.compose.ui.text.input.ImeAction
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.style.TextAlign
```

```
import androidx.compose.ui.unit.dp
import androidx.core.content.ContextCompat
import androidx.core.graphics.drawable.toBitmap
import com.bortnichuk.driveco.android.R
import com.google.android.gms.maps.CameraUpdateFactory
import com.google.android.gms.maps.model.BitmapDescriptorFactory
import com.google.android.gms.maps.model.CameraPosition
import com.google.android.gms.maps.model.LatLng
import com.google.maps.android.compose.*

@OptIn(ExperimentalMaterialApi::class)
@Composable
fun MapScreen(
    uiState: MapUIState,
    onRequestPermissionUpdate: () -> Unit,
    onGetDestinationLocation: (LatLng) -> Unit,
    onBuildRoutes: () -> Unit,
    onUpdateBatteryCapacity: (Double) -> Unit,
    onShowNearestChargeStations: (List<LatLng>) -> Unit
) {

    val kyiv = LatLng(50.4501, 30.5234)

    val cameraPositionState = rememberCameraPositionState {
        position = CameraPosition.fromLatLngZoom(kyiv, 10f)
    }

    val mapProperties by remember { mutableStateOf(MapProperties()) }

    uiState.currentLocation?.let {
        LaunchedEffect(it) {
            val cameraPosition =
                CameraPosition.fromLatLngZoom(LatLng(it.latitude, it.longitude), 10f)
            cameraPositionState.animate(
```

```
            CameraUpdateFactory.newCameraPosition(cameraPosition),
            1_000
        )
    }
}

val launcher = rememberLauncherForActivityResult(
    contract = ActivityResultContracts.RequestMultiplePermissions(),
) { permissions ->
    when {
        permissions.getOrDefault(ACCESS_FINE_LOCATION, false) -> {
            onRequestPermissionUpdate()
        }
        permissions.getOrDefault(ACCESS_COARSE_LOCATION, false) -> {
            // do nothing
        }
        else -> {
            // do nothing
        }
    }
}

val context = LocalContext.current

val scaffoldState = rememberBottomSheetScaffoldState()

BottomSheetScaffold(
    sheetContent = { BottomSheetContent(uiState, onUpdateBatteryCapacity) },
    sheetPeekHeight = 32.dp,
    scaffoldState = scaffoldState
) {
    Box(modifier = Modifier.fillMaxSize(), contentAlignment = Alignment.Center) {
        Map(
            context,
```

```
                uiState,
                launcher,
                cameraPositionState,
                mapProperties,
                onRequestPermissionUpdate,
                onGetDestinationLocation
            )
            uiState.endLocation?.let {
                if (uiState.downloadingStatus == DownloadingStatus.NOT_STARTED)
                    Button(
                        modifier = Modifier
                            .padding(bottom = 32.dp)
                            .align(Alignment.BottomCenter),
                        onClick = onBuildRoutes
                    ) {
                        Text(text = "Build route")
                    }
            }
            BuildingRoute(uiState)
            RouteInfo(uiState, scaffoldState, onShowNearestChargeStations)
        }
    }
}

@Composable
fun BottomSheetContent(
    uiState: MapUIState,
    onUpdateBatteryCapacity: (Double) -> Unit
) {

    var batteryCapacitySearchInput: String by remember {
        mutableStateOf("")
    }
```

```
val focusManager = LocalFocusManager.current

Box(
    Modifier
        .fillMaxWidth()
        .padding(8.dp),
    contentAlignment = Alignment.Center
) {
    Text("Swipe up to see your car specifications")
}
Column(
    modifier = Modifier
        .fillMaxWidth()
        .padding(start = 16.dp, top = 16.dp, end = 16.dp, bottom = 32.dp)
) {
    Text(
        modifier = Modifier.padding(bottom = 16.dp),
        text = "You are driving: ${uiState.car?.name ?: ""}"
    )
    Text(
        text = "- weight: ${uiState.car?.weight ?: ""} kg\n" +
                "- length: ${uiState.car?.length ?: ""} m\n" +
                "- height: ${uiState.car?.height ?: ""} m\n" +
                "- width: ${uiState.car?.width ?: ""} m"
    )
    Row(
        horizontalArrangement = Arrangement.Center,
        verticalAlignment = Alignment.CenterVertically
    ) {
        Text(text = "- battery state of charge: ")
        uiState.car?.batteryCapacity?.let {
            Text(text = it.toString())
        } ?: TextField(
            value = batteryCapacitySearchInput,
```

```
            label = { Text(text = "Enter in kWh") },
            keyboardOptions = KeyboardOptions(
                keyboardType = KeyboardType.Decimal,
                imeAction = ImeAction.Done
            ),
            keyboardActions = KeyboardActions(
                onDone = {
                    focusManager.clearFocus()
                    onUpdateBatteryCapacity(batteryCapacitySearchInput.toDouble())
                }
            ),
            onValueChange = { batteryCapacitySearchInput = it },
            shape = RoundedCornerShape(8.dp),
            singleLine = true,
            colors = TextFieldDefaults.textFieldColors(
                focusedIndicatorColor = Color.Transparent,
                unfocusedIndicatorColor = Color.Transparent,
                disabledIndicatorColor = Color.Transparent
            )
        )
        Text(text = " kWh")
    }

  }
}

@Composable
fun Map(
  context: Context,
  uiState: MapUIState,
  launcher: ManagedActivityResultLauncher<Array<String>, Map<String, @JvmSuppressWildcards
Boolean>>,
  cameraPositionState: CameraPositionState,
  mapProperties: MapProperties,
```

```
    onRequestPermissionUpdate: () -> Unit,
    onGetDestinationLocation: (LatLng) -> Unit,
) {
    GoogleMap(
        modifier = Modifier
            .fillMaxSize()
            .padding(bottom = 16.dp),
        cameraPositionState = cameraPositionState,
        onMapLoaded = {
            when (PackageManager.PERMISSION_GRANTED) {
                ContextCompat.checkSelfPermission(context, ACCESS_FINE_LOCATION),
                ContextCompat.checkSelfPermission(context, ACCESS_COARSE_LOCATION) -> {
                    onRequestPermissionUpdate()
                }
                else -> {
                    launcher.launch(arrayOf(ACCESS_FINE_LOCATION,
ACCESS_COARSE_LOCATION))
                }
            }
        },
        onMapLongClick = {
            onGetDestinationLocation(it)
        },
        properties = mapProperties.copy(isMyLocationEnabled = uiState.currentLocation != null),
    ) {

        uiState.endLocation?.let {
            Marker(
                state = MarkerState(position = it),
            )
        }

        when (uiState) {
            is MapUIState.HasRoutes -> HasRoutes(uiState = uiState, context = context)
```

```
            else -> Unit
        }
    }
}

@Composable
fun HasRoutes(
    uiState: MapUIState.HasRoutes,
    context: Context
) {
    uiState.routes.routes.sortedByDescending { it.estimatedConsumedEnergy }
        .forEach {
            val properties = if (it.isTheMostEfficient) {
                Color.Blue to 13f
            } else {
                Color.Gray to 10f
            }

            Polyline(
                points = it.points.map { point ->
                    LatLng(
                        point.latitude.toDouble(),
                        point.longitude.toDouble()
                    )
                },
                color = properties.first,
                width = properties.second
            )
        }
    if (uiState is MapUIState.HasChargingStations) {

        val bitmap = BitmapFactory.decodeResource(context.resources, R.drawable.ic_charging_station)
        val scaled = Bitmap.createScaledBitmap(bitmap, 75, 100, true)
```

```
        uiState.chargingStations.forEach {
            Marker(
                state = MarkerState(it),
                icon = BitmapDescriptorFactory.fromBitmap(scaled)
            )
        }
    }
}
@Composable
fun BuildingRoute(
    uiState: MapUIState
) {
    if (uiState.downloadingStatus == DownloadingStatus.STARTED) {
        Box(
            modifier = Modifier
                .fillMaxSize()
                .background(color = Color.White.copy(alpha = 0.7f)),
            contentAlignment = Alignment.Center
        ) {
            Text(text = "Searching for the best route...", textAlign = TextAlign.Center)
        }
    }
}
@OptIn(ExperimentalMaterialApi::class)
@Composable
fun RouteInfo(
    uiState: MapUIState,
    scaffoldState: BottomSheetScaffoldState,
    onShowNearestChargeStations: (List<LatLng>) -> Unit
) {
    if (uiState.downloadingStatus == DownloadingStatus.FINISHED && uiState is
MapUIState.HasRoutes) {
        val bestRoute = uiState.routes.routes.first { it.isTheMostEfficient }
        LaunchedEffect(uiState.routes) {
```

```
            val result = scaffoldState.snackbarHostState.showSnackbar(
                message = "Estimated Consumed Energy for the best route: " +
                        "%.${2}f kWh".format(bestRoute.estimatedConsumedEnergy) +
                        "\nOther routes: \n" +
                        uiState.routes.routes.filterNot { it.isTheMostEfficient }.joinToString(
                            separator = "\n"
                        ) {
                            "- %.${2}f kWh".format(it.estimatedConsumedEnergy)
                        },
                actionLabel = "Ok",
                duration = SnackbarDuration.Indefinite
            )
            if (result == SnackbarResult.ActionPerformed) {
                if (uiState.isBatteryCapacityExtended) {
                    val result1 = scaffoldState.snackbarHostState.showSnackbar(
                        message = "Caution! Your current battery capacity (${uiState.car?.batteryCapacity})" +
                                " is lower then estimated consumed energy for the best route! \n" +
                                "Tap on 'Show' to show the nearest charging stations across the route.",
                        actionLabel = "Show",
                        duration = SnackbarDuration.Indefinite
                    )
                    if (result1 == SnackbarResult.ActionPerformed) {
                        onShowNearestChargeStations.invoke(
                            bestRoute.points.filter { bestRoute.points.indexOf(it) % 45 == 0 }
                                .map {
                                    LatLng(it.latitude.toDouble(), it.longitude.toDouble())
                                }
                        )
                    }
                }
            }
        }
    }
}
```

ДОДАТОК Б

Таблиця ефективності електродвигуна

Інструментальні засоби формування енергоефективних маршрутів для електромобілів

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_НН ІАТЕ_ІПЗЕ_ ТВ-12мп

Аркушів 4

2022

T ω	100	465	1000	1562	2000	2659	3000	3390
0	44.51	54.73	58.28	59.84	60.81	62.72	63.35	64.3
1	45.37	54.18	57.62	59.55	60.58	62.51	63.16	64.18
2	45.24	55.66	60.42	62.43	63.37	64.92	65.75	66.75
3	45.12	57.14	63.22	65.31	66.17	67.34	68.34	69.32
4	44.64	58.36	66.01	68.39	69.1	70.24	71.23	72.19
5	44.15	59.58	68.79	71.48	72.05	73.16	74.13	75.07
6	43.47	60.11	70.47	73.45	74.02	75.18	76.15	77.26
7	42.77	60.55	71.99	75.26	75.85	77.09	78.05	79.35
8	41.98	60.59	72.75	76.27	76.95	78.3	79.31	80.73
9	41.17	60.53	73.32	77.07	77.85	79.34	80.41	81.93
10	40.35	60.31	73.61	77.56	78.45	80.09	81.23	82.72
11	39.52	60.03	73.78	77.92	78.93	80.72	81.95	83.34
12	38.72	59.72	73.89	78.22	79.32	81.23	82.49	83.75
13	37.93	59.4	73.95	78.46	79.65	81.66	82.92	84.02
14	37.18	59.08	73.97	78.67	79.94	82.02	83.24	84.19
15	36.48	58.76	73.97	78.84	80.18	82.31	83.45	84.26
16	35.81	58.41	73.92	78.96	80.38	82.54	83.62	84.28
17	35.17	58.03	73.83	79.04	80.53	82.72	83.72	84.23
18	34.52	57.63	73.7	79.08	80.65	82.87	83.79	84.14
19	33.87	57.18	73.52	79.06	80.7	82.96	83.83	83.99
20	33.22	56.72	73.32	79.03	80.74	83.04	83.83	83.82
21	32.56	56.23	73.06	78.94	80.73	83.05	83.78	83.58
22	31.9	55.72	72.79	78.83	80.7	83.05	83.7	83.33
23	31.25	55.2	72.48	78.69	80.63	82.99	83.54	82.98
24	30.6	54.67	72.16	78.53	80.55	82.93	83.37	82.63
25	29.97	54.13	71.81	78.33	80.43	82.8	83.13	82.24
26	29.34	53.59	71.46	78.13	80.3	82.67	82.89	81.84
27	28.75	53.05	71.07	77.89	80.14	82.49	82.61	81.45
28	28.16	52.52	70.68	77.66	79.98	82.3	82.32	81.06
29	27.62	52	70.26	77.39	79.79	82.1	82	80.62
30	27.08	51.49	69.83	77.11	79.61	81.89	81.68	80.18
31	26.57	50.95	69.36	76.82	79.4	81.66	81.32	79.71
32	26.06	50.39	68.89	76.53	79.19	81.43	80.95	79.24
33	25.55	49.82	68.44	76.23	78.97	81.17	80.58	78.84
34	25.04	49.23	67.99	75.93	78.74	80.9	80.22	78.45
35	24.53	48.66	67.56	75.62	78.5	80.59	79.85	78.15
36	24.02	48.09	67.13	75.32	78.25	80.26	79.48	77.89
37	23.51	47.55	66.7	75.01	78	79.88	79.03	NaN
38	23.01	47.03	66.26	74.7	77.74	79.46	78.52	NaN
39	22.52	46.52	65.81	74.38	77.46	78.99	77.97	NaN
40	22.03	46.02	65.35	74.03	77.15	78.47	77.36	NaN
41	21.55	45.51	64.87	73.66	76.81	78.02	76.89	NaN
42	21.09	44.98	64.36	73.26	76.42	77.69	NaN	NaN
43	20.63	44.45	63.85	72.84	76.01	77.37	NaN	NaN
44	20.18	43.89	63.32	72.39	75.57	77.04	NaN	NaN

45	19.73	43.33	62.78	71.93	75.12	76.72	NaN	NaN
46	19.28	42.75	62.2	71.43	74.6	76.37	NaN	NaN
47	18.82	42.16	61.63	70.93	74.11	NaN	NaN	NaN
48	18.36	41.54	61.04	70.5	NaN	NaN	NaN	NaN
49	17.91	40.93	60.45	70.07	NaN	NaN	NaN	NaN
50	17.46	40.29	59.9	NaN	NaN	NaN	NaN	NaN
51	17.02	39.65	59.34	NaN	NaN	NaN	NaN	NaN
52	16.6	39.01	58.74	NaN	NaN	NaN	NaN	NaN
53	16.17	38.38	58.14	NaN	NaN	NaN	NaN	NaN
54	15.68	37.69	57.48	NaN	NaN	NaN	NaN	NaN
55	15.16	37	56.82	NaN	NaN	NaN	NaN	NaN

T ω	**4000**	**4486**	**5000**	**5583**	**6000**	**6679**	**7000**	**7411**
0	64.83	60.61	45.56	42.93	49.5	55.89	56.86	58.56
1	64.97	61.66	50.38	47.87	53.21	58.66	59.72	60.99
2	67.64	66.16	58.96	56.66	60.34	64.4	65.32	65.9
3	70.31	70.66	67.54	65.46	67.48	70.14	70.92	70.82
4	73.54	74.43	72.81	71.55	72.74	74.68	75.24	75.04
5	76.79	78.17	77.97	77.55	77.94	79.18	79.51	79.25
6	79.03	80.28	80.32	80.4	80.69	81.6	81.77	81.53
7	81.14	82.17	82.3	82.82	83.1	83.73	83.77	83.54
8	82.4	83.14	83.21	83.86	84.22	84.66	84.61	84.36
9	83.44	83.87	83.84	84.53	84.99	85.27	85.15	84.87
10	83.99	84.19	84.17	84.84	85.29	85.39	85.19	84.8
11	84.33	84.34	84.38	85	85.38	85.31	85.01	84.48
12	84.44	84.31	84.48	85.11	85.36	85.05	84.64	83.99
13	84.41	84.16	84.53	85.19	85.28	84.69	84.16	83.39
14	84.27	83.9	84.5	85.21	85.11	84.19	83.56	82.74
15	84.04	83.54	84.39	85.18	84.89	83.58	82.85	82.04
16	83.74	83.06	84.15	85.09	84.55	82.92	82.13	81.34
17	83.33	82.43	83.74	84.92	84.08	82.18	81.41	80.66
18	82.88	81.72	83.33	84.68	83.54	81.46	80.72	NaN
19	82.35	80.84	82.93	84.32	82.85	80.78	NaN	NaN
20	81.78	80.05	82.57	83.9	82.2	NaN	NaN	NaN
21	81.14	79.47	82.3	83.33	81.64	NaN	NaN	NaN
22	80.5	78.91	82.09	82.78	81.12	NaN	NaN	NaN
23	79.9	78.41	82.1	82.34	NaN	NaN	NaN	NaN
24	79.31	77.91	82.02	81.92	NaN	NaN	NaN	NaN
25	78.87	77.41	80.96	NaN	NaN	NaN	NaN	NaN
26	78.44	76.91	NaN	NaN	NaN	NaN	NaN	NaN
27	78.22	NaN	NaN	NaN	NaN	NaN	NaN	NaN
28	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

33	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
34	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
35	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
36	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
37	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
38	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
39	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
40	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
41	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
42	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
43	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
44	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
45	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
46	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
47	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
48	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
49	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
51	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
52	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
53	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
54	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
55	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

T ω	**8000**	**8507**	**9000**	**9604**	**10000**	**10700**	**11000**
0	60.62	61.61	63.11	64.14	64.98	66.24	66.22
1	62.87	63.95	64.86	65.95	66.78	67.66	67.81
2	67.71	68.75	68.93	70.15	70.89	71.24	71.48
3	72.55	73.55	72.99	74.35	75.01	74.81	75.15
4	76.3	77.04	76.44	77.37	77.76	77.37	77.53
5	80.01	80.49	79.87	80.35	80.46	79.89	79.87
6	81.94	82.09	81.5	81.62	81.52	80.84	80.69
7	83.63	83.44	82.88	82.64	82.35	81.57	81.31
8	84.16	83.76	83.16	82.68	82.26	81.3	80.92
9	84.38	83.79	83.14	82.45	81.93	80.77	80.28
10	84.07	83.41	82.73	81.83	81.17	79.8	79.19
11	83.54	82.85	82.15	81.05	80.24	78.65	77.92
12	82.91	82.18	81.44	80.17	79.24	NaN	NaN
13	82.23	81.44	80.64	79.25	NaN	NaN	NaN
14	81.56	80.72	79.84	NaN	NaN	NaN	NaN
15	80.91	80.01	NaN	NaN	NaN	NaN	NaN
16	80.26	NaN	NaN	NaN	NaN	NaN	NaN
17	NaN	NaN	NaN	NaN	NaN	NaN	NaN
18	NaN	NaN	NaN	NaN	NaN	NaN	NaN
19	NaN	NaN	NaN	NaN	NaN	NaN	NaN

20	NaN	NaN	NaN	NaN	NaN	NaN	NaN
21	NaN	NaN	NaN	NaN	NaN	NaN	NaN
22	NaN	NaN	NaN	NaN	NaN	NaN	NaN
23	NaN	NaN	NaN	NaN	NaN	NaN	NaN
24	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25	NaN	NaN	NaN	NaN	NaN	NaN	NaN
26	NaN	NaN	NaN	NaN	NaN	NaN	NaN
27	NaN	NaN	NaN	NaN	NaN	NaN	NaN
28	NaN	NaN	NaN	NaN	NaN	NaN	NaN
29	NaN	NaN	NaN	NaN	NaN	NaN	NaN
30	NaN	NaN	NaN	NaN	NaN	NaN	NaN
31	NaN	NaN	NaN	NaN	NaN	NaN	NaN
32	NaN	NaN	NaN	NaN	NaN	NaN	NaN
33	NaN	NaN	NaN	NaN	NaN	NaN	NaN
34	NaN	NaN	NaN	NaN	NaN	NaN	NaN
35	NaN	NaN	NaN	NaN	NaN	NaN	NaN
36	NaN	NaN	NaN	NaN	NaN	NaN	NaN
37	NaN	NaN	NaN	NaN	NaN	NaN	NaN
38	NaN	NaN	NaN	NaN	NaN	NaN	NaN
39	NaN	NaN	NaN	NaN	NaN	NaN	NaN
40	NaN	NaN	NaN	NaN	NaN	NaN	NaN
41	NaN	NaN	NaN	NaN	NaN	NaN	NaN
42	NaN	NaN	NaN	NaN	NaN	NaN	NaN
43	NaN	NaN	NaN	NaN	NaN	NaN	NaN
44	NaN	NaN	NaN	NaN	NaN	NaN	NaN
45	NaN	NaN	NaN	NaN	NaN	NaN	NaN
46	NaN	NaN	NaN	NaN	NaN	NaN	NaN
47	NaN	NaN	NaN	NaN	NaN	NaN	NaN
48	NaN	NaN	NaN	NaN	NaN	NaN	NaN
49	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50	NaN	NaN	NaN	NaN	NaN	NaN	NaN
51	NaN	NaN	NaN	NaN	NaN	NaN	NaN
52	NaN	NaN	NaN	NaN	NaN	NaN	NaN
53	NaN	NaN	NaN	NaN	NaN	NaN	NaN
54	NaN	NaN	NaN	NaN	NaN	NaN	NaN
55	NaN	NaN	NaN	NaN	NaN	NaN	NaN

ДОДАТОК В

Презентація

Інструментальні засоби формування енергоефективних маршрутів для електромобілів

УКР.НТУУ«КПІ ім. Ігоря Сікорського»_НН ІАТЕ_ІПЗЕ_ ТВ-12мп

Аркушів 16

2022

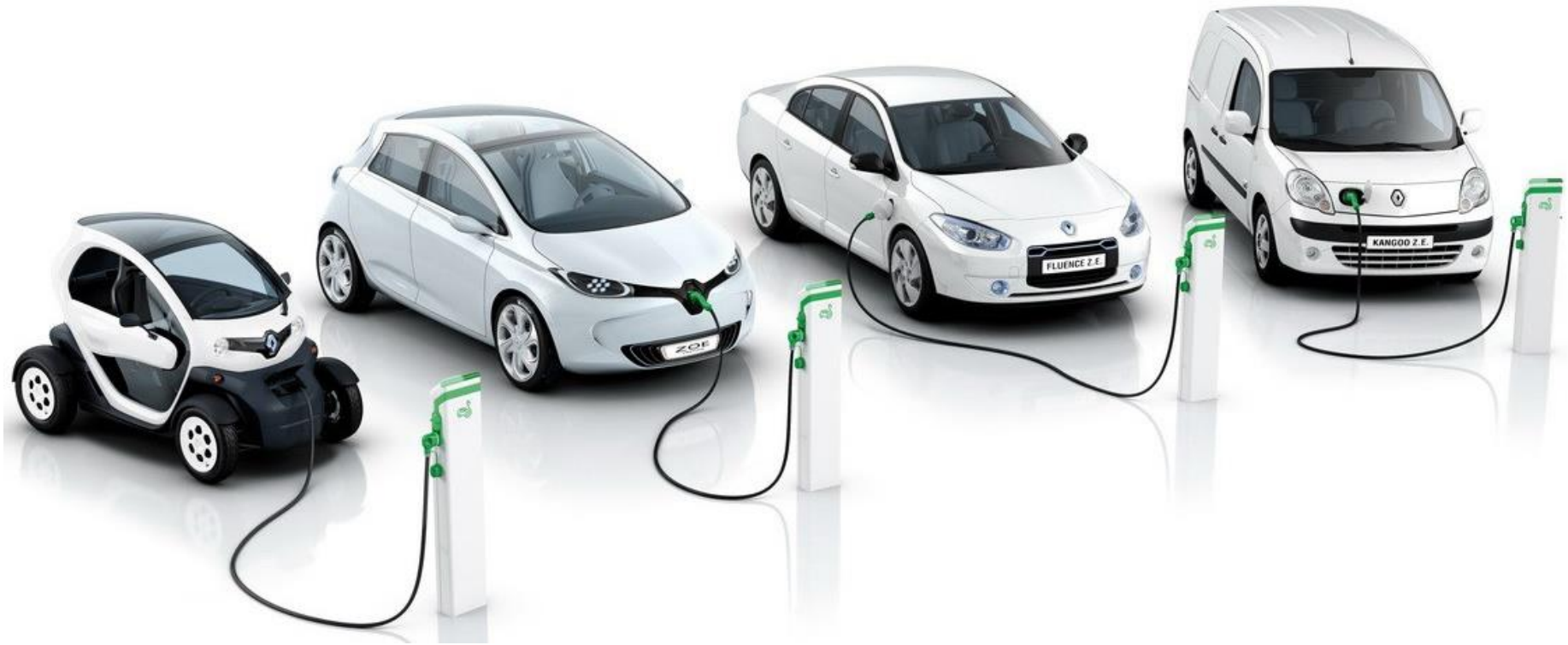
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці

Інструментальні засоби формування енергоефективних маршрутів для електромобілів

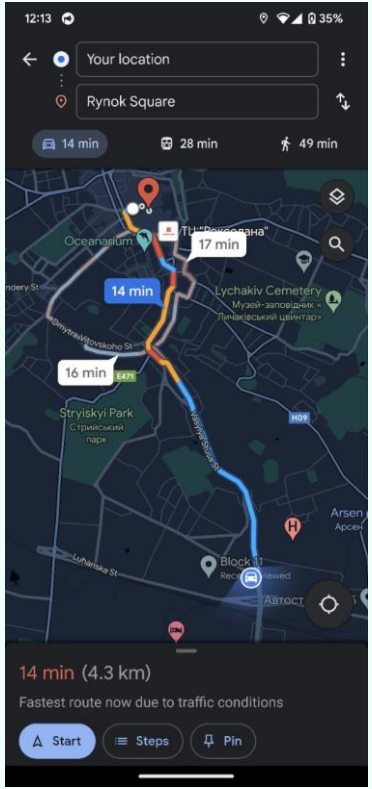
Виконав: студент групи ТВ-12мп
Бортнічук Нікіта Олександрович

Керівник: к.е.н., доцент
Гусєва Ірина Ігорівна

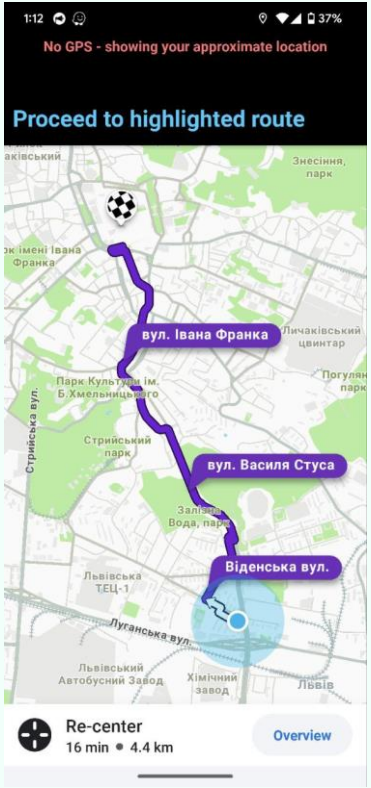
Електромобілі



Існуючі аналоги



Google Maps



Waze

ЗАДАЧА ФОРМУВАННЯ ЕНЕРГОЕФЕКТИВНИХ МАРШРУТІВ ДЛЯ ЕЛЕКТРОМОБІЛІВ

Об'єкт дослідження: Визначення витрат енергії електромобілем.

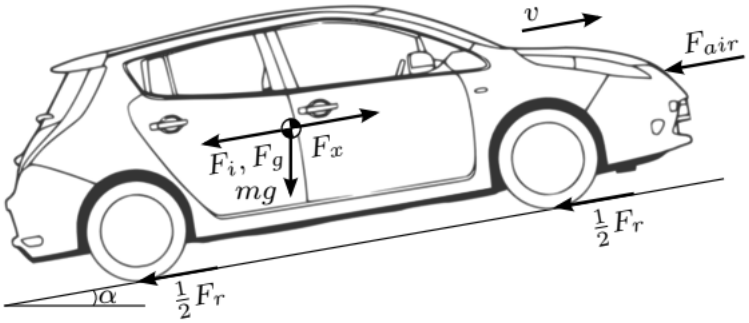
Предмет дослідження: програмне забезпечення для формування енергоефективних маршрутів для електромобілів.

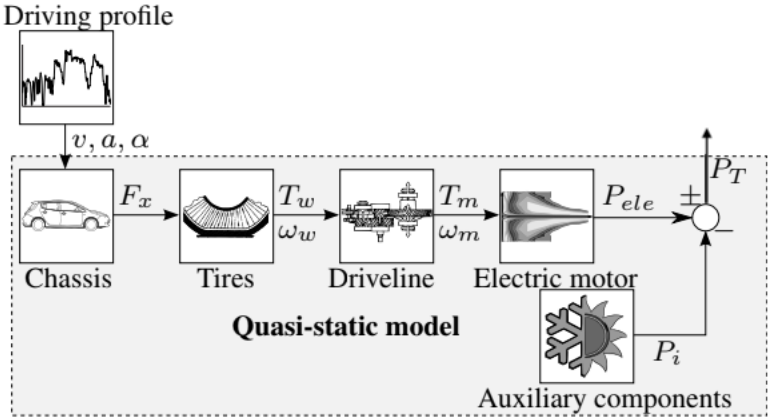
Мета: Розробити програмного забезпечення, яке за допомогою проаналізованих даних та застосування їх до алгоритму обчислення витраченої енергії формує найбільш енергоефективний маршрут та відображає його користувачу через мобільний застосунок.

Задачі, які потрібно розв'язати для досягнення мети:

- Проаналізувати існуючі рішення і дослідження визначення енергозатратності електромобіля;
- Провести аналіз вхідних даних;
- Визначити алгоритм обчислення витраченої енергії за заданим маршрутом;
- Розробити програмне забезпечення для застосування алгоритму і побудови відповідного маршруту.

Квазістатична модель руху електричного транспортного засобу





Визначення спожитої енергії електромобілем

Загальна сила:

$$F_x = F_{air} + F_g + F_r + F_i$$

Аеродинамічна сила лобового опору:

$$F_{air} = \frac{1}{2}\rho_{air} c_w A v^2$$

де ρ_{air} – густина повітря, кг/м^3;
c_w – коефіцієнт лобового опору;
A – площа лобової частини автомобіля, м^2;
v – швидкість руху, м/с.

Сила підйому нагору:

$$F_g = mgsin(\alpha)$$

де m – маса автомобіля, кг;
g – прискорення вільного падіння, м/с^2;
$sin(\alpha)$ – синус кута нахилу дороги.

Сила опору кочення:

$$F_r = mgK_r$$

де K_r – коефіцієнт опору кочення.

Сила прискорення:

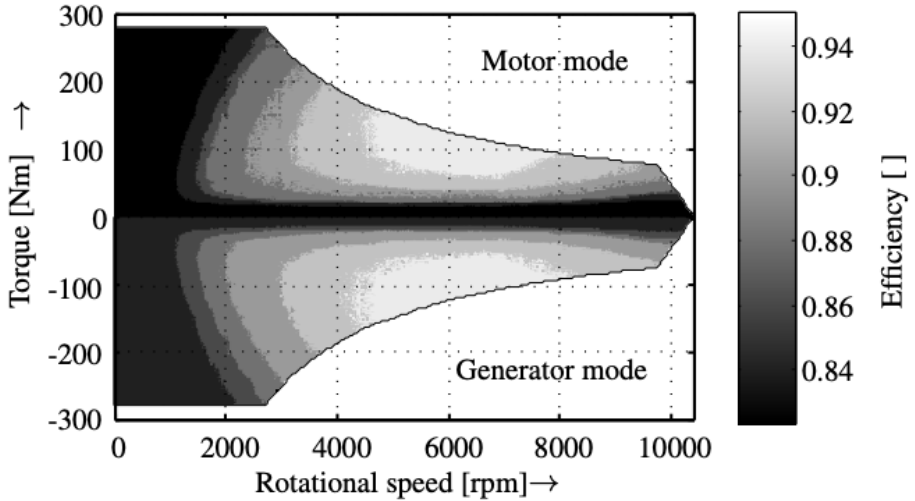
$$F_i = ma$$

Загальна потужність:

$$P_{mec} = F_x v$$

Особливості роботи електричного мотору

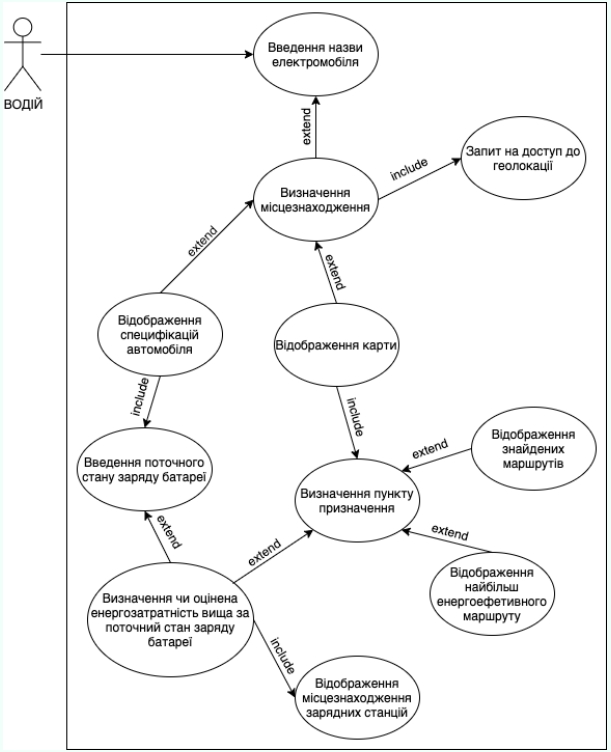
- Електричний мотор має вищий ККД, порівняно з мотором внутрішнього згорання
- Електричний мотор може переходити в стан регенерації енергії під час зниження швидкості



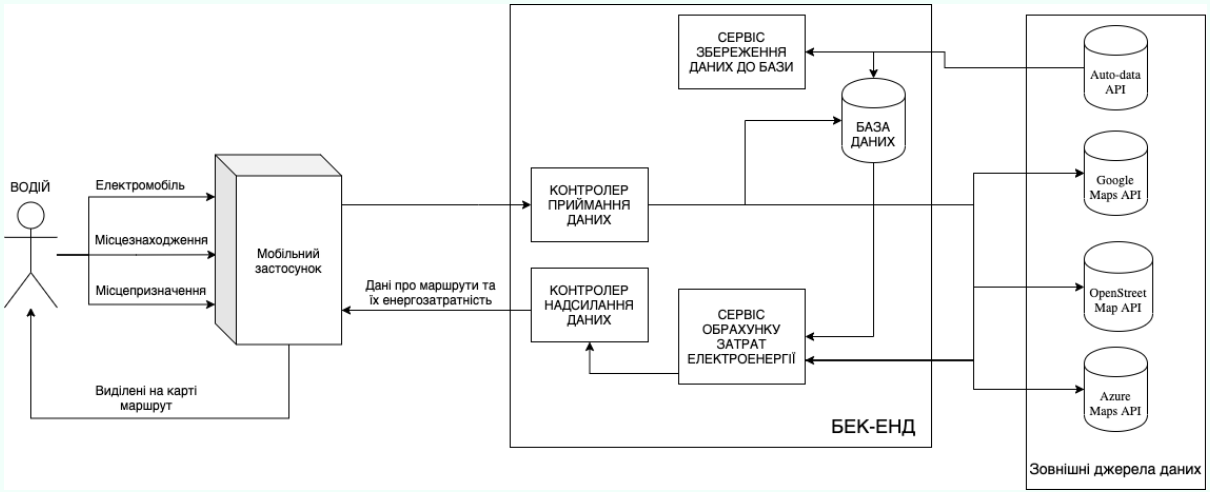
Дані, необхідні для обчислення енергозатратності пройдених маршрутів

Дані	Джерело
Температура, вологість, тиск повітря для визначення **щільності** повітря	Google Maps API
Специфікації електромобіля, такі як: **маса**, **ширина**, **висота, коефіцієнт лобового опору**	Auto-data API
Тип поверхні дороги для визначення **коефіцієнту тертя кочення**	OpenStreetMap API
Висота над рівнем моря, для визначення **кута нахилу** дороги	Google Elevation API
Можливі маршрути у вигляді набору координат(**довгота**, **широта**)	Azure Maps API
Дані про розташування світлофорів для визначення можливих зупинок	OpenStreetMap API
Дані про затори, повороти для визначення моментів уповільнення/прискорення	Azure Maps API

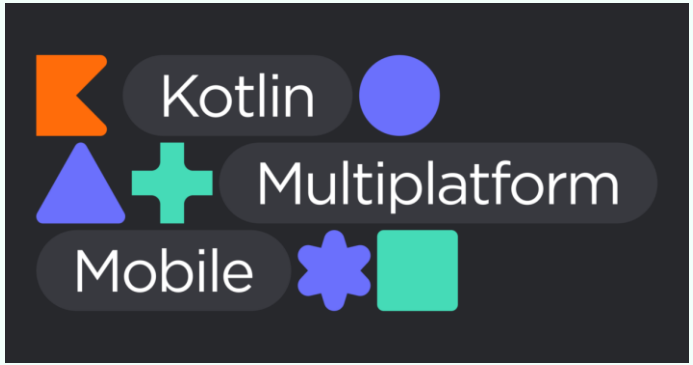
Діаграма прецедентів



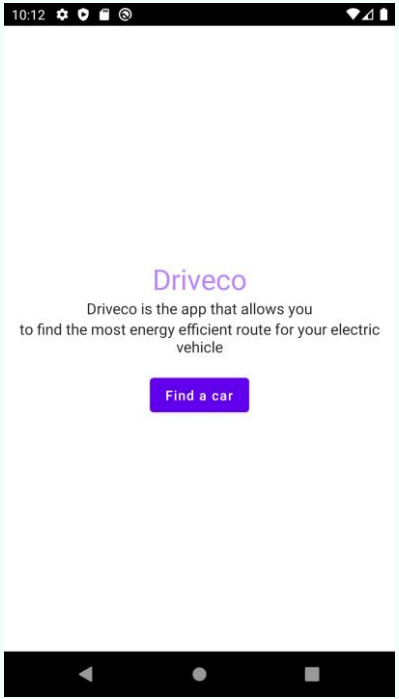
Архітектура ПЗ



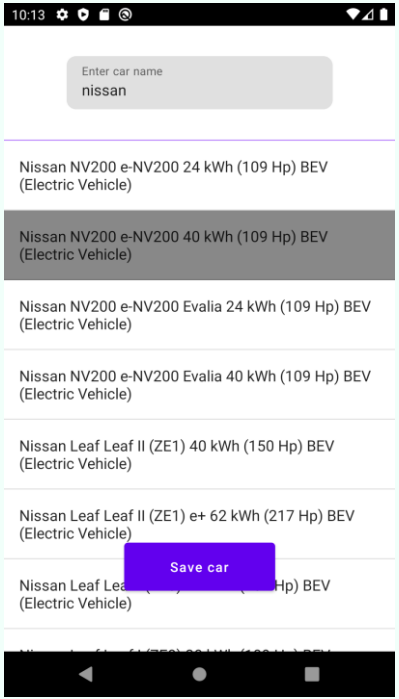
Засоби розробки



Результати розробки програмного забезпечення

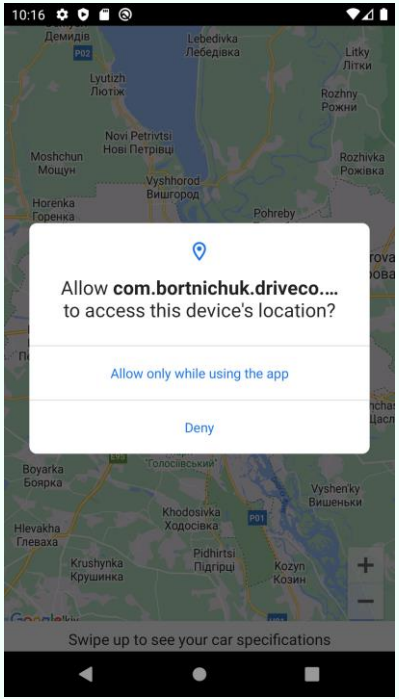


Початковий Екран

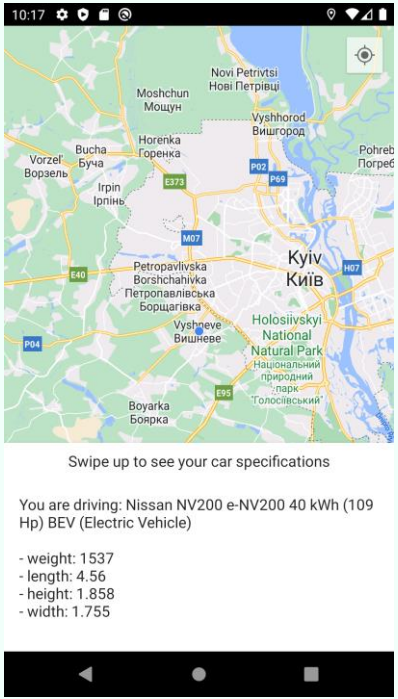


Екран пошуку електромобіля

Результати розробки програмного забезпечення

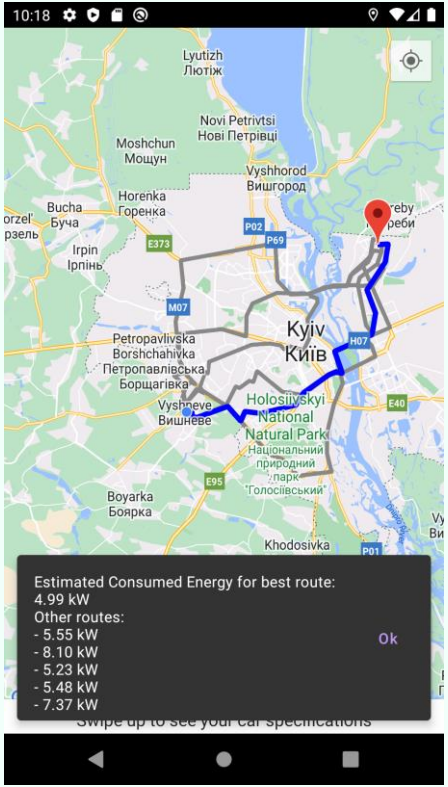


Запит на доступ до геолокації

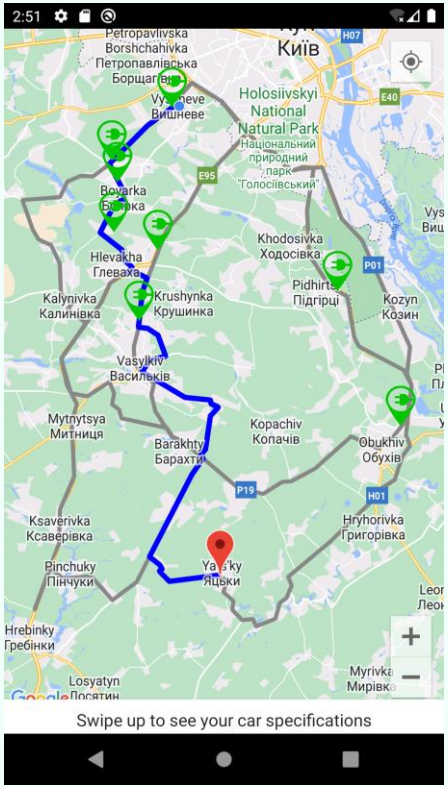


- Визначена геолокація користувача (блакитна точка)
- Відображення специфікацій електромобіля

Результати розробки програмного забезпечення



- Відображення прокладених маршрутів від геолокації до визначеної точки на карті
- Найбільш енергоефективний маршрут виділений синім кольором
- Відображення інформації про енергозатратність кожного з маршрутів

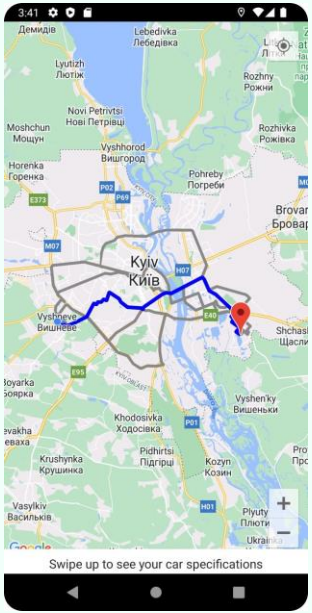
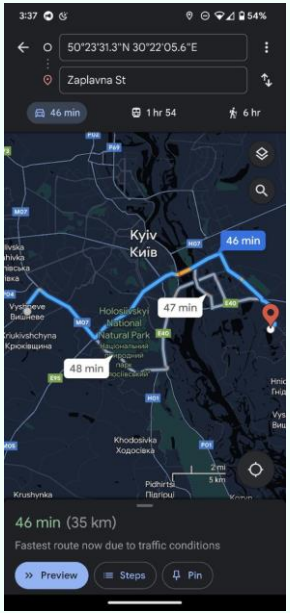
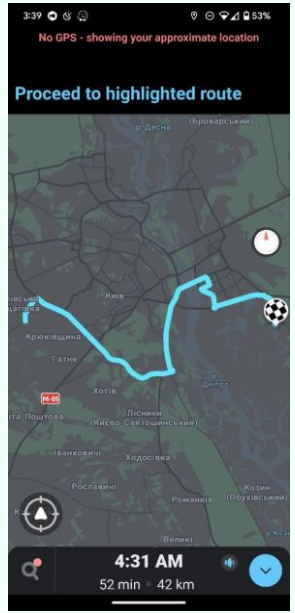


- Місцезнаходження зарядних станцій впродовж маршруту

Порівняння отриманих результатів з існуючими системами

Маршрут:

Вишневе (50.39203,30.36821) – Мікрорайон Бортничі (50.37686,30.69341)

	Driveco	Google Maps	Waze
			
Витрати:	5.9 kWh	6.14 kWh	7.58 kWh

Висновки

- Проаналізовано існуючі методи і моделі реалізації поставленої задачі. Виявлено, що надані елементи підходять для побудови алгоритму системи.
- Проведено пошук, аналіз та збір даних. В результаті необхідні дані були застосовані в обчисленнях.
- Побудовано алгоритм обчислення енергозатратності електромобіля для заданого маршруту з урахуванням особливостей електричного мотора. Алгоритм дозволяє достатньо точно визначити найбільш енергоефективний маршрут.
- Розроблено програмне забезпечення, що реалізує даний алгоритм. Система будує більш оптимальні за енергоефективністю маршрути, в порівнянні з існуючими системами. Це дозволяє економити кількість витраченої електроенергії.

ДЯКУЮ ЗА УВАГУ!