МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРКАСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

Факультет обчислювальної техніки, інтелектуальних та управляючих систем

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 Інженерія програмного забезпечення

До захисту допускаю
Завідувач кафедри ПЗАС
Супруненко О.О.
(ініціали та прізвище)

(підпис)
“____”_____20___ р.

КВАЛІФІКАЦІЙНА РОБОТА

освітнього ступеня “бакалавр”

СЕРВІС ЗАМОВЛЕННЯ ТАКСІ З МОЖЛИВІСТЮ ЧАСТКОВО-СПІЛЬНОЇ ПОЇЗДКИ ПОВ’ЯЗАНОЇ ГРУПИ КЛІЄНТІВ

Студент групи КС-20 (шифр групи) Полигалов Влад Юрійович (прізвище, ім’я, по батькові) _____ (підпис)

Науковий керівник Порубльов І.М., ст.викл. (вчені ступінь та звання, прізвище, ініціали) _____ (підпис)

Черкаси – 2024

ЗМІСТ

ВСТУП 5
РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ СЕРВІСУ ТАКСІ З МОЖЛИВІСТЮ ЧАСТКОВО-СПІЛЬНОЇ ПОЇЗДКИ ПОВ'ЯЗАНОЇ ГРУПИ КЛІЄНТІВ 7
1.1 Актуальність роботи та аналіз проблем програмного забезпечення сервісу таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів 7
1.2 Огляд та аналіз аналогів сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів 7
1.2.1 Сервіс "Uber" 8
1.2.2 Мобільний застосунок "Bolt" 9
1.2.3 Мобільний сервіс "Uklon" 10
1.3 Огляд алгоритмів для розробки сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів 12
1.4 Аналіз підґрунтя для розробки сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів 13
1.5 Постановка задачі на розробку сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів 13
Висновок до першого розділу 14
РОЗДІЛ 2 АНАЛІЗ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ СЕРВІСУ ЗАМОВЛЕННЯ ТАКСІ З МОЖЛИВІСТЮ ЧАСТКОВО-СПІЛЬНОЇ ПОЇЗДКИ ПОВ'ЯЗАНОЇ ГРУПИ КЛІЄНТІВ 15
2.1 Моделювання бізнес-процесів сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів 15
2.2 Формування користувацьких історій сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів 16
2.3 Аналіз вимог сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів з використанням UML-діаграм 17

2.4 Функціональні та нефункціональні вимоги сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 19
2.5 Попереднє архітектурне проектування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 21
2.6 Детальне проектування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 24
2.6.1 Пакет “Керування замовленнями” 24
2.6.2 Пакет “Історія замовлень” 26
2.6.3 Пакет “Розрахунок розподілу” 27
2.7 Проектування моделей бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 28
2.7.1 Концептуальна модель бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 28
2.7.2 Логічна модель бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 29
2.8 Уточнена діаграма архітектури сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 31
Висновки до другого розділу 32
РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ СЕРВІСУ ЗАМОВЛЕННЯ ТАКСІ З МОЖЛИВІСТЮ ЧАСТКОВО-СПІЛЬНОЇ ПОЇЗДКИ ПОВ’ЯЗАНОЇ ГРУПИ КЛІЄНТІВ 33
3.1 Вибір мови, зовнішніх сервісів та середовища програмування для реалізації сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 33
3.2 Структурна схема реалізації сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 34
3.3 Розробка модулів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 35
3.3.1 Розробка компоненту “Створення замовлення” 35
3.3.2 Розробка компоненту “Розрахунок розподілу” 36

3.3.3 Розробка компоненту “Історія замовлень” 40
3.3.4 Розробка бази даних Firebase Realtime для сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 41
3.3.5 Розробка інтерфейсу сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 46
3.3.6 Узагальнена діаграма класів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 53
3.4 Тестування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 55
Висновки до третього розділу 55
ВИСНОВКИ 57
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 59
ДОДАТОК А – Фрагмент коду класу який використовується для запису даних в базу даних 61
ДОДАТОК Б – Фрагмент коду запису даних в firebase realtime database 62
ДОДАТОК В – Узагальнена діаграма класів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 63
ДОДАТОК Г – Тестування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів 64
ДОДАТОК Д – Тестування алгоритму розподілу клієнтів між авто 67
ДОДАТОК Е – Посилання на git-репозиторій із вихідним кодом програмного продукту 69

ВСТУП

З урахуванням зростання популярності та використання таксі, а також стрімкого розвитку технологій, актуальність розробки програмного продукту для оптимізації пасажирських перевезень стає надзвичайно важливою. Забезпечуючи можливість колективних поїздок, ми спрямовуємо свої зусилля на створення ефективної, економічно та екологічно вигідної альтернативи для транспортних потреб сучасного суспільства.

Проте існуючі на сьогоднішній день сервіси таксі, незважаючи на свою популярність, не забезпечують можливості перевозити клієнтів до різних місць призначення в один і той же час. Відсутність такої опції виявляється неефективним та неекономічним для групових поїздок.

Метою даної кваліфікаційної роботи є розробка сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів, який дозволяє не лише забезпечити зручний та оперативний транспорт для користувачів, але й сприяє зменшенню фінансового навантаження на окремих користувачів.

Завдання:

1) дослідити особливості та функціональні можливості існуючих сервісів замовлення таксі;

2) визначити переваги та недоліки у застосунках;

3) спроектувати та розробити як програмний продукт власний сервіс групового замовлення таксі;

4) реалізувати можливість побудови маршруту для групи клієнтів із різними місцями призначення;

5) розробити алгоритм визначення найдешевшого способу розвезення;

6) створити алгоритм рекомендації розподілу між клієнтами групи суми до оплати;

7) зробити опис результатів огляду, проектування, реалізації та тестування застосунку у кваліфікаційній роботі;

8) провести тестування програмного продукту.

Об'єктом розробки є сервіс замовлення таксі, який включає в себе новий функціонал для організації частково-спільних поїздок для пов’язаних груп клієнтів.

Предметом розробки є створення інноваційного модуля для застосунку замовлення таксі, який надасть можливість користувачам створювати частково-спільні поїздки.

Практична цінність даної кваліфікаційної роботи полягає у створенні інноваційного сервісу для замовлення таксі, який дозволяє групам клієнтів здійснювати частково-спільні поїздки. Цей сервіс забезпечує оптимізацію витрат, оскільки дозволяє розподілити вартість поїздки між кількома пасажирами, роблячи послугу більш доступною. Крім того, сервіс має можливість розподіляти клієнтів між різними автомобілями, що дозволяє оптимізувати поїздки і забезпечувати більш вигідні маршрути для користувачів.

Структура та обсяг роботи

Дана робота містить в собі вступ, три розділи, висновки, вісімнадцять пунктів використаних джерел та п’ять додатків. Обсяг роботи складає 69 сторінок, основна частина — 58 сторінок, 26 рисунків та 10 таблиць.

РОЗДІЛ 1 ОГЛЯД ТА АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ РОЗРОБКИ СЕРВІСУ ТАКСІ З МОЖЛИВІСТЮ ЧАСТКОВО-СПІЛЬНОЇ ПОЇЗДКИ ПОВ'ЯЗАНОЇ ГРУПИ КЛІЄНТІВ

1.1 Актуальність роботи та аналіз проблем програмного забезпечення сервісу таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів

Забезпечення ефективної організації колективних поїздок через сервіси замовлення таксі не лише сприятиме зменшенню транспортних заторів, але й відповідатиме зростаючим вимогам споживачів до економічних альтернатив.

У ході аналізу проблем програмних забезпечень, пов'язаних з існуючими сервісами таксі, виявлено кілька ключових аспектів, які потребують уваги та вирішення.

1. Недостатня функціональність для групових поїздок: більшість існуючих сервісів таксі не надають можливості ефективного об'єднання поїздок для груп людей. Це обмеження призводить до втрати можливості економії та непродуктивного використання ресурсів транспорту.

2. Відсутність системи розподілу витрат: брак алгоритмів для розподілу витрат між учасниками поїздки може призвести до конфліктів та невдоволення користувачів.

Розробка програмного продукту, яка враховує ці проблеми, є важливим кроком для покращення функціональності та ефективності сервісів замовлення таксі, що відповідає сучасним потребам споживачів та сприяє сталому розвитку міських транспортних систем.

1.2 Огляд та аналіз аналогів сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів

З огляду на зростання популярності ефективних транспортних рішень, багато компаній у сфері пасажирських перевезень активно розробляють та

вдосконалюють сервіси, які дозволяють клієнтам спільно використовувати транспорт з метою економії коштів. У цьому розділі розглянемо кілька провідних аналогів сервісу замовлення таксі із функцією частково-спільної поїздки.

1.2.1 Сервіс “Uber”

Uber (рис.1.1) — американська компанія, що створила однойменний мобільний застосунок для пошуку, виклику та оплати таксі або приватних водіїв. Застосунок Uber дозволяє користувачам легко та зручно викликати авто та контролювати весь процес подорожі. Він є частиною широкого сервісу Uber, який працює у багатьох містах світу [1].

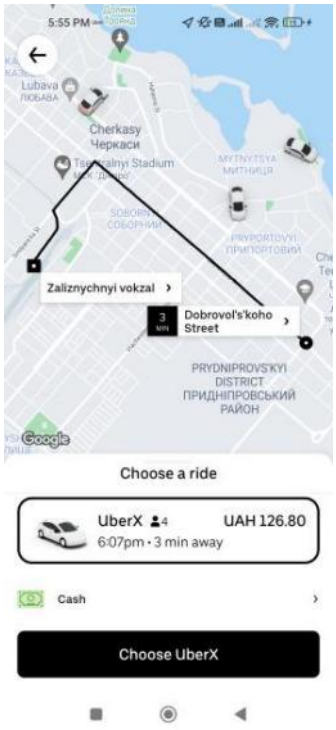


Рисунок 1.1 – Інтерфейс застосунку “Uber”

Після огляду цього застосунку були помічені наступні переваги:

- інтерфейс та карта програми працюють без наявності інтернету;
- наявна історія попередніх маршрутів;
- відображення приблизного часу прибуття авто;

- відображення ціни за поїздку;
- можливість планування поїздки;
- можливість здійснити спільну поїздку.

Сервіс має наступні недоліки:

- функція спільної поїздки має обмеження в 2 особи;
- вибір типу поїздки (економ, комфорт, бізнес) обмежений для деяких регіонів.

1.2.2 Мобільний застосунок “Bolt”

Bolt (рис. 1.2) — це компанія, що працює в сфері транспортних послуг. Заснована в Естонії в 2013 році. Користувачі можуть замовляти поїздки через застосунок Bolt на своїх смартфонах. Основна ідея Bolt - забезпечити зручний та доступний транспорт для користувачів. Вони пропонують різні види транспорту, Bolt (стандартні поїздки), Comfort (комфорт), XL (для більшої кількості пасажирів), Pets (поїздки зі своїми домашніми улюбленцями) [2].

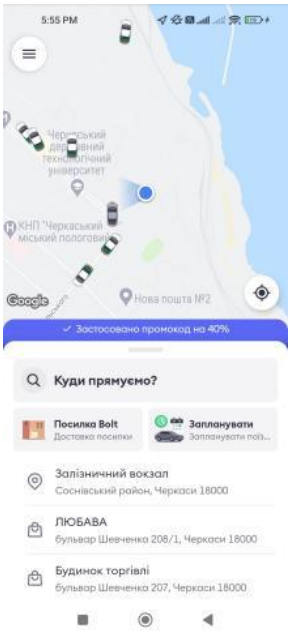


Рисунок 1.2 – Інтерфейс сервісу “Bolt”

Даний застосунок має наступні переваги:

- містить функцію доставки вантажу;
- має можливість запланування перевезення;
- широкий вибір типів поїздок;
- зручний та інтуїтивний інтерфейс для прокладання маршруту;
- надає можливість створювати маршрути з місцями для зупинки.

Програмне забезпечення має такі недоліки:

- відсутність розподілу оплати між клієнтами однієї групи;
- обмежене покриття в деяких регіонах;
- відсутність додаткових сервісів під час замовлення таксі.

1.2.3 Мобільний сервіс “Uklon”

Сервіс “Uklon” (рис. 1.3) — це український сервіс пасажирських перевезень, який надає можливість користувачам замовляти транспорт через мобільний застосунок або веб-сайт. Сервіс дозволяє легко та зручно організовувати переміщення в місті, вибираючи різні типи транспорту та користуючись різними опціями [3].

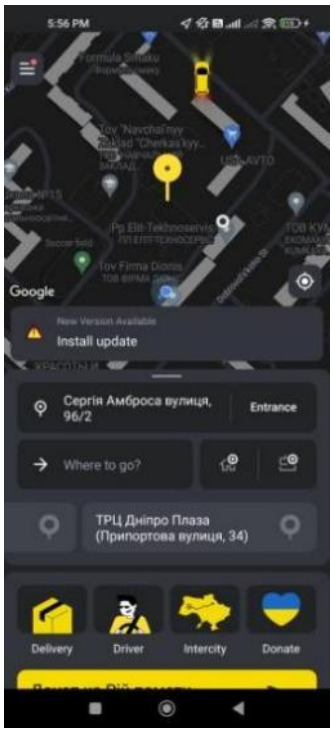


Рисунок 1.3 – Інтерфейс застосунку “Uklon”

Основними перевагами даного застосунку є:

- зручний та зрозумілий інтерфейс;
- можливість замовити додаткові послуги до поїздки (англомовний водій, додатковий багаж, зустріти з табличкою, їду з домашнім улюбленцем);
- можливість змінювати ціну за поїздку, щоб швидше знайти авто;
- можливість додати точки зупинки на маршруті.

Застосунок має наступні недоліки:

- відсутність розділення оплати між клієнтами;
- наявність реклами в застосунку;
- кількість категорій авто залежить від регіону.

Отже розглянувши аналоги можна помітити ряд недоліків які мають оглянуті сервіси замовлення таксі.

1. Обмеження функції спільної поїздки: у деяких сервісах, наприклад, у Uber, функція спільної поїздки може бути обмежена до двох осіб. Це може ускладнити організацію поїздок для більших груп пасажирів.

2. Обмежені вибори типу транспорту: деякі сервіси, наприклад, Bolt, можуть обмежувати вибір типу поїздки (економ, комфорт, бізнес) в певних регіонах. Це може обмежити можливості користувачів вибору підходящого транспорту.

3. Відсутність розділення оплати між клієнтами однієї групи: відсутня функція розділення оплати між користувачами однієї групи, що може бути не зручним для клієнтів, які хочуть поділити вартість поїздки.

4. Обмежене покриття в деяких регіонах: деякі сервіси можуть мати обмежене покриття в певних регіонах, що робить їх менш доступними для користувачів у цих областях.

5. Присутність реклами в застосунку: деякі з розглянутих сервісів можуть включати рекламу в своїх мобільних застосунках, що може впливати на загальний користувацький досвід та сприйняття сервісу.

Незважаючи на описані вище недоліки можна зробити висновок, що найкращою реалізацією даного сервісу є поєднання першого та третього

аналогів, адже вони мають функціонал, що є ключовим для нашого застосунку, а саме: можливість додати точки зупинки на маршруті від другого застосунку та можливість створювати спільні поїздки від застосунку номер один.

1.3 Огляд алгоритмів для розробки сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів

Для оптимізації маршрутів та розподілу вартості поїздки у таксі, важливо використовувати алгоритми та технічні засоби, які дозволять ефективно вирішувати ці завдання. Серед численних алгоритмів, що розв'язують задачу пошуку найкоротшого шляху, можна виділити декілька найпопулярніших та ефективних.

Один із таких алгоритмів – алгоритм Дейкстри, він дозволяє знайти найкоротший шлях від однієї точки графа до всіх інших, враховуючи ваги ребер та надаючи перевагу шляхам з меншою вартістю [4].

Ще одним ефективним алгоритмом для визначення оптимального маршруту є алгоритм A*. Цей метод поєднує у собі переваги алгоритмів Дейкстри та жадібного пошуку, що робить його ефективним і придатним для вирішення задачі маршрутизації в таксі [5].

Існує багато сервісів для побудови оптимального маршруту та навігації. Один із них API MapBox [6]. Він надає широкі можливості для роботи з картами та маршрутами, включаючи розрахунок оптимального маршруту, враховуючи різні умови дорожнього руху та вимоги маршруту. Також цей сервіс надає можливість створювати маршрути, які проходять через декілька заданих точок. Це дозволяє планувати складні маршрути, які включають в себе кілька проміжних точок, наприклад, для пасажирських поїздок з декількома зупинками.

Для розподілу вартості поїздки між учасниками можна використовувати вектор Шеплі. Вектор Шеплі — це метод розподілу вартості поїздки між учасниками, який базується на теорії кооперативних ігор [7]. Він визначає

"внесок" кожного учасника у вибір маршруту та розподіляє вартість поїздки, враховуючи цей внесок. Вектор Шеплі використовується для забезпечення справедливості та ефективності розподілу вартості поїздки між учасниками.

1.4 Аналіз підґрунтя для розробки сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Підґрунтям розробки системи замовлення таксі з можливістю частково-спільної поїздки для пов'язаної групи клієнтів є виявлення визначених недоліків у існуючих транспортних сервісах, а також необхідність вдосконалення та розширення можливостей таких платформ.

Одним із ключових стимулів для створення цієї системи є відсутність адаптованих та ефективних інструментів для замовлення таксі з можливістю частково-спільної поїздки для групи користувачів. Існуючі сервіси можуть не задовольняти потреби клієнтів, які хочуть подорожувати разом, спричиняючи неефективне використання транспортних ресурсів та непрактично високі витрати.

Подальшою основою для цього проекту є впевненість у тому, що використання сучасних технологій та інновацій у сфері транспортних послуг може значно полегшити та зручно організувати подорожі пов’язаної групи людей.

1.5 Постановка задачі на розробку сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Під час реалізації застосунку потрібно вирішити ряд визначених задач.

1. Створити середовище де буде відображатися карта та побудований маршрут.

2. Реалізувати функціонал створення спільної поїздки для пов’язаної групи клієнтів.

3. Додати можливість обрання типу поїздки.

4. Надати користувачам змогу відмінити поїздку.

5. Розробити метод визначення найдешевшого способу розвезення (кому з групи вигідно поєднати поїздку в одному авто, а кому ні).

6. Реалізувати алгоритм рекомендації розподілу між клієнтами групи суми до оплати.

Висновок до першого розділу

У цьому розділі було описано актуальність теми, проведений огляд та аналіз аналогів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів виділені їх переваги та недоліки, описані алгоритми які використовуються для розв’язання задач. Також було виконано аналіз підґрунтя та описано постановку задачі на розробку.

РОЗДІЛ 2 АНАЛІЗ ТА ПРОЕКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ СЕРВІСУ ЗАМОВЛЕННЯ ТАКСІ З МОЖЛИВІСТЮ ЧАСТКОВО-СПІЛЬНОЇ ПОЇЗДКИ ПОВ'ЯЗАНОЇ ГРУПИ КЛІЄНТІВ

2.1 Моделювання бізнес-процесів сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів

На рис. 2.1 зображено модель бізнес процесів для чотирьох груп акторів: гість, клієнт, водій та адміністратор.

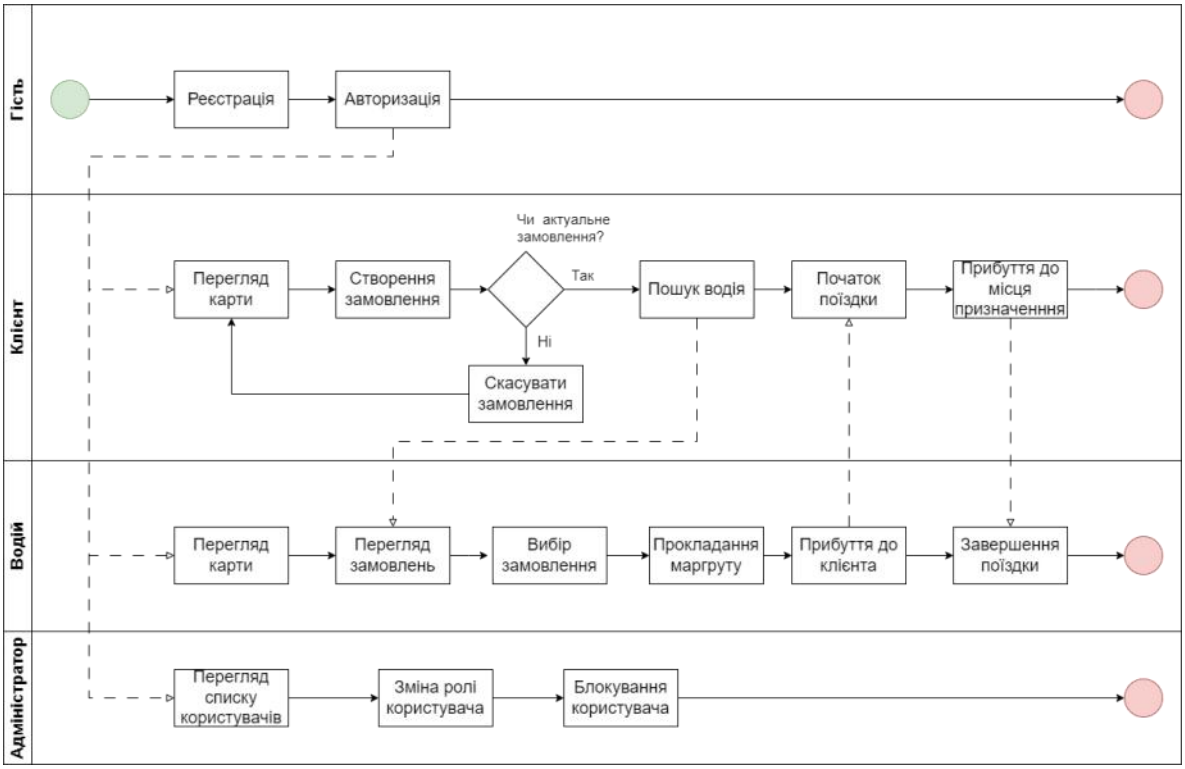


Рисунок 2.1 – Модель бізнес-процесів сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів

Гість може авторизуватися або зареєструватися в застосунку. Він може обрати опцію "Увійти" або "Зареєструватися". У разі вибору "Увійти", гість вводить свої облікові дані (логін та пароль), після чого стає клієнтом і має можливість здійснювати замовлення таксі та інші дії в рамках сервісу. У разі

вибору "Зареєструватися", гість переходить до процесу створення облікового запису, вводить необхідні дані (ім'я, email, пароль тощо) і після успішної реєстрації стає користувачем сервісу.

Клієнт може переглядати карту та створити замовлення вказавши всі необхідні точки прибуття. Після підтвердження маршруту буде сформовано замовлення, а в разі необхідності клієнт може його скасувати. В іншому випадку відбудеться пошук водія та переміщення клієнта з місця відправки до місця прибуття.

Водій може переглядати список замовлення та приймати їх. Після вибору замовлення водієм, застосунок автоматично прокладає маршрут до клієнта. Після завершення поїздки водій підтверджує виконання замовлення.

Адміністратор відповідає за управління користувачами та водіями. Він приймає заявки на реєстрацію водіїв та вирішує про їх підтвердження або відхилення. Також адміністратор блокує клієнтів або водіїв у разі необхідності.

2.2 Формування користувацьких історій сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

У цьому пункті будуть розглянуті сформовані користувацькі історії для сервісу замовлення таксі з акцентом на три основні ролі — гість, клієнт та таксист.

1. Гість матиме змогу створити обліковий запис або увійти, якщо такий існує.
2. Клієнт матиме змогу вийти зі свого акаунту.
3. Клієнт матиме можливість редагувати свій профіль.
4. Клієнт матиме можливість переглядати карту.
5. Клієнт матиме змогу швидко та легко замовити таксі, вказавши своє місце знаходження та місце призначення.
6. Клієнт матиме змогу обрати тип авто (економ, комфорт, мінібус), відповідно до своїх потреб та вподобань.

7. Як клієнт, я хочу бачити вартість та тривалість поїздки перед підтвердженням замовлення.

8. Клієнт матиме можливість скасувати замовлення до прибуття таксі.

9. Клієнт матиме можливість переглянути історію замовлень.

10. Як клієнт як хочу мати можливість формування спільної поїздки вказавши місця призначення.

11. Як клієнт я хочу мати можливість переглянути рекомендації щодо розподілу суми до оплати між всіма клієнтами групи.

12. Таксист матиме можливість увійти та вийти зі свого облікового запису.

13. Як таксист я хочу мати можливість переглянути список замовлень.

14. Як таксист я хочу мати можливість прийняти замовлення.

15. Таксист матиме можливість використовувати навігацію, щоб швидко дістатися до клієнта.

16. Таксист матиме можливість переглянути більш детальну інформацію про замовлення.

17. Таксист матиме змогу підтвердити завершення поїздки.

2.3 Аналіз вимог сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів з використанням UML-діаграм

В даному пункті будуть розглянуті та описані use-case діаграми для визначених вище ролей. Першими на черзі йдуть діаграми акторів гість та клієнт.

Як зображено на рисунку 2.2, зайшовши в застосунок, гість має можливість створити новий обліковий запис або увійти в уже існуючий.

Як видно на діаграмі (рис. 2.2) клієнт може вийти зі свого акаунту, видалити або редагувати його, створювати спільні поїздки, скасувати замовлення, переглянути категорії авто, переглянути історію попередніх замовлень, отримати рекомендації щодо розподілу суми до оплати між групою та зареєструватися у ролі водія.

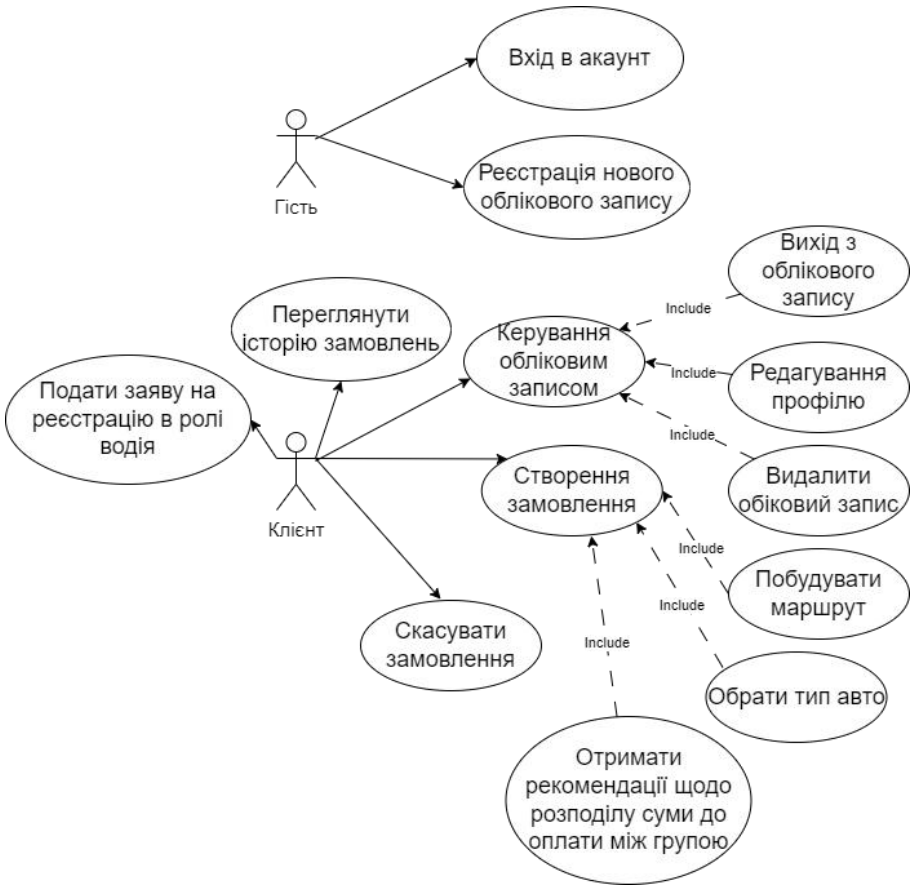


Рисунок 2.2 – Use-case діаграми ролі “Клієнт” та “Гість”

На рисунку 2.3 зображена Use-case діаграма ролі “Водій”.

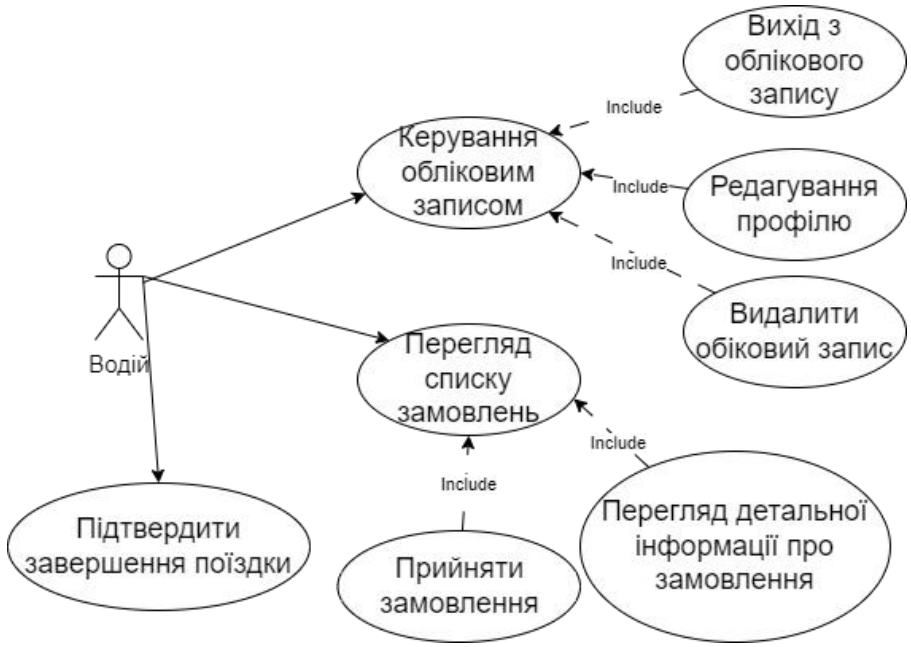


Рисунок 2.3 – Use-case діаграма ролі “Водій”

Водій зможе редагувати свій профіль, видалити його та просто вийти з нього, переглядати список всіх замовлень, переглянути більш детальну інформацію про замовлення, прийняти замовлення, підтвердити завершення поїздки.

На рисунку 2.4 зображена Use-case діаграма ролі “Адміністратор”

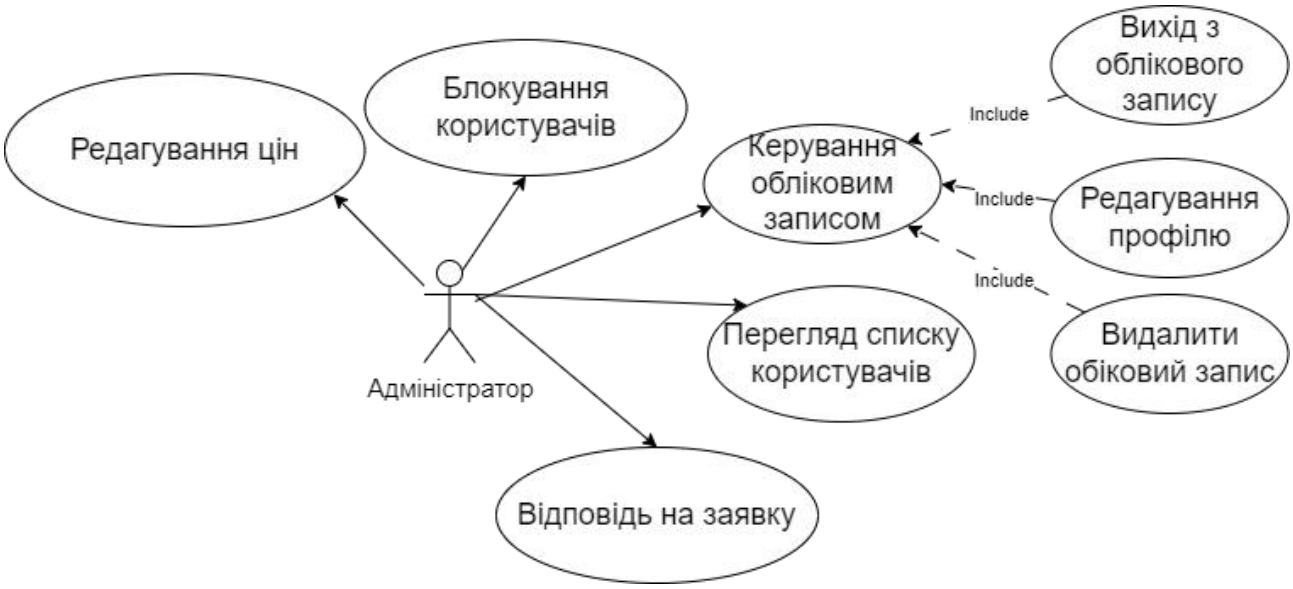


Рисунок 2.4 – Use-case діаграма ролі “Адміністратор”

На даній діаграмі зображено, адміністратор матиме змогу керувати своїм обліковим записом, переглядати списки всіх зареєстрованих користувачів, відповідати на заявки про реєстрацію у ролі водія, блокувати облікові записи користувачів та редагувати значення цін на різні типи поїздок.

2.4 Функціональні та нефункціональні вимоги сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Після проведення аналізу користувацьких історій були сформульовані функціональні вимоги до програмного забезпечення та записані у Product Backlog (таблиця 2.1).

Таблиця 2.1 - Функціональні вимоги до сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

1. №	2. Назва вимоги	3. Важливість	4. Як продемонструвати	5. Примітки
1	Створення замовлення	10	Кнопка “створити замовлення” з’явиться після побудови маршруту	Клієнт повинен буде обрати тип авто після чого він отримає інформацію про поїздку.
2	Розрахунок розподілу	10	Під час створення замовлення	Після обрання типу поїздки клієнт отримає інформації про розподіл клієнтів та суми до сплати.
3	Прийняття замовлення	9	Кнопка “Взяти замовлення”	Таксист сам може обрати замовлення яке його цікавить або просто ігнорувати їх.
4	Авторизація	8	Після відкриття застосунку	Гість матиме можливість створити новий обліковий запис або зайти на вже існуючий.
5	Управління картою	8	Після того як клієнт зайде в свій акаунт	Клієнт матиме можливість переглянути карту.
6	Побудова маршруту	8	Натиснувши на кнопку побудувати маршрут	Після того як користувач натиснув на кнопку відкриється вікно де він зможе обрати стартову та фінішні точки
7	Навігація	8	Після прийняття замовлення	Після того як таксист обрав замовлення на карті прокладається маршрут до клієнта.
8	Перегляд категорій авто	6	Після натискання на кнопку “створити замовлення”	Після того як клієнт створив маршрут він матиме змогу обрати тип поїздки
9	Скасування поїздки	5	Розділ “Історія замовлень” та обрати активне замовлення	Клієнт матиме змогу скасувати поїздку натиснувши на неї у списку та обравши функцію скасувати

Продовження таблиці 2.1

1	2	3	4	5
10	Перегляд деталей замовлення	5	Обрати конкретне замовлення в розділі “список замовлень”	Таксист може переглянути більш детальну інформацію про замовлення натиснувши на нього.
11	Вихід з облікового запису	3	Розділ “профіль”	-
12	Видалення акаунта	3	Розділ “профіль”	Після натискання на кнопку видалення з’явиться вікно про підтвердження дії

Перелік нефункціональних вимог сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів:

- програма повинна зберігати інформацію про користувачів на віддаленому сервері;
- панель адміністратора повинна мати функцію перегляду списку зареєстрованих користувачів;
- карта та побудова маршруту повинні відображатися на головній панелі;
- в обліковому записі має бути можливість зміни інформації про користувача;
- інтерфейс користувача повинен бути простим та інтуїтивно зрозумілим.

2.5 Попереднє архітектурне проектування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Під час попереднього архітектурного проектування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів, важливо визначити архітектуру, яка відповідала б вимогам проекту та забезпечувала б його ефективну реалізацію. Після порівняння трьох архітектурних рішень: MVC [8], SOA [9], та клієнт-сервер [10], було зроблено висновок, що клієнт-серверна архітектура є кращим варіантом для розробки

сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів.

Вона дозволить ефективно розділити функціональність між клієнтом та сервером, забезпечуючи швидке реагування на зміни в інтерфейсі користувача та забезпечуючи гнучкість у масштабуванні для обробки багатьох запитів від клієнтів. Обравши архітектуру, можна перейти до побудови діаграми пакетів (рис.2.5).

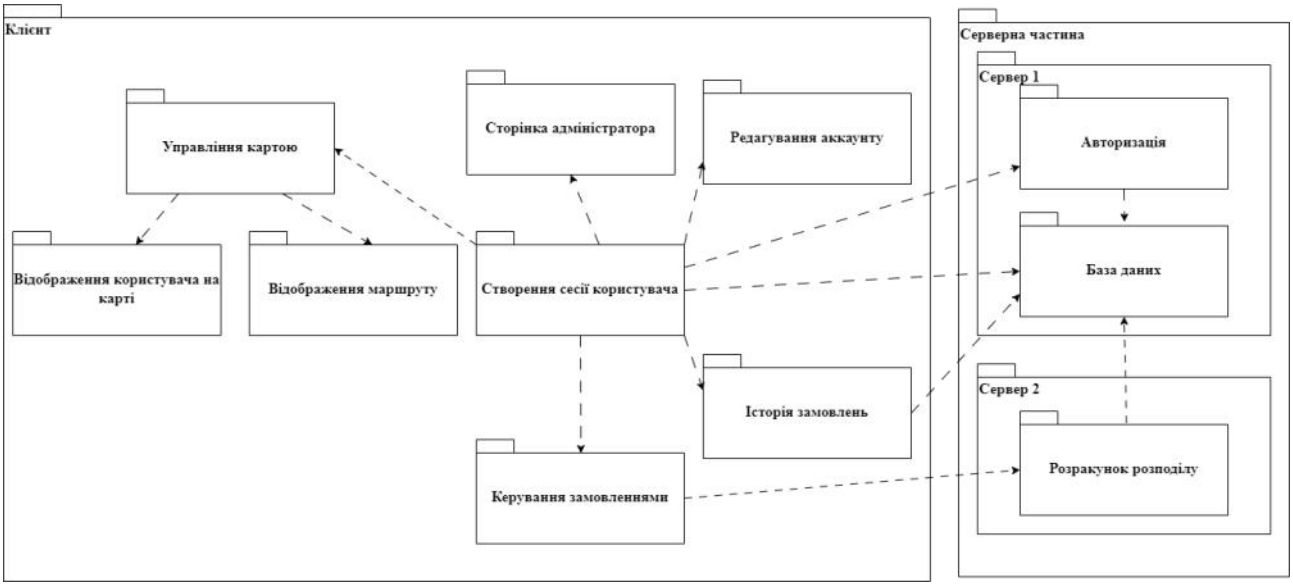


Рисунок 2.5 – Діаграма пакетів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Як можна побачити з діаграми, серверна сторона буде поділена на дві частини: на одній будуть розміщені такі пакети, як авторизація та база даних, що забезпечать безпеку та надійність обробки даних користувачів та замовлень, на іншій розміщений пакет, що буде відповідати за розподіл клієнтів між авто та розрахунок розподілу суми до оплати, а на клієнтській стороні розташовані інші пакети, такі як управління картою, відображення маршруту та інші, які відповідають за інтерактивність та зручність використання застосунку користувачами.

Детальний опис пакетів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів.

1. Управління картою: цей пакет відповідає за інтеграцію з картографічним сервісом для відображення карти в застосунку. Він забезпечує можливість вибору місця призначення на карті та відображення маршруту.

2. Відображення маршруту: цей пакет відповідає за відображення маршруту поїздки на карті.

3. Відображення користувача на карті: цей пакет забезпечує відображення розташування користувача на карті в реальному часі. Він може використовувати дані GPS або інші методи визначення місцезнаходження.

4. Створення сесії користувача: цей пакет відповідає за управління сесіями користувачів, забезпечуючи аутентифікацію та авторизацію.

5. Керування замовленнями: цей пакет відповідає за обробку та керування замовленнями таксі. Він включає в себе можливість створення нового замовлення, відстеження статусу замовлення та його виконання.

6. Сторінка адміністратора: цей пакет надає адміністраторам доступ до панелі управління, де вони можуть переглядати та керувати всіма аспектами застосунку.

7. Редагування акаунту: цей пакет забезпечує можливість користувачам редагувати свій профіль та інформацію в акаунті.

8. Історія замовлень: цей пакет забезпечує можливість перегляду історії замовлень користувача, включаючи деталі та статуси кожного замовлення.

9. Авторизація: цей пакет відповідає за процес авторизації користувачів в застосунку, забезпечуючи безпеку та доступ до персоналізованих функцій.

10. База даних: цей пакет відповідає за зберігання даних, пов'язаних з користувачами, замовленнями, маршрутами та іншою інформацією, необхідною для роботи застосунку.

11. Розрахунок розподілу: цей пакет відповідає за пошук більш економічно вигідних варіантів розвезення користувачів та за розподіл загальної суми до оплати між всіма користувачами.

2.6 Детальне проектування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Розглянемо детальне проектування деяких основних пакетів функцій даного застосунку.

2.6.1 Пакет “Керування замовленнями”

Пакет “Керування замовленнями” відповідає за управління процесом створення та скасування замовлень. Цей пакет забезпечує можливість користувачам здійснювати замовлення та відміняти їх, а також надає необхідні інструменти для внутрішнього управління замовленнями, включаючи їхню обробку та відслідковування. Розглянемо даний пакет більш детально на діаграмі діяльності (рис.2.6).

Рисунок 2.6 – UML-діаграма діяльності пакету “Керування замовленнями”

Ця діаграма демонструє послідовність дій для створення та скасування замовлення в системі. Користувач вибирає опцію "Створення замовлення" у застосунку, після чого вводить необхідні дані. Замовлення створюється та зберігається в базі даних. У разі необхідності користувач може скасувати замовлення.

Далі буде більш детально описаний процес створення замовлення на діаграмі послідовності (рис. 2.7). Цей функціонал відповідає за обробку запитів користувачів щодо створення нових замовлень у системі. Він забезпечує прийняття даних про замовлення від користувача, їх перевірку та збереження в базі даних.

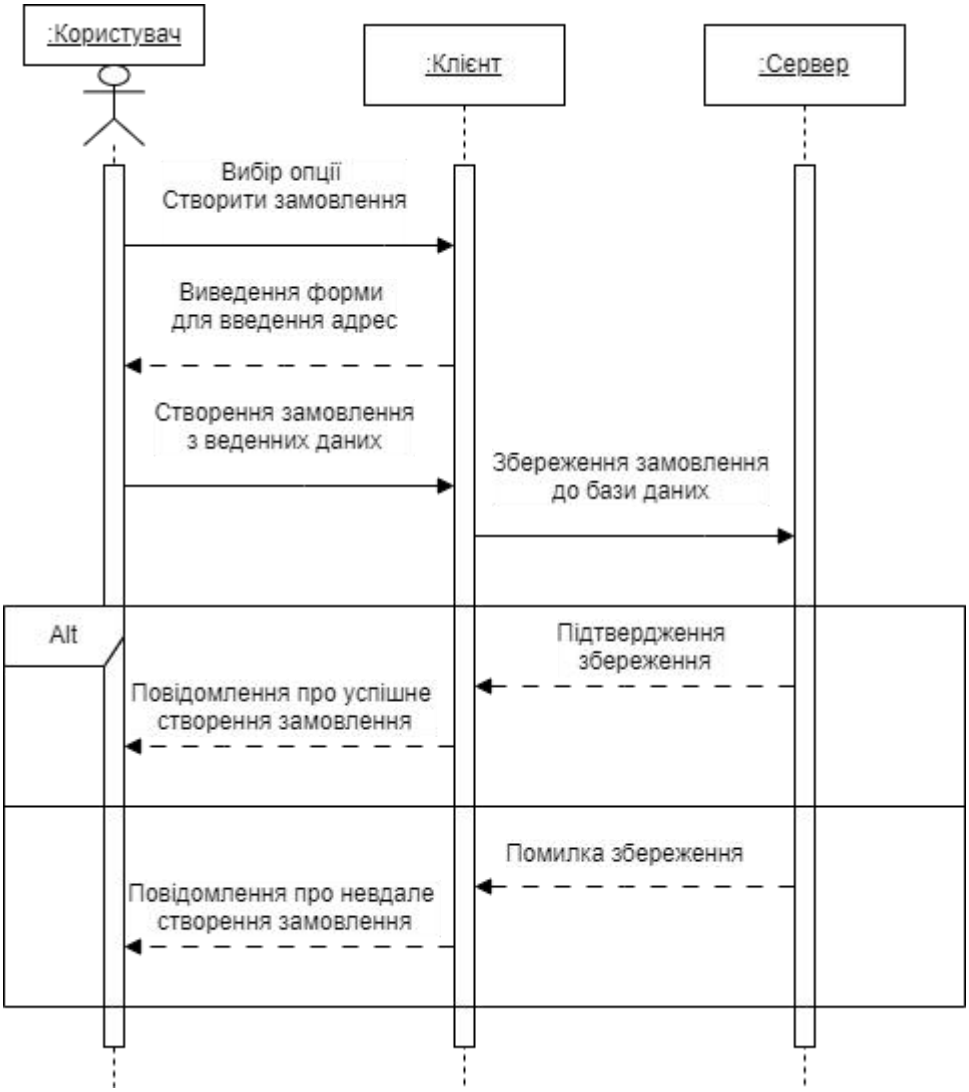


Рисунок 2.7 – UML-діаграма послідовності функції створення замовлення з пакету “Керування замовленнями”

Коли користувач вибирає опцію "Створення замовлення", він матиме можливість ввести деталі замовлення, а саме адреси по яким потрібно побудувати маршрут. Після введення даних користувачем, застосунок надсилає ці дані до серверу. Сервер намагається зберегти замовлення в базі даних. Якщо збереження пройшло успішно, користувач отримає повідомлення про успішно створене замовлення. Однак, якщо під час збереження виникає помилка, сервер повідомляє про невдале створення замовлення, і користувач отримує відповідне повідомлення.

2.6.2 Пакет “Історія замовлень”

Пакет "Історія замовлень" відповідає за збереження та надання доступу до інформації про історію замовлень користувачів. Цей пакет забезпечує функціонал для відображення списку замовлень користувача, деталізації кожного замовлення та можливості перегляду інформації про окремі замовлення. Даний пакет буде описаний за допомогою діаграми комунікації (рис. 2.8).

Рисунок 2.8 – UML-діаграма комунікацій пакету “Історія замовлень”

Як можна побачити на діаграмі спочатку користувач вибирає опцію в застосунку “історія замовлень”. Далі виконується запит до бази даних для отримання всіх замовлень користувача після чого користувач може їх

переглянути. Також він може вибрати конкретне замовлення для перегляду більш детальної інформації.

2.6.3 Пакет “Розрахунок розподілу”

Пакет "Розрахунок розподілу" відповідає за знаходження більш оптимального способу розвезення клієнтів з метою заощадження коштів. Розглянемо більш детально його роботу на діаграмі послідовності (рис. 2.9).

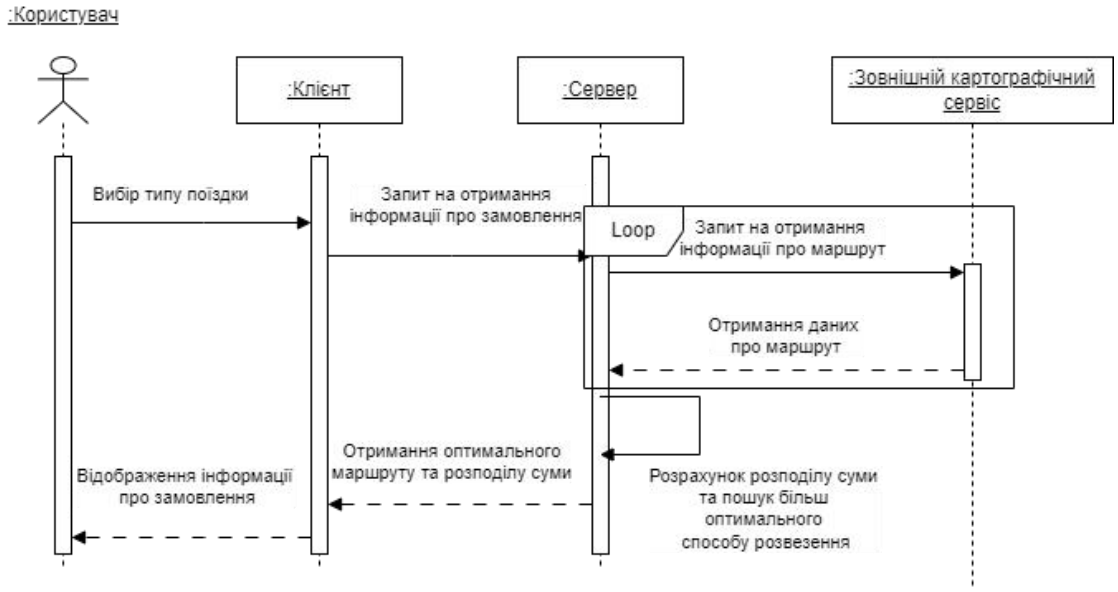


Рисунок 2.9 – UML-діаграма послідовності пакету “Розрахунок розподілу”

На даній діаграмі можна побачити, що після того, як користувач обрав тип поїздки, робиться запит на сервер, який знаходить оптимальний маршрут між всіма заданими точками; в свою чергу, сервер робить запити до зовнішнього сервісу для отримання інформації про маршрут. Після отримання даних сервер шукає найбільш оптимальний спосіб розвезення клієнтів та розраховує розподіл суми до оплати. Далі відповідь сервера повертається на клієнт, після чого клієнту відображається інформація про замовлення з рекомендаціями щодо розвезення клієнтів та розподілу суми до оплати.

2.7 Проектування моделей бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

В даному розділі будуть побудовані та описані концептуальна та логічна моделі бази даних для сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів.

2.7.1 Концептуальна модель бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

На малюнку нижче (рис. 2.10) наведено приклад концептуальної моделі для сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів. Кожна сутність представляє собою об’єкт або концепцію, про які збирається інформація в базі даних, а відношення між сутностями відображають взаємозв’язки між ними.

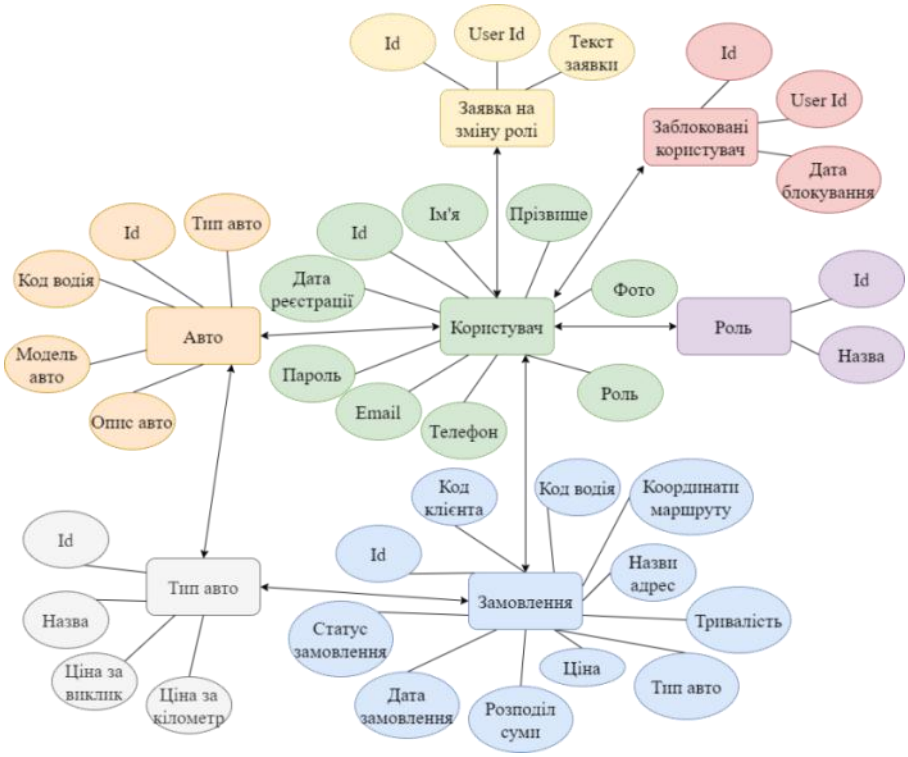


Рисунок 2.10 – Концептуальна модель бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Як можна побачити, на даній моделі зображено сім сутностей (користувач, заявка на зміну ролі, заблоковані користувачі, роль, замовлення, статус замовлення, авто):

- сутність “Користувач” має такі поля: id, ім’я, прізвище, фото, роль, телефон, email, пароль та дата реєстрації;

- сутність “Заявка на зміну ролі” має такі поля: id, код користувача та текст заяви та дата заяви;

- сутність “Заблоковані користувачі” має такі поля: id, код користувача та дата блокування;

- сутність “Роль” має такі поля: id та назва ролі;

- сутність “Авто” має такі поля: id, код водія, модель авто, опис авто, тип авто;

- сутність “Замовлення” має такі поля: id, код клієнта, код водія, маршрут, статус замовлення, дата замовлення, тип авто;

- сутність “Тип авто” має такі поля: id та назва, ціна за виклик, ціна за кілометр.

2.7.2 Логічна модель бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

На рисунку 2.11 наведено логічну модель бази даних для сервісу. Вона вже включає в себе таблиці, поля та зв'язки між ними, що дозволяє краще розуміти, як будуть зберігатися та використовуватися дані в базі даних.

Таблиця “Users” зберігає дані про всіх зареєстрованих користувачів; таблиця “Roles” зберігає ролі користувачів; таблиця “Blocked users” зберігає заблокованих користувачів; таблиця “Cars” зберігає дані про автомобіль водія; таблиця “Orders” зберігає дані про замовлення; таблиця “Car types” зберігає всі можливі типи авто, ціну за виклик авто та ціну за кілометр; таблиця “Statements” зберігає заяви, які подав користувач на роль водія.

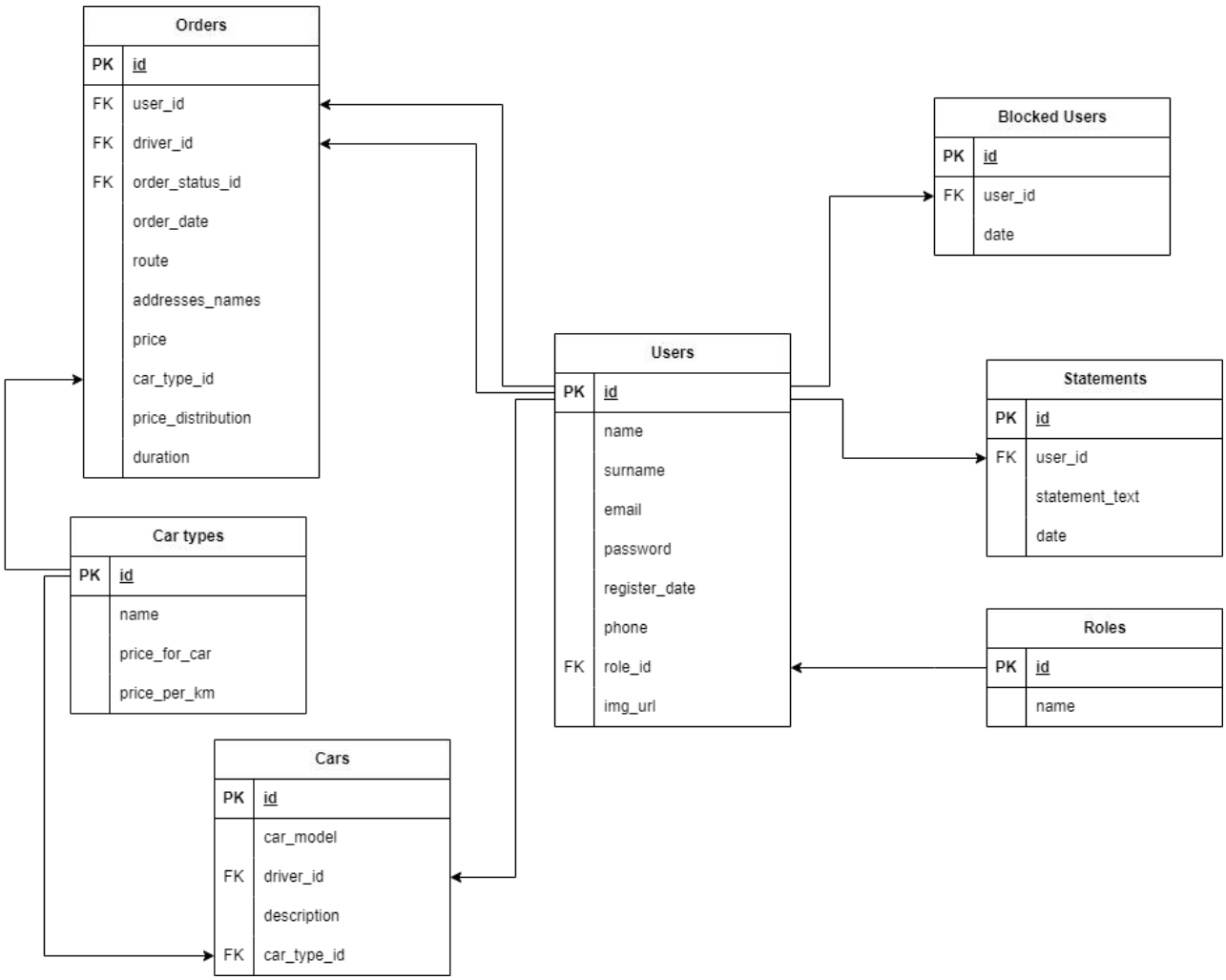


Рисунок 2.11 – Логічна модель бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

На логічній моделі бази даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів наведені наступні зв’язки:

- “Users” та “Roles” — зв’язок один до одного;
- “Users” та “Orders” — зв’язок один до багатьох;
- “Users” та “Blocked users” — зв’язок один до одного;
- “Users” та “Cars” — зв’язок один до одного;
- “Users” та “Statements” — зв’язок один до одного;
- “Car types” та “Orders” — зв’язок один до багатьох;
- “Car types” та “Cars” — зв’язок один до багатьох.

2.8 Уточнена діаграма архітектури сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

У цьому розділі буде представлена уточнена діаграма архітектури сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів.

Під час детального проектування було проведено глибокий аналіз груп вимог, які лягли в основу попередньої архітектури сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів. На основі цього аналізу були чітко визначені ключові компоненти системи та їх взаємодія один з одним.

В результаті була побудована більш детальна та вдосконалена архітектура застосунку, представлена у вигляді діаграми компонентів (рис. 2.12). Ця діаграма чіткіше відображає всі функціональні та нефункціональні вимоги проекту.

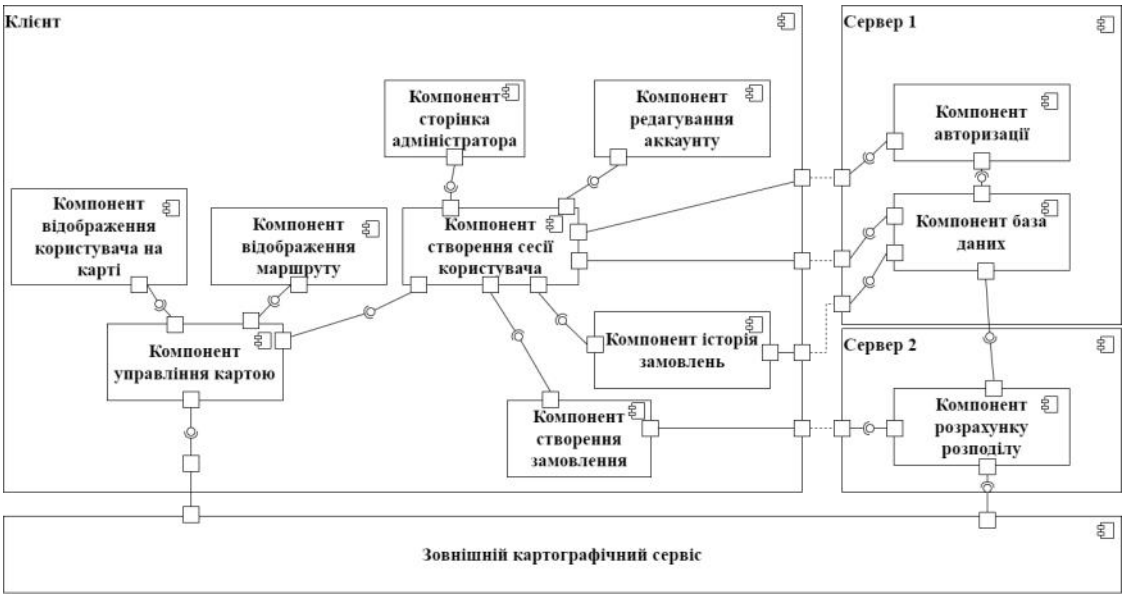


Рисунок 2.12 – Діаграма компонентів сервісу замовлення таксі

Як можна побачити, на діаграмі компонентів виникли зміни щодо попередньої архітектури, що була представлена у вигляді діаграми пакетів. Додався новий компонент “Зовнішній картографічний сервіс”, який відповідає

за інтеграцію з картографічним сервісом для відображення карти в застосунку. Також цей сервіс взаємодіє з серверною стороною застосунку. Картографічні сервіси відповідають за визначення маршруту на основі вхідних даних, у той час як серверна сторона відповідає за обробку цього маршруту, включаючи розподіл клієнтів між автомобілями та розрахунок розподілу суми до оплати.

Коли користувач пробує авторизуватися в застосунку, компонент “Створення сесії користувача” взаємодіє з компонентом “Авторизації”, який відповідає за перевірку валідності даних. Коли користувач реєструє новий обліковий запис, компонент “Створення сесії користувача” звертається до компоненту “База даних” для збереження даних. Під час створення замовлення клієнтом компонент “Створення замовлення” робить запит до компоненту “Розрахунок розподілу”, що розташований на окремому сервері, а сам компонент звертається до зовнішнього картографічного сервісу задля отримання географічних даних маршруту.

Висновки до другого розділу

У другому розділі було проведено моделювання бізнес-процесів для сервісу замовлення таксі з можливістю частково-спільної поїздки для пов'язаних груп клієнтів та визначено основну групу акторів. Далі були сформовані користувацькі історії та визначені основні потреби від сервісу. Також було проведено аналіз вимог за допомогою UML-діаграм для кожного актора. Були сформовані функціональні та нефункціональні вимоги з урахуванням усіх аспектів сервісу. Було обрано архітектурний шаблон та проведено попереднє архітектурне проектування за допомогою діаграми пакетів. Також було спроектовано концептуальну та логічну моделі бази даних та наостанок було побудовано уточнену діаграму архітектури на діаграмі компонентів.

РОЗДІЛ 3 РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ СЕРВІСУ ЗАМОВЛЕННЯ ТАКСІ З МОЖЛИВІСТЮ ЧАСТКОВО-СПІЛЬНОЇ ПОЇЗДКИ ПОВ'ЯЗАНОЇ ГРУПИ КЛІЄНТІВ

3.1 Вибір мови, зовнішніх сервісів та середовища програмування для реалізації сервісу замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів

Щоб реалізувати програмний продукт "Сервіс замовлення таксі з можливістю частково-спільної поїздки пов'язаної групи клієнтів" слід обрати інструменти, які дозволять ефективно втілити функціональні вимоги проекту та забезпечити його стабільну роботу.

Для розробки даного застосунку під Android було обрано мову програмування Java [11]. Її вибір обумовлений широким спектром бібліотек та фреймворків, що полегшують розробку та підтримку застосунків; також вона забезпечує зручний синтаксис та багато функціональних можливостей. Використання Java також дозволяє забезпечити широку сумісність з різними пристроями на базі Android.

Для реалізації мобільного застосунку було обрано Android Studio — інтегроване середовище розробки від Google, що спеціально призначене для розробки Android застосунків. Android Studio забезпечує швидку розробку та налагодження застосунків а також підтримку різних версій Android [12].

Для відображення карти та роботи з географічними даними було обрано сервіс MapBox. Він має потужні інструменти для інтеграції інтерактивних карт у застосунок, включаючи показ маршрутів, маркерів та інших елементів карти. Використання MapBox дасть користувачам можливість зручно взаємодіяти з картою та забезпечить високу точність відображення маршрутів та локацій.

Для забезпечення безпеки та зручності авторизації користувачів та управління базою даних було обрано використання Firebase [13]. Він надає

засоби для реалізації авторизації користувачів, а також інструменти для зберігання та управління даними в режимі реального часу. Це дозволить забезпечити ефективну роботу застосунку та високу безпеку даних користувачів

Для реалізації логіки розподілу клієнтів та розрахунку суми до оплати було вирішено створити власний сервер використовуючи мову програмування C# [14] та фреймворк ASP.NET [15]. Використання цієї мови програмування дозволить створити ефективний та надійний сервер, який забезпечить швидку обробку даних та запитів від клієнтів.

Обрані інструменти та технології дозволять ефективно втілити функціональні вимоги програмного продукту та забезпечать його стабільну роботу.

3.2 Структурна схема реалізації сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Структурна схема сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів буде зображена за допомогою діаграми компонентів з врахуванням доуточненої архітектури програмного забезпечення (рис. 3.1).

Як можна побачити, на структурній схемі наявні певні зміни по відношенню до уточненої архітектури, що була зображена на діаграмі компонентів (рис. 2.12). Компонент “Сторінка адміністратора” включає в себе ще три компоненти: компонент “Блокування користувачів”, “Перегляд заявок” та “Редагування цін”. Також змін зазнав компонент “Створення сесії користувача”, до нього також додався новий компонент, а саме: компонент “Створення заявки на зміну ролі”.

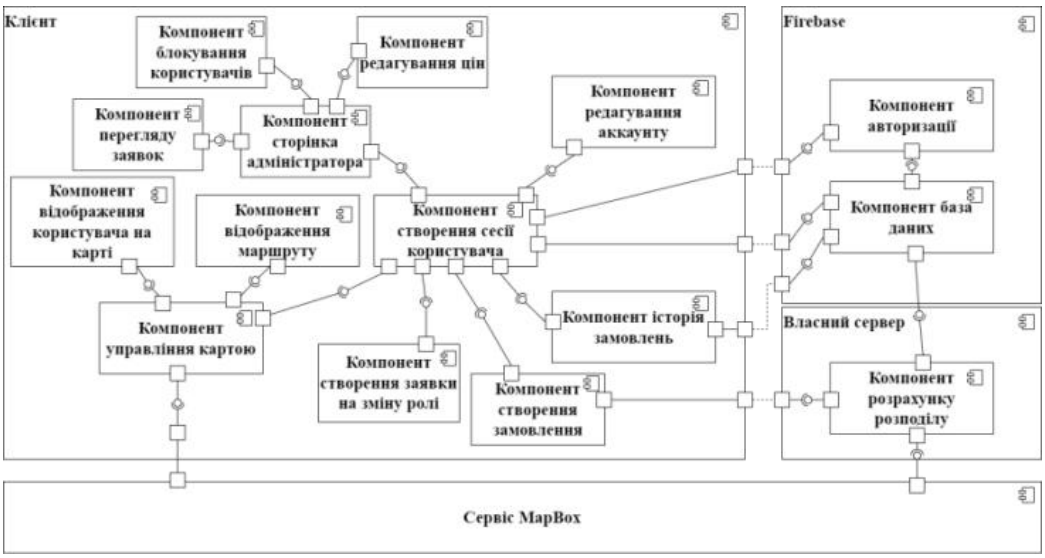


Рисунок 3.1 – Структурна схема реалізації сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

3.3 Розробка модулів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

В даному розділі будуть більш детально розглянуті компоненти необхідні для реалізації сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів.

3.3.1 Розробка компоненту “Створення замовлення”

Даний компонент призначений для створення замовлення та відображення рекомендацій щодо розподілу клієнтів та суми до оплати. Спочатку користувач обирає тип поїздки після чого робиться запит до сервера де відбуваються розрахунки. Якщо все пройшло успішно і від серверу був отриманий результат то користувачу відобразиться інформація про замовлення з рекомендованим розподілом суми до сплати, у разі наявності рекомендацій, щодо розподілу клієнтів вони також будуть відображені для користувача. Користувач матиме змогу обрати чи створити замовлення на одну машину, чи зробити замовлення відповідно до рекомендацій. Приклад реалізації даного компоненту можна розглянути на блок-схемі (рис. 3.2).

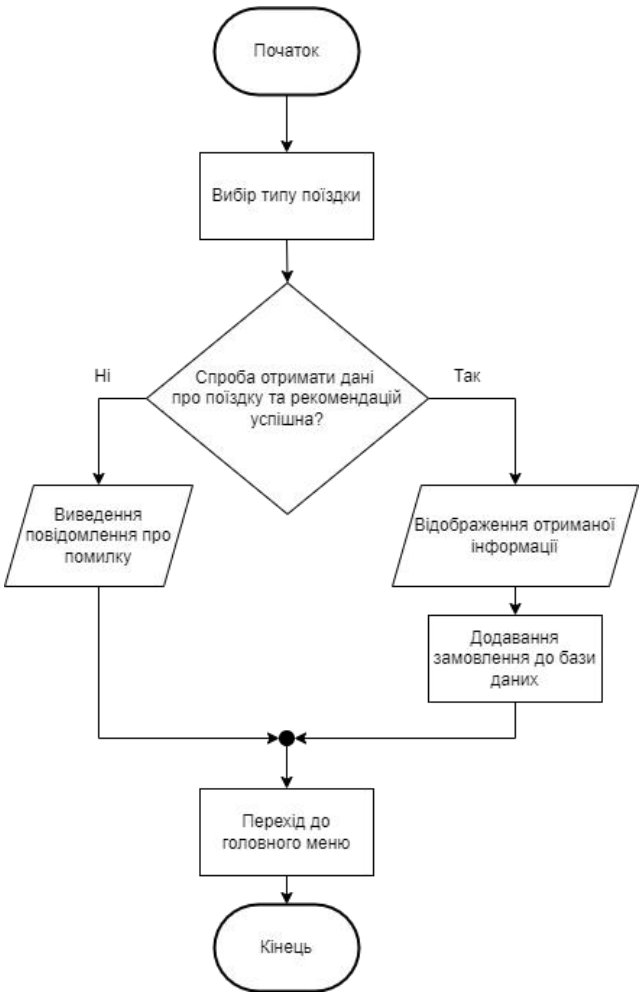


Рисунок 3.2 – Блок-схема компоненту “Створення замовлення”

3.3.2 Розробка компоненту “Розрахунок розподілу”

Даний компонент являє собою окремий сервер, що відповідає за розподіл клієнтів та суми до сплати. Розподіл клієнтів буде виконуватися за допомогою алгоритму вибору найдешевшого розбиття по машинам таксі [16]. Даний алгоритм використовує динамічне програмування для вирішення задачі оптимізації маршрутів таксі. Спочатку формується двовимірний масив, що зберігає відстані між кожною заданою точкою. Після цього формується масив маршрутів, де індекс є числовим значенням бітової маски, що являє собою множину точок через які потрібно проїхати. Для кожної бітової маски шукається шлях з найменшою довжиною. Тривіальними підзадачами вважаються моменти, де пасажир єдиний, а сумарний шлях є шляхом для цього пасажира.

Для розрахунку найдешевшого способу розвезення формується масив, що зберігатиме значення вартості поїздки. В даному масиві значення індексу також є числовим значенням бітової маски. Для всіх 1-елементних множин значення ціни буде відповідати вартості поїздки однією машиною. Для груп з більшою кількістю осіб, поєднання поїздки в одному таксі може бути вигідним, якщо фінішні точки розташовані близько один до одного, або невигідним, якщо фініші у надто різних напрямках від старту. Для кожної бітової маски розглядаються всі її підмножини, які можна розвести одним авто. Потім розраховується сума витрат на дане таксі, якщо значення менше ніж те,що вже записане, тоді воно записується в масив, а також зберігаються значення бітових масок, що утворили мінімальну суму. При цьому, якщо бітова маска являє собою множину з декількох точок, то вартість цієї поїздки також може відповідати перевезенню хоч однією машиною, хоч кількома.

Оскільки вибір оптимального розподілу людей по автомобілям таксі робиться рекурсією, а типовою проблемою рекурсивних реалізацій таких алгоритмів є велика кількість повторних розв'язувань раніше вже розв'язаних задач, слід перевірити, чи нема такої проблеми в пропонованій реалізації.

Класичні описи динамічного програмування (наприклад, розд. 15.3 книги [17]) рекомендують у випадку перекривань підзадач використовувати запам'ятовування вже розв'язаних підзадач.

У пропонованій реалізації, практично цей самий ефект досягається шляхом того, що при рекурсивних викликах методу GetAnswer відбувається перевірка, чи відповідна підзадача вже розв'язувалася раніше, й виклики відбуваються лише для досі нерозв'язаних підзадач.

Також на сервері будуть виконуватись розрахунки щодо розподілу суми до сплати між клієнтами. Реалізації даних алгоритмів можна більш детально розглянути на блок-схемах (рис. 3.3 та рис. 3.4). Результат з серверу буде приходити у вигляді JSON, після чого він буде конвертований в об’єкт.

На рисунку 3.3 зображена блок-схема реалізації алгоритму вибору найдешевшого розбиття по машинам таксі.

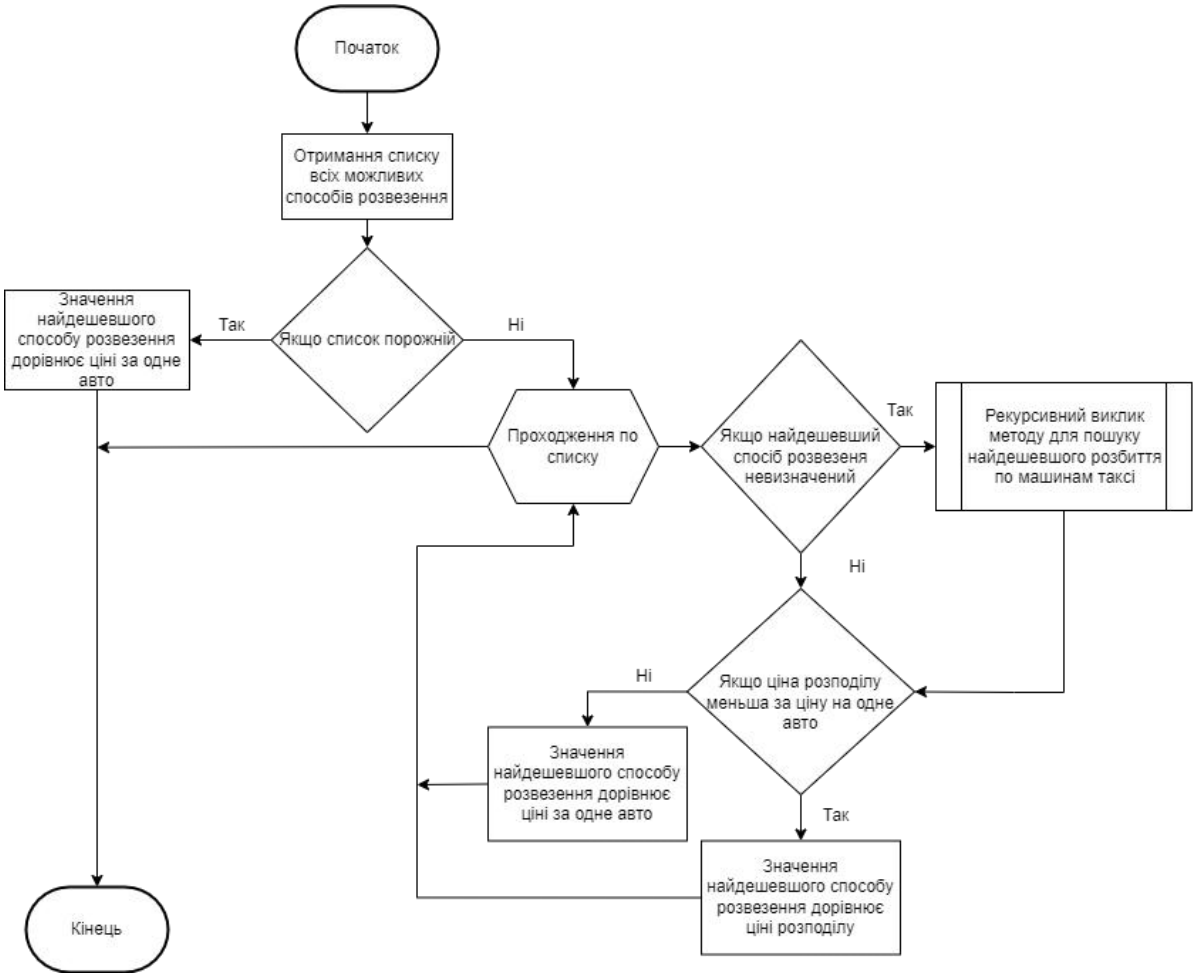


Рисунок 3.3 – Блок-схема алгоритму вибору найдешевшого розбиття по машинам таксі

Даний метод знаходить найдешевший спосіб розвезення підмножин пасажирів, причому кожна підмножина представлена у вигляді бітової маски, тобто якщо маршрут складається з 6 фінішних точок, то бітова маска матиме вигляд 111111, що являє собою число 63. Після цього формується список всіх можливих підмножин заданої бітової маски, який представлятиме собою можливі комбінації для спільної поїздки. Далі йде перебір комбінацій підмножин та перевіряється чи розрахована найнижча ціна за поїздку для кожної підмножини, якщо так, то після цього обчислюється сума цін двох підмножин, в іншому випадку робиться рекурсивний виклик методу для розрахунку найнижчої ціни для заданої підмножини. Далі робиться перевірка суми цін двох підмножин, якщо вона менша за ціну для одного авто,а також

менша за значення, що вже збережене в масив, тоді вона зберігається в масиві найнижчих цін за поїздку. Також зберігаються значення підмножин, які являють собою розподіл клієнтів. У випадку, коли найнижча ціна відповідає ціні за одне авто значення підмножин зберігатися не будуть. В результаті в масиві найнижчих цін під індексом, що відповідає бітовій масці маршруту буде знаходитись значення найнижчої ціни. Також буде створена словникова структура, ключем в якій буде бітова маска, значенням буде список, в якому будуть записані значення бітових масок, що дали найнижчу ціну.

На рисунку 3.4 зображена блок-схема реалізації алгоритму розрахунку розподілу суми до сплати з використанням вектора Шеплі.

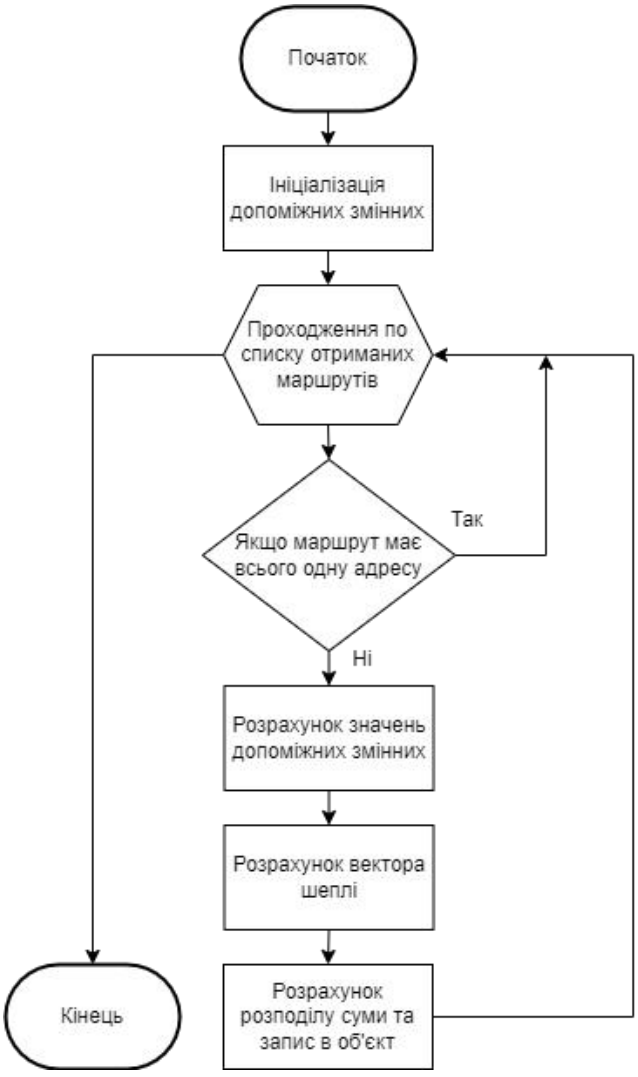


Рисунок 3.4 – Блок-схема алгоритму розрахунку розподілу суми до оплати

На даному малюнку можна побачити роботу методу для розрахунку розподілу суми до оплати. На початку ініціалізуються змінні, а саме створюються два масиви один з яких зберігатиме значення ціни поїздки однією машиною, а другий міститиме в собі значення, скільки можна зекономити коштів за цю поїздку. Ці значення розраховуватимуться наступним чином, від значення ціни однією машиною віднімається значення суми цін однією машиною для кожного учасника, що входить в маршрут. Якщо маршрут складається з однієї адреси то розподіл рахуватися не буде. Після цього будується двовимірний масив, де кількість рядків відповідатиме кількості перестановок клієнтів, а кількість колонок відповідатиме кількості учасників маршруту. Далі робиться перебір всіх перестановок та розрахунок зекономлених коштів для кожної перестановки, враховуючи порядок учасників. Після цього значення зекономлених коштів кожного учасника сумуються та діляться на кількість перестановок, в результаті буде отримано масив зі значеннями вектора Шеплі. Отримавши значення вектору Шеплі ми розраховуємо, хто скільки мусить заплатити за поїздку, для цього від ціни за поїздку однією машиною відповідного учасника віднімається його значення вектора Шеплі, отримане значення і буде його платою за поїздку. В кінці зберігаємо отриманий розподіл суми в об’єкт.

3.3.3 Розробка компоненту “Історія замовлень”

Компонент призначений для для створення списку замовлень клієнта. Клієнт матиме змогу переглянути всі свої створені замовлення, а разі необхідності натиснути на якесь конкретна замовлення після чого відкриється вікно з детальною інформацією про замовлення. Елементи цього списку будуть відсортовані по даті створення замовлення. Більш детально розглянути процес відображення списку замовлень можна на блок-схемі (рис. 3.5).

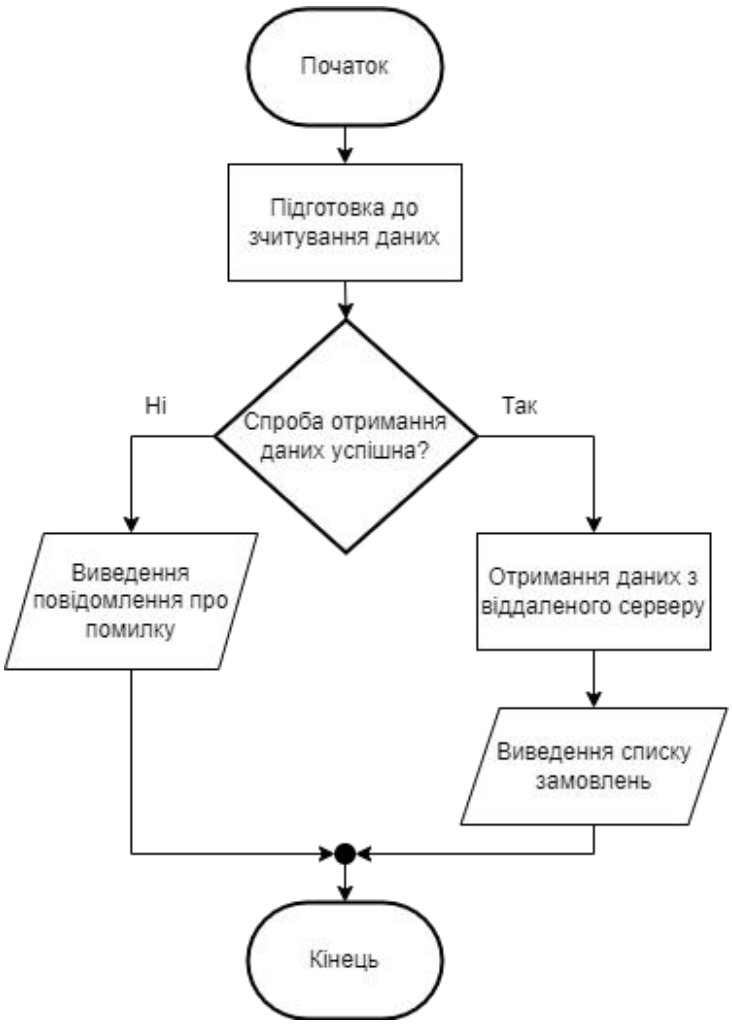


Рисунок 3.5 – Блок-схема компоненту “Історія замовлень” сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Як можна побачити на даній блок-схемі, спочатку робиться спроба отримати дані з бази даних на віддаленому сервері, у разі невдачі буде виведено повідомлення про помилку та відображений пустий список, в іншому випадку всі створенні користувачем замовлення будуть додані до списку та відображені для користувача.

3.3.4 Розробка бази даних Firebase Realtime для сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Для забезпечення збереження та синхронізації даних сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів, буде

використовуватися Firebase Realtime Database. Ця база даних є нереляційною (NoSQL) і працює у форматі об’єктів JSON.

Для початку роботи з даною базою даних необхідно створити проект на вебсайті Firebase та додати файл «google-service.json» до проекту Android. В даному файлі містяться конфігураційні дані проекту Firebase.

Для збереження інформації використовується метод setValue(). Він дозволяє передати в нього об’єкт, дані якого будуть збережені в базі даних. Для цього необхідно створити класи з необхідними полями, які можна буде передати в цей метод. Вигляд одного з класів який використовується для запису даних буде наведено в додатках (Додаток А). Задля того, щоб додати дані в базу даних спочатку потрібно створити унікальний ключ та вказати шлях (це можна зробити за допомогою методу child()). Код збереження даних в Firebase Realtime Database буде наведений в додатках (Додаток Б).

Щоб зчитувати дані з бази даних використовується метод getValue(). Цей метод дозволяє отримати дані та одразу перетворити їх в екземпляр відповідного класу.

Для видалення даних використовується метод removeValue(). Спочатку потрібно вказати шлях до об’єкту після чого викликати цей метод. В результаті дані будуть повністю видалені з бази даних.

Далі розглянемо більш детально кожну модель, що була створена для збереження даних до бази даних. Опираючись на концептуальну та логічну моделі бази даних були реалізовані наступні моделі:

- Users;
- Cars
- CarTypes
- Orders
- Roles
- BlockedUsers
- Statements

Модель Users (табл. 3.1) зберігає в собі інформацію про користувача. Атрибут uid є унікальним ключем запису. Атрибут name зберігає ім’я користувача. Атрибут lastname містить прізвище користувача. Атрибут password зберігає пароль. Атрибут phone містить номер телефону. Атрибут registrationDate містить дату реєстрації. Атрибут roleId зберігає ключ ролі. Атрибут urlImage містить url-адресу на фото. Атрибут email зберігає електрону пошту користувача.

Таблиця 3.1 — Модель Users

№	Назва поля	Тип даних
1	uid	String
2	email	String
3	lastname	String
4	name	String
5	password	String
6	registrationDate	String
7	roleId	String
8	urlImage	String

Модель Orders (табл. 3.2) зберігає в собі інформацію про замовлення. Атрибут uid є унікальним ключем. Атрибут addresses зберігає назви адрес. Атрибут destributionPrice містить розподіл суми до сплати. Атрибут driverId зберігає унікальний ключ водія. Атрибут duration містить інформацію про тривалість поїздки. Атрибут orderDate містить дату створення замовлення. Атрибут orderStatus зберігає статус замовлення. Атрибут route містить значення координат заданих адрес. Атрибут userId зберігає унікальний ключ користувача. Атрибут price містить ціну за замовлення.

Таблиця 3.2 — Модель Orders

№	Назва поля	Тип даних
1	addresses	String
2	destributionPrice	String
3	driverId	String
4	duration	String
5	orderDate	String
6	orderStatus	String
7	price	String
8	route	String
9	uid	String
10	userId	String

Модель Roles (табл. 3.3) зберігає в собі інформацію про роль. Атрибут uid є унікальним ключем. Атрибут name зберігає назву ролі.

Таблиця 3.3 — Модель Roles

№	Назва поля	Тип даних
1	uid	String
2	name	String

Модель Cars (табл. 3.4) зберігає в собі інформацію про авто. Атрибут uid є унікальним ключем. Атрибут carModel зберігає назву моделі авто. Атрибут carPlate містить значення номерного знаку авто. Атрибут carTypeId зберігає унікальний ключ типу авто. Атрибут driverId містить унікальний ключ водія.

Таблиця 3.4 — Модель Cars

№	Назва поля	Тип даних
1	uid	String

Продовження таблиці 3.4

1	2	3
2	carModel	String
3	carPlate	String
4	carTypeId	String
5	driverId	String

Модель CarTypes (табл. 3.5) зберігає в собі інформацію про ціни за певний тип авто. Атрибут uid є унікальним ключем. Атрибут name зберігає назву типу авто. Атрибут priceForCar містить значення ціни за виклик авто. Атрибут pricePerKm зберігає значення ціни за кілометр маршруту.

Таблиця 3.5 — Модель CarTypes

№	Назва поля	Тип даних
1	uid	String
2	priceForCar	Double
3	pricePerKm	Double
4	name	String

Модель BlockedUsers (табл. 3.6) зберігає в собі інформацію про заблокованих користувачів. Атрибут uid є унікальним ключем. Атрибут name зберігає назву типу авто. Атрибут userId містить унікальний ключ користувача. Атрибут date зберігає дату блокування користувача.

Таблиця 3.6 — Модель BlockedUsers

№	Назва поля	Тип даних
1	uid	String
2	userId	String
3	date	String

Модель Statements (табл. 3.7) зберігає в собі інформацію про заявку на зміну ролі. Атрибут uid є унікальним ключем. Атрибут userId зберігає унікальний ключ користувача. Атрибут statementText містить текст заяви. Атрибут statementDate зберігає дату створення заявки.

Таблиця 3.7 — Модель Statements

№	Назва поля	Тип даних
1	uid	String
2	userId	String
3	statementText	String
4	statementDate	String

3.3.5 Розробка інтерфейсу сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

У даному розділі буде розглянуто інтерфейс сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів для кожної ролі користувачів. Описано основні елементи і функції до яких матиме доступ користувач під час використання застосунку.

На рисунку 3.6 зображено макет інтерфейсу головного меню клієнта.

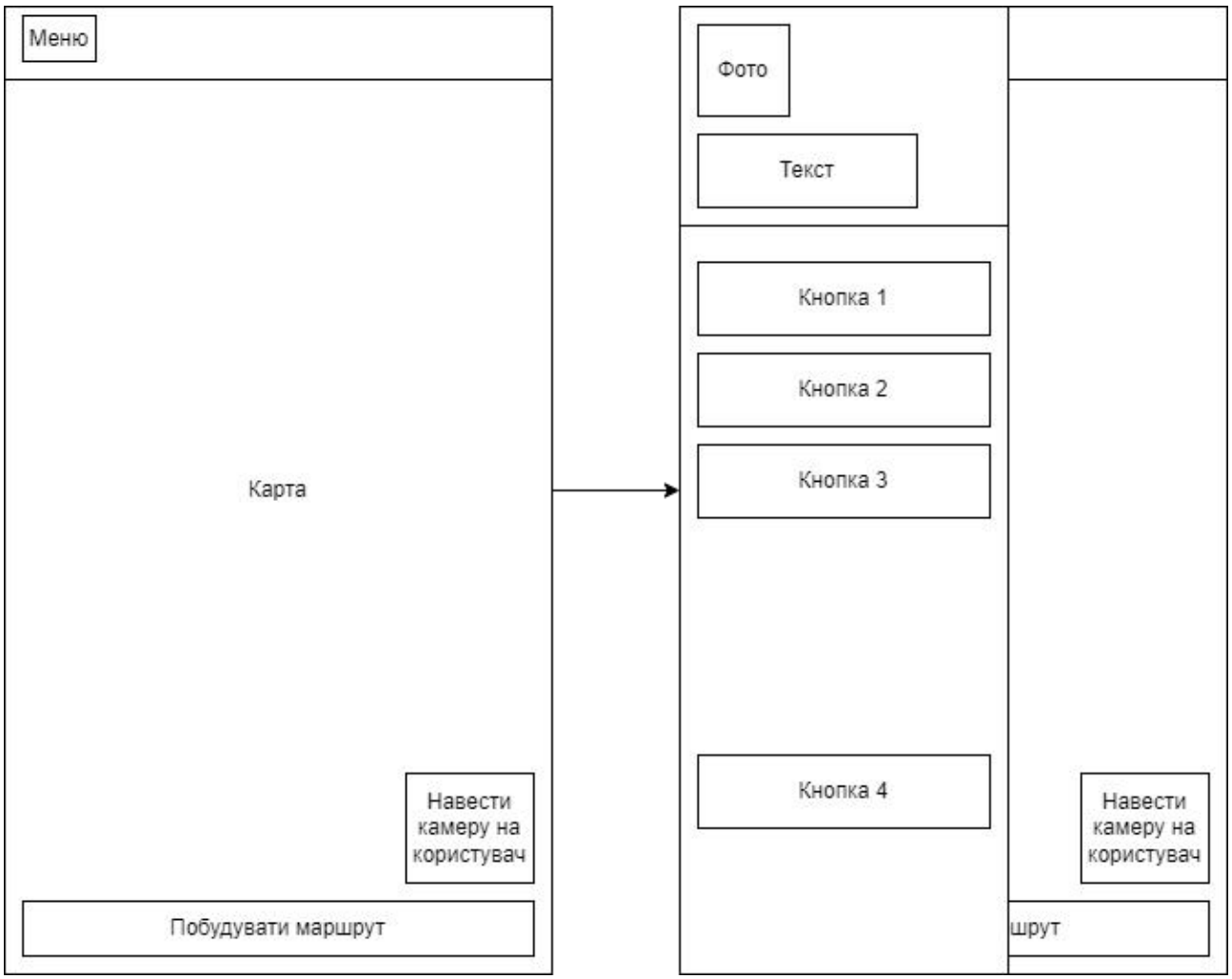


Рисунок 3.6 – Макет інтерфейсу головного меню користувача клієнта

На даному рисунку зображено шаблон вікна, що зустріне користувача після авторизації в застосунку. Перед клієнтом одразу відкриється карта на якій буде зображено його поточне місце находження, кнопка для побудови маршруту, кнопка, яка буде фокусувати камеру на позиції користувача після її натискання та кнопка для відкриття бокової навігаційної панелі. На самій панелі буде відображена інформація про клієнта та кнопки для навігації по іншим вікнам. Користувач може перейти до свого профілю натиснувши кнопку 1, кнопка 2 відкриє головне меню, кнопка 3 відкриє вікно історії замовлень та кнопка 4 відкриє форму для створення заявки на отримання ролі водія.

Далі розглянемо рисунок 3.7, на якому зображено вигляд макету розділу “Історія замовлень”.

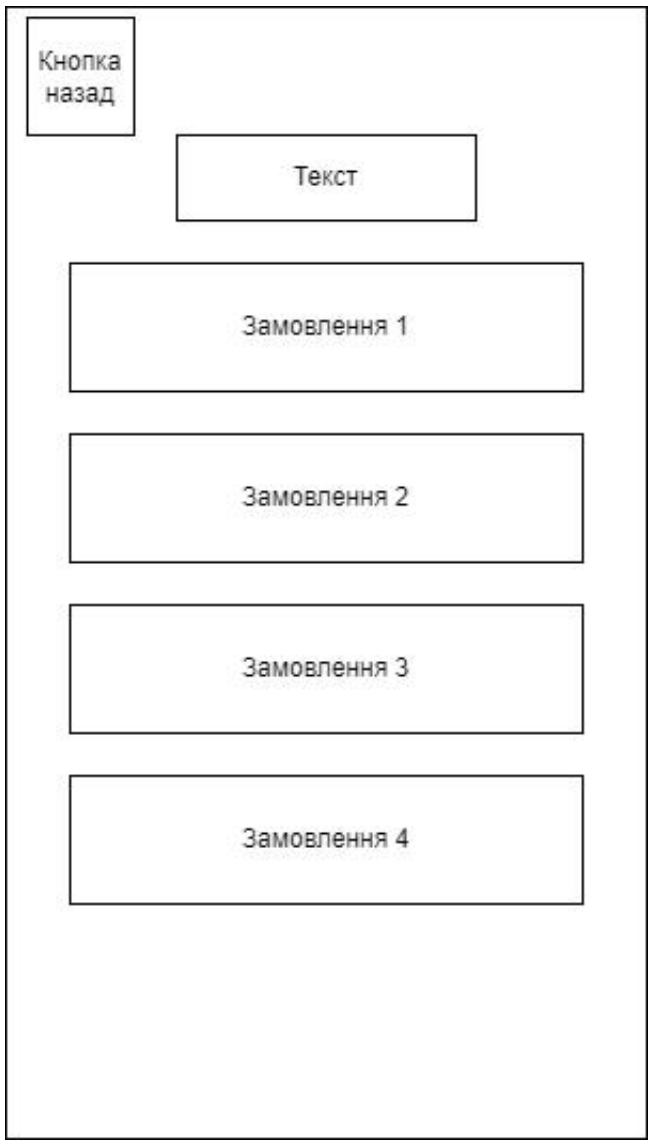


Рисунок 3.7 – Макет інтерфейсу розділу “Історія замовлень”

На даній картинці зображено вигляд розділу “Історія замовлень”. Він відповідає за відображення всіх створених замовлень клієнта та показує деяку інформацію про саме замовлення, а саме адреси, статус замовлення, ціну та тривалість. Даний макет містить наступні елементи “кнопка назад”, після її натискання користувач повернеться в головне меню, елемент “Текст”, що міститиме в собі назву розділу “Історія замовлень”, та декілька елементів, що відобразять замовлення користувача.

На рисунку 3.8 зображено макет вікна створення замовлення.

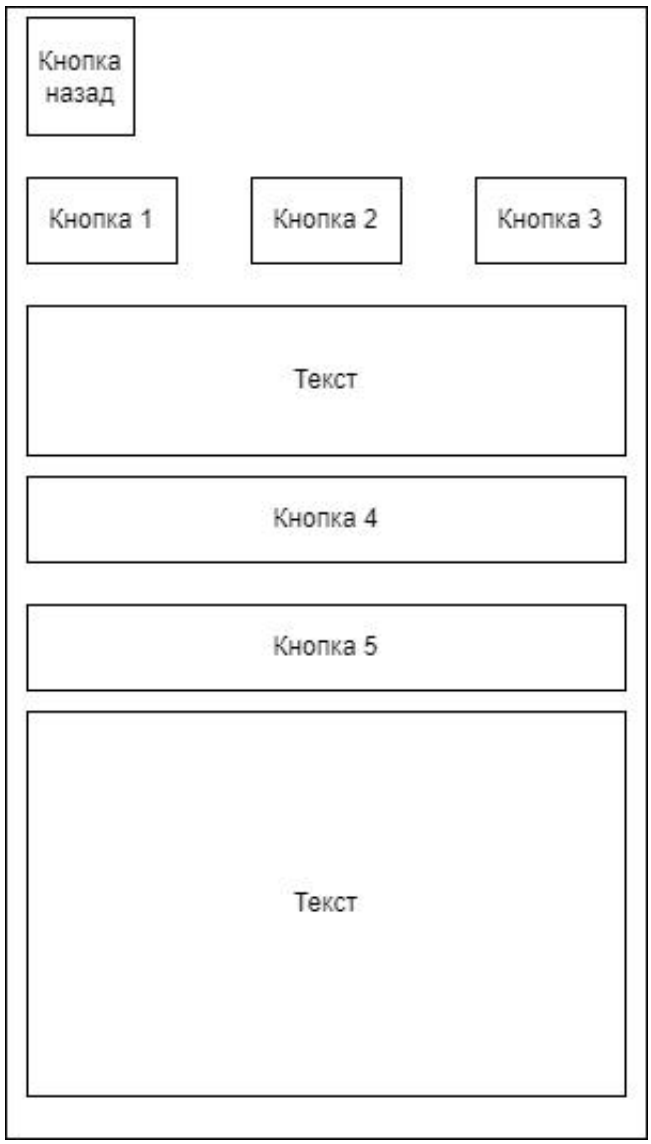


Рисунок 3.8 – Макет вікна створення замовлення

На даному макеті зображено вікно створення замовлення на якому відображені всі необхідні елементи, а саме кнопки для вибору типу поїздки (кнопка 1, кнопка 2, кнопка 3), кнопка 4 відповідає за створення замовлення на одне авто, кнопка 5 відповідає за створення замовлення відповідно до рекомендацій, поле в якому відображається інформація про поїздку, якщо брати одне авто та поле в якому відображаються рекомендації щодо розподілу клієнтів між авто для більш економічно вигідного варіанту розвезення, також відображається інформація про те хто скільки повинен заплатити, враховуючи місце, куди їде клієнт.

Розглянемо вигляд макету інтерфейсу профілю користувача (рис. 3.10). На ньому відображається інформація про користувача: фото, ім’я та прізвище, номер телефону. Також там розташовані кнопки які відповідають за певний функціонал: кнопка 1 відповідає за відкриття вікна в якому користувач може змінити свої дані, кнопка 2 — після натискання відкриється вікно в якому користувач матиме змогу змінити свій пароль, кнопка 3 відповідає за видалення акаунту, після натискання з’явиться діалогове вікно для підтвердження дії після чого, якщо користувач підтвердив дію акаунт буде видалено, кнопка 4 відповідає за вихід з облікового запису.

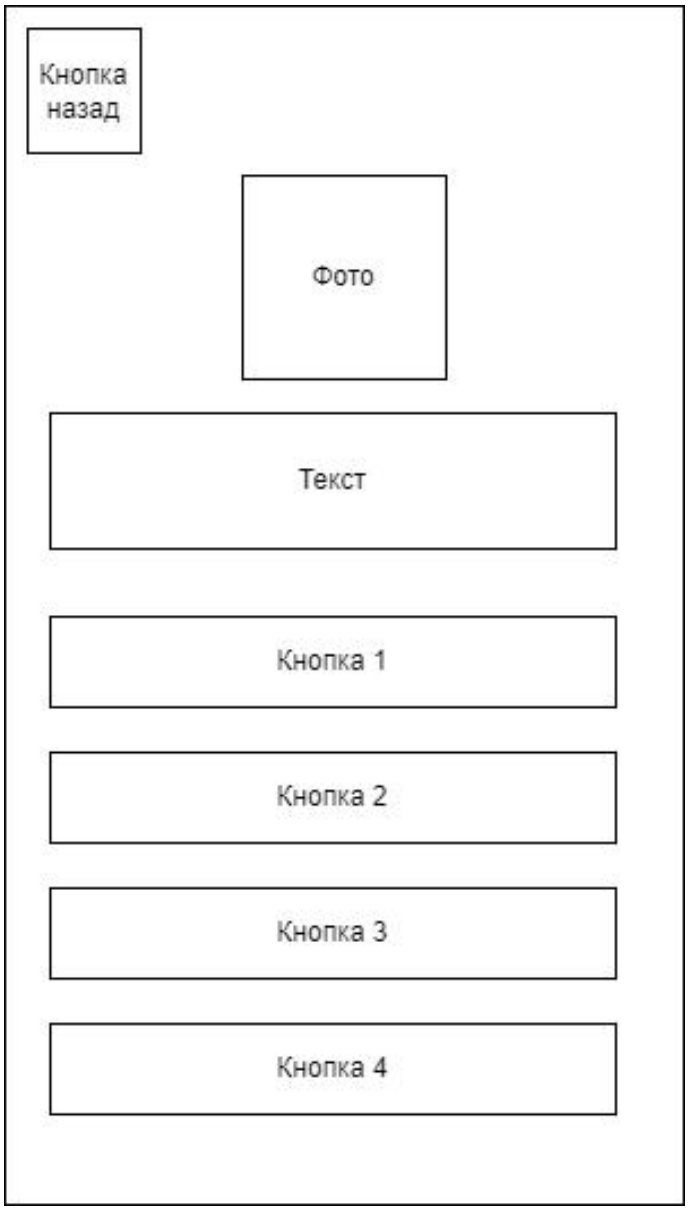


Рисунок 3.10 – Вигляд макету інтерфейсу профілю користувача

На рисунку 3.11 зображено макет головного меню адміністратора з усіма необхідними елементами: фото та поле для відображення імені адміністратора, кнопки, що відповідають за різний функціонал. Кнопка 1 призначена для відображення списку користувачів з можливістю блокування, кнопка 2 відповідає за додавання авто до бази даних, кнопка 3 відкриє вікно для перегляду списку заявок, кнопка 4 призначена для відображення списку заблокованих користувачів, кнопка 5 відкриє вікно для редагування цін, кнопка 6 відповідає за вікно, де можна назначити водія на автомобіль.

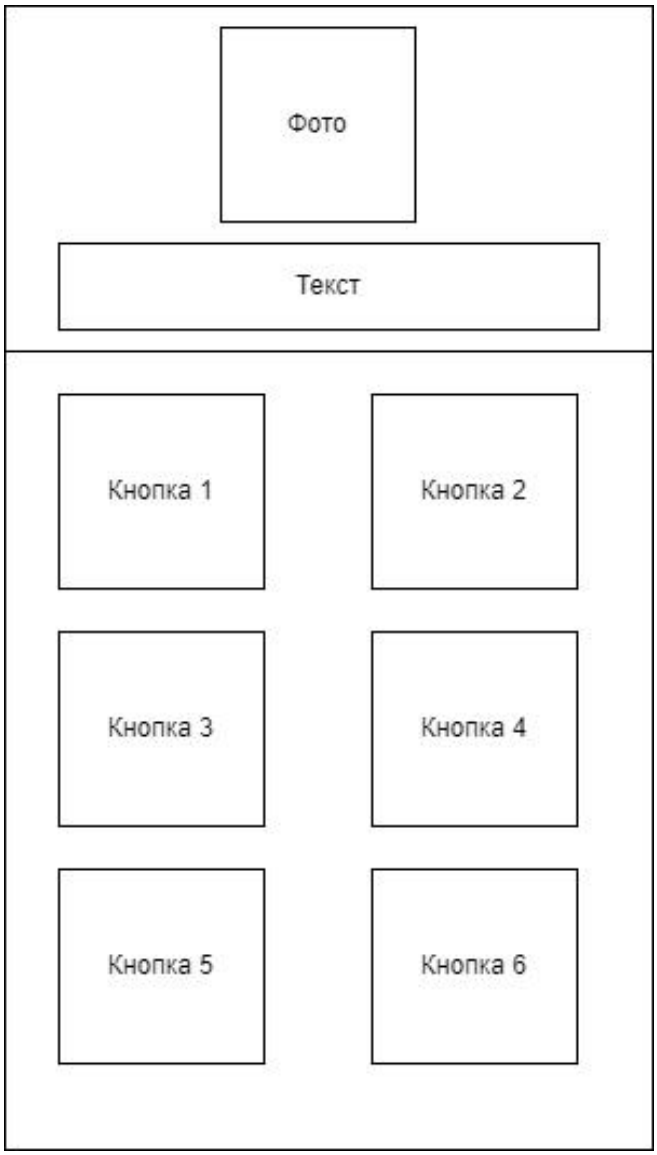


Рисунок 3.11 – Макет інтерфейсу головного меню адміністратора

На рисунку 3.12 зображені макети вікон авторизації та реєстрації. Вони відповідають за вхід користувача в обліковий запис та створення нового акаунту. Вікно авторизації включає в себе два поля для введення пошти та паролю, кнопку, що відповідає за вхід в акаунт та текстове поле, після натискання на яке відкриється вікно реєстрації нового облікового запису, а на вікні реєстрації наявні поля для ведення імені та прізвища, номеру телефону, електронної пошти, під якою буде створений акаунт, та поле для введення пароля та його підтвердження, кнопка, що відповідає за створення нового акаунту та текстове поле, після натискання на яке ми повернемося до вікна авторизації.

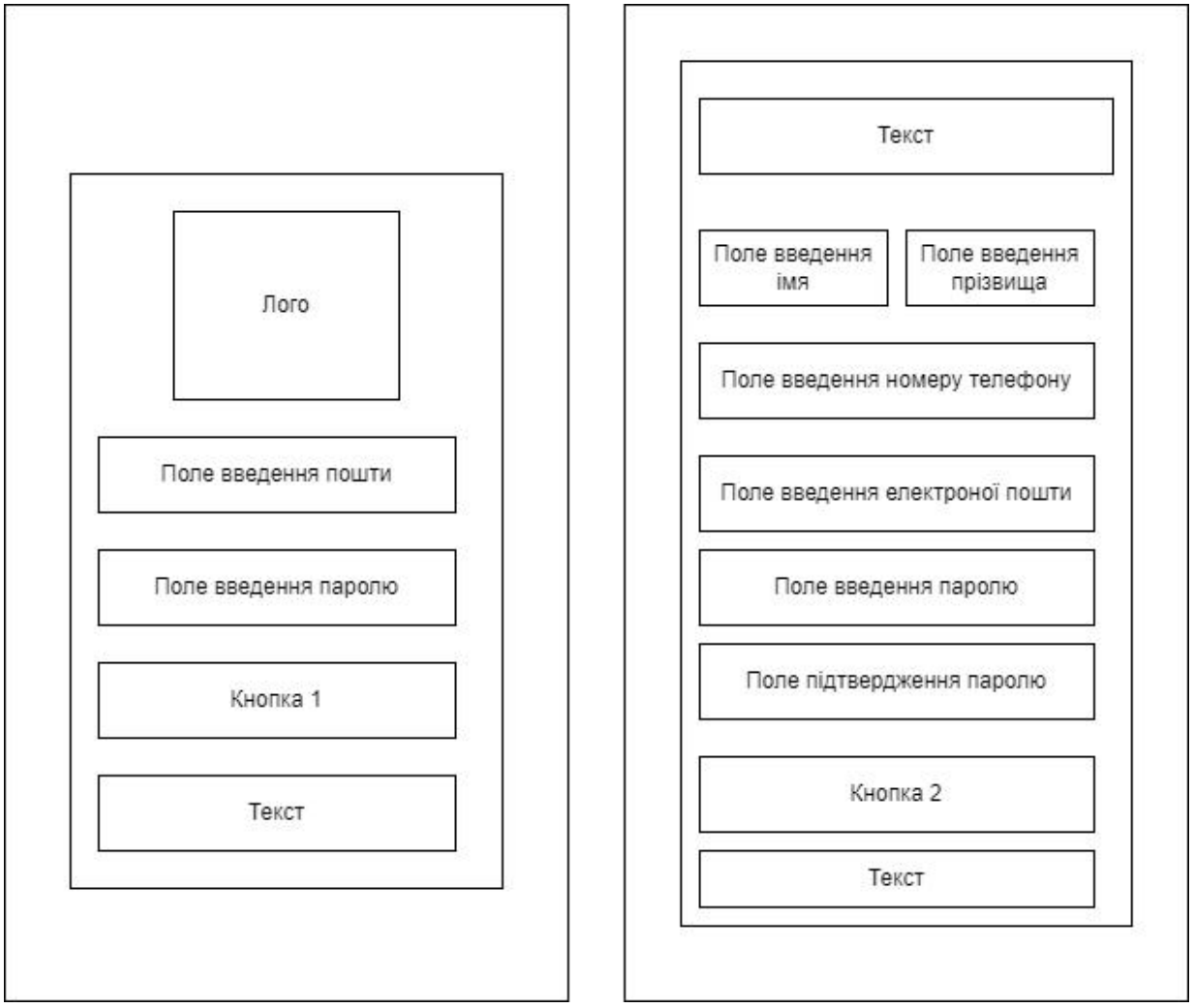


Рисунок 3.12 – Макети вікон авторизації та реєстрації

3.3.6 Узагальнена діаграма класів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

Реалізувавши основні компоненти, що необхідні для сервісу сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів можна побудувати діаграму класів. Вид узагальненої діаграми класів буде винесено в додатки (Додаток В).

Розглянемо більш детально класи що були відображені на діаграмі. Розпочнемо з класів що стосуються клієнта: UserHomeActivity — даний клас відповідає за відображення маршруту на карті та навігації по іншим розділам застосунку, StatementActivity — цей клас відповідає за створення запиту на отримання ролі водія, ClientOrderDetailsActivity — даний клас відповідає за відображення інформації про замовлення яке обрав клієнт, CreateGroupRideActivity — клас відповідає за побудову маршруту користувач може обрати точку стару та точки по яким потрібно всіх розвести, CreateRequestActivity — цей клас відповідає за відображення інформації про замовлення, створення замовлення та додавання його до бази даних, OrderHistoryActivity — даний клас призначений для відображення всіх замовлень користувача.

Тепер розглянемо класи що стосуються водія, а саме : DriverHomeActivity — даний клас відповідає за відображення користувача та маршруту на карті та навігації по іншим розділам, DriverTakeOrderActivity — цей клас відповідає за відображення всіх доступних замовлень, DriverOrderDetailsActivity — даний клас відповідає за відображення більш детальної інформації про замовлення та надає функціонал для прийняття замовлення.

Класи, що стосуються адміністратора: AdminHomeActivity — відповідає за навігацію між іншими розділами застосунку, AddCarActivity — призначений для додавання нового авто в базу даних, AssignDriverActivity — відповідає за призначення водія до авто, BlockedUserListActivity — цей клас відображає всіх заблокованих користувачів з можливістю їх розблокування, CarListActivity — в

даному класі зберігаються всі авто які не маються водія та відображаються для адміністратора, CategoryPricesActivity — відповідає за відображення типів поїздок та їх цін, EditCategoryPrices — цей клас призначений для редагування інформації про тип поїздки, StatementDetailsActivity — відповідає за відображення детальної інформації заяви, StatementsListActivity — призначений для відображення всих активних заявок на зміну ролі, UsersListActivity — клас що відповідає за відображення всіх клієнтів та водії з можливістю їх блокування.

Тепер розглянемо загальні класи: SplashScreenActivity — відповідає за відображення початкового екрану при запуску застосунку, LoginActivity — призначений для реалізації авторизації користувача в застосунок, RegisterActivity — цей клас відповідає за реєстрацію нового користувача та додавання його до бази даних, UserProfileActivity — відповідає за відображення інформації про користувача та надає функції виходу з облікового запису та видалення акаунту, ChangePasswordActivity — цей клас надає функціонал ля зміни паролю, ChangeUserInfoActivity — цей клас надає функціонал для зміни інформації користувача.

Також на діаграмі зображені такі класи як: Statement, Order, User, CarTypes, Car, BlockedUserModel. Вони призначені для взаємодії з нереляційною базою даних.

Отже, було зроблено огляд узагальненої діаграми класів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів та описано класи, які відображають функціональність системи. Кожен клас має визначені поля та методи, що дає можливість зрозуміти їхню роль в системі та взаємодію з іншими класами.

3.4 Тестування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

В даному підрозділі буде описано тестування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів. Цей етап є важливою складовою під час розробки програмного забезпечення, адже в ході тестування можна перевірити відповідність застосунку визначеним вимогам, виявити помилки та недоліки в роботі програми. Для реалізації даного пункту було обрано один з типів тестування, а саме функціональне тестування [18].

Функціональне тестування спрямоване на перевірку відповідності програмного забезпечення вимогам функціональності, які були визначені в розробленому Product Backlog. Оскільки робота має дуже істотні складові як багатокомпонентної програмної системи, так і нетривіальні алгоритми, то слід відтестувати роботу, як багатокомпонентної програмної системи, так і роботу реалізованого алгоритму. Результати тестування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів та алгоритму будуть відображені в додатках (Додаток Г та Додаток Д).

Переглянувши результати функціонального тестування можна дійти висновку, що застосунок повністю відповідає функціональним вимогам, що описані в Product Backlog. Виявлені помилки та недоліки були виправлені, а всі функції працюють стабільно.

Висновки до третього розділу

Виконуючи даний розділ було виконано низку задач пов’язаних з розробкою та тестуванням програмного продукту сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів.

Першим завданням був огляд та вибір мови та середовища програмування (мова Java, середовище Android Studio), та інших додаткових сервісів, що були необхідні для реалізації застосунку, а саме: сервіси Firebase, які відповідають за авторизацію та збереження даних та MapBox, що надає можливість

відображення карт в застосунку та отримання різних географічних даних. Також в ході виконання цього завдання було прийнято рішення реалізувати власний сервер на мові програмування C# для проведення різного роду розрахунків.

Після цього було проведено побудову структурної схеми програмного продукту (рис.3.1), а також описано реалізацію компонентів програмного забезпечення. Сама реалізація була описана за допомогою блок-схем. Також було розроблено інтерфейс сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів з його описом. Була побудована узагальнена діаграма класів (Додаток В) та описано про призначення кожного класу.

У фінальному етапі було проведено тестування застосунку на виявлення помилок та перевірку на відповідність до функціональних вимог. В результаті було виявлено що програмний продукт повністю відповідає зазначеним вимогам.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було визначено та описано актуальність даного програмного забезпечення. Було оглянуто та проаналізовано три аналоги сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів та виділені їх переваги та недоліки. Також було виконано аналіз підґрунтя та описано постановку задачі на розробку.

Було проведено моделювання бізнес-процесів для сервісу замовлення таксі з можливістю частково-спільної поїздки для пов'язаних груп клієнтів та визначено основну групу акторів. Далі були сформовані користувацькі історії та визначені основні потреби від сервісу. Також було проведено аналіз вимог за допомогою use-case діаграм для кожного актора. Були сформовані функціональні та нефункціональні вимоги з урахуванням усіх аспектів сервісу. Було обрано архітектурний шаблон (клієнт-серверна архітектура) та проведено попереднє архітектурне проектування за допомогою діаграми пакетів. Під час виконання детального проектування були розглянуті такі пакети як: пакет “Керування замовленнями”, пакет “Історія замовлень”, пакет “Розрахунок розподілу”, за допомогою UML-діаграм (діаграма послідовності, діаграма діяльності та діаграма комунікацій). Також було спроектовано концептуальну та логічну моделі бази даних розглянуті їх сутності, зв’язки та атрибути, а наостанок було побудовано уточнену діаграму архітектури на діаграмі компонентів.

Під час роботи над етапом розробки застосунку було обрано мову та середовище програмування, а також інші додаткові сервіси, що були необхідні для реалізації деяких функцій застосунку. Також був реалізований власний сервер який відповідає за знаходження більш оптимального способу розвезення клієнтів, а також розраховує розподілу суми до оплати. було проведено побудову структурної схеми на діаграмі компонентів з урахуванням доуточненої архітектури (рис.3.1), а також описано реалізацію компонентів

програмного забезпечення. Сама реалізація була описана за допомогою блок-схем. Також було розроблено інтерфейс сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів з його описом. Була побудована узагальнена діаграма класів (Додаток В) та описано про призначення кожного класу.

У фінальному етапі було проведено тестування застосунку на виявлення помилок та перевірку на відповідність до функціональних вимог, та сформовано таблицю яка відображає результати тестів (Додаток Г). Крім того, окремо (Додаток Д) була протестована суто алгоритмічна складова визначення оптимальних розподілів та маршрутів.

Розроблений сервіс має великий потенціал для подальшого вдосконалення і розвитку. Одним із таких вдосконалень може бути онлайн оплата. Додавання такого функціоналу може забезпечити зручність для клієнтів. Також можна додати можливість спілкування з водієм, такий функціонал може бути дуже корисним якщо клієнт захоче надати водію якусь додаткову інформацію або вирішити якісь питання. Ці покращення можуть зробити сервіс більш зручним та привабливим для користувачів, підвищити його конкурентоспроможність та покращити загальне враження від користування ним.

Отже, можна зробити висновок, що в ході роботи було успішно реалізовано функціонуючий сервіс, який повністю відповідає завданню кваліфікаційної роботи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Uber [Електронний ресурс]. Режим доступу: https://www.uber.com/ua/uk/about/?uclick_id=2ecda89d-7da1-4255-a127-70cb0ade07e0 Перевірено 05.05.2024.

2. Bolt [Електронний ресурс]. Режим доступу: https://bolt.eu/uk-ua/ Перевірено: 05.05.2024.

3. Uklon [Електронний ресурс]. Режим доступу: https://uklon.com.ua/online-order/?gad_source=1&gclid=Cj0KCQjw_-GxBhC1ARIsADGgDjt7QFGkBRpmOeh6OB9pZxPGHb03Je8ziz0-3oMX54ifotvI_FDaW2YaAj4hEALw_wcB Перевірен: 05.05.2024.

4. Алгоритм дейкстри [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%94%D0%B5%D0%B9%D0%BA%D1%81%D1%82%D1%80%D0%B8 Перевірено: 05.05.2024.

5. Алгоритм A* [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%BF%D0%BE%D1%88%D1%83%D0%BA%D1%83_A* Перевірено: 05.05.2024.

6. MapBox [Електроний ресурс]. Режим доступу: https://www.mapbox.com/ Перевірено: 05.05.2024.

7. Вектор Шеплі [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%BA%D1%82%D0%BE%D1%80_%D0%A8%D0%B5%D0%BF%D0%BB%D1%96 Перевірено: 05.05.2024.

8. MVC [Електронний ресурс]. Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller Перевірено: 05.05.2024.

9. SOA [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%A1%D0%B5%D1%80%D0%B2%D1%96%D1%81%D0%BD%D0%BE-%D0%BE%D1%80%D1%96%D1%94%D0%BD%D1%82%D0%BE%D0%B2%D0

%B0%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0 Перевірено: 05.05.2024.

10. Клієнт-серверна архітектура [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%9A%D0%BB%D1%96%D1%94%D0%BD%D1%82-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80%D0%BD%D0%B0_%D0%B0%D1%80%D1%85%D1%96%D1%82%D0%B5%D0%BA%D1%82%D1%83%D1%80%D0%B0 Перевірено: 05.05.2024.

11. Java [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Java Перевірено: 05.05.2024.

12. Android Studio [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio Перевірено: 05.05.2024.

13. Firebase [Електронний ресурс]. Режим доступу: https://firebase.google.com/ Перевірено: 05.05.2024.

14. C# [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/C_Sharp Перевірено: 05.05.2024.

15. ASP.NET [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/ASP.NET Перевірено: 20.05.2024.

16. Порубльов І.М. Задача визначення оптимального способу доїзду на таксі групи клієнтів зі спільним стартом і різними фінішами. Інформаційні моделюючі технології, системи та комплекси: V міжнародна науково-практична конференція. 18-19 квітня 2024 р., Черкаси : Черкаський національний університет імені Богдана Хмельницького, 2024. С. 65– 67.

17. Charles E. Leiserson , Ronald L. Rivest, Clifford Stein, Thomas H. Cormen. Introduction to Algorithms, Second Edition, MIT Press, 2001, P. 290-298.

18. Функціональне тестування [Електронний ресурс]. Режим доступу: https://uk.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F Перевірено: 05.05.2024.

ДОДАТОК А

Фрагмент коду класу який використовується для запису даних в базу даних

```
public class User {
    private String _uid;
    private String _name;
    private String _lastname;
    private String _email;
    private String _password;
    private String _urlImage;
    private String _phone;
    private String _roleId;
    private String _registrationDate;

    public User(String _uid, String _name, String _lastname,
String _email, String _password, String _urlImage, String _phone,
String _roleId, String _registrationDate) {
        this._uid = _uid;
        this._name = _name;
        this._lastname = _lastname;
        this._email = _email;
        this._password = _password;
        this._urlImage = _urlImage;
        this._phone = _phone;
        this._roleId = _roleId;
        this._registrationDate = _registrationDate;
    }
```

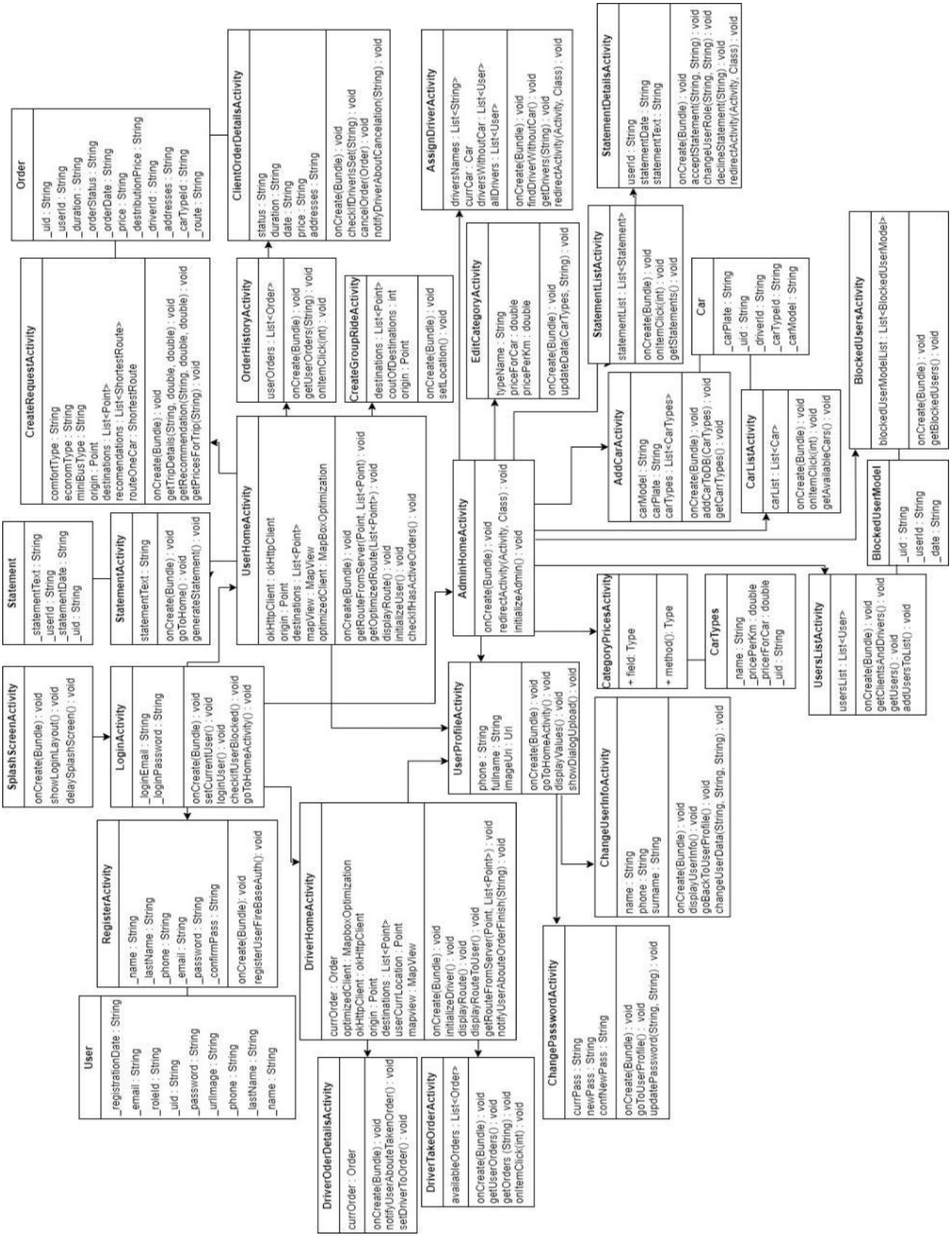
ДОДАТОК Б

Фрагмент коду запису даних в firebase realtime database

```
    Car car = new Car(uid,"",model,carTypes.get_uid(),plate);
    FirebaseDatabase.getInstance().getReference(Common.CARS_REFEREN
CE).child(uid).setValue(car).addOnCompleteListener(task -> {
     if(task.isSuccessful()){
    Toast.makeText(getBaseContext(),"Car    successfully    added",
Toast.LENGTH_SHORT).show();
                    }
                });
```

ДОДАТОК В

Узагальнена діаграма класів сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів



ДОДАТОК Г

Тестування сервісу замовлення таксі з можливістю частково-спільної поїздки пов’язаної групи клієнтів

1.№	2. Тест	3. Очікуваний результат	4. Фактичний результат
1	Авторизація з коректними даними	Вхід в акаунт та перехід до головного меню	Після того як користувач увів правильні дані для авторизації відкривається головне меню
2	Авторизація з некоректними даними	Виведення повідомлення про некоректні дані	Коли користувач вводить неправильні дані виводиться повідомлення про те що дані некоректні.
3	Авторизація заблокованого користувача	Виведення повідомлення про те що акаунт заблокований	Коли заблокований користувач пробує увійти в акаунт виводиться повідомлення про те, що цей акаунт був заблокований
4	Побудова маршруту	Відображення маршруту на карті	Після того як користувач обрав стартову та фінішні точки на карті відображається маршрут, що проходить через всі точки
5	Створення замовлення	Збереження замовлення в базу даних	Після того як користувач обрав ти поїздки та підтвердив замовлення воно додається до бази даних
6	Отримання рекомендацій	Виведення рекомендацій, щодо розподілу клієнтів	Якщо існує більш оптимальний спосіб розвезення, то він буде відображений для користувача

1	2	3	4
7	Отримання розподілу суми	Виведення розподілу суми до сплати.	Під час формування замовлення користувачу відображається інформація про те хто скільки мусить заплатити за дану поїздку.
8	Прийняття замовлення	Для водія прокладається маршрут до клієнта, а у клієнта з’являється повідомлення про те що його замовлення було прийняте.	Коли водій приймає замовлення одразу прокладається маршрут до клієнта, а у клієнта в цей час з’являється вікно в якому відображається ім’я водія та інформація про авто.
9	Блокування користувача	Додавання користувача до таблиці заблокованих користувачів.	Після блокування користувача він додається до таблиці заблокованих користувачів.
10	Редагування профілю	Збереження зміненої інформації про користувача	Змінені дані користувача зберігаються в базі даних
11	Видалення акаунту	Видалення даних користувача з бази даних	Після видалення акаунту дані користувача видаляються з бази даних
12	Перегляд історії замовлень	Виведення списку замовлень	Коли користувач відкриває розділ “Історія замовлень”, то йому відображаються всі його створені замовлення.
13	Скасування замовлення	Зміна статусу замовлення та інформування водія про скасування	Статус замовлення змінюється на “скасоване”, а водій отримує повідомлення про те, що замовлення було скасоване.

1	2	3	4
14	Завершення замовлення	Інформування клієнта про те що замовлення було завершене	Після завершення замовлення клієнт отримує повідомлення про те, що замовлення було виконано.
15	Зміна цін	Збереження змінених цін в базу даних	Після того як адміністратор змінив ціни на відповідну категорію поїздок, змінені дані записуються до бази даних
16	Реєстрація	Створення нового користувача та запис до бази даних	Після заповнення всіх полів валідними даними та натискання кнопки “Зареєструватися” створюється новий користувач та записується до бази даних

ДОДАТОК Д

Тестування алгоритму розподілу клієнтів між авто

1.№	2. Тест	3. Очікуваний результат	4. Фактичний результат
1	Створення поїздки з кількістю фінішних точок більше 4, що знаходять відносно близько один одного	Якщо для поїздки було обрано авто, що може помістити максимум 4 особи, то ми маємо отримати розподіл клієнтів на декілька авто, а якщо авто може вмістити всіх, то ніякого розподілу ми не отримуватимемо	Після створення поїздки з 5 фінішними точками та обрання авто, що може помістити максимум 4 особи, поїздку було розбито на 2 окремі авто. Клієнти були розподілені так: в одному авто поїхало 3 особи, а в іншому — 2. Обравши авто з більшою місткістю ніякого розподілу не було отримано, всіх можна було розвести одним авто.
2	Створення поїздки фінішні точки якої знаходяться в різних напрямках відносно старту	При створенні поїздки з чотирма фінішними точками, в яких три знаходяться відносно близько один одного, а остання розташована в іншому напрямку відносно старту, ми маємо отримати розподіл клієнтів на 2 окремих авто. В одному авто їде 3 особи, фінішні точки яких розташовані близько один до одного і окремо їде особа, точка якої знаходиться в іншому напрямку.	Після створення даної поїздки було отримано очікуваний результат, а саме: поїздка була розбита на два окремих авто. В одному поїхало 3 особи в іншому 1.

1	2	3	4
3	Створення поїздки з шістьма фінішними точками, п’ять з яких знаходяться в одному напрямку, а остання в іншому.	Якщо для поїздки було обрано авто, що може помістити максимум 4 особи, то ми маємо розподіл на три окремі авто. Клієнти повинні будуть бути розподілені наступним чином: особа, що їду в іншому напрямку буде їхати одна, а інші п’ять осіб точки яких знаходяться близько один до одного також будуть розподілені між окремими авто: троє їдуть в одній машині і двоє в іншій. Якщо ж було обрано авто з більшою місткістю, то в результаті ми маємо отримати розподіл лише на два авто.	Після створення даної поїздки було отримано очікуваний результат, а саме: якщо було обрано авто з місткістю в 4 особи, то було отримано очікуваний розподіл на три окремих авто, у випадку коли було обрано авто з місткістю в шість осіб поїздка було розподілена лише на два окремих авто

ДОДАТОК Е

Посилання на git-репозиторій із вихідним кодом програмного продукту

Вихідний код програми розміщений в Git-репозиторії:

Клієнт — https://github.com/Castomery/TaxiApp.

Сервер — https://github.com/Castomery/TaxiApp_Server.