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

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

_____Сергій СТІРЕНКО

«___»_____2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних систем»

спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Мобільний додаток для управління системою спільних поїздок»

Виконав:
студент IV курсу, групи ІП-62
Гук Микола Олексійович _____

Керівник:
Асистент,
Алєнін Олег Ігорович _____

Консультант з нормоконтролю:
Професор кафедри ОТ, д.т.н.,
Сімоненко Валерій Павлович _____

Рецензент:
Посада, науковий ступінь, вчене звання,
Прізвище, ім'я, по батькові _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Сергій СТІРЕНКО

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Гуку Миколі Олексійовичу

1. Тема проекту «Мобільний додаток для управління системою спільних поїздок», керівник проекту Алєнін Олег Ігорович, асистент, затверджені наказом по університету від «07» травня 2020 р. № 1081-с

2. Термін подання студентом проєкту _____

3. Вихідні дані до проекту: технічна документація, теоретичні дані, розроблений додаток для управління системою спільних поїздок.

4. Зміст пояснювальної записки: огляд існуючих рішень, проектування системи спільних поїздок, реалізація системи спільних поїздок, інструкція користувача

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)

Алгоритм пошуку спільного маршруту(1 шт.)

Діаграма бази даних(1 шт.)

Структурна схема системи(1 шт.)

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сімоненко В.П., проф.		

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Затвердження теми роботи	01.09.2019	
2	Вивчення та аналіз завдання	01.01.2019	
3	Аналіз існуючих передумов для розробки	28.02.2020	
4	Проектування архітектури системи	20.03.2020	
5	Розробка додатку	25.04.2020	
6	Оформлення матеріалів роботи	22.05.2020	
7	Передзахист	26.05.2020	
8	Захист		

Студент Микола ГУК

Керівник Олег АЛЄНІН

* Якщо визначені консультанти. Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

У даній бакалаврській роботі були досліджені способи кооперація власників транспортних засобів і пасажирів для організації спільних поїздок. Були розглянуті засоби комунікації, що сприяють кооперації пасажирів і водіїв. В результаті був створений додаток на основі ОС Android для кооперації і комунікації водіїв і пасажирів.

Створена програма надає можливість користувачам розміщувати свої маршрути, шукати спільні з іншими користувачами маршрути і комунікувати.

ABSTRACT

In this bachelor's thesis, ways to unite divers and passangers for performing joint trips. The means of communication that promote cooperation between passengers and drivers were considered. As a result, an Android-based software application was created to unite users for joit trips and provide way to communicate.

The created program allows users to post their routes, search for routes shared with other users and communicate.

№ рядка	Формат	Позначення	Найменування	Кільк. аркушів	Примітка
1			Документація		
2					
3	А4		Завдання	2	
4			на дипломний проект		
5					
6	А4	ІАЛЦ.467800.001 ВП	Відомість	1	
7			дипломного проекту		
8					
9	А4	ІАЛЦ. 467800.002 ТЗ	Технічне завдання	3	
10					
11	А4	ІАЛЦ.467800.003 ПЗ	Пояснювальна записка	63	
12					
13	А4	ІАЛЦ.467800.004 Д1	Додаток 1. Алгоритм	1	
14			пошуку маршрутів		
15					
16	А4	ІАЛЦ.467800.005 Д2	Додаток 2. Діаграма	1	
17			бази даних		
18					
19	А4	ІАЛЦ.467800.006 Д3	Додаток 3. Структурна	1	
20			схема системи		

					ІАЛЦ.467800.001 ВП
Зм	Аркуш	№ докум	Підпис	Дата	
Розробив		Гук М.О.			Мобільний додаток для управління системою спільних поїздок Відомість дипломного проекту
Перевірив		Алєнін О.І.			
Т. контр.					
Н. контр.		Сімоненко В.П.			
Затверд.					

Літ.	Аркуш	Аркушів
	1	1

НТУУ «КПІ ім. Ігоря Сікорського» ФІОТ
Група ІП-62

ТЕХНІЧНЕ ЗАВДАННЯ

до дипломного проєкту
освітньо-кваліфікаційного рівня бакалавр

на тему: “Мобільний додаток для управління системою спільних поїздок”

Київ – 2020 року

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Найменування: «Мобільний додаток для управління системою спільних поїздок».

Область застосування: мобільний додаток може використовуватися в повсякденному житті.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання бакалаврської дипломної роботи, затверджене кафедрою обчислювальної техніки Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського".

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Метою розробки є кооперація людей для здійснення поїздок в приватному транспорті.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література, публікації в спеціалізованих виданнях та мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до програмного продукту, що розробляється

Додаток, що розробляється, повинен:

- надавати можливість користувачам реєструватися та входити в систему за допомогою акаунта Google;
- розміщувати маршрути користувачів;
- відображати маршрути користувачів;
- фільтрувати маршрути користувачів;
- надавати можливість редагування маршрутів;
- надавати можливість приховування маршрутів;
- надавати способи комунікації.

5.2. Вимоги до апаратного забезпечення мобільного пристрою

- оперативна пам'ять об’ємом не менше 1 Гб;

- вбудована пам'ять об’ємом не менше 4 Гб;
- діагональ екрана не менше 4 дюймів.

5.3. Вимоги до програмного забезпечення мобільного пристрою

- версія ОС Android не менше 5.0.

6. ЕТАПИ РОЗРОБКИ

Назва етапів виконання	Термін виконання
Затвердження теми роботи	01.09.2019-01.01.2020
Вивчення та аналіз завдання	01.01.2020-28.02.2020
Аналіз існуючих передумов для розробки	28.02.2020-20.03.2020
Проектування архітектури системи	20.03.2020-25.04.2020
Розробка додатку	25.04.2020-22.05.2020
Оформлення матеріалів роботи	22.05.2020-26.05.2020
Передзахист	26.05.2020
Захист	

Пояснювальна записка
до дипломного проєкту

на тему: «Мобільний додаток для управління системою спільних поїздок»

Київ – 2020

ЗМІСТ

ВСТУП

В повсякденному житті іноді виникає проблема зв’язана з транспортом, тобто поїздки на роботу, в навчальний заклад, тренажерний зал або будь-які інші регулярні поїздки можуть спричинити багато незручностей: натовпи в громадському транспорті, незручний графік руху або взагалі відсутній графік руху, незручний маршрут, який займає багато часу через об’їзди, велика кількість пересадок, необхідність проходити великі відстані пішки.

Ціна на утримання транспортного засобу, ціна на пальне створюють незручності для власників особистих транспортних засобів, хоч і вирішують проблему вказану вище. Кожен власник автомобіля щодня витрачає, відчутну суму на пальне і регулярно платить за обслуговування і частіше всього їздить в машині на 5 місць, один.

Ще однією проблемою є збільшення числа транспортних засобів на дорогах загального користування, що призводить погіршення трафіку, утворення заторів на дорогах, аварій і погіршення екологічної обстановки.

Більшість людей, що стикаються з цією проблемою володіють засобами мобільного зв’язку і створення мобільного додатку для кооперації людей і управління поїздками допоможе вирішити ці проблеми.

Кількість мобільних додатків з кожним роком збільшується, вони дозволяють в будь-який момент отримати найактуальнішу інформацію і візуалізувати її в який-завгодно спосіб, забезпечують миттєвий зв'язок при цьому вони доступні кожному. Накопичення регулярних маршрутів власників автомобільних засобів і осіб без особистих транспортних засобів допоможе знайти спільні маршрути і об’єднати користувачів додатку для спільних поїздок. Застосунок може дати швидкий і зручний доступ до даних, підібрати найбільш оптимальний маршрут і забезпечити зв’язок між учасниками поїздки. Саме тому написання мобільного додатку для

управління спільними поїздками на особистому транспорті це адекватна тема.

РОЗДІЛ 1
ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Концепт системи для управління спільними поїздками і кооперації для спільних поїздок

Точка маршруту – точка на географічній площині, що належить до певного маршруту і характеризується геокординатами – широтою і довготою.

Кінці маршруту – крайні точки маршруту.

Маршрут – це спосіб представлення траєкторії руху з чітко визначеними характеристиками – послідовністю точок маршруту і часом початку поїздки.

Спільний маршрут – маршрут, який частково або повністю співпадає (має однакові або близькі кінці маршруту) з маршрутом іншого користувача і час старту поїздки однаковий або мало відрізняється. Наявність спільного маршруту означає, що пасажири можуть скооперуватися для спільної поїздки. Щоб кооперуватися для спільних поїздок необхідно накопичити якомога більше маршрутів користувачів системи, відобразити всі спільні маршрути для користувача системи, надати спосіб звязку для користувачів системи, реагувати на зміни в маршрутах і надавати лише актуальну інформацію.

1.2 Особливості системи для управління спільними поїздками і кооперації для спільних поїздок

Особливості системи повинні вирішити задачі поставлені перед системою.

Система повинна ідентифікувати користувачів для подальшої роботи по накопичуванню їхніх маршрутів і пошуку спільних маршрутів.

Система повинна надавати можливість введення і редагування маршруту.

Система повинна забезпечити доступ користувача до повного списку актуальних маршрутів.

Система повинна фільтрувати маршрути і представляти користувачу лише спільні маршрути.

Система повинна давати можливість деактивувати маршрут, тобто зробити його невидим для інших користувачів, якщо поїздка не відбудеться у визначений час

Система повинна надавати засіб комунікації між користувачами для кооперації на спільну поїздку

1.3 Огляд аналогічних додатків

BlaBlaCar (рис. 1.1) – застосунок, що дозволяє шукати попутників для разових спільних поїздок з метою зеконмити на переїзді. Реалізований для платформ web, android та ios.

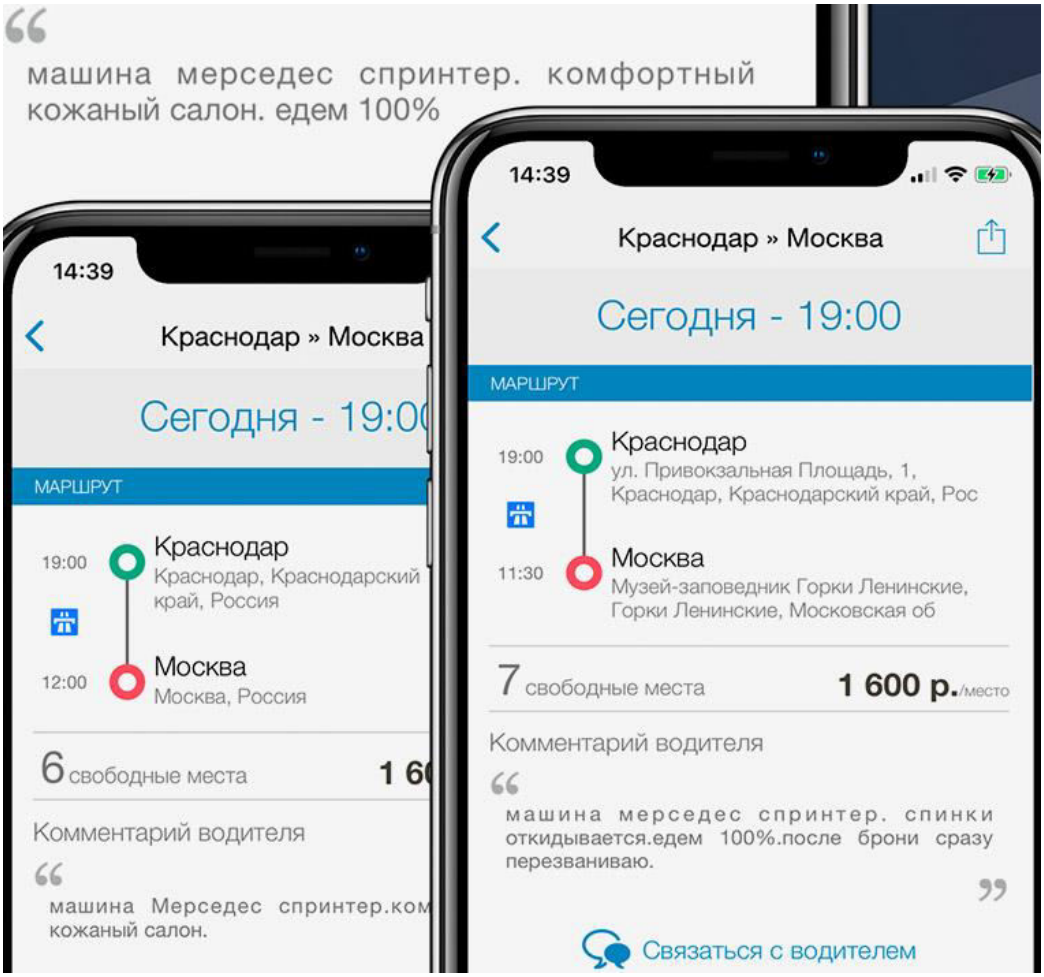


Рис. 1.1 Інтерфейс застосунку «BlaBlaCar»

Після реєстрації або авторизації, додаток дозволяє переглянути історію поїздок, знайти попутника, вказавши адресу точки відправлення, адресу точки прибуття, бажану дату поїздки і число пасажирів, є можливість ввести точку відправлення за допомогою геолокації вказавши своє поточне

місцезнаходження, як точку відправлення, також є можливість вказати своє місцезнаходження безпосередньо поставивши маркер на елекронній карті або ввести адресу свого місцезнаходження, для цього способу працює автодоповнення, яке зекономить час на введення адреси, підказуючи потрібну. Після введення всіх необхідних даних система видає список всіх поїздок і коротку інформацію про них, тут же до будь-якої поїздки можна долучитися, поїздки можна відфільтрувати. Також можна подивитися детальну інформацію про конкретну поїздку: ціна за участь в поїздці, час відправлення – час прибуття, ім'я водія, модель автомобіля, кількість вільних і забронюваних місць, рейтинг водія, список обмежень в автомобілі.

Щоб виступити в якості водія, потрібно ввести пункт відправлення, пункт прибуття, проміжні пункти на маршруті, дату і час поїздки кількість вільних місць, обмеження в автомобілі, ціну забронювання місця, ймовірність поїздки в зворотню сторону та іншу, додаткову інформацію текстово. Водій має можливість самостіймо підтверджувати бронювання місця пасажиром або дозволити робити це системі. Після чого поїздка буде опублікована і доступна для інших користувачів системи. В додатку реалізована система push-сповіщень, яка інформує користувача при виявленні нових маршрутів.

Для комунікації між учасниками поїздки в додатку наявний месенджер.

Основна особливість програми – пошук попутника для одноразової поїздки, за гроші. У нашому випадку даний застосунок не вирішує проблему регулярних поїздок. Також недоліком данного додатку є те, що він не підтримується на старих версіях операційної системи.

Greendrive (рис. 1.2) – мобільний застосунок, який допомагає шукати попутників для разових або регулярних поїздок. Реалізований для платформ android та ios.

Після реєстрації або авторизації, додаток дозволяє переглянути поїздки користувача: як завершені поїздки, так і майбутні.

Можна запропонувати поїздку, для цього потрібно ввести адресу точки відправлення і адресу точки прибуття, є можливість ввести точку відправлення за допомогою геолокації вказавши своє поточне місцезнаходження, як точку відправлення, наявна функція автодоповнення, яка зекономить час на введення адреси, підказуючи потрібну, час прибуття або час відправлення, вказати чи поїздка повторюється регулярно і періодичність поїздки, кількість вільних місць і ціну бронювання. Після створення поїздки вона стане доступною для приєднання других користувачів, також після створення показується preview заявки і список користувачів, які почали шукати даний маршрут ще до того як він був створений. В додатку реалізована система push-сповіщень, яка інформує користувача при виявленні нових маршрутів. Також можна запросити поїздку, для цього потрібно вказати адресу пункту відправлення і адресу пункту назначення, час відправлення або час прибуття, вказати чи поїздка повинна відбуватися на регулярній основі і періодичність поїздки і дату останнього прибуття. Після створення запиту на поїздку, вона почне показуватись для водіїв які пропонують таку поїздку, також відобразиться preview заявки і список водіїв, які пропонують таку поїздку.

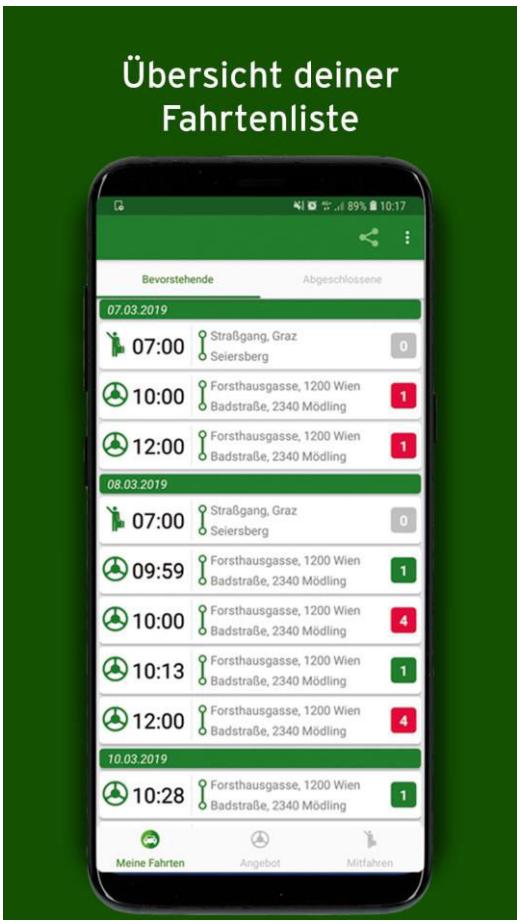


Рис. 1.2 Інтерфейс застосунку «Greendrive»

Отже мета додатку – спільні одноразові або регулярні поїздки з метою економії.

Недоліками даного додатку є відсутність локалізації для більшості мов, зокрема української, неможливість вказати точку маршруту на карті, незначне ком'юніті на території України, що призводить до неможливості знайдення необхідного маршруту і ускладнений інтерфейс користувача. Цей додаток не вирішує основну проблему – спільні поїздки.

Попутчик (Рис 1.3) – мобільний застосунок який допомагає шукати попутників для одноразових і регулярних поїздок, спільних подорожей і спільного відвідування різних заходів. Реалізований для платформ android та ios.

Рис. 1.3 Інтерфейс застосунку "Попутчик"

Після реєстрації або авторизації , які є необов'язковими, можна зайти в якості гостя, мобільний додаток переводить користувача в флоу пошуку водія, тобто пошуку маршруту, в якості пасажира і надає два способи це зробити – вказати маршрут на карті (пункт відправлення і пункт призначення) на карті є текстовий помічник з можливістю голосового вводу, який дає змогу швидше знайти необхідну точку на карті, поточне місцезнаходження користувача можна вказати однією із точок маршруту за допомогою геолокації, можна змінити напрямок маршруту або за допомогою текстового пошуку вказавши у відповідних полях адреси пункту відправлення і пункту призначення, поточне місцезнаходження користувача можна вказати однією із точок маршруту за допомогою геолокації, можна змінити напрямок маршруту. Для полів адреси доступна функція автодоповнення, яка зекономить час на введення адреси. Необхідно уточнити час відправлення указавши чи це разова поїздка чи щотижнева і точну дату або день тижня відповідно. Після заповнення форми система сповістить всіх водіїв з підходящим маршрутом про появу нового потенційного попутника і відобразить список всіх водіїв з потрібними поїздками. Також можна

подивитися детальну інформацію про конкретну поїздку: ціна за участь в поїздці, час відправлення, час в дорозі, ім’я водія, модель автомобіля, кількість вільних і забронюваних місць, ціну поїздки. На цьому ж етапі можна забронювати місце.

Для водіїв в системі передбачений широкий спектр дій з маршрутами. Для додавання нового маршруту користувач має бути авторизованим. Є два способи створення нового маршруту - вказати маршрут на карті (пункт відправлення і пункт призначення) на карті є текстовий помічник з можливістю голосового вводу, який дає змогу швидше знайти необхідну точку на карті, поточне місцезнаходження користувача можна вказати однією із точок маршруту за допомогою геолокації, можна змінити напрямок маршруту або за допомогою текстового пошуку вказавши у відповідних полях адреси пункту відправлення і пункту призначення, поточне місцезнаходження користувача можна вказати однією із точок маршруту за допомогою геолокації, можна змінити напрямок маршруту. Для полів адреси доступна функція автодоповнення, яка зекономить час на введення адреси. Необхідно уточнити час відправлення указавши чи це разова поїздка чи щотижнева і точну дату або день тижня відповідно. Після заповнення форм, система сповістить всіх пасажирів про появу нового потенційного водія-попутника. За наявності маршрутів, їх перелік буде відображений в історії маршрутів, щойно додані маршрути класифікуються системою, як нові і будуть відображені ще й окремо.

В застосунку наявний вбудований месенджер для комунікації між попутниками.

Push-сповіщення налаштовуються, вони можуть бути тимчасово або повністю вимкнені, вимкнені для певних осіб – чорний список, наявна історія сповіщень.

Цей додаток має низку недоліків: збої в програмі на різних етапах роботи – авторизації, створення або знайдення маршруту, відображення

карти, що перешкоджає його використанню за призначенням, додаток має громіздкий занадто ускладнений інтерфейс і надлишок функцій, ще одним недоліком є незначне ком'юніті, що призводить до неможливості знайдення необхідного маршруту.

Uber (рис. 1.4) – система таксі. Реалізована для платформ android та ios.

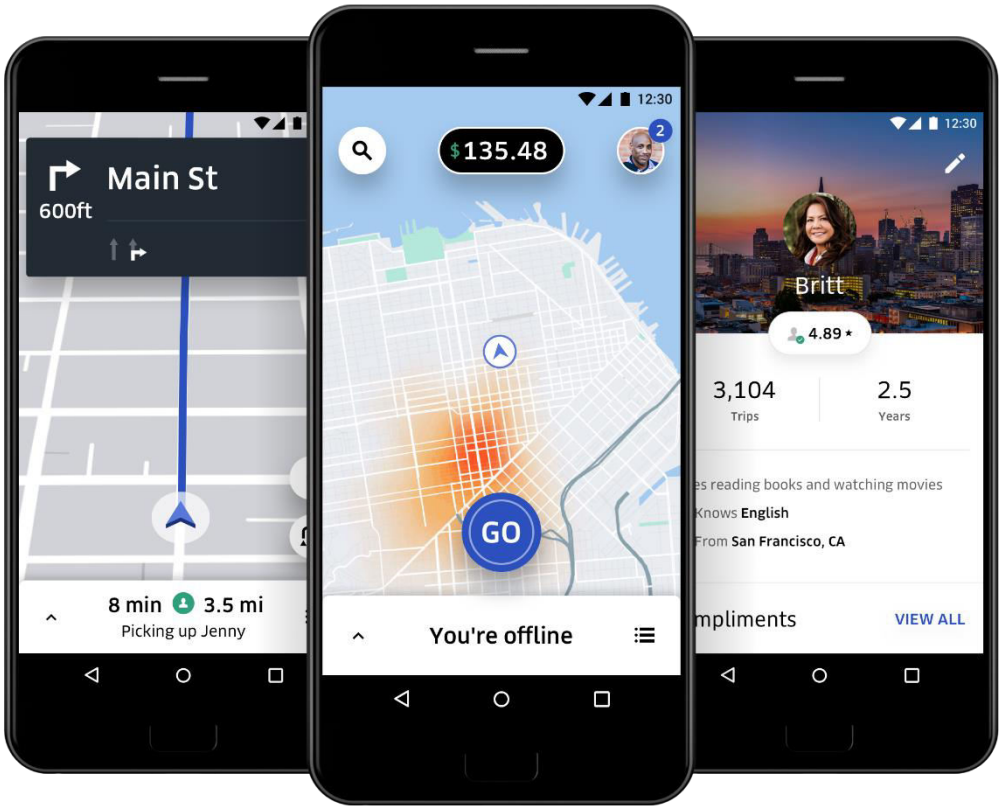


Рис. 1.4 Інтерфейс застосунку "Uber"

Для роботи із застосунком необхідна реєстрація / авторизація. Система має простий та інтуїтивно-зрозумілий користувацький інтерфейс. Користувач має можливість викликати таксі, вказавши точку відправлення: текстово вписавши адресу, наявне автодоповнення, яке зекономить час на введення адреси, підказуючи потрібну, відмітивши точку на карті або вибравши свое поточне місцезнаходження, як початок маршруту за допомогою геолокації і точку призначення текстово вписавши адресу або відмітивши точку на карті. Наявні функції geocoding – знаходження адреси за допомогою геокоординат і

reverse geocoding – знаходження геокоординат за введеною адресою. Застосунок знаходить найближче вільне таксі і сповіщає водія. По прибуттю екіпажу користувачу також приходить сповіщення, переміщення машини постійно відображається на карті. Вбудована система оплати поїздок. Є функція історії для перегляду попередніх поїздок. В налаштуваннях профілю є можливсть вказати місце роботи і місце проживання і подальшого застосування цих даних для побудови маршрутів.

Основним недоліком даної системи є висока ціна послуг. Використання даної системи також неможливе в малонаселених регіонах, немає локалізацій крім англійської, тому вона не вирішує поставлених задач.

Uklon (рис. 1.5) – система таксі. Реалізована для платформ android та ios.

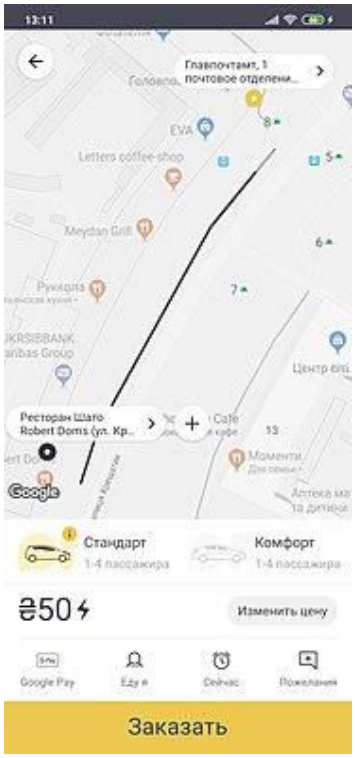


Рис. 1.5 Інтерфейс застосунку "Uklon "

Для роботи із застосунком необхідна реєстрація / авторизація. Користувач має можливість викликати таксі, вказавши точку відправлення: текстово вписавши адресу, наявне автодоповнення, яке зекономить час на введення адреси, підказуючи потрібну, відмітивши точку на карті або

вибравши свое поточне місцезнаходження, як початок маршруту за допомогою геолокації і точку призначення текстово вписавши адресу або відмітивши точку на карті. Наявні функції geocoding – знаходження адреси за допомогою геокоординат і reverse geocoding – знаходження геокоординат за введеною адресою. Застосунок знаходить найближче вільне таксі і сповіщає водія. По прибуттю екіпажу користувачу також приходить сповіщення, переміщення машини постійно відображається на карті. Вбудована система оплати поїздок. Є функція історії для перегляду попередніх поїздок.

Основним недоліком даної системи є висока ціна послуг. Використання даної системи також неможливе в малонаселених регіонах, тому вона не вирішує поставленої задачі.

GoogleMaps (рис. 1.6) – система елекронних карт з можливістю прокладання маршрутів. Реалізована на платформах web, android та ios.

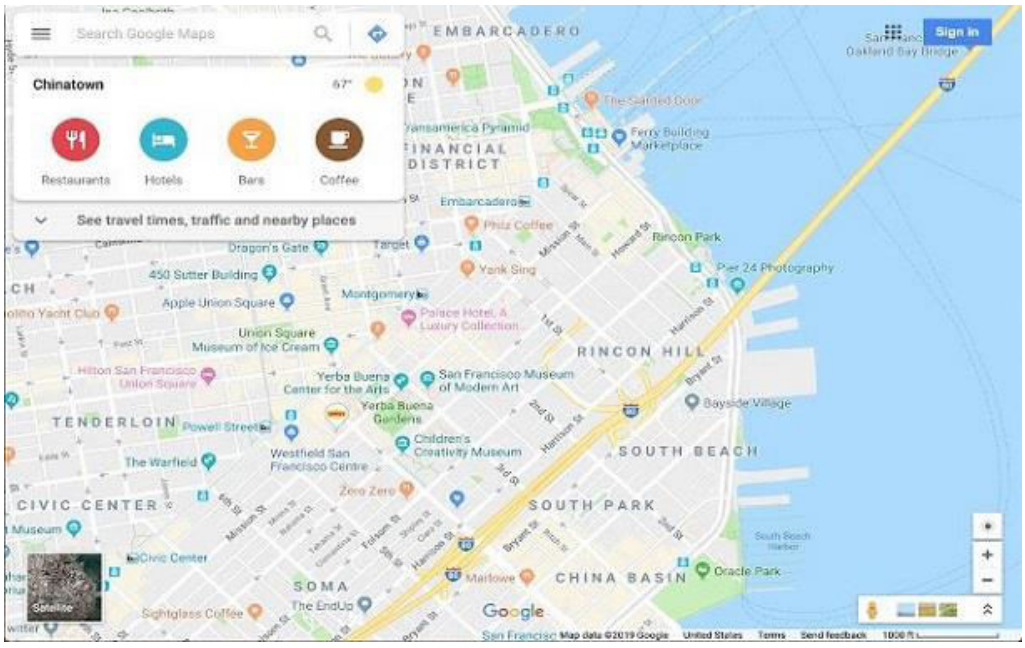


Рис. 1.6 Інтерфейс застосунку "Google Maps"

Застосунок виконує авторизацію автоматично використовуючи дані аккаунтів збережених на пристрої. Додаток дає можливість переглядати карти, знаходити місця за введеною адресою і відображати їх на карті, навпаки за вибраною на карті точкою встановлювати адресу, будувати декілька альтернативних маршрутів для пішоходів, громадського транспорту

або для особистого транспорту за вказаними точками і оцінювати час на проходження маршруту. Є можливість шукати місця поблизу за вказаними тегами. В даної системи є API на основі якого побудовано багато програм.

Основним недоліком даного додатку є те, що він не поширює маршрути користувачів і не дає змогу кооперуватися з іншими користувачами для спільних поїздок, також немає функціоналу для управління поїздками, тому він не вирішує поставлені задачі.

SIXT: Car rental, Carsharing & Taxi (рис. 1.7) **–** система оренди автомобілей, каршерінгу і таксі. Реалізована на платформах android та ios.

Для роботи із застосунком необхідна реєстрація / авторизація. Система сворена для управління процесом здачі в оренду автомобілей і виклику таксі. Користувач має можливість викликати таксі, вказавши точку відправлення: текстово вписавши адресу, наявне автодоповнення, яке зекономить час на введення адреси, підказуючи потрібну, відмітивши точку на карті або вибравши свое поточне місцезнаходження, як початок маршруту за допомогою геолокації і точку призначення текстово вписавши адресу або відмітивши точку на карті. Після відправлення форми вона йде на опрацювання співробітниками системи. Оренда автомобіля являє собою місця розташування автомобілів на карті, з детальною інформацією про угоду: технічні характеристики автомобіля, тарифи оренди, запас пального. Щоб здійснити оренду, необхідно її спочатку оплатити, після чого в додатку генерується QR код, який є одноразовим ключем для відмикання автомобіля. Коли час оренди добігає кінця, система попереджає користувача за допомогою push – сповіщення, є можливість продовжити оренду виконавши оплату ще раз.

Перевагами системи є простий і зрозумілий інтерфейс, вона вирішує проблеми зв'язані з незручностями громадського транспорту. Недоліками системи є відсутність локалізацій крім англійської, мале число автомобілів які можна орендувати і найголовніше вона не сприяє кооперації користувачів

для спільних поїздок і не мая функціоналу для управління поїздками, тому дане рішення не підходить для поставлених задач.

Рис. 1.7 Інтерфейс застосунку "Sixt "

OsmAnd (рис. 1.8) – система для перегляду карт і навігатор. Реалізована на платформах android і ios.

Додаток призначений для перегляду карт, побудови маршрутів і навігації. Для введення точок маршруту використовується текстове введення адреси з автодоповненням, позначення точки на карті і геолокація. Наявні функції geocoding – знаходження адреси за допомогою геокоординат і reverse

geocoding – знаходження геокоординат за введеною адресою. Підтримується режим роботи без підключення до глобальної мережі за допомогою сервісів GPS і локального збереження файлів карт. Система надає можливість будувати пішохідні, велосипедні і автомобільні маршрути, а також маршрути громадського транспорту, рахує довжину траєкторії маршруту та оцінює час проходження маршруту.

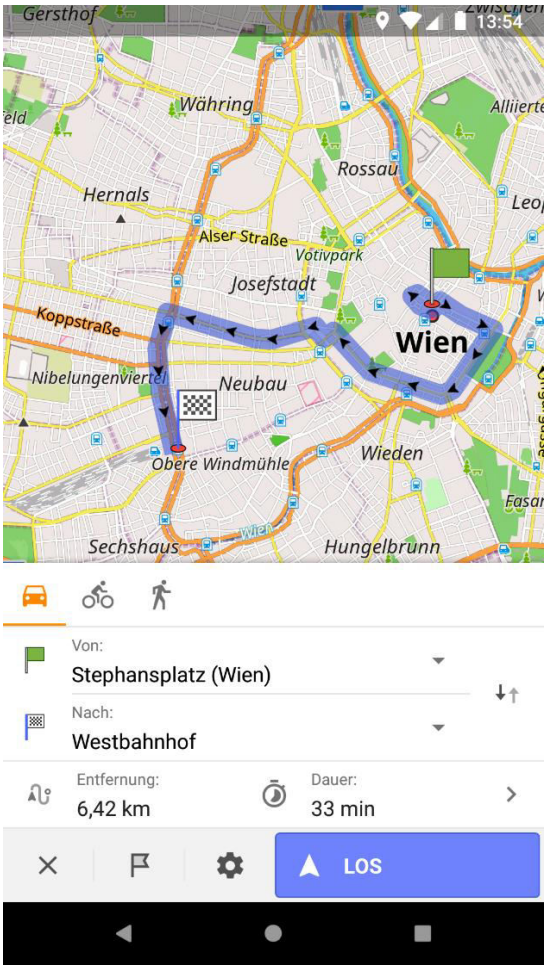


Рис. 1.8 Інтерфейс додатку "OsmAnd"

Додаток не має функціональності для забезпечення кооперації між користувачами і управління їхніми спільними поїздками, тому він не є рішенням поставлених задач.

1.4 Порівняння оглянутих застосунків

Таблиця 1.1 Порівняння додатків

Назва додатку	**Створення, редагування маршрутів**	**Відображення наявних маршрутів**	**Фільтрація маршрутів**	**Управління поїздками**	**Засоби комунікації**
BlaBlaCar	+	+	+	+	+
Greendrive	+	+	+	+	+
Попутчик	+	+	+	+	+
Uber	+	+	-	+	-
Uklon	+	+	-	+	-
GoogleMap	+	+	-	-	-
Sixt	+	-	-	+	-
OsmAnd	+	+	-	-	-

Таблиця 1.2 Порівняння додатків

Назва додатку	**Geocoding reverse geogcoding**	**Платформа**	**Автозаповнення**	**Сповіщення**
BlaBlaCar	+	web, android, ios	+	+
Greendrive	+	android, ios	+	+
Попутчик	+	android, ios	+	+
Uber	+	android, ios	+	+
Uklon	+	android, ios	+	+
GoogleMap	+	web, android, ios	+	-
Sixt	-	android, ios	+	-
OsmAnd	+	android, ios	+	-

Висновки до розділу 1

В ході аналізу існуючих рішень для роботи з картами, геолокацією і побудовою маршрутів з’ясувалося, що:

1) Для забезпечення постійного доступу користувача до інформації про маршрути і поїздки, а також для створення можливості в будь який момент ініціювати пошук або створення маршруту необхідно реалізувати систему саме в вигляді мобільного додатку.
2) Кожен з переглянутих додатків вирішує частину поставлених задач, але жоден з них не є повним рішенням всіх проблем. Тому виникає необхідність створення самостійного мобільного додатку для ініціювання спільних поїздок і управління поїздками.
3) Для спрощення інтерфейсу додатку необхідно забезпечити якомога більше способів введення і редагування маршруту даючи можливість користувачу використати найбільш зручний спосіб.
4) Візуалізація геоданих на карті полегшує взаємодію користувача з програмою.
5) Реалізація функції автодоповнення адреси заощаджує час на введення тексту, ти самим робота з додатком стає більш динамічною, також вона запобігає виникненню орфографічних помилок при введенні тексту і збільшую вірогідність побудови правильного маршруту.

РОЗДІЛ 2
ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Варіанти використання системи

Функціонал системи передбачає лише дві роль – авторизований користувач і користувач який ще не авторизувався.

Неавторизований користувач має доступ до таких варіантів використання:

Авторизація

Авторизований користувач має доступ до таких варіантів використання:

- Перегляд повного списку маршрутів
- Перегляд деталей конкретного маршруту
- Перегляд власного профілю
- Пошук маршруту
- Комунікація із власником маршруту
- Створення маршруту
- Редагування деталей маршруту
- Деактивація маршруту
- Реактивація маршруту
- Видалення маршруту
- Вихід із системи

Варіанти використання зображені у вигляді діаграми на рис. 2.1

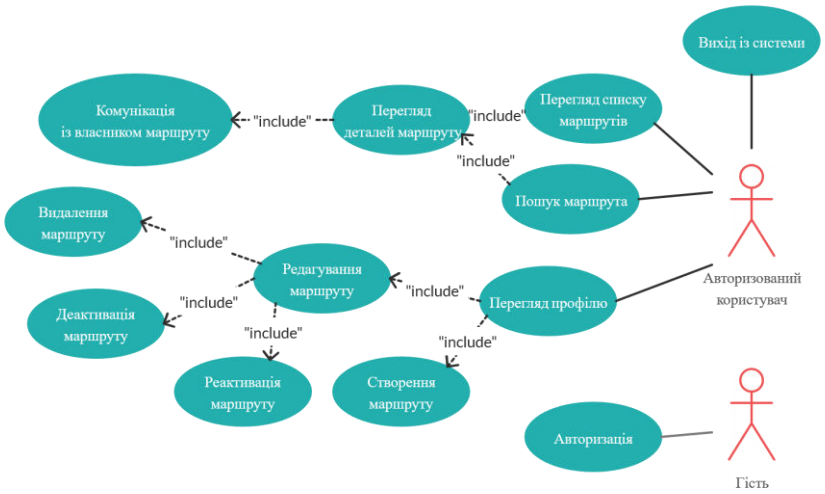


Рис. 2.1 Діаграма варіантів використання

Далі йде опис сценаріїв використання програмної системи для управління спільними поїздками

Сценарій авторизації в системі (рис. 2.2)

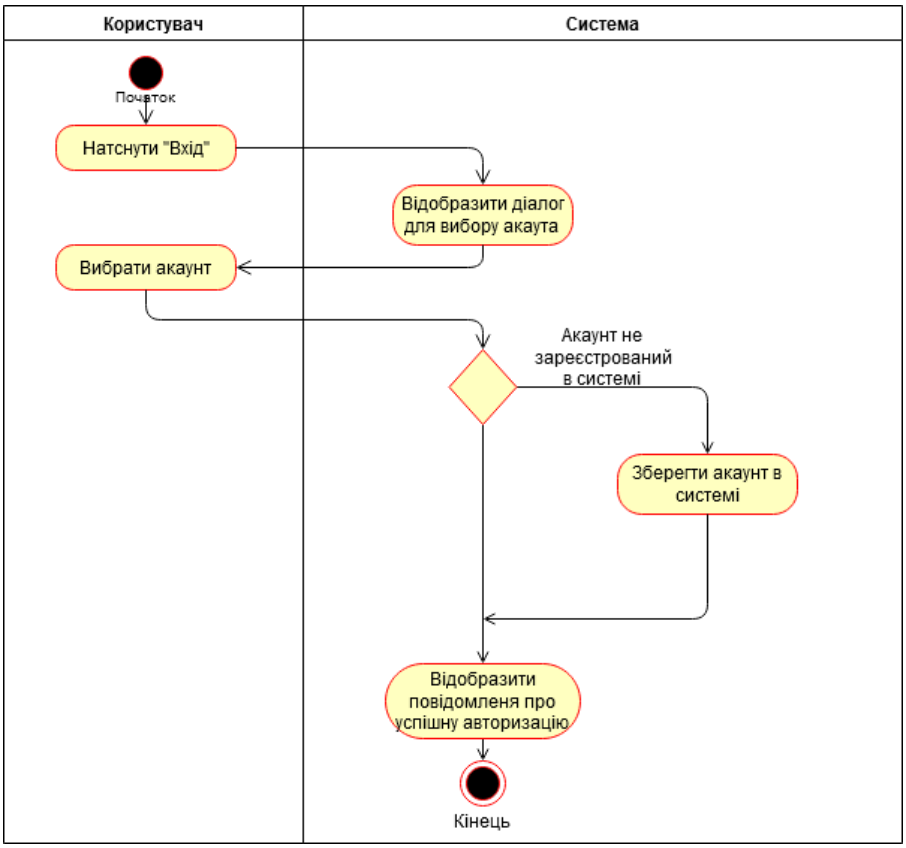


Рис. 2.2 Авторизація

Учасники: користувач і система

Передумови: в системі немає авторизованих користувачів

Сценарій:

1. Користувач натискає на кнопку «Вхід»
2. Система відображає діалог для вибору акаунта
3. Користувач вибирає акаунт
4. Система зберігає токен авторизації даного користувача

Результат: користувач отримує доступ до функціоналу системи, який доступний лише для авторизованих користувачів

Виключні ситуації: не вдалося доставити запит на авторизацію серверу

Сценарій перегляду всіх маршрутів (рис. 2.3)

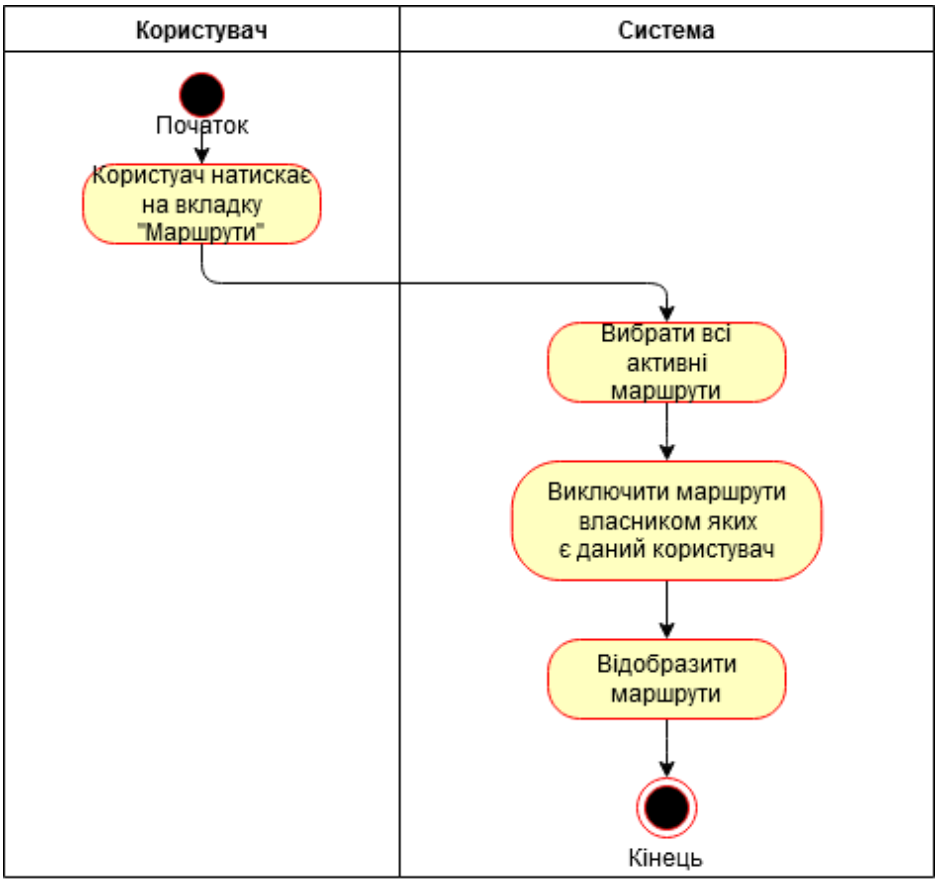


Рис. 2.3 Перегляд усіх маршрутів

Учасники: користувач і система

Передумови: авторизований користувач

Сценарій:

1. Користувач натискає на вкладку «Маршрути»

2. Система вибирає всі активні маршрути для яких даний користувач не є власником.
3. Система відображає знайдені маршрути

Результат: користувач отримав інформацію про всі наявні маршрути

Виключні ситуації: не вдалося встановити з'єднання з сервером для отримання списку маршрутів

Сценарій перегляду профілю (рис. 2.4)

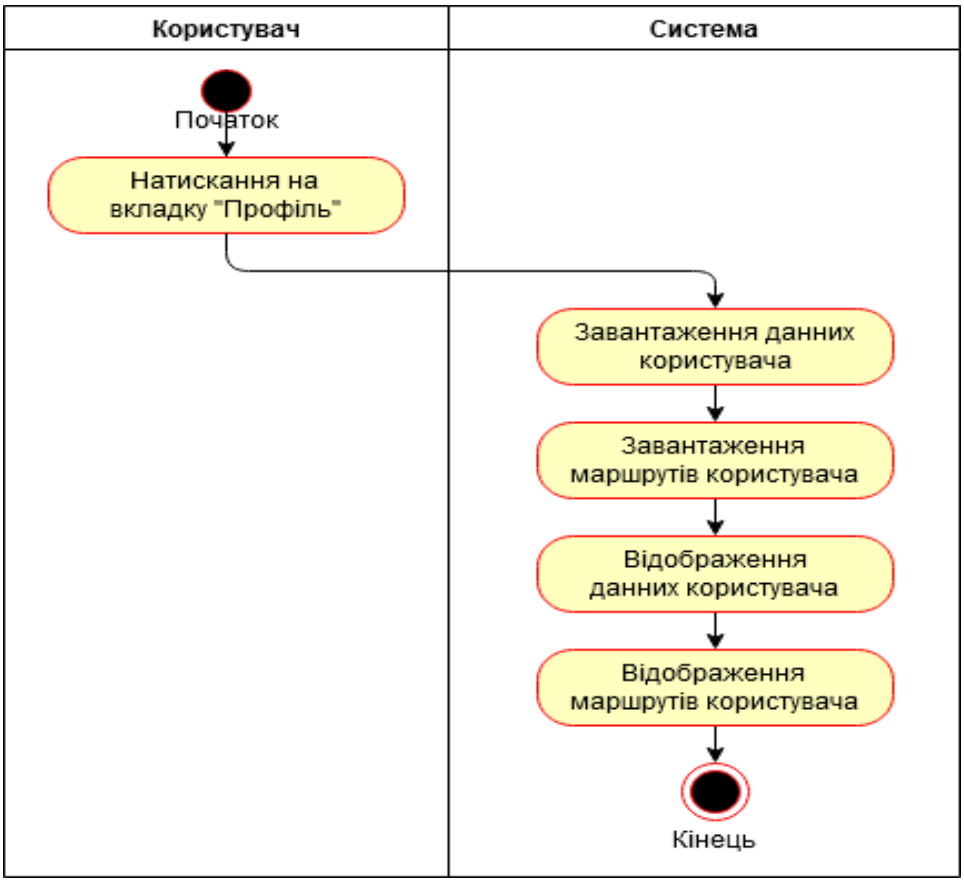


Рис. 2.4 Перегляд профілю

Учасники: користувач і система

Передумови: авторизований користувач

Сценарій:

1. Користувач натискає на вкладку «Профіль»
2. Система завантажує дані користувача і маршрути користувача
3. Система відображає дані користувача і маршрути користувача

Результат: користувач отримав інформацію свого профілю і свої маршрути

Виключні ситуації: не вдалося встановити з’єднання з сервером для отримання даних профілю і маршрутів

Сценарій пошуку маршруту (рис. 2.5)

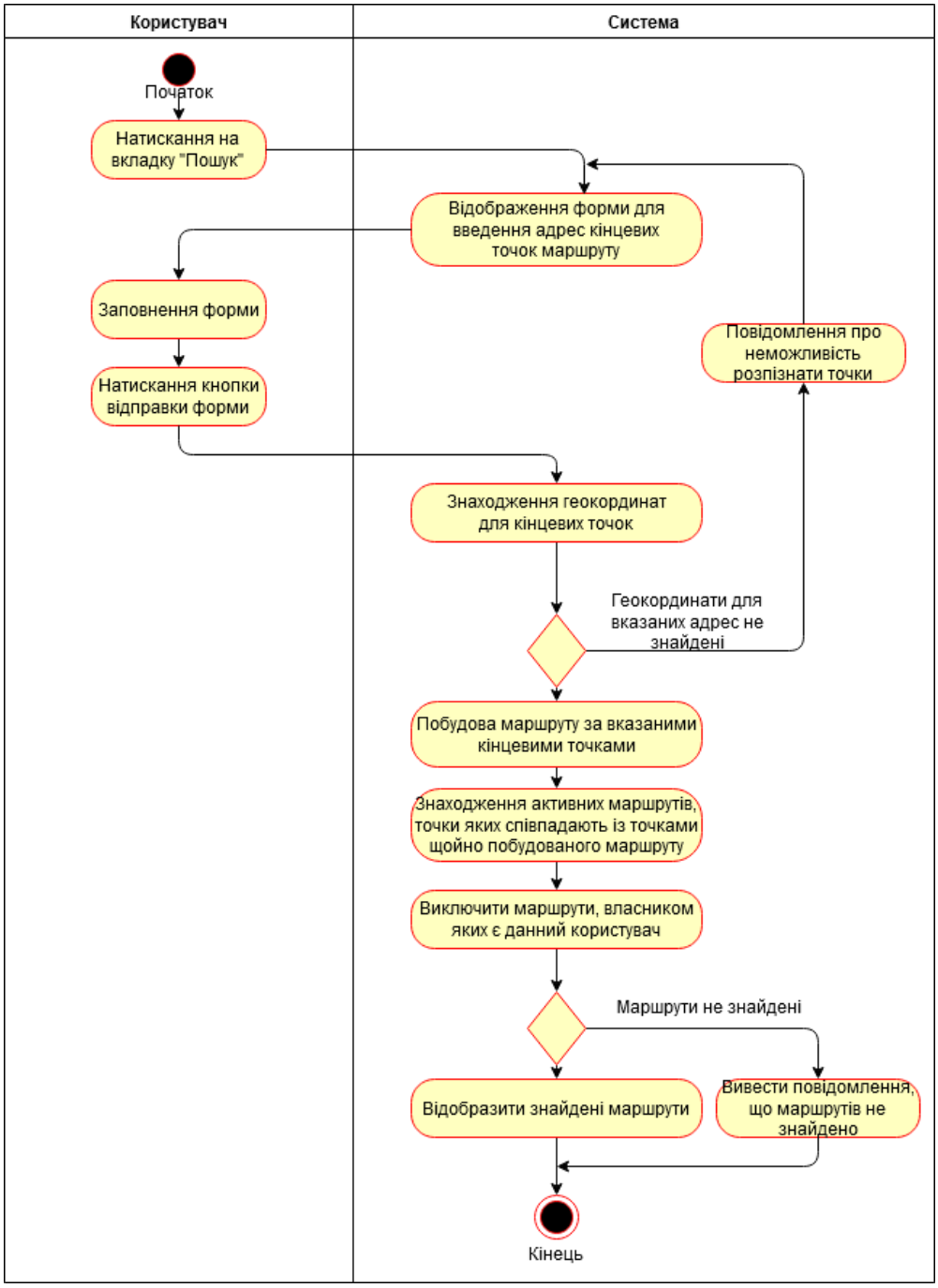


Рис. 2.5 Пошук маршруту

Учасники: користувач і система

Передумови: авторизований користувач

Сценарій:

1. Користувач натискає на вкладку пошук
2. Система відображає форму для введення адрес кінцевих точок маршруту

3. Користувач заповнює форму
4. Користувач натискає кнопку відправки форми
5. Система знаходить геокординати для даних кінцевих точок маршрута
6. Система будує маршрут за геокординатами кінцевих точок
7. Система шукає активні маршрути, точки яких частково або повністю співпадають з точками щойно побудованого маршруту і для яких данний користувач не є власником.
8. Система відображає інформацію про знайдені маршрути

Результат: користувач отримує інформацію про шукані маршрути.

Виключні ситуації: не вдалося встановити з'єднання з сервером для отримання маршрутів

Сценарій комунікації з власником маршруту (рис. 2.6)

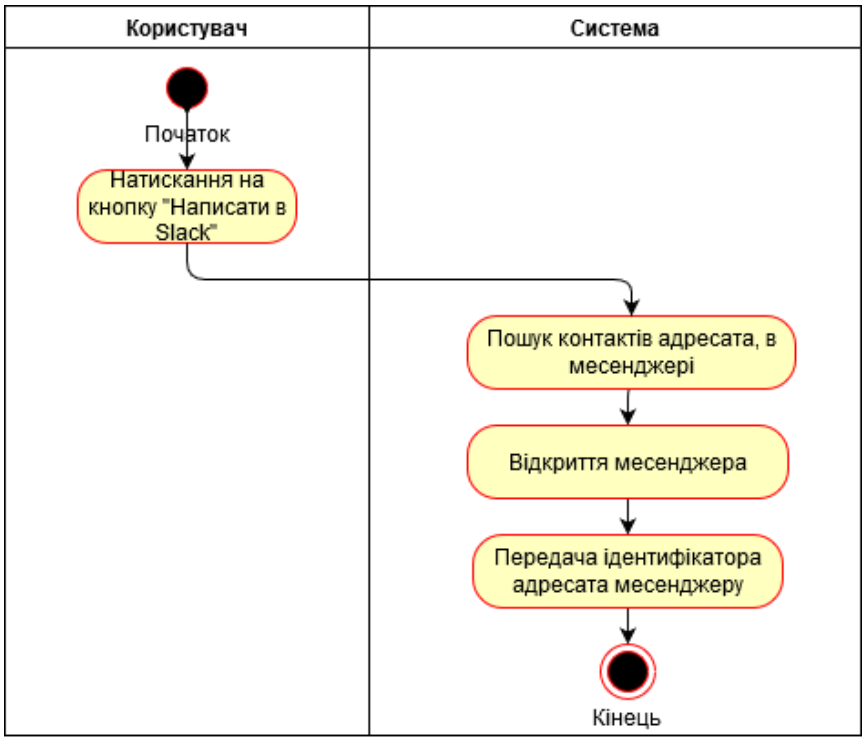


Рис. 2.6 Комунікація з власником маршруту

Учасники: користувач і система

Передумови: авторизований користувач, користувач перейшов до деталей маршруту

Сценарій:

1. Користувач натискає кнопку «Написати в Slack»
2. Система знаходить ідентифікатор власника маршруту в месенджері
3. Система відкриває додаток месенджера
4. Система передає ідентифікатор власника маршруту в месенджер

Результат: користувач перейшов до діалогу з власником маршруту

Виключні ситуації: для власника маршруту не знайдено ідентифікатора в месенджері

Сценарій редагування маршруту (рис. 2.7)

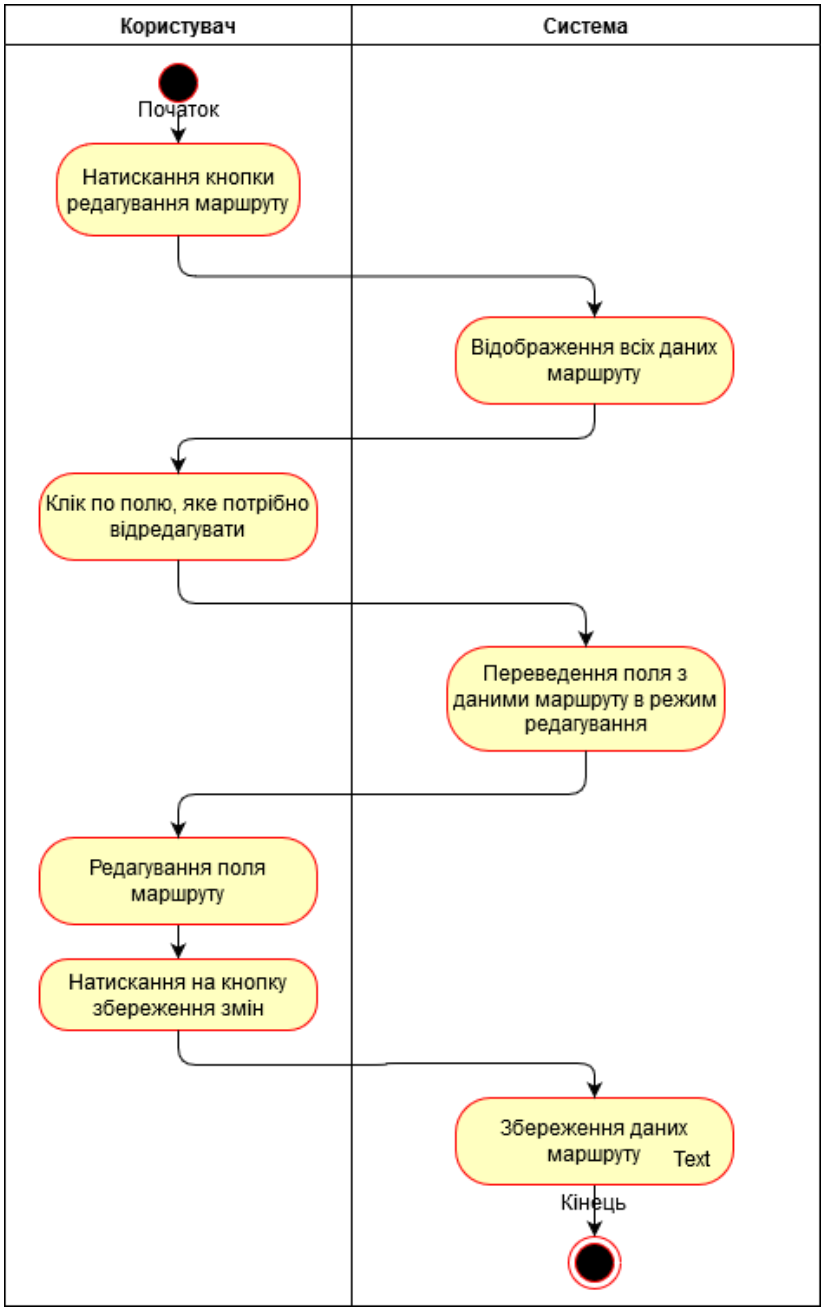


Рис. 2.7 Редагування маршруту

Передумови: авторизований користувач, користувач є власником маршруту

Сценарій:

1. Користувач натискає кнопку редагування маршруту
2. Система відображає всі дані маршруту
3. Користувач клікає по полю, яке треба відредагувати
4. Система переводить поле в режим редагування

5. Користувач редагує дані маршруту
6. Користувач натискає на кнопку збереження змін
7. Система зберігає дані маршруту

Результат: користувач відредагував дані маршруту

Виключні ситуації: не вдалося встановити з'єднання з сервером для збереження маршруту

Сценарій деактивації маршруту (рис. 2.8)

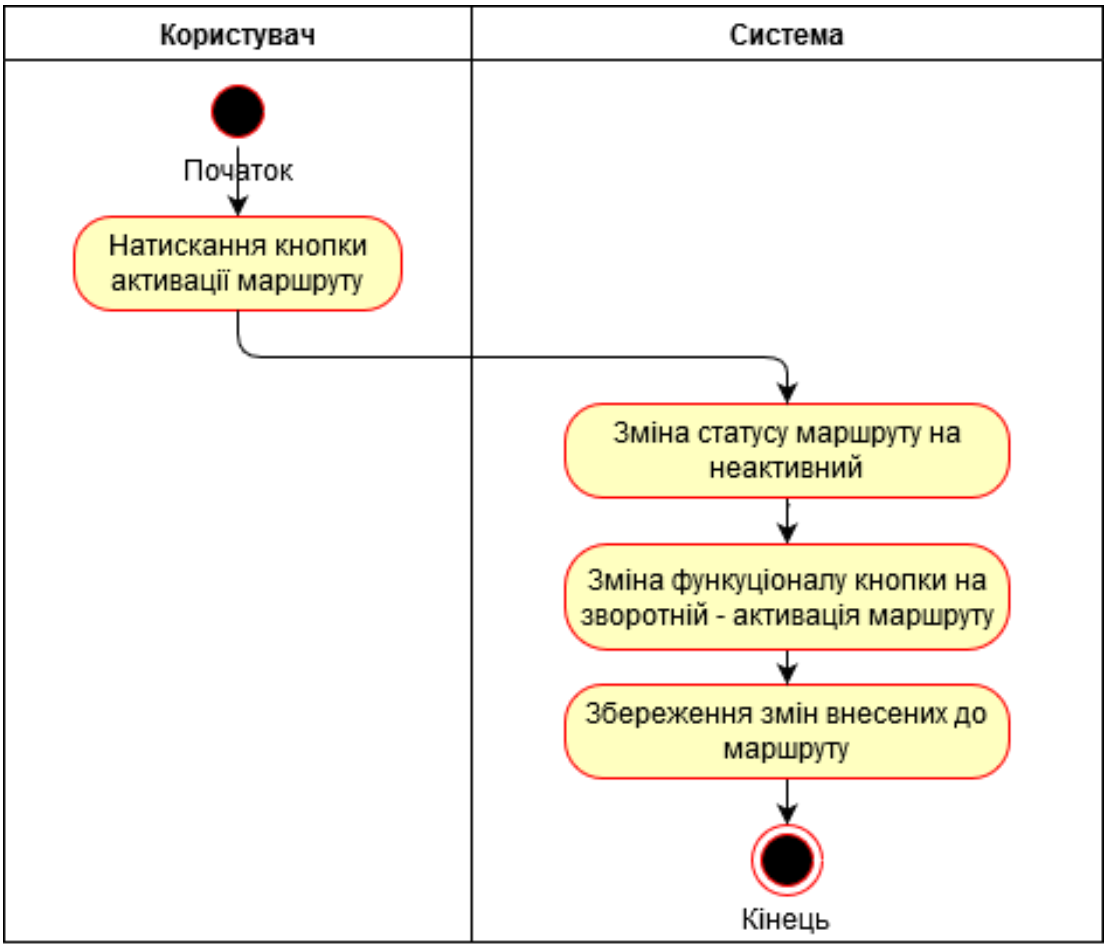


Рис. 2.8 Деактивація маршруту

Передумови: авторизований користувач, користувач є власником маршруту, маршрут активний

Сценарій:

1. Користувач натискає на кнопку активації маршруту
2. Система змінює статус маршруту на неактивний

3. Система змінює функціонал кнопки на зворотній – активація маршруту
4. Система зберігає зміни внесені до маршруту

Результат: статус маршруту змінено на неактивний, він не відображається для інших користувачів системи

Виключні ситуації: не вдалося встановити з'єднання з сервером для збереження змін внесених до маршруту

Сценарій активації маршруту (рис. 2.9)

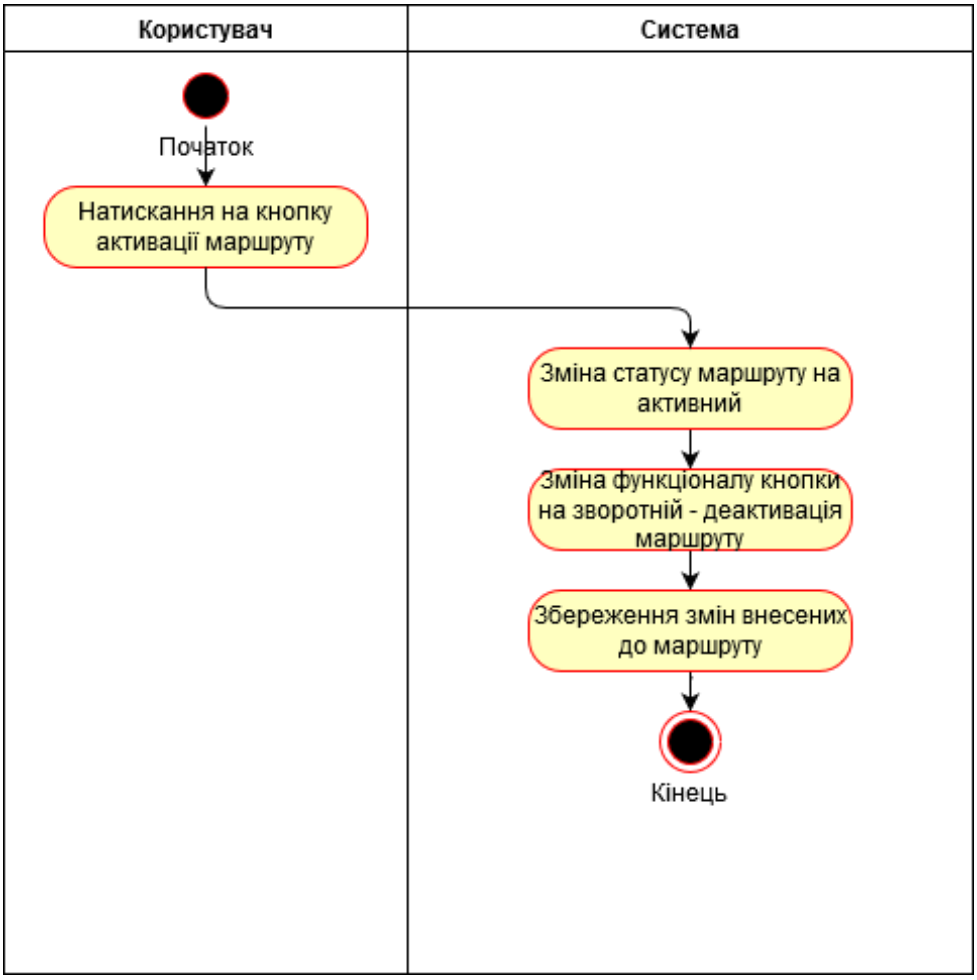


Рис. 2.9 Активація маршруту

Передумови: авторизований користувач, користувач є власником маршруту, маршрут деактивований

Сценарій:

1. Користувач натискає на кнопку активації маршруту
2. Система змінює статус маршруту на активний

3. Система змінює функціонал кнопки на зворотній – деактивація маршруту
4. Система зберігає зміни внесені до маршруту

Результат: статус маршруту змінено на активний, він відображається для інших користувачів

Виключні ситуації: не вдалося встановити з'єднання з сервером для збереження змін внесених до маршруту

Сценарій видалення маршруту (рис. 2.10)

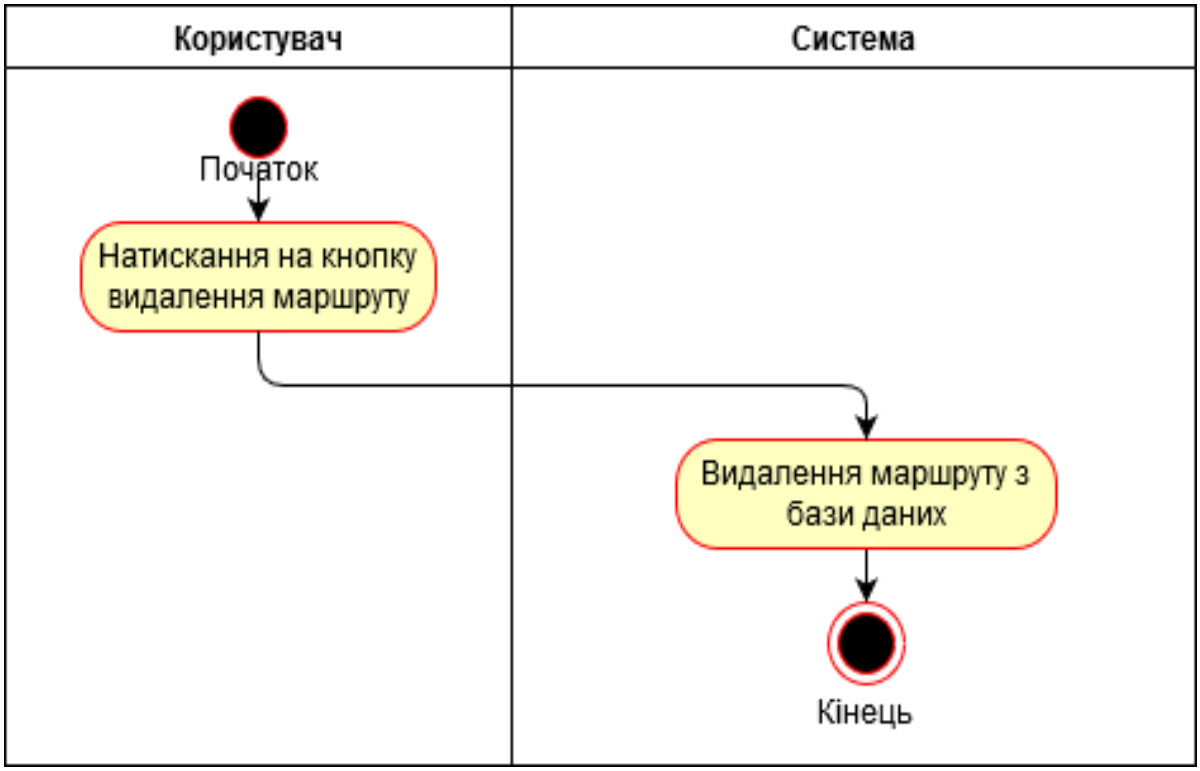


Рис. 2.10 Сценарій видалення маршруту

Передумови: авторизований користувач, користувач є власником маршруту

Сценарій:

1. Користувач натискає на кнопку видалення маршруту
2. Система видаляє маршрут

Результат: маршрут видалено, він не відображається для користувачів

Виключні ситуації: не вдалося встановити з'єднання з сервером для видалення маршруту

Сценарій виходу із системи (рис. 2.11)

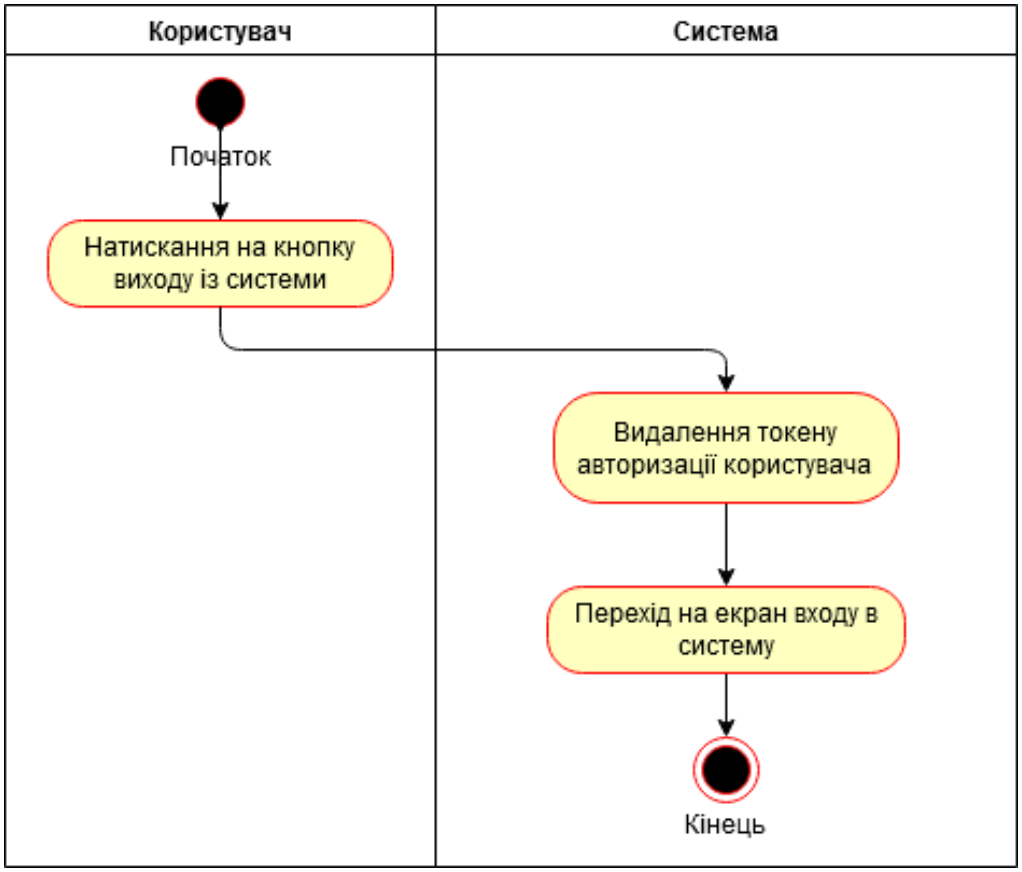


Рис. 2.11 Сценарій виходу із системи

Передумови: авторизований користувач

Сценарій:

1. Користувач натискає кнопку виходу з системи
2. Система видаляє токен авторизації користувача
3. Система відкриває екран авторизації

Результат: користувач не авторизований

Виключні ситуації: немає

2.2 Нефункціональні вимоги

Для додатків, які використовуються на постійній основі, одним із яких є мобільні додатки нефункціональні вимоги є дуже важливими. До функціональних вимог належать узгоджені стандарти, маштабування, надійність, безпека та конфіденційність, обробка помилок, продуктивність, зручність використання, збереження даних.

Нефункціональні вимоги впливають на розробку системи в цілому і накладають багато обмежень при розробці. Нефункціональні вимоги для системи управління спільними поїздками транспорті:

1. Операції активації/деактиації мршруту, редагування, створення доступні тільки для власника маршруту
2. Плавна робота при скролі списків маршрутів, швидкий підбір адрес при геокодінгу, відсутність лагів при натисканні кнопок, взаємодії з полями для вводу тексту і валідації, плавний перехід між екранами в додатку.
3. Швидкий відклик сервера, особливо при пошуку спільних маршрутів – до 2 сек.
4. Економія заряду батареї
5. Інтуїтивно-зрозумілий інтерфейс користувача андроїд.
6. Актуальність геокодингу, реверс геокодингу і автодоповнення адрес для регіону користувача.

Для досягнення задоволення цих вимог були прийняті такі рішення під час розробки.

Додавання реєстрації та авторизації користувача. Авторизація була реалізована за допомогою сервісу Google SignIn (рис. 2.10). Він побудований на основі стандарту авторизації OAuth2 [1].

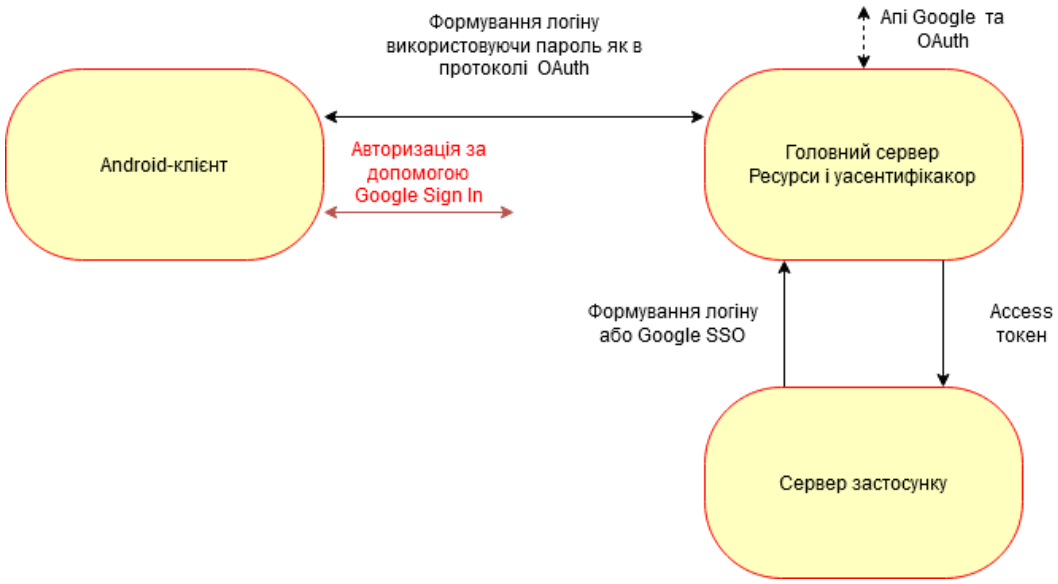


Рис. 2.12 Google SignIn

Сервіс Google Sign In на основі вибраного акаунта Google генерує access-токен, після чого повертає токен клієнту додатка, після чого клієнт за допомогою методу POST протоколу HTTPS пересилає токен серверу додатка, де проводиться валідація токена. При успішній валідації клієнт отримує успішну відповідь на POST запит, код 200, токен користувача зберігається на стороні сервера і на стороні клієнта і в подальшому використовується при комунікації клієнта і сервера. В разі неуспішної валідації клієнт отримує код помилки, як відповідь від сервера на свій запит POST. В даному випадку код – 401 Unauthorised, access-токен не зберігається, користувач залишається неавторизованим і подальші запити до сервера не будуть успішні, поки користувач залишається неавторизованим.

Для пришвидшення роботи користувацького інтерфейсу були використані view-елементи андроїд sdk, а також компонент навігації андроїд. Елемент RecyclerView (рис. 2.13) дозволяє ефективно відображати великі колекції колекції елементів, які часто змінюються. Стандартні реалізації

дозволяють

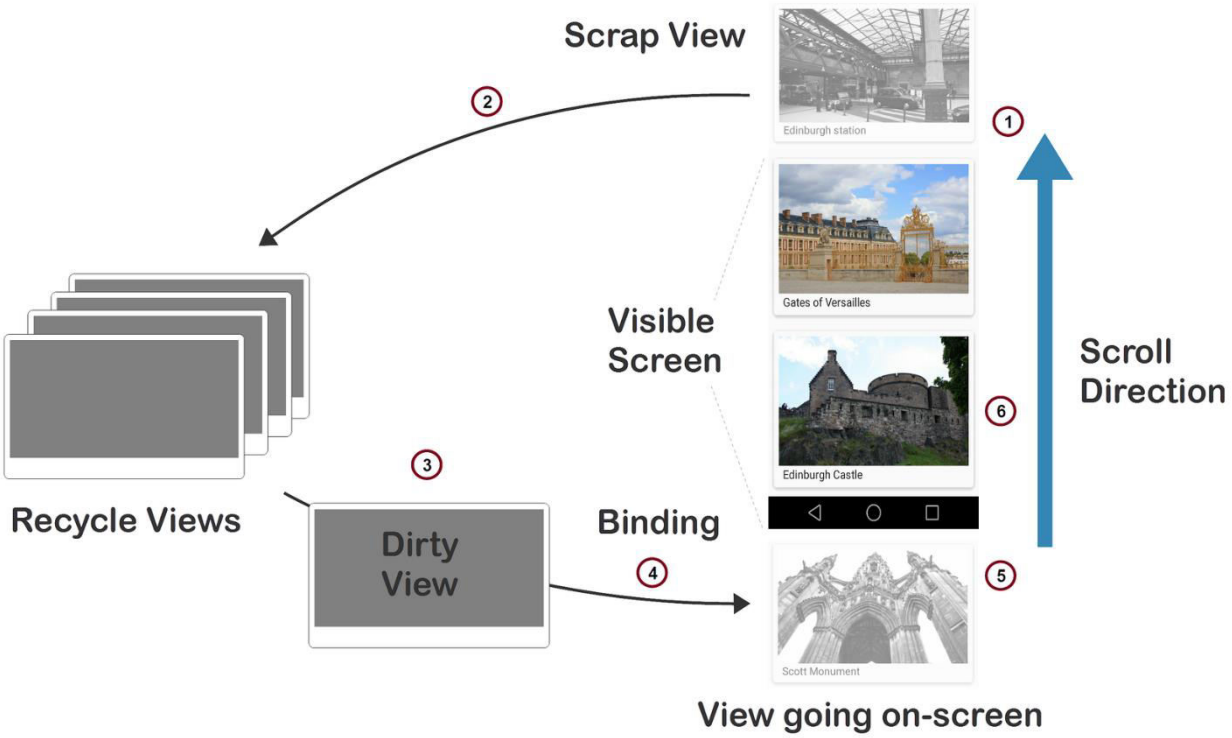


Рис. 2.13 RecyclerView

відображати дані у вигляді списків або таблиць. Дані елементів списку зберігаються в об'єктах ViewHolder, кожному елементу даних відповідає один об'єкт ViewHolder. RecyclerView створює рівно стільки view-елементів, щоб вистачило заповнити місце на екрані плюс декілька про запас [2]. При скролі списку або таблиці, даний елемент, використовує view-елемент, який зникає з екрану для відображення наступного члена колекції. Для створення і управління об'єктами ViewHolder призначений клас RecyclerView.Adapter. Він створює об'єкти ViewHolder на основі даних моделі і тим самим визначає, як будуть візуалізовані ті чи інші дані. Для оптимізації відображення великих колекцій данних використовується принцип пагінації. Пагінація дозволяє завантажувати і відображати малі порції даних раз за разом. Завантаження даних порціями, збільшує пропускну здатність мережі і понижує використання системних ресурсів. Ключовим елементом бібліотеки пагінації є PagedList, він відповідає за заввантаження порцій даних застосунку або сторінок. Кожен екземпляр класу PagedList отримує

актуальний знімок даних додатку з відповідного джерела даних – DataSorce [3]. DataSorce може надавати дані з різних джерел, наприклад файлів, бази даних, сервера або навіть комбінувати джерела для отримання найбільш повних і актуальних вибірок даних (рис. 2.14).

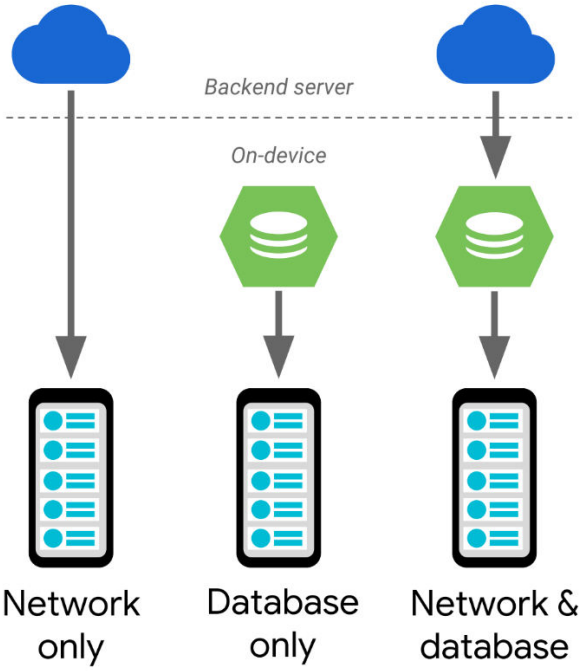


Рис. 2.14 Джерела даних для PagedList

Під час навіть незначного оновлення даних, для їх відображення система перерисовує весь список або таблицю, витрачаючи багато ресурсів на перерисовку данних, які не змінилися. Механізм DiffUtils.Callback дозволяє вираховувати різницю між наборами даних і формувати список операцій оновлення, що конвертують старий список в актуальний новий список. Обчислення різниці списків можна проводити синхронно, для уникнення блокування потоку користувацького інтерфейсу. RecyclerView.Adapter використовує DiffUtils.Callback для вирахування різниці між списками і перерисовування лише тієї частини даних, яка зазнала змін.

Навігація виникає між кількома кінцевими точками в додатку, кінцевими точками можуть бути різні зони розміщення контенту в додатку. Переході між кінцевими точками, здійснюється за допомогою логічних

звязків або дій (англ. actions). Компонент навігації андроїд дозволяє представити навігацію у вигляді графа, де кінцеві точки – це вершини графа, а дії - звязки в графі. Для здійснення навігації використовується NavigationController, параметром для якого слугують ті самі дії – звязки в графі [4]. Компонент навігації андроїд дозволяє не тільки переміщатися користувачу по додатку, він також дає змогу визначити анімаційю переходу між кінцевими точками, передавати дані між кінцевими точками і навіть програмно відкривати інші додатки, передаючи URI (Universal Resource Identifier), як параметр.

Швидкий підбір адрес під час геокодингу, реверс-геокодингу і автодоповнення, а також їх актуальність забезпечується за допомогою налаштувань АПІ для роботи з геоданими, зокрема, обмеження регіону пошуку адреси країною, містом або площиною. Настройки апі дозволяють також отримувати локалізовані адреси, сортувати їх за віддаленістю до певної географічної точки, популярністю або алфавітом.

Починаючи з версії андроїд 6.0 в операційній системі запроваджено дві особливості, що продовжують роботу девайса на одному заряді батареї завдяки контролю над додатками під час автономної роботи пристрою. Перший принцим називається Doze, він полягає в відкладенні активної роботи процесора мережевої активності для застосунків, якщо смартфон не використовувавсядовгий період часу. Другий принцип – режим очікування,

полягає у відстрочці фонової роботи для додатків, що довго не використовувалися.

Якщо користувач залишає пристрій не на зарядці на деякий сталий проміжок часу, з виключеним екраном, пристрій переходить в режим doze (рис. 2.15) .В режимі doze операційна система намагається заощадити заряд батареї, обмежуючи доступ додатків до мережі інтернет, вісрочує завдання, синхронізації і алярми, а також обмежує активне використання процесора

застосунками[5]. Операційна система покидає режим doze періодично, дозволяючи додаткам виконати всі відкладені задачі, синхронізації, алярми.

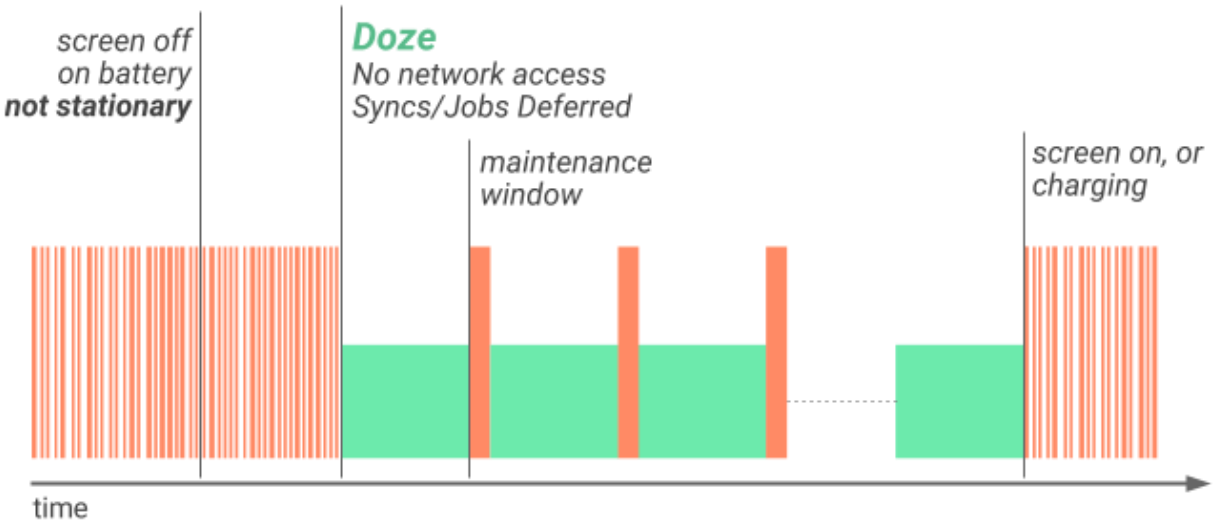


Рис. 2.15 Режим doze

Під час даного часового вікна, додаткам відкривається доступ в інтернет. В кінці часового вікна, система знову входить в режим doze. Вихід з режиму doze і поява часового вікна відбувається періодичноале період появи вікна щоразу більшає, для ефективнішої економії заряду батареї.

Режим очікування дає системі можливість визначити коли застосунок простоює і користувач не використовує його активно. Для визначення простою існують певні критерії. Додаток простоює, якщо:

1. Користувач явно не запустив додаток
2. Додаток не відображується на екрані
3. Додаток не генерує сповіщень, які відображаються на екрані блокування смартфона або на шторці сповіщень
4. Додаток не має прав адміністратора на цьому пристрої.

Коли користувач підключає девайс до електро-мережі, застосунок виходить із стану простою, отримує доступ до інтернету і має можливість виконати всі відкладені задачі. Якщо застосунок дуже довго перебуває в стані простою, система надає йому доступ до інтернету і дозволяє виконати відкладені задачі лише, приблизно, раз в день.

Для створення інтуїтивно-зрозумілого і зручного користувацького інтерфейсу був застосований підхід матеріал дизайн (рис. 2.16). Це набір вимог, правил і рекомендацій щодо побудови користувацького інтерфейсу.

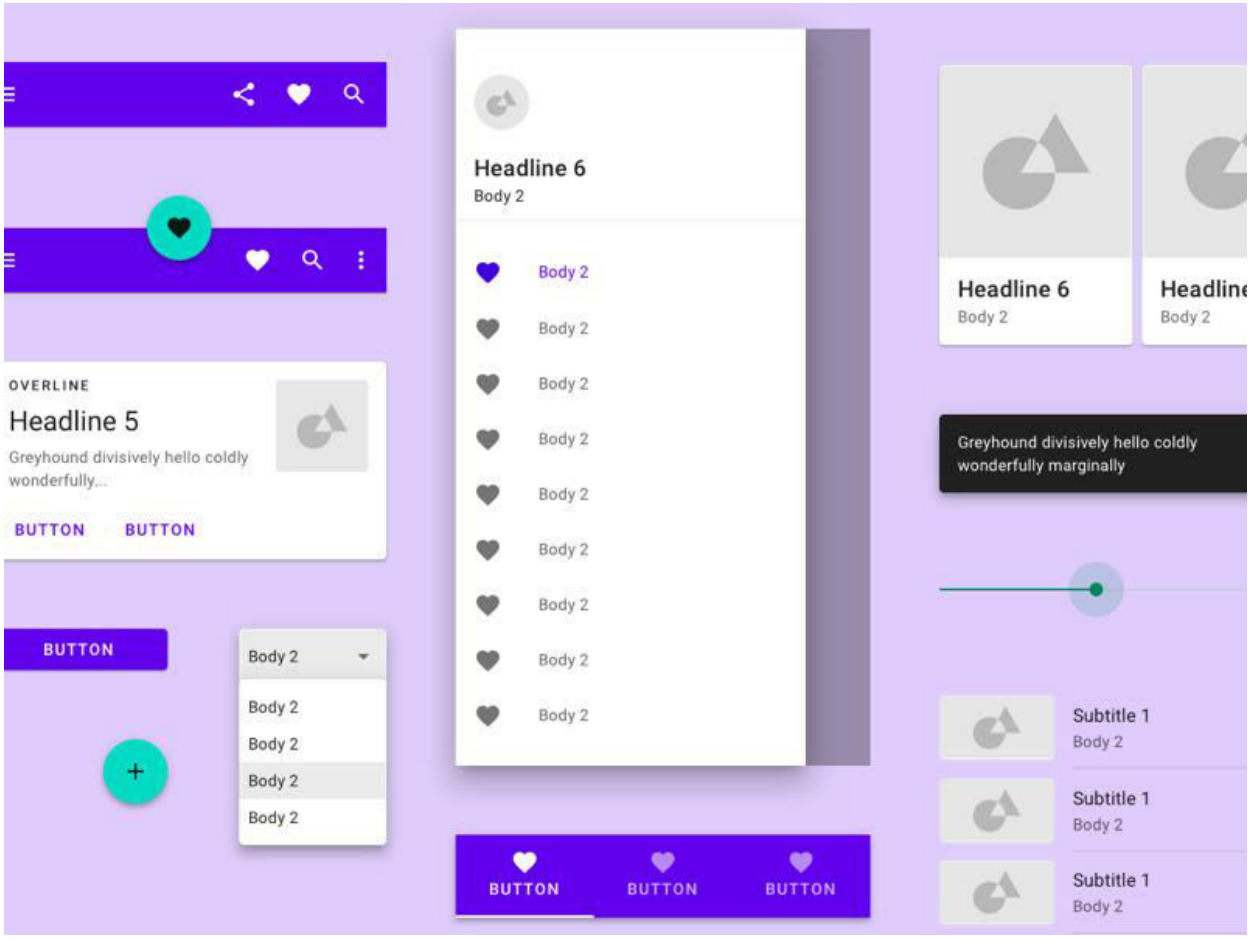


Рис. 2.16 Material design

Він описує поведінку елементів користувацького інтерфейсу, їхнє положення на екрані та один відносно іншого, сферу використання того чи іншого елемента. Також цей підхід визначає вигляд кожного елемента інтерфейсу і застосунку в цілому вигляд і тривалість анімацій, діалогових вікон тощо[6].

2.3 Мобільний додаток, як вибір платформи

Мобільний додаток — програмне забезпечення, яке використовується для запуску на телефонах, планшетах, годинниках та інших мобільних пристроях [7]. З самого початку, мобільні додатки були задумані, як органайзер чи то помічник і призначалися для перегляду електронної пошти, звукозапису, доступу до баз даних, відслідковування часу. Але попит на ці

додатки сприяв їхньому поширенню і в інших сферах: відслідковування замовлень, відеозапис, автоматизація, онлайн-покупки, стрімінг, GPS, мобільні ігри та інше. Розповсюдженню мобільних додатків найбільше сприяють дві платформи GooglePlayStore (операційна система Android) та AppStore (операційна система iOS).

Велике число мобільних пристроїв поставляється користувачам з деякими вже встановленими програмами – музичний плеєр, електронні карти, календар браузер, поштова скринька емейл або навіть більше. Застосунки, які не встановленні на девайсі спочатку, поширюються за допомогою спеціальних платформ або магазинів застосунків. Діючі зараз магазини застосунків – App store, BlackBerryWorld, Windows Phone Store і Google Play. Також додаток може бути встановлений на смартфон за допомогою загрузочного файлу, що має назву пакет застосунку андроїд, APK (англ. Android application package).

Додатки в основному призначені для комунікації, підвищення продуктивності користувача, отримання різноманітної інформації, доступ до інформації фондових ринків, календаря, контактів, інформації про погоду. Проте зі зростанням попиту сфера застосування мобільних додатків щороку збільшується.

Застосунки можна класифікувати багатьма способами. Частіше за все їх поділяють на веб-додатки, гібридні і нативні.

Сучасні мобільні веб-додатки створюються на основі технологій JavaScript або CSS і HTML5. Для коректної поведінки додатку необхідні доступ в інтернет і дотримання певних принципів дизайну користувацького інтерфейсу під час розробки. Перевагою цього виду застосунків є те, що вони займають дуже мало простору на носії інформації, особливо, в порівнянні з нативними і гібридними додатками. Дані зберігаються на інтернет серверах і стають доступними користувачу за наявності інтернету.

Гібридний застосунок - це поєднання нативного і веб-застосунку. Такі додатка створюються на основі технологій React Native, Apache Cordova, Xamarin, Flutter та інших подібних технологій. Це дозволяє підтримувати як веб, так і нативні технології на різних платформах. Такі застосунки швидко й легко розробляти написаний код буде виконуватися на пристроях з різними операційними системами. Недоліком гібридних додатків є втрата в продуктивності, обмеженість при розробці і можливі незручності для користувачів певних платформ, так як інтерфейс користувача на всіх платформах буде однаковий і можливо не звичний для користувачів певних платформ, що зіпсує користувацький досвід.

Нативними називаються всі застосунки, які призначені лише для однієї платформи, наприклад, додаток розроблений для платформи iOS не може бути запущений в середовищі Android. Розробка нативних додатків дає змогу розробити найзручніший інтерфейс користувача, забезпечує надійність, продуктивність і позитивний досвід для користувача. Розробка застосунку для конкретної платформи дозволить користувачу без зусиль переміщатися між додатками в системі і передавати дані між додатками. Нативні застосунки не накладають ніяких обмежень на використання.

Розробка мобільних застосунків проводиться з урахуванням особливостей мобільних приладів і певних обмежень. Так мобільні додатки працюють на заряді батареї, мають менш потужний в порівнянні з персональним комп’ютером процесор, особливості - засоби для геолокації, різноманітні сенсори – магнітометр, акселерометр, гіроскоп, NFC, Bluetooth, датчики руху, сканер відбитка пальця і т.д., камеру. Розробники повинні враховувати наявність великої кількості дисплеїв з різною діагоналлю, специфіку апаратних засобів пристрою.

2.4 REST API. Серверна частина

REST API – програмний інтерфейс додатку, який використовує методи протоколу HTTP – GET, PUT, POST і DELETE для виконання операцій над даними. (рис. 2.17)

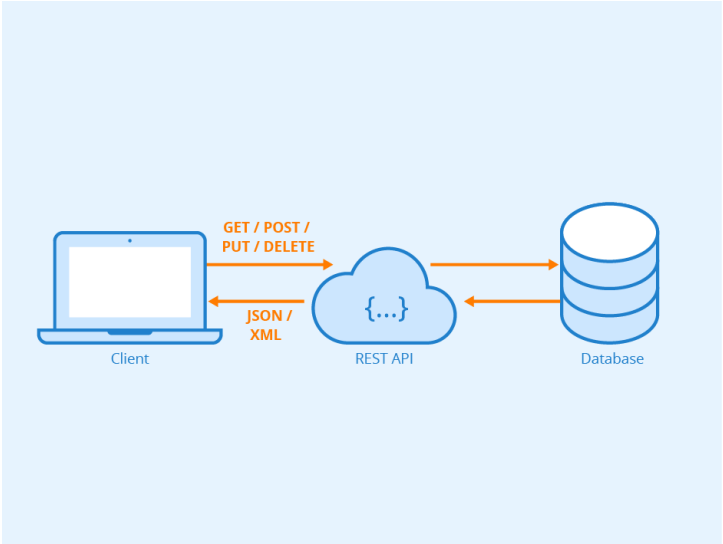


Рис. 2.17 REST API

Де GET – зчитування даних, PUT – запис даних, POST – оновлення даних, DELETE – видалення даних, детальніше ці методи описані в стандарті RFC 2616. API – це код який дозволяє комп'ютерним програмам взаємодіяти між собою. API визначає спосіб комунікації для комунікації з операційною системою або іншими програмами [8].

Технологія REST має перевагу над більш надійним підходом - Simple Object Access Protocol (SOAP), бо REST менше навантажує мережу, збільшуючи тим самим пропускну здатність мережі і тому є більш зручною в використанні і ефективною технологією. REST API розбиває всю транзакцію на менші модулі. Кожен модуль адресується конкретній окремій частині транзакції. Така модульність забезпечує велику гнучкість при розробці, але значно збільшує кількість роботи для розробників, якщо розробляти систему з нуля, тому окремі сервіси пропонують моделі для розробників, наприклад, Cloud Date Management Interface скорочено CDMI, Amazon S3, OpenStack Swift – це найбільш популярні.

REST-запити не мають стану, тому ця технологія широко використовується в хмарних програмах. Компоненти, що не мають стану, можуть бути легко перезавантажені, якщо щось піде не так, також вони легко маштабуються, щоб витримувати великі навантаження. Це можливо завдяки перенаправленню запитів на інші екземпляри компонентів, нічого не

потрібно зберігати, для проведення наступної транзакції. Це робить REST кращим для використання в web, а хмарні обчислення і мікросервіси зроблять REST основним в майбутньому.

Щоб відповідати справжньому REST-API, web-сервіси мають дотримуватися шістьох пунктів:

1. Використання уніфікованого інтерфейсу (Uniform interface). Ресурси повинні бути унікально ідентифіковані за допомогою однієї URL і використовувати лише методи протоколу HTTP – GET, PUT, POST, DELETE для маніпулювання даними.
2. Клієнт-серверна архітектура. Повинне бути чітке розділення на клієнт і сервер. Інтерфейс користувача і збір запитів – відбувається на стороні клієнта. Управління навантаженням, безпека і доступ до даних здійснюються на стороні сервера.
3. Операції, що не зберігають стан. Операції не повинні зберігати стан, а управління станом, якщо воно необхідне, повинно відбуватися на стороні клієнта, не на сервері.
4. Кешування ресурсів в стилі REST. Дозволяється кешування всіх ресурсів, якщо це не заборонено явно.
5. Поділ системи на шари. REST дозволяє ділити архітектуру на декілька шарів серверів.
6. Код на запит. В більшості випадків сервер повертає статичне представлення ресурсів в формі XML або JSON. Проте, коли це необхідно, сервер може повертати код для виконання на клієнті.

2.5 Чиста архітектура

Існує багато архітектур, підходів до розробки програмного забезпечення. Серед них:

- Гексагональна архітектура
- Цибулева архітектура
- Кричуща архітектура
- DCI

- DCE

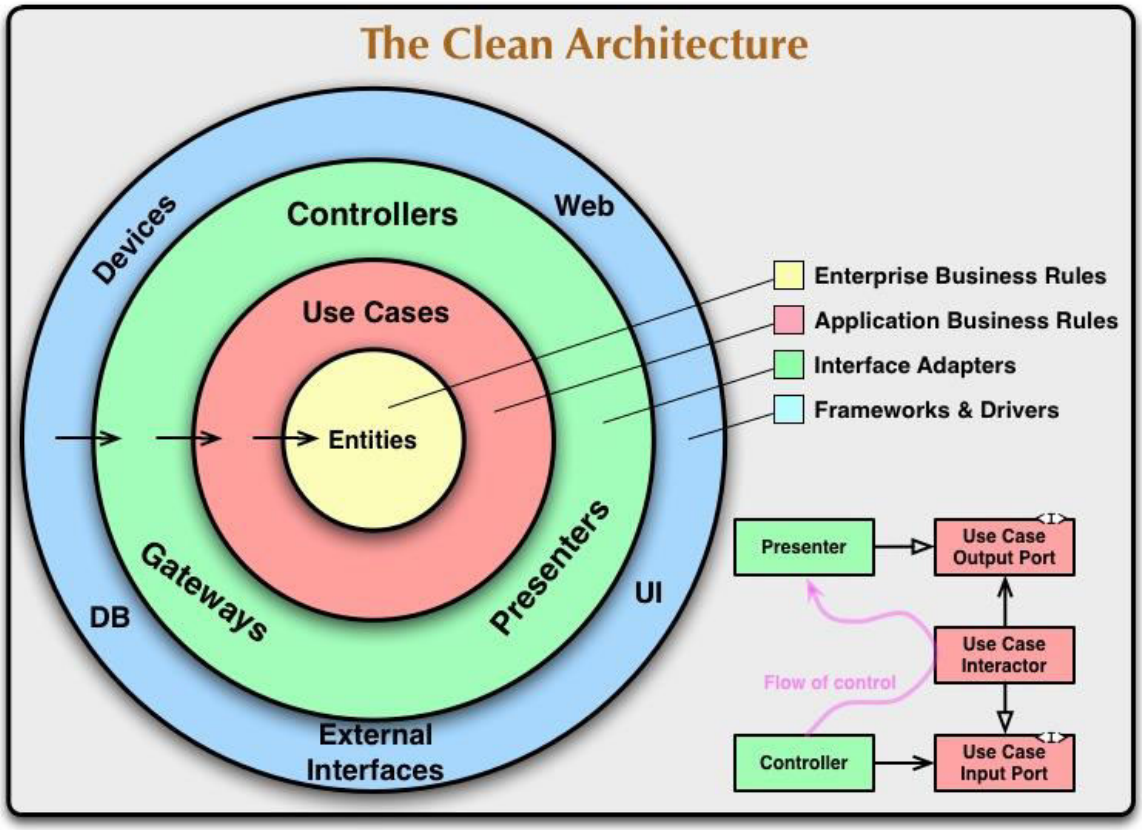


Рис. 2.18 Загальна схема архітектури

Всі ці архітектури (див. рис. 2.18) варіюються в деяких їхніх деталях, проте вони дуже схожі. Усі вони мають однакову мету, які полягає в зменшенні звязності. Зменшення звязності досягається завдяки розділенню програми на шари. Кожна програма повинна мати як мінімум шар бізнес логіки і інші шари з інтерфейсами[9].

Всі ці архітектури дозволяють створити систему, яка:

1. Незалежна від фреймворків. Архітектура не зав’язана на існуванні якоїсь бібліотеки. Це дозволяє використовувати фреймворки всього-лише як інструменти, а не робити їх ключовими в системі, накладаючи обмеження на розробку.
2. Тестована. Бізнес-логіка може бути протестована незалежно від серверу, бази даних, інтерфейсу користувача чи інших зовнішніх елементів.

3. Незалежна від користувацького інтерфейсу. Зміни в користувацькому інтерфейсі вносяться легко без внесення змін в решту програми. Наприклад web-інтерфейс може бути замінено на інтерфейс консолі без внесення змін до бізнес логіки.
4. Незалежна від бази даних. Зміна бази даних, як то SQL Server на MongoDB або Firebase на Realm, не впливає на бізнес логіку.
5. Незалежність від зовнішніх факторів. Бізнес-логіка просто не має доступу до будь-яких зовнішніх факторів, тому вони не мають ніякого впливу.

Концентричні кола в схемі архітектури відображають різні області програми. Чим глибше знаходиться коло, тим вищий рівень абстракції програми воно відображає. Зовнішні кола – це засоби. Внутрішні кола – це логіка. Ключове правило, що змушує архітектуру працювати – правило залежностей, воно гласить, що залежності в сирцевому коді мають вказувати лине на внутрішні шари кола. Ніщо у внутрішніх ділянках кола не може зсилатися на зовнішні рівні. Також формати і типи даних визначені в зовнішніх шарах не повинні використовуватися на внутрішніх шарах.

Суб’єкт. Суб’єкт містить в собі значущі для розробки правила бізнес-логіки. Це може бути структура даних з функціями або клас з методами. Одні й ті ж самі суб’єкти використовуються в усій системі в цілому, а не тільки в певній програмі або додатку. Суб’єкт містить, найбільш загальні правила, які стосуються бізнес логіки, вони найменше піддаються змінам, коли міняються зовнішні шари програми, наприклад вони не повинні бути затрогнуті при змінах в навігації додатку або мір безпеки. Ніяка зміна зовнішнього шару в будь-якій програмі системи, не повинна викликати змін в шарі суб’єкта.

Варіанти використання. Програма на цьому рівні містить логіку яка стосується, лише цієї конкретної програми, а не системи в цілому. Рівень містить в собі правила щодо всіх варіантів використання системи. Ці сценарії оперують потоком даних з шару суб’єктів і користуються правилами бізнес

логіки визначеними в суб’єкті для досягнення мети сценарію. Зміни в шарі варіантів використання не повинні впливати, на шар суб’єктів, також зовнішні шари і зміни в них, в свою чергу не повинні впливати на шар варіантів використання. Проте зміни в шарі варіантів використання будуть впливати на будову і зміст зовнішніх шарів.

Інтерфейс адаптери. Функціонал цього шару представляє собою набір адаптерів, що конвертують дані з шарів суб’єктів і варіантів використання в формати найбільш підходящі для зовнішніх засобів, як то база даних або Web. Це той рівень на якому, наприклад, використовується патерн MVC для графічного інтерфейсу користувача. Презентери, контролери і вью – все це знаходиться на рівні інтерфейс адаптерів. Моделі на цьому рівні – це структури даних, які використовуються для передавання даних від контролерів і презентерів на рівень варіантів використання і навпаки – від шару варіантів використання до контролерів і презентерів. Дані на цьому рівні конвертуються з форматів, якими оперують варіанти використання в формати якими оперують фреймворк для зберігання даних, сервер, тощо. На код всередині цього шару ніяк не впливають чинники з зовнішнього шару, такі як база даних, сервер чи інші додатки з якими комунікує програма. Також цей шар містить адаптери для конвертації даних із зовнішього шару в дані, які сприймають шари варіантів використання і суб’єктів.

Фреймворки і драйвери. Самий зовнішній шар архітектури який складається з фреймворків та інструментів таких як база даних, веб-фреймворк і так далі. Весь код в цьому рівні призначений для комунікації вжготових інструментів і рішень з внутрішніми шарами.

Необов’язково обмежуватися цими чотирма шарами, хоча в більшості випадків, чотирьох достатньо, головне дотримуватися правила залежнотей.

Для передачі даних між шарами без порушення правила залежностей користуються принципом інверсії залежностей. Він може бути реалізований за допомогою інтерфейсів. Для реалізації принципу необхідно виділити

абстракцію для комунікації з зовнішнім рівнем, а реалізація класу вже знаходить на зовнішньому рівні. Таким чином внутрішній рівень прив’язується до абстракцій кожна кожна конкретна реалізація не впливає на внутрішній рівень.

2.6 Патерн MVVM

Патерн MVVM дозволяє відділити логіку застосунку від графічного інтерфейсу. Цей патерн є архітектурним, а значить він впливає на архітектуру всієї програми. MVVM складається з трьох компонентів моделі, вью-моделі і вью [10]. (рис. 2.19)

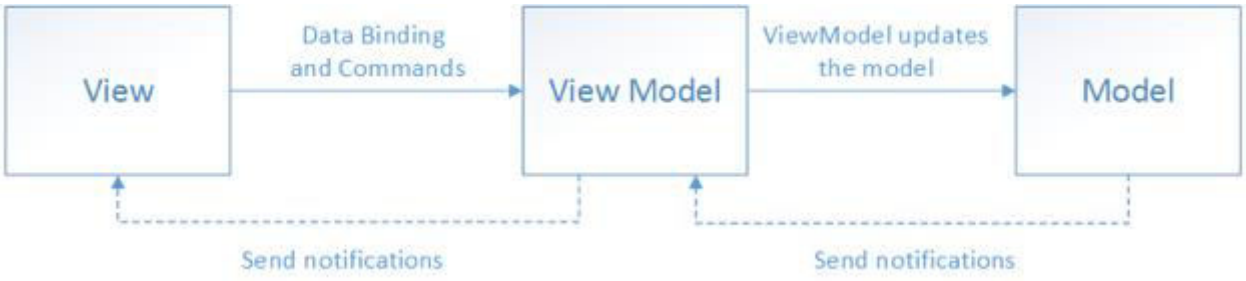


Рис. 2.19 Патерн MVVM

Модель описує дані, які використовуються в застосунку. Також модель містить логіку звязану з цими даними, наприклад логіку ваідації. В той же час модель не повинна містити логіку звязану з візуалізацією і відображенням даних і не повинна ніяк взаємодіяти з візуальними елементами.

Вью або представлення визначає графічний інтерфейс, за допомогою якого користувач взаємодіє з системою. Всередині вью не повинна бути розміщена логіка, додатку, лише логіка відображення даних.

Вью-модель або модель представлення звязує модель і вью через механізм прив’язки даних. Вью модель містить логіку для отримання даних з моделі, які потім передаються у вью, також вью модель містить логіку для оновлення даних моделі. Вью модель напряму не комунікує з вью, для цього використовується патерн обзервер, вью модель сповіщає про зміну даних моделі, а вью підписується на ці зміни і реагує відповідно.

Висновок до розділу 2

В другому розділі були вироблені функціональні і нефункціональні вимоги, розглянуті інструменти для проектування системи управління спільними поїздками.

Головними частинами системи є:

- Сервер
- Клієнт (мобільний додаток)
- Система авторизації Google Sign In
- База даних

Під час проектування системи були досліджені різні засоби для реалізації взаємодії сервера і клієнта, система авторизації на основі протоколу OAuth2, підходи для проектування і реалізації користувацького інтерфейсу, чотирьохшарова архітектура для розробки додатків – Clean architecture, архітектурний патерн MVVM.

В результаті проектування було вибрано наступні інструменти:

- Android sdk – для розробки інтерфейсу клієнтської частини
- Android MVVM – для розробки рівня презентації застосунку
- GraphQL – для звязку клієнта і рервера
- PostgreSQL – для збереження даних
- Retrofit – для побудови запитів на стороні клієнта
- Koin – для предоставлення залежностей всередині програми

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Реалізація об’єктної моделі бізнес логіки

Звязки між суб’єктами в бізнес логіці

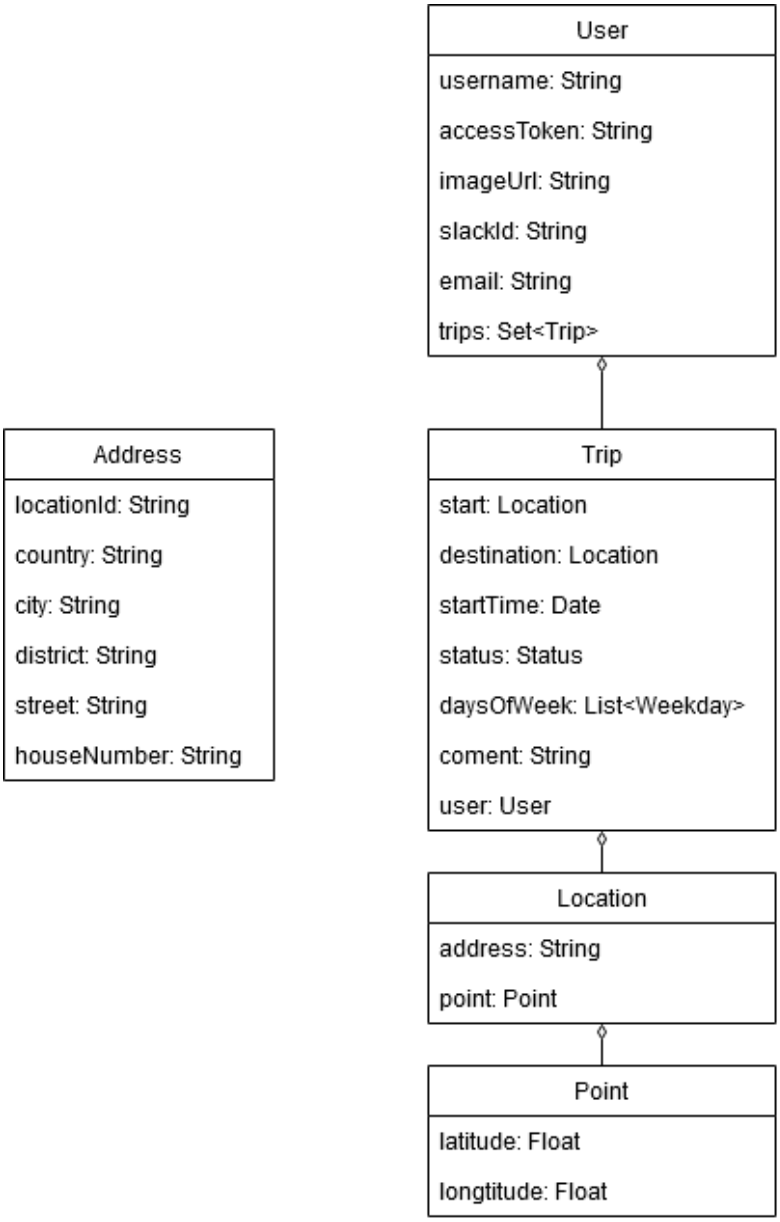


Рис. 3.1 Звязки між суб'єктами

- Користувач (User)
- Поїздка (Trip)
- Місце розташування (Location)
- Географічна точка (Point)
- Адреса (Address)

Суб’єкт User (рис. 3.2) – клас користувача

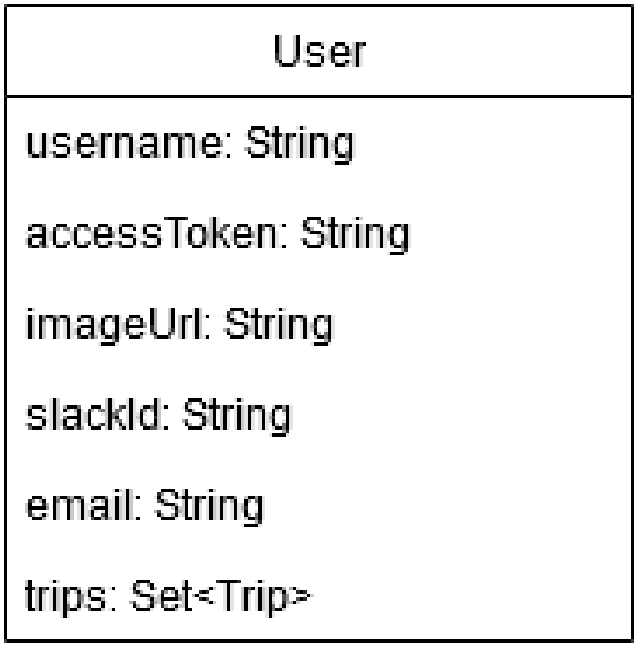


Рис. 3.2 Суб'єкт User

username – ім’я користувача в системі, використовується для відображення профілю користувача

accessToken – токен авторизації користувача, використовується при відправці запитів на сервер

imageUrl – посилання на аватар користувача, використовується при відображенні профілю користувача

slackId – ідентифікатор користувача в месенджері, використовується для комунікації між користувачами

email – електронна адреса користувача, використовується при авторизації і відображенні профілю

trips – маршрути, власником яких є даний користувач, використовується для відображення профілю, створення, редагування, активації, деактивації і видалення маршрутів користувача, при пошуку спільних маршрутів

Суб’єкт Trip (рис. 3.3) – клас поїздки

Рис. 3.3 Суб'єкт Trip

start – місце знаходження початку маршруту, задає напрямок маршруту, використовується при побудові маршруту і відображення маршруту на карті

destination - місце знаходження кінця маршруту, задає напрямок маршруту, використовується при побудові маршруту і відображення маршруту на карті

startTime – час початку поїздки, використовується для відображення даних про маршрут і пошуку спільних поїздок

status – статус поїздки, значення {active (активна), inactive(не активна)}, використовується для відображення поїздок, активації або деактивації поїздки, пошуку спільних поїздок

daysOfWeek – дні тижня в які відбувається поїздка, значення {Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, Saturday}, використовується для відображення поїздок, пошуку спільних поїздок

coment – додаткова текстова інформація від автора маршруту, використовується для відображення маршрутів

user – власник маршруту, використовується для відображення профілю користувача, відображення поїздок, пошуку спільних поїздок, комунікації пасажира з власником маршруту

Суб’єкт Location (рис. 3.4) – клас місця знаходження

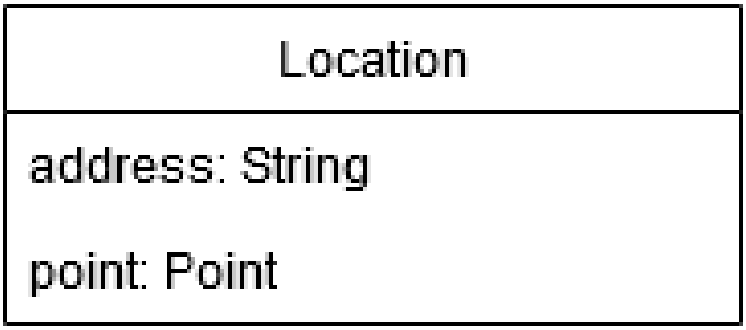


Рис. 3.4 Суб'єкт Location

address – текстова адреса, використовується для відображення деталей маршруту

point – географічна координата, використовується для позначення точки на карті, побудови маршруту

Суб’єкт Point (рис. 3.5) – клас географічної точки

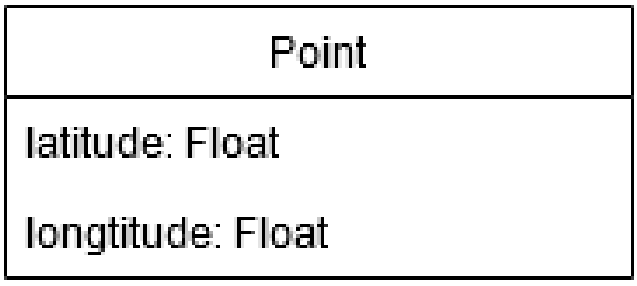


Рис. 3.5 Суб'єкт Point

latitude – географічна широта

longtitude – географічна довгота

Суб’єкт address (рис. 3.6) – клас адреси

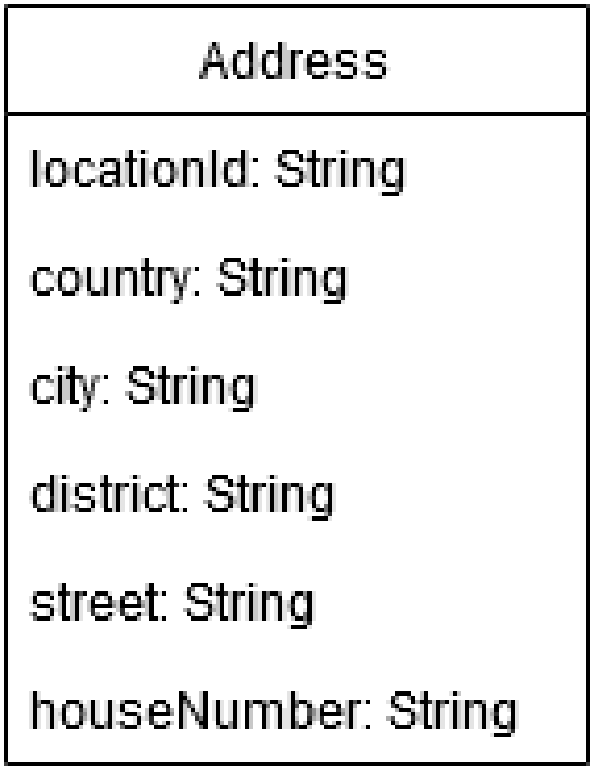


Рис. 3.6 Суб'єкт Address

locationId – ідентифікатор адреси, використовується при автодоповненні і реверс геокодінгу

country – назва країни, використовується при автодоповненні, створенні маршруту

city – назва міста, використовується при автодоповненні, створенні маршруту

district – назва району, використовується при автодоповненні, створенні маршруту

street – назва вулиці, використовується при автодоповненні, створенні маршруту

houseNumder – номер будинку, використовується при автодоповненні, створенні маршруту

3.2 Інструкція користувача

Розділ відображай основні частини інтерфейсу додатку і описує їх функціональні здібності.

Екран авторизації (рис. 3.7)

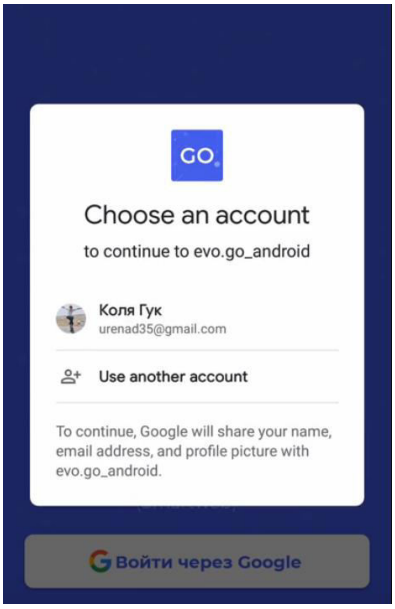


Рис. 3.7 Екран авторизації

На даному екрані користувач має можливість авторизаватися в один клік , просто вибравши один із гугл-акаунтів збережених на смартфоні.

Екран зі списком всіх маршрутів (рис. 3.8)

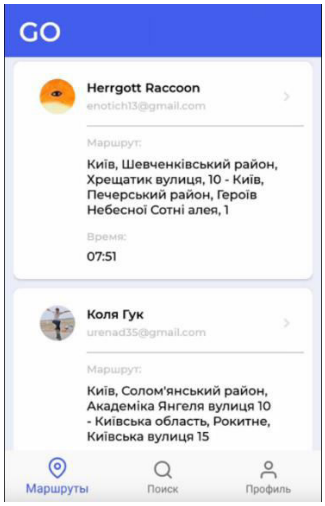


Рис. 3.8 Список всіх маршрутів

На даному екрані користувач може бачити всі активні маршрути інших користувачів вистеми. Внизу є панель для навігації по додатку.

Екран пошуку маршруту (рис. 3.9)

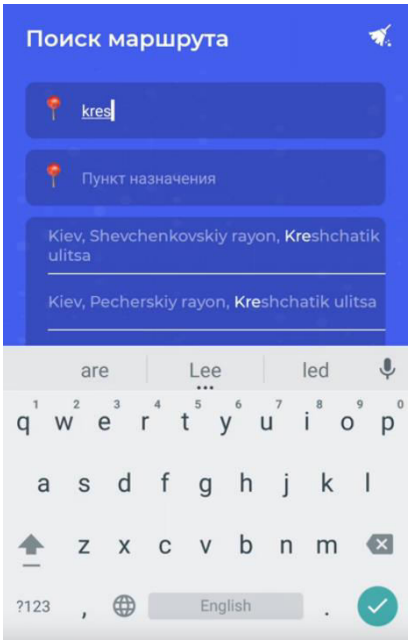


Рис. 3.9 Пошук маршруту

На даному екрані, користувач може відфільтрувати маршрути, задавши початкову і кінцеву точку маршруту. Для введення адреси, наявна функція автодоповнення.

Екран профілю користувача (рис. 3.10)

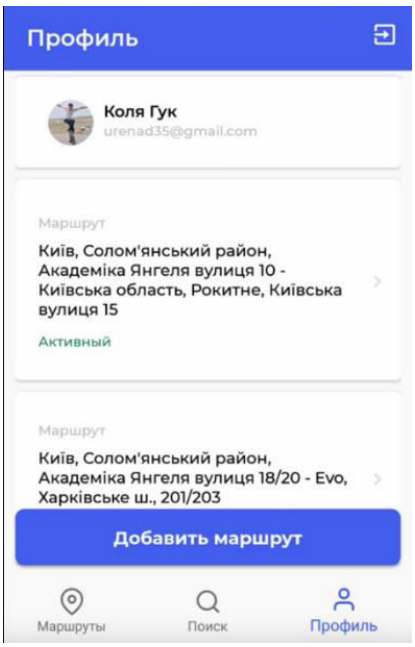


Рис. 3.10 Екран профілю

На даному екрані користувач може переглянути свої маршрути, додати новий маршрут, вийти з системи.

Екран деталей маршруту (рис. 3.11)

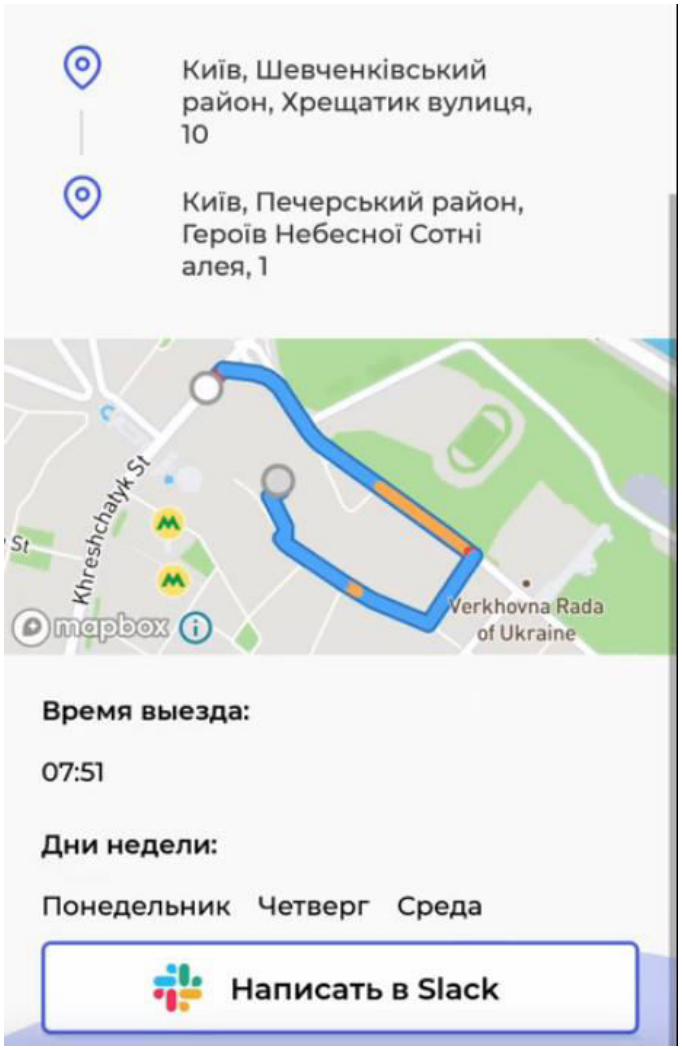


Рис. 3.11 Деталі маршруту

На даному екрані користувач може продивитися деталі маршруту, зокрема початкову і кінцеву точку маршруту, дні і точний час коли відбувається поїздка, вид маршруту на карті. Також користувач може перейти до комунікації з власником маршруту.

Екран управління маршрутом (рис. 3.12)

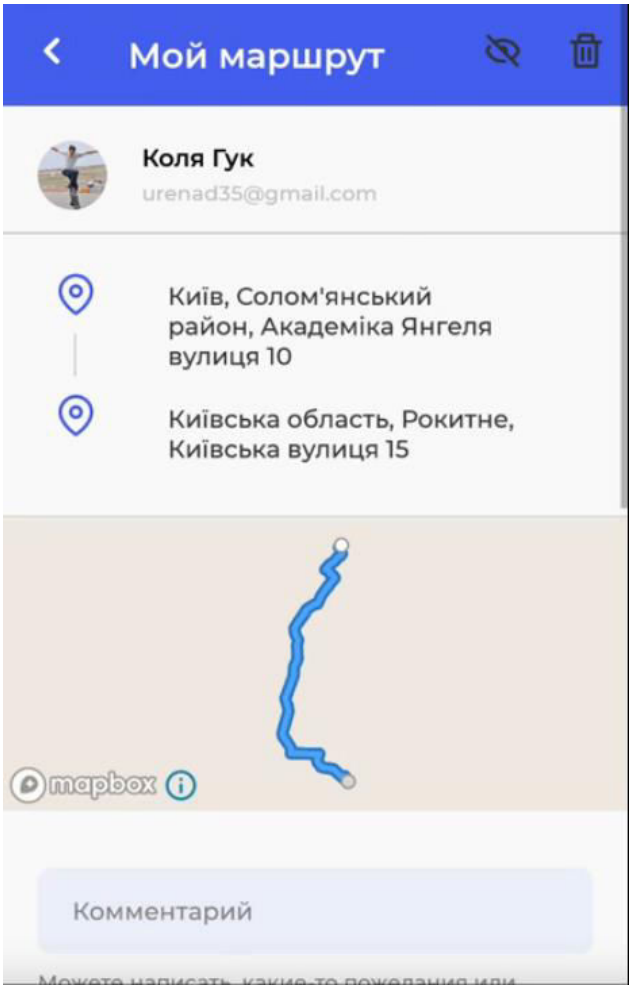


Рис. 3.12 Управління маршрутом

На даному екрані користувач може продивлятися деталі свого маршруту, редагувати коментар до маршруту, дні тижня і точний час коли відбувається поїздка. Також тут можна активувути/деактивевати маршрут або видалити маршрут.

Еран додавання нового маршруту (рис. 3.13)

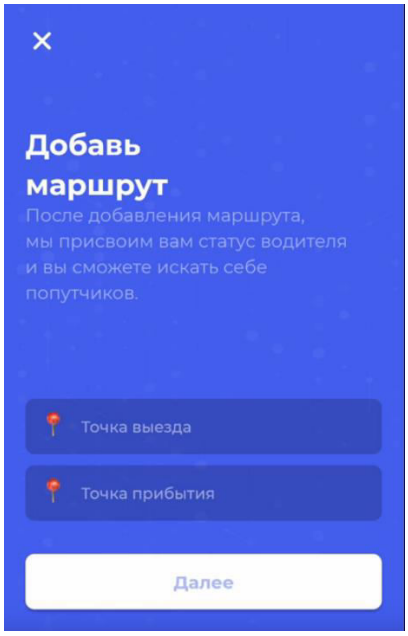


Рис. 3.13 Додавання нового маршруту

На даному екрані можна вказати кінцеві точки нового маршруту, для зручності користувача працює автодоповнення адреси.

Екран попереднього перегляду маршруту (рис. 3.14)

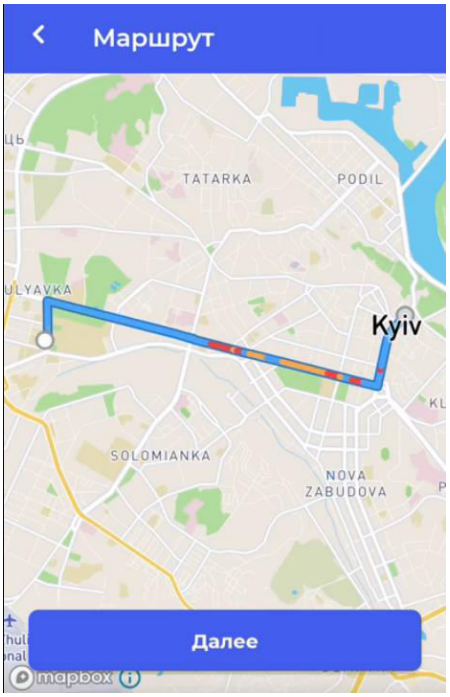


Рис. 3.14 Попередній перегляд маршруту

На даному екрані, користувач може переглянути, як буде виглядіти маршрут, до його збереження

Екран для введення деталей маршруту (рис. 3.15)

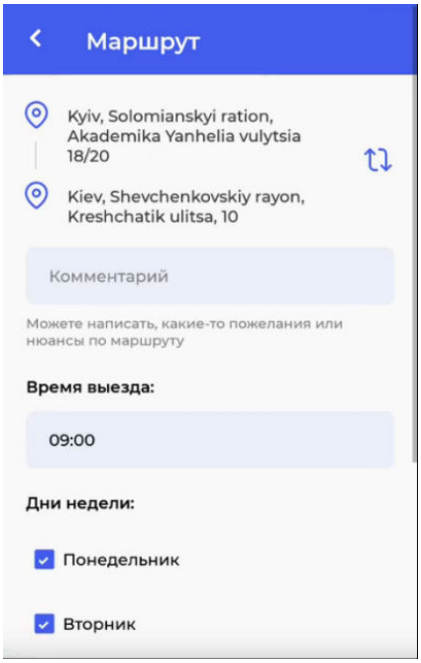


Рис. 3.15 Додавання деталей маршруту

На даному екрані користувач може ввести деталі маршруту, зокрема, коментар, дні тижня і точний час коли відбувається поїздка.

Екран перезавантаження даних (рис 3.16)

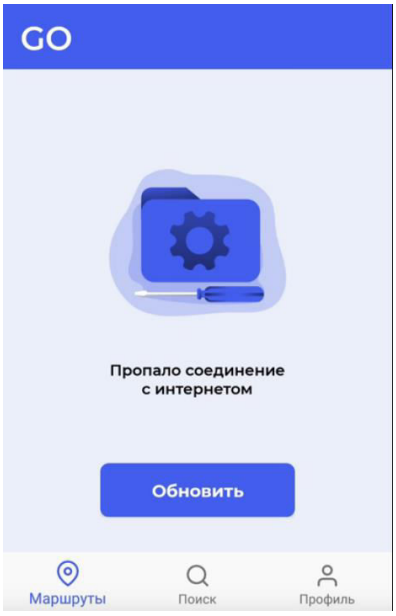


Рис. 3.16 Перезавантаження маршрутів

В додатку передбачена можливість перезавантажити дані, якщо тимчасово втрачено з’єднання з мережею

Висновки до розділу 3

Клієнтська частина додатку розроблена на основі фреймворка андроїд, на мові програмування Kotlin. Додаток був спроектований згідно положень чистої архітектури. Програма розділена н чотири рівня – рівень бізнес сущностей, рівень варіантів використання, рівень інтерфейс адаптерів і рівень фреймворків і драйверів. На рівні інтерфейсів адаптерів використовується архітектурний патерн MVVM, який дозволяє відділити логіку роботи з даними від логіки відображення, що дозволяє швидко і легко вносити зміни в інтерфейс додатку не затрогуючи при цьому шар роботи з даними.

Для забезпечення актуальності даних клієнт постійно комунікує з сервером і отримує найновіші дані. Для комунікації клієнта і сервера використовується мова запитів GraphQL.

Програм має інтуїтивно зрозумілий інтерфейс і простий дизайн, в більшості частин програми є такі функції як візуалізація маршрутів на карті і механізм автодоповнення адреси. При дизайні графічного інтерфейсу були дотримані правила розробку матеріал дизайну і основні правила user experience в андроїд.

ВИСНОВОК

В результаті виконання дипломного проекту був розроблений мобільний додаток на платформі андроїд. Для реалізації системи була вибрана платформа мобайл, адже тільки вона зможе в будь-який час забезпечити доступ до даних користувача.

Платформа була спроектована на базі найкращих практик для розробки андроїд, а саме чотирохшарової архітектури – чистої архітектури, з використанням архітектурного патерна MVVM, архітектури клієнт-сервер і мови запитів GraphQL, формування запитів і обробка відповідей з сервера реалізовані в реактивному стилі за допомогою бібліотеки RxJava2, дотримані принципи програмування SOLID і DRY, залежності в проекті вирішуються за допомогою принципу ін’єкції залежностей, у проектуванні дизайну враховані положення матеріал дизайну та user experience андроїд. Програма реалізована на основі фреймворків android та android jetpack на мові програмування Kotlin, для розмітки використовувалась мова XML.

Розроблена програма надає доступ до маршрутів, надає максимально актуальні дані про маршрути, є стабільною, безпечною і швидкодійною. Додаток має простий дизайн та інтуїтивно зрозумілий графічний інтерфейс.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Google Sign-In for Websites [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com/identity/sign-in/web/backend-auth
2. Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/topics/ui/layout/recyclerview
3. Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/topic/libraries/architecture/paging
4. Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/guide/navigation/navigation-getting-started
5. Developers [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/training/monitoring-device-state/doze-standby
6. Material design [Електронний ресурс] – Режим доступу до ресурсу: https://material.io/design
7. Wikipedia [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Mobile_app
8. Search Architecture [Електронний ресурс] – Режим доступу до ресурсу: https://searchapparchitecture.techtarget.com/definition/RESTful-API
9. Clean Architecture: A Craftsman's Guide to Software Structure and Design (Robert C. Martin Series) - Prentice Hall; 1 edition (Вересень 20, 2017) - 428 ст.
10. Metanit [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/wpf/22.1.php

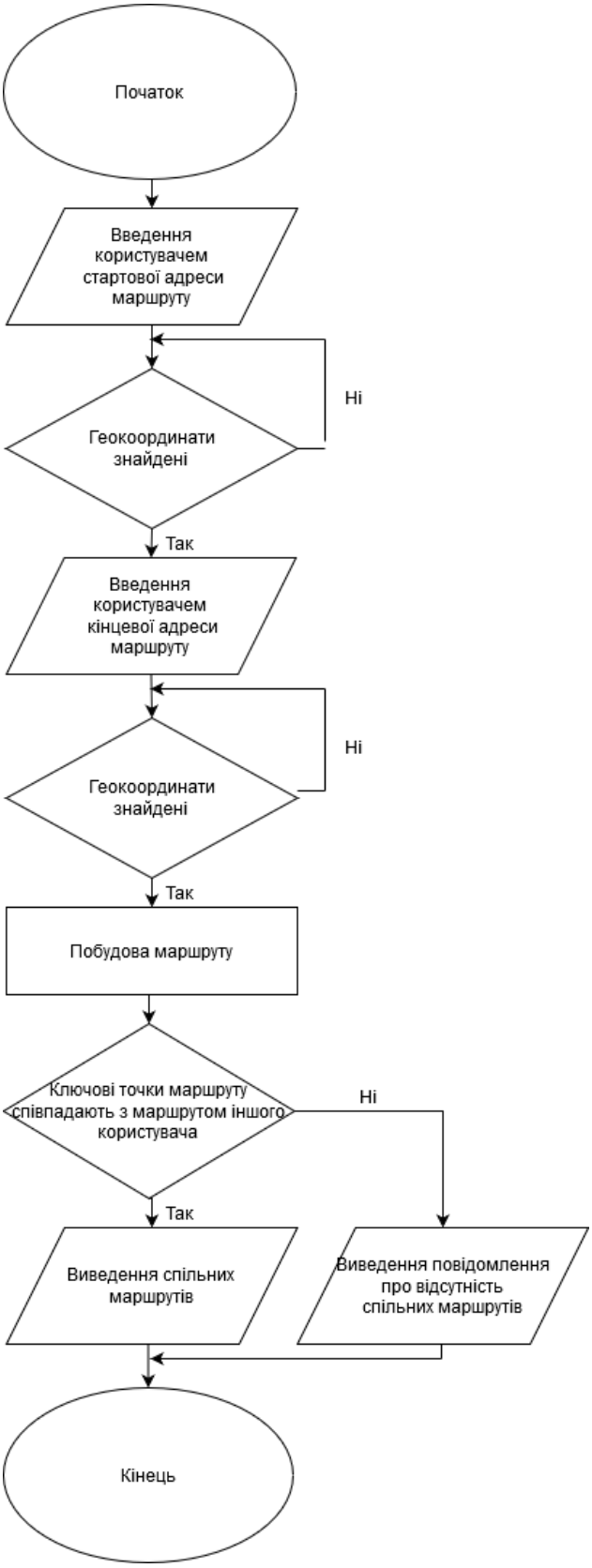
ДОДАТОК 1

МОБІЛЬНИЙ ДОДАТОК ДЛЯ УПРАВЛІННЯ СИСТЕМОЮ СПІЛЬНИХ ПОЇЗДОК

АЛГОРИТМ ПОШУКУ МАРШРУТУ

Аркушів 1

Київ – 2020



					ІАЛЦ 467800.004 Д1	
Зм.	*Арк.*	*Прізвище*	*Підпис*	*Дата*		
Розроб.		Гук М.О.			*Мобільний додаток для управління системою спільних поїздок* Алгоритм пошуку маршрутів	*Арк.* 2 / *Аркушів*
Перевірив						
Н. кон.						*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, ІП-62*
Затв.						

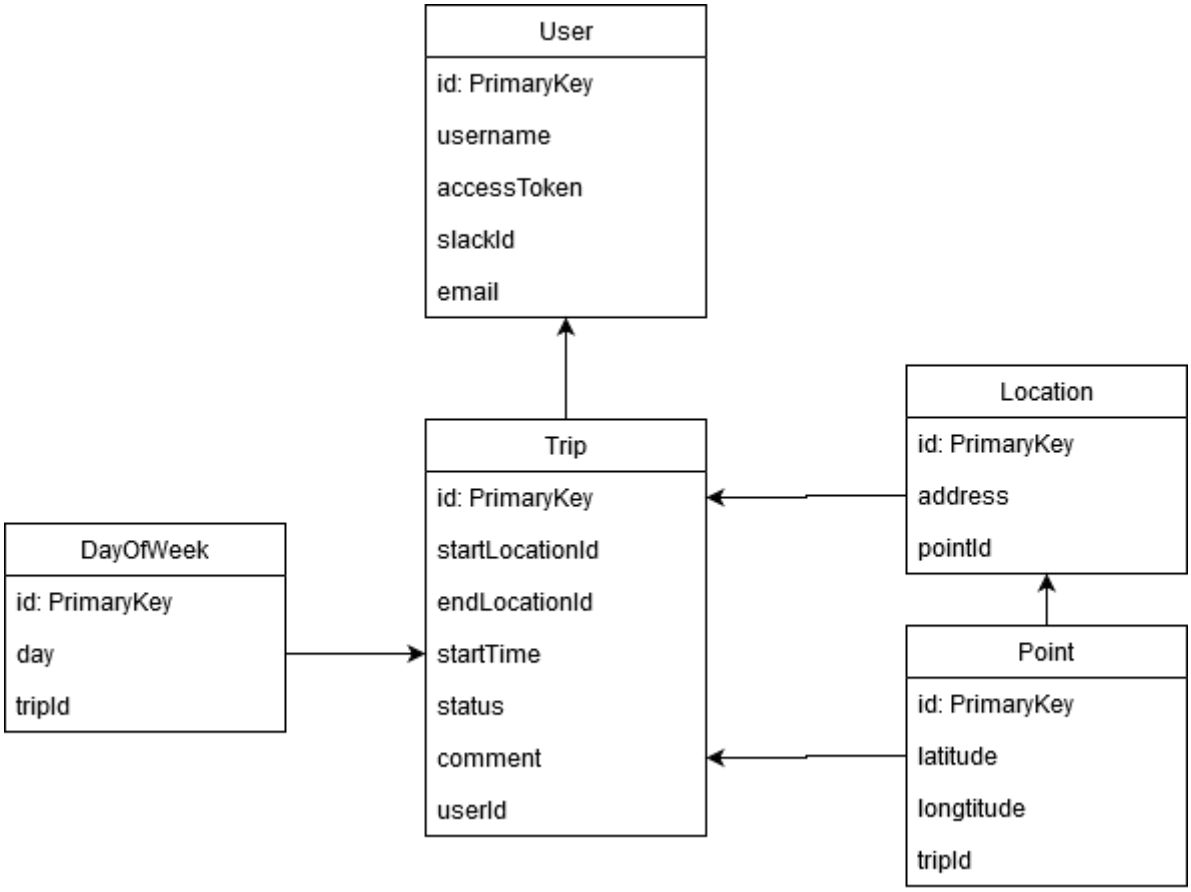
ДОДАТОК 2

МОБІЛЬНИЙ ДОДАТОК ДЛЯ УПРАВЛІННЯ СИСТЕМОЮ СПІЛЬНИХ ПОЇЗДОК

СХЕМА БАЗИ ДАНИХ

Аркушів 1

Київ – 2020



					ІАЛЦ 467800.005 Д2	
Зм.	*Арк.*	*Прізвище*	*Підпис*	*Дата*		
Розроб.		Гук М.О.			*Мобільний додаток для управління системою спільних поїздок* Діаграма бази даних	*Арк.* 2 / *Аркушів*
Перевірив						
Н. кон.						*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, ІП-62*
Затв.						

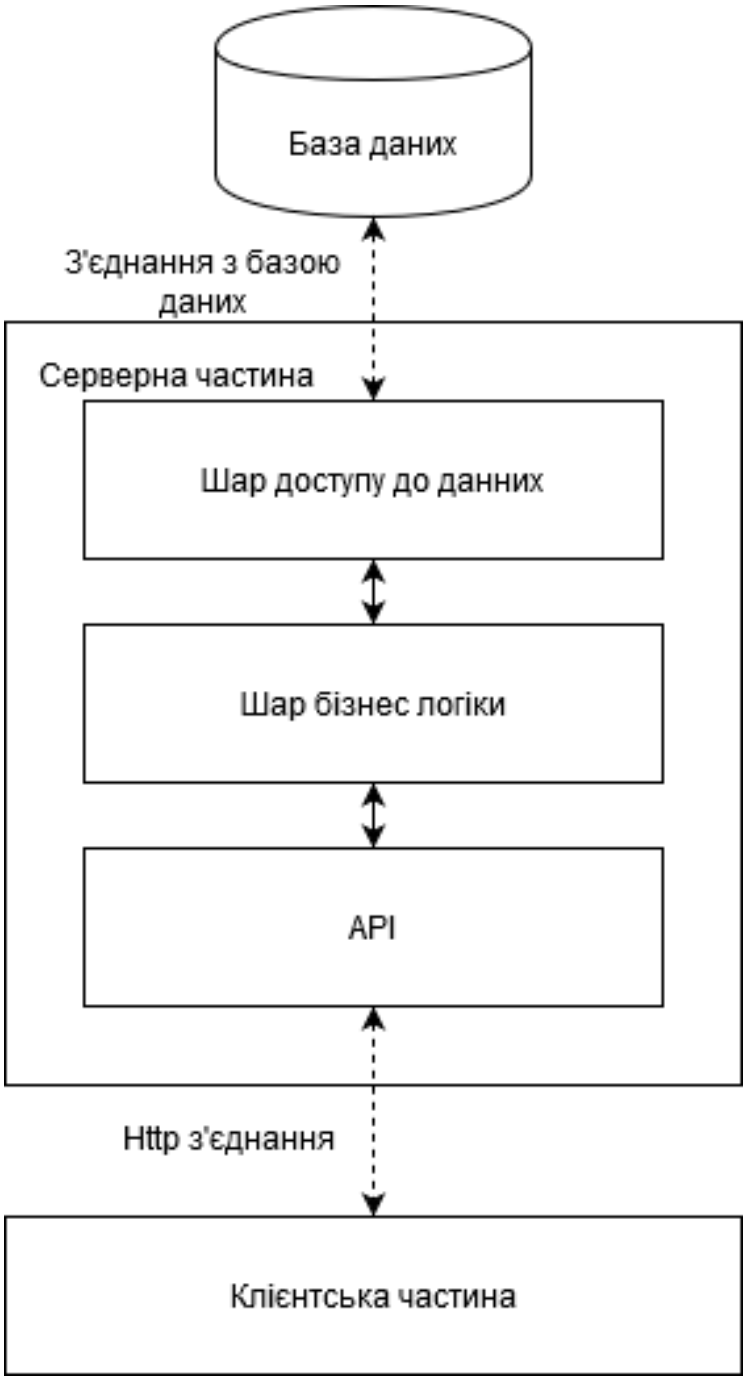
ДОДАТОК 3

МОБІЛЬНИЙ ДОДАТОК ДЛЯ УПРАВЛІННЯ СИСТЕМОЮ СПІЛЬНИХ ПОЇЗДОК

СТРУКТУРНА СХЕМА СИСТЕМИ

Аркушів 1

Київ – 2020



					ІАЛЦ 467800.006 Д3	
Зм.	*Арк.*	*Прізвище*	*Підпис*	*Дата*		
Розроб.		Гук М.О.			*Мобільний додаток для управління системою спільних поїздок* Структурна схема системи	*Арк.* 2 / *Аркушів*
Перевірив						
Н. кон.						*НТУУ «КПІ ім. Ігоря Сікорського», ФІОТ, ІП-62*
Затв.						