

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра автоматизації та управління в технічних системах**

«До захисту допущено»

Завідувач кафедри

_____ О.І. Ролік

«__» _____ 2019 р.

**Дипломний проект
на здобуття ступеня бакалавра
з напрямку підготовки 6.050103 «Програмна інженерія»
на тему: «Мобільне застосування для обслуговування потреб у галузі таксі»**

Виконав:

студент IV курсу, групи ІТ-51 Подосьонов Ігор Євгенович _____

Керівник:

Асистент, Шинкевич Микола Костянтинович _____

Рецензент:

Посада, науковий ступінь, вчене звання Прізвище, ініціали _____

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

Студент _____

Київ – 2019 рік

Пояснювальна записка
до дипломного проекту
на тему: «Мобільне застосування для
обслуговування потреб у галузі таксі»

Київ – 2019 рік

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Напрямок підготовки – 6.050103 «Програмна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ О.І. Ролік

«__» _____ 2019 р.

ЗАВДАННЯ

на дипломний проект студенту

Подосьонову Ігорю Євгеновичу

1. Тема проекту «Мобільне застосування для обслуговування потреб у галузі таксі», керівник проекту Шинкевич Микола Костянтинович, інженер 1-ї категорії кафедри АУТС,, затверджені наказом по університету від «__» _____ 2019 р. № _____

2. Термін подання студентом проекту 18.05.2019

3. Вихідні дані до проекту

4. Зміст пояснювальної записки

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

7. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка

Студент

І.Є. Подосьонов

Керівник проекту

М.К. Шинкевич

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЕКТУ

[illegible]

[illegible]

АНОТАЦІЯ

Пояснювальна записка дипломного проекту складається з п'яти розділів, містить 7 таблиць, 1 додаток та 15 джерел – загалом 62 сторінки.

Об'єкт дослідження: Інструменти розробки, тестування та прототипування мобільних застосунків. Розробка мобільних застосунків

Мета дипломного проекту: створити мобільне застосування для задоволення потреб у галузі таксі.

У першому розділі було виконано постановку задачі.

У другому розділі було проведено аналіз існуючих рішень, їх переваги та недоліки.

У третьому розділі була наведена теоретична частина у якій розглядалися операційні системи IOS та Android.

У четвертому розділі було проведено опис основних програмних засобів.

У п'ятому розділі було спроектовано та реалізовано мобільне застосування у галузі потреб таксі.

КЛЮЧОВІ СЛОВА: МОБІЛЬНЕ ЗАСТОСУВАННЯ, IOS, ГАЛУЗЬ ПОТРЕБ ТАКСІ

ABSTRACT

The explanatory note of the diploma project consists of five sections, contains 7 tables, 1 application and 15 sources - a total of 62 pages.

The object of study: single page applications used for corporate correspondence.

The aim of the diploma project: create a mobile application to meet the needs of the taxi industry.

In the first section, the task was performed.

In the second section, we analyzed the EXISTING solutions, their advantages and disadvantages.

The third section outlined the theoretical part in which IOS and Android operating systems were considered.

In the fourth section, a description of the main software tools was provided.

In the fifth section, the mobile application was designed and implemented.

KEYWORDS: MOBILE APPLICATION, IOS, TAXI INDUSTRY NEEDS

ЗМІСТ

Зміст.....	11
ПЕРЕЛІК УМОВНИХ ТЕРМІНІВ, ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ	13
Вступ.....	14
1. ПОСТАНОВКА ЗАДАЧІ.....	15
2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ.....	16
3. ТЕОРЕТИЧНА ЧАСТИНА.....	19
3.1. Мобільна операційна система IOS та її переваги над Android	19
3.2. Основи програмування IOS	21
4. ОПИС ОСНОВНИХ ПРОГРАМНИХ ЗАСОБІВ.....	23
4.1. SWIFT	23
4.1.1. Основні концепції мови програмування SWIFT.....	24
4.1.2. Порівняння з мовою програмування Objective-c	25
4.2. Firebase.....	27
4.3. XCode	29
4.3.1. CocoaPods	30
4.3.2. Storyboard	32
4.3.3. Simulator	34
4.4. Sketch	36
4.5. Apple MapKit та CoreLocatoін	37
5. ПРОЕКТУВАННЯ І РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУВАННЯ ..	41
5.1. Проектування прототипу графічного інтерфейсу	41
5.2. Архітектура застосування.....	44
5.3. Автентифікація користувача	47
5.3.1. Створення облікового запису.....	48
5.4. Відображення місцезнаходження	51
5.4.1. Відображення пасажирів	52

					IT51.017БАК.009 ПЗ			
Вим	Лист	№ докум.	Підпис	Дата	Мобільне застосування для обслуговування потреб у галузі таксі	Лім.	Лист	Листів
Розробив		Подосьонов І.Є.						
Перевірів		Шинкевич М.К.						
Реценз.						КПІ ім. Ігоря Сікорського ФІОТ ІТ-51		
Н. Контр.								
Затвердив		Шинкевич М.К.						

5.4.2. Відображення водіїв.....	54
5.5. Здійснення поїздки	55
Висновок	58
Список використаної літератури	59
ДОДАТОК А. СХЕМА БАЗИ ДАНИХ.....	61

ПЕРЕЛІК УМОВНИХ ТЕРМІНІВ, ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

UX – user experience

UI – user interface

ФП – функціональне програмування

BaaS – Backend as a Service

IDE – integrated development environment

MVC – model view controller

UID – unique identifier

					IT51.017БАК.009 ПЗ	Лист
Ізм.	Лист	№ докум.	Підпис	Дата		13

ВСТУП

Більшість людей у розвиненому світі має доступ до мобільного інтернету, все більше спостерігається тенденція зв'язана зі зменшенням долі відвідувачів веб-сайтів через десктопне обладнання, а доля мобільних пристроїв зростає. Також фактор того, що через веб-браузер неможливо використати деякі найважливіші функції смартфона збільшує потребу у розробці мобільних застосунків замість звичних веб-сайтів.

Перевагами мобільного застосування над веб-сайтом є тісна інтеграція з операційною системою, більш швидкий інтерфейс, більше можливостей для доступу до пристрою. Але складність пошуку спеціалістів для розробки мобільних застосунків та складності публікації мобільних застосунків змушують застосовувати їх тільки у випадках коли переваги є більш значними аніж недоліки.

Одним із таких випадків є галузь таксі. Мобільне застосування спрощує процес замовлення, його обробки та зменшує витрати на диспетчерів та обслуговуючий персонал, а також використовує можливості геопозиціонування мобільного пристрою.

1. ПОСТАНОВКА ЗАДАЧІ

Основною задачею дипломного проекту є дослідження засобів для дизайну та розробки мобільних застосунків та створення мобільного застосування для задоволення потреб у галузі таксі. Цю задачу можна поділити на наступні підзадачі:

1. Дослідити середу розробки Xcode. Провести дослідження основних програмних засобів та обрати ті, що задовольняють потреби розробки мобільних застосунків.

2. Спроекувати прототип графічного інтерфейсу застосування.

3. Визначити архітектуру мобільного додатку, обрати спосіб збереження даних та засоби клієнт-серверної взаємодії.

4. Реалізувати мобільне застосування за допомогою обраних програмних засобів та на основі створеного прототипу графічного інтерфейсу.

2. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

На даний момент на ринку існують наступні основні гравці у сфері мобільних додатків для замовлення таксі. Кожне застосування має свої недоліки та переваги, що будуть наведені нижче.

1. Uber

Робочий процес застосування Uber:

- Користувач відкриває застосування: Користувач вказує пункт призначення в поле «Куди?» У верхній частині екрану, вибирає тип поїздки в залежності від часу очікування, місткості автомобіля і вартості, підтверджує місце посадки і оформляє замовлення;
- Водій приймає замовлення: Водій, який знаходиться недалеко від місця посадки, бачить і приймає замовлення. Користувачу буде повідомлено, коли до прибуття водія залишається близько хвилини;
- Водій прибуває до місця посадки: Водій і користувач звіряють імена один одного і пункт призначення. Після цього водій починає поїздку;
- Водій доставляє користувача в пункт призначення: застосування показує докладний маршрут, тому водій може повністю зосередитися на дорозі, а користувач - насолодитися поїздкою;
- Водій і користувач ставлять один одному оцінки: Після закінчення поїздки водій і користувач можуть поставити один одному оцінку - від 1 до 5 зірок. У додатку для користувачів можна написати комплімент водієві, а в деяких містах - залишити водієві невелику доплату в знак подяки.

Перевагами додатку є:

- можливість оплати у застосуванні за допомогою картки;
- можливість слідкувати як автомобіль їде на замовлення;

- інформація про автомобіль, його колір, номерні знаки та можливість подзвонити водію при замовлені;
- поширеність застосування по всьому світу;
- можливість оцінки водіїв за різними критеріями, від вправності керування транспортним засобом до чистоти салону.

Недоліками додатку є:

- якість перевірки інформації про водіїв залишає бажати кращого;
- співробітники Uber мають доступ до особистих даних користувачів, що означає можливість витіку даних;
- застосування несе серйозну загрозу застарілим службам таксі, що працювали і ділили ринок між собою довгі роки.

2. Uklon

Перевагами додатку є:

- Можливість анонімно користуватися сервісом;
- Збереження маршрутів, знижки;
- Можливість швидкого виклику машини з користувацької історії замовлень.

Недоліками додатку є:

- Сервіс працює лише у Києві

3. IQTaxi

Завдяки вбудованим до автомобілів сервісу передавачам можна слідкувати за статистикою автомобілів а також точно знати кількість машин що знаходяться поряд з вами. Ще одною перевагою є можливість здійснити замовлення через веб-сайт компанії.

Під час очікування автомобіля можна як слідкувати за його положенням так і переглядати цікаві факти у мобільному застосуванні.

4. KwikPik

					IT51.017БАК.009 ПЗ	Лист
						17
Ізм.	Лист	№ докум.	Підпис	Дата		

Даний сервіс працює не тільки в Україні а також за кордоном – в США, Росії і навіть Казахстані. Головною його особливістю є можливість обирати між водіями що погодилися виконати ваше замовлення, а завдяки гнучкій системі оцінювання водіїв та можливості переглядати оцінки пасажир може розраховувати на гарну якість обслуговування. Недоліком такого підходу є те що така політика сервісу не завжди до вподоби водіям, що може відштовхнути їх від використання сервісу, коли саме водії створюють потребу користувачів у цьому сервісі.

5. Taxi Select

Універсальне застосування, створено для тих кому терміново потрібно замовити таксі, працює майже по всьому світу. За допомогою одного застосування можливо замовити таксі де завгодно. Застосування визначить ваше місцезнаходження та покаже список можливих перевізників.

Основними перевагами сервісу Taxi Select є:

- Можливість користуватися додатком без підключення до інтернету;
- Сервіс в той же час є довідником;
- Можливість викликати машину як по телефону, так і через інтернет;

3. ТЕОРЕТИЧНА ЧАСТИНА

3.1.Мобільна операційна система IOS та її переваги над Android

Операційна система IOS – мобільна операційна система розробником якої є корпорація Apple. Розроблена для використання на смартфонах, планшетах та носимих пристроях від компанії Apple. Являється одним із головних конкурентів операційної системи з відкритим кодом Android.

Основними перевагами IOS над операційною системою Android є високий рівень захищеності, швидкодія, більш інтуїтивний та зручний інтерфейс, що в свою чергу покращує рівень задоволення від користувацького досвіду (User Experience). Також Apple пропонує оновлення операційної системи для усіх девайсів, що спроможні працювати з новим софтом. Це означає що девайси під управлінням операційної системи IOS залишаються актуальними довший час.

Давайте розглянемо ці переваги більш детально:

1. Інтерфейс – через те що Android є відкритою операційною системою багато виробників смартфонів, що реалізують свою продукцію з встановленим Android кастомізують інтерфейс під свої уявлення, встановлюючи так звані «оболонки» і через це немає єдиного консенсусу як виглядає інтерфейс операційної системи. У випадку з IOS існують чіткі рекомендації з дизайну, завдяки чому користувачі не мають проблем із взаємодією з новими застосуваннями завдяки схожому за логікою інтерфейсом;
2. Швидкодія – завдяки тому що операційна система IOS розроблена під відносно невелику кількість девайсів це дозволяє оптимізувати її під конкретні апаратні пристрої. Завдяки тісній інтеграції хардверної та софтверної частини кінцеві користувачі отримують кращі показники у швидкодії операційної системи;

3. Захищеність – хакери рідше пишуть шкідливі застосування для IOS пристроїв через огляд усіх програм корпорацією Apple та перевірку ідентичності видавців програм. Про те, що Android-пристрої є менш захищеними, свідчить те що ціна за знайдену вразливість нульового дня (тобто невідому до цього) у Android є значно меншою ніж за знайдену вразливість у IOS.

Виходячи з даних що представлені нижче (таблиця 3.1), можна зробити висновок що операційна система IOS завдяки високим стандартам якості програмного забезпечення а також уваги до користувацького досвіду має перевагу над операційною системою Android.

Таблиця 3.1 – Порівняння операційних систем IOS та Android

	IOS	Android
Джерело	Закрите, з відкритими компонентами	Відкрите
Сімейство ОС	OS X, UNIX	Linux
Розробник	Apple Inc.	Google
Безпека	Загрози безпеці рідкісні, тому що iOS має закрите ядро, а завантаження додатків з App Store ускладнене.	Виробники, як правило, відстають у висуванні оновлень. Таким чином, у будь-який час переважна більшість Android-пристроїв використовують застаріле програмне забезпечення для ОС.
Інтерфейс	Тачскрін. Нормалізований інтерфейс в усій системі	Тачскрін. Через можливості кастомізації інтерфейс не нормалізований, може бути різним

Продовження таблиці 3.1

Швидкодія	Висока. Завдяки взаємодії хардверної та софтверної частин.	Залежить від девайсу.
-----------	--	-----------------------

3.2.Основи програмування IOS

Навички та інструменти потрібні для створення власних застосування для операційної системи IOS:

- xcode: інтегроване середовище розробки яке використовується для створення застосувань. Він має редактор коду, організатор проекту, компілятор, відгадчик і багато інших важливих інструментів для створення застосувань;
- swift: потужна мова програмування, яку ви використовуєте для кодування застосувань iOS, macOS, tvOS і watchOS. Прийшов на заміну Objective-C, Swift пакує багато функцій, які допомагають продуктивно кодувати;
- створення інтерфейсів: кожна програма потребує інтерфейсу користувача (UI). Інтерфейси користувача складаються з кнопок, переглядів, навігації, міток, зображень, елементів керування тощо;
- кодування логіки: код Swift керує тим, що відбувається у застосуванні та в який час. Це називається логікою, вона має вирішальне значення для будь-якої програми;
- архітектура додатків: Написання чіткого, розширюваного і підтримуваного коду не менш важливе, ніж написання коду, який працює. Архітектура додатків є складною темою, і для вивчення рішення, яке найкраще працює в різних сценаріях, потрібен час.

Розробники можуть зобразити архітектуру IOS як чотири шари абстракції:

- cocoa Touch: підтримує основну інфраструктуру додатків і забезпечує ключові рамки додатків, такі як push-сповіщення, багатозадачність і сенсорний вхід;
- media: дозволяє програмі надавати аудіо, відео та графічні можливості;
- core Services: основні системні послуги, такі як Core Foundation і Foundation Framework. Цей шар також підтримує такі функції, як геопозиціонування та мережні служби;
- core OS: надає такі послуги, як безпека, локальна автентифікація та інші фреймворки.

Apple рекомендує розробникам писати код на найвищому шарі застосування і використовувати фреймворки глибокого рівня на більш високих рівнях лише коли це необхідно.

Щоб створити успішне застосування, розробники повинні ставити у пріоритет дизайн інтерфейсу(UI) та користувацького досвіду(UX) на кожному кроці.

Користувачі повинні мати можливість переглядати, розуміти та взаємодіяти з вмістом без постійного вдавання до прокручування або масштабування - отже, текст повинен бути розбірливим. Користувачі також повинні мати можливість використовувати функції сенсора та виконувати завдання з найменшими можливими кроками.

4. ОПИС ОСНОВНИХ ПРОГРАМНИХ ЗАСОБІВ

4.1.SWIFT

Swift – відносно нова мова програмування для розробки застосувань для операційних систем OS X та IOS. Swift використовує сучасні функції та патерни безпечного програмування, увібрав в себе найкраще від сучасних мов програмування та спроектований таким чином щоб мати можливість кодувати не лише прості застосування але й цілі операційні системи. Код на Swift скомпільований і оптимізований, для того щоб отримувати найбільшу віддачу від сучасного обладнання.

Згідно з даними представленими PYPL ^[1] Swift зрівнявся у популярності з Objective-c всього за рік (рисунок 4.1) завдяки постійним оновленням та підтримці від Apple.

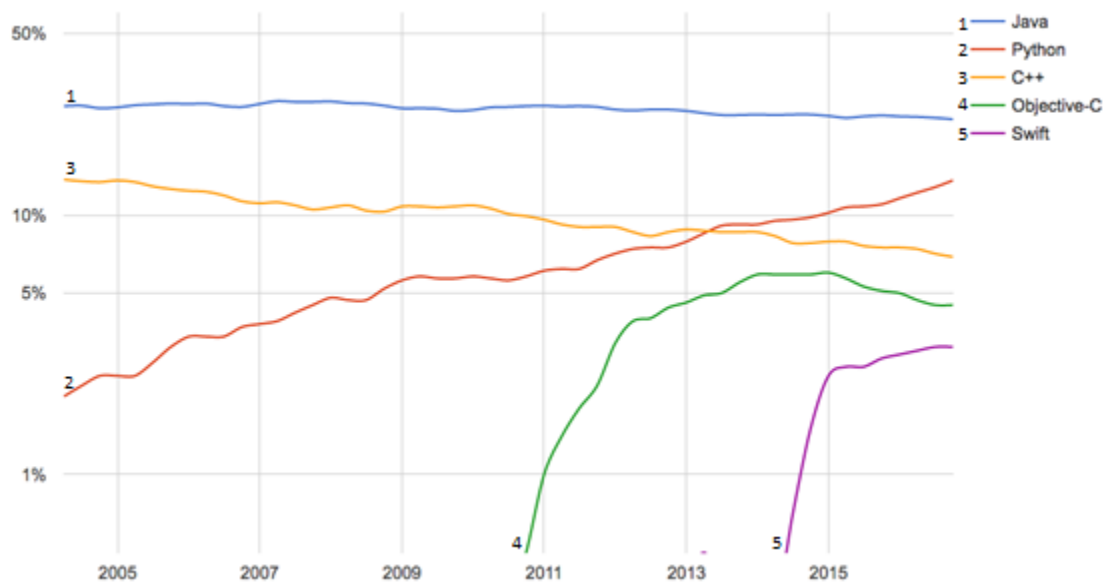


Рисунок 4.1 – Популярність мов програмування згідно індексу PYPL [1]

Тому розробка застосування на мові Swift є цілком виправданою і підходить для поставленої задачі.

4.1.1. Основні концепції мови програмування SWIFT

Функціональне програмування (ФП) – програмування за допомогою функцій, кодування в більш математичному стилі, коли завжди є вхідні і вихідні дані, створюючи з них потік, через який проходять дані.

Основні стовпи ФП:

- Незмінність вхідних даних;
- Зміна рівню абстракції за рахунок функцій вищих порядків.

Розглянемо їх більш детально. У мові Swift є два типи даних: посилання (reference type) та значення (value type). Тип посилання лише зберігає у собі посилання на об'єкт або змінну, в той час коли структура, тобто тип значення, немає ніяких залежностей і усі данні ізольовані. Зміна рівня абстракції за рахунок функцій вищих порядків означає що можна приймати функцію у середині іншої функції та використовувати її як аргумент. Також можна поводитись з функціями як із звичайними змінними, тобто приймати як аргумент у середині функції, повертати у якості результату іншої функції. У мові Swift реалізовані базові функції ФП (рисунок 4.2), такі як «map», «filter», «reduce».

```
1 let numbers = [1,2,3]
2 let squares = numbers.map { $0 * $0 }
3 let odds = numbers.filter { $0 * 2 == 1 }
4 let sum = numbers.reduce(0) { $0 + $1 }
5
6 let act1SceneCount = romeoAndJuliet.reduce(0) { count, title in
7     title.hasPrefix("Act 1") ? count + 1 : count
8 }
```

Рисунок 4.2 – Приклад використання базових функцій ФП

Також у Swift є опціональні типи даних, так звані опціонали (optionals) – зручний механізм для обробки ситуацій, коли значення змінної може бути відсутнім.

4.1.2. Порівняння з мовою програмування Objective-C

Спочатку створена на початку 80-х, Objective-C вперше отримала свою популярність як основна мова для операційної системи NEXTSTEP. Коли NEXTSTEP, творець NeXT, був об'єднаний з Apple в 1996 році, NEXTSTEP став основою нової ОС Apple, зміцнивши Objective-C як основну мову для систем Apple на довгі роки.

Довгий час Objective-C був єдиною основною мовою програмування для створення OS X і iOS-додатків. У своїй основі Objective-C є розширенням мови C з доданими об'єктно-орієнтованими функціями та динамічною середовищем виконання.

Нова мова програмування Swift від Apple найкраще описана як Objective-C без C. Вперше була представлена Apple на WWDC 2014. Це мова кодування Mac OS і IOS, яка також підтримує tvOS і watchOS, саме тому що мова Swift створена бути заміною застарілому Objective-C порівняння цих мов має сенс.

Swift поєднує в собі як сильні сторони C, так і Objective-C, не маючи прямого вбудованої C-сумісності і всіх обмежень, які йдуть разом з ним. Сучасні набори функцій та безпечні шаблони програмування є основними перевагами Swift над Objective-C, що дозволяє забезпечувати гнучкість і швидкість розробки застосувань. Завдяки підтримці популярних фреймворків, таких як Cocoa та Cocoa Touch, Swift повністю змінює наше розуміння того, як виготовляти програмне забезпечення для продуктів Apple.

Переваги та недоліки Objective-C та Swift повинні розглядатися до того, як зробити остаточний вибір. Однак, зрештою, обидві мови є різними і є корисними в різних обставинах, тому важливо вибрати мову на основі кожного проекту.

У Objective-C написано неймовірно велика кількість сторонніх бібліотек, які зосереджують увагу як на інтерфейсі користувача, так і на логіці програми (бази даних, мережа, багатопоточність, аналіз даних тощо).

Swift, з іншого боку, не має стільки бібліотек третіх сторін, однак, це лише питання часу, оскільки це молода мова що розвивається. Але за допомогою Swift можна викликати методи Objective-C бібліотеки, що вирішує цю проблему.

Синтаксис Swift є схожим на сучасні мови програмування такі як C#, Java, Rust, що дозволяє писати код зручніше та швидше. В той час коли Objective-c має у собі багато застарілих практик кодування.

Swift далеко попереду Objective-C, коли справа доходить до стабільності самого коду. Завдяки своїй суворій типізації і способам обробки помилок, ви можете легко написати код, який буде враховувати всі потенційні помилки.

Виходячи з того, що мова Swift має більше переваг ніж Objective-c, що можна прослідкувати у приведеній нижче таблиці (таблиця 4.1), можна зробити висновок що її використання є доцільним. Завдяки сучасному синтаксису та простоті у поєднанні з потужністю мова Swift є гарним вибором для розробки мобільного додатку для операційної системи IOS.

Таблиця 4.1 – Плюси та мінуси мов програмування Swift та Objective-c

	Swift	Objective-c
Відкрите джерело	+	-
Доступна велика кількість бібліотек та фреймворків	+/-	+
Сучасний синтаксис (на подоби C#, Java, Go)	+	-
Простий у вивченні	+	-
Стабільність та надійність перевірена часом	-	+
Активна підтримка та просування від Apple	+	-

4.2. Firebase

Firebase – це платформа для розробки мобільних і веб-додатків, яка надає розробникам безліч інструментів і послуг, які допомагають їм розробляти високоякісні застосування, збільшувати базу користувачів і заробляти більше прибутку. Після придбання компанією Google у 2014 році Firebase швидко перетворився на багатофункціонального гіганта для розробки мобільних та веб застосувань.

Firebase пропонує ряд сервісів для розробників мобільних та веб застосувань (рисунок 4.3) які можна поділити на дві основні групи: розробка та тестування вашого додатку, а також розвиток і залучення аудиторії.



Рисунок 4.3 – Сервіси Firebase [2]

Розглянемо сервіси які буде використано для розробки мобільного застосування.

База даних реального часу (Realtime Database) – більшість баз даних вимагають здійснення викликів HTTP для отримання та синхронізації даних, а

також передають дані лише тоді, коли ви робите запит. Коли ви підключаєте свою програму до Firebase, замість HTTP підключення ви підключаєтесь через WebSocket.

WebSocket набагато швидше HTTP підключення, а також не потребує постійних викликів, оскільки одного підключення до сокету достатньо. Усі дані автоматично синхронізуються за допомогою єдиного веб сокету настільки швидко, наскільки швидко працює мережа користувача.

Firebase відсилає нові дані одразу після оновлення завдяки чому усі підключені до бази даних клієнти отримують нові дані майже миттєво.

База даних реального часу є NoSQL базою даних, що за своєю суттю є одним великим JSON об'єктом що дозволяє вам зберігати та синхронізувати дані між користувачами застосування у реальному часі.

Автентифікація (Authentication) – перевірка автентичності Firebase забезпечує серверні служби, прості у використанні SDK і готові бібліотеки інтерфейсу користувача для автентифікації користувачів вашої програми.

Як правило, потрібно декілька місяців, щоб створити власну систему автентифікації. І навіть після цього потрібно буде зберегти спеціальну команду для підтримки цієї системи. Але якщо ви використовуєте Firebase, ви можете налаштувати всю систему декількома рядками коду, які будуть обробляти все для вас, включаючи складні операції, такі як злиття акаунтів. Також Firebase Authentication підтримує стандарт OAuth2.

Ви можете перевірити автентичність користувачів програми за допомогою таких методів:

- Електронна пошта та пароль;
- Номери телефонів;
- Google;
- Facebook;
- Twitter.

4.3.XCode

Xcode – інтегрована середовище розробки (IDE) тісно зв'язана з фреймворками Cocoa та Cocoa Touch, що дозволяє створювати застосування для пристроїв під управлінням операційних систем iOS та OS X сторонніми розробниками. IDE – це програма, або їх набір, що включає в себе інструменти для різних аспектів створення програмного забезпечення. Xcode, зокрема, об'єднує редактор коду, компілятор, відладчик, конструктор інтерфейсів і симулятор (який технічно є окремою програмою, запущеною з Xcode).

Xcode має основні вікна робочої області (рисунки 4.4). Вони знаходяться в напрямку за годинниковою стрілкою зліва:

- область навігації;
- область редактору;
- область інструментів;
- область відладки (debug).

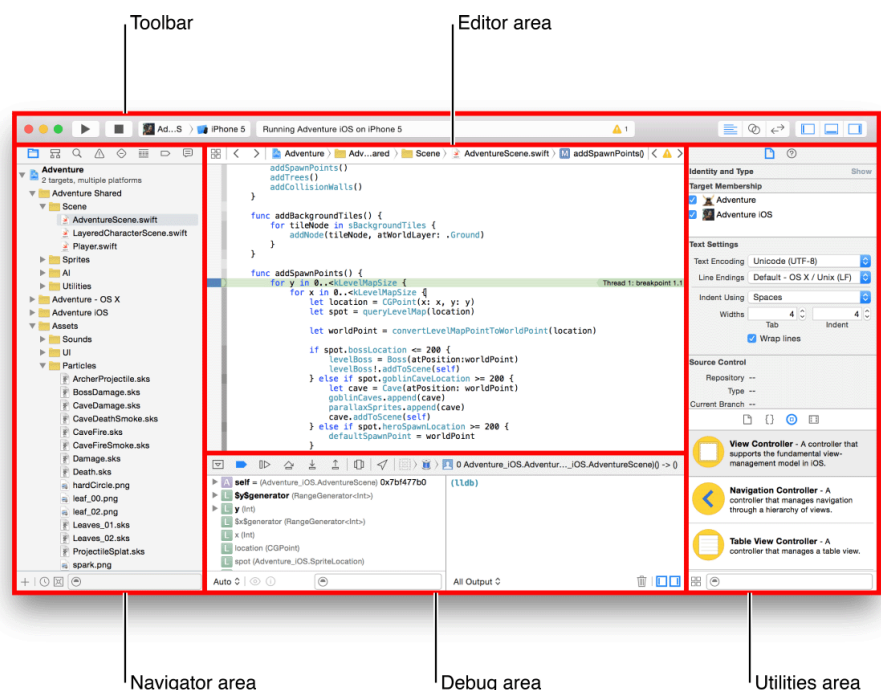


Рисунок 4.4 – Вікно робочої області Xcode [15]

4.3.1. CocoaPods

CocoaPods – це дуже потужний інструмент, який потрібен кожному iOS розробнику, це спосіб керування залежностями, що розробники використовують при створенні застосунків. З його допомогою можливо легко, швидко і просто підключити різні бібліотеки та інструменти, які значно покращують стан при розробці програми.

При створенні мобільного додатку не доцільно писати кожен частину коду власноруч, для цього на допомогу нам приходять накладки інших розробників. Як правило вихідний код додають у корінь проекту. Цей підхід має суттєву кількість недоліків:

- складно слідкувати за версіями бібліотек та їх зв'язком між собою;
- немає загального місця де можна переглянути список усіх доступних бібліотек;
- необхідно завжди пам'ятати про те що вихідний код таких бібліотек потрібно обновляти;
- інколи є потреба зберігати проект у репозиторії без зайвих файлів бібліотек. А той хто завантажує проект для роботи з ним має власноруч додавати ці файли до проекту.

Засоби керування залежностями CocoaPods допомагають вирішити ці питання. CocoaPods допоможе з залежностями між бібліотеками, які ви використовуєте, створить і буде підтримувати структуру проекту в належному вигляді.

Для того щоб створити файл зі списком залежностей потрібно використовуючи термінал перейти у директорії проекту та запустити команду «pod init». Ця команда створить спеціальний файл під назвою «Podfile» що буде зберігати у собі список потрібних залежностей для вашого проекту.

Для того щоб встановити нові залежності вам потрібно описати їх у файлі файлі зі списком залежностей та запустити команду «pod install» (рисунок 4.5).

```

MacBook-Pro-Igor:Flash-Chat-iOS11 igor$ pod install
Analyzing dependencies
Downloading dependencies
Using ChameleonFramework (2.1.0)
Using Firebase (5.4.1)
Using FirebaseAnalytics (5.0.1)
Using FirebaseAuth (5.0.2)
Using FirebaseCore (5.0.6)
Using FirebaseDatabase (5.0.2)
Using FirebaseInstanceID (3.1.1)
Using GTMSessionFetcher (1.2.0)
Using GoogleToolboxForMac (2.1.4)
Using SVProgressHUD (2.2.5)
Using leveldb-library (1.20)
Using nanopb (0.3.8)
Generating Pods project
Integrating client project
Sending stats
Pod installation complete! There are 5 dependencies from the Podfile and 12 total pods installed.
MacBook-Pro-Igor:Flash-Chat-iOS11 igor$

```

Рисунок 4.5 – Результат виконання команди «pod install»

Для пошуку потрібних фреймворків можна використовувати ресурс cocoapods.org, він має функціонал пошуку бібліотек (рисунок 4.6) і документацію для кожної з них.

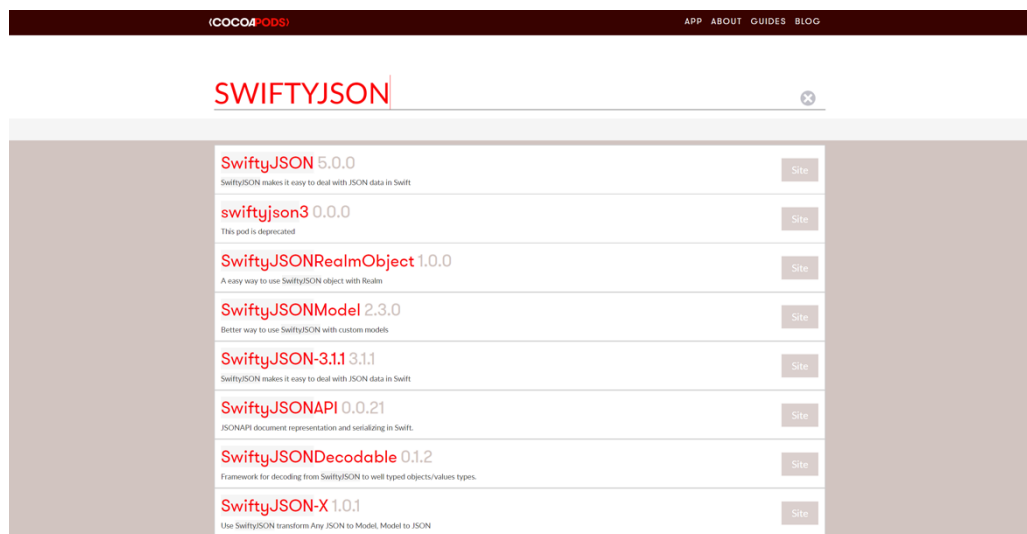


Рисунок 4.6 – Пошук бібліотек на ресурсі cocoapods.org

Після виконання налаштування залежностей створюється нова заготовка проекту. Тепер працювати з проектом потрібно через згенерований файл з розширенням «xcworkspace». З цього моменту відкривати проект можна тільки

через цей файл, тому що він містить в собі потрібні залежності, так звані «pods».

4.3.2. Storyboard

Storyboard - це корисна функція, вперше представлена в iOS 5, яка економить час на побудові інтерфейсів користувача для ваших програм. Storyboard дозволяє створювати декілька контролерів вигляду (View Controller) у межах одного файлу.

До Storyboard вам довелося б використовувати XIB-файли, і ви могли використовувати тільки один XIB-файл на контролер (UITableViewCell, UITableView або інші підтримувані типи UIView).

Завдяки цій функції ви можете візуально створити декілька контролерів вигляду та налаштувати зв'язки та переходи (segues) між ними (рисунк 4.7). Такий підхід є дуже зручним та візуально зрозумілим.

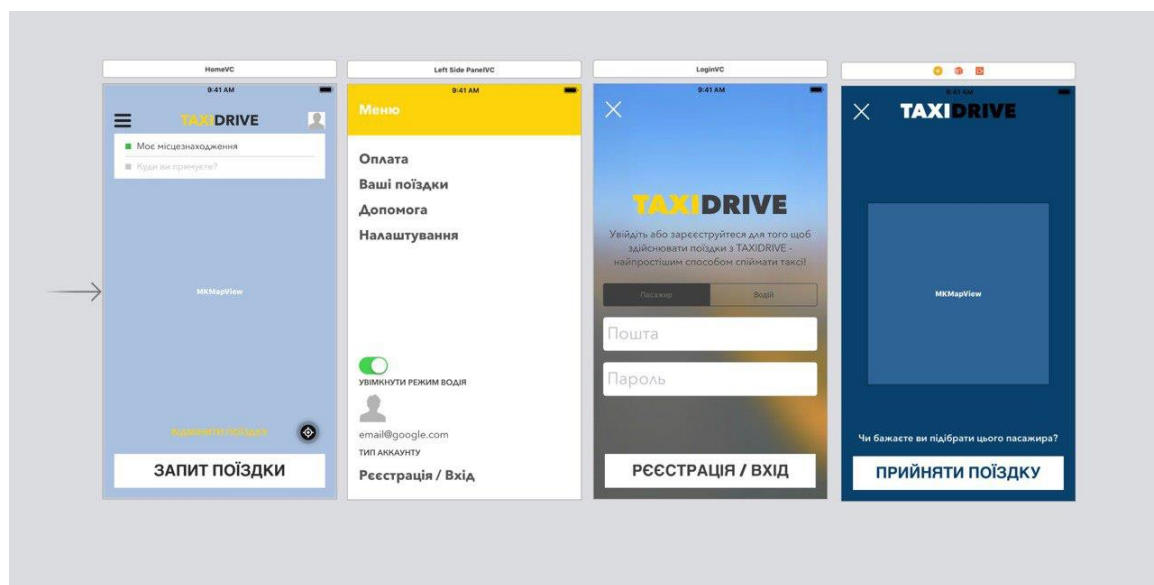


Рисунок 4.7 – Вигляд Storyboard зі створеними контролерами та зв'язками між ними

Storyboard має ряд переваг:

- можливість візуально викласти всі ваші контролери у так звані сцени (scenes) та описати зв'язки між ними;
- описати переходи між різними сценами. Ці переходи називаються «segues» і завдяки ним вам потрібно менше коду для обробки інтерфейсу користувача;
- спрощене застосування «Auto Layout», функції, яка дозволяє визначити математичні відношення між елементами визначаючи їх положення і розміри. Ця потужна функція спрощує роботу з пристроями із різним розміром екрану.

Існує багато типів переходів, так званих «segues». Однак не всі з них можна використовувати у всіх ситуаціях. Вибір переходу між двома конкретними контролерами залежить від структури інтерфейсу та бажаного ефекту. Деякі переходи мають сенс лише в деяких сценаріях. Існують наступні типи переходів:

- push segue – дозволяє переміщатися у обох напрямках стеку контролерів. Через цей контролер ви можете «підштовхнути» новий контролер поверх попереднього контролеру, або «викинути» контролер зі стеку для відкриття нижчого контролеру у стеку;
- present modally – також є одним з найбільш поширених типів переходів у iOS. Він показує новий контролер перегляду поверх попереднього. Тим не менш, модульний контролер перегляду не повинен бути частиною будь-якої ієрархії або стека екранів. Він просто розташований на верхньому контролері попереднього перегляду;
- rorup – показує новий контролер перегляду, обмежений у спливаючому вікні. Очевидно, що цей тип переходів має сенс лише в iPad. На пристроях з меншими екранами, як iPhone, цей перехід поводить як звичайний модальний перехід;

- show detail – особливий тип перегляду, який корисний для контролерів розділення перегляду.

4.3.3. Simulator

Симулятор дозволяє швидко прототипувати і тестувати збірки вашого додатка під час процесу розробки. Симулятор є частиною інструментів Xcode, Simulator працює як стандартне застосування та емулює мобільний пристрій під який ведеться розробка. Симулятор це інструмент попереднього тестування, який слід використовувати, перш ніж тестувати програму на власному пристрої.

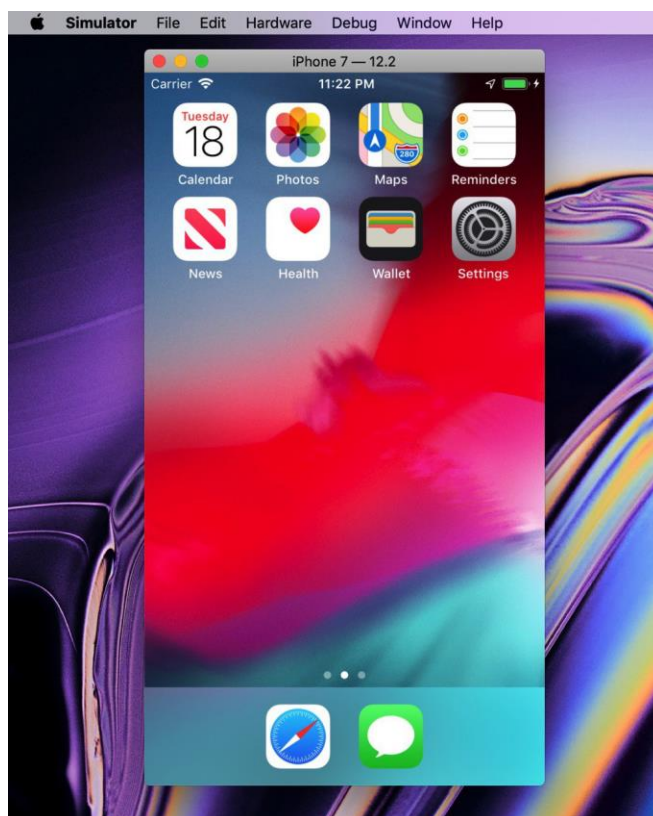


Рисунок 4.8 – Xcode Simulator у дії

Симулятор дозволяє імітувати пристрої з різними розмірами екрану та версіями ОС (рисунок 4.8). Кожна комбінація модельованого пристрою та версії програмного забезпечення вважається власним середовищем

моделювання, незалежним від інших, з його власними налаштуваннями та файлами. Ці параметри та файли існують на кожному пристрої, який ви тестуєте в середовищі моделювання.

Моделюючи роботу програми в Simulator, ви можете:

- знайти основні проблеми у вашому застосуванні під час розробки та раннього тестування;
- перевірити свою програму за допомогою інструментів розробника, доступних лише для симулятора.

Симулятор є корисним інструментом, але він не повинен бути єдиним способом перевірити програму. Оскільки симулятор - це програма, що працює на Mac, вона має доступ до ресурсів комп'ютера, включаючи процесор, пам'ять і мережеве з'єднання. Всі ці ресурси, швидше за все, будуть швидшими, ніж ті, які знайдені на мобільному пристрої. У результаті симулятор не є точним тестом продуктивності програми, використання пам'яті та швидкості роботи мережі. З цієї ж причини завжди перевіряйте продуктивність користувацького інтерфейсу вашої програми на пристрої. У симуляторі користувацький інтерфейс вашої програми може працювати швидше і більш плавно, ніж на пристрої. Для того щоб протестувати застосування безпосередньо на пристрої можна використовувати так званий «Sideloadіng», для цього лише треба підключити смартфон до вашого комп'ютера та у вікні вибору пристрою (рисунок 4.9) обрати свій пристрій.

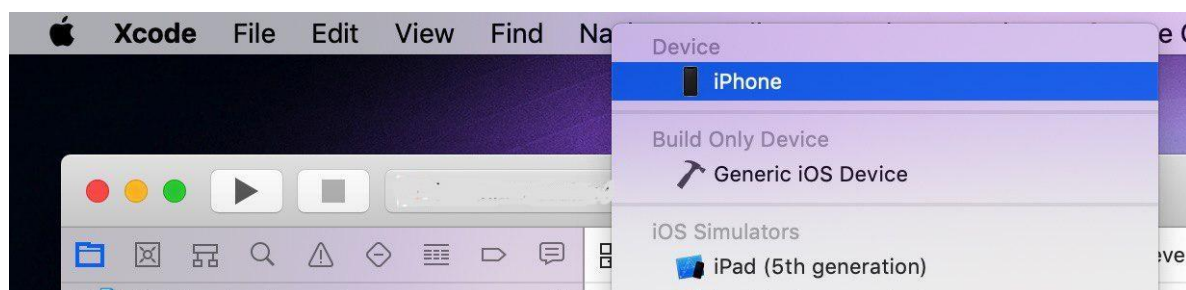


Рисунок 4.9 – Sideloadіng у Xcode

4.4.Sketch

Дизайн має важливе значення у мобільному застосуванні. І спершу, ніж створювати Storyboard потрібно створити макет, бо ранній перехід прямо в Xcode може закінчитися великою кількістю втраченого часу та написанням непотрібного коду.

Sketch – відмінний інструмент для створення логотипів і користувальницьких значків. Він базується на векторній графіці, тобто ви можете змінювати розмір без втрати якості. Оскільки Sketch включає в себе набір розмірів екранів iPhone і вбудований шаблон UIKit, ви також можете використовувати Sketch для створення макетів мобільних додатків. Крім того, Sketch досить простий, щоб розробник міг швидко навчитися його використовувати.

Шаблони – шаблон оформлення iOS UI Design (рисунок 4.10) це набір компонентів UIKit у векторному форматі. Він має панель вкладок, навігаційну панель, таблицю та багато іншого. Ви можете використовувати цей шаблон як базу для створення макетів.

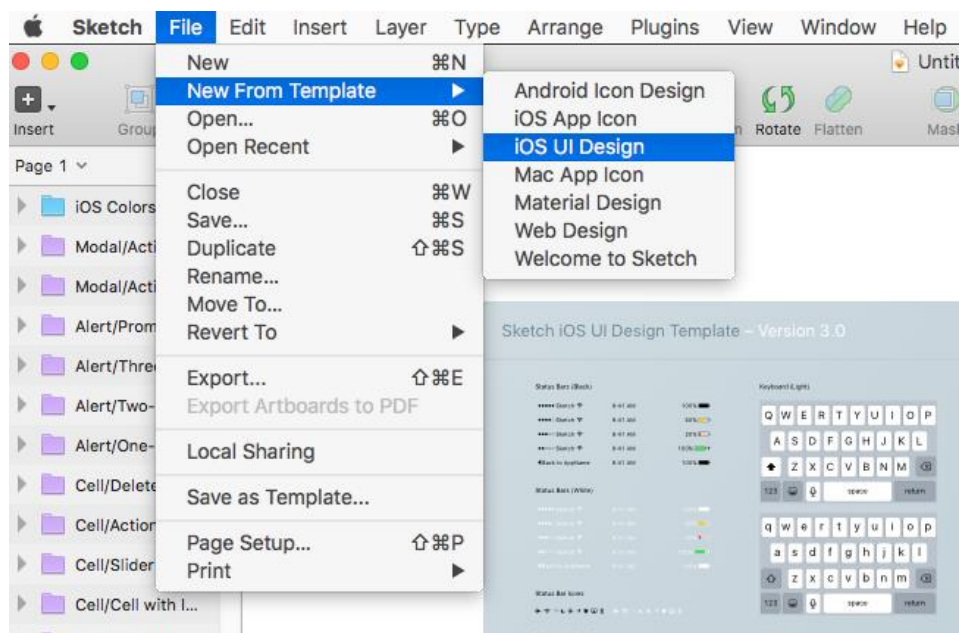


Рисунок 4.10 – Шаблон оформлення IOS UI Design [11]

Artboards – можна малювати будь-де на нескінченному полотні Sketch, але якщо створюєте макет застосування з кількома екранами, зручно використовувати використовувати монтажні області. Artboard організовує слої всередині фіксованого кадру. Коли ви експортуєте Artboard, всі шари всередині об'єднуються в один файл. Подумайте про них як про сцену контролерів переглядів у Storyboard.

Ви можете додати стільки монтажних областей, скільки хочете. Ескіз має розміри для iPhone, Apple Watch, і навіть iPad Pro. Якщо вам коли-небудь потрібно позбутися Artboard, просто натисніть назву і натисніть "Видалити".

Також у Sketch є можливість роботи зі слоями, векторними формами та іншими інструментами що властиві додаткам для роботи з векторною графікою.

4.5. Apple MapKit та CoreLocation

Mapkit – це фреймворк, що доступний на пристроях IOS, котрий дозволяє легко відображати супутникові та схематичні мапи, наносити на мапу місцезнаходження об'єктів, прокладати маршрути, а також наносити інші фігури поверх мапи [2].

CoreLocation – фреймворк що дозволяє визначити геопозицію пристрою з вказаною точністю, це дозволяє використовувати його максимально ефективно з точки зору економії заряду батареї мобільного пристрою, для його використання користувач застосування має дати дозвіл на використання своєї геопозиції. Для визначення геопозиції використовуються різні апаратні елементи мобільного пристрою, такі як Wi-Fi, GPS-модуль, Bluetooth-модуль, барометр, магнітометр [3].

Перед тим як використовувати дані про геопозицію користувача потрібно запросити дозвіл на використання сервісу визначення місцеположення. Для

виконання запиту на дозвіл потрібно викликати метод `requestWhenInUseAuthorization` для того щоб зчитувати дані при використанні застосування, або `requestAlwaysAuthorization` щоб отримувати дані коли застосування неактивне або знаходиться у фоновому режимі. Ще однією умовою для доступу до геопозиції користувача є спеціальний запис у файлі списку інформаційних властивостей застосування `Info.plist`. Цей файл є набором властивостей у форматі XML із структурою “ключ-значення”. У файлі `Info.plist` повинні бути вказані наступні властивості для доступу до геопозиції:

- “Privacy - Location When In Use Usage Description” для виклику методу `requestWhenInUseAuthorization`;
- “Privacy - Location Always and When In Use Usage Description” для виклику методу `requestAlwaysAuthorization`.

Використовуючи `MapKit` частина мапи, що відображається на екрані називається регіоном. Регіон визначається місцеположенням у центрі та розмахом області відображення. Всередині перегляду мапи можна створити анотацію із зазначенням точки інтересу. Широта та довгота точки відображення визначається за допомогою структури `CLLocationCoordinate2d`. Анотація створюється у вказаних координатах за допомогою класу `MKPointAnnotation` та додається у уявлення за допомогою методу `addAnnotation`.

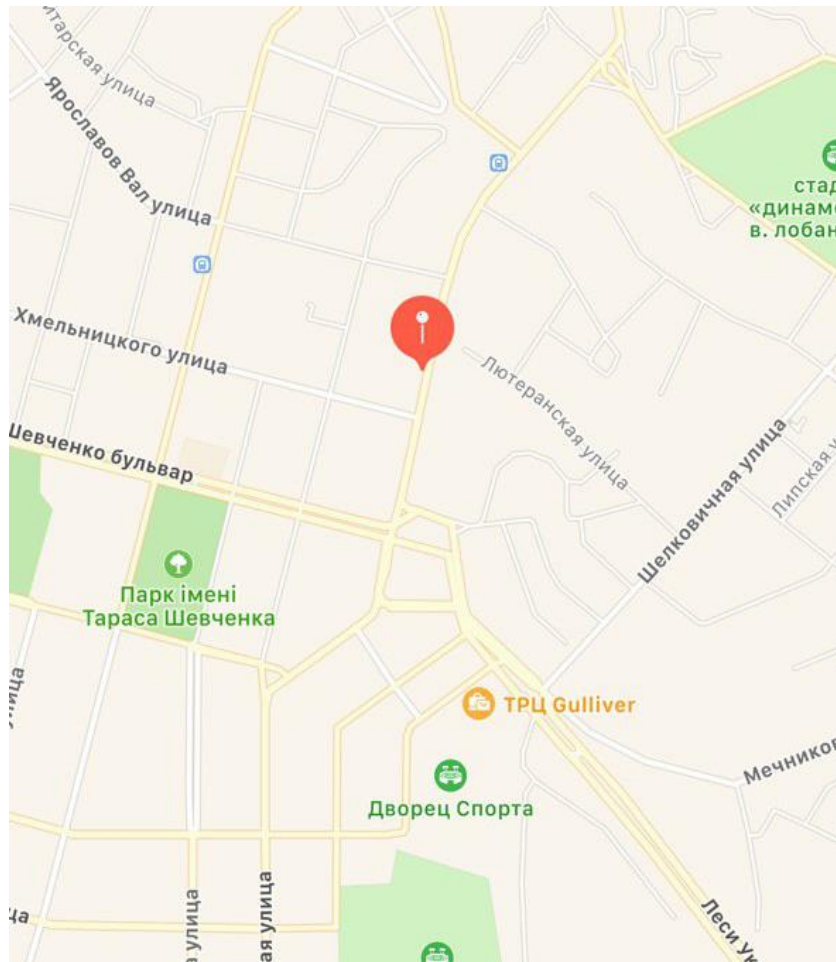


Рисунок 4.11 – MKMapView із анотацією

Для того щоб відобразити шлях між двома локаціями на мапі нам потрібні координати обох локацій. Отримавши координати обох місць можна створити об'єкт MKPolyline що відображає шлях на мапі, а потім помістити його на MKMapView за допомогою MKPolylineRender.

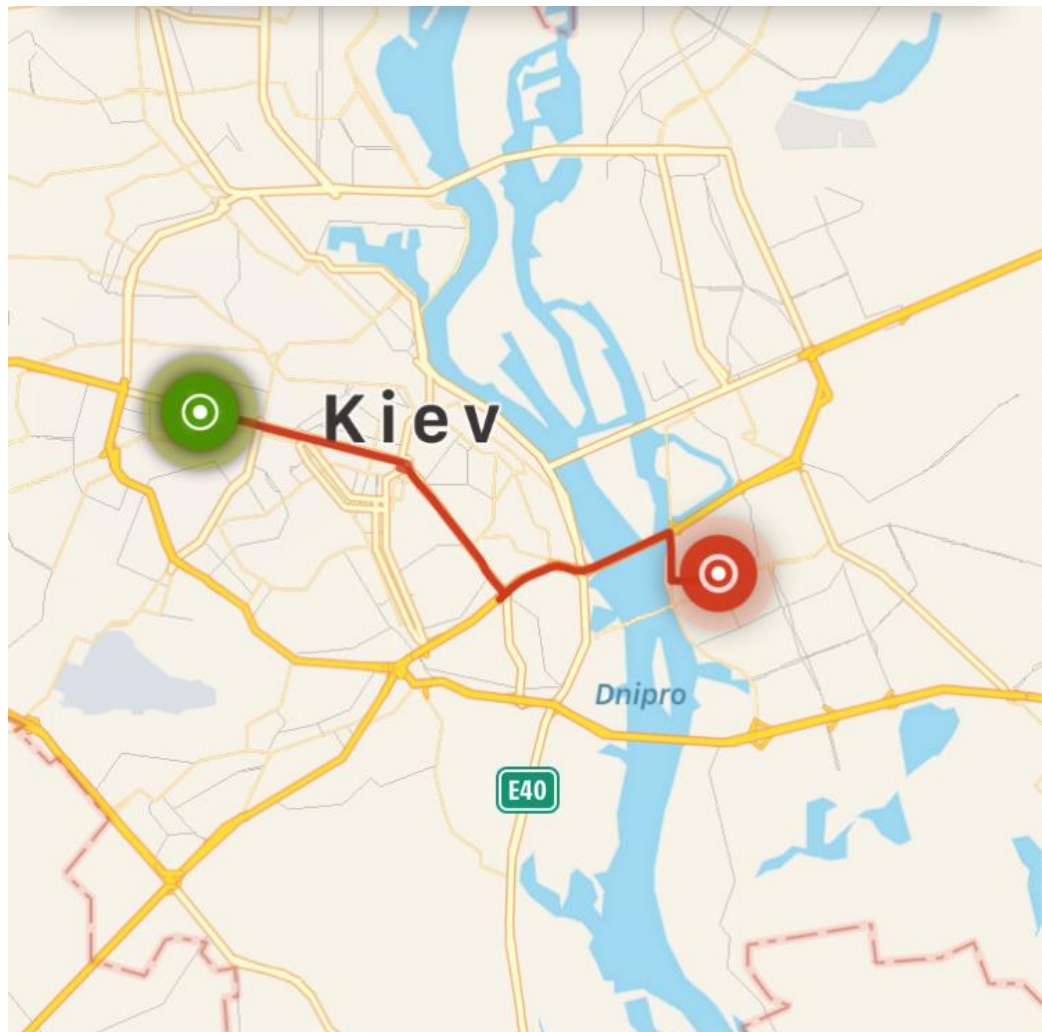


Рисунок 4.12 – MKMapView з лінією MKPolyline

5. ПРОЕКТУВАННЯ І РЕАЛІЗАЦІЯ МОБІЛЬНОГО ЗАСТОСУВАННЯ

У цьому розділі буде спроектовано та реалізовано інтерфейс а також основні функції розроблюваного застосування. Буде розглянута архітектура і реалізація основних функцій застосування.

5.1.Проектування прототипу графічного інтерфейсу

Інструментами для створення інтерфейсу застосування, що виглядає однаково на пристроях з різним розміром екрану є Auto Layout та Stack View.

Auto Layout – інструмент що виконує динамічне обчислення позиції та розміру усіх елементів інтерфейсу на основі правил заданих для того чи іншого з елементів. Цей інструмент дає розробнику можливість створити один єдиний інтерфейс без потреби у підгонці розмірів під певні розміри екранів пристроїв. Auto Layout на основі правил робить це самостійно, динамічно змінюючи інтерфейс застосування в залежності від зміни розмірів екрану пристрою.

Stack View – інструмент автоматичного позиціонування елементів у вікні застосування. Stack View дозволяє створити стек елементів та налаштувати їх позиціонування за наступними параметрами:

- Axis – горизонтальна або вертикальна орієнтація;
- Distribution – визначає положення елементів у даній орієнтації;
- Spacing – визначає простір між сусідніми елементами.

Першим та головним контроллером застосування буде HomeVC (рисунок 5.1). Головний елементом у цьому контроллері є MKMapView, що відображає мапу на якій будуть відображатися основні інформаційні елементи, такі як місцезнаходження водіїв, геолокація користувача а також візуально зображений маршрут майбутньої поїздки. У верхній частині знаходиться кнопка відкриття меню, назва застосування а також фото користувача. Під цими елементами знаходиться вертикально орієнтований Stack View із полями

введення тексту для вводу адреси куди прямує користувач. У нижній частині знаходяться кнопки для відміни поїздки, запиту поїздки а також для фокусу мапи на поточному місцезнаходженні користувача.

Наступним контроллером є LeftSidePanelVC (рисунок 5.1). Він буде відображатися при натисканні на кнопку відкриття меню. Основними його елементами є перемикач режимів для водія, що відображається тільки для користувачів з роллю “водій”, горизонтальний Stack View для відображення основної інформації про користувача, а також кнопка що веде до контроллера реєстрації та входу.

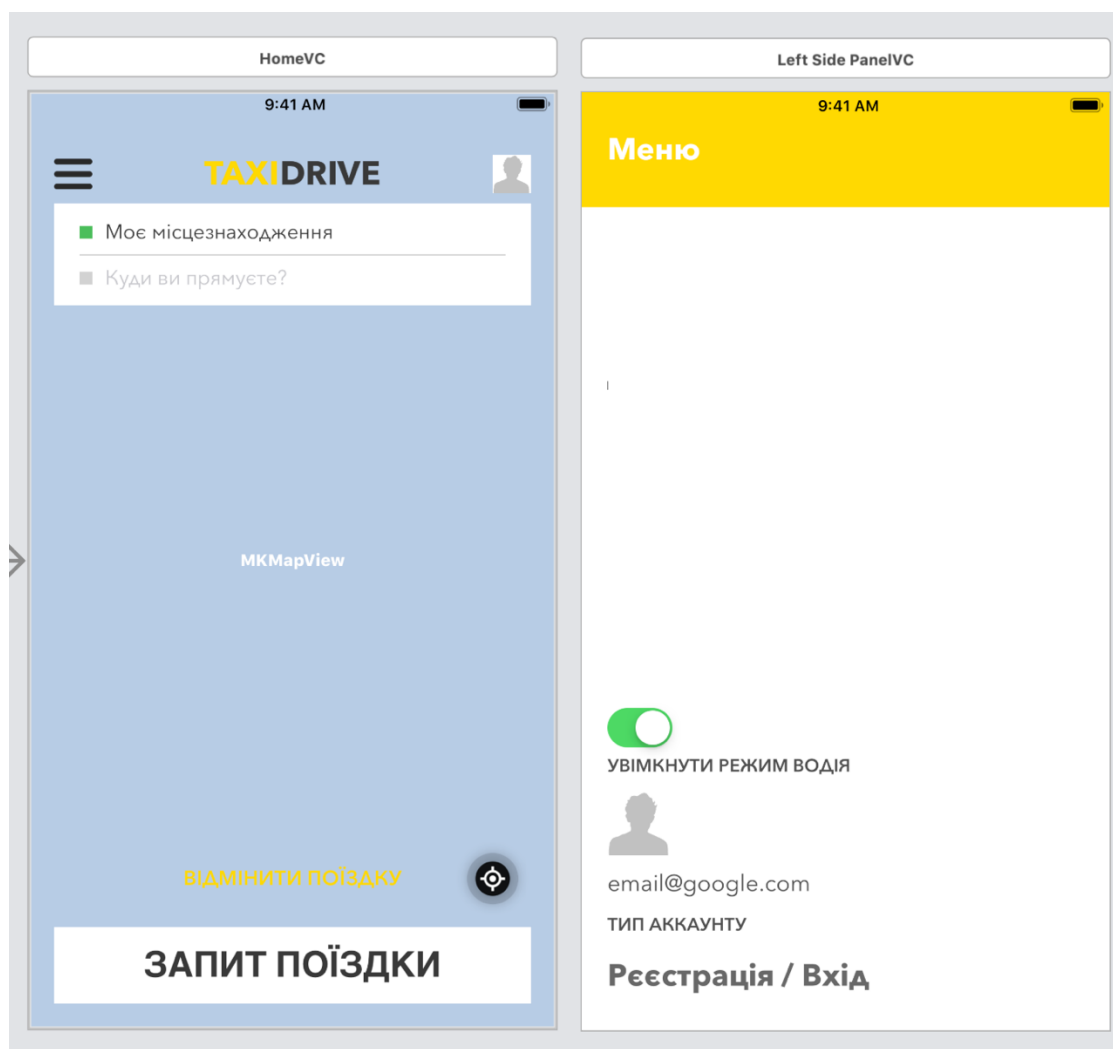


Рисунок 5.1 – Інтерфейси контроллерів HomeVC та LeftSidePanelVC

Далі було розроблено інтерфейс контролеру LoginVC (рисунок 5.2). У ньому знаходиться вертикальний Stack View з короткою інформацією про застосування, елементом Segmented Control для вибору ролі користувача, текстовими полями для вводу пошти та паролю користувача кнопкою входу та реєстрації а також кнопкою закриття вікна.

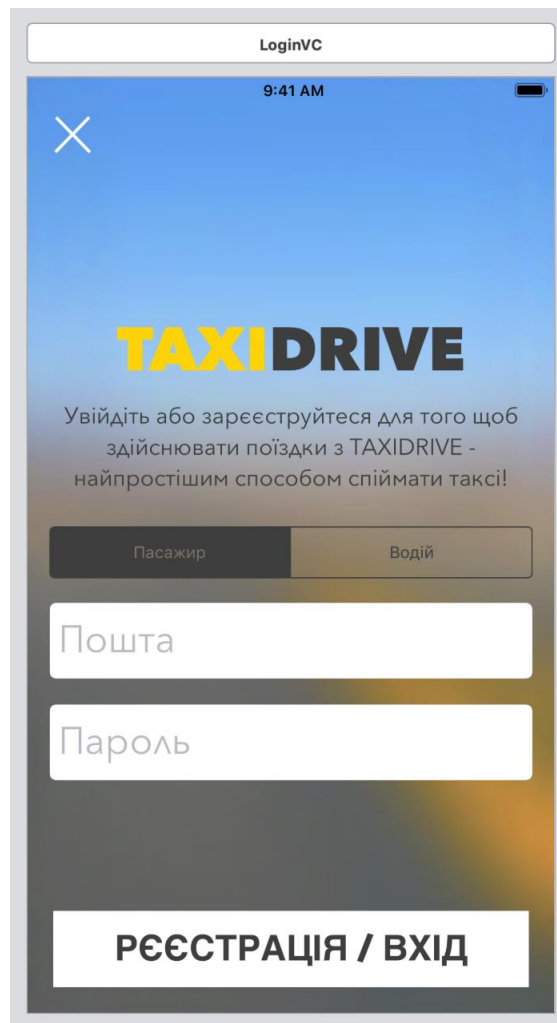


Рисунок 5.2 – Інтерфейс контролеру LoginVC

Останнім розробленим інтерфейсом є інтерфейс контролеру PickupVC (рисунок 5.3). Цей контролер відображається тільки для водіїв що знаходяться поряд із користувачем що виконав запит поїздки, а також у даний момент не виконують іншу поїздку. Основними елементами цього контролеру є

МКMapView для відображення майбутньої поїздки, а також кнопок прийняття або відміни поїздки.

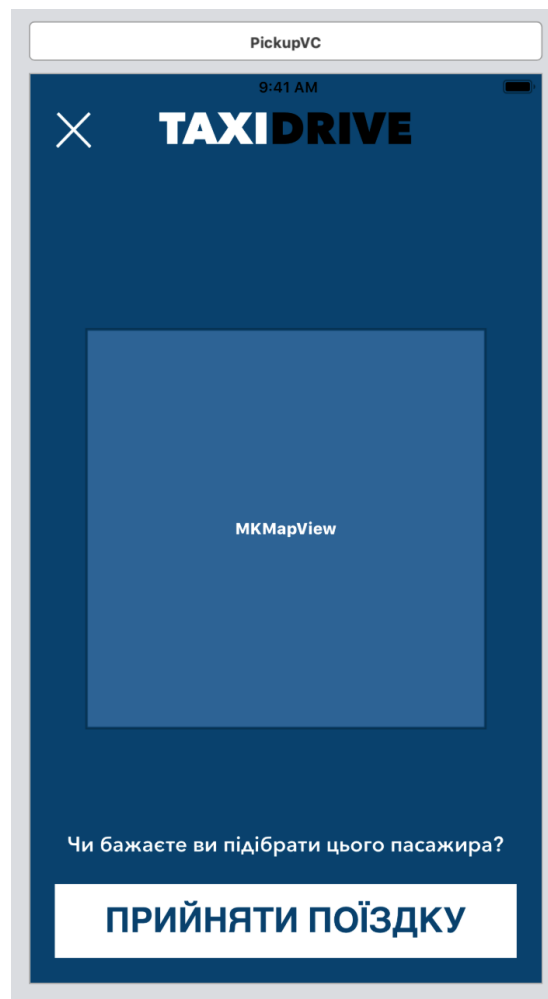


Рисунок 5.3 – Інтерфейс контролеру LoginVC

5.2.Архітектура застосування

Класичний MVC (Model View Controller) – патерн програмування що розділяє застосування три шари абстракції, що дозволяє у подальшому полегшити внесення змін до застосування завдяки чіткому розподілу між компонентами застосування. Основна проблема, яку має вирішити MVC це чіткий поділ між компонентами відповідальності на модель, уявлення та контролер (рисунок 5.4).

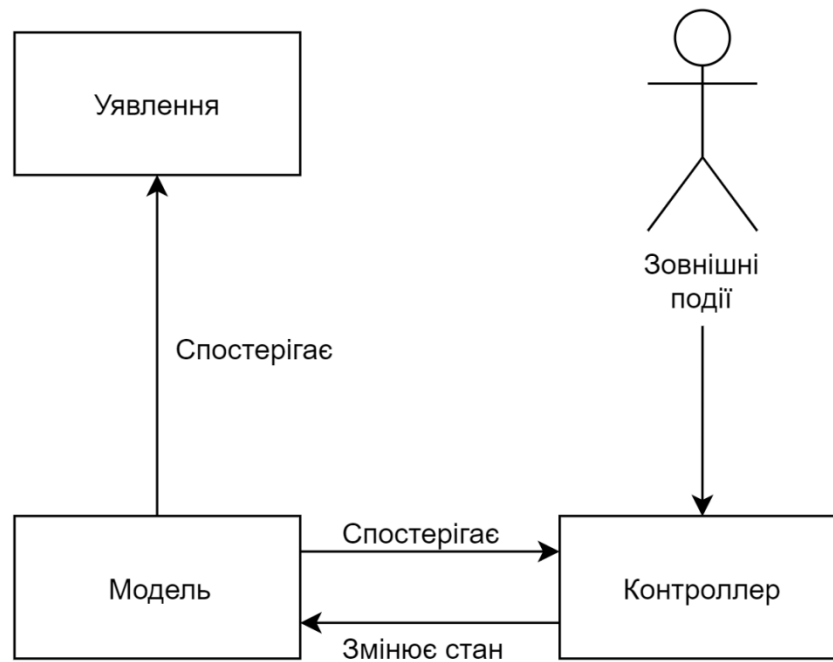


Рисунок 5.4 – Схема класичного MVC

Модель - це набір класів, які інкапсулюють дані конкретної предметної області, їх алгоритми перевірки. У класичному MVC модель також включає логіку обробки («бізнес-логіка»). Існує два типи моделей: активний і пасивний. Активна модель може сповіщати про зміни у своєму стані (як правило, за шаблоном спостерігача). Класична реалізація MVC вважається активною моделлю. Модель нічого не знає про View і Controller і може працювати незалежно. Ця вимога відіграє важливу роль, коли мова йде про тестування.

Уявлення - клас, який відповідає за графічне (хоча і не обов'язково) відображення даних. У класичному MVC View має прямий доступ до моделі тільки в режимі читання. Уявлення не повинне безпосередньо змінювати стан моделі. Це відповідальність Контролера.

Контролер - реагує на зовнішні подразники. Як реакція на зовнішні подразники, контролер виконує певну логіку, включаючи зміну стану моделі. Однак контролер не реагує на перегляд безпосередньо, а також не зберігає свій

стан. Контролер не є посередником між уявленням і моделлю, також він не несе відповідальності за передачу даних від моделі до уявлення.

Одна зі спроб адаптування класичного MVC патерну – Apple MVC (рисунок 5.4). Цей патерн використовується фреймворками Cocoa та CocoaTouch, а отже і розроблюваним застосуванням.

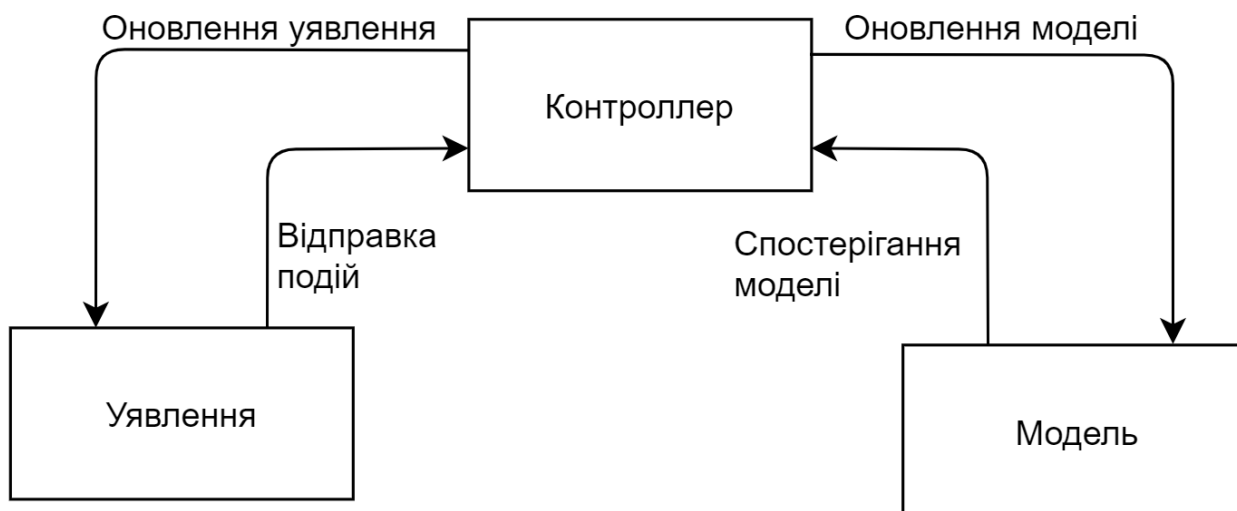


Рисунок 5.4 – Схема патерну Apple MVC

Основними відмінностями від класичного MVC є те, що контролер взаємодіє із уявленням, що посилає події на контролер, а той в свою чергу оновлює уявлення. Контролер змінює стан моделі при необхідності, а коли стан змінюється контролер буде повідомлений і знову повинен вирішити як обробляти ці зміни.

Реалізоване застосування потребує якимсь чином зберігати та передавати інформацію між користувачами, також ці дані повинні передаватися у реальному часі. Досягти цього можна завдяки базі даних у хмарі, та підключення через вебсокет. Але на розробку такої системи знадобиться велика кількість часу та команда людей що будуть підтримувати цей проект. Тому раціонально використовувати послугу BaaS (Backend-as-a-Service) що

дозволить користувачам обмінюватися даними через хмару. Крім того, масштабування програм важке, і коли ваша аудиторія зростає, щоб зробити це ефективно, вам потрібен сервер. Раніше бекенд потрібно було розробляти з нуля, але завдяки технологічним досягненням, тепер ви можете безпосередньо інтегрувати їх у вашу програму. Тому я буду використовувати базу даних реального часу від Firebase.

Backend-as-a-Service (BaaS) – це модель хмарного сервісу, в якій розробники аутсорсингують всі аспекти поза мобільного додатку, так що їм доводиться писати і підтримувати інтерфейс. Постачальники BaaS надають попередньо написане програмне забезпечення для дій, що відбуваються на серверах, таких як автентифікація користувачів, управління базами даних, віддалене оновлення та push-сповіщення (для мобільних додатків), а також зберігання та хостинг.

5.3.Автентифікація користувача

Для реалізації функцій автентифікації було використано сервіс Firebase Authentication, а саме спосіб з використанням пошти та паролю користувача. Після реєстрації користувачу присвоюється унікальний ідентифікатор, а його пошта та пароль зберігаються у хмарі Firebase.

У додатку був використаний єдиний контролер для реєстрації та входу, що потребувало розробити алгоритму для реалізації цього підходу (рисунок 5.5), коли юзер натискає кнопку «Логін/Реєстрація» відбувається запит до серверу автентифікації з даними що впровадив користувач, після цього відбувається перевірка чи вдалою була автентифікація, якщо невдала, то видаємо помилку у випадку невірно введеного паролю, або ж реєструємо нового користувача згідно з впровадженими даними.

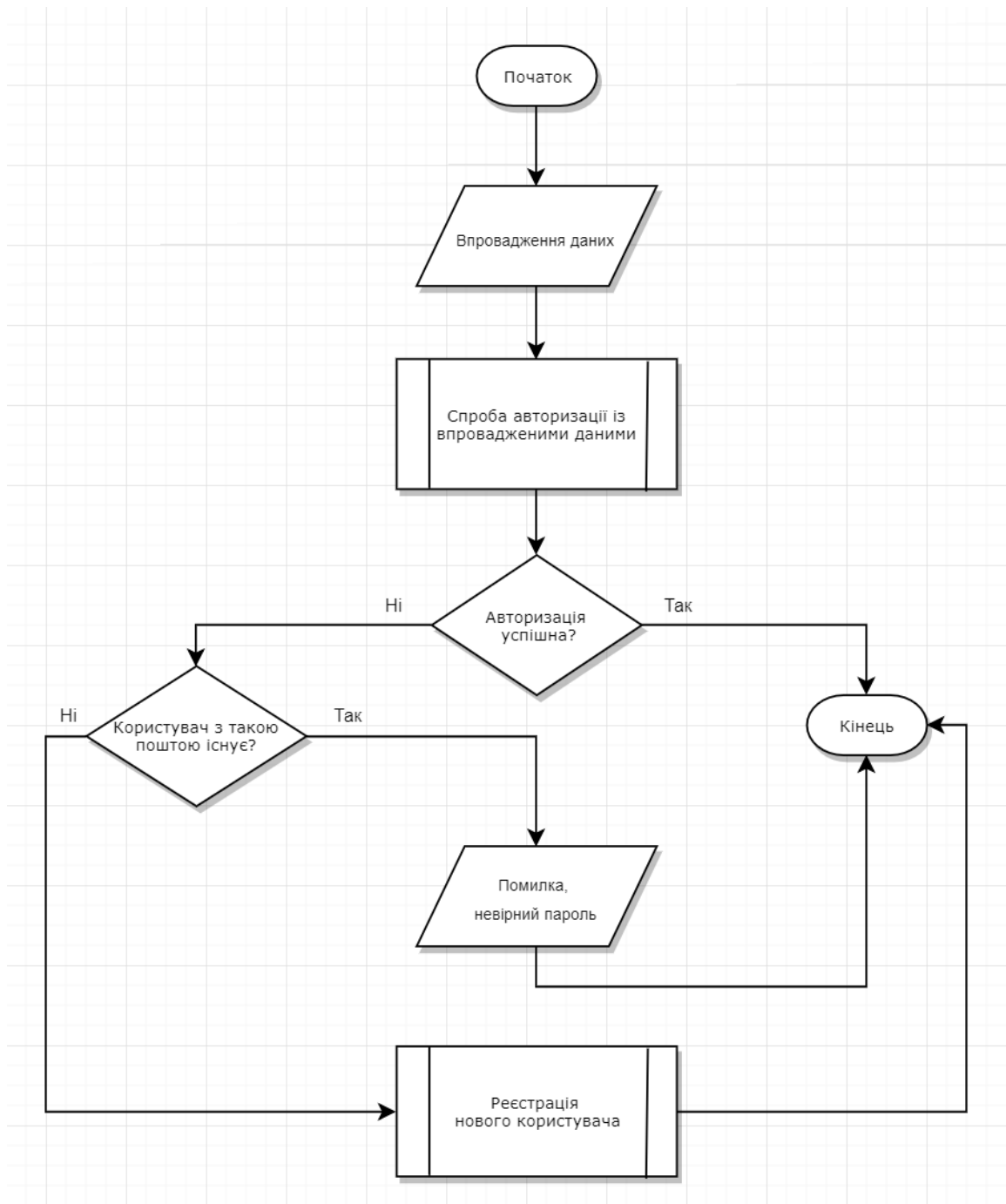


Рисунок 5.5 – Блок-схема алгоритму реєстрації та входу

5.3.1. Створення облікового запису

Головним класом для авторизації, автентифікації та реєстрації у фреймворку Firebase є клас Auth, нижче приведена таблиця основних функцій цього класу (таблиця 5.1).

Таблиця 5.1 – Клас фреймворку Firebase Auth

Клас Firebase Auth		
Властивості		
app	currentUser	languageCode
settings		
Функції		
signInWithCustomToken	signInWithCredential	signInWithEmailLink
signInWithPopup	signOut	createUser

При створенні облікового запису використовується функція “createUser” реалізація якої інкапсульована у фреймворку Firebase у класі Auth (таблиця 5.1). У цю функцію передається пошта та пароль введений користувачем. У випадку якщо після запиту до серверу Firebase не відбулося помилок, викликана функція має повернути об’єкт класу User, що має у собі наведені у таблиці нижче (таблиця 5.2) функції та властивості.

Таблиця 5.2 – Клас фреймворку Firebase User

Клас Firebase User		
Властивості		
displayName	email	emailVerified
isAnonymous	metadata	phoneNumber
photoURL	providerData	providerId
refreshToken	uid	
Функції		
delete	getIdToken	getIdTokenResult
updateEmail	updatePassword	updatePhoneNumber

Як видно з таблиці (таблиця 5.1) об'єкт класу User має властивість “uid”, це унікальний ідентифікатор що асоціюється з об'єктом користувача. Унікальний ідентифікатор (UID) – це числове або буквено-цифрове значення що являє собою унікальне ім'я об'єкту у межах певної системи. Основною задачею UID є бути унікальним показником певного об'єкту, наприклад у базі даних, для того щоб отримати доступ до певної сутності та взаємодіяти з нею.

Унікальний ідентифікатор користувача повертаємого об'єкту класу User використано у базі даних реального часу Realtime Database у якості ідентифікатору користувача за яким можна отримати інформацію про нього, а саме про його місцезнаходження, координати його напрямку, а також інші логічні значення (Додаток А). На основі впроваджених користувачем даних про його роль у контроллері реєстрації, створюється об'єкт у базі даних у середині об'єкту “drivers” або “users”.

Оскільки база даних є нереляційною, кожен запис є об'єктом із набором властивостей та інших об'єктів, а сама база даних являє собою один великий JSON-об'єкт (рисунок 5.6).



Рисунок 5.6 – Вигляд нереляційної бази даних

5.4.Відображення місцезнаходження

Широта і довгота – це глобальна адреса місця на землі. Кожне місцезнаходження на землі має глобальну адресу. Оскільки ця “адреса” відображається у цифрах, люди можуть повідомляти про місцезнаходження незалежно від мови на якій вони можуть говорити а також використовувати ці значення для геолокації у форматі чисел. Глобальна адреса надається як два числа, які називаються координатами. Два числа - це географічна широта та довгота.

Використання широти та довготи відрізняється від текстової адреси. Замість того щоб мати певну адресу, чисельне позначення працює з нумерованою системою сітки, яку ви можете побачити, коли ви дивитеся на паперову мапу. Вона має горизонтальні лінії і вертикальні лінії, які перетинаються. Місце розташування може бути відображене або знайдено на сітці шляхом надання двох чисел, які є горизонтальними і вертикальними координатами розташування; або, кажучи іншим чином, "перехрестя", де розташоване місце.

Лінії широти та довготи також є системою картографічної сітки. Але замість того, щоб бути прямими лініями на гладкій поверхні лінії широти та довготи оточують Землю, як горизонтальні кола і вертикальні півкола.

Горизонтальні лінії відображення на Землі це лінії широти. Вони відомі як "паралелі" широти, тому що вони проходять паралельно екватору. Один з простих способів візуалізувати це – думати про наявність уявних горизонтальних "обручів" навколо Землі, з найбільшим “обручем” навколо екватора, а потім поступово менших, що укладаються вище і нижче, щоб досягти Північного і Південного полюсів.

Лінії широти є числовим способом, щоб визначити, наскільки далеко на північ або південь від екватора розташоване місце. Екватор є відправною точкою для вимірювання широти - тому він позначений як 0 градусів широти.

					IT51.017БАК.009 ПЗ	Лист
Ізм.	Лист	№ докум.	Підпис	Дата		51

Число градусів широти буде більше, чим далі від екватора розташоване місце, аж до 90 градусів широти на полюсах. Місцеположення в географічній зоні наведено у вигляді кількості градусів на північ або кількості градусів на південь.

Вертикальні лінії відображення на Землі це лінії довготи, відомі як "меридіани". Один простий спосіб візуалізувати це – думати про те, що “обручі” обрізаються навпіл та вертикально розташовані з одним кінцем на Північному полюсі, а іншим на Південному полюсі.

Лінії довготи є чисельним способом, щоб показати або виміряти, наскільки далеко розташування знаходиться на схід або на захід від універсальної вертикальної лінії, яка називається Первинний меридіан. Лінія "Перший меридіан" проходить вертикально на північ і південь, прямо над Британською королівською обсерваторією Грінвіч у Англії, від Північного полюса до Південного полюса. В якості вертикальної відправною точкою довготи початковий меридіан має нульову довжину 0 градусів.

Щоб виміряти довготу на схід або на захід від початкового меридіана, є 180 вертикальних ліній довготи на схід від початкового меридіана і 180 вертикальних довготних ліній на захід від початкового меридіана, тому довгота розташована на певній кількості градусів на схід або на захід.

5.4.1. Відображення пасажирів

Для визначення місцеположення як водіїв так і пасажирів використовується CLLocationManager. Дані про місцезнаходження користувачів зберігаються у базі даних в об'єкті “coordinate”, що знаходиться всередині об'єкту даних користувача та реактивно оновлюються зі зміною позиції. Для пасажирів їх поточне місцезнаходження відображається синьою крапкою на мапі (рисунок 5.7).

При натисканні на кнопку фокусу відбувається центрування MKMapView на теперішньому місцезнаходженні користувача.

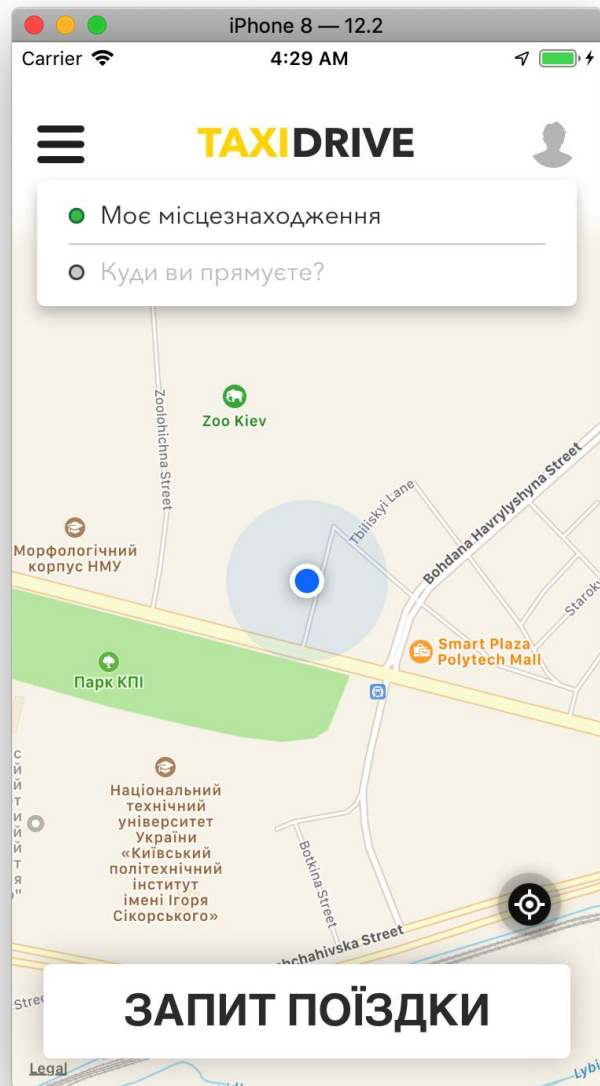


Рисунок 5.7 – Відображення поточного місцезнаходження користувача

Кнопка фокусу поточного місцезнаходження створена для того щоб у випадку випадкового пересування фокусу мапи у інше положення користувач міг легко повернутися до свого місцеположення на мапі.

5.4.2. Відображення водіїв

Водії на мапі відображаються червоною точкою з іконкою таксі у середині (рисунк 5.8). Кожен користувач може бачити де знаходиться водій що не виконує поїздки у даний момент та має активований перемикач режиму водія. Координати водіїв оновлюються завдяки CCLocationManager, зберігаються у базі даних та реактивно оновлюються. Місцеположення зберігається у об'єкті “coordinate”, де значення з ключем “0” відповідає за широту, а значення з ключем “1” відповідає за довготу.

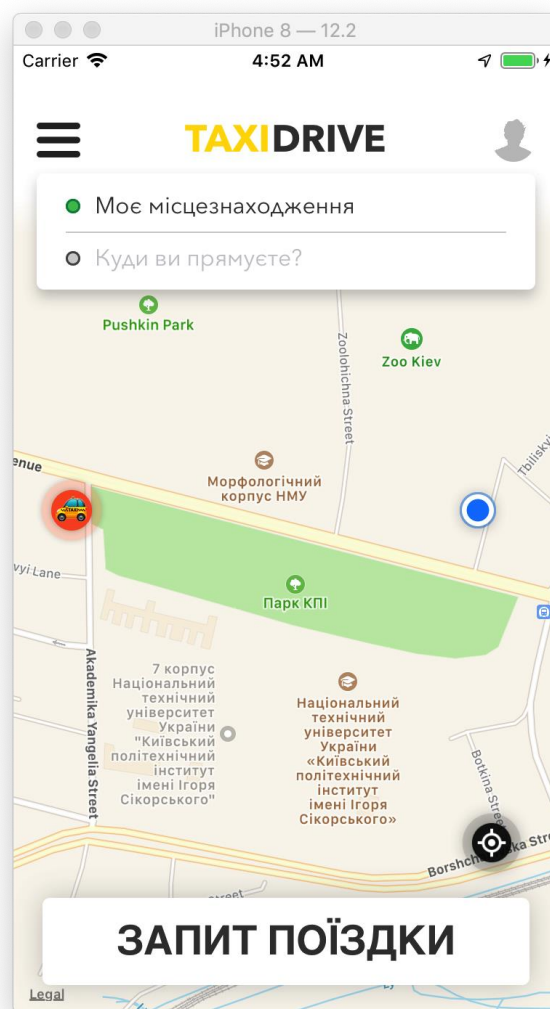


Рисунок 5.8 – Відображення водіїв на мапі

5.5.Здійснення поїздки

Цикл здійснення поїздки відбувається коли користувач вводить адресу напрямку у текстове поле і натискає кнопку пошуку. На цьому етапі відбувається запит до MKLocalSearch, класу що виконує пошук адрес або визначених місць на мапі. Після завершення запиту результат передається до обробника завершення запиту, що відображає результати на TableView. Коли користувач обирає один із варіантів адрес MKTableView будує PolyLine від поточної геопозиції до місця призначення (рисунок 5.9).

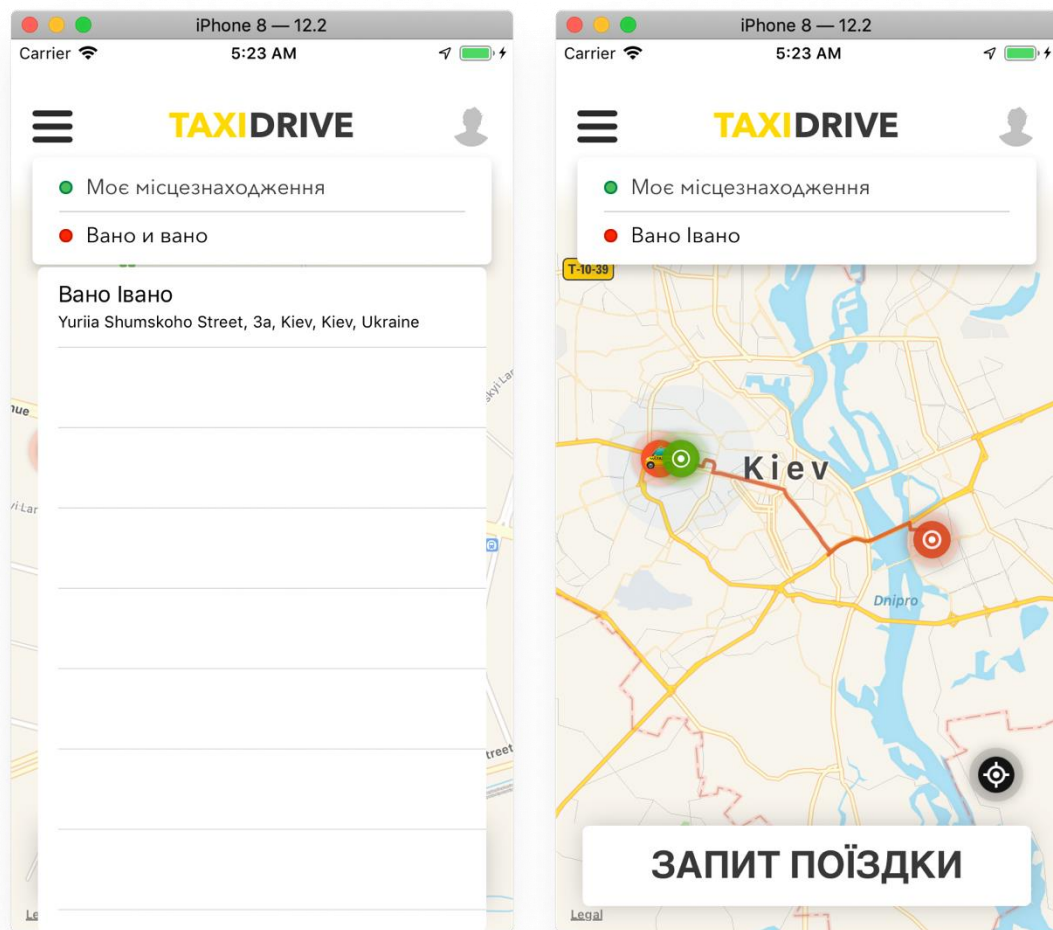


Рисунок 5.9 – Пошук і вибір місця напрямку користувачем

Після цього користувач може натиснути кнопку “Запит поїздки”, що в свою чергу відобразить контроллер PickupVC з місцезнаходженням пасажирів на мапі у водіїв що знаходяться поряд. У водія є можливість натиснути кнопку “Прийняти поїздки”, або закрити вікно.

Після того як водій прийняв поїздки для нього буде прокладений маршрут до місця посадки пасажирів, а також з’явиться кнопка “Відкрити навігатор”, що відкриває навігатор з впровадженням місцем посадки пасажирів. Коли місцезнаходження водія буде у радіусі двохсот метрів від пасажирів, водій зможе почати поїздки натиснувши кнопку “Почати поїздки” (рисунки 5.10).

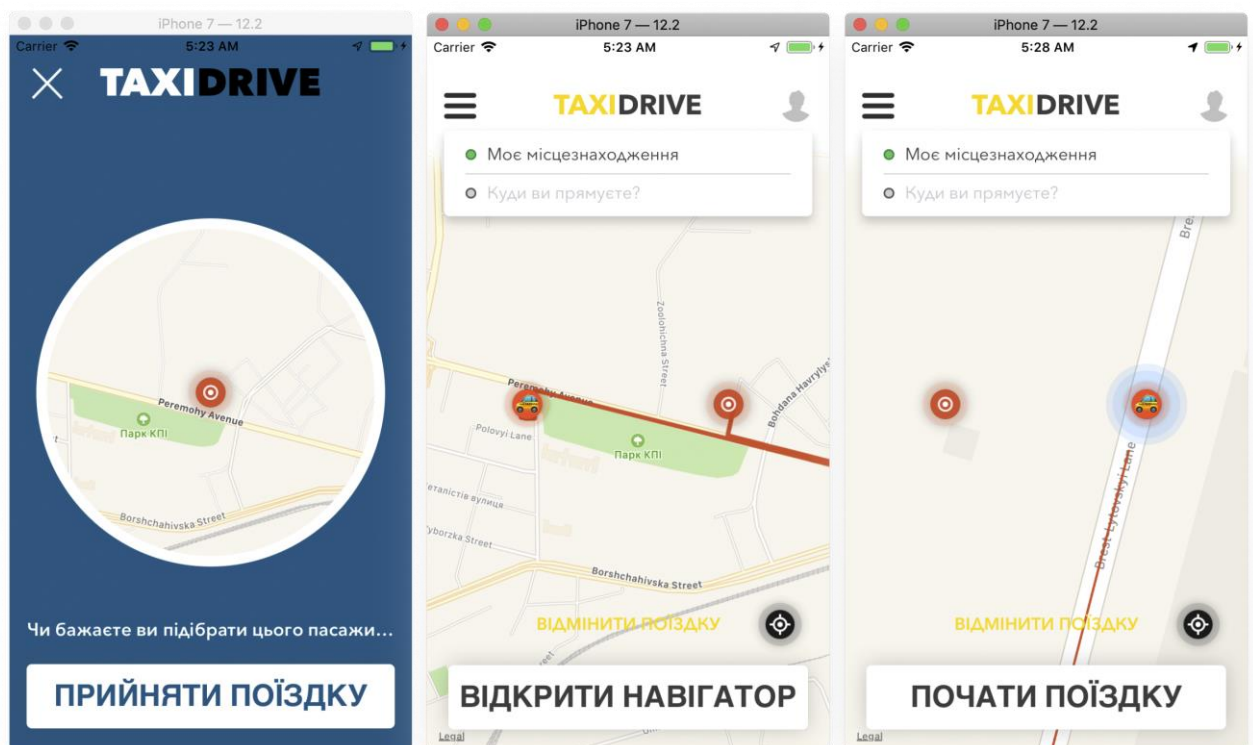


Рисунок 5.10 – Вигляд прийняття та початку поїздки для водія

Після початку поїздки буде прокладений PolyLine від місця підбору пасажирів до місця призначення а також буде доступна кнопка “Відкрити навігатор” . Після прибуття в радіус місця призначення водію буде доступна

кнопка “Завершити поїздку”, після її натискання статус поїздки буде змінений на “завершений”, МКMapView буде очищений від зайвих анотацій і масштаб мапи буде змінено на більш широкий (рисунк 5.11).

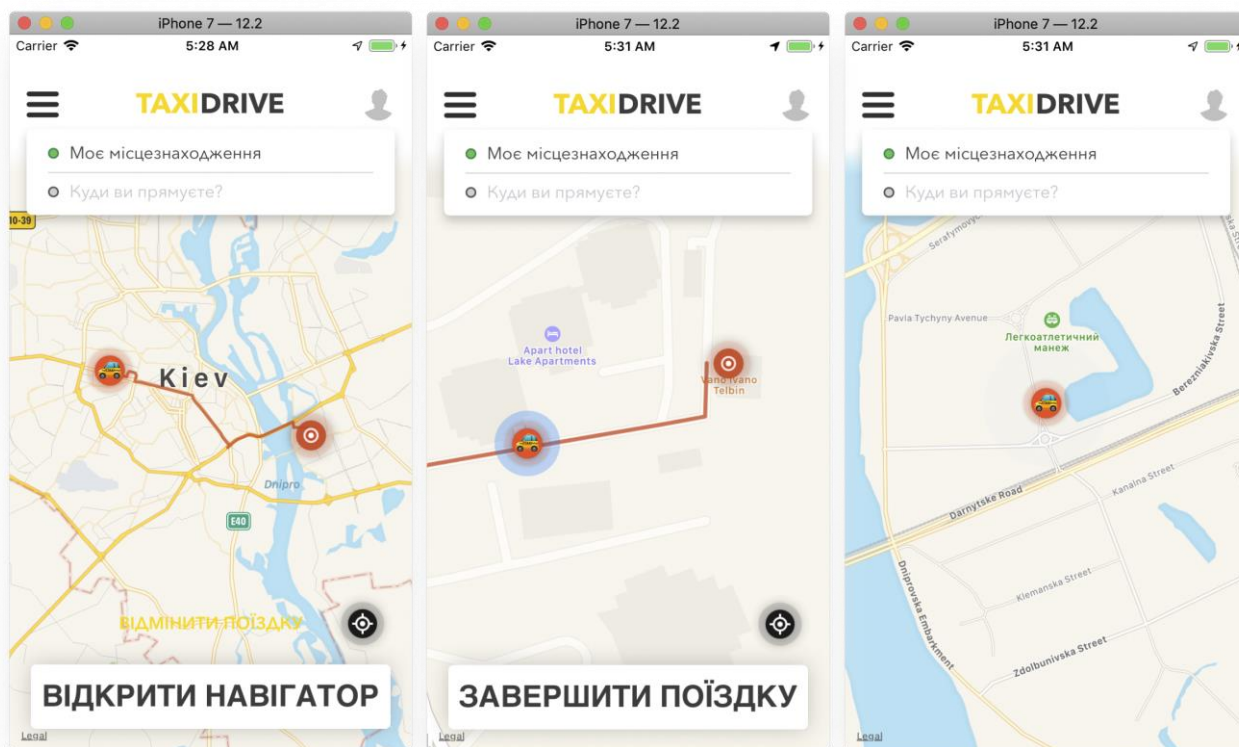


Рисунок 5.11 – Вигляд початку та завершення поїздки

Ізм.	Лист	№ докум.	Підпис	Дата

ВИСНОВОК

У дипломній роботі було розроблено мобільне застосування для обслуговування потреб у галузі таксі.

Здійснено аналіз існуючих рішень для вирішення поставленої у дипломній роботі задачі. Досліджено основні програмні засоби що використовуються для розробки мобільних додатків, розглянуто інструменти для дизайну та прототипування.

Проведено огляд існуючих методів для розробки та тестування мобільних додатків. Розроблено макет мобільного застосування а також елементи його дизайну.

					IT51.017БАК.009 ПЗ	Лист
						58
Ізм.	Лист	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. PYPL (Popularity of Programming Language Index) [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <http://pypl.github.io/PYPL.html>
2. A brief post about what Firebase is all about, and it's new NoSQL Database Cloud Firestore [Електронний ресурс] – 2017. – Режим доступу до ресурсу: <https://hackernoon.com/introduction-to-firebase-218a23186cd7>
3. MapKit Framework [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <https://developer.apple.com/documentation/mapkit>
4. MapKit Framework [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <https://developer.apple.com/documentation/corelocation>
5. Firebase [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <https://uk.wikipedia.org/wiki/Firebase>
6. iOS против Android. Что всё-таки лучше? [Електронний ресурс] – 2017. – Режим доступу до ресурсу: <https://www.iphones.ru/iNotes/759337>
7. A Swift tour [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <https://docs.swift.org/swift-book/GuidedTour/GuidedTour.html>
8. Firebase Realtime database [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <https://firebase.google.com/docs/database>
9. Apple Xcode IDE от Apple [Електронний ресурс] – 2019. – Режим доступу до ресурсу: <http://wnfx.ru/apple-xcode-ide-ot-apple>
10. Основы использования Cocos2d в разработке приложений под iOS [Електронний ресурс] – 2015. – Режим доступу до ресурсу: <https://habr.com/ru/post/261711>
11. Sketch Tutorial for iOS Developers [Електронний ресурс] – 2016. – Режим доступу до ресурсу: <https://www.raywenderlich.com/1379-sketch-tutorial-for-ios-developers>

12. Основы Auto Layout — Концепция, строение, применение [Электронный ресурс] – 2016. – Режим доступа до ресурсу: <https://habr.com/ru/post/312782>

13. Математические основы Auto Layout [Электронный ресурс] – 2019. – Режим доступа до ресурсу: <https://habr.com/ru/company/oleg-bunin/blog/437584>

14. NoSQL базы данных: понимаем суть [Электронный ресурс] – 2012. – Режим доступа до ресурсу: <https://habr.com/ru/post/152477>

15. About an Xcode IDE for Apple application development [Электронный ресурс] – 2018. – Режим доступа до ресурсу: <https://medium.com/worst-ios-developer/about-an-xcode-ide-for-apple-application-development-68b161088e53>

ДОДАТОК А. СХЕМА БАЗИ ДАНИХ

Таблиця А.1

Users		
Унікальний ідентифікатор користувача (UID)		
Ключ	provider	Строкове значення що визначає походження реєстрації
Coordinate		
Ключ	0	Широта
Ключ	1	Довгота

Таблиця А.2

Drivers		
Унікальний ідентифікатор водія (UID)		
Ключ	driverIsOnTrip	Булеве значення що визначає чи виконує поїздку водій у даний момент
Ключ	isPickupModeEnabled	Булеве значення що визначає чи увімкнув водій режим підбору пасажирів
Ключ	provider	Строкове значення що визначає походження реєстрації
Coordinate		
Ключ	0	Широта

Продовження таблиці А.2

Ключ	1	Довгота
------	---	---------

Таблиця А.3

Trips		
Унікальний ідентифікатор користувача що зробив замовлення (UID)		
Ключ	driverKey	Унікальний ідентифікатор водія що прийняв замовлення
Ключ	passangerKey	Унікальний ідентифікатор пасажира що створив замовлення
pickupCoordinate		
Ключ	0	Широта
Ключ	1	Довгота
destinationCoordinate		
Ключ	0	Широта
Ключ	1	Довгота