НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Індивідуальний дослідницький проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Інтерактивна система бронювання столиків у ресторані»

Виконав:
студент IV курсу, групи ІА-82
Саволюк Данило Дмитрович _____

Керівник:
Доцент кафедри ІСТ, к. т. н
Шимкович Володимир Миколайович _____

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

Студент _____

Київ – 2022 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАВДАННЯ

на індивідуальний дослідницький проєкт студенту

Саволюку Данилу Дмитровичу

1. Тема проєкту «Інтерактивна система бронювання столиків у ресторані», керівник проєкту Шимкович Володимир Миколайович, доцент кафедри ІСТ, к. т. н

2. Термін подання студентом проєкту: 15 червня 2022 року

3. Вихідні дані до проєкту: Інтерактивна система бронювання столиків у ресторані, розроблена у вигляді андроїд додатку. Цей застосунок дозволить бронювати столики, замовляти їжу наперед та оплачувати замовлення.

4. Зміст пояснювальної записки: вступ, призначення та галузь , технічні характеристики, огляд та аналіз існуючих рішень, аналіз обраної системи, організація процесу розробки, програмне та технічне забезпечення, розроблення діаграм програмного додатку, інформаційне забезпечення, програмна реалізація застосунку, інструкція користувача, висновок

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) діаграма компонентів, структурна схема бази даних, діаграма варіантів використання, діаграма послідовності, діаграма класів.

6. Дата видачі завдання 1 грудня 2021 року

Календарний план

№ з/п	Назва етапів виконання індивідуального дослідницького проєкту	Термін виконання етапів проєкту	Примітка
1	Огляд та аналіз існуючих рішень	15.03.2022	
2	Аналіз обраної системи	01.04.2022	
3	Розроблення діаграми компонентів	10.04.2022	
4	Огляд засобів розробки	15.04.2022	
5	Розроблення UML діаграм програмного додатку	20.04.2022	
6	Програмна реалізація застосунку	10.05.2022	
7	Розроблення інструкції користувача	30.05.2022	
8	Оформлення документації проєкту	05.06.2022	

Студент Данило САВОЛЮК

Керівник Володимир ШИМКОВИЧ

АНОТАЦІЯ

Саволюк Д. Д. Інтерактивна система бронювання столиків у ресторані. КПІ ім. Ігоря Сікорського, Київ, 2022.

Проєкт містить 89 с. тексту, 50 рисунків, 11 таблиць, посилання на 17 джерел, та 5 конструкторських документів.

Ключові слова: бронювання столиків, система бронювань, Android додаток, автоматизація, клієнт-серверна архітектура.

Об’єктом розробки є система бронювання столиків у ресторанах.

Мета розробки – автоматизація процесу бронювання столиків у закладах харчування.

Даний індивідуальний дослідницький проєкт має за ціль розробку системи, основна задача якої - це бронювання столиків у ресторані, використовуючи лише додаток на своєму смартфоні.

Дана система дозволяє користувачу зручно забронювати столик у ресторані, а додаткові функції, такі як перегляд меню, оформлення замовлення наперед та онлайн оплата через застосунок дозволять ще більше часу економити відвідувачам, а ресторану - ефективніше розподіляти навантаження.

Програмна складова додатку реалізована на високорівневій мові програмування Kotlin. Back-end програми, що виконує основну логіку та зберігає інформацію у базах даних, написаний на NodeJS.

В даному індивідуальному дослідницькому проєкті розроблено: додаток для зручної взаємодії відвідувача та ресторану, структурна схема програми, структурна схема бази даних, діаграма класів, діаграма варіантів використання, діаграма послідовності та графічні елементи.

SUMMARY

Savoliuk D.D. Interactive restaurant table reservation system. Igor Sikorsky KPI, Kyiv, 2022.

The project contains 89 pages. text, 50 figures, 11 tables, references to 17 sources, and 5 design documents.

Keywords: table reservation, reservation system, Android application, automation, client-server architecture.

The object of development is a system for tables reservation in restaurants.

The purpose of the development is to automate the process of booking tables in catering establishments.

This individual research project aims to develop a system, the main task of which is to book tables in a restaurant only with the help of application in your smartphone .

This system allows the user to conveniently book a table in the restaurant, and additional features such as menu viewing, pre-ordering and online payment through the application will save even more time for visitors, and will help the restaurant more efficiently distribute the load.

The software component of the application is implemented with the help of the high-level Kotlin programming language. Back-end of program that executes basic logic and stores information in databases, written in NodeJS.

In this individual research project were developed: an application for easy interaction between the visitor and the restaurant, the block diagram of the program, the block diagram of database, class diagram, diagram of use cases, sequence diagram and graphic elements.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА82.230БАК.003.ПЗ	Пояснювальна записка	89		
6	А3	ІА82.230БАК.003 Д1	Інтерактивна система	1		
7			бронювання столиків у			
8			ресторані. Діаграма компонентів			
9	А3	ІА82.230БАК.003 Д2	Інтерактивна система	1		
10			бронювання столиків у			
11			ресторані. Діаграма варіантів			
12			використання			
13	А3	ІА82.230БАК.003 Д3	Інтерактивна система	1		
14			бронювання столиків у			
15			ресторані. Діаграма			
16			послідовності			
17	А3	ІА82.230БАК.003 Д4	Інтерактивна система	1		
18			бронювання столиків у			
19			ресторані. Діаграма класів			
20	А3	ІА82.230БАК.003 Д5	Інтерактивна система	1		
21			бронювання столиків у			
22			ресторані. Структурна схема			
23			бази даних			
24						
25						

Зм.	Арк.	ПІБ	Підп.	Дата	ІА82.230БАК.003 ТП			
Розробн.		Саволюк Д.Д			Інтерактивна система бронювання столиків у ресторані. Відомість проєкту	Літ.	Арк.	Аркушів
Керівн.		Шимкович В.М.					1	1
.						КПІ ім. Ігоря Сікорського		
.						Група ІА-82		
Затв..								

Пояснювальна записка
до індивідуального дослідницького проєкту на тему:
«Інтерактивна система бронювання столиків у ресторанах»

Київ – 2022 року

ЗМІСТ

СПИСОК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – База Даних.

API – Application Programming Interface.

SDK – Software Development Kit.

UML – Unified Modeling Language.

UI – User Interface.

DI – Dependency Injection

RAM – Random-Access Memory

SHA-1 – Secure Hash Algorithm

HTTP – Hypertext Transfer Protocol

IDE – Integrated Development Environment

SQL – Structured Query Language

VSC – Version Control System

ПЗ – Програмне Забезпечення

ВСТУП

Сучасний світ все швидшими темпами рухається до цифровізації, починаючи від малого бізнесу і закінчуючи рівнем держави. У багатьох людей є персональні комп'ютери, а ще частіше – власні смартфоні, якими вони користуються кожен день. Постійно для цих смартфонів виходять нові застосунки, які так чи інакше спрощують наше життя, а іноді і взагалі – кардинально його змінюють.

Зараз існує багато додатків зі сфери послуг, які дозволяють людям ефективніше розпоряджатись їхнім часом. Яскравим прикладом є застосунки таких типів як: доставка, зокрема не тільки товарів, а й їжі; таксі; мессенджери; бронювання – будь то готелі чи квитки на громадський транспорт. Іноді може здатись, що всі можливі інформаційні продукти вже створені і ідей для нових вже не залишилось.

Проте, проаналізувавши ринок, можна зрозуміти, що попри широкий вибір додатків все ще не всі ніші заповнені. Однією з таких ніш є додатки для бронювання столиків у ресторанах чи кафе.

Часто буває таке, що хочеться відвідати ввечері у свій улюблений ресторан, але коли туди приходиш, то всі місця виявляються зайнятими. Звісно, можна подзвонити до ресторану напередодні і замовити столик. Проте сучасні люди звикли до комфорту: їм набагато простіше зробити декілька кліків у смартфоні замість непотрібних дзвінків. Тим паче, що за допомогою такого додатку можна не просто бронювати столики, а ще й переглядати меню закладу, робити замовлення наперед та оплачувати онлайн. На додачу можна буде одразу переглянути рейтинг ресторану, а після відвідування залишити відгук.

Такий додаток принесе користь не тільки відвідувачу, але й ресторану. Офіціантам не доведеться приймати замовлення вручну, оскільки можна зробити його онлайн, що дозволить уникнути помилок, зекономити час та приготувати замовлення завчасно.

Такий додаток дозволить бізнесу ефективніше використовувати свої ресурси, а клієнту – з більшим комфортом отримувати послуги.

Метою даного індивідуального дослідницького проєкту є розроблення застосунку з назвою «Bookresto» на Android, який дозволить бронювати столики у закладах харчування, замовляти меню наперед та оплачувати своє замовлення без наявності карти чи готівки. Додаток буде містити й інші корисні функції: карту з ресторанами навколо, список улюблених закладів та функцію залишку чайових для офіціанта.

В ході роботи над індивідуальним дослідницьким проєктом було виконано наступні задачі:

- огляд і порівняння існуючих систем для резервації столиків;
- вибір типу застосунку та мови програмування;
- огляд засобів розробки;
- зроблено аналіз обраної системи;
- розроблення діаграми компонентів у додатку;
- розроблення UML- діаграм системи;
- дизайн та розроблення застосунку;
- розроблення інструкції користувача.

1 ПРИЗНАЧЕННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

За допомогою додатку, розробленого у ході реалізації індивідуального дослідницького проєкту, люди зможуть зручніше взаємодії з ресторанам, шляхом резервації столиків, оформленням замовлення заздалегідь та онлайн оплатою. Можна буде переглянути відгуки та оцінки будь-якого закладу та обрати найкращий, а також залишити власну рецензію після відвідування.

Ресторанному бізнесу ж стане простіше приймати замовлення, оскільки вони будуть в онлайн форматі; вони зможуть ретельніше використовувати свої ресурси. Зручна система онлайн взаємодії буде приваблювати клієнтів, що дозволить збільшити прибуток. Також завдяки тому, що всі дані будуть зберігатися в електронному вигляді, можна буде в майбутньому проводити аналітику вподобань клієнтів і використовувати її для покращення бізнесу.

Оскільки в сучасному світі майже всі операції відбуваються банківськими картками, багато людей не носить з собою готівку, через що офіціанти можуть лишатися чайових. У додатку ж буде можливість додати чайові до оплати замовлення, тому всі залишаться задоволені.

2 ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

Одним з основних етапів проєктування додатків є вибір технічних характеристик пристрою, щоб він міг виконувати всі завдання. Після аналізу функцій, необхідних для реалізації, були визначені наступні характеристики пристрою:

- архітектура процесора ARMv7, MIPS або вище;
- операційна система Android 8.0 (API рівня 26) або вище;
- 1 ГБ RAM (RAM) або більше;
- 50 МБ вільного місця пам’яті;
- екран з роздільною здатністю 1280x720 пікселів і вище;
- можливість виходу в Інтернет через модуль Wi-Fi, GSM, 3G, 4G або 5G.

3 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

Для аналізу існуючих рішення насамперед було перевірено Google Play Market – основну платформу додатків на Android – на наявність схожих додатків.

Проте незалежно від платформи, кожна система резервації для ресторанів пропонує унікальний спосіб реалізації функцій, які можна адаптувати до будь-якої платформи. Тому дуже важливо проаналізувати ринок існуючого програмного забезпечення, а не обмежуватися лише мобільними платформами. Через це було також досліджено і рішення для інших платформ, а саме web. Серед найпопулярніших сервісів можна виділити наступні:

- «RestOn» - андроїд додаток;
- «TheFork» - андроїд додаток;
- «OpenTable» - веб-сервіс.

3.1 Додаток «RestOn»

«RestOn» – це сервіс онлайн бронювання столиків у ресторані будь-якого міста. На рисунку 3.1 можна побачити, що у застосунку є список доступних закладів прямо на карті, що достатньо зручно, або списком. Також є різні фільтри, такі як місто, діапазон середнього чеку, тип закладу, кухня, район, метро тощо. Вони допоможуть переглядати тільки ті місця, які точно підходять користувачу.

У цьому застосунку також є сторінка попередніх бронювань та сторінка обраних ресторанів. Будь-який ресторан можна додати у обране, просто натиснувши на відповідну кнопку на сторінці закладу. Завдяки цьому користувачу не доведеться вдруге шукати улюблений ресторан, можна просто знайти його на цих сторінках.

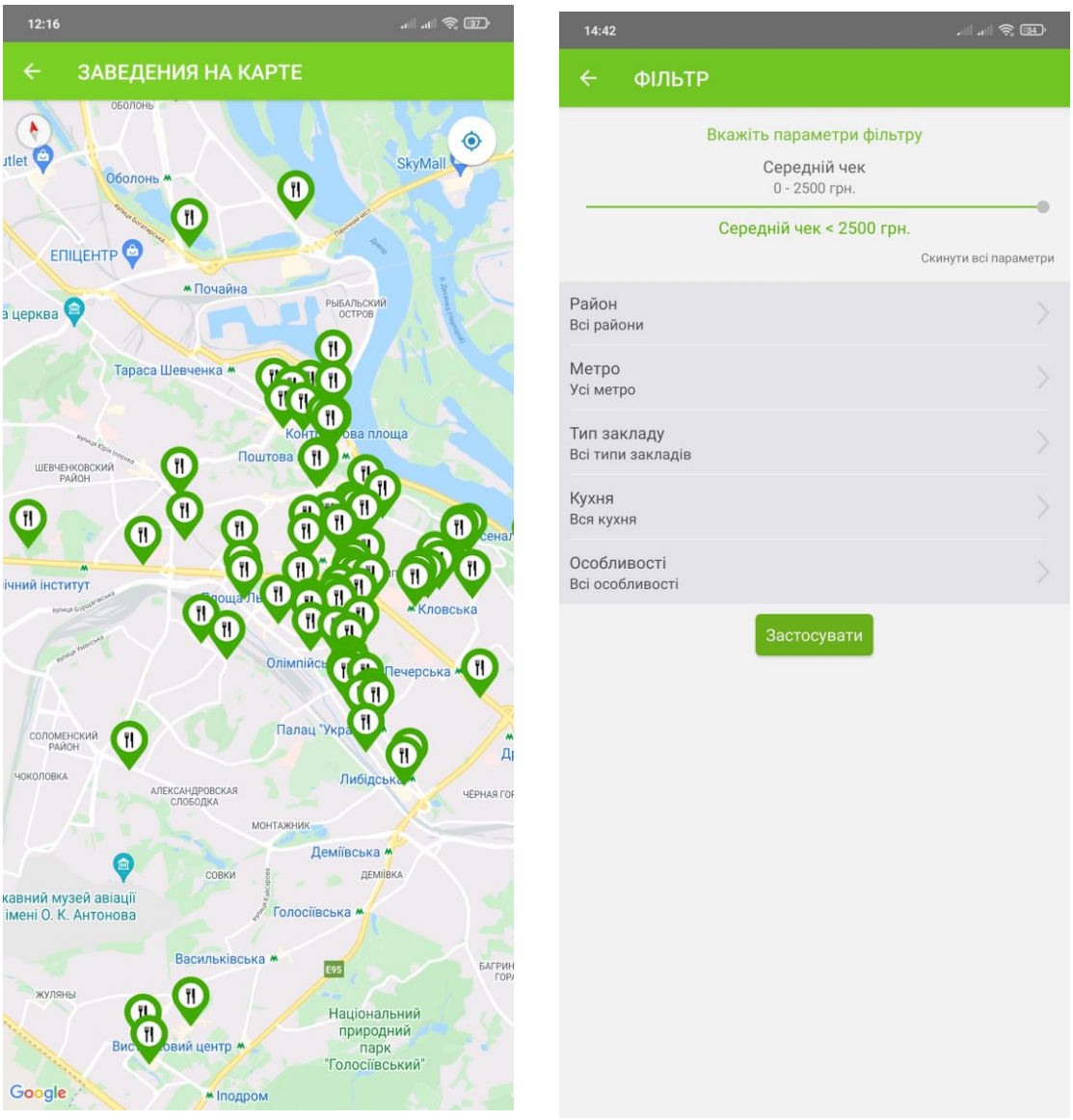

Рисунок 3.1 – Відображення ресторанів на карті та сторінка фільтрації у додатку «RestOn»

Перейшовши у обраний ресторан можна побачити відгуки про нього та його оцінку. Є можливість також поставити власну оцінку та відгук закладу. Також є функція перегляду меню закладу, проте, на жаль, його можна тільки переглядати, вибирати ж з попереднє замовлення з нього немає можливості.

Далі можна перейти до резервації столика, обрати час, кількість гостей, вказати свій номер телефону і додати коментар, рисунок 3.2.

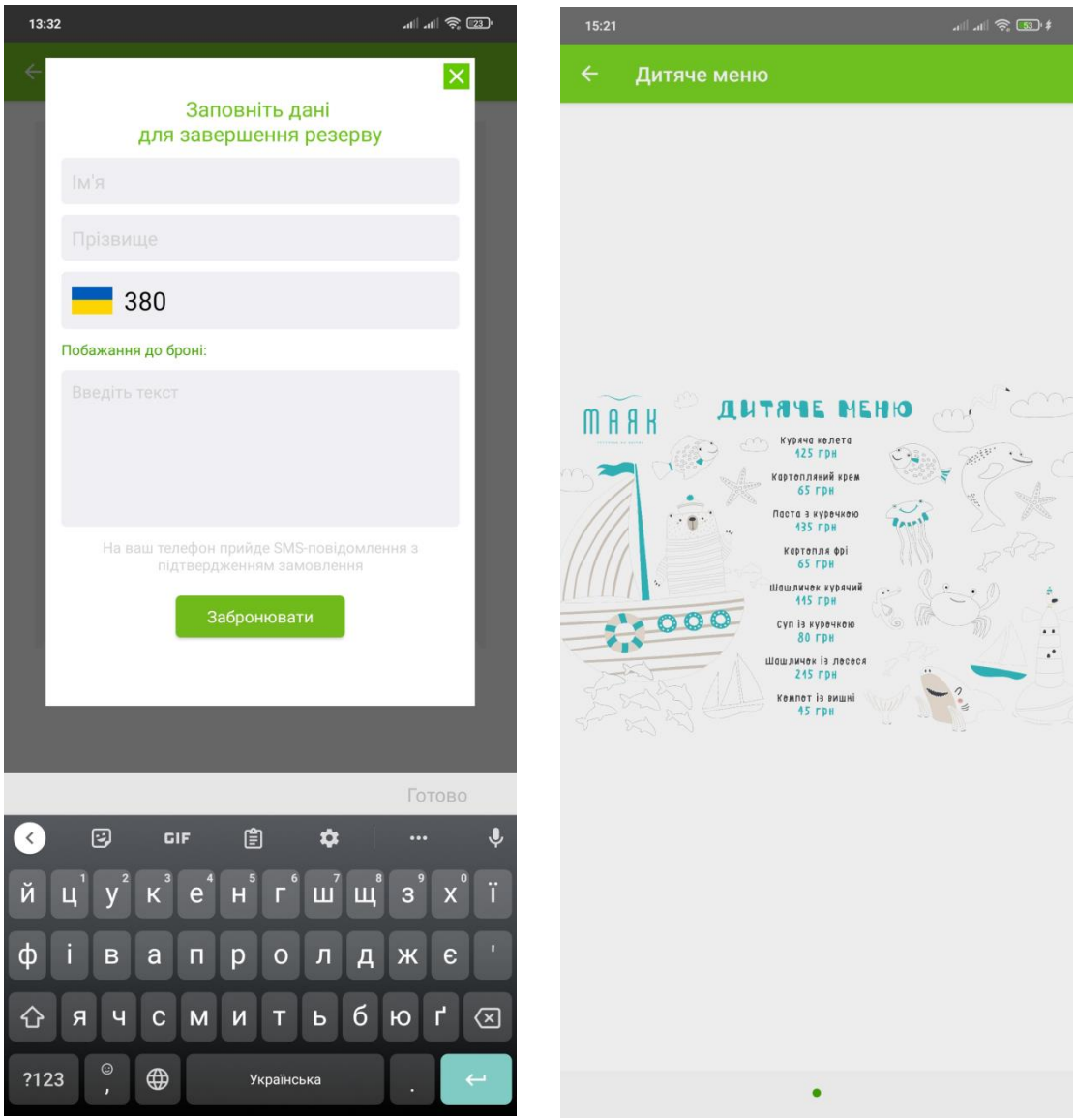

Рисунок 3.2 – Процес бронювання столику та перегляд меню

Проте застосунок має декілька суттєвих мінусів. Як вже було сказано раніше, меню в додатку можна тільки переглядати, але не можна додавати з нього страви у передзамовлення. Звісно, можна додати це у коментарі, але це не дуже зручно.

По-друге, одразу не зрозуміло, на який час столик є вільним, тому ресторан може відповісти відмовою після перегляду заявки бронювання. Також тут не можна візуально обрати столик, який бронюєш. Було б добре мати можливість або побачити схему закладу і після цього обирати стіл, або хоча б мати його фото. І останнє – в застосунку немає оплати онлайн та функції додавання

чайових, тобто якщо у відвідувача немає готівки, то чайові залишити не буде можливості.

3.2 Застосунок «TheFork»

Наступним на розгляді аналогом сервісу бронювання є додаток «TheFork». Насправді він дуже схожий на попередній. В ньому так само присутня мапа усіх закладів поблизу, є просто перегляд ресторанів списком та пошук по назві або місцю розташування. Це можна побачити на рисунку 3.3

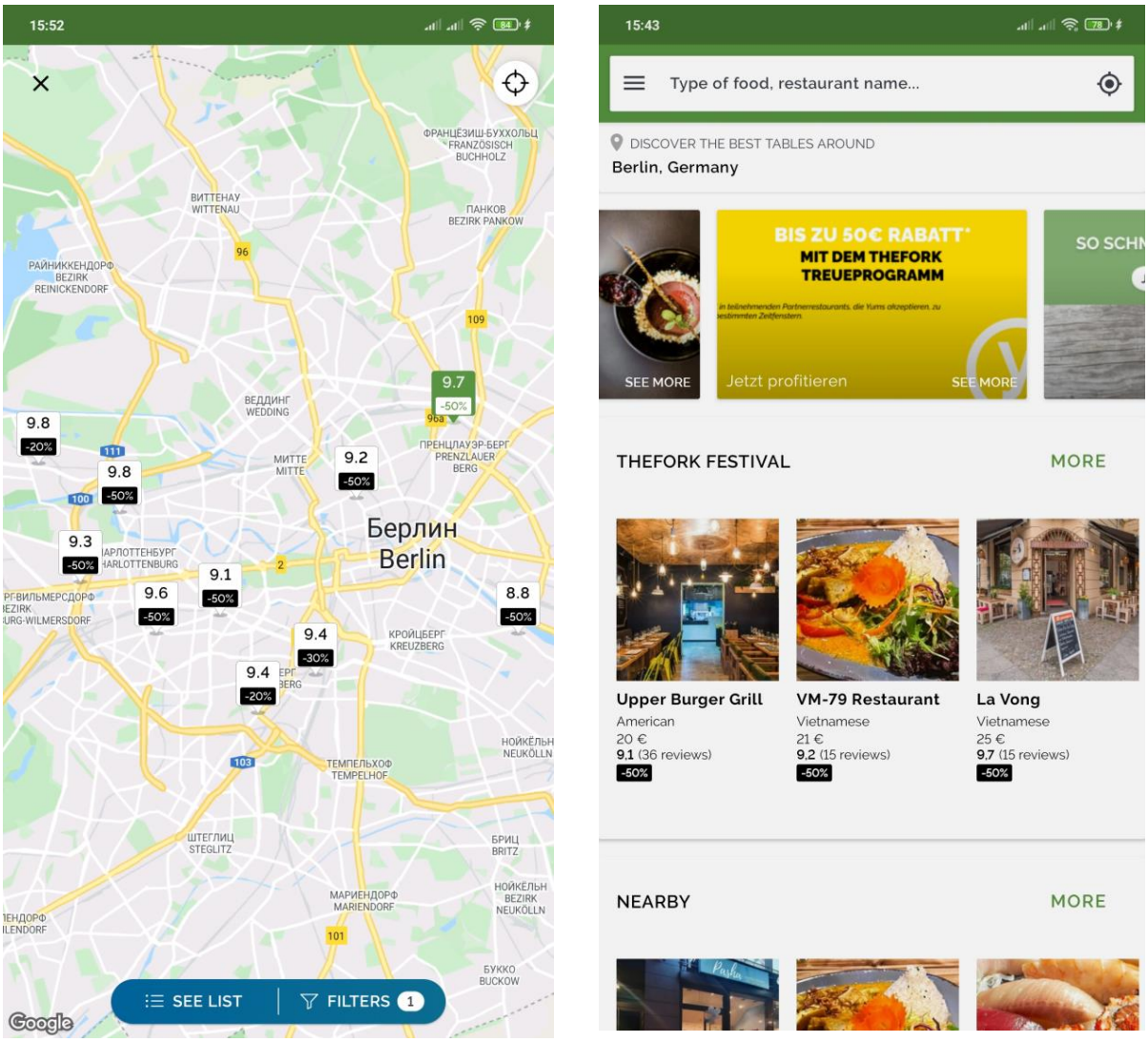

Рисунок 3.3 – Мапа закладів та їх список у додатку «TheFork»

У цьому додатку дещо змінений процес бронювання. Користувачу одразу показує вільні дати, які можна обрати для відвідування. Після того, як користувач обирає дату, йому буде запропоновано вільний час, на який можна замовити резервацію. Проте, юзер вказує лише годину, о котрій він прийде у ресторан, при цьому не вказуючи ймовірний час свого перебування. Це може створити деякі складнощі для управляючих рестораном, адже вони не будуть знати, чи звільниться стіл одного клієнту до часу бронювання іншого. Логічніше зробити можливість обрання користувачем певного проміжку часу, нехай навіть цей проміжок не завжди буде максимально коректним. Це дозволить кухарям, офіціантам та адміністрації краще розпоряджатися часом та ефективніше використовувати свої ресурси. Процес бронювання в даному додатку зображено на рисунку 3.4

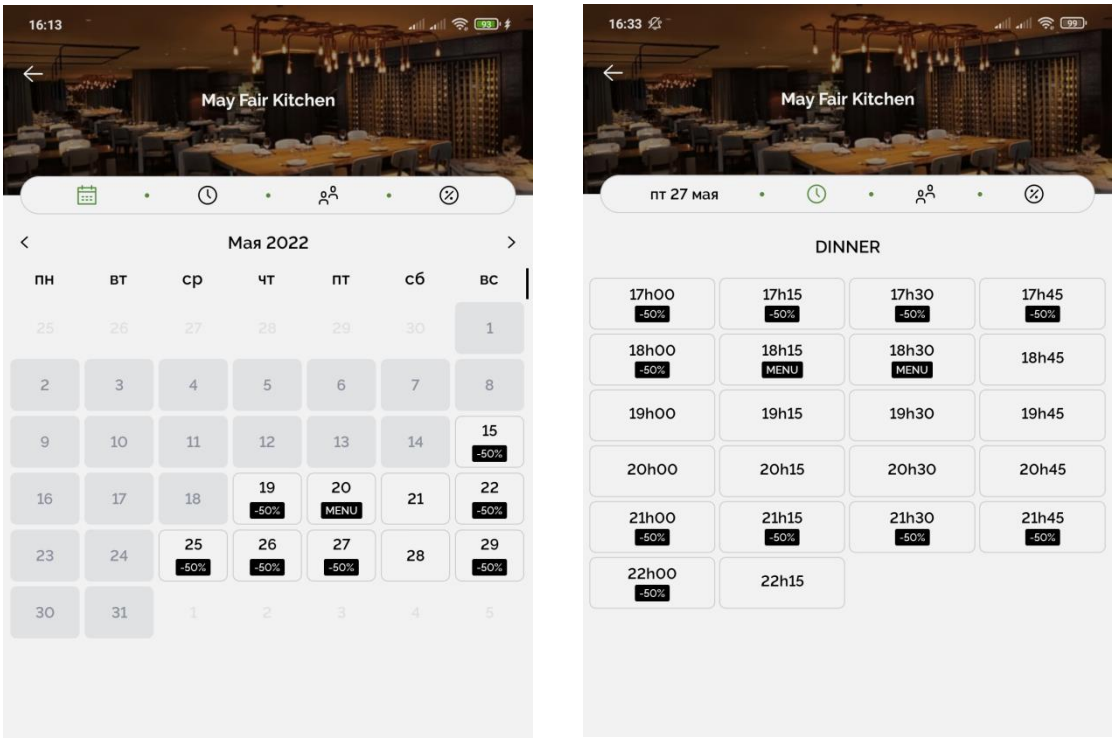

Рисунок 3.4 – Процес бронювання столику

На відміну попереднього додатку, у цьому є плюс: тут присутня програма лояльності. Чим більше юзер буде користуватись за стосунком, тим більше у

нього буде балів, які він зможе використати в якості оплати у закладі. Це буде заохочувати клієнтів повертатись до додатку та активно ним користуватись.

Проте у цьому додатку все ще такі ж проблеми, як і у попередньому

– не можна обрати столик зі схеми або по фото;
– не можна зробити замовлення наперед ;
– немає онлайн оплати та функції для залишання чайових.

3.3 Веб-сайт «OpenTable»

Останнім було розглянуто веб-сервіс для бронювання «OpenTable». Оскільки він адаптований до мобільних пристроїв, його досить зручно використовувати і на телефоні.

На сайті є головна сторінка, на якій можна знайти заклад по назві або по своєму місту. Інтерактивної мапи з розташуванням ресторанів, як в попередніх додатках, немає. Це видно на рисунку 3.5.

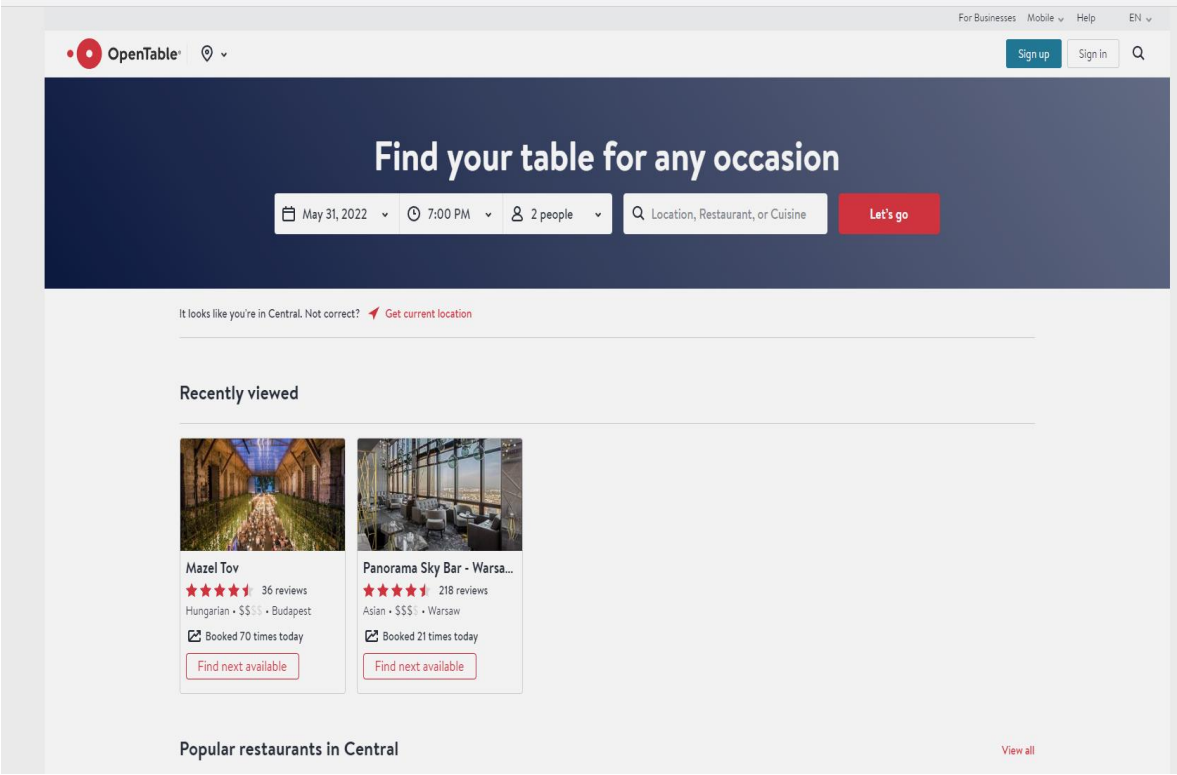

Рисунок 3.5 – Головна сторінка сайту «OpenTable»

На цьому ресурсі є багато параметрів фільтрації, які допоможуть легко знайти потрібний заклад, рисунок 3.6. При резервації одразу перевіряється наявність вільних місць на обрану дату та час. В коментарях до бронювання можна вказати в коментарях певні побажання.

Також на сторінці закладу можна подивитися фото закладу, меню та відгуки відвідувачів.

На жаль, як і у всіх переглянутих додатків , немає інтерактивного вибору місць у залі, можливості замовити заздалегідь та оплатити все онлайн.

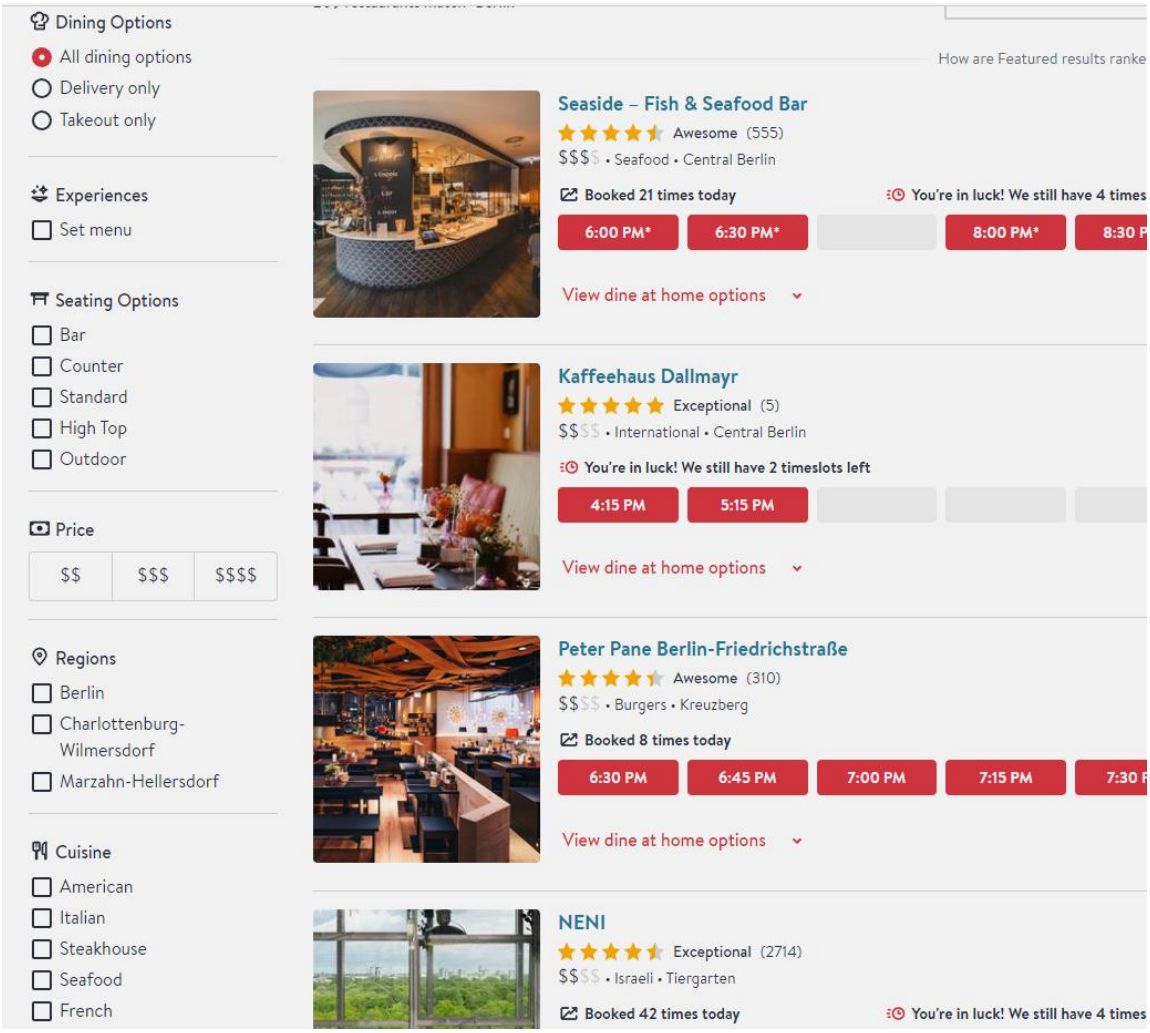

Рисунок 3.6 – Параметри фільтрації для вибору ресторану

Порівняльний опис розглянутих аналогів існуючих систем дозволяє виділити їх основні переваги та недоліки, які показано в таблиці 3.1.

Таблиця 3.1 – Порівняльна характеристика усіх схожих за функціоналом додатків

Назва програмного застосунку	«RestOn»	«TheFork»	«OpenTable»
Платформа	Android OS	Android OS	Веб-застосунок
Користувацький інтерфейс	Не надто зручний інтерфейс. Процес бронювання може бути складним для користувача, оскільки не відображує одразу вільних слотів.	Інтерфейс більш зрозумілий. Чіткий процес бронювання столику, окрім того, що не можна обрати проміжок часу резервації.	Цей сервіс не має інтерактивної карти закладів, на відміну від попередніх. Процес бронювання зрозумілий.
Функціонал для клієнтів	Є карта з відображення ресторанів поблизу. Можна переглянути відгуки, меню	Також має мапу з закладами поблизу. Є зручний процес бронювання, в якому одразу	Сервіс не має графічного відображення кафе поблизу, проте має досить зручні параметри

	закладу та зробити бронювання.	відображені вільні дати та час. Також плюсом є наявність у додатку програми лояльності.	фільтрації. Процес бронювання схожий як у додатку.
Основні функції додатку	Пошук ресторанів з фільтрацією; перегляд карти з закладами поблизу; ознайомлення з відгуками та можливість їх залишити; бронювання столика з можливістю залиши додаткові коментарі.	Перегляд мапи ресторанів поряд; пошук закладів по ключовим словам та місцю розташування; додавання закладу у список обраних; участь у програмі лояльності; бронювання одразу на вільний слот у часі.	Детальні параметри фільтрації для пошуку; перегляд календаря своїх бронювань; бронювання на певну годину з функцію додавання коментаря.

Недоліки	Немає відображення одразу вільних слотів у додатку; не можна зробити замовлення наперед; немає функції онлайн оплати.	Немає можливості зробити бронювання на проміжок часу; немає функції онлайн-платежів; мало категорій фільтрації; не можна зробити замовлення заздалегідь.	Немає карти з відображенням закладів поблизу; відсутня схема розташування столиків у ресторані або їх фотографії; немає функції онлайн оплати та передзамовлення.

В процесі аналізу було виділено основні переваги, які можна отримати від кожної програми, та недоліки, яких слід уникнути. Система «RestOn» має зручну карту з ресторанами поблизу. У додатку «TheFork» зручний варіант бронювання столиків, у якому одразу відображуються доступні та недоступні дати, а також зручний список обраних закладів. У сервісі «OpenTable» широкий вибір фільтрів. Всі ці корисні функції можна використати у проєкті. Також можна додати функціонал, який відсутній у всіх цих застосунках, проте може бути дуже корисним для користувача:

- фотографії столиків або схема ресторану;
- меню з можливістю оформлення перед замовлення;
- оплата через додаток;
- функція для залишення чайових;
- вибір ймовірного часового проміжку проведеного у закладі.

4 АНАЛІЗ ОБРАНОЇ СИСТЕМИ

4.1 Роль інформаційних технологій в процесі бронювання столиків

Ресторани – це сфера обслуговування, у якій край необхідно застосовувати нові технологія. Зважаючи на прискорений темп сучасного життя, коли кожна хвилина на рахунку, багато людей залишаються незадоволеними наданою якістю послуг, які пропонуються при обслуговуванні в ресторанах, а іноді й взагалі залишаються не обслуговуються, коли у закладі немає вільного столику. Інформаційні технології, якими сьогодні користується переважна більшість людства, може допомогти вирішити ці проблеми. Одним із важливих напрямків розвитку технологій громадського харчування є мобільний онлайн-сервіс бронювання столиків із багатьма перевагами:

– мобільний телефон - завжди поряд з користувачем, тому забронювати, наприклад, столик можна навіть під час роботи;

– усі дані про клієнтів та їх бронювання зберігаються не в книзі бронювання, а в єдиному електронному вигляді, де вся інформація чітко структурована;

– можливість обрати столик за планом залу або за його зображенням;

– оплата замовлення онлайн без будь-яких додаткових комісій;

– функція попереднього замовлення, яка економить безліч часу як у кухні, так і у клієнта.

Мабуть очевидно, що з такою кількістю переваг не могло обійтись і без недоліків. Одним з них є ситуація, коли клієнт бронює столик на конкретний проміжок часу, а потім не приходить до закладку, від чого ресторан зазнає тільки збитків.

Проте відомі кухарі ресторанів «Alinea» і «Next» в Чикаго Нік Коконас та Грант Акація знайшли рішення цієї проблеми.. Вони розробили власну систему бронювання Tock, у якій, подібно то процесу придбання квитків до кінотеатру, клієнт бронює столик у закладі одразу сплативши за своє замовлення[1]. Таким

чином, ресторан нічого не втратить у випадку, коли замовник не прийде у свій заброньований час.

Аналогічно цьому рішенню, можна додати й альтернативне. Якщо, до прикладу, користувач не має змоги сплатити все замовлення онлайн або він не знає, які страви буде замовляти, ресторан може стягувати певну суму, яку буде вважати за потрібне, при бронюванні столику, і повертати її назад лише у тому разі, коли клієнт все ж таки прийде до закладу у заброньований час.

Сьогодні в Україні майже 40 відсотків операцій з оплати рахунків в ресторанах і кафе відбувається в безготівковій формі. У найбільших містах частка сягає близька 50%, у регіонах ж трохи менше – біля 30 відсотків[2]. У наш час люди все рідше користуються готівкою. У зв'язку з цим дохід офіціантів почав стрімко знижуватися, оскільки далеко не скрізь є можливість залишити чайові карткою. Тому в даному застосунку повинна бути функція безготівкових чайових, які клієнт зможе залишити через додаток по завершенню свого візиту. Реалізувати це дозволять онлайн платежі. Цінність цієї функції для ресторанів полягає в наступному:

– підвищити дохід офіціантів і зменшити «мобільність» співробітників;

– надати гостям можливість залишити чайові навіть за відсутності готівкою;

– збільшити частку гостей, які активно беруть участь у програмі лояльність.

Однією з останніх новинок у ресторанній справі є електронне меню. Під час пандемії майже всі ресторани замінили свої паперові меню на електронні, доступ до яких клієнт може отримати зчитавши QR-код. Електронне меню – інтерактивна система відображення списку страв, яка є альтернативою сучасному паперовому меню. Воно, як і звичайне, містить повну інформацію про список їжі та напоїв , але доступ до нього можна отримати з будь-якого пристрою. Представлення меню у цей спосіб є зручнішим по декільком причинам:

– з гігієнічної точки зору електронна версія є кращим, оскільки людині не треба торкатися паперового меню, яке до тебе брали невідомі люди;

– звичайне меню втрачає товарний вид з часом, тому власникам закладу доведеться його змінювати та витрачати на це додаткові кошти. Використовуючи ж електронне в якості альтернативи цього можна уникнути. Також це важливо з екологічної точки зору, оскільки це значно економить папір, а отже і природні ресурси;

– не треба очікувати, поки офіціант звільниться і принесе меню, можна самому відкрити його з будь-якого місця та девайсу на системі Android;

– зважаючи на маркетинг, електронне меню – це хороший інструмент для залучення більшої кількості клієнтів, оскільки всі хочуть користуватись зручним сервісом;

– з точки зору автоматизації, якщо в додатку є електронне меню та функція попереднього замовлення, то це може зменшити навантаження персоналу та краще структурувати виконання роботи кухаря.

На часі популярними є програми лояльності у різних закладах, будь то кінотеатри, продуктові магазини чи магазини одягу, або ж ресторани. Системи лояльності є одним із важелів ресторанного маркетингу. Це не творить чудес і не збільшить кількість відвідувачів удвічі, проте лояльність клієнтів просто забезпечує емоційну прихильність до вашого закладу і мотивує їх знову і знову повертатися до ресторану . Це надасть можливість поступово збільшувати свій прибуток і стати популярнішими на ринку.

Завдяки додатку електронного бронювання столиків можна з легкістю реалізувати систему лояльності для будь-якого ресторану, оскільки всі дані у електронному варіанті і ними просто оперувати.

4.2 Оптимізація бізнес-процесів закладу шляхом розробки додатку

Багато компаній усвідомлюють, що зміни мають вирішальне значення в тому, аби залишатися на плаву, розвиватися та не втрачати актуальність, але не

всі знають, якими ці зміни мають бути, яку ланку ланцюга слід замінити або видалити, щоб отримати максимальну віддачу.. Тільки компанії, які мають чітке уявлення про те, який продукт або результат цікавлять споживачів і в якій конкретній формі, можуть знайти найкращий шлях для досягнення своїх цілей. Це вимагає креативного мислення, сміливості, і, що ще важливіше, бажання та можливість вкладати гроші в проєкти, оскільки зміна варіантів управління часто вимагає значних інвестицій.

Підприємці прагнуть замінити дорогу робочу силу автоматизацією, механізацією та електронікою. Проте автоматизація процесів не передбачає простий механізм заміни людей, а поступово визначає рутинні, повторювані операції в будь-якій професії та переводить їх на електронне обладнання. Щодо сфери обслуговування ресторан - це те, що виділяє його серед інших закладів високим рівнем обслуговування та приготуванням складних страв. Як і будь-який інший бізнес, який активно розвивається, робота з адаптації всіх бізнес-процесів в ресторані дуже важлива, особливо прийом і обробка замовлень. Впровадження електронних меню дозволить повністю автоматизувати замовлення з застосунку на належному рівні, уникнути помилок офіціантів та лишніх витрат, пов’язаних з виправленням цих недоліків, що гарантує:

- точність обслуговування;
- зростання іміджу ресторану;
- підвищення конкурентоспроможності;
- зменшення навантаження на працівників.

До того ж не потрібно завантажувати окремо додатки кожного ресторану (якщо окремий ресторан взагалі має його), щоб оплатити замовлення безготівковим розрахунком, оскільки в цій системі є безліч ресторанів. Всі дані з застосунку зберігаються в хмарному сховищі. Це гарантує безпеку всієї інформації та транзакцій, які відбуваються в системі, а також дозволяє легко оперувати даними для різного типу аналізу систем та впровадження нового функціоналу.

5 ОРГАНІЗАЦІЯ ПРОЦЕСУ РОЗРОБКИ

Сьогодні важко уявити роботу в будь-якій ІТ-команді без сучасних інструментів управляння. Команди на деяких проєктах можу сягати десятків, а то і сотень людей, а сам проєкт зазвичай поділений на менші функціональні частини. Тому для таких проєктів завжди потрібно використовувати певні системи моніторингу та управління задачами, аби наладити бізнес-процеси, економити час та ресурси.

Під час написання даного індивідуального дослідницького проєкту було використано інструмент організації робочого процесу Jira. Це система керування проєктами, яка має доволі багатий функціонал: створення задач, вибір виконавців, встановлення пріоритетів та відстеження виконання вказаних задач. Jira є дуже популярною серед agile-команд, команд керування проєктами та розробки програмного забезпечення, для DevOps команд та менеджерів задач [3].

Оскільки процес написання проєкту є достатньо об’ємним та кропітким, такий функціонал Jira, як управління задачами, може забезпечити оптимізацію роботи та якісні результати у кінці.

По-перше, змога створювати команди у цій системі значно полегшить комунікацію студента та його дипломного керівника. Долучившись до студента викладач зможе спостерігати за задачею, відстежувати її виконання та отримувати сповіщення після завершення певного етапу.

По-друге, змога декомпозувати задачу, тобто розбити її на підзадачі, допоможе розбити весь об’єм роботи на зручні частини та відслідковувати її виконання. Переглянути стан та статус кожної задачі можна на дошці керування. На рисунку 5.1 зображено інтерфейс дошки керування в Jira.

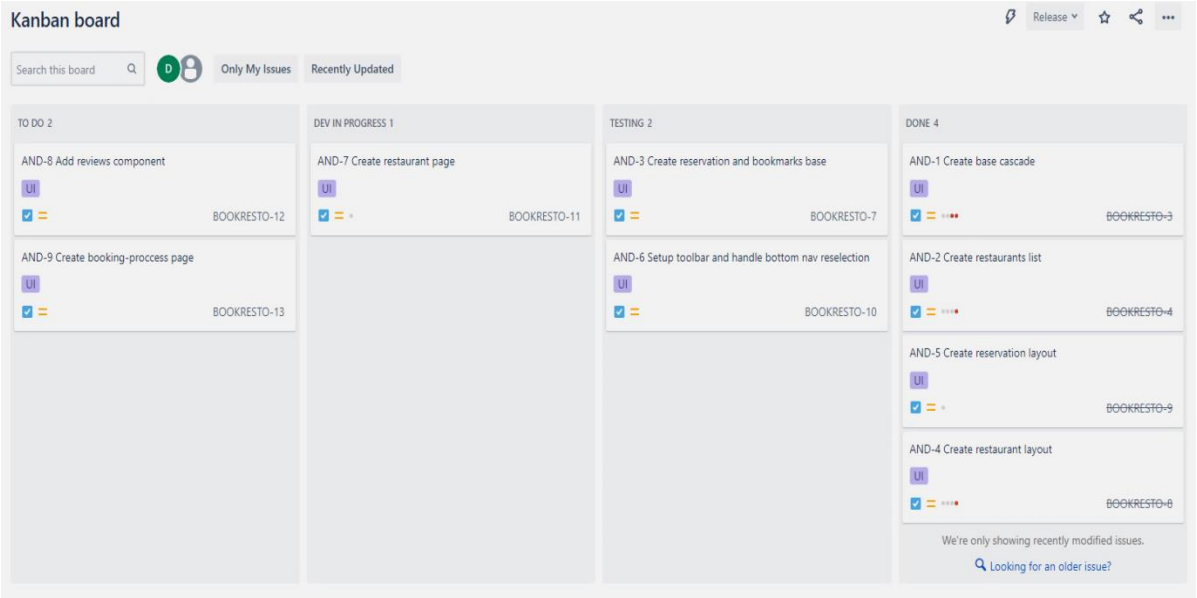

Рисунок 5.1 - Інтерфейс дошки керування в Jira

Дошка умовно поділена на чотири групи: задачі на виконання, задачі у процесі виконання, тестування та вже виконані задачі. Кожна група відповідає статусу задачі.

Для даного проєкту було обрано дошку Kanban. Kanban – це гнучка методологія управління проєктами, яка є однією з найбільш популярних на чолі зі Scrum.

Суть Kanban полягає у досягненні максимально ефективної роботи за рахунок візуалізації завдання та обмеження об’єму незавершеної роботи. Kanban-команди намагаються якомога краще скоротити час виконання задачі від початку до кінця. Метою цієї методології є досягнення виконання задачі [4].

Методологія Scrum, у свою чергу, базується на використанні певних проміжків часу, за які має бути створена основна частина програмного забезпечення. Ці проміжки називаються спринтами і тривають до двох тижнів. Scrum-команди використовують спеціальні ролі, використовують особливі артефакти та проводять регулярні зустрічі під час спринт-циклу для ефективного перебігу роботи. Метою Scrum-методології є вчасне завершення спринту [4].

Оскільки успішне виконання проєкту залежить від послідовного виконання задач та їх підзадач, було прийняте рішення використання гнучкої методології Kanban для його завершення.

На рисунку 5.2 зображено інтерфейс Roadmap у системі Jira.

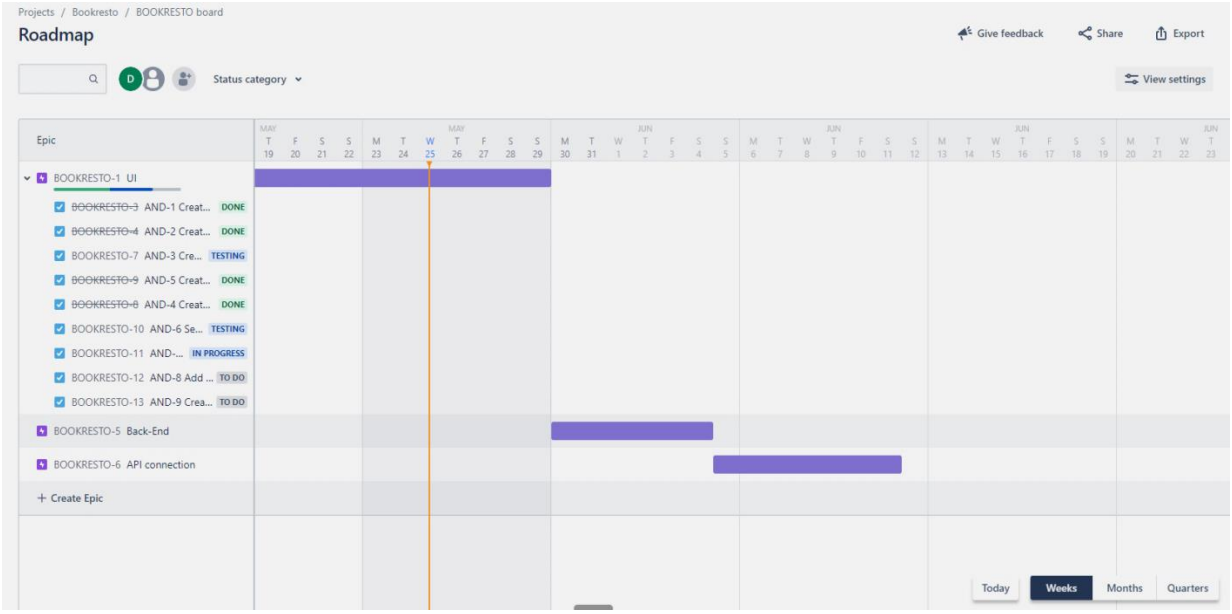

Рисунок 5.2 - Інтерфейс Roadmap у системі Jira

Roadmap складається з об’ємних задач, які ще називають епіс. На рисунку можна бачити 3 великі задачі, що відносяться до UI частини, Back-End частини та API connection частини. Зазвичай епіс декомпозують на підзадачі, вказавши при цьому їх приналежність до головної задачі. Процес та час виконання кожної під задачі можна відстежити використовуючи дошку календаря у вигляді дня, тижня, місяця, кварталу тощо.

У таблиці порівняння методологій Scrum і Kanban.

Таблиця 5.1- Порівняння методологій Scrum і Kanban

Методологія	Scrum	Kanban
Походження	Розроблення програмного забезпечення	Бережливе виробництво
Ідеологія	Навчання за рахунок отримання досвіду,	Використання візуальних елементів для покращення

	самоорганізація, визначення пріоритетів та аналіз перемог та поразок для постійного вдосконалення	незавершеної роботи
Каденція	Регулярні спринти фіксованої довжини (два тижні)	Безперервний потік
Практики	Планування спринту, спринт, огляд спринту, ретроспектива спринту	Візуалізація ходу роботи, обмеження незавершених робіт, керування потоком, застосування циклів зворотного зв’язку
Ролі	Власник продукту, скрам-майстер, команда розробників	Немає обов'язкових ролей

Ще одним інструментом організації робочого процесу є система контролю версій Git. Контроль версій – це принцип відслідковування змін програмного коду та його керування. Системи контролю версій (VSC) – це програмній інструменти, котрі використовуються розробниками для управління змінами в початковому варіанті коду з часом. З ходом розвитку й ускладнення середовищ розробки вони допомагають розробникам працювати швидше та ефективніше.

ПЗ контролю версій відслідковує всі зміни, які вносяться, у спеціальній базі даних. Якщо було знайдено помилку, розробник може повернутися до більш ранніх версій коду для виправлення помилок. Це допоможе мінімізувати проблеми усіх учасників команди.

Хороше програмне забезпечення для керування версіями підтримує бажаний робочий процес розробника і не нав’язує певний спосіб роботи. Воно також може працювати на будь-якій платформі та будь-якій операційній

системі. Якісні VCS забезпечують розробнику неперервний процес розробки та внесення змін у коді, уникає громіздких блокувань файлів, що спрощує роботу усій команді.

Переваги систем контролю версій:

1. Повна історія змін кожного файлу проєкту за тривалий час. До змін можна віднести створення та видалення файлів та редагування їх вмісту. Різноманітні інструменти VCS відрізняються якістю операції перейменування та переміщення файлів. До історії також мають належати деталі про автора, дата та коментар з описом цілі кожної зміни. Наявність повної історії змін дає можливість повертатися до попередніх версій з метою аналізу основних причин виникнення помилок та їх вирішення. Якщо ПЗ перебуває в активній фазі розробки, то «старою версією» можна вважати майже весь код цього проєкту. На рисунку 5.3 зображено вигляд історії комітів проєкту.

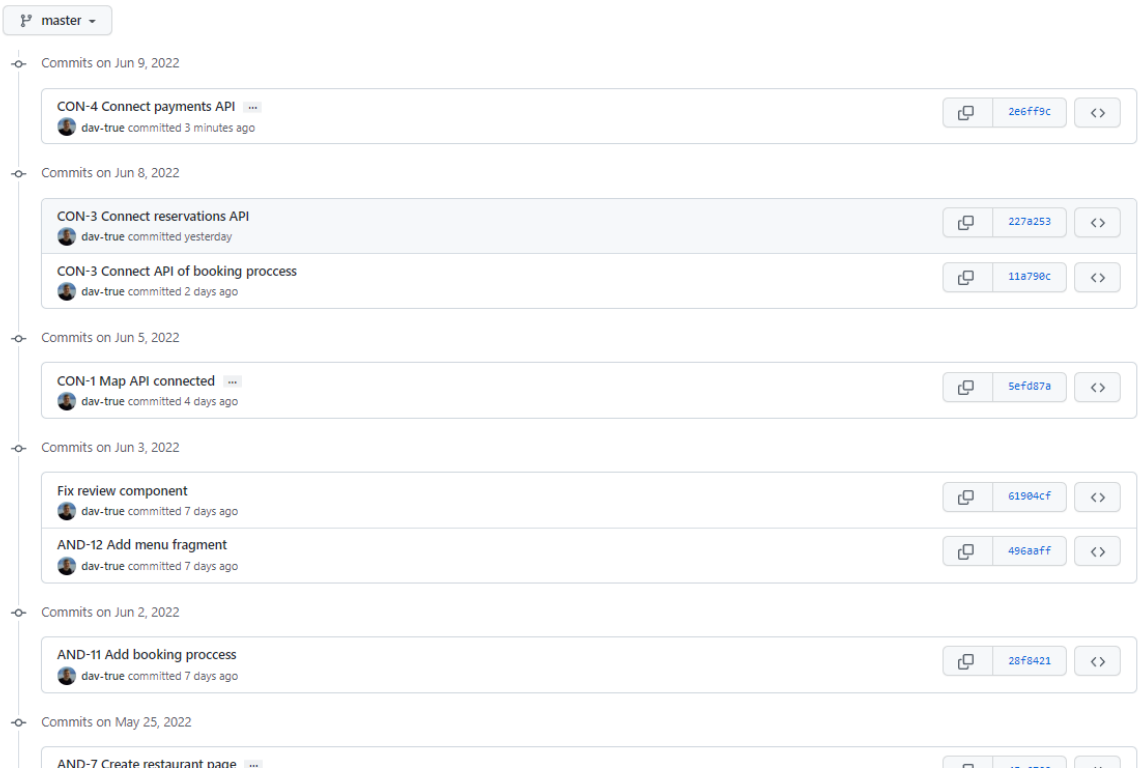

Рисунок 5.3 - Вигляд історії зміни коду в проєкту

2. Гілки та злиття. Створення «гілок» в інструментах VCS надає можливість мати кілька незалежних один від одного напрямків розробки , а також виконувати їх злиття. Таке поєднання дозволяє розробнику перевіряти внесені зміни та бачити, що вони не конфліктують між собою. Часто створюють окремі «гілки» для кожної функціональності або релізу. Наявність багатьох варіацій робочих процесів дозволяє команді обирати найбільш зручний для неї спосіб використання гілкування і злиття у VCS.

3. Відстеження. Можливість відслідковувати кожну зміну в ПЗ і поєднувати його з ПЗ керування проєктами та відстеження помилок, наприклад Jira, а також функція збереження коментаря з описом цілі змін може допомогти під час аналізу основних причин виникнення помилок тощо. Історія з використанням коментарів під час прочитання коду полегшує процес його аналізу.

Загалом, розроблення ПЗ можлива і без систем контролю версій, але такий підхід має ряд ризиків, що слід враховувати. Таким чином питання не в тому, чи використовувати VCS, а в тому, яку саме систему обрати [5].

Висновки до розділу

У цьому розділі описані головні інструменти організації робочого процесу, які були застосовані під час написання даного індивідуального дослідницького проєкту, а саме: Jira та Git. Під час використання системи Jira було обрано методологію Kanban для розробки ПЗ. Цей вибір аргументовано перевагами даної методології та порівняно її з іншою не менш популярною методологією Scrum у вигляді таблиці. Далі було описано технічну систему розробки програмного забезпечення, як Git. Наведено загальне призначення системи контролю версій та її переваги.

6 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

6.1 Засоби розробки

Для реалізації даної інформаційної системи було використано наступний стек технологій:

- Android Studio;
- Android SDK;
- MVVM;
- Kotlin;
- JavaScript;
- Node.js;
- API
- XML;
- MySQL.

Android Studio – офіційне інтегроване середовище розробки для розробки Android додатків. Воно базується на InteliJ IDEA, інтегрованому середовищі розробки на мові Java, та має вбудовані інструменти для процесу розробки редагування коду. Android Studio використовує систему збірки на основі Gradle, шаблони коду, емулятор девайсу та інтеграцію з Github для підтримки розробки програм у операційній системі Android. Також тут застосовується функція Instant Push, що надсилає зміни ресурсів та коду в запущену програму. Редактор коду, у свою чергу, пропонує розробнику доповнення та аналіз коду. Програми, що були створені в цьому середовищі, компілюються у формат APK для майбутнього використання в Google Play Store. Запуск цього ПЗ вперше було анонсовано в травні 2013 року, а сама збірка була випущена вже в 2014. Середовище Android Studio доступне для стаціонарних платформ Windows, Mac і Linux. Як основу для розробки додатків Android, Android Studio витіснило Eclipse Android Development Tools (ADT) [6].

Android SDK – це набір інструментів розробки, що використовуються під час розробки програмних забезпечень для платформи Android. Android SDK має:

- бібліотеки;
- дебагер;
- емулятор;
- документація для програмних інтерфейсів (API) Android;
- зразок вихідного коду;
- посібники для операційної системи Android [7].

MVVM (Model-View-ViewModel) – це шаблон проєктування ПЗ, який використовуються для розділення логіки програми та елементів управління користувацького інтерфейсу. MVVM раніше був відомий, як model-view-binder, та був розроблений архітекторами Microsoft Кеном Купером та Джоном Госсманом [8].

Шар Model відображає дані та бізнес-логіку Android програми. До нього входять бізнес-логіка, віддалене та локальне джерела даних, класи моделі та репозиторій.

Рівень View складається з коду користувацького інтерфейсу (фрагмент, діяльність) та XML. Він відправляє користувачу ViewModel, але не отримує відповідь безпосередньо. Для отримання відповіді View має підписатися на спостережувані елементи, надані йому ViewModel.

ViewModel, у свою чергу, виступає зв’язувальним компонентом між рівнями представлення та моделлю, тобто бізнес-логіки. Він не знає, яким саме компонентом View буде використовуватися, оскільки не має прямого посилання на View. З цього випливає, що ViewModel не має знати про View, з яким буде взаємодіяти. Він взаємодіє з моделюю і відкриває те, що буде спостерігатися за допомогою View [9].

Kotlin - це сучасна мова програмування статичного типу, що швидко розвивається та була створена компанією JetBrains. Ця мова як об'єктно-орієнтоване, так процедурне програмування та використовується понад 60%

професійних розробників Android. Kotlin дозволяє створювати кросплатформний код, який буде застосовуватися на будь-яких платформах. Наприклад, веб-додаток, серверні та браузер-клієнтські застосунки, десктопні застосунки [10].

JavaScript – динамічна мова програмування, яка застосовується для розробки веб-застосунків, ігор тощо. JavaScript дозволяє реалізувати динамічні функції, які неможливо зробити лише за допомогою HTML і CSS. Більшість браузерів використовують цю мову програмування у якості мови сценаріїв для створення динамічних об’єктів на веб-сторінках [11].

Node.js – це платформа розробки з відкритим вихідним кодом для роботи з кодом JavaScript на стороні сервера. Node.js корисний для створення за стосунків, що вимагають підключення браузера до сервера. Також вона часто застосовується у додатках режиму реального часу, наприклад чат за стосунки, новини та веб push повідомлення [12].

API (Application Programming Interface) – це інтерфейс прикладного програмування, що є набором визначень та протоколів для створення й інтеграції прикладного ПЗ [13].

XML (Extensible Markup Language) – це розширювана мова розмітки. Мова розмітки – це набір тегів та коду, який описує текст цифрового документу. Найвідомішою мовою розмітки є мова HTML, яка використовується для форматування веб-сторінок. XML є більш гнучкою і дозволяє оперувати складними бізнес-операціями на веб-сторінках [14].

MySQL – це система керування реляційною БД, яка заснована на мові структурних запитів SQL. MySQL застосовується для багатьох цілей, включаючи сховища даних, реєстраційні програми та електронну комерцію. Але найбільш поширеним використанням цієї системи є створення веб-баз даних [15].

7 РОЗРОБЛЕННЯ ДІАГРАМ ПРОГРАМНОГО ДОДАТКУ

7.1 Діаграма компонентів

Діаграма компонентів представлена на кресленику ІА82.230БАК.003 Д1. На цій діаграмі показано взаємодія між такими основними елементами, як мобільний застосунок, сервер та база даних.

Першим найголовнішим компонентом є сам мобільний застосунок, написаний під операційну систему Android. Мобільний застосунок можна умовно поділити на модулі, у даному випадку це:

- модуль авторизації;
- модуль ресторану;
- модуль процесу бронювання;
- модуль резервацій.

Перша частина відповідає за авторизацію юзера у системі. Користувач або входить, або в реєструється у додатку, після чого отримує спеціальний токен від серверу. Тільки таким чином він зможе отримати доступ до інших функцій застосунку.

Друга частина відповідає за розміщення ресторанів на карті та у списку, а також за фільтрацію закладів по різного виду параметрам. Також в цей модуль входить реалізація фрагменту, який відображає сторінку ресторану, на якій є можливість переглянути та залишити відгуки.

Наступний рівень включає в себе проведення операції бронювання. Туди входить вибір насамперед дати та часу резервації. Далі за фото користувач обирає столик, який його задовольняє. Уже після цього юзера буде перенаправлено на сторінку вибору меню. В кінці даного процесу клієнт матиме змогу оплатити своє замовлення.

Останній модуль цієї компоненти відповідає за список бронювань у додатку. Він дозволить користувачу переглядати всі минулі та наявні резервації, а також їх деталі: меню, час, суму до сплати.

Наступною компонентою цього програмного забезпечення є безпосередньо зовнішній сервер, який буде виконувати функцію API. Додаток Android за допомогою інтерфейсів та HTTP-client буде надсилати запити до серверу. Ці запити будуть оброблятися фреймворком Express JS, після чого відповідь направлятиметься назад у додаток.

При обробці запитів сервер повинен десь брати та зберігати дані, тому остання компонента, а саме база даних MySQL, буде відповідати за їх збереження.

7.2 Діаграма варіантів використання мовою UML

Діаграма використання - це діаграма, що ілюструє функціональність системи, або іншими словами, показує, що система буде робити під час своєї роботи. Діаграма використання — це початкове концептуальне уявлення або концептуальна модель системи в процесі проєктування та розробки. Основна мета діаграми полягає в тому, що система представлена як набір сутностей або акторів, які за допомогою варіантів використання взаємодіють із програмою. У цьому контексті актор або учасник — це будь-яка сутність, яка взаємодіє з системою ззовні.

Діаграма варіантів використання програмного застосунок представлена на кресленику ІА82.230БАК.003 Д2. На ній зображено одного актора - користувача. Після того, як юзер авторизувався, увійшовши або зареєструвавшись у додатку, в нього з'являється можливість переглядати мапу з ресторанами поблизу, або ж доступні заклади зі списку. У списку є додаткова можливість перейти на сторінку фільтрів та обрати параметри фільтрації, такі як тип кухні, мінімальний рейтинг та назву. Також він може перейти у будь-який ресторан та подивитися його деталі, а саме фотографії, рейтинг, адресу, цінову політику, відгуки, меню. Обравши підходяще місце для обіду, користувач переходить до процесу бронюванню. Спочатку він обирає дату та час початку, а також тривалість відвідування. На основі отриманих даних

система вибирає вільні столики і відображає їх юзеру. Після обрання місця, йому буде запропоновано зробити передзамовлення, проте цей крок можна пропустити. Після успішного бронювання користувач може одразу оплатити замовлення картою, або ж може перейти у список резервацій і вже там зробити сплату.

У списку бронювань відображені всі резервації користувача з часом, датою, номером стола, його фотографією, адресою та назвою ресторану. Далі можна перейти у деталі кожного окремого бронювання, де буде також наявна інформація про замовлення, статус оплати та кількість залишених чайових. Якщо замовлення не було оплачене раніше, можна перейти на сторінку оплати. Те ж саме стосується чайових.

Також в додатку є можливість додавати заклад у обране. Список обраних ресторанів буде відображений на відповідній сторінці. З цього списку можна одразу перейти у заклад та виконати бронювання.

Останньою функціональною сторінкою у додатку є профіль юзера. Там відображена його фотографія, електронна адреса та список кредитних карток. З цієї сторінки також можна перейти на фрагмент додавання карток, і прикріпити свою картку до застосунку, що дозволить не вводити її дані кожного разу при оплаті замовлення, а обирати з доданих.

7.3 Діаграма послідовності мовою UML

Діаграми послідовності є одним із типів діаграм, розроблених на мові моделювання UML. Діаграма послідовності показує зв’язок між об’єктами, розташованими в часі. Цей тип діаграми показує задіяні об’єкти та порядок процесів.

Основна задача програмного забезпечення — бронювання столиків у ресторані за допомогою додатку і внесення інформації про резервацію в базу даних , але для початку користувачу необхідно пройти авторизацію, після якої

він отримає спеціальний токен, який буде використовуватись у запитах, де необхідні дані юзера.

Діаграма послідовності програмного додатку зображена на кресленику ІА82.230БАК.003 Д3. В даній діаграмі показано послідовність дій клієнта від авторизації до оплати. Після того, як користувач авторизувався, він обирає ресторан з запропонованих. Далі він переходить на сторінку закладу і починає процес бронювання. Після закінчення даного процесу в нього з'являється можливість оплатити своє замовлення, якщо воно було зроблене. Згодом нове бронювання відображується у списку.

Оскільки процес бронювання є достатню комплексним, було розроблено його окрему діаграму послідовності. Вона зображена на рисунку 7.1.

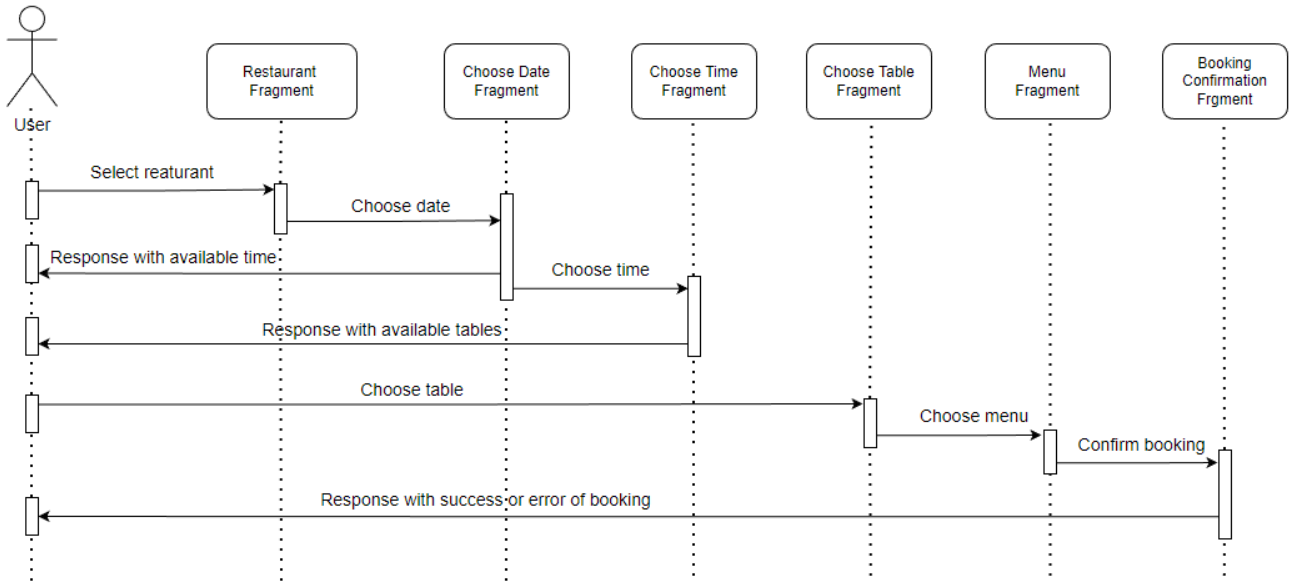

Рисунок 7.1 – Діаграма послідовності процесу бронювання

На цій діаграмі видно, що процес бронювання починається з вибору закладу. Далі користувач обирає дату, час початку та тривалість відвідування, після чого сервер перевіряє наявність вільних столиків з цими параметрами. Якщо столики наявні, то сервер відправляє їх користувачу, якщо ж ні, то він надішле відповідне повідомлення, після чого юзеру потрібно буде обрати інші параметри для бронювання.

Після успішного обрання столика буде запропоновано обрати страви для передзамовлення за бажанням. Далі потрібно підтвердити замовлення і оплатити його одразу або пізніше.

Також, для кращого розуміння процесу авторизації було розроблено його детальну діаграму діяльності. Її можна побачити на рисунку 7.2

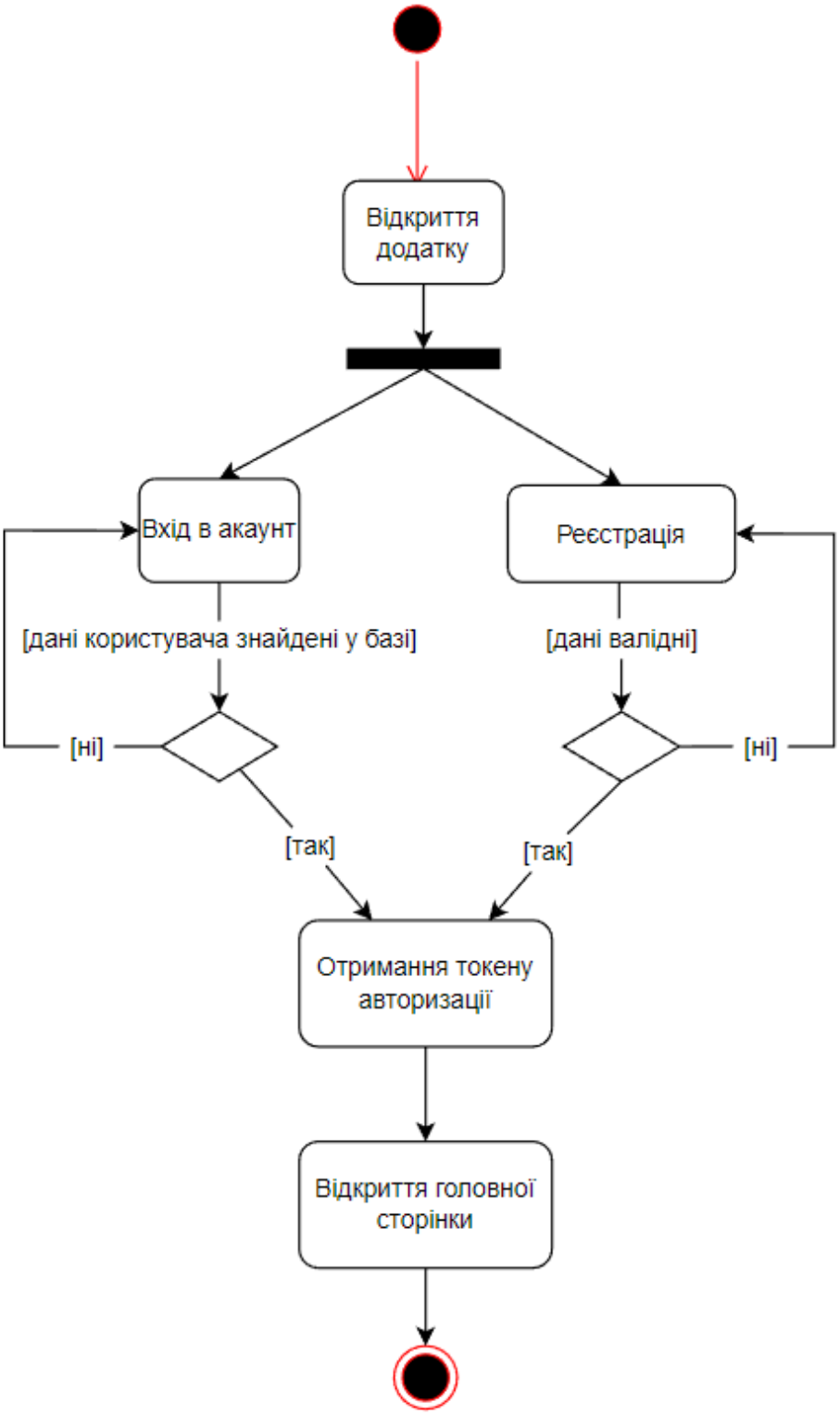

Рисунок 7.2 – Діаграма діяльності процесу авторизації

На ній видно, що спочатку користувач може авторизуватись за допомогою функції входу, якщо він уже був зареєстрований раніше. Для цього йому потрібно ввести валідну електронну адресу і пароль. Якщо ж користувача не знайдено у базі даних, він отримає помилку і не зможе авторизуватися. Проте юзер може перейти на сторінку реєстрації, де йому потрібно буде вказати електронну адресу та пароль, двічі. Якщо дані введені правильно, користувач зможе пройти авторизацію. У противному разі сервер відправить у відповідь помилку і вона буде відображена у застосунку.

У процесі авторизації користувач передає свою електронну адресу та пароль, які на стороні серверу шифруються у спеціальний токен і надсилаються користувачу у відповідь. Цей токен надалі можна використовувати у запитах, які потребують дані клієнта, оскільки сервер може розшифрувати їх з цього токену. Такий спосіб є зручним, оскільки не потрібно кожного разу надсилати повний список даних юзера, а також безпечним, тому що навіть перехопивши токен, зловмисники не зможуть його розшифрувати та витягнути звідти персональні дані.

7.4 Діаграма класів додатку

Діаграма класів описує структуру мобільного додатку і зображує більшість його класів, такі як:

- BaseFragment;
- MainActivity;
- HomeFragment;
- MapFragment;
- RestaurantsListFragment;
- HomeViewModel;
- RestaurantsListDomain;
- RestaurantListApiService;

- RestaurantsPagingSource;
- RestaurantsPagingAdapter;
- ReservationsFragment;
- ReservationDetailsFragment;
- ReservationsViewModel;
- ReservationsDomain;
- ReservationsApiService;
- ReservaitionsPagingSource;
- ReservationsPagingAdapter;
- AuthorizationsActivity;
- AuthViewModel;
- AuthDomain;
- AuthApiService;
- LoginFragment;
- RegistrationFragment;
- WriteReviewFragment;
- TipsFragment;
- ProfileFragment;
- ProfileViewModel;
- ProfileDomain;
- ProfileApiService;
- AddCardFragment;
- PaymentFragment;
- PaymentViewModel;
- PaymentDomain;
- PaymentApiService;
- MenuFragment;
- BookmarksFragment;
- BookmarksViewModel;

– BookmarksDomain;
– BookmarksPagingSource
– BookmarksApiService;
– RestaurantFragment;
– RestaurantViewModel;
– RestaurantDomain;
– RestaurantApiService;
– BookingApiService;
– SearchFilterFragment;
– ChooseMenuFragment;
– ChooseTimeFragment;
– ChooseDateFragment;
– ChooseTableFragment;
– BookingConfirmationFragment;
– BookingContainer;
– BookingViewModel;
– BookingDomain;
– BookingApiService.

Опис функцій та призначення основних класів застосунку наведено у таблиці 7.1.

Таблиця 7.1 – Опис функцій та призначення основних класів застосунку

Клас	Опис
AuthorizationActivity	Клас є лаунчером додатку, тобто з нього починається робота застосунку. Якщо користувач уже раніше виконав вхід у додаток, він буде одразу перенаправлений на MainActivity. AuthorizationActivity слугує контейнером для LoginFragment та

	RegistrationFragment і дозволяє переключатися між ними.
LoginFragment	Цей фрагмент відображає юзеру вікно входу, яке включає в себе поле для вводу електронної адреси та поле для вводу паролю. При натисканні на кнопку входу дані з полів відправляються на сервер за допомогою AuthViewModel, і у відповідь буде отримано або токен авторизації, або помилку за умови відправки некоректних даних для входу.
RegistrationFragment	У цьому фрагменті користувач має можливість зареєструвати новий обліковий запис. Дані відправляються до серверу через AuthViewModel. У відповідь юзер отримує токен авторизація, або ж повідомлення про помилку, якщо якісь користувач з такою електронною адресою уже зареєстрований у системі.
AuthViewModel	Методи цього класу відповідають за ініціацію запитів на сервер, які пов’язані з авторизацією. Коли юзер натискає на кнопку входу або реєстрації, методи AuthViewModel через AuthDomain відправляють запит на сервер, після чого відповідь кладуть у спеціальну StateFlow змінну, на яку «підписані» LoginFragment та RegistrationFragment, і при зміні значення цієї змінної одразу його оброблюють.
AuthDomain	Клас, який робить запит на сервер через AuthApiService, оброблює відповідь та

	обгортає її у клас Response, в залежності від того успішною чи ні була відповідь.
AuthApiService	Інтерфейс, в якому описані http-запити, пов’язані з авторизацією, їх метод, URL, хедера та параметри запиту, а також вказаний тип отримуваних даних.
MainActivity	Ця активність є контейнером для усіх фрагментів, окрім тих, що пов’язані з авторизацією. У вікні MainActivity розташований тулбар, який відображає назву фрагменту, а також стрілочку «назад». Також у цієї активності є BottomNavigationView – компонента, яка відображає чотири основних вкладки додатку, між якими можна переключатись: Home, Reservations, Bookmarks та Profile.
HomeFragment	Цей клас є стартовим фрагментом у MainActivity. Він відповідає за відображення MapFragment та RestaurantListFragment.
MapFragment	Макет цього фрагменту включає в себе лише карту, яка інтегрована через бібліотеку Google Maps. На мапі можна побачити маркери ресторанів поряд, а при натисканні на маркер буде відображено віконечко за більш інформацією про заклад і можливістю перейти до нього.
RestaurantsListFragment	У вікні RestaurantsListFragment відображено список ресторанів, а також кнопку з можливістю перейти на фрагмент пошуку та фільтрації. Дані про заклади цей фрагмент

	отримує через HomeViewModel.
HomeViewModel	Відповідає за відправку запитів списку ресторанів та отримання відповіді від серверу.
SearchFilterFragment	У цьому фрагменті користувач може відфільтрувати список ресторанів за трьома категоріями: назва, тип кухні та рейтинг. Обрані фільтри зберігаються у HomeViewModel і автоматично додаються до запиту на сервер. Також фільтрацію можна скинути, натиснувши кнопку Reset.
RestaurantsPagingAdapter	Цей клас є адаптером для списку ресторанів. Цей адаптер дозволяє нам відслідковувати його стан та відображати це у UI , а також він самостійно відправляє запит на наступну сторінку та показує стан завантаження, коли користувач долистав список до низу.
RestaurantsPagingSource	Цей клас дозволяє нам інтегрувати RestaurantsPagingAdapter та перевизначити метод load(), який відповідає за завантаження нової сторінки з ресторанами.
RestaurantFragment	Фрагмент RestaurantFragment відображає інформацію про ресторану, а саме галерею, рейтинг, робочий час, розташування, тип кухні, цінову політику та відгуки. Також з цього фрагменту можна перейти до бронювання, переглянути меню або ж написати відгук.
RestaurantViewModel	Цей клас відправляє запити на сервер, пов’язані з рестораном, такі як завантаження інформації про заклад, додавання його до

	обраних закладів користувача, отримання меню ресторану або написання відгуку юзером.
BookingContainer	BookingContainer дозволяє «розшарити» BookingViewModel по всьому графу бронювання. Граф бронювання включає в себе всі фрагменти, що стосуються цього процесу. Тобто всі обрані користувачем параметри (столик, дата, меню тощо) будуть зберігатись при переміщені юзера від одного фрагмента до іншого, і будуть очищенні тільки коли юзер вийде з процесу бронювання або завершить його.
BookingViewModel	Відповідає за запити та дані, що стосуються бронювання. Методи цього класу дозволяють отримати доступний час бронювання, доступні столик, список страв закладу, а також зберігати обрані користувачем параметри при бронювання.
ChooseDateFragment	У вікні цього фрагменту користувач обирає дату бронювання, після чого сервер відправляє вільний час для резервації на цей день.
ChooseTimeFragment	Цей фрагмент відображає вільний час для бронювання, які юзер може обрати. Також користувач вибирає тривалість відвідування (від одної до двох годин). Після вибору сервер надсилає список вільних місць.
ChooseTableFragment	Цей клас відображає список вільних столиків з фотографією та інформацією про максимальну кількість персон, які вміщаються за стіл.

ChooseMenuFragment	Даний фрагмент відображає меню закладу з можливістю обрати страви для передзамовлення.
ChooseMenuFragment	На цьому фрагменті юзер бачить всю інформацію про майбутнє бронювання і список обраних страв. Якщо все обрано правильно, юзер підтверджує бронювання натиснувши на відповідну кнопку.
ReservationsFragment	Цей фрагмент відображує список резервацій юзера з інформацією про кожне окреме бронювання. Детальну інформацію про бронювання можна побачити, натиснувши на будь-яке з списку та перейшовши на ReservationDetailsFragment.
ReservationDetailsFragment	Відображує всі деталі бронювання, а також страви, які замовив користувач. На цьому фрагменті можна побачити суму замовлення та перейти до його оплати. Також можна залиши чайові.
PaymentFragment	У цьому фрагменті юзер може оплатити своє замовлення. Він може або обрати одну із карт, які додані до його профілю, або додати нову.
TipsFragment	TipsFragment дає можливість користувачу залишити чайові. Для цього юзер повинен ввести бажаний розмір чайових та картою, якою він хоче це оплатити.

BookmarksFragment	Цей фрагмент зображає список ресторанів, які юзер додав у обране. Користувач може додати будь-який ресторан у обране, натиснувши відповідну кнопку на сторінці закладу.
ProfileFragment	Сторінка профілю відображає фотографію користувача, його електронну адресу та список доданих карт. Будь-яку карту можна видалити, або ж додати, натиснувши кнопку Add card.
AddCardFragment	У цьому фрагменті юзер можна додати нову карту, ввівши її номер, дату та cvv-код.
WriteReviewFragment	У цьому класі користувач може залишити відгук про ресторан, а також додати оцінку закладу.

Діаграма класів додатку зображена на кресленику ІА82.230БАК.003 Д4.

Висновки до розділу

У цьому розділі було розроблено різні діаграми програмного застосунку, такі як:

– діаграма варіантів використання, яка відображає функції, що можуть бути здійснені користувачем;

– діаграма послідовності всього додатку, на якій показано послідовність процесів, які виконує юзер;

– окрема діаграма послідовності процесу бронювання;

– діаграма діяльності процесу авторизації;

– діаграма класів андроїд додатку.

Також було описано функції основних класів програмного забезпечення.

8 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

8.1 Вхідні дані

Вхідні дані системи вказуються користувачем застосунку.

На вході користувач обирає наступні параметри:

- ресторан (номер);
- дату відвідування;
- час відвідування;
- тривалість відвідування (максимум 2 години);
- меню (за бажанням)
- столик (номер).

8.2 Вихідні дані

Вихідними даними є деталі замовлення в певному ресторані. Вони відображаються у вигляді переліку на сторінці бронювань.

8.3 Опис структури бази даних

Структура таблиць бази даних показана в таблицях 8.1-8.8.

Таблиця 8.1 – Структура таблиці Restaurants (таблиця ресторанів)

Назва	Тип даних	Опис поля
id	INT	Первинний ключ
name	CHAR	Назва ресторану
rating	DOUBLE	Рейтинг ресторану
cuisine	CHAR	Тип кухні
location	TEXT	Розташування
latitude	DOUBLE	Координати ширини
longtitude	DOUBLE	Координати довготи

photo	TEXT	Зображення ресторану
priceRange	INT	Діапазон цін
workingTime	TEXT	Час роботи
tablesCount	INT	Кількість столиків

Таблиця 8.2 – Структура таблиці tables (таблиця столиків)

Назва	Тип даних	Опис поля
id	INT	Первинний ключ
restaurantId	INT	Ідентифікатор ресторану
image	TEXT	Зображення
personsCount	INT	Кількість осіб для столику
tableNumber	INT	Номер столику

Таблиця 8.3 – Структура таблиці gallery (таблиця фотогалереї ресторану)

Назва	Тип даних	Опис поля
restaurantId	INT	Ідентифікатор ресторану
imageUrl	TEXT	URL-адреса зображення

Таблиця 8.4 – Структура таблиці users (таблиця користувачів)

Назва	Тип даних	Опис поля
id	INT	Первинний ключ
email	TEXT	Електронна адреса
name	TEXT	Ім’я
password	TEXT	Пароль
avatar	TEXT	Аватар

Таблиця 8.5 – Структура таблиці dishes (таблиця страв)

Назва	Тип даних	Опис поля

id	INT	Первинний ключ
restaurantId	INT	Ідентифікатор ресторану
name	TEXT	Назва
price	DOUBLE	Вартість
description	TEXT	Опис вмісту
image	TEXT	Зображення

Таблиця 8.6 – Структура таблиці bookings (таблиця бронювань)

Назва	Тип даних	Опис поля
id	INT	Первинний ключ
restaurantId	INT	Ідентифікатор ресторану
userId	INT	Ідентифікатор користувача
date	TEXT	Дата
time	TEXT	Час
isPaid	BOOLEAN	Оплата замовлення
tableId	INT	Ідентифікатор столика
orderTotal	DOUBLE	Загальна вартість замовлення
tips	DOUBLE	Сума чайових

Таблиця 8.7 – Структура таблиці orders (таблиця замовлень)

Назва	Тип даних	Опис поля
id	INT	Первинний ключ
bookingId	INT	Ідентифікатор бронювання
dishId	INT	Ідентифікатор страви
quantity	INT	Кількість

Таблиця 8.8 – Структура таблиці reviews (таблиця відгуків)

Назва	Тип даних	Опис поля
id	INT	Первинний ключ

userId	INT	Ідентифікатор користувача
review	TEXT	Текст відгуку
rating	DOUBLE	Оцінка
restaurantId	INT	Ідентифікатор ресторану

Структура таблиць бази даних даного мобільного застосунку наведена у графічному матеріалі ІА82.230БАК.003 Д5.

Висновки до розділу

У цьому розділі було визначено вхідні параметри системи, їх джерело та вихідні дані. У даному за стосунку вхідні дані це параметри, які обирає користувачі для створення замовлення. Він обирає ресторан та деталі для його бронювання. Вихідними даними системи є заброньований ресторан з деталями замовлення. Вони відображаються у вигляді переліку на сторінці бронювань. Також у цьому розділі була визначена та описана структура бази даних. Вона складається з восьми таблиць, які містять інформацію про ресторани, столики, фотогалерею, користувачів, страви, бронювання, замовлення, відгуки.

9 ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

9.1 Розроблення андроїд застосунку

Для початку роботи треба створити новий проєкт у Android Studio. При створення проєкту потрібно обрати йому назву та назву пакету, а також мову програмування та мінімальний андроїд, який зможе встановити даний додаток. У даному випадку було обрано мову програмування Kotlin, а мінімальна версія андроїду – 8.0. Така версія андроїду була вибрана тому, що 87% телефонів мають андроїд 8.0 або вище, а обравши версію нижче довелося б додавати багато лишнього коду і тим самим ускладнювати процес розробки додатку. Вікно створення проєкту можна побачити на рисунку 9.1.

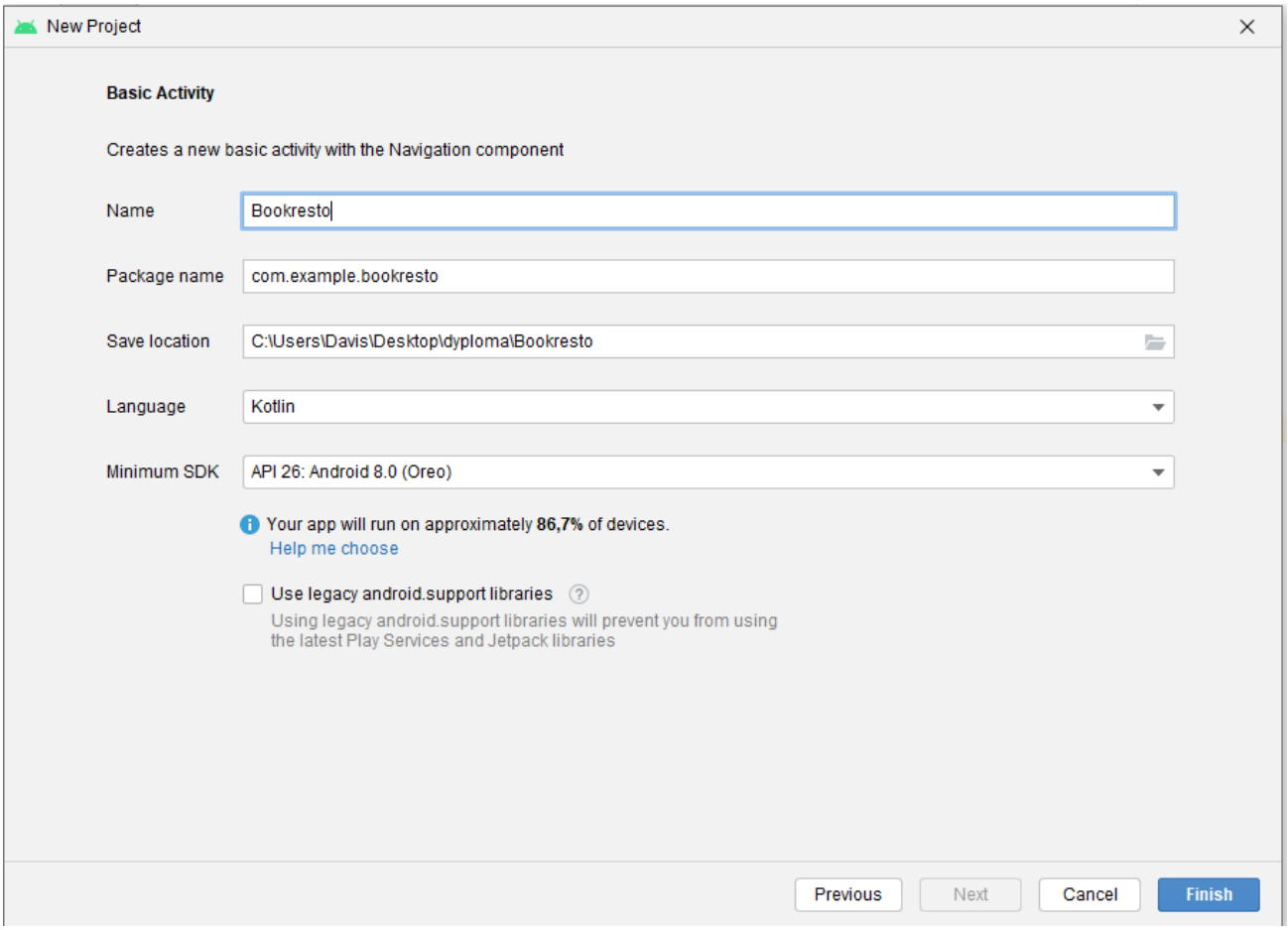

Рисунок 9.1 – Вікно створення проєкту в IDE Android Studio

Після створення проєкту потрібно перейти у файл build.gradle і додати бібліотеки, які будуть використовуватись у додатку. На рисунку 9.2 можна побачити, що у даний застосунок було додано бібліотеки для навігації, пагінації, DI, view model, http-запитів, корутин та різні UI бібліотеки, до прикладу, бібліотека для відображення компоненти для додавання карток, бібліотека для анімації, бібліотека для календаря тощо.

```
//Dependency injection
implementation "com.google.dagger:hilt-android:2.39.1"
kapt "com.google.dagger:hilt-compiler:2.38.1"

//Jetpack navigation
implementation "androidx.navigation:navigation-fragment-ktx:$nav_version"
implementation "androidx.navigation:navigation-ui-ktx:$nav_version"
implementation "androidx.navigation:navigation-dynamic-features-fragment:$nav_version"
implementation "androidx.navigation:navigation-compose:$nav_version"

//lifecycle
implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.4.1"
implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.4.1"
implementation "androidx.lifecycle:lifecycle-livedata-ktx:2.4.1"
implementation "androidx.lifecycle:lifecycle-viewmodel-savedstate:2.4.1"

implementation 'com.google.android.gms:play-services-maps:18.0.2'
implementation 'com.google.maps.android:android-maps-utils:2.3.0'

//paging library
implementation 'android.arch.paging:runtime:1.0.1'
implementation "androidx.paging:paging-runtime-ktx:3.1.1"
testImplementation "androidx.paging:paging-common:3.1.1"

//viewmodel and lifecycle
implementation 'androidx.lifecycle:lifecycle-extensions:2.2.0'

//retrofit
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
implementation 'com.squareup.okhttp3:logging-interceptor:4.9.0'

//coroutines
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.1'
implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.6.1'
implementation 'com.jakewharton.retrofit:retrofit2-kotlin-coroutines-adapter:0.9.2'

// Images download
implementation 'com.github.bumptech.glide:glide:4.13.0'
annotationProcessor 'com.github.bumptech.glide:compiler:4.13.0'

//UI
implementation 'com.github.aids61517:EasyRatingView:1.1.1'
implementation "com.tbuonomo:dotsindicator:4.3"
implementation 'de.hdodenhof:circleimageview:3.1.0'
implementation 'com.applandeo:material-calendar-view:1.9.0-rc03'
implementation 'nl.bryanderidder:themed-toggle-button-group:1.4.1'
implementation 'com.airbnb.android:lottie:5.2.0'
```

Рисунок 9.2 – Залежності бібліотек у проєкті

Після того, як залежності додано, можна приступати до розробки. Спочатку потрібно створити активність, з якої буде починатись робота додатку. Цією активністю буде AuthorizationActivity. У її макет ми додаємо Toolbar та FragmentContainerView. У коді активності ми спочатку перевіряємо, чи збережений у токен користувача у shared preferences. Shared preferences – це постійне сховище на платформі Android, яке використовується додатками для зберігання, до прикладу, налаштувань, в даному ж випадку – токену авторизації[16]. Якщо токен є у сховищі, юзера одразу перенаправляє на MainActivity, а якщо немає, то відкриється вікно входу. Перевірка наявності токену у сховищі виконується класом SessionState, який інжектиться у клас активності за допомогою бібліотеки для DI.

Це можна побачити на фрагменті коду, рисунок 9.3.

```
@AndroidEntryPoint
class AuthorizationActivity : AppCompatActivity() {

    @Inject
    lateinit var sessionState: SessionState

    val binding by viewBinding(ActivityAuthorizationBinding::inflate)
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        if(sessionState.userLoggined) {
            val intent = Intent( packageContext: this, MainActivity::class.java)
            startActivity(intent)
        }
        setContentView(binding.root)
```

Рисунок 9.3 – Перевірка наявності токена авторизації в AuthorizationActivity

У MainActivity ми реалізуємо BottomNavigationView. BottomNavigationView – це нижня панель навігації, яка дозволяє переходити між екранами – пунктами призначення навігації, а також наочно інформує користувача про те, на якому екрані він знаходиться[17]. Для імплементації цієї компоненти потрібно спочатку створити файл навігації в форматі XML, який буде називатись

main_navigation.xml, після чого потрібно передати його у компоненту. У цьому файлі ми включаємо інші графи, які відносять окремо до кожної вкладки меню навігації. Це можна побачити на рисунку 9.4.

```
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    app:startDestination="@id/home_nav"
    android:id="@+id/main_navigation">

    <include app:graph="@navigation/home_nav"/>
    <include app:graph="@navigation/reservations_nav"/>
    <include app:graph="@navigation/bookmarks_nav"/>
    <include app:graph="@navigation/profile_nav"/>
</navigation>
```

Рисунок 9.4 – Основний граф навігації.

Далі потрібно встановити залежність NavigationController з панеллю навігації та з Toolbar. Таким чином, при перемиканні між пунктами меню нижньої панелі буде встановлюватись потрібний граф, а також в тулбарі кнопка меню буде відображатись тільки тоді, коли фрагменти не є «рутовим». Також зробити функцію повернення до рутового фрагменту графу за умови подвійного натискання на елемент меню. Реалізацію цього зображено на рисунку 9.5.

```
        // Setup the bottom navigation view with navController
        binding.bottomNavigation.setupWithNavController(navController)

        binding.bottomNavigation.setOnItemReselectedListener { it: MenuItem
            btmNavManager.handleDoubleClick(it, navController, currentFragment)
        }


        val appBarConfiguration = AppBarConfiguration(
            setOf(
                R.id.home_fragment,
                R.id.reservations_fragment,
                R.id.bookmarks_fragment,
                R.id.profile_fragment
            )
        )
        setupWithNavController(binding.toolbar, navController, appBarConfiguration)
    }
```

Рисунок 9.5 – Імплементація нижньої панелі меню

Далі потрібно створити HomeFragment, у якому буде ViewPager, який буде контейнером для MapFragment та RestaurantListFragment. Для його реалізації на потрібно створити FragmentStateAdapter і встановити його у ViewPager, а також зв’язати ViewPager з TabLayout, який матиме дві кнопки – Map та Restaurant. Ці кнопки дозволять користувачу переключатись між фрагментом карти та фрагментом списку.

У MapFragment буде застосовано GoogleMaps API. Для цього потрібно створити проєкт на Google Cloud і увімкнути там GoogleMaps API. Далі потрібно створити API-ключ, вказавши пакет додатку та SHA-1 ключ проєкту. Цей процес зображено на рисунку 9.6

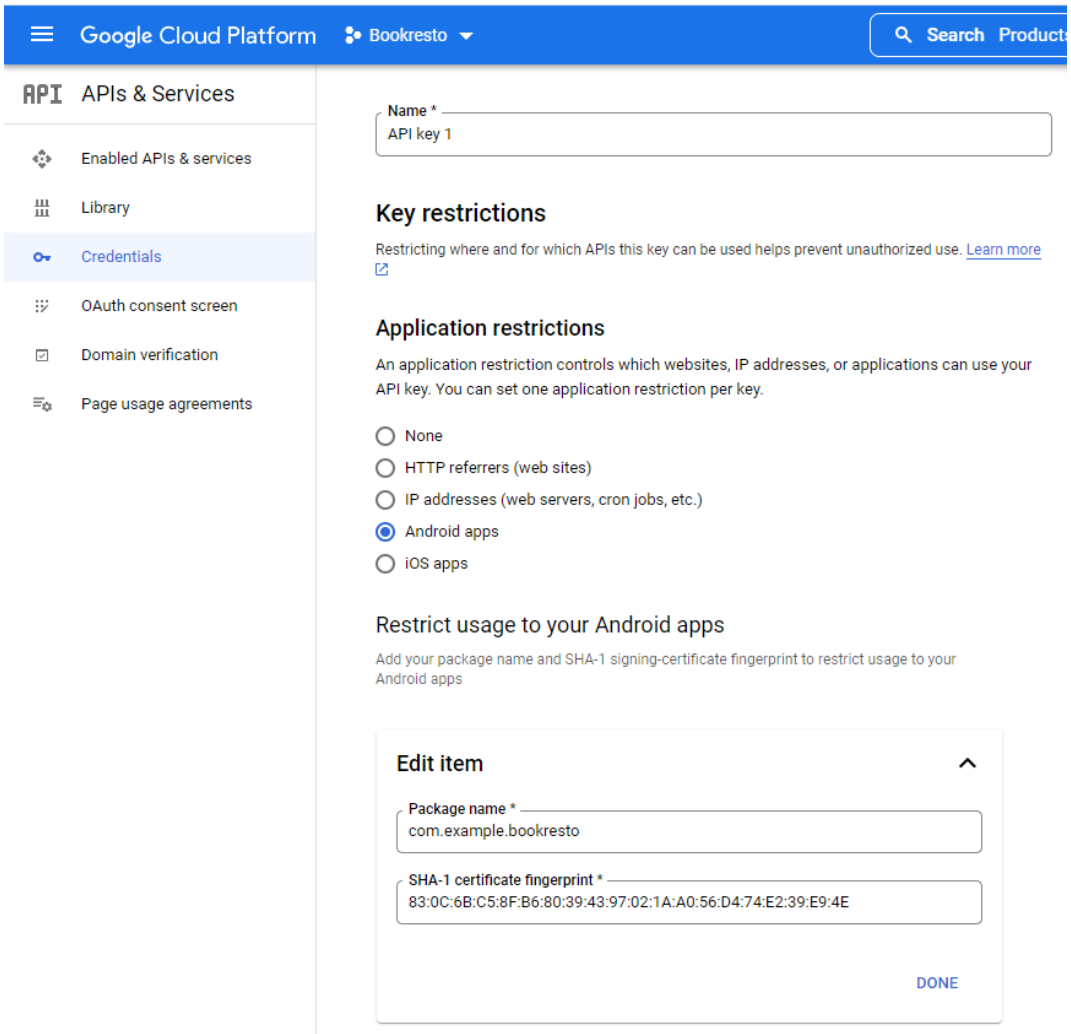

Рисунок 9.6 – Отримання API-ключа для GoogleMaps API

Після отримання ключа необхідно додати його у AndroidManifest файл.

Далі необхідно додати колбек у MapFragment, який буде спрацьовувати, коли карта буде готова до використання. Там ми встановлюємо початкову точку відображення на карті, а також listener на маркер, для того аби з'являлось віконечко з обраним рестораном при натисканні на маркер. Це видно на рисунку 9.7.

```
inner class MapCallback : OnMapReadyCallback {
    override fun onMapReady(map: GoogleMap) {
        gMap = map
        map.uiSettings.isMapToolbarEnabled = false

        val point = CameraUpdateFactory.newLatLngZoom(
            LatLng( latitude: 50.460069177309165,  longitude: 30.546350275999362),
             zoom: 12f
        )
        map.moveCamera(point)

        map.animateCamera(point)

        map.setOnMarkerClickListener { marker ->
            viewModel.mapFlow.value.response?.find { it.id.toString() == marker.tag.toString() }
                ?.let {  it: Restaurant
                    binding.restaurant.restaurantCuisine.text = it.cuisine
                    binding.restaurant.restaurantName.text = it.name
                    Glide.with(requireContext()).load(it.photo)
                        .into(binding.restaurant.restaurantImage)
                    binding.restaurant.restaurantRating.text = it.rating.toString()
                    binding.restaurant.restaurantRatingBar.rating = it.rating.toFloat()
                    binding.restaurant.priceRating.rating = it.priceRange.toFloat()
                    binding.restaurant.restaurantLocation.text = it.location
                    binding.restaurant.root.setOnClickListener { _ ->
                        val deepLink =
                            deepLinkFactory.createRestaurantDeepLink(it.name, it.id)
                        screenNavigationDeepLink.navigate(deepLink)
                    }
                    binding.restaurantWrapper.visible()
                }
            true  ^setOnMarkerClickListener
        }
    }
}
```

Рисунок 9.7 – Імплементація MapCallback

У фрагменті RestaurantListFragment відображається RecyclerView з список ресторанів, реалізований за допомогою PagingAdapter. PagingAdapter дозволяє створити список, який буде автоматично підвантажувати наступну сторінку, коли користувач догортає список до самого низу. Також цей адаптер дозволяє встановити listener, за допомогою якого можна відслідковувати різні стани

списку. На рисунку 9.8 зображено, як встановлюється listener для обробки стану, коли список ресторанів пустий.

```
private fun getCurrentAdapter(): ConcatAdapter {
    restaurantsAdapter = pagingAdapterProvider.get()
    return with(restaurantsAdapter) {  this: RestaurantsPagingAdapter

        restaurantClickListener = RestaurantClickListener()

        //Add state listener to render UI depending on adapter loading state
        addLoadStateListener { combinedLoadStates -> renderUIStates(combinedLoadStates) }

        //Add footer to display page appending loading
        withLoadStateFooter(loadingStateAdapter)  ^with

    }
}

private fun renderUIStates(loadStates: CombinedLoadStates) {
    val isListEmpty =
        loadStates.source.refresh is LoadState.NotLoading && restaurantsAdapter.itemCount == 0
    binding.emptyList.visibleIf(isListEmpty)
    binding.restaurantsRecyclerView.visibleIf(!isListEmpty)

}
```

Рисунок 9.8 – Встановлення listener для обробки станів адаптеру

Для переміщення між фрагментами застосовуються класи ScreenNavigationDeepLink та DeepLinkFactory. Спочатку створюється deeplink, в який передається id фрагменту, а також параметри, які потрібно передати у фрагмент, за необхідності. Такі deeplink створюються для кожного фрагменту і знаходяться у класі DeepLinkFactory. Далі потрібно викликати функцію navigate() класу ScreenNavigationDeepLink і передати у нього створений діплінк. Цей метод знайде NavigationController, який в конкретний момент використовується у додатку і перенаправить користувача на фрагмент із діплінка, передавши також параметри.

Для зменшення повторюваного коду у проєкт було додано клас BaseFragment. Надалі від цього класу будуть унаслідуватись усі фрагменти. У BaseFragment базові змінні та методи, які будуть застосовуватись у будь-якому фрагменті додатку. Це дозволить уникнути написання однакового коду у

фрагментах і користуватись змінним і методами з базового фрагменту. Реалізація BaseFragment зображена на рисунку 9.9.

```
@AndroidEntryPoint
open class BaseFragment(@LayoutRes layout: Int) : Fragment(layout), FragmentContract {

    @Inject
    lateinit var screenNavigationDeepLink: ScreenNavigationDeepLink

    @Inject
    lateinit var deepLinkFactory: DeepLinkFactory

    @Inject
    lateinit var viewModelFactory: StubViewModelFactory

    override fun configureToolbar() {
        activity?.title = toolbarTitle()
    }

    fun configureBackButtonFragments(@IdRes fragmentId: Int?) {
        (activity as? MainActivity)?.updateToolbarIncludeConfirmation(fragmentId)
    }

    open fun toolbarTitle(): String = "Fragment"

    override fun onResume() {
        super.onResume()
        configureToolbar()
    }

    override fun isActionBarBackButton(): Boolean = false

}
```

Рисунок 9.9 – Реалізація BaseFragment

Для того, аби під час процесу бронювання не зникали параметри, які обрав юзер, було вирішено створити клас BookingContainer. Цей клас дозволити не створювати кожного разу новий клас view model, а ділити його між усім booking графом. Для цього потрібно створити реалізувати ініціалізацію view model за допомогою методу navGraphViewModel().

Далі у кожному фрагменті, який є у booking_nav_graph потрібно ініціалізовувати view model з BookingContainer за допомогою lazy ініціалізації.

Реалізацію цього контейнеру та розгортання view model у фрагменті можна побачити на рисунку 9.10.

```
class BookingContainer @Inject constructor(
) {

    //region PUBLIC FUNCTIONS ----------------------------------------------------

    /**
     * Returns [BookingViewModel]
     */
    fun bookingViewModel(fragment: BaseFragment): BookingViewModel {
        return fragment.navGraphViewModels<BookingViewModel>(R.id.booking_nav) {
            fragment.defaultViewModelProviderFactory
        }.value
    }

    //endregion
}
```

```
@AndroidEntryPoint
class ChooseDateFragment : BaseFragment(R.layout.fragment_choose_date) {

    val binding by viewBindingWithBinder(FragmentChooseDateBinding::bind)

    @Inject
    lateinit var bookingContainer: BookingContainer

    private val viewModel: BookingViewModel by lazy {
        bookingContainer.bookingViewModel( fragment: this)
    }
```

Рисунок 9.10 – Реалізація BookingContainer і його використання у фрагменті

У BookingViewModel зберігаються всі параметри, які обрав користувач при бронюванні, і вони не втрачаються при навігації вперед та назад. Проте, для того аби не було конфліктів, коли користувач обрав, до приклад, вільний столик, а потім повернувся назад і змінив час бронювання, збережена інформація про обраний столик очиститься. Теж саме буде, якщо юзер обере час бронювання, повернеться назад та вибере інакшу дату. Обраний час зіб’ється. Проте меню

обране меню буде збережено, оскільки користувач уже не може змінити ресторан, перейшовши у процес бронювання.

Частина класу BookingViewModel зображено на рисунку 9.11.

```
@HiltViewModel
class BookingViewModel @Inject constructor(
    val bookingDomain: BookingDomain,
    val sessionState: SessionState
) : ViewModel() {

    var restaurantId: Long = 0

    var selectedDate: Calendar? = null
        set(value) {
            availableTablesFlow.value = Response.Loading()
            availableTimeFlow.value = Response.Loading()
            field = value
        }
    var selectedTime: String? = null
        set(value) {
            availableTablesFlow.value = Response.Loading()
            field = value
        }
    var duration: Int = 1
    var selectedTable: Int? = null

    var selectedMenu: HashMap<Int, Int> = hashMapOf()

    val formattedDate: String?
        get() = selectedDate?.let {  it: Calendar
            "${it.get(Calendar.DAY_OF_MONTH)}/${it.get(Calendar.MONTH) + 1}/${it.get(Calendar.YEAR)}"
        }
```

Рисунок 9.11 – Клас BookingViewModel

Хоча в Android SDK наявні багато різних елементів для реалізацію UI, іноді їх недостатню і треба зробити власний custom view, аби винести логіку в окремий клас і перевикористовувати його. Саме через було створено два нестандартних компоненти: QuantityComponent та ReviewsComponent. Перший застосовується при виборі кількості страв під час замовлення, а другий для того, або відображати відгуки про заклад. ReviewsComponent має функцію addReviews(), в яку передається список відгуків. Проте спочатку відображується не весь список, а лише три елементи та кнопка для підвантаження ще трьох елементів і так доти, доки не будуть завантажені усі відгуки. Коли всі відгуки

завантаження, кнопка зникає. Ця логіка реалізована в методі loadMoreReviews(). Детальний код класу зображений на рисунку 9.12.

```
fun addReviews(reviewsList: List<Review>?) {
    binding.reviewsContainer.removeAllViews()
    reviewsLoaded = 0
    this.reviewsList = reviewsList
    loadMoreReviews()
}

fun loadMoreReviews() {
    reviewsList?.let {  it: List<Review>
        if (it.size > 3 + reviewsLoaded) {
            it.subList((reviewsLoaded), (reviewsLoaded) + 3).forEach { review ->
                val reviewBinding =
                    ReviewItemBinding.inflate(LayoutInflater.from(context),  parent: null,  attachToParent: false)
                reviewBinding.name.text = review.name
                reviewBinding.reviewText.text = review.review
                reviewBinding.rating.text = review.rating.toString()
                reviewBinding.ratingBar.rating = review.rating.toFloat()
                Glide.with(context).load(review.avatar).into(reviewBinding.profileImage)
                binding.reviewsContainer.addView(reviewBinding.root)
            }
            binding.loadMoreBtn.visibility = View.VISIBLE
            reviewsLoaded += 3
        } else {
            it.subList((reviewsLoaded), it.size).forEach { review ->
                val reviewBinding =
                    ReviewItemBinding.inflate(LayoutInflater.from(context),  parent: null,  attachToParent: false)
                reviewBinding.name.text = review.name
                reviewBinding.reviewText.text = review.review
                reviewBinding.rating.text = review.rating.toString()
                reviewBinding.ratingBar.rating = review.rating.toFloat()
                Glide.with(context).load(review.avatar).into(reviewBinding.profileImage)
                binding.reviewsContainer.addView(reviewBinding.root)
            }
            binding.loadMoreBtn.visibility = View.GONE
            reviewsLoaded += (it.size - reviewsLoaded)

        }
    }
```

Рисунок 9.12 – Код класу ReviewsComponent

По всьому проєкту застосовується бібліотека Kotlin Coroutines. Вона призначена для асинхронної роботи з кодом, тобто коли нам потрібно зробити запит у базу даних чи на сервер, головний потік андроїд, який відповідає за оброблення натискань та відмальовування UI, не займається, тому наш додаток працює без проблем.

Також у view model використовуються State Flow зміні, на які можна «підписатися» на стороні фрагменти. Тобто, є функція для виконання запиту на сервер та змінна StateFlow. Після того, як виконався запит і повернувся

результат, він кладеться у StateFlow. «Спостерігач» бачить зміну у цій змінній, і виконує певну функцію, в залежності від значення цієї змінної.

Нижче на рисунку 9.13 зображена реалізація RestaurantViewModel та «підписування» на StateFlow у фрагменті ресторану.

```
@HiltViewModel
class RestaurantViewModel @Inject constructor(
    val restaurantDomain: RestaurantDomain,
    val bookingDomain: BookingDomain,
) : ViewModel() {

    val restaurantFlow = MutableStateFlow<Response<Restaurant>>(Response.Loading())
    val menuFlow = MutableStateFlow<Response<List<MenuItem>>>(Response.Loading())

    fun getRestaurant(id: Long) {
        viewModelScope.launch {  this: CoroutineScope
            restaurantFlow.value = restaurantDomain.getRestaurant(id)
        }
    }
```

```
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    super.onViewCreated(view, savedInstanceState)

    viewLifecycleOwner.lifecycleScope.launch {  this: CoroutineScope
        viewModel.restaurantFlow.collectLatest {  it: Response<Restaurant>
            when (it) {
                is Response.Success -> {
                    renderUI(it.response!!)
                    binding.loading.gone()
                }
                is Response.Loading -> binding.loading.visible()
                is Response.Error -> Toast.makeText(context, it.message!!, Toast.LENGTH_LONG)
                    .show()
            }
        }
    }

    viewModel.getRestaurant(args.getLong(RESTAURANT_ID))
```

Рисунок 9.13 – Реалізація RestaurantViewModel

Тобто спочатку підписуємось на restaurantFlow за допомогою методу collectLatest, і в цьому методі обробляємо UI в залежності від типу відповіді від серверу: Success, Error чи Loading. Далі робимо запит, виконавши метод viewModel.getRestaurants(). Коли відповідь прийде від серверу, вона буде

присвоєна значенню restaurantFlow, про що фрагмент отримає сповіщення і обробить відповідь.

Такий підхід використовується у всіх фрагментах.

9.2 Розроблення серверу

Для розробки серверної частини було вирішено використовував NodeJS з застосування ExpressJS,а також MySQL в якості сховища даних.

Для початку потрібно створити базу даних bookrestodb, а також користувача, наділеного правами додавання, читання видалення та оновлення. Далі потрібно створити файл з розширенням .js. Після чого потрібного імпортувати бібліотеки, які будуть використовуватись у проєкті, а також підключитися до бази даних і запустити сервер. Реалізація цього процесу зображена на рисунку 9.14.

```
const express = require('express')
const jwt = require("jsonwebtoken")
const mysql = require('mysql');

const app = express()

const port = 3005

const config = {
  host: "127.0.0.1",
  user: "danil",
  password: "1111",
  database: "bookrestodb",
}

const pool = mysql.createPool(config);

pool.getConnection((err) => {
  if (!err) {
    console.log("Connected to MySQL");
  } else {
    console.log(err);
  }
})

app.listen(port, () => {
  console.log(`App listening at http://localhost:${port}`)
})
```

Рисунок 9.14 – Підключення до бази даних та старт серверу

До бази даних підключаємось з даними користувача, якого ми створили та наділили всіма правами.

Після успішного запуску серверу потрібно створити функції, які будуть часто використовуватись. Цими функціями є getUserDataFromToken() та verifyToken(). Їх реалізація зображена на рисунку 9.15.

```
function getUserDataFromToken(token) {
  return jwt.verify(token, 'secretkey', (err, authData) => {
    if (err) {
      const errorBody = {
        error: {
          code: 403,
          message: "Unauthorized"
        }
      }
      return errorBody
    } else {
      return authData
    }
  })
}

function verifyToken(req, res, next) {
  const bearerHeader = req.headers['authorization'];

  if (bearerHeader) {
    const bearer = bearerHeader.split(' ');
    const bearerToken = bearer[1];
    req.token = bearerToken;
    next();
  } else {
    // Forbidden
    res.json({
      success: false,
      error: {
        code: 403,
        message: "Unauthorized"
      }
    });
  }
}
```

Рисунок 9.15 – Реалізація функцій є getUserDataFromToken() та verifyToken()

Вони будуть використовуватись для запитів, в який нам потрібні дані користувача. Функція verifyToken() перевіряє, чи валідний токен, який був надісланий з додатку. Якщо токен невірний, сервер присилає помилку, якщо вірний – то виконання запиту продовжується. Метод getUserDataFromToken() достає з токену електронну адресу користувача, яку можна використовувати для отримання його ідентифікатора. На рисунку 9.16 зображено, як при запиті інформації ресторану перевіряється токен за допомогою verifyToken(), а потім з нього витягається електронна адреса через функцію getUserDataFromToken(), а далі через адресу отримується ідентифікатор користувача, який дозволяє перевірити, чи входить даний заклад у список його обраних.

```
app.get('/api/restaurant/:id', verifyToken, (req, res) => {

  let id = parseInt(req.params.id)
  let email = getUserDataFromToken(req.token).user.email

  pool.getConnection(async (err, connection) => {
    let userId = await new Promise(function (resolve, reject) {
      connection.query(`SELECT id FROM users WHERE email=?`, [email], (err, _result) => {
        result = JSON.parse(JSON.stringify(_result))[0]
        if (result !== undefined) {
          resolve(result.id)
        } else {
          res.json({
            error: {
              code: 403,
              message: ""
            }
          })
          return false
        }
      })
    })
```

Рисунок 9.16 – Перевірка токену та витягування з нього даних

Найскладнішою функцією на стороні серверу є getAvailableTime(), за допомогою якої формується список вільного часу для бронювання. Спочатку у неї передається список всіх бронювань для даного столику та час роботи ресторану. Далі за допомогою циклу for час кожного бронювання розділяється

на початок та кінець і переводиться з типу String у тип Integer. Далі кожен заброньований проміжок у вигляді масиву додається у масив забронюваного часу. Потім з в циклі for з початку роботу до кінця роботи ресторану перевіряється чи вільна дана година для бронювання за трьома параметрами. Якщо година не є початком іншого бронювання, не я початком та одночасно кінцем двох бронювань, а також не є серединою одного з бронювань, тоді цю годину можна вважати доступною для резервація. Детальний код даної функції зображено на фрагменті на рисунку 9.17.

```
function getAvailableTime(allBookingsTimeForTable, workingTime) {
  let reservedTime = []

  for (let index = 0; index < allBookingsTimeForTable.length; index++) {
    let start = timeToInt(allBookingsTimeForTable[index].split("-")[0])
    let end = timeToInt(allBookingsTimeForTable[index].split("-")[1])

    let reservedTimeFrame = []
    for (let i = start; i <= end; i++) {
      reservedTimeFrame.push(i)
    }
    reservedTime.push(reservedTimeFrame)
  }
  reservedTime.sort((first, second) => first[0] - second[0])

  let workingTimeStart = timeToInt(workingTime[0])
  let workingTimeEnd = timeToInt(workingTime[1])
  let availableTime = []
  for (let i = workingTimeStart; i < workingTimeEnd; i++) {
    let startOfSomeBooking = false
    let endOfSomeBooking = false
    let containsOfSomeBooking = false
    for (let l = 0; l < reservedTime.length; l++) {
      if (i == reservedTime[l][0]) {
        startOfSomeBooking = true
      } else if (i == reservedTime[l][reservedTime[l].length - 1]) {
        endOfSomeBooking == true
      }

      if (reservedTime[l].length == 3 && reservedTime[l][1] == i) {
        containsOfSomeBooking = true
      }
    }

    if (!(startOfSomeBooking) && !(startOfSomeBooking && endOfSomeBooking) && !containsOfSomeBooking) {
      availableTime.push(i)
    }
  }

  return availableTime
}
```

Рисунок 9.17 – Реалізація функції getAvailableTime()

Для коректної роботи додатку потрібно зберігати сервер запущеним локально або ж задеплоїти його у хмару.

10 ІНСТРУКЦІЯ КОРИСТУВАЧА

Після входу в додаток користувач потрапляє на сторінку авторизації. Вона зображена на рисунку 10.1. З цієї сторінки користувач може здійснити авторизацію в систему, якщо був раніше зареєстрований. Для цього слід ввести логін та пароль і натиснути кнопку “Login”.

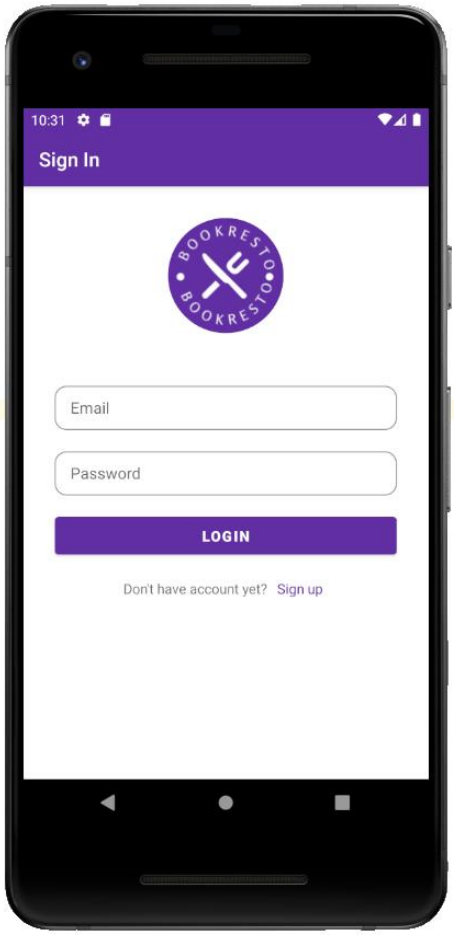

Рисунок 10.1 – Сторінка авторизації

У випадку, коли користувач увійшов у систему вперше, він може зареєструватися, створивши новий обліковий запис. Для цього потрібно натиснути кнопку “Sign up”. Потрапивши на сторінку реєстрації, користувач вказує адресу своєї електронної пошти, створює пароль та підтверджує його. Після заповнення всіх полів потрібно натиснути кнопку “Sign up”. Сторінка реєстрації зображена на рисунку 10.2.

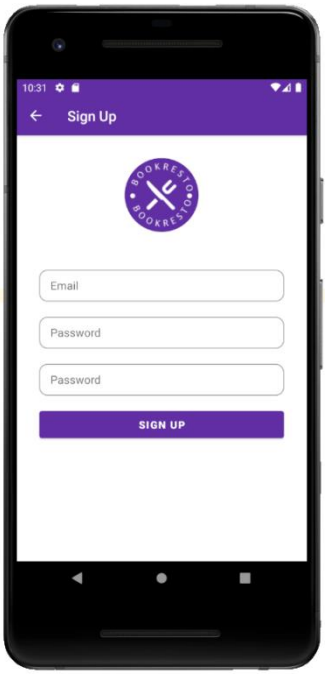

Рисунок 10.2 – Сторінка реєстрації

Після успішної реєстрації чи авторизації в системі користувач автоматично переходить на головну сторінку пошуку ресторану. Головна сторінка має вигляд мапи та зображена на рисунку 10.3.

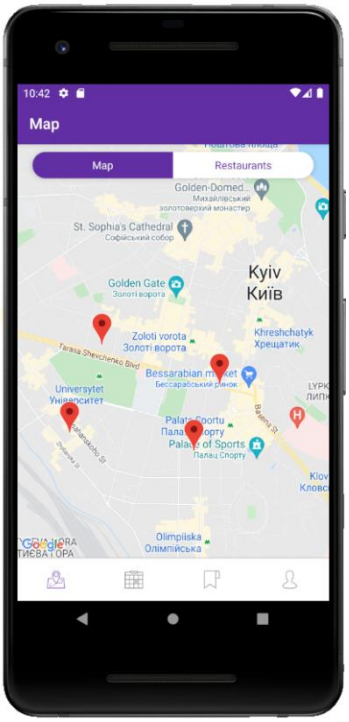

Рисунок 10.3 – Головна сторінка

На головній сторінці користувач має можливість переглянути перелік ресторанів у вигляді безпосередньої локації ресторану на карті (рис. 10.3) або у вигляді списку. Щоб перейти до переліку ресторанів у вигляді списку, потрібно натиснути кнопку “Restaurants”. Такий тип переліку зображено на рисунку 10.4.

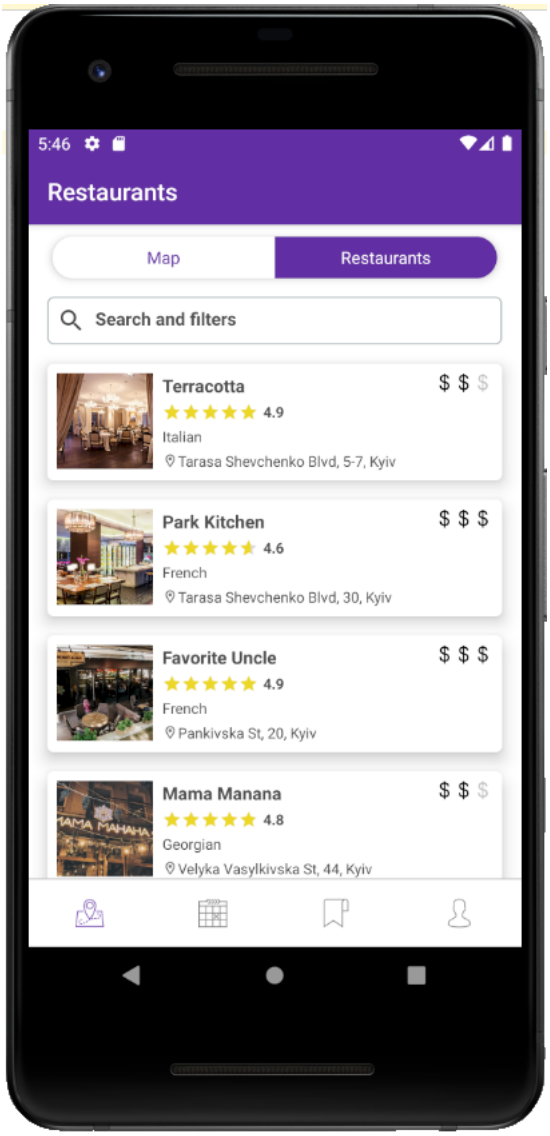

Рисунок 10.4 – Перелік ресторанів у вигляді списку

Щоб отримати коротку інформацію про заклад, користувач може натиснути на червоний маркер ресторану, що його цікавить. Після цього на екрані з’явиться інформаційне вікно відповідного закладу. Сторінку мапи з інформаційним вікном зображено на рисунку 10.5.

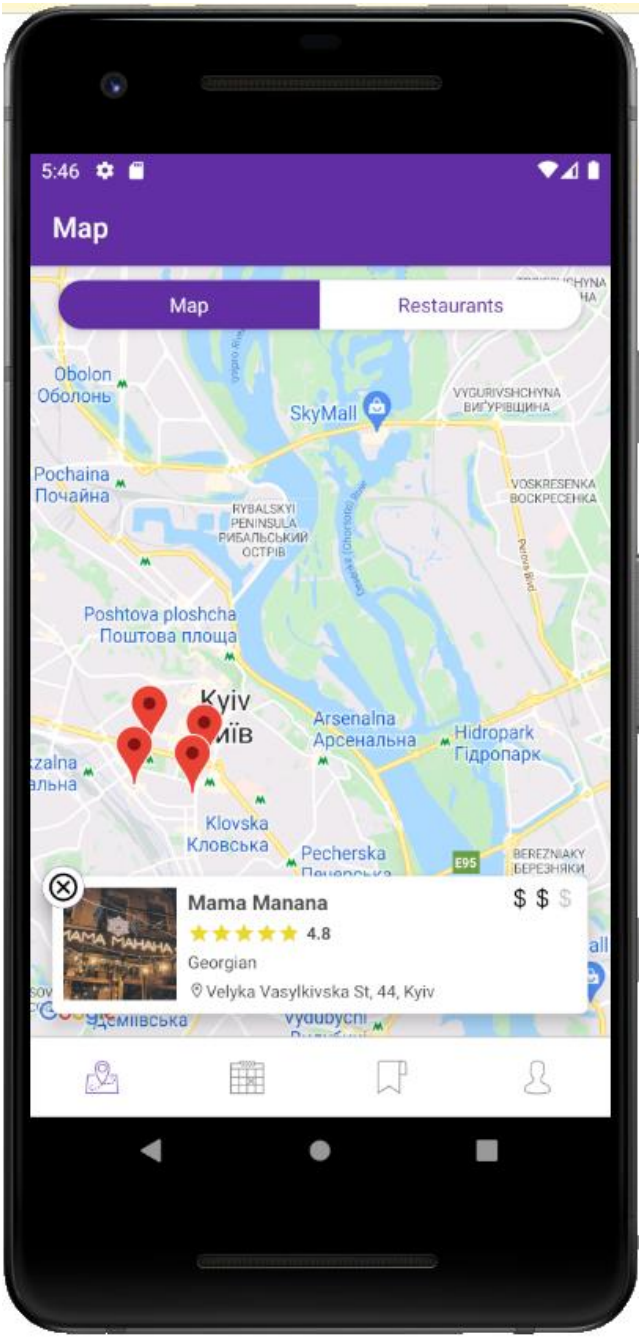

Рисунок 10.5 – Сторінка мапи з інформаційним вікном

На сторінці переліку ресторанів у вигляді списку (рис. 10.4) можна обрати функції додаткового пошуку та фільтрації закладів, натиснувши на поле “Search and filters”. Після цього користувач потрапляє на сторінку фільтрації, яка зображена на рисунку 10.6. На цій сторінці можна відфільтрувати заклади за назвою, типом кухні чи рейтингом. Вказавши потрібні параметри, потрібно натиснути на кнопку “Apply filters”. Для видалення вибраних параметрів можна натиснути на кнопку “Reset”.

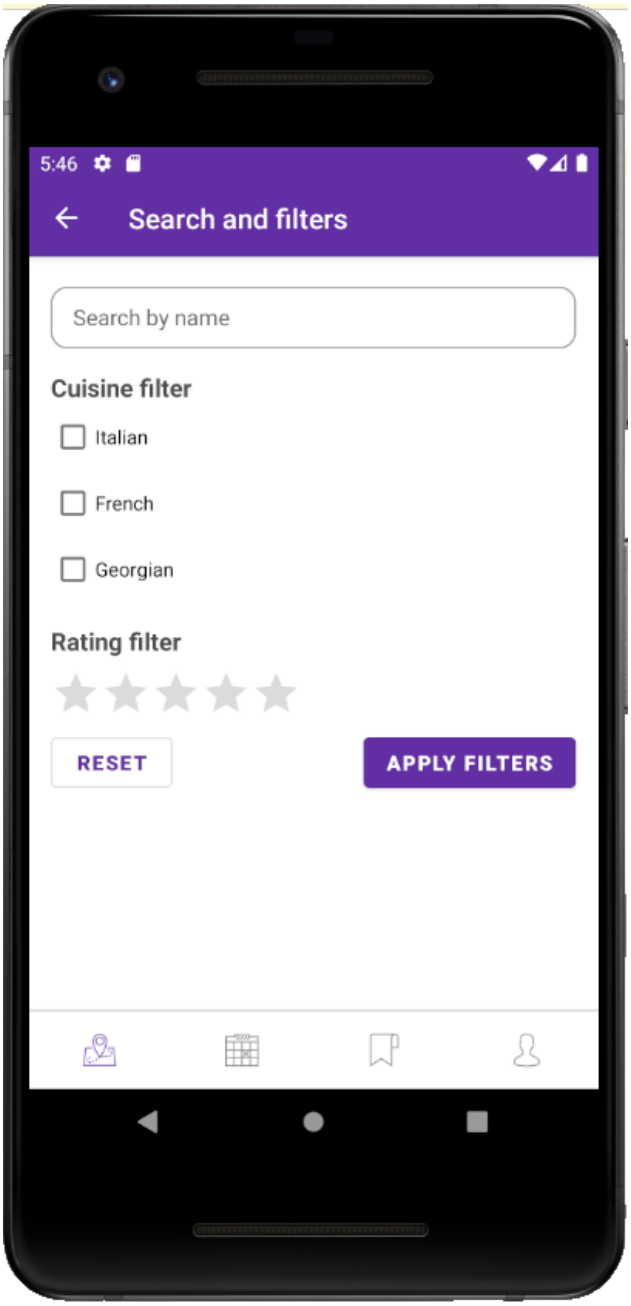

Рисунок 10.6 – Сторінка фільтрації закладів

Після вибору закладу, що цікавить користувача, він потрапляє на сторінку самого ресторану, де вказана уся інформація про нього та відгуки користувачів. Вигляд сторінки з інформацією про ресторан наведено на рисунку 10.7. Тут користувач може залишити відгук про заклад після відвідин, натиснувши на кнопку “Write a review”. У разі вибору потрібного ресторану користувач може забронювати столик, натиснувши на кнопку “Book a table”.

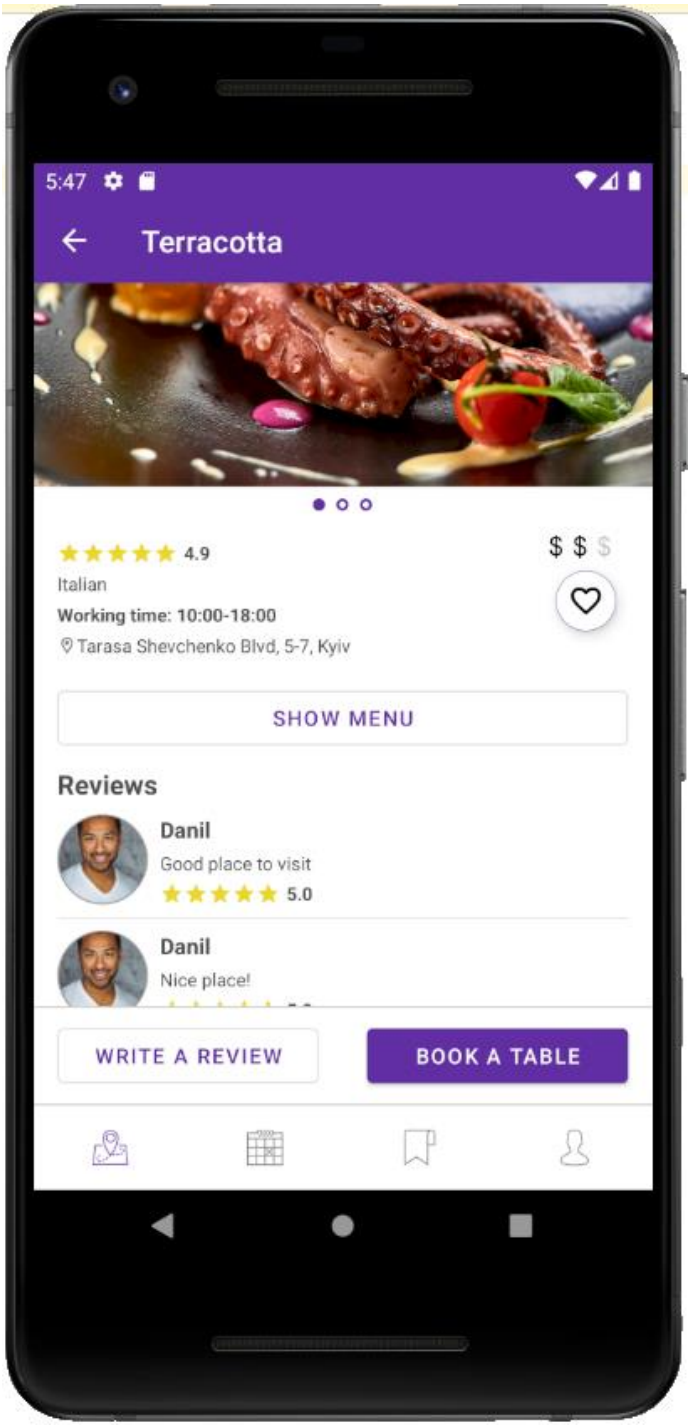

Рисунок 10.7 – Сторінка з інформацією про ресторан

Також зі сторінки про ресторан можна перейти на сторінку меню обраного закладу, натиснувши на кнопку “Show menu”. У цьому вікні буде відображено назву, зображення, вартість та опис страви. Сторінка меню зображена на рисунку 10.8.

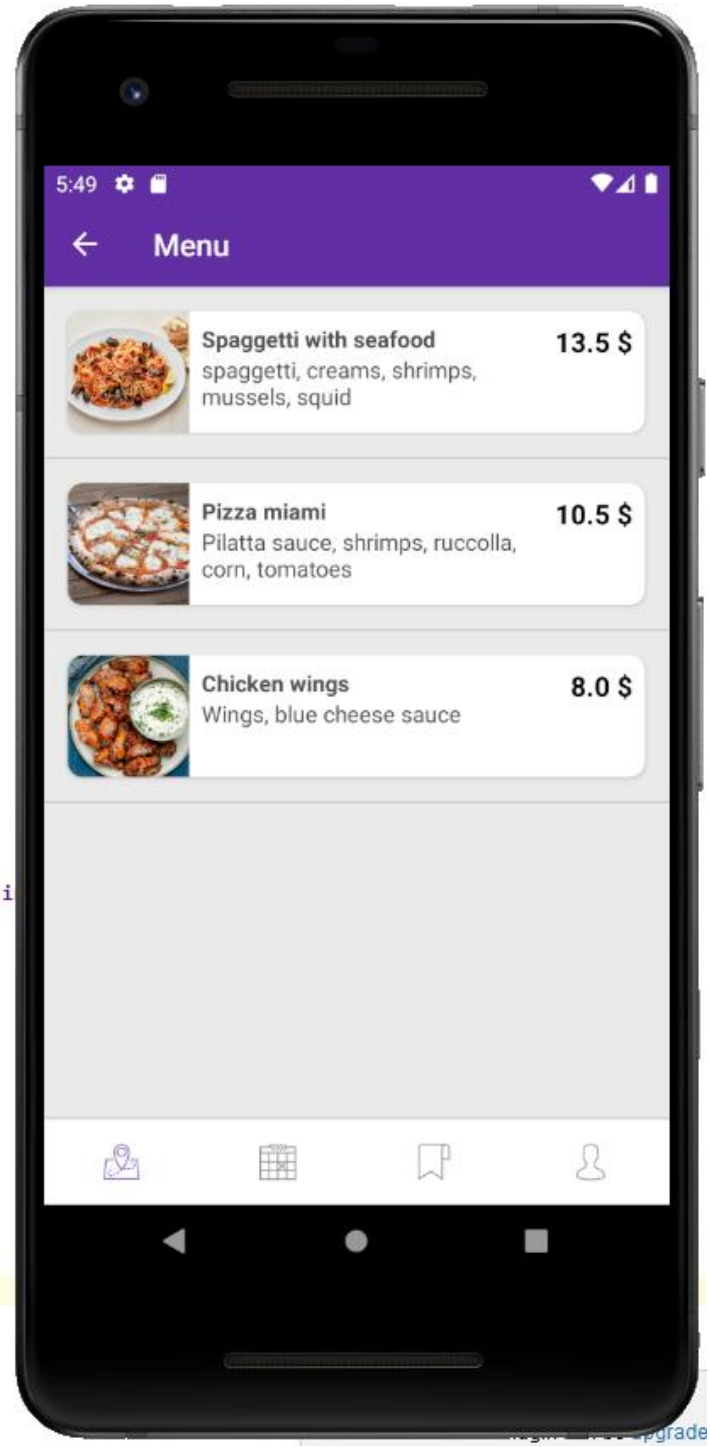

Рисунок 10.8 – Сторінка меню

Після натискання на кнопку “Book a table” користувач потрапляє на сторінку з календарем, де має вибрати дату бронювання. Сторінка з календарем зображена на рисунку 10.9.

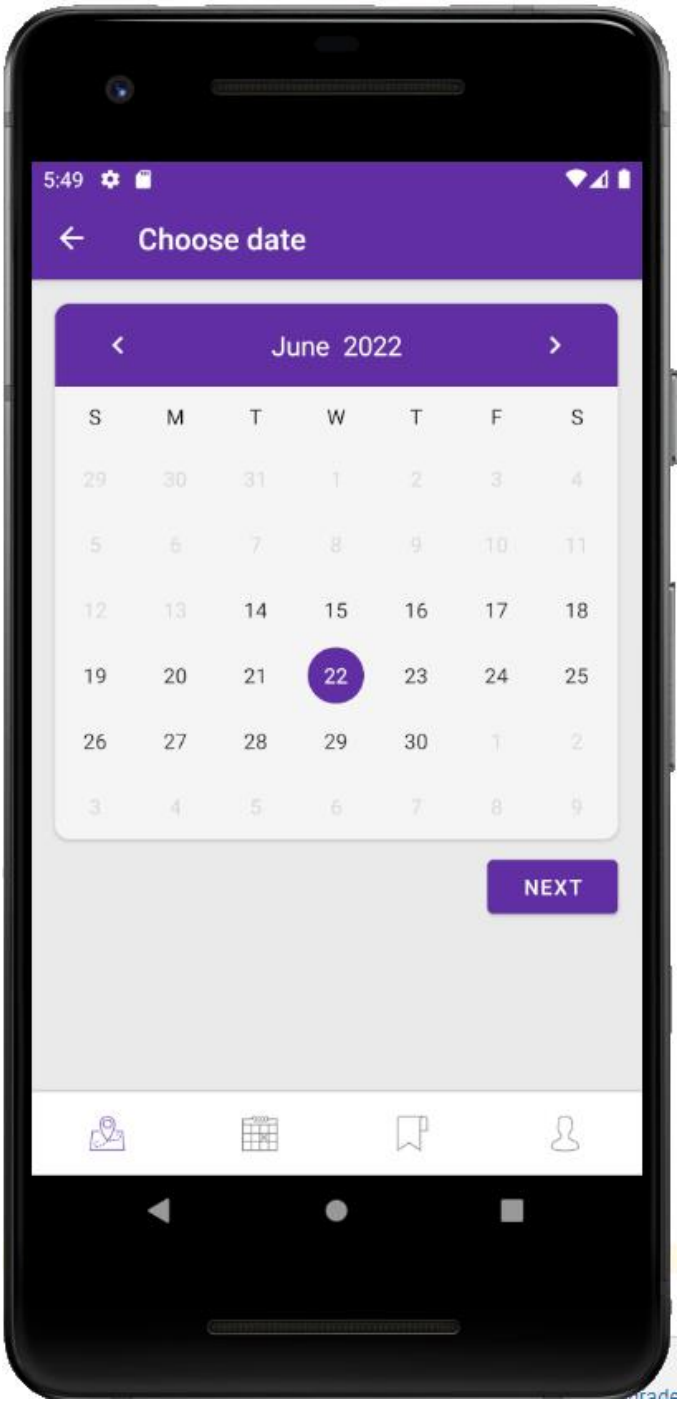

Рисунок 10.9 – Сторінка з календарем бронювання

Обравши дату, користувач вказує годину, на котру хоче забронювати столик. Бронювання триває від 1 до 2 годин. Бажана тривалість може бути обрана за допомогою перемикача часу. Сторінка вибору параметрів часу зображена на рисунку 10.10. Щоб перейти до наступного етапу бронювання, потрібно натиснути на кнопку “Next”.

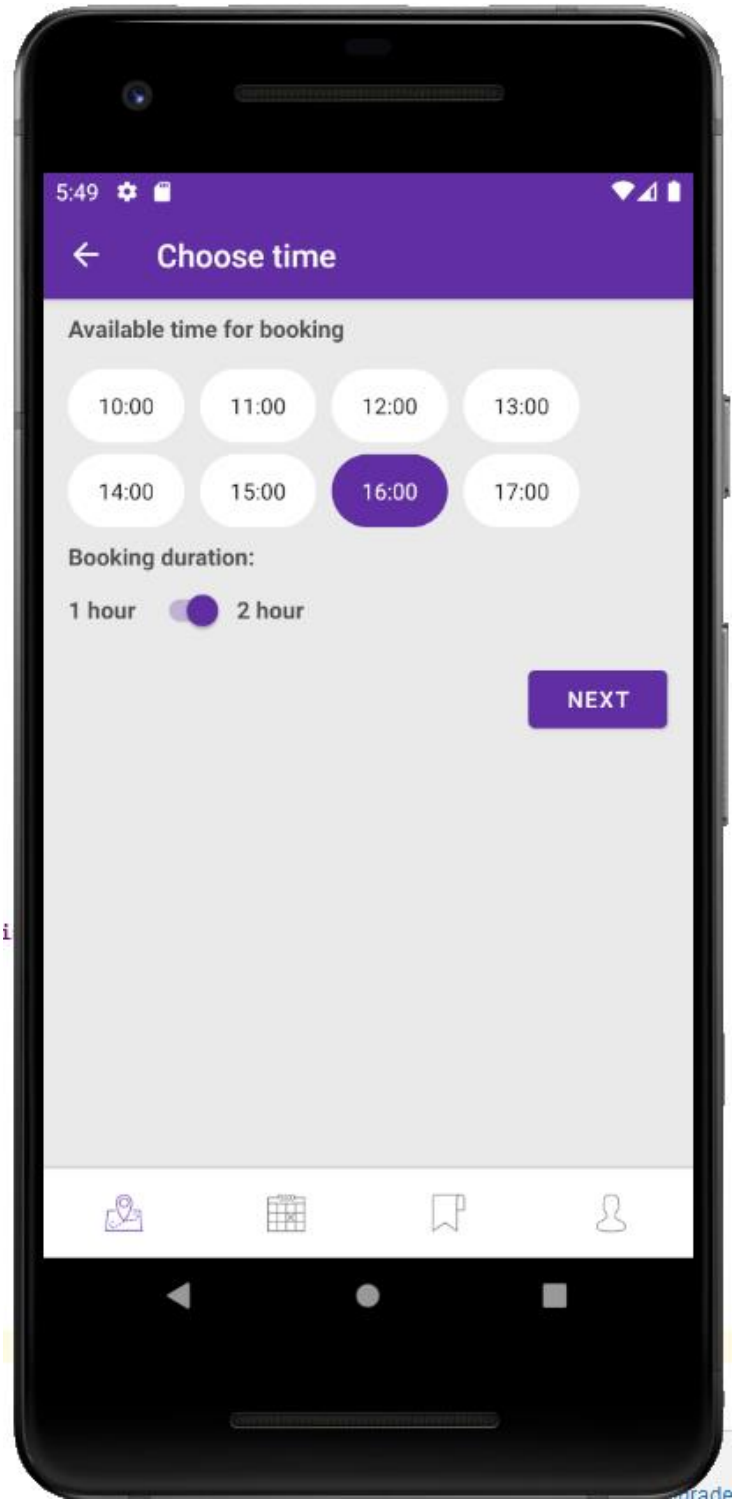

Рисунок 10.10 – Сторінка вибору параметрів часу

Обравши час бронювання та очікувану тривалість відвідування, користувач переходить на сторінку вибору бажаного столика, вигляд якої зображено на рисунку 10.11. Щоб перейти далі, потрібно натиснути кнопку “Next”.

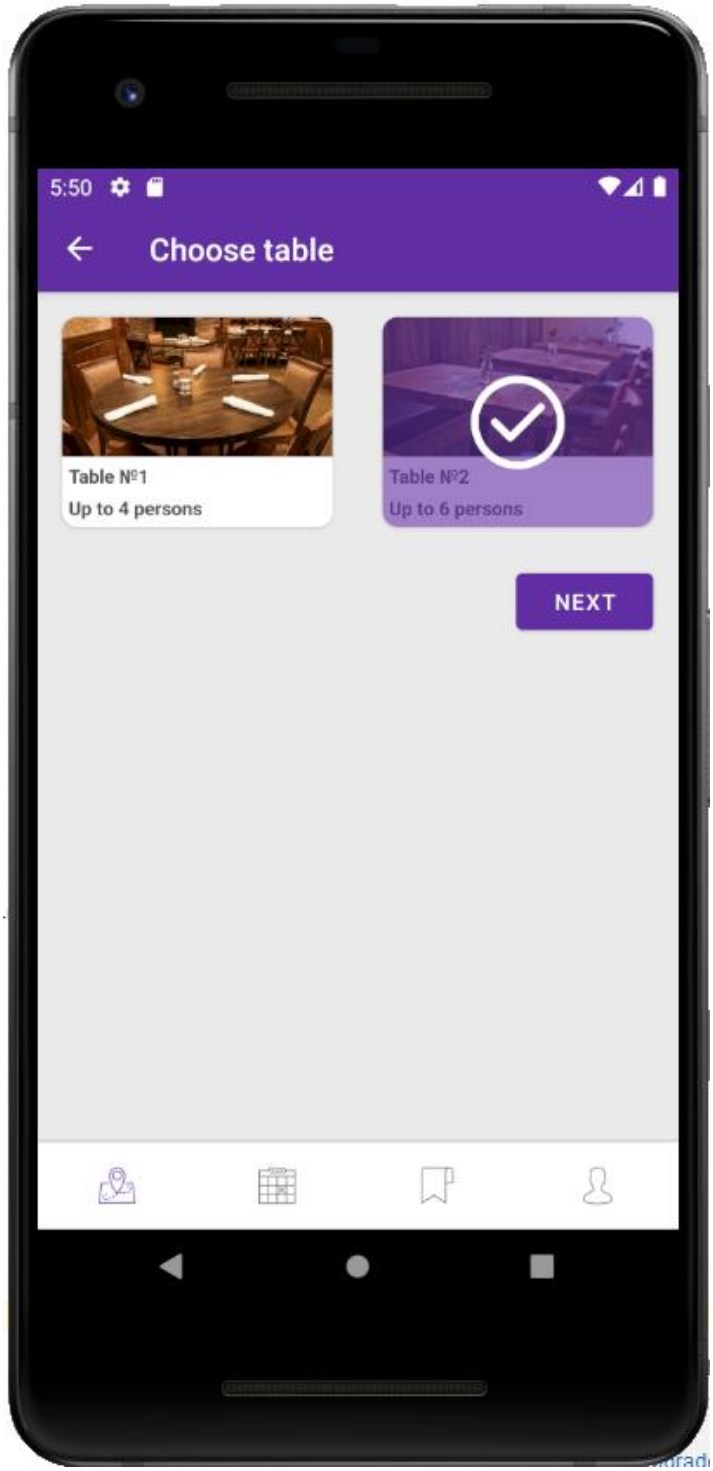

Рисунок 10.11 – Сторінка вибору столика

Після вибору столика, користувач потрапляє на сторінку вибору меню ресторану. Вона зображена на рисунку 10.12. Для переходу на сторінку підтвердження замовлення потрібно натиснути кнопку “Next”.

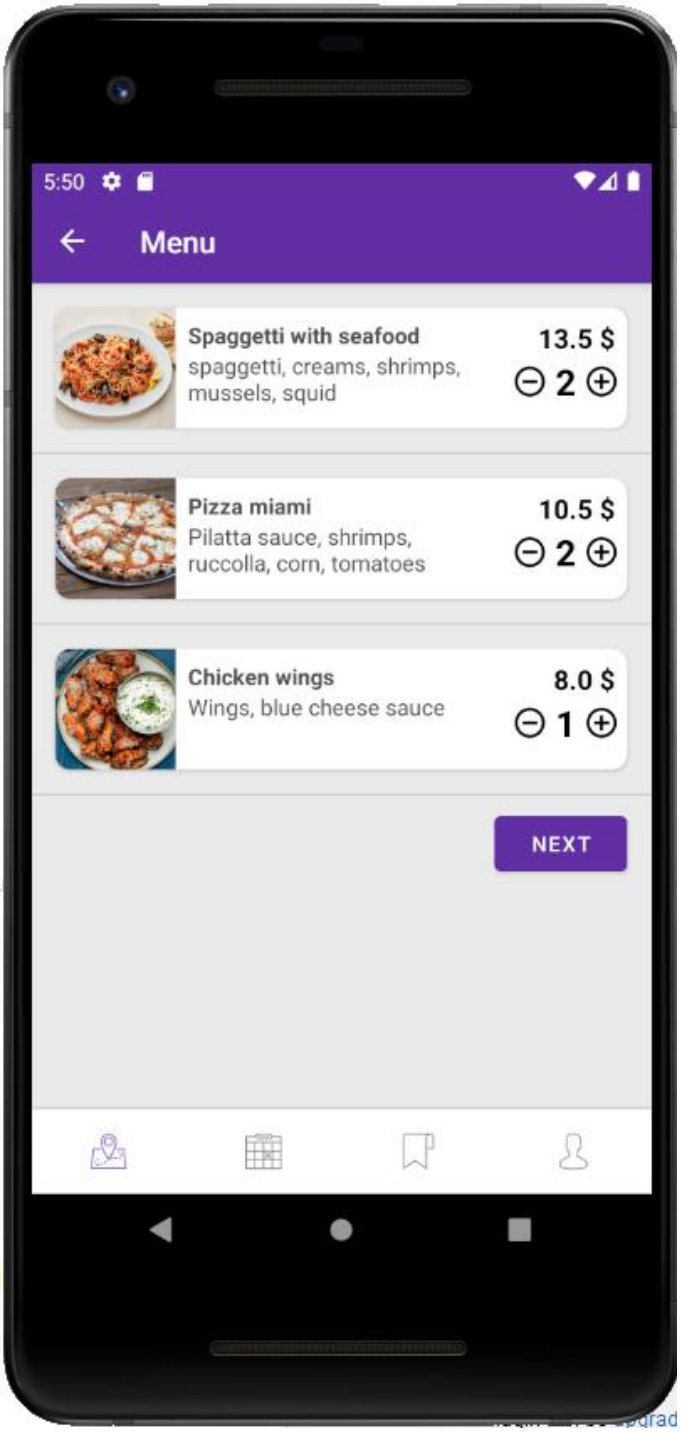

Рисунок 10.12 – Сторінка вибору меню ресторану

На сторінці підтвердження замовлення користувач переглядає інформацію про замовлення перед його підтвердженням. Її вигляд зображено на рисунку 10.13. Якщо замовлення задовольняє користувача, йому потрібно натиснути на кнопку “Confirm” для підтвердження.

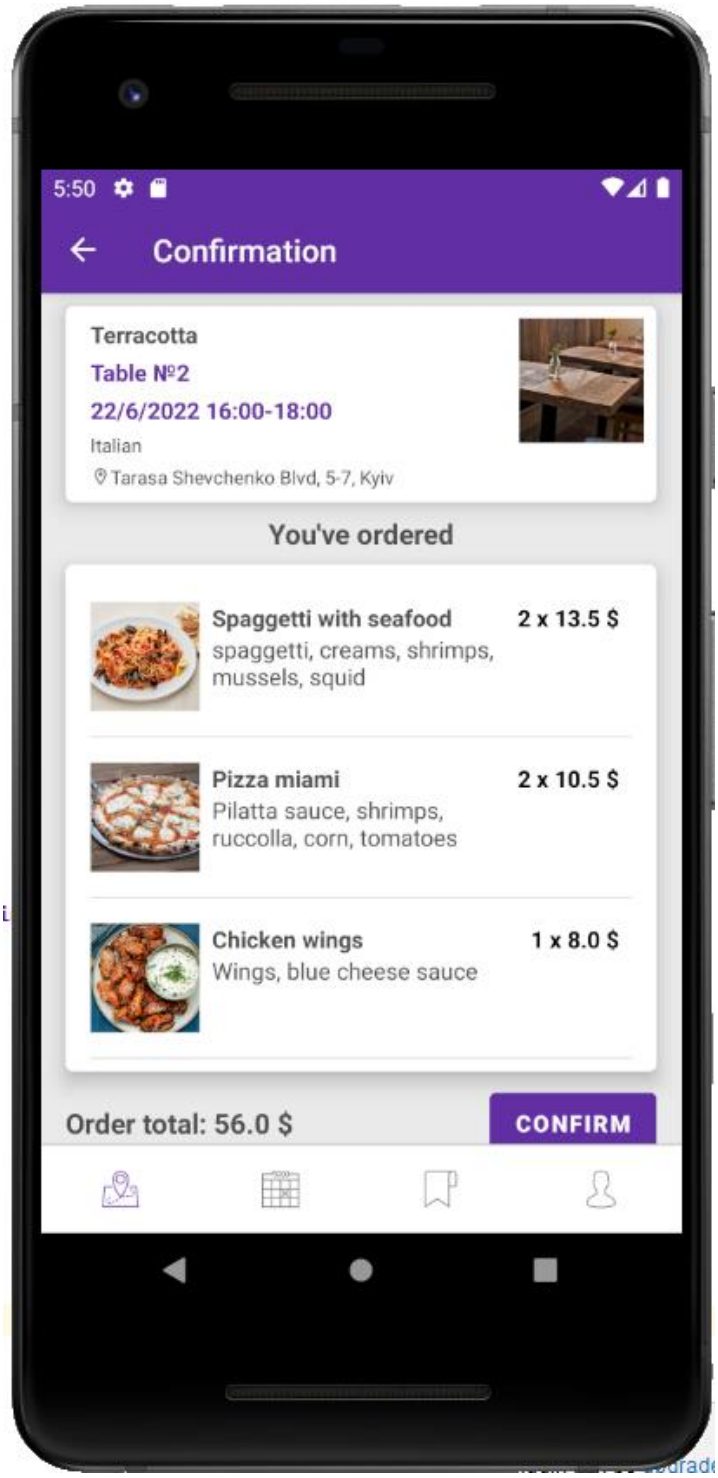

Рисунок 10.13 – Сторінка підтвердження замовлення

Після підтвердження замовлення користувач автоматично переходить на сторінку, де відображається повідомлення про успішне створення замовлення, а також кнопки “Bookings” та “Pay online”. Ця сторінка зображена на рисунку 10.14.

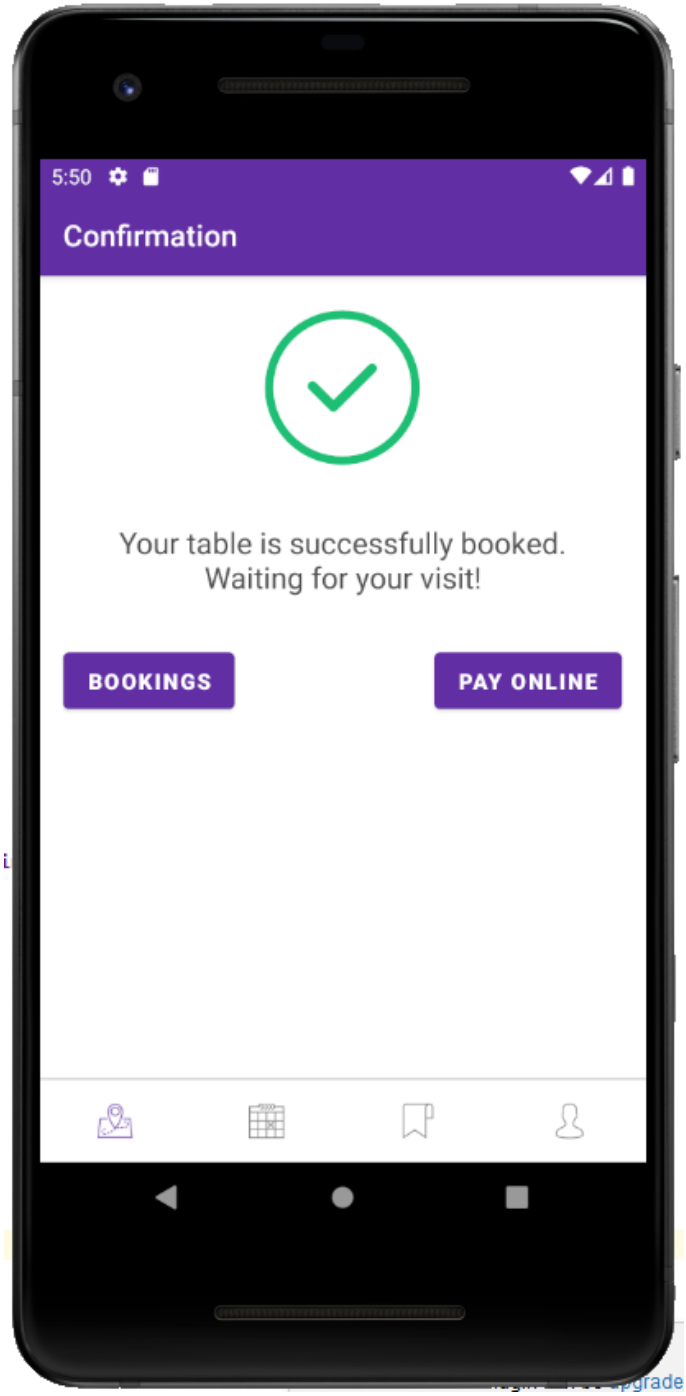

Рисунок 10.14 – Відображення успішного створення замовлення

Аби переглянути усі бронювання, створені користувачем, потрібно натиснути на кнопку “Bookings”. Користувач відразу потрапляє на сторінку з переліком зроблених бронювань, вигляд якої зображено на рисунку 10.15.

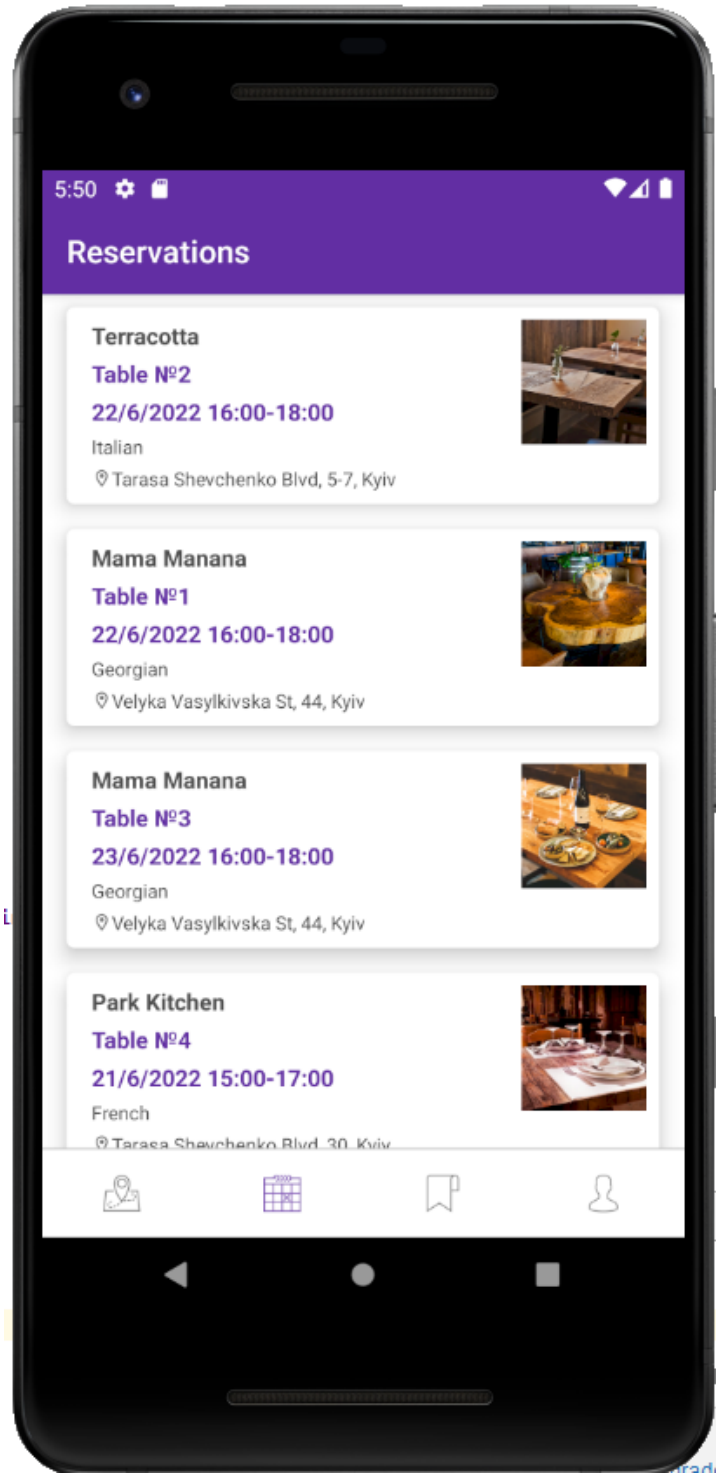

Рисунок 10.15 – Сторінка з переліком зроблених бронювань

Натиснувши на одне з бронювань, користувач потрапляє на сторінку деталей обраного бронювання. На цій сторінці відображено окрім загальної інформації, такої як час, дата, столик та назва ресторану, ще й замовлення користувача. Сторінка з деталями бронювання зображена на рисунку 10.16.

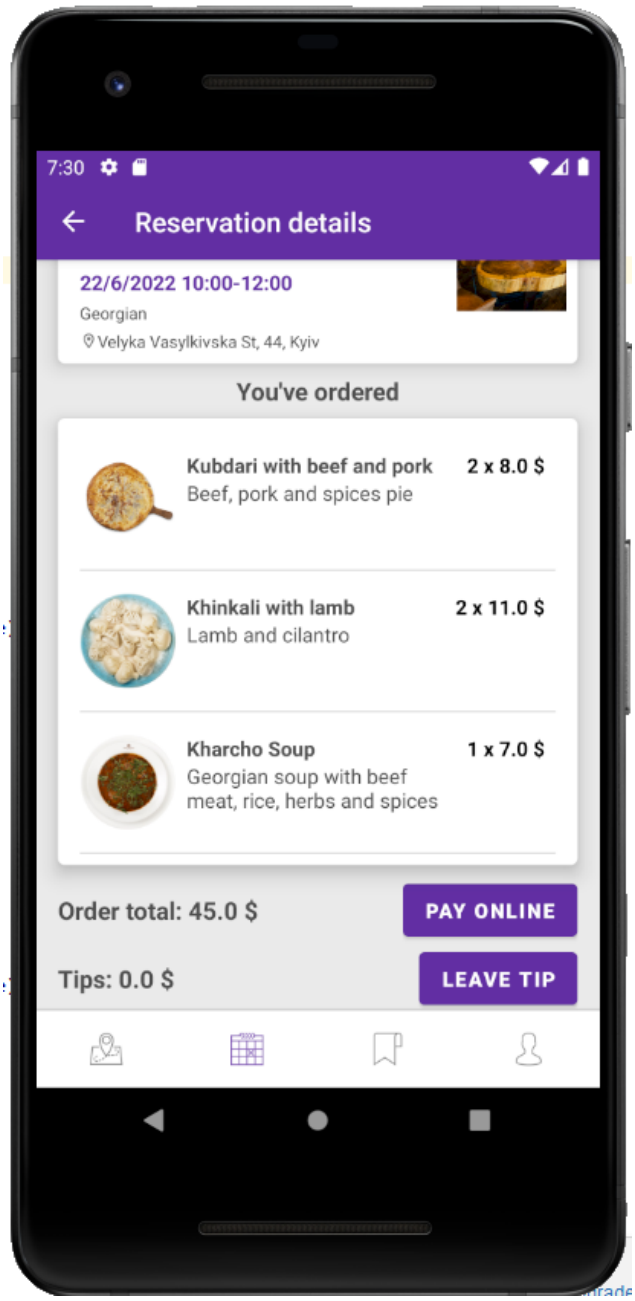

Рисунок 10.16 – Сторінка з деталями бронювання

Оплатити замовлення можна онлайн, натиснувши на кнопку “Pay online” зі сторінки відображення успішного створення замовлення (рис. 10.14) або зі сторінки з деталями бронювання (10.16). Користувач переходить на сторінку оплати, де може вибрати потрібну банківську картку, натиснувши на неї. Сторінка оплати зображена на рисунку 10.17. Для завершення операції оплати потрібно натиснути на кнопку “Pay”, де вказана сума замовлення.

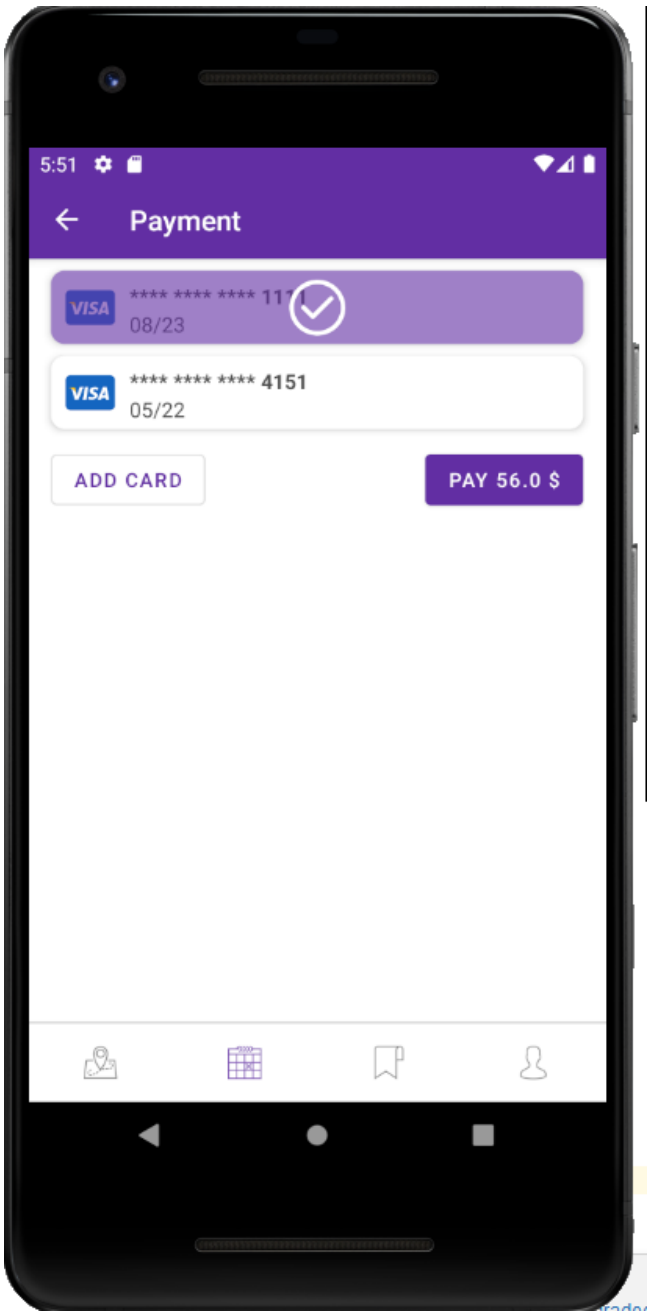

Рисунок 10.17 – Сторінка оплати

Також можна додати нову банківську карту, натиснувши на кнопку “Add card”. Після натискання користувач переходить на сторінку додавання картки, вигляд якої зображено на рисунку 10.18. Тут потрібно вказати всі дані нової картки та натиснути кнопку “Done” для підтвердження.

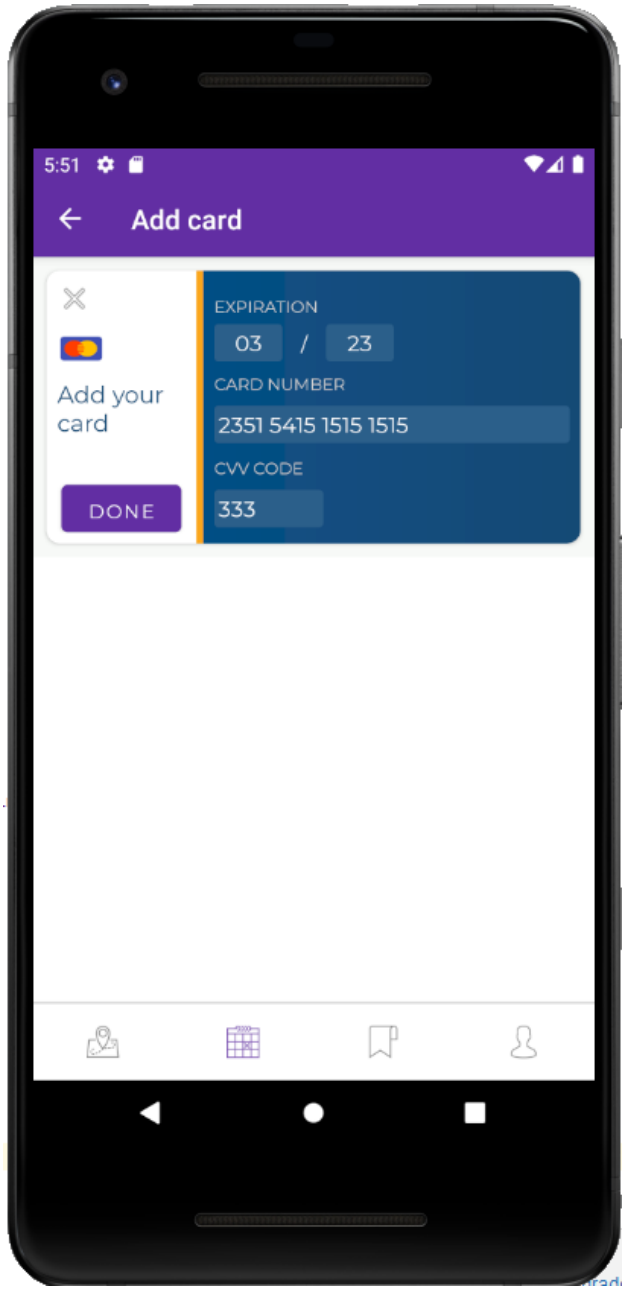

Рисунок 10.18 – Сторінка додавання картки

У системі є функція чайових, яка не є обов'язковою. Для того, аби залишити онлайн чайові за зроблене замовлення. Для цього на сторінці деталей бронювання (рис. 10.16) потрібно натиснути на кнопку “Leave tip”. Користувач буде автоматично направлений на сторінку оплати чайових, яка зображена на рисунку 10.19. Тут потрібно вказати суму чайових у полі “Tips value”, обрати банківську картку та натиснути кнопку “Pay”. Також є можливість додати нову картку для оплати, натиснувши кнопку “Add card”.

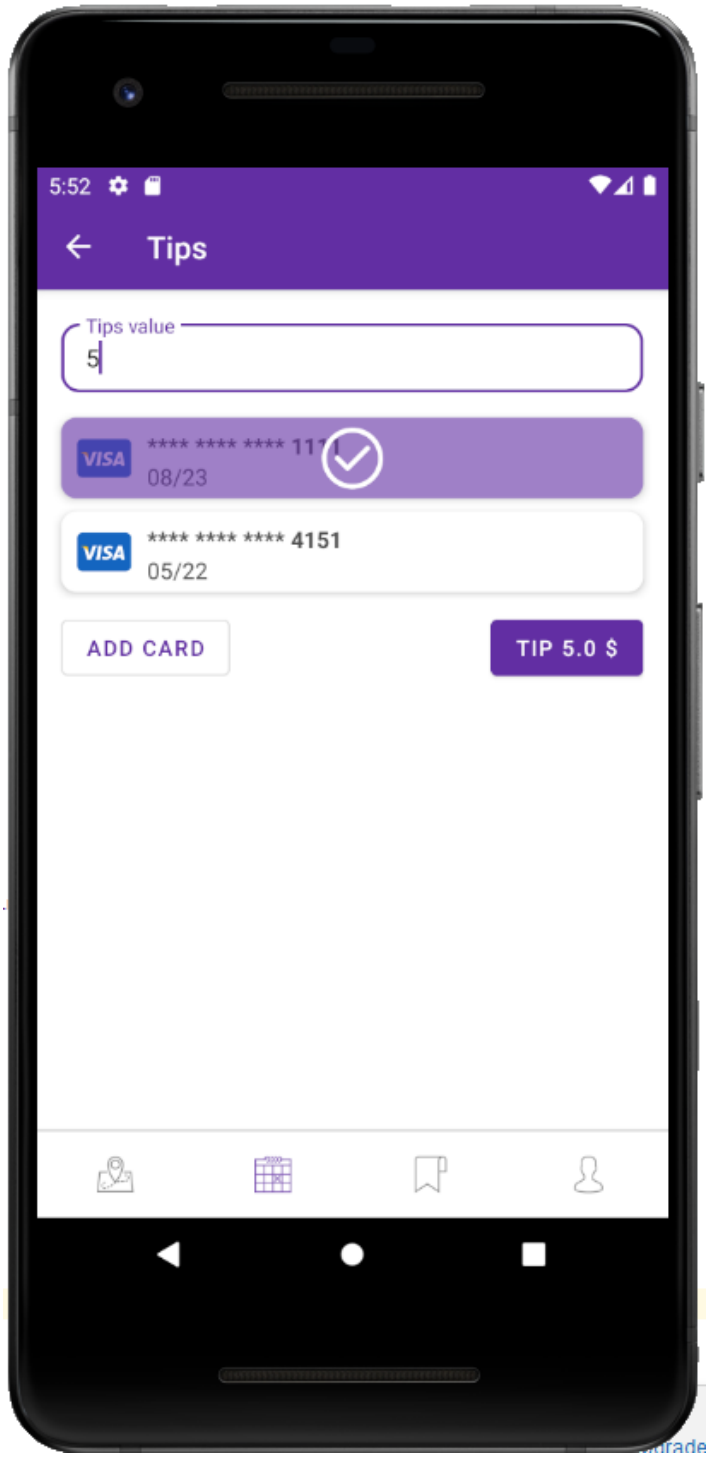

Рисунок 10.19 – Сторінка оплати чайових

Після оплати замовлення та чайових (за бажанням) статус замовлення зміниться на сторінці деталей бронювання, що вказано на зображенні 10.20.

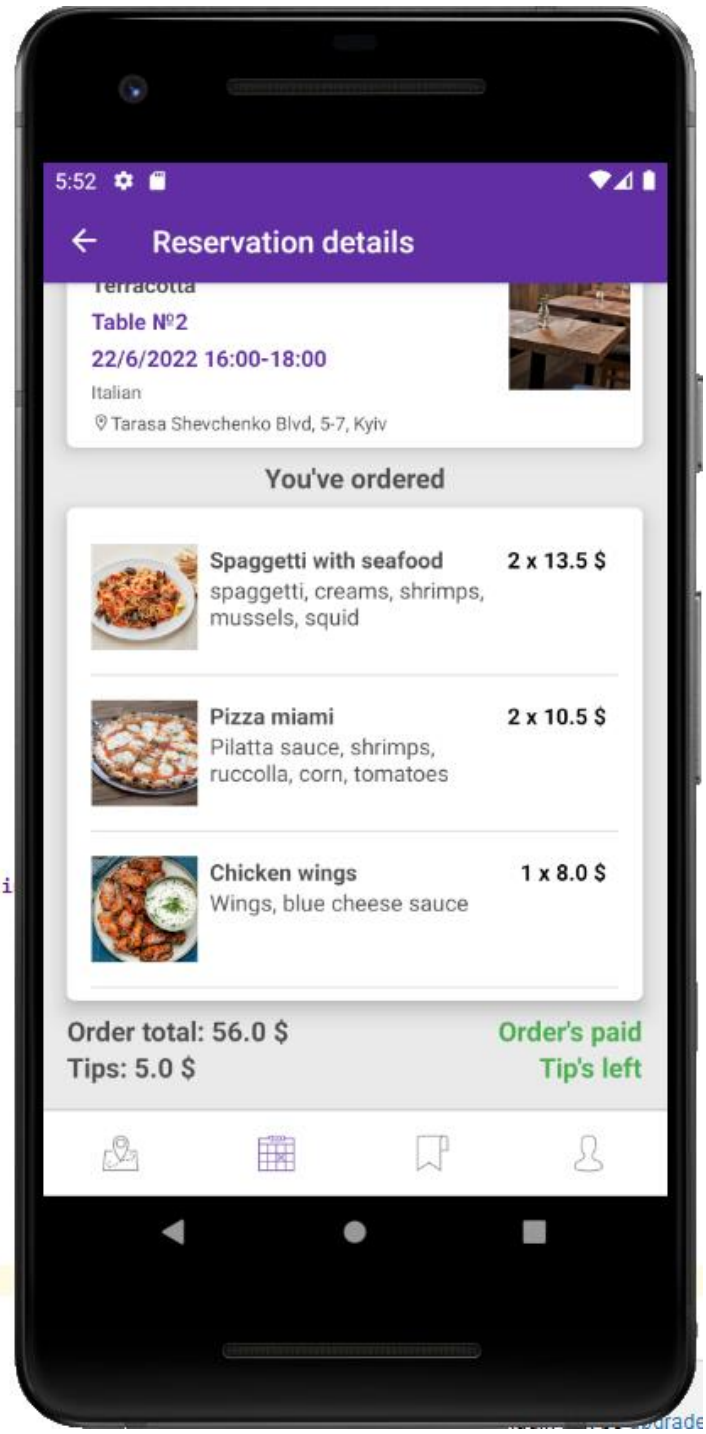

Рисунок 10.20 – Сторінка зі зміненим статусом замовлення

На нижній панелі екрану можна обрати елемент «Обране» для перегляду збережених ресторанів. Вигляд сторінки «Обране» зображено на рисунку 10.21.

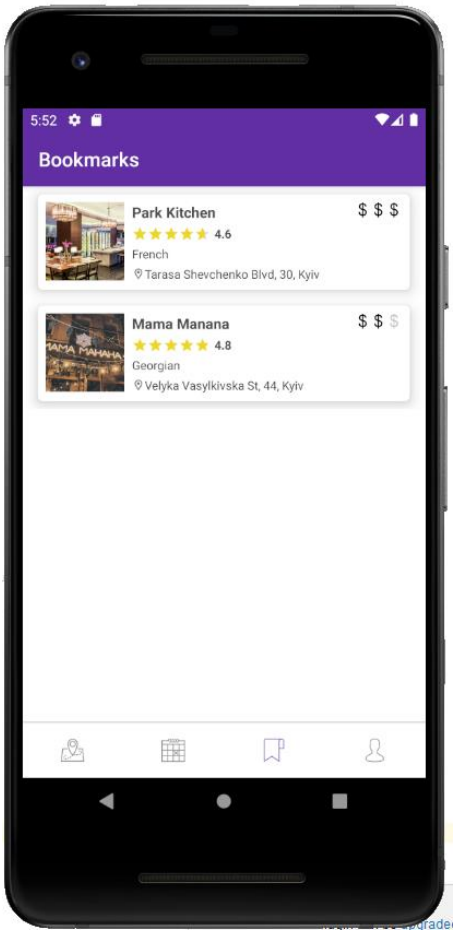

Рисунок 10.21 – Вигляд сторінки «Обране»

Останній елемент на нижній панелі екрану – кнопка профілю користувача. Натиснувши на неї, користувач потрапляє на сторінку свого профілю, де вказана його особиста інформація, а саме: фото користувача,адреса електронної пошти та актуальна банківська картка. Карту можна видалити, натиснувши на хрестик, що знаходиться праворуч на картці. Також є функція додавання карти, яку можна виконати, натиснувши на кнопку “Add card”.

Щоб вийти з облікового запису, потрібно натиснути на кнопку “Log out”. Фрагмент профілю зображений на рисунку 10.22.

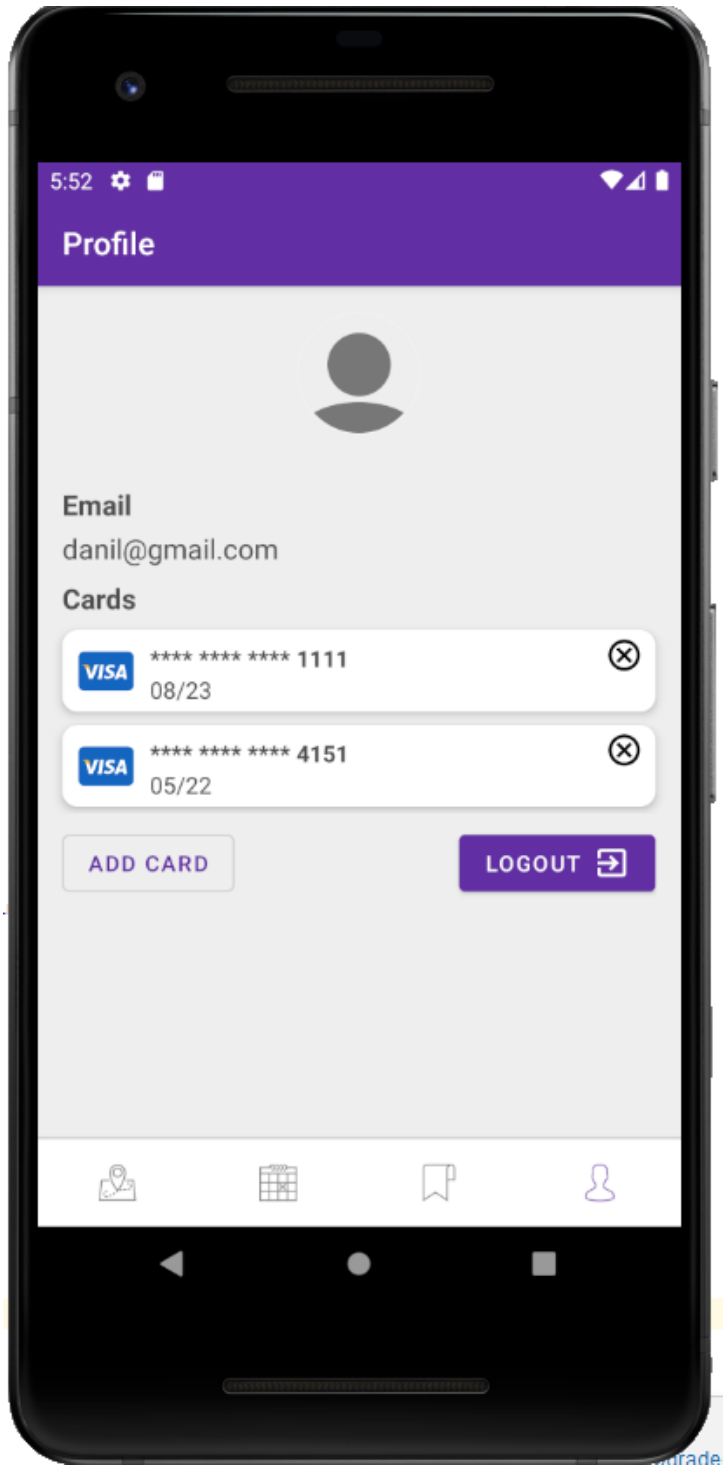

Рисунок 10.22 – Профіль користувача

ВИСНОВКИ

В результаті розробки індивідуального дослідницького проєкту створено мобільний додаток для резервування столиків у ресторанах. Однією з головних переваг програми є те, що вона поєднує в собі всі найкращі функції зі своїх аналогів.

На початку розробки проєкту встановлюються та описуються технічні характеристики початкової заявки. У процесі розробки було розглянуто існуючі рішення, щоб визначити їх сильні та слабкі сторони та розробити необхідний функціонал для проєктованого додатка.

Для опису процесу системи та для представлення взаємодій між компонентами та модулями було розроблено структурну. схему Вона показує основні елементи системи та як взаємодіють основні компоненти програми між собою. Діаграма дає можливість зрозуміти, як додаток працює з сервером та базою даних

У розділі програмного та технічного забезпечення було описано інструменти розробки, бібліотеки та патерни архітектури.

Було розроблено UML діаграми додатку різного виду, такі як діаграма варіантів використання, діаграма послідовності та діаграма класів. Також було зроблено окрему діаграму послідовності для процесу бронювання та діаграму діяльності для процесу авторизації.

Результатом виконання індивідуального дослідницького проєкту є програмне забезпечення, що надає можливість автоматизувати процес бронювання столиків у закладах харчування і використовувати додаткові корисні функції при бронюванні. Розроблений додаток повністю відповідає технічним умовам і без проблем може бути використаний по призначенню.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. How reservation startup Tock saved the restaurant industry in the nick of time. URL: https://www.fastcompany.com/90600405/tock-most-innovative-companies-2021

2. Частка безготівкової виручки у ресторанах України: дослідження Poster. URL: https://joinposter.com/ua/post/chastka-bezhotivkovoyi-vyruchky-u-restoranakh-Ukrayiny

3. Jira для команд разработки ПО. URL: https://www.atlassian.com/ru/software/jira/guides/use-cases/what-is-jira-used-for#jira-for-software-development-teams

4. Сравнение Scrum и Kanban. URL: https://www.atlassian.com/ru/agile/kanban/kanban-vs-scrum

5. What is version control. URL: https://www.atlassian.com/git/tutorials/what-is-version-control

6. About Android Studio IDE. URL: https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio

7. What Does Android SDK Mean. URL: https://www.techopedia.com/definition/4220/android-sdk

8. Model-View-ViewModel (MVVM) Definition. URL: https://www.techtarget.com/whatis/definition/Model-View-ViewModel

9. MVVM Architecture. Step by step guide. URL: https://blog.mindorks.com/mvvm-architecture-android-tutorial-for-beginners-step-by-step-guide

10. Kotlin docs. URL: https://kotlinlang.org/docs/home.html

11. Definition of the JS Programming Language. URL: https://www.freecodecamp.org/news/what-is-javascript-definition-of-js/

12. What is Node.js. URL: https://www.techtarget.com/whatis/definition/Nodejs#:~:text=James%20Denman-,Node.,feeds%20and%20web%20push%20notifications.

13. How do APIs work. URL: https://www.redhat.com/en/topics/api/what-are-application-programming-interfaces

14. What are XML’s advantages over HTML. URL: https://hbr.org/2000/07/explaining-xml

15. MySQL database implementation. URL: https://www.123-reg.co.uk/support/servers/what-is-mysql-and-why-do-i-need-it/

16. SharedPreferences. Сохранение данных в постоянное хранилище Android. URL: https://www.fandroid.info/sharedpreferences-sohranenie-dannyh-v-postoyannoe-hranilishhe-android/

17. Android Navigation and bottom navigation view. URL: https://www.fandroid.info/17-android-bottom-navigation/