Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

# КВАЛІФІКАЦІЙНА РОБОТА

## Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Комп'ютерна система безконтактного замовлення в закладах харчування

(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-6
Денис ГОЛОПЬОРОВ
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Віталій МАРТОВИЦЬКИЙ
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

# ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Голопьорову Денису Максимовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Комп'ютерна система безконтактного замовлення в закладах харчування

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи Приклади схожих мобільних застосунків

4. Перелік питань, що потрібно опрацювати у роботі

1) Аналіз вимог до програмного забезпечення: основні визначення та терміни, опис предметного середовища, огляд існуючих технічних рішень та відомих програмних продуктів, розробка функціональних та нефункціональних вимог

2) Моделювання та конструювання програмного забезпечення: моделювання та аналіз програмного забезпечення, засоби розробки, технічні рішення, архітектура програмного забезпечення

3) Розгортання та впровадження програмного забезпечення

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій 13 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1 )

| Найменування розділу | Консультант (посада, прізвище, ім'я, по батькові) | Позначка консультанта про виконання розділу | |
|---|---|---|---|
| | | підпис | дата |
| | | | |
| | | | |

## КАЛЕНДАРНИЙ ПЛАН

| № | Назва етапів роботи | Строк / терміни виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз проблеми та огляд існуючих рішень | 27.05.25-30.05.25 | |
| 2 | Вибір технології розробки та | 31.05.25-02.06.25 | |
| | інструментальних засобів | | |
| 3 | Розробка алгоритмічного забезпечення | 03.06.25-05.06.25 | |
| 4 | Розробка програмних модулів | 06.06.25-09.06.25 | |
| 5 | Відлагодження програмних модулів | 10.06.25-11.06.25 | |
| 6 | Оформлення матеріалів кваліфікаційної роботи | 12.06.25-13.06.25 | |
| 7 | Подання кваліфікаційної роботи керівникові та | 13.06.25-14.06.25 | |
| | її попередній захист | | |
| 8 | Подання кваліфікаційної роботи на рецензу- | 15.06.22-16.06.25 | |
| | вання | | |
| | | | |

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Віталій МАРТОВИЦЬКИЙ
(підпис) (посада, власне ім'я, прізвище)

# РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 56 с., 16 рис., 2 табл., 1 дод., 12 джерел.

КОМП’ЮТЕРНА СИСТЕМА, ІНТЕРНЕТ, FLUTTER, ANDROID, FIREBASE, СКБД, NOSQL.

Метою кваліфікаційної роботи є розробка застосунку для безконтактних замовлення в закладах харчування. Метою роботи є скорочення часу очікування замовлення в закладах харчування за рахунок зменшення часу на пошук відповідного місця, спілкування з персоналом закладів, автоматизації процесів створення та обробки замовлень.

У ході виконання кваліфікаційної роботи було проведено огляд існуючих рішень, було описано існуючі застосунки зі схожою функціональністю та представлено їх переваги і недоліки. Описано основні засоби розробки застосунку і визначено вимоги до технічного забезпечення. Описано архітектуру та реалізацію програмного забезпечення.

# ABSTRACT

Bachelor’s thesis: 56 pages, 16 figures, 2 tables, 1 appendices, 12 sources.

COMPUTER SYSTEM, INTERNET, FLUTTER, ANDROID, FIREBASE, DBMS, NOSQL.

The major goal of this thesis is to develop an application for remote orders in catering establishments. The aim of the work is to reduce the waiting time for orders in catering establishments by reducing the time to find a suitable place, communicate with the staff of establishments, automate the processes of creating and processing orders.

In the course of the qualification work, an overview of existing solutions was conducted, existing applications with similar functionality were described and their advantages and disadvantages were presented. The main means of application development are described and the requirements for technical support are defined. Describes the architecture and implementation of the software.

ЗМІСТ

# СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API – Application Programming Interface – сукупність засобів та правил, що вможливлюють взаємодію між окремими складниками програмного забезпечення

JSON – JavaScript Object Notation – нотація об'єкта javascript

REST – Representational State Transfer, підхід до архітектури мережевих протоколів

UX – user experience design – дизайн досвіду користувача

БД – база даних

ПЗ – програмне забезпечення

СУБД – система управління базами даних

# ВСТУП

В умовах ринкової економіки поряд із харчовою промисловістю та сферою торгівлі активно формується й розвивається нова галузь – система громадського харчування. Вона виступає альтернативою домашньому приготуванню їжі, забезпечуючи масове виробництво страв за допомогою сучасних технологічних рішень та обладнання.

Громадське харчування виникло як наслідок розподілу праці у суспільстві та стало чинником економії трудових ресурсів. Воно створює зручні умови для прийому їжі у безпосередній близькості до місць роботи, навчання чи відпочинку. В умовах сучасного ритму життя багато людей не мають змоги приділяти достатньо уваги своєму харчуванню, режиму прийому їжі, тому звертаються до швидких рішень – фастфудів та напівфабрикатів. Через прагнення до кар'єрного росту люди часто нехтують речами, які не можна повернути за жодні гроші – здоров'ям і часом.

Розвиток закладів громадського харчування дає змогу істотно збільшити вільний час працівників, що позитивно впливає на їх всебічний розвиток – як фізичний, так і духовний. Це, у свою чергу, веде до зростання ефективності праці в масштабах суспільства. Подібна форма організації харчування допомагає швидше відновлювати витрачену енергію, підтримує працездатність людей.

Розміщення об’єктів громадського харчування поблизу заводів, фабрик, навчальних закладів та наукових інституцій сприяє дотриманню раціонального режиму прийому їжі, а також дозволяє заощаджувати час для відпочинку та особистих справ.

Саме з огляду на вищезазначене, основною метою цього дипломного проєкту є створення мобільного застосунку, що дозволить здійснювати віддалене замовлення їжі в закладах громадського харчування.

За інформацією аналітичної компанії Statista з Німеччини, яка

займається збором статистичних даних, у 2020 році кількість активних смартфонів у світі сягнула 3,6 мільярда одиниць. Очікується, що у 2023 році ця цифра зросте до приблизно 4,3 мільярда [1]. Такий показник свідчить про широке поширення мобільних пристроїв – більшість людей мають хоча б один, а дехто – кілька гаджетів.

У зв’язку з цим розробка мобільних застосунків набуває особливої актуальності, і ця тенденція продовжуватиме зростати. Проте створення окремих (нативних) додатків для різних платформ – iOS та Android – є доволі затратним процесом. Це потребує наявності щонайменше двох окремих команд розробників, що є суттєвим фінансовим тягарем для малих компаній або стартапів. Виходом у такій ситуації є кросплатформенна розробка. Одним із популярних інструментів для цього є Flutter.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ОПИС ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## 1.1 Аналіз предметної області

Сьогодні ресторанна індустрія в Україні активно розвивається. Відкривається все більше закладів з різними кухнями світу, тематичними інтер'єрами та розважальними програмами. Такий широкий вибір дозволяє клієнтам обирати місце для дозвілля відповідно до своїх смаків і фінансових можливостей. Водночас власники закладів змушені постійно вдосконалювати бізнес-процеси, підвищувати якість сервісу, впроваджувати сучасні технології та зміцнювати свої позиції на ринку.

Автоматизація закладів громадського харчування вже давно стала необхідністю, а не просто бажаною інновацією. Головна проблема полягає у виборі програмного рішення, яке відповідатиме формату та масштабам підприємства. Якщо для великих ресторанів існують потужні системи автоматизації, вартість яких стартує з 20 000 грн (близько 1000 дол. США) і може сягати 80 000 грн (до 4000 дол. США), то для невеликих кафе чи бістро такі витрати часто є недосяжними. При додаванні витрат на обладнання, загальні інвестиції в автоматизацію можуть коливатись від 40 000 до 150 000 грн (2000–5500 дол. США).

Внаслідок цього виникла потреба в доступному та простому у використанні програмному забезпеченні, яке можна легко впровадити. Одним із таких рішень є система Stuffer, спеціально розроблена для малого та середнього бізнесу в сфері громадського харчування.

«Stuffer» ідеально підходить для автоматизації кафе, піцерій, бістро, фаст-фудів, барів, їдалень, а також невеликих ресторанів та закладів розваг. Особливістю цієї програми є можливість ефективного управління діяльністю закладу з боку власника або керуючого.

Ефективна робота підприємств ресторанного господарства базується на гармонійному поєднанні таких елементів: цінова політика, якість страв, обслуговування, атмосфера, асортимент послуг. Важливо, що в сфері харчових послуг виробництво, реалізація та споживання часто відбуваються одночасно і в одному місці. Послуги можуть мати як матеріальну, так і нематеріальну складову.

У нормативних документах, що регулюють роботу закладів харчування, визначено перелік послуг, які мають надаватися:

- послуги харчування;
- виготовлення кулінарних і кондитерських виробів;
- організація процесу споживання;
- реалізація готової продукції;
- організація дозвілля;
- консультаційні послуги тощо.

Відносини між споживачами й підприємствами регулюються «Правилами роботи закладів ресторанного господарства» (наказ Міністерства економіки від 24.07.2002 № 219), а також законами «Про захист прав споживачів» і «Про безпечність та якість харчових продуктів». Послуги закладу визначаються його типом, а для ресторанів і барів – також класом. У разі потреби діяльність сертифікується згідно з державними стандартами. Для продажу алкоголю чи тютюнових виробів обов’язково потрібна ліцензія.

Сьогодні ресторани та кафе є важливою частиною соціального життя міста. Щоб залишатися конкурентоспроможними, власники активно запроваджують програми лояльності: дисконтні картки, подарунки до дня народження, бонусні страви при великих замовленнях, дегустації, спеціальні рецепти від шеф-кухаря, вікторини з призами тощо.

Сучасний заклад харчування – це не лише місце, де можна поїсти, а й простір для відпочинку. Інтер’єр, оформлений у стилі певної епохи або культури, декоративні елементи (фонтани, каміни, акваріуми) створюють унікальну атмосферу. Престиж закладу визначається рівнем сервісу,

кулінарною майстерністю, асортиментом, доброзичливим прийомом та комфортом.

Однак високий рівень обслуговування потребує значних витрат, зокрема на навчання персоналу. Багато власників інвестують у тренінги та курси для офіціантів, але результат не завжди виправдовує очікування. Недосвідченість працівників, порушення стандартів обслуговування, затримки – усе це негативно впливає на якість послуг.

Автоматизація процесів обслуговування може значно покращити ситуацію. Використання сучасних IT-рішень допомагає підвищити ефективність роботи персоналу, зменшити час обслуговування клієнтів, усунути черги.

Інформаційні технології класифікуються за рядом критеріїв: способами реалізації в системах, охопленням управлінських функцій, типами користувацького інтерфейсу, рівнем інтеграції з мережею.

Управління – ключова функція будь-якої організації. Система управління охоплює планування, облік, прогнозування, контроль, аналіз і коригування. Усі ці процеси супроводжуються обміном інформацією як всередині підприємства, так і з його зовнішнім середовищем.

Автоматизовані інформаційні системи – це поєднання методів, моделей, технічних засобів і людських ресурсів, спрямованих на обробку інформації й ухвалення управлінських рішень.

З технічного боку, інформаційна система – це сукупність елементів, що працюють разом для збору, обробки, збереження й передачі даних, які допомагають у прийнятті управлінських рішень. Вони також сприяють аналізу проблем, моделюванню процесів, створенню нових продуктів.

У великих організаціях функціонують окремі інформаційні системи для основних сфер: маркетингу, продажів, виробництва, обліку, фінансів і HR. У межах кожної сфери можуть існувати ще більш вузькі спеціалізовані підсистеми, наприклад: управління запасами, планування виробництва, технічне обслуговування тощо.

## 1.2 Огляд аналітичних існуючих рішень

Poster – це система, яка забезпечує автоматизацію процесів у кафе чи ресторанах та надає широкий спектр функціональних можливостей. Цей сервіс дозволяє ефективно організувати всі аспекти роботи закладу, відповідаючи сучасним стандартам якості обслуговування у сфері ресторанного бізнесу.

Серед основних переваг програмного комплексу Poster можна виокремити наступні:

- можливість доступу з будь-якого місця у світі;
- стабільна робота навіть за відсутності інтернет-з’єднання;
- легкість у встановленні та освоєнні;
- прийнятна вартість;
- зручний інтерфейс для роботи офіціанта.

Доступність з будь-якого місця. Оскільки Poster є хмарним рішенням, у користувача завжди є можливість переглядати статистику, контролювати залишки на складі та стежити за фінансовими показниками бізнесу з будь-якої точки світу, де є інтернет-зв’язок.

Працездатність без інтернету. Навіть за умови тимчасової відсутності інтернету в закладі, система продовжує працювати в автономному режимі: офіціанти можуть приймати замовлення, а чеки – друкуватися. Щойно з’єднання відновлюється, усі дані автоматично синхронізуються з хмарним сховищем.

На рисунку 1.1 представлено інтерфейс офіційного сайту програмного забезпечення Poster, який демонструє зручність та інтуїтивність використання сервісу.

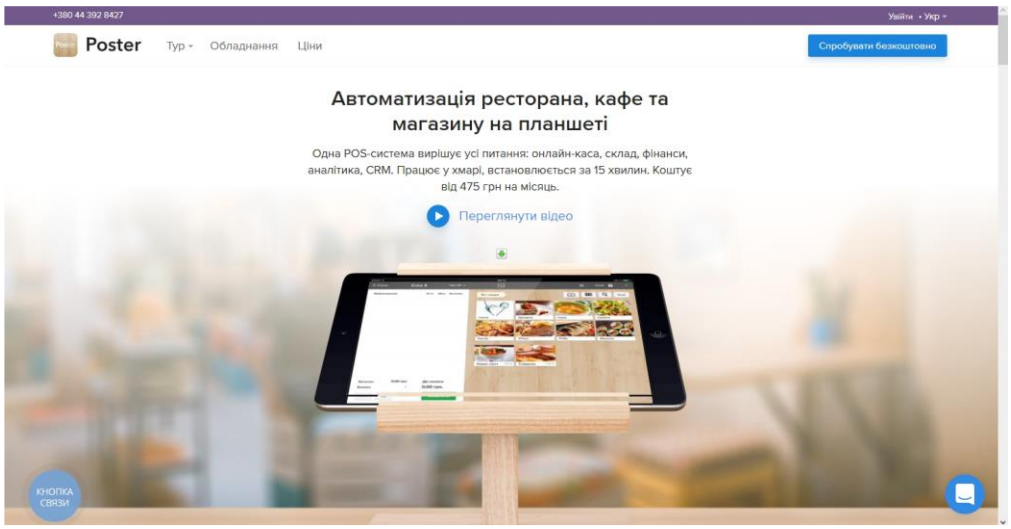


Рисунок 1.1 – Зовнішній вигляд офіційної сторінки Poster

Простота встановлення та легкість у навчанні. Команда розробників Poster прагне створювати продукти, які є водночас інтуїтивними та функціональними. Завдяки простому інтерфейсу та зручному налаштуванню, система не потребує тривалого навчання персоналу – працівники можуть почати роботу вже через кілька хвилин після встановлення.

Доступність за ціною. Poster працює за принципом підписки, що дозволяє уникнути великих одноразових витрат на впровадження автоматизації. Мінімальний тариф складає лише 475 грн на місяць, що робить систему вигідною для малого та середнього бізнесу.

Оптимізоване робоче середовище для офіціанта. Головне завдання персоналу – ефективна взаємодія з гостями, а не витрачання часу на розбір складного ПЗ. Саме тому інтерфейс Poster розроблений максимально простим, швидким та зручним у користуванні як для офіціантів, так і для барменів чи бариста.

Комплектація програмного забезпечення Poster включає такі елементи:

- термінал для роботи офіціанта;
- грошові ящики для зберігання виручки;
- фіскальні реєстратори;
- сканери для зчитування штрих-кодів;
- принтери для друку чеків.

На рисунках 1.2 та 1.3 показано приклади обладнання з підтримкою фіскалізації. Poster дозволяє працювати як з фіскальним, так і з нефіскальним обладнанням. Сервіс сумісний з усіма принтерами чеків, які підтримують протокол ESC/POS та мають підключення через Ethernet-інтерфейс.

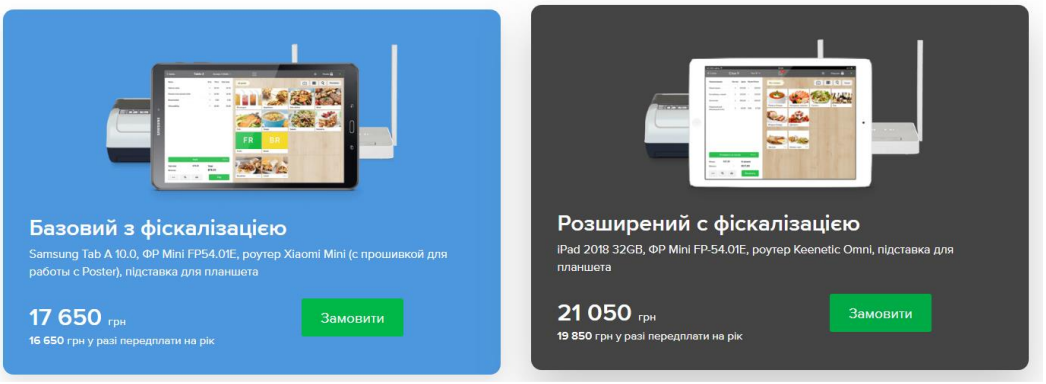


Рисунок 1.2 – Зразки обладнання з підтримкою фіскалізації

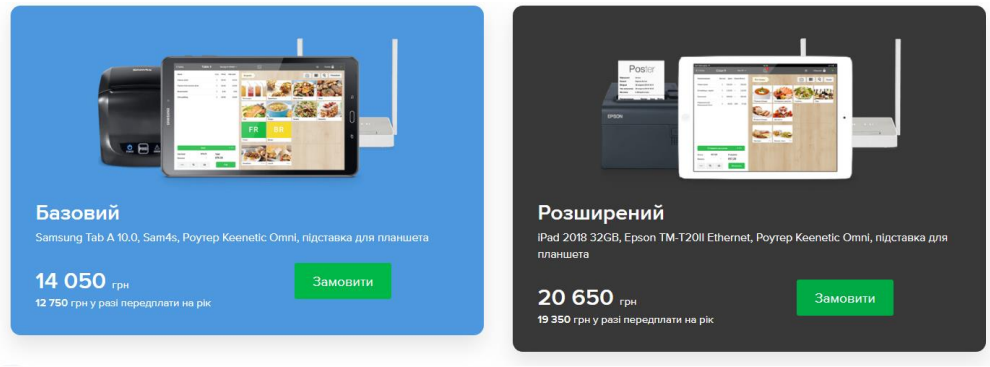


Рисунок 1.3 – Зразки обладнання без підтримкою фіскалізації

Термінал офіціанта. Робоче середовище для обслуговуючого персоналу може бути запущене на будь-якій популярній операційній системі, включаючи iOS, Android, Windows та macOS, що забезпечує гнучкість у виборі обладнання. Грошові ящики. Для організації розрахунків готівкою використовується грошова скринька, яка автоматично відкривається при друкуванні фіскального чека – за умови підключення до чекового принтера.

Фіскальні реєстратори. Сервіс Poster відповідає вимогам Федерального закону № 54 (ФЗ-54) щодо фіскалізації в торгівлі. Система підтримує роботу з такими популярними брендами обладнання, як Atol, Штрих-М та Viki Print.

Сканери штрих-кодів. Poster також сумісний із пристроями для зчитування штрих-кодів. Передбачена підтримка Bluetooth-сканерів для платформ iOS та Android, а також USB-сканерів для систем на базі Windows.

Чекові принтери. Сервіс інтегрується з більш ніж 20 моделями термопринтерів, серед яких Epson, Star, XPrinter, Sam4s, Mprint, Mini та інші. Це дозволяє адаптувати систему під різні технічні умови закладу.

Yelp – один із провідних застосунків у ресторанній індустрії. Незважаючи на значну інформаційну увагу навколо цього сервісу, його популярність зумовлена широким функціоналом та високою якістю реалізації. На рисунку 1.4 представлено інтерфейс мобільного додатку Yelp, який широко використовується як в США, так і у світі.


Рисунок 1.4 – Зовнішній вигляд інтерфейсу мобільного застосунку Yelp

Yelp – це одна з найпопулярніших світових платформ для пошуку та оцінювання закладів громадського харчування та інших видів бізнесу. Сервіс надає доступ до понад 50 мільйонів відгуків, що дозволяє користувачам легко знаходити ресторани поблизу, ознайомлюватися з численними рецензіями, переглядати фотографії закладів та знаходити актуальні пропозиції. Зручна система фільтрів дає змогу звузити результати пошуку за

критеріями: відстань, цінова категорія та рейтинг, що дозволяє прийняти обґрунтоване рішення при виборі закладу.

Платформа була заснована у 2004 році Джеремі Стоппельманом та Расселом Сіммонсом – колишніми співробітниками компанії PayPal. У наступні роки Yelp активно розвивався, отримав кілька інвестиційних раундів, і вже до 2010 року його річний дохід досяг 30 мільйонів доларів, а на сайті було опубліковано близько 4,5 мільйона відгуків. У період з 2009 по 2012 рік компанія почала активно розширювати свою присутність на європейському та азійському ринках. Зокрема, в 2009 році велись перемовини з Google щодо можливого придбання. У березні 2012 року Yelp вийшов на фондовий ринок, а вже через два роки став прибутковим.

Yelp функціонує як онлайн-довідник бізнесу та платформа краудсорсингових відгуків, а також є публічною компанією з головним офісом у Сан-Франциско, Каліфорнія. До сфери її діяльності входить розробка та супровід веб-сайту Yelp.com і мобільного додатку Yelp, що містять відгуки про різні підприємства – від ресторанів до сервісних компаній. Також сервіс пропонує функціонал онлайн-бронювання столиків під брендом Yelp Reservations.

Станом на другий квартал 2019 року, кількість унікальних користувачів платформи становила 61,8 мільйона з десктопів та 76,7 мільйона з мобільних пристроїв. За офіційними даними інвесторської сторінки компанії, на 30 червня 2019 року на Yelp було опубліковано понад 192 мільйони відгуків.

Проте діяльність компанії неодноразово піддавалась критиці. Yelp звинувачували в недобросовісних бізнес-практиках, зокрема у просуванні негативних відгуків про підприємства, які не користуються її платними рекламними послугами, та у розміщенні реклами конкурентів поряд з профілями таких компаній. Крім того, надходили скарги на агресивну або маніпулятивну поведінку з боку деяких торгових представників компанії.

Ще однією проблемою, з якою стикається Yelp, є надійність системи відгуків. Зустрічалися випадки фальсифікації відгуків, як позитивних – задля

штучного підвищення рейтингу власного бізнесу, так і негативних – з метою дискредитації конкурентів. Така практика, відома як "астротурфінг", завдає шкоди довірі до платформи, з чим компанія намагається боротися за допомогою внутрішніх алгоритмів та модерації.

На рисунку 1.5 зображено екран пошуку ресторанів у мобільному додатку Yelp.

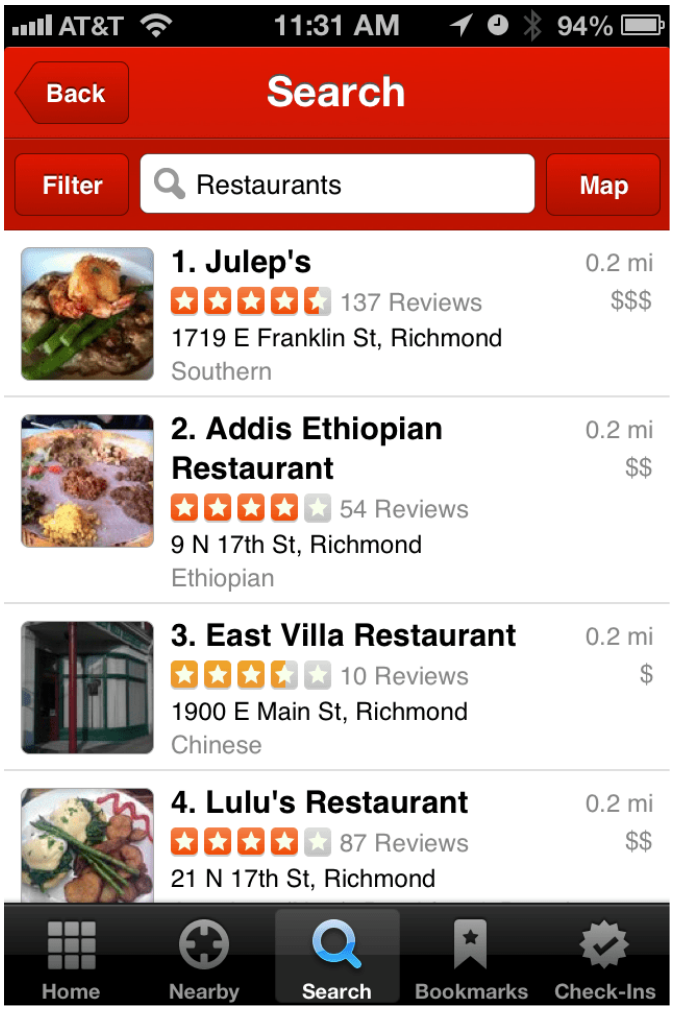


Рисунок 1.5 – Екран пошуку ресторанів

Ще одним популярним мобільним застосунком, орієнтованим на ресторанний сектор, є Urbanspoon. Цей сервіс надає користувачам можливість порівнювати ресторани за рейтингом, відстанню, типом кухні та популярністю, що особливо корисно під час перебування в незнайомому місті.

Користувачі можуть ознайомлюватися з меню, фотографіями страв, а також читати відгуки критиків та гастрономічних ентузіастів. Додатковою зручною функцією є можливість бронювання столиків безпосередньо через додаток, що робить Urbanspoon універсальним інструментом – свого роду «швейцарським ножем» у сфері вибору місць для харчування.

Сервіс Urbanspoon був заснований у 2006 році колишніми працівниками компанії Jobster. Спочатку платформа орієнтувалася на Північну Америку, а згодом поширилася на деякі англомовні регіони Австралії та Європи. У січні 2015 року компанію придбала індійська технологічна компанія Zomato, яка спеціалізується на пошуку та відкритті закладів громадського харчування.

На рисунку 1.6 представлено інтерфейс мобільного додатку Urbanspoon.

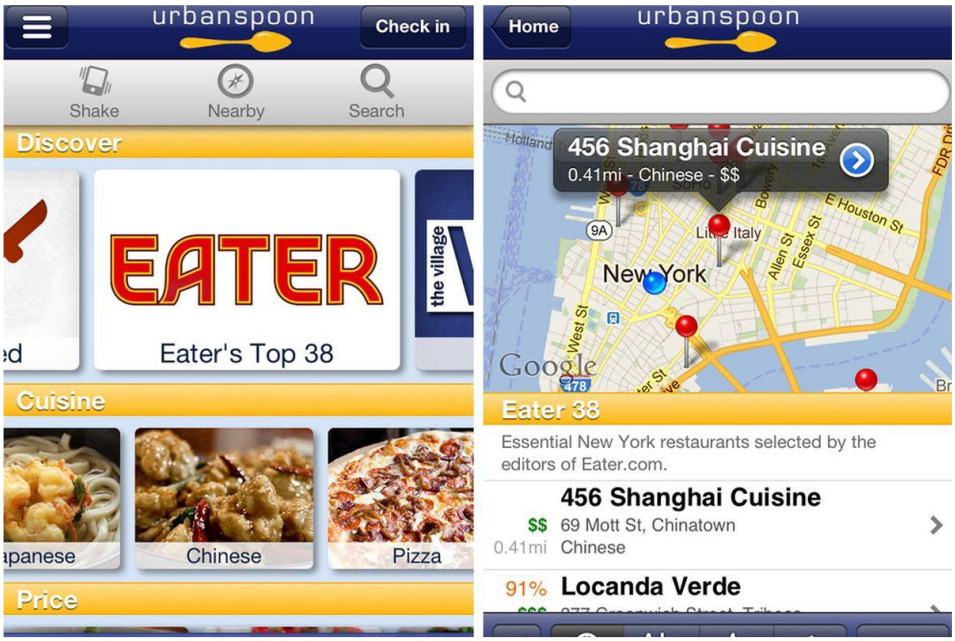


Рисунок 1.6 – Інтерфейс користувача у застосунку Urbanspoon

Urbanspoon розпочав свою діяльність із веб-сайту та операційного офісу в Сіетлі, де команда займалася збором даних про ресторани шляхом безпосереднього контакту з закладами – тобто за принципом "від дверей до дверей". Ця модель ефективно працювала й була схожа на підхід, який

Zomato застосовував у себе на батьківщині – в Індії.

Однак після придбання Urbanspoon компанією Zomato у 2015 році, платформа поступово припинила своє існування як окремий бренд. 1 червня 2015 року веб-сайт Urbanspoon було офіційно закрито, а весь користувацький трафік перенаправлено на сайт Zomato. При цьому логотип Urbanspoon було частково інтегровано в нову візуальну айдентику Zomato.

Незабаром після цього розробка додатків Urbanspoon була зупинена, що призвело до відходу інженерної команди з Сіетла до серпня 2015 року. У жовтні того ж року було остаточно закрито декілька офісів Zomato у США, включаючи колишню штаб-квартиру Urbanspoon у Сіетлі. За повідомленнями бізнес-видань станом на жовтень 2015 року, або усі операції Zomato-Urbanspoon у США було припинено, або збережено лише кілька офісів, зокрема в Далласі, де на той момент розташовувались адміністративні функції компанії в Північній Америці.

Zomato – це індійський онлайн-агрегатор закладів харчування, а також сервіс доставки їжі, який був заснований у 2008 році Deepinder Goyal та Pankaj Chaddah. Платформа надає доступ до інформації про ресторани, меню, рейтингів та відгуків користувачів. Крім того, сервіс забезпечує можливість замовлення доставки їжі з партнерських ресторанів у визначених містах.

Станом на 2016 рік, Zomato здійснював свою діяльність у 24 країнах світу, охоплюючи ключові міжнародні ринки.

На рисунку 1.7 представлено рекламний постер сервісу Zomato.

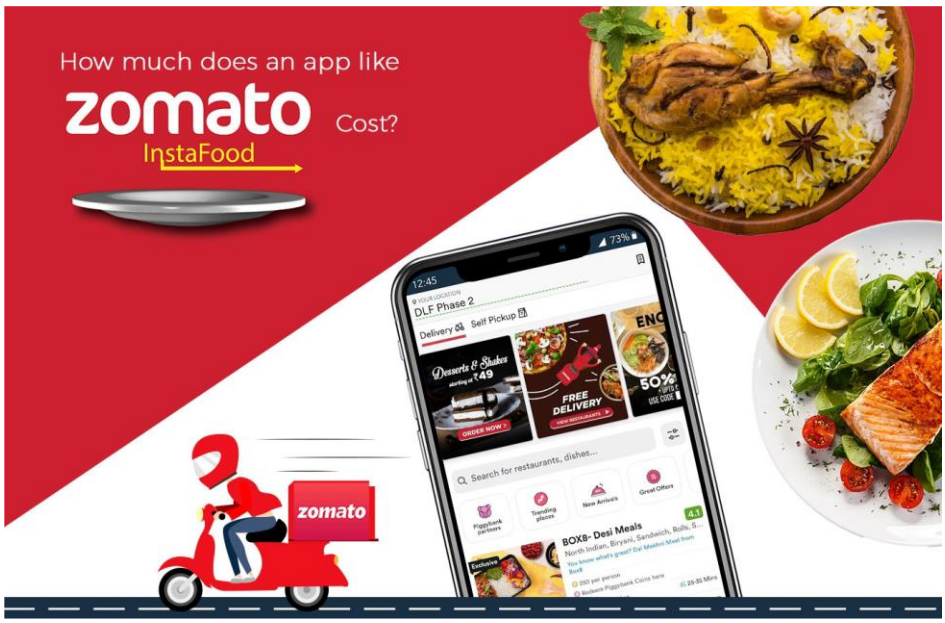


Рисунок 1.7 – Постер сервісу Zomato

Завдяки впровадженню технології хмарної кухні, компанія Zomato прагнула надати ресторанам можливість розширення своїх послуг без необхідності нести постійні операційні витрати. У вересні 2017 року компанія оголосила, що досягла прибутковості на всіх 24-х ринках, де вела діяльність на той момент. Одночасно було представлено концепцію «ресторанів з нульовою комісією» для партнерських закладів, яка передбачала відсутність плати за замовлення, зроблені через платформу.

Однак, уже наприкінці 2017 року Zomato зіткнулась із критикою користувачів: сервіс припинив приймати оновлення від активних користувачів, не здійснюючи модерації інформації про ресторани. У результаті, багато сторінок із закладами залишалися неактуальними, що впливало на точність даних. Окрім того, з'явилися скарги на збої в роботі функцій онлайн-оплати, які щойно були впроваджені.

Ще одним авторитетним сервісом у сфері оцінювання закладів громадського харчування є Zagat. Цей проєкт було започатковано у 1979 році подружжям Тімом та Ніною Загат як засіб збору й систематизації відгуків про ресторани, отриманих під час особистих візитів. Спочатку Zagat функціонував у вигляді друкованих гідів, але згодом еволюціонував у

цифрову платформу з користувацьким рейтингом.

На рисунку 1.8 представлено інтерфейс онлайн-сервісу Zagat.

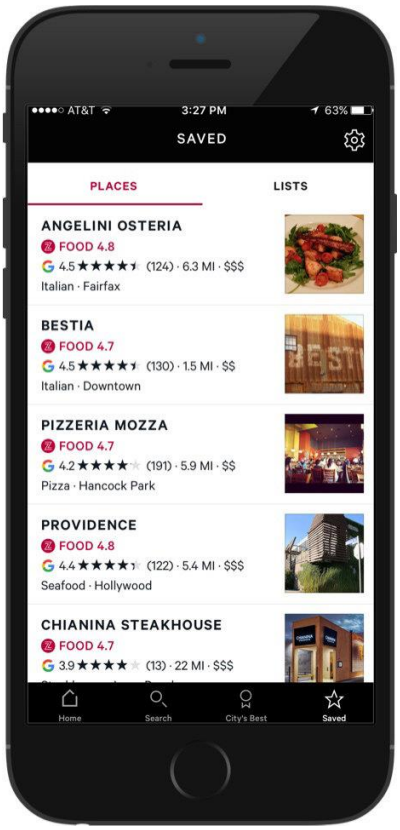


Рисунок 1.8 – Навігаційний екран програми Zagat

Перші опитування Zagat проводились у вузькому колі – автори проєкту зверталися до власних знайомих. Проте з часом охоплення сервісу суттєво зросло. Уже в 2005 році рейтинги Zagat охоплювали 70 міст, а база даних формувалася на основі відгуків близько 250 000 користувачів. У гідах оцінювались не лише ресторани, а й готелі, торгові центри, заклади нічного життя, театри, кінотеатри, музеї, зоопарки, музичні події, гольф-клуби та навіть авіакомпанії. Довідники традиційно публікувалися у друкованій формі, а доступ до повної онлайн-версії на сайті Zagat надавався лише за платною підпискою.

У вересні 2011 року компанію Zagat було придбано Google за суму, що перевищувала 150 мільйонів доларів США. Після придбання сервіс був інтегрований до підрозділу "Geo and Commerce" корпорації Google, і його

функціонал поступово поєднувався з іншими геолокаційними сервісами компанії.

29 липня 2013 року Google оновив сайт Zagat, запропонувавши новий інтерфейс та змінений формат подання контенту. Водночас кількість міст, що охоплювались у момент запуску, скоротили з 30 до 9 основних локацій. Упродовж наступних днів компанія поступово розширювала базу даних, додаючи інформацію з ще 21 міста, з метою подальшого охоплення нових регіонів на оновленій платформі.

Однак у грудні 2012 року Google оголосив про звільнення більшості штатних працівників Zagat, які на момент придбання були найняті на умовах підряду. Це спричинило хвилю песимістичних прогнозів у бізнес-пресі, де передбачалося поступове згортання друкованих видань Zagat. Подальші звіти підтвердили скорочення поліграфічного напряму діяльності компанії.

Попри ці зміни, інтеграція Zagat у екосистему Google забезпечила бренду сильні позиції у сфері локальних рекомендацій ресторанів, а також багатий контент для геолокаційного пошуку.

## 1.3 Постановка задачі

Для реалізації поставленого завдання необхідно виконати формалізацію задачі, що є ключовим етапом розробки, який включає побудову структури довідників, визначення взаємозв’язків між документами та довідниками, а також опис алгоритмів обробки інформаційних потоків.

У сфері громадського харчування важливо забезпечити своєчасний та точний облік продуктів і замовлень. Саме тому автоматизація процесів обробки інформації в ресторані є одним із пріоритетних напрямів розвитку підприємства. Головна мета цієї задачі полягає у забезпеченні оперативного обліку замовлень клієнтів, що, у свою чергу, дозволяє формувати необхідні звітні документи та іншу внутрішню документацію.

Впровадження автоматизованої системи дає змогу значно зменшити

трудові витрати та мінімізувати ймовірність помилок, допущених персоналом під час обробки даних. Це сприяє економії часу, підвищенню ефективності прийняття рішень, а також достовірності та цілісності інформації завдяки централізованому зберіганню. Вся інформація, необхідна для реалізації функцій автоматизації, зберігається в хмарній базі даних Firebase Firestore.

Автоматизована система також дозволяє формувати звіти за заданими критеріями, здійснювати підрахунок підсумкових показників та забезпечувати збереження історичних даних для подальшого аналізу.

Успішна діяльність закладу громадського харчування – це результат чітко налагоджених бізнес-процесів, що забезпечують високий рівень обслуговування та злагоджену роботу персоналу.

Проблематиці розвитку ресторанної галузі присвячено низку наукових досліджень вітчизняних авторів, таких як Архіпова В.В., Мостова Л.М., Новікова О.В., Мальська М.П., Пандяк І.Г. та ін. У своїх працях вони наголошують, що в умовах високої конкуренції підприємства повинні впроваджувати сучасні системи управління, основу яких становлять автоматизовані інформаційні технології.

Сьогодні на ринку комп’ютерного забезпечення представлені як універсальні аналітичні платформи, так і вузькоспеціалізовані програмні продукти, орієнтовані на окремі галузі. Більшість користувачів обирає саме універсальні програмні рішення через їх здатність до гнучкої адаптації під потреби конкретного закладу ресторанного бізнесу.

Управлінська діяльність сучасних підприємств відбувається в умовах динамічного середовища, тому потребує аналізу, прогнозування та оцінювання, як внутрішніх процесів, так і взаємодії із зовнішнім середовищем – як мікро-, так і макрорівня. Це, у свою чергу, висуває підвищені вимоги до інформаційного забезпечення всіх рівнів управління.

Інтеграція цифрових технологій значно прискорює виконання завдань, оптимізує комунікацію та створює зручні умови для гостей закладу,

підвищуючи загальний рівень сервісу.

Враховуючи це, слід окремо проаналізувати функціональні обов’язки офіціанта в закладах громадського харчування (кафе, ресторанах, барах тощо), оскільки запропонований програмний продукт спрямований на покращення ефективності їхньої роботи. Система розробляється з урахуванням використання мобільного додатку на платформі Android, що дозволяє персоналу зручно керувати замовленнями, не відволікаючись від обслуговування клієнтів.

# 2 ПРОЕКТУВАННЯ МОБІЛЬНОГО ЗАСТОСУНКУ

## 2.1 Моделювання та аналіз програмного забезпечення

Основний бізнес-процес передбачає взаємодію між двома мобільними застосунками – для менеджера закладу та для клієнта – з використанням серверної частини, реалізованої на платформі Firebase. Деталізована схема бізнес-процесу подана на рисунку 2.1.

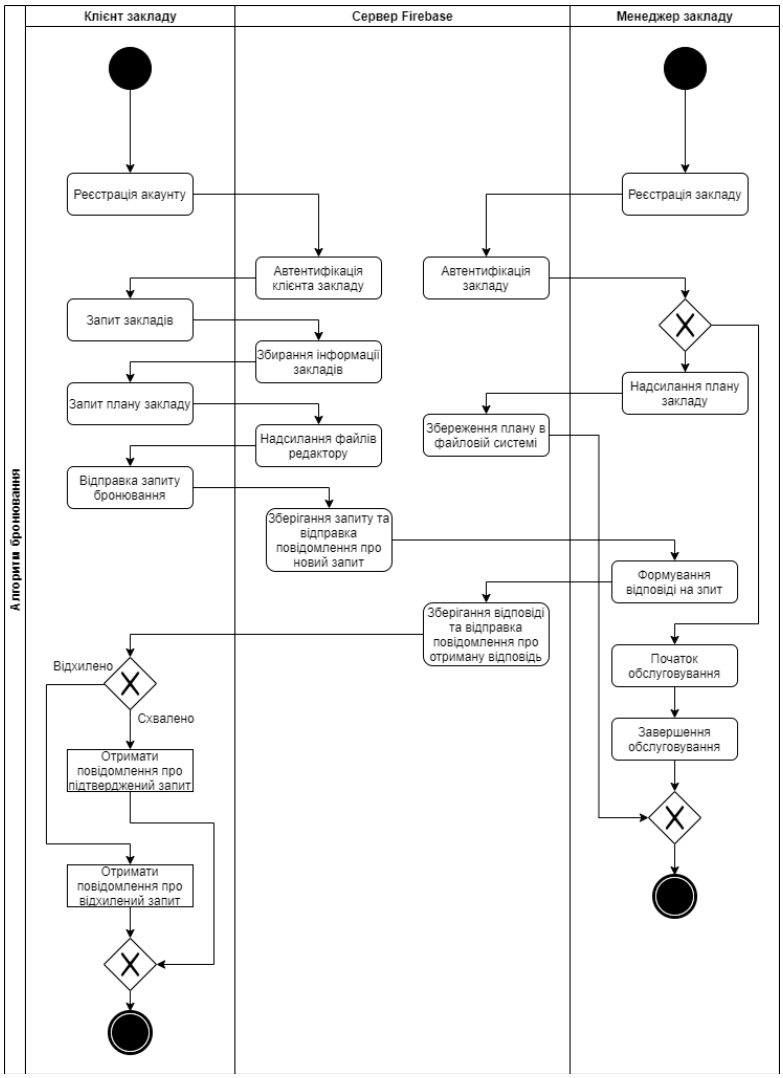


Рисунок 2.1 – Структурна схема бізнес-процесу взаємодії клієнтського та менеджерського застосунків

На початковому етапі обидва мобільні застосунки – як для клієнта, так і для менеджера – виконують авторизацію, під час якої користувачі вводять необхідні облікові дані. Ця інформація зберігається в базі даних на сервері, що реалізований на платформі Firebase.

Після успішного входу клієнтський застосунок надсилає запит на отримання переліку всіх закладів, які зберігаються в базі даних. Після отримання цієї інформації користувач має змогу обрати конкретний заклад і ініціювати запит на отримання файлу з планом розміщення столиків у залі.

Цей файл обробляється вбудованим у додаток спеціалізованим редактором, який візуалізує схему ресторану та дозволяє клієнту інтерактивно обрати вільний столик. Після цього користувач надсилає запит на бронювання обраного місця.

Зі свого боку менеджер закладу отримує повідомлення про нову заявку. Після її розгляду він формує відповідь, яка відправляється на сервер. У свою чергу, клієнт отримує повідомлення про результат бронювання – підтвердження або відмову.

Додатково, менеджер може ініціювати процес обслуговування клієнта, що включає реєстрацію замовлення, контроль виконання та облік у системі.

## 2.2 Архітектура програмного забезпечення

Клієнтський мобільний застосунок здійснює взаємодію з менеджерським застосунком через серверну частину на рисунку 2.2, яка реалізована на базі сервісів Firebase. Сам сервер функціонує як посередник у комунікації між двома сторонами, забезпечуючи обробку даних, їх зберігання та контроль доступу.

До основних компонентів Firebase, що використовуються в системі, належать:

- Firebase Authentication (FirebaseAuth) – для реалізації процесу автентифікації користувачів обох застосунків (менеджера та клієнта);

- Firebase Firestore – як основне сховище NoSQL-даних, яке дозволяє зберігати та структурувати інформацію у форматі колекцій і документів;

- Firebase Storage – для зберігання великоформатних файлів, зокрема планів закладу, фотографій та іншого мультимедійного контенту;

- Firebase Cloud Functions – для реалізації додаткової серверної логіки, яка сприяє підвищенню безпеки та гнучкості обробки подій між клієнтськими та менеджерськими застосунками.

Уся динамічна інформація зберігається у базі даних Firebase Firestore, яка побудована за принципом документо-орієнтованої NoSQL-моделі. Структурна схема бази даних наведена на рисунку 2.3.

Однією з ключових колекцій є RestaurantAdmins, яка відповідає за зберігання інформації про заклади. Вона містить:

- унікальний ідентифікатор закладу,
- контактні дані (номер телефону, адреса),
- геолокаційні координати,
- назву ресторану.

Колекція RestaurantAdmins пов’язана з іншими елементами бази даних за принципом "один до багатьох", зокрема:

- з колекцією ClientSubscriptions – для зберігання даних про підписників закладу;
- з RestaurantComments – що містить відгуки клієнтів;
- з тематичними колекціями тегів для пошуку:
- SearchCuisineTags,
- SearchFoodTags,
- SearchSpecialTags,
- SearchTypeTags,
- SearchCostTags;
- з RestaurantServicingTables – яка містить інформацію про наявні столики та їхню доступність для резервування.

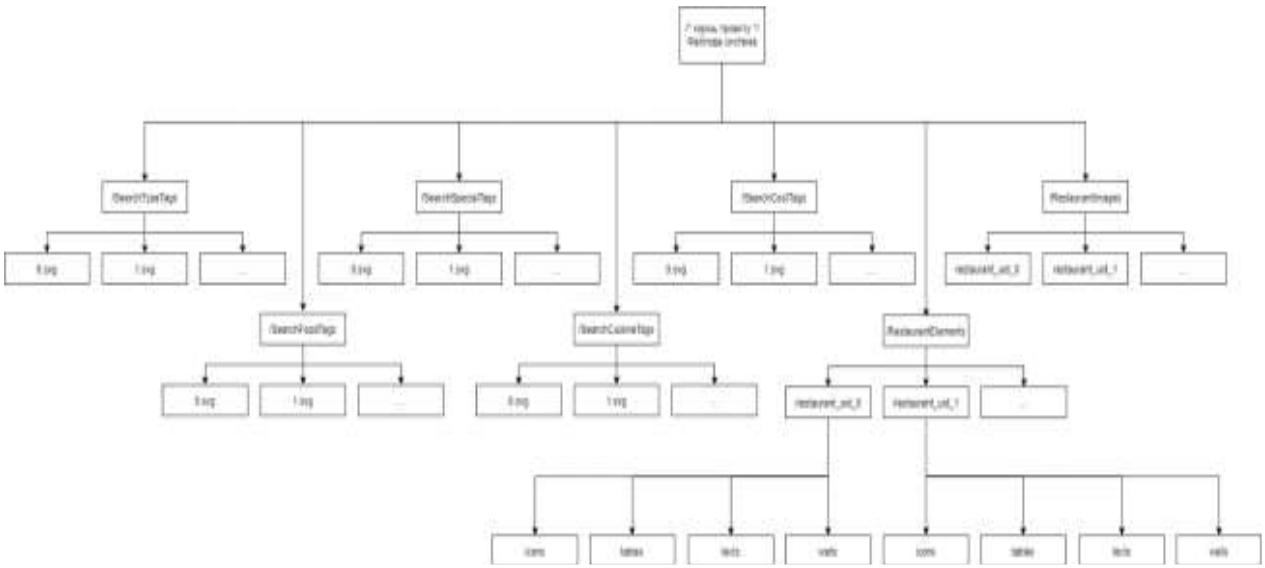

Рисунок 2.2 – Організація файлової системи клієнтського додатку

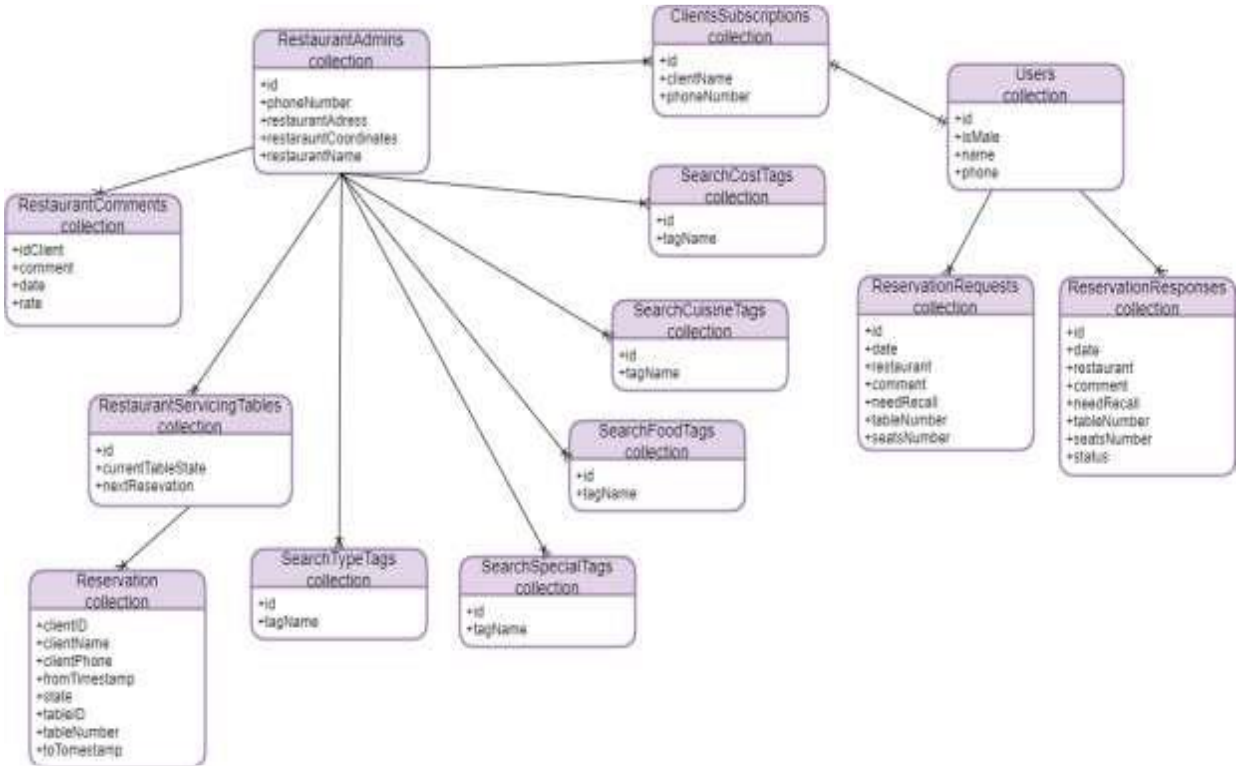


Рисунок 2.3 – Схема БД

У структурі бази даних Firebase Firestore, яка використовується для зберігання інформації в рамках розроблюваної системи, передбачено низку взаємопов’язаних колекцій.

ClientSubscriptions – колекція, призначена для зберігання даних про підписників конкретного закладу. Вона містить унікальний ідентифікатор

підписника, ім’я та номер телефону. Ця колекція має зв’язок один до одного з колекцією Users, що забезпечує цілісність інформації про клієнта.

RestaurantComments – колекція, в якій зберігаються відгуки користувачів про заклади громадського харчування. Кожен запис містить унікальний ідентифікатор автора коментаря, дату публікації, числову оцінку закладу та текстовий коментар.

Users – колекція, що містить персональні дані зареєстрованих користувачів. У ній зберігається унікальний ідентифікатор користувача, ім’я, номер телефону та стать. Колекція Users пов’язана з колекціями ReservationRequests та ReservationResponses за принципом один до багатьох, що дає змогу вести історію бронювань та відповідей на них.

ReservationRequests та ReservationResponses – колекції, призначені для зберігання запитів клієнтів на бронювання та відповідей менеджерів на них. Дані, які зберігаються у запитах: дата бронювання, назва закладу, коментар до замовлення, необхідність зворотного дзвінка, номер столика та кількість посадкових місць. Колекція ReservationResponses додатково містить поле статусу запиту: "прийнято", "не прийнято", "в очікуванні".

Колекції SearchCuisineTags, SearchFoodTags, SearchSpecialTags, SearchTypeTags, SearchCostTags призначені для зберігання категоризованих тегів, які використовуються при фільтрації закладів за відповідними ознаками (тип кухні, страви, спеціальні особливості, ціновий рівень тощо).

RestaurantServicingTables – колекція, що описує столики у залі ресторану. Вона містить інформацію про поточний стан столика (вільний, зайнятий, заброньований) та дані про наступну резервацію. Колекція має зв’язок один до багатьох із колекцією Reservation, де зберігається детальна інформація про бронювання.

Reservation – колекція, призначена для фіксації підтверджених резервацій. Кожен документ містить унікальний ідентифікатор резервації, ім’я клієнта, номер телефону, дату бронювання, статус резервації, номер столика та його ідентифікатор.

На рисунку 2.3 представлено схему зв’язків між основними колекціями бази даних Firestore.

## 2.3 Архітектура серверу Firebase

Архітектура серверної частини Firebase базується на використанні множини мікросервісів, кожен із яких відповідає за окремий функціональний аспект системи. Основні мікросервіси, що застосовуються в межах реалізації даного проєкту, наведені нижче.

Firebase Firestore – мікросервіс, що забезпечує взаємодію з NoSQL базою даних. Дозволяє створювати колекції та документи, здійснювати їх редагування, розширення або видалення. Реалізовано систему налаштувань прав доступу до окремих документів або колекцій. Додатково підтримується фільтрація даних за заданими умовами, аналогічно до оператора SELECT у SQL, а також можливість сортування результатів запиту.

Firebase Storage – мікросервіс для роботи з файловою системою. Дозволяє створювати, змінювати та видаляти файли та папки. Як і Firestore, він підтримує налаштування прав доступу на будь-якому рівні ієрархії (до окремого файлу чи каталогу).

Firebase Auth – мікросервіс, відповідальний за автентифікацію користувачів. Підтримує створення облікових записів за логіном і паролем, номером телефону, обліковим записом Google, а також у режимі анонімного входу. Надає функції для оновлення особистих даних (логін, ім’я), а також для відновлення доступу у разі втрати пароля. Додатково дозволяє ідентифікувати користувача під час кожного запиту до серверу.

Firebase Messaging – сервіс, що використовується для надсилання повідомлень користувачам, зокрема push-повідомлень для Android-клієнтів.

Cloud Functions – мікросервіс, який дає змогу розміщувати на сервері код, написаний на мові програмування Node.js. Завантажені функції можуть бути викликані як із клієнтської частини, так і всередині самого серверного

середовища. Цей підхід дозволяє реалізувати додаткову логіку, обробку подій, перевірку прав доступу та інші дії, що підвищують безпеку і гнучкість системи.

У таблиці 2.1 наведено специфікацію функціональних можливостей, реалізованих за допомогою Cloud Functions для менеджера закладу.

Таблиця 2.1 – Специфікації функцій Cloud Functions для менеджера закладу

| Функція | Призначення | Аргументи |
|---|---|---|
| canBookTable | Виконує перевірку можливості бронювання столика на заданий проміжок часу в заданому закладі | restaurantId, startMillisDate, endMillisDate, tableId |
| addReservation | Виконує додавання бронювання на заданий проміжок часу, на заданий столик, кількість людей з опціональним коментарем та вказанням чи потрібно передзвонювати. Функцію можна викликати тільки зі сторони авторизованого менеджера закладу | client, startMillisDate, endMillisDate, tableId, tableNumber, peopleCount, comment, needsRecall |
| getCanBook Tables | Визначає чи доступне бронювання заданих столиків на заданий проміжок часу | restaurantId, tablesIdsList, startMillisDate, endMillisDate |
| subscribeFor NotificAtion | Підписує заданий ідентифікатор пристрою на розповсюдження повідомлень для менеджера закладу. Функцію можна викликати тільки зі сторони авторизованого менеджера закладу | deviceId |

У таблиці 2.2 представлені специфікації набору функцій Cloud Functions для клієнтів закладу.

Таблиця 2.2 – Специфікації функцій Cloud Functions для клієнтів закладу

| Функція | Призначення | Аргументи |
|---|---|---|
| canBookTable | Підписує заданий ідентифікатор пристрою на розповсюдження повідомлень для менеджера закладу. Функцію можна викликати тільки зі сторони авторизованого клієнта закладу | deviceId |
| unsubscribeFor Notification | Відписує заданий ідентифікатор пристрою від розповсюдження повідомлень для менеджера закладу. Функцію можна викликати тільки зі сторони авторизованого клієнта закладу | deviceId |
| createReservation Request | Створює запит на бронювання заданого столика на заданий проміжок часу для заданої кількості людей з коментарем та вказанням чи потрібний зворотній дзвінок для визначення деталей бронювання. Функцію можна викликати тільки зі сторони авторизованого клієнта закладу | restaurantId, tableId, tableNumber, startMillisDate, endMillisDate, peopleCount, comment, needsRecall |
| subscribeForRestaur antNotification | Підписатися на повідомлення від конкретного закладу Функцію можна викликати тільки зі сторони авторизованого клієнта закладу | restaurantId, clientName, phoneNumber |
| unsubscribeFromRe staurantNotification | Відписатися від повідомлень від конкретного закладу | restaurantId |

## 2.4 Аналіз безпеки даних

Для забезпечення безпечної взаємодії між клієнтським та менеджерським додатками були реалізовані спеціальні правила доступу до даних, що зберігаються у Firebase Firestore та Firebase Storage. Ці правила визначають, які саме дії може виконувати конкретний користувач над певними об'єктами даних, виходячи з його прав та ідентифікатора.

Зокрема, основна інформація про ресторан (назва, адреса, локація тощо) є відкритою для читання усім користувачам системи. Водночас право на редагування таких даних надається виключно авторизованому користувачу, унікальний ідентифікатор якого збігається з ідентифікатором відповідного документа.

Аналогічні правила реалізовані й для файлового сховища Firebase Storage. Наприклад, файли з планом розміщення столиків у ресторані доступні для перегляду всім користувачам, проте можливість їх редагування має лише менеджер закладу, який пройшов автентифікацію та ідентифікований відповідним чином.

Крім того, до виклику серверних функцій (Cloud Functions) також застосовуються обмеження за доступом. Окремі функції можуть бути викликані тільки з боку авторизованого менеджера, тобто клієнти закладу не мають можливості ініціювати виконання таких дій. У свою чергу, функціонал, призначений для клієнтської частини, доступний виключно для авторизованих користувачів зі статусом клієнта. Це забезпечує ізольованість логіки обох додатків і запобігає несанкціонованому доступу до адміністративних функцій.

# 3 РЕАЛІЗАЦІЯ ДОДАТКУ ДЛЯ БРОНЮВАННЯ СТОЛИКІВ

## 3.1 Технічне завдання та використані засоби

Технічне завдання полягало у створенні двох типів програмного забезпечення: мобільного застосунку для користувачів, які бажають забронювати столик у ресторані, та веб-застосунку для адміністраторів закладів громадського харчування, які можуть створювати сторінки ресторанів і керувати процесом бронювання. Обидва застосунки реалізовані з використанням інструментів SDK фреймворку Flutter.

У процесі реалізації було створено мобільний додаток для клієнтів, який надає функцію бронювання столиків, а також веб-додаток для адміністрації ресторанів, у якому власники мають змогу додавати заклади, редагувати їхні дані, переглядати активні бронювання та обробляти запити від користувачів.

Крім стандартних бібліотек Flutter SDK, у проєкті були використані такі додаткові бібліотеки:

GetX – легковажна бібліотека, що спрощує роботу з керуванням станами, навігацією, ін’єкцією залежностей, а також підтримує багатомовність. У цьому проєкті застосовується реактивний підхід до управління станами.

flutter_neumorphic – набір віджетів для створення інтерфейсів у стилі неоморфізму, що був особливо популярним у 2020–2021 роках.

flutter_svg – бібліотека, яка дозволяє працювати з векторною графікою у форматі SVG.

equatable – забезпечує правильне перевизначення операторів порівняння та генерацію хеш-кодів для класів даних.

json_serializable – бібліотека для автоматичної генерації функцій серіалізації/десеріалізації JSON для класів даних.

Для реалізації серверної логіки, автентифікації, зберігання та обміну даними було використано хмарні сервіси платформи Firebase.

Firebase Authentication – сервіс, який забезпечує автентифікацію користувачів за допомогою електронної пошти, номера телефону, акаунтів соціальних мереж, а також дозволяє реалізувати анонімну авторизацію. Підтримує інтеграцію з OAuth 2.0 та OpenID Connect. Також надає інструменти для відновлення пароля, оновлення облікових даних та зв'язку з іншими сервісами Firebase.

Firestore Database – це хмарна документо-орієнтована NoSQL база даних, яка зберігає інформацію у вигляді колекцій і документів. Підтримуються основні типи даних, у тому числі рядки, числа, булеві значення, об'єкти, масиви тощо. Дані оновлюються у реальному часі, що дозволяє оперативно синхронізувати зміни між усіма клієнтами. Також реалізована офлайн-підтримка для мобільних застосунків і гнучка система доступу, інтегрована з Firebase Authentication.

Firebase Storage – сервіс для зберігання файлів, таких як плани залів, зображення, документація тощо. Дозволяє створювати директорії, завантажувати файли та керувати ними. Має інтеграцію з Google Cloud Storage APIs, що дозволяє виконувати складні операції над файлами, наприклад, обробку зображень або перекодування відео. Реалізовано систему доступу на основі ролей, пов'язану з Firebase Authentication.

Firebase Analytics – сервіс для збору аналітичних даних про дії користувачів у застосунку. Особливо корисний у рамках маркетингового аналізу та покращення UX/UI.

Firebase Crashlytics – інструмент для відстеження та аналізу збоїв у роботі застосунків на платформах Android та iOS. Дозволяє оперативно реагувати на помилки, що виникають у користувачів.

У таблиці 2.1 подано перелік реалізованих функцій на базі Cloud Functions, призначених для адміністратора ресторану.

## 3.2 Історія Flutter

У 2015 році компанія Google вперше анонсувала Flutter SDK – фреймворк для розробки мобільних застосунків, який базується на мові програмування Dart. Початково доступ до Flutter був обмеженим, а розробка здійснювалася виключно для операційної системи Android.

У 2017 році на конференції Google I/O було представлено першу публічну альфа-версію Flutter, яка вже підтримувала кросплатформену розробку. Компанія офіційно рекомендувала використання Flutter для створення мобільних застосунків.

У 2018 році було випущено Flutter 1.0 – першу стабільну версію SDK. Це дало сигнал спільноті, що фреймворк готовий до використання у продакшн-середовищі. Серед нових можливостей з'явилися підтримка Google Maps, WebView, а також Flare – інструмент для створення складних векторних анімацій. Крім того, було анонсовано Flutter for Web, що значно розширило сферу застосування фреймворку.

У 2019 році Flutter отримав підтримку для розробки не тільки мобільних, але й веб та десктопних застосунків (Windows, Linux, macOS) – поки що у режимі публічного альфа-доступу.

Наступне важливе оновлення відбулося у березні 2021 року на презентації Flutter Engage, коли вийшов Flutter 2. У цій версії веб-розробка отримала статус стабільної, а також з'явилася бета-підтримка десктопних застосунків. Додано нові віджети, інтегровано Google Mobile Ads. Одночасно з цим вийшла мова Dart 2.12 з підтримкою null safety. Компанія Canonical оголосила, що Flutter стає основним інструментом для розробки застосунків на операційній системі Ubuntu.

Flutter базується на декларативному стилі програмування, що є альтернативою імперативному. Імперативна парадигма полягає в описі послідовності дій (приклади: C++, Java), тоді як декларативна – у формулюванні бажаного результату без уточнення алгоритму (приклади:

HTML, SQL).

Більшість UI-фреймворків у минулому реалізовувалися за імперативною парадигмою. Натомість Flutter, як і React, дотримується декларативного підходу: інтерфейс створюється автоматично відповідно до стану, яким керує розробник.

В основі Flutter лежить концепція "все є віджетом". Увесь користувацький інтерфейс описується у вигляді деревоподібної структури віджетів, кожен з яких відповідає за певний елемент інтерфейсу. Під час зміни стану будь-якого віджета викликається метод build(), який оновлює візуальне представлення.

Існує два основних типи віджетів: Stateless (незмінні) та Stateful (ті, що змінюють свій стан). StatelessWidget не зберігає стану і складається лише з одного класу. StatefulWidget складається з двох класів: сам віджет (StatefulWidget) і клас стану (State), у якому реалізовано логіку зміни стану.

Функція build() є ключовим методом у обох типах віджетів. Вона повертає дерево віджетів, які описують структуру сторінки. Наприклад, для створення типової сторінки Material Design використовується віджет Scaffold. У його параметрі body можна розмістити віджет Center, який містить Text – для виводу тексту. У параметрі floatingActionButton передається кнопка типу FloatingActionButton, яка має параметр onPressed. У цьому параметрі викликається функція, яка використовує метод setState для оновлення стану – саме це запускає процес перерисовки інтерфейсу.

У прикладі сторінки з лічильником, створюється клас IncrementorPage, що наслідується від StatefulWidget. Метод createState() створює екземпляр класу _IncrementorPageState, де описано логіку оновлення значення лічильника (_increment) за допомогою setState.

Для складніших сценаріїв керування станами можуть використовуватись спеціальні бібліотеки, такі як Provider, BLoC, Redux, MobX, GetX та інші.

## 3.3 Середовище розробки

Для розробки застосунків на базі Flutter зазвичай використовуються два основні інтегровані середовища розробки: Android Studio та Visual Studio Code. Обидва середовища вимагають попереднього встановлення відповідних розширень – Flutter та Dart – для забезпечення повноцінної роботи з проєктами на Flutter.

Android Studio вважається більш функціональним середовищем, яке забезпечує розширений інструментарій для розробки та тестування застосунків. Однак його використання потребує значних апаратних ресурсів, що може впливати на продуктивність комп'ютера під час розробки.

У зв'язку з цим для виконання практичного завдання було обрано більш легке за ресурсоспоживанням середовище – Visual Studio Code. Воно є кросплатформеним редактором коду, розробленим компанією Microsoft, і забезпечує достатній функціонал для реалізації Flutter-застосунків. Visual Studio Code підтримує розширення для різних мов програмування та технологій, має вбудовану підтримку системи контролю версій Git, а також надає інструменти для зневадження коду.

Для забезпечення коректної роботи з Flutter у Visual Studio Code необхідно встановити відповідні розширення з офіційної бібліотеки, що доступна безпосередньо через інтерфейс середовища. На рисунку 3.1 представлено приклад інсталяції необхідного плагіна.

Ці розширення забезпечують можливість створення нових проєктів, їх збирання та запуск на різних емуляторах або фізичних пристроях. Розробник може легко обирати доступні пристрої безпосередньо з меню середовища розробки представленої на рисунку 3.2, що значно спрощує процес тестування застосунків у різних середовищах.

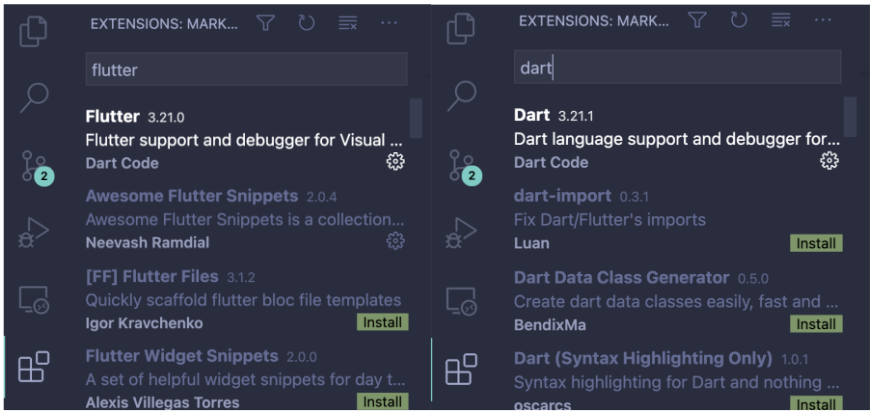


Рисунок 3.1 – Налаштування середовища розробки

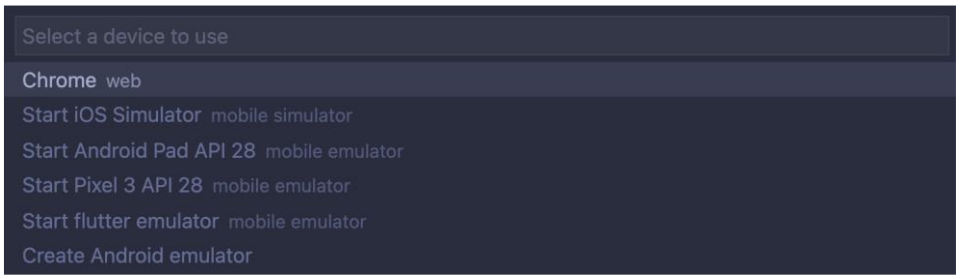


Рисунок 3.2 – Вибір платформи розробки

Dart DevTools – це вбудована утиліта, яка надає розширені можливості для налагодження застосунків, написаних на мові Dart, зокрема для проєктів на Flutter. Вона інтегрується із середовищем розробки та дозволяє розробнику:

- відстежувати структуру користувацького інтерфейсу та поточні стани віджетів у Flutter у реальному часі;
- виявляти та діагностувати проблеми, пов’язані з продуктивністю застосунка;
- переглядати споживання оперативної пам’яті, навантаження на процесор і використання мережевих ресурсів;
- виконувати аналіз обсягу коду та розміру кінцевого застосунка;
- досліджувати дерево віджетів для глибшого розуміння структури інтерфейсу та швидкого пошуку помилок.

## 3.4 Архітектура проекту

Під час розробки застосунку була реалізована класична трирівнева архітектура, яка включає такі рівні:

1. Рівень доступу до даних – відповідає за взаємодію з базою даних. Тут був застосований паттерн "Репозиторій", що дозволяє інкапсулювати складні запити до Firestore та забезпечує гнучкість: у майбутньому джерело даних можна легко змінити, просто перевизначивши методи репозиторію в іншому класі.

2. Рівень бізнес-логіки – відповідає за обробку даних, прийняття рішень, виконання перевірок та логіку між рівнями доступу до даних і представлення.

3. Рівень представлення – реалізує інтерфейс користувача. Він був умовно поділений на три частини:

- веб-інтерфейс (для браузера) – використовується адміністрацією ресторанів для управління бронюваннями та створення нових закладів.

- мобільний інтерфейс – дає можливість користувачам переглядати ресторани, їхній план та бронювати столи.

- спільні компоненти – авторизація, реєстрація та універсальні віджети, які використовуються в обох типах інтерфейсів (мобільному та веб).

Такий підхід дозволяє підтримувати чисту архітектуру, забезпечує повторне використання коду та полегшує масштабування проєкту.

## 3.5 Опис розробки та застосунку

Для розробки практичної частини використовувалася Flutter версії 2.0.4, яка вийшла якраз під час реалізації проєкту. На початковому етапі був створений проєкт за допомогою Flutter CLI під назвою GoRes командою flutter create gores. До проєкту було додано систему контролю версій Git, а також підключено можливість запуску застосунку у веб-браузері, оскільки

Flutter for Web на той момент ще не входив до стандартного SDK.

Було створено проєкт у Firebase та підключено всі необхідні бібліотеки для взаємодії з його сервісами. Після цього були виконані додаткові налаштування для Android-платформи. Після успішного підключення всіх необхідних залежностей та бібліотек було створено початкову структуру (скелет) застосунку. Також було реалізовано сервіс для керування мовами інтерфейсу. Для організації залежностей було впроваджено бібліотеку GetX.

На наступному етапі почалася розробка авторизації та реєстрації рисунок 3.3. Було створено окремі класи: AuthRepository для обробки запитів до Firebase Authentication, LoginController та SignUpController для керування логікою авторизації і реєстрації, а також LoginPage і SignUpPage – відповідні екрани. Для зберігання та управління користувачами використовувався Firebase Authentication. Усі етапи реалізації авторизації були побудовані згідно з класичною трирівневою архітектурою застосунку.

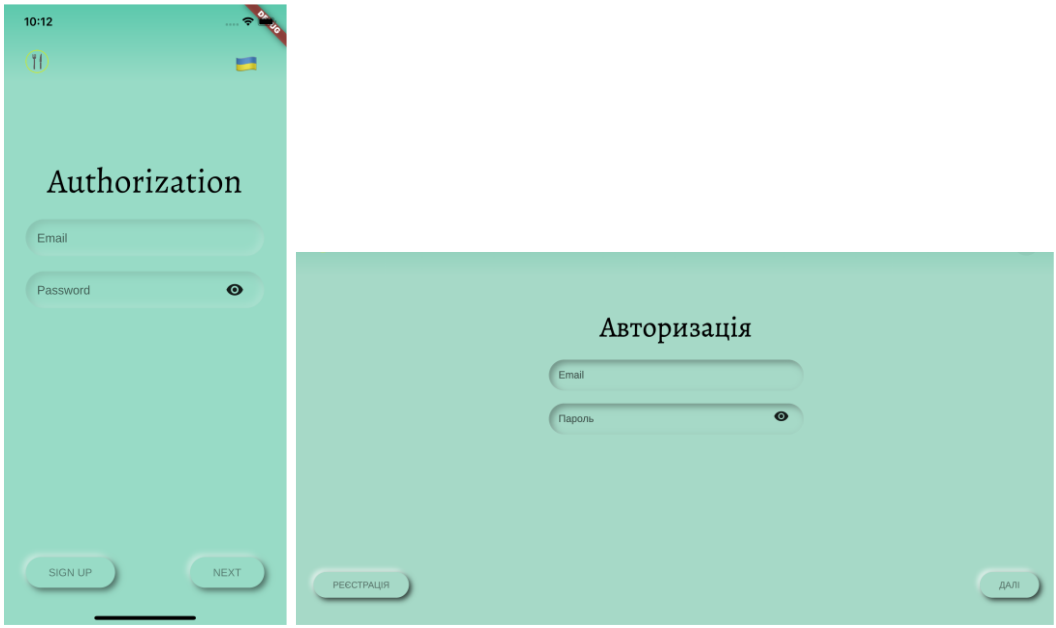


Рисунок 3.3 – Вигляд LoginPage

У верхній частині сторінки був розміщений спеціальний віджет DefaultAppBar, який містить логотип застосунку та кнопку перемикання мови. Для цього був створений окремий віджет, який відповідає за

інтерфейсну локалізацію.

Нижче, з невеликим відступом, розташовані текстові поля для введення електронної пошти (email) та пароля. Для цих цілей були використані власні віджети DefaultTextField та DefaultPassword. У полі пароля праворуч розміщена кнопка, яка дозволяє показати або приховати введений пароль.

У нижній частині екрана розміщені дві кнопки, реалізовані через віджет DefaultButton. Кнопка “Реєстрація” відкриває сторінку створення нового акаунта, а кнопка “Далі” запускає процес авторизації.

LoginController містить поля для збереження введених користувачем даних: електронної пошти та пароля. Після натискання кнопки “Далі” проводиться валідація введених значень: перевіряється, чи заповнені обидва поля, чи введено коректний email. Якщо валідація проходить успішно, дані передаються у функцію login класу AuthRepository. У разі, якщо login повертає true, користувач авторизується і його перенаправляють на головну сторінку. У веб-версії вона одна, у мобільній – інша. Якщо ж повертається false, користувач бачить повідомлення про помилку.

Клас AuthRepository реалізує взаємодію з Firebase Authentication. Він відповідає за авторизацію, реєстрацію користувачів, а також за підтримку поточного стану користувача. У функції login викликається метод signInWithEmailAndPassword з FirebaseAuth, куди передається email і пароль. Якщо авторизація проходить успішно, викликається функція initProfile, яка отримує з Firestore дані про користувача. Якщо користувач не знайдений або сталася помилка, повертається відповідне повідомлення про невдачу.

Після успішної авторизації або реєстрації користувача автоматично перенаправляють на головну сторінку рисунок 3.4, відповідно до типу платформи. На головному екрані користувач бачить список ресторанів, кнопку для пошуку та кнопку виходу з облікового запису. Кожен ресторан представлений карткою, що містить зображення, назву, опис, цінову категорію та коротку інформацію. Після вибору ресторану через пошук, користувача одразу перекидає на сторінку обраного закладу.

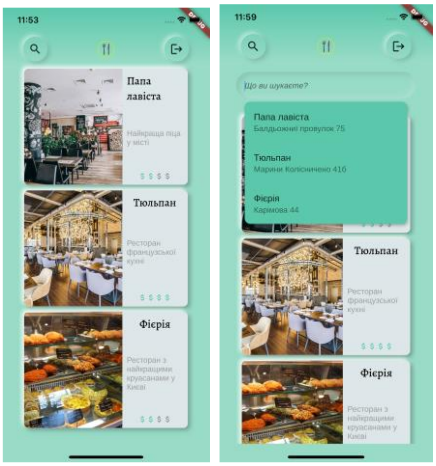


Рисунок 3.4 – Вигляд екрану замовлення

На сторінці ресторану користувач може переглядати зображення закладу, які доступні у вигляді галереї. Крім візуальної інформації, доступна функція бронювання. Для цього потрібно заповнити кілька параметрів: обрати дату, кількість людей, тривалість перебування, бажаний час та підтвердити бронювання.

Після заповнення усіх полів та надсилання запиту система перевіряє наявність вільних місць у заданий час. Якщо вільні столики є – запит успішно надсилається на сервер. У випадку, якщо на вибраний час немає доступних столиків, з’являється відповідне повідомлення про відсутність вільних місць (рисунок 3.5).

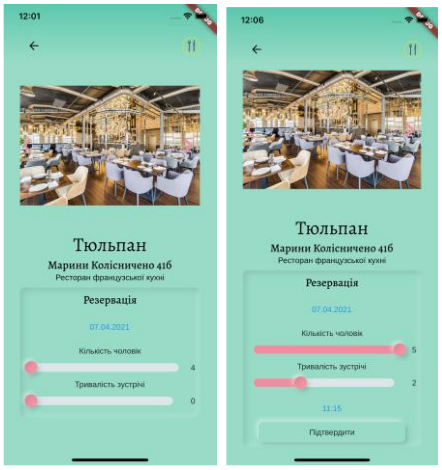


Рисунок 3.5 – Процес підтвердження замовлення

Таким чином, мета кваліфікаційної роботи була досягнута: було розроблено мобільний застосунок для клієнтів і веб-застосунок для адміністраторів закладів громадського харчування, що дозволяють ефективно здійснювати процес бронювання столиків. Реалізовані функції авторизації, перегляду інформації про ресторани, інтерактивного вибору столиків та обробки запитів забезпечують зручність як для користувачів, так і для персоналу. Застосування Flutter і Firebase дозволило створити сучасне, кросплатформене рішення з гнучкою архітектурою та надійною роботою в реальному часі.

# ВИСНОВКИ

У даній кваліфікаційній роботі було реалізовано програмне забезпечення, орієнтоване на сервісне обслуговування клієнтів закладів громадського харчування, зокрема – можливість здійснення онлайн-бронювання конкретних столиків. Рішення охоплює мобільні додатки для клієнтів і менеджерів на платформі Android, а також серверну частину, яка забезпечує надійний обмін даними між додатками та централізоване зберігання інформації.

Проведений аналіз існуючих аналогічних рішень показав, що на момент розробки не існувало повноцінних мобільних сервісів з функціоналом наочного вибору конкретного столика на інтерактивному плані ресторану. Це обґрунтувало актуальність обраного підходу.

Було спроєктовано архітектуру системи, яка відповідає принципам масштабованості та легкості у підтримці. Розроблено схему NoSQL бази даних, яка зберігає як основну інформацію про ресторани (назва, адреса, геолокація, контакти, посилання на сайт і меню), так і дані користувачів (ім’я, номер телефону), а також записи про створені бронювання.

Окрему увагу приділено серверній частині, побудованій на базі сервісів Firebase. Була реалізована логіка взаємодії між клієнтським та менеджерським додатками через сервер, описані ключові функції та захищені інтерфейси, що гарантують коректну та безпечну роботу системи.

# ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Statista.com. Number of smartphone users worldwide from 2016 to 2023 [Електронний ресурс]. – Режим доступу: https://www.statista.com/statistics/330695/number-of-smartphone-usersworldwide/ – Дата звернення: 22.06.2025.

2. GitHub. GetX [Електронний ресурс]. – Режим доступу: https://github.com/jonataslaw/getx – Дата звернення: 22.06.2025.

3. Pub.dev. flutter_neumorphic [Електронний ресурс]. – Режим доступу: https://pub.dev/packages/flutter_neumorphic – Дата звернення: 22.06.2025.

4. Pub.dev. flutter_svg [Електронний ресурс]. – Режим доступу: https://pub.dev/packages/flutter_svg – Дата звернення: 22.06.2025.

5. Firebase. Firebase Authentication [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/auth – Дата звернення: 22.06.2025.

6. Firebase. Cloud Firestore [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/firestore – Дата звернення: 22.06.2025.

7. Firebase. Cloud Storage [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/storage – Дата звернення: 22.06.2025.

8. Firebase. Google Analytics [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/analytics – Дата звернення: 22.06.2025.

9. Firebase. Firebase Crashlytics [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/crashlytics – Дата звернення: 22.06.2025.

10. Medium.com. A Brief History of Flutter [Електронний ресурс] / Pragmatic Programmers. – Режим доступу: https://medium.com/pragmatic-programmers/a-brief-history-of-flutter-939645f93255 – Дата звернення: 22.06.2025.

11. Medium.com. Flutter 1.0: An Introduction [Електронний ресурс] / The Startup. – Режим доступу: https://medium.com/swlh/flutter-1-0-an-introduction-

cb9540b2e1a – Дата звернення: 22.06.2025.

12. Codemagic.io. Flutter Versus Other Mobile Development Frameworks: A UI and Performance Experiment [Електронний ресурс] / Codemagic Blog. – Режим доступу: https://blog.codemagic.io/flutter-vs-ios-android-reactnative-xamarin/ – Дата звернення: 22.06.2025.