Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

КВАЛІФІКАЦІЙНА МАГІСТЕРСЬКА РОБОТА

на тему: «Розробка автоматизованої системи управління службою доставки»

Виконав студент 2 курсу (номер курсу), групи ЕК.м-11 (шифр групи)
Спеціальності 051 «Економіка»
(«Економічна кібернетика»)
Ревенко А. В.
(прізвище та ініціали студента)
Керівник: доцент, к.т.н., Гриценко К. Г.
(посада, науковий ступінь, прізвище та ініціали керівника)

Суми – 2022 рік

РЕФЕРАТ

кваліфікаційної магістерської роботи на тему

«Розробка автоматизованої системи управління службою доставки»

студента Ревенка Артема Володимировича

(прізвище, ім’я, по батькові студента)

Актуальність теми, обраної для дослідження, визначається тим, що за останні роки Україна зіштовхнулася з розповсюдженням короновірусної хвороби (COVID-19), що призвело до локдауну. Велика кількість людей перестала виходити з власних будинків для запобігання захворювання. Подане явище визвало зростання кількості служб доставки їжі. Спочатку попит на подібні послуги був лише у великих містах, проте останнім часом дана тенденція стає актуальною і для малих міст. Окремим фактором розповсюдження доставок їжі є швидкий ритм життя або ненормований робочий день у сучасних людей, що не дозволяє годинами простоювати у плити. Для більшості людей буває важко знайти час, щоб піти за продуктами, не кажучи вже про приготування їжі вдома. Використання служби доставки є також актуальною для замовлення у офіс, для співробітників. Тим самим підприємству не потрібно тримати свою кухню, що впливає на чистий дохід.

Мета кваліфікаційної магістерської роботи полягає у розробці прототипу мобільного додатку для автоматизації процесу управління службою доставки готових страв.

Об’єктом дослідження виступає процес доставки готових страв.

Предметом дослідження виступають сучасні засоби автоматизації процесу доставки готових страв.

Завданнями дослідження є:

– охарактеризувати процес доставки продукції;

– проаналізувати ринок автоматизованих систем служб доставки;

– сформувати вимоги до системи та обрати технології розробки;

– розробити модель бізнес-процесів;

– описати архітектуру автоматизованої системи та обрати технології вирішення поставлених завдань;

– визначити функціональну структуру завдання;

– описати структуру та особливості реалізації інформаційного та алгоритмічного забезпечень;

– надати контрольний приклад та інструкцію щодо використання;

– оцінити очікуваний ефект від впровадження системи.

Для досягнення поставленої мети та завдань дослідження були використані такі методи дослідження: дедукція, аналіз, синтез, порівняння, системний аналіз, проектування та моделювання.

Інформаційною базою кваліфікаційної магістерської роботи є матеріали, ТОВ «ТЕК СОЛЮШНС ЛТД», аналітичні огляди й інформаційні дані служб доставки та різноманітні дослідження в галузі автоматизації.

Основний науковий результат кваліфікаційної магістерської роботи полягає в такому: проаналізовано ринок автоматизованих служб доставки, побудовано комплекс прикладних програм для користувача та кур'єра, створена для досягнення мети автоматизації процесів доставки у місті Суми.

Одержані результати можуть бути використані ресторанами, кафе та іншими закладами харчування для підвищення ефективності продажів та автоматизації системи управління службою доставки, а також всіма бажаючими жителями міста, котрі мають намір задовольнити власні потреби.

Ключові слова: автоматизація, автоматизована система, інформаційна система, автоматизована інформаційна система, служба доставки, мобільний додаток, мобільний застосунок.

Зміст кваліфікаційної магістерської роботи викладено на 61 сторінці. Список використаних джерел із 40 найменувань, розміщений на 4 сторінках. Робота містить 53 рисунки, а також 2 додатки, розміщених на 4 сторінках.

Рік виконання кваліфікаційної роботи – 2022 рік.

Рік захисту роботи – 2022 рік.

Міністерство освіти і науки України
Сумський державний університет
Навчально-науковий інститут бізнесу, економіки та менеджменту
Кафедра економічної кібернетики

ЗАТВЕРДЖУЮ
Завідувач кафедри
доцентка, к.е.н.,
_____ В. В. Койбічук
“__” _____ 2022 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ МАГІСТЕРСЬКУ РОБОТУ
спеціальність 051 «Економіка» (Економічна кібернетика)
студента 2 курсу, групи ЕК.м-11

Ревенко Артем Володимирович
(прізвище, ім’я, по батькові студента)

1. Тема роботи: Розробка автоматизованої системи управління службою доставки
затверджена наказом по університету від «12» грудня 2022 року № 1212-VI
2. Термін подання студентом закінченої роботи «17» грудня 2022 року
3. Мета кваліфікаційної роботи: розробка прототипу мобільного додатку для автоматизації процесу управління службою доставки готових страв.
4. Об’єкт дослідження: процес доставки готових страв.
5. Предмет дослідження: сучасні засоби автоматизації процесу доставки готових страв
6. Кваліфікаційна робота виконується на матеріалах: ТОВ «ТЕК СОЛЮШНС ЛТД», аналітичних оглядів й інформаційних даних служб доставки та різноманітних дослідженнях в галузі автоматизації.
7. Орієнтовний план кваліфікаційної роботи, терміни подання розділів керівникові та зміст завдань для виконання поставленої мети
Розділ 1. ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ БІЗНЕС-ПРОЦЕСІВ СЛУЖБИ ДОСТАВКИ – 15 листопада 2022 року
(назва – термін подання)

У розділі 1 – Теоретична характеристика процесу доставки продукції, аналіз ринку автоматизованих систем служб доставки, формування вимог до системи.
(зміст конкретних завдань до розділу, які повинен виконати студент)

Розділ 2. ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ – 23 листопада 2022 року
(назва – термін подання)

У розділі 2 – Розробка моделі бізнес-процесів, архітектура автоматизованої інформаційної системи та технології вирішення поставлених завдань, функціональна структура завдання

(зміст конкретних завдань до розділу, які повинен виконати студент)

Розділ 3. РЕАЛІЗАЦІЯ ПРОТОТИПУ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ – 01 грудня 2022 року

(назва – термін подання)

У розділі 3 – Структура та особливості реалізації інформаційного забезпечення, структура та особливості реалізації алгоритмічного забезпечення, контрольний приклад та інструкція щодо використання, оцінювання очікуваного ефекту від впровадження автоматизованої інформаційної системи

(зміст конкретних завдань до розділу, які повинен виконати студент)

8. Консультації з роботи:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1			
2			
3			

9. Дата видачі завдання: «28» жовтня 2022 року

Керівник кваліфікаційної роботи _____ К. Г. Гриценко

(підпис) (ініціали, прізвище)

Завдання до виконання одержав _____ А. В. Ревенко

(підпис) (ініціали, прізвище)

ЗМІСТ

ВСТУП

За останні роки Україна зіштовхнулася з розповсюдженням короновірусної хвороби (COVID-19), що призвело до локдауну [40]. Велика кількість людей перестала виходити з власних будинків для запобігання захворювання. Подане явище визвало зростання кількості служб доставки їжі. Спочатку попит на подібні послуги був лише у великих містах, проте останнім часом дана тенденція стає актуальною і для малих міст. Окремим фактором розповсюдження доставок їжі є швидкий ритм життя або ненормований робочий день у сучасних людей, що не дозволяє годинами простоювати у плити. Для більшості людей буває важко знайти час, щоб піти за продуктами, не кажучи вже про приготування їжі вдома. Використання служби доставки є також актуальною для замовлення у офіс, для співробітників. Тим самим підприємству не потрібно тримати свою кухню, що впливає на чистий дохід. Саме тому й зростає попит на послуги доставки їжі.

Об’єктом даного дослідження виступає процес доставки готових страв.

Предметом дослідження виступають сучасні засоби автоматизації процесу доставки готових страв.

Метою дослідження є розробка прототипу застосунку для автоматизації процесу управління службою доставки готових страв.

Завдання:

– охарактеризувати процес доставки продукції;

– проаналізувати ринок автоматизованих систем служб доставки;

– сформувати вимоги до системи та обрати технології розробки;

– розробити модель бізнес-процесів;

– описати архітектуру автоматизованої системи та обрати технології вирішення поставлених завдань;

– визначити функціональну структуру завдання;

– описати структуру та особливості реалізації інформаційного та алгоритмічного забезпечень;

– надати контрольний приклад та інструкцію щодо використання;

– оцінити очікуваний ефект від впровадження системи.

Результати дипломної роботи можуть бути використаними ресторанами, кафе та іншими закладами харчування для підвищення ефективності продажів та автоматизації системи управління службою доставки, а також всіма бажаючими жителями міста, котрі мають намір задовольнити власні потреби.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ОСОБЛИВОСТЕЙ БІЗНЕС-ПРОЦЕСІВ СЛУЖБИ ДОСТАВКИ

1.1 Теоретична характеристика процесу доставки продукції

Служба доставки – це послуга доставки продуктів харчування та інших товарів до покупця додому/офіс. Завдяки службам доставки їжі клієнти можуть найняти кур'єрів, які забиратимуть їжу від їхнього імені.

За останні роки в Україні великою мірою зросла кількість служб доставки їжі. Спочатку відбувалось зростання попиту на подібні послуги у великих містах, проте останнім часом дана тенденція стає актуальною й для малих міст. Це пояснюється розповсюдженням коронавірусної хвороби (COVID-19). Новий коронавірус 2019 року – це новий вірус [31, 6], який спричиняє розвиток респіраторних захворювань у людей та може передаватися від людини до людини. Саме тому велика кількість людей перестала виходити з власних будинків на вулицю для запобігання захворювання. Це і визвало популяризацію кур'єрських служб. Іншим фактором розповсюдження доставок є відсутність бажання у сучасних людей готувати вдома через швидкий ритм життя або ненормований робочий день, що не дозволяє годинами простоювати біля плити. Для більшості людей буває важко знайти час, щоб піти за продуктами, не кажучи вже про приготування їжі вдома.

З економічної сторони відбудеться задоволення наростаючого попиту, власне підприємства в цій зоні обслуговування почнуть отримувати прибуток і швидко окуплять початкові витрати.

Тому зростає попит на послуги доставки їжі. Ті, хто починає бізнес з доставки їжі, можуть заробляти на цьому простому бажанні людей. Можливість доставки додому під час пандемії актуальна як ніколи. Тут підійде служба доставки додому. Бізнес у сфері доставки їжі досить

диверсифікований, і є куди розвиватися та діяти. Великою перевагою цієї сфери діяльності є те, що тут є місце як досвідченим підприємцям (власникам кафе чи ресторанів), так і бізнесменам-початківцям [34].

Стосовно історії виникнення доставки готових страв, необхідно звернутися до давніх історій. Так, ідея «візьми із собою» відома ще з давньоримських часів, коли існували термополіуми. Це були закусочні, де корінне населення мало можливість придбати готові страви. Як і сучасний фаст-фуд, їх подавали на винос у великому глиняному посуді, який підтримував температуру. Для багатьох римлян, які не могли дозволити собі власну кухню, це був єдиний шанс поїсти гарячої їжі [30].

Цікава історія доставки від легенди про піци. У 1889 році король Італії Умберто та королева Маргарита замовили доставку піци до своїх палаців у Неаполі. На їхнє прохання шеф-кухар приготував піцу з моцарелою, помідорами та базиліком. Таким чином відбулось не лише народження знаменитої піци Маргарита, а й народження концепції доставки.

Окремо варто сказати про доставку парного молока. У міру індустріалізації Сполучених Штатів дедалі менше людей жили у сільській місцевості. У той час молоко було швидкопсувним, а холодильників дома були мало в кого, тому були потрібні щоденні поставки. В результаті ранкові постачання молока стали давньою традицією для багатьох американців [12].

Таким чином, служби доставки користувалися великим попитом ще з давніх часів, полегшуючи життя звичайного населення та задовольняючи наростаючий попит.

1.2 Аналіз ринку автоматизованих систем служб доставки

Провівши дослідження ринку мобільних додатків на рахунок існування альтернатив для служб доставки, мною було знайдено декілька варіантів: Glovo, Mister.Am та Rocket.

Glovo – це іспанська служба доставки [11], популярна у більш ніж 20 країнах, та реалізована через мобільний додаток. Користувачі мають змогу замовляти будь-які товари до 9кг. Застосунок містить великий вибір ресторанів, саме тому найбільшою популярністю користується саме доставка готових страв. В додатку реалізована можливість відстеження переміщення кур'єра на мапі в режимі реального часу. Поданий сервіс працює у web, iOS та Android системах (рис. 1.1-1.2).

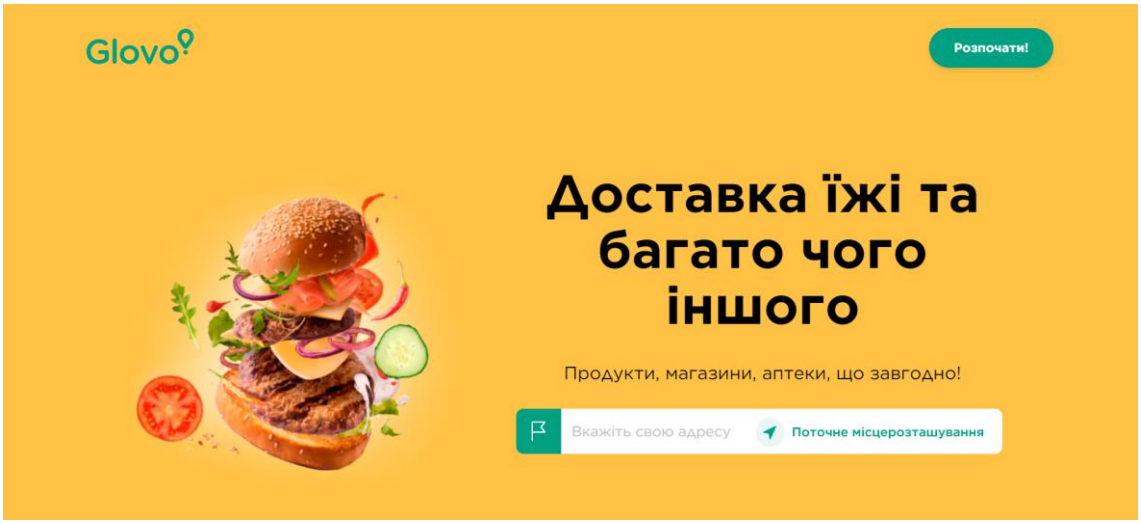

Рисунок 1.1 – Офіційна сторінка Glovo

Рисунок 1.2 – Прикладні програми сервісу

Mister.Am – популярна в Україні служба доставки [18], яка працює у 16 містах країни. Її сервіс – єдина система, котра поєднує сотні кафе, ресторанів, піцерій та суші-барів з відмінною репутацією. Має сучасний сервіс та високий рівень відповідальності. Сервіс має бонусну систему у вигляді кешбеку, який

можна застосувати для сплати наступних покупок або погашення частини суми від замовлення. Прикладна програма реалізована в web, iOS та Android системах (рис. 1.3-1.4).

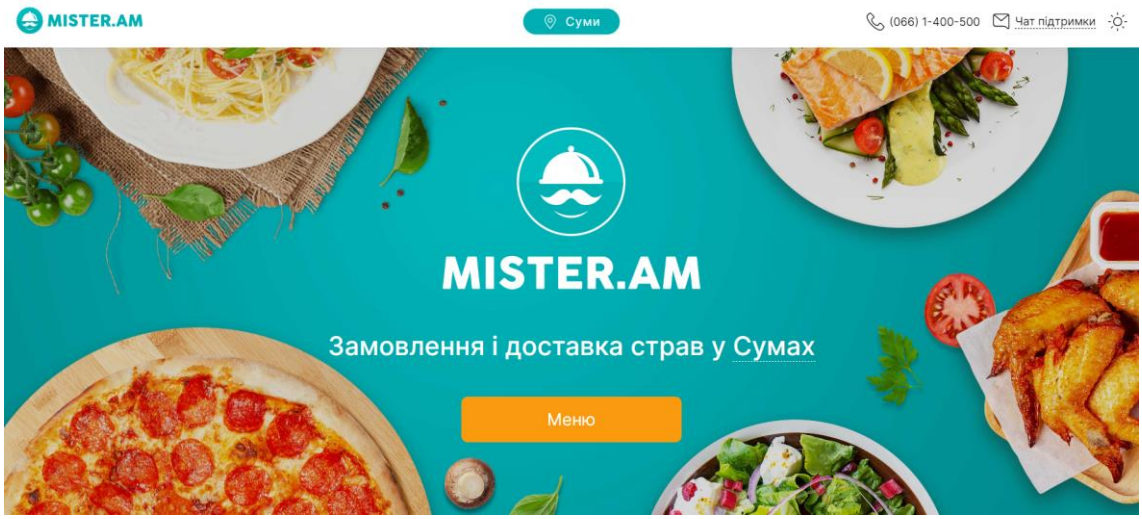

Рисунок 1.3 – Офіційна сторінка Mister.Am

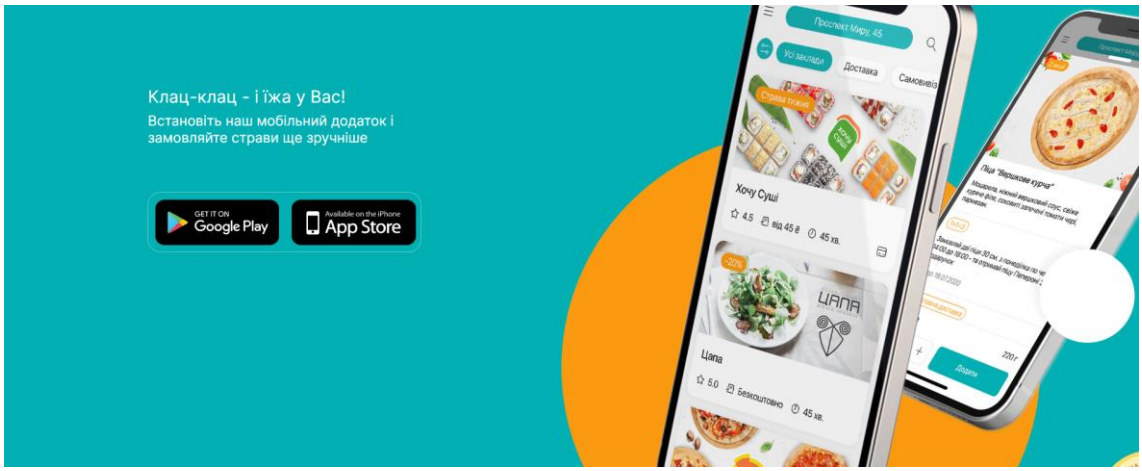

Рисунок 1.4 – Прикладні програми служби доставки

Rocket – українська компанія, що надає послуги з доставки їжі та продуктів під брендом Rocket за допомогою мобільного застосунку [35]. Основною послугою компанії є можливість замовлення доставки з партнерських магазинів та закладів харчування користувачем. Проте з 18 березня 2022 року, після повномасштабного вторгнення росії в Україну, співзасновники повідомили про припинення роботи в Україні.

Аналізуючи інформацію про вищезгадані автоматизовані системи, можна зробити висновок про те, що можливість зайняти сегмент ринку служб доставки дійсно існує. Представлену ідею можна реалізувати шляхом створення власної автоматизованої системи служби доставки з урахуванням особливостей конкуруючих служб та оптимізованою системою управління.

Отже, система SumyEats – це комплекс прикладних програм для користувача та кур’єра, створена для досягнення мети автоматизації процесів доставки у місті Суми.

1.3 Формування вимог до системи

Зі сторони взаємодії з користувачем та кур’єром система повинна гарантувати зручний, швидкий, безпечний та комбінований з іншими мовами програмування процес створення мобільного додатку, який забезпечить виконання основного завдання – автоматизацію служби доставки.

Таким чином, основними вимогами є:

– орієнтованість на користувачів обох платформ: iOS та Android для більшого охвату;

– можливість взаємодії з сервісом backend-у для прискорення онлайн реалізації системи;

– можливість створення, редагування та видалення основних таблиць з даними;

– інтуїтивно зрозумілий та простий інтерфейс.

Усі вимоги поданої системи поділяються на функціональні (процеси обробки вхідних масивів даних) та нефункціональні (зручність використання застосунків, надійність даних та продуктивність). Стосовно зручності використання – система повинна бути інтуїтивно зрозумілою для користувача, інтерфейс повинен бути простим, а управління логічним. Стосовно надійності – система повинна мати стійкість до помилок. Продуктивність повинна проявлятися у швидкій обробці даних за найменший проміжок часу.

РОЗДІЛ 2. ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Розробка моделі бізнес-процесів

Бізнес-процес – це послідовність операцій у результаті створення нового продукту/послуги, тобто, це інструмент для вирішення бізнес-проблем. Подані операції лежать в основі діяльності кожної організації, котрі визначаються місією та цілями підприємства [39]. Бізнес-процеси використовують для виконання всіх дій організацією та її працівниками, забезпечуючи виконання корпоративних функцій [38]. Таким чином, представлено основний бізнес-процес автоматизованої системи служби доставки (рис. 2.1):

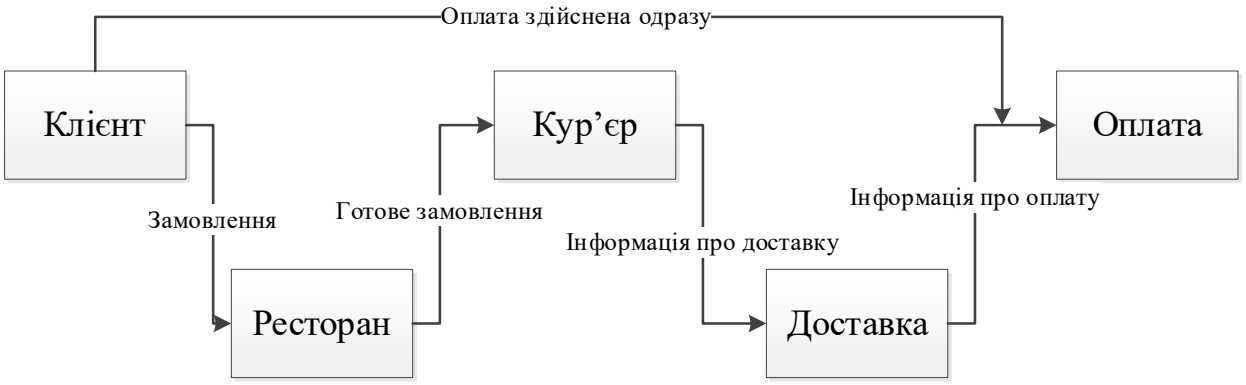

Рисунок 2.1 – Основний бізнес-процес служби доставки

На етапі замовлення клієнт обирає зі списку ресторанів з меню та вартістю страв, інформація про замовлення передається до ресторану

На етапі ресторану відбувається приготування страви, формується її вартість, змінюється статус замовлення та відбувається видача страв.

Кур'єр по побудованому оптимальному маршруту до клієнта в залежності від обраного транспортного засобу, формування ціни та часу доставки передається ця інформація на етап доставки.

Етап доставки відбувається за рахунок побудованого оптимального маршруту. Головне в цьому етапі - врахування затримки - часу доставки до клієнта.

Останнім етапом є оплата. Замовлення вважається завершеним при доставці та оплаті за умови оплати готівкою, або ж доставці (при умові, що було сплачено онлайн під час формування замовлення).

2.2 Архітектура автоматизованої інформаційної системи та технології вирішення поставлених завдань

Для реалізації бекенду, мною була обрана система AWS Amplify.

AWS Amplify – це набір спеціально створених інструментів і функцій, які дозволяють веб-розробникам веб-інтерфейсу та мобільних пристроїв швидко й легко створювати повний стек програм на AWS. Amplify надає дві послуги: Amplify Hosting і Amplify Studio. Amplify Hosting забезпечує робочий процес на основі git для розміщення повного стека безсерверних веб-програм із постійним розгортанням. Цей посібник користувача містить інформацію, необхідну для початку роботи з Amplify Hosting [25].

AWS Amplify дозволяє:

1) створити серверну частину AWS для веб-програми, програми для iOS або Android із автентифікацією, даними, сховищем;
2) візуально створити зовнішній інтерфейс користувача та підключити інтерфейс користувача до серверної частини;
3) розгортати та розміщувати швидкі, безпечні, надійні веб-сайти та додатки, що відображаються на стороні сервера;
4) розширити свою програму за допомогою 175+ служб AWS для підтримки нових варіантів використання, практик DevOps і збільшення кількості користувачів.

Автоматизована система SumyEats має клієнтську частину у вигляді двох мобільних додатків (SumyEatsUser та SumyEatsCourier), та серверну частину, на якому власне і відбувається управління.

Конструктор моделі даних надає API GraphQL таблиці бази даних [3]. Сервіс надає автентифікацію за допомогою соціальних постачальників, та допомагає визначати правила авторизації для захисту ваших даних. Доступ до згенерованих Amplify Studio серверних модулів можна отримати за допомогою Amplify CLI, виконавши команду amplify pull.

Після виконання вищевказаної команди, застосунок має наступну модель, готову для взаємодією з таблицями БД (рис. 2.2).

```
const OrderStatus = {
  "NEW": "NEW",
  "COOKING": "COOKING",
  "READY_FOR_PICKUP": "READY_FOR_PICKUP",
  "PICKED_UP": "PICKED_UP",
  "COMPLETED": "COMPLETED",
  "ACCEPTED": "ACCEPTED"
};

const TransportationModes = {
  "DRIVING": "DRIVING",
  "BICYCLING": "BICYCLING",
  "WALKING": "WALKING"
};

const { User, Order, OrderDish, Dish, Restaurant, Basket, BasketDish, Courier } = initSchema(schema);

export {
  User,
  Order,
  OrderDish,
  Dish,
  Restaurant,
  Basket,
  BasketDish,
  Courier,
  OrderStatus,
  TransportationModes
};
```

Рисунок 2.2 – Модель взаємодії з БД

Порівняння основних інструментів для створення мобільного додатку:

У 2022 році найпопулярнішими інструментами для створення мобільних додатків є інтегровані середовища розробки з нативними мовами програмування Xcode і Android Studio та фреймворки Flutter і ReactNative.

1. Xcode – це інтегроване середовище розробки (IDE) від Apple для macOS, яке використовується для розробки програмного забезпечення для macOS, iOS, iPadOS, watchOS і tvOS. Спочатку він був випущений наприкінці 2003 року; остання стабільна версія – версія 14.2, випущена 13 грудня 2022 року через Mac App Store з macOS Monterey [28]. Набір програмного забезпечення надається безкоштовно. Зареєстровані розробники можуть завантажити попередні випуски та попередні версії набору через веб-сайт Apple Developer. Xcode містить інструменти командного рядка, які дозволяють розробку в стилі UNIX через програму Terminal у macOS. Їх також можна завантажити та встановити без графічного інтерфейсу користувача [27].

На сьогодні основною мовою програмування поданого середовище є мова Swift. Swift – це потужна та інтуїтивно зрозуміла мова програмування для iOS, iPadOS, macOS, tvOS і watchOS. Написання коду Swift інтерактивне та веселе, синтаксис стислий, але виразний, а Swift містить сучасні функції, які подобаються розробникам. Код Swift є безпечним за своєю конструкцією та створює програмне забезпечення, яке працює блискавично [22].

Плюси використання Swift для нативної розробки iOS [23]:

1) Швидкий процес розвитку – чисту та виразну мову зі спрощеним синтаксисом і граматикою, Swift легше читати та писати. Він дуже стислий, а це означає, що для виконання того самого завдання потрібно менше коду порівняно з Objective-C. Створення додатків для iOS за допомогою Swift зазвичай займає менше часу.

2) Легше масштабувати продукт і команду – окрім швидшого часу розробки, ви отримуєте продукт, який готовий до майбутнього та може бути розширений новими функціями за потреби. Таким чином, проекти Swift зазвичай легше масштабувати.

3) Покращена продуктивність та швидкість розробки – завдяки зосередженню на продуктивності та швидкості мова спочатку була розроблена, щоб перевершити свою попередницю. Крім того, Swift було

створено з фреймворком компілятора LLVM, який перекладає мову асемблера в машинний код і оптимізує код, прискорюючи розробку.

4) Безпека – його потужна система введення та обробки помилок запобігає збоям у коді та помилкам у виробництві. Таким чином, Swift має коротший цикл зворотного зв’язку, що дозволяє миттєво бачити помилки в коді та виправляти їх на ходу, що значно скорочує час і зусилля, необхідні для виправлення помилок, і усуває ризики розгортання коду низької якості.

5) Автоматичне керування пам'яттю за допомогою ARC – Swift використовує автоматичний підрахунок посилань – технологію, спрямовану на додавання функції збирача сміття, яка раніше не була представлена в iOS. ARC Swift визначає, які екземпляри більше не використовуються, і позбавляється від них від вашого імені. Це дозволяє підвищити продуктивність програми без затримки пам’яті чи ЦП.

6) Повний потенціал стека та підтримка різних пристроїв – ініціатива розміщення мови в хмарі, яку активно просуває IBM, наразі була досить успішною.

Мінуси мови програмування Swift [16]:

1) Мова ще досить молода – Swift був представлений у світі в 2014 році, що здається давно, але самій мові насправді лише 7 років, порівняно з Objective C, що живе з 1980-х років. Нещодавнє оновлення забезпечило стабільність ABI на платформах Apple, зворотну сумісність версій Swift і оновлену документацію. Це величезні кроки до того, щоб зробити Swift більш зрілою мовою.

2) Обмежений резерв талантів – хоча спільнота Swift швидко зростає, вона все ще значно менша порівняно з будь-якою іншою мовою з відкритим кодом. Згідно з останнім опитуванням розробників StackOverflow, лише 5.1 відсотка з 83 053 респондентів використовують Swift. Це означає, що при виникненні помилки під час створення мобільного додатку, допомогти усунути її буде нікому.

3) Погана взаємодія зі сторонніми інструментами та IDE – значною мірою через часті оновлення, як згадувалося вище, часто буває важко знайти потрібні інструменти для вирішення певних завдань. Крім того, офіційна Apple IDE, XCode, відстає в плані інструментарію та підтримки Swift. Розробники часто повідомляють про проблеми з підсвічуванням синтаксису, автозаповненням, інструментами рефакторингу та компіляторами.

4) Неповна кросплатформна підтримка – Swift підтримує всі платформи Apple, а також Linux і Windows. Однак він був спочатку створений і досі найкраще працює для нативної розробки iOS, залишаючи міжплатформний ринок React Native і Flutter.

Таким чином, проаналізувавши вищезазначені плюси і мінуси, мною було відхилено інтегроване середовище розробки Xcode, використовуючи мову програмування Swift через яскраво виражені недоліки поганої взаємодії зі сторонніми інструментами IDE, неповною кросплатформною підтримкою та відсутністю створення мобільного додатку на операційній системі Android, що є однією з основним вимог створення автоматизованої служби доставки.

2. Android Studio – це офіційне інтегроване середовище розробки (IDE) для розробки додатків Android [13]. Він заснований на IntelliJ IDEA, інтегрованому середовищі розробки Java для програмного забезпечення, і містить його інструменти для редагування коду та розробника [14, 15].

Для підтримки розробки додатків в операційній системі Android Android Studio використовує систему збірки на основі Gradle, емулятор, шаблони коду та інтеграцію з Github [10]. Кожен проект в Android Studio має одну або кілька модальностей із вихідним кодом і файлами ресурсів. Ці модальності включають модулі програми Android, модулі бібліотеки та модулі Google App Engine.

Android Studio використовує функцію Instant Push для надсилання коду та змін ресурсів у запущену програму. Редактор коду допомагає розробнику писати код і пропонує доповнення, заломлення та аналіз коду. Програми,

створені в Android Studio, потім компілюються у формат APK для надсилання в магазин Google Play [5].

Програмне забезпечення було вперше оголошено на Google I/O у травні 2013 року, а перша стабільна збірка була випущена в грудні 2014 року. Android Studio доступний для настільних платформ Mac, Windows і Linux. Він замінив Eclipse Android Development Tools (ADT) як основну IDE для розробки додатків Android. Android Studio та пакет розробки програмного забезпечення можна завантажити безпосередньо з Google [4].

Android Studio є основним інструментом для створення мобільних додатків для смартфонів на Android. На сьогодні основною мовою програмування поданого середовище є мова Kotlin [17].

Kotlin – це безкоштовна, статично типізована «прагматична» мова програмування загального призначення з відкритим вихідним кодом, яка спочатку була розроблена для JVM (віртуальна машина Java) та Android і поєднує функції об’єктно-орієнтованого та функціонального програмування. Він зосереджений на сумісності, безпеці, чіткості та підтримці інструментів. Версії Kotlin, націлені на JavaScript ES5.1 і нативний код (з використанням LLVM) для ряду процесорів, також знаходяться у виробництві [26].

Kotlin виник у JetBrains, компанії, що стоїть за IntelliJ IDEA, у 2010 році, а з 2012 року є відкритим кодом. Проект Kotlin на GitHub має понад 770 учасників; хоча більшість команди працює в JetBrains, у проекті Kotlin брали участь майже 100 зовнішніх учасників. JetBrains використовує Kotlin у багатьох своїх продуктах, включаючи свій флагман IntelliJ IDEA.

Переваги Kotlin для розробки програм Android [1]:

1) легко зрозуміти – Kotlin прагне не лише переписати, але й покращити функціональність Java. Кожен проект, пов’язаний із Kotlin, також може бути предметом усіх навичок, які розробник мобільних додатків навчився та отримав під час роботи на Java;

2) має менше ймовірності помилок – Kotlin не залишає місця для помилок для більш простої та легкої кодової бази. Також видаються більш стабільні вихідні коди. Kotlin стає безпечнішою альтернативою Java;

3) більш надійна мова – у порівнянні з багатьма іншими традиційними мовами програмування, Kotlin є набагато досконалішою. У 2011 році народився Котлін. Перш ніж запустити остаточну версію, Kotlin пройшов численні бета- та альфа-рівні з моменту випуску;

4) простота в обслуговуванні – Kotlin підтримує багато IDE, включаючи Android Studio та багато іншого програмного забезпечення SDK. Це має на меті збільшити загальну продуктивність розробника, оскільки він може вмістити набори інструментів, які вони все ще часто використовують;

5) узгодження з існуючим кодом Java – інтероперабельна в Java мова програмування Kotlin. Це Java та кілька інших фреймворків і програмного забезпечення, пов’язаних із проектом. Це робить можливим перехід на Kotlin, і дві мови програмування можна використовувати одночасно;

6) підвищення продуктивності – Kotlin це мова програмування на основі Java. Необхідно усунути громіздкість і застарілість Java. Kotlin має інтуїтивно зрозумілий і стислий синтаксис, крім того, це проста, портативна та потужна мова. Таким чином Kotlin може оптимізувати загальну продуктивність цілої команди розробників.

Недоліки Kotlin для розробки програм Android

1) Kotlin має мінімальні можливості навчання – хоча більшість розробників переходять на Kotlin, невелика кількість розробників доступна по всьому світу. Він надає мінімальні інструменти для вивчення мови програмування та різноманітних запитів у процесі розробки програмного забезпечення;

2) швидкість компіляції нижча – у деяких випадках Kotlin працює швидше, ніж Java, особливо під час інкрементних конструкцій. Але зауважте, що коли справа доходить до акуратної побудови, Java завжди буде розвиватися;

3) значущі відмінності від Java – навіть якщо Kotlin і Java мають деякі подібності, вони все одно мають кілька основних відмінностей. Після детального вивчення Kotlin розробники мобільних додатків не можуть переходити на іншу мову програмування.

Отже, провівши аналіз плюсів та мінусів поданого середовища розробки та мови програмування, мною було вирішено не використовувати його для розробки мобільного додатку через відсутність створення прикладної програми для операційної системи iOS, а це є однією з основним вимог створення автоматизованої служби доставки.

3. Flutter – це розроблений Google фреймворк з відкритим програмним кодом, який дозволяє просто і швидко створювати мобільні додатки для iOS і Android [9].

При цьому в роботі Flutter не використовує нативні компоненти зовсім. Замість цього всі UI-елементи у фреймворку створюються за допомогою власного графічного движка. Flutter дозволяє створювати всі елементи призначеного для користувача інтерфейсу додатку з готових віджетів. У цьому Flutter схожий з іншими фреймворками – React і Vue, і в той же час має ряд відмінностей від них. Так, він не використовує мову програмування Javascript, натомість Flutter використовує мову Dart [36].

Переваги Flutter для розробки мобільних додатків [8]:

1) Має велику спільноту та підтримку від Google, що можна з упевненістю припустити, що Flutter є одним з найкращих виборів, коли йдеться про розробку мобільних додатків.

2) Продуктивність – незважаючи на те, що його досить легко підтримувати на високому рівні, у випадках, коли виникають проблеми з оптимізацією, ви можете використовувати «накладення продуктивності», яке дозволяє перевірити, які дії є дорогими для рендерингу, а потім легко оптимізувати їх.

3) Підтримка анімацій – все плавно та легко у використанні, зі стабільною частотою оновлення 60/120 кадрів на секунду.

4) Можливість використання особливостей платформи – завдяки простим у користуванні каналам платформи, які дозволяють прямий зв’язок із рідною платформою, вам не доведеться турбуватися про те, що вас заблокують у використанні рідні особливості.

5) Завдяки величезній бібліотеці пакетів Flutter, добре продуманим API та чудовим віджетам, які надаються одразу, команда може писати нові функції швидше та дешевше з меншим кодом.

Недоліки Flutter для розробки мобільних додатків:

1) Розмір додатку – дазовий додаток для Android у Flutter більш ніж у 3 рази більший, ніж еквівалент у Kotlin. Якщо доступ до стабільного мережевого з’єднання обмежений або дуже дорогий, для деяких людей це може стати перешкодою.

2) Слабка веб-підтримка – хоч вона і має великий потенціал, але не замінить веб-програми в таких технологіях, як React, у найближчому майбутньому. Особливо для веб-порталів, які значною мірою залежать від трафіку, створеного SEO та веб-позиціонуванням.

3) Програми Flutter на iOS зазвичай дотримуються Material Design замість Рекомендацій щодо людського інтерфейсу. Уніфікований інтерфейс для iOS і Android забезпечує велику економію, але якщо дизайнери UI/UX сильно прив’язані до поведінки однієї платформи, користувачі іншої платформи вважатимуть інтерфейс менш інтуїтивно зрозумілим.

4) Існує також випадок деяких нових функцій, специфічних для платформи. Між додаванням чогось до операційної системи та наданням доступності у Flutter є затримка, і може знадобитися власне впровадження, перш ніж це зробить команда Flutter.

Таким чином, можна стверджувати, що фреймворк Flutter з мовою програмування Dart є чудовим варіантом для створення мобільного додатку в майбутньому. Але зараз він є дуже молодим та має недоліки у порівнянні з наступним фреймворком, тому поданий фреймворк відходить на другий план.

4. ReactNative – це фреймворк, котрий має відкритий вихідний код для створення мобільних додатків на мовах JavaScript та TypeScript, який підтримує такі мобільні операційні системи, як Android та IOS одночасно. Це є значною перевагою поданого фреймворку, бо пишучи один код, створюється 2 мобільні додатки для різних ОС [21,24].

Існує два популярних способи створення програми React Native використовуючи Expo CLI або React Native. Необхідно провести глибокий аналіз ReactNative Init та Expo щоб з'ясувати, що найкраще підходить для поданого проєкту [20].

ReactNative став популярним фреймворком для створення кросплатформних додатків за допомогою JavaScript. Його головною перевагою є можливість створювати кросплатформні програми. Вони дозволяють надати набагато кращий інтерфейс користувача, ніж поточні гібридні варіанти на ринку, ближче до рідних програм. Примітиви React рендеряться до рідного інтерфейсу користувача платформи. Це означає, що ваша програма використовуватиме ті самі API рідної платформи, що й інші програми. Щоб почати, вам знадобиться Xcode або Android Studio.

Expo – це платформа для створення додатків React Native. Це набір інструментів і служб, створених для React Native. Це допоможе вам легко розпочати створення додатків React Native. Він надає вам список інструментів, які спрощують створення та тестування програми React Native. Крім того, Expo забезпечує більш надійний і зручний робочий процес розробки з гнучкістю.

Отже, Expo та React Native App – це інструменти «Кросплатформенної мобільної розробки».

Плюси React Native:

1) Існує можливість включати власні модулі на Java/Objective-C.
2) Розробляти файли .apk і .ipa набагато легше, ніж за допомогою Expo.
3) Має багаторазовий код і готові компоненти.
4) Спрощений інтерфейс користувача.

5) Підтримка сторонніх плагінів.

6) Модульна архітектура.

React Native зробив великий поштовх у розробці мобільних додатків, але він має деякі недоліки. Нещодавно Airbnb і Udacity поділилися своїм досвідом у React Native. Вони дійшли висновку, що ця платформа має багато переваг. Але ви не можете використовувати його для створення будь-яких мобільних додатків.

Мінуси React Native CLI:

1) Для запуску проектів потрібні Android Studio та XCode.

2) Ви не можете створити програму для iOS, не маючи Mac.

3) Для тестування потрібно підключити пристрій через USB.

4) У XCode немає необхідності імпортувати шрифти вручну.

5) Якщо ви хочете надіслати свою програму комусь, вам потрібно надіслати весь файл .apk/.ipa.

6) Ви повинні встановити та підключити, наприклад, npm Push-Notifications, Asset Manager.

7) Правильно налаштувати робочий проект досить складно і може зайняти час.

8) Він вимагає високого рівня конфігурації, для цього потрібні базові знання структури папок Android та iOS.

Плюси Expo CLI:

1) Немає необхідності в посиланні та безлічі бібліотек – Expo створив для вас бібліотеку, і ви можете легко її інтегрувати, також існує багато бібліотек JavaScript, які можна використовувати з Expo.

2) Кращий досвід розробки – розробникам не потрібно підключати телефон до комп’ютера, щоб запустити його. Він може запускати програму, яку ви кодуєте, через Wi-Fi, а також може синхронізувати між різними телефонами, що чудово. Тож є можливість набирати текст на своєму комп’ютері, а коли ви вносите зміни, телефони інших людей також оновлюються. І це дуже приємно.

3) Легше оновити до нових версій – оновлення відбувається набагато плавніше. У документі оновлення розписані кроки, які потрібно виконати, щоб оновити його до нової версії.

4) Легше розгорнути в Apple/Google Store – набагато простіше розгортати мобільні додатки в магазинах Apple/Google Play. Він обробляє ваші ключі та підписує облікові дані та сертифікати за вас. Це робить речі більш простими.

5) Мобільний інтерфейс – набагато простіше переглядати як на Android, так і на iPhone за допомогою сканування штрих-коду. Це також легше переглядати за допомогою онлайн-симуляторів iPhone та Android.

6) Expo SDK доступний – програми Expo поставляються з Expo SDK. І це відкриває багато можливостей, які допоможуть вам. Сканер штрих-кодів, MapView, ImagePicker та багато інших доступні для використання.

7) Створення файлів .apk і .ipa – Expo передбачає створення файлу IPA для iOS і файлу APK для Android.

Мінуси Expo CLI:

1) Обмеження – Expo не має можливості використовувати нативні модулі.

2) Бібліотеки – Expo створює версії багатьох бібліотек, і ви можете вільно використовувати їх у своїх програмах. Отже, є конкретні рідні бібліотеки модулів або власні модулі, які потрібні для розробки, але їх не можна використовувати в Expo.

3) Деякі API для iOS і Android недоступні – деякі з API пристрою не підтримуються: ні Bluetooth, ні Web RTC. Але багато функцій зараз розробляються, тому це хороший спосіб перевірити список запитів на функції.

Таким чином, мною було прийняте рішення використовувати саме Expo CLI для створення додатків React Native. Подана платформа містить всі необхідні інструменти для розробки автоматизованої служби доставки та відповідає всім вищевказаним вимогам.

2.3 Функціональна структура завдання

Система повинна реалізовувати автоматизоване управління службою доставки.

Функції системи поділяються на дві частини:

1) Для задоволення потреб користувача для клієнта;

2) Для покращення умов праці для кур’єра.

Для клієнта система повинна забезпечувати наступне:

– можливість перегляду інформації про заклади харчування для ефективної оцінки;

– можливість вибору страв, необхідних для задоволення власних потреб;

– можливість відображення ціни кожної страви для аналізу власних можливостей купівлі того чи іншого товару;

– можливість відображення вартості доставки до пункту призначення;

– можливість відслідковування статусу замовлення та його переміщення до кінцевого результату – а саме до отримання на руки.

Для кур’єра повинні забезпечуватися наступні елементи:

– можливість вибору транспортного засобу, необхідних для переміщення по місту;

– можливість вибору замовлень з короткою інформацією про кожне;

– можливість прокладення найкращого маршруту в залежності від обраного транспортного засобу;

– можливість отримання гідної винагороди за виконану роботу.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОТОТИПУ АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Структура та особливості реалізації інформаційного забезпечення

Інформаційне забезпечення автоматизованої системи служби доставки відбувається за допомогою сервісу AWS Amplify.

Це комплексне рішення дозволяє створити таблиці БД як за допомогою візуальний редактор, так і схеми GraphQL.

Перед представленням таблиць, необхідно звернути увагу на найпоширеніші типи даних, використаних у поданій системі:

1) ID – унікальний тип даних у системі AWS Amplify, використовується для зберігання ідентифікаційного номеру.

2) Float – це число з плаваючою точкою — це тип даних, що складається з числа, яке не є цілим числом, оскільки воно включає дріб, представлений у десятковому форматі.

3) Int – тип даних INTEGER зберігає цілі числа в діапазоні від -2 147 483 647 до 2 147 483 647 з точністю 9 або 10 цифр.

4) String – рядковий тип даних традиційно є послідовністю символів, або як літеральна константа, або як якась змінна.

Отже, система SumyEats має наступні таблиці:

1. Таблиця Restaurant – відображає головну інформацію про ресторан (рис. 3.1, додаток Б лістинг Б.1), містить наступні поля:

– id – ідентифікаційний номер, обов’язковий та унікальний параметр типу ID;

– name – назва ресторану, обов’язковий параметр, тип даних String;

– rating – рейтинг ресторану, тип Float;

– deliveryFee – вартість доставки (вказується мінімальне значення, котре в подальшому використовується для розрахунку), обов’язковий параметр типу Float;

– minDeliveryTime – мінімальний час доставки, обов’язковий параметр типу Int;

– maxDeliveryTime – максимальний час доставки, обов’язковий параметр типу Int;

– address – адреса, обов’язковий параметр типу String;

– lat – широта ресторану, обов’язковий параметр типу Float;

– lng – довгота ресторану, обов’язковий параметр типу Float.

Таблиця Restaurant має зв’язок один до багатьох до таблиці Basket та зв’язок один до багатьох до таблиці Dish.

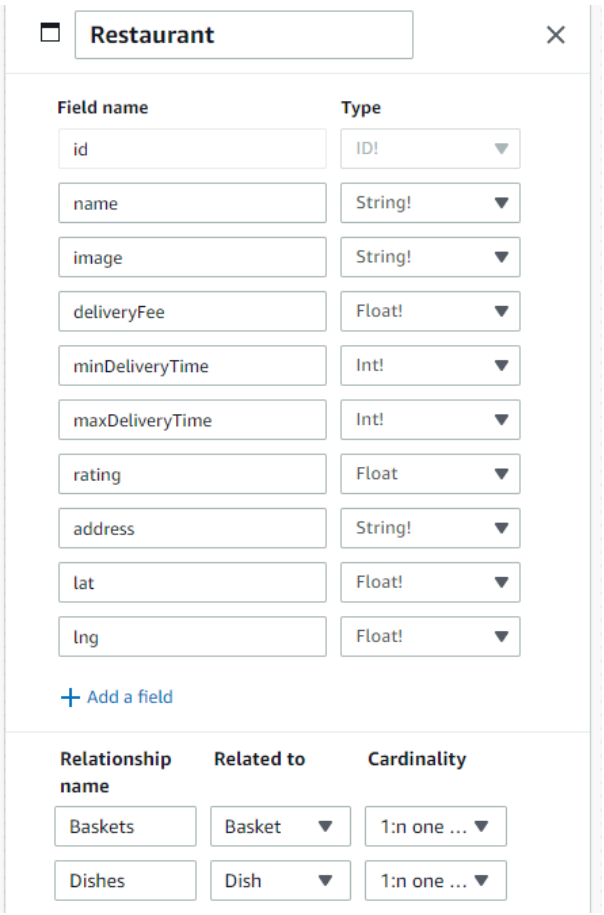

Рисунок 3.1 – Візуальний редактор таблиці Restaurant

2. Таблиця Dish – містить основну інформацію щодо страви (рис. 3.2, додаток Б лістинг Б.2), має наступні поля:

– id – ідентифікаційний номер, обов’язковий та унікальний параметр типу ID;

– name – назва страви, обов’язковий параметр, тип даних String;

– image – зображення страви, тип даних String;

– description – опис страви, тип даних String;

– price – ціна за одиницю страви, обов’язковий параметр типу Float;

– restauranID – посилання на таблицю Restaurant, створене за допомогою зв’язку з поданої таблиці.

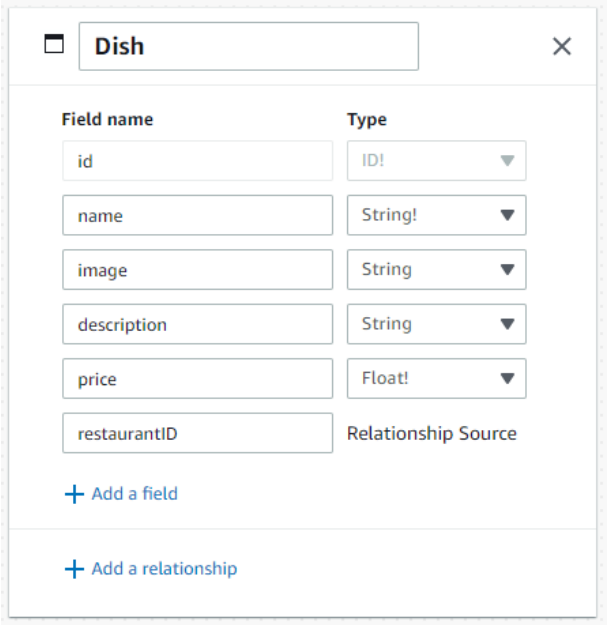

Рисунок 3.2 – Візуальний редактор таблиці Dish

3. Таблиця Basket – містить основну інформацію про кошик (рис. 3.3, додаток Б лістинг Б.3), має наступні поля:

– id – ідентифікаційний номер, обов’язковий та унікальний параметр типу ID;

– userID – посилання на таблицю User, створене за допомогою зв’язку з поданої таблиці;

– restaurantID – посилання на таблицю Restaurant, створене за допомогою зв’язку з поданої таблиці.

Таблиця Basket має зв’язок один до багатьох до таблиці BasketDish.

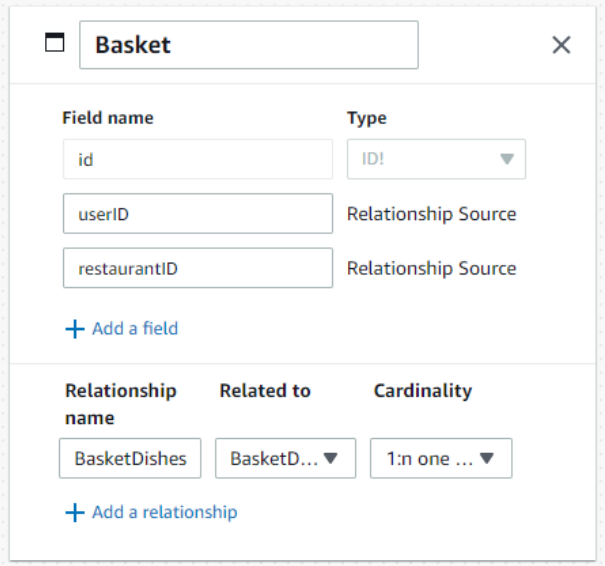

Рисунок 3.3 – Візуальний редактор таблиці Basket

4. Таблиця BasketDish – містить основну інформацію про страву кошика (рис. 3.4, додаток Б лістинг Б.4), має наступні поля:

– id – ідентифікаційний номер, обов’язковий та унікальний параметр типу ID;

– quantity – кількість страв, обов’язковий параметр, тип даних Int;

– basketID – посилання на таблицю Basket, створене за допомогою зв’язку з поданої таблиці.

Таблиця BasketDish має зв’язок один до одного до таблиці Dish.

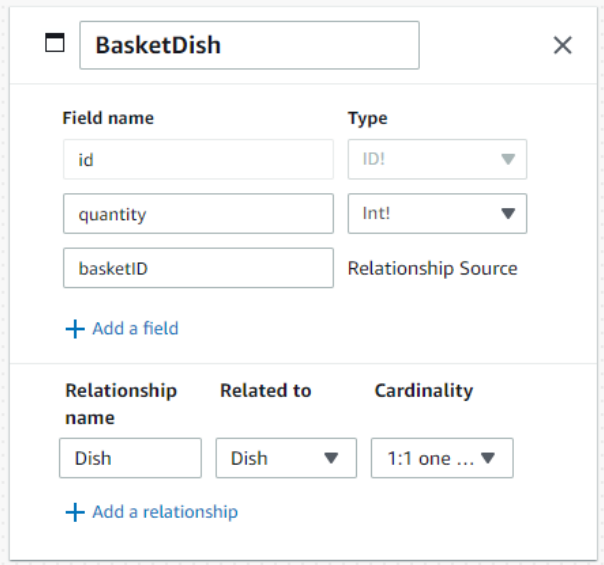

Рисунок 3.4 – Візуальний редактор таблиці BasketDish

5. Таблиця User – містить головну інформацію про юзера (рис. 3.5, додаток Б лістинг Б.5), має наступні поля:

– id – ідентифікаційний номер, обов’язковий та унікальний параметр типу ID;

– name – ім’я користувача, обов’язковий параметр, тип даних String;

– address – адреса користувача, обов’язковий параметр, тип даних String;

– lat – широта розташування користувача, обов’язковий параметр типу Float;

– lng – довгота розташування користувача, обов’язковий параметр типу Float;

– sub – основний ключ користувача, обов’язковий параметр типу String.

Таблиця User має зв’язок один до багатьох до таблиці Order та зв’язок один до багатьох до таблиці Basket.

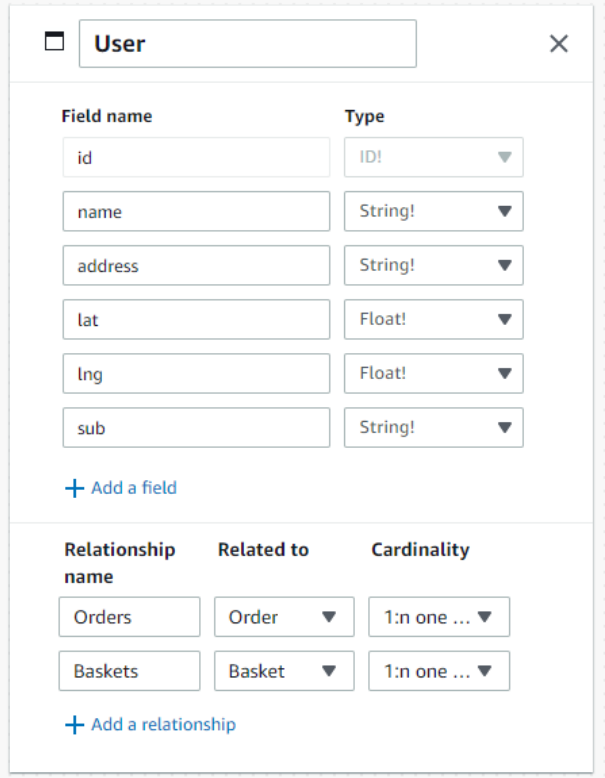

Рисунок 3.5 – Візуальний редактор таблиці User

6. Таблиця Courier – відображає головну інформацію про кур'єра (рис. 3.6, додаток Б лістинг Б.6), містить наступні поля:

– id – ідентифікаційний номер, обов'язковий та унікальний параметр типу ID;

– name – ім'я кур'єра, обов'язковий параметр, тип даних String;

– sub – основний ключ користувача, обов'язковий параметр типу String.

– lat – широта розташування користувача, обов'язковий параметр типу Float;

– lng – довгота розташування користувача, обов'язковий параметр типу Float;

– transportationMode – містить посилання на талицю TransportationModes, тип даних Enums.

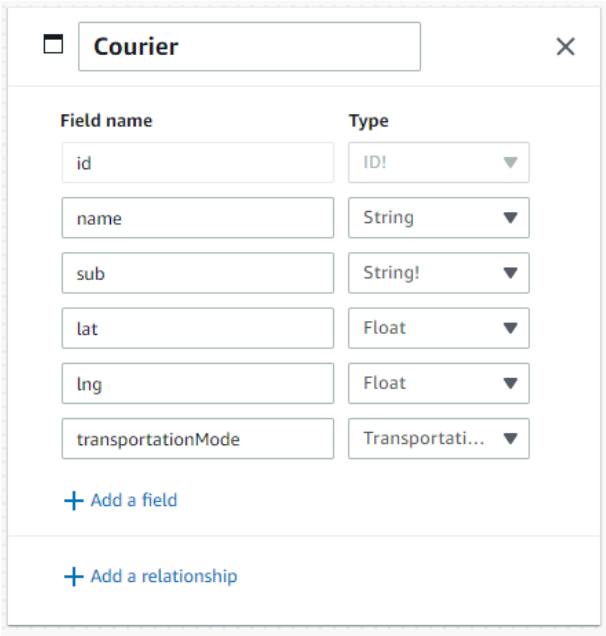

Рисунок 3.6 – Візуальний редактор таблиці Courier

7. TransportationModes – таблиця з типом даних посилання (рис. 3.7, додаток Б лістинг Б.7), доповнює таблицю Courier. Містить основні види транспорту для кур’єра.

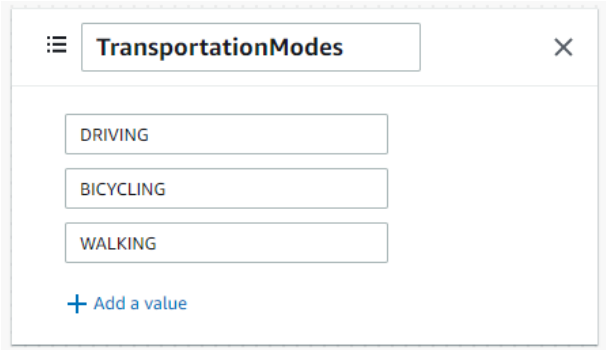

Рисунок 3.7 – Візуальний редактор таблиці TransportationModes

8. Таблиця Order – відображає головну інформацію про замовлення (рис. 3.8, додаток Б лістинг Б.8), містить наступні поля:

– id – ідентифікаційний номер, обов’язковий та унікальний параметр типу ID;

– userID – посилання на таблицю User, створене за допомогою зв’язку з поданої таблиці;

– status – містить посилання на талицю OrderStatus, тип даних Enums;

– total – загальна вартість замовлення, обов’язковий параметр типу Float.

Таблиця Order має зв’язок один до багатьох до таблиці OrderDish, зв’язок один до одного до таблиці Restaurant та зв’язок один до одного до таблиці Courier.

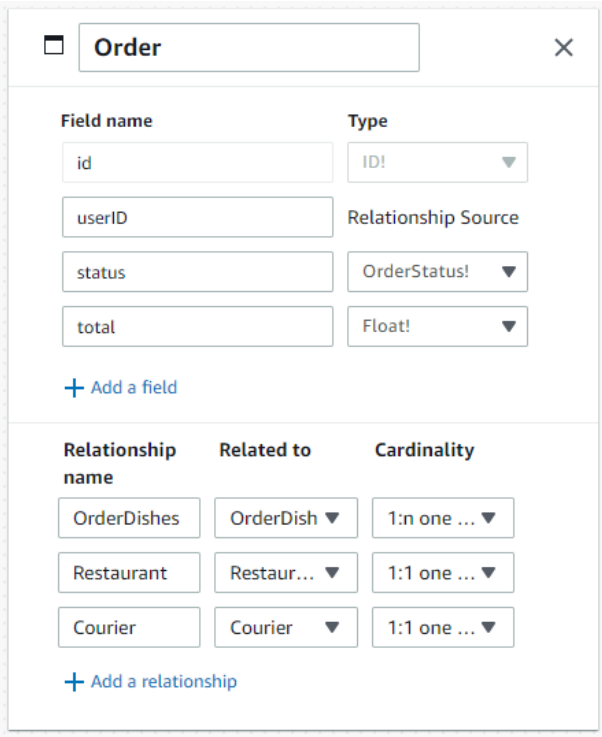

Рисунок 3.8 – Візуальний редактор таблиці Order

9. Таблиця OrderDish – відображає головну інформацію про замовлення (рис. 3.9, додаток Б лістинг Б.9), містить наступні поля:

– id – ідентифікаційний номер, обов’язковий та унікальний параметр типу ID;

– quantity – кількість страв, обов’язковий параметр, тип даних Int;

– orderID – посилання на таблицю Order, створене за допомогою зв’язку з поданої таблиці.

Таблиця OrderDish має зв’язок один до одного до таблиці Dish.

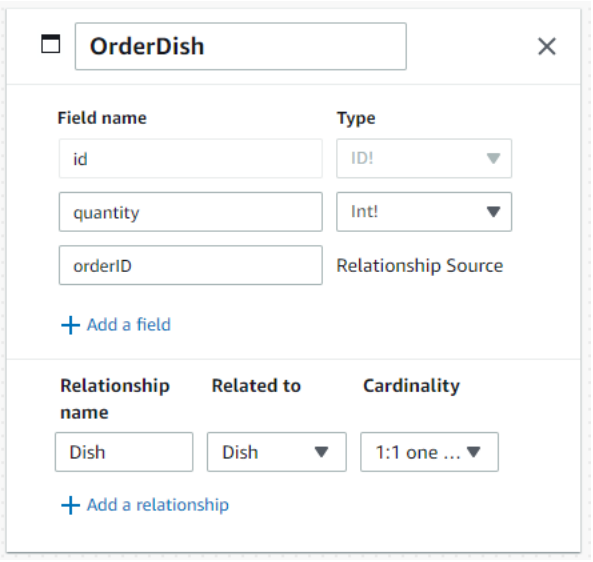

Рисунок 3.9 – Візуальний редактор таблиці OrderDish

10. OrderStatus – таблиця з типом даних посилання (рис. 3.10, додаток Б лістинг Б.10), доповнює таблицю Order. Містить статуси замовлення.

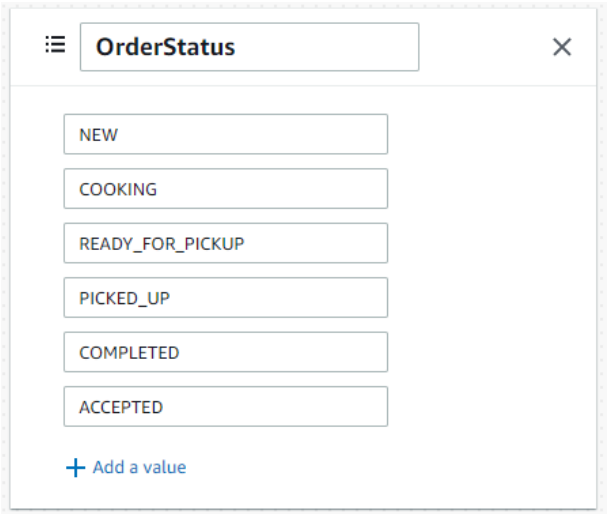

Рисунок 3.10 – Візуальний редактор таблиці OrderStatus

Для початку роботи з системою SumyEats, необхідно заповнити таблицю Restaurant, адже вона є однією з основних таблиць для подальшого заповнення даними (рис. 3.11).

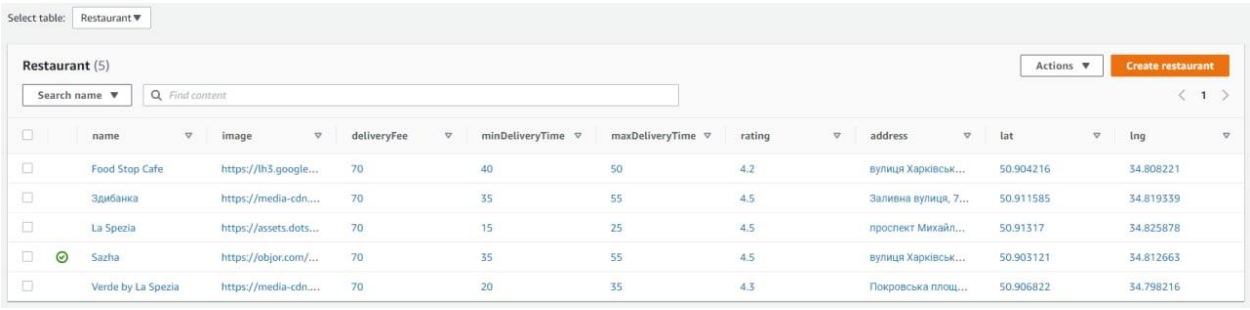

Рисунок 3.11 – Заповнена даними таблиця Restaurant

Наступним етапом підготовки системи є заповнення таблиці Dish (рис. 3.12).

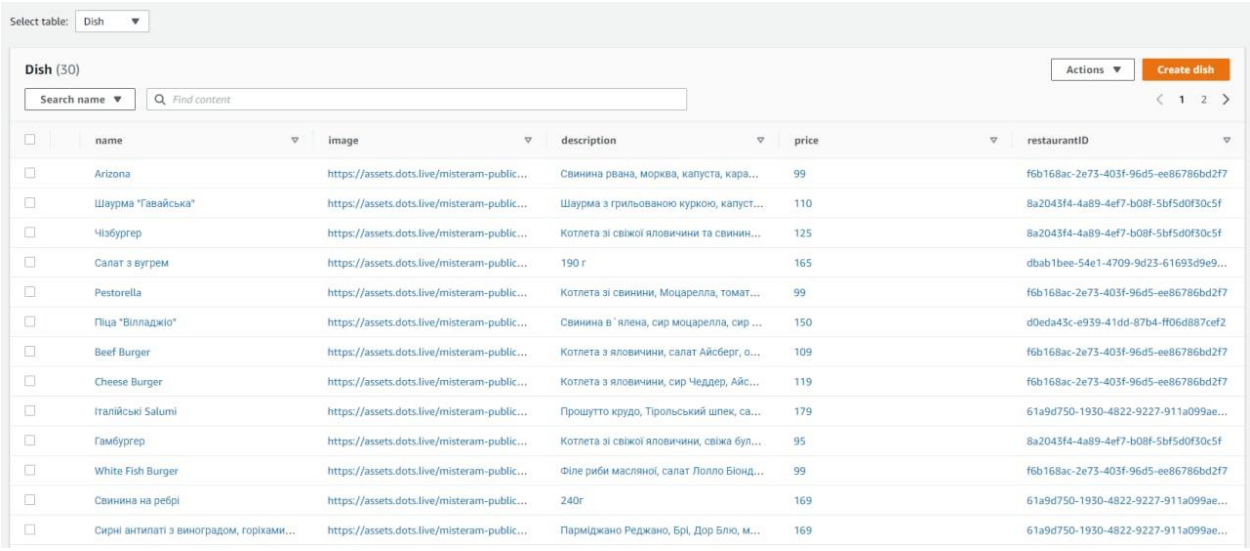

Рисунок 3.12 – Заповнена даними таблиця Dish

Основні елементи таблиці є заповненими. Тепер необхідно додати користувачів для використання. Отже, таблиця User набуватиме наступного вигляду:

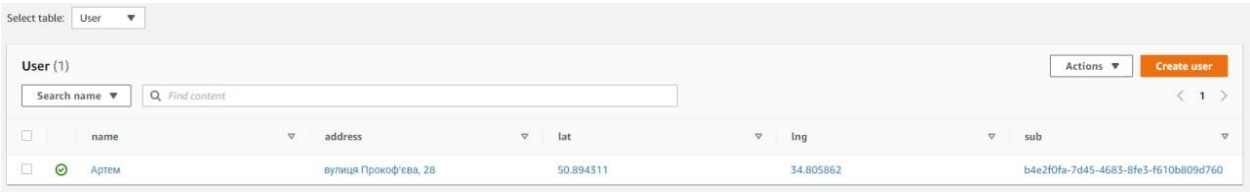

Рисунок 3.13 – Заповнена даними таблиця User

Для здійснення доставки системі необхідно мати кур’єра, рисунок 3.14.

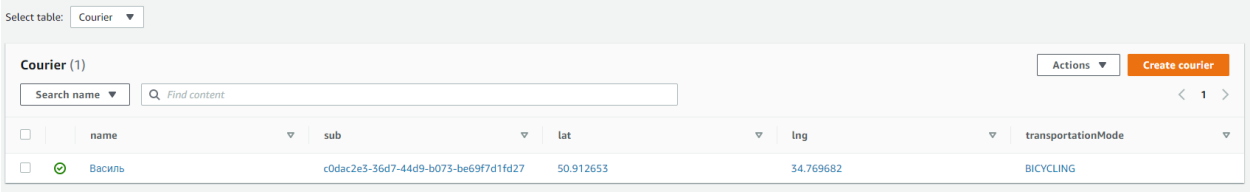

Рисунок 3.14 – Заповнена даними таблиця Courier

Отже, маючи основну інформацію для вибору замовлення, відбудеться заповнення таблиці Basket. Поданий процес відбувається коли користувач обрав страву.

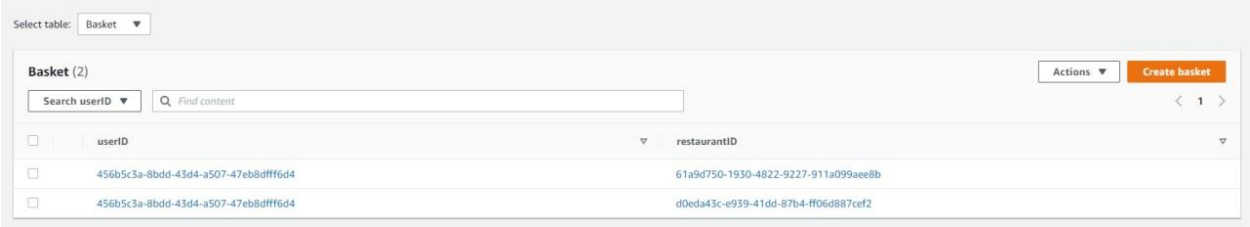

Рисунок 3.15 – Заповнена даними таблиця Basket

Таблиця Basket має зв’язок до багатьох елементів BasketDish, тому маємо:

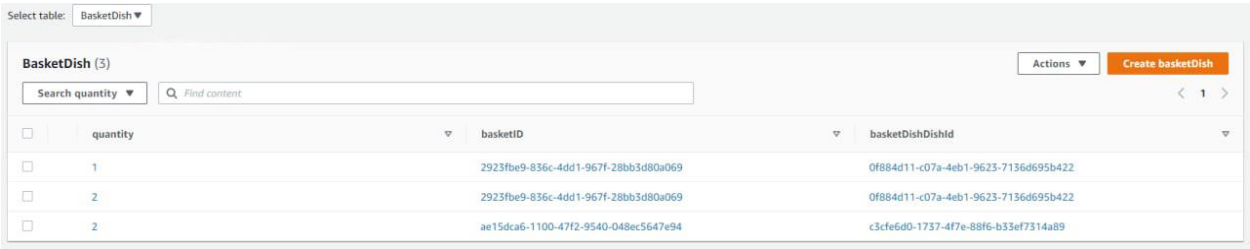

Рисунок 3.16 – Заповнена даними таблиця BasketDish

Після того, як користувач підтвердив створення замовлення, ми маємо дані в таблиці Order.

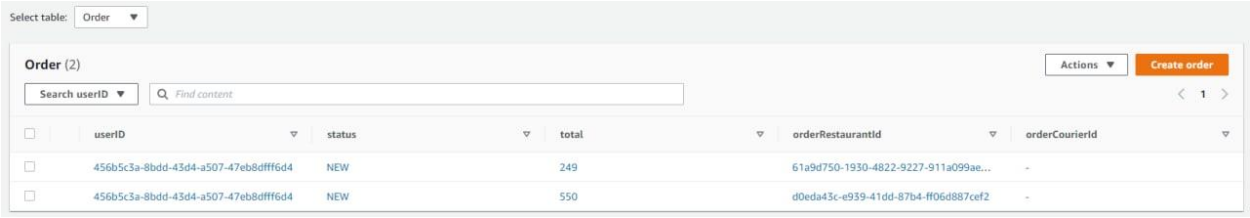

Рисунок 3.17 – Заповнена даними таблиця Order

Таблиця Order має зв’язок до багатьох елементів OrderDish, тому маємо:

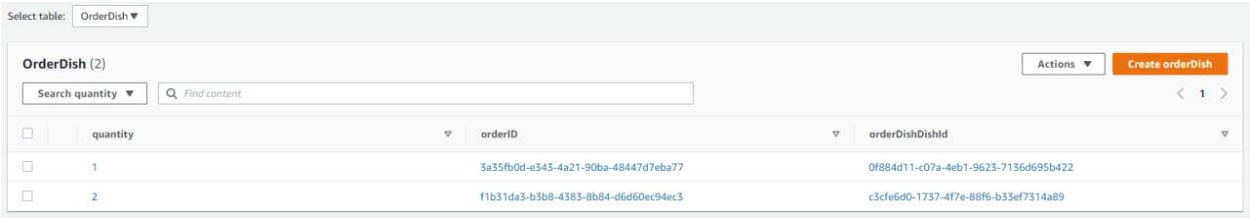

Рисунок 3.18 – Заповнена даними таблиця OrderDish

Таким чином, було продемонстровано роботу з логічного заповнення таблиць системи SumyEats.

3.2 Структура та особливості реалізації алгоритмічного забезпечення

Моделювання – це метод дослідження прототипів реальних і абстрактних об'єктів на звичайних зображеннях, схемах, фізичних об'єктах, що відрізняються від них, але схожі за структурою або типом поведінки, за допомогою аналогії, теорії подібності та теоретико-експериментальної обробки даних [32, 19].

Проектування, у свою чергу, є процесом визначення архітектури, компонентів, інтерфейсів та інших характеристик системи або частини системи. Результатом проектування є елемент – набір моделей, властивостей або характеристик, описаних у формі, придатній для реалізації системи [29].

Архітектура мобільного додатку – це спосіб побудови додатку, абстракція елементів системи на певному етапі їх функціонування [33].

Система може складатися з декількох рівнів абстракції і мати кілька етапів роботи, кожен з яких може мати окрему архітектуру [7].

Отже, використовуючи сайт app.diagrams.net мною були розроблені структури 3х частин мобільного додатку: реєстрацію користувача у системі (є однаковою для обох додатків), застосунки SumyEatsUser та SumyEatsCourier відповідно. Подані структури надають можливість мати попереднє уявлення про функціонування додатків, їх візуальне подання та логічні процеси, що зображено на рисунках 3.19-3.23.

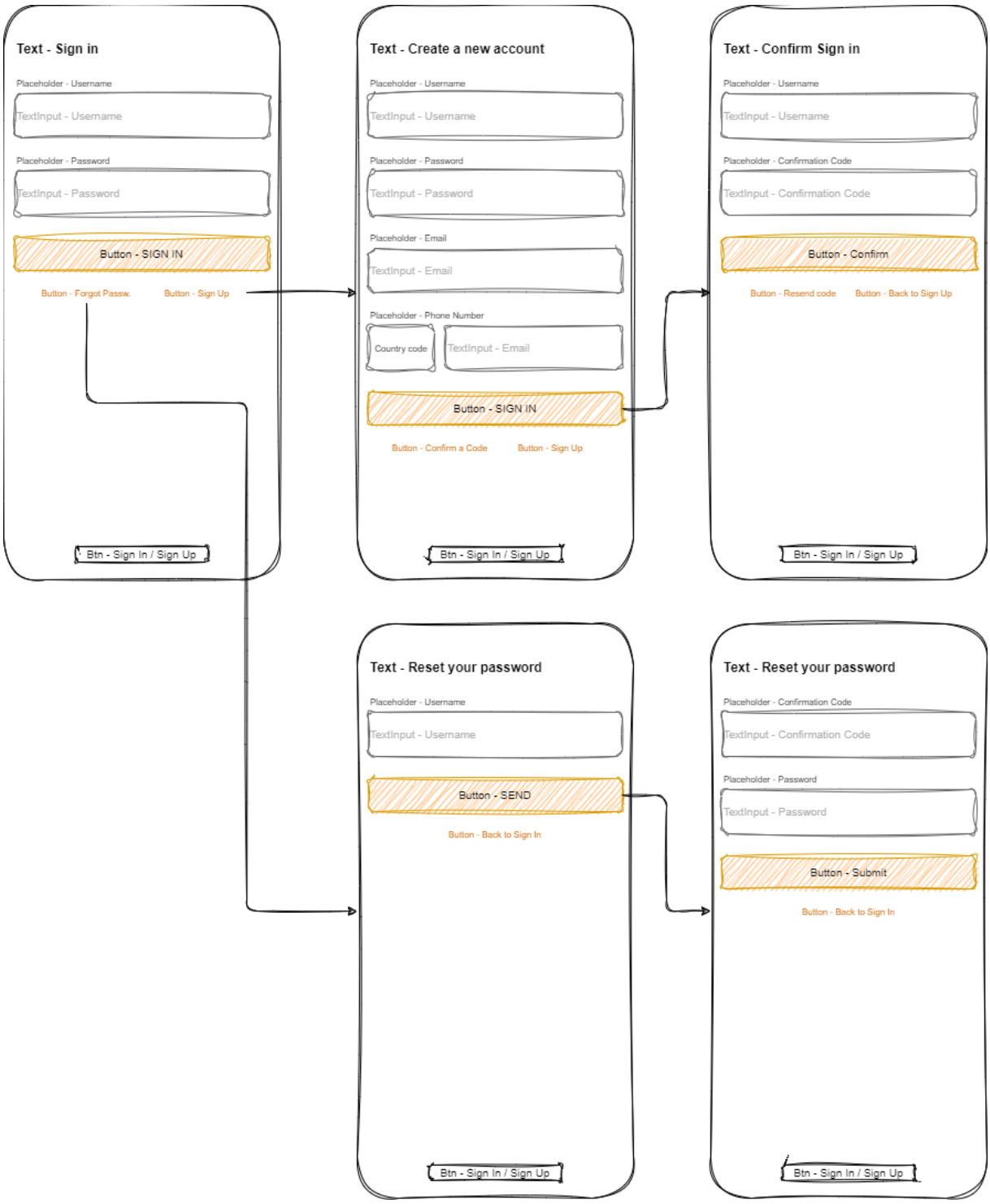

Рисунок 3.19 – Архітектура процесу реєстрації користувача у системі

Так, як подані форми є частиною використання системи AWS Amplify, то абстракція елементів системи була надана окремо.

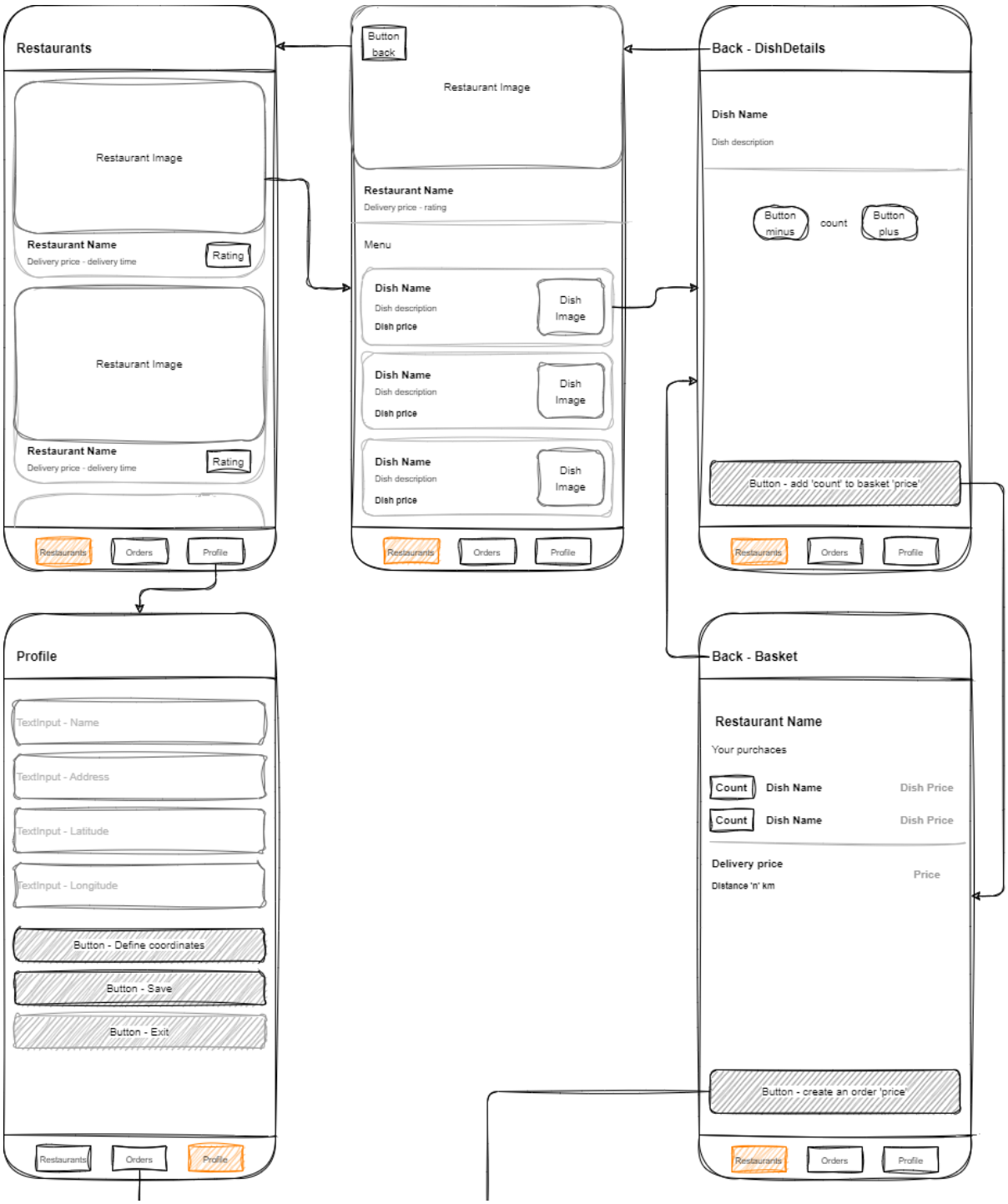

Рисунок 3.20 – Архітектура мобільного додатку SumyEatsUser. 1 частина

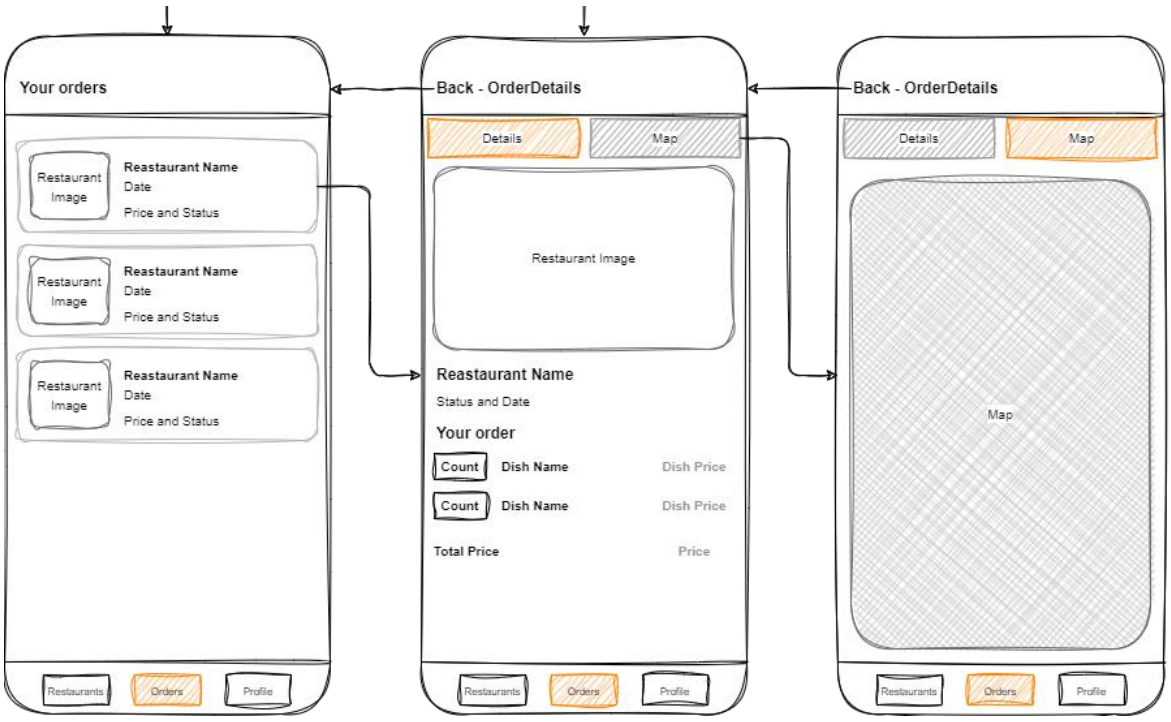

Рисунок 3.21 – Архітектура мобільного додатку SumyEatsUser. 2 частина

Застосунок SumyEatsUser має 3 головних та 5 другорядних сторінок. До головних можна віднести Restaurants, Orders та Profile.

Restaurants має список унікальних значень, при на тисканні на котре відбувається поглиблений процес ознайомлення з обраним значенням (рестораном), що зображено на сторінці RestaurantDetails.

RestaurantDetails містить перелік страв (Dishes) з неповторними ідентифікаторами. При натисканні на певний пункт, відбувається перехід до сторінки DishDetails.

DishDetails відображає коротку інформацію про обрану страву та має логіку про додавання її до кошика (форма Basket) з урахуванням кількості.

Basket містить інформацію про додані страви та вартість і відстань доставки. Створення замовлення відбувається саме з цієї форми.

Наступною головною формою є Orders. Відображає список створених користувачем замовлень з коротким описом. При натисканні на певне з них, відбувається перехід до OrdersDetails.

OrdersDetails – містить 2 форми: Details та Map. Details – має повну інформацію про замовлення, враховуючи список страв та оплату.

Map – відображає статус замовлення та переміщення кур’єра на мапі.

Останньою головною сторінкою є Profile. Має ключову інформацію для здійснення доставки, та логіку встановлення розташування користувача.

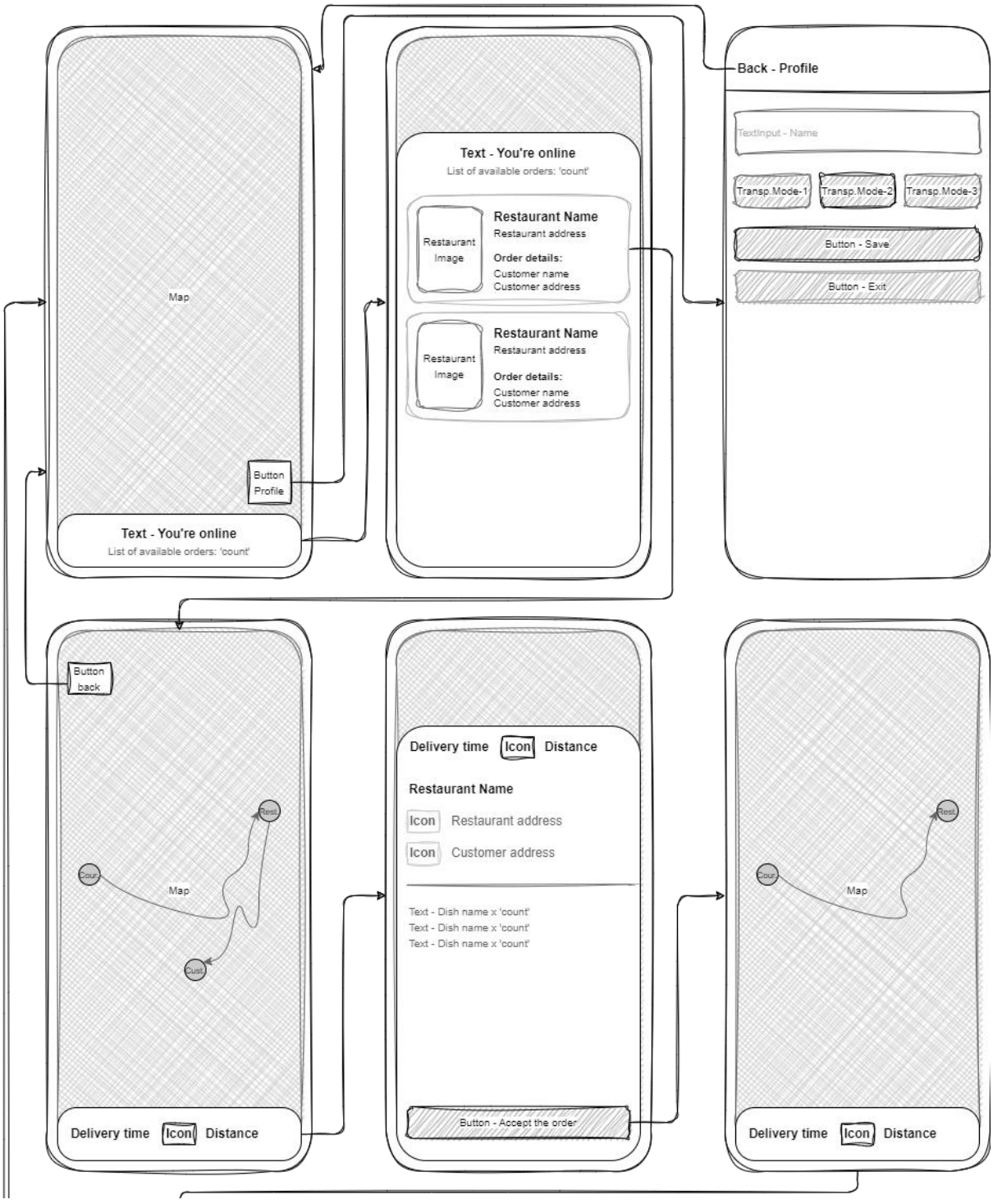

Рисунок 3.22 – Архітектура мобільного додатку SumyEatsCourier. 1 частина

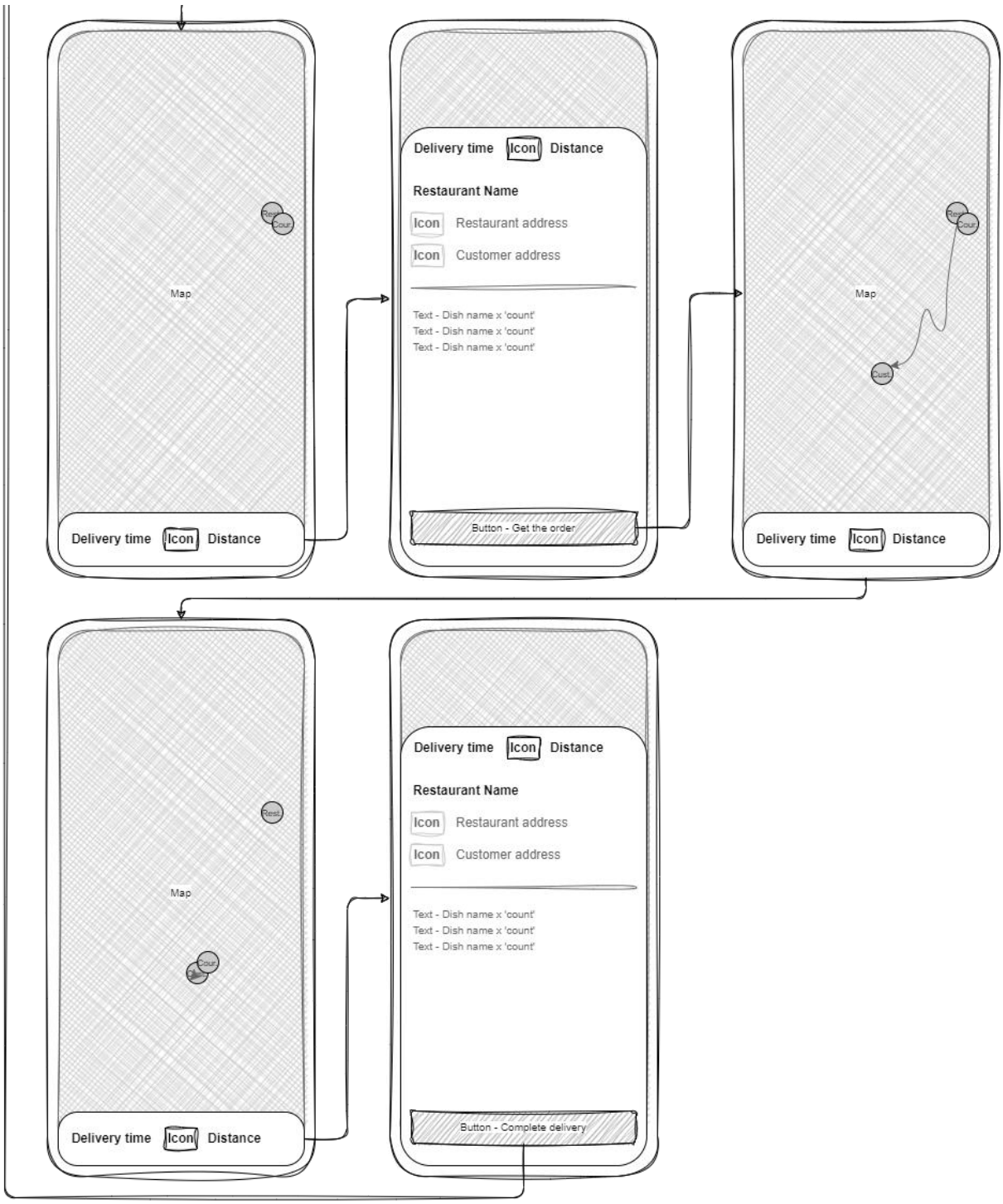

Рисунок 3.23 – Архітектура мобільного додатку SumyEatsCourier. 2 частина

Застосунок SumyEatsCourier містить 2 головні та 1 другорядну сторінки. До головних можна віднести Orders та Profile, до другорядної – OrdersDetails.

Форма Orders має мапу з розташуванням кур’єра, доступні замовлення та список поданих замовлень.

Сторінка Profile відображає ім’я та тип транспорту (впливає на побудову маршрута) кур’єра.

Основний функціонал у даному застосунку відбувається на формі OrdersDetails. Перед прийняттям замовлення сторінка містить кнопку «Назад», котре поверне кур’єра до головної форми, повне відображення маршруту (від кур’єра до закладу харчування, та від ресторана до місцезнаходження замовника), час та відстань, необхідні для створення замовлення, та основну інформацію про страви.

В залежності від статусу та розташування кур’єра, видозмінюється основна кнопка взаємодії з замовленням. На першому етапу вона має назву «Прийняти замовлення», пізніше, коли кур’єр добереться до ресторану, дана кнопка змінить назву на «Отримати замовлення», після чого, коли кур’єр буде поруч з замовником та віддасть замовлення, кнопка буде містити текст «Завершити замовлення».

Разом зі зміною кнопки, змінюється й маршрут на мапі. До того, як кур’єр прийняв замовлення, йому доступний весь шлях. Після цього, на карті буде відображено лише шлях до ресторану. І вже коли кур’єр отримає замовлення, буде показано шлях до замовника.

Також для покращення взаємодії користувачів системою SumyEats у двох прикладних програмах, були розроблені алгоритми дій [37]. Це послідовні команди, котрі необхідно виконувати для досягнення поставленої мети [2].

При першому вході до будь-якого з додатків, користувача необхідно зареєструватися у системі. Таким чином, алгоритм створення аккаунту набуває наступного вигляду (рис. 3.24):

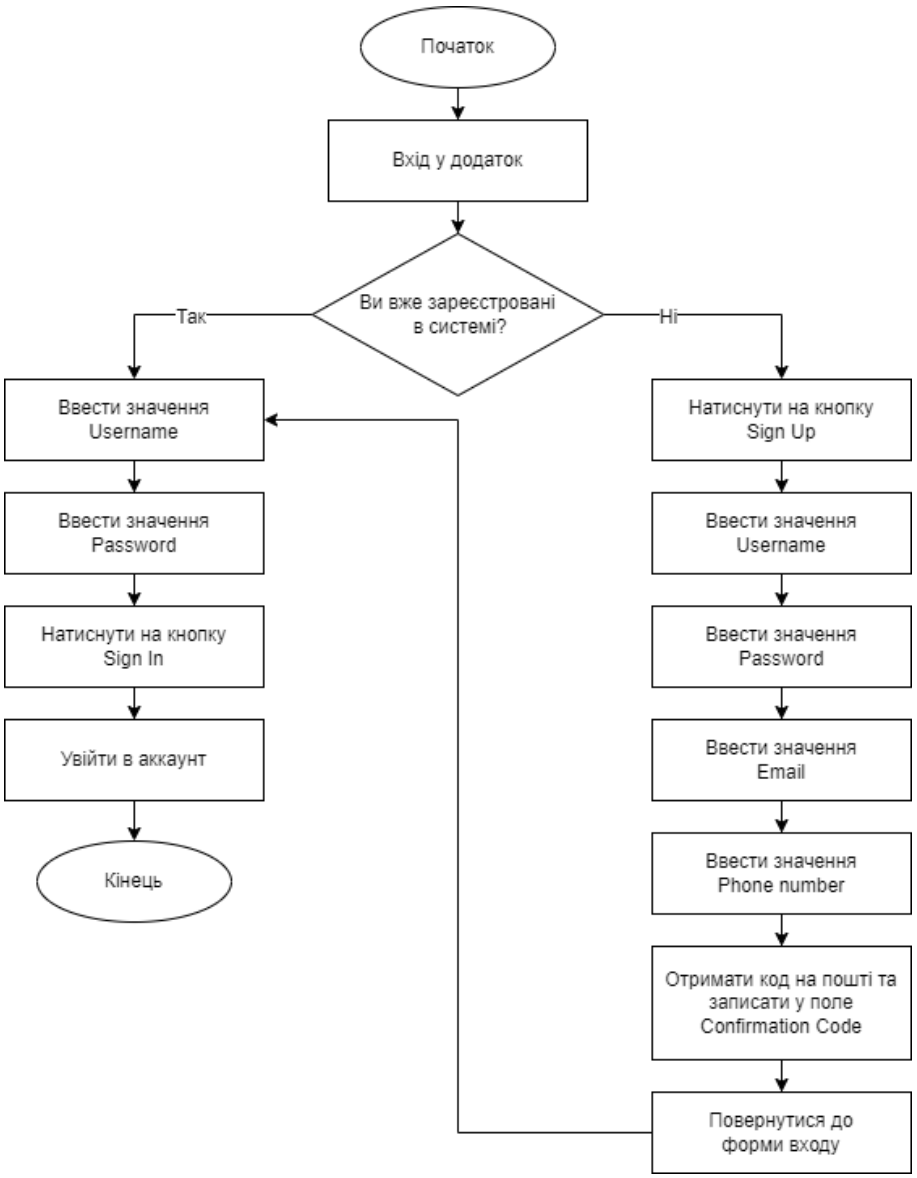

Рисунок 3.24 – Алгоритм створення/входу до аккаунту в системі

Поданий комплекс дій створений для більш швидкого адаптування користувача у системі автоматизованої служби доставки. Він відображає ключові дії для користуванням застосунками, а саме – створення особистого аккаунту на сервері SumyEats, або вхід до нього.

Основне розгалуження регулює наявність аккаунту в системі. При його відсутності, алгоритм декларує лінійну послідовність дій для його створення. Після чого відбувається повернення до основної форми, де є процес аторизації.

Наступним алгоритмом є процес заповнення даних користувачем (рис. 3.25).

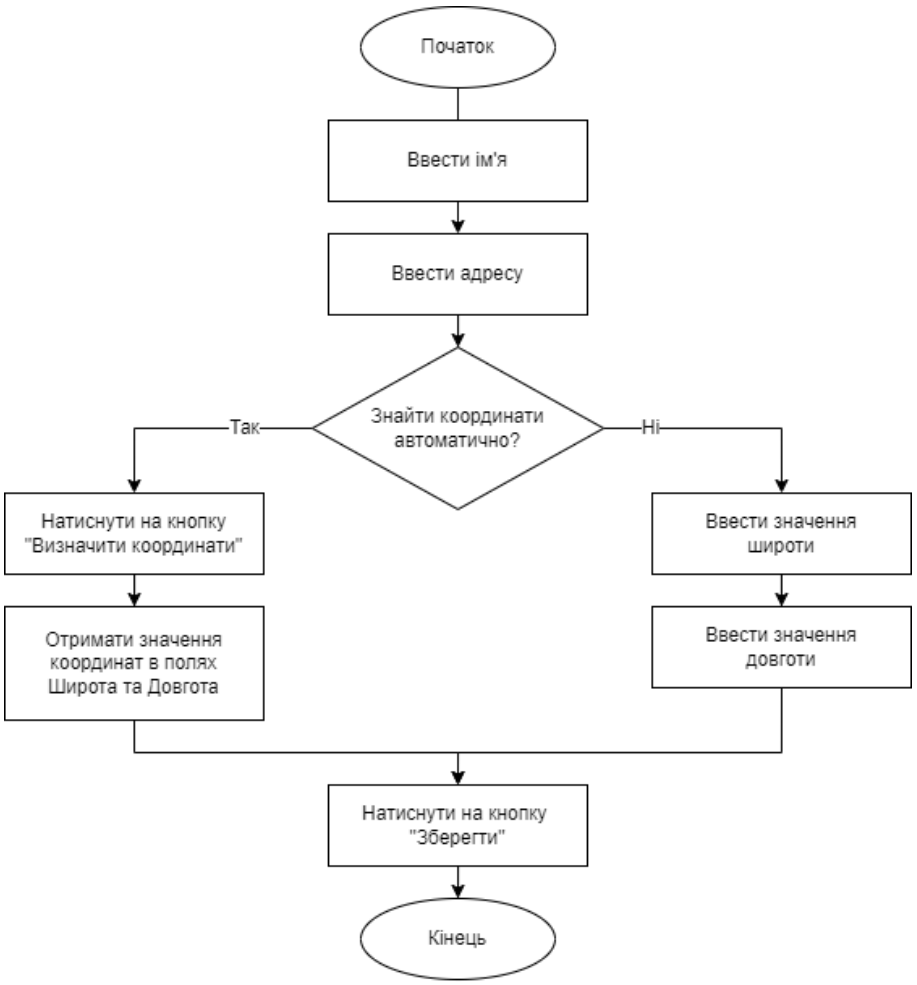

Рисунок 3.25 – Алгоритм внесення даних про користувача

Виконання даного алгоритму забезпечить необхідну інформацію, необхідну для здійснення доставки. Розгалуження дає варіативність для введення координат місця доставки. Це необхідно для реалізації випадку, коли користувач створює замовлення, не знаходячись поруч з місцем призначення.

Розглянемо тепер послідовність дій, необхідних для заповнення даних про кур’єра (рис. 3.26).

Рисунок 3.26 – Алгоритм внесення даних про кур’єра

Даний алгоритм декларує послідовне виконання дій, необхідних для досягнення поставленої мети. Після вказання необхідної інформації, кур’єру буде доступна можливість вибирати та здійснювати замовлення.

Тепер, коли ми маємо зареєстровані 2 типи користувачів, необхідно вказати алгоритм створення замовлення (рис. 3.27).

Наступним етапом для замовника є процес отримання замовлення. Поданий алгоритм дій розписаний на рисунку ?. Він відображає послідовні дії, необхідні для задоволення основної потреби користувача – отримати замовлення від кур’єра (рис. 3.28).

Рисунок 3.27 – Алгоритм створення замовлення

Рисунок 3.28 – Алгоритм отримання замовлення

Знаючи комплекс дій зі сторони клієнта, потрібно ознайомитися з процесом доставки замовлення кур’єром. Це відображено на рисунку 3.29.

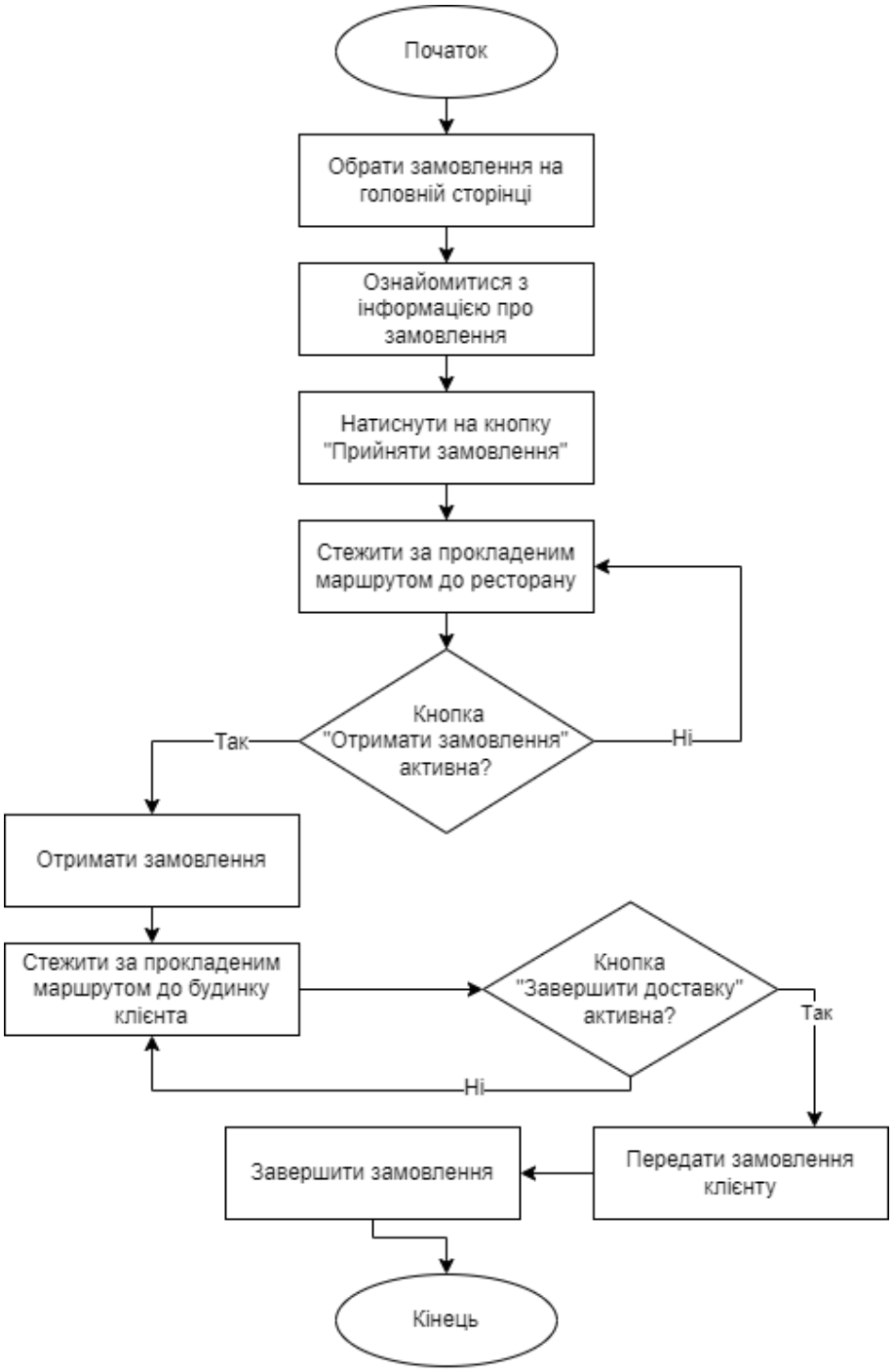

Рисунок 3.29 – Алгоритм доставки замовлення

Подана послідовність дій має основну мету – успішно завершити замовлення кур’єром. Алгоритм має декілька розгалужень, пов’язаних зі статусом замовлення.

3.3 Контрольний приклад та інструкція щодо використання

Додаток SumyEatsUser складається з двох типів форм:

1. Реєстрація в системі

2. Власне сам мобільний додаток

До форм реєстрації у системі відносяться наступні:

1) Форма входу до свого аккаунту «Sign in to your account». Подана форма вимагає ввести Username та Password користувача для подальшої роботи з додатком. За умови першого запуску або ж відсутності аккаунту, необхідно натиснути на кнопку «Sign up» для реєстрації.

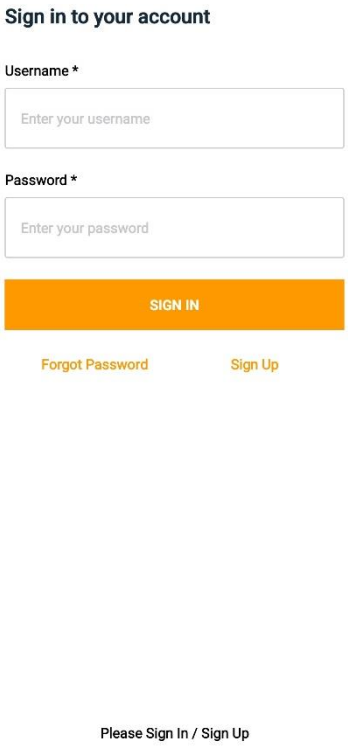

Рисунок 3.30 – Sign in to your account

2) Сторінка «Create a new account». Містить 4 обов’язкові для заповнення поля (Username, Password, Email, Phone Number). Після вказання інформації, відбувається перехід до форми підтвердження аккаунту.

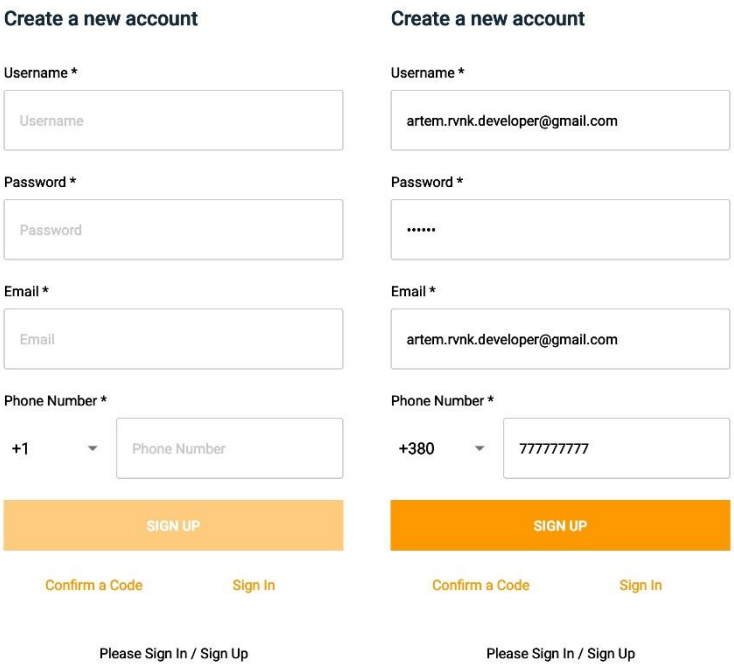

Рисунок 3.31 – Create a new account

3) Форма «Confirm Sign Up». Має 2 обов’язкові поля, серед яких значення Username вже є заповненим, а в поле Confirmation Code необхідно ввести код, котрий прийшов на раніше вказану пошту користувача. Після його введення, відбувається перехід до сторінки «Sign in to your account», де необхідно ввести вже створені дані Username та Password.

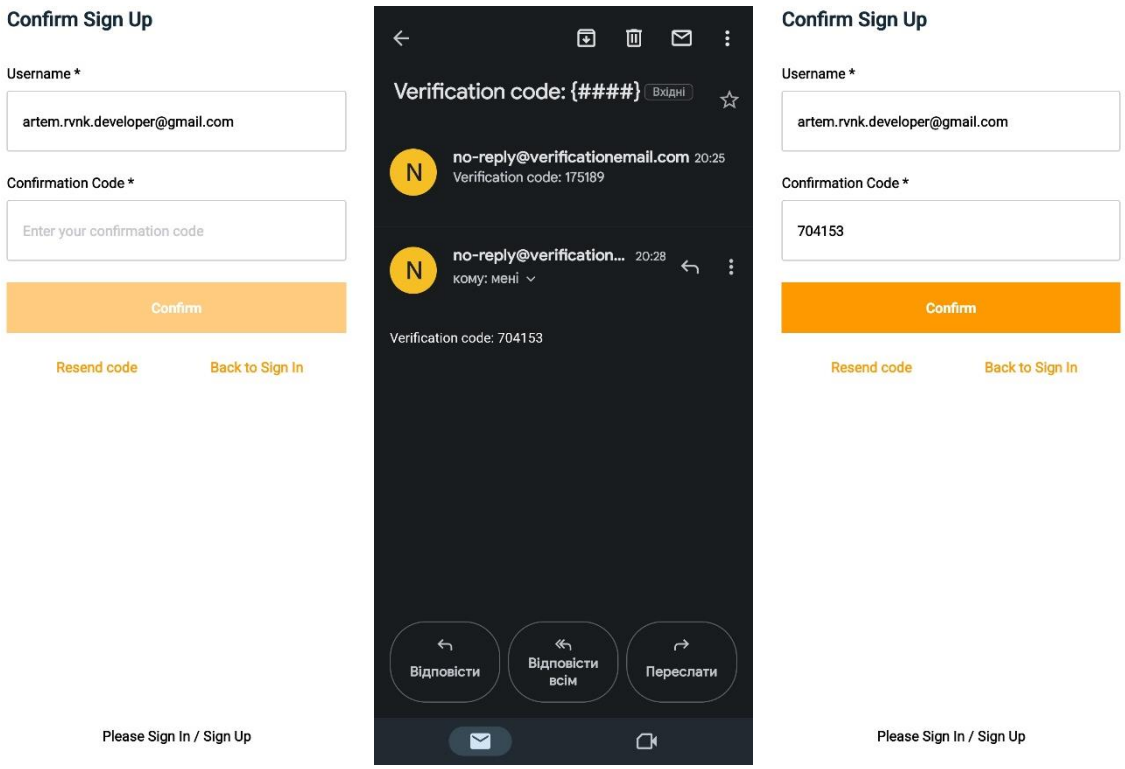

Рисунок 3.32 – Confirm Sign Up

Додаток SumyEatsUser складаються з наступних сторінок:

1) Профіль – є першою формою, з якою зіткнеться користувач під час першого запуску додатку. На поданій сторінці необхідно надати доступ до місцезнаходження пристрою для подальшої роботи з додатком, та ввести значення імені, адреси та координат – широти та довготи. Це можна зробити як самостійно, так й за допомогою кнопки «Визначити координати», котра автоматично заповнить подані поля даними про місцезнаходження користувача.

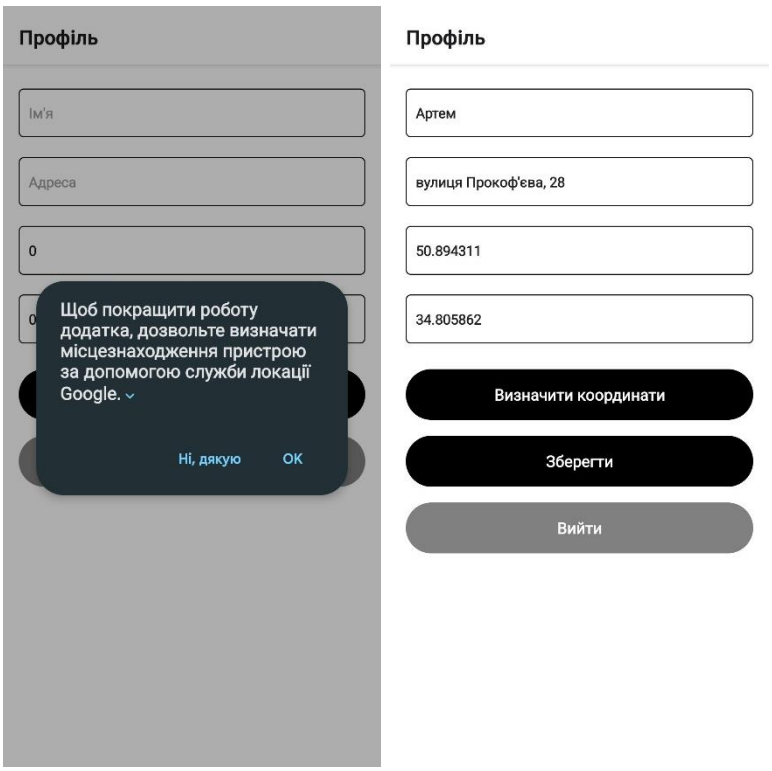

Рисунок 3.33 – Профіль

2) Основною сторінкою, з котрою буде працювати користувач під час створення замовлення – це форма Ресторани. Подана область містить список доступних ресторанів, з короткою інформацією про кожен. Якщо натиснути на певний ресторан, то користувач перейде до наступної сторінки з детальною інформацією про обраний ресторан.

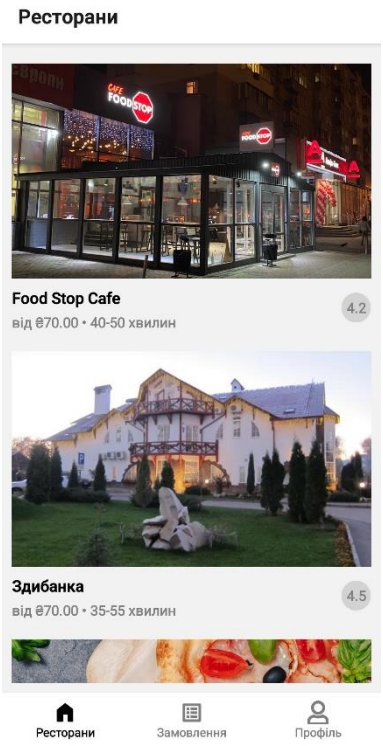

Рисунок 3.34 – Ресторани

3) Сторінка Про ресторан містить коротку інформацію про заклад харчування та меню, яке може замовити користувач. Власне меню складається зі списку страв, котрі містять назву, опис, зображення та вартість. При натисканні на елемент меню, користувач перейде до сторінки Про страву.

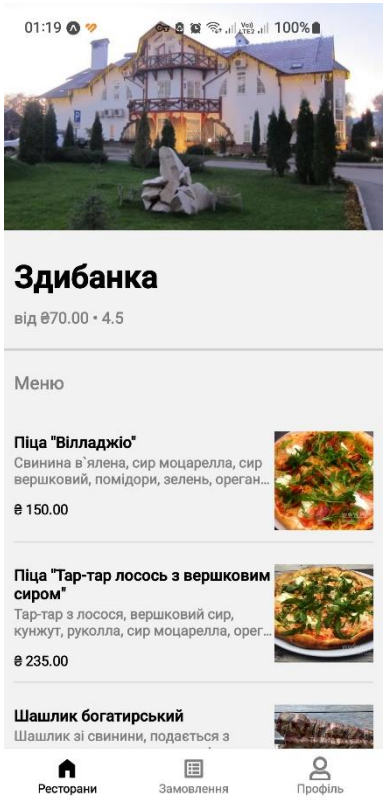

Рисунок 3.35 – Про ресторан

4) Сторінка Про страву надає змогу користувачу додати обрану ним страву до кошика, вказавши її кількість. Кнопка Додати до кошику автоматично відображає кількість страв та їх сумарну вартість. Після натискання на подану кнопку відбувається створення кошику, до якого додається обрана користувачем страва.

Після цього відбувається перехід до сторінки Про ресторани, де буде відображатися кнопка «До кошика» з вказаною кількістю страв. Після натискання на яку користувач побачить нову сторінку Кошик.

Рисунок 3.36 – Про страву

5) Кошик – остання форма перед створенням замовлення. На ній відображається назва ресторану, в якому користувач замовляє страву, список обраних товарів з вказаною кількістю кожного з них та розрахованою вартістю в залежності від кількості, вартість доставки з відображенням відстані, та власне основну кнопку Створити замовлення з підсумованою вартістю. Після оформлення замовлення, користувачу буде представлена сторінка Деталі замовлення.

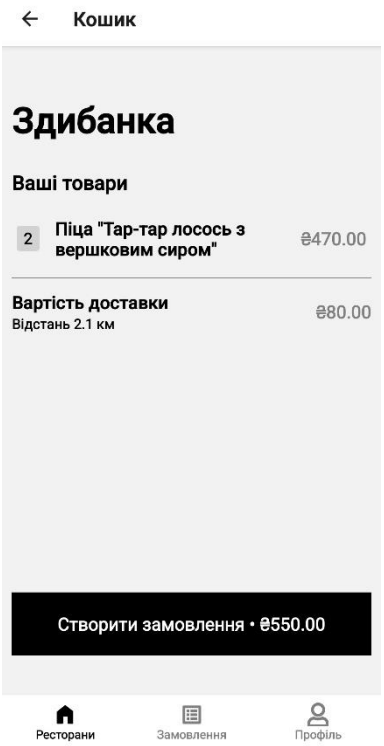

Рисунок 3.37 – Кошик

6) Замовлення – основна форма для взаємодії та контролю зі створеними замовленнями, які доступні у вигляді списку. Після натискання на будь-який пункт, відбудеться перехід до Деталей замовлення.

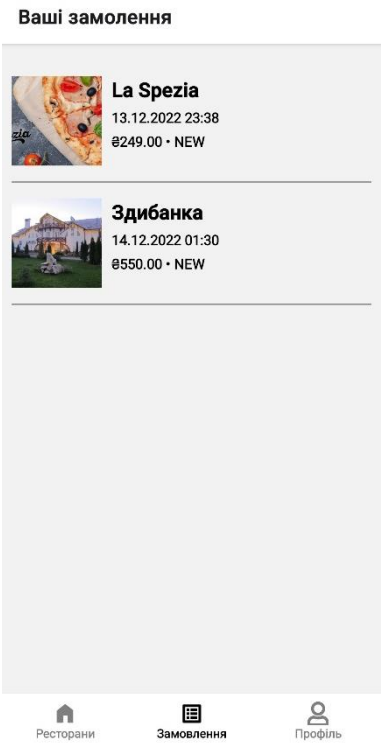

Рисунок 3.38 – Замовлення

7) Деталі замовлення – форма з детальним описом. Містить інформацію про ресторан, про обрану страву, про вартість та про відображення процесу доставки у пункті Карта.

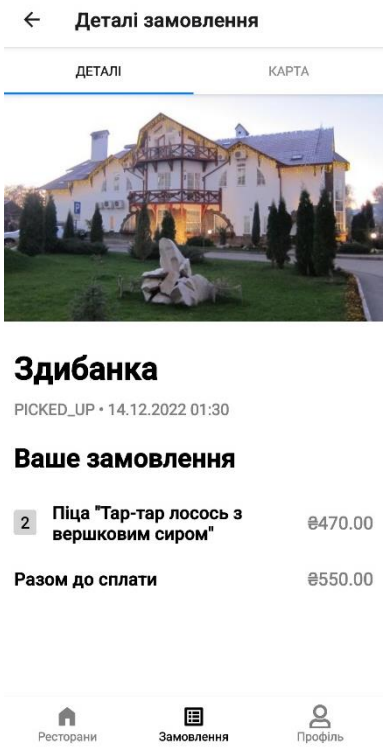

Рисунок 3.39 – Деталі замовлення

Тепер необхідною умовою для продовження процесу доставки, необхідно перейти до додатку SumyEatsCourier, розроблений для робітників служби доставки.

Додаток SumyEatsCourier складається з двох типів форм:

1. Реєстрація у системі – є однаковою з додатком SumyEatsUser, та відображена на рисунках 3.30-3.32.

2. Сторінки мобільного додатку.

Отже, мобільний додаток SumyEatsCourier складаються з наступних сторінок:

1) Профіль – форма, на якій кур’єру необхідно вказати ім’я та обрати вид транспорту на якому він буде здійснювати доставку (у майбутньому це можна змінити).

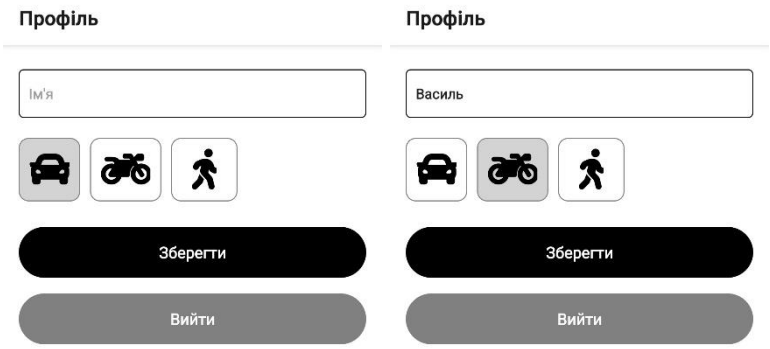

Рисунок 3.40 – Профіль. Кур’єр

Перед переходом на наступну форму, кур’єру необхідно надати доступ для визначення місцезнаходження пристрою для подальшої роботи.

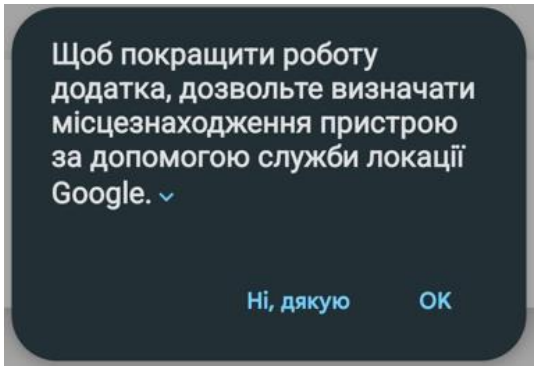

Рисунок 3.41 – Модальне вікно

2) Головна сторінка – основна форма додатку, котра містить карту з відображеній на ній розташуванням кур’єра та ресторанами, кнопкою переходу до профіля та нижнє модальне вікно, котре відображає кількість доступних замовлень та їх перелік.

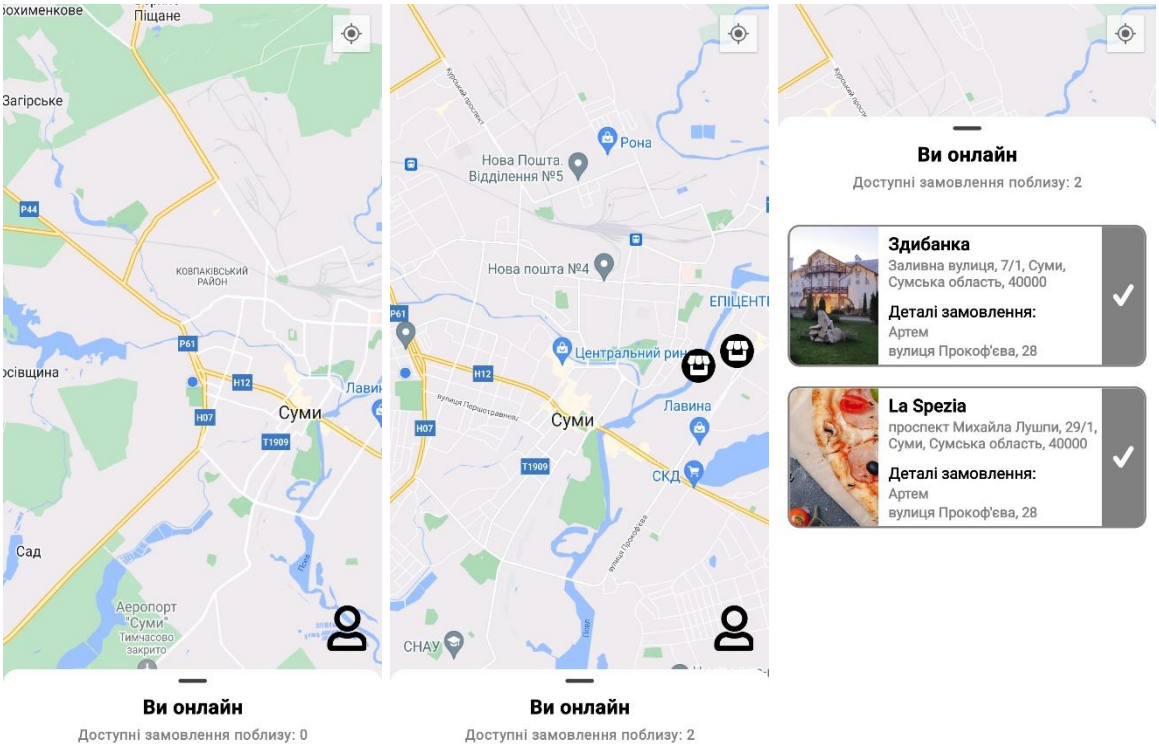

Рисунок 3.42 – Головна сторінка

3) Форма Доставки – сторінка, з відображенням найкращого шляху для виконання замовлення. Поданий маршрут видозмінюється в залежності від обраного кур’єром виду транспортного засобу. На початку подана форма відображає загальний шлях виконання маршруту, його час та відстань.

Нижнє модальне вікно містить інформацію про замовлення: назву ресторану, його адресу, адресу замовника та список замовлених страв. Головним елементом поданої панелі є кнопка взаємодії зі статусом замовлення. Перед прийняттям замовлення вона відображає текст «Прийняти замовлення». Її першою функцією – є фіксування за кур’єром поданого замовлення.

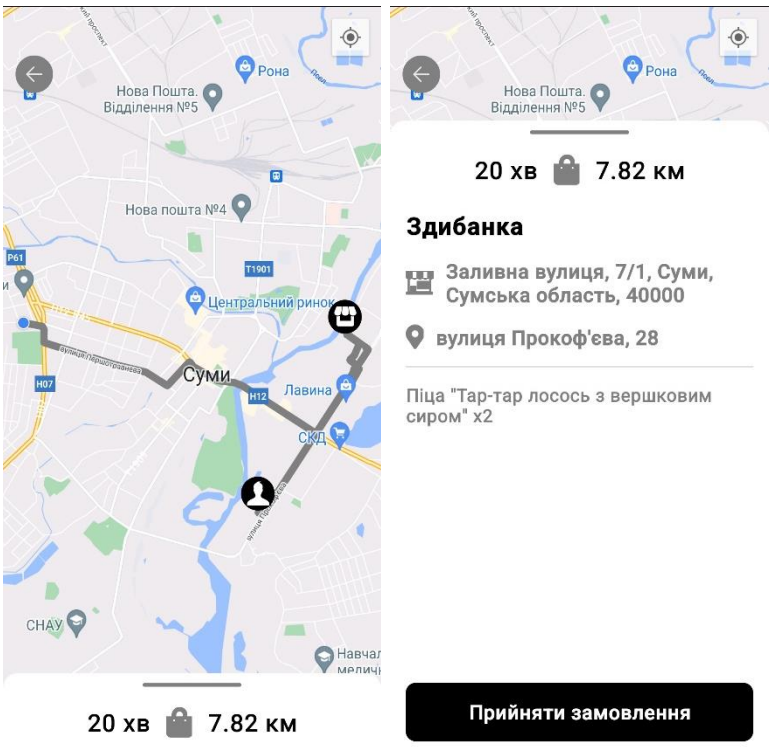

Рисунок 3.43 – Доставка (загальна)

Після прийняття замовлення, маршрут на карті змінюється і відображає лише шлях до ресторану, час, необхідний щоб дістатися до нього та відстань, а власне кнопка взаємодії стане заблокованою, тобто статус змінити не можливо до тих пір, поки коли кур’єр не забере замовлення в ресторані. Поданий процес відображений на рисунках ?-?.

Власне коли кур’єр забере замовлення в ресторані та оновить статус доставки, на карті з’явиться новий маршрут, котрий буде відображати шлях до замовника, очікуваний час подорожі та відстань.

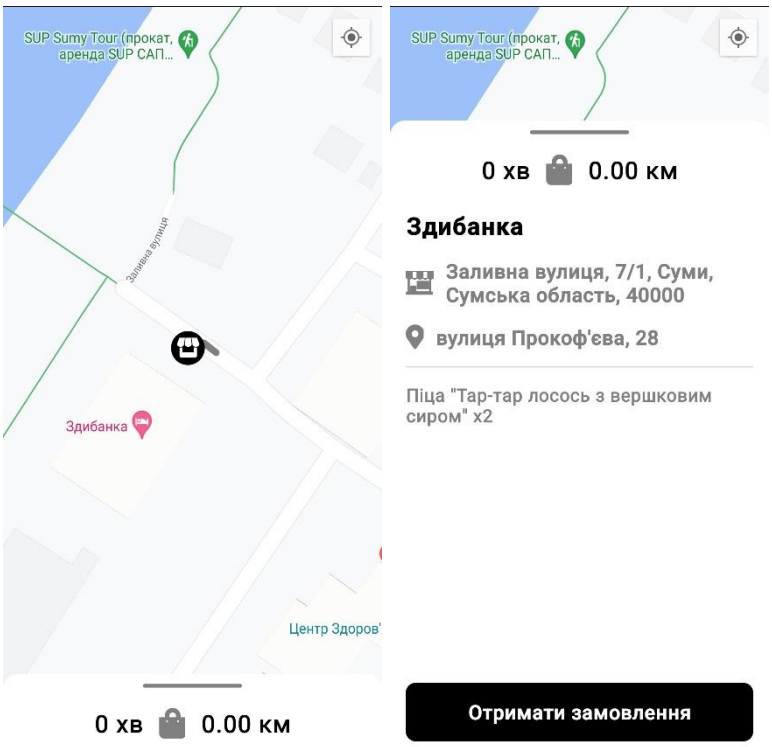

Рисунок 3.44 – Взаємодія з замовленням

Коли кур’єр добереться фінальної точки, йому буде доступна кнопка оновлення статусу.

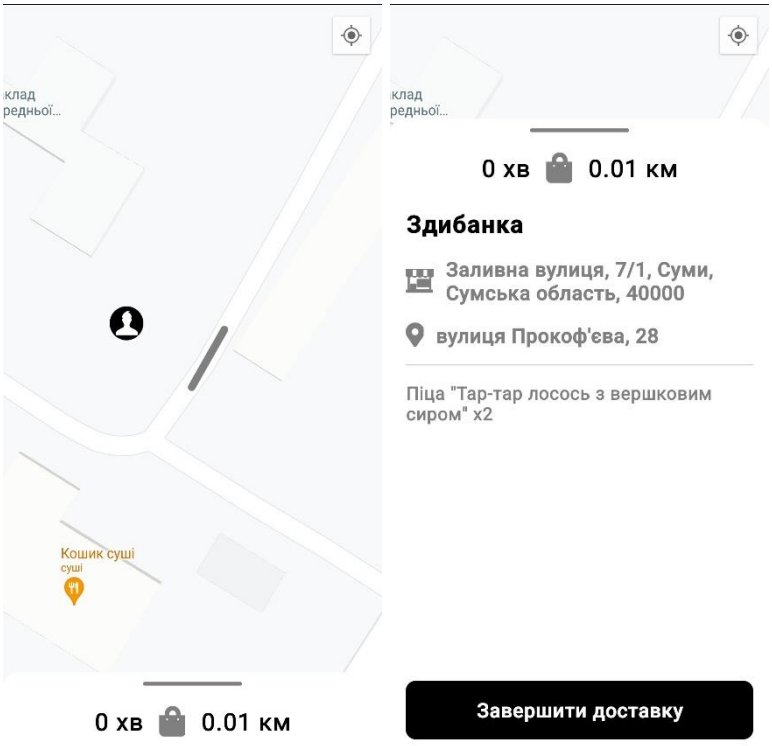

Рисунок 3.45 – Завершення доставки

І тепер, коли замовник отримав свою страву, кур’єр оновлює статус замовлення та закриває його.

А тепер необхідно розглянути процес доставки зі сторони клієнта. Отже, замовнику необхідно перейти до сторінки всіх замовлень та обрати потрібне йому замовлення, після чого натиснути на кнопку Карта.

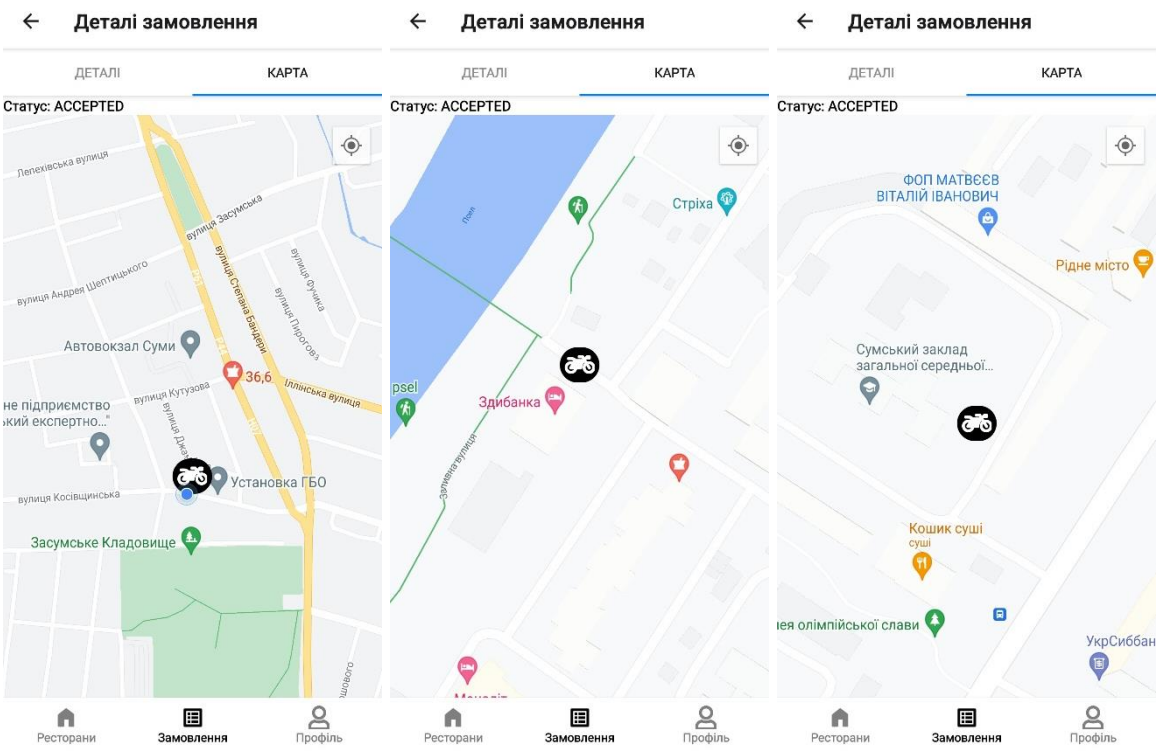

Рисунок 3.46 – Карта з відображенням доставки

На поданій формі користувачу буде доступна інформація щодо доставки його замовлення шляхом відображення статусу та власне переміщення кур’єра.

Після завершення доставки на головній сторінці кур’єра зникне подане замовлення, а на сторінці замовлень користувача буде оновлено статус на завершено (рис. 3.47).

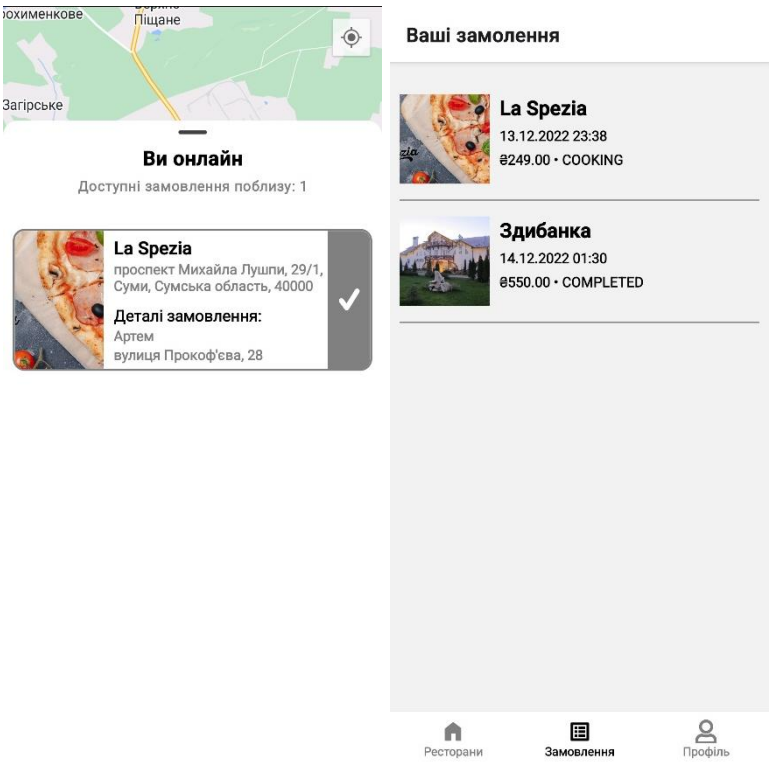

Рисунок 3.47 – Завершений етап доставки

3.4 Оцінювання очікуваного ефекту від впровадження автоматизованої інформаційної системи

Прототип мобільного застосунку служби доставки SumyEats матиме декілька шляхів отримання прибутку.

Першим шляхом є укладення договору з ресторанами про взаємну допомогу. Його основною метою буде виплата фіксованої суми в місяць від закладу харчування в обмін на надання кур’єрів для своєчасної доставки замовлень.

Другим шляхом отримання прибутку є комісія розміром 10% від кожного замовлення, виконаного кур’єром.

Розрахунок оплати кур’єра є наступним: ресторан має фіксовану мінімальну ставку доставки (для прототипу її розмір становить 70 грн). Вона є актуальною для відстані до 2-х км включно. Якщо шлях доставки від закладу харчування до замовника є більшим за 2 км, то відбувається доплата розміром 10 грн за кожний 1 км.

Поданий функціонал відображений на лістингу 1:

Лістинг 1 – Код розрахунку вартості замовлення

```
useEffect(() => {
   if (basket) {
      let distance = getDistance(
         {latitude: restaurant?.lat, longitude: restaurant?.lng},
         {latitude: dbUser?.lat, longitude: dbUser?.lng}
      ) / 1000;
      setDeliveryDistance(distance.toFixed(1));

      if (distance > 2) {
         setDeliveryPrice(Math.ceil((distance - 2)) * 10 + restaurant?.deliveryFee);
      } else {
         setDeliveryPrice(restaurant?.deliveryFee)
      }
   }
}, [basket]);
```

Таким чином, служба доставки SumyEats матиме чистий дохід розміром 10% від кожного виконаного замовлення кур’єром та буде отримувати фіксовану суму в місяць від ресторанів за надані послуги.

ВИСНОВКИ

У ході написання кваліфікаційної магістерської роботи був досліджений процес доставки готових страв та сучасні засоби автоматизації поданого процесу.

Була досягнена основна мета кваліфікаційної магістерської роботи, а саме був розроблений прототип мобільного додатку для автоматизації процесу управління службою доставки готових страв.

Були вирішені основні завдання кваліфікаційної магістерської роботи. А саме було:

– охарактеризовано процес доставки готових страв;

– проаналізовано ринок автоматизованих систем служб доставки;

– сформовано вимоги до системи;

– розроблено модель бізнес-процесів;

– описано архітектуру автоматизованої системи та обрано технології вирішення поставлених завдань;

– визначено функціональну структуру завдання;

– описано структуру та особливості реалізації інформаційного та алгоритмічного забезпечень;

– надано контрольний приклад та інструкцію щодо використання;

– оцінено очікуваний ефект від впровадження системи.

Отже, за рахунок виконання основної мети кваліфікаційної магістерської роботи, можна досягти значних результатів в автоматизації управління службою доставки. Результати дипломної роботи можуть бути використаними ресторанами, кафе та іншими закладами харчування для підвищення ефективності продажів та автоматизації системи управління службою доставки, а також всіма бажаючими жителями міста, котрі мають намір задовольнити власні потреби.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Advantages and disadvantages of Kotlin. Krify. URL: https://krify.co/advantages-and-disadvantages-of-kotlin/ (дата звернення 17.11.2022)

2. Algorithm. TechTarget. URL: https://www.techtarget.com/whatis/definition/algorithm (дата звернення 27.11.2022)

3. Amplify Studio. Amplify Dev Center. URL: https://docs.amplify.aws/console/#visual-backend-builder---works-with-amplify-cli (дата звернення 21.10.2022)

4. Android Studio. Developers. URL: https://developer.android.com/studio/ (дата звернення 15.11.2022)

5. APK file (Android Package Kit file format). TheServerSide. URL: https://www.techtarget.com/whatis/definition/APK-file-Android-Package-Kit-file-format (дата звернення 15.11.2022)

6. Coronavirus disease (COVID-19). World Health Organization. URL: https://www.who.int/health-topics/coronavirus#tab=tab_1 (дата звернення 01.10.2022)

7. Fielding Roy. Architectural Styles and the Design of Network-based Software Architectures. Каліфорнійський університет в Ірвайні. 2000. (дата звернення 25.11.2022)

8. Flutter Pros and Cons 2022 - Summary and Recommendations. LeanCode. URL: https://leancode.co/blog/flutter-pros-and-cons-2022-summary (дата звернення 18.11.2022)

9. Flutter. Build apps for any screen. URL: https://flutter.dev/ (дата звернення 18.11.2022)

10. GitHub. TheServerSide. URL: https://www.techtarget.com/searchitoperations/definition/GitHub (дата звернення 11.11.2022)

11. Glovo. Доставка їжі та багато чого іншого. URL: https://glovoapp.com/ua/uk/ (дата звернення 03.10.2022)

12. History of Food Delivery. Grubtech. URL: https://blog.grubtech.com/history-of-food-delivery (дата звернення 02.10.2022)

13. Integrated development environment (IDE). TechTarget. URL: https://www.techtarget.com/searchsoftwarequality/definition/integrated-development-environment (дата звернення 06.11.2022)

14. IntellJ IDEA. TheServerSide. URL: https://www.theserverside.com/definition/IntellJ-IDEA (дата звернення 07.11.2022)

15. Java. TheServerSide. URL: https://www.theserverside.com/definition/Java (дата звернення 10.11.2022)

16. Madi Connors. What is Xcode and why do I need it? Quora. 2016. URL: https://www.quora.com/What-is-Xcode-and-why-do-I-need-it (дата звернення 04.11.2022)

17. Meet Android Studio. Android Developers. Android Studio. URL: https://developer.android.com/studio/intro#find_sample_code (дата звернення 17.11.2022)

18. Mister.Am. Замовлення і доставка страв у Сумах. URL: https://misteram.com.ua/ (дата звернення 04.10.2022)

19. Modeling. Cambridge English Dictionary. URL: https://dictionary.cambridge.org/dictionary/english/modeling (дата звернення 24.11.2022)

20. React Native Init vs Expo 2022: What Are the Differences? Fulcrum. URL: https://fulcrum.rocks/blog/react-native-init-vs-expo#what-is-react-native-init (дата звернення 19.11.2022)

21. React Native. Learn once, write anywhere. React Native. URL: https://reactnative.dev/ (дата звернення 19.11.2022)

22. Swift. Apple Developer. Apple. URL: https://developer.apple.com/swift/ (дата звернення 25.10.2022)

23. The Good and the Bad of Swift Programming Language. Alexsoft. URL: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language/ (дата звернення 26.10.2022)

24. Thinkwik. React Native: What is it? and, Why is it used? Medium. 2018. URL: https://medium.com/@thinkwik/react-native-what-is-it-and-why-is-it-used-b132c3581df (дата звернення 19.11.2022)

25. Welcome to AWS Amplify Hosting. AWS. URL: https://docs.aws.amazon.com/amplify/latest/userguide/welcome.html (дата звернення 21.10.2022)

26. What is Kotlin? The Java alternative explained. InfoWorld. URL: https://www.infoworld.com/article/3224868/what-is-kotlin-the-java-alternative-explained.html (дата звернення 17.11.2022)

27. Xcode 14 – Apple Developer. Apple. URL: https://developer.apple.com/xcode/ (дата звернення 27.10.2022)

28. Xcode. Mac App Store. URL: https://apps.apple.com/us/app/xcode/id497799835 (дата звернення 25.10.2022)

29. Архітектура та проектування програмного забезпечення. URL: https://learn.ztu.edu.ua/mod/book/view.php?id=278&chapterid=71 (дата звернення 25.11.2022)

30. Доставка їжі: історія та комунікації. Bazilik. URL: https://bazilik.media/dostavka-izhi-istoriia-ta-komunikatsii/ (дата звернення 02.10.2022)

31. Коронавірус: симптоми та профілактика. Клініка MEDIKOM. URL: https://medikom.ua/koronavirus-simptomy-i-profilaktika/ (дата звернення 01.10.2022)

32. Лопатьєв А.О. Моделювання як методологія пізнання. Львівський державний університет фізичної культури, Центр математичного моделювання ІППММ ім. Я. С. Підстригача НАН України. 2007. № 8. URL: https://core.ac.uk/download/pdf/304295518.pdf (дата звернення 24.11.2022)

33. Пройдаков Е. М., Теплицький Л. А. Англо-Український тлумачний словник з обчислювальної техніки, Інтернету і програмування. СофтПрес. С. 552. (дата звернення 25.11.2022)

34. Служба доставки їжі – простий і рентабельний стартап. SPAR URL: https://spar.ua/blogs/sluzhba-dostavki-izhi-prostiy-i-rentabelniy-startap (дата звернення 02.10.2022)

35. Українська «ракета», у яку ніхто не вірив: історія сервісу доставки їжі Rocket. Економічна правда. URL: https://www.epravda.com.ua/publications/2021/04/1/672523/ (дата звернення 04.10.2022)

36. Фреймворк Flutter. Avada-media URL: https://avada-media.ua/ua/services/flutter/ (дата звернення 18.11.2022)

37. Що таке алгоритм? МійКлас. URL: https://miyklas.com.ua/p/informatica/5-klas/algoritmi-ta-programi-python-51129/algoritm-ta-iogo-vlastivosti-48217/re-11a12614-aa85-4655-b188-695b028496ad (дата звернення 27.11.2022)

38. Що таке бізнес-процеси в підприємницькій діяльності? Школа бізнесу НОВА ПОШТА. URL: https://online.novaposhta.education/blog/scho-take-biznes-protsesi-v-pidpriyemnitskij-diyalnosti#648 (дата звернення 20.10.2022)

39. Що таке бізнес-процеси та які їхні основні елементи? IT-Artel. URL: https://it-artel.ua/instruction/chto-takoe-byznes-proczessu-y-kakye-yh-osnovnue-elementu/ (дата звернення 20.10.2022)

40. Що таке локдаун: які обмеження передбачає та умови його запровадження в Україні. УНІАН. URL: https://www.unian.ua/society/lokdaun-shcho-ce-chi-zaprovadyat-v-ukrajini-lokdaun-yaki-obmezhennya-ochikuyut-novini-ukrajini-11235149.html (дата звернення 01.10.2022)

ДОДАТКИ

Додаток А

SUMMURY

Revenko A.V. Development of an automated delivery service management system. Master's qualification work. Sumy State University, Sumy, 2022.

The work explored the peculiarities of the business processes of the delivery service. An analysis of modern automated systems of delivery services was carried out. The basic requirements for the system were given and the choice of development technologies was made. The architecture of the mobile application was designed, and the prototype of the application program was implemented.

Keywords: automation, automated system, information system, automated information system, delivery service, mobile application.

АНОТАЦІЯ

Ревенко А. В. Розробка автоматизованої системи управління службою доставки. Кваліфікаційна магістерська робота. Сумський державний університет, Суми, 2022 р.

У роботі було досліджено особливості бізнес-процесів служби доставки. Було проведено аналіз сучасних автоматизованих систем служб доставки. Були наведені основні вимоги до системи та був здійснений вибір технологій розробки. Було спроектовано архітектуру мобільного додатку та реалізовано прототип прикладної програми.

Ключові слова: автоматизація, автоматизована система, інформаційна система, автоматизована інформаційна система, служба доставки, мобільний додаток, мобільний застосунок.

Додаток Б

Лістинг Б.1 – Код таблиці Restaurant

```
type Restaurant @model @auth(rules: [{allow: public}]) {
  id: ID!
  name: String!
  image: String!
  deliveryFee: Float!
  minDeliveryTime: Int!
  maxDeliveryTime: Int!
  rating: Float
  address: String!
  lat: Float!
  lng: Float!
  Baskets: [Basket] @hasMany(indexName: "byRestaurant", fields: ["id"])
  Dishes: [Dish] @hasMany(indexName: "byRestaurant", fields: ["id"])
}
```

Лістинг Б.2 – Код таблиці Dish

```
type Dish @model @auth(rules: [{allow: public}]) {
  id: ID!
  name: String!
  image: String
  description: String
  price: Float!
  restaurantID: ID! @index(name: "byRestaurant")
}
```

Лістинг Б.3 – Код таблиці Basket

```
type Basket @model @auth(rules: [{allow: public}]) {
  id: ID!
  BasketDishes: [BasketDish] @hasMany(indexName: "byBasket", fields: ["id"])
  userID: ID! @index(name: "byUser")
  restaurantID: ID! @index(name: "byRestaurant")
}
```

Лістинг Б.4 – Код таблиці BasketDish

```
type BasketDish @model @auth(rules: [{allow: public}]) {
  id: ID!
  quantity: Int!
  Dish: Dish @hasOne
  basketID: ID! @index(name: "byBasket")
}
```

Лістинг Б.5 – Код таблиці User

```
type User @model @auth(rules: [{allow: public}]) {
  id: ID!
  name: String!
  address: String!
  lat: Float!
```

```
  lng: Float!
  Orders: [Order] @hasMany(indexName: "byUser", fields: ["id"])
  Baskets: [Basket] @hasMany(indexName: "byUser", fields: ["id"])
  sub: String!
}
```

Лістинг Б.6 – Код таблиці Courier

```
type Courier @model @auth(rules: [{allow: public}]) {
  id: ID!
  name: String
  sub: String!
  lat: Float
  lng: Float
  transportationMode: TransportationModes
}
```

Лістинг Б.7 – Код таблиці TransportationModes

```
enum TransportationModes {
  DRIVING
  BICYCLING
  WALKING
}
```

Лістинг Б.8 – Код таблиці Order

```
type Order @model @auth(rules: [{allow: public}]) {
  id: ID!
  userID: ID! @index(name: "byUser")
  status: OrderStatus!
  OrderDishes: [OrderDish] @hasMany(indexName: "byOrder", fields: ["id"])
  total: Float!
  Restaurant: Restaurant @hasOne
  Courier: Courier @hasOne
}
```

Лістинг Б.9 – Код таблиці OrderDish

```
type OrderDish @model @auth(rules: [{allow: public}]) {
  id: ID!
  quantity: Int!
  orderID: ID! @index(name: "byOrder")
  Dish: Dish @hasOne
}
```

Лістинг 3.10 – Код таблиці OrderStatus

```
enum OrderStatus {
  NEW
  COOKING
  READY_FOR_PICKUP
  PICKED_UP
  COMPLETED
  ACCEPTED
}
```