МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки,

освітньо-професійної програми «Інформаційні технології проектування» на тему: «Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку запасів»

Здобувача групи ІТ-03-2 Лиховида Максима Романовича .
(шифр групи) (прізвище, ім'я, по батькові)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Максим ЛИХОВИД
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник доцент кафедри ІТ, к.т.н., доцент Юлія ПАРФЕНЕНКО
_____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«_____» _ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Лиховид Максим Романович

1 Тема роботи *Додаток «*Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку запасів*»*

керівник роботи *Парфененко Юлія Вікторівна, к.т.н., доцент*,

затверджені наказом по університету від «*07*» травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.

3 Вхідні дані до роботи перелік вимог по розробці вебдодатку «Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку запасів», графічні матеріали для наповнення додатку

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) аналіз предметної області, моделювання та проектування додатку, програмна реалізація вебдодатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) актуальність, мета, задачі, функціональні вимоги, аналіз аналогів, порівняльна таблиця аналогів, контекстна діаграма ведення щоденника IDEF0, діаграма варіантів використання, приклади документів бази даних, діаграма послідовностей, засоби реалізації, демонстрація вебдодатку, висновки, апробація.

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Аналіз предметної області	11.04.2024 – 12.04.2024	виконано
	Планування робіт	13.04.2024 - 18.04.2024	виконано
	Аналіз продуктів-аналогів	19.04.2024 - 20.04.2024	виконано
	Складання технічних завдань	21.04.2024 - 23.04.2024	виконано
	Модельована та проектування вебдодатку	25.04.2024 - 01.05.2024	виконано
	Розробка вебдодатку	02.05.2024 - 21.05.2024	виконано
	Тестування вебдодатку	22.05.2024	виконано
	Оформлення пояснювальної записки	23.05.2024 - 25.05.2024	виконано

Студент _____ Максим ЛИХОВИД

(підпис)

Керівник роботи _____ Юлія ПАРФЕНЕНКО

(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра «Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку запасів».

Пояснювальна записка складається зі вступу, 3 розділів, висновків, списку використаних джерел із 11 найменувань, додатків. Загальний обсяг роботи − 123 сторінка, у тому числі 74 сторінки основного тексту, 1 сторінки списку використаних джерел, 49 сторінкок додатків.

Актуальність цієї кваліфікаційної роботи випливає з постійно зростаючих вимог до оптимізації управління запасами в ресторанному бізнесі. Сучасний ринок диктує потребу в автоматизації процесів, що дозволяє зменшити втрати, оптимізувати закупівлі і підвищити загальну ефективність діяльності. В цьому контексті, розробка мобільного додатку з підсистемою обліку запасів є критично важливою для підтримки ведення ресторанного бізнесу, забезпечення точного обліку і швидкого доступу до необхідної інформації.

Мета роботи полягає у розробці та впровадженні мобільного додатку, який дозволяє ресторанам ефективно управляти запасами та замовленнями, оптимізувати робочі процеси, зменшити витрати та покращити якість обслуговування клієнтів. Додаток має включати інтуїтивно зрозумілий інтерфейс, широкий спектр функціональності для обліку товарів, автоматизації замовлень і аналітики даних.

Ключові слова: мобільний додаток, підсистема обліку запасів, оптимізація робочих процесів, react native.

ЗМІСТ

ВСТУП

У сучасному світі ресторанний бізнес динамічно розвивається і потребує ефективних інструментів управління. Мобільні додатки стають все більш популярними завдяки своїй зручності та доступності, що робить їх перспективним рішенням для підтримки ведення ресторанного бізнесу. Підсистема обліку запасів є однією з ключових компонентів будь-якого ресторану, адже вона дозволяє відстежувати наявність продуктів харчування, оптимізувати закупівлі та мінімізувати втрати.

Впровадження мобільного додатку з підсистемою обліку запасів може значно покращити управління рестораном та призвести до численних переваг для бізнесу. Зниження витрат стане можливим завдяки точному контролю за наявністю продуктів і оптимізації закупівель, що сприятиме підвищенню рентабельності. Автоматизація процесів обліку та звітності підвищить ефективність управління, дозволяючи працівникам зосередитися на стратегічних завданнях. Швидке та точне прийняття замовлень завдяки мобільному додатку покращить обслуговування клієнтів і забезпечить їхнє задоволення [1].

Ця тема дослідження актуальна через зростання популярності мобільних додатків, які стають важливим інструментом для замовлення їжі та ведення бізнесу у ресторанній галузі. Конкуренція на ринку ресторанного обслуговування підтримує необхідність вдосконалення операцій та оптимізації управління. Недолік комплексних мобільних рішень для обліку запасів у ресторанах створює простір для дослідження та розробки ефективних інструментів управління запасами. Такий підхід дозволить ресторанам підвищити свою конкурентоспроможність, ефективно використовувати ресурси та задовольняти потреби клієнтів, що в кінцевому результаті призведе до збільшення прибутку і успішного розвитку бізнесу.

Метою даної роботи є розробка підсистеми обліку запасів мобільного додатку підтримки ведення ресторанного бізнесу.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- Провести аналіз останніх досліджень і публікацій у сфері мобільних додатків для ресторанного бізнесу.
- Проаналізувати наявні програмні продукти-аналоги.
- Розробити концепцію мобільного додатку, який буде включати підсистему обліку запасів.
- Спроектувати та реалізувати підсистему обліку запасів мобільного додатку підтримки ведення ресторанного бізнесу.
- Провести тестування мобільного додатку та оцінити його ефективність.

Результати цього дослідження мають великий потенціал для практичного використання у сфері ресторанного бізнесу. Вони можуть бути застосовані для розробки та впровадження мобільних додатків, спрямованих на поліпшення управління та ефективності ресторанного бізнесу. Ці результати дозволять знизити витрати, покращити обслуговування клієнтів та збільшити прибуток ресторанів.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

У сучасному конкурентному середовищі ресторанного бізнесу ефективне управління запасами є ключовим для досягнення успіху та збереження конкурентної переваги. Мобільні додатки стають необхідним інструментом для ресторанів у цьому процесі, надаючи зручні та інноваційні інструменти для обліку, оптимізації та аналізу запасів.

Ефективне управління запасами в ресторанному бізнесі включає точний облік запасів, своєчасне замовлення необхідних інгредієнтів, оптимізацію процесу інвентаризації та підвищення ефективності ланцюжка постачання. Це допомагає ресторанам уникати недостачі товарів, зменшує витрати на зберігання запасів та підвищує задоволеність клієнтів через постійну наявність улюблених страв [2].

Останні дослідження підтверджують, що мобільні додатки можуть значно полегшити управління запасами у ресторанах. Деякі з ключових можливостей мобільних додатків у цій сфері включають:

Інтеграція з системами точок продажу (POS): Це дозволяє відстежувати використання запасів у режимі реального часу, що сприяє точному обліку запасів та своєчасному замовленню інгредієнтів.

Технології сканування штрих-кодів: Вони спрощують процес інвентаризації, швидко та точно скануючи продукти і зменшуючи час та ресурси, необхідні для цього процесу [3].

Розширені функції додатків: Сучасні мобільні додатки для управління запасами мають розширені функції, такі як моніторинг запасів, створення замовлень, відстеження поставок та аналітика даних, що сприяє покращенню ефективності управління та зниженню витрат.

Останні дослідження та публікації підтверджують важливість інноваційних технологій у цій галузі:

Прогнозування попиту за допомогою штучного інтелекту: Використання штучного інтелекту дозволяє ресторанам прогнозувати попит на страви, що допомагає оптимізувати запаси та зменшує псування продуктів.

Інтеграція з системами POS: Це покращує ефективність ланцюжка постачання та дозволяє знижувати витрати на управління запасами.

Використання технологій сканування штрих-кодів: Це підвищує точність інвентаризації та економізує час у процесі управління запасами.

Загалом, мобільні додатки для управління запасами стають незамінним інструментом для ресторанів, які прагнуть оптимізувати свій бізнес, підвищити конкурентоспроможність та задовольнити потреби клієнтів у високоякісному обслуговуванні. Їх інтеграція допомагає забезпечити ефективне управління запасами та підвищити ефективність ресторанного бізнесу. Наукові публікації підтверджують важливість та потенціал мобільних додатків для управління запасами в ресторанному бізнесі. Ось деякі з останніх досліджень:

- Прогнозування попиту за допомогою штучного інтелекту: Дослідження досліджують, як штучний інтелект може використовуватися для прогнозування попиту на страви в ресторанах, що допомагає їм оптимізувати запаси та мінімізувати псування продуктів [9].
- Інтеграція з системами POS: Дослідження вивчають переваги інтеграції мобільних додатків для управління запасами з системами POS для покращення ефективності ланцюжка постачання та зниження витрат [10].
- Використання технологій сканування штрих-кодів: Дослідження підкреслюють важливість технологій сканування штрих-кодів для підвищення точності інвентаризації та економії часу [11].

Мобільні додатки для управління запасами стають незамінним інструментом для ресторанів, які прагнуть оптимізувати свій бізнес, підвищити конкурентоспроможність та покращити задоволеність клієнтів. Інтеграція цих інноваційних технологій.

1.2 Аналіз програмних продуктів - аналогів

Підсистема обліку запасів є важливою частиною мобільного додатку підтримки ведення ресторанного бізнесу. Вона дозволяє ресторанам відстежувати свої запаси продуктів харчування, напоїв та інших товарів, а також допомагає їм оптимізувати закупівлі та мінімізувати втрати [4].

Apicurio: Apicurio - це платформа управління даними API, яка пропонує функції обліку запасів, такі як відстеження запасів, управління замовленнями та оптимізація закупівель. Apicurio добре підходить для ресторанів, які хочуть інтегрувати свою підсистему обліку запасів з іншими системами.

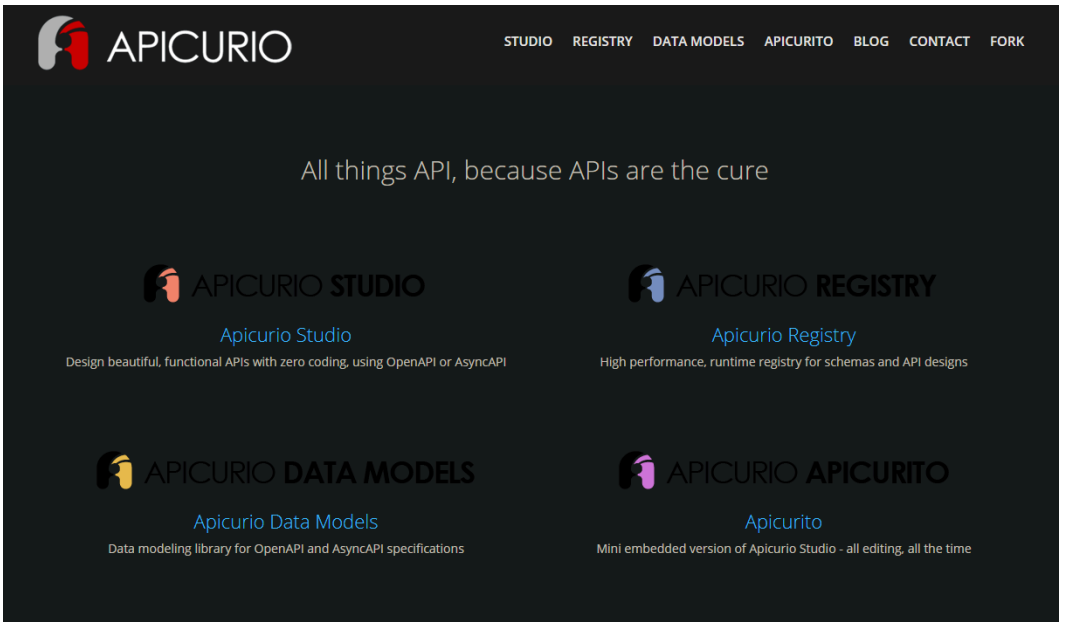

Рисунок 1.1 Apicurio

Chef's Plate: Chef's Plate - це програмне забезпечення для планування меню та замовлення продуктів харчування, яке пропонує функції обліку запасів, такі як відстеження запасів, створення списків покупок та управління рецептами. Chef's Plate добре підходить для ресторанів, які хочуть оптимізувати свої закупівлі та мінімізувати втрати.

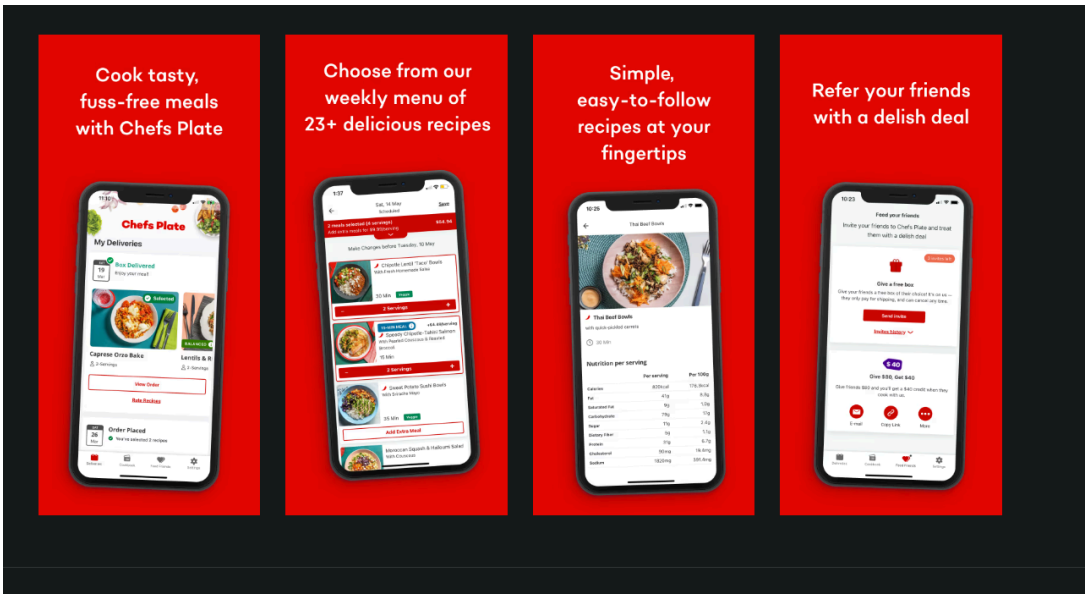

Рисунок 1.2 Chef's Plate

Restaurant Inventory: Restaurant Inventory - це просте у використанні програмне забезпечення для обліку запасів, яке пропонує функції, такі як відстеження запасів, створення звітів та управління замовленнями. Restaurant Inventory добре підходить для малих та середніх ресторанів, які шукають доступне рішення для обліку запасів.

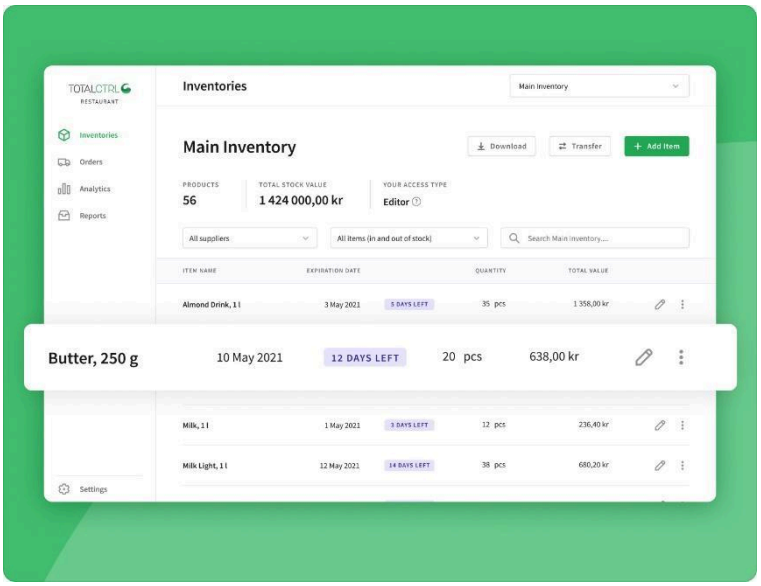

Рисунок 1.3 Chef's Plate

Функція	Apicurio	Chef's Plate	Restaurant Inventory
Ціна	За запитом	Від 99$ на місяць	Від 29$ на місяць
Облік запасів	Так	Так	Так
Відстеження витрат	Так	Так	Так
Створення звітів	Так	Так	Так
Управління рецептами	Ні	Так	Ні
Інтеграція з постачальниками	Так	Так	Ні
Мобільний додаток	Так	Так	Ні
Підтримка клієнтів	24/7	24/7	24/7

1.3 Мета та задачі дослідження

Метою роботи є розробка та впровадження мобільного додатку для підтримки ведення ресторанного бізнесу, який включає підсистему обліку запасів з урахуванням підсистеми замовлень.

Для розроблення мобільного додатку першим етапом є визначення потреб та вимог користувачів, проведення опитувань та інтерв'ю з власниками ресторанів, менеджерами та персоналом, щоб визначити їхні потреби та вимоги щодо мобільного додатку для підтримки ведення ресторанного бізнесу.

Далі необхідно виконати проектування мобільного додатку вцілому та підсистеми обліку запасів, створення прототипів та інтерфейсу користувача мобільного додатку, який включає підсистему обліку запасів та підсистему замовлень з урахуванням потреб та вимог користувачів [5].

Наступним етапом є розробка та впровадження підсистеми обліку запасів, створення та інтеграція підсистеми обліку запасів у мобільний додаток, яка дозволяє відстежувати рівень запасів, автоматично генерувати замовлення на поповнення запасів, оптимізувати закупівлі та мінімізувати втрати. Інтеграція підсистеми обліку запасів з підсистемою замовлень, створення зв'язку між підсистемою обліку запасів та підсистемою замовлень дозволить гарантувати, що замовлення на поповнення запасів генеруються на основі фактичного рівня запасів та прогнозованого попиту.

Після розроблення мобільного додатку слід виконати тестування та оцінювання мобільного додатку, проведення тестування мобільного додатку з користувачами для виявлення помилок, збоїв та покращення загального користувацького досвіду.

Заключними етапами є впровадження та популяризація мобільного додатку, розробка стратегії впровадження та популяризації мобільного додатку серед власників ресторанів, менеджерів та персоналу.

Очікувані результати впровадження мобільного додатку для ресторанного бізнесу з підсистемою обліку запасів та замовлень є значущими з практичної точки зору. Цей додаток принесе численні переваги для ресторанів та їх клієнтів.

Перш за все, розробка та впровадження такого додатку дозволить підвищити ефективність управління запасами. Це стане можливим завдяки автоматизації процесів замовлення та поповнення запасів, що в свою чергу зменшить витрати на закупівлі та мінімізує втрати продуктів харчування.

Крім того, такий додаток сприятиме покращенню комунікації та співпраці між різними відділами ресторану, що є важливим елементом успішного управління. Це дозволить забезпечити більш швидке та якісне обслуговування клієнтів, що, в свою чергу, збільшить їх задоволеність та лояльність. [6]

Підсистема обліку запасів, поєднана з підсистемою замовлень, вирішить проблему дефіциту та надлишків запасів, оптимізуючи закупівлі та забезпечуючи наявність необхідних продуктів для приготування страв. Це дозволить ресторанам більш точно планувати свою діяльність, ефективно управляти витратами та підвищити свою прибутковість.

У підсумку, впровадження мобільного додатку з підсистемою обліку запасів та замовлень є кроком у майбутнє, що дозволить ресторанам покращити свою ефективність, знизити витрати та підвищити задоволеність клієнтів.

2 Моделювання та проектування

2.1 Моделювання інформаційної системи

Роботу підсистеми обліку запасів змодельовано за допомогою методології функціонального моделювання IDEF0. Рис. 2.1 представляє контекстну діаграму, модельовану в одному блоці, яка показує основні входи, елементи керування, виходи та мехінізми, пов’язані з системою.

Рисунок 2.1 – Контекстна діаграма в нотації IDEF0

Розділивши складний процес купівлі товару на складові функції, рис. 2.2 показує ієрархічну декомпозицію.

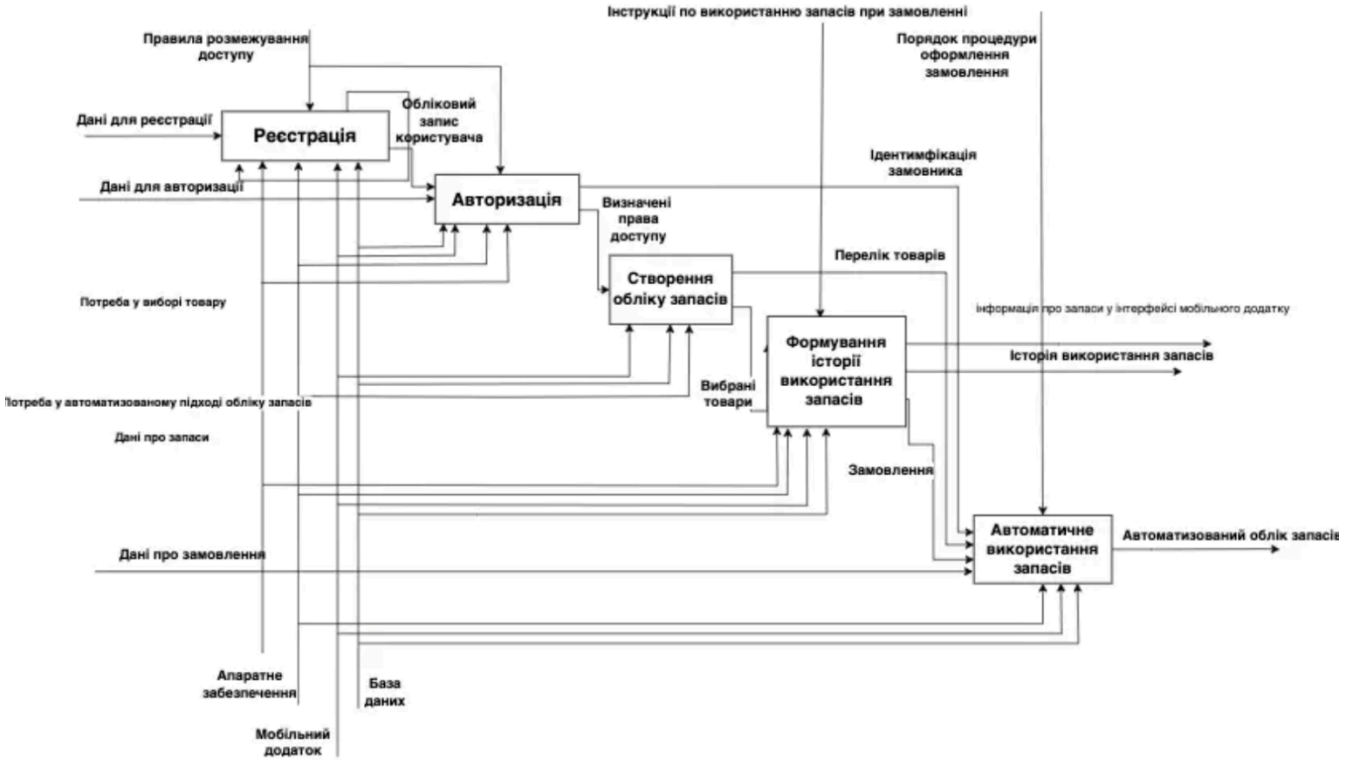

Рисунок 2.2 – Декомпозиція контекстної діаграми

2.2 Діаграма варіантів використання

Рис. 2.3 демонструє діаграму варіантів використання, яка безпосередньо показує варіанти використання, актори та їхні зв’язки.

Актори:

- Власник компанії – це особа, яка має право переглядати інформацію про замовлення, реєструватися, оформляти, обробляти та перевіряти, чи були вони витрачені.

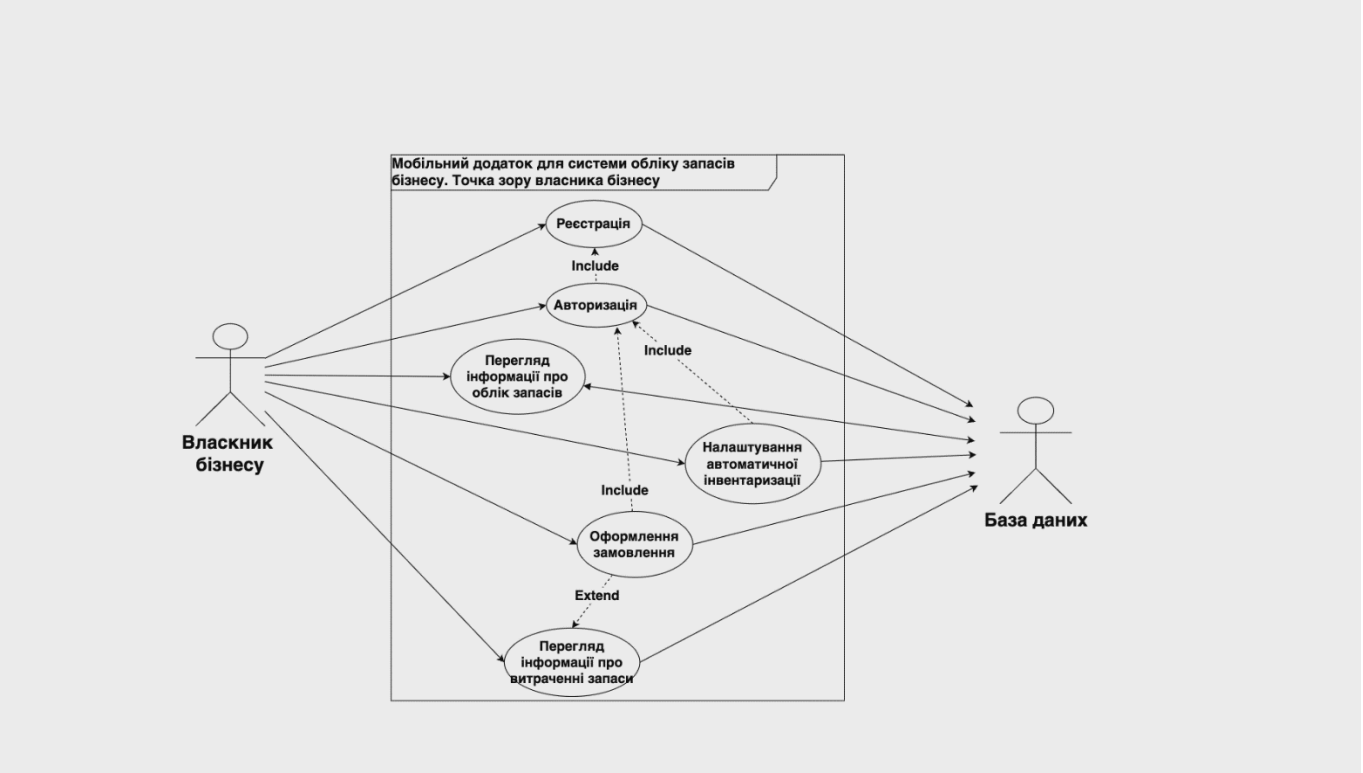

Рисунок 2.3 – Діаграма варіантів використання

Варіанти використання:

- ВВ Реєстрація та авторизація – надає дозвіл реєструватися на авторизовуватися;
- ВВ Робота з переглядом інформація про облік запасів – дозволяє власнику переглядати усю інформацію про облік запасів;
- ВВ Оформляти замовлення – надає дозвіл фласнику оформляти замовлення;
- ВВ Обробка замовлень – надає дозвіл власнику обробляти замовлення;
- ВВ Перегляд інформація про витрачені товари – надає дозвіл власнику переглядати інформацію про витрачені товари;

2.3 Проектування моделі бази даних

Щоб гарантувати ефективну реалізацію етапів життєвого циклу бази даних, на етапі проектування бази даних необхідно розробити комплексний стратегічний план. На даний момент необхідно розглянути низку важливих питань, включаючи: [7]

1. Аналіз існуючих інформаційних систем: Дослідження та оцінка наявних інформаційних систем, включаючи їх переваги, недоліки та можливі проблеми.
2. Доцільність створення нової інформаційної системи: Визначення необхідності створення нової системи на основі вимог користувачів та недоліків існуючої системи, враховуючи технічні, організаційні та фінансові аспекти.
3. Оцінка вартості проекту, необхідного обсягу робіт і ресурсів: оцінка вартості проекту, необхідного обсягу робіт, ресурсів (матеріальних, фінансових і людських) і ресурсів.
4. Визначення формату та методів збору даних: Розробка плану збору даних, включаючи методи їх збору, організації та зберігання. Визначення найкращого формату даних для потреб інформаційної системи.
5. Визначення послідовності проектування та реалізації додатків: Встановлення етапів проектування та реалізації бази даних, вибір відповідних технологій, розробка моделей і інтерфейсів.

У відповідь на отримані дані було створено ERD-діаграму реляційної бази даних (рис. 2.4), а також детальний опис інформаційної системи (табл. 2.1) і відповідних полів даних (табл. 2.2), які показують структуру та взаємозв’язки даних у системі.

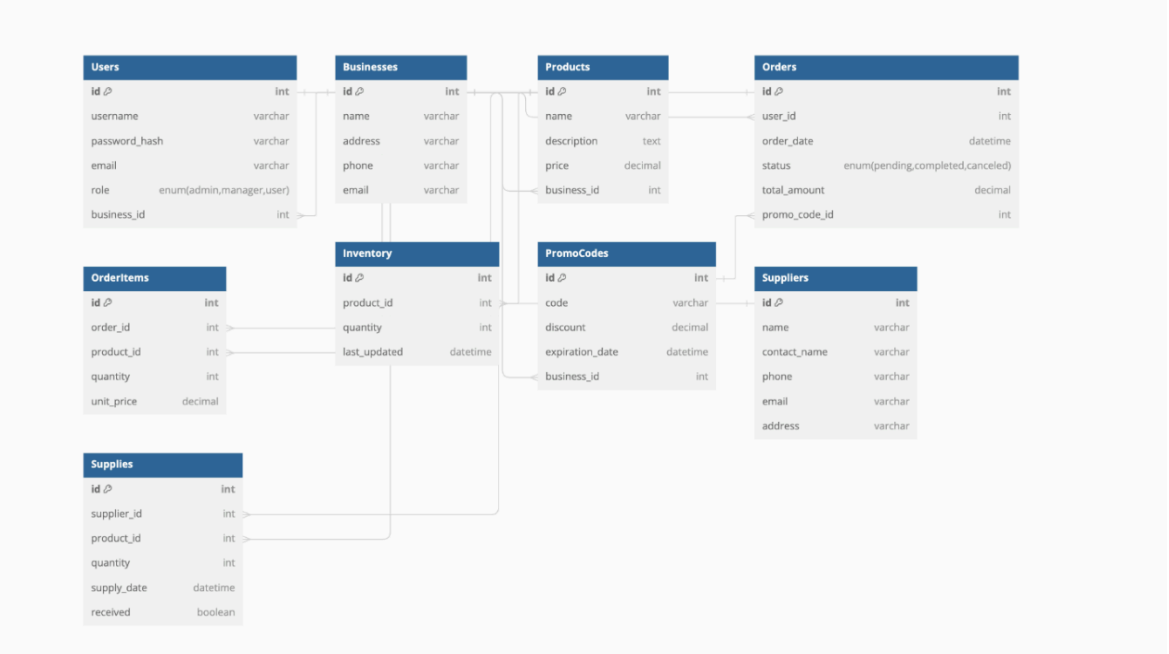

Рисунок 2.4 – ERD діаграма бази даних

Для підсистеми обліку запасів було створено поля Users, Businesses, Inventory, Suppliers та Supplies із загальної схеми бази даних.

Таблиця 2.1 – Опис таблиць бази даних

№	Назва таблиці	Призначення
1	Users	Зберігає інформацію про користувачів системи.
2	Businesses	Зберігає інформацію про бізнеси.
3	Inventory	Зберігає інформацію про запаси на складі.
4	Suppliers	Зберігає інформацію про постачальників.
5	Supplies	Зберігає інформацію про постачання.

Таблиця 2.2 – Опис полів бази даних

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
1	Users	id	Унікальний ідентифікатор	INT	PK	Не пустий
		username	Логін користувача	VARCHAR		Не пустий
		password_hash	Хеш паролю	VARCHAR		Не пустий
		email	Електронна пошта	VARCHAR		Не пустий
		role	Роль користувача (admin, manager, user)	ENUM		Не пустий
		business_id	Зовнішній ключ до Businesses	INT	FK	Не пустий
2	Businesses	id	Унікальний ідентифікатор	INTEGER	PK	Не пустий
		name	Назва бізнесу	VARCHAR		Не пустий
		address	Адреса	VARCHAR		Не пустий Не пустий
		phone	Телефон	VARCHAR		Не пустий
		email	Електронна пошта	VARCHAR		Не пустий
3	Inventory	id	Унікальний ідентифікатор	INT	PK	Не пустий
		product_id	Зовнішній ключ до Products	INT	FK	Не пустий
		quantity	Кількість на складі	INT		Не пустий
		last_updated	Останнє оновлення	DATETIME		Не пустий
4	Suppliers	id	Унікальний ідентифікатор	INT	PK	Не пустий
		name	Назва постачальника	VARCHAR		Не пустий
		contact_name	Контактна особа	VARCHAR		Не пустий
		phone	Телефон	VARCHAR		Не пустий
		email	Електронна пошта	VARCHAR		Не пустий
		address	Адреса	VARCHAR		Не пустий
5	Supplies	id	Унікальний ідентифікатор	INT	PK	Не пустий
		supplier_id	Зовнішній ключ до Suppliers	INT	FK	Не пустий

№	Таблиця	Поле	Зміст	Тип	Ключі	Обмеження
		product_id	Зовнішній ключ до Products	INT	FK	Не пустий
		quantity	Кількість	INT		Не пустий
		supply_date	Дата постачання	DATETIME		Не пустий
		received	Отримано (true/false)	BOOLEAN		Не пустий

На рис. 2.4 представлено фізичну модель бази даних.

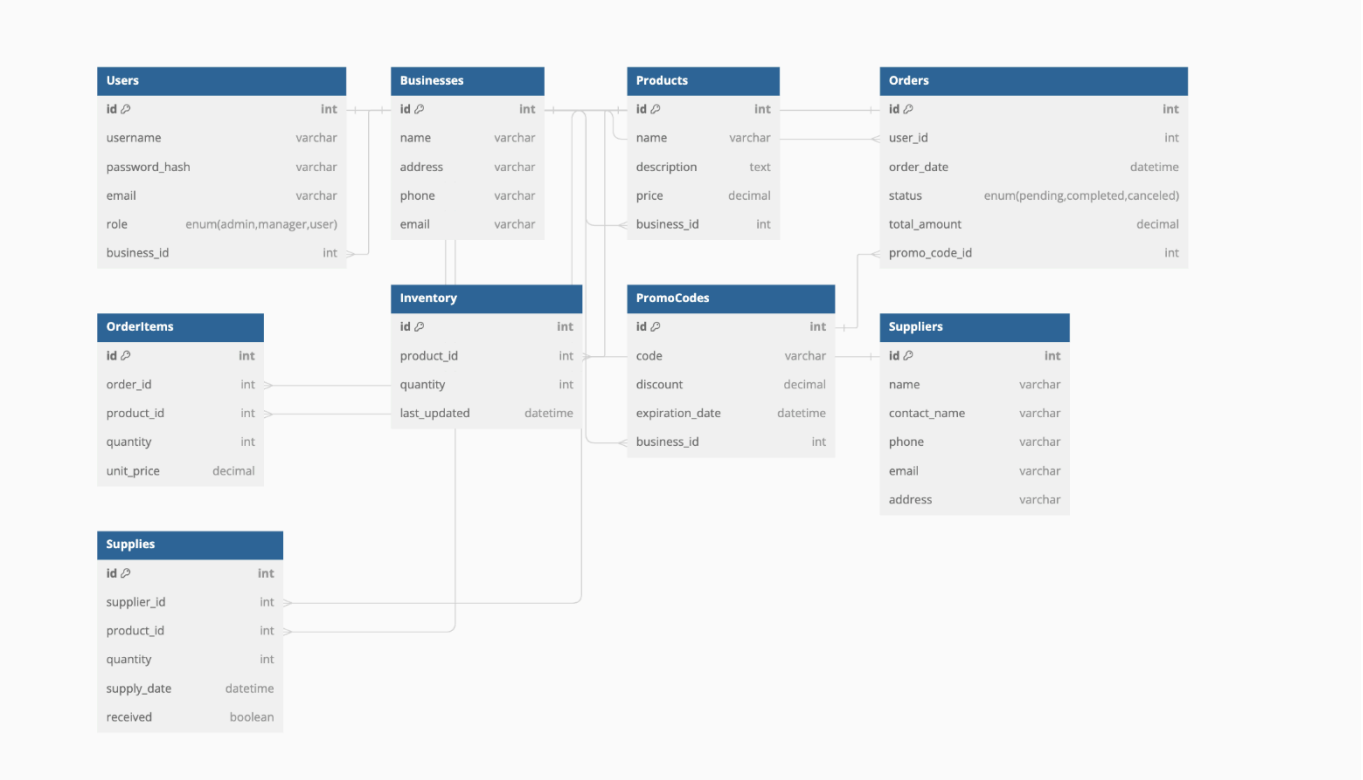

Рисунок 2.4 – База даних

3 Програмна реалізація

3.1 Архітектура додатку

Розробка користувача інтерфейсу була розроблена таким чином, щоб він був простим у використанні та зручним для користувача. Він демонструє структуру програми та дозволяє швидко та логічно переходити між екранами та розділами. Таким чином, користувачі можуть легко знайти необхідні дані та легко користуватися додатком. Підсистема обліку запасів повинна відповідати трьом основним стандартам: візуальному наповненню, змісту та структурі. Візуальне наповнення є компонентом дизайну додатку, який робить його привабливим для користувачів і покращує досвід користування. контент буде якісною та інформативною, а також матиме функції та зручні способи використання. Структура програми повинна бути логічною та легкою для використання, щоб користувачі могли швидко знайти та використовувати інформацію. [8]

Архітектура мобільного додатку складається з рівнів відображення (користувацький інтерфейс), бізнес-логіки та інформації (база даних). Користувач, який взаємодіє з мобільним додатком, розташований на верхньому рівні структури. Дві основні частини наступного рівня, мобільного додатку, є відображенням і бізнес-логікою. Користувацький інтерфейс, який містить усі елементи, з якими користувач безпосередньо взаємодіє, і логіка відображення, яка відповідає за правильне відображення даних і забезпечує зручну взаємодію користувача з додатком, є двома основними компонентами підсистеми відображення. Підсистема бізнес-логіки складається з двох підсистем: Підсистеми обліку запасів, яка контролює та керує запасами, і Підсистеми обліку замовлень, яка облікує та керує замовленнями користувачів. Секція Інформація складається з трьох основних компонентів, які можна знайти нижче: Доступ до даних, який забезпечує механізми доступу до баз даних та іншої інформації; Утиліти, які є допоміжними програмами та інструментами, необхідним для роботи додатка; і Сервіси, які надають додаткові

функції та послуги, необхідні для роботи додатка. На найнижчому рівні знаходиться компонент Дані та кеш, який забезпечує збереження та обробку даних, а також кешування, щоб пришвидшити їх доступ. Кожен із цих шарів виконує певну функцію і взаємодіє з іншими компонентами системи, щоб гарантувати, що мобільний додаток

Підтримка в управлінні ресторанним бізнесом. Перед авторизацією та після авторизацією є двома основними частинами підсистеми обліку запасів. У результаті цього розподілу користувачі можуть отримувати доступ до різних функціональних модулів і сторінок відповідно до того, який статус користувача має.

Щоб розпочати роботу з додатком, потрібно зареєструватися або отримати авторизацію. Після авторизації користувач отримує додаткові можливості та привілеї. Він може отримати доступ до персоналізованої інформації, такої як історія інветар, налаштування профілю тощо, а також брати участь у виборі свого бізнесу. Розподіл дозволяє забезпечити відповідний рівень функціональності та доступу до інформації в залежності від потреб користувачів і створити середовище для користування, зручне для користувачів.

Перелік функціональних можливостей для незареєстрованого користувача:

- реєстрація, або авторизація;

Перелік функціональних можливостей для зареєстрованого користувача:

- вибор бізнесу;
- продукти;
- інвентар;
- постачальники;
- робітники;
- аналітика;
- налаштування.

3.2 Програмна реалізація

Програма підтримки ведення ресторанного бізнесу працює за допомогою JavaScript і фреймворку React Native. Створення проекту з використанням фреймворку React Native можна розділити на кілька етапів. Нижче наведено основні етапи розробки програми.

1. Встановлення React Native:

По-перше, встановіть фреймворк React Native локально. Для цього можна використовувати команду MyReactNativeApp npx react-native init. Після завершення установки можна приступити до створення нового проекту.

2. Створення маршрутів:

React Native використовує маршрути, щоб визначити, який код використовувати при запитах до вашого додатку. Маршрути знаходяться в папці «Навігація». Приклад маршрутів:

```
const CustomDrawerItem = ({
  navigation,
  state,
  screenName,
  label,
  iconName,
}: any) => {
  const dispatch = useDispatch();

  const handleNavigation = () => {
    navigation.navigate(screenName);
  };
  return (
    <TouchableOpacity
      activeOpacity={1}
      onPress={() => {
        if (iconName === 'Logout') {
          //@ts-ignore
          dispatch(logout());
        } else {
          handleNavigation();
```

```
        }
      }}>
      <View style={styles.elementWrapper}>
        <ActionButton
          size="large"
          iconName={screenName ?? iconName}
          onPress={() => {}}
          disabled
        />
        <Divider width={15} />
        <Text style={styles.drawerItemText as any}>{label}</Text>
      </View>
    </TouchableOpacity>
  );
};

const SideBar = () => (
  <View style={{ flex: 1 }}>
    <Drawer.Navigator
      drawerContent={CustomDrawerContent}
      initialRouteName="NewOrder"
      screenOptions={{
        headerShown: false,
        drawerStyle: {
          width: Dimensions.get('window').width,
        },
        drawerStatusBarAnimation: 'slide',
      }}>
      <Drawer.Screen name="NewOrder" component={NewOrder} />
      <Drawer.Screen name="Products" component={Products} />
      <Drawer.Screen name="Inventory" component={Inventory} />
      <Drawer.Screen name="Suppliers" component={Suppliers} />
      <Drawer.Screen name="Workers" component={Workers} />
      <Drawer.Screen name="Analytics" component={Analytics} />
```

```
            <Drawer.Screen name="OrderHistory" component={OrderHistory}
/>
                                <Drawer.Screen    name="BusinessSettings"
component={BusinessSettings} />
                                 <Drawer.Screen    name="AccountSettings"
component={AccountSettings} />
                                   <Drawer.Screen    name="Subscriptions"
component={Subscriptions} />
            <Drawer.Screen name="Promocodes" component={Promocodes} />
          </Drawer.Navigator>
        </View>
      );

      export const HomeRouter: FC = () => {
        return (
          <HomeStack.Navigator
            initialRouteName="BusinessList"
            screenOptions={{
              headerShown: false,
              animation: 'slide_from_right',
            }}>
                              <HomeStack.Screen    name={'CreateBusiness'}
component={CreateBusiness} />
                                <HomeStack.Screen    name="CreateProduct"
component={CreateProduct} />
                               <HomeStack.Screen    name="CreateInventory"
component={CreateInventory} />
            <HomeStack.Screen name="Basket" component={Basket} />
                                <HomeStack.Screen    name="PickInventory"
component={PickInventory} />
             <HomeStack.Screen name="Categories" component={Categories}
/>

                               <HomeStack.Screen    name={'BusinessList'}
component={BusinessList} />
```

```
        <HomeStack.Screen name={'Business'} component={SideBar} />
      </HomeStack.Navigator>
    );
  };
```

3. Створення контролерів:

Контролери React Native групують логіку обробки запитів. У контролері описано методи обробки різних запитів. Наприклад:

```
  export const getBusinessCategories = data => {
                                                                  return
axiosInstance.get(/business-api/category/get-all/${data});
  };

  export const createBusinessProduct = data => {
    return fetch(BASE_API_URL + /business-api/product/create, {
      method: 'POST',
      body: data,
      headers: {
                                                        Authorization:
axiosInstance.defaults.headers.common.Authorization,
        'Content-Type': 'multipart/form-data',
      },
    });
  };

  export const updateBusinessProduct = data => {
    return fetch(BASE_API_URL + /business-api/product/update, {
      method: 'PUT',
      body: data,
      headers: {
                                                        Authorization:
axiosInstance.defaults.headers.common.Authorization,
        'Content-Type': 'multipart/form-data',
      },
    });
```

```
};

export const deleteBusinessProduct = data => {
  return axiosInstance.delete(/business-api/product/delete, {
    data: { productId: data },
  });
};
```

4. Використання структури та логіки:

React Native відображає сторінки вашого додатку за допомогою структури та логіки. Вони відповідають за окремі сторінки та відображають загальну структуру сторінки. Вони знаходяться в директорії «екрани». Наприклад:

```
export const Inventory: React.FC<InventoryProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();

  const { inventory } = useSelector((store: any) => store.inventory);
  const { profile } = useSelector((store: any) => store.auth);

  const [inventoryList, setInventroyList] = useState<any>(null);
  useEffect(() => {
    setInventroyList(inventory);
  }, [inventory]);
  return (
    <SafeAreaView style={styles.area}>
      <View style={styles.container}>
        <Header />
        <Divider height={20} />
        <View style={styles.headerWrapper}>
          <Text style={styles.titleText}>{t('yourInventory')}</Text>
          <ActionButton
            iconName="plus"
            onPress={() => {
              if (true) {
```

```
                navigation.navigate('CreateInventory');
              } else {
                //TOAST
              }
            }}
            size="large"
          />
        </View>
        <Divider height={20} />

        <FlatList
          showsVerticalScrollIndicator={false}
          data={inventoryList ?? []}
          renderItem={({ item }) => (
            <ProductCard
              title={item.name}
              image={item?.image ?? ''}
              price={item?.amount}
              onEdit={() => {
                     navigation.navigate('CreateInventory', { ...item,
edit: true });
              }}
              onDelete={() => {
                dispatch(
                  deleteInventory(
                    item?.id,
                    () => {
                      Toast.show({
                                                                  text1:
TOASTS[i18n.language].SUCCESS_DELETE_INVENTORY,
                      });
                    },
                    (error: string) => {
                      Toast.show({
                        text1: TOASTS[i18n.language].ERROR,
```

```
                        text2:
                            TOASTS[i18n.language][error] ?? 'Unexpected
error',
                        type: 'error',
                      });
                    },
                  ) as any,
                );
              }}
            />
          )}
          style={styles.list}
        />
      </View>
    </SafeAreaView>
  );
};
```

Розглянемо структуру додатку на рисунку 3.2 та детальний опис кожного із каталогів в таблиці 3.3

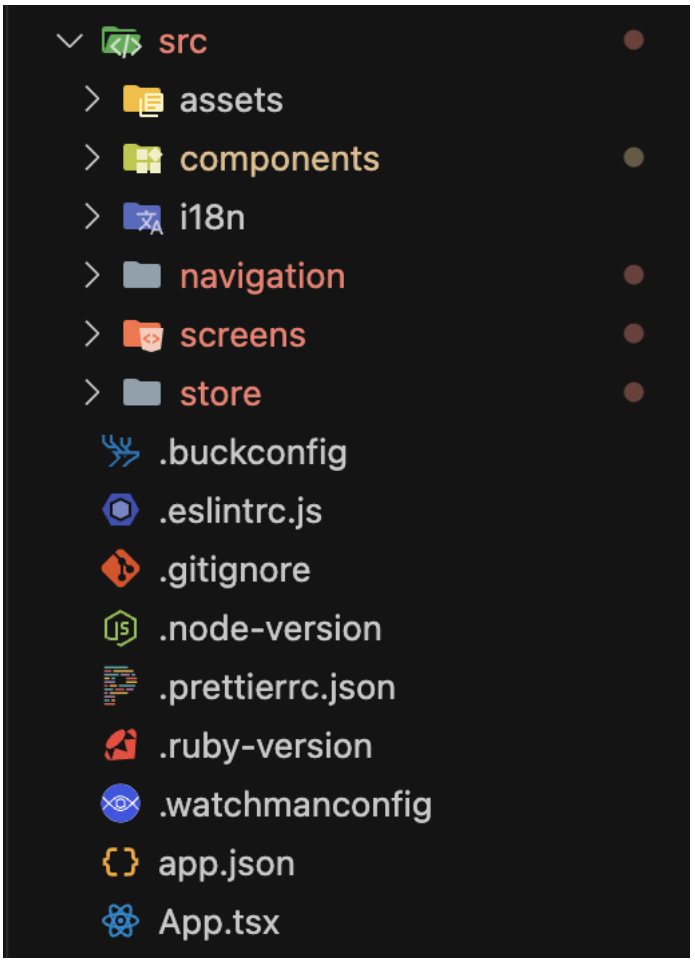

Рисунок 3.2 – Структура проекту

Таблиця 3.3 – Опис структури проекту

№	Назва	Опис
1	«src»	Загальна директорія проєкту.
2	«assets»	Директорія в якій зберігаються усі медіа проєкту.
3	«components»	Директорія де знаходяться усі компоненти додатка.
4	«i18n»	Різні переклади для застосунку.
5	«navigation»	Містить усю навігацію проєкту.
6	«screens»	Знаходиться навігація сторінок.
7	«store»	Містить запити, стейтменеджмент та інше.

Отже, результатом є підготовлена структура проекту із всіма потрібними шаблонами, файлами та налаштуваннями.

3.3 Використання програмного додатку

При завантаженні програми вимагається авторизація (рис. 3.3)

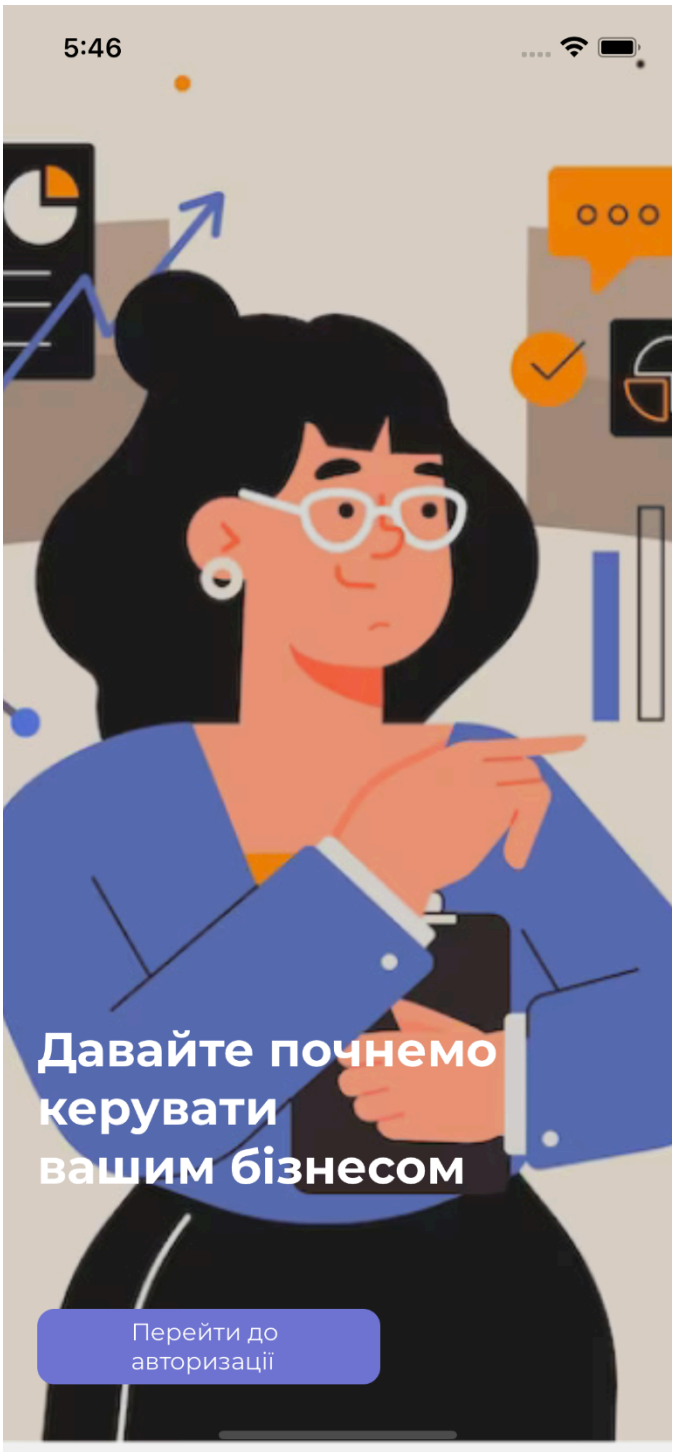

Рисунок 3.3 – Головна сторінка додатку

Після натискання на кнопку «Перейти до авторизації», ви зможете перейти на екран, призначений для авторизації або реєстрації.(рис. 3.4 та 3.5).

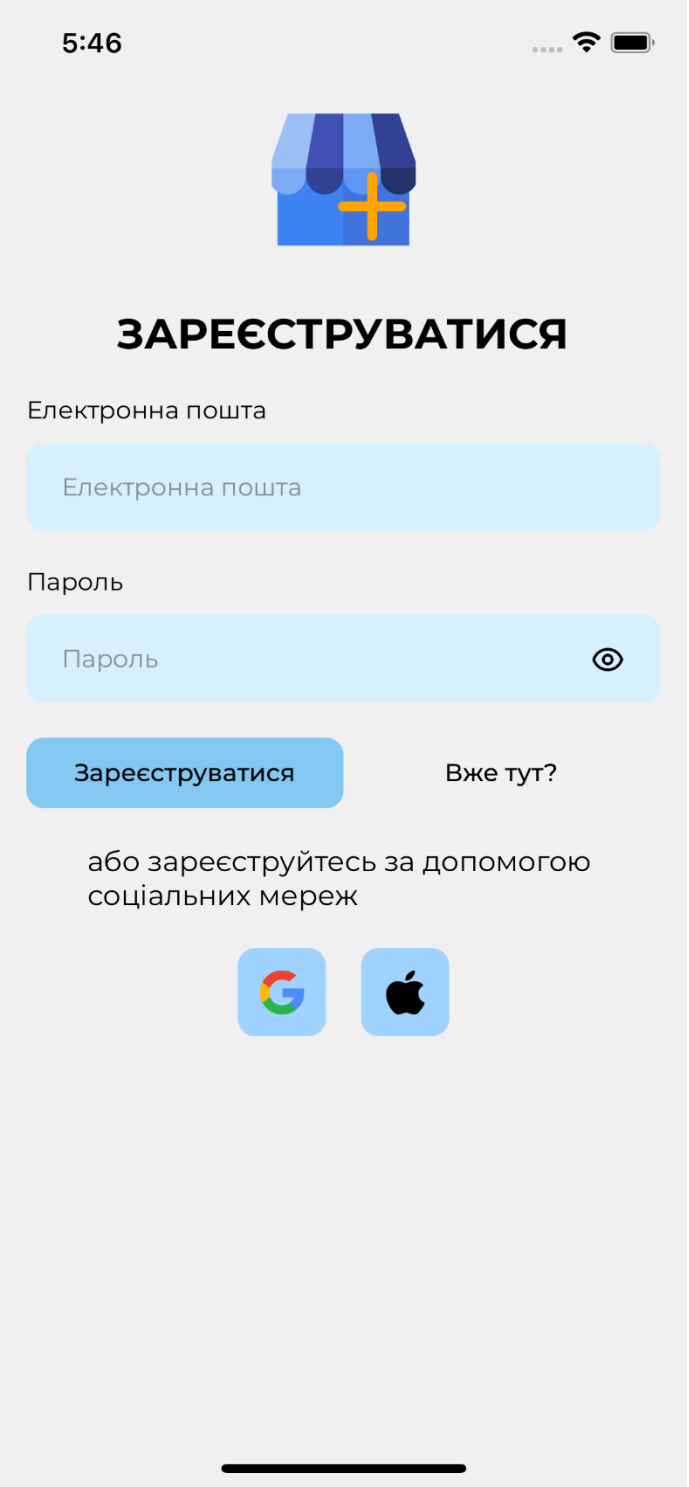

Рисунок 3.4 – Сторінка реєстрації

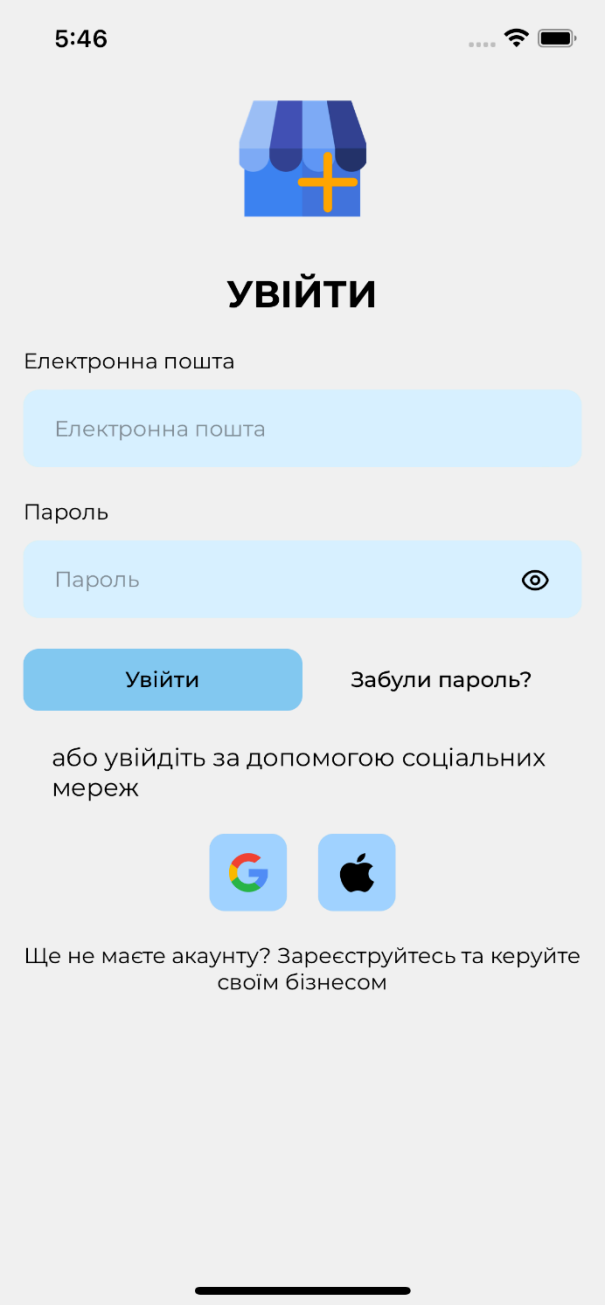

Рисунок 3.5 – Сторінка авторизації

Після отримання даних вам запропонують заснувати ваш бізнес. (рис 3.6)

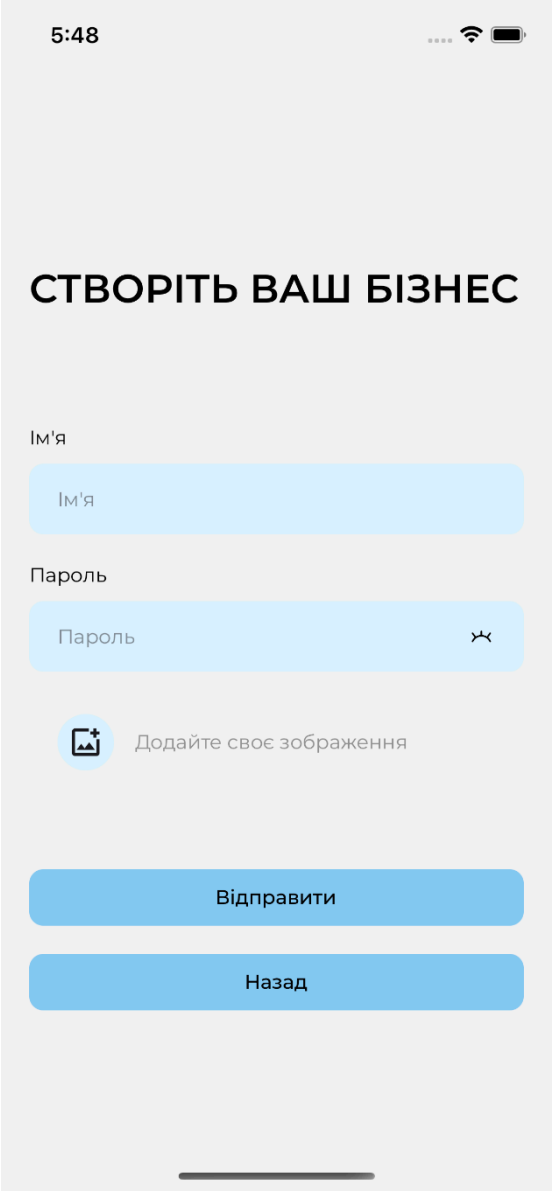

Рисунок 3.6 – Створення бізнесу

Коли ви створите необхідний бізнес, ви побачите екран, на якому є вибір бізнесів. (рис. 3.7)

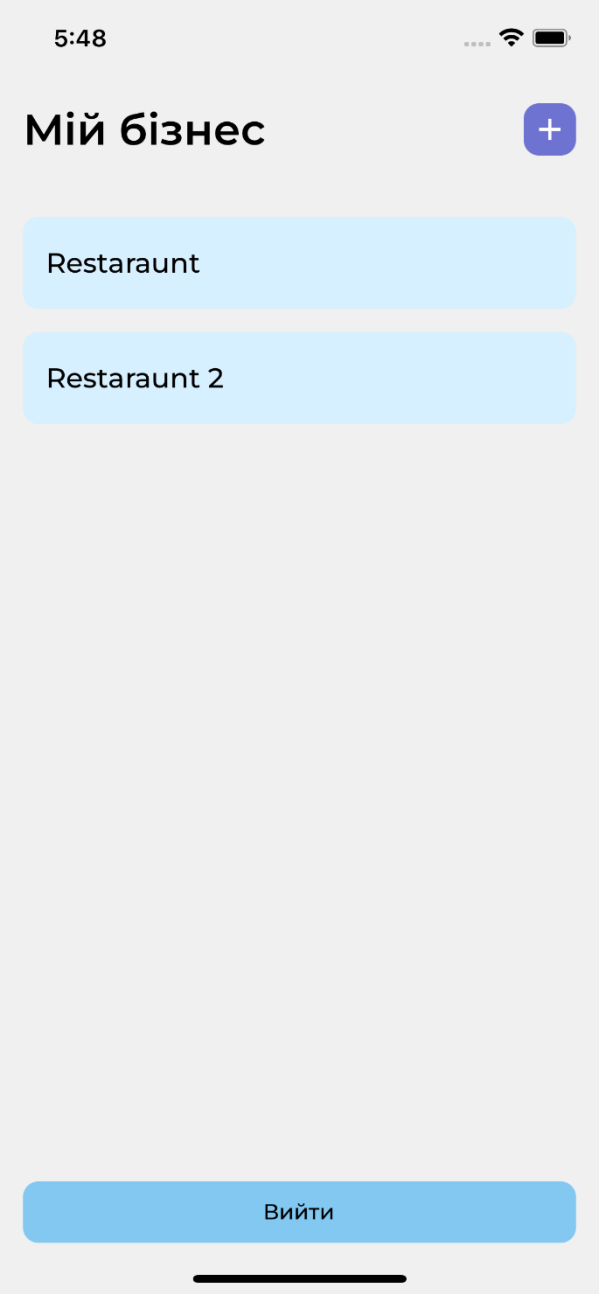

Рисунок 3.7 – Екран вибору бізнесів

Після вибору бізнесу ви побачите навігаційне меню з усіма функціями. (рис. 3.8)

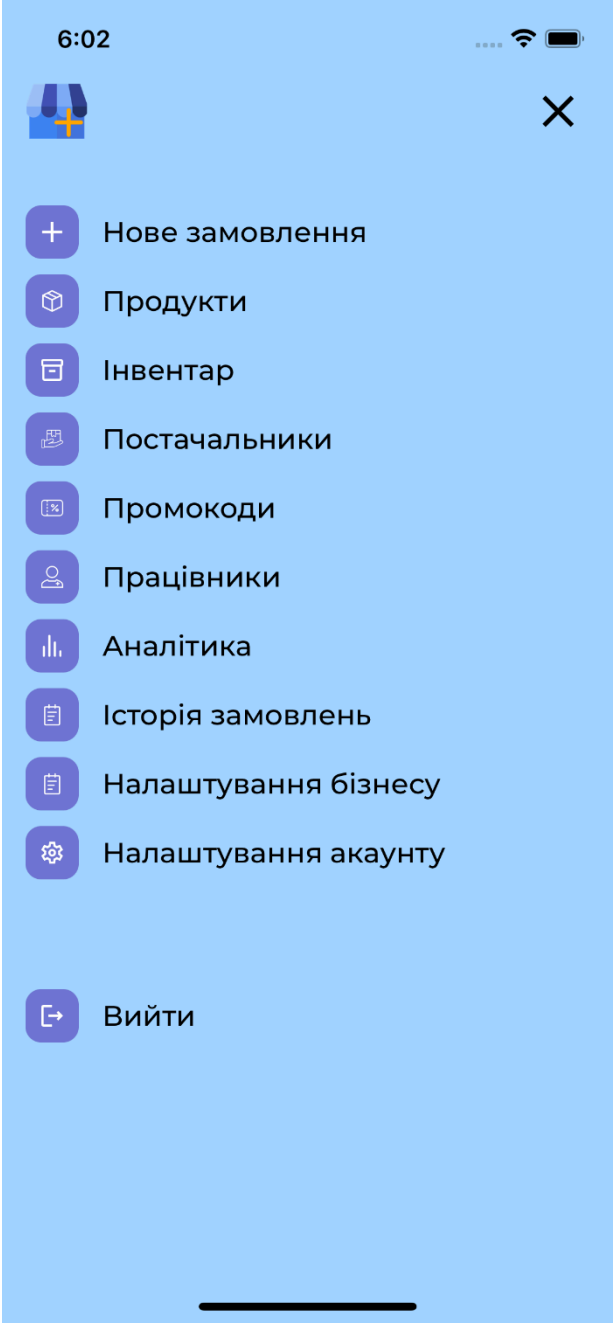

Рисунок 3.8– Навігаційне меню

Меню включаю список з всього функціоналу додатку, перший з якого «Продукти», де можна переглядати наявні продукти, редагувати їх, видаляти та додавати нові (рис 3.9)

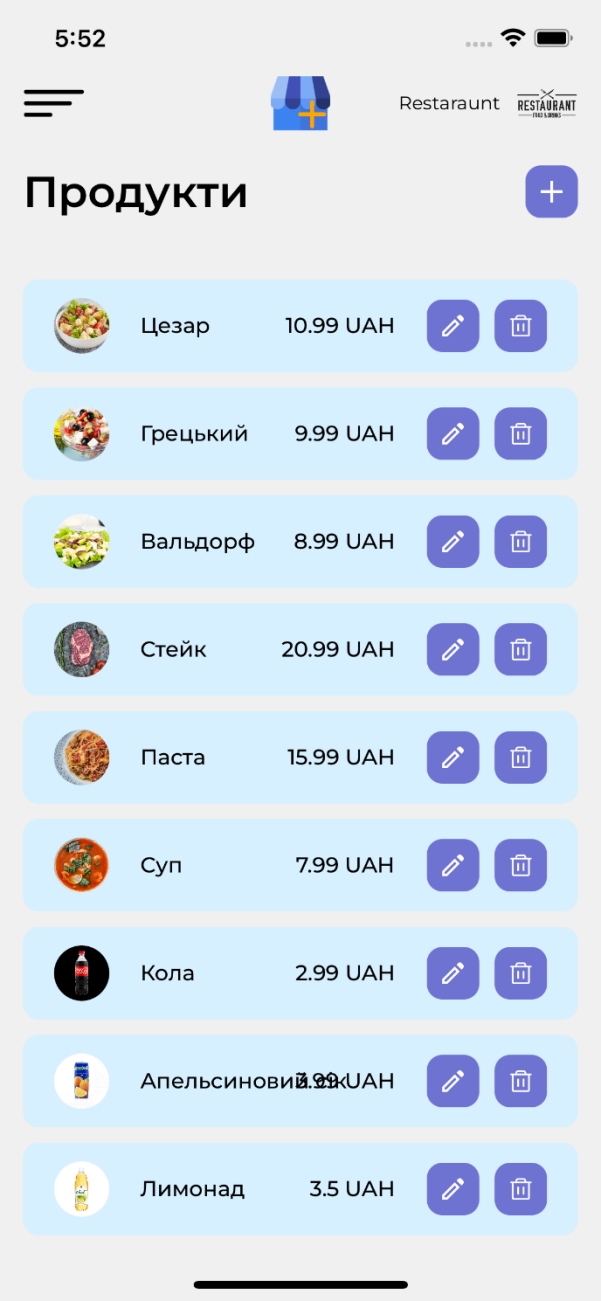

Рисунок 3.9 – Екран з продуктами

При натисканні кнопки для додавання ви побачите наступну форму (рис 3.10)

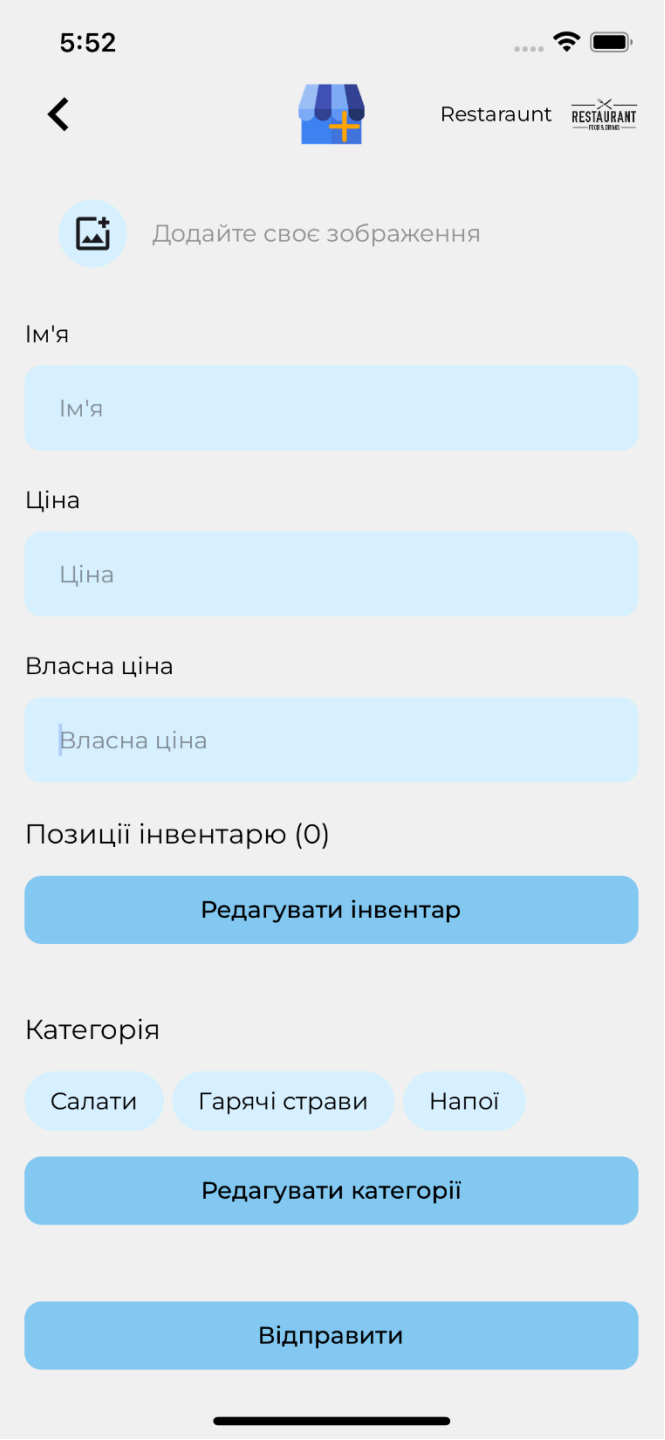

Рисунок 3.10 – Форма для додавання продуктів

Тут користувач заповнює відповідні поля, є можливість редагування інвентарю (рис.3.11) та редагуванню категорій (рис. 3.11)

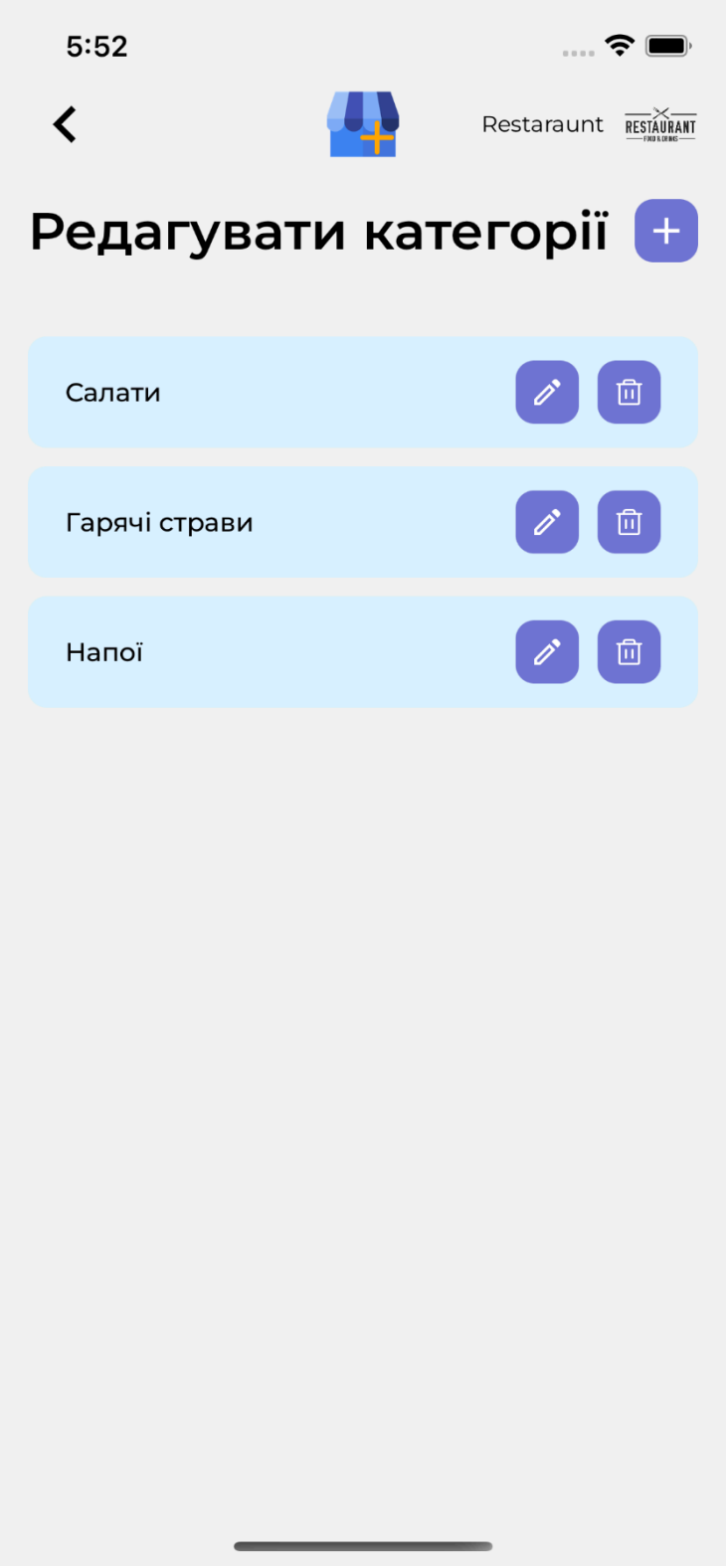

Рисунок 3.11 – Редагування категорій

Перегляд редагування категорій (рис 3.12)

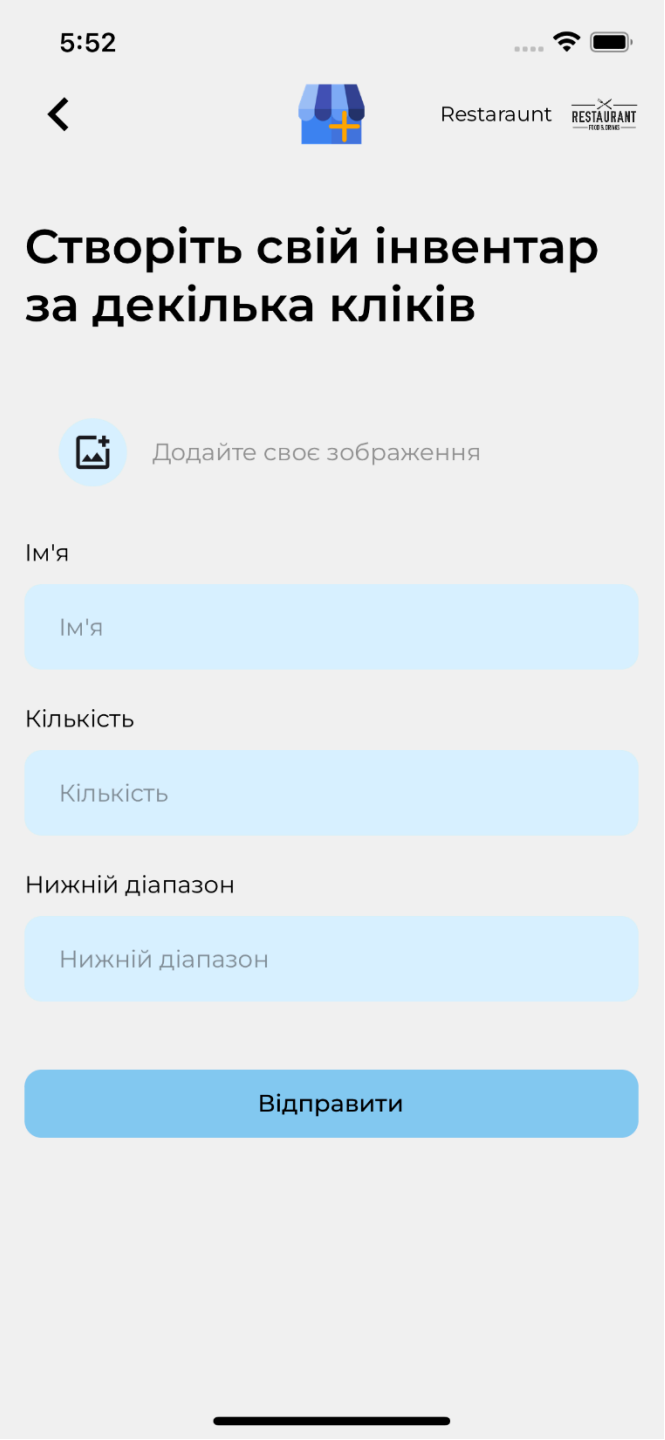

Рисунок 3.12 – Створення товару для інвентарю

Ще користувач може додати новий товар для інвентарю, які може переглядати якщо вибрати в меню пункт «Інвентар» (рис 3.13).

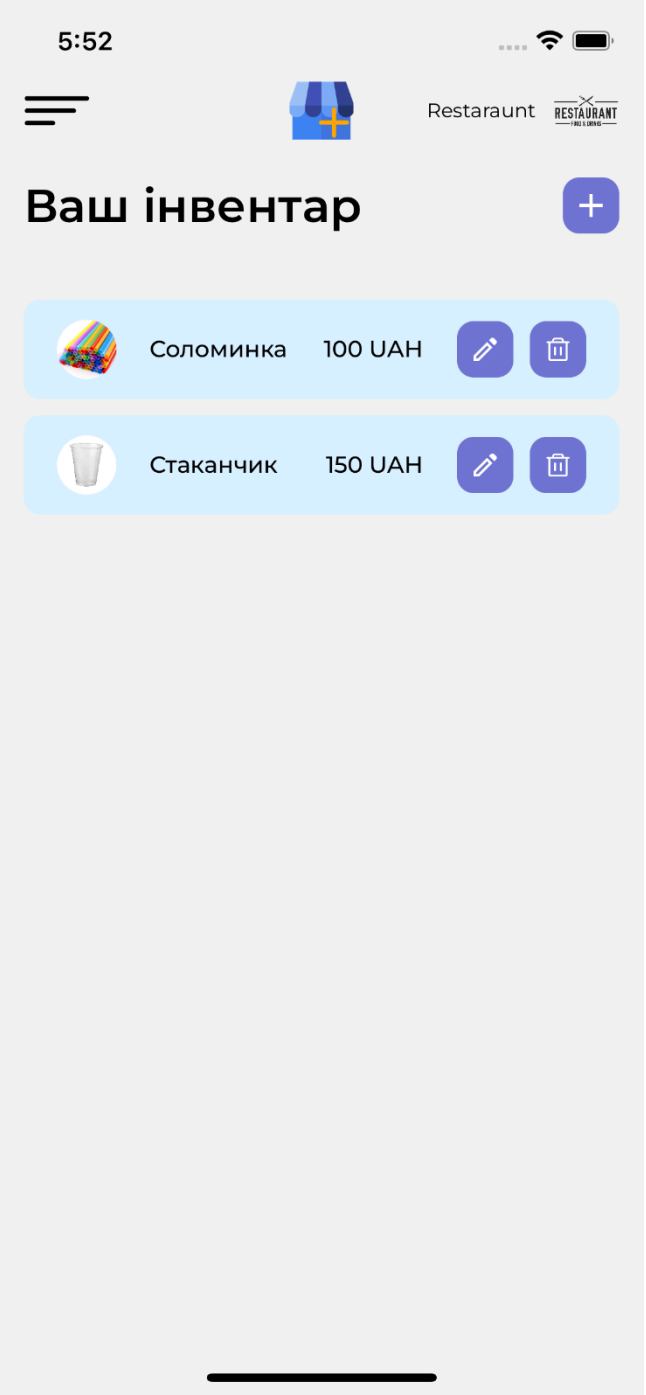

Рисунок 3.13 – «Інвентар»

Тут власник бізнесу може коригувати наявний інвентар, наприклад: видаляти, додавати (рис. 3.12) або редагувати товари (рис. 3.14).

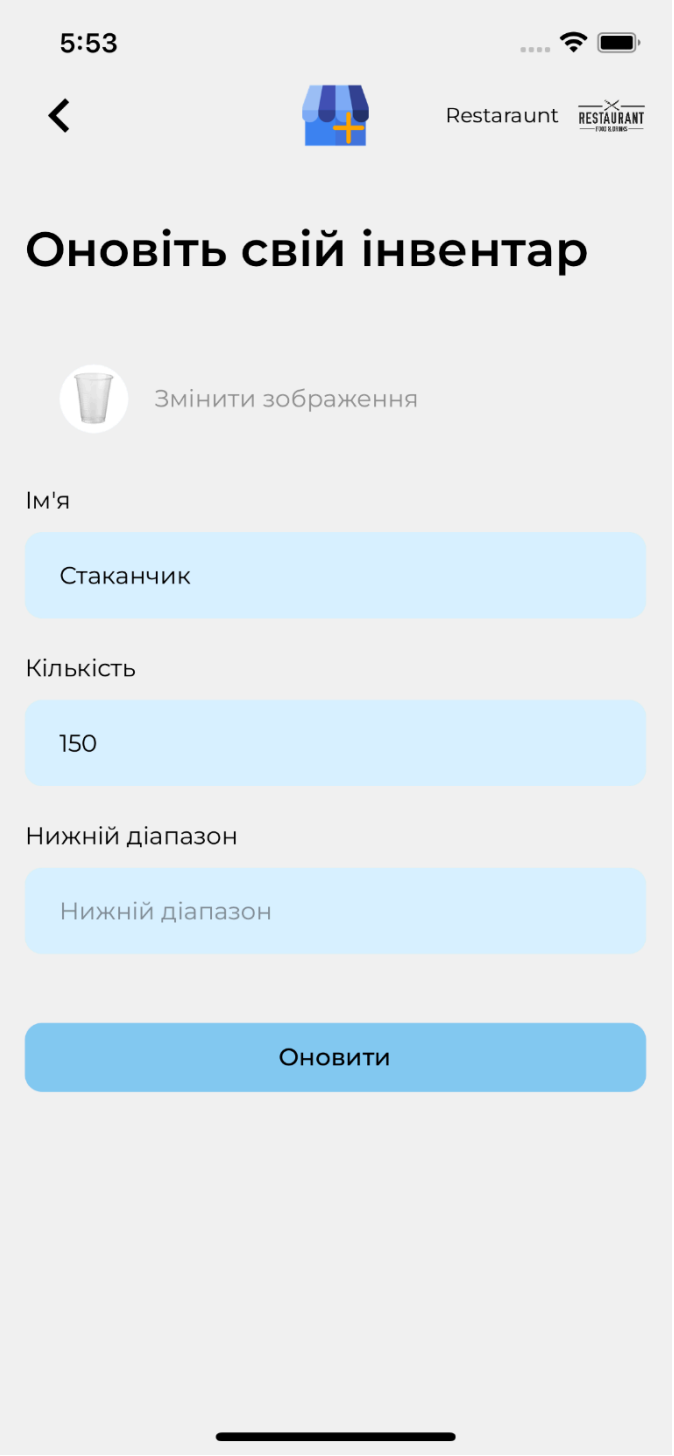

Рисунок 3.14 – Редагування товару в інвентарю

Важливою функцією в додатку є додавання постачальників (рис 3.15).

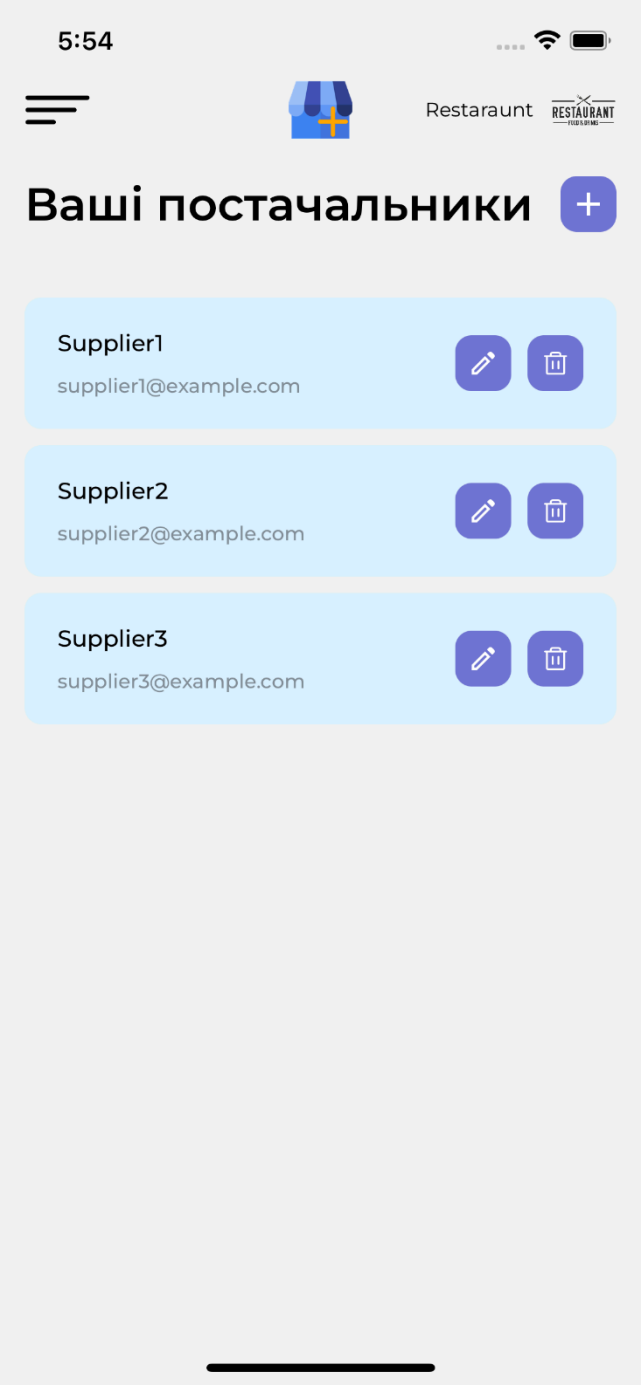

Рисунок 3.15 – Постачальники

На цьомку екрані власник додає, редагує та видаляє потрібних постачальників. (рис. 3.16).

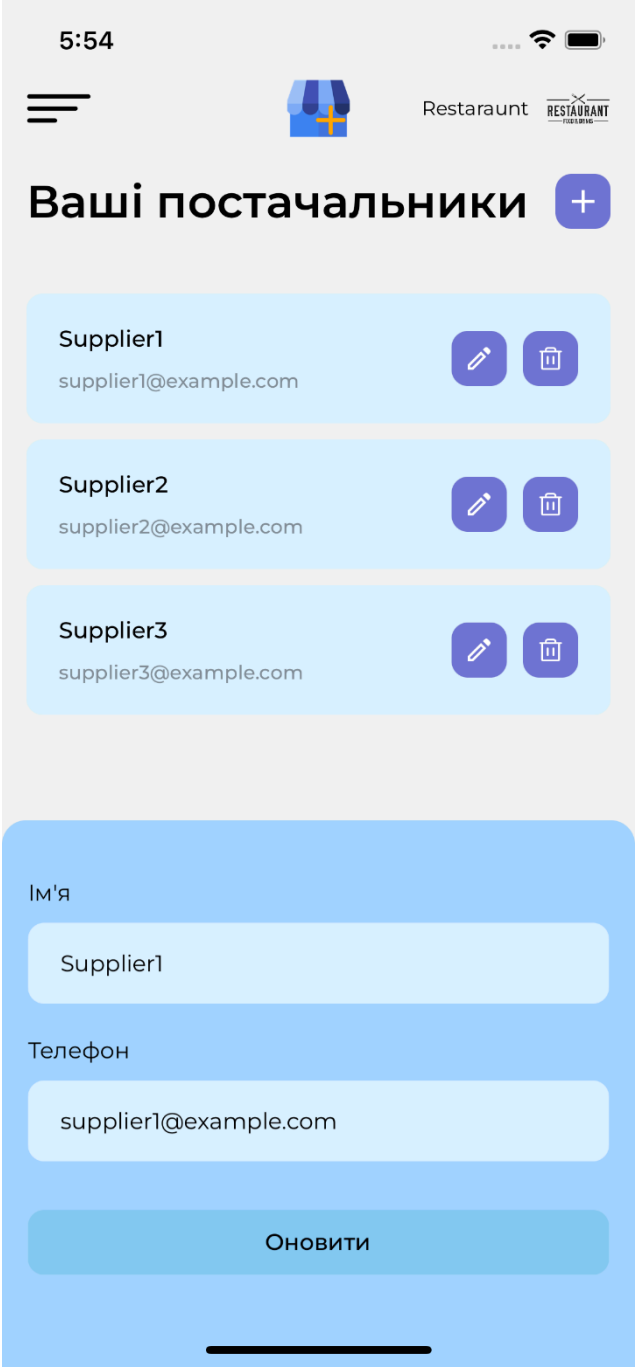

Рисунок 3.16 – Приклад додавання постачальнику

Окрім постачальників власник коригує своїх працівників (рис 3.17).

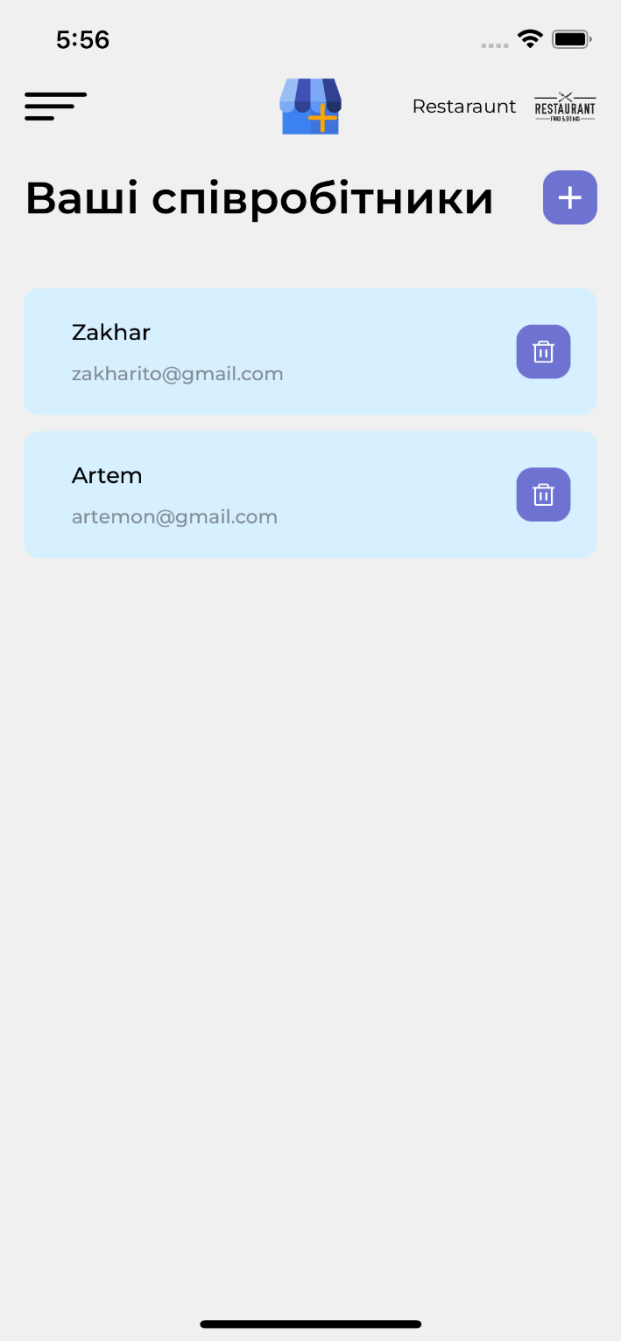

Рисунок 3.17 – Працівники

Він може видаляти їх, або додавати нових (рис 3.18)

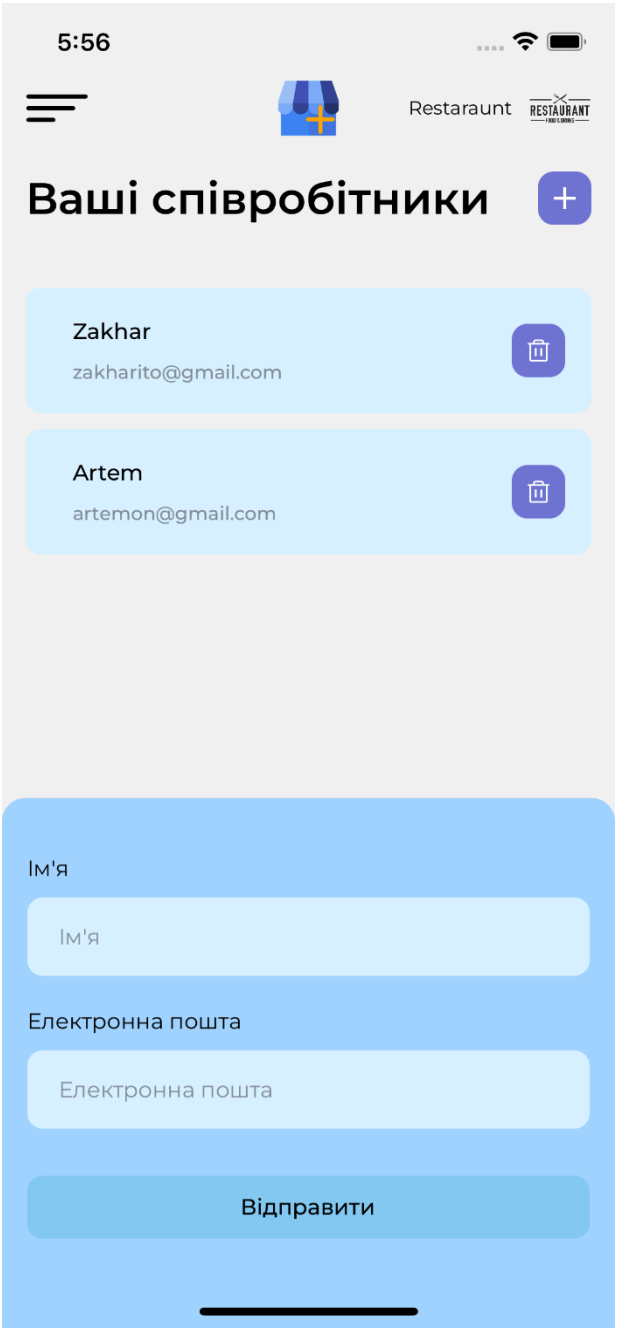

Рисунок 3.18 – Додавання працівників

ВИСНОВКИ

Результати, одержані у ході виконання кваліфікаційної роботи бакалавра: проведено аналіз останніх досліджень та публікацій у сфері мобільних додатків для ресторанного бізнесу, проаналізовано наявні програмні продукти-аналоги, розроблено концепцію мобільного додатку, який буде включати підсистему обліку запасів, спроектовано та реалізовано мобільний додаток, проведено тестування мобільного додатку та оцінено його ефективність.

Розроблений мобільний додаток для ресторанного бізнесу з підсистемою обліку запасів є важливим інструментом, що принесе численні переваги управлінцям та власникам ресторанів. Переваги цього додатку варто розглянути у контексті його впливу на ефективність, витрати, прибуток та зручність використання.

По-перше, мобільний додаток покращує ефективність управління запасами через автоматизацію процесів замовлення та поповнення запасів, відстеження рівня запасів та прогнозування попиту. Це дозволить ресторанам оптимізувати закупівлі та покращити управління запасами.

По-друге, додаток знижує витрати за рахунок мінімізації втрат продуктів харчування, оптимізації закупівлі та зниження витрат на зберігання. Це сприятиме підвищенню прибутку ресторанів та їхній фінансовій стійкості.

Крім того, додаток підвищує задоволеність клієнтів через більш швидке та якісне обслуговування, покращення комунікації та співпраці між відділами ресторану. Це сприятиме залученню та утриманню клієнтів у закладі.

Зручність використання є ще однією перевагою цього додатку, оскільки він надає мобільний доступ до інформації про запаси, замовлення та інші аспекти ведення ресторанного бізнесу. Це зробить роботу персоналу більш ефективною та зручною.

Щоб максимізувати користь від впровадження такого додатку, рекомендується надати навчання персоналу щодо його використання, а також регулярно оновлювати та вдосконалювати його на основі відгуків користувачів. Крім того, для майбутніх

досліджень важливо розглянути можливість інтеграції з іншими системами ресторанного бізнесу, розробку додаткових функцій та використання додатку у інших сферах бізнесу. Загалом, розробка та впровадження мобільного додатку для підтримки ведення ресторанного бізнесу з підсистемою обліку запасів та урахуванням підсистеми замовлень може стати значним кроком вперед у напрямку оптимізації ведення ресторанного бізнесу, підвищення його ефективності та прибутковості.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. "Mobile Inventory Management Solutions for Restaurants: Best Practices and Implementation Strategies", Industry Whitepaper by XYZ Inventory Solutions, 2023

2. Стаття "Streamlining Restaurant Inventory with Mobile Apps", [Електронний ресурс] – Режим доступу до ресурсу: https://www.restaurant.org/articles/news/streamlining-restaurant-inventory-with-mobile-apps (дата звернення: 10.07.2023)

3. "Optimizing Restaurant Inventory Control through Mobile Technology", Research Report by ABC Research Institute, 2022

4. "Case Study: Implementation of Mobile Inventory Management in a Fast-Casual Restaurant Chain", Case Study Report by DEF Consulting, 2021

5. "Mobile Apps for Real-Time Inventory Tracking in Restaurants: Challenges and Solutions", Conference Presentation, International Conference on Restaurant Technology, 2023

6. "Inventory Management Best Practices for Mobile Restaurant Apps", Webinar Recording, Restaurant Tech Solutions, 2022

7. "The Role of Mobile Apps in Inventory Optimization for Restaurants", Expert Panel Discussion Summary, Restaurant Management Summit, 2023

8. Книга "Mobile Applications in Inventory Management for Restaurants", автор Назва Автора, видавництво, 2020

9. Examining the impact of artificial intelligence on hotel employees through job insecurity perspectives [https://www.sciencedirect.com/science/article/abs/pii/S0278431920303157]

10. A POS Solution for Restaurants Journal of Foodservice Business Research [https://www.researchgate.net/publication/232990525_A_POS_Solution_for_Restaurants]

11. Best Restaurant Inventory Management Software 2024 [https://www.forbes.com/advisor/business/software/restaurant-inventory-management-software/]

ДОДАТОК А. Технічне завдання

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку додатку

«Мобільний додаток підтримки ведення ресторанного бізнесу. Підсистема обліку запасів»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій

_____ Парфененко Ю.В.

Студент групи ІТ-03-2

_____ Лиховид М.Р

Суми 2024

1. Призначення й мета створення додатку

1.1 Призначення мобільного додатку

Підсистема обліку запасів у мобільному додатку для ресторанного бізнесу є ключовим елементом, спрямованим на автоматизацію та оптимізацію управління запасами продуктів харчування та напоїв. Ця підсистема має важливе значення для ресторанів з огляду на свої можливості та переваги, які вона надає.

Впровадження підсистеми обліку запасів дозволяє ресторанам ефективно знижувати витрати через кращий контроль за наявністю продуктів та оптимізацію закупівель. Це створює умови для більш раціонального використання ресурсів і зниження непотрібних витрат.

Крім того, підсистема сприяє покращенню ефективності ресторану, оскільки вона автоматизує процеси обліку та звітності. Це дозволяє економити час і зусилля персоналу, які можуть бути спрямовані на більш важливі аспекти роботи закладу.
Зменшення втрат є ще однією важливою перевагою підсистеми обліку запасів. Шляхом кращого прогнозування попиту та своєчасного замовлення продуктів, ресторани можуть уникати надмірних запасів та недостачі товарів, що сприяє економічній ефективності бізнесу.

Не менш важливою є можливість підвищення рівня обслуговування клієнтів завдяки швидкому та точному облобленню запасів. Підсистема обліку запасів у поєднанні з іншими функціями мобільного додатку створює сприятливі умови для покращення взаємодії з клієнтами та задоволення їхніх потреб.

Отже, підсистема обліку запасів у мобільному додатку для ресторанного бізнесу відіграє важливу роль у підтримці оптимального управління ресурсами та підвищенні якості обслуговування клієнтів.

1.1 Мета створення додатку

Мобільний додаток для управління запасами у ресторанному бізнесі має важливі функціональні особливості, спрямовані на полегшення та оптимізацію робочих процесів. Основні характеристики такого додатку включають:

Автоматизація обліку запасів: Додаток забезпечує автоматичне введення даних про надходження та витрачання продуктів харчування, обчислення наявних запасів, формування замовлень на закупівлю та створення звітів щодо динаміки запасів. Це дозволяє значно економити час та уникати помилок в обліку.

Підвищення зручності роботи: Мобільний додаток повинен бути максимально простим у використанні для всього персоналу ресторану, незалежно від їхнього рівня технічної підготовки. Інтуїтивний інтерфейс та доступність на різних пристроях роблять роботу з додатком максимально зручною.

Інтеграція з іншими системами: Додаток повинен бути інтегрований з іншими ключовими системами ресторанного бізнесу, такими як система управління замовленнями, система управління персоналом та система бухгалтерського обліку. Це забезпечує однорівневу обробку даних і покращує загальну ефективність управління.

1.3 Цільова аудиторія

Цільовою аудиторією мобільного додатку є персонал ресторану, який займається:

- Прийманням та обробкою замовлень: офіціанти, бармени, кухарі;
- Закупівлею продуктів харчування: менеджери з закупівель;
- Зберіганням та обліком запасів: складські працівники.

2 Вимоги до додатку

2.1 Вимоги до додатку в цілому

2.1.1 Вимоги до структури й функціонування додатку

Застосунок має бути доступним в основних магазинах на телефоні або планшеті. Додаток повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2.1.2 Вимоги до персоналу

- Навички роботи з мобільними пристроями**:** персонал ресторану повинен володіти загальними навичками роботи з мобільними пристроями та стандартними мобільними операційними системами (Android, iOS).
- **Знання інтерфейсу мобільного додатку:** персонал ресторану повинен бути ознайомлений з інтерфейсом мобільного додатку та його основними функціональними можливостями..

2.1.3 Вимоги до збереження інформації

Уся інформація надана у додатку буде зберігатися у базі даних реалізованій засобами системи управління базами даних MongoDB.

2.1.4 Вимоги до розмежування доступу

Додаток буде доступний для всіх користувачів без обмежень.

Завантаження буде можливе з App Store та Google Play.

Для доступу до адміністративної панелі потрібна авторизація та аутентифікація.

2.2 Структура додатку

2.2.1 Загальна інформація про структуру додатку

Структура додатку являє собою набір сторінок, які також є пунктами головного меню.

Такими розділами є:

Авторизація – на сторінці зображене поле для авторизації користувачів.

Головна – сторінка для вибору потрібних функції.

Інвентарь – інформація про наявний інвентарь.

Продукти – інформація про наявну продукці..

Постачальники – інформація про існуючих постачальників продуктів.

Аналітика – сторінка з аналітикою вашого бізнесу.

Налаштування бізнесу – сторінка с налаштуванням створеного бізнесу у додатку.

2.2.2 Навігація

Відповідно до бажаного замовником дизайну додатку, для навігації, у шапці буде створена система контент меню. Меню необхідне для швидкого переміщення користувача по усім доступним сторінкам. Меню буде відображатися на всіх сторінках, щоб відвідувач міг в будь-який момент часу перейти на будь-яку сторінку додатку.

2.2.3 Наповнення додатку (контент)

Для управління контентом додатку буде використана внутрішня система управління.

Заповнення та редагування контенту додатку має бути зроблено через панель керування, використовуючи інформацію з бази даних.

Всю інформацію для наповнення додатку має надавати ресторан який використовує додаток.

2.2.4 Дизайн та структура додатку

Мобільний додаток буде мати стильний та сучасний дизайн, що буде зручним для користувачів. Основними кольорами додатку будуть синій та білий. Інтерфейс додатку буде простим і зрозумілим для користувачів, щоб вони могли легко знаходити потрібну інформацію та виконувати необхідні дії без зайвих проблем.

Розташування елементів на головній сторінці додатку схематично показано на рисунку А.1.

Рисунок А.1 – Схема головної сторінки

2.2.5 Система навігації (карта додатку)

Карта додатку зображена на рисунку А.2.

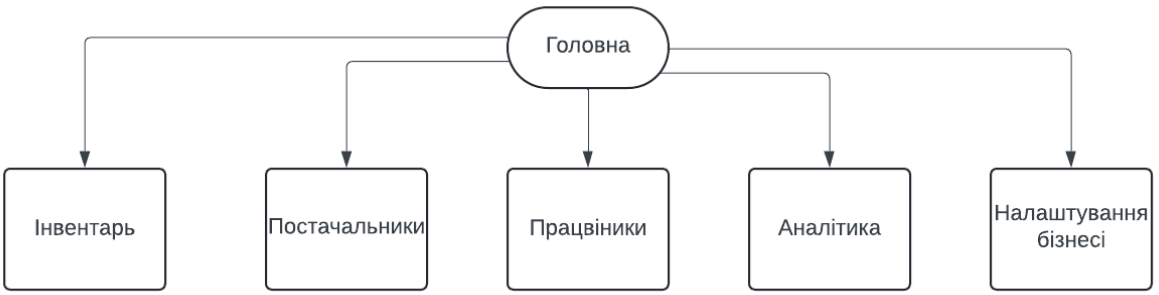

Рисунок А.2 – Карта додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Потреби користувача, визначені на основі рішення замовника, представлені у таблиці А.1.

Таблиця А.1 – Потреби користувача

ID	**Потреби користувача**	**Джерело**
UN-01	Можливість переглянути та додати продукти	Працівник
UN-02	Можливість переглянути інвентар	Працівник
UN-03	Перегляд та додання працівників	Власник
UN-04	Перегляд аналітики	Працівник, власник
UN-05	Прийняття та обробка постачальників	Працівник, власник

UN-06	Налаштування параметрів бізнесу	Власник

2.3.2 Функціональні вимоги

На основі потреб користувача були визначені такі функціональні вимоги:

-реєстрація та авторизація користувачів;

-перегляд та обробка інвентарю;

-перегляд та обробка продуктів;

-перегляд та обробка постачальників;

-перегляд та обробка працівників;

-аналітика;

-адміністрування інформації, видалення, зміну користувацької групи, надання користувацьких прав;

2.3.3 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 – Системні вимоги

ID	**Системні вимоги**	**Пріоритет**	**Опис**
R-01	Наявність модуля інвентарю	M	Надає можливість створити та переглянути інветарь
R-02	База даних із інвентарем, продуктами та постачальниками	M	Зберігає інформацію про весь інвентарь,

			продукти та всіх їх постачальників
R-03	Модуль обробки продуктів	M	Надає можливість клієнтам відстежувати статус своїх замовлень
R-04	Модуль налаштування працівників	M	Надає можливість керувати працівниками
R-05	Модуль постачальників	M	Надає можливість керувати постачальниками продуків

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація додатку відбувається з використанням:

- React Native

- Node JS
- MongoDB

2.4.2 Вимоги до лінгвістичного забезпечення

Додаток має бути виконаний українською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

Телефон або планшет:

- **Android:** 4.4 і вище.
- **iOS:** 8.0 і вище.

3 Склад і зміст робіт зі створення додатку

Докладний опис етапів роботи зі створення додатку наведено в таблиці А.3.

Таблиця А.3 – Етапи створення додатку

	Склад і зміст робіт	Ст

	Постановка цілей необхідних для досягнення певного результату	1
	Складання технічного завдання	3
	Підготовка прототипу	2

	Створення макету дизайну додатку	2
	Верстка	3
	Робота над модулями для додатку	2
	Робота з контентом	1
	Перевірка працездатності додатку	1
	Завершення роботи	1
	Загальна тривалість робіт	17

4 Вимоги до складу й змісту робіт із введення додатку в експлуатацію

Для того, щоб клієнти могли користуватися додатком, його потрібно розмістити в Інтернеті, а саме в AppStore та PlayMarket. Додаток переноситься в магазини разом з наповненням бази даних, яке потім доробляється.

ДОДАТОК Б

Планування робіт

Деталізація мети проекту методом SMART. Продуктом дипломного проекту є додаток доповненої реальності системи дизайну інтер’єру.

Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific (конкретна)	Створити додаток доповненої реальності системи дизайну інтер’єру, для візуалізації предметів дизайну в реальному просторі.
Measurable (вимірювана)	Результатом роботи проекту є оцінка замовника.
Achievable (досяжна)	Реалізації системи здійснюється за допомогою середовища розробки React Native, з використанням бекенд Node JS, та бібліотекою MongoDB.
Relevant (реалістична)	У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.
Time-framed (обмежена у часі)	Ціль має часове обмеження. Робота повинна бути виконана у терміни, що були оговорені замовником проекту. Проект повинен бути виконаний згідно з календарним планом.

Планування змісту структури робіт. Основним інструментом для планування змісту структури робіт служить WBS діаграма − графічне подання згрупованих елементів проекту у вигляді пакета робіт, які ієрархічно пов’язані з продуктом проекту. Побудуємо структуру WBS, у якій детально опишемо роботи, які

потрібно виконати на кожному етапі створення проекту. Виконаємо декомпозицію робіт для даного проекту. Діаграма WBS зображена на рис. Б.1.

Планування структури організації, для впровадження готового проекту (OBS). Після побудови WBS розробимо організаційну структуру виконавців OBS. Організаційна структура проекту стосується тільки внутрішньої організаційної структури проекту і не стосується відносин проектних груп чи учасників з батьківськими організаціями. Діаграма OBS зображена на рис. Б.2. Список виконавців, що функціонують в проекті знаходиться в табл. Б.2.

Таблиця Б.2 − Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Лиховид М.Р.	Виконує розробку основного функціоналу проекту та інтерфейс користувача
Тестувальник	Лиховид М.Р	Відповідає за тестування функціоналу та дизайну додатку, перевірку моделі на адекватність
Консультант проекту	Парфененко Ю.В.	Формує завдання на розробку проекту
Менеджер проекту	Лиховид М.Р.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних

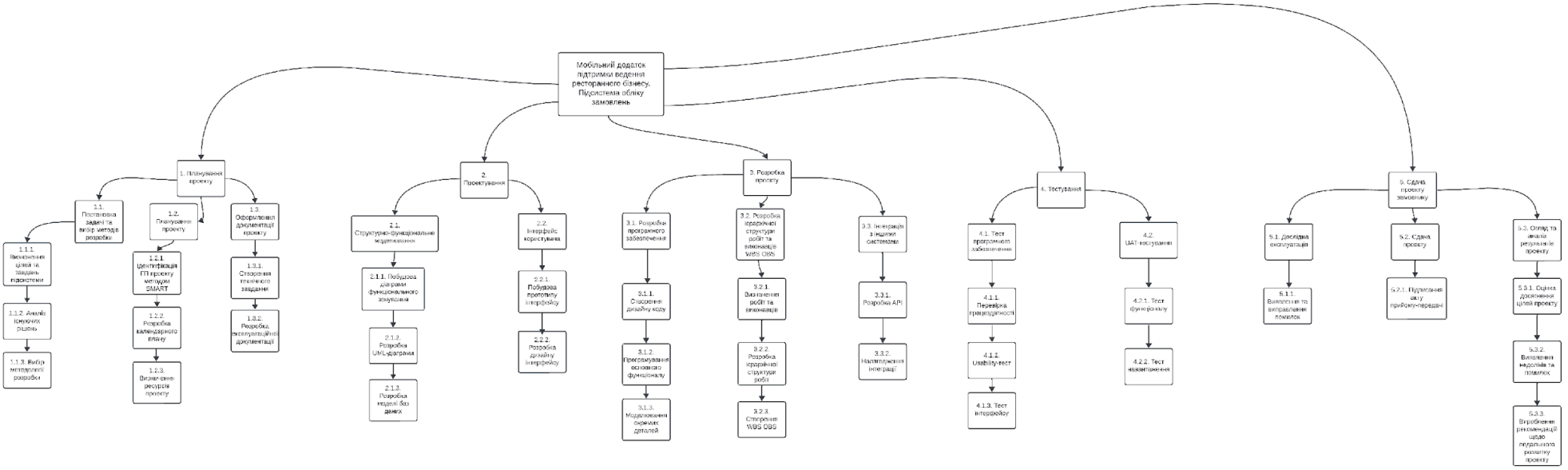

Рисунок Б.1 − WBS. Структура робіт проекту

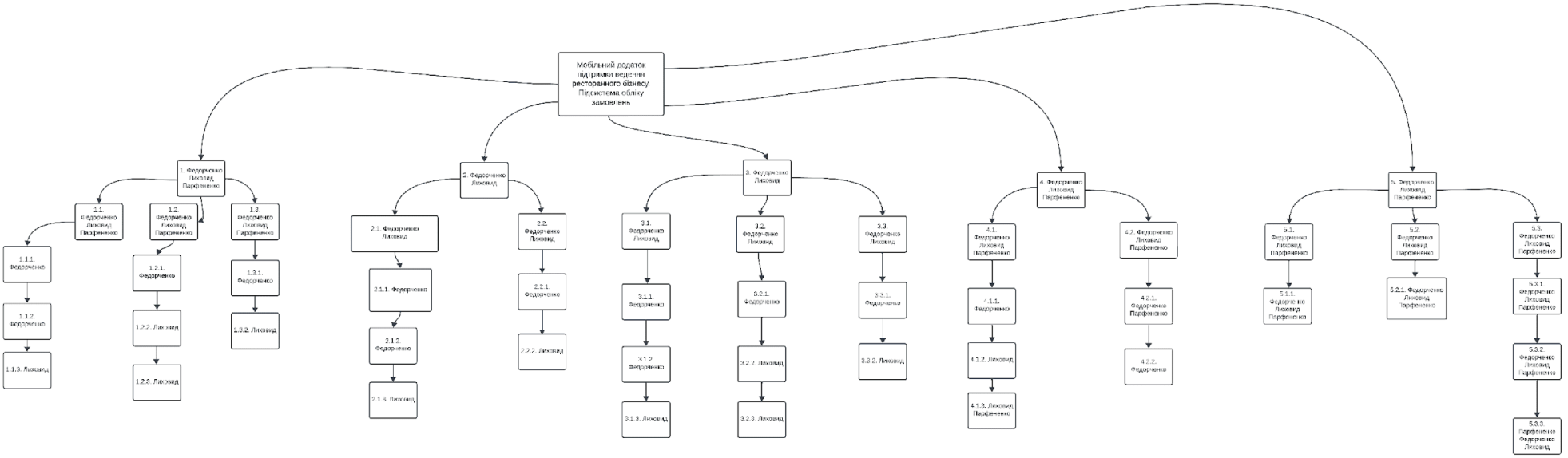

Рисунок Б.2 − Організаційна структура проекту (OBS)

Діаграма Ганта. Далі побудуємо календарний план виконання дипломного проекту. Найпоширеніший формат графіка в будь-якій галузі — діаграма Ганта. Цей графік дозволяє менеджерам проекту і всій команді розробників візуалізувати графіки часу і взаємозв'язок між окремими завданнями та етапами роботи над проектом. Тривалість виконання робіт зазначена в днях, але фактична тривалість виконання робіт приблизно дорівнює 2-3 години на день. Для того щоб мати реальне уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, з урахуванням вихідних та святкових днів, побудовано календарний графік. Діаграма Ганта та список робіт діаграми Ганта зображені на рис. Б.3-Б.6.

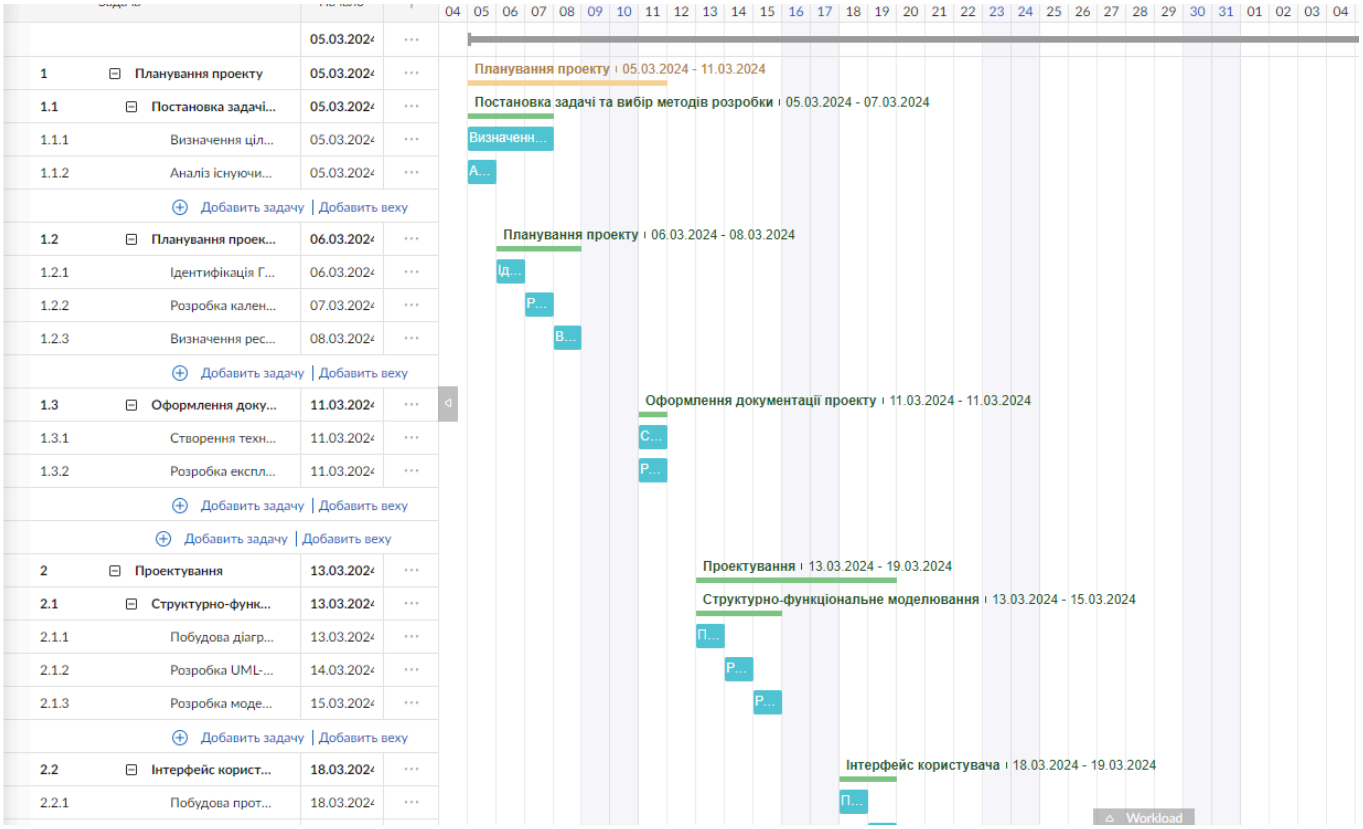

Рисунок Б.3 − Діаграма Ганта

Рисунок Б.4 – Продовження діаграми Ганта

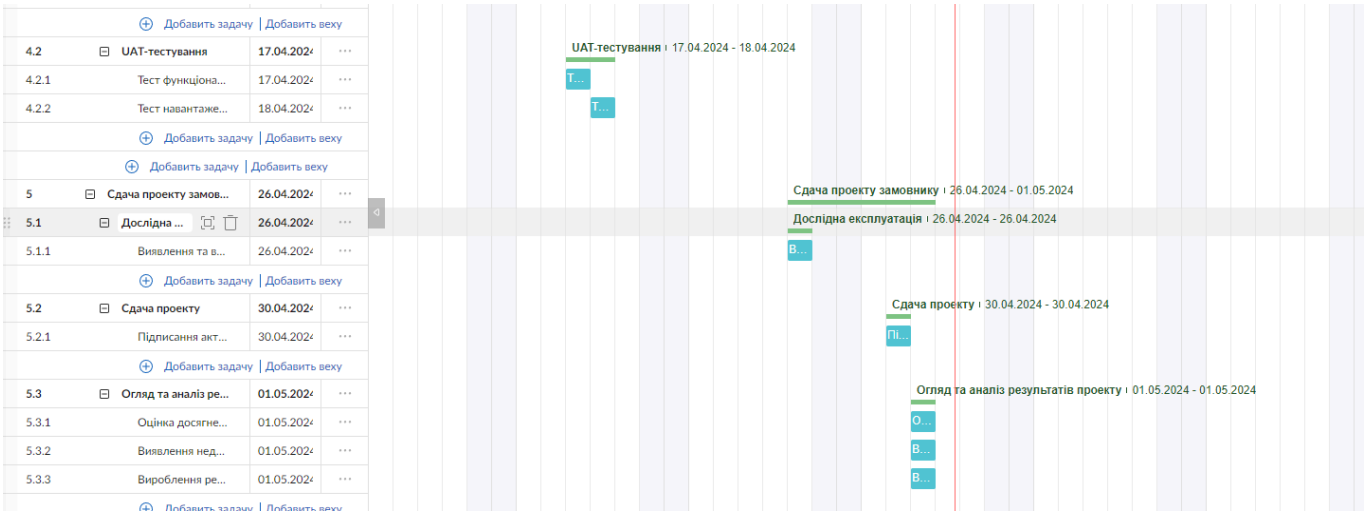

Рисунок Б.5 – Продовження діаграми Ганта

Аналіз ризиків. Виконаємо якісну і кількісну оцінку ризиків роботи. При якісній оцінці визначимо ризики, що потребують швидкого реагування. Така оцінка визначить ступінь важливості ризику і дозволить вибрати спосіб реагування. Кількісна оцінка ризиків буде виконана для більш повної ідентифікації ризиків та ступеня їхнього впливу на виконання проекту. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків. У табл. Б.5 знаходиться

класифікація ризиків за показниками ймовірності виникнення ризику та величині втрат.

Далі виконаємо планування реагування на ризики — це розробка методів і технологій зниження негативного впливу ризиків на проект. Визначимо ефективність розробки реагування на проект, визначимо чи будуть наслідки впливу ризику на проект позитивними або негативним. Оцінюємо ризики за показниками, що знаходяться в табл. Б.3. На основі оцінки будуємо матрицю ймовірності виникнення ризиків та впливу ризику, що зображена на рис. Б.7.

Таблиця Б.3 − Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

Ймовірність виникнення				
	3	RS_2	RS_3,	RS_5, RS_9
	2	RS_1, RS_13	RS_4, RS_6	RS_7, RS_14
	1	RS_12	RS_8, RS_11	RS_10, RS_15
		1	2	3

Вплив ризику

Рисунок Б.7 − Матриця ймовірності виникнення ризиків та впливу ризику

- зелений колір – прийнятні ризики;
- жовтий колір – виправданні ризики;
- червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в табл. А.4.

Таблиця Б.4 − Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	1,8,11,12,13
2	Виправдані	3≤R≤4	2,4,6,10,15
3	Недопустимі	6≤R≤9	3,5,7,9,14

Таблиця Б.5 − Оцінка ймовірності виникнення, величини витрат та індексу ризику

Таблиця Б.5 − Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Недотримання санітарних норм при обліку запасів	Низька	Високий	3	1. Розробити та впровадити чіткі інструкції щодо санітарних норм при роботі з запасами 2. Регулярно проводити навчання персоналу з питань санітарних норм 3. Контролювати дотримання санітарних норм.	Попередження	Зупинити роботу з обліку запасів, якщо будуть виявлені порушення санітарних норм.
RS_2	Відкритий	Недосяжність підсистеми	Низька	Середній	4	1. Використовувати надійні хостинг-провайдерів 2. Регулярно створювати резервні копії даних 3. Розробити план відновлення роботи підсистеми на випадок збоїв.	Попередження	Використовувати резервні методи обліку запасів на випадок недосяжності підсистеми.

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_3	Відкритий	Несанкціонований доступ до даних про запаси	Низька	Високий	3	1. Використовувати надійні методи шифрування даних про запаси 2. Застосовувати протоколи безпечного доступу до даних 3. Контролювати доступ до даних про запаси.	Попередження	Відновити дані про запаси з резервних копій у разі несанкціонованого доступу.
RS_4	Відкритий	Недосконалість системи звітності	Середня	Середній	4	1. Провести аналіз потреб користувачів щодо звітності 2. Розробити гнучкі та інформативні звіти про запаси. >3. Збирати та аналізувати відгуки користувачів щодо звітності.	Попередження	Внести зміни до системи звітності на основі відгуків користувачів.
RS_5	Теоретичний	Зміна ринкових цін на продукти харчування	Низька	Високий	3	1. Регулярно моніторити зміни ринкових цін 2. Розробити механізм автоматичного оновлення цін у підсистемі 3. Бути готовим до ручного оновлення цін, якщо це буде необхідно.	Попередження	Внести зміни до підсистеми на основі відгуків користувачів.

Продовження таблиці Б.5

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_6	Відкритий	Недосконалість інтеграції з іншими підсистемами мобільного додатку	Середня	Середня	4	1. Провести ретельне тестування інтеграції з іншими підсистемами 2. Використовувати стандартизовані інтерфейси програмування 3. Залучити до тестування інтеграції фахівців з інших підсистем.	Попередження	Використовувати ручні методи обліку запасів, якщо виникнуть проблеми з інтеграцією.
RS_7	Відкритий	Незадоволеність персоналу роботою з підсистемою	Середня	Середній	4	1. Провести навчання персоналу з питань роботи з підсистемою 2. Збирати та аналізувати відгуки персоналу щодо підсистеми 3. Вносити зміни до підсистеми на основі відгуків персоналу.	Попередження	Використовувати резервні методи обліку запасів, якщо персонал не зможе працювати з підсистемою.
RS_8	Теоретичний	Зміна технологій обліку запасів	Низька	Високий	3	1. Регулярно моніторити нові технології обліку запасів 2. Зберігати гнучкість у плануванні розвитку підсистеми 3. Бути готовим до оновлення підсистеми, якщо це буде необхідно.	Попередження	-

Додаток В

КОД РЕАЛІЗАЗІЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ

```
import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import { BottomSheetTextInput } from '@gorhom/bottom-sheet';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { BusinessListProps } from './types';
import { useDispatch, useSelector } from 'react-redux';
import { loginBusiness } from '../../store/slices/business';
import { getMe, logout } from '../../store/slices/auth';
import axiosInstance from '../../store/axios';
import { useTranslation } from 'react-i18next';
import i18next from 'i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const BusinessList: React.FC<BusinessListProps> = () => {
  const navigation = useNavigation<any>();
```

```
const dispatch = useDispatch<any>();
const { t } = useTranslation();

const [open, setOpen] = useState(false);
const [password, setPassword] = useState('');

const [isValidForm, setIsValidForm] = useState({
  password: true,
});

const validateForm = (onSuccess: any) => {
  let isValid = true;

  if (password.length > 6) {
    setIsValidForm(prev => ({ ...prev, password: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, password: false }));
  }
  if (isValid) {
    onSuccess();
  }
};

const [current, setCurrent] = useState<any>(null);

const onPressDismiss = () => {
  Keyboard.dismiss();
};

const { profile, token } = useSelector((store: any) => store.auth);
useEffect(() => {
  !profile?.businesses?.length && navigation.navigate('CreateBusiness');
}, [profile?.businesses]);

useEffect(() => {
  setTimeout(() => {
    dispatch(
      getMe(),
      () => {},
      (error: string) => {
        Toast.show({
```

```
          text1: TOASTS[i18n.language].ERROR,
          text2: TOASTS[i18n.language][error] ?? 'Unexpected error',
          type: 'error',
        });
      },
    );
  }, 1000);
}, [token]);

useEffect(() => {
  profile?.language && i18next.changeLanguage(profile?.language);
}, [profile?.language]);

useEffect(() => {
  navigation.navigate('Business');
}, []);

const renderItem = ({ item }: any) => (
  <TouchableOpacity
    style={styles.listItem}
    onPress={() => {
      setOpen(true);
      setCurrent(item);
    }}>
    <Text style={styles.listText}>{item.name}</Text>
  </TouchableOpacity>
);
// console.log(profile.subscription.businessLength);

return (
  <SafeAreaView style={styles.area}>
    <View style={styles.headerWrapper}>
      <Text style={styles.titleText}>{t('myBusiness')}</Text>
      <ActionButton
        iconName="plus"
        onPress={() => {
          if (
            profile?.businesses?.length <
            profile?.subscription?.businessLength
          ) {
            navigation.navigate('CreateBusiness');
          } else {
```

```
            //TOAST
          }
        }}
        size="large"
      />
    </View>
    <Divider height={20} />

    <TouchableWithoutFeedback onPress={onPressDismiss}>
      <FlatList
        showsVerticalScrollIndicator={false}
        data={profile?.businesses ?? []}
        renderItem={renderItem}
        style={styles.list}
      />
    </TouchableWithoutFeedback>
    <View style={styles.buttonWrapper}>
      <Button
        onPress={() => {
          dispatch(logout());
        }}
        text={t('logout')}
      />
</View>



    <BottomSheet
      open={open}
      snapPoints={['40%']}
      onDismiss={() => {
        setOpen(false);
      }}>
      <Input
        placeholder={t('password')}
        onChange={setPassword}
        secureTextEntry
        inBottomSheet
        isValid={isValidForm.password}
      />

      <Divider height={40} />
```

```
            <Button
              text={t('openBusiness')}
              mode="large"
              onPress={() => {
                validateForm(() =>
                  dispatch(
                    loginBusiness(
                      {
                        businessId: current?.id,
                        password,
                      },
                      () => {
                        setOpen(false);
                        navigation.navigate('Business');
                      },
                      (error: string) => {
                        Toast.show({
                          text1: TOASTS[i18n.language].ERROR,
                                text2: TOASTS[i18n.language][error] ?? 'Unexpected
error',
                          type: 'error',
                        });
                      },
                    ),
                  ),
                );
              }}
            />
          </BottomSheet>
        </SafeAreaView>
      );
    };

    const styles = StyleSheet.create({
      buttonWrapper: {
        width: '100%',
        paddingHorizontal: 15,
        marginBottom: Platform.OS === 'ios' ? 0 : 20,
      },
      textInput: {
        width: '100%',
        height: 50,
```

```
    backgroundColor: '#D9F0FF',
    borderRadius: 10,
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-Regular',

    paddingLeft: 20,
  },
  headerWrapper: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
    paddingHorizontal: 15,
    paddingTop: 20,
  },
  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
  area: { flex: 1 },
  list: { paddingHorizontal: 15, paddingTop: 20, height: '70%' },
  listItem: {
    height: 60,
    width: '100%',
    backgroundColor: '#D9F0FF',
    borderRadius: 10,
    marginBottom: 15,
    justifyContent: 'center',
    paddingLeft: 15,
  },
  listText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 18,
    fontWeight: '500',
    color: '#000000',
  },
});

import React, { useState } from 'react';
import {
```

```
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  ImageInput,
  Input,
  KeyboardAware,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { BusinessSettingsProps } from './types';
import { useDispatch, useSelector } from 'react-redux';
import { deleteBusiness, updateBusiness } from '../../store/slices/business';

import { useTranslation } from 'react-i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const BusinessSettings: React.FC<BusinessSettingsProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch<any>();
  const { t } = useTranslation();

  const onPressDismiss = () => {
    Keyboard.dismiss();
  };

  const { currentBusiness } = useSelector((store: any) => store.business);
```

```
const [name, setName] = useState<any>(currentBusiness?.name);
const [currency, setCurrency] = useState<any>(
  currentBusiness?.currency ?? '',
);
const [isValidForm, setIsValidForm] = useState({
  name: true,
  currency: true,
});

const validateForm = (onSuccess: any) => {
  let isValid = true;

  if (name.length > 3 && name.length < 20) {
    setIsValidForm(prev => ({ ...prev, name: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, name: false }));
  }
  if (currency.length > 1 && name.length < 5) {
    setIsValidForm(prev => ({ ...prev, currency: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, currency: false }));
  }
  if (isValid) {
    onSuccess();
  }
};

const [photo, setPhoto] = useState<any>('');
const [imageUrl, setImageUrl] = useState<any>(currentBusiness?.image ?? '');

return (
  <SafeAreaView style={styles.area}>
    <KeyboardAware>
      <TouchableWithoutFeedback onPress={onPressDismiss}>
        <View style={styles.container}>
          <Header />
          <Divider height={20} />
          <Text style={styles.titleText}>{t('businessSettings')}</Text>
          <Divider height={30} />
          <Text style={styles.descText}>{t('businessDesc')}</Text>
```

```
<Divider height={40} />

<ImageInput onSelect={setPhoto} imageUrl={imageUrl} />
<Divider height={20} />

<Input
  placeholder={t('name')}
  value={name}
  onChange={setName}
  isValid={isValidForm.name}
/>
<Divider height={20} />

<Input
  placeholder={t('currency')}
  value={currency}
  onChange={setCurrency}
  isValid={isValidForm.currency}
/>

<Divider height={40} />

<View style={styles.buttonWrapper}>
  <View style={styles.buttonWrapper2}>
    <Button
      text={t('submit')}
      onPress={() => {
        const formData = new FormData();
        formData.append('name', name);
        formData.append('currency', currency);



        formData.append('businessId', currentBusiness?.id);
        photo.path &&
          formData.append('image', {
            name: photo.filename,
            type: photo.mime ?? 'image/jpeg',
            uri: photo.path,
          } as any);
        validateForm(() =>
          dispatch(
```

```
                        updateBusiness(
                          formData,
                          () => {
                            Toast.show({
                              text1:
                                TOASTS[i18n.language].SUCCESS_UPDATE_BUSINESS,
                            });
                            navigation.navigate('NewOrder');
                          },

                          (error: string) => {
                            Toast.show({
                              text1: TOASTS[i18n.language].ERROR,
                              text2:
                                TOASTS[i18n.language][error] ??
                                'Unexpected error',
                              type: 'error',
                            });
                          },
                        ),
                      ),
                    );
                  }}
                />
              </View>

              <ActionButton
                iconName="delete"
                onPress={() => {
                  dispatch(
                    deleteBusiness(
                      currentBusiness?.id,
                      () => {
                        Toast.show({
                                                                    text1:
TOASTS[i18n.language].SUCCESS_DELETE_BUSINESS,
                        });
                        navigation.navigate('BusinessList');
                      },
                      (error: string) => {
                        Toast.show({
                          text1: TOASTS[i18n.language].ERROR,
```

```
                              text2:
                                       TOASTS[i18n.language][error] ?? 'Unexpected
error',
                              type: 'error',
                            });
                          },
                        ),
                      );
                    }}
                    size="large"
                  />
                </View>
              </View>
            </TouchableWithoutFeedback>
          </KeyboardAware>
        </SafeAreaView>
      );
    };

    const styles = StyleSheet.create({
      buttonWrapper2: { width: '70%' },
      createbuttonWrapper: { alignSelf: 'center' },
      area: { flex: 1 },
      container: { paddingHorizontal: 15, height: '100%' },
      buttonWrapper: {
        alignSelf: 'center',
        // position: 'absolute',
        flexDirection: 'row',
        justifyContent: 'space-between',
        alignItems: 'center',
        width: '90%',
        // marginTop: 20,
        // bottom: 10,
      },

      list: {
        marginTop: 20,
        height: '87%',
      },

      titleText: {
        fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',
```

```
    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
  descText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-Regular',

    color: '#000000',
    opacity: 0.4,
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { InventoryProps } from './types';
import { ProductCard } from './components/ProductCard';
import { useDispatch, useSelector } from 'react-redux';
import {
  deleteInventory,
  getInventoryList,
} from '../../store/slices/inventory';
```

```
import { useTranslation } from 'react-i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Inventory: React.FC<InventoryProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();

  const { inventory } = useSelector((store: any) => store.inventory);
  const { profile } = useSelector((store: any) => store.auth);

  const [inventoryList, setInventroyList] = useState<any>(null);
  useEffect(() => {
    setInventroyList(inventory);
  }, [inventory]);
  return (
    <SafeAreaView style={styles.area}>
      <View style={styles.container}>
        <Header />
        <Divider height={20} />
        <View style={styles.headerWrapper}>
          <Text style={styles.titleText}>{t('yourInventory')}</Text>
          <ActionButton
            iconName="plus"
            onPress={() => {
              if (true) {
                navigation.navigate('CreateInventory');
              } else {
                //TOAST
              }
            }}
            size="large"
          />
        </View>
        <Divider height={20} />

        <FlatList
          showsVerticalScrollIndicator={false}
          data={inventoryList ?? []}
          renderItem={({ item }) => (
```

```
                <ProductCard
                  title={item.name}
                  image={item?.image ?? ''}
                  price={item?.amount}
                  onEdit={() => {
                      navigation.navigate('CreateInventory', { ...item, edit: true
});
                  }}
                  onDelete={() => {
                    dispatch(
                      deleteInventory(
                        item?.id,
                        () => {
                          Toast.show({
                            text1: TOASTS[i18n.language].SUCCESS_DELETE_INVENTORY,
                          });
                        },
                        (error: string) => {
                          Toast.show({
                            text1: TOASTS[i18n.language].ERROR,
                            text2:
                              TOASTS[i18n.language][error] ?? 'Unexpected error',
                            type: 'error',
                          });
                        },
                      ) as any,
                    );
                  }}
                />
              )}
              style={styles.list}
            />
          </View>
        </SafeAreaView>
      );
    };

    const styles = StyleSheet.create({
      headerWrapper: {
        flexDirection: 'row',
        justifyContent: 'space-between',
        alignItems: 'center',
```

```
  },
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },

  list: {
    marginTop: 20,
    height: '83%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  ImageInput,
  Input,
  KeyboardAware,
} from '../../components';
import { useNavigation, useRoute } from '@react-navigation/native';
import { CreateInventoryProps } from './types';
```

```
import { useDispatch, useSelector } from 'react-redux';
import {
  createInventory,
  getInventoryList,
  updateInventory,
} from '../../store/slices/inventory';

import { useTranslation } from 'react-i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const CreateInventory: React.FC<CreateInventoryProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();
  const { params } = useRoute<any>();

  const { currentBusiness } = useSelector((store: any) => store.business);

  const onPressDismiss = () => {
    Keyboard.dismiss();
  };

  const [name, setName] = useState(params?.name ?? '');
  const [amount, setAmount] = useState(
    params?.amount ? params?.amount + '' : '',
  );
  const [lower, setLower] = useState(
    params?.lowerRange ? params?.lowerRange + '' : '',
  );
  const [photo, setPhoto] = useState<any>('');
  const [imageUrl, setImageUrl] = useState<any>(params?.image ?? '');
  const [isValidForm, setIsValidForm] = useState({
    name: true,
    lower: true,
    amount: true,
    photo: true,
  });

  const validateForm = (onSuccess: any) => {
    let isValid = true;
```

```
  if (name.length > 2 && name.length < 20) {
    setIsValidForm(prev => ({ ...prev, name: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, name: false }));
  }
  if (Number(amount) > 0) {
    setIsValidForm(prev => ({ ...prev, amount: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, amount: false }));
  }
  if (Number(lower) > 0) {
    setIsValidForm(prev => ({ ...prev, lower: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, lower: false }));
  }
  if (photo?.filename) {
    setIsValidForm(prev => ({ ...prev, photo: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, photo: false }));
  }

  if (isValid) {
    onSuccess();
  }
};

return (
  <SafeAreaView style={styles.area}>
    <KeyboardAware>
      <TouchableWithoutFeedback onPress={onPressDismiss}>
        <View style={styles.container}>
          <Header withGoBack />
          <Divider height={40} />

          <Text style={styles.titleText}>
            {params?.edit ? t('updateInventory') : t('createInventory')}
          </Text>
```

```
<Divider height={40} />

<ImageInput
  onSelect={setPhoto}
  imageUrl={imageUrl}
  isValid={isValidForm.photo}
/>
<Divider height={20} />

<Input
  placeholder={t('name')}
  onChange={setName}
  value={name}
  isValid={isValidForm.name}
/>
<Divider height={20} />
<Input
  placeholder={t('amount')}
  onChange={setAmount}
  value={amount}
  isValid={isValidForm.amount}
/>
<Divider height={20} />
<Input
  placeholder={t('lowerRange')}
  onChange={setLower}
  value={lower}
  isValid={isValidForm.lower}
/>
<Divider height={40} />



{/* <View style={styles.buttonWrapper}> */}
<Button
  text={params?.edit ? t('update') : t('submit')}
  onPress={() => {
    const formData = new FormData();
    formData.append('name', name);
    formData.append('lowerRange', lower);
    formData.append('amount', amount);
    formData.append('businessId', currentBusiness?.id);
```

```
params?.id && formData.append('inventoryId', params?.id);

photo.path &&
  formData.append('image', {
    name: photo.filename,
    type: photo.mime ?? 'image/jpeg',
    uri: photo.path,
  } as any);
if (params?.edit) {
  validateForm(() =>
    dispatch(
      updateInventory(
        formData,
        () => {
          Toast.show({
            text1:
              TOASTS[i18n.language].SUCCESS_UPDATE_INVENTORY,
          });
          navigation.navigate('Inventory');
        },
        (error: string) => {
          Toast.show({
            text1: TOASTS[i18n.language].ERROR,
            text2:
              TOASTS[i18n.language][error] ??
              'Unexpected error',
            type: 'error',
          });
        },
      ) as any,
    ),
  );
} else {
  validateForm(() =>
    dispatch(
      createInventory(
        formData,
        () => {
          Toast.show({
            text1:
              TOASTS[i18n.language].SUCCESS_CREATE_INVENTORY,
```

```
                        });
                        navigation.navigate('Inventory');
                      },
                      (error: string) => {
                        Toast.show({
                          text1: TOASTS[i18n.language].ERROR,
                          text2:
                            TOASTS[i18n.language][error] ??
                            'Unexpected error',
                          type: 'error',
                        });
                      },
                    ) as any,
                  ),
                );
              }
            }}
          />
          {/* </View> */}
        </View>
      </TouchableWithoutFeedback>
    </KeyboardAware>
  </SafeAreaView>
 );
};

const styles = StyleSheet.create({
 createbuttonWrapper: { alignSelf: 'center' },
 area: { flex: 1 },
 container: { paddingHorizontal: 15, height: '100%' },
 buttonWrapper: {
   alignSelf: 'center',
   marginTop: 20,
   // position: 'absolute',
   // bottom: 10,
 },

 list: {
   marginTop: 20,
   height: '87%',
 },
```

```
  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { CategoriesProps } from './types';
import { PromocodeCard } from './components/SupplierCard';
import { useDispatch, useSelector } from 'react-redux';

import {
  createCategory,
  deleteCategory,
  updateCategory,
} from '../../store/slices/products';
import { useTranslation } from 'react-i18next';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
```

```
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Categories: React.FC<CategoriesProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();

  const [open, setOpen] = useState(false);
  const [edit, setEdit] = useState(false);
  const [item, setItem] = useState<any>(null);

  const [name, setName] = useState('');

  const [isValidForm, setIsValidForm] = useState({
    name: true,
  });

  const validateForm = (onSuccess: any) => {
    let isValid = true;

    if (name.length > 2 && name.length < 20) {
      setIsValidForm(prev => ({ ...prev, name: true }));
    } else {
      isValid = false;
      setIsValidForm(prev => ({ ...prev, name: false }));
    }

    if (isValid) {
      onSuccess();
    }
  };

  const { categories } = useSelector((store: any) => store.products);
  const { currentBusiness } = useSelector((store: any) => store.business);

  const [categoriesList, setCategoriesList] = useState<any>(null);

  useEffect(() => {
    setCategoriesList(categories);
  }, [categories]);
```

```
return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header withGoBack />
      <Divider height={20} />
      <View style={styles.headerWrapper}>
        <Text style={styles.titleText}>{t('editCategory')}</Text>
        <ActionButton
          iconName="plus"
          onPress={() => {
            setOpen(true);
            setEdit(false);
            setName('');
          }}
          size="large"
        />
      </View>
      <Divider height={20} />

      <FlatList
        showsVerticalScrollIndicator={false}
        data={categoriesList ?? []}
        renderItem={({ item }) => (
          <PromocodeCard
            name={item?.name}
            phone={''}
            onDelete={() => {
              dispatch(
                deleteCategory(
                  item?.id,
                  () => {
                    Toast.show({
                      text1: TOASTS[i18n.language].SUCCESS_DELETE_CATEGORY,
                    });
                  },
                  (error: string) => {
                    Toast.show({
                      text1: TOASTS[i18n.language].ERROR,
                      text2:
                        TOASTS[i18n.language][error] ?? 'Unexpected error',
                      type: 'error',
                    });
```

```
              },
            ) as any,
          );
        }}
        onEdit={() => {
          setItem(item);
          setEdit(true);
          setName(item.name);
          setOpen(true);
        }}
      />
    )}
    style={styles.list}
  />
</View>
<BottomSheet
  open={open}
  snapPoints={['30%']}
  onDismiss={() => {
    setOpen(false);
  }}>
  <Input
    placeholder={t('name')}
    value={name}
    onChange={setName}
    inBottomSheet
    isValid={isValidForm.name}
  />
  <Divider height={20} />



  <Button
    text={edit ? t('update') : t('submit')}
    mode="large"
    onPress={() => {
      if (edit) {
        const formData = new FormData();

        formData.append('name', name);
        formData.append('categoryId', item?.id);
        validateForm(() =>
```

```
      dispatch(
        updateCategory(
          formData,
          () => {
            Toast.show({
              text1: TOASTS[i18n.language].SUCCESS_UPDATE_CATEGORY,
            });
            setOpen(false);
          },
          (error: string) => {
            Toast.show({
              text1: TOASTS[i18n.language].ERROR,
              text2:
                TOASTS[i18n.language][error] ?? 'Unexpected error',
              type: 'error',
            });
          },
        ) as any,
      ),
    );
  } else {
    const formData = new FormData();

    formData.append('name', name);
    formData.append('businessId', currentBusiness?.id);
    validateForm(() =>
      dispatch(
        createCategory(
          formData,
          () => {
            Toast.show({
              text1: TOASTS[i18n.language].SUCCESS_CREATE_CATEGORY,
            });
            setOpen(false);
          },
          (error: string) => {
            Toast.show({
              text1: TOASTS[i18n.language].ERROR,
              text2:
                TOASTS[i18n.language][error] ?? 'Unexpected error',
              type: 'error',
            });
```

```
                    },
                  ) as any,
                ),
              );
            }
            setItem(null);
          }}
        />
      </BottomSheet>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  headerWrapper: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },

  list: {
    marginTop: 20,
    height: '77%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
```

```
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { SuppliersProps } from './types';
import { SupplierCard } from './components/SupplierCard';
import { useDispatch, useSelector } from 'react-redux';
import {
  createSupplier,
  deleteSupplier,
  getSuppliersList,
  updateSupplier,
} from '../../store/slices/suppliers';

import { useTranslation } from 'react-i18next';
import { phoneReg } from '../../store/config';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Suppliers: React.FC<SuppliersProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();

  const [open, setOpen] = useState(false);
  const [edit, setEdit] = useState(false);
  const [item, setItem] = useState<any>(null);

  const [name, setName] = useState('');
```

```
const [phone, setPhone] = useState('');

const [isValidForm, setIsValidForm] = useState({
  name: true,
  phone: true,
});

const validateForm = (onSuccess: any) => {
  let isValid = true;

  if (name.length > 2) {
    setIsValidForm(prev => ({ ...prev, name: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, name: false }));
  }

  if (phoneReg.test(phone)) {
    setIsValidForm(prev => ({ ...prev, phone: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, phone: false }));
  }

  if (isValid) {
    onSuccess();
  }
};

const { suppliers } = useSelector((store: any) => store.suppliers);
const { profile } = useSelector((store: any) => store.auth);

const { currentBusiness } = useSelector((store: any) => store.business);

const [suppliersList, setSuppliersList] = useState<any>(null);

useEffect(() => {
  setSuppliersList(suppliers);
}, [suppliers]);

return (
  <SafeAreaView style={styles.area}>
```

```
<View style={styles.container}>
  <Header />
  <Divider height={20} />
  <View style={styles.headerWrapper}>
    <Text style={styles.titleText}>{t('yourSuppliers')}</Text>
    <ActionButton
      iconName="plus"
      onPress={() => {
        if (true) {
          setOpen(true);
          setEdit(false);
          setName('');
          setPhone('');
        } else {
          //TOAST
        }
      }}
      size="large"
    />
  </View>
  <Divider height={20} />



  <FlatList
    showsVerticalScrollIndicator={false}
    data={suppliersList ?? []}
    renderItem={({ item }) => (
      <SupplierCard
        name={item.name}
        phone={item.contact}
        onDelete={() => {
          dispatch(
            deleteSupplier(
              item?.id,
              () => {
                Toast.show({
                  text1: TOASTS[i18n.language].SUCCESS_DELETE_SUPPLIER,
                });
              },
              (error: string) => {
                Toast.show({
```

```
                    text1: TOASTS[i18n.language].ERROR,
                    text2:
                      TOASTS[i18n.language][error] ?? 'Unexpected error',
                    type: 'error',
                  });
                },
              ) as any,
            );
          }}
          onEdit={() => {
            setItem(item);
            setEdit(true);
            setName(item.name);
            setPhone(item.contact);
            setOpen(true);
          }}
        />
      )}
      style={styles.list}
    />
  </View>
  <BottomSheet
    open={open}
    snapPoints={Platform.OS === 'ios' ? ['40%'] : ['50%']}
    onDismiss={() => {
      setOpen(false);
    }}>
    <Input
      placeholder={t('name')}
      value={name}
      onChange={setName}
      inBottomSheet
      isValid={isValidForm.name}
    />
    <Divider height={20} />

    <Input
      placeholder={t('phone')}
      value={phone}
      onChange={setPhone}
      inBottomSheet
      isValid={isValidForm.phone}
```

```
/>
<Divider height={30} />
<Button
  text={edit ? t('update') : t('submit')}
  mode="large"
  onPress={() => {
    if (edit) {
      validateForm(() =>
        dispatch(
          updateSupplier(
            {
              name,
              contact: phone,
              supplierId: item?.id,
            },
            () => {
              Toast.show({
                text1: TOASTS[i18n.language].SUCCESS_UPDATE_SUPPLIER,
              });
              setOpen(false);
            },
            (error: string) => {
              Toast.show({
                text1: TOASTS[i18n.language].ERROR,
                text2:
                  TOASTS[i18n.language][error] ?? 'Unexpected error',
                type: 'error',
              });
            },
          ) as any,
        ),
      );
    } else {
      validateForm(() =>
        dispatch(
          createSupplier(
            {
              name,
              contact: phone,
              businessId: currentBusiness?.id,
            },
            () => {
```

```
                    Toast.show({
                      text1: TOASTS[i18n.language].SUCCESS_CREATE_SUPPLIER,
                    });
                    setOpen(false);
                  },
                  (error: string) => {
                    Toast.show({
                      text1: TOASTS[i18n.language].ERROR,
                      text2:
                        TOASTS[i18n.language][error] ?? 'Unexpected error',
                      type: 'error',
                    });
                  },
                ) as any,
              ),
            );
          }
          setItem(null);
        }}
      />
    </BottomSheet>
  </SafeAreaView>
);
};

const styles = StyleSheet.create({
  headerWrapper: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },



  list: {
    marginTop: 20,
    height: '77%',
  },

  titleText: {
```

```
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  ImageInput,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { WorkersProps } from './types';
import { WorkerCard } from './components/WorkerCard';
import { useTranslation } from 'react-i18next';
import { useDispatch, useSelector } from 'react-redux';
import { addUser, deleteUser } from '../../store/slices/business';
import { Platform } from 'react-native';
import { emailReg } from '../../store/config';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Workers: React.FC<WorkersProps> = () => {
```

```
const navigation = useNavigation<any>();
const dispatch = useDispatch<any>();

const { t } = useTranslation();

const [open, setOpen] = useState(false);
const [name, setName] = useState('');
const [email, setEmail] = useState('');

const [isValidForm, setIsValidForm] = useState({
  name: true,
  email: true,
});

const validateForm = (onSuccess: any) => {
  let isValid = true;

  if (name.length > 2) {
    setIsValidForm(prev => ({ ...prev, name: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, name: false }));
  }

  if (emailReg.test(email)) {
    setIsValidForm(prev => ({ ...prev, email: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, email: false }));
  }

  if (isValid) {
    onSuccess();
  }
};

const { currentBusiness } = useSelector((store: any) => store.business);
const { profile } = useSelector((store: any) => store.auth);

const [workerList, setWorkerList] = useState<any>(null);

useEffect(() => {
```

```
  setWorkerList(currentBusiness?.workers);
}, [currentBusiness]);

return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header />
      <Divider height={20} />
      <View style={styles.headerWrapper}>
        <Text style={styles.titleText}>{t('yourWorkers')}</Text>
        <ActionButton
          iconName="plus"
          onPress={() => {
            if (true) {
              setOpen(true);
            } else {
              //TOAST
            }
          }}
          size="large"
        />
      </View>
      <Divider height={20} />

      <FlatList
        showsVerticalScrollIndicator={false}
        data={workerList}
        renderItem={({ item }) => (
          <WorkerCard
            name={item.name}
            email={item.email}
            onDelete={() => {
              dispatch(
                deleteUser(
                  {
                    businessId: currentBusiness?.id,
                    email: item.email,
                  },
                  () => {
                    Toast.show({
                      text1: TOASTS[i18n.language].SUCCESS_DELETE_USER,
                    });
```

```
              },
              (error: string) => {
                Toast.show({
                  text1: TOASTS[i18n.language].ERROR,
                  text2:
                    TOASTS[i18n.language][error] ?? 'Unexpected error',
                  type: 'error',
                });
              },
            ) as any,
          );
        }}
      />
    )}
    style={styles.list}
  />
</View>
<BottomSheet
  open={open}
  snapPoints={Platform.OS === 'ios' ? ['40%'] : ['50%']}
  onDismiss={() => {
    setOpen(false);
  }}>
  <Input
    placeholder={t('name')}
    inBottomSheet
    onChangeText={setName}
    isValid={isValidForm.name}
  />
  <Divider height={20} />



  <Input
    placeholder={t('email')}
    inBottomSheet
    onChangeText={setEmail}
    isValid={isValidForm.email}
  />
  <Divider height={30} />
  <Button
    text={t('submit')}
```

```
              mode="large"
              onPress={() => {
                validateForm(() =>
                  dispatch(
                    addUser(
                      {
                        businessId: currentBusiness?.id,
                        name,
                        email,
                      },
                      () => {
                        Toast.show({
                          text1: TOASTS[i18n.language].SUCCESS_CREATE_USER,
                        });
                      },
                      (error: string) => {
                        Toast.show({
                          text1: TOASTS[i18n.language].ERROR,
                                text2: TOASTS[i18n.language][error] ?? 'Unexpected
error',
                          type: 'error',
                        });
                      },
                    ),
                  ),
                );

                setOpen(false);
              }}
            />
          </BottomSheet>
        </SafeAreaView>
      );
    };

    const styles = StyleSheet.create({
      headerWrapper: {
        flexDirection: 'row',
        justifyContent: 'space-between',
        alignItems: 'center',
      },
      area: { flex: 1 },
```

```
  container: { paddingHorizontal: 15 },

  list: {
    marginTop: 20,
    height: '77%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
  Icon,
  ImageInput,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { WorkersProps } from './types';
import { WorkerCard } from './components/WorkerCard';
import { useTranslation } from 'react-i18next';
import { useDispatch, useSelector } from 'react-redux';
```

```
import { addUser, deleteUser } from '../../store/slices/business';
import { Platform } from 'react-native';
import { emailReg } from '../../store/config';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Workers: React.FC<WorkersProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch<any>();

  const { t } = useTranslation();

  const [open, setOpen] = useState(false);
  const [name, setName] = useState('');
  const [email, setEmail] = useState('');

  const [isValidForm, setIsValidForm] = useState({
    name: true,
    email: true,
  });

  const validateForm = (onSuccess: any) => {
    let isValid = true;

    if (name.length > 2) {
      setIsValidForm(prev => ({ ...prev, name: true }));
    } else {
      isValid = false;
      setIsValidForm(prev => ({ ...prev, name: false }));
    }

    if (emailReg.test(email)) {
      setIsValidForm(prev => ({ ...prev, email: true }));
    } else {
      isValid = false;
      setIsValidForm(prev => ({ ...prev, email: false }));
    }

    if (isValid) {
      onSuccess();
    }
```

```
};

const { currentBusiness } = useSelector((store: any) => store.business);
const { profile } = useSelector((store: any) => store.auth);

const [workerList, setWorkerList] = useState<any>(null);

useEffect(() => {
  setWorkerList(currentBusiness?.workers);
}, [currentBusiness]);

return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header />
      <Divider height={20} />
      <View style={styles.headerWrapper}>
        <Text style={styles.titleText}>{t('yourWorkers')}</Text>
        <ActionButton
          iconName="plus"
          onPress={() => {
            if (true) {
              setOpen(true);
            } else {
              //TOAST
            }
          }}
          size="large"
        />
      </View>
      <Divider height={20} />

      <FlatList
        showsVerticalScrollIndicator={false}
        data={workerList}
        renderItem={({ item }) => (
          <WorkerCard
            name={item.name}
            email={item.email}
            onDelete={() => {
              dispatch(
                deleteUser(
```

```
              {
                businessId: currentBusiness?.id,
                email: item.email,
              },
              () => {
                Toast.show({
                  text1: TOASTS[i18n.language].SUCCESS_DELETE_USER,
                });
              },
              (error: string) => {
                Toast.show({
                  text1: TOASTS[i18n.language].ERROR,
                  text2:
                    TOASTS[i18n.language][error] ?? 'Unexpected error',
                  type: 'error',
                });
              },
            ) as any,
          );
        }}
      />
    )}
    style={styles.list}
  />
</View>
<BottomSheet
  open={open}
  snapPoints={Platform.OS === 'ios' ? ['40%'] : ['50%']}
  onDismiss={() => {
    setOpen(false);
  }}>
  <Input
    placeholder={t('name')}
    inBottomSheet
    onChangeText={setName}
    isValid={isValidForm.name}
  />
  <Divider height={20} />



  <Input
```

```
            placeholder={t('email')}
            inBottomSheet
            onChangeText={setEmail}
            isValid={isValidForm.email}
          />
          <Divider height={30} />
          <Button
            text={t('submit')}
            mode="large"
            onPress={() => {
              validateForm(() =>
                dispatch(
                  addUser(
                    {
                      businessId: currentBusiness?.id,
                      name,
                      email,
                    },
                    () => {
                      Toast.show({
                        text1: TOASTS[i18n.language].SUCCESS_CREATE_USER,
                      });
                    },
                    (error: string) => {
                      Toast.show({
                        text1: TOASTS[i18n.language].ERROR,
                              text2: TOASTS[i18n.language][error] ?? 'Unexpected
error',
                        type: 'error',
                      });
                    },
                  ),
                ),
              );

              setOpen(false);
            }}
          />
        </BottomSheet>
      </SafeAreaView>
    );
  };
```

```
const styles = StyleSheet.create({
  headerWrapper: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },

  list: {
    marginTop: 20,
    height: '77%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});

import React, { useEffect, useState } from 'react';
import {
  FlatList,
  Keyboard,
  Platform,
  SafeAreaView,
  StyleSheet,
  Text,
  TouchableOpacity,
  TouchableWithoutFeedback,
  View,
} from 'react-native';
import {
  ActionButton,
  BottomSheet,
  Button,
  Divider,
  Header,
```

```
  Icon,
  Input,
} from '../../components';
import { useNavigation } from '@react-navigation/native';
import { SuppliersProps } from './types';
import { SupplierCard } from './components/SupplierCard';
import { useDispatch, useSelector } from 'react-redux';
import {
  createSupplier,
  deleteSupplier,
  getSuppliersList,
  updateSupplier,
} from '../../store/slices/suppliers';

import { useTranslation } from 'react-i18next';
import { phoneReg } from '../../store/config';
import { Toast } from 'react-native-toast-message/lib/src/Toast';
import { TOASTS } from '../../i18n/toasts';
import i18n from '../../i18n';

export const Suppliers: React.FC<SuppliersProps> = () => {
  const navigation = useNavigation<any>();
  const dispatch = useDispatch();
  const { t } = useTranslation();

  const [open, setOpen] = useState(false);
  const [edit, setEdit] = useState(false);
  const [item, setItem] = useState<any>(null);

  const [name, setName] = useState('');
  const [phone, setPhone] = useState('');

  const [isValidForm, setIsValidForm] = useState({
    name: true,
    phone: true,
  });

  const validateForm = (onSuccess: any) => {
    let isValid = true;

    if (name.length > 2) {
      setIsValidForm(prev => ({ ...prev, name: true }));
```

```
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, name: false }));
  }

  if (phoneReg.test(phone)) {
    setIsValidForm(prev => ({ ...prev, phone: true }));
  } else {
    isValid = false;
    setIsValidForm(prev => ({ ...prev, phone: false }));
  }

  if (isValid) {
    onSuccess();
  }
};

const { suppliers } = useSelector((store: any) => store.suppliers);
const { profile } = useSelector((store: any) => store.auth);

const { currentBusiness } = useSelector((store: any) => store.business);

const [suppliersList, setSuppliersList] = useState<any>(null);

useEffect(() => {
  setSuppliersList(suppliers);
}, [suppliers]);

return (
  <SafeAreaView style={styles.area}>
    <View style={styles.container}>
      <Header />
      <Divider height={20} />
      <View style={styles.headerWrapper}>
        <Text style={styles.titleText}>{t('yourSuppliers')}</Text>
        <ActionButton
          iconName="plus"
          onPress={() => {
            if (true) {
              setOpen(true);
              setEdit(false);
              setName('');
```

```
        setPhone('');
      } else {
        //TOAST
      }
    }}
    size="large"
  />
</View>
<Divider height={20} />



<FlatList
  showsVerticalScrollIndicator={false}
  data={suppliersList ?? []}
  renderItem={({ item }) => (
    <SupplierCard
      name={item.name}
      phone={item.contact}
      onDelete={() => {
        dispatch(
          deleteSupplier(
            item?.id,
            () => {
              Toast.show({
                text1: TOASTS[i18n.language].SUCCESS_DELETE_SUPPLIER,
              });
            },
            (error: string) => {
              Toast.show({
                text1: TOASTS[i18n.language].ERROR,
                text2:
                  TOASTS[i18n.language][error] ?? 'Unexpected error',
                type: 'error',
              });
            },
          ) as any,
        );
      }}
      onEdit={() => {
        setItem(item);
        setEdit(true);
```

```
            setName(item.name);
            setPhone(item.contact);
            setOpen(true);
          }}
        />
      )}
      style={styles.list}
    />
  </View>
  <BottomSheet
    open={open}
    snapPoints={Platform.OS === 'ios' ? ['40%'] : ['50%']}
    onDismiss={() => {
      setOpen(false);
    }}>
    <Input
      placeholder={t('name')}
      value={name}
      onChange={setName}
      inBottomSheet
      isValid={isValidForm.name}
    />
    <Divider height={20} />

    <Input
      placeholder={t('phone')}
      value={phone}
      onChange={setPhone}
      inBottomSheet
      isValid={isValidForm.phone}
    />
    <Divider height={30} />
    <Button
      text={edit ? t('update') : t('submit')}
      mode="large"
      onPress={() => {
        if (edit) {
          validateForm(() =>
            dispatch(
              updateSupplier(
                {
                  name,
```

```
            contact: phone,
            supplierId: item?.id,
          },
          () => {
            Toast.show({
              text1: TOASTS[i18n.language].SUCCESS_UPDATE_SUPPLIER,
            });
            setOpen(false);
          },
          (error: string) => {
            Toast.show({
              text1: TOASTS[i18n.language].ERROR,
              text2:
                TOASTS[i18n.language][error] ?? 'Unexpected error',
              type: 'error',
            });
          },
        ) as any,
      ),
    );
  } else {
    validateForm(() =>
      dispatch(
        createSupplier(
          {
            name,
            contact: phone,
            businessId: currentBusiness?.id,
          },
          () => {
            Toast.show({
              text1: TOASTS[i18n.language].SUCCESS_CREATE_SUPPLIER,
            });
            setOpen(false);
          },
          (error: string) => {
            Toast.show({
              text1: TOASTS[i18n.language].ERROR,
              text2:
                TOASTS[i18n.language][error] ?? 'Unexpected error',
              type: 'error',
            });
```

```
                    },
                  ) as any,
                ),
              );
            }
            setItem(null);
          }}
        />
      </BottomSheet>
    </SafeAreaView>
  );
};

const styles = StyleSheet.create({
  headerWrapper: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    alignItems: 'center',
  },
  area: { flex: 1 },
  container: { paddingHorizontal: 15 },



  list: {
    marginTop: 20,
    height: '77%',
  },

  titleText: {
    fontFamily: Platform.OS === 'ios' ? 'Montserrat' : 'Montserrat-SemiBold',

    fontSize: 28,
    fontWeight: '600',
    color: '#000000',
  },
});
```