

Національний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

КУРСОВА РОБОТА

з дисципліни «Веб-технології та веб-дизайн»

Тема «Розробка веб-застосунку для автоматизації замовлень піци»

Студента 3 курсу AI-221 групи
Спеціальності 122 – «Комп'ютерні науки»

Семиволос Л.Ю.

(прізвище та ініціали)

Керівник доцент, Червоненко П. П.

(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____

Кількість балів: _____

Оцінка: ECTS _____

Члени комісії

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

(підпис)

(прізвище та ініціали)

Національний університет «Одеська політехніка»
Інститут комп'ютерних систем
Кафедра інформаційних систем

ЗАВДАННЯ
НА КУРСОВУ РОБОТУ

студенту Семиволоса Лева Юрійовича

група AI-221

1. Тема роботи
«Розробка веб-застосунку для автоматизації замовлень піци»
2. Термін здачі студентом закінченої роботи 11.12.2024
3. Зміст розрахунково-пояснювальної записки (перелік питань, які належить розробити)
Вступ, актуальність та аналіз аналогів, проектування інформаційної системи, реалізація інформаційної системи, тестування додатку
4. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Діаграма прикладів використання, діаграми послідоності, діаграми активності, ER-діаграма, діаграма класів, скриншоти заповнених таблиць

Додаток А (код продукту)

Завдання видано

(підпис викладача)

Завдання прийнято до виконання

(підпис студента)

АНОТАЦІЯ

Розглянуто процес створення веб-додатку для піцерії, який забезпечує зручне управління замовленнями, перегляд меню та роботу з профілями користувачів.

Запропоновано сучасну архітектуру, яка використовує Java Spring Boot на серверній стороні та React на клієнтській.

Розроблено REST API для взаємодії з веб-сервісом, сервіси для реалізації бізнес-логіки, Entity-класи та репозиторії для роботи з базою даних.

Реалізовано функціонал для аутентифікації, авторизації, перегляду замовлень, керування профілем користувача та створення нових замовлень.

ABSTRACT

The process of creating a web application for a pizzeria that provides convenient order management, menu browsing and working with user profiles is considered.

A modern architecture is proposed, which uses Java Spring Boot on the server side and React on the client side.

Developed REST API for interacting with the web service, services for implementing business logic, Entity classes and repositories for working with the database.

Functionality for authentication, authorization, viewing orders, managing user profile and creating new orders has been implemented.

ЗМІСТ

ВСТУП	5
1 Актуальність, аналіз аналогів	7
1.1 Аналіз предметної області	7
1.2 Огляд аналогів	9
2 Проектування інформаційної системи	11
2.1 User Stories	11
2.2 Функціональні вимоги	12
2.3 Моделювання діаграм	14
2.4 Опис стеку технологій	20
3 Реалізація інформаційної системи	22
3.1 Реалізація серверної частини	22
3.2 Реалізація клієнтської частини	27
4 Тестування додатку	34
4.1. Цілі та завдання тестування	34
4.2. Методи та інструменти тестування	34
4.3. Тестування API	35
4.4. Тестування інтерфейсу користувача	44
Висновки	45
Перелік використаних джерел	47

ВСТУП

Метою даної курсової роботи є закріплення та поглиблення знань, отриманих у курсі "Розробка веб-додатків", шляхом створення функціонального веб-додатку для управління замовленнями піцерії. Робота спрямована на розробку багатoshарової архітектури, яка використовує сучасні технології для забезпечення ефективності, безпеки та зручності використання.

Веб-додаток побудований за клієнт-серверною архітектурою, що є одним із найпоширеніших підходів у сучасній веб-розробці. Клієнтська частина, створена за допомогою React, відповідає за взаємодію з користувачем і забезпечує зручний інтерфейс для перегляду меню, оформлення замовлень, управління профілем і перегляду історії замовлень. Серверна частина, реалізована на основі Java Spring Boot, виконує роль обробки бізнес-логіки, роботи з базою даних та забезпечення безпеки через використання JWT для аутентифікації та авторизації.

Розробка веб-додатку включає такі ключові етапи:

- Проектування архітектури, яка передбачає розподіл відповідальностей між клієнтською і серверною частинами.
- Реалізацію функціональності RESTful API для взаємодії клієнта з сервером.
- Інтеграцію клієнтської частини з серверною для забезпечення безперебійного обміну даними.
- Забезпечення адаптивного дизайну інтерфейсу для коректного відображення на різних пристроях.
- Реалізацію функцій управління замовленнями, включаючи створення, перегляд та скасування замовлень.

Для розробки клієнтської частини було використано React, що дозволяє створювати інтерактивні односторінкові додатки (SPA), використовуючи React Router для маршрутизації та Context API для управління станом. Серверна частина реалізована за допомогою Spring Boot, який забезпечує роботу з базою даних, бізнес-логіку та механізми безпеки.

Клієнт-серверна архітектура є однією з ключових концепцій сучасного програмного забезпечення, яка визначає взаємодію між клієнтськими та серверними компонентами через мережу. Цей підхід широко використовується у веб-додатках, оскільки забезпечує ефективну обробку запитів, модульність та масштабованість систем. Основна ідея клієнт-серверної архітектури полягає в розподілі обов'язків між компонентами: сервер виконує обчислення, обробляє запити та керує даними, тоді як клієнтська частина відповідає за взаємодію з користувачем і відображення інформації.

Серверна частина, створена за допомогою Java Spring Boot, забезпечує швидкий старт проекту завдяки інтеграції з такими технологіями, як JPA для управління базами даних, Spring Security для реалізації аутентифікації та авторизації, а також зручними механізмами для реалізації RESTful API. Java Spring Boot є одним із найпопулярніших фреймворків для серверної розробки завдяки своїй продуктивності, гнучкості та підтримці великої кількості інструментів.

Завдяки використанню клієнт-серверної архітектури та сучасних технологій, веб-додаток забезпечує:

- Масштабованість: можливість обробки великої кількості користувачів та запитів.
- Безпеку: використання JWT для аутентифікації та шифрування конфіденційних даних.
- Модульність: чіткий поділ логіки між клієнтською і серверною частинами.
- Зручність використання: адаптивний дизайн інтерфейсу, що дозволяє працювати з додатком на різних пристроях.

1 Актуальність, аналіз аналогів

1.1 Аналіз предметної області

Розвиток сучасних технологій та масове використання Інтернету створюють значні можливості для автоматизації бізнес-процесів у сфері обслуговування. Однією з найбільш популярних галузей, що активно використовує інформаційні системи, є ресторанний бізнес. Зокрема, служби доставки їжі та піцерії мають особливу потребу в ефективних та зручних для користувачів рішеннях. Основними проблемами, які виникають у цій сфері, є складність управління замовленнями, забезпечення швидкого доступу до меню, підтримка взаємодії з клієнтами, а також мінімізація помилок, що виникають у процесі обробки замовлень.

Існуючі системи часто не відповідають сучасним вимогам користувачів щодо інтуїтивного інтерфейсу, адаптивності до різних пристроїв та швидкості роботи. Також проблемою є обмежена інтеграція з іншими сервісами, такими як платіжні системи або модулі аналізу даних. У контексті локальних бізнесів, таких як невеликі піцерії, виклики стають ще гострішими, оскільки готові рішення часто є дорогими або складними для впровадження.

Відсутність інтерактивного веб-додатку, який би ефективно поєднував клієнтську та серверну частини, ускладнює роботу як для клієнтів, так і для персоналу. Наприклад, клієнти можуть відчувати незручності при оформленні замовлення, якщо сайт має повільний інтерфейс або складну навігацію. Для персоналу труднощі можуть виникати через відсутність автоматизованого обліку замовлень або помилки в обробці даних.

1.1.1 Опис проблеми, яку вирішує додаток

Розроблений веб-додаток для управління замовленнями піцерії спрямований на вирішення ключових проблем, які виникають у процесі взаємодії клієнтів із сервісом доставки. Основними завданнями додатку є:

Автоматизація процесу замовлення: зменшення ручної роботи завдяки інтегрованій системі оформлення та обробки замовлень.

Покращення зручності користувача: створення інтуїтивно зрозумілого інтерфейсу, який дозволяє легко переглядати меню, обирати параметри пицці та відслідковувати статус замовлення.

Оптимізація роботи персоналу: надання інструментів для швидкого доступу до інформації про замовлення, їх поточний статус та потреби клієнтів.

Мінімізація помилок: забезпечення коректності даних через інтеграцію із системою обліку замовлень та перевірку інформації на рівні серверу.

Додаток дозволяє клієнтам не лише оформлювати замовлення, але й переглядати історію попередніх замовлень, що сприяє збільшенню лояльності клієнтів. З іншого боку, менеджери піцерії можуть аналізувати дані, покращуючи якість обслуговування та адаптуючи свої послуги до потреб користувачів.

1.1.2 Опис актуальності обраної теми

Актуальність розробки веб-додатку для управління замовленнями піцерії обумовлена кількома факторами. По-перше, у сучасному світі все більше людей надають перевагу онлайн-сервісам замість традиційних методів замовлення. Це зумовлено потребою в економії часу, доступністю послуг 24/7 та можливістю персоналізації замовлень. По-друге, конкурентоспроможність ресторанного бізнесу значною мірою залежить від ефективності використання цифрових технологій, які дозволяють швидше адаптуватися до змін ринку та потреб клієнтів.

Окрім цього, впровадження автоматизованих систем сприяє підвищенню ефективності роботи бізнесу, скороченню операційних витрат та зменшенню людського фактору в процесі обробки замовлень. Таким чином, створення сучасного, функціонального та зручного веб-додатку для піцерії є актуальним завданням, яке відповідає вимогам ринку та потребам користувачів.

1.2 Огляд аналогів

На сучасному ринку веб-додатків існує безліч платформ, які надають можливість управління замовленнями у сфері громадського харчування, зокрема піцерій. Розглянемо основні приклади таких систем, їх переваги та недоліки.

1.2.1 Системи для замовлення їжі

Серед найбільш популярних платформ для замовлення їжі можна виділити такі, як Glovo, Uber Eats, Wolt та інші. Ці сервіси дозволяють користувачам переглядати меню різних ресторанів, оформлювати замовлення та здійснювати оплату онлайн.

Переваги таких платформ:

- Зручність використання: простий та інтуїтивний інтерфейс, що дозволяє швидко знайти необхідний заклад або страву.
- Широкий вибір: можливість замовлення їжі з різних ресторанів у межах одного додатку.
- Інтеграція з платіжними системами: забезпечує зручний спосіб оплати замовлення.

Недоліки:

- Висока комісія: більшість платформ стягують значну комісію з ресторанів, що може підвищувати кінцеву вартість замовлення для користувача.
- Обмежені можливості для індивідуалізації: такі сервіси не дозволяють ресторанам реалізовувати унікальні функції, які могли б покращити взаємодію з клієнтами.
- Відсутність прямого контакту: клієнт не може напряду взаємодіяти з рестораном, що іноді ускладнює уточнення деталей замовлення.

1.2.2 Сайти індивідуальних піцерій

Багато піцерій мають власні веб-сайти, такі як Domino's Pizza, Papa John's, Pizza Hut тощо. Ці платформи дозволяють клієнтам замовляти піцу, обираючи розмір, тип тіста, додаткові інгредієнти та інші параметри.

Переваги таких сайтів:

- Індивідуалізація замовлення: можливість гнучко налаштувати замовлення під потреби клієнта.
- Брендова ідентичність: унікальний дизайн сайту дозволяє компаніям виділятися на фоні конкурентів.
- Прямий зв'язок із клієнтами: відсутність посередників забезпечує прозорість взаємодії.

Недоліки:

- Відсутність універсальності: користувач може замовляти їжу лише з одного закладу, що обмежує його вибір.
- Застарілі технології: деякі сайти мають повільний інтерфейс або недостатньо адаптивний дизайн, що ускладнює використання на мобільних пристроях.
- Недостатня інтеграція: у багатьох випадках відсутня інтеграція з популярними платіжними сервісами або системами управління замовленнями.

2 Проектування інформаційної системи

Процес проектування інформаційних систем є ключовим етапом у розробці програмного забезпечення, оскільки саме на цьому етапі визначаються структура системи, функціональність її компонентів і взаємодія між ними. Одним із ефективних інструментів для розробки користувацьких сценаріїв і визначення вимог до програмного забезпечення є підхід, заснований на User Stories (користувацьких історіях). Цей підхід дозволяє описати потреби користувачів і забезпечити їхнє задоволення шляхом розробки відповідного функціоналу.

User Stories описують конкретні дії, які користувачі можуть виконувати в системі, а також результати, які вони очікують отримати. Цей метод дозволяє уникнути розмитих формулювань і сфокусуватися на реальних задачах, які вирішує програмне забезпечення. У контексті даної курсової роботи User Stories слугують основою для визначення функціональних вимог до веб-додатку для піцерії.

2.1 User Stories

Для гостей (Guest):

- US1: Як гість, я хочу переглядати меню піц, щоб дізнатися, які варіанти доступні, перш ніж реєструватися.
- US2: Як гість, я хочу додати піци до кошика, щоб підготувати замовлення.
- US3: Як гість, я хочу налаштовувати розмір піци та тип тіста, щоб піца відповідала моїм вподобанням.
- US4: Як гість, я хочу ввести свої дані для доставки під час оформлення замовлення і зробити замовлення без обов'язкової реєстрації.
- US5: Як гість, я хочу бачити орієнтовний час доставки після оформлення замовлення, щоб знати, коли його очікувати.

Для зареєстрованих користувачів (Registered User, Customer):

- US6: Як клієнт, я хочу зберігати свою адресу доставки та контактні дані, щоб замовлення оформлялися швидше в майбутньому.
- US7: Як клієнт, я хочу переглядати історію своїх замовлень, щоб повторно замовляти улюблені піци або відстежувати поточні замовлення.
- US8: Як клієнт, я хочу входити в систему і залишатися автентифікованим під час перегляду, додавання товарів до кошика та оформлення замовлення, щоб отримати персоналізований досвід.

Для адміністратора (Admin):

- US9: Як адміністратор, я хочу додавати нові піци до меню, щоб меню завжди залишалось актуальним.
- US10: Як адміністратор, я хочу оновлювати статуси замовлень, щоб клієнти були поінформовані про їхній прогрес.
- US11: Як адміністратор, я хочу редагувати або видаляти піци з меню, щоб ефективно управляти життєвим циклом продуктів.

2.2 Функціональні вимоги

Функціональні вимоги визначають основну функціональність, яку система повинна забезпечити, щоб задовольнити потреби користувачів. Ці вимоги розподілено за ключовими модулями, такими як управління користувачами, управління меню, кошик і оформлення замовлень, локація магазинів, інтерфейс користувача та системна безпека. Наведена таблиця структуровано описує всі вимоги для ефективної розробки системи.

Таблиця 2.1 – опис функцій з наданням унікальних ієрархічних ідентифікаторів.

FR	Текст
FR1	Управління користувачами
FR1.1	Реєстрація користувачів

FR1.1.1	Користувач може зареєструватися, надавши дійсну електронну пошту та пароль.
FR1.1.2	Користувач повинен вказати ім'я, електронну пошту та пароль під час реєстрації.
FR1.1.3	Система повинна хешувати та безпечно зберігати паролі користувачів.
FR1.1.4	Після успішної реєстрації користувач отримує підтвердження на екрані.
FR1.2	Вхід користувача (з використанням JWT)
FR1.2.1	Користувач може увійти до системи, використовуючи свою електронну пошту та пароль.
FR1.2.2	Система видає JWT-токен після успішного входу.
FR1.2.3	JWT-токен має бути включений у заголовок авторизації для доступу до захищених запитів.
FR1.2.4	У разі невдалого входу (наприклад, невірний пароль) користувач бачить повідомлення про помилку.
FR1.3	Управління профілем користувача
FR1.3.1	Авторизований користувач може переглядати та редагувати свою особисту інформацію (ім'я, номер телефону, адресу).
FR1.3.2	Зміни в профілі користувача зберігаються та відображаються в наступних сесіях.
FR1.4	Ролі та права доступу
FR1.4.1	Має бути щонайменше дві ролі: CUSTOMER та ADMIN.
FR1.4.2	ADMIN має доступ до функцій управління меню та замовленнями, які недоступні для CUSTOMER.
FR1.4.3	Доступ на основі ролей забезпечується перевіркою атрибутів у JWT-токені.
FR2	Управління меню (піци та додаткові товари)*
FR2.1	Перегляд меню

FR2.1.1	Гості та авторизовані користувачі можуть переглядати список піц.
FR2.1.2	Список піц включає зображення, назву, ціну (залежно від розміру), тип тіста та короткий опис.
FR2.1.3	Користувачі можуть фільтрувати або сортувати піци за категоріями (наприклад, "Новинки", "Вегетаріанські", "Гострі").
FR2.2	Деталі піци та налаштування
FR2.2.1	Натискання на картку піци відкриває модальне вікно або окрему сторінку з детальною інформацією про піцу (інгредієнти, розміри, типи тіста, ціну та додаткові опції, як-от додатковий сир, соус).
FR2.2.2	Користувач може обрати розмір (наприклад, 25см, 30см, 35см) та тип тіста (тонке, традиційне) перед додаванням у кошик.
FR2.2.3	Користувач може додати додаткові інгредієнти (якщо доступно) за додаткову плату.
FR2.3	Управління меню адміністратором
FR2.3.1	Адміністратор може додати нові позиції (з назвою, описом, ціною, зображенням, категоріями).
FR2.3.2	Адміністратор може редагувати існуючі позиції (змінювати ціну, опис, доступність).
FR2.3.3	Адміністратор може видаляти піци з меню за потреби.

2.3 Моделювання діаграм

Для ефективного проектування та реалізації веб-додатку важливо детально моделювати ключові аспекти системи, включаючи взаємодію користувачів із системою, процеси обробки даних та структуру бази даних. Для цього використовуються різні типи діаграм, такі як діаграми випадків використання (Use Case), послідовності (Sequence), активності (Activity),

реляційної моделі бази даних (ERD) та класова діаграма. Кожна з цих діаграм допомагає візуалізувати та уточнити функціональність, взаємодії та структуру компонентів системи.

Діаграма (рис 2.1) показує ключові функціональні можливості системи для різних категорій користувачів: гостя, зареєстрованого користувача та адміністратора. Вона відображає, як кожен користувач взаємодіє із системою для виконання основних завдань.

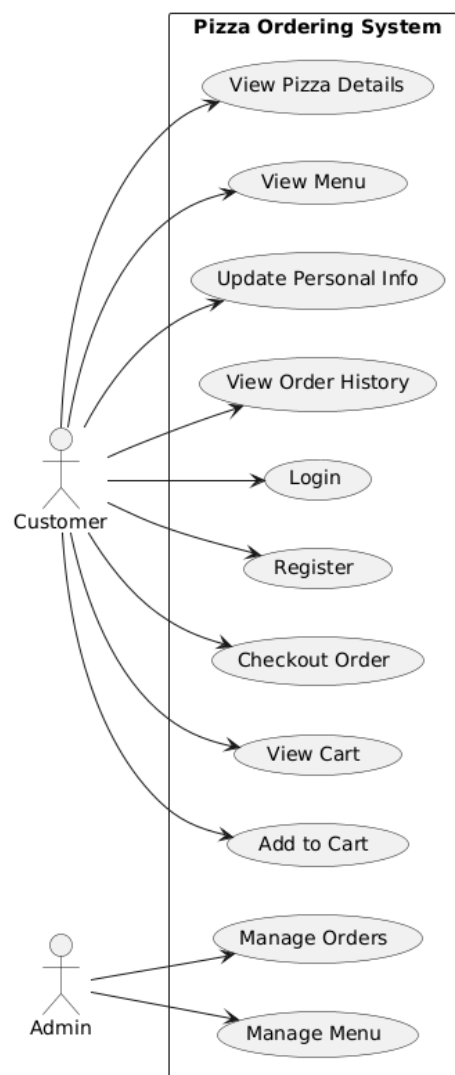


Рис. 2.1 - Діаграма випадків використання (Use Case Diagram)

Ця діаграма демонструє, які дії доступні для гостей, зареєстрованих користувачів і адміністраторів у межах системи замовлення піци.

Діаграми послідовності (рис 2.2-2.4) описують взаємодію між компонентами системи під час виконання конкретних процесів, таких як оформлення замовлення або реєстрація користувача.

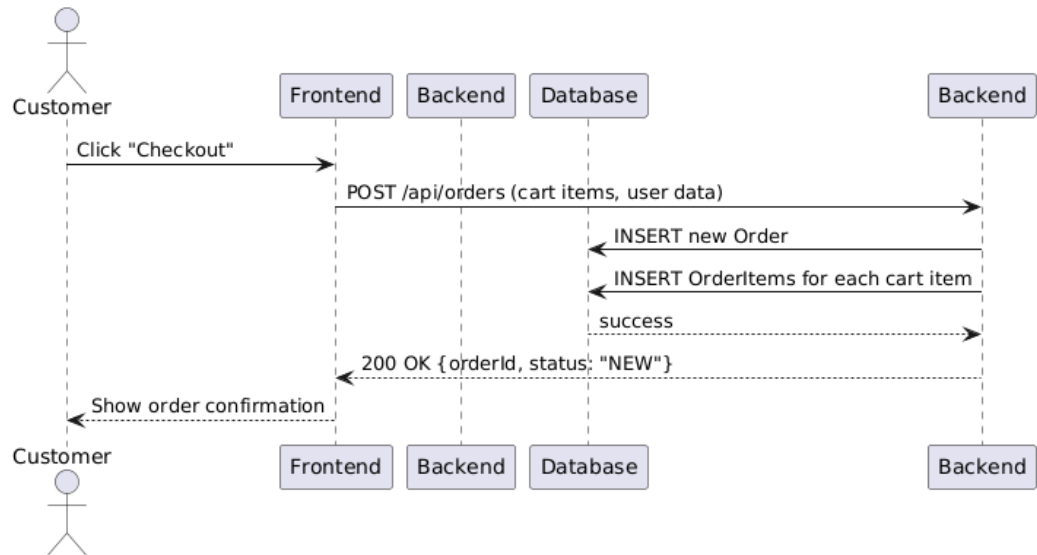


Рис. 2.2 - Діаграма послідовності оформлення замовлення (Checkout Process)

Демонструє процес передачі даних між користувачем, клієнтською частиною, сервером і базою даних під час створення нового замовлення.

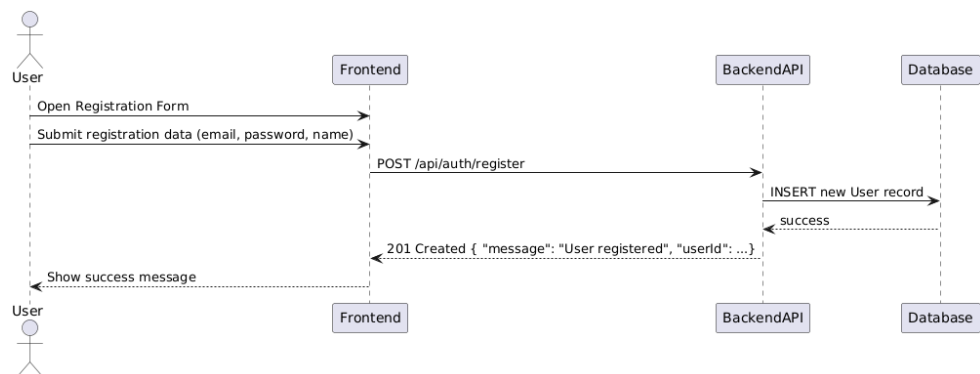


Рис. 2.3 - Діаграма послідовності реєстрації користувача (Registration Sequence Diagram)

Відображає етапи взаємодії між користувачем, фронтендом, бекендом і базою даних під час реєстрації.

Діаграми активності (рис 2.4-2.5) ілюструють логіку виконання різних функціональних вимог системи, наприклад, управління користувачами або меню піцерії.

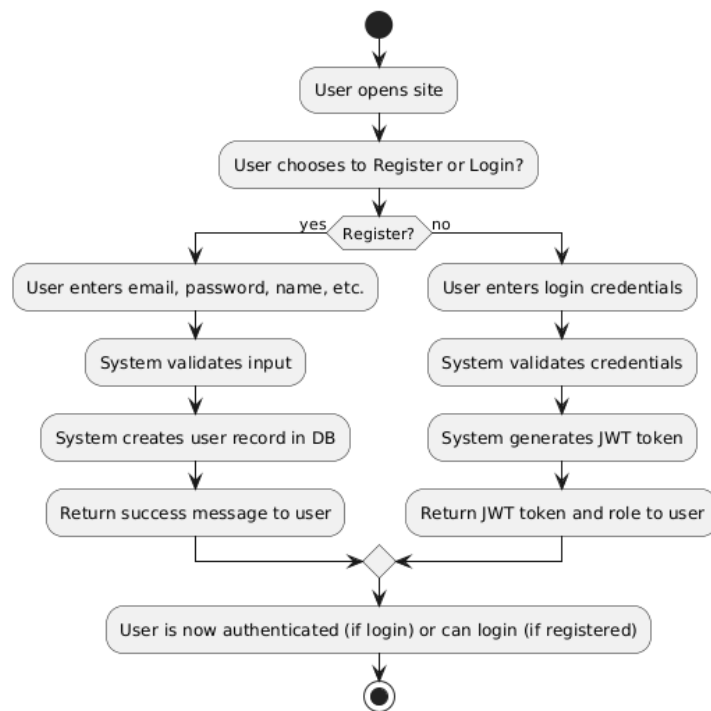


Рис. 2.4 - Діаграма активності управління користувачами (User Management)

Описує основні дії під час реєстрації, входу в систему та управління профілем.

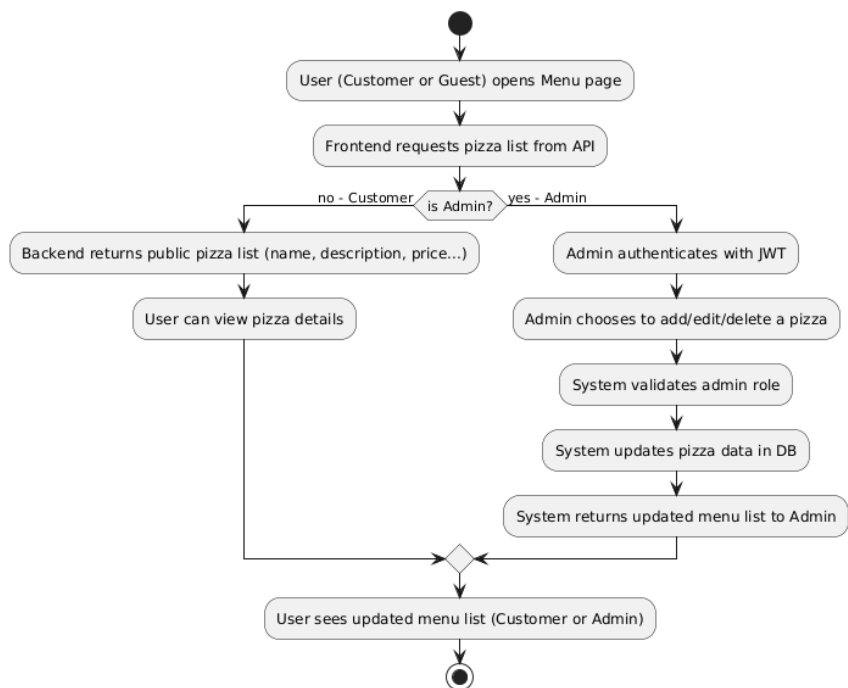


Рис. 2.5 - Діаграма активності управління меню піцерії (Menu Management)

Демонструє дії, які виконує адміністратор для додавання, редагування та видалення позицій у меню.

Реляційна модель бази даних (рис 2.6) ілюструє структуру зберігання даних, взаємозв'язки між сутностями, такими як користувачі, замовлення, піци тощо.

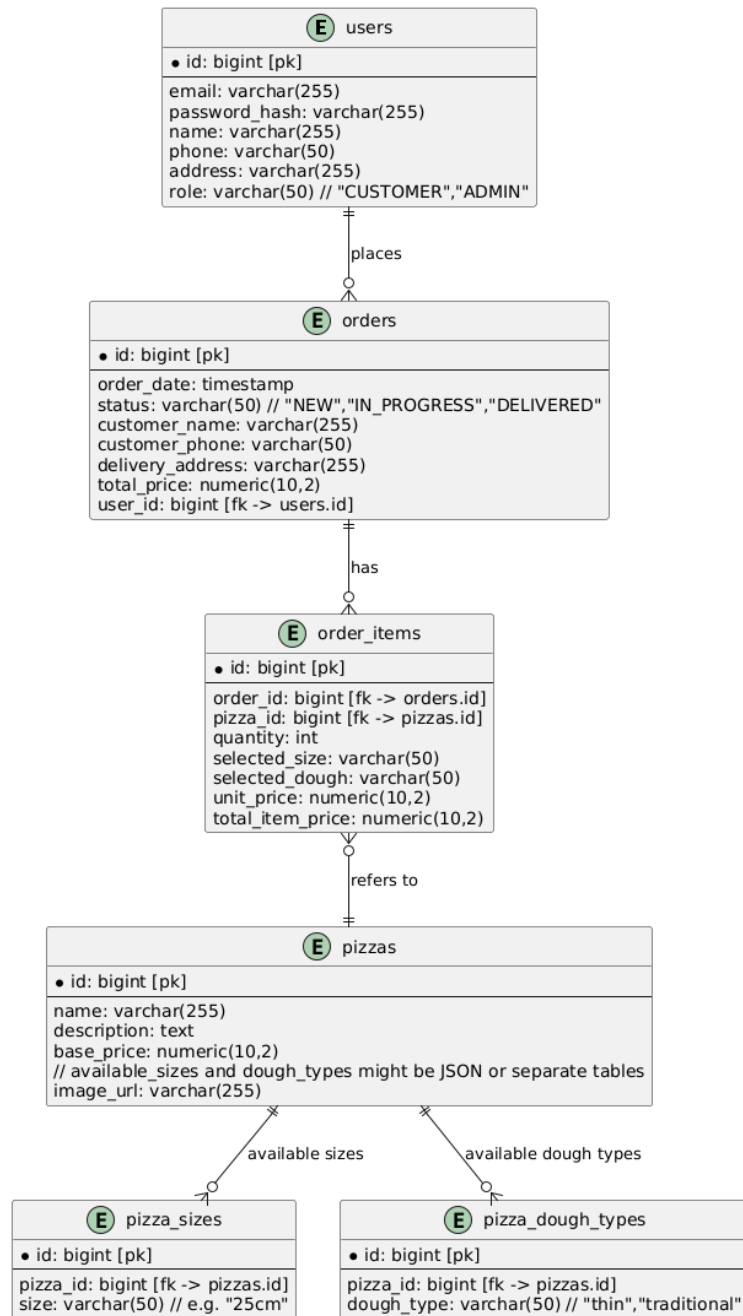


Рис. 2.7 - Діаграма реляційної моделі бази даних (ERD)

Демонструє таблиці бази даних, їхні атрибути та зв'язки між ними.

Класова діаграма (рис 2.8) представляє структуру об'єктів, їхні атрибути та методи, які реалізуються в системі.

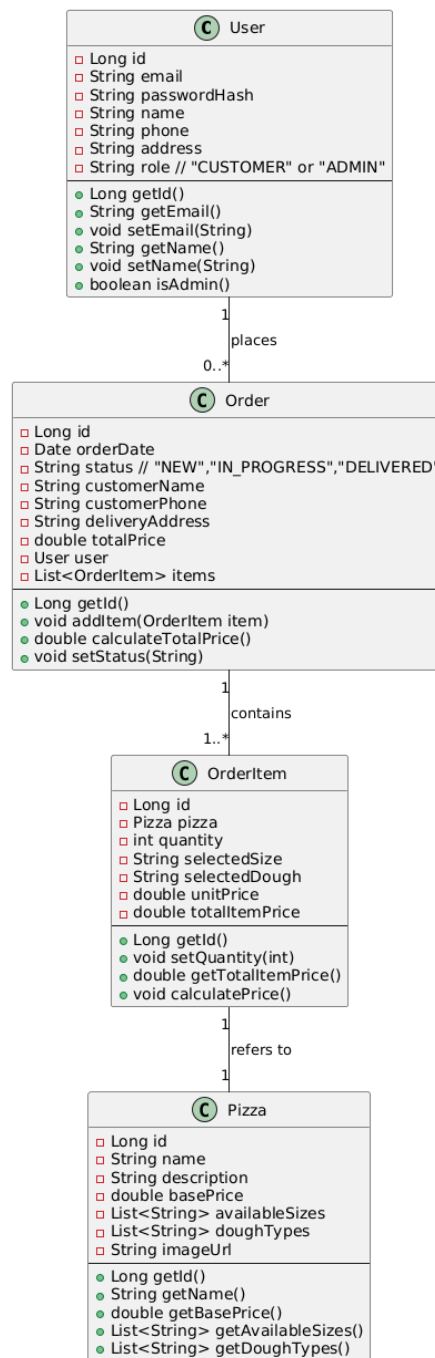


Рис. 2.8 - Класова діаграма

Описує класи, які використовуються в системі, їхню взаємодію, атрибути та методи.

2.4 Опис стеку технологій

Стек технологій, використаний у проєкті

1. Клієнтська частина (Frontend):

2. React — бібліотека для створення компонентів інтерфейсу користувача.
3. React Router — бібліотека для маршрутизації та створення SPA.
4. Axios — HTTP-клієнт для виконання запитів до API.
5. Material-UI (MUI) — бібліотека компонентів для побудови сучасних інтерфейсів із використанням концепцій Material Design.
6. React Context API — інструмент для управління глобальним станом додатку, наприклад, аутентифікацією або вмістом кошика.
7. Formik — бібліотека для управління формами та їх валідацією.
8. Yup — бібліотека для валідації форм, інтегрована з Formik.
9. React Toastify — бібліотека для відображення сповіщень (toast).

Серверна частина (Backend):

1. Java Spring Boot — фреймворк для створення RESTful API та управління бекенд-логікою.
2. Spring Security — забезпечення безпеки API через JWT, автентифікацію та авторизацію.
3. Spring Data JPA — ORM для роботи з базою даних через об'єкти Java.
4. Hibernate — бібліотека ORM, яка інтегрується зі Spring Data JPA.
5. JSON Web Tokens (JWT) — механізм для аутентифікації користувачів і передачі захищених даних.
6. Validation API (javax.validation) — для перевірки вхідних даних, наприклад, електронної пошти чи паролів.
7. PostgreSQL — реляційна база даних для зберігання даних користувачів, замовлень і меню.
8. Postman — інструмент для тестування API.

3 Реалізація інформаційної системи

3.1 Реалізація серверної частини

Серверна частина побудована на основі Java Spring Boot, що дозволяє ефективно реалізовувати RESTful API для клієнтської частини. У даному розділі описано основні компоненти серверної архітектури, зокрема контролери, сервіси та репозиторії, які взаємодіють між собою для забезпечення необхідної функціональності додатку.

Опис контролерів:

1. UserController.

Методи:

- registerUser(): Приймає дані нового користувача, зберігає їх у базі даних, повертає підтвердження.
- loginUser(): Приймає email та пароль, генерує JWT токен у разі успіху.
- getUserProfile(): Повертає інформацію про поточного автентифікованого користувача.
- updateProfile(): Оновлює особисту інформацію користувача.
- changePassword(): Оновлює пароль користувача після перевірки старого пароля.

2. PizzaController.

Методи:

- getAllPizzas(): Повертає список піц з можливістю сортування та фільтрації.
- getPizzaDetails(Long pizzaId): Повертає детальну інформацію про конкретну піцу.
- addPizza(): Додає нову піцу в меню (тільки ADMIN).
- updatePizza(Long pizzaId): Оновлює інформацію про піцу (тільки ADMIN).
- deletePizza(Long pizzaId): Видаляє піцу з меню (тільки ADMIN).

3. OrderController.

Методи:

- `createOrder()`: Приймає дані замовлення, зберігає у базі даних, повертає підтвердження.
- `getMyOrders()`: Повертає список замовлень автентифікованого користувача.
- `cancelOrder(Long orderId)`: Скасовує замовлення за ідентифікатором (тільки NEW).
- `updateOrderStatus(Long orderId, String status)`: Оновлює статус замовлення (тільки ADMIN).
- `getAllOrders()`: Повертає список усіх замовлень (тільки ADMIN).

Опис сервісів

1. UserService.

Функціональність: Обробка логіки реєстрації, аутентифікації та роботи з профілем.

Методи:

- `registerUser(UserDto userDto)`: Реалізує бізнес-логіку реєстрації, включаючи хешування пароля.
- `authenticateUser(LoginDto loginDto)`: Перевіряє дані входу, генерує JWT токен.
- `getUserById(Long userId)`: Отримує інформацію про користувача за ідентифікатором.
- `updateUserProfile(UserUpdateDto userUpdateDto)`: Оновлює дані профілю користувача.
- `changeUserPassword(ChangePasswordDto dto)`: Перевіряє старий пароль і оновлює на новий.

2. PizzaService.

Функціональність: Обробка логіки роботи з піццями (перегляд, додавання, редагування, видалення).

Методи:

- `getAllPizzas(FilterDto filterDto)`: Отримує список піц з урахуванням фільтрів і сортування.
- `getPizzaById(Long pizzaId)`: Отримує деталі конкретної піци.
- `addPizza(PizzaDto pizzaDto)`: Зберігає нову піцу в базі даних.
- `updatePizza(Long pizzaId, PizzaDto pizzaDto)`: Оновлює інформацію про піцу.
- `deletePizza(Long pizzaId)`: Видаляє піцу з бази даних.

3. OrderService.

Функціональність: Обробка логіки створення, перегляду та управління замовленнями.

Методи:

- `createOrder(OrderDto orderDto)`: Створює нове замовлення.
- `getOrdersByUser(Long userId)`: Отримує всі замовлення для конкретного користувача.
- `cancelOrder(Long orderId)`: Скасовує замовлення за ідентифікатором.
- `updateOrderStatus(Long orderId, String status)`: Оновлює статус замовлення.
- `getAllOrders()`: Повертає список усіх замовлень.

Опис репозиторіїв

1. UserRepository

Функціональність: Робота з даними користувачів у базі даних.

Методи:

- `findByEmail(String email)`: Пошук користувача за email.
- `findById(Long id)`: Пошук користувача за ідентифікатором.

2. PizzaRepository

Функціональність: Робота з даними піц у базі даних.

Методи:

- `findAll(Specification<Pizza> spec)`: Повертає список піц з урахуванням специфікацій (фільтрів).
- `findById(Long id)`: Пошук піци за ідентифікатором.

OrderRepository

Функціональність: Робота з даними замовлень у базі даних.

Методи:

- `findById(Long id)`: Пошук замовлення за ідентифікатором.
- `findByUserId(Long userId)`: Повертає замовлення конкретного користувача.
- `findAll()`: Повертає всі замовлення.

Опис реалізації безпеки

Забезпечення безпеки є критично важливим аспектом у розробці веб-додатків, особливо тих, що включають функції аутентифікації, авторизації та обробки конфіденційних даних користувачів. Одним із сучасних та ефективних підходів до реалізації безпеки є використання токенів JWT (JSON Web Token).

JWT — це стандарт відкритого формату для передачі інформації між сторонами у вигляді компактного та самостійного токена. Він використовується для передачі ідентифікаційної інформації про користувача або інші дані, які можна перевірити та довіряти. Основними характеристиками JWT є:

- Компактність: Токени легко передаються через URL, HTTP-заголовки або збереження в `localStorage`.
- Самостійність: JWT містить усю необхідну інформацію для ідентифікації користувача, тому серверу не потрібно зберігати сесію користувача.
- Безпека: Дані в JWT підписуються за допомогою алгоритмів шифрування (наприклад, HMAC або RSA), що дозволяє перевірити їхню цілісність.

JWT складається з трьох частин:

- Header (заголовок): Містить інформацію про тип токена та алгоритм підпису.
- Payload (корисне навантаження): Містить дані (claims), такі як id користувача, його роль, час створення токена та термін дії.
- Signature (підпис): Використовується для забезпечення цілісності токена та його неможливості підробки.

Реалізація безпеки в серверній частині проєкту здійснена з використанням Spring Security та JWT.

Під час реєстрації користувач надає свої дані (ім'я, email, пароль тощо). Пароль хешується за допомогою алгоритму BCrypt і зберігається у базі даних.

Під час входу користувач надає email та пароль.

Система перевіряє правильність введених даних. У разі успіху генерується JWT токен.

JWT токен у даному проєкті містить такі claims:

- sub: ідентифікатор користувача.
- role: роль користувача (CUSTOMER або ADMIN).
- iat: час створення токена.
- exp: час закінчення дії токена.

Spring Security налаштовано для перевірки всіх запитів, які потребують аутентифікації:

Доступ до різних ендпоінтів налаштовано за допомогою анотацій Spring Security (@PreAuthorize, @Secured) або конфігурації безпеки.

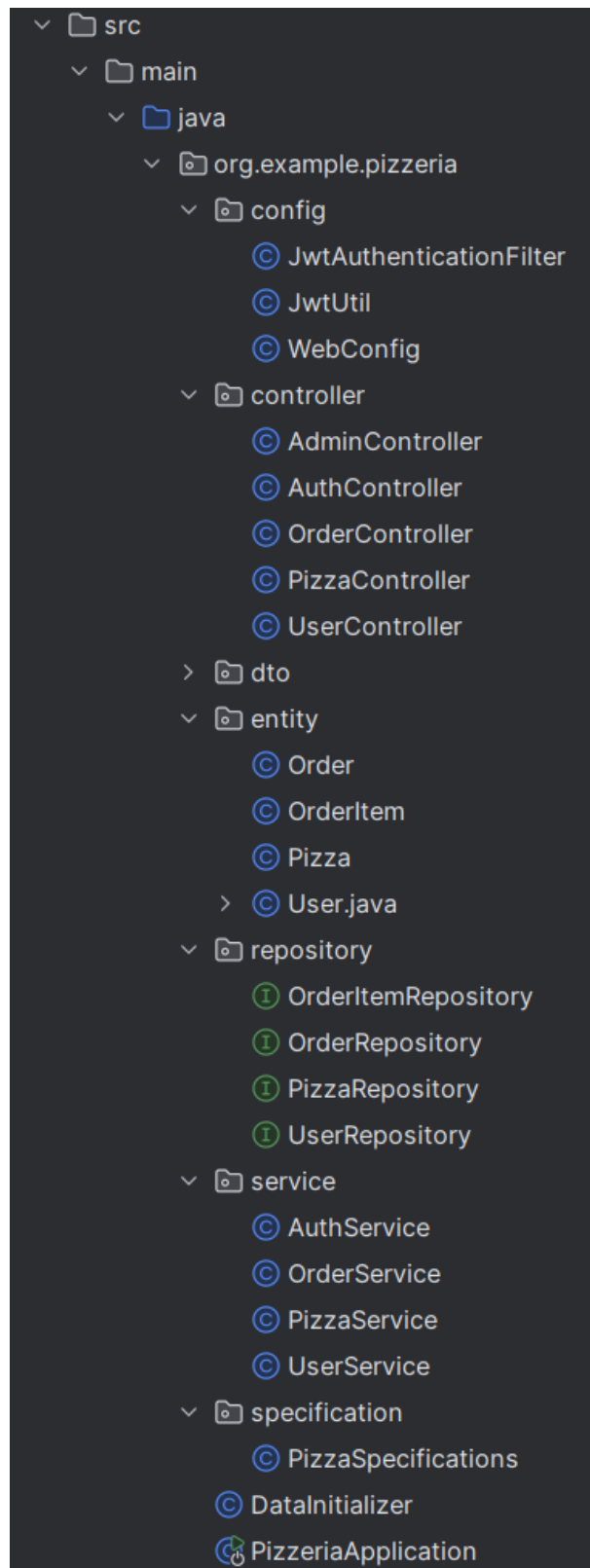


Рис. 3.1— Структура проекту backend

3.2 Реалізація клієнтської частини

Клієнтська частина проєкту створена з використанням сучасного фреймворку React, що дозволяє розробляти інтерактивні односторінкові веб-додатки (SPA). Основна мета клієнтської частини – забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів, що дозволяє переглядати меню, оформляти замовлення, керувати обліковими записами та взаємодіяти із серверною частиною через RESTful API.

Використання SPA забезпечує швидкий доступ до функціоналу додатка без необхідності перезавантаження сторінки, що значно покращує користувацький досвід.

Для організації переходів між сторінками застосовується бібліотека React Router. Основні маршрути включають:

- Головна сторінка з переглядом меню.
- Сторінка входу та реєстрації.
- Кошик з переглядом замовлень.
- Сторінка оформлення замовлення.
- Сторінка профілю користувача для редагування особистих даних та перегляду історії замовлень.

Управління станом клієнтської частини реалізовано за допомогою Context API. Це забезпечує централізоване зберігання даних користувача (JWT токен, роль, дані профілю) та кошика.

Уся функціональність розбита на невеликі повторно використовувані компоненти, що відповідають принципам модульності. Основні компоненти включають:

- Header: відображає навігацію, кошик та дані користувача.
- PizzaCard: картка з інформацією про піцу.
- PizzaModal: модальне вікно для вибору розміру, типу тіста та додавання до кошика.
- Cart: компонент для перегляду та редагування кошика.
- Forms: компоненти для входу, реєстрації та редагування профілю.

Для забезпечення основної функціональності реалізовані такі сторінки:

- Головна сторінка: показує список доступних піц з можливістю фільтрації та сортування.
- Кошик: дозволяє переглядати додані піци, змінювати кількість та оформляти замовлення.
- Сторінка профілю: перегляд особистої інформації та історії замовлень.

Взаємодія з серверною частиною здійснюється через бібліотеку Axios.

Усі форми мають перевірки на стороні клієнта (коректність email, довжина пароля, обов’язковість полів).

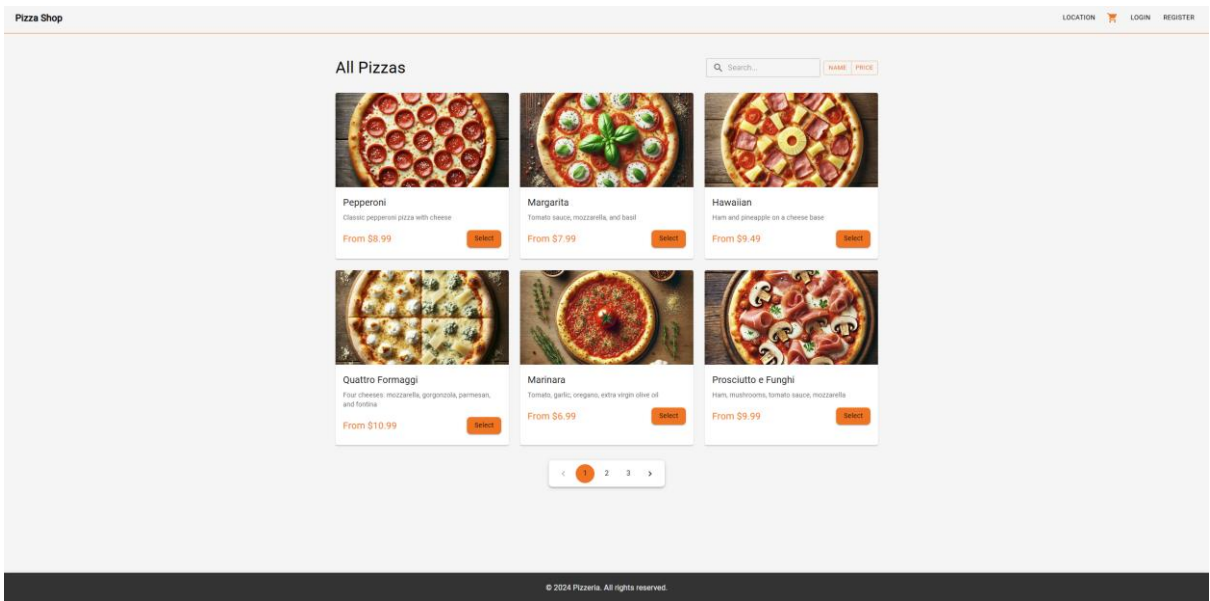


Рис. 3.2— Головна сторінка

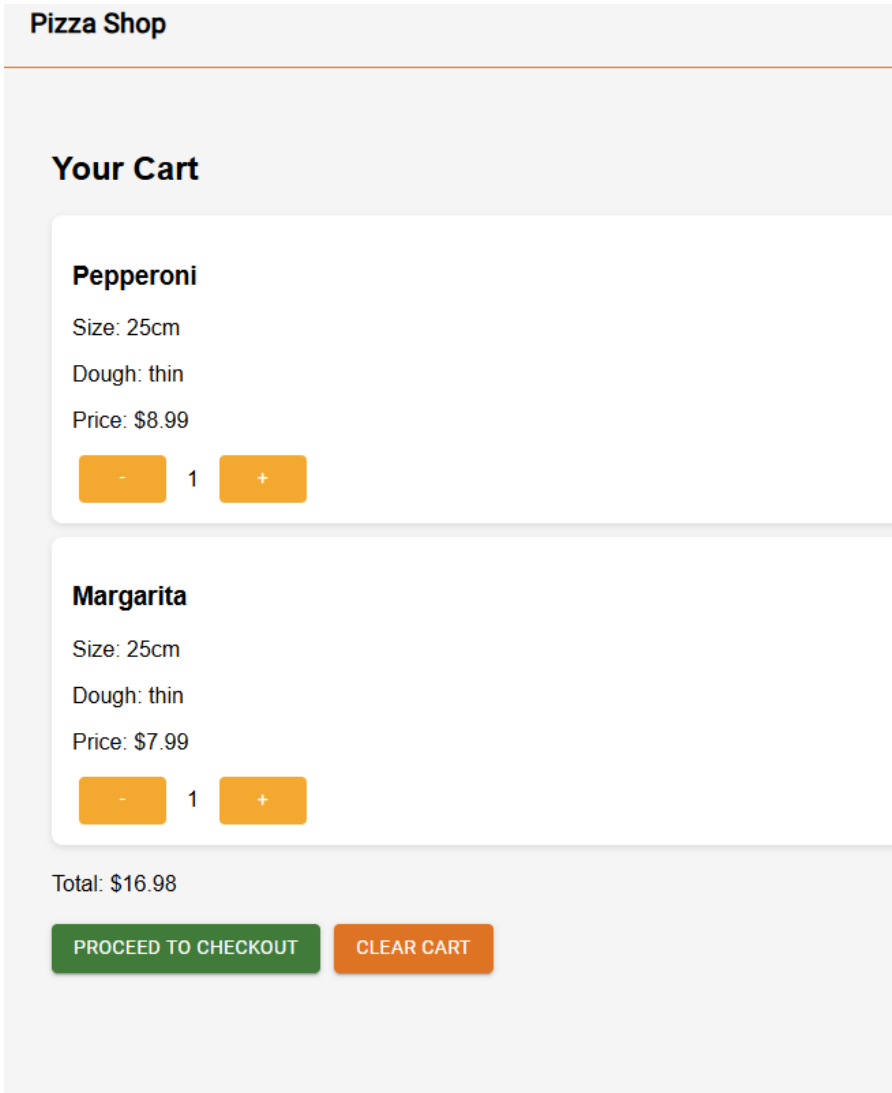


Рис. 3.3— Кошик

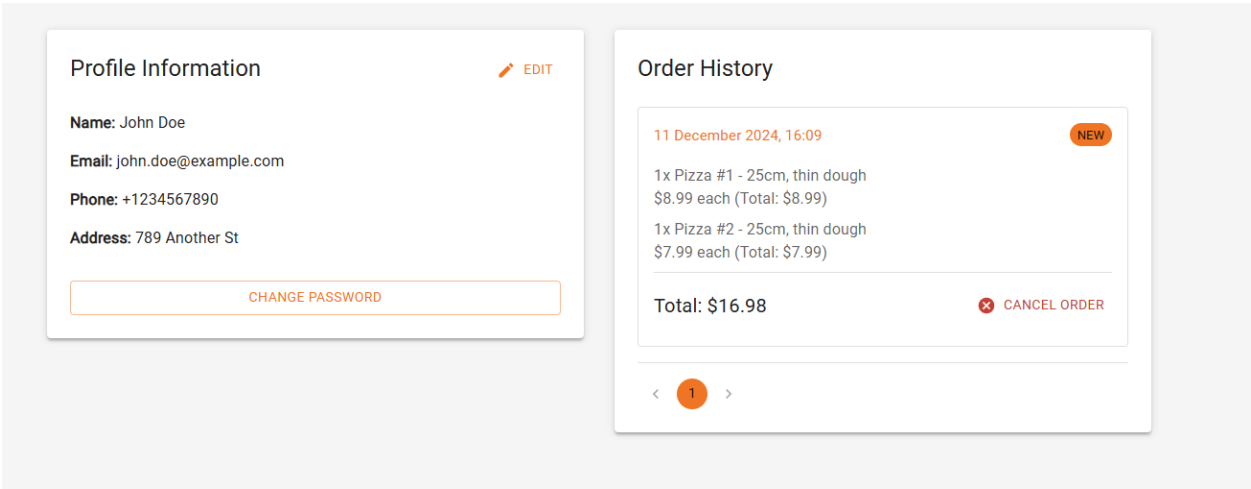
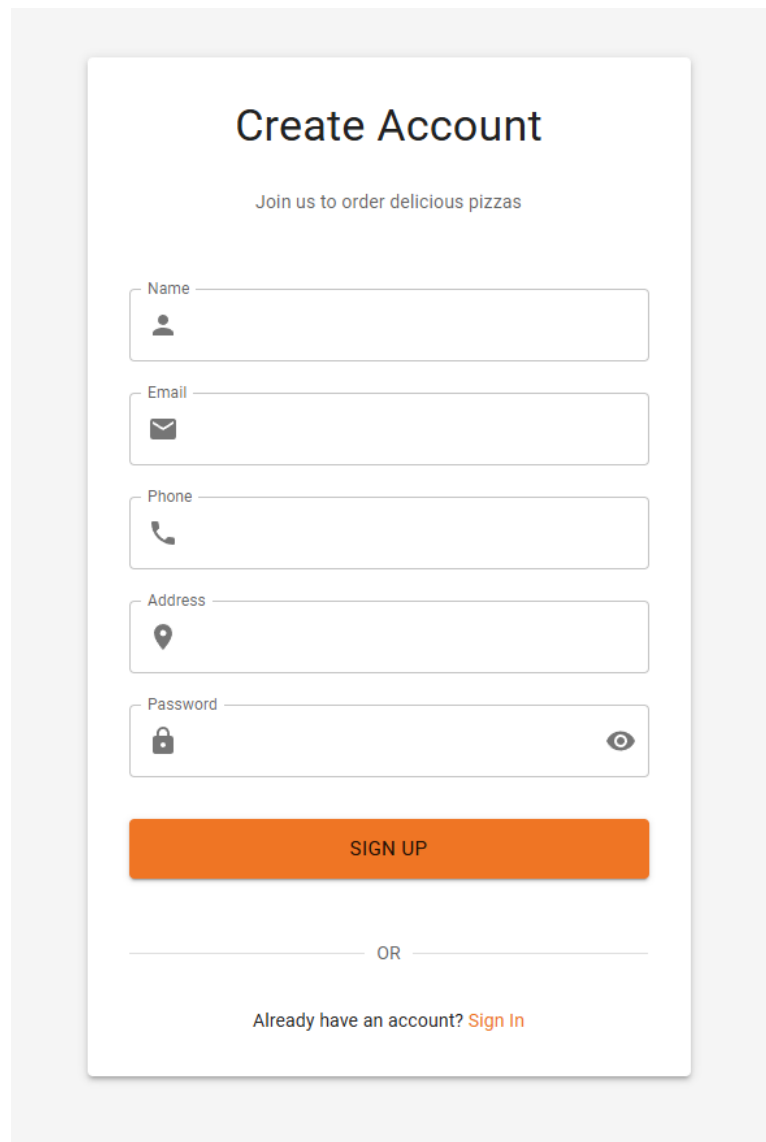


Рис. 3.4— Сторінка профілю



The image shows a 'Create Account' registration form. At the top, the title 'Create Account' is centered in a large, bold, black font. Below it, a subtitle 'Join us to order delicious pizzas' is centered in a smaller, regular black font. The form consists of five input fields, each with a label and an icon: 'Name' with a person icon, 'Email' with an envelope icon, 'Phone' with a telephone handset icon, 'Address' with a location pin icon, and 'Password' with a padlock icon. To the right of the password field is an eye icon for toggling visibility. Below the input fields is a prominent orange button with the text 'SIGN UP' in white, uppercase letters. Underneath the button is a horizontal line with the word 'OR' centered. At the bottom, the text 'Already have an account?' is followed by a link 'Sign In' in orange text.

Create Account

Join us to order delicious pizzas

Name

Email

Phone

Address

Password

SIGN UP

OR

Already have an account? [Sign In](#)

Рис. 3.5— Сторінка реєстрації

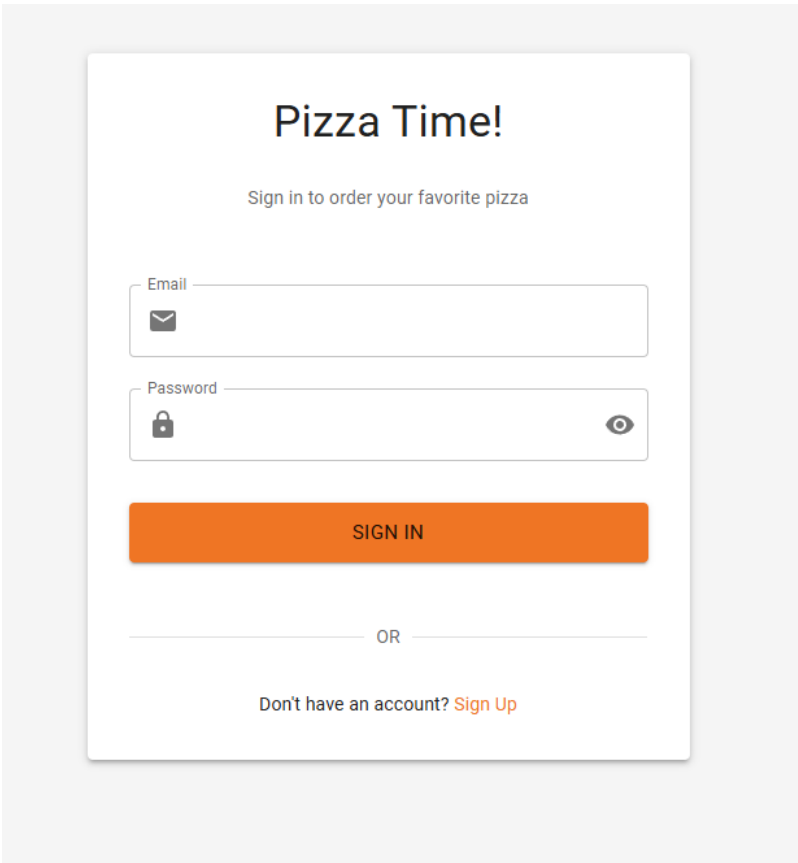


Рис. 3.6— Сторінка профілю

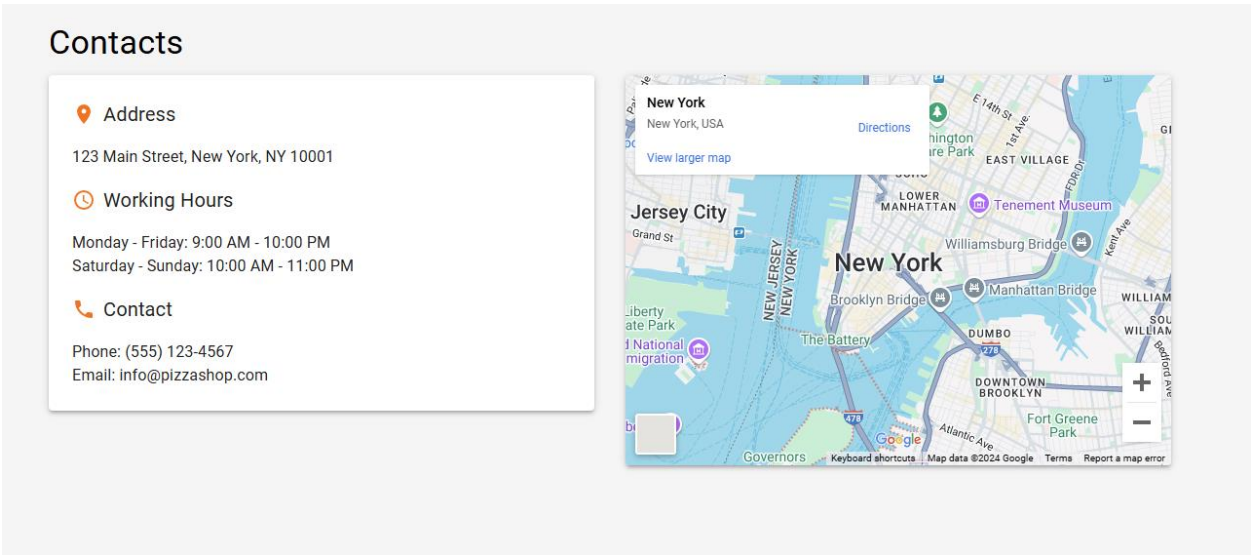


Рис. 3.7— Сторінка контактів

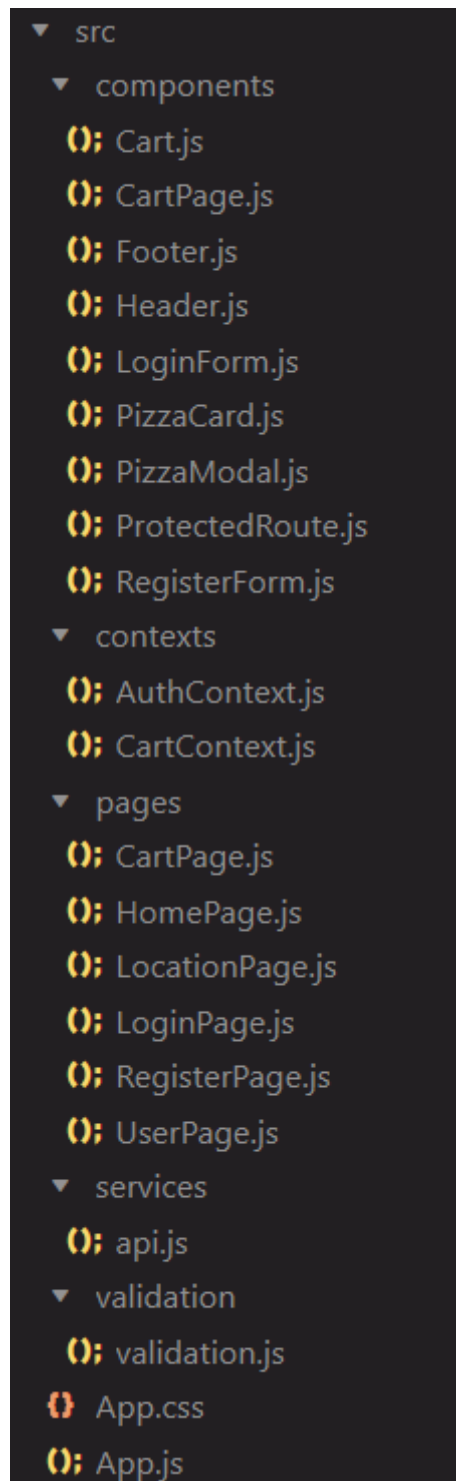


Рис. 3.8— Структура проекту frontend

4 Тестування додатку

4.1. Цілі та завдання тестування

Тестування веб-застосунку для вивчення англійської мови має на меті забезпечити його коректну роботу, відповідність функціональним вимогам та виявлення потенційних помилок. Основними цілями тестування є:

- 1) Перевірка функціональності API: Забезпечити, щоб усі API-ендпоінти працювали коректно та повертали очікувані дані, а саме:
 - правильність отримання карток за ID, словом та типом;
 - коректність створення, оновлення та видалення карток;
 - правильність отримання карток для навчання;
 - обробка відповідей користувача;
 - отримання статистики навчання.
- 2) Перевірка логіки навчання: Забезпечити, щоб алгоритм інтервального повторення та оновлення прогресу навчання працювали правильно, відповідно до заданих параметрів та логіки, описаної в розділі 3.
- 3) Перевірка роботи Spring Security: Забезпечити, щоб автентифікація та авторизація працювали коректно, а доступ до ресурсів був обмежений відповідно до ролей користувачів. Адміністратор має мати доступ до керування картками, а звичайні користувачі - лише до навчання та перегляду статистики.
- 4) Перевірка обробки помилок: Забезпечити, щоб застосунок коректно обробляв помилки, такі як некоректні запити, неавторизований доступ, відсутність даних тощо, та повертав відповідні коди стану HTTP.
- 5) Перевірка роботи інтерфейсу користувача: Забезпечити зручність та інтуїтивність інтерфейсу користувача, коректність відображення даних, обробки введених даних та взаємодії з API.

4.2. Методи та інструменти тестування

Для тестування веб-застосунку було використано наступні методи та інструменти:

- Ручне тестування API: використання Postman для надсилення запитів до API та перевірки відповідей сервера.
- Ручне тестування інтерфейсу користувача: взаємодія з інтерфейсом користувача в браузері для перевірки його функціональності та зручності.

4.3. Тестування API

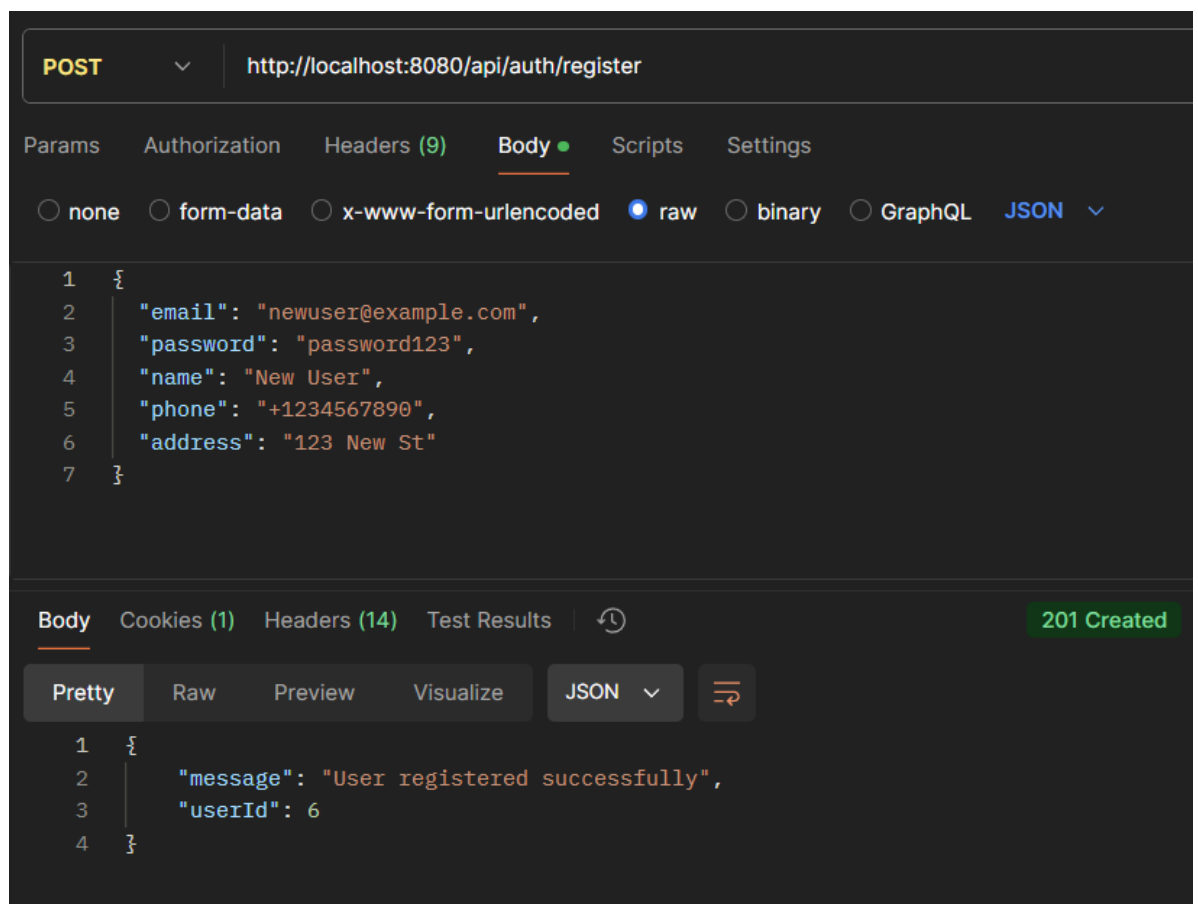


Рис. 4.1 – Реєстрація нового користувача

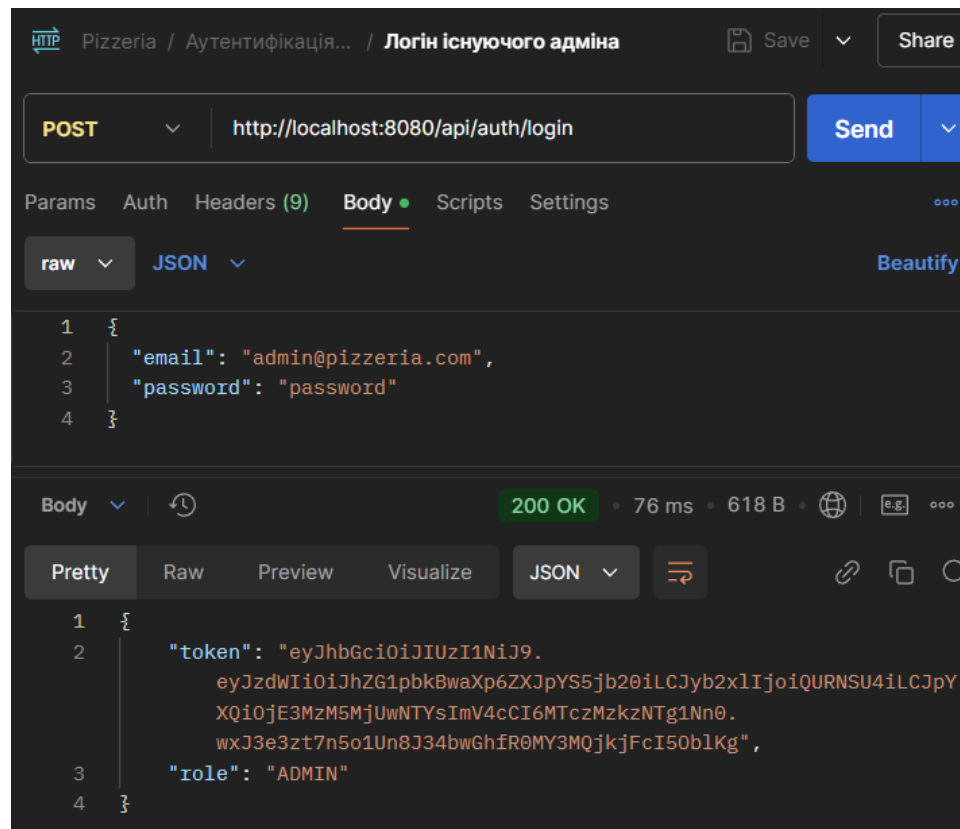


Рис. 4.2 – Логін існуючого адміна

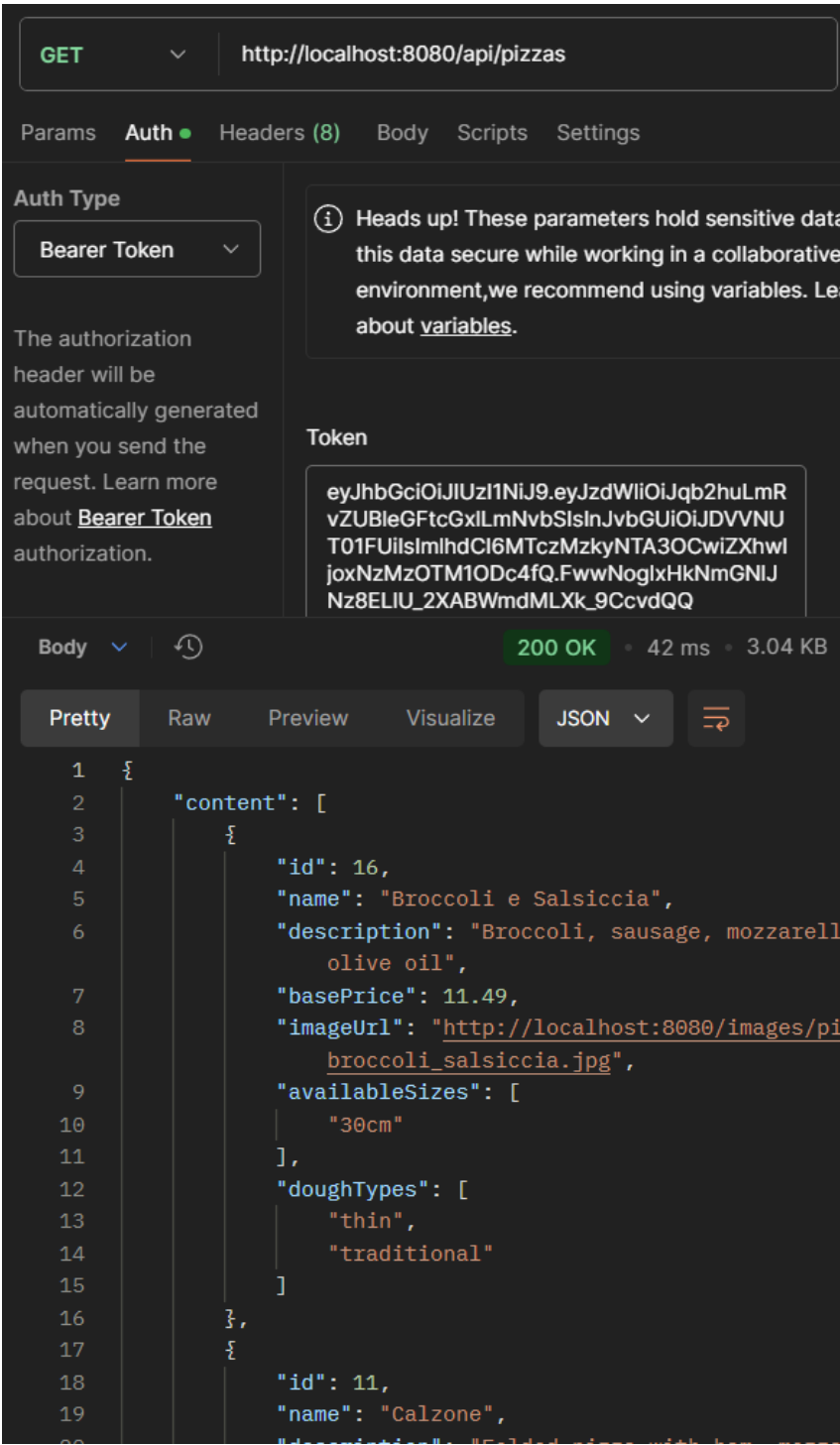


Рис. 4.3 – Перегляд меню піц

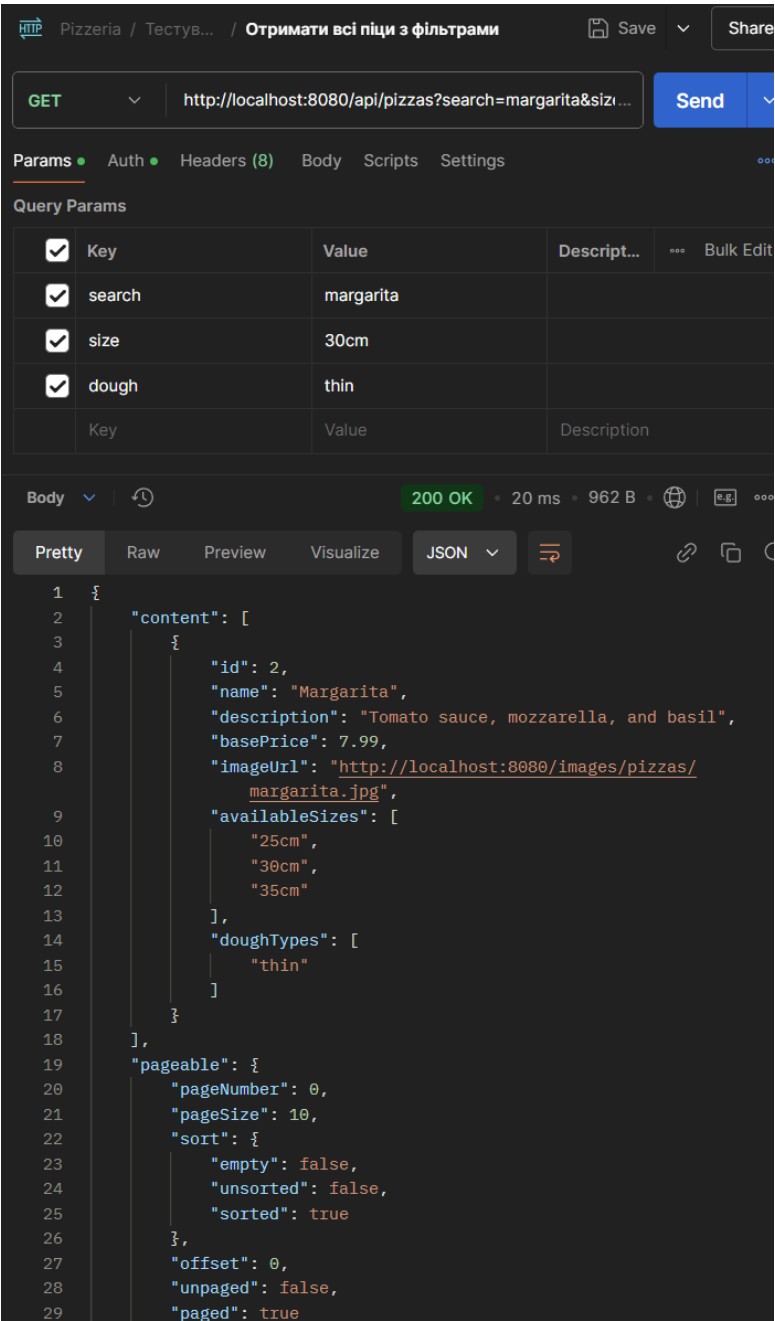


Рис. 4.4 – Отримати всі піци з фільтрами

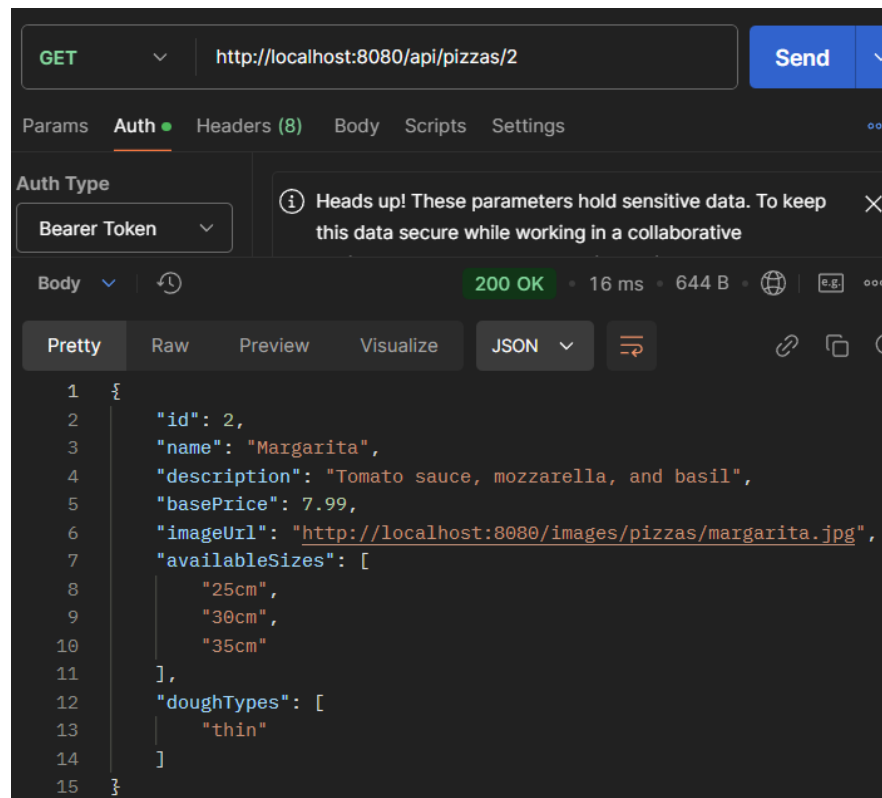


Рис. 4.5 – Деталі піци

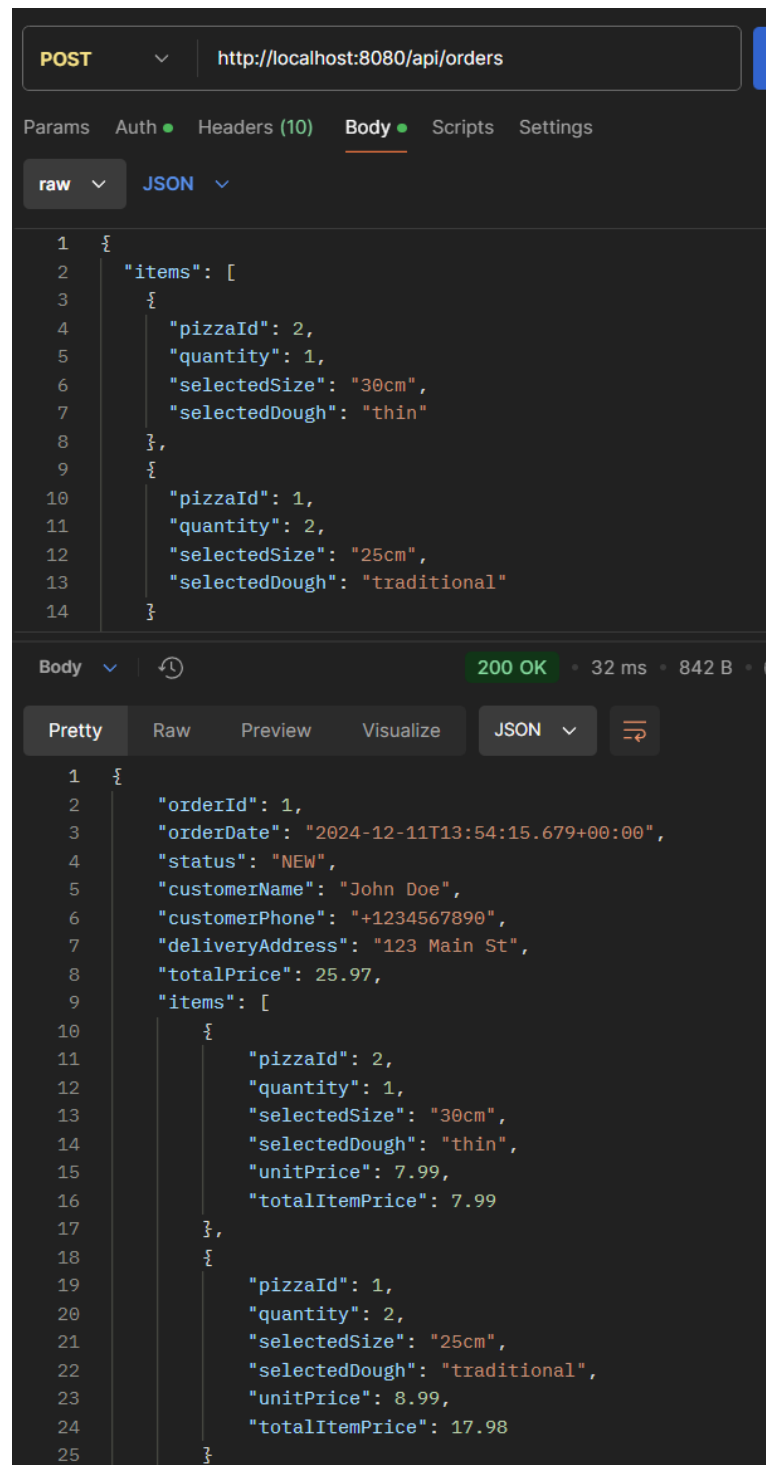


Рис. 4.6 – Створення замовлення

The screenshot shows a web client interface with the following details:

- URL:** `http://localhost:8080/api/orders/my`
- Method:** `GET`
- Status:** `200 OK` (17 ms, 844 B)
- Response Body (JSON):**

```

1  [
2    {
3      "orderId": 1,
4      "orderDate": "2024-12-11T13:54:15.679+00:00",
5      "status": "NEW",
6      "customerName": "John Doe",
7      "customerPhone": "+1234567890",
8      "deliveryAddress": "123 Main St",
9      "totalPrice": 25.97,
10     "items": [
11       {
12         "pizzaId": 2,
13         "quantity": 1,
14         "selectedSize": "30cm",
15         "selectedDough": "thin",
16         "unitPrice": 7.99,
17         "totalItemPrice": 7.99
18       },
19       {
20         "pizzaId": 1,
21         "quantity": 2,
22         "selectedSize": "25cm",
23         "selectedDough": "traditional",
24         "unitPrice": 8.99,
25         "totalItemPrice": 17.98
26       }
27     ]
28   }
29 ]

```

Рис. 4.7 – Перегляд історії замовлень

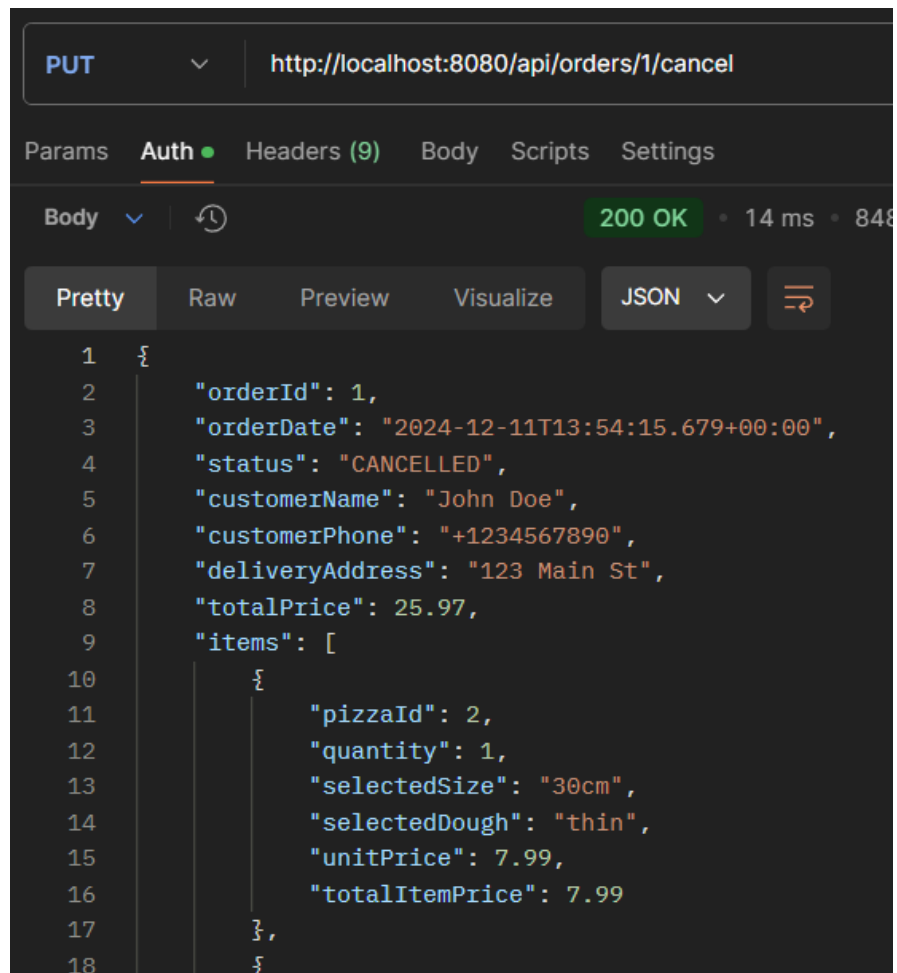


Рис. 4.8 – Відміна замовлення

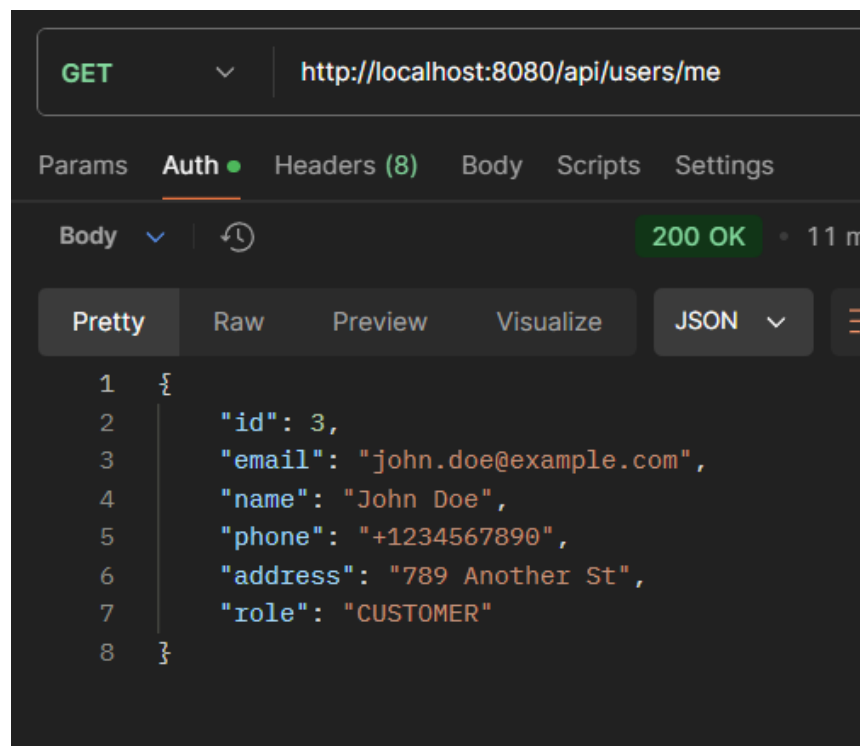


Рис. 4.9 – Інформація профіля

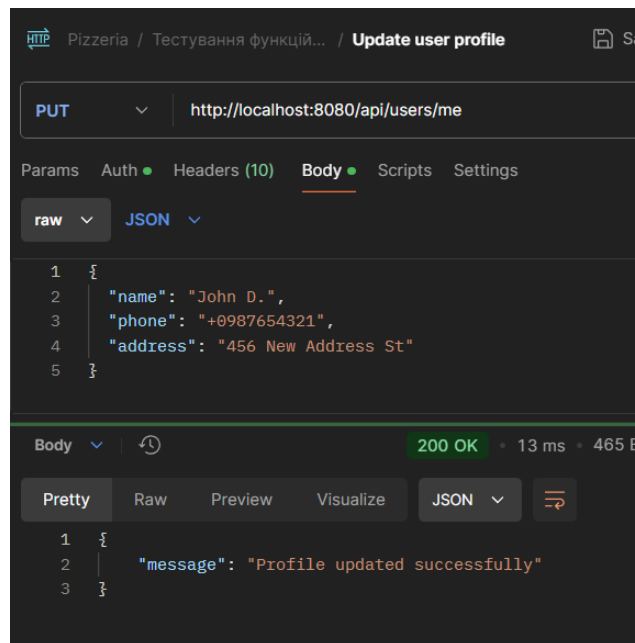


Рис. 4.10 – Оновити інформацію профіля

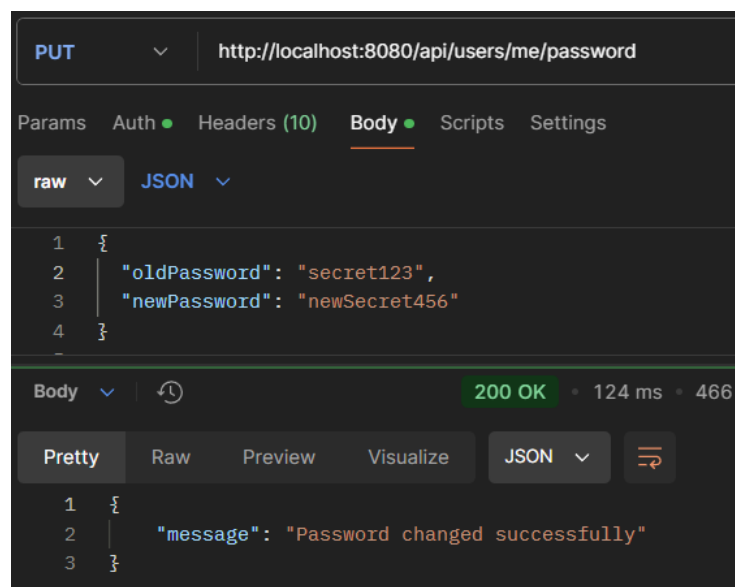


Рис. 4.11 – Змінити пароль

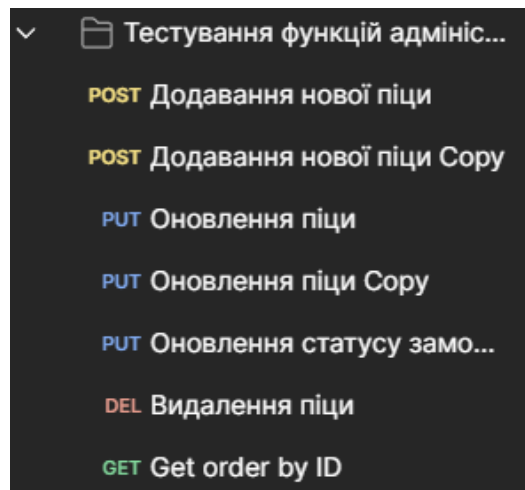


Рис. 4.12 – Функції адманістратора

4.4. Тестування інтерфейсу користувача

Тестування інтерфейсу користувача проводилося шляхом ручної взаємодії з веб-сторінками застосунку в браузері. Перевірялася коректність відображення даних, робота форм, переходи між сторінками, обробка введених даних, а також загальна зручність та інтуїтивність інтерфейсу.

Висновки

В рамках даної курсової роботи було успішно розроблено веб-застосунок для управління замовленнями піцерії, який базується на клієнт-серверній архітектурі. Цей проєкт дозволив закріпити теоретичні знання, отримані в курсі "Розробка веб-додатків", і здобути практичний досвід створення функціонального, безпечного та зручного у використанні веб-застосунку.

Розроблено багаторівневу архітектуру системи, яка включає серверну частину, реалізовану на основі Java Spring Boot, та клієнтську частину, створену за допомогою React. Серверна частина забезпечує обробку бізнес-логіки, управління базою даних та захист інформації за допомогою механізму JWT-аутентифікації. Клієнтська частина надає інтуїтивно зрозумілий інтерфейс, який дозволяє користувачам переглядати меню, оформляти замовлення, управляти профілем і переглядати історію замовлень.

Було реалізовано RESTful API, яке забезпечує зручну та ефективну взаємодію між клієнтом і сервером. Функціонал API включає обробку запитів на реєстрацію, авторизацію, управління меню, оформлення замовлень та оновлення їх статусу. Використання сучасних технологій, таких як Spring Security і JPA, дозволило створити систему з високим рівнем безпеки та надійності.

Додаток включає гнучкий механізм управління замовленнями, який дозволяє користувачам вибирати розмір піци, тип тіста та додаткові інгредієнти. Реалізовано функцію оформлення замовлення з можливістю вказати адресу доставки та контактні дані. Для адміністраторів надано можливості редагування меню та управління статусами замовлень через спеціалізовану панель.

Крім основної функціональності, забезпечено високу зручність використання через адаптивний дизайн, який коректно відображається на різних пристроях. Використання сучасних бібліотек і підходів, таких як React

Router і Context API, дозволило створити односторінковий застосунок (SPA), який забезпечує плавну навігацію та швидке завантаження сторінок.

Система була протестована на відповідність функціональним вимогам, включаючи реєстрацію, авторизацію, управління замовленнями та безпеку даних. Проведене тестування підтвердило коректність роботи всіх компонентів і відповідність реалізованого функціоналу заявленим вимогам.

Розроблений застосунок є надійним інструментом для управління замовленнями в піцерії та може бути легко адаптований до інших типів бізнесу. У майбутньому він може бути розширений такими можливостями:

- Інтеграція з платіжними системами для онлайн-оплати замовлень.
- Додавання функції відстеження замовлень у реальному часі.
- Розробка мобільного застосунку на основі існуючого API.
- Впровадження аналітики для адміністраторів, яка дозволить відслідковувати популярність піц і ефективність доставки.
- Покращення UX/UI через використання інтерактивних елементів та оновлення дизайну.

Розроблений застосунок демонструє актуальність клієнт-серверних технологій у сучасному світі, їхню зручність і гнучкість у використанні для бізнес-завдань. Отриманий досвід і знання дозволяють вважати цю роботу вагомим внеском у професійний розвиток і підготовку до подальшої кар'єри у сфері інформаційних технологій.

Перелік використаних джерел

1. React Documentation. — [Електронний ресурс]. — Режим доступу: <https://react.dev/>
2. Spring Framework Documentation. — [Електронний ресурс]. — Режим доступу: <https://spring.io/>
3. RESTful API Design Guide. — [Електронний ресурс]. — Режим доступу: <https://www.restapitutorial.com/>
4. A Guide to JWT (JSON Web Tokens). — [Електронний ресурс]. — Режим доступу: <https://jwt.io/>
5. Material-UI React Library Documentation. — [Електронний ресурс]. — Режим доступу: <https://mui.com/>
6. Робочі матеріали лекцій та практичних занять з курсу «Розробка веб-додатків».

ДОДАТОК А

КОД ПРОГРАМИ

Github репозиторій з бекендом:

https://github.com/levsemyvolos/Pizzeria_server

Github репозиторій з фронтендом:

https://github.com/levsemyvolos/Pizzeria_front