МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНА З ВИКОРИСТАННЯМ REACT ТА NODE.JS»**

Виконав: студент 4 курсу, групи 6.1211-2пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Д.С. Батаєв
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD Чопорова О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2024 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Батаєву Дмитру Сергійовичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет-магазина з використанням React та Node.js

керівник роботи Чопорова Оксана Володимирівна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 02.06.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація інтернет-магазина.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 26.12.2024 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	08.01.2025	
2.	Збір вихідних даних.	29.01.2025	
3.	Обробка методичних та теоретичних	10.02.2025	
	джерел.		
4.	Розробка першого та другого розділу.	17.03.2025	
5.	Розробка третього розділу.	19.05.2025	
6.	Оформлення та нормоконтроль	26.05.2025	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	18.06.2025	

Студент _____ Д.С. Батаєв
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Чопорова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет-магазина з використанням React та Node.js»: 43 с., 29 рис., 7 джерел, 4 додатка.

БАЗА ДАНИХ, ІНТЕРНЕТ МАГАЗИН, BACK-END, FRONT-END, NODE.JS, REACT.

Об’єкт дослідження – фреймворки Node.js і React.

Мета роботи: розробка сучасного вебзастосунку інтернет-магазину на основі технологій React (frontend) та Node.js з використанням Prisma ORM (backend), який дозволяє користувачам реєструватися, переглядати каталог товарів, додавати їх до кошика, оформляти замовлення та зберігати інформацію в базі даних PostgreSQL.

Метод дослідження – дослідження предметної області, включаючи аналіз та систематизацію інформації.

У цій роботі розроблено повнофункціональний вебзастосунок – інтернет-магазин бургерів. Він включає фільтрацію товарів, модальні вікна з описом продукції, кошик, а також системи реєстрації, авторизації і зберігання замовлень у базі даних PostgreSQL, використовуючи Prisma ORM.

SUMMARY

Bachelor’s qualifying paper “Online Store Development Using React and Node.js”: 43 pages, 29 figures, 7 references, 4 supplements.

BACK-END, DATABASE, FRONT-END, NODE.JS, ONLINE STORE, REACT.

The object of the study is framework Node.js and framework React.

The aim of the study is the development of a modern web application for an online store based on React (frontend) and Node.js technologies using Prisma ORM (backend), which allows users to register, view the product catalog, add them to the cart, place orders, and store information in a PostgreSQL database.

The method of research is study of the subject area, including analysis and systematization of information.

This work presents a fully functional web application – a burger online store. It includes product filtering, modal windows with product descriptions, a shopping cart, as well as registration and authentication systems, and order storage in a PostgreSQL database using Prisma ORM.

ЗМІСТ

Завдання на кваліфікаційну роботу……..…2
Реферат 4
Summary 5
Вступ..... 8
1 Теоретичні відомості 9
1.1 Основні концепції інтернет-магазину..... 9
1.2 Огляд технологій React та Node.js..... 9
1.2.1 React..... 10
1.2.2 Node.js 10
1.3 Огляд баз даних для інтернет-магазину 11
1.4 Аналіз вимог до інтернет-магазину 12
2 Проєктування інтернет-магазину 14
2.1 Проєктування структури інтернет-магазину..... 14
2.1.1 Інформаційна архітектура..... 14
2.1.2 Створення діаграми бази даних..... 15
2.2 Розробка макетів та прототипів..... 17
3 Розробка інтернет-магазину..... 18
3.1 Розробка Back-end..... 18
3.2 База даних 19
3.2.1 Підключення до бази даних..... 21
3.2.2 Розробка моделей бази даних 22
3.3 Реалізація API для основних операцій..... 24
3.4 Створення сторінки авторизації та реєстрації 25
3.5 Створення головної сторінки..... 27
3.6 Тестування 29
Висновки 35
Перелік посилань..... 37

Додаток А Файл schema.prisma 38
Додаток Б Файл index.js з серверної частини 40
Додаток В Файл App.jsx з front-end частини 41
Додаток Г Файл index.js з front-end частини 43

ВСТУП

В даний час комп’ютери та програмне забезпечення використовуються для автоматизації звичайних процесів і підвищення їх ефективності. Особливо це стосується електронної комерції, де швидкість, комфорт і легкий доступ відіграють важливу роль в успіху.

Створення інтернет-магазину є актуальною роботою в умовах зростаючої тенденції онлайн-продажів. Електронні магазини допомагають компаніям розширити список клієнтів, зменшити витрати на оренду та персонал і надають покупцям простий спосіб робити замовлення в будь-який час і з будь-якого місця.

Цей проєкт спрямований на створення інтернет-магазину з використанням нових технологій React і Node.js. React допоможе побудувати живий і простий користувальницький інтерфейс, а Node.js – потужну і адаптовану серверну частину.

Мета дослідження – розробка ефективного та функціонального інтернет-магазину з використанням React та Node.js.

Об’єкт дослідження – процес розробки інтернет-магазину.

Предмет дослідження – функціональні можливості, інтерфейс та технічна реалізація інтернет-магазину засобами React та Node.js.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- проаналізувати предметну область та визначити вимоги;
- розробити структуру та функціональність інтернет-магазину;
- реалізувати front-end частину з використанням React;
- реалізувати back-end частину з використанням Node.js;
- провести тестування та оцінку ефективності розробленого інтернет-магазину.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Основні концепції інтернет-магазину

Інтернет-магазин – це ІС, яка виконує функції купівлі-продажу товарів чи послуг через Інтернет. Він набув величезної популярності, оскільки покупці можуть легко знаходити потрібні товари, порівнюючи їх ціни та роблячи замовлення, не виходячи з дому.

Велике значення в роботі інтернет-магазину мають такі поняття:

– каталог товарів: презентація товарів з описом, фото, цінами та іншою інформацією;
– кошик для покупок: покупець збирає продукти, вибрані для розміщення замовлення;
– обробка замовлення: можливість оформити замовлення, вибрати спосіб оплати та доставки;
– керування користувачами: реєстрація, авторизація та особистий кабінет покупця;
– пошук і фільтрація: інструменти, які допомагають покупцям швидко знаходити потрібні продукти;
– адміністрування: функції для керування продуктами, замовленнями, користувачами та іншим вмістом магазину.

Ці ідеї є основою для створення легкого та корисного вебмагазину.

1.2 Огляд технологій React та Node.js

При розробці сучасних вебдодатків, особливо таких складних, як інтернет-магазин, вибір правильних технологій відіграє вирішальну роль. У цьому проєкті я використовую React для створення користувацького інтерфейсу (front-end) та Node.js для розробки серверної частини (back-end).

1.2.1 React

React – це JavaScript-бібліотека для створення інтерфейсів користувача. Розроблена компанією Facebook, вона дозволяє розробникам будувати великі вебдодатки, які можуть змінювати дані без перезавантаження сторінки [1].

Переваги React:

– компонентний підхід: React дає можливість розбивати інтерфейс на незалежні частини (компоненти), що полегшує розробку, тестування та підтримку коду;
– віртуальний DOM: React використовує віртуальний DOM, що робить оновлення інтерфейсу швидкими та ефективними;
– декларативний синтаксис: розробники описують, як має виглядати інтерфейс, а React сам оновлює DOM, що спрощує розробку;
– велика спільнота та екосистема: React має велику спільноту розробників та багато готових бібліотек і інструментів.

Застосування React в інтернет-магазині:

– створення інтерактивних елементів (кнопки, форми, фільтри);
– відображення каталогу товарів;
– реалізація кошика та процесу оформлення замовлення;
– управління станом додатку.

1.2.2 Node.js

Node.js – це JavaScript-середовище виконання, яке дозволяє запускати JavaScript-код на сервері. Воно базується на рушії V8 JavaScript від Google Chrome [2].

Переваги Node.js:

– асинхронність та неблокуючий I/O: Node.js добре підходить для додатків, які обробляють багато одночасних з’єднань, таких як інтернет-магазини;

– javaScript на сервері та клієнті: використання JavaScript на обох сторонах розробки спрощує розробку та обмін кодом;
– NPM (Node Package Manager): величезна кількість готових пакетів та бібліотек, які можна використовувати у проєктах;
– масштабованість: Node.js дозволяє легко масштабувати додатки.

Застосування Node.js в інтернет-магазині:

– обробка запитів від клієнтів;
– робота з базою даних;
– реалізація API (Application Programming Interface) для взаємодії з front-end;
– аутентифікація та авторизація користувачів.

1.3 Огляд баз даних для інтернет-магазину

База даних є вкрай важливим складником будь-якого інтернет-магазину. Вона використовується для збереження всієї інформації, потрібної для функціонування магазину, включаючи дані про товари, користувачів, замовлення, платежі та іншу важливу інформацію.

При виборі бази даних для інтернет-магазину важливо враховувати такі фактори:

– тип даних: які типи даних потрібно буде зберігати (текст, числа, зображення, тощо);
– розмір бази даних: який обсяг даних передбачається зараз і в майбутньому;
– швидкість: наскільки швидко потрібен доступ до даних;
– масштабованість: чи можна буде легко збільшити обсяг бази даних та кількість користувачів;
– надійність: наскільки важливо забезпечити збереження даних та запобігти їх втраті;
– вартість: яка ціна ліцензії, обслуговування та підтримки бази даних.

Існує два основних типи баз даних – реляційні бази даних та NoSQL бази даних.

Реляційні бази даних (RDBMS):

– зберігають дані у вигляді таблиць з рядками та стовпцями;
– використовують SQL (Structured Query Language) для запитів та управління даними;
– добре підходять для зберігання структурованих даних, таких як дані про товари, користувачів та замовлення;
– приклади: MySQL, PostgreSQL, Oracle, Microsoft SQL Server.

NoSQL бази даних:

– зберігають дані у різних форматах, таких як JSON, XML, ключ-значення, графи;
– не використовують SQL, а часто мають свої власні мови запитів;
– добре підходять для зберігання неструктурованих або напівструктурованих даних, а також для великих обсягів даних та високої швидкості доступу;
– приклади: MongoDB, Cassandra, Redis, Couchbase.

Для інтернет-магазину часто використовуються як реляційні, так і NoSQL бази даних, в залежності від конкретних потреб проєкту. Наприклад, для зберігання основних даних про товари, користувачів та замовлення може використовуватись реляційна база даних, а для зберігання даних про сесії користувачів, кешування або обробки великих обсягів даних – NoSQL база даних.

1.4 Аналіз вимог до інтернет-магазину

Перед розробкою інформаційної системи інтернет-магазину необхідно чітко визначити її функціональні та нефункціональні вимоги. Це дозволить сформувати цілісне бачення системи та уникнути проблем у процесі реалізації.

Функціональні вимоги. Інтернет-магазин бургерів повинен забезпечувати користувачам набір основних функцій для перегляду асортименту, оформлення замовлень, а також адміністрування контенту.

Основні функціональні вимоги включають:

- реєстрація та авторизація користувача: система повинна дозволяти створювати облікові записи, входити в систему за допомогою логіна та пароля, виходити з облікового запису;
- перегляд каталогу бургерів: користувачі повинні мати можливість переглядати список доступних бургерів із фотографією, назвою, ціною, складом і вагою;
- фільтрація та сортування: користувачі можуть фільтрувати бургери за категоріями (наприклад: класичні, веганські, гострі) та сортувати за ціною або популярністю;
- детальна інформація про товар: при натисканні на картку бургера відкривається модальне вікно з докладним описом складу та ваги;
- додавання до кошика та оформлення замовлення: користувач може додати обрані бургери до кошика, переглянути його вміст, змінити кількість позицій та оформити замовлення;
- адміністративна частина: можливість додавання нових бургерів, редагування наявних, перегляд списку замовлень.

2 ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ

2.1 Проєктування структури інтернет-магазину

Проєктування структури інтернет-магазину включає два основних аспекти: інформаційну архітектуру користувацького інтерфейсу та структуру бази даних для збереження інформації про користувачів, товари та замовлення.

2.1.1 Інформаційна архітектура

Інформаційна архітектура визначає логіку побудови навігації, структуру сторінок і розділів сайту.

У проєкті інтернет-магазину бургерів передбачено основні сторінки та функціональні компоненти, розглянемо їх детальніше.

Основні розділи та сторінки:

– /login – сторінка входу в систему;
– /register – сторінка реєстрації нового користувача;
– /home – головна сторінка з каталогом бургерів;
– /burger/:id – сторінка з детальною інформацією про конкретний бургер;
– /cart – сторінка кошика із доданими товарами.

Навігаційне меню (Navbar):

– головна;
– кошик (із лічильником позицій);
– кнопка входу/виходу (залежно від авторизації);
– ім’я користувача (після входу).

Інтерфейс навігаційного меню представлено на рисунку 2.1.

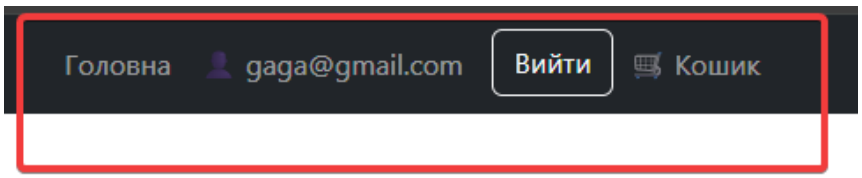


Рисунок 2.1 – Навігаційне меню

Додаткові компоненти:

– фільтр категорій (класичні / гострі / веганські) (див. рис. 2.2);

– модальне вікно для детального перегляду бургера (див. рис. 2.3).

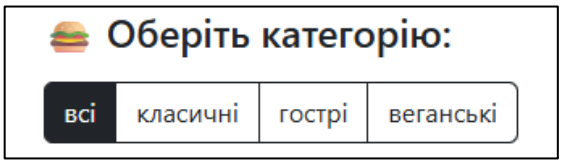


Рисунок 2.2 – Фільтр категорій

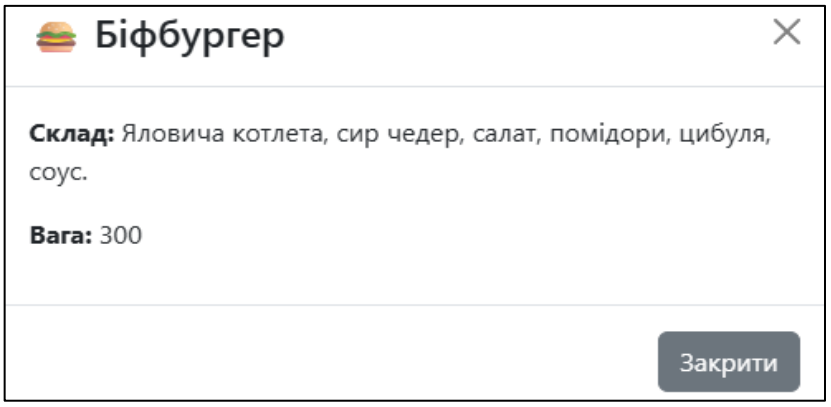


Рисунок 2.3 – Модальне вікно

2.1.2 Створення діаграми бази даних

Розробка діаграми бази даних складається з кількох послідовних етапів. Кожен з них має своє приначення і разом вони дають змогу створити чітке відображення структури даних.

Збір вимог: спочатку потрібно зрозуміти, яка інформація зберігатиметься в базі даних, та як різні частини цієї інформації взаємопов’язані.

Створення концептуальної моделі: далі розробляється загальна схема, яка показує основні сутності (наприклад, «клієнт», «товар», «замовлення») та їхні взаємозв'язки. Важливо, щоб ця модель була зрозумілою та логічною.

Створення логічної моделі: на цьому етапі концептуальна модель доповнюється деталями, важливими для бази даних. Це можуть бути типи даних (наприклад, текст, число, дата) та обмеження на дані (наприклад, обов'язковість поля).

Створення фізичної моделі: логічна модель перетворюється на фізичну, яка враховує особливості конкретної системи керування базами даних (СУБД). Тут визначаються конкретні типи даних, ключі та індекси.

Розгортання бази даних: після того, як діаграма готова, створюється сама база даних. Це містить створення таблиць та інших об'єктів відповідно до розробленої структури.

Налагодження бази даних: останній етап – оптимізація бази даних для ефективної роботи. Сюди входить створення індексів для швидкого пошуку, встановлення зв'язків між таблицями та налаштування прав доступу. Також може здійснюватися оптимізація запитів для підвищення продуктивності (див. рис. 2.4).

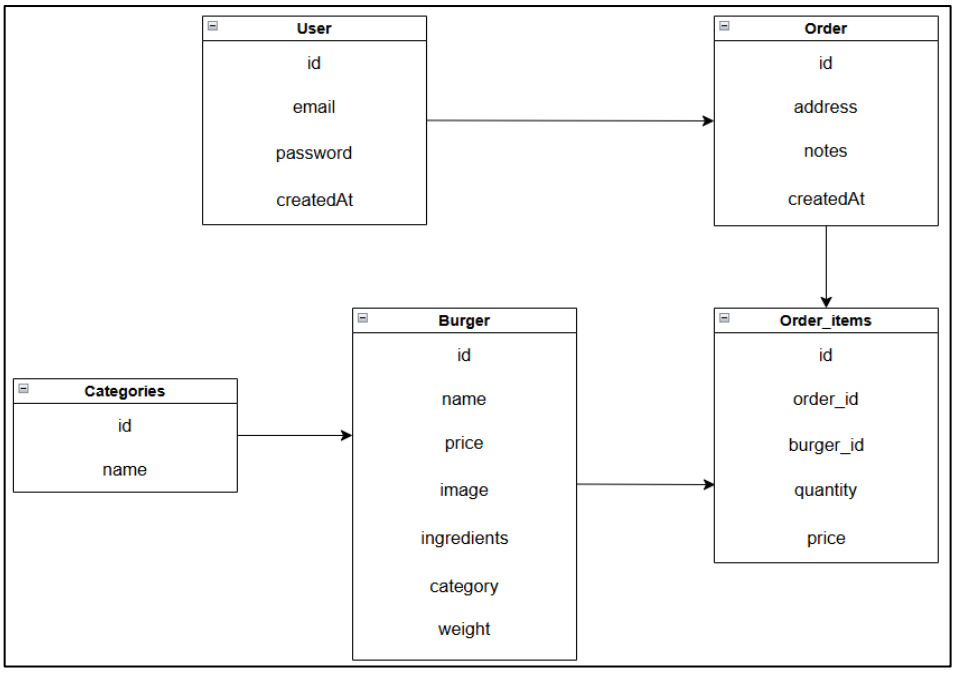


Рисунок 2.4 – Діаграма бази даних

2.2 Розробка макетів та прототипів

Для початкового етапу проєктування UI були створені wireframes (каркасні макети) сторінок системи. Це дозволило продумати структуру інтерфейсу, розміщення елементів та логіку взаємодії користувача із системою до початку розробки коду.

Основні сторінки, для яких були створені прототипи:

- головна сторінка каталогу бургерів (Home): перелік бургерів у вигляді карток, фільтрація за категоріями, кнопка «Додати в кошик»;
- кошик (Cart): перелік обраних бургерів із кнопками видалення, підрахунок загальної суми;
- сторінка оформлення замовлення (Checkout): форма для введення адреси доставки, коментарів до замовлення, кнопка підтвердження;
- сторінки реєстрації та авторизації: стандартні форми входу/реєстрації із валідацією.

3 РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ

3.1 Розробка Back-end

Створення інтернет-магазину на Node.js виконується в кілька етапів [3]:

– этап № 1: встановлення Node.js;
– этап № 2: вибір фреймворка;
– этап № 3: налаштування бази даних;
– этап № 4: розробка API;
– этап № 5: ініціалізація проєкту.

Ініціалізація проєкту. Створюємо в директорії теку для серверу і виконуємо ініціалізацію проєкту [4] (див. рис. 3.1).

```
PS C:\Users\0112d\Desktop\diplommm\server> npm init -y
```

Рисунок 3.1 – Ініціалізація проєкту

Це робиться для того, щоб провести через серію питань для налаштування файлу package.json, який містить інформацію про проєкт . Тобто дозволяє налаштувати вихідну структуру проєкту. Це може допомогти зробити код організованим та зрозумілим для інших програмістів.

Після цього встановлюються залежності, які потрібні на проєкті. Для даного проєкту встановлюється jsonwebtoken для реєстрації та авторизації користувачів, dotenv для конфігурації підключення до БД і prisma client [5] (див. рис. 3.2).

```
PS C:\Users\0112d\Desktop\diplommm\server> npm install @prisma/client
```

Рисунок 3.2 – Встановлення prisma

У межах розробки серверної частини інтернет-магазину бургерів було обрано бібліотеку Prisma як інструмент для взаємодії з базою даних

PostgreSQL. Prisma – це сучасна ORM (Object-Relational Mapping) для Node.js і TypeScript, яка значно спрощує роботу з реляційними базами даних.

Переваги використання Prisma:

– типізація та автодоповнення: Prisma автоматично генерує типи TypeScript, що дозволяє виявляти помилки на етапі розробки й забезпечує зручну підтримку автодоповнення у редакторі коду;
– інтуїтивний синтаксис: запити до бази даних пишуться зрозумілою мовою (наприклад, prisma.user.findMany()), що є більш безпечним і зручним порівняно з написанням «сирих» SQL-запитів;
– автоматична міграція: Prisma підтримує зручну систему міграцій, яка дозволяє оновлювати структуру бази даних без ризику втрати даних;
– просте налаштування зв’язків між таблицями: завдяки Prisma Schema можна легко описати структуру таблиць, ключі, зовнішні зв’язки (наприклад, зв’язок між користувачем і замовленнями), що забезпечує цілісність даних;
– підтримка PostgreSQL: Prisma добре інтегрується з PostgreSQL, яка використовується як основна СУБД у проєкті [6].

Prisma виконує такі функції у розроблюваному інтернет-магазині:

– створення та оновлення структури бази даних за допомогою схем (schema.prisma);
– генерація типів і клієнта для виконання запитів;
– реалізація CRUD-операцій (створення, читання, оновлення, видалення) для моделей User, Burger, Order, CartItem;
– підтримка реляційних запитів – наприклад, отримання всіх замовлень певного користувача або товарів у кошику.

3.2 База даних

PgAdmin – це провідний інструмент керування з відкритим вихідним кодом для Postgres, бази даних з відкритим вихідним кодом [7].

Цей інструмент надає клієнт і вебінтерфейс, до якого можна підключатися віддалено. У разі використання в локальній мережі інтернет не потрібен. Також можна надати доступ своїй команді, якщо вона є. Треба буде просто роздати логіни та паролі для підключення до інтерфейсу.

Створюємо новий сервер через pgAdmin (див. рис. 3.3 і 3.4).

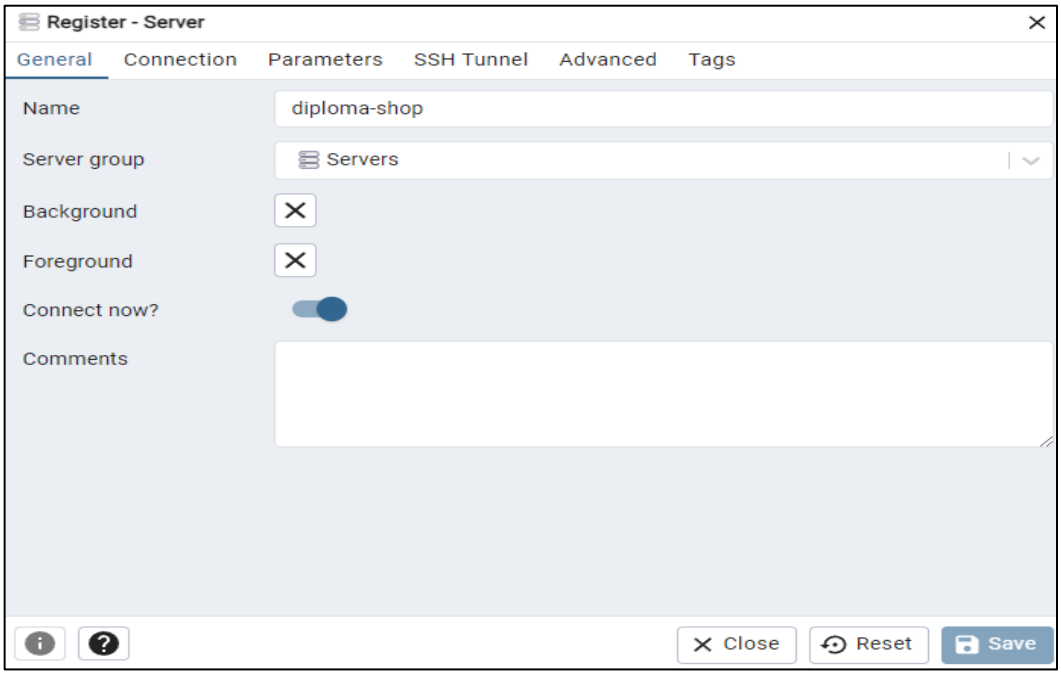


Рисунок 3.3

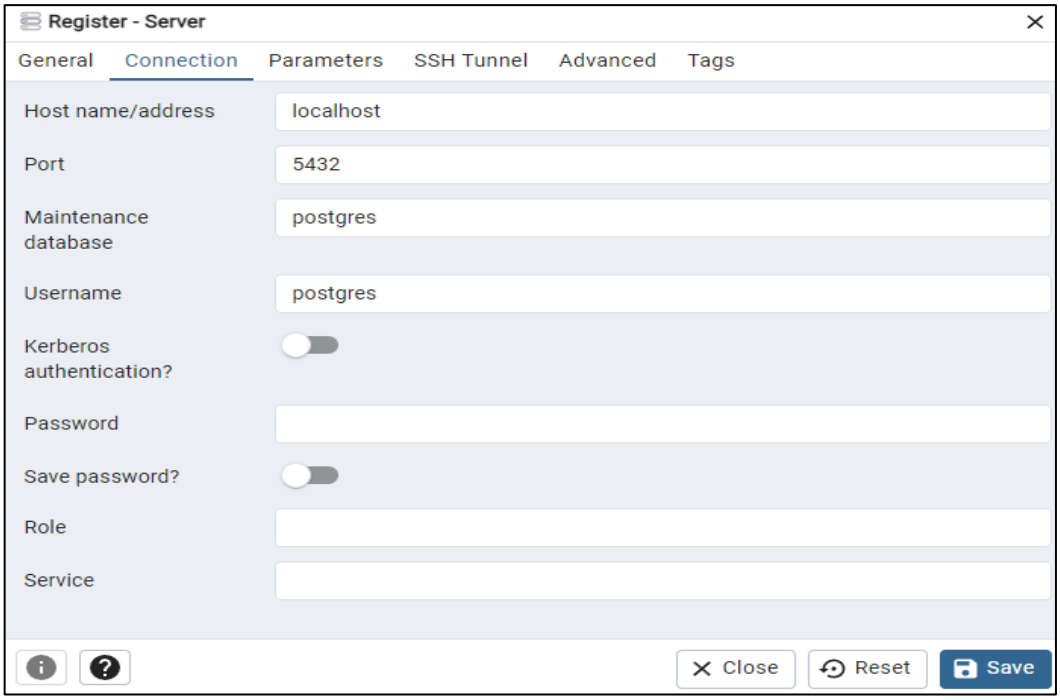


Рисунок 3.4

Після цього створюємо базу даних під назвою mydb (див. рис. 3.5 і 3.6).

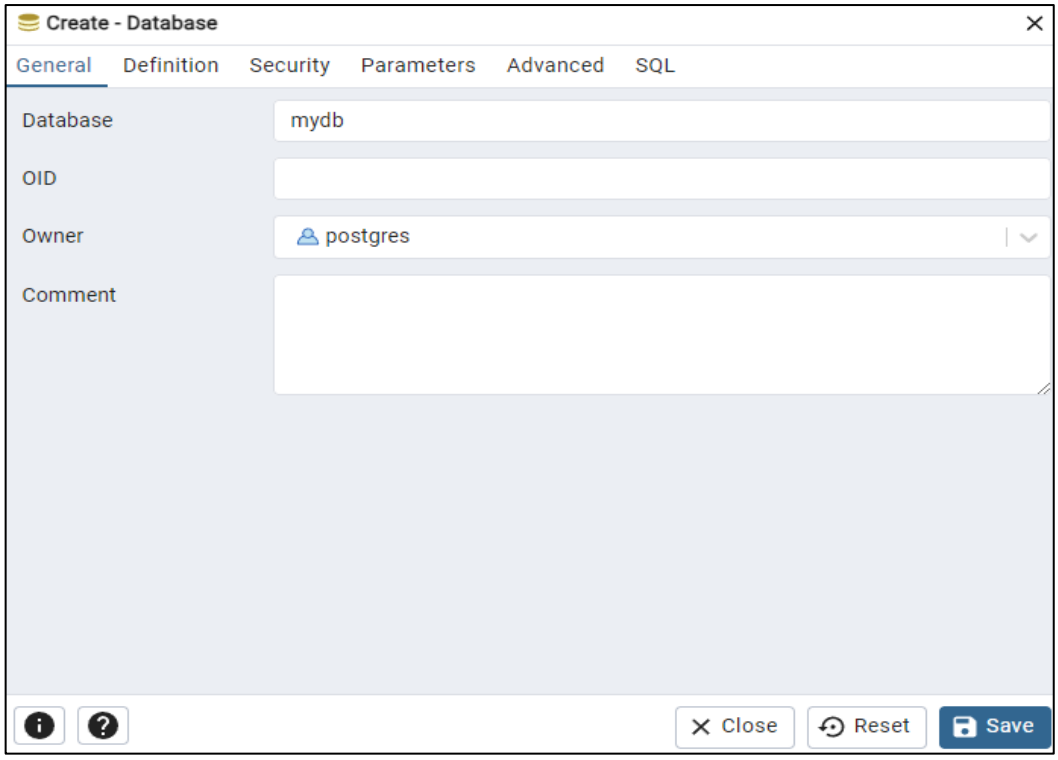


Рисунок 3.5 – Створення бази даних

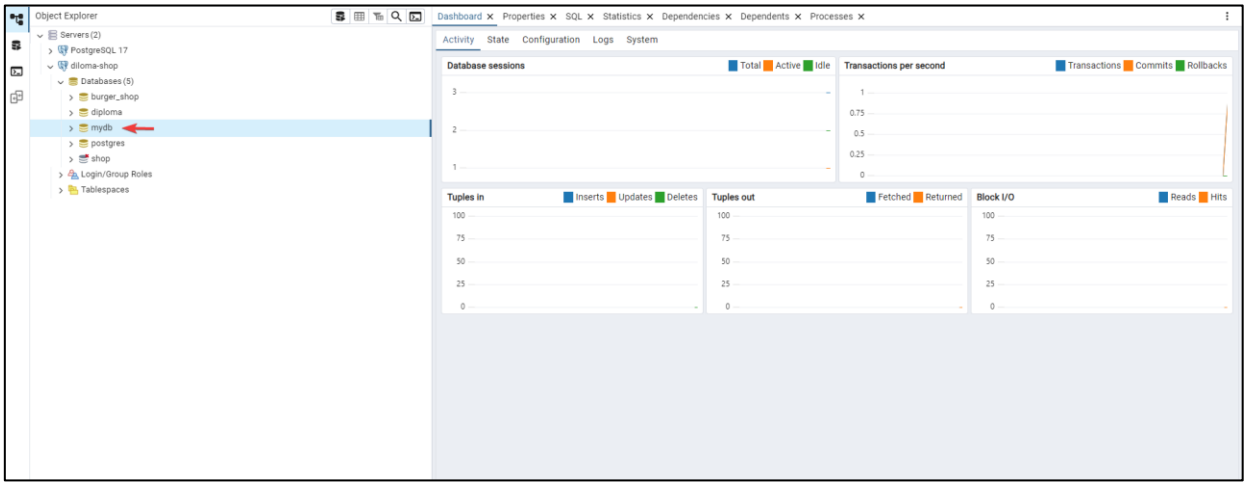


Рисунок 3.6 – Інтерфейс програми pgAdmin

3.2.1 Підключення до бази даних

Заходжу в файл конфігурації .env і запускаю дані моєї бази даних (див. рис. 3.7).



Рисунок 3.7 – Файл .env

Які дані тут введені:

– postgres – ім’я користувача;
– 1242 – пароль;
– localhost – host name;
– 5432 – порт серверу;
– mydb – назва бази даних.

Також було введено JWT_SECRET. Це секретний ключ, який використовується на бекенді для:

– підписування токенів при авторизації користувача: коли користувач успішно входить у систему (логін), сервер створює токен з його ID (та іншими даними) і підписує його цим секретним ключем;
– перевірки токена при подальших запитах: при кожному запиті до захищених маршрутів сервер перевіряє, чи токен валідний і чи був підписаний цим же ключем.

У інтернет-магазині JWT-токени використовуються для того, щоб:

– контролювати доступ користувачів до особистих даних (замовлення, кошик тощо);
– захищати API маршрути (наприклад, /orders, /cart) від неавторизованих користувачів;
– зберігати сесію користувача після логіну без потреби зберігати дані у сесії чи cookies.

3.2.2 Розробка моделей бази даних

Заходжу у schema.prisma і створюю моделі, визначаючи їх поля та типи даних (див. рис. 3.8).

server > prisma > schema.prisma > ...

```
Generate
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id        Int      @id @default(autoincrement())
  email     String   @unique
  password  String
  createdAt DateTime @default(now())
}

model Burger {
  id        Int     @id @default(autoincrement())
  name      String
  price     Int
  image     String
  category  String
  ingredients String?
  weight     Int?
}

model Order {
  id        Int      @id @default(autoincrement())
  address   String
  notes     String?
  createdAt DateTime @default(now())
}
```

Рисунок 3.8 – Файл .env

Блок *generator client* вказує Prisma створити клієнтський JavaScript/TypeScript код для доступу до бази даних.

Блок *datasource db* описує джерело даних – PostgreSQL базу даних. URL до неї береться з .env файлу.

Блок User описує таблицю користувачів:

– id – унікальний ідентифікатор (автоматично зростає);

– email – унікальна електронна адреса користувача;

– password – хешований пароль;

– createdAt – дата створення облікового запису.

Блок Burger описує товари (бургери) в інтернет-магазині:

– name, price, image, category – основні поля для каталогу;

– ingredients – необов’язкове поле з описом інгредієнтів;

– weight – вага бургера (необов’язкове поле).

Блок Order описує замовлення:

– address – адреса доставки;

– notes – додаткові побажання (необов’язкове);

– createdAt – час оформлення замовлення.

Schema.prisma описує повністю бізнес-логіку: користувачів, бургери та замовлення, автоматизує роботу з базою через Prisma Client і дозволяє швидко створювати міграції та структуру бази без написання SQL вручну.

3.3 Реалізація API для основних операцій

Для взаємодії між клієнтською частиною (React) та сервером було реалізовано REST API на базі фреймворку Express.js. Це API забезпечує виконання основних CRUD-операцій (Create, Read, Update, Delete) для трьох ключових сутностей системи – користувачів, товарів (бургерів) та замовлень.

Користувачі (Users):

– POST /api/auth/register – створення нового користувача (реєстрація) (валідація електронної пошти, хешування паролю за допомогою bcrypt, збереження у базу через Prisma);

– POST /api/auth/login – автентифікація користувача (перевірка пароля, генерація JWT-токена);

– GET /api/users/:id – отримання інформації про користувача за ID;

– DELETE /api/users/:id – видалення користувача (тільки за потреби, наприклад, адміністратором).

Товари (Burgers):

– GET /api/burgers – отримання списку всіх бургерів;

– GET /api/burgers/:id – отримання інформації про окремий бургер;

– POST /api/burgers – створення нового бургера (для адміністратора);

– PUT /api/burgers/:id – редагування інформації про бургер;

– DELETE /api/burgers/:id – видалення бургера.

Замовлення (Orders):

– POST /api/orders – створення нового замовлення з даними користувача та вмістом кошика;
– GET /api/orders – отримання списку замовлень (може бути обмежено для адміністратора);
– GET /api/orders/:id – отримання деталей окремого замовлення;
– DELETE /api/orders/:id – скасування замовлення (опційно).

Особливості реалізації API:

– всі маршрути реалізовані через модулі Express і підключені в server/index.js;
– для захисту приватних маршрутів використовується middleware перевірки JWT (verifyToken);
– робота з базою даних реалізована через Prisma ORM, що забезпечує безпечну та ефективну взаємодію з PostgreSQL;
– структура відповіді сервера дотримується принципів REST – використовуються відповідні HTTP-коди, а відповідь містить зрозумілі об’єкти або повідомлення про помилки.

Інструменти та технології:

– Express.js – для створення серверного застосунку та маршрутизації;
– Prisma – для взаємодії з базою даних PostgreSQL;
– bcrypt – для хешування паролів;
– jsonwebtoken – для створення та перевірки токенів авторизації;
– dotenv – для керування змінними середовища (наприклад, JWT_SECRET).

3.4 Створення сторінки авторизації та реєстрації

Для забезпечення доступу користувачів до функціоналу інтернет-магазину, таких як створення замовлень або доступ до кошика, були реалізовані сторінки реєстрації та авторизації.

Реєстрація користувача (див. рис. 3.9). Сторінка реєстрації дозволяє створити новий обліковий запис шляхом заповнення форми з такими полями:

– email;

– пароль.

Основні кроки:

– користувач вводить дані у форму;

– після підтвердження даних відправляється POST-запит на сервер (/api/register);

– сервер перевіряє унікальність email, хешує пароль (bcrypt) і створює нового користувача в базі даних;

– у відповідь повертається JWT-токен, який зберігається на клієнті (наприклад, у localStorage).

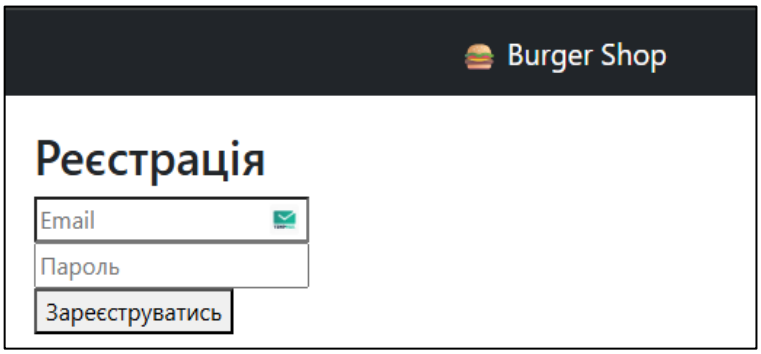


Рисунок 3.9 – Сторінка реєстрації

Авторизація користувача (див. рис. 3.10). На сторінці входу користувач заповнює:

– email;

– пароль.

Логіка роботи:

а) після заповнення форми надсилається POST-запит на /api/login;

б) сервер перевіряє наявність користувача та валідність пароля;

в) у разі успішної авторизації повертається JWT-токен;

г) після авторизації користувача перенаправляє на головну або іншу доступну сторінку.

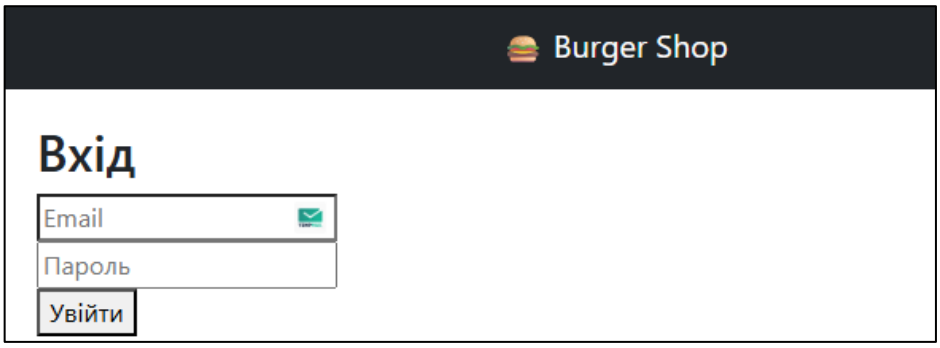


Рисунок 3.10 – Сторінка авторизації

Інтерфейс (UI). Для зручності користувача реалізовано:

– інтуїтивно зрозумілий інтерфейс з кнопками «Зареєструватися» / «Увійти»;
– валідацію форм (перевірка на порожні поля, коректність email);
– повідомлення про помилки (наприклад, «Неправильний пароль» або «Email вже зайнятий»).

Захист маршрутів. Після входу з токеном, користувач отримує доступ до захищених сторінок. Без валідного токена маршрути автоматично перенаправляють на сторінку авторизації.

Технології:

– React – для реалізації компонентів сторінок;
– React Router – для навігації між сторінками;
– Axios – для надсилання HTTP-запитів;
– JWT – для авторизації;
– Bootstrap – для швидкого оформлення інтерфейсу.

3.5 Створення головної сторінки

Головна сторінка є центральною частиною інтернет-магазину бургерів. Вона призначена для відображення доступного асортименту продукції, фільтрації бургерів за категоріями, перегляду інформації про товар, а також додавання обраного товару до кошика (див. рис. 3.11).

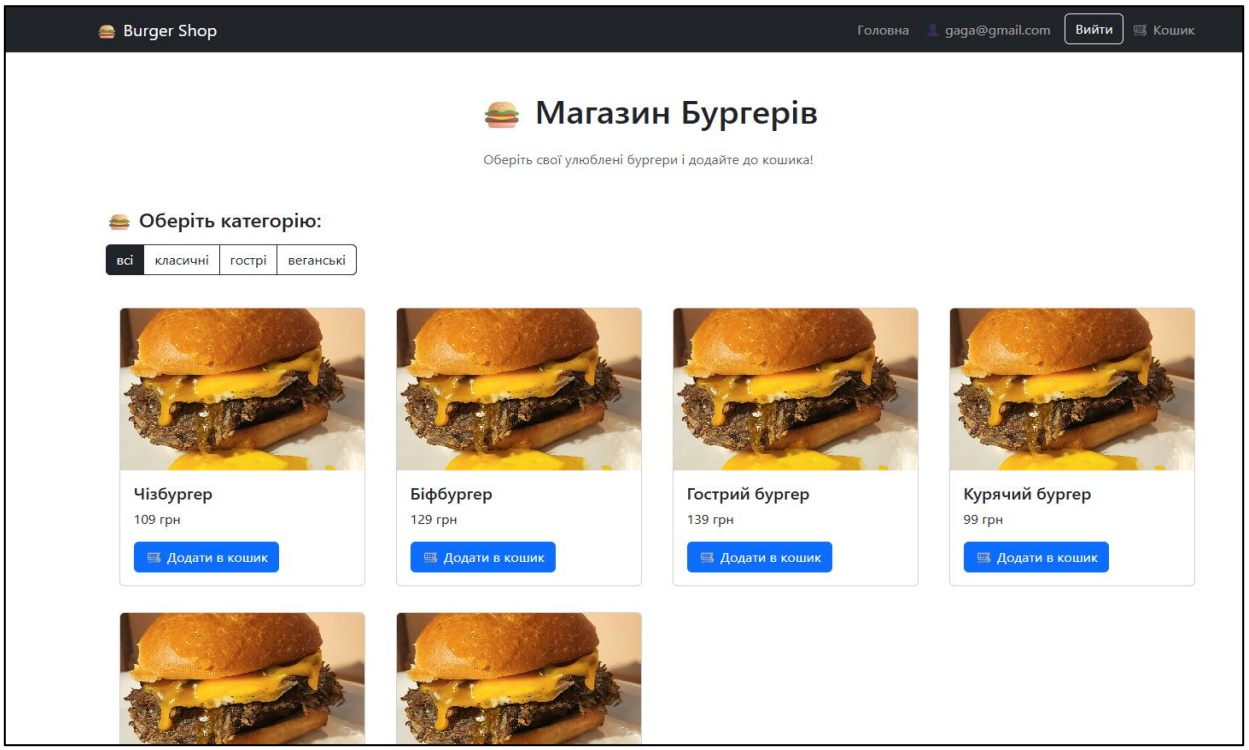


Рисунок 3.11 – Головна сторінка

Функціональні можливості головної сторінки:

– завантаження та виведення списку бургерів з бази даних;
– відображення товарів у вигляді карток;
– фільтрація за категоріями: класичні, веганські, гострі;
– модальне вікно з детальною інформацією (склад, вага);
– кнопка «Додати в кошик»;
– адаптивна сітка, зручна для перегляду на мобільних пристроях.

Опис реалізації. На фронтенді реалізація головної сторінки побудована за допомогою бібліотеки React. Для виведення списку бургерів використано компонент BurgerList, який отримує дані з бекенду через HTTP-запит (axios.get('/api/burgers')). Кожен бургер відображається у вигляді компонента BurgerCard. ***Компонент містить***:

– назву бургера;
– фото;
– ціну;
– кнопку для додавання до кошика;
– подію відкриття модального вікна з детальним описом.

Для зручності користувачів реалізовано фільтрацію товарів за категоріями, яка дозволяє швидко знайти потрібний тип бургерів.

Інтерфейс сторінки:

а) категорії реалізовані у вигляді кнопок (ButtonGroup) з можливістю активного стану;

б) бургери виводяться у сітці (Grid) з використанням Bootstrap (Row, Col) або Flexbox;

в) модальне вікно відкривається при кліку на картинку бургера і містить:

– склад інгредієнтів;

– вагу бургера;

– ціну.

3.6 Тестування

Процес випробовування інформаційної системи являє собою перевірку та аналіз дієвості функціонування програмних компонентів, що складають цю систему. Основне його завдання – переконатися, що система працює правильно та відповідає визначеним вимогам і сподіванням кінцевих користувачів.

В рамках випробовування створюються особливі сценарії та вхідні дані для оцінки різних характеристик системи, зокрема функціональності, стабільності, продуктивності, безпеки та сумісності. Виконання цих сценаріїв дає змогу виявити потенційні помилки й недоліки в програмному забезпеченні, аби усунути їх ще до запуску системи в реальних умовах експлуатації.

Після запуску у браузері відкривається localhost:3000/login, на якому треба авторизуватися або зареєструватися. Спробую зареєструватися (див. рис. 3.12).



Рисунок 3.12 – Спроба зареєструватися

Я увійшов під користувачем dima@gmail.com (див. рис. 3.13) і з’явився новий user (див. рис. 3.14).

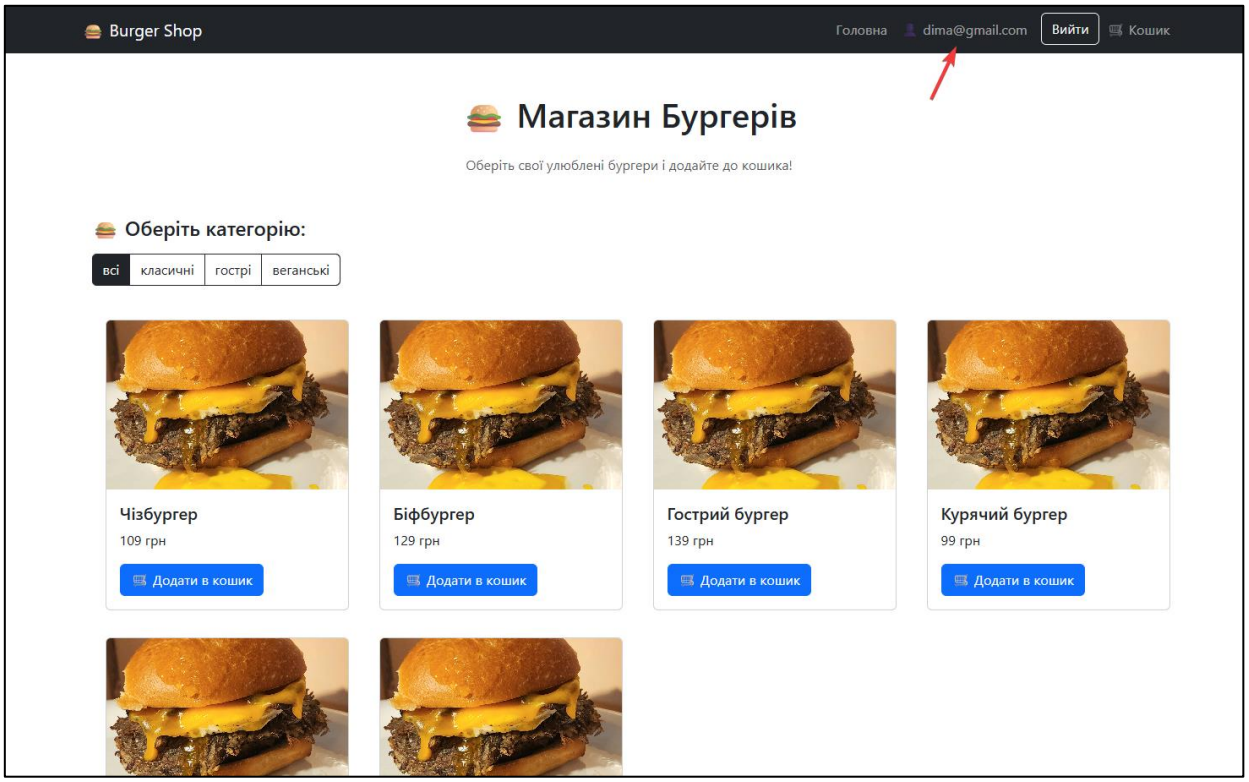


Рисунок 3.13 – Після авторизації

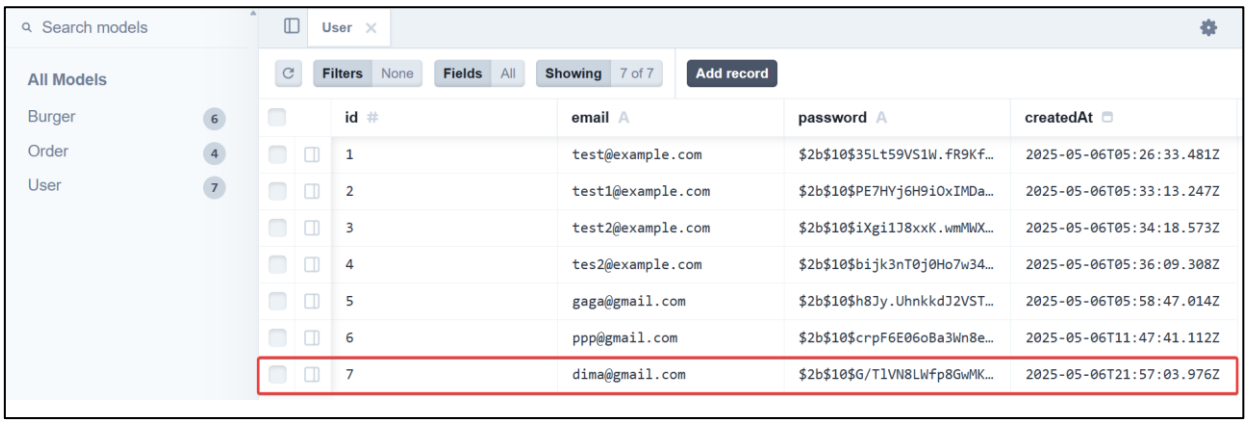


Рисунок 3.14 – Новий user

На головній сторінці бачу 6 різних бургерів. Протестую кнопки які відповідають за вибір категорії (див. рис. 3.15 – 3.17).

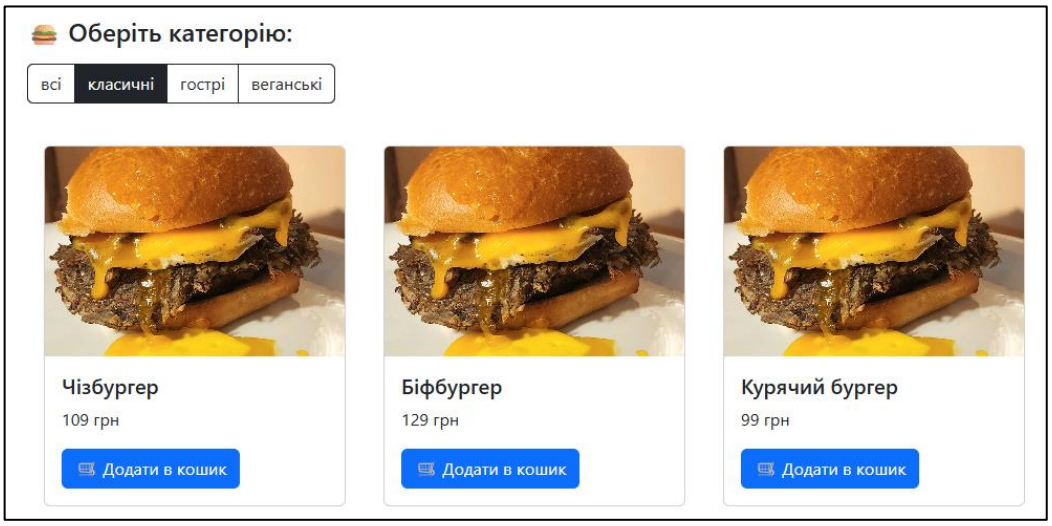


Рисунок 3.15 – Класичні бургери

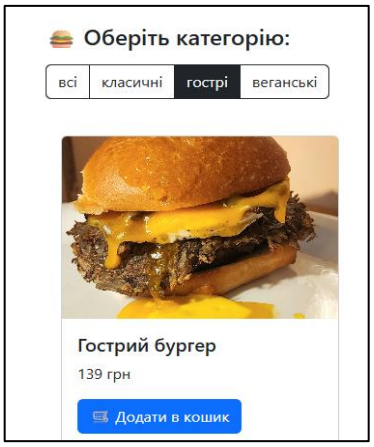


Рисунок 3.16 – Гострі бургери

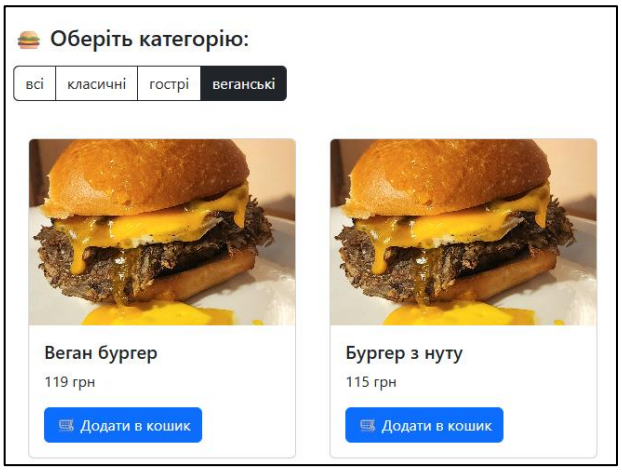


Рисунок 3.17 – Веганські бургери

Тепер протестую модальне вікно з інформацією про товар для випадкових двух бургерів (див. рис. 3.18 – 3.19). Для кожного бургера повинна бути різна інформація.

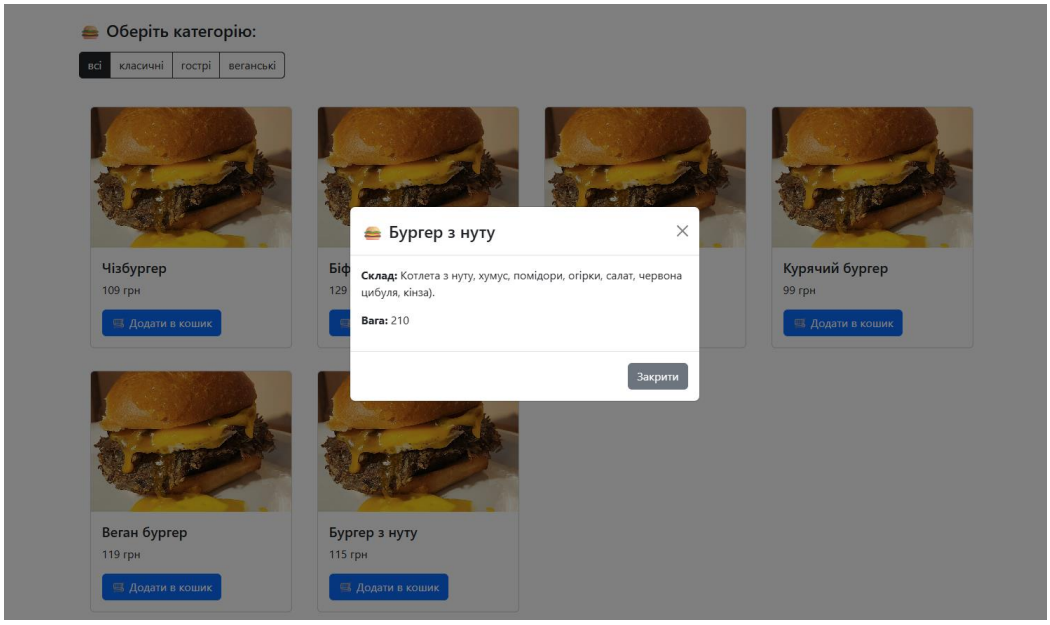


Рисунок 3.18 – Інформація про бургер з нуту

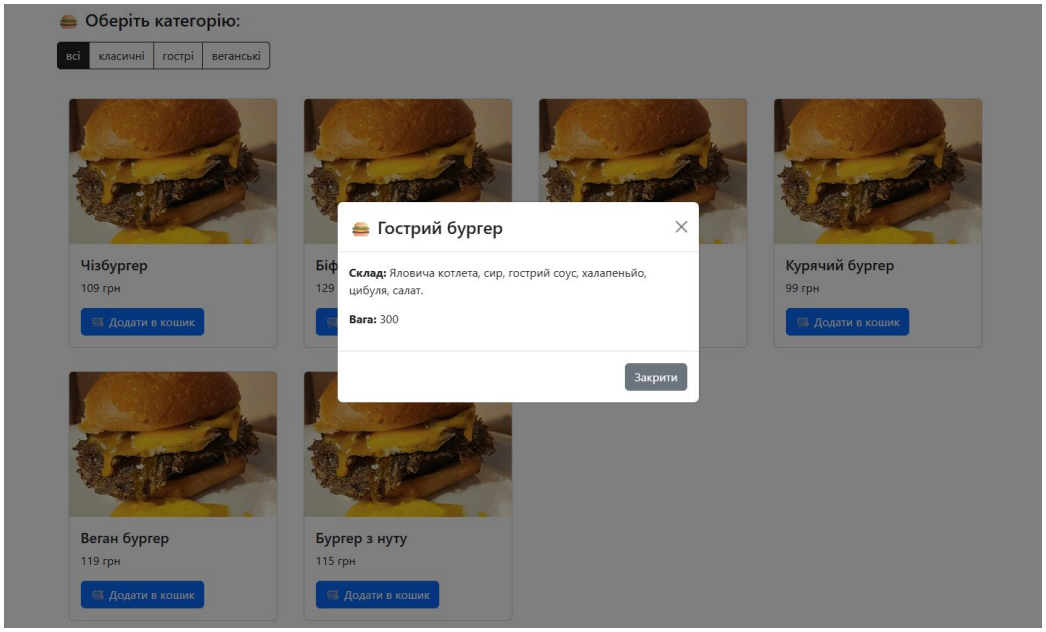


Рисунок 3.19 – Інформація про гострий бургер

Протестую додавання та видалення з кошику. На головній сторінці нажимаю на кнопку «Додати в кошик». Біля кошику з’явилось число товарів (див. рис. 3.20).

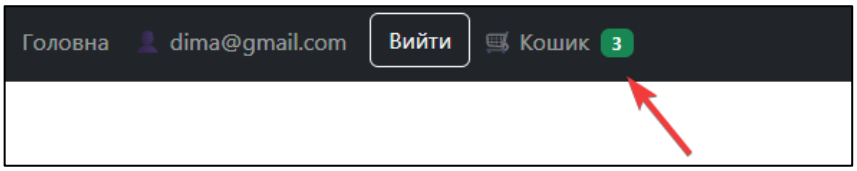


Рисунок 3.20 – Кількість товарів у кошику

Переходжу до кошика і бачу там всі 3 товари які були додані і загальна сума для оплати (див. рис. 3.21).

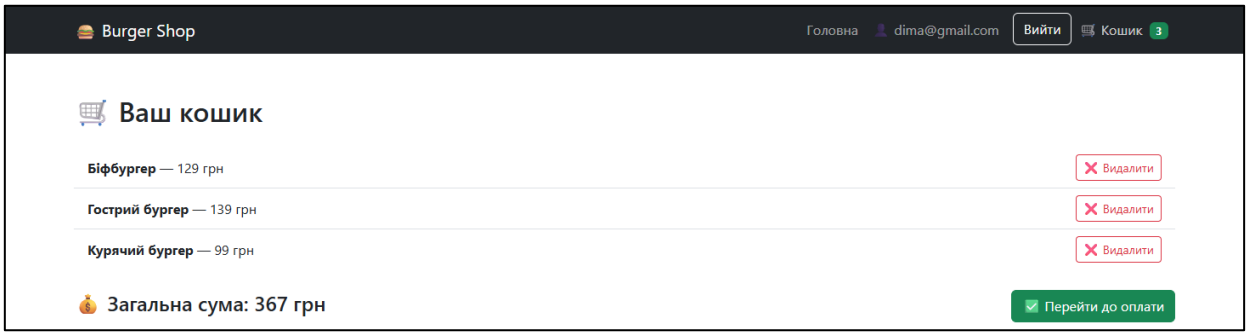


Рисунок 3.21 – Кошик

Протестую видалення товару з кошика (див. рис. 3.22 і 3.23).

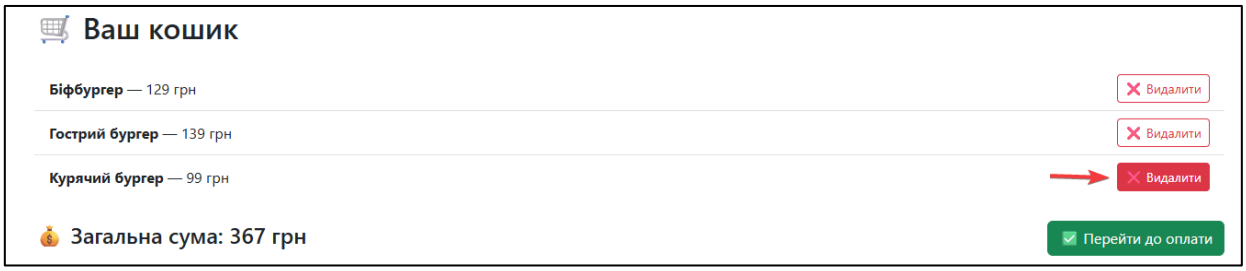


Рисунок 3.22 – Процес видалення



Рисунок 3.23 – Оновлений кошик

Протестую функцію «Перейти до оплати». Ввожу адресу доставки і підказку для кур’єра (див. рис. 3.24).

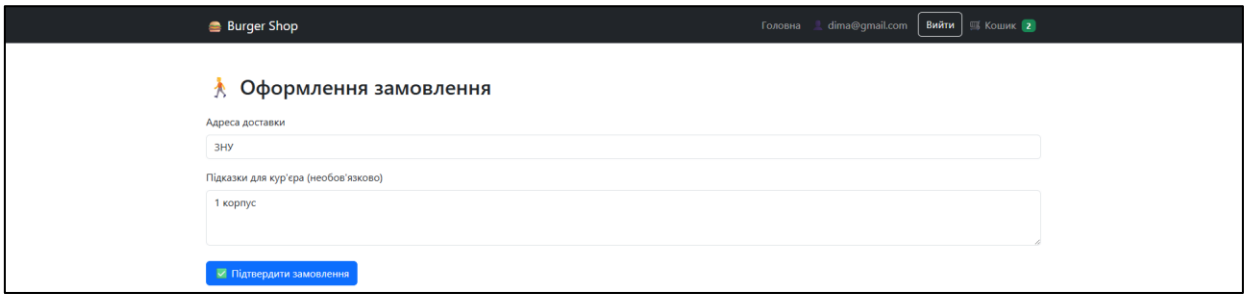


Рисунок 3.24 – Оформлення замовлення

Протестую кнопку «Підтвердити замовлення». Повинно з’явитись повідомлення, що замовлення успішно оформлено і кошик повинен стати порожнім (див. рис. 3.25).

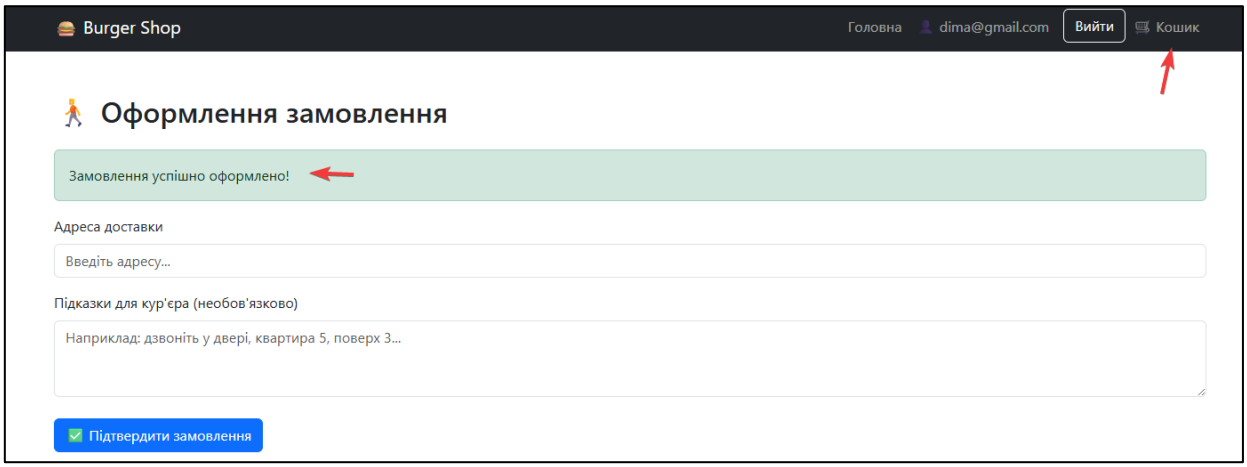


Рисунок 3.25 – Замовлення успішне

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було успішно реалізовано повнофункціональний вебзастосунок – інтернет-магазин бургерів, який дозволяє користувачам переглядати товари, додавати їх до кошика, оформляти замовлення та взаємодіяти із системою через зручний інтерфейс.

У ході розробки були досягнуті всі поставлені цілі:

– розроблено front-end частину на основі React з використанням компонентного підходу;
– створено back-end на Node.js з Express, який обробляє всі запити до бази даних;
– застосовано ORM Prisma для типобезпечної роботи з базою даних PostgreSQL;
– реалізовано JWT-аутентифікацію для захисту маршрутів та збереження сесій користувачів;
– забезпечено зручний та адаптивний користувацький інтерфейс із використанням Bootstrap;
– організовано структуру бази даних відповідно до потреб онлайн-магазину;
– реалізовано фільтрацію, модальні вікна, оформлення замовлення, збереження замовлень у базі.

Під час роботи над кваліфікаційною роботою були здобуті практичні навички з повноцінної розробки вебдодатку: від проєктування бази даних і API до створення клієнтського інтерфейсу та його з’єднання з бекендом.

Кваліфікаційна робота відповідає сучасним вимогам до інформаційних систем, є масштабованим та зручним для подальшого розвитку. У майбутньому можливе розширення функціоналу: інтеграція онлайн-оплати, реалізація адміністративної панелі, рейтингів, історії замовлень та push-сповіщень.

Результати роботи підтверджують доцільність вибору обраних технологій і обґрунтованість архітектурних рішень. Створений застосунок є реальним прикладом ефективної інформаційної системи, здатної вирішувати прикладні завдання електронної комерції.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке React JS і для чого він потрібен? *Dan.it Education.* URL: https://dan-it.com.ua/uk/blog/chto-takoe-react-js-i-dlja-chego-on-nuzhen/ (дата звернення: 01.04.2025).

2. Rud A. Що таке Node.js та для чого він потрібен? *HyperHost.* URL: https://hyperhost.ua/info/uk/shho-take-nodejs-ta-dlya-cogo-vin-potriben (дата звернення: 01.04.2025).

3. Створення веб-сервера на Node.js : покрокове керівництво. *FoxmindEd*. URL: https://foxminded.ua/stvorennia-veb-servera-na-node-js/ (дата звернення: 01.04.2025).

4. Документація npm-init. *npm Docs*. URL: https://docs.npmjs.com/cli/v8/commands/npm-init (дата звернення: 06.04.2025).

5. Prisma Documentation. *Prisma | Simplify working and interacting with databases*. URL: https://www.prisma.io/docs (дата звернення: 06.04.2025).

6. Prisma pros and cons. *Reddit*. URL: https://www.reddit.com/r/nextjs/comments/16ngm0d/prisma_pros_and_cons/ (дата звернення: 06.04.2025).

7. Як підключитися до бази даних PostgreSQL. *ALexHost SRL*. URL: https://alexhost.com/uk/faq/yak-pidklyuchytysya-do-bazy-danyh-postgresql/#31_pgAdmin (дата звернення: 06.04.2025).

ДОДАТОК А

Файл schema.prisma

```
generator client {
  provider = "prisma-client-js"
}

datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model User {
  id        Int      @id @default(autoincrement())
  email     String   @unique
  password  String
  createdAt DateTime @default(now())
}

model Burger {
  id        Int     @id @default(autoincrement())
  name      String
  price     Int
  image     String
  category  String
  ingredients String?
  weight     Int?
}
```

```
model Order {
  id        Int      @id @default(autoincrement())
  address   String
  notes     String?
  createdAt DateTime @default(now())
}
```

ДОДАТОК Б

Файл index.js з серверної частини

```
import express from 'express'
import cors from 'cors'
import dotenv from 'dotenv'
import { PrismaClient } from '@prisma/client'
import authRoutes from './routes/auth.js'
import burgerRoutes from './routes/burgers.js'
import orderRoutes from './routes/order.routes.js'

dotenv.config()
console.log('JWT_SECRET:', process.env.JWT_SECRET)
const prisma = new PrismaClient()
const app = express()

app.use(cors())
app.use(express.json())
app.use('/api/burgers', burgerRoutes)

app.use('/api/auth', authRoutes)
app.use(express.json())
app.use('/api/orders', orderRoutes)

const PORT = process.env.PORT || 5000
app.listen(PORT, () => console.log(`Server running on
http://localhost:${PORT}`))
```

ДОДАТОК В

Файл App.jsx з front-end частини

```
import { BrowserRouter, Routes, Route, Navigate } from 'react-router-dom'
import { useAuth } from './context/AuthContext'
import LoginPage from './pages/LoginPage'
import RegisterPage from './pages/RegisterPage'
import HomePage from './pages/HomePage'
import Navbar from './components/Navbar'
import CartPage from './pages/CartPage'
import { useCart } from './context/CartContext'
import CheckoutPage from './pages/CheckoutPage'

const PrivateRoute = ({ children }) => {
  const { user } = useAuth()
  return user ? children : <Navigate to="/login" />
}

const App = () => {
  const { cart, setCart } = useCart()

  const removeFromCart = (burger) => {
    setCart(cart.filter(item => item.id !== burger.id))
  }

  return (
    <BrowserRouter>
      <Navbar />
      <Routes>
```

```
        <Route path="/" element={<PrivateRoute><HomePage /></PrivateRoute>}
/>
        <Route path="/login" element={<LoginPage />} />
        <Route path="/register" element={<RegisterPage />} />
        <Route path="/cart" element={<CartPage
removeFromCart={removeFromCart} />} />
        <Route path="/checkout" element={<CheckoutPage />} />
      </Routes>
    </BrowserRouter>
  )
}

export default App
```

ДОДАТОК Г

Файл index.js з front-end частини

```
import React from 'react'
import ReactDOM from 'react-dom/client'
import App from './App'
import { AuthProvider } from './context/AuthContext'
import { CartProvider } from './context/CartContext'
import './index.css'
import 'bootstrap/dist/css/bootstrap.min.css'

const root = ReactDOM.createRoot(document.getElementById('root'))
root.render(
  <React.StrictMode>
    <CartProvider>
      <AuthProvider>
        <App />
      </AuthProvider>
    </CartProvider>
  </React.StrictMode>
)
```