МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА E-COMMERCE ПОРТАЛУ З ВИКОРИСТАННЯМ REACT ТА NODEJS»**

Виконав: студент 4 курсу, групи 6.1219-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
В.Е. Гугля
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Мильцев О.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення

Освітня програма програмна інженерія

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

_____ Лісняк А.О.
(підпис)
« 07 » 02 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Гуглі Владиславу Едуардовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка E-commerce порталу з використанням React та NodeJS

керівник роботи Мильцев Олександр Михайлович, доцент, к.ф.-м.н.
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 26 » січня 2023 року № 102-с

2. Строк подання студентом роботи 07.06.2023 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз вимог.
2. Проєктування.
3. Реалізація та тестування програмного забезпечення.

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

6. Консультанти розділів роботи

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

7. Дата видачі завдання 07.02.2023 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.	08.02.2023	
2.	Збір вихідних даних.	01.03.2023	
3.	Обробка методичних та теоретичних	20.03.2023	
	джерел.		
4.	Розробка першого розділу.	10.04.2023	
5.	Розробка другого розділу.	24.04.2023	
6.	Розробка третього розділу.	17.05.2023	
7.	Оформлення та нормоконтроль	01.06.2023	
	кваліфікаційної роботи.		
8.	Захист кваліфікаційної роботи.	21.06.2023	

Студент _____ В.Е. Гугля
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.М. Мильцев
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

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

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка E-commerce порталу з використанням React та NodeJS»: 67 с., 86 рис., 15 табл., 9 джерел.

AWS S3, DOCKER, FIGMA, JAVASCRIPT, NESTJS, NODE.JS, POSTMAN, PRISMA, REACT.JS, TYPESCRIPT.

Об’єкт дослідження – E-commerce портал з продажу побутових матеріалів.

Мета роботи: розробка E-commerce порталу з продажу побутових матеріалів.

Метод дослідження – аналітичний, проєктувальний, програмний.

В сучасному світі електронна комерція є одним з найбільш швидко зростаючих сегментів бізнесу. За останні роки споживачі все більше віддають перевагу онлайн-покупкам, оскільки вони зручні, доступні та пропонують широкий вибір товарів та послуг. Цей тренд створює великі можливості для підприємців, які бажають розширити свої бізнеси або запустити нові проєкти в галузі електронної комерції.

Дипломна робота присвячена розробці E-commerce сайту з використанням двох популярних технологій – React і Node.js. React – це потужна JavaScript бібліотека для створення користувацьких інтерфейсів, в той час як Node.js – це відкрите середовище виконання JavaScript, яке дозволяє розробникам створювати сучасні та масштабовані веб-додатки.

SUMMARY

Bachelor’s qualifying paper «Development of an E-commerce Portal using React and NodeJS»: 67 p., 86 figures, 15 tables, 9 references.

AWS S3, DOCKER, FIGMA, JAVASCRIPT, NESTJS, NODE.JS, POSTM AN, PRISMA, REACT.JS, TYPESCRIPT.

The object of research is an E-commerce portal for the sale of household materials.

Purpose: to develop an E-commerce portal for the sale of household materials.

Research methods: analytical, design, software.

In the modern world, e-commerce is one of the fastest-growing business segments. In recent years, consumers have increasingly preferred online shopping because it is convenient, affordable and offers a wide range of goods and services. This trend creates great opportunities for entrepreneurs who want to expand their businesses or launch new e-commerce projects.

This thesis is dedicated to the development of an E-commerce website using two popular technologies – React and Node.js. React is a powerful JavaScript library for creating user interfaces, while Node.js is an open-source JavaScript runtime that allows developers to create modern and scalable web applications.

ЗМІСТ

ВСТУП

В сучасному світі електронна комерція є одним з найбільш швидко зростаючих сегментів бізнесу. За останні роки споживачі все більше віддають перевагу онлайн-покупкам, оскільки вони зручні, доступні та пропонують широкий вибір товарів та послуг. Цей тренд створює великі можливості для підприємців, які бажають розширити свої бізнеси або запустити нові проєкти в галузі електронної комерції.

Дипломна робота присвячена розробці E-commerce сайту з використанням двох популярних технологій – React і Node.js. React – це потужна JavaScript бібліотека для створення користувацьких інтерфейсів, в той час як Node.js – це відкрите середовище виконання JavaScript, яке дозволяє розробникам створювати сучасні та масштабовані веб-додатки.

Метою дипломної роботи є розробка повноцінного E-commerce сайту, який надасть користувачам можливість зареєструватися, переглядати та вибирати товари, додавати їх у кошик, шукати товари по різним фільтрам, здійснювати покупки, відстежувати статус замовлень, і т.д .

Основними функціональними вимогами до E-commerce сайту є створення привабливого та інтуїтивно зрозумілого інтерфейсу, ефективного механізму пошуку, безпеки та надійності операцій, впровадження процесів аутентифікації і авторизації для запобігання несанкціонованого доступу деяких модулів порталу, а також можливості взаємодії зовнішніх сервісів різних платіжних систем.

У наступних розділах ми детальніше розглянемо методику та процес розробки E-commerce сайту з використанням React і Node.js, а також представимо практичні результати і висновки.

1 АНАЛІЗ ВИМОГ ПЗ

1.1 Опис проєкту

У цьому розділі дипломної роботи ми надаємо детальний опис E-commerce сайту, який ми розробляємо для продажу будматеріалів [1]. Наш сайт має на меті забезпечити зручний та швидкий доступ до різноманітних будівельних матеріалів для покупців, включаючи будівельні компанії, підрядників, приватних забудовників та всіх, хто зацікавлений у здійсненні будівельних проєктів.

1.1.1 Огляд функціональності сайту

При розробці проєкту замислювалися такі бізнес-функції:

- **реєстрація та аутентифікація користувачів**: користувачі зможуть створювати облікові записи, входити в систему та управляти своїм профілем;
- **пошук і фільтрація**: користувачі зможуть швидко знайти потрібні будматеріали за допомогою пошуку за назвою, категорією, сабкатегорією або іншими параметрами;
- **каталог товарів**: сайт буде мати широкий каталог будматеріалів, включаючи ціни, описи, технічні характеристики, зображення кожного товару;
- **кошик покупок**: користувачі зможуть додавати товари в кошик, керувати їх кількістю та здійснювати замовлення;
- **оплата та доставка**: сайт буде підтримувати безпечні та зручні способи оплати;
- **відгуки та рейтинги**: користувачі зможуть залишати відгуки та

оцінки для товарів, що допоможе іншим покупцям при прийнятті рішення про покупку;

– **особливості сайту**: інтеграція зі складською системою: Сайт буде підключений до системи управління складом, що дозволить оновлювати наявність товарів в режимі реального часу.

1.1.2 Технології та інструменти

– **фронтенд розробка**: для розробки користувацького інтерфейсу ми використаємо React, одну з найпопулярніших JavaScript бібліотек, для створення динамічних та інтерактивних елементів;
– **бекенд розробка**: для створення серверної логіки ми використаємо Node.js, що дозволить нам побудувати швидку та масштабовану інфраструктуру;
– **база даних:** ми будемо використовувати базу даних PostgreSQL, для зберігання і управління інформацією про товари, користувачів та замовлення.

Цей E-commerce сайт з продажу будматеріалів має на меті забезпечити зручну та ефективну платформу для покупців, щоб вони могли швидко знайти та замовити необхідні матеріали для будівництва. Розробка цього сайту дозволить нам отримати практичний досвід у розробці E-commerce рішень з використанням React і Node.js та сприятиме подальшому розвитку сфери електронної комерції в галузі будівельних матеріалів.

1.2 Use-cases

Виконуючи дипломну роботу, замислювалися наступні use-cases.

Підрядник, який шукає матеріали. Підрядник займається

будівництвом та потребує різних будматеріалів для своїх проєктів. Він відвідує E-commerce портал, використовуючи пошук та фільтри, щоб знайти потрібні матеріали за категоріями, брендами або характеристиками. Він переглядає описи, технічні характеристики та ціни, додає певні товари до кошика та робить замовлення з доставкою на будівельний об'єкт.

Будівельна компанія, яка здійснює оптові закупівлі. Велика будівельна компанія здійснює регулярні оптові закупівлі будматеріалів. Вони використовують E-commerce портал, щоб швидко знайти потрібні матеріали за оптовими цінами, порівняти варіанти від різних постачальників та здійснити більші обсяги замовлень через систему каталогу та кошика.

Приватний забудовник, який планує будівництво свого будинку. Приватна особа, яка має намір збудувати власний будинок, відвідує E-commerce портал, щоб знайти та придбати різноманітні будматеріали. Вона може шукати товари за специфічними вимогами та характеристиками, переглядати відгуки та рейтинги від інших покупців, а також отримувати рекомендації щодо оптимальних варіантів для свого будівельного проєкту.

Торгова компанія, яка пропонує будматеріали. Компанія, яка спеціалізується на торгівлі будматеріалами, використовує E-commerce портал як свою платформу для продажу товарів. Вони заводять свій каталог товарів на порталі, додають описи, зображення, ціни та наявність товарів. Вони також пропонують різні спеціальні пропозиції, акції та знижки для залучення клієнтів. Клієнти можуть робити замовлення безпосередньо на порталі, і компанія обробляє їх та організовує доставку.

1.3 User stories

На рисунках 1.1 – 1.28 представлено опис бізнес-логіки E-commerce порталу у вигляді user-stories.

Як користувач, я можу створити свій особистий аккаунт на порталі (реєстрація)
Design:
https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=696-2&t=y3b2qmWXiFg45KO5-4
Given – Користувач заходить на сайт і відкриває форму реєстрації.
And – Користувач вводить валідні данні у всі обов'язкові поля для створення аккаунту (Імя, Прізвище, Емейл, Пароль, Телефон).
When – Користувач клікає на кнопку «Sign Up».
Then – Дані юзера заносяться в базу даних; тепер користувач може авторизуватись, використовуючи новостворені емейл і пароль.

Рисунок 1.1 – Registration user-story

Як користувач, я можу увійти в свій особистий аккаунт (аутентифікація)
Design:
https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=1-581&t=y3b2qmWXiFg45KO5-4
Given – Користувач заходить на сайт і відкриває форму авторизації
And – Користувач вводить валідні данні у всі обов'язкові поля для авторизації (Емейл, Пароль)
When – Користувач клікає на кнопку «Sign In»
Then – Юзер проходить процес ідентифікації і авторизації і може користуватися своїм особистим аккаунтом та іншими функціями порталу

Рисунок 1.2 – Authentication user-story

Як користувач, я можу змінити мої особисті дані у особистому кабінеті
Design:
https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=1-1309&t=QTAaLzQ0w2MHdBci-4
Given – Користувач заходить на портал
And – Користувач проходить процес аутентифікації і відкриває особистий кабінет
And – Користувач переходить на вкладку «Personal Info»
When – Користувач змінює свої персональні дані (Імя, Прізвище, Емейл, Телефон) і клікає на кнопку «ОК»
Then – Портал оновлює персональні дані користувача в базі і виводить актуальну інформацію у особистому кабінеті

Рисунок 1.3 – "Update user info" user-story

Як користувач, я можу змінити мій пароль у особистому кабінеті

Given – Користувач заходить на портал

And – Користувач проходить процес аутентифікації і відкриває особистий кабінет

And – Користувач переходить на вкладку «Change Password»

When – Користувач вводить валідний старий пароль, валідний новий пароль, такий же підтверджуючий новий пароль і клікає на кнопку «ОК»

Then – Портал оновлює пароль користувача в базі, вилогінує його і пропонує аутентифікуватись ще раз з ново-оновленим паролем.

Рисунок 1.4 – “Reset password” user-story

Як користувач, я можу створити/оновити віртуальну кредиту картку у особистому кабінеті

Design:

https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=1-1206&t=QTAaLzQ0w2MHdBci-4

Given – Користувач заходить на портал

And – Користувач проходить процес аутентифікації і відкриває особистий кабінет

And – Користувач переходить на вкладку «Wallet»

When –Користувач вводить валідні данні кредитної картки (Номер, Термін дії, Валюта, CVC) і клікає на кнопку «ОК»

Then – Портал створює/оновлює крединту картку у базі даних і відображає її у особистому кабінеті; також користувач зможе обрати картку як засіб платежу на етапі оформлення замовлення

Рисунок 1.5 – “Create/update card” user-story

Як користувач, я можу додати/оновити мою адресу у особистому кабінеті

Given – Користувач заходить на портал

And – Користувач проходить процес аутентифікації і відкриває особистий кабінет

And – Користувач переходить на вкладку «Address»

When – Користувач вводить валідні данні адреси (Вулиця, Місто, Країна, Поштовий індекс) і клікає на кнопку «ОК»

Then – Портал створює/оновлює адресу у базі даних і відображає її у особистому кабінеті; також портал буде відображати адресу на етапі оформлення замовлення

Рисунок 1.6 – “Create/update user address” user-story

Як користувач, я можу продивитися список категорій і сабкатегорій

Design -

https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=124-499&t=QTAaLzQ0w2MHdBci-4

Given – Користувач заходить на сайт

When – Користувач відкриває каталог

Then – Портал відображає всі категорії, а також сабкатегорії які відносяться тільки до певних категорій

Рисунок 1.7 – “View categories/subcategories” user-story

Як користувач, я можу продивитися список продуктів за категорією/сабкатегорією

Given – Користувач заходить на сайт

And – Користувач відкриває каталог

When – Користувач обирає будь-яку категорію/сабкатегорію

Then – Портал відображає тільки ті продукти, які є у відповідно обраній категорії/сабкатегорії

Рисунок 1.8 – “View category/subcategory products” user-story

Як користувач, я можу відфільтрувати список продуктів за ім’ям або ціною за зростанням/спаданням

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

When Користувач обирає фільтр «Name» і порядок «Ascending/Descending» і натискає кнопку «Submit»

Then – Портал відображає продукти за ім’ям у порядку зростання (від А до Я) / спадання (від Я до А)

When – Користувач обирає фільтр «Price» і порядок «Ascending/Descending» і натискає кнопку «Submit»

Then – Портал відображає продукти за ціною у порядку зростання (від найдешевших до найдорожчих) / спадання (від найдорожчих до найдешевших)

Рисунок 1.9 – “Products filter” user-story

Як користувач, я можу знайти продукти за частковим ім’ям

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку літеру в полі «Пошук»

When – Користувач натискає на кнопку «ОК»

Then – Портал відображає продукти, в яких міститься введена літера

Рисунок 1.10 – “View product by search query” user-story

Як користувач, я можу відкрити продукт і подивитись його деталі

Design:

https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=1-216&t=QTAaLzQ0w2MHdBci-4

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

When – Користувач відкриває продукт

Then – Портал відображає детальну інформацію про продукт: його ціну, характеристики, зображення продукту коментарі інших користувачів щодо цього продукту, і т.д.

Рисунок 1.11 – “View product details” user-story

Як користувач, я можу оцінити продукт і додати коментар до нього

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

And – Користувач відкриває продукт

And – Користувач ставить оцінку від 1 до 5 і додає якийсь коментар

When – Користувач натискає на кнопку «Опублікувати коментар»

Then – Портал відображає новостворений коментар і списку коментарів для обраного продукту

Рисунок 1.12 – “Comment product” user-story

Як користувач, я можу додати продукт у список бажаного

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

And – Користувач відкриває продукт

When – Користувач натискає на кнопку «Додати в список бажаного»

Then – Портал додає продукт у список бажаного і помічає продукт як бажаний

Рисунок 1.13 – “Add product to wishlist” user-story

Як користувач, я можу видалити продукт зі списку бажаного

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

And – Користувач відкриває продукт

And – Користувач натискає на кнопку «Додати в список бажаного»

When – Користувач ще раз натискає на кнопку «Додати в список бажаного»

Then – Портал видаляє продукт зі списку бажаного

Рисунок 1.14 – “Delete product from wishlist” user-story

Як користувач, я можу додати продукт у кошик

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

And – Користувач відкриває продукт

When – Користувач натискає на кнопку «Додати в кошик»

Then – Портал додає продукт у кошик і додає його ціну у загальну ціну кошику

Рисунок 1.15 – “Add product to cart” user-story

Як користувач, я можу продивитися, які продукти є в моєму кошику і яка його загальна ціна

Design:

https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=1-774&t=y3b2qmWXiFg45KO5-4

Given – Користувач заходить на сайт

When – Користувач натискає на кнопку «Кошик»

Then – Портал показує вікно кошика з усіма продуктами в ньому і правильно розрахованою ціною кошику

Рисунок 1.16 – “View cart” user-story

Як користувач, я можу змінити кількість одного товару в кошику

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

And – Користувач відкриває продукт

And – Користувач натискає на кнопку «Додати в кошик»

When – Користувач змінює кількість для замовлення якогось одного товару

Then – Портал правильно перераховує ціну одного товару, включаючи його кількість і загальну вартість кошика і виводить це користувачеві на клієнті.

Рисунок 1.17 – “Change cart item quantity” user-story

Як користувач, я можу видалити товар з кошику

Given – Користувач заходить на сайт

And – Користувач вводить будь-яку марку якоїсь лінійки товарів

And – Користувач натискає на кнопку «ОК»

And – Користувач відкриває продукт

And – Користувач натискає на кнопку «Додати в кошик»

When – Користувач видаляє товар з кошику

Then – Портал правильно перераховує вже зменшену загальну вартість кошика і виводить це користувачеві на клієнті.

Рисунок 1.18 – “Delete product from cart” user-story

Як користувач, я можу оформити замовлення згідно даних у кошику

Design:

https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=1-1405&t=y3b2qmWXiFg45KO5-4

Given – Користувач заходить на сайт

And – Користувач натискає на кнопку «Кошик»

When – Користувач натискає на кнопку «Оформити замовлення»

у відкритому вікні кошика

Then – Портал переходить до екрану замовлення де відображаються дані з кошика, персональні дані користувача, його кредитна картка, яка використовується для оплати замовлень

Рисунок 1.19 – “Create order” user-story

Як користувач, я можу оплатити замовлення

Given – Користувач заходить на сайт

And – Користувач натискає на кнопку «Кошик»

And – Користувач натискає на кнопку «Оформити замовлення» у відкритому вікні кошика

And – Користувач вводить дані своєї кредитної картки (або обирає вже існуючу)

When – Користувач натискає на кнопку «Оплатити замовлення» у відкритій сторінці замовлення

Then – Портал передає керування процесом платежу сервісу Stripe, який знімає точну і коректну кількість коштів з картки, яка була вказана у кошику; замовлення відправляється користувачеві; замовлення з'являється у особистому кабінеті користувача для його відстежування.

Рисунок 1.20 – "Pay for order" user-story

Як користувач, я можу відстежувати статус замовлень в особистому кабінеті

Design:

https://www.figma.com/file/G9CPuFwDVUSPHUq9NGbEQp/Diplom?type=design&node-id=1-1010&t=QTAaLzQ0w2MHdBci-4

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію і заходить у особистий кабінет

When – Користувач переходить на вкладку «Мої Замовлення»

Then – Портал показує усі замовлення користувача, замовленні товари, їх ціну і поточний статус

Рисунок 1.21 – "View order" user-story

Як адміністратор, я можу створювати/оновлювати продукт

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію як адміністратор і заходить у особистий кабінет

And – Користувач переходить на вкладку «Товари»

And – Користувач натискає на кнопку «Додати товар» / «Оновити товар»

When – Користувач вводить валідні дані для створення продукту (Ім'я, Ціна, Опис, Кількість на складі, Сабкатегорія, Категорія, Фото) і натискає на кнопку «ОК»

Then – Портал створює/оновлює запис продукту в базі і відображає його коректно на клієнті

Рисунок 1.22 – "Create product" user-story

Як адміністратор, я можу створювати/оновлювати категорію

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію як адміністратор і заходить у особистий кабінет

And – Користувач переходить на вкладку «Категорії»

And – Користувач натискає на кнопку «Додати категорію» / «Оновити категорію»

When – Користувач вводить валідні дані для створення/оновлення категорії (Ім'я,Фото) і натискає на кнопку «ОК»

Then – Портал створює/оновлює запис категорії в базі і відображає її коректно на клієнті

Рисунок 1.23 – "Create category" user-story

Як адміністратор, я можу створювати/оновлювати сабкатегорію

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію як адміністратор і заходить у особистий кабінет

And – Користувач переходить на вкладку «Сабкатегорії»

And – Користувач натискає на кнопку «Додати сабкатегорію» / «Оновити сабкатегорію»

When – Користувач вводить валідні дані для створення/оновлення категорії (Ім'я, Категорія, Фото) і натискає на кнопку «ОК»

Then – Портал створює/оновлює запис сабкатегорії в базі і відображає її коректно на клієнті

Рисунок 1.24 – "Create subcategory" user-story

Як адміністратор, я можу додавати знижку до товару

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію як адміністратор і заходить у особистий кабінет

And – Користувач переходить на вкладку «Товари»

And – Користувач натискає на кнопку «Додати знижку» на цікавлячому товарі

When – Користувач вводить валідні дані для створення/оновлення категорії (Ім'я, Процент знижки, Термін дії знижки) і натискає на кнопку «ОК»

Then – Портал додає знижку до товару і виводить коректно розраховану ціну зі знижкою

Рисунок 1.25 – "Create discount" user-story

Як адміністратор, я можу видаляти знижку з товару

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію як адміністратор і заходить у особистий кабінет

And – Користувач переходить на вкладку «Товари»

When – Користувач натискає на кнопку «Видалити знижку» на інтересуючому товарі

Then – Портал видаляє знижку з товару і виводить коректно розраховану ціну без знижки

Рисунок 1.26 – “Delete discount” user-story

Як системний адміністратор, я можу блокувати користувачів

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію як системний адміністратор і заходить у особистий кабінет

And – Системний адміністратор переходить на вкладку «Користувачі»

When – Системний адміністратор натискає на кнопку «Заблокувати» на потрібному йому користувачі

Then – Портал блокує користувача і той не зможе аутентифікуватися на цьому порталі, допоки системний адміністратор не розблокує його.

Рисунок 1.27 – “Ban user” user-story

Як системний адміністратор, я можу міняти ролі користувачів

Given – Користувач заходить на сайт

And – Користувач проходить аутентифікацію як системний адміністратор і заходить у особистий кабінет

And – Системний адміністратор переходить на вкладку «Користувачі»

And – Системний адміністратор натискає на кнопку «Змінити роль» на потрібному йому користувачі

When – Системний адміністратор обирає потрібну йому роль і натискає ОК

Then – Портал змінює роль користувача і видає йому відповідні до нової ролі дозволи.

Рисунок 1.28 – “Change user role” user-story

2 ПРОЄКТУВАННЯ ПЗ

2.1 База даних

Розділ "База даних" є важливим етапом в процесі розробки нашого E-commerce порталу для продажу будматеріалів [2]. У цьому розділі ми детально описуємо структуру та компоненти бази даних, які використовуються для зберігання, управління та обробки інформації нашого порталу.

Метою цього розділу є представлення чіткого та систематичного огляду бази даних, яка є фундаментальною складовою нашого E-commerce порталу. Ми розглянемо структуру таблиць, поля, взаємозв'язки між ними та розроблені операції для ефективного управління даними.

Опис бази даних включає в себе розкриття основних сутностей, які використовуються у нашому порталі, таких як товари, користувачі та замовлення. Також ми розглянемо різні види зв’язків, які будують нетворк між таблицями бази.

Цей розділ є критичним для розуміння структури та функціональності нашого E-commerce порталу. Ретельно спроєктована та оптимізована база даних забезпечить ефективну обробку даних, надійність операцій та зручне використання порталу як для наших адміністраторів, так і для наших користувачів.

2.1.1 ER-діаграма

У ході проєктування бази даних, була сконструйована така ER-діаграма (див. рис. 2.1) [3].

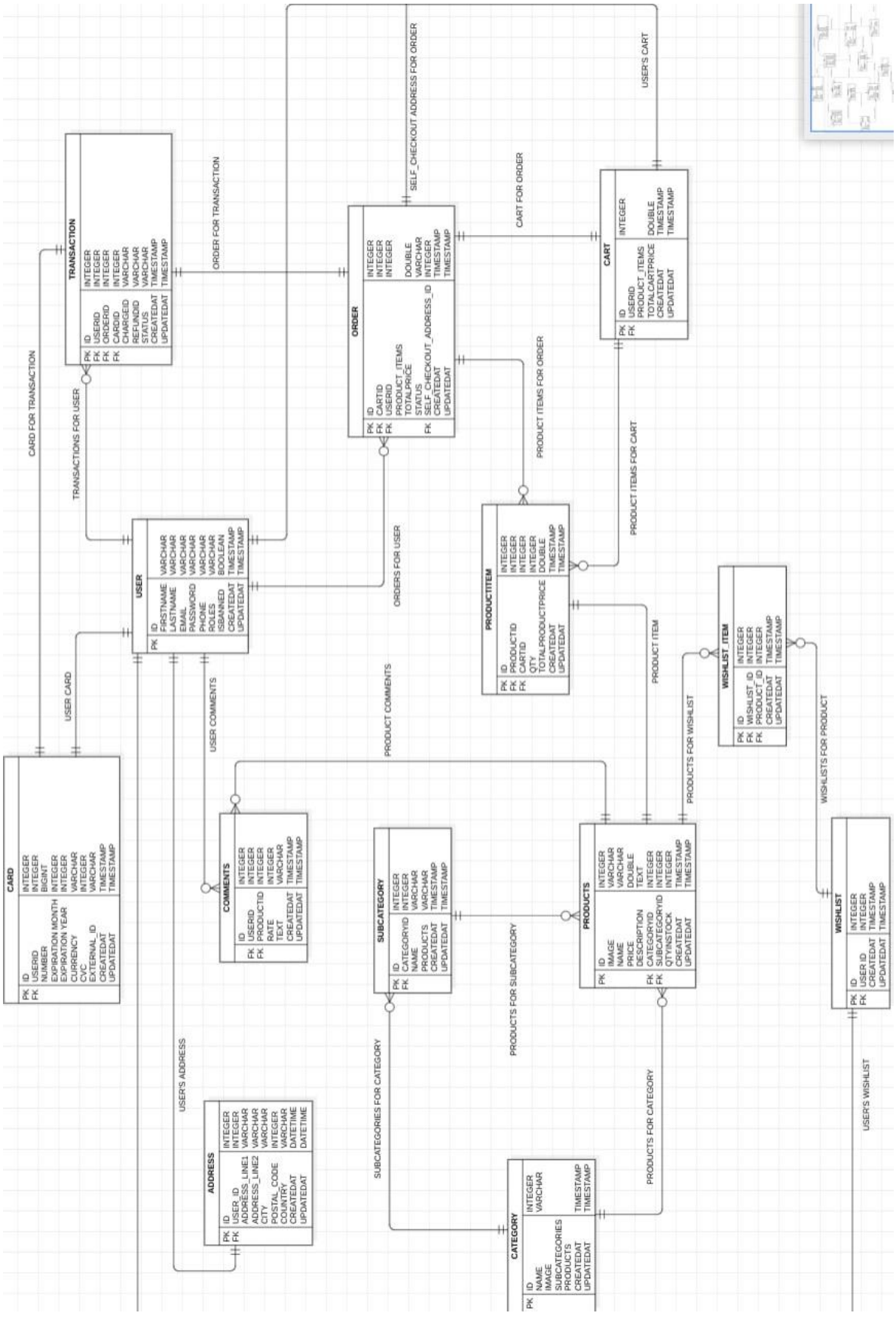

Рисунок 2.1 – ER-діаграма

2.1.2 Таблиці

У таблицях 2.1 – 2.15 і на рисунках 2.2 – 2.16 представлений фізичний опис таблиць бази даних маркетплейсу та їх вигляд в перекладі на мову SQL.

Таблиця 2.1 – Опис таблиці «User»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор користувача
FIRST_NAME	STRING	Ім'я користувача
LAST_NAME	STRING	Прізвище користувача
EMAIL	STIRNG	Емейл користувача
PASSWORD	STRING	Пароль користувача
PHONE	STRING	Телефон користувача
ROLES	ENUM	Роль користувача
IS_BANNED	BOOLEAN	Флаг для стану аккаунту користувача (заблокований/не заблокований)
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE "User" (
  id                SERIAL PRIMARY KEY,
  firstName         VARCHAR,
  lastName          VARCHAR,
  email             VARCHAR UNIQUE,
  password          VARCHAR,
  phone             VARCHAR,
  roles             TEXT[] DEFAULT ARRAY['CUSTOMER'],
  isBanned          BOOLEAN DEFAULT FALSE,
  createdAt         TIMESTAMPTZ DEFAULT NOW(),
  updatedAt         TIMESTAMPTZ
);
```

Рисунок 2.2 – SQL-код таблиці «User»

Таблиця 2.2 – Опис таблиці «Address»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор адреси користувача
USER_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "User"
ADDRESS_LINE_1	STRING	Перший рядок адреси
ADDRESS_LINE_2	STIRNG	Другий рядок адреси
POSTAL_CODE	INTEGER	Поштовий код адреси
CITY	STRING	Місто, в якому проживає користувач
COUNTRY	STRING	Країна, в якій проживає користувач
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Address" (
  "id" SERIAL PRIMARY KEY,
  "userId" INT UNIQUE,
  "address_line1" VARCHAR,
  "address_line2" VARCHAR,
  "city" VARCHAR,
  "postal_code" VARCHAR,
  "country" VARCHAR,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.3 – SQL-код таблиці «Address»

Таблиця 2.3 – Опис таблиці «Card»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор кредитної картки
USER_ID	INTEGER	Зовнішній ключ, що посилається на

Продовження табл. 2.3

ПОЛЕ	ТИП ДАНИХ	ОПИС
		таблицю "User"
NUMBER	STRING	Унікальний номер картки
EXPIRATION MONTH	INTEGER	Місяць закінчення терміну діїї картки
EXPIRATION YEAR	INTEGER	Рік закінчення терміну діїї картки
CURRENCY	STRING	Валюта, яку приймає картка
CVC	INTEGER	CVC картки
CARD_TOKEN	STRING	Унікальний віртуальний ідентифікатор Stripe для кредитної картки
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Card" (
  "id" SERIAL PRIMARY KEY,
  "userId" INT UNIQUE,
  "number" VARCHAR,
  "expMonth" INT,
  "expYear" INT,
  "cvc" INT,
  "currency" VARCHAR,
  "cardToken" VARCHAR,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.4 – SQL-код таблиці «Card»

Таблиця 2.4 – Опис таблиці «Category»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор категорії
NAME	STRING	Ім'я категорії
IMAGE	STRING	Іконка категорії

Продовження табл. 2.4

ПОЛЕ	ТИП ДАНИХ	ОПИС
SUBCATEGORIES	STIRNG []	Підкатегорії категорії
PRODUCTS	STIRNG []	Продукти категорії
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE "Category" (
  id                SERIAL PRIMARY KEY,
  name              VARCHAR UNIQUE,
  createdAt       TIMESTAMPTZ DEFAULT NOW(),
  updatedAt       TIMESTAMPTZ
);
```

Рисунок 2.5 – SQL-код таблиці «Category»

Таблиця 2.5 – Опис таблиці «Subcategory»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор підкатегорії
NAME	STRING	Ім'я підкатегорії
IMAGE	STRING	Іконка підкатегорії
CATEGORY_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Categories"
PRODUCTS	STRING []	Продукти категорії
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE "Subcategory" (
  id                SERIAL PRIMARY KEY,
  name              VARCHAR UNIQUE,
  subcategoryIconId  INT UNIQUE,
  categoryId        INT,
  createdAt         TIMESTAMPTZ DEFAULT NOW(),
  updatedAt         TIMESTAMPTZ,
  FOREIGN KEY ("subcategoryIconId")
  REFERENCES   "SubcategoryIcon (id),
  FOREIGN KEY ("categoryId")
  REFERENCES "Category" (id) ON DELETE CASCADE
);
```

Рисунок 2.6 – SQL-код таблиці «Subcategory»

Таблиця 2.6 – Опис таблиці «Product»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор продукту
NAME	STRING	Ім'я продукту
PRICE	DOUBLE	Ціна одного продукту
DESCRIPTION	TEXT	Детальний опис продукту
IMAGE	STRING	Картинка продукту
CATEGORY_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Categories"
SUBCATEGORY_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Subcategories"
QTY_IN_STOCK	INTEGER	Кількість товару на складі
DISCOUNT_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Discount"
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```sql
CREATE TABLE IF NOT EXISTS "Product" (
  "id" SERIAL PRIMARY KEY,
  "name" VARCHAR UNIQUE,
  "price" FLOAT,
  "discountPrice" FLOAT,
  "productImageId" INT UNIQUE,
  "description" VARCHAR,
  "subcategoryId" INT,
  "categoryId" INT,
  "qtyInStock" INT,
  "discountId" INT UNIQUE,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.7 – SQL-код таблиці «Product»

Таблиця 2.7 – Опис таблиці «Comment»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор коментарію
USER_ID	STRING	Зовнішній ключ, що посилається на таблицю "User"
PRODUCT_ID	DOUBLE	Зовнішній ключ, що посилається на таблицю "Product"
RATE	ENUM	Оцінка товару від 1 до 5
TEXT	TEXT	Безпосередньо сам коментарій
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Comment" (
  "id" SERIAL PRIMARY KEY,
  "userId" INT,
  "productId" INT,
  "text" VARCHAR NOT NULL,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.8 – SQL-код таблиці «Comment»

Таблиця 2.8 – Опис таблиці «CartItem»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор предмету кошику
PRODUCT_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Product"
PRODUCT_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Cart"

Продовження табл. 2.8

ПОЛЕ	ТИП ДАНИХ	ОПИС
QUANTITY	INTEGER	Бажана кількість для замовлення товару
SUB_TOTAL_PRICE	DOUBLE	Загальна ціна предмету кошику, враховуючи кількість і ціну за одиницю товару
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "CartItem" (
  "id" SERIAL PRIMARY KEY,
  "cartId" INT,
  "productId" INT,
  "quantity" INT,
  "subTotalPrice" FLOAT DEFAULT 0.00,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.9 – SQL-код таблиці «CartItem»

Таблиця 2.9 – Опис таблиці «Cart»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор кошику
USER_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Product"
PRODUCT_ITEMS	PRODUCT_ITEM[]	Масив предметів для кошику
TOTAL_CART_PRICE	DOUBLE	Загальна вартість кошику
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Cart" (
  "id" SERIAL PRIMARY KEY,
  "userId" INT UNIQUE,
  "totalPrice" FLOAT DEFAULT 0.00,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.10 – SQL-код таблиці «Cart»

Таблиця 2.10 – Опис таблиці «WishlistItem»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор предмету списку бажаного
PRODUCT_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Product"
WISHLIST_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Wishlist"
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "WishlistItem" (
  "id" SERIAL PRIMARY KEY,
  "wishlistId" INT,
  "productId" INT,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.11 – SQL-код таблиці «WishlistItem»

Таблиця 2.11 – Опис таблиці «Wishlist»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор предмету списку бажаного
USER_ID	INTEGER	Зовнішній ключ, що

Продовження табл. 2.11

ПОЛЕ	ТИП ДАНИХ	ОПИС
		посилається на таблицю "User"
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Wishlist" (
  "id" SERIAL PRIMARY KEY,
  "userId" INT UNIQUE,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.12 – SQL-код таблиці «Wishlist»

Таблиця 2.12 – Опис таблиці «Order»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор предмету замовлення
USER_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "User"
CART_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Cart"
TOTAL_PRICE	DOUBLE	Вартість кошику, який підтягується через поле «CART_ID»
STATUS	ENUM	Статус замовлення
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Order" (
  "id" SERIAL PRIMARY KEY,
  "userId" INT,
  "cartId" INT UNIQUE,
  "amount" FLOAT,
  "orderStatus" VARCHAR,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.13 – SQL-код таблиці «Order»

Таблиця 2.13 – Опис таблиці «Transaction»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор транзакції
USER_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "User"
ORDER_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Order"
CARD_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Card"
CHARGE_STATUS	ENUM	Статус транзакції
CHARGE_TOKEN	STRING	Унікальний ідентифікатор Stripe для виконаного платежу
REFUND_TOKEN	STRING	Унікальний ідентифікатор Stripe для виконаного рефанду
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Transaction" (
  "id" SERIAL PRIMARY KEY,
  "userId" INT,
  "orderId" INT UNIQUE,
  "cardId" INT,
  "chargeStatus" VARCHAR,
  "chargeToken" VARCHAR,
  "refundToken" VARCHAR,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.14 – SQL-код таблиці «Transaction»

Таблиця 2.14 – Опис таблиці «SelfCheckoutCity»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор міста, в якому зберігаються відділення для самовивізу
CITY	STRING	Назва міста, в якому зберігаються відділення для самовивізу
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "Self_checkout_city" (
  "id" SERIAL PRIMARY KEY,
  "city" VARCHAR,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.15 – SQL-код таблиці «SelfCheckoutCity»

Таблиця 2.15 – Опис таблиці «Self_checkout_address»

ПОЛЕ	ТИП ДАНИХ	ОПИС
ID	INTEGER	Унікальний ідентифікатор міста, в якому зберігаються відділення для самовивізу
SELF_CHECKOUT_ADDRESS_ID	INTEGER	Зовнішній ключ, що посилається на таблицю "Self_checkout_city"
SECTION_NUMBER	INTEGER	Номер відділення для

Продовження табл. 2.15

ПОЛЕ	ТИП ДАНИХ	ОПИС
		самовивізу
SECTION_ADDRESS	STRING	Адрес відділення для самовивізу
CREATED_AT	DATETIME	Дата, коли запис був створений
UPDATED_AT	DATETIME	Дата, коли запис був оновлений

```
CREATE TABLE IF NOT EXISTS "SelfCheckoutSection" (
  "id" SERIAL PRIMARY KEY,
  "selfCheckoutId" INT,
  "sectionNumber" INT,
  "sectionAddress" INT,
  "createdAt" TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  "updatedAt" TIMESTAMP
);
```

Рисунок 2.16 – SQL-код таблиці «Self_checkout_address»

2.2 Діаграма класів

Розділ "Діаграма класів" є важливим етапом в процесі розробки нашого E-commerce порталу для продажу будматеріалів (див. рис. 2.17). У цьому розділі ми представимо детальний огляд класів, їх взаємозв'язків та структури системи, що допоможе нам краще розуміти логіку і функціональні можливості нашого порталу.

Метою цього розділу є надання глибокого інсайту в архітектуру нашої системи та сприяння зручному розумінню взаємозв'язків між класами. Діаграма класів дозволяє нам візуалізувати структуру програмного забезпечення, його компоненти та взаємодію між ними.

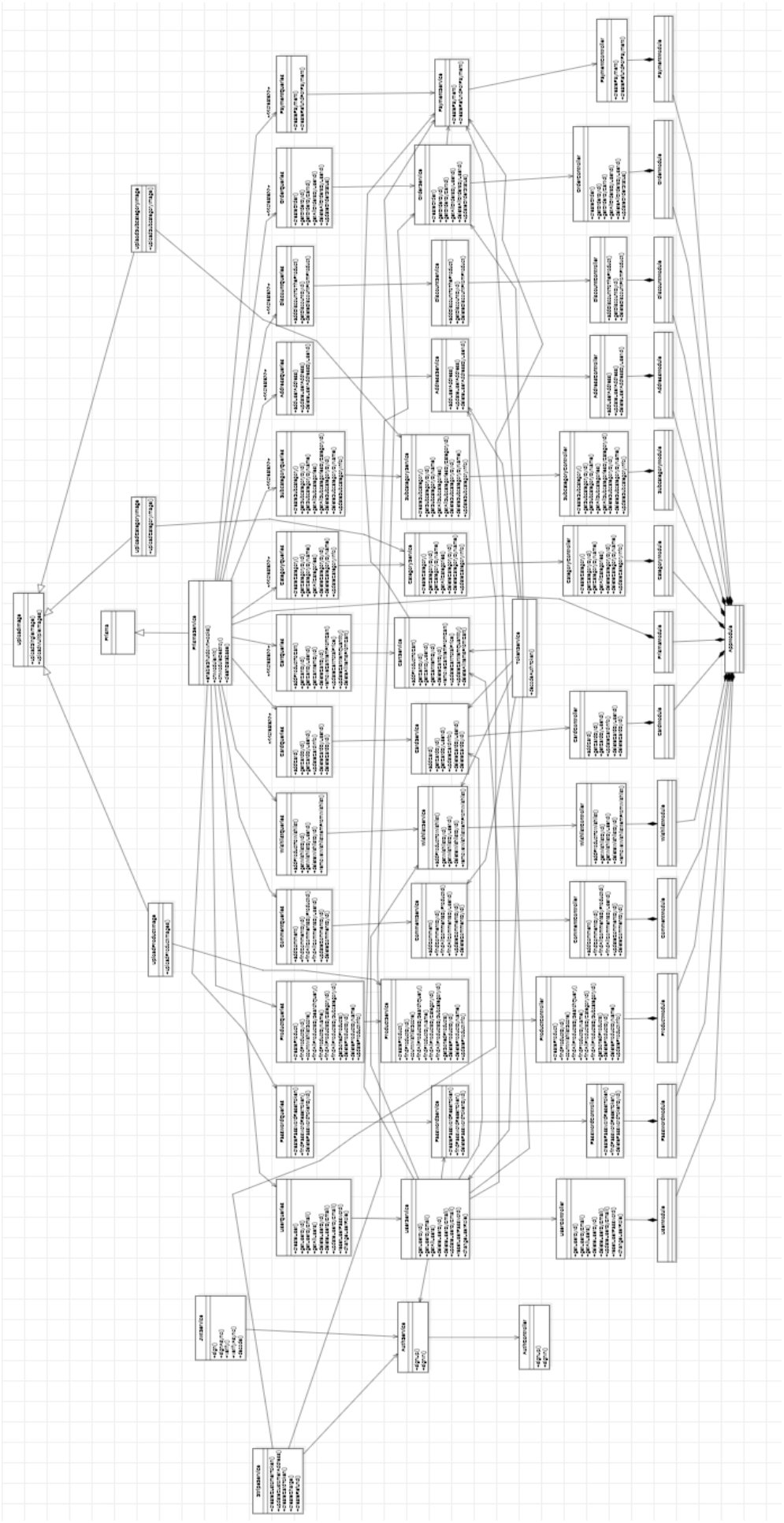

Рисунок 2.17 – Діаграма класів

У цьому розділі ми розглянемо основні класи, які використовуються у нашому E-commerce порталі, такі як "Користувач", "Товар", "Кошик", "Замовлення" та інші. Ми проаналізуємо їх атрибути, методи та взаємозв'язки, які сприяють реалізації функціональності порталу.

Діаграма класів є потужним інструментом для моделювання складних систем та відображення взаємозв'язків між класами. Цей розділ є важливим кроком для нашого проєкту, оскільки допоможе нам краще організувати код, розподілити функціональні обов'язки та забезпечити масштабованість та розширюваність нашого порталу.

Наведена діаграма класів (див. рис. 2.17) допоможе розробникам та членам команди легше спілкуватися, розуміти функціональність системи та здійснювати необхідні зміни та модифікації. Це важлива складова проєкту, яка сприятиме успішній розробці та впровадженню нашого E-commerce порталу.

2.3 Опис технологій реалізації

React – це JavaScript бібліотека для розробки користувацьких інтерфейсів [4]. Він дозволяє створювати ефективні, інтерактивні та масштабовані веб-додатки. Основною перевагою React є його компонентний підхід до розробки, що дозволяє розділити інтерфейс на незалежні, повторно використовувані компоненти.

Однією з основних переваг React є віртуальне DOM (Document Object Model), яке дозволяє ефективно оновлювати тільки необхідні частини сторінки, забезпечуючи швидку реакцію на зміни. Це дозволяє покращити продуктивність веб-додатків та забезпечити плавну взаємодію з користувачем.

Крім того, React має велику спільноту розробників та екосистему інструментів, які допомагають розробникам швидко розгортати та

підтримувати веб-додатки. Такі інструменти, як Redux, React Router і Axios, роблять розробку складних функціональних можливостей більш простою та ефективною.

У контексті розробки маркетплейсу, React надає багато переваг. Він дозволяє створювати переважно односторінкові додатки, що реагують миттєво на дії користувачів. Компонентна структура React дозволяє створювати повторно використовувані блоки, такі як товарні карточки, кошики або фільтри, що спрощує розробку та підтримку коду.

Крім того, React має простий та зрозумілий синтаксис, що сприяє швидкому навчанню та ефективній командній роботі. Він також інтегрується з багатьма іншими бібліотеками та фреймворками, що дозволяє легко розширювати функціональність вашого маркетплейсу.

Узагалі, React є потужним інструментом для розробки веб-додатків, особливо тих, які потребують швидкого реагування на зміни, великої кількості взаємодій з користувачем та масштабованості. Використання React у розробці маркетплейсу допоможе створити сучасний та ефективний інтерфейс, який забезпечить зручну та задоволену користувачів платформу для продажу будматеріалів.

Node.js – це середовище виконання JavaScript, яке дозволяє розробникам використовувати JavaScript на серверній стороні [5]. Однією з ключових переваг Node.js є його асинхронна та подієво-орієнтована модель програмування, що дозволяє ефективно обробляти багато запитів одночасно без блокування інших операцій.

Одна з найважливіших переваг Node.js – це швидкість. Його легкий та ефективний двигун V8 забезпечує швидке виконання коду. Крім того, Node.js має велику кількість модулів та пакетів, які полегшують розробку і розширення функціональності веб-додатків.

Node.js також є дуже масштабованим. Він дозволяє легко створювати високонавантажені додатки, що здатні обробляти багато запитів одночасно. Це особливо важливо для маркетплейсу, оскільки він має обробляти

одночасно багато запитів від користувачів, які шукають, додають або купують товари.

Крім того, Node.js має велику активну спільноту розробників, що забезпечує підтримку, документацію та постійні оновлення. Його екосистема включає такі інструменти, як Express.js, що спрощують створення серверних додатків та розробку API.

Використання Node.js у розробці маркетплейсу дозволяє створити швидкий, масштабований та ефективний сервер для вашого додатку. Ви зможете здійснювати обробку запитів, управляти базою даних, забезпечувати безпеку та взаємодію з іншими сервісами для створення повнофункціонального маркетплейсу.

NestJS є фреймворком для розробки серверної частини веб-додатків на базі Node.js [6]. Він базується на принципах модульності, компонентності та розширюваності. NestJS поєднує в собі потужність Node.js з патернами проєктування, відомими з фреймворків, таких як Angular.

Однією з головних переваг NestJS є його структурований підхід до розробки. Він надає модульну архітектуру, яка дозволяє організувати код у логічні блоки, такі як контролери, провайдери, сервіси і моделі даних. Це полегшує розуміння та підтримку кодової бази.

Ще однією перевагою NestJS є його вбудована підтримка TypeScript. TypeScript дозволяє писати код з типізацією, що полегшує виявлення та усунення помилок на етапі розробки. NestJS пропонує зручну інтеграцію з TypeScript, що дозволяє розробникам використовувати потужні можливості цієї мови програмування.

Також варто зазначити, що NestJS має вбудовану підтримку для веб-сокетів, мікросервісної архітектури та аутентифікації. Він надає розширені можливості для роботи з базами даних та підключення зовнішніх сервісів. Крім того, NestJS пропонує велику кількість допоміжних модулів та інструментів, що полегшують розробку та підтримку серверної частини.

У порівнянні з іншими Node.js фреймворками, NestJS вирізняється

своєю структурованістю, підтримкою TypeScript і великим екосистемою модулів. Це робить його потужним інструментом для розробки серверної частини веб-додатків, зокрема для маркетплейсу з продажу будматеріалів.

Prisma – це сучасний ORM (Object-Relational Mapping), який надає швидкий та зручний спосіб взаємодії з базою даних в додатках [7]. Вона підтримує багато різних баз даних, таких як PostgreSQL, MySQL, SQLite і деякі інші.

Однією з ключових переваг Prisma є його типізація. Він генерує типи TypeScript на основі схеми бази даних, що дозволяє вам отримати переваги статичної типізації та автодоповнення в редакторі коду. Це забезпечує високу робочу ефективність та допомагає уникнути помилок на ранніх етапах розробки.

Prisma також пропонує зручні механізми для виконання різних операцій з базою даних, таких як створення, читання, оновлення та видалення записів. Вона надає декларативний підхід до роботи з даними, що полегшує їхню маніпуляцію та забезпечує чистоту коду.

Іншою вагомою перевагою Prisma є його швидкодія. Вона використовує розумне кешування запитів та оптимізацію доступу до бази даних, що дозволяє підвищити продуктивність додатка. Крім того, Prisma автоматично генерує ефективні SQL-запити, що сприяє оптимальній роботі з базою даних.

Окрім цього, Prisma надає зручні механізми міграцій, що дозволяють зручно оновлювати структуру бази даних та управляти версіями схеми.

Загалом, Prisma є потужним інструментом для роботи з базами даних у розробці додатків. Вона пропонує переваги типізації, зручність взаємодії з базою даних, швидкодію та інші функціональні можливості, що роблять її привабливим вибором при розробці маркетплейсу з продажу будматеріалів.

PostgreSQL є потужною об'єктно-реляційною системою керування базами даних (СКБД), яка пропонує розширені можливості для зберігання і обробки даних [8]. Ось деякі з переваг PostgreSQL, які зробили його

популярним вибором для розробки маркетплейсів.

Надійність і стабільність: PostgreSQL відомий своєю стабільністю та надійністю. Він використовує транзакційну систему з підтримкою ACID (атомарність, консистентність, ізольованість, довіреність), що гарантує цілісність даних та захист від втрати інформації.

Розширюваність: PostgreSQL надає широкі можливості для розширення. Він підтримує реплікацію, шардування, кластеризацію та інші технології масштабування, що дозволяють забезпечити високу продуктивність та доступність для великих обсягів даних.

Розширені типи даних: PostgreSQL має багатий набір вбудованих типів даних і дозволяє створювати користувацькі типи, що робить його відмінним вибором для обробки різноманітної інформації. Він підтримує географічні дані, JSON, XML, даних у форматі зображень та багато інших типів.

Потужна мовний підтримка: PostgreSQL підтримує багато мов програмування, включаючи SQL, PL/pgSQL, JavaScript, Python, Ruby та багато інших. Це дає розробникам більшу свободу вибору мови для взаємодії з базою даних.

Розширені можливості запитів: PostgreSQL надає потужні інструменти для складних запитів і аналітики даних. Він підтримує ряд агрегатних функцій, віконні функції, CTE (загальні табличні вирази) та інші функціональні можливості, що дозволяють здійснювати розширену обробку та аналіз даних.

У контексті розробки маркетплейсу з продажу будматеріалів, PostgreSQL є привабливим вибором, оскільки він забезпечує надійну збереження та обробку даних, підтримує масштабування для росту великими обсягами інформації, має розширені можливості для роботи з різними типами даних та надає потужні засоби для аналізу даних.

Docker – це відкрите програмне забезпечення, яке надає можливість контейнеризації додатків [9]. Воно дозволяє упаковувати програми та їх

залежності в стандартизовані контейнери, які можна легко переносити та запускати на різних середовищах.

Ось деякі переваги Docker, які роблять його популярним вибором для розробки маркетплейсів.

Відокремленість і ізоляція: Docker контейнери забезпечують відокремленість додатків та їх залежностей, що дозволяє запускати їх на одній системі, не впливаючи один на одного. Це дозволяє уникнути конфліктів між різними компонентами вашого маркетплейсу та забезпечити більшу стабільність і надійність системи.

Портативність: Docker контейнери є портативними, тобто ви можете створювати їх на одній системі та запускати на інших безпосередньо, нехай навіть ці системи мають різні конфігурації або операційні системи. Це забезпечує зручність в розгортанні вашого маркетплейсу на різних середовищах, включаючи розробку, тестування та продакшн.

Швидкість розгортання: Docker надає швидкий процес розгортання, оскільки контейнери можна створювати, запускати та зупиняти швидше, ніж традиційні віртуальні машини. Ви можете масштабувати свої маркетплейси залежно від потреб, додаючи або видаляючи контейнери залежно від навантаження.

Системна ефективність: Docker використовує ресурси системи ефективно, оскільки він ділить ядро операційної системи з хост-системою, уникнуваючи додаткових накладних витрат. Це дозволяє оптимізувати використання обчислювальних ресурсів та забезпечити більшу продуктивність вашого маркетплейсу.

Екосистема та підтримка: Docker має широку спільноту розробників та багато готових образів контейнерів, що полегшує розгортання та управління вашим маркетплейсом. Крім того, Docker інтегрується з різними інструментами та сервісами, такими як Kubernetes, що дозволяє легко масштабувати та управляти вашими контейнерами.

У розробці маркетплейсу Docker забезпечує простоту в розгортанні,

управлінні та масштабуванні додатків, робить процес розробки та тестування більш ефективним і дозволяє забезпечити стабільність та портативність вашого маркетплейсу на різних середовищах.

3 РЕАЛІЗАЦІЯ ПЗ

Користувач вводить валідні дані в усі необхідні поля: FirstName, LastName, Email, Password, і натискає на кнопку “Sign Up”. Якщо користувач з таким емейлом вже існує в базі, з серверу прийде 400 помилка, яка інформує, що такий емейл вже зареєстрований. Якщо такого емейлу немає, і всі дані валідні, то в базі створиться новий запис з інформацією про користувача. Пароль буде зашифрований бібліотекою bcrypt для більшої секьюрності, і в кінцевому JSON об’єкті він не буде повертатися. Також створюється customerToken, який потрібен для проведення платіжок через Страйп (див. рис. 3.1 – 3.3).

```
public async signUp(user: UserDto) : Promise<User> {
  const hashedPassword : string = await bcrypt.hash(user.password, saltOrRounds: 10);
  try {
    const createdUser : User = await this.userService.signUp( data: {
      ...user,
      password: hashedPassword,
    });
    createdUser.password = undefined;
    const userToken : Stripe.Response<Stripe.Custome... = await this.stripeService.createCustomerToken(
      createdUser.firstName,
      createdUser.lastName,
      createdUser.email,
    );
    await this.prismaService.user.update( args: {
      where: { id: createdUser.id },
      data: {
        customerToken: userToken.id,
      },
    });
    return createdUser;
  } catch (e) {
    if (e instanceof Prisma.PrismaClientKnownRequestError) {
      if (e.code === 'P2002') {
        throw new BadRequestException( objectOrError: `User with such email already exists`);
      }
    }
    throw e;
```

Рисунок 3.1 – Код для виконання процесу реєстрації

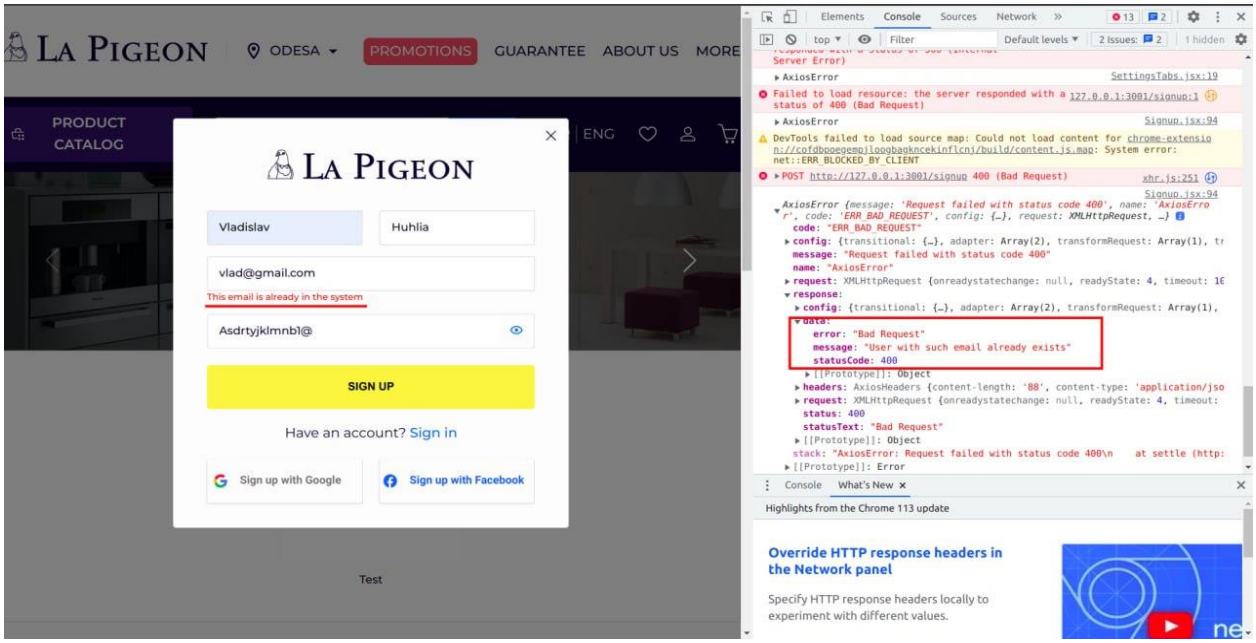

Рисунок 3.2 – Помилка при спробі реєстрації з емейлом, який вже існує

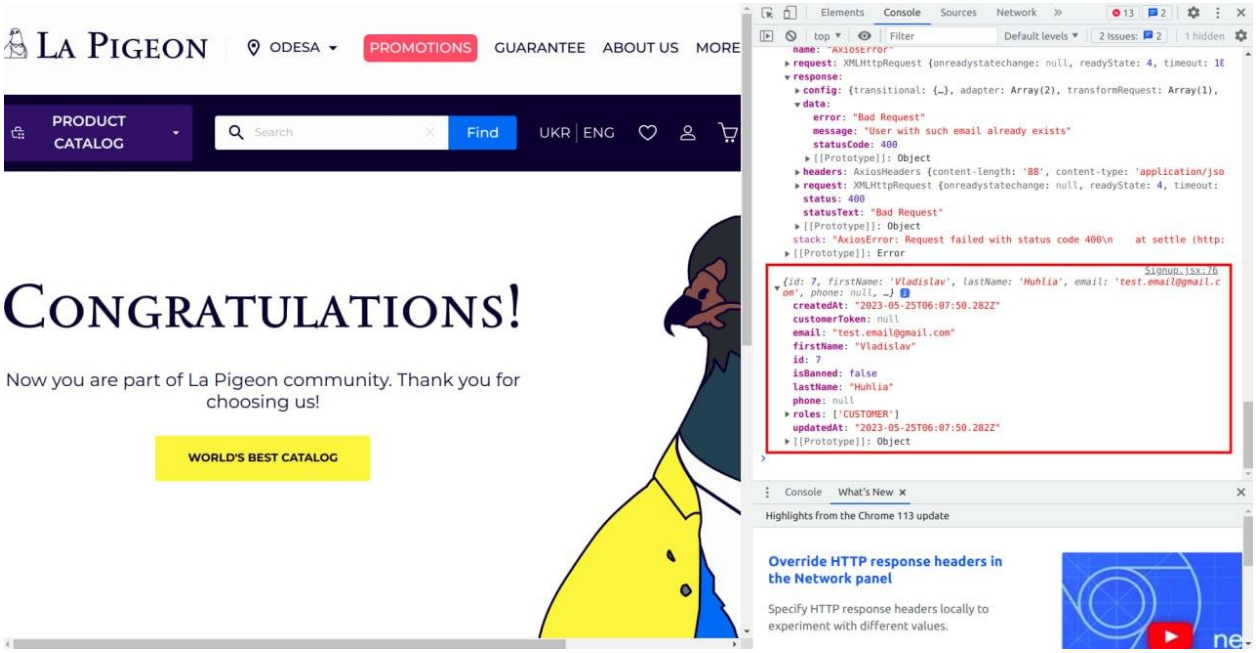

Рисунок 3.3 – Успішна реєстрація користувача

Користувач вводить емейл і пароль у валідному форматі у формі «Sign In». Якщо введеного емейла не існує в базі, сервер поверне 404 помилку з месседжом «User doesn’t exists».Якщо емейл корректний, але пароль неправильний, сервер поверне 401 помилку з месседжом «Invalid user credentials». Якщо емейл корректний і пароль до нього підходить, сервер підписує JWT токен, куди записується емейл користувача, його роль та

isBanned поле, яке показує, забанений юзер чи ні. Потім цей токен він повертає у об’єкті, де на клієнті він сетиться у browserStorage, і відтепер, усі реквести, зроблені з клієнту, будуть автоматично мати цей токен у headers, як показник, що юзер залогінений. На клієнті юзер одразу переходить на сторінку свого персонального аккаунту (див. рис. 3.4 – 3.7).

```
async signIn(data: LoginDto) : Promise<{access_token: string}... {
  const user : User & {comments: Comment[], c... = await this.userService.findUserByEmail(data.email);

  if (!user || !(await bcrypt.compare(data.password, user.password))) {
    throw new UnauthorizedException( objectOrError: 'Invalid user credentials');
  }

  if (user.isBanned === true) {
    throw new ForbiddenException(
      objectOrError: 'You are banned from this portal. Please, refer to administrator',
    );
  }
  const access_token : string = this.jwtService.sign( payload: {
    id: user.id,
    email: user.email,
    roles: user.roles,
    isBanned: user.isBanned,
  });
  return { access_token };
}
```

Рисунок 3.4 – Код для виконання процесу аутентифікації користувача

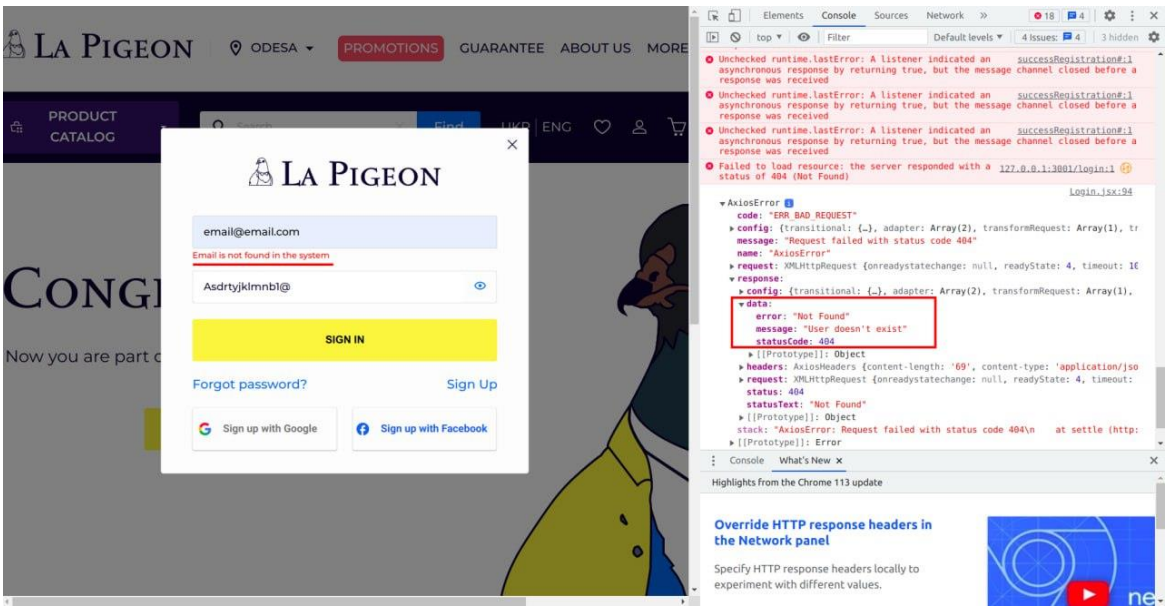

Рисунок 3.5 – Помилка при спробі логіну з неіснуючим в базі емейлом

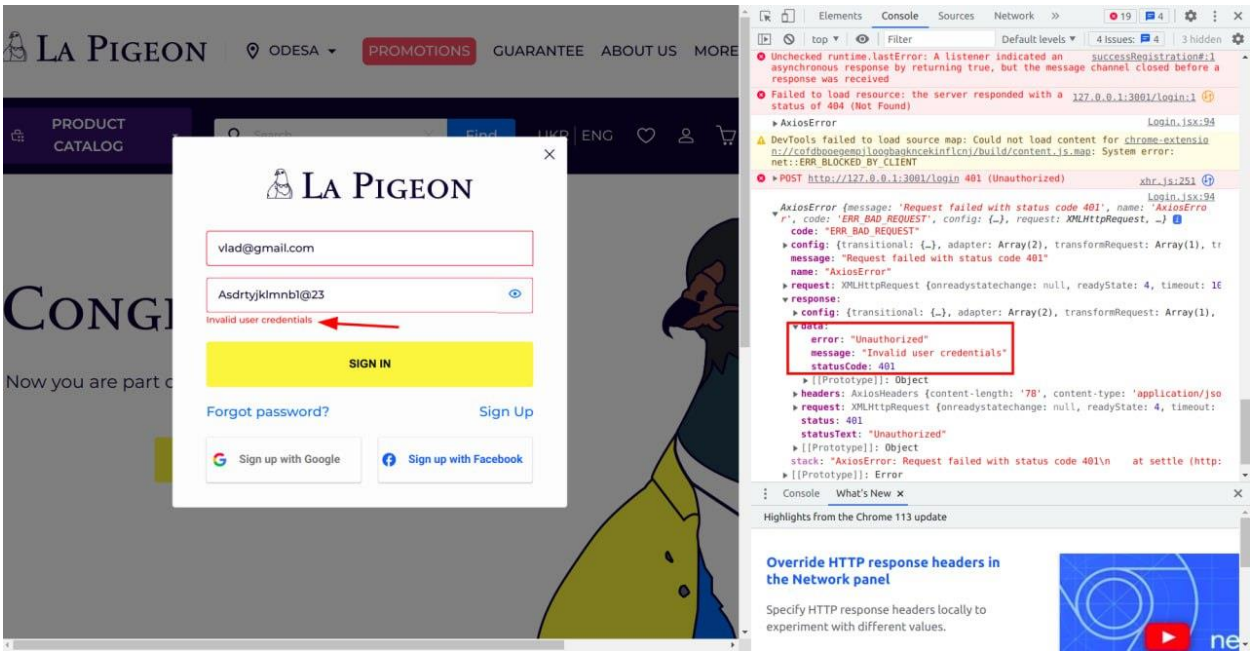

Рисунок 3.6 – Помилка при спробі логіну з неправильним паролем

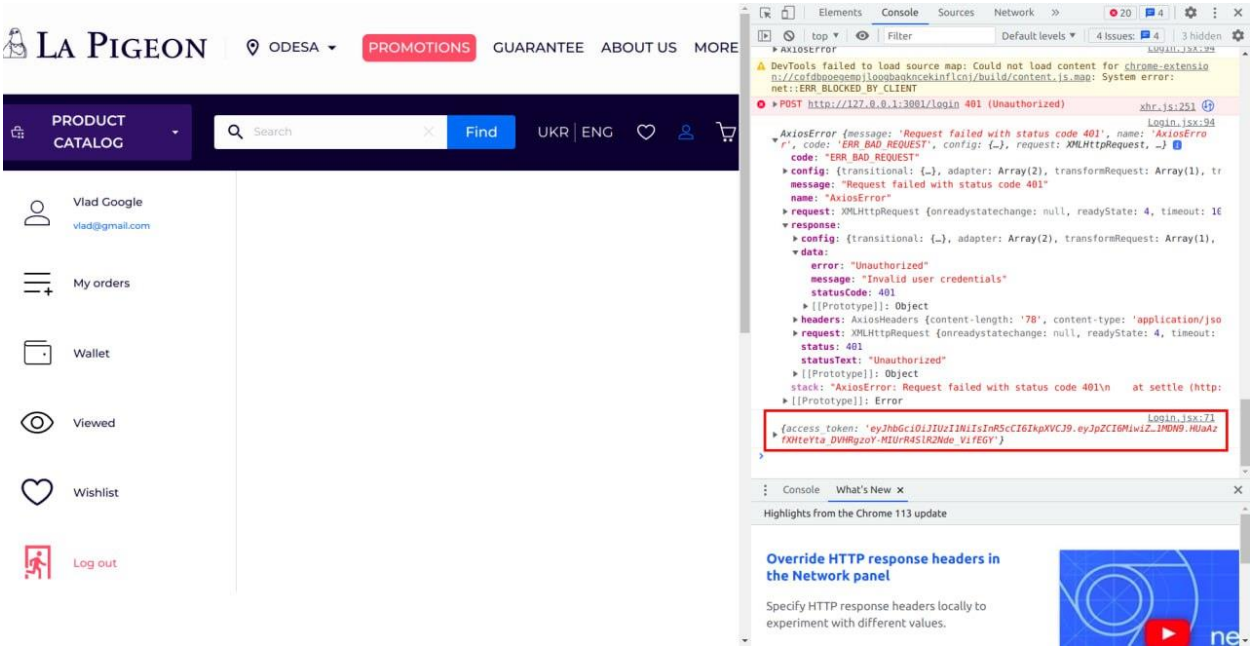

Рисунок 3.7 – Успішна аутентифікація користувача

Користувач вводить валідні дані у всі необхідні поля: номер картки, рік закінчення терміну дії картки, місяць закінчення терміну дії картки, її CVC і валюта картки (див. рис. 3.8 та 3.9). Після відправки реквесту на сервер, створюється унікальний токен картки, який потрібен потім для проведення платіжок і створюється запис в базі, який прив’язується до поточного

користувача, в аккаунті якого була створена картка.

```
1 usage   VladGoogle
async addCard(data: CardDto, authHeader: string): Promise<Card> {
  const decodedPayload : PayloadInterface  = await this.tokenService.decodeAuthToken(authHeader);
  const user : User & {comments: Comment[], c...  = await this.userService.findUserByEmail(decodedPayload.email);
  await this.cardQueries.addCard( data: {
    ...data,
    userId: user.id,
  });

  const cardSource : Stripe.Response<Stripe.Custome...  = await this.stripeService.createCardToken(
    user.customerToken,
    cardToken: 'tok_mastercard',
  );

  return this.prisma.card.update( args: {
    where: { userId: user.id },
    data: {
      cardToken: 'tok_mastercard',
      cardSource: cardSource.id,
    },
    include: {
      payments: true,
    },
  });
}
```

Рисунок 3.8 – Код для виконання процесу додавання крединтої картки

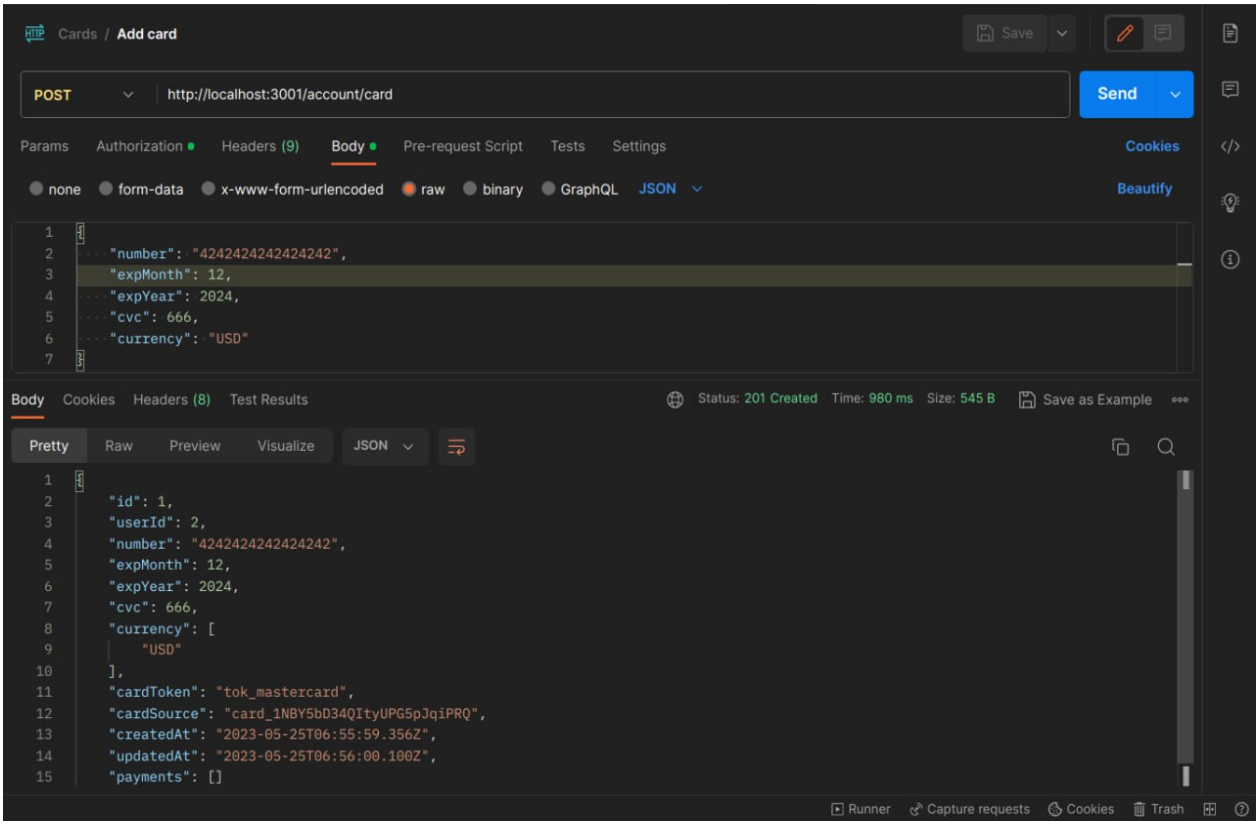

Рисунок 3.9 – Успішне додавання кредитної кратки користувача

Користувач з правами адміністратора або системного адміністратора може створити категорію, ввівши її ім’я і завантаживши іконку для категорії (див. рис. 3.10, рис. 3.11).

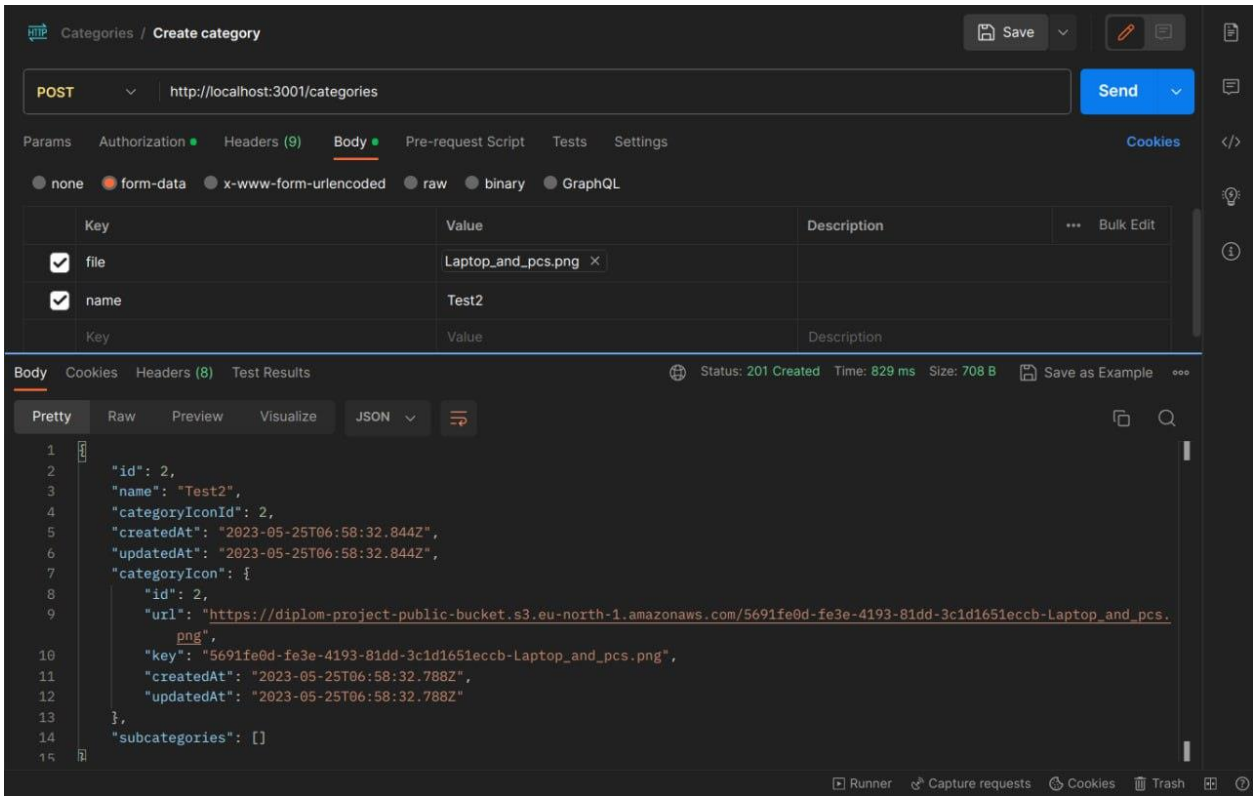

Рисунок 3.10 – Успішне створення категорії

```
1 usage   VladGoogle
async createCategory(
  data: CategoryDto,
  buffer: Buffer,
  filename: string,
): Promise<Category> {
  const icon : CategoryIcon  = await this.categoryIconService.uploadCategoryIcon(
    buffer,
    filename,
  );
  return await this.categoryQueries.createCategory( data: {
    ...data,
    categoryIconId: icon.id,
  });
}
```

Рисунок 3.11 – Код для виконання процесу створення категорії

Після відправлення реквесту, іконка відсилається на опрацювання AWS сервісу під назвою S3. Якщо з даними все гаразд, то фотографія завантажиться на віддалений сервер, і поверне об’єкт, в якому є двоє важливих для категорії, полів – url, який містить посилання на завантажений файл (тобто, на фотографію категорії), і по якому його можна завантажити це файл, і key – унікальний ідентифікатор файлу. Після цього створюється запис категорії в базі, куди записується ім’я категорії і ідентифікатор новоствореного запису іконки категорії.

Користувач з правами адміністратора або системного адміністратора може створити сабкатегорію, ввівши її ім’я, ідентифікатор категорії, до якої буде належати сабкатегорія, і завантаживши іконку для сабкатегорії (див. рис. 3.12, рис. 3.13) Після відправлення реквесту, іконка відсилається на опрацювання AWS сервісу під назвою S3. Якщо з даними все гаразд, то фотографія завантажиться на віддалений сервер, і поверне об’єкт, в якому є двоє важливих для категорії, полів – url, який містить посилання на завантажений файл (тобто, на фотографію категорії), і по якому його можна завантажити це файл, і key – унікальний ідентифікатор файлу. Після цього створюється запис сабкатегорії в базі, куди записується ім’я сабкатегорії, ідентифікатор категорії і ідентифікатор новоствореного запису іконки категорії.

```
1 usage   VladGoogle
async createSubcategory(data: SubcategoryDto) :Promise<Subcategory & {categor...  {
  try {
    return await this.prisma.subcategory.create({
      data: {
        ...data,
      },
      include: {
        subcategoryIcon: true,
        category: {
          include: {
            categoryIcon: true,
          },
        },
      },
    });
  } catch (e) {
    if (e instanceof Prisma.PrismaClientKnownRequestError) {
      if (e.code === 'P2002') {
        throw new BadRequestException(
          objectOrError: `Subcategory with such name already exists`,
        );
      }
    }
    throw e;
  }
}
```

Рисунок 3.12 – Код для процесу створення сабкатегорії

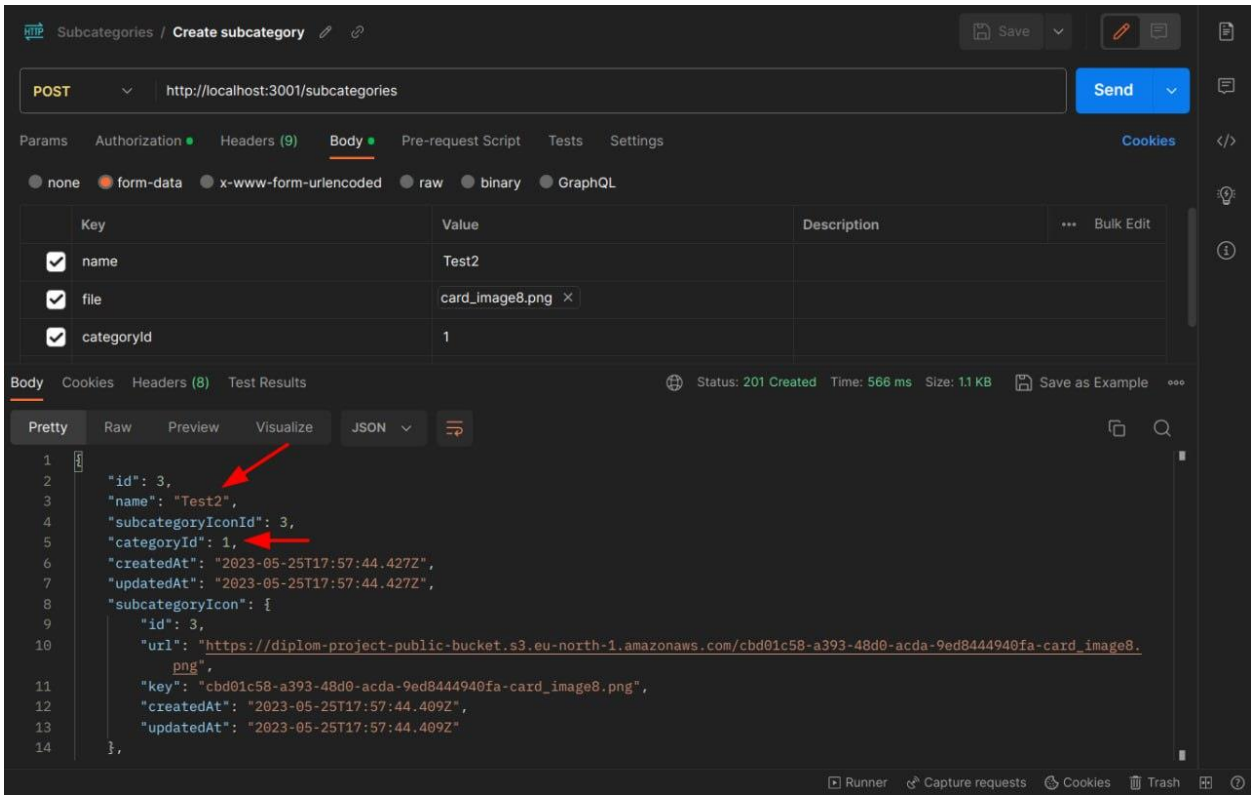

Рисунок 3.13 – Успішне створення сабкатегорії

Користувач з правами адміністратора або системного адміністратора може створити продукт, ввівши всі валідні дані (див. рис. 3.14, рис. 3.15). Користувач може додати до продукту стільки фотографій, скільки він захоче.

```
1 usage  VladGoogle
async createProduct(data: ProductDto) : Promise<Product & {subcategory... {
  try {
    return await this.prisma.product.create({
      data: {
        ...data,
      },
      include: {
        productImages: true,
        category: true,
        subcategory: true,
        discount: true,
      },
    });
  } catch (e) {
    if (e instanceof Prisma.PrismaClientKnownRequestError) {
      if (e.code === 'P2002') {
        throw new BadRequestException(
          objectOrError: `Product with such name already exists`,
        );
      }
    }
    throw e;
  }
}
```

Рисунок 3.14 – Код створення продукту

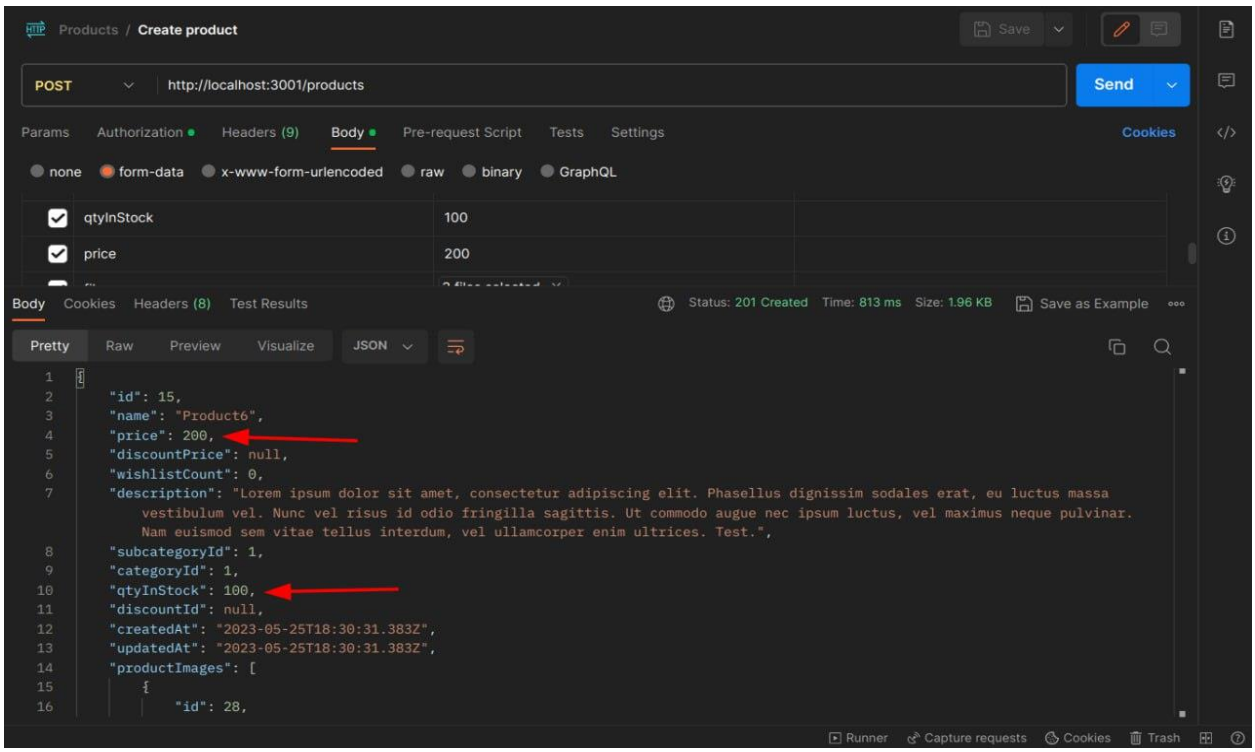

Рисунок 3.15 – Успішне створення продукту

Користувач може додати коментар до продукту: поставити оцінку від 1 до 5 і написати якийсь текст (див. рис. 3.16 та 3.17).

```
1 usage   VladGoogle
async addComment(data: CommentDto) : Promise<Comment & {user: {emai...  {
  try {
    return await this.prisma.comment.create({
      data: {
        ...data,
      },
      include: {
        user: {
          select: {
            firstName: true,
            lastName: true,
            email: true,
            phone: true,
            password: false,
          },
        },
        product: true,
      },
    });
  } catch (e) {
    throw e;
  }
}
```

Рисунок 3.16 – Код для процесу створення коментаря до продукту

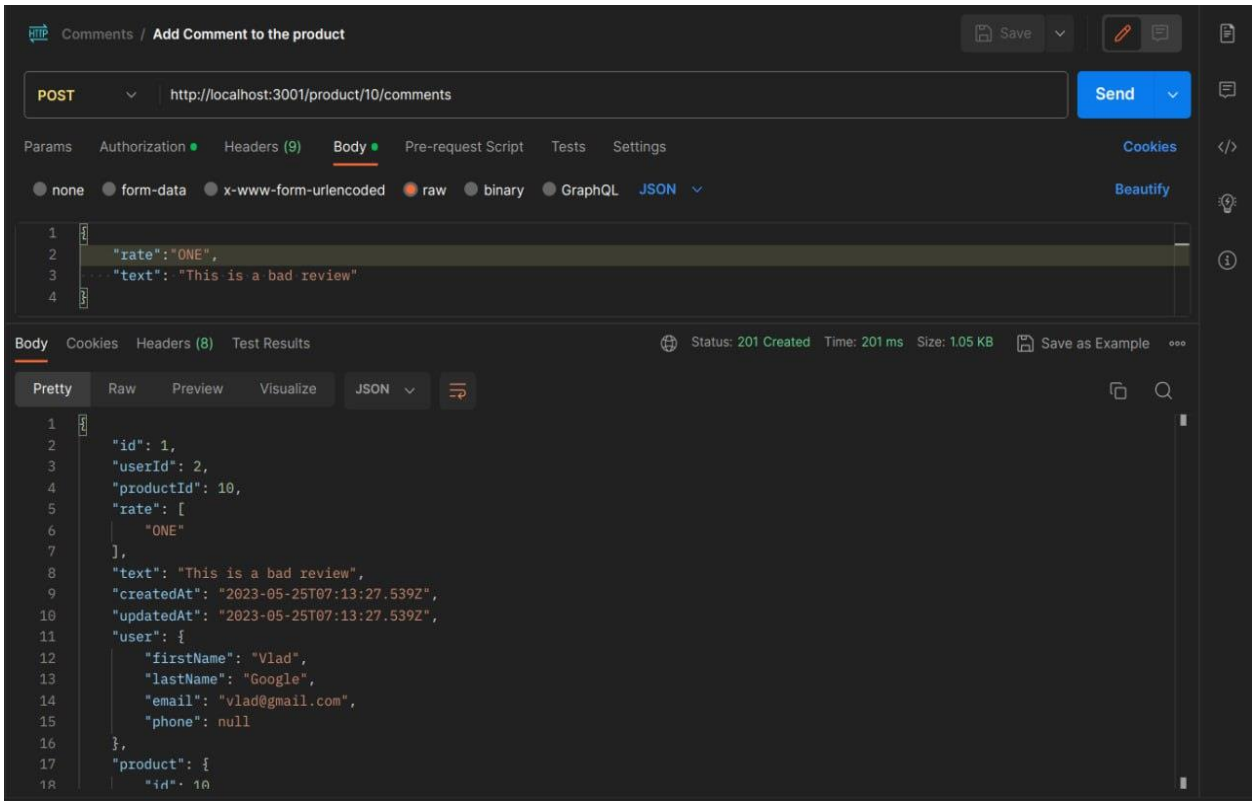

Рисунок 3.17 – Успішне створення коментаря до продукту

Сервер витягне з токену інформацію про користувача, який писав коментар (ім’я та прізвище), і створить запис у базі з підв’язаним до коментаря продуктом і користувачем.

Є ендпойнти для отримання всіх коментарів одного користувача, неважливо, на яких продуктах він це робив, і також всіх коментарів одного продукту.

Користувач з правами адміністратора і системного адміністратора може додати знижку на продукт (див. рис. 3.18, рис. 3.20).

Користувач вводить назву знижки, її процент і статус активності.

Потім сервер розраховує за формулою і встановлює нову ціну з урахуванням знижки у поле “discountPrice”, при цьому, стара ціна залишається незмінною і зберігається у полі “price”.

Коли адміністратор видаляє знижку, значення discountPrice стає null (див. рис. 3.19, рис. 3.21).

```
1 usage  VladGoogle
async addDiscountToTheProduct(id: number, data: DiscountDto) : Promise<Product & {discount: D...  {
  try {
    const discount : Discount  = await this.prisma.discount.create({
      data: {
        ...data,
      },
    });

    const product : Product & {subcategory: Subcat...  = await this.productService.findProductById(id);
    return await this.prisma.product.update( args: {
      where: { id: id },
      data: {
        discountId: discount.id,
        discountPrice:
          product.price - product.price * (discount.discount_percent / 100),
      },
      include: {
        discount: true,
      },
    });
  } catch (e) {
    throw e;
  }
}
```

Рисунок 3.18 – Код для додавання знижки на продукт

```
1 usage  VladGoogle
async deleteDiscountFromProduct(productId: number) : Promise<...>  {
  try {
    const product : Product & {subcategory: Subcat...  = await this.productService.findProductById(productId);
    if (product.discountId === null) {
      return {
        message: `Product with id:${product.id} don't have a discount`,
      };
    } else {
      await this.prisma.discount
        .delete( args: {
          where: {
            id: product.discountId,
          },
        })
        .then( onfulfilled: async () : Promise<void>  => {
          await this.prisma.product.update( args: {
            where: {
              id: productId,
            },
            data: {
              discountPrice: null,
            },
          });
        });
      return await this.productService.findProductById(productId);
    }
  } catch (e) {
```

Рисунок 3.19 – Код для видалення знижки з продукту

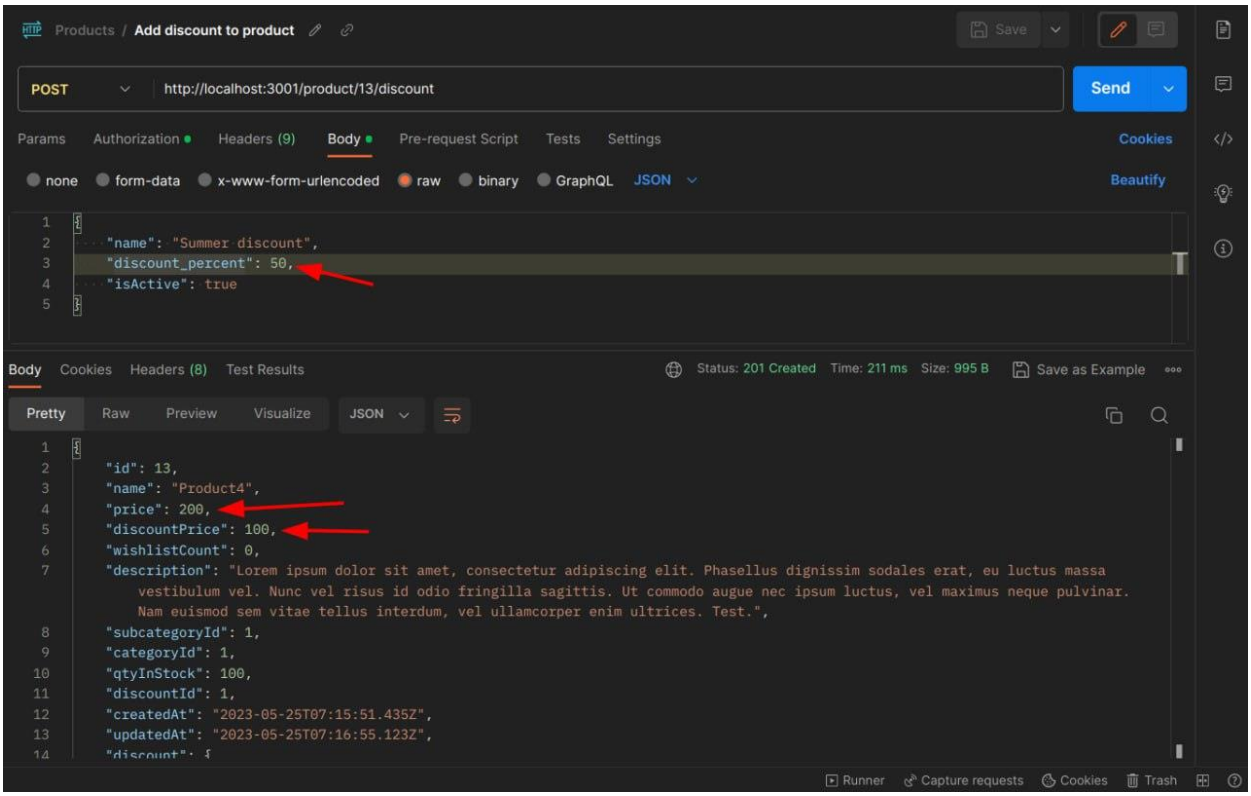

Рисунок 3.20 – Успішне додавання знижки до продукту

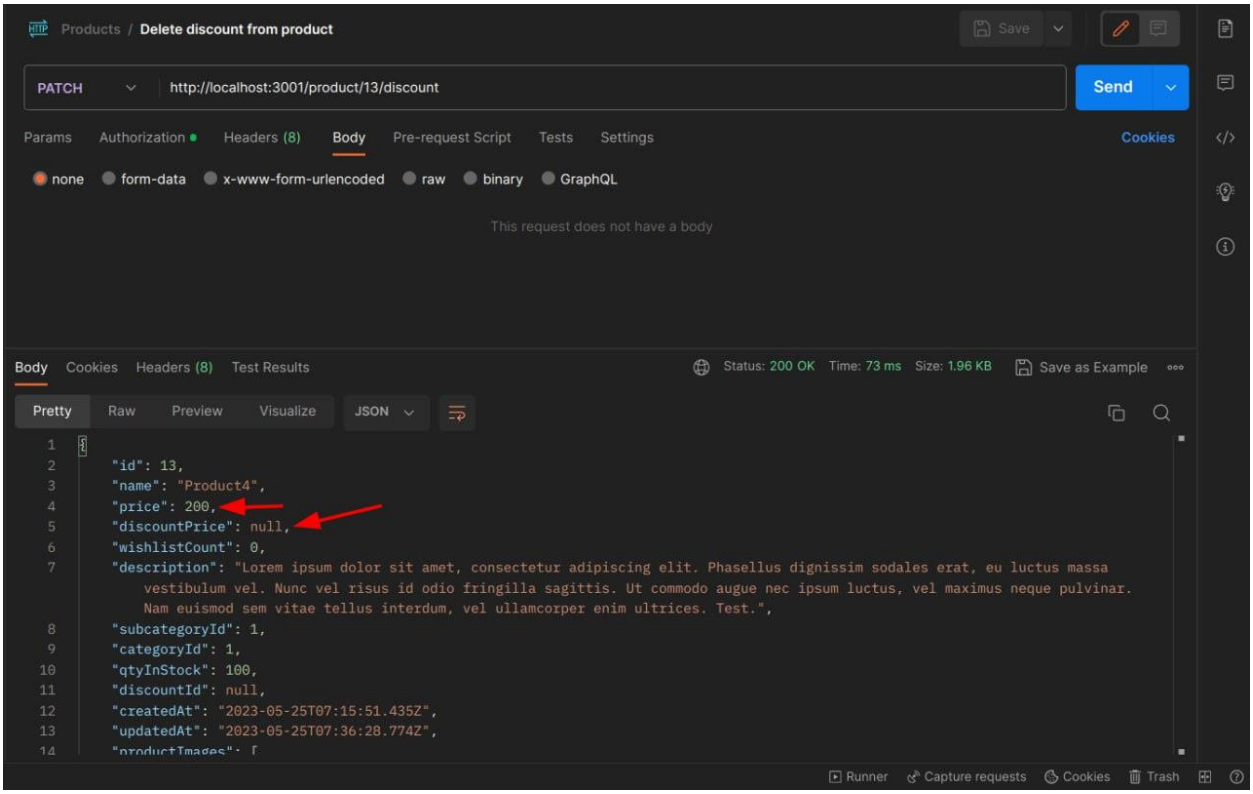

Рисунок 3.21 – Успішне видалення знижки з продукту

Користувач може додати продукт у кошик (див. рис. 3.22, рис. 3.24). Спочатку йде GET запит для отримання кошику по юзер токену на сервер.

Якщо кошика не існує для поточного юзера, то цей кошик створюється і туди зразу же додається продукт з початковою кількістю в одиницю. Рахується підсума одного предмета кошику і загальна вартість кошику. Якщо кошик вже існує, то туди просто додається новий продукт. Загальна вартість кошику збільшується кожний раз з додаванням нового продукту в кошик на вартість доданого продукту, і зменшується, коли якийсь продукт з кошика видаляють, відповідно. Якщо змінюється кількість одного продукту для замовлення, загальна вартість кошику заново перераховується. Також користувач може видалити продукт з кошика (див. рис. 3.23, рис. 3.25).

Рисунок 3.22 – Код для процесу додавання продукту до кошика

Рисунок 3.23 – Код для процесу видалення продукту з кошика

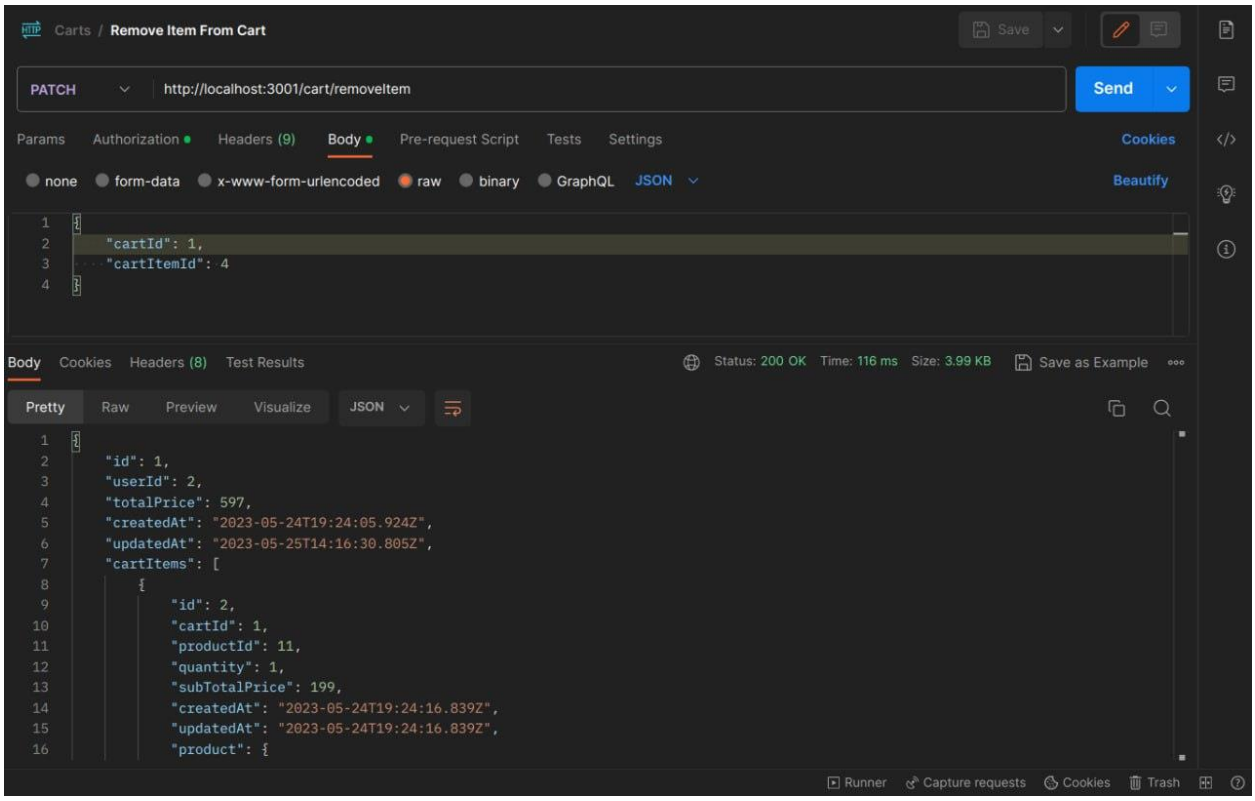

Рисунок 3.24 – Успішне додавання продукту до кошика

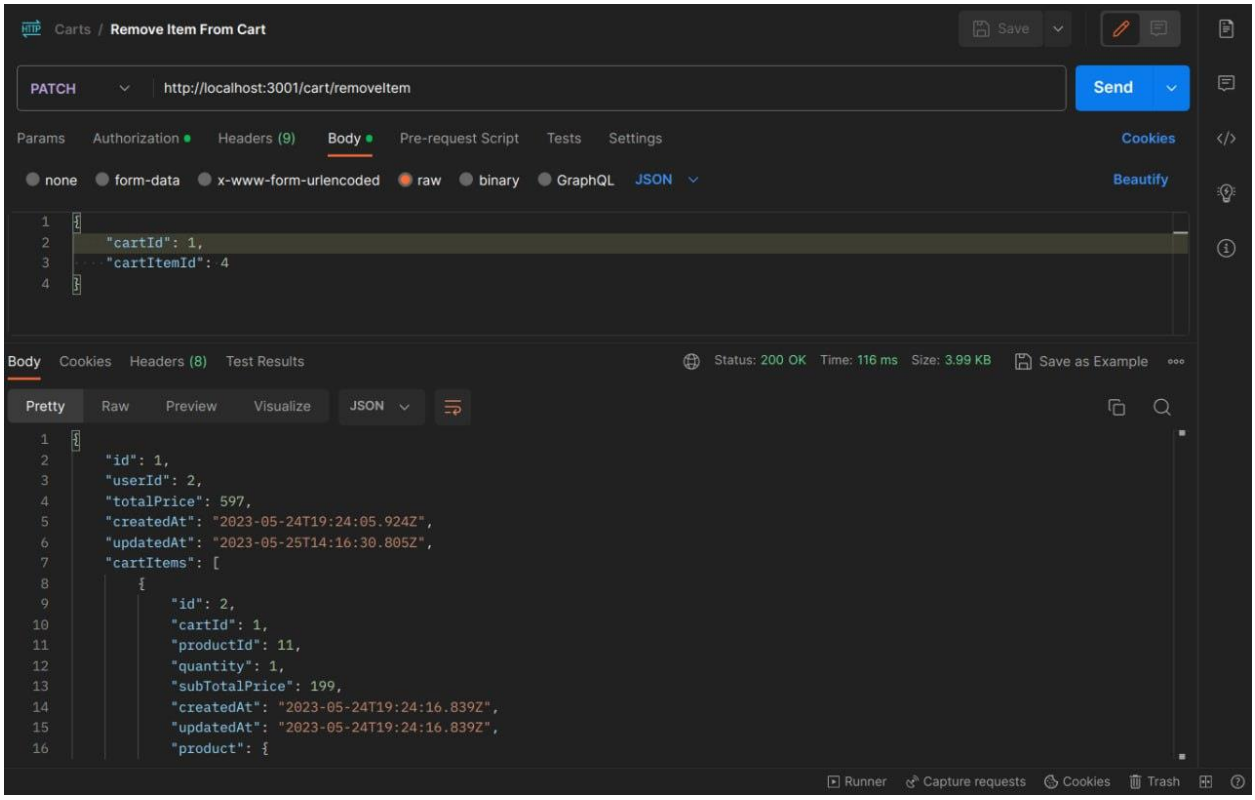

Рисунок 3.25 – Успішне видалення продукту з кошика

Оновлення кількості замовленого продукту в кошику. Користувач може оновити кількість продукту в кошику для замовлення (див. рис. 3.26).

Він передає у реквест ідентифікатор предмета кошику і бажану кількість продукту. Після відправлення реквесту підціна перераховується і оновлюється в запису в базі даних. Зразу після цього заново перераховується загальна ціна кошика і також оновлюється в баз і даних (див. рис. 3.27).

```
async updateCartItemQuantity(data: UpdateCartItemQuantityDto) {
  try {
    const productPrice = await this.getCartItemById(data.cartItemId).then(
      (data) => {
        return data.product.price;
      },
    );
    await this.prisma.cartItem.update({
      where: { id: data.cartItemId },
      data: {
        quantity: data.quantity,
        subTotalPrice: productPrice * data.quantity,
      },
    });
    return await this.updateCartTotalPrice(data.cartId);
  } catch (e) {
    if (e instanceof Prisma.PrismaClientKnownRequestError) {
      if (e.code === 'P2025') {
        throw new NotFoundException(`Item doesn't exist`);
      }
    }
    throw e;
  }
}
```

Рисунок 3.26 – Код для процесу оновлення кількості продукту в кошику

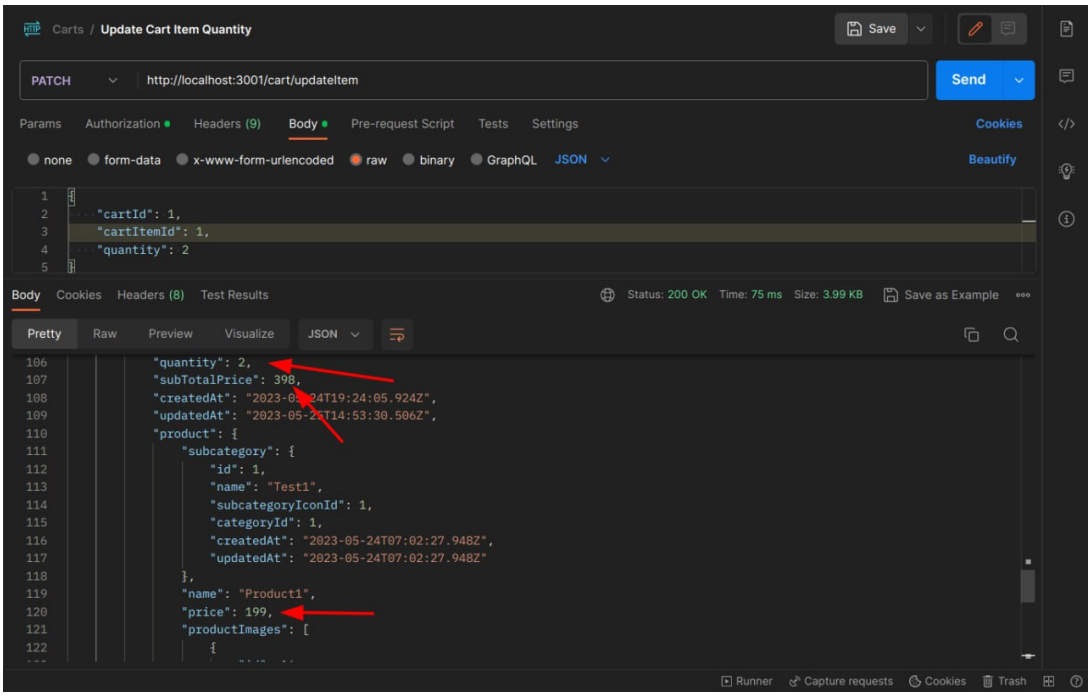

Рисунок 3.27 – Успішне оновлення кількості продукту в кошику

Користувач може додати продукт у список бажаного (див. рис. 3.28, рис. 3.30). Спочатку йде GET запит для отримання кошику по юзер токену на сервер. Якщо списку не існує для поточного юзера, то цей кошик створюється і туди зразу же додається продукт. Одразу після цього, поле «wishlistCounter» доданого продукту збільшується на одиницю, як показник, скільки даний продукт був доданий у кошик різними користувачами загалом. Також користувач може видалити продукт зі списку бажаного (див. рис. 3.29, рис. 3.31).

Рисунок 3.28 – Код для процесу додавання продукту в список бажаного

Рисунок 3.29 – Код для процесу додавання продукту в список бажаного

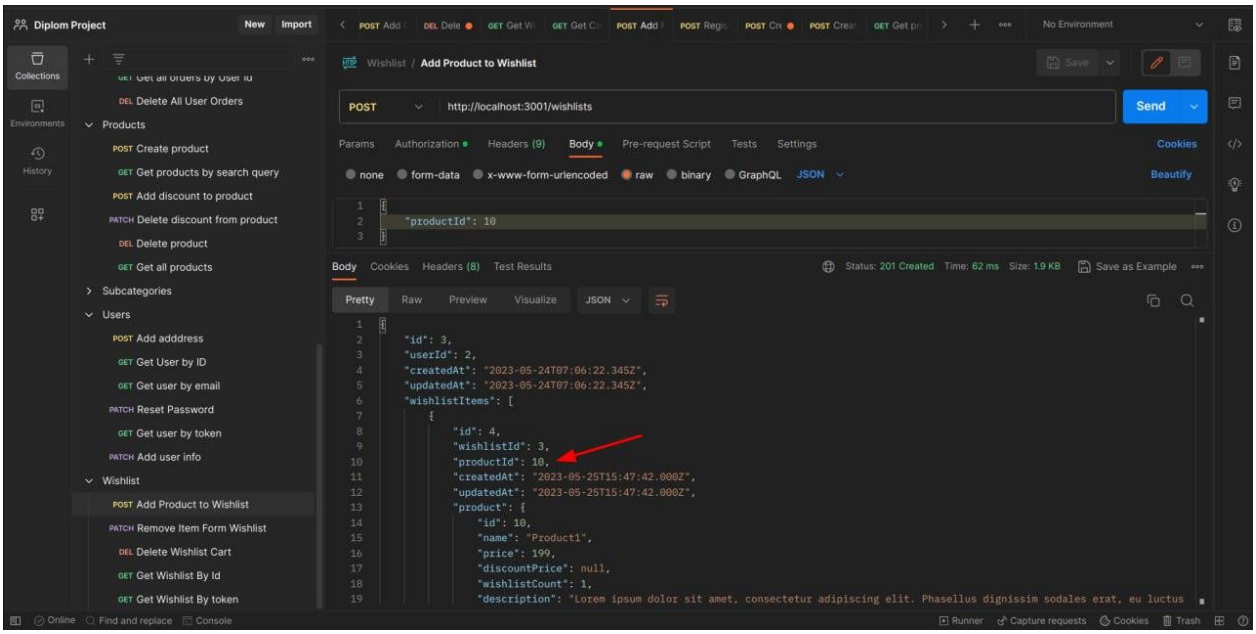

Рисунок 3.30 – Успішне додавання продукту в список бажаного

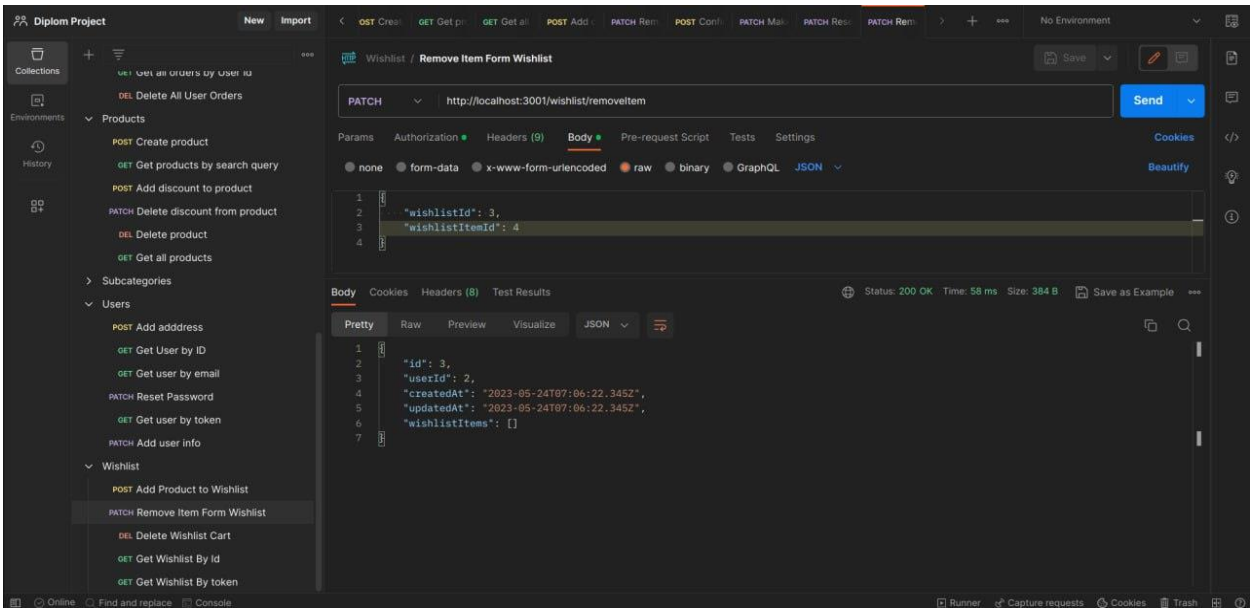

Рисунок 3.31 – Успішне видалення продукту з списку бажаного

Після того, як користувач визначився з кошиком і продуктами, він може перейти до етапу оформлення замовлення (див. рис. 3.32). Після відправки реквесту на створення замовлення, сервер створює новий запис в таблиці Orders, копіює в цей новий запис загальну вартість кошика а також масив замовлених продуктів з їх кількістю. Якщо користувач захоче забрати замовлення з пункту самовидачі, сервер додатково встановить ідентифікатор обраного ним відділення у поле selfCheckoutId (див. рис. 3.33)

```
1 usage   VladGoogle
async createOrder(data: OrderDto, cartId: number) : Promise<Order & {user: {addres...  {
  try {
    await this.getOrderByCartId(cartId).then((order : Order & {user: {address: Addre... ) : void  => {
      if (order) {
        throw new BadRequestException( objectOrError: 'Order with such cart already exists');
      }
    });

    const cart : Cart & {cartItems: (CartItem &...  = await this.cartService.getCartById(cartId);
    return await this.prisma.order.create({
      data: {
        ...data,
        cartId: cartId,
        amount: cart.cartItems.reduce(
          (acc : number , currentValue : CartItem & {product: {name: st... ) => acc + currentValue.subTotalPrice,
          0,
        ),
        orderItems: {
          create: cart.cartItems.map((item : CartItem & {product: {name: st... ) : {...}  => ({
            product: { connect: { id: item.productId } },
            quantity: item.quantity,
            subTotalPrice: item.subTotalPrice,
          })),
        },
      },
      include: {
        user: {
          select: {
```

Рисунок 3.32 – Код для процесу створення замовлення

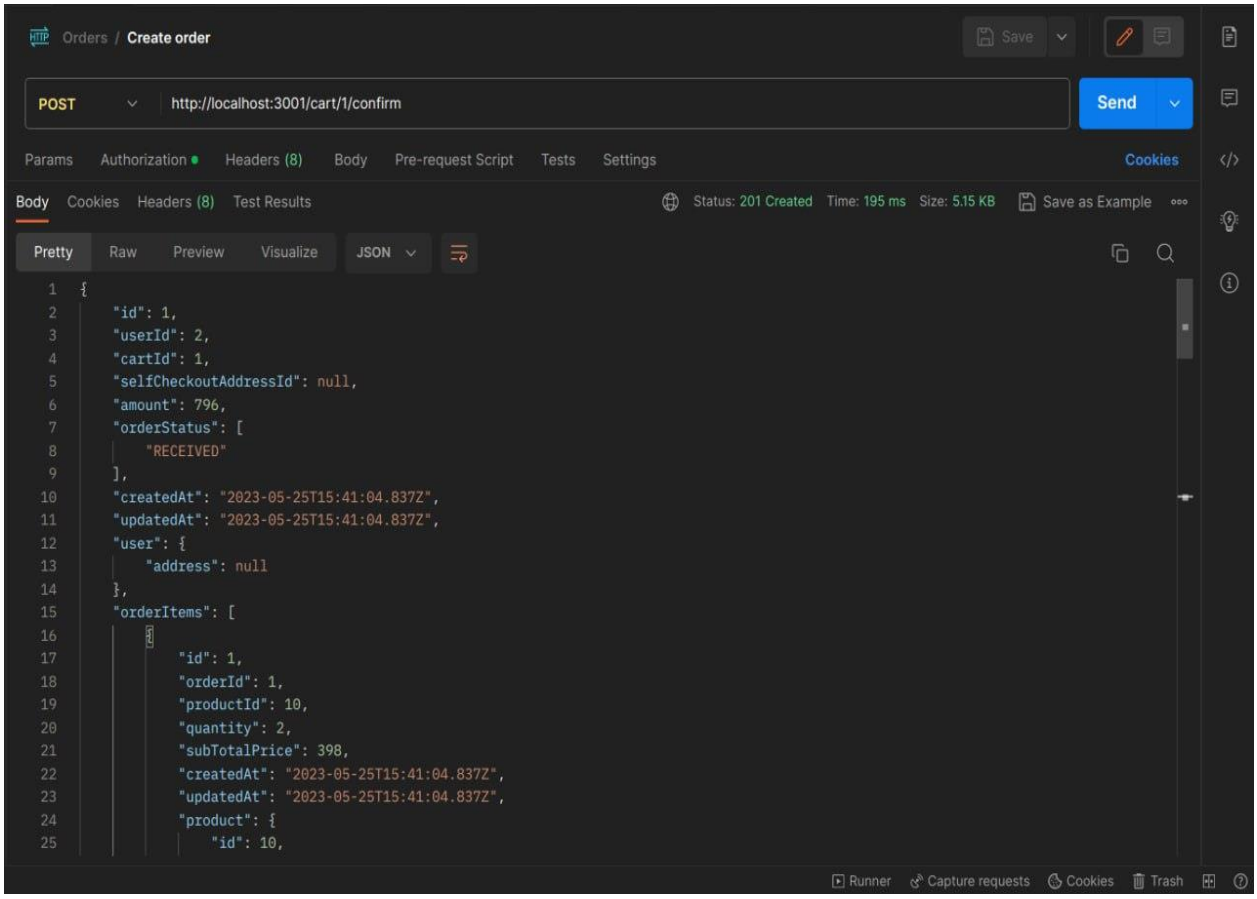

Рисунок 3.33 – Успішне створення замовлення

Користувач може оплатити замовлення (див. рис. 3.34, рис. 3.35) Коли користувач відправляє запит в базу, 3-party бібліотека Stripe виконує charge, в яких входить інформація про його ціну, токен користувача, який виконує оплату, токен кредитної картки, з якої проходить оплата, а також валюта, в якій проходить оплата. Після того, як charge виконався, кошик користувача автоматично видаляється, а також змінюється значення «qtyInStock» кожного замовленого продукту на ту кількість, яка була вказана в кошику на момент оформлення замовлення. У кінці, сервер створює запис у базі у таблицю Payment, куди встановлює ідентифікатор замовлення, ідентифікатор користувача, ідентифікатор картки, а також унікальний ключ чарджу, який повернув Stripe і який був збережений у константі (див. рис. 3.36).

```
const charge : Stripe.Response<Stripe.Charge>  = await this.stripeService.createCharge( data: {
  currency: user.card.currency.toString(),
  amount: Math.floor( x: order.amount * 100),
  source: user.card.cardSource,
  customer: user.customerToken,
});

//Delete the cart
await this.cartService.deleteCartByUserId(authHeader);

//Update products quantity
const orderItems : (OrderItem & {product: (Produc...  = order.orderItems;
await Promise.all(
  orderItems.map(async (orderItem : OrderItem & {product: (Product... ) : Promise<void>  => {
    const product : Product  = await this.prisma.product.findUnique( args: {
      where: { id: orderItem.productId },
    });
    await this.prisma.product.update( args: {
      where: { id: product.id },
      data: { qtyInStock: product.qtyInStock - orderItem.quantity },
    });
  }),
);

//Create payment record in the DB
return await this.paymentQueries.createPayment( data: {
  orderId: orderId,
  userId: user.id,
  cardId: user.card.id
```

Рисунок 3.34 – Код для виконання процесу створення транзакції

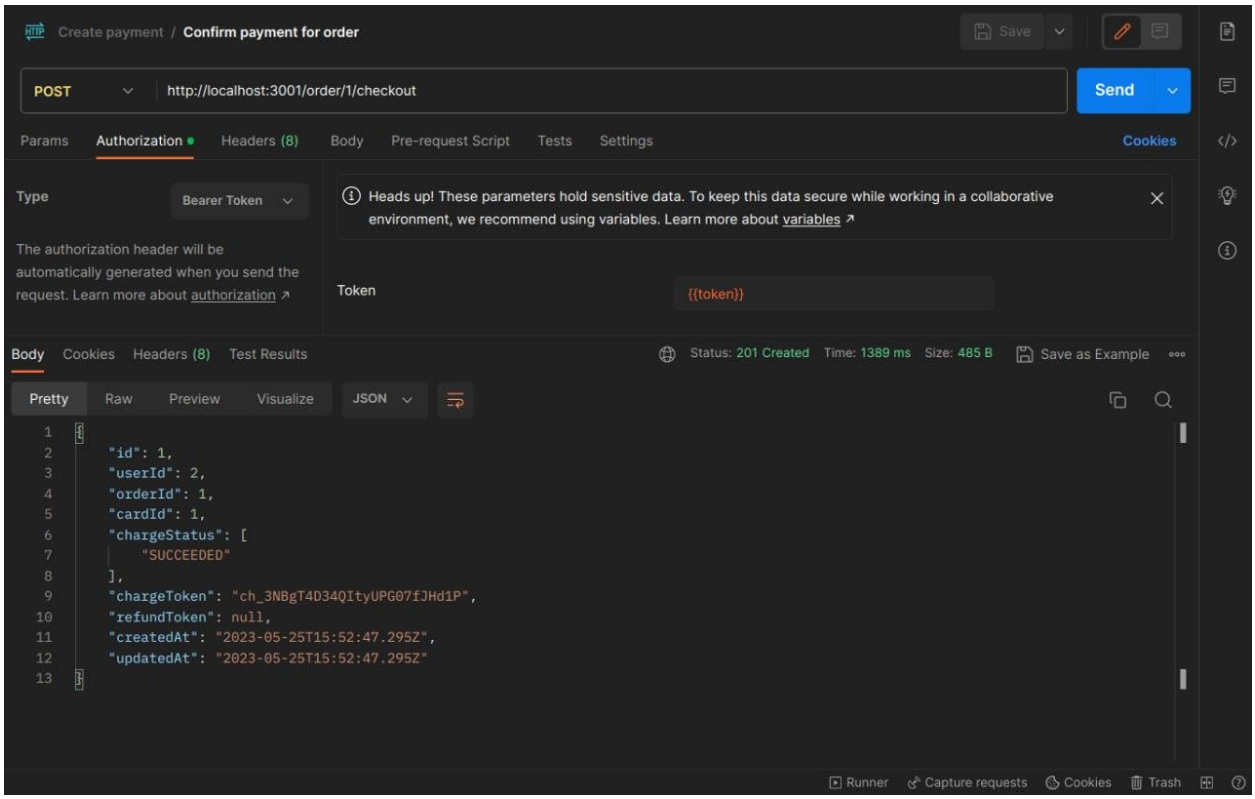

Рисунок 3.35 – Успішне створення замовлення

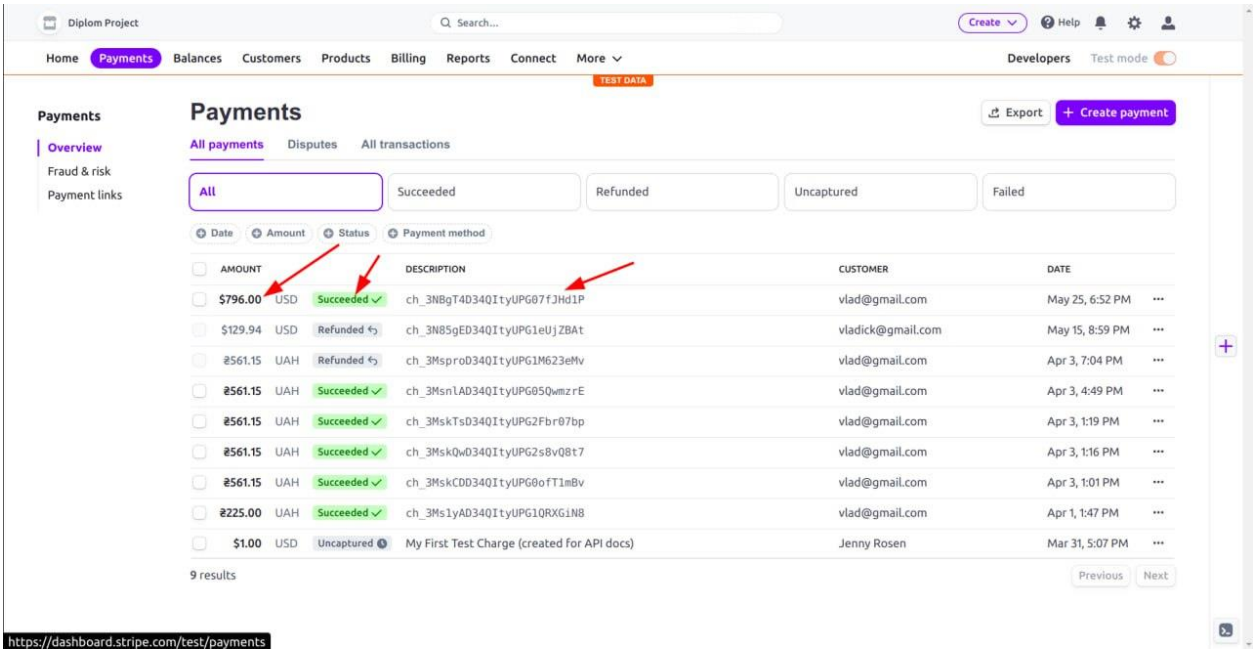

Рисунок 3.36 – Перевірка виконання транзакції в аккаунті Stripe

Користувач може виконати повернення замовлення замовлення (див. рис. 3.37) У тілі реквесту відправляється токен чарджу, який був створений при оформленні оплати замовлення, де Stripe вже використовує свій API для виконання рефанду. Статус транзакціїї змінюється на «REFUNDED», статус

замовлення – на «RETURNED», і в запису транзакції створюється refundToken (див. рис. 3.38, рис. 3.39).

```
async createRefundForPayment(refundToken: string, id: number) {
  try {
    return await this.prisma.payment.update({
      where: { orderId: id },
      data: {
        chargeStatus: ['REFUNDED'],
        refundToken: refundToken,
      },
      include: {
        order: true,
      },
    });
  } catch (e) {
    if (e instanceof Prisma.PrismaClientKnownRequestError) {
      if (e.code === 'P2025') {
        throw new NotFoundException(`Order doesn't exist`);
      }
    }
    throw e;
  }
}
```

Рисунок 3.37 – Код для виконання процесу створення рефанду

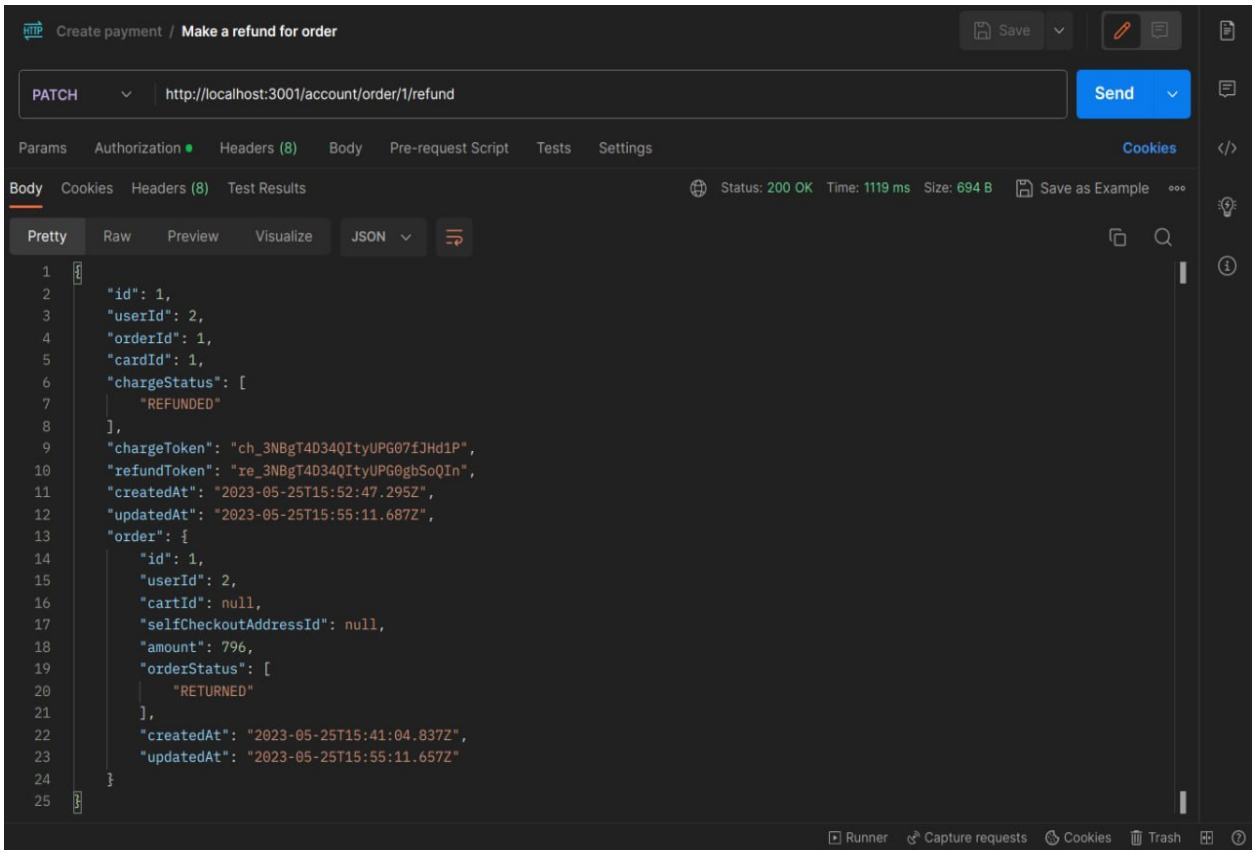

Рисунок 3.38 – Успішне створення рефанду

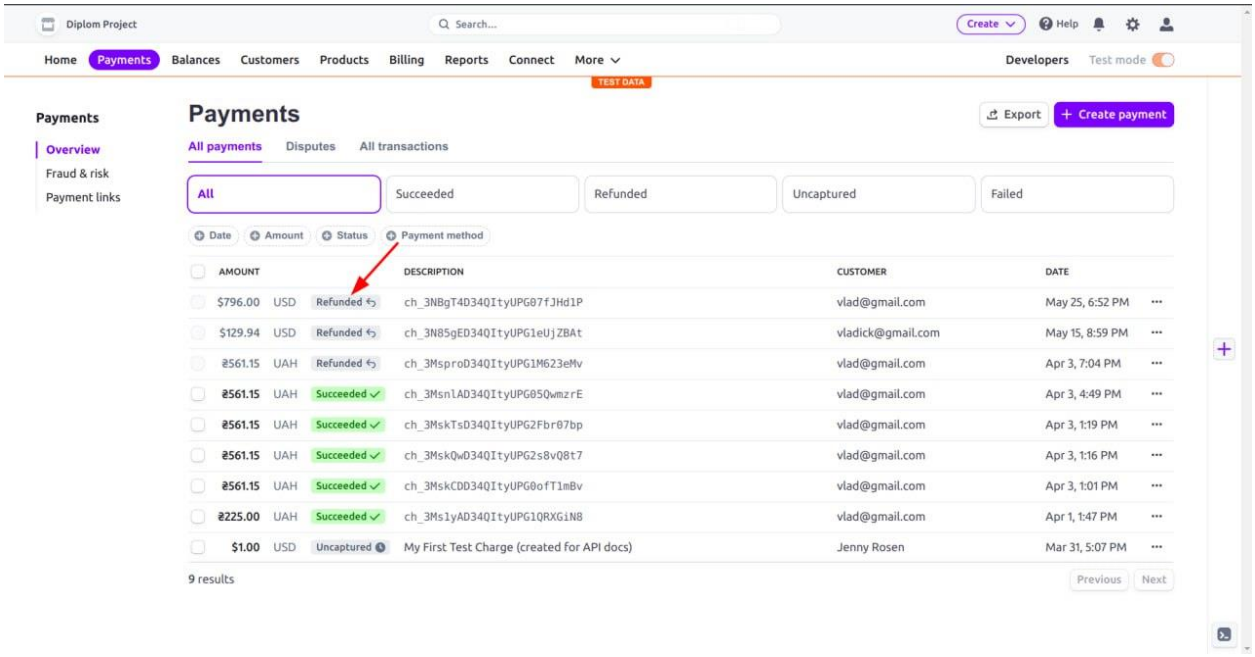

Рисунок 3.39 – Перевірка виконання рефанду в аккаунті Stripe

Проєкт можна контейнеризувати і запустити через Докер. Спочатку, Докер побудує Docker Image (див. рис. 3.40), який описаний в докер-файлі. В докерфайлі спочатку створюється і робиться активною в файловій системі контейнера нова директорія, де і будуть зберігатися файли проєкта. Потім в цю дерикторію копіюються package.json, package-lock.json, yarn.lock і папка prisma (ці всі файли необхідні для коректного встановлення залежностей). Після цього запускається команда «yarn install --only=production»,яка встановлює залежності і бібліотеки, які об’явлені в package.json файлі. Після цього до поточної директорії копіюються усі інші файли, включно з папкою «node_modules», яка з’явилася після попереднього кроку. Фінальним кроком імеджу в докерфайлі є запуск команди «yarn build:server», яка компілює ТайпСкрипт код у зрозумілий для браузера і Node.js, JavaScript код. Сам контейнер серверу стартує з команди «yarn run start:server:dev», який запускає збудований командою «yarn build:server» код (тобто, сервер) у watch моді, який перезапускає сервер при кожній зміні коду. Але перед запуском контейнера сервера, спочатку піднімається контейнер з базою даних. Після того, як контейнер з сервером і базою даних активні, нарешті піднімається контейнер з клієнтом, який будує веб портал, на якому потенційний

користувач буде здійснювати замовлення (див. рис. 3.41).

```
FROM node:19-alpine As Development

ENV NODE_ENV=development

WORKDIR /usr/src/app

COPY package*.json ./
COPY yarn.lock ./
COPY prisma ./prisma/

RUN yarn install --only=production

COPY . .

EXPOSE 3001

RUN yarn build:server;
```

Рисунок 3.40 – Docker Image для контейнера «server»

```
server:
  container_name: server
  build:
    context: .
    dockerfile: Dockerfile_server
    target: development
  volumes:
    - .:/usr/src/app
    - /usr/src/app/node_modules
  env_file: '.env'
  networks:
    - appnet
  depends_on:
    - database
  ports:
    - '3001:3001'
  command: yarn run start:server:dev

database:
  container_name: database
  image: postgres:15
  restart: always
  networks:
    - appnet
  ports:
    - '5432:5432'
  volumes:
    - db_val:/var/lib/postgresql/data
  env_file: '.env'
  environment:
    - POSTGRES_USER=${POSTGRES_USER}
    - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
```

Рисунок 3.41 – Docker Compose файл для контейнерів «server» і «database»

ВИСНОВКИ

У цій дипломній роботі була розглянута тема розробки E-commerce порталу з використанням React та NodeJS. Метою роботи було створення функціонального та ефективного онлайн-магазину, який би задовольняв потреби як продавців, так і покупців. У рамках дослідження було проведено аналіз сучасних тенденцій в галузі електронної комерції, вивчено основні принципи розробки React та NodeJS-додатків, а також розроблено та протестовано власний E-commerce портал.

Під час розробки було використано сучасні технології та інструменти, які дозволяють ефективно будувати складні веб-додатки. React, як фронтенд-бібліотека, надавав можливість створювати компонентну структуру і забезпечувати швидку та інтерактивну роботу з додатком. NodeJS, у свою чергу, забезпечував побудову серверної частини, а також взаємодію з базою даних.

Під час розробки E-commerce порталу були реалізовані основні функціональні можливості, такі як реєстрація та авторизація користувачів, перегляд каталогу товарів, додавання товарів до кошика, здійснення покупок, керування замовленнями, оплата та доставка. Було також забезпечено можливість пошуку товарів, фільтрації за різними параметрами та відображення деталей товарів.

Під час тестування розробленого додатку було перевірено його функціональність, стійкість та швидкодію. Тести показали, що портал працює стабільно, забезпечує коректне виконання запитів і відповіді на них в прийнятний час.

Результати роботи свідчать про те, що розроблений E-commerce портал є функціональним та ефективним рішенням для онлайн-торгівлі. Використання React та NodeJS дозволяє створити масштабований та легко розширюваний додаток, який може задовольнити потреби як маленьких, так і

великих підприємств. Багатофункціональний і зручний інтерфейс порталу сприяє зручному та швидкому використанню як для продавців, так і для покупців.

Однак, варто відзначити, що розроблений додаток має потенціал для подальшого розширення та вдосконалення. Наприклад, можна додати функцію створення персоналізованих рекомендацій для покупців або розробити мобільну версію порталу. Крім того, можна вдосконалити процес оптимізації завантаження сторінок та покращити безпеку системи.

У цілому, розробка E-commerce порталу з використанням React та NodeJS є актуальною та перспективною задачею. Технології, використані у цій роботі, дозволяють створювати потужні та ефективні додатки, які можуть привернути широке коло користувачів і забезпечити успішну онлайн-торгівлю.

ПЕРЕЛІК ПОСИЛАНЬ

1. Git Repository with project. GitHub. URL: https://github.com/VladGoogle/diplom-project (дата звернення: 08.04.2023).

2. StarUML Homepage. StarUML Documentation. URL: https://staruml.io/ (дата звернення: 06.04.2023).

3. ER and Class diagrams link. Google Drive. URL: https://drive.google.com/file/d/1tQ0Jyig5E3C7QbSXUz9n8f37MclMDxMv/view?usp=sharing (дата звернення: 09.04.2023).

4. React Homepage. React Documentation. URL: https://react.dev/ (дата звернення: 01.04.2023).

5. Node.js homepage. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 02.03.2023).

6. NestJS introduction. NestJS Documentation. URL: https://docs.nestjs.com/ (дата звернення: 03.04.2023).

7. Prisma Homepage. Prisma Documentation. URL: https://www.prisma.io/docs (дата звернення: 04.03.2023).

8. ChatGPT homepage. ChatGPT bot. URL: https://openai.com/blog/chatgpt (дата звернення: 07.03.2023).

9. Docker «Get Started» page. Docker Documentation. URL: https://docs.docker.com/get-started/ (дата звернення: 05.04.2023).