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

«До захисту допущено»
В.о. завідувача кафедри

_____Світлана ВАЩЕНКО

____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»,
освітньо-професійної програми «Інформаційні технології проектування»
на тему: "Web-додаток підтримки продажу електронної техніки"
Здобувача групи ІТ-91 Куликова Олексія Олександровича

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

_____ ___Олексій КУЛИКОВ___
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник _____Анна МАРЧЕНКО_____ _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2023

Сумський державний університет

Факультет електроніки та інформаційних технологій

Кафедра інформаційних технологій

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ

В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО

«__» _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Куликову Олексію Олександровичу

1 Тема роботи *Web-додаток підтримки продажу електронної техніки*

керівник роботи *Марченко Анна Вікторівна, к.т.н., доцент*

затверджені наказом по університету від « 29 » травня 2023 р. №0588-VI

2 Строк подання студентом роботи « 7 » червня 2023 р.

3 Вхідні дані до роботи Технічне завдання на розробку web-додатку підтримки продажу електронної техніки

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) Аналіз предметної області, моделювання та проектування web-додатку, розробка web-додатку

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Актуальність, постановка проблеми, порівняння існуючих web-додатків, функціональні вимоги, моделювання роботи web-додатку, діаграма

варіантів використання, архітектура web-додатку, технології та засоби реалізації, опис технологій та засобів реалізації для клієнтської та серверної частин web-додатку, демонстрація web-додатку

6 Консультанти розділів роботи:

Розділ	**Консультант**	**Підпис, дата**	
		Завдання видав	**Завдання прийняв**

7 Дата видачі завдання 12.09.2022

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Розробка дизайну web-додатку	16.01.23 – 24.01.23	
2	Розробка бази даних	25.01.23 – 14.02.23	
3	Back-end розробка	15.02.23 – 28.03.23	
4	Front-end розробка	29.03.23 – 09.05.23	
5	Наповнення web-додатку контентом	10.05.23 – 16.05.23	
6	Оформлення документації	17.05.23 – 05.06.23	
7	Захист	14.06.23 – 19.06.23	

Студент _____ Олексій КУЛИКОВ

(підпис)

Керівник роботи _____ к.т.н., доц. Анна МАРЧЕНКО

(підпис)

ЗМІСТ

ВСТУП

З кожним днем все більше розширюється використання інформаційних технологій, оскільки вони стають невід'ємною частиною нашого щоденного життя. Сучасні компанії активно розширюють свою діяльність в онлайні. Одним із актуальних питань є автоматизація різноманітних процесів у сфері надання послуг та товарів. Люди щодня відвідують багато магазинів, щоб придбати товари, техніку, одяг тощо. Розноманітність магазинів та торгових центрів вражає. Щоб бути конкурентоспроможним, магазину потрібно мати свою унікальну концепцію, яка допоможе виділятися серед інших, привертати увагу клієнтів та завойовувати популярність.

Мати власний інтернет-магазин є великою перевагою для будь-якого роздрібного закладу. Web-додатки є ефективним інструментом реклами. Клієнти мають можливість скористатися web-сайтом, щоб отримати більш детальну інформацію про розклад роботи, концепцію, асортимент товарів та послуг, що надає компанія. Вони можуть переглянути фотографії товарів, розмістити замовлення, зв'язатися з технічною підтримкою для отримання додаткових відомостей та зробити ще багато інших речей. Web-додатки допомагають ефективно організувати роботу магазинів і вже давно займають важливе місце в онлайн-середовищі та користуються великою популярністю серед користувачів. Проте багато таких сайтів, які були розроблені, залишилися на низькому рівні.

Таким чином, це дослідження спрямоване на розробку web-додатку для підтримки продажу електронної техніки та впровадження інноваційних рішень, що спростять процес вибору та покупки для користувачів. Основними цілями дослідження є поліпшення взаємодії між користувачем і продуктом, зменшення навантаження інформацією, підвищення задоволення користувачів під час процесу покупки товарів, а також підвищення ефективності продажів для бізнесу.

Для досягнення цілей проекту необхідно виконати наступні завдання: оцінити актуальність роботи, визначити цільову аудиторію та провести дослідження предметної області, проаналізувати існуючі web-додатки для виявлення їх переваг та недоліків, розробити модель та структуру додатку, вибрати технології для розробки, створити прототип, реалізувати структуру та розробити функціонал додатку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Розробка web-додатку онлайн-магазину електроніки – це процес, що потребує глибокого розуміння технологій та специфіки електронної комерції. Останні дослідження та публікації в галузі розробки web-додатків надають цінну інформацію та рекомендації щодо побудови оптимальних web-додатків.

Одним з головних аспектів розробки онлайн-магазинів є їхній інтерфейс. Дослідження показують, що ефективний та зручний інтерфейс сприяє збільшенню конверсії та задоволення користувачів. У своїй статті "The Psychology of Web Design: How Colors, Typefaces and Spacing Affect your Mood" (Психологія web-дизайну: як кольори, шрифти та простір впливають на ваш настрій) Емілі Рудд розглядає вплив кольорів, шрифтів та інші аспекти web-дизайну на почуття та настрій користувачів [5].

Крім того, останні дослідження в області розробки web-додатків звертають увагу на важливість швидкості роботи та завантаження сторінок. У статті "How Website Speed Can Impact Your Business" (Як швидкість web-сайту може вплинути на ваш бізнес) автор Браян Джонсон зазначає, що повільно працюючі сайти можуть втратити понад 50% відвідувачів, що впливає на конверсію та прибуток [6].

Ще одним важливим аспектом розробки онлайн-магазинів є безпека. У своїй статті "Security Issues in E-commerce: A Review" (Проблеми безпеки в електронній комерції: огляд) Душянт Каушик та інші співавтори розглядають проблеми безпеки в електронній комерції та надають рекомендації з покращення безпеки web-додатків. Дослідження показали, що недостатня безпека може призвести до втрати даних користувачів, що може вплинути на репутацію бренду та призвести до втрати клієнтів [7].

Нарешті, останні дослідження в області розробки web-додатків звертають увагу на важливість аналітики та відстеження даних. У статті "Customer Analytics: Where the Answers to All of Your Ecommerce Marketing Questions Live" (Аналітика клієнтів: де живуть відповіді на всі ваші маркетингові запитання електронної комерції) Алекс МакПік розглядає важливість аналізу даних для розуміння поведінки користувачів та оптимізації web-додатків. Дослідження показали, що аналітика може допомогти зрозуміти, які товари популярні серед клієнтів, які сторінки web-додатку відвідуються найчастіше та як користувачі взаємодіють з сайтом [8].

Аналіз останніх досліджень і публікацій в області розробки web-додатків зазначає важливість реалізації дружнього, інтуїтивно-зрозумілого дизайну web- сторінок з урахуванням вподобань та емоційного сприйняття користувачем форми подання інформації для отримання очікуваного ефекту від впровадження інтерне-магазину в роботу.

1.2 Аналіз існуючих продуктів-аналогів

Вже давно в мережі знаходяться web-додатки для інтернет-магазинів. Однак, більшість невеликих компаній, які мають свої сайти, залишилися на примітивному рівні після їх розробки. На таких сайтах можна знайти застарілі фотографії низької якості, відсутність актуальних новин, а також відсутність будь-яких сучасних додаткових функцій [9]. Отже, важливим елементом конкурентоспроможності web-додатків є відповідність усім сучасним тенденціям розробки [10]. Для встановлення вимог до майбутнього web-додатку, було здійснено дослідження наявних аналогів онлайн-магазинів у місті Суми, а саме «Compservice» [11], «Eldorado» [12] та «Bicom» [13].

На головній сторінці web-сайту магазину «Compservice» (рис. 1.1) розміщені можливі способи зв'язку, графік, навігація по категоріям товару,

рядок пошуку та блок з вкладками “популярні”, “новинки” та “акційні” товари, також є короткий опис закладу і кнопка для онлайн-зв'язку з консультантом.

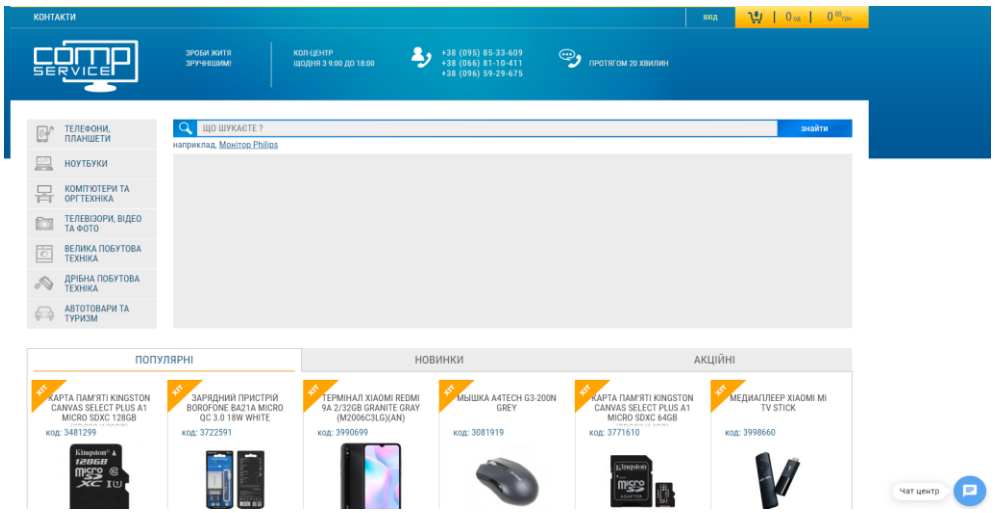


Рисунок 1.1 – Головна сторінка web-сайту “Compservice”

Щоб потрапити на сторінку каталогу (рис. 1.2), потрібно обрати одну з категорій, або вписати найменування товару у рядок пошуку.

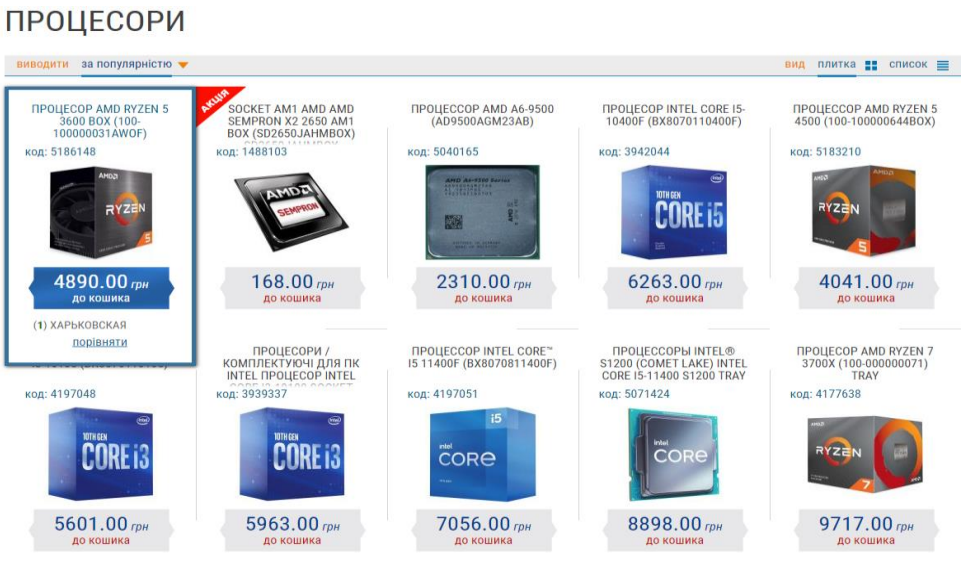


Рисунок 1.2 – Сторінка каталогу web-сайту “Compservice”

Замовити продукцію можна, додавши товар до кошика (рис. 1.3), а потім на сторінці “Кошик” можна оформити замовлення.

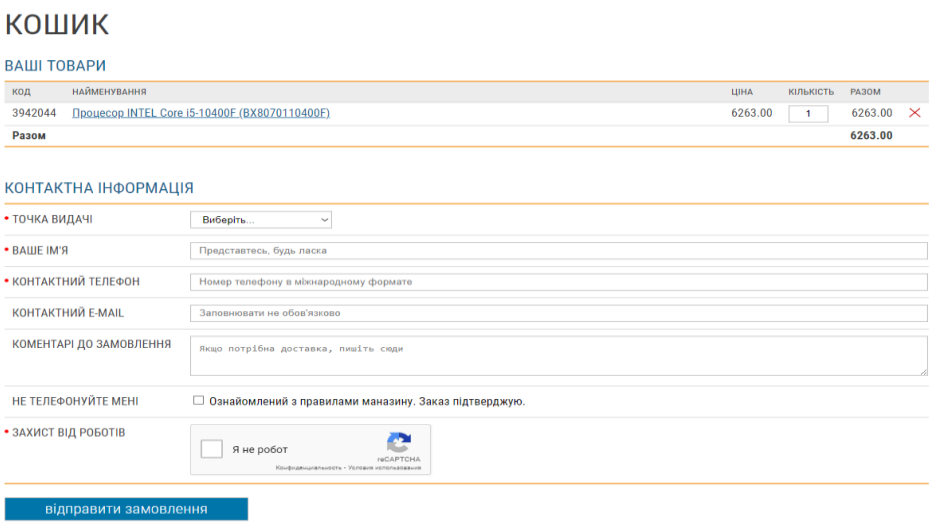


Рисунок 1.3 – “Кошик” web-сайту “Compservice”

Головна сторінка web сайту магазину “Eldorado” (рис. 1.4) дещо схожа на вище описану. Тут також присутні контакти, список категорій товару, кілька вкладок блоку “Наші пропозиції”, але з нового, на сторінці є карусель акційних товарів та маркетингових пропозицій і кілька “карток” з акційними актуальними категоріями.

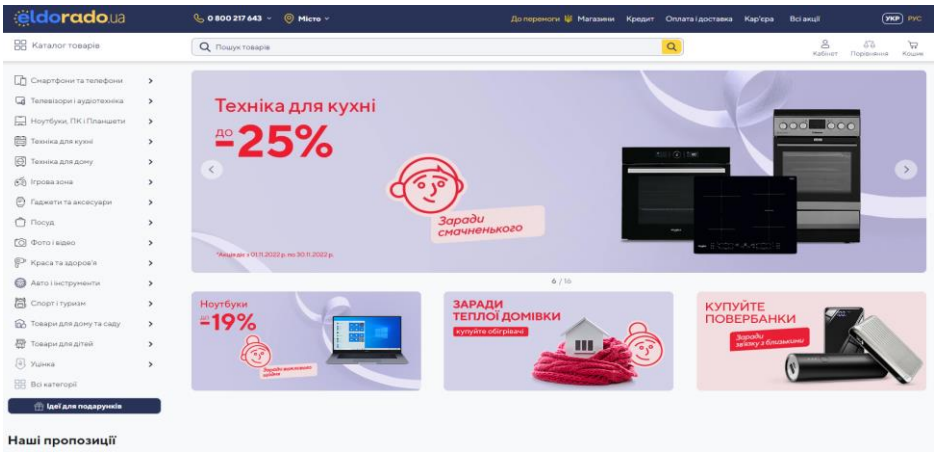


Рисунок 1.4 – Головна сторінка web-сайту “Eldorado”

До каталогу можна потрапити так само, обравши потрібну категорію товарів у списку (рис. 1.5).

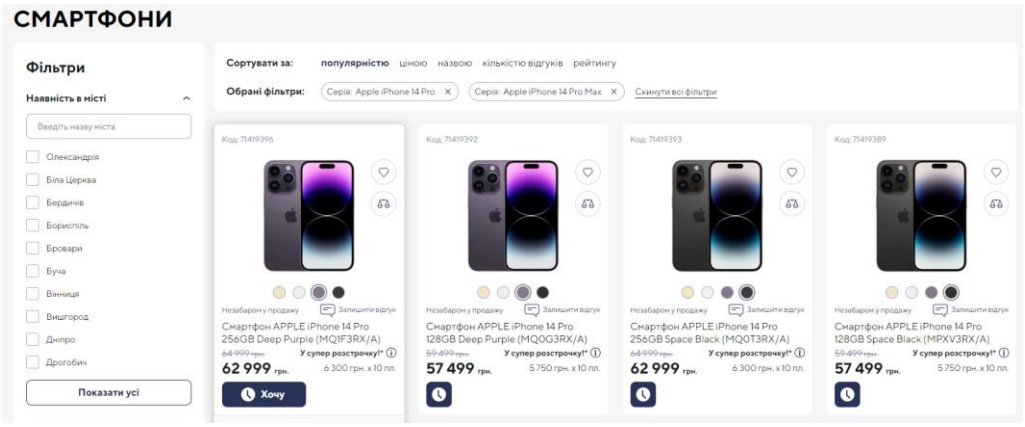


Рисунок 1.5 – Сторінка каталогу web-сайту “Eldorado”

На сайті магазину доступна функція передзамовлення. Якщо натиснути кнопку “Хочу”, потрібно буде ввести ім'я, номер телефону та місто доставки. Проте, щоб додати до кошика і замовити одразу декілька товарів, необхідно пройти реєстрацію на сайті.

Головна сторінка магазину “Bicom” (рис. 1.6) містить схожу інформацію з вище описаними онлайн-магазинами, проте оформлення дещо сучасніше від інших. Також на сторінці присутній так званий підвал сайту, в якому розміщені додаткові можливості по типу вже переглянути, обраних, порівняних та доданих до кошику товарів, що є дуже зручним і досить цікаво оформленим функціоналом.

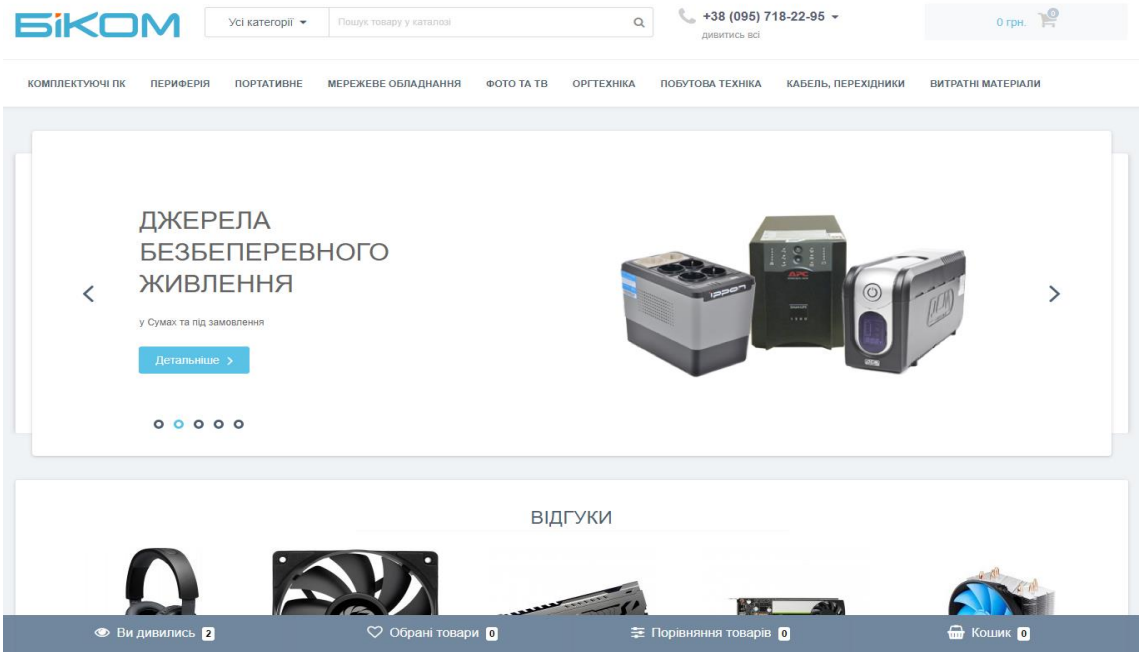


Рисунок 1.6 – Головна сторінка web-сайту “Bicom”

До каталогу товарів (рис 1.7) можна потрапити шляхом обрання запропонованих категорій, що знаходяться нижче шапки сайту. А переглянути список усіх категорій можна, натиснувши на кнопку “Усі категорії”.

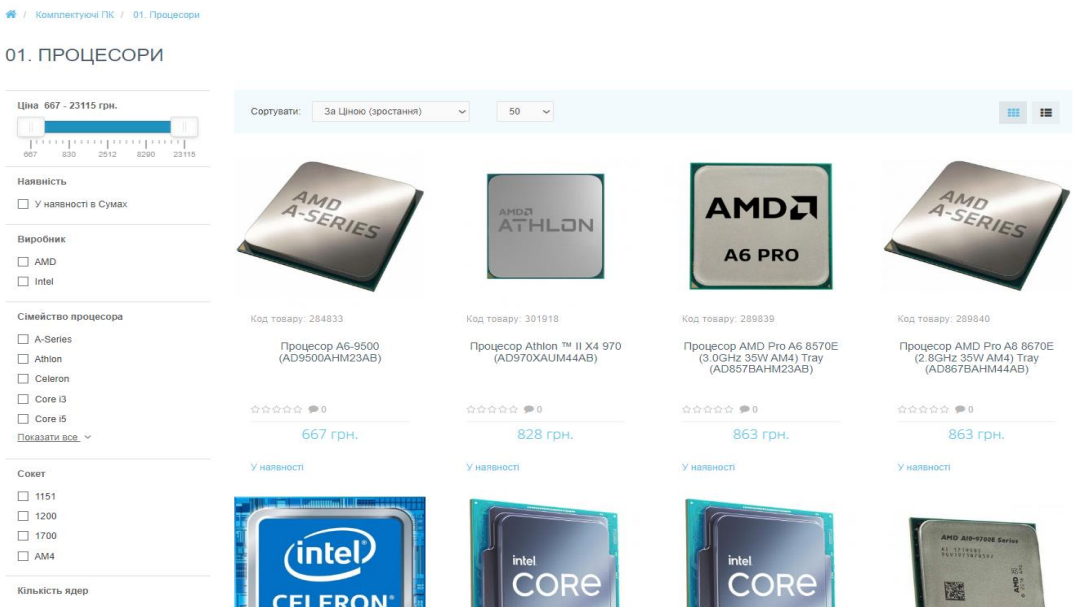


Рисунок 1.7 – Сторінка каталогу web-сайту “Bicom”

Оформлення замовлення (рис. 1.8) відбувається на окремій сторінці, на яку можна потрапити шляхом натиснення кнопки “Купити” у кошику. Також можна обрати спосіб оплати: готівкою, банківським переказом або безготівковим розрахунком.

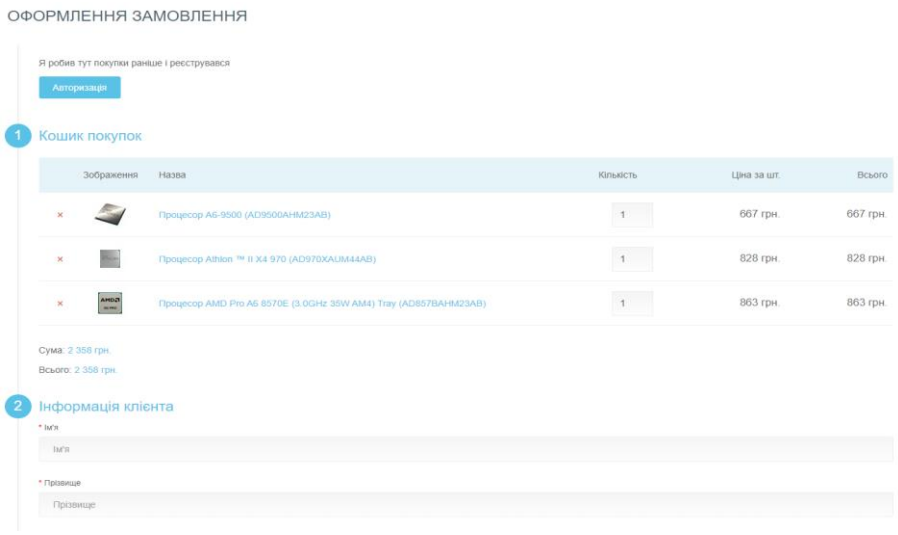


Рисунок 1.8 – Сторінка оформлення замовлення на web-сайті “Bicom”

Після ретельного дослідження web-додатків-аналогів магазинів електронної техніки, було визначено їх переваги та недоліки. Загалом, можна сказати, що “Bicom” значно краще підійшов до організації процесу оформлення замовлень та зовнішнього вигляду web-сайту ніж його конкуренти. Результати аналізу представлені в таблиці 1.1.

Таблиця 1.1 – Порівняльна таблиця характеристик аналогів web-додатків

Магазин / Характеристика	“Compservice”	“Eldorado”	“Bicom”	“Silex”
Сучасний дизайн	-	-	+	+
Зручний інтерфейс	-	-	+	+
Інтерактивність	-	+	+	+
Функціональність	+	+	+	+

Продовження таблиці 1.1

Навігація	+	+	+	+
Реєстрація користувачів	+	+	+	+
Замовлення товарів онлайн без обов'язкової реєстрації	+	-	+	+
Наявність онлайн зв'язку з тех. підтримкою	+	-	-	+

Дані з таблиці 1.1 надають змогу під час розробки звернути увагу на цікаві функціональні доповнення, які можна використати, і недоліки, які варто подолати. Web-додаток повинен мати сучасний дизайн, зручну навігацію та інтерфейс, інтерактивність web-сторінок. З функціональних доповнень варто виділити реєстрацію користувачів, замовлення товарів онлайн без обов'язкової реєстрації, можливість онлайн зв'язку з технічною підтримкою.

1.3 Постановка задачі

Це дослідження має на меті розробку web-додатку для онлайн-магазину електронної техніки, який буде мати зручний та легкий у використанні інтерфейс. Цей додаток сприятиме ефективній організації продажу товарів шляхом автоматизації бізнес-процесів і впровадження інноваційних рішень. Такий підхід допоможе спростити процес вибору та покупки товарів для користувачів. Це включає в себе вивчення поведінки споживачів, їхніх потреб і проблем, пов'язаних з вибором товарів, а також розробку технологічних рішень, які забезпечують зручний і ефективний досвід покупки. Основними цілями дослідження є покращення взаємодії між користувачем і продуктом,

зменшення інформаційного перевантаження, підвищення рівня задоволеності користувачів під час процесу придбання товару, і збільшення ефективності продажів для бізнесу.

Основні вимоги до створюваного web-додатку є наступними:

- підтримка модулю реєстрації/авторизації;
- підтримка модулю оформлення замовлення;
- підтримка пошуку і фільтрації товару за різними критеріями;
- підтримка якісного наповнення каталогу товарів.

Для досягнення мети проекту потрібно виконати такі завдання:

- встановити актуальність проекту, визначити цільову аудиторію та провести дослідження предметної області;
- здійснити аналіз web-додатків-аналогів для виявлення їх переваг та недоліків;
- розробити модель та структуру web-додатку;
- вибрати технології для розробки;
- створити прототип web-додатку;
- реалізувати структуру web-додатку;
- розробити функціонал web-додатку.

Технічне завдання на розробку проекту (додаток А) містить опис вимог до проекту, до структури додатку, до видів забезпечення та функціонування системи. Для реалізації цього web-додатку були вибрані наступні технології: HTML (HyperText Markup Language) для створення основи сторінок сайту, CSS (Cascading Style Sheets) для надання сторінкам візуальних ефектів та пристосованості до різних розмірів екранів, динамічну мову програмування JavaScript для асинхронної взаємодії з серверною частиною додатку та web-фреймворк Spring [14] для створення бекенду додатку. Такий вибір обумовлюється дуже широкими можливостями даного фреймворку. Він дозволяє зосередитися саме на організації логіки додатку, а решту процесів автоматизує всередині себе, тобто “під капотом”.

Для створення бази даних у розроблюваному web-додатку було вибрано систему управління базами даних (СУБД) PostgreSQL [15], враховуючи такі переваги, як гнучке налаштування бази даних, розширений функціонал, а також інтеграція з Spring фреймворком.

1.4 Висновки по першому розділу

У результаті аналізу предметної області було сформовано мету розробки проекту, технічне завдання на розробку (Додаток А) і планування робіт (Додаток Б) проведено аналіз потреб користувачів для прийняття необхідних мір для задоволення потреб. У якості зразків для порівняння існуючих додатків аналогів, було обрано Bicom, Compservice та Eldorado. При детальному дослідженні можливостей сайтів даних магазинів, було прийнято рішення до вдосконалення цих недоліків для власного проекту. Розкрито мету розробки web-додатку онлайн магазину електронної техніки, були визначені цілі проекту та основні функції, які має включати в себе web-додаток. Крім того, проведено аналіз можливих підходів до реалізації поставленої задачі.

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ ПІДТРИМКИ ПРОДАЖУ ЕЛКТРОННОЇ ТЕХНІКИ

2.1 Функціональне моделювання web-додатку в IDEF0

Методологія IDEF0 використовується для формалізації та опису бізнес-процесів шляхом функціонального моделювання та графічного представлення. Одним з основних аспектів IDEF0 є його підкреслення ієрархічного представлення об'єктів, що сприяє полегшенню розуміння предметної області. Крім того, усі сигнали управління також відображаються в цій моделі. Ця модель є однією з найбільш передових і застосовується для управління проектами, де передбачається моделювання адміністративних та організаційних процесів [16].

На рисунку 2.1 представлена діаграма функціонального моделювання IDEF0. Вона складається з функціонального блоку, вхідних та вихідних даних, елементів управління та механізмів для виконання функції.

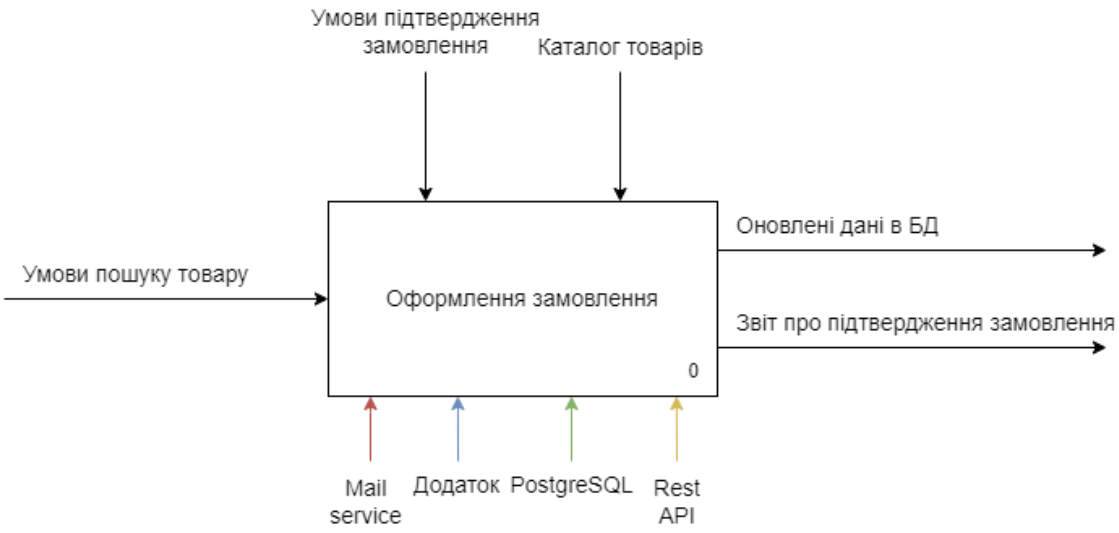


Рисунок 2.1 – Діаграма бізнес-процесів в нотації IDEF0. Конептуальний рівень

Вхідні дані:

– Умови пошуку товару (умови, за якими користувач шукає товар)

Контролюючі та обмежуючі механізми:

– Умови підтвердження замовлення (необхідні умови, при яких менеджер підтверджує замовлення: коректність введених даних, підтвердження замовлення користувачем після дзвінка менеджера)

– Каталог товарів (обмеження у вигляді доступності представленного товару)

Механізми, що використовуються для виконання роботи:

– Mail Service (використовується для відправлення повідомлень користувачам про статуси їхніх замовлень)

– Додаток (серверна частина додатку, яка запускається на вбудованому в Spring Framework сервері)

– PostgreSQL (система управління даними, за допомогою якої відбувається обмін даними між клієнтом та сервером)

– REST API (інтерфейс, що використовується для безпечного обміну інформацію через інтернет)

Вихідні дані:

– Оновлені дані в БД

– Звіт про оформлення замовлення, яке надсилається користувачу

Для отримання більш детального розуміння основних функцій web-додатку була розроблена докладна модель його роботи. Шляхом аналізу вхідних даних було визначено головний потік їх використання у функціональних модулях. Була створена послідовна схема, що ілюструє використання основного призначення web-додатку поетапно. Кожен новий етап випливає з попереднього, утворюючи послідовність кроків для виконання основної мети додатку, і сприяє вирішенню поставленої задачі та виконанню основної функції додатку. У початковій стадії цього процесу додаток отримує вхідні дані, які потім піддаються обробці. На виході додаток повертає

результати цієї обробки. У схемі також відображені зв'язки з механізмами, які використовуються для вирішення поставленої задачі [17].

Декомпозиція функціональної моделі web-додатку представлена на рисунку 2.2.

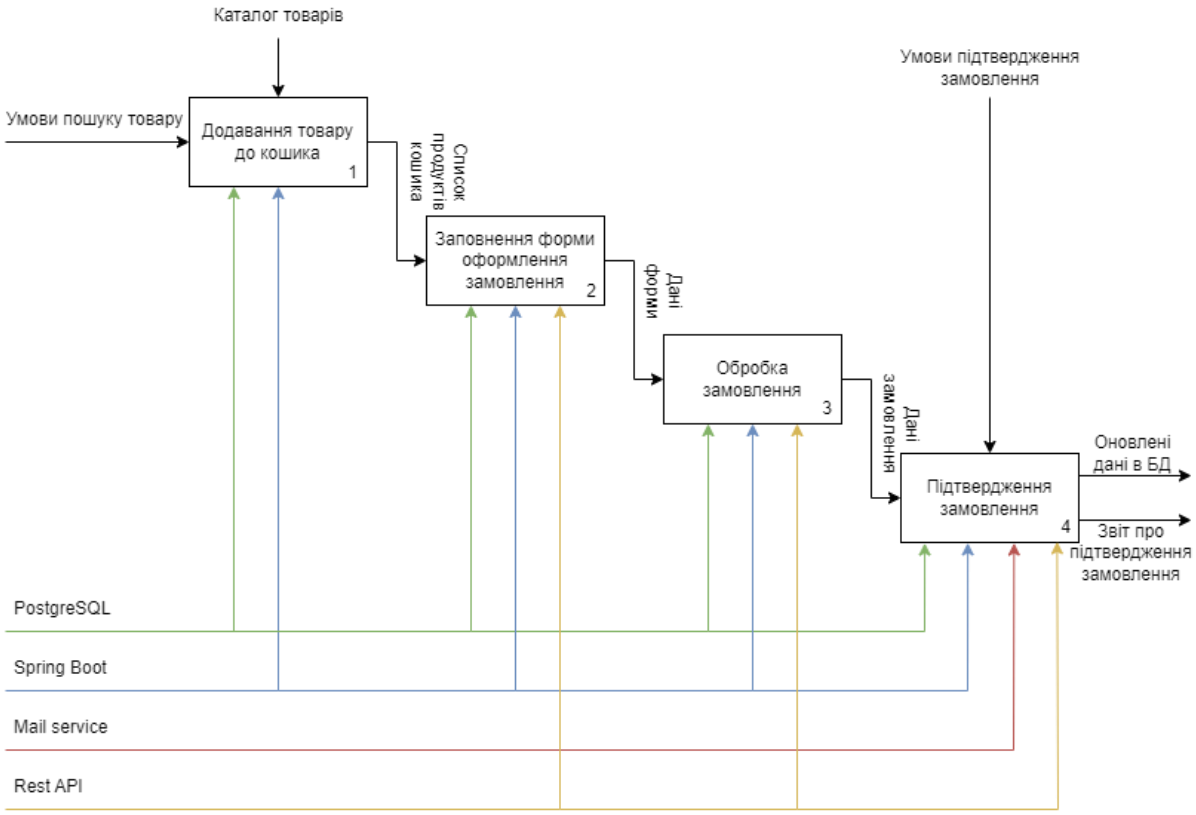


Рисунок 2.2 – Декомпозиція бізнес процесу

2.2 Проектування діаграми варіантів використання

Для досягнення функціональних цілей системи спочатку створюється діаграма варіантів використання (use-case diagram), яка служить простою формою поведінкових діаграм UML. Ця діаграма визначає функціональне призначення системи і має велику користь для наглядного представлення функціональних особливостей програми, особливо для замовників

програмного забезпечення. Вона допомагає описати цілі, які ставляться користувачами, будь то люди або інші програми, що використовують систему. [18].

Діаграма варіантів використання представлена на рисунку 2.3.

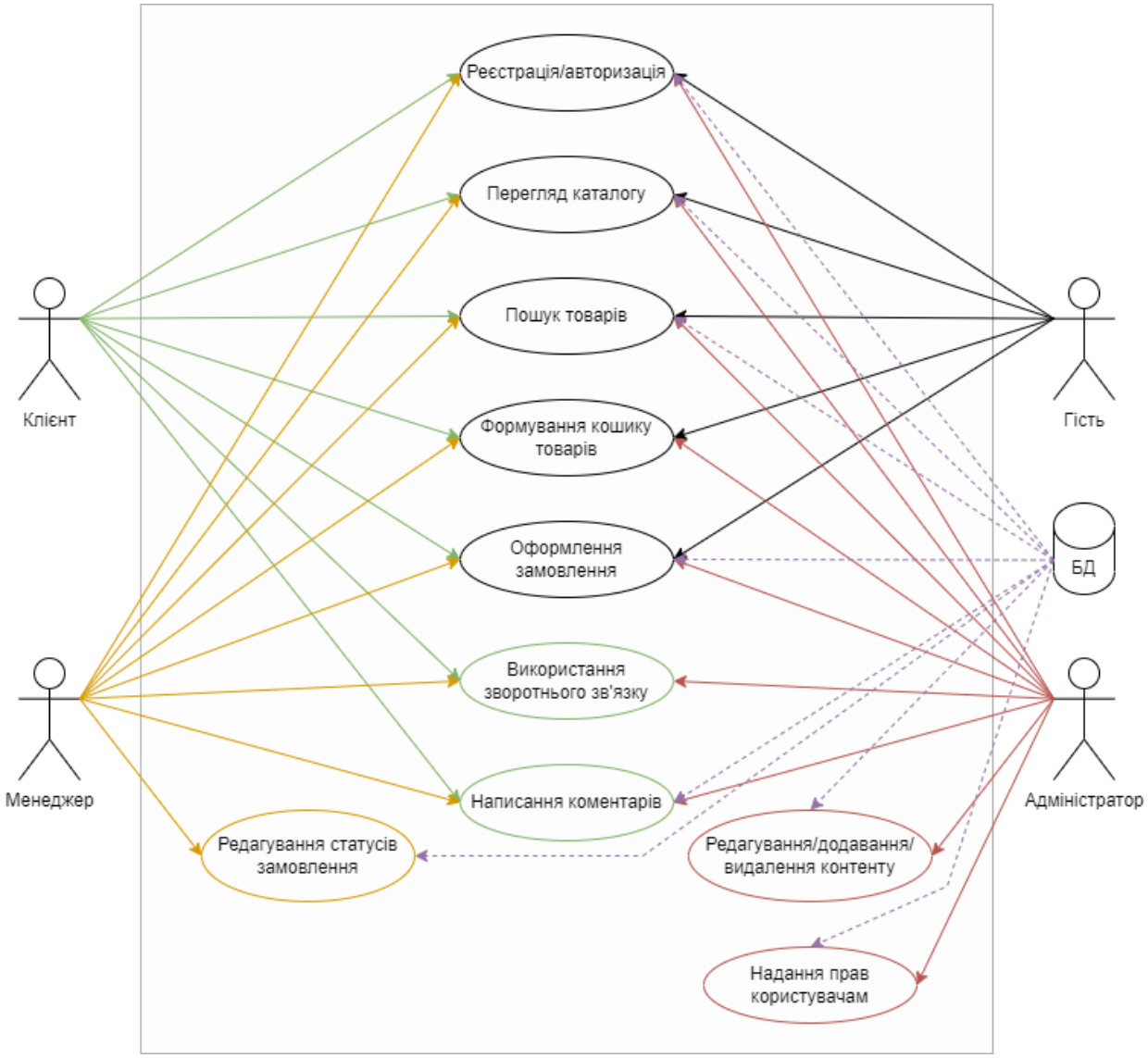


Рисунок 2.3 – Діаграма варіантів використання

На діаграмі наведено 5 акторів:

– Гість. Незареєстрований користувач;
– Клієнт. Зареєстрований користувач;
– Менеджер. Працівник, що обробляє замовлення;

– Адміністратор. Працівник, що контролює продукцію та надає ролі користувачам;

– PostgreSQL. База даних, яка надає можливість зберігати, читати, редагувати, та видаляти дані, а також забезбечує їх цілісність.

2.3 Структурне моделювання web-додатку

Структурне моделювання дозволяє візуалізувати рух та взаємодію між базою даних та додатком, що дозволяє описати ключові операції, які можуть бути виконані у web-додатку за допомогою бази даних. Діаграма послідовності (sequence diagram) - це графічний засіб, що демонструє взаємодію об'єктів у відповідності до їх послідовності у часі. На такій діаграмі основні елементи розташовуються зліва направо: об'єкти, вертикальні лінії, які відображають тривалість виконання дій об'єктом, та стрілки, які показують дії, які виконуються об'єктами [19].

Структурне моделювання web-додатку у вигляді діаграми послідовності представлена на рисунку 2.4.

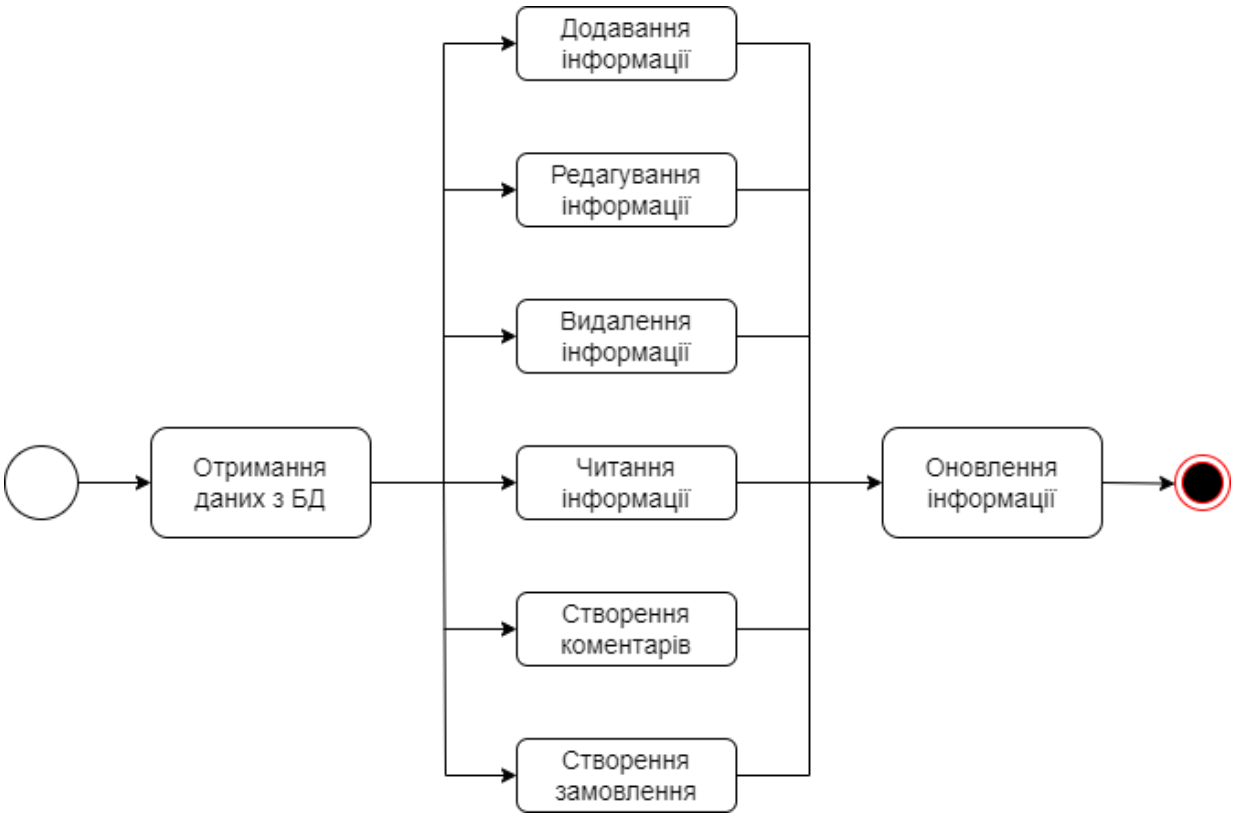


Рисунок 2.4 – Діаграма послідновності web-додатку

Основними елементами діаграми є потік даних та операції, що застосовуються до цих даних. Спочатку здійснюється доступ до інформації з бази даних. Потім з цими даними можна виконувати наступні прості операції:

- додавання інформації, наприклад товару, або реєстрація користувачів;
- редагування інформації про товари або користувачів;
- видалення інформації про товари, категорії;
- читання інформації про товари, коментарі.

До складних операцій можна віднести наступні:

- додавання коментарів для конкретних товарів;
- оформлення замовлення на покупку продукції.

2.4 Проектування діаграми компонентів web-додатку

Для того, щоб реалізувати конкретну фізичну систему, необхідно перетворити всі елементи логічного опису на фізичні компоненти з визначеними матеріалами. Для опису цих реальних елементів використовується фізичне подання моделі.

Діаграми компонентів UML надають концептуальне уявлення про взаємодію між різними системами. Вони можуть бути використані як у логічному, так і у фізичному моделюванні. Компоненти в UML є автономними. Вони є модульними елементами системи, які можуть бути замінені альтернативними компонентами. Вони можуть містити конструкції будь-якою складності та функціонувати незалежно. Взаємодія між компонентами відбувається лише через їхні інтерфейси. Компоненти мають свої власні інтерфейси, але також можуть використовувати операції та служби інших компонентів за допомогою їхніх інтерфейсів. На діаграмах компонентів також відображаються зв'язки та залежності в архітектурі програмного забезпечення [20].

Діаграма компонентів web-додатку представлена на рисунку 2.5.

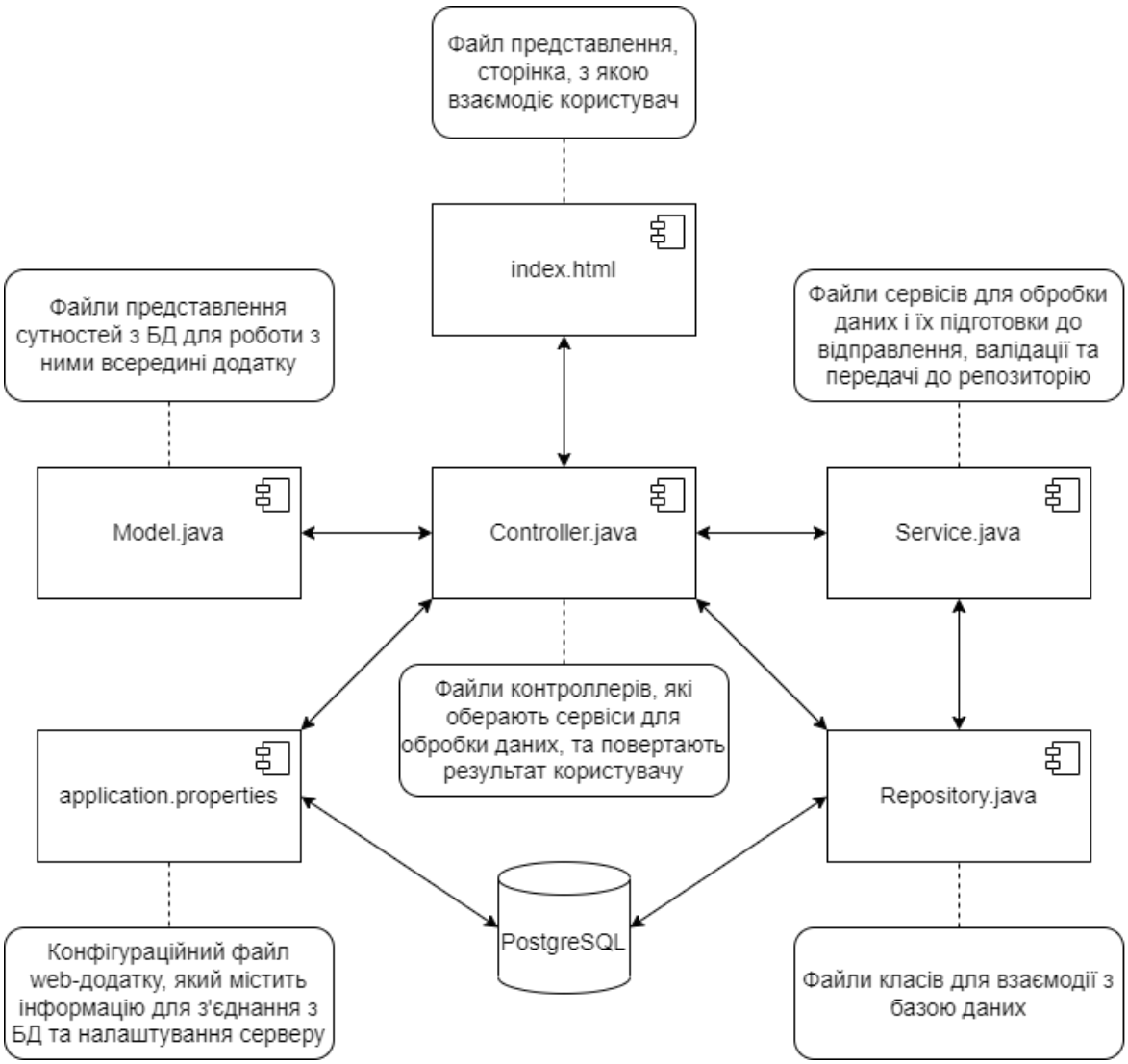


Рисунок 2.5 – Діаграма компонентів web-додатку

2.5 Проектування моделі бази даних

Проектування моделі бази даних є невід'ємною складовою процесу розробки інформаційних систем. Від правильного проектування моделі бази даних залежить якість та продуктивність системи, а також зручність роботи з даними для користувачів. Під час проектування необхідно встановити структуру бази даних, що включає композицію таблиць, їхню організацію та логічні зв'язки. Структура реляційної таблиці визначається за допомогою

компонентів, таких як набір стовпців, їхній порядок, типи даних, розміри стовпців та ключі таблиці [21].

У процесі проектування бази даних визначено наступні сутності:

– Категорії (categories)
– Користувачі (users)
– Товари (products)
– Коментарі (comments)
– Замовлення (orders)
– Статуси замовлень (order_statuses)
– Ролі користувачів (permissions)

Фізичну модель бази даних web-додатку зображено на рисунку 2.6.

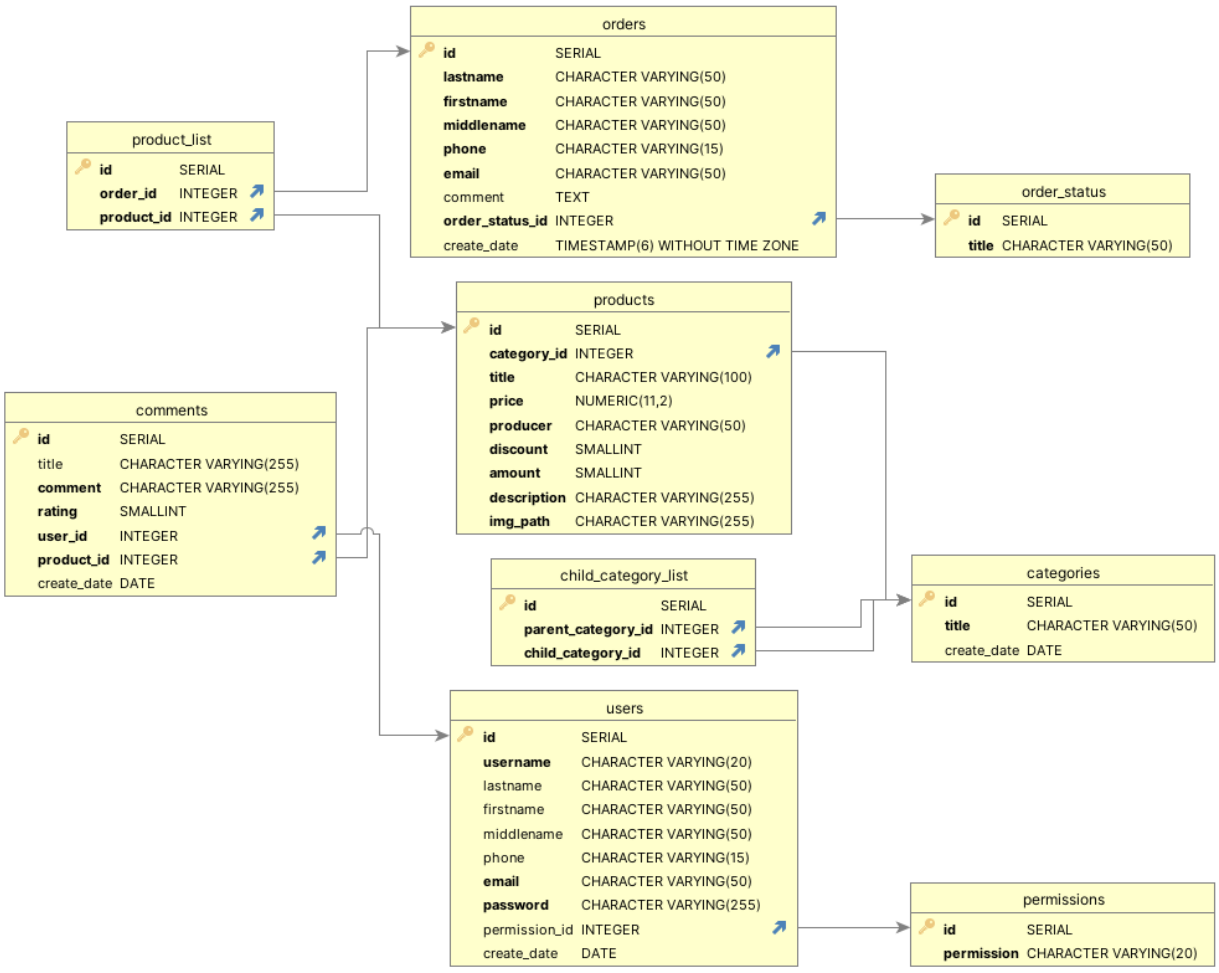


Рисунок 2.6 – Фізична модель бази даних

3 РОЗРОБКА WEB-ДОДАТКУ ПІДТРИМКИ ПРОДАЖУ ЕЛЕКТРОННОЇ ТЕХНІКИ

3.1 Архітектура web-додатку

На етапі початкової розробки web-додатку, була побудована його архітектура на базі патерну MVC (Model-View-Controller). Ця модель відображає взаємодію компонентів у створюваному програмному продукті. Моделі (Model) відповідають за управління даними та їх структурою. Вони забезпечують збереження та організацію даних. Подання (View) відповідальний за відображення цих даних користувачу. Він створює інтерфейс, який дозволяє користувачеві бачити та взаємодіяти з даними. Контролери (Controller) виконують роль керівників і керують компонентами системи. Вони отримують команди у вигляді дій, які виконує користувач, і керують відповідними діями та обробкою даних [22].

Послідовність роботи web-додатку за допомогою MVC можна описати наступним чином: при завантаженні web-додатку, користувач потрапляє на головну сторінку сайту. Якщо користувач вирішив зареєструватися на сайті, він переходить на відповідну сторінку, де заповнює реєстраційну форму. Дані з форми надсилаються на сервер, де відповідний контролер здійснить оброку та валідацію цих даних і, за допомогою сервісу, надішле запит до бази даних, в якій збережеться інформація про користувача. Далі контролер поверне користувачу відповідне повідомлення про успішну реєстрацію на сайті, або ж навпаки, про помилку при валідації даних, їх збереженні тощо.

Архітектура web-додатку підтримки продажу електронної техніки зображена на рисунку 3.1.

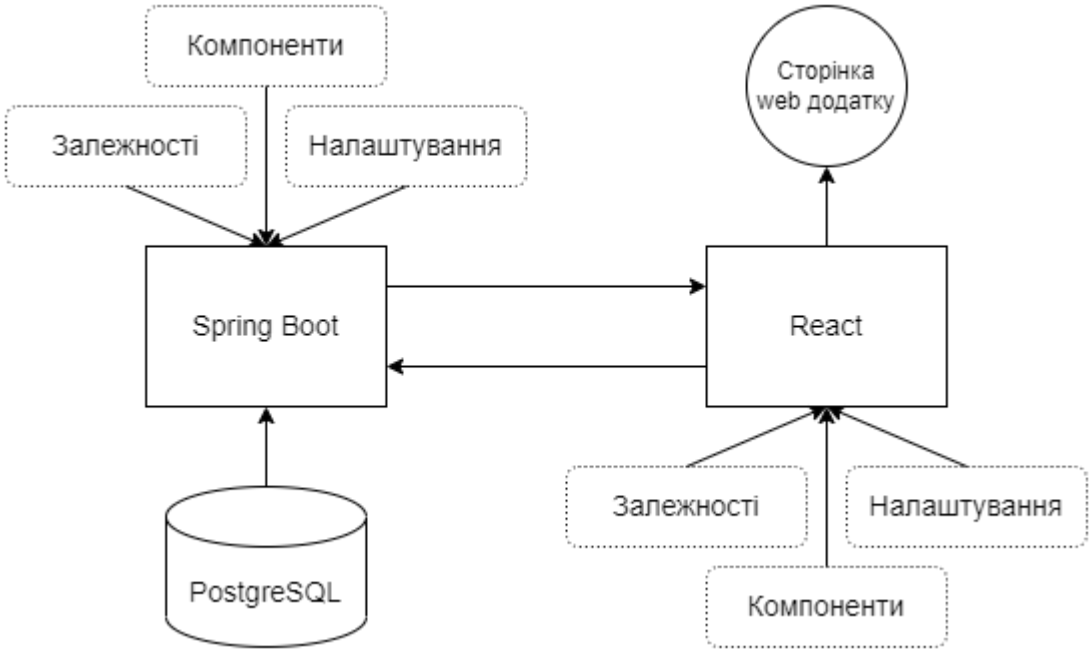


Рисунок 3.1 – Схема архітектури web-додатку

3.2 Створення дизайну web-додатку

У процесі розробки web-додатку передбачається створення макету його структури, що є наступним етапом реалізації проекту. За допомогою програмного забезпечення Figma були розроблені макети сторінок, відповідно до вимог, визначених у додатку А.

Дизайн розроблених сторінок представлений на рисунках 3.2 – 3.7.

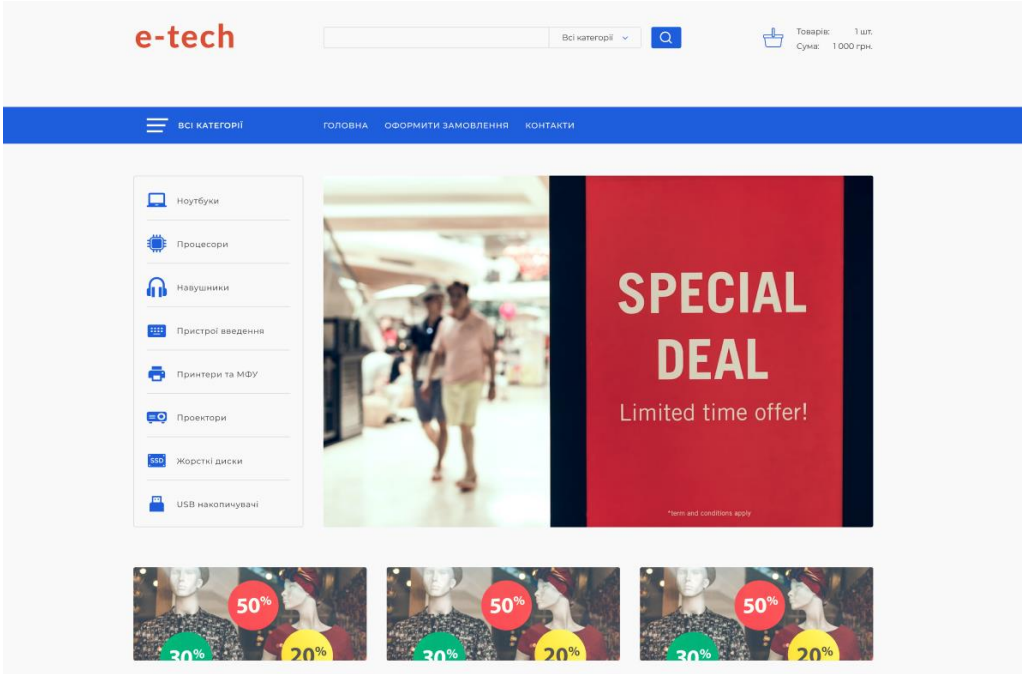


Рисунок 3.2 – Головна сторінка web-додатку

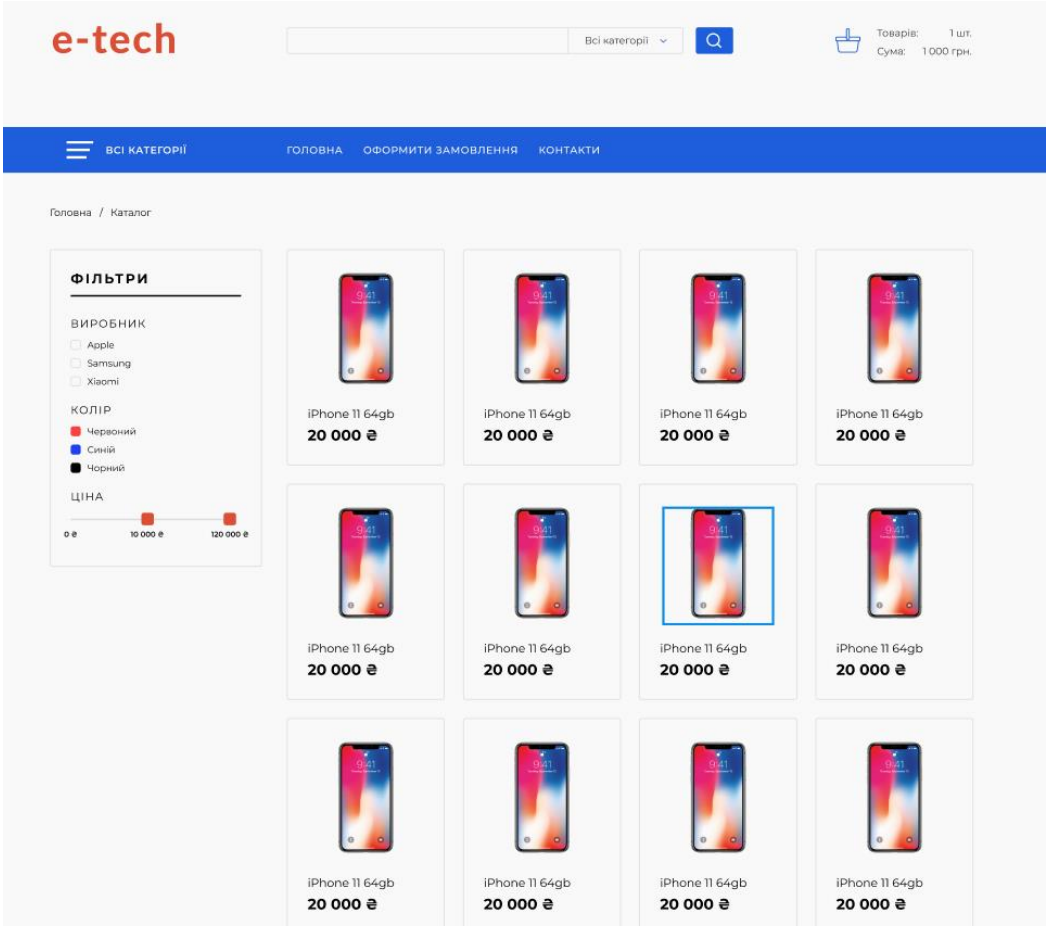


Рисунок 3.3 – Каталог web-додатку

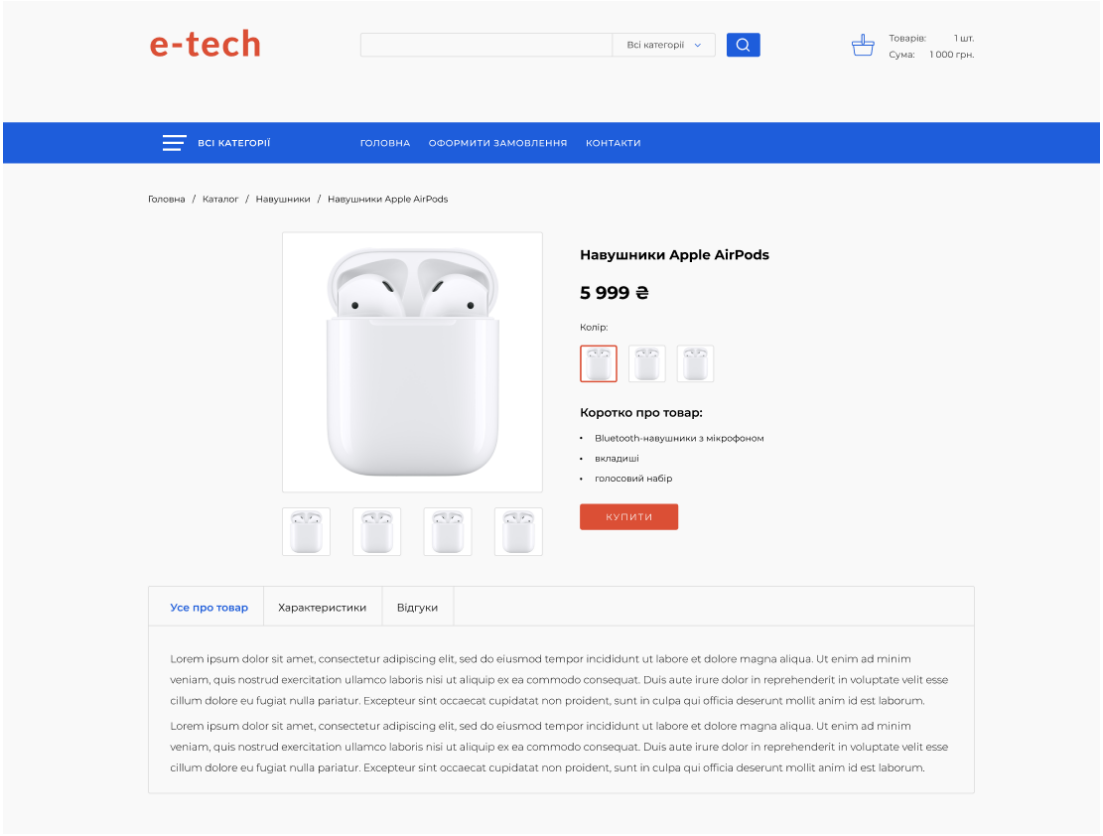


Рисунок 3.4 – Сторінка детальної інформації про товар

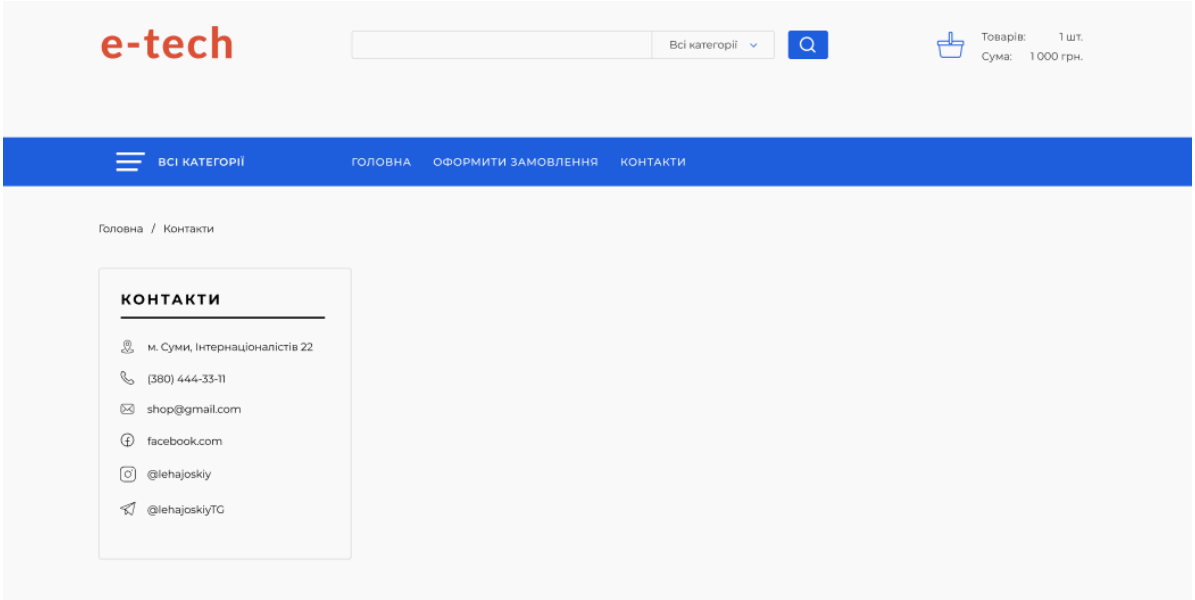


Рисунок 3.5 – Сторінка способів зв’язку з адміністрцією/тех. підтримкою web-додатку

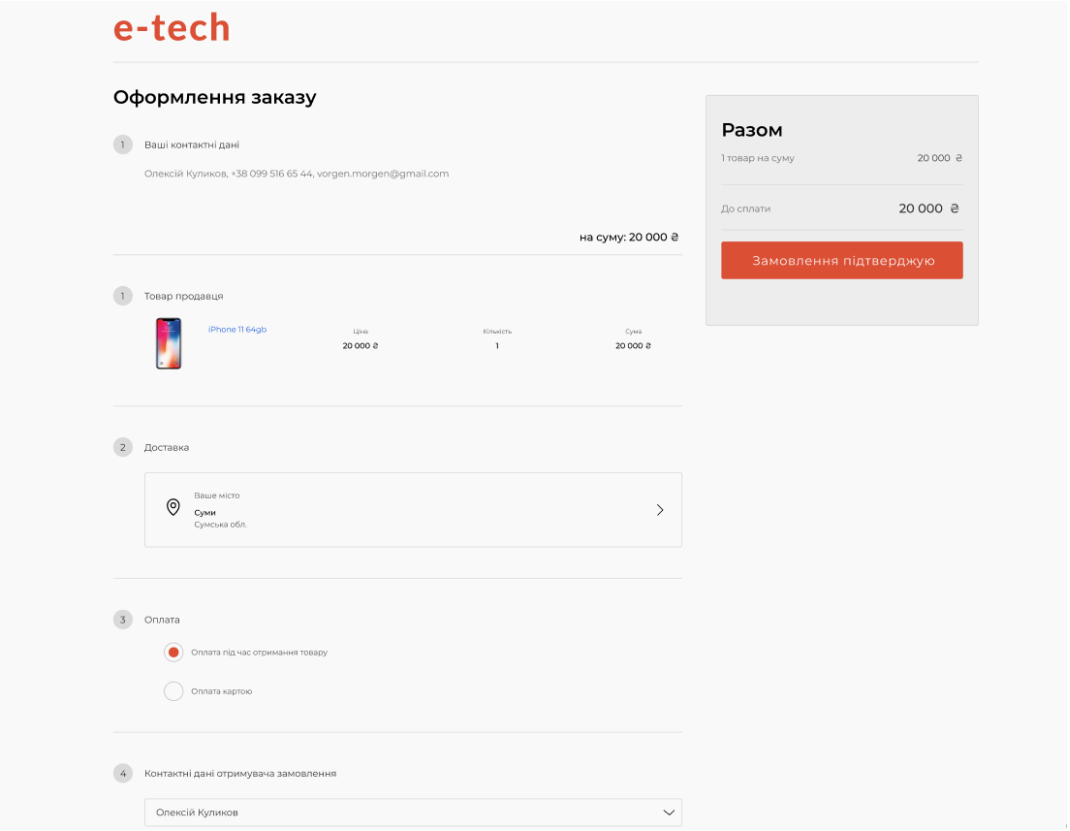


Рисунок 3.6 – Сторінка оформлення замовлення

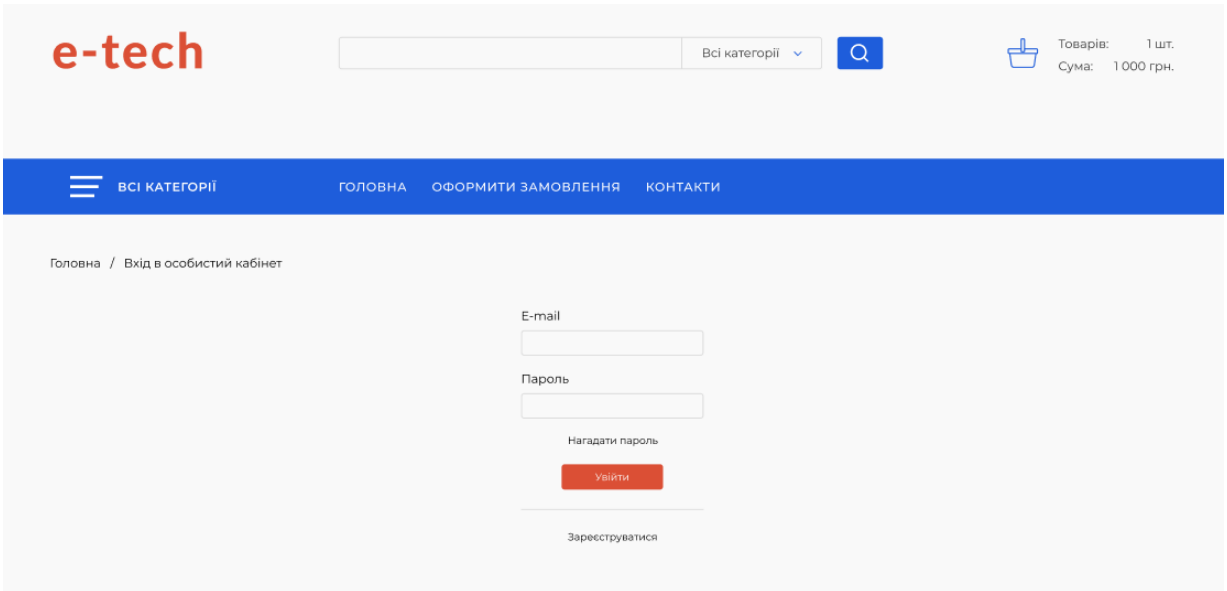


Рисунок 3.7 – Сторінка авторизації у web-додатку

Для оформлення зовнішнього вигляду сторінок web-додатку використано фреймворк Tailwind CSS, який дозволяє швидко стилізувати web-сторінки. Також, застосовані у web-додатку шрифти, було завантажено з web-ресурсу Google Fonts. На кожній сторінці закріплена шапка сайту, на якій розташовані логотип, рядок пошуку, кошик та навігаційна панель web-додатку.

3.3 Програмна реалізація web-додатку

База даних. Розробка бази даних з використанням СУБД PostgreSQL є першим кроком у реалізації web-додатку підтримки продажу електронної техніки. За допомогою утиліти PgAdmin 4 було створено таблиці, відповідні до попередньо спроектованої моделі бази даних (рис. 3.8).

Name	Schema	Owner	Tablespace	Has Indexes	Has Rules	Has Triggers	Is Shared	N Cols	Comment
categories	public	postgres	(null)	☑	☐	☑	☐	3	(null)
child_category_list	public	postgres	(null)	☑	☐	☑	☐	3	(null)
comments	public	postgres	(null)	☑	☐	☑	☐	7	(null)
order_status	public	postgres	(null)	☑	☐	☑	☐	2	(null)
orders	public	postgres	(null)	☑	☐	☑	☐	4	(null)
permissions	public	postgres	(null)	☑	☐	☑	☐	2	(null)
product_list	public	postgres	(null)	☑	☐	☑	☐	3	(null)
products	public	postgres	(null)	☑	☐	☑	☐	9	(null)
users	public	postgres	(null)	☑	☐	☑	☐	6	(null)

Рисунок 3.8 – Список таблиць бази даних web-додатку

Серверна частина. Для розробки серверної частини web-додатку було обрано мову об’єктно орієнтовану мову програмування Java у поєднанні з Spring Boot фреймворком, який спростить розробку і дозволить зосередитися на створенні бізнес-логіки додатку.

Сервіс формується з наступних обов’язкових компонентів:

– шар контролерів;

– шар моделей;
– сервісний шар;
– шар репозиторіїв;
– шар мапінгу;
– конфігурація додатку.

Конфігурація додатку містить статичні дані, які використовуються в різних частинах коду, наприклад для конфігурування, так званих бінів (сервісних компонентів Spring Boot фреймворку), для з'єднання з базою даних тощо. Це дозволяє налаштовувати роботу web-додатку залежно від різних ситуацій. Приклад файлу конфігурації представлено на рисунку 3.9.

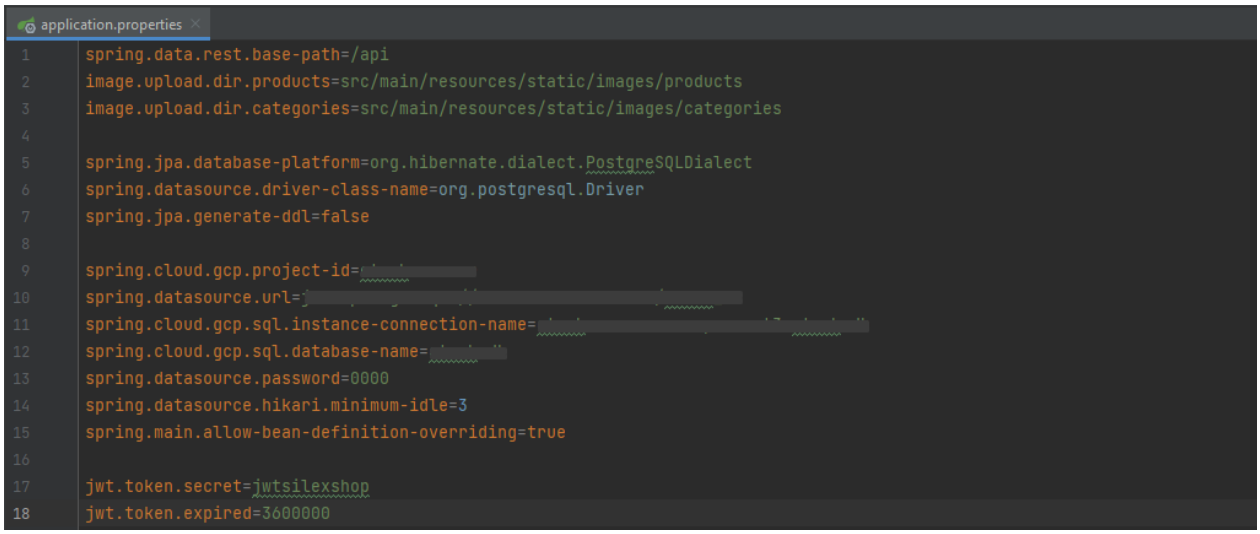


Рисунок 3.9 – Приклад конфігураційного файлу

Шар контролерів відповідає за отримання та передачу запитів для подальшої обробки даних. Коли сервер отримує HTTP запит від клієнта, він містить URI (endpoint), заголовки, параметри та тіло запиту. Контролер з використанням мапінгу визначає, який метод використовувати для обробки запиту, а потім передає його до сервісного шару.

Приклад контролеру, який відповідає за товари представлено на рисунку 3.10.

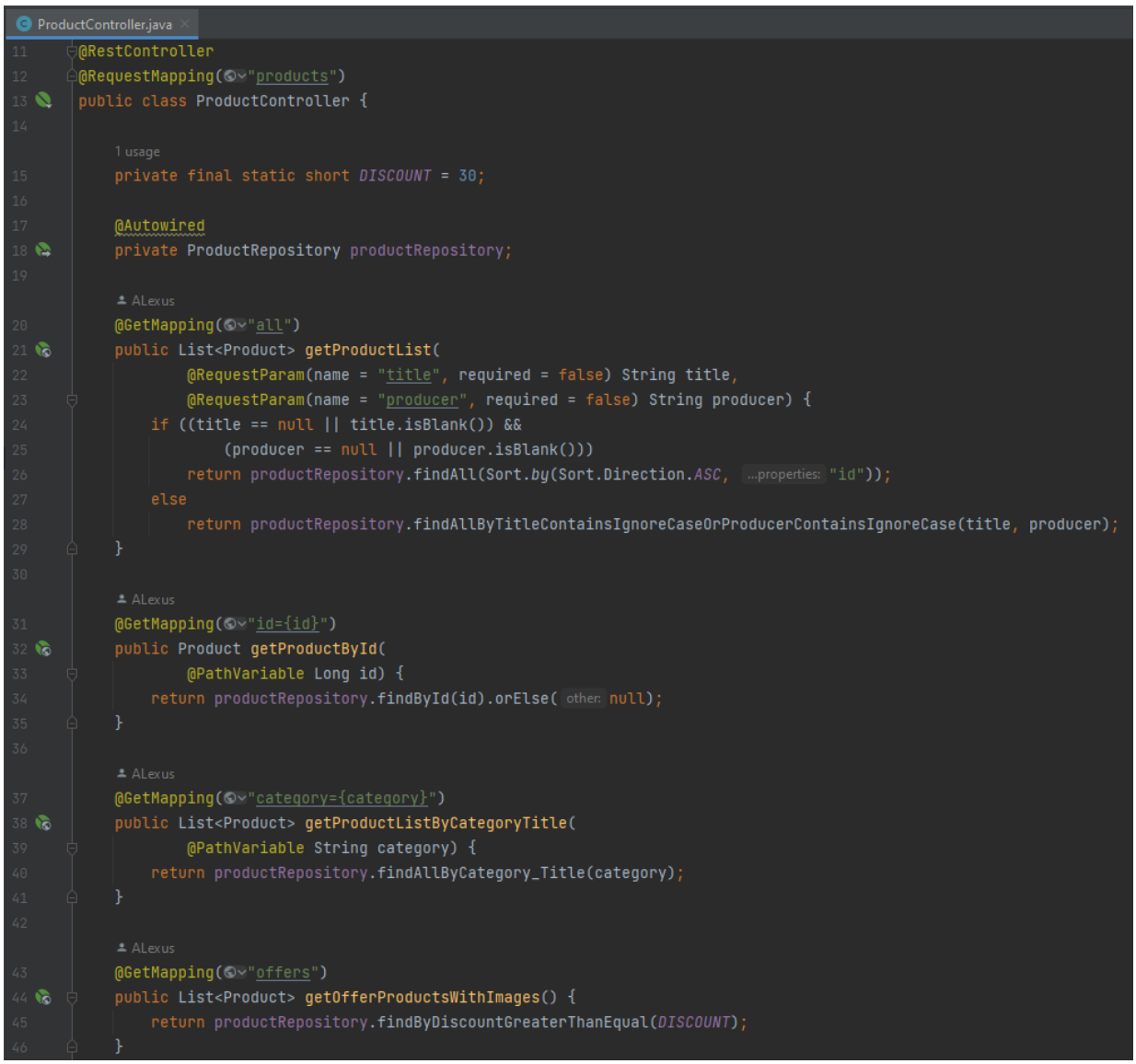


Рисунок 3.10 – Приклад контролеру, що відповідає за CRUD операції з товарами

Сервісний шар відповідає за основну логіку обробки даних. У цьому шарі виконуються такі операції: витягування та перевірка даних, звернення до репозиторію та інших сервісів за додатковою інформацією, виконання правил для обробки даних, фільтрація та трансформація даних, а також виклики для перетворення даних перед їх надсиланням. Після обробки, дані повертаються до контролера, який передає їх клієнту.

Приклад сервісу, що відповідає за авторизацю користувачів, наведений на рисунку 3.11.

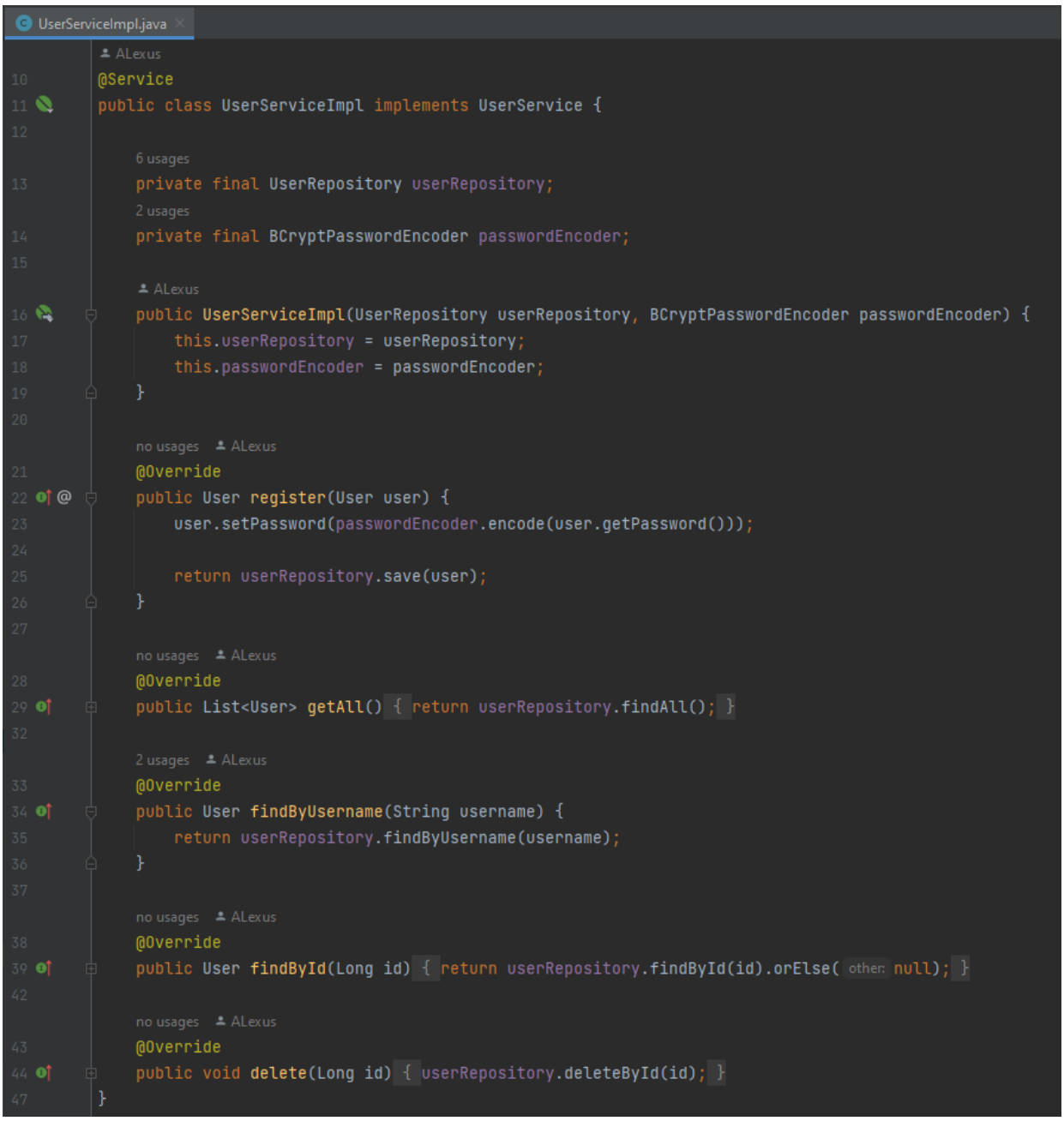


Рисунок 3.11 – Приклад сервісу, що відповідає за авторизацію користувачів

Шар репозиторію взаємодіє з джерелами інформації, такими як база даних. Використання цього шару дозволяє легко заміняти джерело даних без змін в сервісному шарі.

Шар бізнес-моделей представляє об'єкти, які зберігають дані з джерел інформації та піддаються обробці.

Приклад бізнес-моделі, що відповідає за представлення об’єкту замовлення наведено на рисунку 3.12.

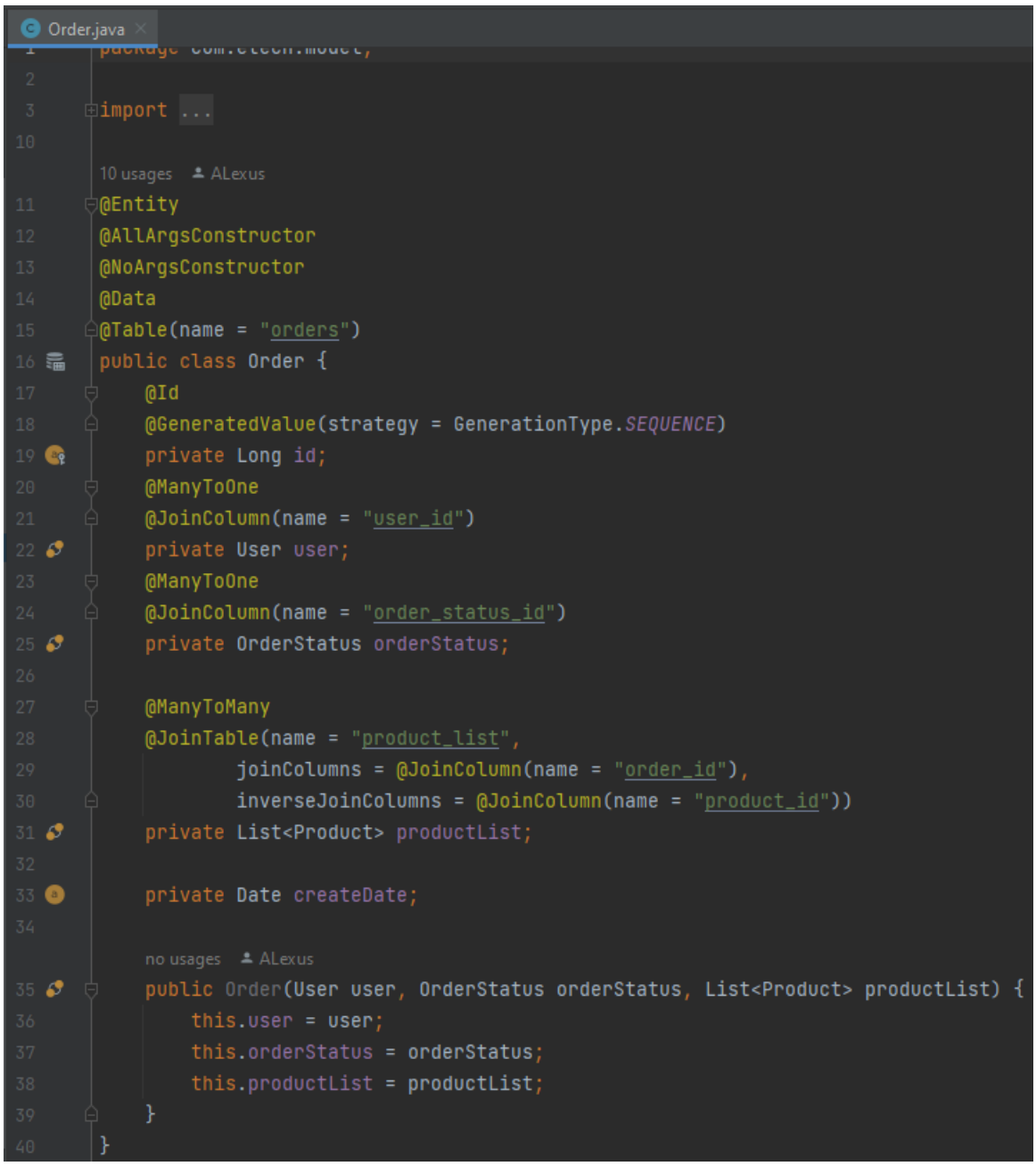


Рисунок 3.12 – Приклад моделі, що відповідає за представлення об’єкту замовлення

Шар мапінгу відповідає за розділення логіки моделей та їх представлень, які передаються клієнту. Під час відправлення запиту до сервера, дані перетворюються на DTO (Data Transfer Object), які містять необроблені дані. Ці дані проходять валідацію на серверній стороні. Після валідації шар мапінгу переносить дані з об'єкта DTO на бізнес-модель. При відправці даних до клієнта також використовуються DTO, що дозволяє додавати нові поля для

клієнта та приховувати непотрібні поля бізнес-моделі. Наприклад, модель списку продуктів не повинна містити додаткову інформацію, яка потрібна лише клієнту, таку як кількість сторінок, номер сторінки, сортування тощо. Це дозволяє забезпечити слабку залежність сервісу від рівня репозиторію та легку заміну репозиторію на іншу реалізацію.

Також важливою складовою додатку є автоматизована збірка, яка використовує популярний інструмент для збірки - Maven. Його основна ідея полягає в описі декларативних команд, які виконують наступні операції:

- Завантаження залежностей та бібліотек з інтернет-репозиторіїв;
- Запуск етапу тестування;
- Компіляція та збірка проекту для подальшого запуску;
- Запуск отриманого додатку;

Клієнтська частина. Фронтенд частина додатку реалізована з використанням фреймворку React [23] і Redux [24]. Основна концепція React полягає у створенні компонентів, які можуть бути використані повторно. Механізми, які впроваджені в цей фреймворк, надають можливості для гнучкої розробки клієнтських додатків. Тому важливо розбити клієнтську частину додатку на окремі компоненти при проектуванні. Головний підхід Redux полягає у створенні локальної бази даних, в якій зберігається виключно інформація конкретного користувача. Це надає можливість формувати, наприклад, кошик користувача, не надсилаючи запит на сервер кожного разу при додаванні нового товару, або зберігати інформацію для авторизації користувача у web-додатку.

Структура web-додатку клієнтської частини виглядає наступним чином:

- файл конфігурації додатку;
- головний компонент;
- картинки;
- сторінки;
- дочірні компоненти сторінок;

– файли конфігурації клієнтського сховища;

– утилітні модулі;

– стилі компонентів

Конфігураційний файл включає в собі список необхідних залежностей (бібліотек) та налаштування для запуску проекту (рис. 3.13).

package.json

```
{
  "name": "etech-client",
  "private": true,
  "version": "0.0.0",
  "type": "module",
  "scripts": {
    "dev": "vite",
    "build": "vite build",
    "preview": "vite preview"
  },
  "dependencies": {
    "@material-tailwind/react": "^2.0.1",
    "@reduxjs/toolkit": "^1.9.5",
    "@trendyol-js/react-carousel": "^3.0.2",
    "axios": "^1.4.0",
    "cookies-js": "^1.2.3",
    "google-fonts": "^1.0.0",
    "react": "^18.2.0",
    "react-cookie": "^4.1.1",
    "react-dom": "^18.2.0",
    "react-icons": "^4.8.0",
    "react-redux": "^8.0.5",
    "react-router-dom": "^6.11.1",
    "react-scroll": "^1.8.9",
    "redux": "^4.2.1",
    "redux-persist": "^6.0.0"
  },
  "devDependencies": {
    "@types/react": "^18.0.28",
    "@types/react-dom": "^18.0.11",
    "@vitejs/plugin-react": "^3.1.0",
    "autoprefixer": "^10.4.14",
    "postcss": "^8.4.22",
    "tailwindcss": "^3.3.1",
    "vite": "^4.2.0"
  }
}
```

Рисунок 3.13 – Приклад конфігураційного файлу

Головний компонент web-додатку, відповідаючи аналогії головної сторінки на сайті, містить усі елементи, які використовуються для навігації по сайту, це називається роутинг (рис. 3.14).

```
function App () {
  return (
    <>
      <Routes>
        <Route path='/' element={<Layout />}>
            <Route index element={ <HomePage /> } />
            <Route path='/products/category/:category' element={<ProductList />} />
            <Route path='/products/id/:id/' element={<ProductDetails />} />
            <Route path='/products' element={<ProductList />} />
            <Route path='/cabinet' element={ <CabinetPage /> } />
            <Route path='/contact' element={ <ContactPage /> } />
            <Route path='/error' element={<ErrorPage />} />
            <Route path='/authentication' element={<AuthPage/>} />
            <Route path='/admin' element={<AdminPage/>} />
        </Route>
      </Routes>
    </>
  );
}

2 usages  skinner.x8
export default App;
```

Рисунок 3.14 – Приклад головного компоненту

Сторінки описують в собі сторінки web-додатку, які будуть відображатися при навігації, яка згадується вище, але які не мають дочірніх компонентів для відображення (рис 3.15).

AuthPage.jsx

```jsx
export function AuthPage() {
    const dispatch : Dispatch<AnyAction>  = useDispatch();
    const navigate : NavigateFunction  = useNavigate();

    1 usage  ALexus
    async function handleSubmit(e) : Promise<void>  {
        e.preventDefault()
        let username = e.target.username.value;
        let password = e.target.password.value;

        await axios.post( url: local + '/auth/login',  data: {username, password})
            .then(response : AxiosResponse<any>  => {
                dispatch(saveToken(response.data.token, response.data.username, response.data.role));
                navigate('/');
            })
            .catch(e => {
                console.log(e.message)
            })
    }

    return (
        <div className='w-[1110px] mx-auto' onSubmit={handleSubmit}>
            <form className='w-min mx-auto my-20'>
                <label htmlFor="username" className='text-md block'>Логін</label>
                <input type="text" name='username' id='username' placeholder='Логін'
                       className='border border-black/20 rounded-[3px] py-1 px-3'
                       required/>

                <label htmlFor="password" className='text-md block mt-3'>Пароль</label>
                <input type="password" name='password' id='password' placeholder='Пароль'
                       className='border border-black/20 rounded-[3px] py-1 px-3'
                       required/>

                <button className='button blue-button block mx-auto w-min mt-3'>
                    Увійти
                </button>
            </form>
        </div>
    )
}
```

Рисунок 3.15 – Приклад сторінки авторизації

Дочірні компоненти сторінок містять в собі логіку взаємодії з web-додатком. Вони поділяються на пакети, в яких кожна папка відповідає за компонент, що буде відображатися користувачу (рис 3.16).

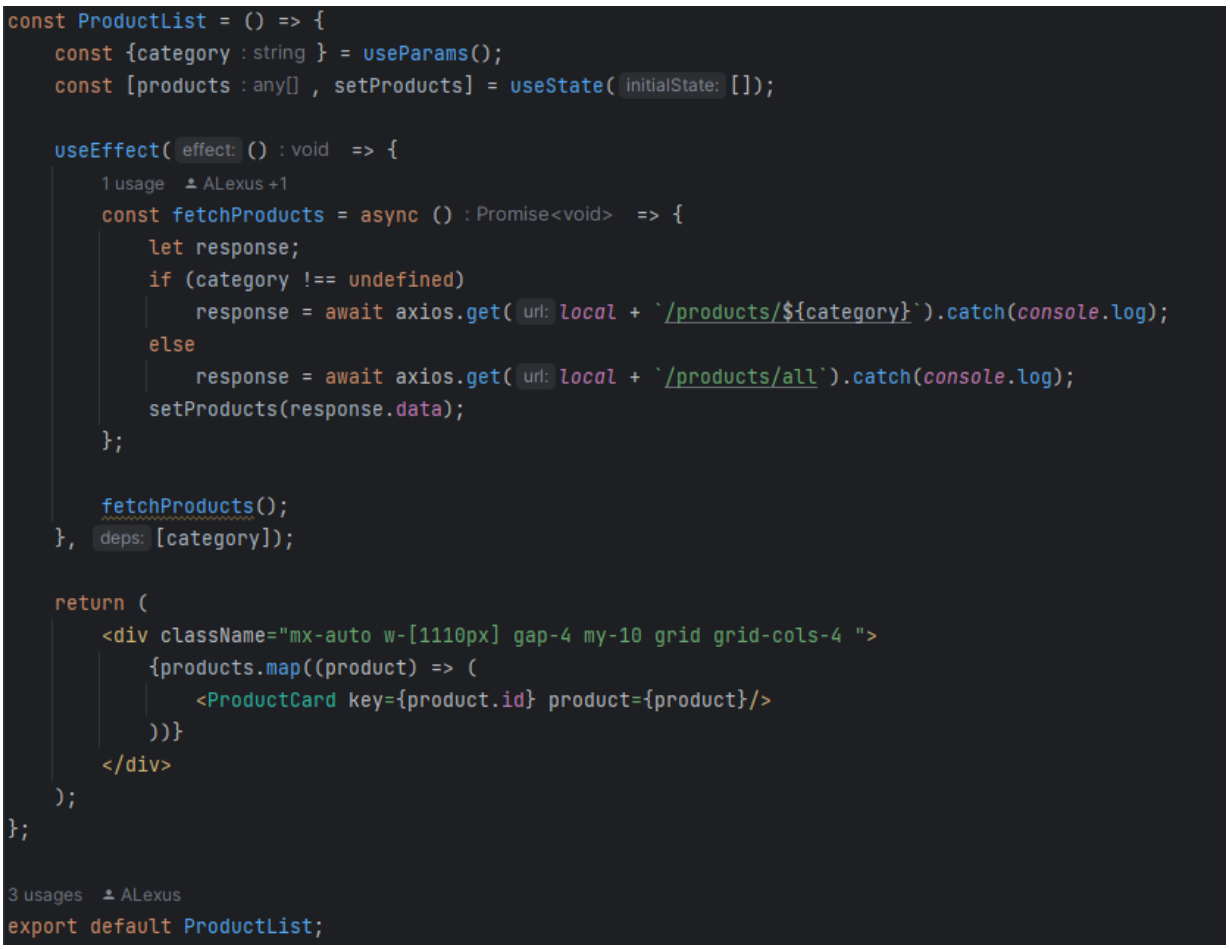


Рисунок 3.16 – Приклад дочірнього компонента, що відображає список продуктів

Файли конфігурації локального сховища клієнта визначають логіку зберігання даних на клієнтському пристрої (рис. 3.17). Наприклад, кошик використовує таке сховище для збереження даних. Це зроблено з метою зменшення навантаження на сервер. Тепер, замість того, щоб надсилати запит на сервер кожного разу, коли користувач додає товар до кошика, дані зберігаються в локальному сховищі. При оформленні замовлення, список товарів з кошика передається до бази даних одним запитом.

Рисунок 3.17 – Приклад файлу конфігурації локального сховища (кошик)

Утильні модулі містять скрипти, які можна використовувати повторно в різних компонентах. Наприклад, такі модулі можуть включати функції для збереження або завантаження даних з кукі. Головна мета цього підходу - уникнути дублювання коду.

Стилі компонентів містять у собі перелік класів, за допомогою яких здійснюється стилізація компонентів web-додатку.

3.4 Використання web-додатку

Перша сторінка, на яку потрапляє користувач при завантаженні додатку – головна сторінка (рис. 3.18).

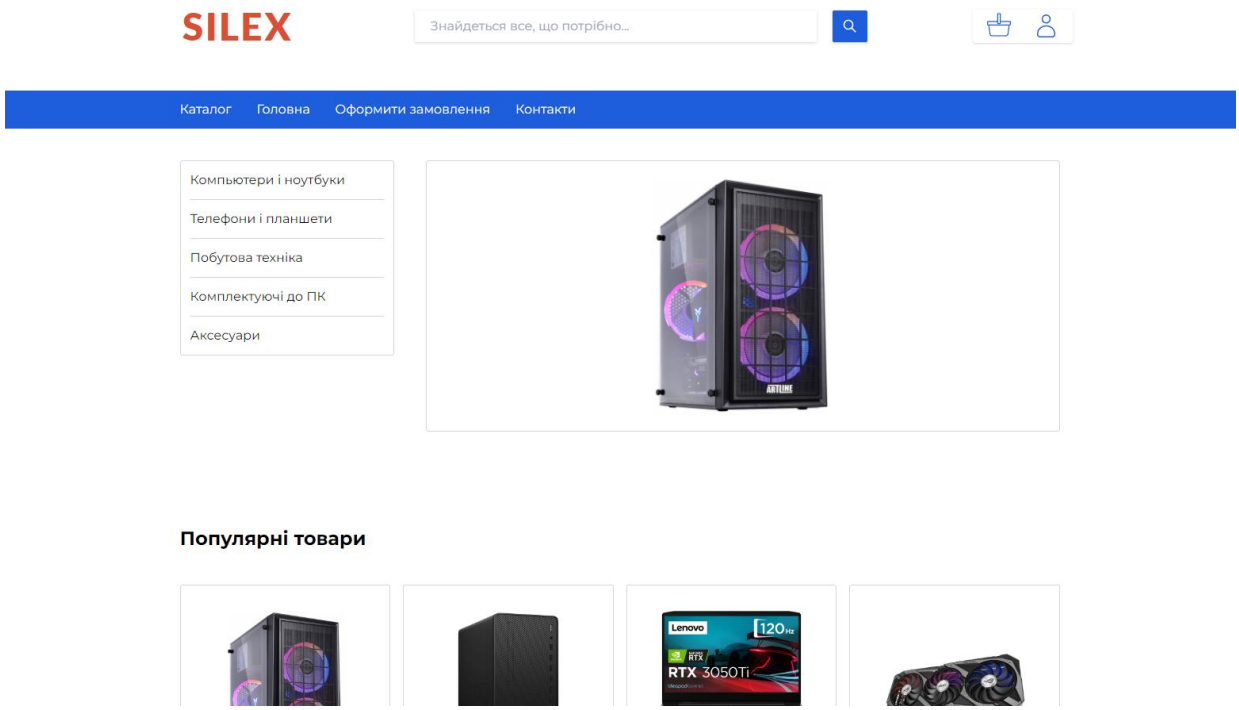


Рисунок 3.18 – Головна сторінка web-додатку

Далі, користувач може авторизуватися в додатку, за допомогою форми авторизації (рис. 3.19), або ж одразу перейти до вибору товару за допомогою вибору відповідної категорії категорії, пошукового рядку (рис. 3.20) або натиснувши на посилання “Каталог” (рис. 3.21).

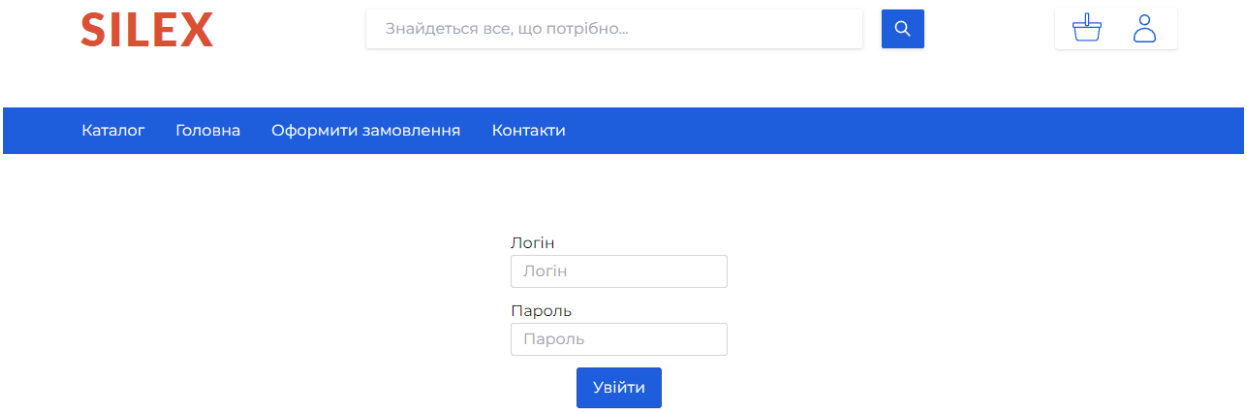


Рисунок 3.19 – Сторінка авторизації у web-додатку

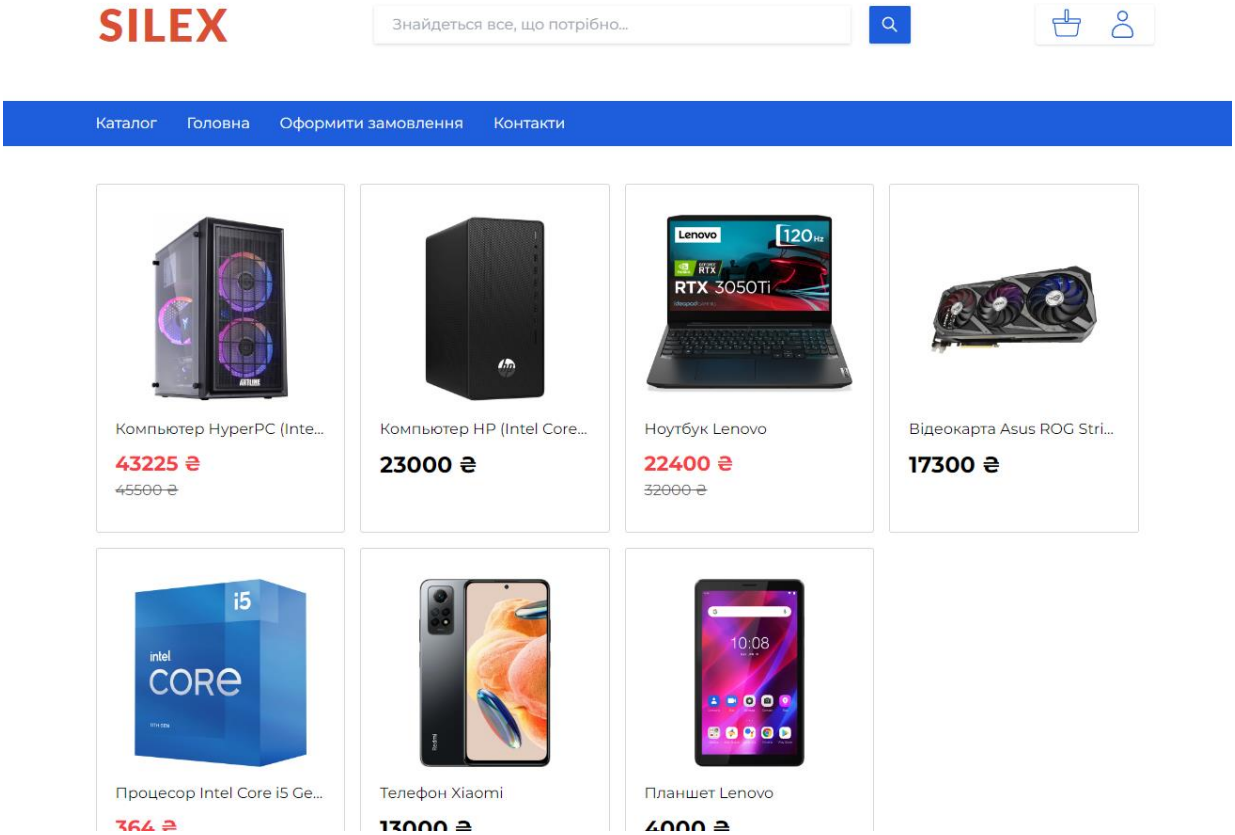


Рисунок 3.21 – Каталог web-додатку

Після того, як користувач обрав необхідні товари, вони будуть відображатися в кошику (рис. 3.22) і на сторінці оформлення замовлення (рис. 3.23)

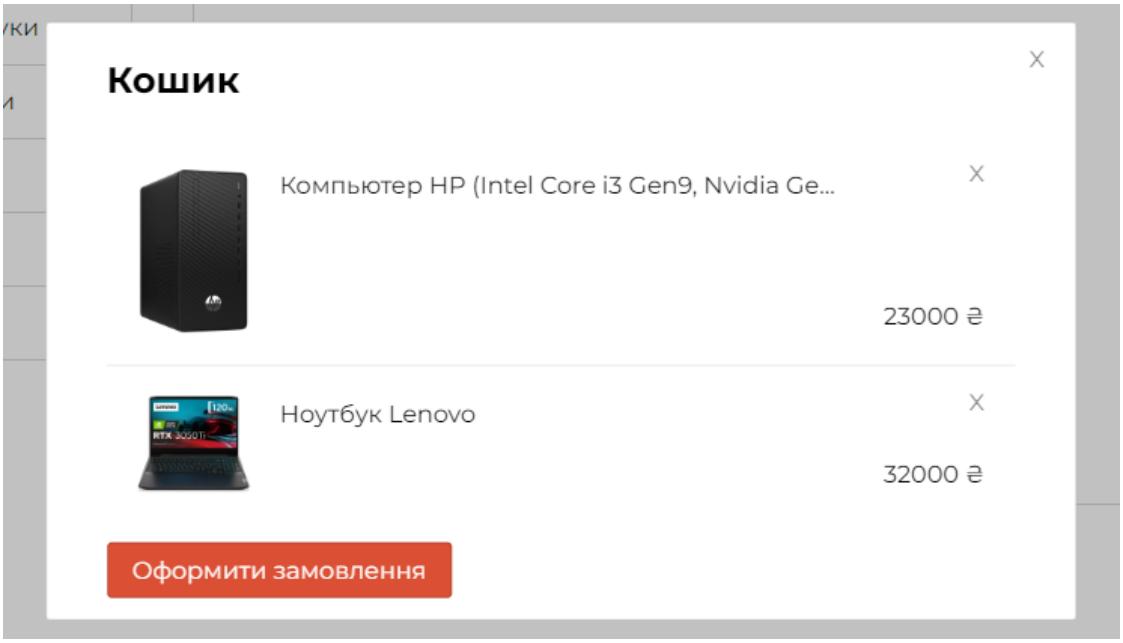


Рисунок 3.22 – Модальне вікно кошику

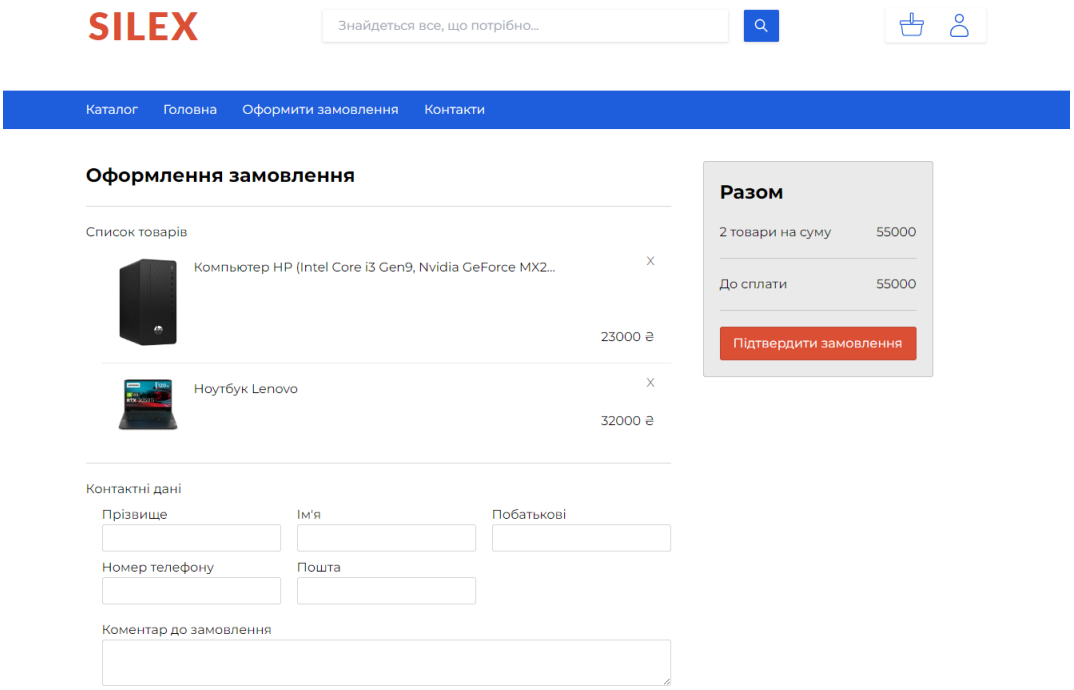


Рисунок 3.23 – Сторінка оформлення замовлення

Після заповнення всієї необхідної інформації, користувач підтверджує замовлення і отримує повідомлення про успішне оформлення замовлення (рис. 3.24), чи навпаки, про помилку (рис. 3.25).

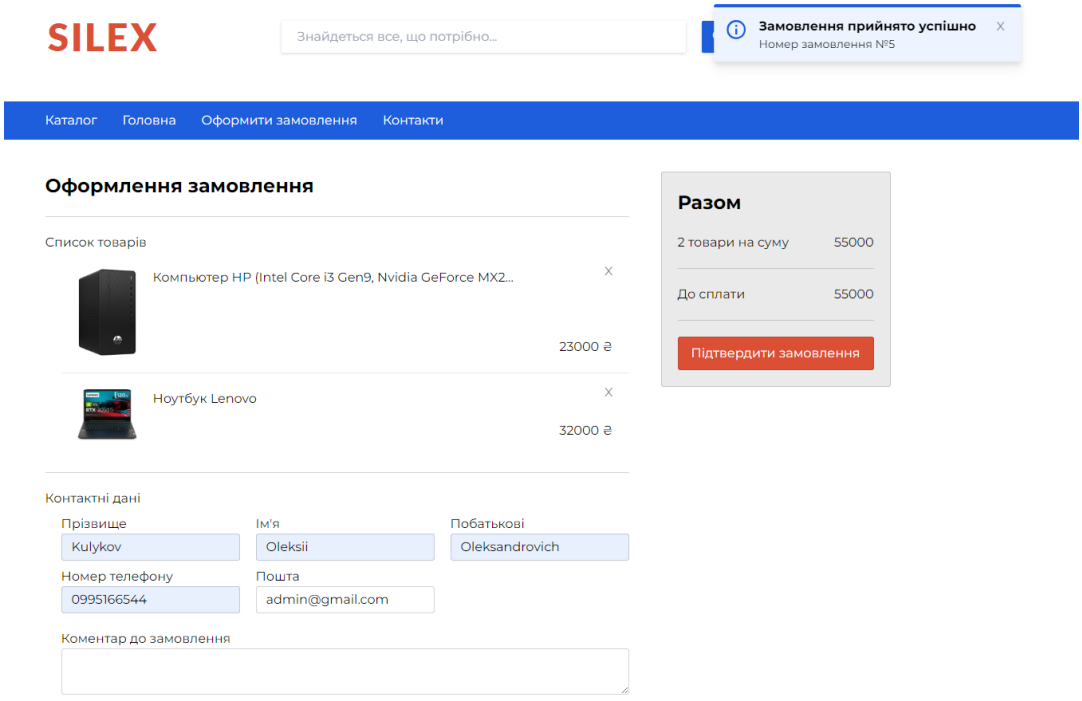


Рисунок 3.24 – Повідомлення про успішне оформлення замовлення

В ситуації, при якій підтвердження замовлення неможливе, користувач також отримає відповідне повідомлення (рис. 3.25).

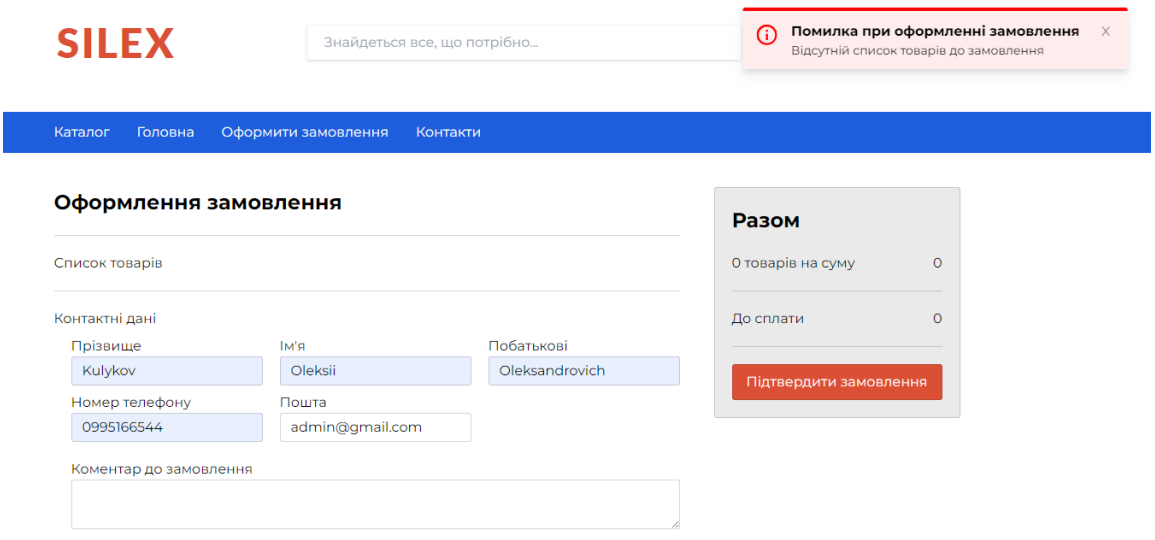


Рисунок 3.25 – Повідомлення про помилку оформлення замовлення

Розроблений функціонал в повному обсязі відповідає технічному завданню.

ВИСНОВКИ

Результатом виконання кваліфікаційонї роботи бакалавра є web-додаток підтримки продажу елктронної техніки, який є надійним, інтерактивним та з сучаним дизайном. Web-додаток було розміщено на хостингу Google Cloud Platform для серверної частини, і Netlify для клієнтської частини додатку.

У ході дослідження предметної області було визначено актуальність автоматизації бізнес-процесів функціонування додатку, представлено результати аналізу предметної області та огляду існуючих аналогів, в результаті чого сформовано специфікацію функціональних вимог до додатку.

Сформульовано мету роботи, обрано інструменти і технології для реалізації програмного продукту у вигляді web-додатку. Метою дослідження є розробка інноваційних рішень, які допоможуть спростити процес вибору та покупки товарів для користувачів. Основними цілями дослідження є покращення взаємодії між користувачем і продуктом, зменшення інформаційного перевантаження, підвищення рівня задоволеності користувачів під час процесу придбання товару, а також збільшення ефективності продажів для бізнесу.

Результати моделювання бізнес-процесів web-додатку, які будуть автоматизовані в розроблюваному додатку, представлені у формі діаграм бізнес-процесів на контекстному та першому рівні декомпозиції, варіантів використання, діаграми послідовності і компонентів.

Web-додаток було розроблено за допомогою мови програмування Java і фреймворку Spring Boot для серверної частини, і мови програмування Java Script та фреймворку React з вкористанням Redux для клієнтської частини. Також, для зберігання інформації web-додатку, була створена база даних в СУБД Postgresql, сутності якої описані у проектній частині КРБ. Результати її моделювання представлені у вигляді логічної моделі бази даних. Для забезпечення пітримки цілісноті даних були реалізовані тригерні функції.

Результати дослідження були апробовані на конференції ІМА 2023. Опубліковані тези доповіді наведено в додатку В.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інформаційні технології в повсякденному житті URL: https://studfile.net/preview/7383905/page:3/ (Дата звернення: 16.01.23)
2. Е. Ю. Терещенко. Розвиток інтернет-торгівлі в умовах сучасного бізнес-середовища URL: http://www.economy.nayka.com.ua/?op=1&z=7381 (Дата звернення: 16.01.23)
3. Р.Л. Лупак, Т.Г. Васильців. Конкурентоспроможність підприємства С.108 URL: http://www.lute.lviv.ua/fileadmin/www.lac.lviv.ua/data/kafedry/Ekonomiky/Docs/Konkurentospromozhnist_pva_Lupak__Vasilciv.pdf (Дата звертання: 16.01.23)
4. Переваги інтернет-магазину над звичайною торговою точкою URL: https://webdevandseo.com.ua/online-store-benefits/ (Дата звернення: 16.01.23)
5. The psychology of Web design: How colors, typefaces and spacing affect your mood URL: https://thenextweb.com/news/psychology-web-design (Дата звернення: 17.01.23)
6. How Website Speed Can Impact Your Business URL: https://pagecrafter.com/how-website-speed-can-impact-your-business/ (Дата звернення: 17.01.23)
7. E-Commerce Security Challenges: A Review URL: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3595304 (Дата звернення: 17.01.23)
8. Customer Analytics: Where the Answers to All of Your Ecommerce Marketing Questions Live URL: https://www.bigcommerce.com/blog/ecommerce-customer-analytics/ (Дата звернення: 17.01.23)
9. Іванина Роман, Клюєв Ерік. Комерційні фактори ранжування інтернет-магазину URL: https://elit-web.ua/ua/blog/kommercheskie-faktory-ranzhirovanija (Дата звернення: 17.01.23)

10. Каркунова Катерина. Бізнес-план для інтернет-магазину URL: https://horoshop.ua/ua/blog/business-plan-for-an-online-store/ (Дата звернення: 18.01.23)

11. Інтернет-магазин “Compservice” URL: https://compservice.in.ua (Дата звертання: 20.01.23)

12. Інтернет-магазин “Eldorado” URL: https://eldorado.ua/uk/ (Дата звертання: 20.01.23)

13. Інтернет-магазин “Bicom” URL: https://bicom.net.ua (Дата звертання: 20.01.23)

14. Spring Framework: Overview URL: https://spring.io/projects/spring-framework (Дата звертання: 15.02.23)

15. PostgreSQL Documentation URL: https://www.postgresql.org/docs/current/index.html (Дата звертання: 26.01.23)

16. Вікіпедія. IDEF0 URL: https://uk.wikipedia.org/wiki/IDEF0 (Дата звертання: 01.04.23)

17. What is Use Case Diagram? URL: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/ (Дата звертання: 01.04.23)

18. Застосування uml в дипломних роботах. URL: https://dut.edu.ua/ua/news-1-626-7758-zastosuvannya-uml-v-diplomnih-robotah_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy (Дата звертання: 02.04.23)

19. Марченко Анна. Проектування інформаційних систем. URL: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2a161439391120eef364dd8ce4d8e5e/20160217112601/content-20160217112601.pdf (Дата звертання: 05.04.23)

20. Підручник зі створення діаграми компонентів UML. Частина 3. URL: https://www.mindonmap.com/uk/blog/uml-component-diagram/#part3 (Дата звертання: 10.04.23)

21. Проектування баз даних. URL: https://pidru4niki.com/11570718/bankivska_sprava/proektuvannya_baz_danih (Дата звертання: 05.04.23)

22. Вікіпедія. Модель-вид-контролер. URL: https://uk.wikipedia.org/wiki/Модель-вид-контролер (Дата звертання: 15.02.23)

23. React Learn URL: https://react.dev/learn (Дата звернення: 30.01.23)

24. Redux Tutorial URL: https://redux.js.org/tutorials/essentials/part-1-overview-concepts (Дата звернення: 15.02.23)

25. Що таке SMART-цілі? URL: https://experience.dropbox.com/uk-ua/resources/smart-goals (Дата звертання: 01.03.23)

26. Колпакова Ольга. Як сплануєш, так і проведеш. Оцінюємо проєкт з максимальною точністю URL: https://dou.ua/lenta/articles/evaluating-project-with-maximum-accuracy/ (Дата звертання: 03.03.23)

27. Л.Є. Довгань, Г.А. Мохонько, І.П. Малик. Управління проектами. С.36 URL: https://ela.kpi.ua/jspui/bitstream/123456789/19481/1/DMM_UP_2017.pdf (Дата звертання: 14.03.23)

28. Вікіпедія. Діаграма Ганта URL: https://uk.wikipedia.org/wiki/Діаграма_Ґанта (Дата звернення: 15.03.23)

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Web-додаток підтримки продажу електронної техніки»

ПОГОДЖЕНО:

Доцент кафедри комп’ютерних наук

_____ Марченко А.В.

Студент групи ІТ-91

_____ Куликов О.О.

Суми 2023

1 ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ WEB ДОДАТКУ

Призначення web додатку

Додаток призначений для легкого і швидкого оформлення замовлень товарів електронної техніки відповідно до уподобань користувача та різних налаштувань пошуку товару (ціна, категорія, характеристики).

Мета створення web додатку

Головною метою створення web додатку є забезпечення автоматизованої, гнучкої та легкої у використанні системи онлайн магазину, що допоможе користувачам обрати потрібний товар за короткий проміжок часу та з мінімальною кількістю необхідних дій.

Цільова аудиторія

Цільовою аудиторією даного проекту є користувачі, які зацікавлені у самостійному виборі якісного товару електронної техніки. Це доволі обширна аудиторія, так як електронна техніка є необхідністю майже для кожної людини, як у побуті, так і в роботі.

2 ВИМОГИ ДО WEB ДОДАТКУ

2.1 Вимоги до web додатку в цілому

2.1.1 Вимоги до структури й функціонування web додатку

Web додаток повинен бути реалізований за допомогою web-інструментів та забезпечувати визначений набір функціональних можливостей. Кінцевий продукт даного проекту має бути представлений web додатком, який містить високоякісний контент та графічні матеріали.

Розроблюваний web-додаток онлайн магазину елетронної техніки повинен бути завершений в рамках встановлених термінів та обмежень ресурсів. Сайт повинен мати адаптивний інтерфейс, який забезпечить належне функціонування магазину та автоматизацію основних бізнес-процесів.

2.1.2 Вимоги до персоналу

Співробітники магазину не повинні мати особливих технічних навичок для роботи з web-додатком (єдиною вимогою є наявність навичок користування ПК та браузером). Виключенням є технічні розробники, які працюють над розвитком та підтримкою web додатку. Такі працівники мають бути кваліфікованими спеціалістами, оскільки рівень кваліфікації впливає на результативність роботи над проектом.

2.1.3 Вимоги до збереження інформації

Уся інформація надана у web додатку повинна зберігатися у базі даних реалізованій засобами системи управління базами даних PostgreSQL.

2.1.4 Вимоги до розмежування доступу

Розроблюваний web-додаток має бути загальнодоступним у мережі Інтернет. Права доступу до інформації розмежовані за групами користувачів: адміністратор, технічна підтримка, авторизований користувач та неавторизований користувач.

Адміністратор має необмежений доступ до даних з правами перегляду, додавання, редагування та видалення інформації. Доступ до адміністративної панелі надається за наявністю відповідної ролі у користувача.

Неавторизований користувач web додатку може переглядати інформацію на головній сторінці web додатку, де викладена коротка інформація про онлайн магазин, каталог товарів, а також оформити швидке замовлення.

У авторизованого користувача web-додатку спектр доступу до інформації ширший за неавторизованого. До переліку його можливостей входять ті, які визначені в групі користувачів “неавторизований користувач”, а також доступ до особистого кабінету з можливістю перегляду історії замовлень та особистої інформації. При оформленні замовлення, авторизованому користувачу не потрібно кожного разу вводити свої дані, вони автоматично будуть підтягуватися на основі вказаних в особистому кабінеті.

Фахівець технічної підтримки має такі самі права доступу як авторизований користувач, але також користувачі з цими правами мають доступ до онлайн-чату технічної підтримки.

2.2 Структура web додатку

2.2.1 Загальна інформація про структуру web додатку

До структури web додатку входять усі його компоненти, які є загальнодоступними, а також адміністративна панель для користування співробітниками магазину.

Перелік сторінок web-додатку наступний:

– Головна сторінка містить навігаційне меню, яке закріплене на кожній

сторінці web-додатку, карусель із зображеннями акційних пропозицій та оголошень, популярні товари, та товари, які нещодавно надійшли до магазину;

– На сторінку каталогу можна потрапити, обравши певну категорію товару

або при пошуку товару за назвою. Тут розміщений асортимент товарів магазину. При виборі конкретного товару повинен здійснюватися перехід на сторінку з повною інформацією про нього;

– Кошик призначений для перегляду користувачем обраних товарів, загальну ціну за замовлення з можливістю видаляти товар зі списку або редагувати кількість однакових товарів до замовлення;

– На сторінці “Оформити замовлення” можна оформити замовлення, заповнивши особисті дані та контактну інформацію;

– В особистому кабінеті користувач може переглянути історію замовлень та редагувати особисті дані.

– На сторінці “Контакти” можна знайти інформацію про зв’язок з адміністрацією сайту, а також відправити листа з питанням про повернення чи проблемою з товаром до технічної підтримки онлайн магазину.

2.2.2 Навігаційне меню

Для зручної навігації в межах web-додатку необхідно створити закріплене меню, яке забезпечить швидке переміщення користувача між усіма доступними сторінками. Це меню повинно розташовуватися вгорі (у шапці) на кожній сторінці web-додатку.

2.2.3 Дизайн та структура web додатку

Для дизайну web-додатку потрібно використовувати мінімалістичний та сучасний стиль. Корпоративна палітра онлайн-магазину складається з відтінків білого та блакитного кольорів, тому при розробці веб-додатку необхідно використовувати саме ці кольори.

Розміри та типи шрифтів повинні бути зручними для читання. Інформаційні блоки, графічні матеріали та інші елементи веб-сторінок повинні мати логічне та зручне розташування, анімаційні ефекти можуть бути використані за потреби.

Прототип дизайну майбутнього web додатку зображено на рисунку А.1.

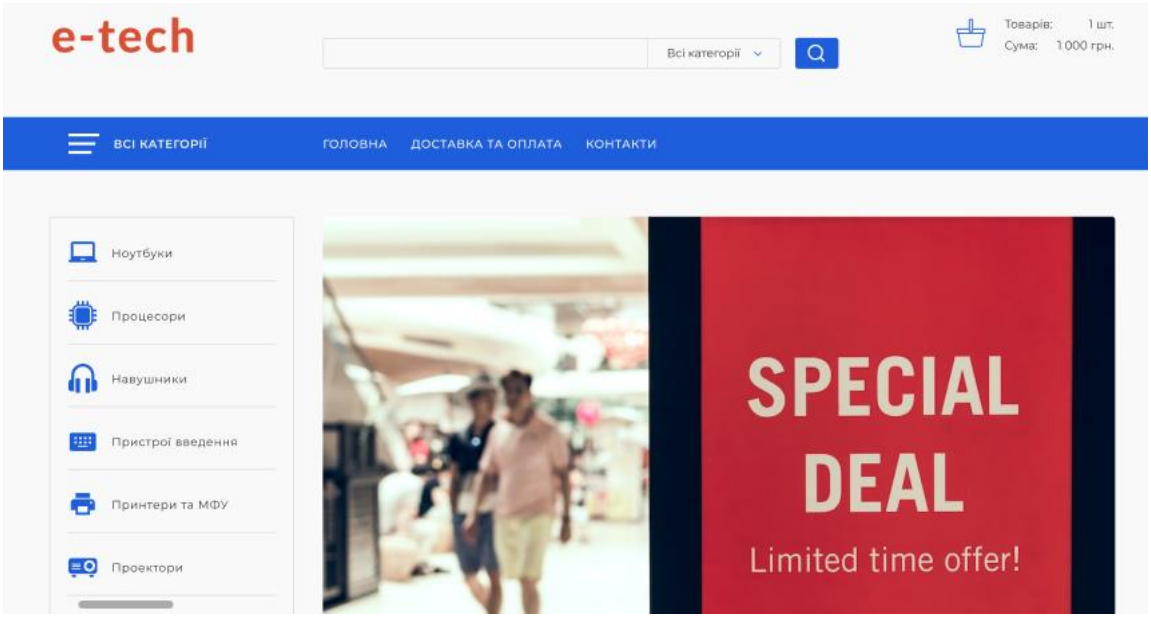


Рисунок А.1 – Прототип дизайну web додатку (головна сторінка)

2.3 Вимоги до функціонування системи

Відповідно до вимог, розроблений web Додаток має задовольняти функціональні потреби, які наведені у таблиці А.1.

Таблиця А.1 – Функціональні потреби користувача

ID	Потреби користувача	Джерело
UN-01	Перегляд головної сторінки сайту	Неавторизований користувач
UN-02	Перегляд каталогу товарів онлайн магазину	Неавторизований користувач
UN-03	Замовлення товарів онлайн	Неавторизований користувач

Продовження таблиці А.1

UN-04	Доступ до особистого кабінету	Авторизований користувач
UN-08	Додавання, редагування, видалення інформації	Адміністратор

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web додатку відбувається із використанням наступних технологій та інструментів:

- Технології та інструменти для розробки серверної частини додатку
  - IntelliJ IDEA
  - Java 17, Maven
  - Spring Framework
  - PostgreSQL
  - REST API
- Технології та інструменти для розробки клієнтської частини додатку
  - React
  - Redux
  - HTML, CSS, JS

2.4.2 Вимоги до лінгвістичного забезпечення

Web додаток буде містити один лінгвістичний пакет для локалізації інтерфейсу – українська мова.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

- Google Chrome – версія 66 і вище
- Opera – версія 53 і вище
- Mozilla Firefox – версія 60 і вище
- Apple Safari – версія 12 і вище
- Microsoft Edge – версія 79 і вище

3 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ WEB ДОДАТКУ

Детальний опис етапів створення web додатку наведено на рисунку А.2.

Розробка web-додатку підтримки продажу електронної техніки	**87 дней**	**Пн 16.01.23**	**Вт 16.05.23**
Розробка дизайну web-додатку	**7 дней**	**Пн 16.01.23**	**Вт 24.01.23**
Створення прототипу головної сторінки	1 день	Пн 16.01.23	Пн 16.01.23
Створення прототипу сторінки каталогу	1 день	Вт 17.01.23	Вт 17.01.23
Створення прототипу сторінки з повною інформацією про товар	1 день	Ср 18.01.23	Ср 18.01.23
Створення прототипу сторінки оформлення замовлень	1 день	Чт 19.01.23	Чт 19.01.23
Створення прототипу сторінки особистого кабінету	1 день	Пт 20.01.23	Пт 20.01.23
Розробка бази даних	**15 дней**	**Ср 25.01.23**	**Вт 14.02.23**
Розробка таблиць необхідних для функціонування веб-додатку	10 дней	Ср 25.01.23	Вт 07.02.23
Розробка допоміжних таблиць для більш зручного користування базою даних	5 дней	Ср 08.02.23	Вт 14.02.23
Back-end розробка	**21 дней**	**Ср 15.02.23**	**Ср 15.03.23**
Розробка механізму обміну інформацією з базою даних	4 дней	Ср 15.02.23	Пн 20.02.23
Розробка основних механізмів обміну даними з клієнтською частиною	5 дней	Вт 21.02.23	Пн 27.02.23
Розробка модулю авторизації	4 дней	Вт 28.02.23	Пт 03.03.23
Розробка модулю кошика	4 дней	Пн 06.03.23	Чт 09.03.23
Розробка модулю оформлення замовлень	4 дней	Пт 10.03.23	Ср 15.03.23
Front-end розробка	**30 дней**	**Чт 16.03.23**	**Ср 26.04.23**
Верстка каркасу основних сторінок веб-додатку	10 дней	Чт 16.03.23	Ср 29.03.23
Розробка механізму запиту даних з серверу	5 дней	Чт 30.03.23	Ср 05.04.23
Надання веб-сторінкам зовнішнього виду згідно з дизайном	10 дней	Чт 06.04.23	Ср 19.04.23
Надання окремим елементам анімацій, візуальних ефектів	5 дней	Чт 20.04.23	Ср 26.04.23
Наповнення веб-додатку контентом	**5 дней**	**Чт 27.04.23**	**Ср 03.05.23**
Фотографування товару з різних ракурсів	2 дней	Чт 27.04.23	Пт 28.04.23
Формування короткої інформації про товар	3 дней	Пн 01.05.23	Ср 03.05.23
Тестування web-додатку	**2 дней**	**Чт 04.05.23**	**Пт 05.05.23**
Тестування серверної частини	1 день	Чт 04.05.23	Чт 04.05.23
Тестування клієнтської частини	1 день	Пт 05.05.23	Пт 05.05.23

Рисунок А.2 – Етапи створення web додатку

4 ВИМОГИ ДО СКЛАДУ Й ЗМІСТУ РОБІТ ІЗ ВВЕДЕННЯ WEB ДОДАТКУ В ЕКСПЛУАТАЦІЮ

Для введення web додатку в експлуатацію необхідно розгорнути серверну і клієнтську частину на локальному хості робочої станції.

ДОДАТОК Б

Планування робіт

Деталізація мети проекту методом SMART [25]. Результатом виконання дипломного проекту є web додаток онлайн магазину електронної техніки. Результати деталізації методом SMART розміщені у табл. Б.1.

Таблиця Б.1 − Деталізація мети методом SMART

Specific	Створити web додаток онлайн магазину електронної техніки для автоматизації бізнес процесів
Measurable	Результатом роботи проекту є оцінка замовника
Achievable	Є спеціалісти для розробки і постійної підтримки сучасного web додатку, ліцензійний софт, ресурси. Працівники для організації технічної підтримки магазину. Постачальники новітніх і якісних товарів
Relevant	Для стабільного прибутку і можливості подальшого розширення магазину або реалізування нових проектів
Time-bound	Робота повинна бути виконана у терміни, що були обговорені із замовником.

Планування змісту структури робіт. WBS діаграма [26] є головним інструментом для створення структури робіт в проекті. Це графічне зображення згрупованих елементів проекту в пакети робіт, які ієрархічно пов’язані з продуктом проекту. Побудуємо WBS структуру, щоб детально описати роботи, які потрібно виконати на кожному етапі створення проекту, а також виконаємо декомпозицію робіт для даного проекту. Діаграма WBS зображена на рисунку Б.1.

Планування структури організації, для впровадження готового проекту. Після створення WBS розробимо OBS - організаційну структуру виконавців [27]. OBS стосується виключно внутрішньої організаційної структури проекту та не має відношення до взаємин між проектними групами або учасниками та їх батьківськими організаціями. Діаграма OBS зображена на рисунку Б.2. Список виконавців, що фігурують в проекті знаходиться в таблиці Б.2.

Таблиця Б.2 – Виконавці проекту

Роль	**Ім'я**	**Проектна роль**
Розробник	Куликов О. О.	Виконує front-end та back-end розробку
Проектувальник	Куликов О. О.	Виконує проектування бази даних та розробляє структуру web додатку
Тестувальник	Кущ Б. С. Марченко А. В.	Відповідає за тестування функціоналу та дизайну web додатку.
Керівник проекту	Куликов О. О.	Формує завдання на розробку проекту.
Менеджер проекту	Куликов О. О.	Відповідає за виконання термінів, розподіл ресурсів та завдань між учасниками. Виконує збір та аналіз даних.

Рисунок Б.1 – WBS структура робіт проекту

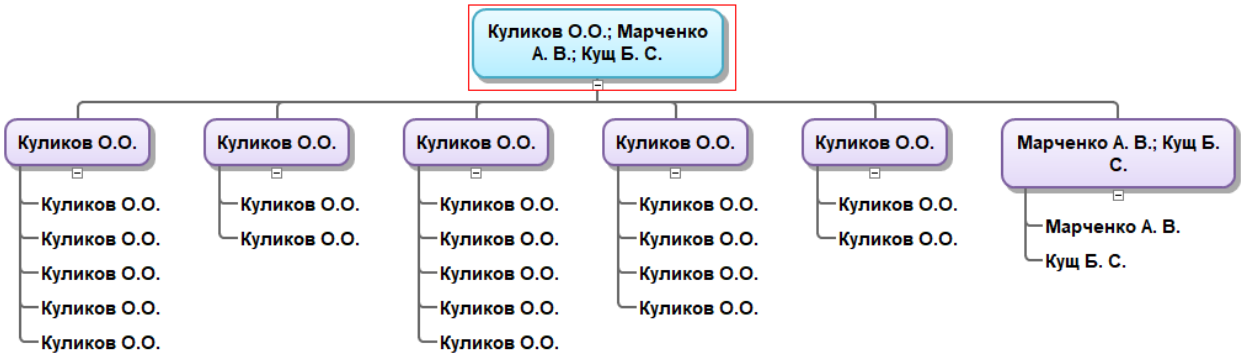
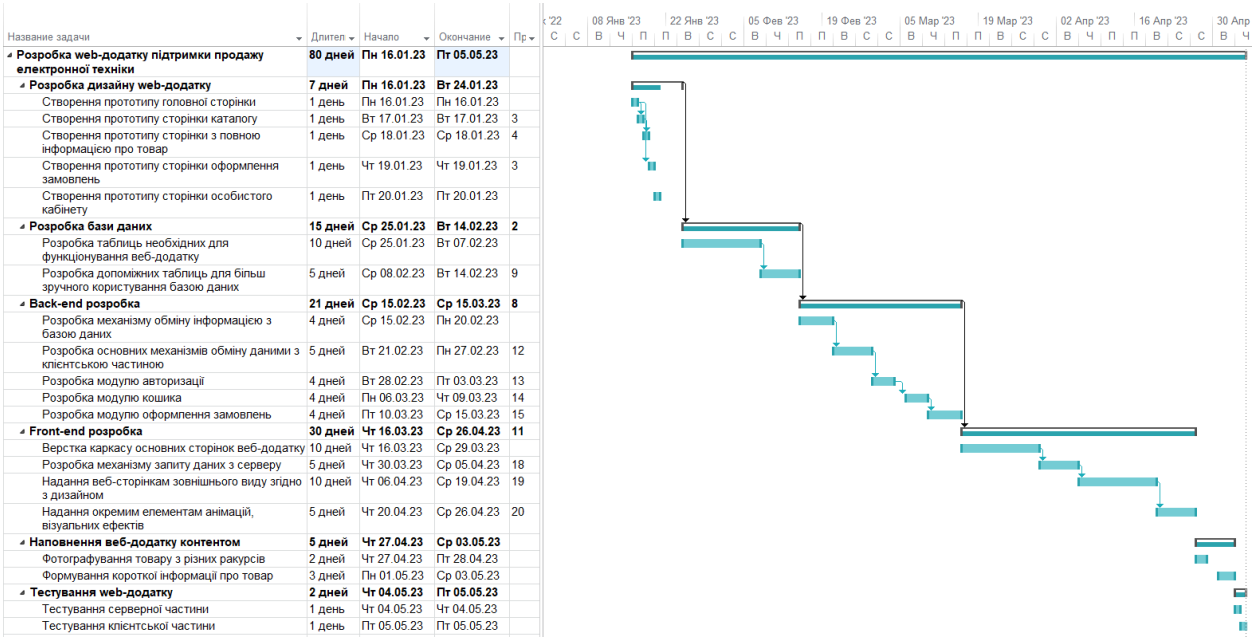


Рисунок Б.2 – OBS структура робіт проекту

Діаграма Ганта [28]. Далі створимо графік виконання дипломного проекту, використовуючи діаграму Ганта – найпоширеніший формат графіка в будь-якій галузі. Цей графік дозволяє менеджерам проекту та команді розробників візуалізувати графіки часу та взаємозв'язок між завданнями та етапами проекту. Тривалість виконання завдань зазначена в днях, але фактичний час, потрібний на їх виконання, становить близько 2-3 годин на день. Щоб мати реалістичне уявлення про тривалість виконання робіт з урахуванням обмеженості ресурсів, вихідних та святкових днів, було побудовано календарний графік. Діаграма Ганта і список робіт зображена на рисунку Б.3.



Название задачи	Длител	Начало	Окончание	Пр
Розробка web-додатку підтримки продажу електронної техніки	**80 дней**	**Пн 16.01.23**	**Пт 05.05.23**	
Розробка дизайну web-додатку	**7 дней**	**Пн 16.01.23**	**Вт 24.01.23**	
Створення прототипу головної сторінки	1 день	Пн 16.01.23	Пн 16.01.23	
Створення прототипу сторінки каталогу	1 день	Вт 17.01.23	Вт 17.01.23	3
Створення прототипу сторінки з повною інформацією про товар	1 день	Ср 18.01.23	Ср 18.01.23	4
Створення прототипу сторінки оформлення замовлень	1 день	Чт 19.01.23	Чт 19.01.23	3
Створення прототипу сторінки особистого кабінету	1 день	Пт 20.01.23	Пт 20.01.23	
Розробка бази даних	**15 дней**	**Ср 25.01.23**	**Вт 14.02.23**	**2**
Розробка таблиць необхідних для функціонування веб-додатку	10 дней	Ср 25.01.23	Вт 07.02.23	
Розробка допоміжних таблиць для більш зручного користування базою даних	5 дней	Ср 08.02.23	Вт 14.02.23	9
Back-end розробка	**21 дней**	**Ср 15.02.23**	**Ср 15.03.23**	**8**
Розробка механізму обміну інформацією з базою даних	4 дней	Ср 15.02.23	Пн 20.02.23	
Розробка основних механізмів обміну даними з клієнтською частиною	5 дней	Вт 21.02.23	Пн 27.02.23	12
Розробка модулю авторизації	4 дней	Вт 28.02.23	Пт 03.03.23	13
Розробка модулю кошика	4 дней	Пн 06.03.23	Чт 09.03.23	14
Розробка модулю оформлення замовлень	4 дней	Пт 10.03.23	Ср 15.03.23	15
Front-end розробка	**30 дней**	**Чт 16.03.23**	**Ср 26.04.23**	**11**
Верстка каркасу основних сторінок веб-додатку	10 дней	Чт 16.03.23	Ср 29.03.23	
Розробка механізму запиту даних з серверу	5 дней	Чт 30.03.23	Ср 05.04.23	18
Надання веб-сторінкам зовнішнього виду згідно з дизайном	10 дней	Чт 06.04.23	Ср 19.04.23	19
Надання окремим елементам анімацій, візуальних ефектів	5 дней	Чт 20.04.23	Ср 26.04.23	20
Наповнення веб-додатку контентом	**5 дней**	**Чт 27.04.23**	**Ср 03.05.23**	
Фотографування товару з різних ракурсів	2 дней	Чт 27.04.23	Пт 28.04.23	
Формування короткої інформації про товар	3 дней	Пн 01.05.23	Ср 03.05.23	
Тестування web-додатку	**2 дней**	**Чт 04.05.23**	**Пт 05.05.23**	
Тестування серверної частини	1 день	Чт 04.05.23	Чт 04.05.23	
Тестування клієнтської частини	1 день	Пт 05.05.23	Пт 05.05.23	

Рисунок Б.3 – Діаграма Ганта

Аналіз ризиків. Проведемо аналіз ризиків, включаючи якісну і кількісну оцінку. Якісна оцінка дозволить ідентифікувати ризики, які потребують негайної реакції, і визначити їх важливість для вибору стратегії реагування. Кількісна оцінка потрібна для повного розуміння ризиків та їх впливу на проект. Якісна та кількісна оцінки можуть використовуватися окремо або в поєднанні, залежно від обмежень в часі, бюджеті та потреби в оцінці ризиків. У таблиці Б.6 наведена класифікація ризиків залежно від ймовірності виникнення та наслідків. Далі буде розроблений план реагування на ризики, включаючи методи та технології для зниження їх негативного впливу на проект. Буде оцінена ефективність плану реагування та вплив ризиків на проект, враховуючи можливі позитивні або негативні наслідки. Оцінка ризиків буде здійснюватися за показниками, що наведені в табл. Б.3, і на основі цього буде побудована матриця ймовірності виникнення ризиків та їх впливу, яка приведена в таблиці Б.4.

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

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

Таблиця Б.4 – Матриця ймовірності виникнення ризиків та впливу ризику

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

Вплив ризику

На підставі отриманого значення індексу, ризики класифікують за рівнем ризику, що наведено в таблиці Б.5

Таблиця Б.5 – Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, що входять (номери)
1	Прийнятні	$1 \leq R \leq 2$	1, 4, 6, 8, 10, 11, 12
2	Виправдані	$3 \leq R \leq 4$	2, 3, 9, 13, 14
3	Недопустимі	$6 \leq R \leq 9$	5, 7, 15

Таблиця Б.6 – Оцінка ймовірностей виникнення, величини втрат та індексу ризику

ID ризику	Статус	Опис	Ймовірність	Вплив	Ранг	План А	Тип стратегії	План Б
RS_1	Відкритий	Непорозуміння між розробником та замовником	Низька	Середній	2	Налагодження гарних відносин між розробником та керівником. Дотримання ділового етикету спілкування. Створення комфортних умов для співпрацю	Зменшення	Обговорення причини непорозуміння та створення здорової атмосфери в колективі.

Продовження таблиці Б.6

RS_2	Відкритий	Поява альтернативного продукту	Низька	Високий	3	Провести попереднє дослідження альтернативних продуктів. Вибрати унікальну стратегію створення проекту	Прийняття	
RS_3	Відкритий	Часте внесення змін у ТЗ	Низька	Високий	3	Завчасне обговорення усіх фактичних і можливих виправленнь. Домовлення з замовником про можливість виправлень на розсуд розробника	Зменшення	Фокусування на виправленнях замовника, потім продовження роботи над проектом
RS_4	Закритий	Неоптимальний розподіл часу	Низька	Середній	2	Редагування календарного плану робіт	Ухилення	Обговорення з замовником щодо відкладання терміну здачі проекту

Продовження таблиці Б.6

RS_5	Закритий	Нестача робочої сили	Висока	Середній	6	Збільшення робочих годин	Прийняття	Залучення сторонніх довірених осіб до роботи
RS_6	Закритий	Помилки з програмним забезпеченням	Низька	Низький	1	Вирішення проблеми з програмним забезпеченням	Ухилення	Використання іншого програмного забезпечення
RS_7	Закритий	Вибір не ефективної технології розробки	Середня	Високий	6	Проаналізувати методи та засоби, для виконання проекту. Обрати зрозумілу та легку в використанні технологію розробки	Пом’якшення	Виділити час та ресурси на покращення обраної технології. Застосувати допоміжні ресурси

Продовження таблиці Б.6

RS_8	Закритий	Неправильна оцінка в масштабі проекту	Низька	Середній	2	Провести детальний аналіз проекту Визначити основні його етапи, розподілити час на виконання. Проаналізувати масштаби проекту на основі додаткових джерел	Пом’якшення	Переоцінка масштабів проекту Перебудова стратегії реалізації проекту
RS_9	Закритий	Помилки проектування	Низька	Високий	3	На етапі проектування тісно співпрацювати із замовником та демонструвати проміжні результати	Пом’якшення	Здійснювати проміжний контроль результатів в ході виконання проекту

Продовження таблицці Б.6

RS_10	Відкритий	Реалізація непотрібного функціоналу	Середня	Низький	2	Попередити замовника про можливість додаткового функціоналу	Використання	Обговорити вигоду і збитки від можливих змін проекту
RS_11	Відкритий	Відсутність резервних копій даних	Низька	Середній	2	Налаштувати автоматичне збереження даних. Зберігати дані на різних носіях інформації	Попередження	Робити копію даних після кожного виконаного етапу
RS_12	Відкритий	Невиконання моніторингу проекту	Середня	Низький	2	Здіснювати проміжний контроль результатів в ході виконання проекту	Перенос	Здійснювати контроль проміжних результатів замовником
RS_13	Відкритий	Зміна вимог замовника в процесі розробки проекту	Низька	Високий	3	Узгодити всі питання на початкових етапах розробки, щоб мінімізувати кількість змін під час розробки	Пом’якшення	Переоцінка проекту кожного разу, коли вимоги змінюються

Продовження таблиці Б.6

RS_14	Відкритий	Низька кваліфікація розробників проекту	Середня	Середній	4	Підвищити кваліфікацію персоналу. Переглянути онлайн ресурси для підвищення рівня знань	Пом’якшення	Врахувати час на підготовку працівників. Видати літературу, переглянути онлайн ресурси
RS_15	Відкритий	Нечітке завдання на розробку	Середня	Високий	6	Однозначно обговорити із замовником усі види вимог. Періодичний контроль етапів роботи замовником	Попередження	При виявленні Невідповідностей деяких характеристик web додатку заявленим вимогам, потрібно уважно та чітко окреслити те, що було виконано невірно та зробити правки

ДОДАТОК В

Web додаток онлайн магазину електронної техніки

Куликов О.О., *студент ІТ-91*; Марченко А.В., *доцент*
Сумський державний університет, м. Суми, Україна

Стрімкий розвиток інформаційних технологій та їх вчасне впровадження в бізнес-процеси підприємництва сприяє підвищенню конкурентоспроможності та надання послуг на найвищому рівні якості. Реалізація web додатку для підтримки онлайн продажу техніки оптимізує ведення бізнесу за рахунок раціонального використання ресурсів та значного розширення аудиторії споживачів послуг.

Основним призначенням web додатку є легке і швидке оформлення замовлень товарів електронної техніки відповідно до уподобань користувача та гнучких налаштувань пошуку товару за різними параметрами.

Для реалізації web-додатку була обрана мікросервісна архітектура. Специфікою обраної архітектури є предметно розподілена логіка між сервісами, кожен з яких підпорядковується предметно-орієнтованому дизайну. А отже, тестування та налагодження коректної роботи сервісів для інтегрованої роботи відбувається набагато швидше і легше. Крім цього мікросервісний підхід дозволяє забезпечити стійку роботу через можливість швидкого відновлення сервісу за допомогою оркестру для контейнерів, наприклад, таких як k8s. Під час вибору системи управління базою даних для реалізації сховища даних web-додатку головними факторами були гнучкість при формуванні бізнес-об’єктів та простота моделей даних, безпека бази даних та її висока продуктивність. Враховуючи дані критерії, було обрано систему управління базою даних PostgreSQL, яка використовує підхід формування моделей у вигляді гнучких структурованих об'єктів. В процесі розробки web-додатку були використані такі технології та інструменти для реалізації серверної частини, як IntelliJ IDEA, Java 17, Maven, Spring Framework, PostgreSQL, REST API. Клієнтська частина була розроблена з використанням технологій React, Redux та HTML, CSS, JS. Модульне та мануальне тестування розробленого додатку пройшло успішно. Результати тестування встановили повну відповідність розробленого функціоналу специфікації вимог. Інтерфейс web додатку онлайн магазину відповідає сучасним вимогам та може бути впроваджений в роботу магазинів продажу електронної техніки.

ДОДАТОК Г

Клієнтська частина web-додатку

App.css

```
#root {
  margin: 0 auto;
  padding: 0;

}

body {

}

.button {
  @apply py-2 px-4 border border-black/20 rounded-[3px] transition-all hover:scale-105
}

.white-button {
  @apply bg-white text-black
}

.blue-button {
  @apply bg-blue text-white active:bg-clicked-blue
}

.orange-button {
  @apply bg-orange text-white transition-all active:bg-clicked-orange;
}
```

env.js

```
export const server = 'https://etech-5fydkirpga-lm.a.run.app'
export const local = 'http://serverhost:8080'
export const roles = {
    ADMIN: 'ROLE_ADMIN',
    MANAGER: 'ROLE_MANAGER',
    USER: 'ROLE_USER',
}
```

CartModal.css

```
.cartModal {
    @apply relative w-max border border-black/20
    bg-white overflow-auto rounded-[3px] outline-none py-5 px-10
    top-1/2 left-1/2 -translate-x-1/2 -translate-y-1/2;
}

.cartHeader {
    @apply font-bold text-[24px];
}

.cartBody {
    @apply py-5;
}

.x-button {
    @apply text-black/60 text-xl absolute top-2 right-5
}

.cartOverlay {
    @apply fixed top-0 left-0 right-0 bottom-0 bg-black/30 z-10;
}
```

CategoryList.css

```
.root-category-list {
    @apply border border-black/20 rounded-[3px] h-min w-full px-3
}

.root-category-list li {
    @apply py-3 border-b border-black/20 cursor-pointer
transition-all
}

.root-category-list li:last-child {
    border: none;
}

.child-category-list li {
    @apply py-3 ml-3
}

.child-category-list li:first-child {
}

.child-category-list li:last-child {
    @apply pb-0
}

.child-category-list p {

}
```

ShippingPage.css

```
.contacts, .comments {
    @apply w-max mt-5;
}
```

```
.contacts div div, .comments div {
    @apply flex flex-col mt-2 ml-5;
}

.contacts input {
    @apply border border-black/20 rounded-[3px] py-1 px-3;
}

.comments textarea {
    @apply border border-black/20 rounded-[3px] py-1 px-3;
}

.order-listing {
    @apply ml-5;
}

.order-listing > a {
    @apply py-5;
}

.order-summary {
    @apply w-1/3 border border-black/20 rounded-[3px] bg-black/10
p-5 h-max ml-10;
}

.order-summary > div {
    @apply flex justify-between border-b border-black/20 py-5
}
```

store.js

```
import {configureStore} from "@reduxjs/toolkit";
import          {cartReducer,          loadFromCookie}          from
"./reducers/cartReducer.js";
```

```
import {getCookie, setCookie} from "../utils/cookieUtils.js";
import {authReducer, saveToken} from "./reducers/authReducer.js";


export const store = configureStore({
    reducer: {
        auth: authReducer,
        cart: cartReducer,
    }
})

store.subscribe(() => {
    const authState = store.getState().auth;
    const cartState = store.getState().cart;
    setCookie('authState', JSON.stringify(authState));
    setCookie('cartState', JSON.stringify(cartState));
});

const authState = getCookie('authState');
const cartState = getCookie('cartState');
if (authState) {
    const parsedState = JSON.parse(authState);
    store.dispatch(saveToken(parsedState)
    );
}
if (cartState) {
    const parsedState = JSON.parse(cartState);
    store.dispatch(loadFromCookie(parsedState.cart));
}
```

authReducer.js

```
import {createSlice} from '@reduxjs/toolkit';

const authSlice = createSlice({
```

```
    name: 'auth',
    initialState: {
        username: null,
        lastname: null,
        firstname: null,
        middlename: null,
        email: null,
        token: null,
        role: null,
    },
    reducers: {
        saveToken: (state, action) => {
            state.username = action.payload.username;
            state.lastname = action.payload.lastname;
            state.firstname = action.payload.firstname;
            state.middlename = action.payload.middlename;
            state.email = action.payload.email;
            state.token = action.payload.token;
            state.role = action.payload.role;
        },
        removeToken: (state, action) => {
            state.username = null;
            state.lastname = null;
            state.firstname = null;
            state.middlename = null;
            state.email = null;
            state.token = null;
            state.role = null;
        }
    }
});

export const authReducer = authSlice.reducer;

export const {
    saveToken,
```

```
    removeToken,
} = authSlice.actions;
```

cartReducer.js

```
import {createSlice} from '@reduxjs/toolkit';

const cartSlice = createSlice({
    name: 'cart',
    initialState: {
        cart: [],
    },
    reducers: {
        loadFromCookie: (state, action) => {
            state.cart = action.payload;
        },
        addToCart: (state, action) => {
            const itemInCart = state.cart.find((item) => item.id
=== action.payload.id);
            if (itemInCart) {
                itemInCart.quantity++;
            } else {
                state.cart.push({...action.payload, quantity: 1});
            }
        },
        incrementQuantity: (state, action) => {
            const item = state.cart.find((item) => item.id ===
action.payload);
            item.quantity++;
        },
        decrementQuantity: (state, action) => {
            const item = state.cart.find((item) => item.id ===
action.payload);
            if (item.quantity === 1) {
                item.quantity = 1
```

```
                } else {
                    item.quantity--;
                }
            },
            removeItem: (state, action) => {
                state.cart = state.cart.filter((item) => item.id !==
action.payload);
            },
            clearCart: (state, action) => {
                state.cart = [];
            }
        },
    });

    export const cartReducer = cartSlice.reducer;
    export const {
        loadFromCookie,
        addToCart,
        incrementQuantity,
        decrementQuantity,
        removeItem,
        clearCart,
    } = cartSlice.actions;
```

Alert.css

```
    .alert {
        @apply fixed right-1/4 top-10 translate-x-1/2 border-t-4
rounded-[3px] px-4 py-3 drop-shadow-lg
    }

    .success {
        @apply bg-lightBlue border-blue;
    }
```

```
.error {
    @apply bg-lightRed border-[red];
}
```

Файл: \client\src\ui\Button.css

Файл: \client\src\ui\Carousel.css

```
.carousel {
    position: relative;
    @apply min-w-[800px] ml-10 min-h-[300px]
}

.carousel-items {
    @apply border border-black/20 w-full rounded-[3px] p-5
}

.carousel-item {
    @apply block w-max mx-auto z-10 relative;
    flex-shrink: 0;
    transition: transform 0.5s ease;
    transform: translateX(-100%);
}

.carousel-item[data-active="true"] {
    transform: translateX(0);
}

.carousel-item img {
    @apply max-h-[300px] mx-auto;
}

.carousel-controls {
    position: absolute;
```

```
        top: 50%;
        left: 0;
        right: 0;
        transform: translateY(-50%);
        display: flex;
        justify-content: space-between;
        @apply h-full
    }

    .carousel-control {
        background-color: transparent;
        border: none;
        outline: none;
        cursor: pointer;
        @apply w-20
    }

    .carousel-control p {
        font-size: 32px;
        font-family: Montserrat, sans-serif;

    }
```

cookieUtils.js

```
    export function setCookie(name, value, days) {
        const expirationDate = new Date();
        expirationDate.setDate(expirationDate.getDate() + days);

        const cookieValue = encodeURIComponent(value) + (days ? `;
expires=${expirationDate.toUTCString()}` : '');
        document.cookie = `${name}=${cookieValue}; path=/`;
    }

    export function getCookie(name) {
```

```
        const cookieName = `${name}=`;
        const cookies = document.cookie.split(';');

        for (let i = 0; i < cookies.length; i++) {
            let cookie = cookies[i];
            while (cookie.charAt(0) === ' ') {
                cookie = cookie.substring(1);
            }
            if (cookie.indexOf(cookieName) === 0) {
                return
decodeURIComponent(cookie.substring(cookieName.length, cookie.length));
            }
        }

        return null;
    }
```

App.jsx

```
    import React from 'react'
    import './App.css'
    import {Route, Routes} from 'react-router-dom'
    import {HomePage} from './pages/HomePage'
    import {Layout} from './components/Layout'
    import {ContactPage} from './pages/ContactPage'
    import ErrorPage from './pages/ErrorPage'
    import ProductList from './components/products/ProductList.jsx'
    import                    ProductDetails                    from
"./components/products/ProductDetails.jsx";
    import CabinetPage from './pages/CabinetPage';
    import {AuthPage} from "./pages/AuthPage.jsx";
    import {AdminPage} from "./pages/AdminPage.jsx";
    import ShippingPage from "./pages/ShippingPage.jsx";

    function App () {
      return (
```

```
        <>
          <Routes>
            <Route path='/' element={<Layout />}>
                <Route index element={ <HomePage /> } />
                <Route             path='/products/category/:category'
element={<ProductList />} />
                <Route                          path='/products/id/:id/'
element={<ProductDetails />} />
                <Route path='/products' element={<ProductList />} />
                <Route path='/cabinet' element={ <CabinetPage /> } />
                <Route path='/contact' element={ <ContactPage /> } />
                <Route path='/error' element={<ErrorPage />} />
                <Route  path='/authentication'  element={<AuthPage  />}
/>
                <Route path='/admin' element={<AdminPage />} />
                <Route path='/shipping' element={<ShippingPage />}/>
            </Route>
          </Routes>
        </>
      );
    }

    export default App;
```

main.jsx

```
    import React from 'react'
    import ReactDOM from 'react-dom/client'
    import App from './App'
    import "./index.css"
    import {BrowserRouter} from 'react-router-dom'
    import {Provider} from "react-redux";
    import {store} from "./store/store.js"
```

```
ReactDOM.createRoot(document.getElementById('root')).render(
    <React.StrictMode>
        <BrowserRouter>
            <Provider store={store}>
                <App/>
            </Provider>
        </BrowserRouter>
    </React.StrictMode>
);
```

CartModal.jsx

```
import React, {useState} from "react";
import Modal from "react-modal"
import './CartModal.css';
import {useSelector} from "react-redux";
import CartItem from "./cart/CartItem.jsx";
import {Link} from "react-router-dom";

Modal.setAppElement('#root')

//TODO
export default function CartModal() {
    const {cart} = useSelector((state) => state.cart)
    const [open, setOpen] = useState(false);


    const toggleModal = () => setOpen(!open);

    return (
        <div className='h-[35px]'>
            <button onClick={toggleModal}
                    className='px-3 rounded-[3px] hover:bg-blue/10
transition-all duration-400 active:bg-clicked-blue/20'>
```

```
                    <img   src='/src/images/cart.svg'   alt='cart.svg'
className=''/>
                </button>

                <Modal   isOpen={open}   onRequestClose={toggleModal}
className='cartModal' overlayClassName='cartOverlay'>
                    <div className='cartHeader'>РљРsС€РёРє</div>

                    <div className="cartBody">
                        {
                            cart.length > 0 ? cart.map(item => (
                                    <CartItem            key={item.id}
item={item}/>
                                )) :
                                <Link                    to='/products'
onClick={toggleModal}>Р’  РєРsС€РёРєСѓ  PSPµPjР°С”  С,РsPIР°СЂС–PI<br
/>РќР°С,РёСЃPSС–С,СЊ  СЃСЂРґРё,  С‰РsР±  РsР±СЂР°С,Рё  С‰РsСЃСЊ  С†С–
РєР°PIPµPSCЊРєРµ</Link>
                        }
                    </div>

                    <Link to={'/shipping'} className={`button orange-
button       ${cart.length       ===       0       &&       'hidden'}`}
onClick={toggleModal}>РћС„РsСЂРjРёС,Рё Р·Р°РjРsPIP»РµPSPSCЦ</Link>
                    <button                          className='x-button'
onClick={toggleModal}>x</button>
                </Modal>
            </div>
    );
    }
```

Footer.jsx

```
    import React from 'react';
    import { Link } from 'react-router-dom';
```

```
import { animateScroll as scroll } from 'react-scroll';

const Footer = () => {
  return (
    <footer className="bg-blue bottom-0 w-full text-white min-w-[1110px]">
      <div className="w-[1110px] mx-auto grid gap-8 pt-10 items-center">
        <div className="grid grid-cols-3 gap-4">
          <div className="grid">
            <p className="text-orange font-bold text-4xl leading-[61.7px]">Silex</p>
            <p className="pt-10">
              P†PSC,PµCЂPSPµC, PjP°PiP°P·PëPS PµP»PµPєC,CЂPsPSPSPsC— C,PµC…PSC–PєPë,
              PI CЏPєPsPjCґ P·PSP°P№PґPµC,CЊCЃCЏ PICЃPµ, C‰Ps PïPsC,CЂC–P±PSPs
            </p>
          </div>
          <div className="grid border-l-2 border-white/30 justify-center text-start pt-2">
            <span className="text-lg font-semibold text-16 uppercase">PµCЂPs PєPsPjPïP°PSPëCЋ</span>
            <Link to="/error" className="block" onClick={() => scroll.scrollTo('error')}>P”PsCЃC,P°PIPєP°</Link>
            <Link to="/checkout" className="block" onClick={() => scroll.scrollTo('checkout')}>PЋPïP»P°C,P°</Link>
            <Link to="/contact" className="block" onClick={() => scroll.scrollTo('contact')}>PЪPsPSC,P°PєC,Pë</Link>
            <Link to="/products" className="block" onClick={() => scroll.scrollTo('products', { smooth: true, duration: 500 })}>PЪP°C,PµPiPsCЂC–C—</Link>
          </div>
        </div>
        <div className="justify-center flex items-center border-t-2 border-white/60 py-6">
```

```
              <span          className="text-white/70">B©          {new
Date().getFullYear()} - PќP°C€ PSP°PNºPєCЂP°C‰PёPNº PjP°PiP°P·PёPS</span>
            </div>
          </div>
        </footer>
      );
    };

    export default Footer;
```

Header.jsx

```
    import React from 'react'
    import ProductSearch from '../ui/ProductSearch'
    import {Link} from 'react-router-dom'
    import NavbarHeader from '../ui/NavbarHeader'
    import {useSelector} from "react-redux";
    import {CiUser} from "react-icons/all.js";
    import CartModal from "./CartModal.jsx";


    const Header = () => {
        const {token} = useSelector((state) => state.auth);

        return (
            <div className='min-w-[1110px]'>
                <div className='flex justify-between items-center px-
[20%] py-[50px]'>
                    <Link to="/"
                          className='text-orange   font-[900]   font-
[Lato] text-[48px] leading-[61.7px] tracking-[0.05em] px-5 '>
                        SILEX
                    </Link>
```

```
                    <ProductSearch/>
                    <div className='flex rounded-[3px] p-1 shadow'>
                        <CartModal/>
                        {token === null ?
                            <Link to={'/authentication'}
                                  className='block px-3 rounded-[3px]
hover:bg-blue/10   transition-all   duration-400   active:bg-clicked-
blue/20'>
                                <CiUser                 size={'35px'}
fill={'#1E5DDB'}/>
                            </Link>
                            :
                            <Link to={'/cabinet'}
                                  className='block px-3 rounded-[3px]
hover:bg-blue/10   transition-all   duration-400   active:bg-clicked-
blue/20'>
                                <CiUser                 size={'35px'}
fill={'#1E5DDB'}/>
                            </Link>
                        }
                    </div>
                </div>
                <NavbarHeader/>
            </div>
        )
    }

    export default Header
```

Layout.jsx

```
import React from 'react'
import { Outlet } from 'react-router'
import Header from './Header'
import Footer from './Footer'
```

```
const Layout = () => {
  return (
    <>
        <Header />
        <Outlet/>
        <Footer />
    </>
  )
}

export {Layout}
```

CartItem.jsx

```
import React from "react";
import {Link} from "react-router-dom";
import {server} from "../../env.js";
import {useDispatch} from "react-redux";
import {removeItem} from "../../store/reducers/cartReducer.js";

const CartItem = ({item}) => {
    const dispatch = useDispatch();

    function handleRemoveItem() {
        dispatch(removeItem(item.id));
    }
    return (
        <div  className='relative  border-b  border-black/10  p-5
last:border-none'>
            <Link to={`/products/id/id=${item.id}`}
                  className="flex ">
                <img                src={server                +
`/images/product/${item.imgPath}`}  alt={item.title}  className='max-w-
[75px]'/>
```

```
                    <div className='flex flex-col pl-5 w-full justify-
between'>
                        <p className='w-4/5 truncate'>{item.title}</p>
                        <p className='text-right'>{item.price} в,г</p>
                    </div>
                </Link>
                <button                              className='x-button'
onClick={handleRemoveItem}>x</button>
            </div>
        );
    };

    export default CartItem;
```

CartTotal.jsx

```
    import React from 'react';
    import { useSelector } from 'react-redux';

    const CartTotal = () => {
      const cartItems = useSelector(state => state.cart);
      const total = cartItems.reduce((acc, item) => acc + item.price *
item.quantity, 0);

      return (
        <div className="cart-total">
          <h3>Total: ${total}</h3>
        </div>
      );
    };

    export default CartTotal;
```

CategoryList.jsx

```
import React, {useEffect, useState} from "react";
import axios from "axios";
import {server} from "/src/env.js"
import {Link} from "react-router-dom";
import "./CategoryList.css"
import {useSelector} from "react-redux";

const CategoryList = () => {
    const [categories, setCategories] = useState([]);
    const [openCategory, setOpenCategory] = useState(null);
    const {token} = useSelector((state) => state.auth);

    useEffect(() => {
        const fetchCategories = async () => {
            const response = await axios.get(server + "/categories/all").catch(console.log)
            setCategories(response.data);
        };

        fetchCategories();
    }, []);

    const config = {
        headers: {
            Authorization: `Bearer_${token}`,
        }
    }

    const handleMouseClick = (categoryId) => {
        setOpenCategory(categoryId);
    };

    return (
```

```
            <ul className="root-category-list">
                {categories.length === 0 ? <li>Loading...</li> :
categories.map((category) => (
                    category.childCategories.length > 0 && (
                        <li key={category.id}>
                            <p onClick={() =>
handleMouseClick(category.id)}>
                                {category.title}

                            </p>
                            {openCategory === category.id && (
                                <ul className='child-category-list'>

{category.childCategories.map((childCategory) => (
                                        <li key={childCategory.id}>
                                            <p>
                                                <Link
to={`/products/category/category=${childCategory.title}`}>

{childCategory.title}
                                                </Link>
                                            </p>
                                        </li>
                                    ))}
                                </ul>
                            )}
                        </li>
                    ))}
            </ul>
        );
    };

    export default CategoryList;
```

ProductCard.jsx

```
import React from "react";
import {Link} from "react-router-dom";
import {server} from '../../env.js'

const ProductCard = ({product}) => {
    function getPriceWithDiscount(price, discount) {
        return discount !== 0 ? price - price / 100 * discount :
price;
    }

    if (!product) {
        return <div>Loading...</div>;
    }

    return (
        <Link to={`/products/id/id=${product.id}`}>
            <div  className="mx-auto  px-5  py-8  border  border-
black/20 rounded-[3px] gap-4 h-full hover:shadow-lg">
                <div className='h-[200px] text-center'>
                    <img
                        src={server                                +
`/images/product/${product.imgPath}`}
                        alt={product.title}
                        className="max-h-[200px] inline-block"
                    />
                </div>
                <p   className="mt-4   text-[16px]   text-gray-700
truncate">
                    {product.title}
                </p>
                <p classNam={product.discount >= 1 ? 'mt-2 text-
[22px]          font-bold          text-[#f84147]'          :          '
hidden'}>{getPriceWithDiscount(product.price, product.discount)} в,г
```

```
                    </p>
                    <p className={product.discount > 0 ? 'font-[400] text-[16px] text-black/80 line-through ' : " mt-2 font-[700] text-[24px]"}>{product.price} в,г</p>
                </div>

            </Link>
        );
    };

    export default ProductCard;
```

ProductDetails.jsx

```
    import React, {useEffect, useState} from "react";
    import axios from "axios";
    import {server} from "/src/env.js"
    import {useParams} from "react-router-dom";
    import Warranty from "../../ui/Warranty";
    import AdditionalForProduct from "../../ui/AdditionalForProduct";
    import AboutProduct from "../../ui/AboutProduct";
    import ProductSame from "./ProductSame";
    import {useDispatch} from "react-redux";
    import {addToCart} from "../../store/reducers/cartReducer.js";

    const ProductDetails = () => {
        const { id } = useParams();
        const [product, setProduct] = useState(null);
        const dispatch = useDispatch();

        useEffect(() => {
            const getData = async () => {
                try {
                    const response = await axios.get(
```

```
                    server + `/products/${id}`
                );
                setProduct(response.data);
            } catch (e) {
                console.log(e);
            }
        };
        getData();
    }, [id]);

    if (!product) {
        return null;
    }

    function getPriceWithDiscount(price, discount) {
        return discount !== 0 ? price - (price / 100) * discount :
price;
    }

    function getSavedPrice(price, discount) {
        return price - getPriceWithDiscount(price, discount);
    }

    function handleClickBuy() {
        dispatch(addToCart(product));
    }

    return (
        <div>
            <div className="w-[1110px] mx-auto flex justify-center
my-10">
                <div className="mr-10 min-w-[40%] p-5 shadow-lg ">
                    <img              src={server              +
`/images/product/${product.imgPath}`}
                         alt={product.title} className="w-full"/>
                </div>
```

```
                    <div className="ml-10">
                        <h2        className="font-[700]        text-
[24px]">{product.title}</h2>
                        <div className="flex gap-4 mt-4">
                        <p className='border border-black/30 p-1 text
rounded-[3px] mb- ml-auto w-min'>{product.producer}</p>
                        <div   className="border   border-[#00bc52]/60
rounded-[3px]  p-1  float-right  text-[#00bc52]">P’  PSP°СЏPIPSPsСЃС,С-
</div>
                        </div>

                        <p className={product.discount > 0 ?
                            "font-[400]   text-[16px]   text-black/80
line-through " : "font-[700] text-[24px]"}>
                            {product.price} в,ґ
                        </p>

                        {product.discount > 0 && (
                            <div className="flex gap-4 items-baseline
">

                                <p  className="font-[700]  text-[28px]
text-[#f84147]">

{getPriceWithDiscount(product.price, product.discount)} в,ґ
                                </p>
                                <p  className="font-[700]  text-[18px]
text-black/80">
                                    {getSavedPrice(product.price,
product.discount)} в,ґ
                                </p>

                            </div>
                        )}
                        <Warranty />
```

```
                        <AdditionalForProduct />
                        <button className='button orange-button mt-4'
onClick={handleClickBuy}>
                            РљСѓРїРёС,Ре
                        </button>
                    </div>
                </div>
                <div className=" w-[1110px] mx-auto flex justify-center
mt-20">
                    <AboutProduct />
                </div>
                <div className=" w-[1110px] mx-auto flex justify-center
mt-0">
                    <ProductSame />
                </div>
            </div>
        );
    };

    export default ProductDetails;
```

ProductList.jsx

```
    import React, {useEffect, useState} from 'react';
    import {useParams} from "react-router-dom";
    import axios from "axios";
    import ProductCard from "./ProductCard.jsx";
    import {server} from "/src/env.js"

    const ProductList = () => {
        const {category} = useParams();
        const [products, setProducts] = useState([]);

        useEffect(() => {
```

```
            const fetchProducts = async () => {
                let response;
                if (category !== undefined)
                    response = await axios.get(server + `/products/${category}`).catch(console.log);
                else
                    response = await axios.get(server + `/products/all`).catch(console.log);
                setProducts(response.data);
            };

            fetchProducts();
        }, [category]);

        return (
            <div className="mx-auto w-[1110px] gap-4 my-10 grid grid-cols-4 ">
                {products.map((product) => (
                    <ProductCard key={product.id} product={product}/>
                ))}
            </div>
        );
    };

    export default ProductList;
```

ProductNew.jsx

```
    import React, {useEffect, useState} from 'react';
    import {useParams} from "react-router-dom";
    import axios from "axios";
    import ProductCard from "./ProductCard.jsx";
    import {server} from "/src/env.js"
```

```
    const ProductNew = () => {
        const {category} = useParams();
        const [products, setProducts] = useState([]);

        useEffect(() => {
            const fetchProducts = async () => {
                try {
                    let response;
                    if (category !== undefined )
                        response = await axios.get(server +
`/products/${category}`);
                    else
                        response = await axios.get(server +
`/products/all`)
                    setProducts(response.data);
                } catch (error) {
                    console.log(error);
                }
            };

            fetchProducts();
        }, [category]);

        return (
            <div className='mt-[7%]'>
                <span  className='font-bold  text-[24px]'>PќPsPIC-
PSP°PrC…PsPrP¶PµPSPSCЏ</span>
                <div  className="mx-auto  w-[1110px]  gap-4  my-10  grid
grid-cols-4 ">
                {products.map((product, index) => (
                    index  <  4  &&  <ProductCard  key={product.id}
product={product}/>
                ))}
            </div>
            </div>
```

```
        );
    };

    export default ProductNew
```

ProductPopular.jsx

```
    import React, {useEffect, useState} from 'react';
    import {useParams} from "react-router-dom";
    import axios from "axios";
    import ProductCard from "./ProductCard.jsx";
    import {server} from "/src/env.js"

    const ProductPopular = () => {
        const {category} = useParams();
        const [products, setProducts] = useState([]);

        useEffect(() => {
            const fetchProducts = async () => {
                try {
                    let response;
                    if (category !== undefined )
                        response = await axios.get(server +
`/products/${category}`);
                    else
                        response = await axios.get(server +
`/products/all`)
                    setProducts(response.data);
                } catch (error) {
                    console.log(error);
                }
            };

            fetchProducts();
        }, [category]);
```

```
        return (
            <div className='mt-[7%]'>
                <span            className='font-bold            text-
[24px]'>РџPsPїCґP»СЏСЂPSC– C,PsPIP°CЂPë</span>
                <div  className="mx-auto  w-[1110px]  gap-4  my-10  grid
grid-cols-4 ">
                {products.map((product, index) => (
                    index   <   4   &&   <ProductCard   key={product.id}
product={product}/>
                ))}
            </div>
            </div>

        );
    };

    export default ProductPopular
```

ProductSame.jsx

```
    import React, {useEffect, useState} from 'react';
    import {useParams} from "react-router-dom";
    import axios from "axios";
    import ProductCard from "./ProductCard.jsx";
    import {server} from "/src/env.js"

    const ProductSame = () => {
        const {category} = useParams();
        const [products, setProducts] = useState([]);

        useEffect(() => {
            const fetchProducts = async () => {
                try {
                    let response;
```

```
                    if (category !== undefined )
                        response = await axios.get(server +
`/products/${category}`);
                    else
                        response = await axios.get(server +
`/products/all`)
                    setProducts(response.data);
                } catch (error) {
                    console.log(error);
                }
            };

            fetchProducts();
        }, [category]);

        return (
            <div className='mt-[7%]'>
                <span  className='font-bold  text-[24px]'>РЎС…РѕР¶С-
С,РѕРІР°СЂРё</span>
                <div  className="mx-auto  w-[1110px]  gap-4  my-10  grid
grid-cols-4 ">
                {products.map((product, index) => (
                    index  <  4  &&  <ProductCard  key={product.id}
product={product}/>
                ))}
            </div>
            </div>

        );
    };

    export default ProductSame
```

AdminPage.jsx

```
import {useSelector} from "react-redux";
import {useNavigate} from "react-router-dom";
import {useEffect} from "react";
import {roles} from "../env.js";


export function AdminPage() {
    const {role} = useSelector((state) => state.auth);
    const navigate = useNavigate()
    useEffect(() => {
        if (role !== roles.ADMIN)
            navigate("/")
    })

    return (
        <div className='w-[1110px] mx-auto'>
            <p>ADMINKA</p>
        </div>
    )
}
```

AuthPage.jsx

```
import React from 'react'
import axios from "axios";
import {server} from "../env.js"
import {useDispatch} from "react-redux";
import {useNavigate} from "react-router-dom"
import {saveToken} from "../store/reducers/authReducer.js";

export function AuthPage() {
    const dispatch = useDispatch();
    const navigate = useNavigate();
```

```
        async function handleSubmit(e) {
            e.preventDefault()
            let username = e.target.username.value;
            let password = e.target.password.value;

            await axios.post(server + '/auth/login', {username,
password})
                .then(response => {
                    dispatch(saveToken(response.data));
                    navigate('/');
                })
                .catch(e => {
                    console.log(e.message)
                })
        }

        return (
            <div            className='w-[1110px]            mx-auto'
onSubmit={handleSubmit}>
                <form className='w-min mx-auto my-20'>
                    <label   htmlFor="username"   className='text-md
block'>P›PsPiC–PS</label>
                    <input type="text" name='username' id='username'
placeholder='P›PsPiC–PS'
                           className='border border-black/20 rounded-
[3px] py-1 px-3'
                           required/>

                    <label htmlFor="password" className='text-md block
mt-3'>РџР°СЂPsP»СЊ</label>
                    <input       type="password"       name='password'
id='password' placeholder='РџР°СЂPsP»СЊ'
                           className='border border-black/20 rounded-
[3px] py-1 px-3'
                           required/>
```

```
                    <button  className='button  blue-button  block  mx-
auto w-min mt-3'>
                        PJPIC-PNºC,Pë
                    </button>
                </form>
            </div>
        )
    }
```

CabinetPage.jsx

```
    import React from "react";
    import {Link, useNavigate} from "react-router-dom";
    import {useDispatch} from "react-redux";
    import {removeToken} from "../store/reducers/authReducer.js";

    export function CabinetPage() {
        const dispatch = useDispatch();
        const navigate = useNavigate();

        const data = [
            {
                id: 1,
                number: "148-869",
                date: "26.11.2023",
                link: "silex.com",
                amount: "25000",
            },
            {
                id: 2,
                number: "148-869",
                date: "26.11.2023",
```

```
            link: "silex.com",
            amount: "25000",
        },
        {
            id: 3,
            number: "148-869",
            date: "26.11.2023",
            link: "silex.com",
            amount: "25000",
        },
        {
            id: 4,
            number: "148-869",
            date: "26.11.2023",
            link: "silex.com",
            amount: "25000",
        },
        {
            id: 5,
            number: "148-869",
            date: "26.11.2023",
            link: "silex.com",
            amount: "25000",
        },
    ];

    const handleClickLogOut = () => {
        dispatch(removeToken());
        navigate('/')
    }

    return (
        <div className="w-[1110px] mx-auto">
            <div className="flex w-full my-10">
                <div  className="border  border-black/20  rounded-
[3px] p-5 min-w-[250px]">
```

```
                        <p className="font-[700] text=[16px] leading-
[20px] tracking-[0.15em]">
                            PµP PђP¤P†P›P¬
                        </p>
                        <div className="h-0 border my-2"></div>

                        <Link to="/error"
                              className="font-[400]       text-[14px]
leading-[15px] my-4 flex">
                            <span>PђCЃPsP±PëCЃC,C- PґP°PSC-</span>
                        </Link>
                        <Link to="/error" className="font-[400] text-
[14px] leading-[15px] my-4 flex">
                            <span>PњPsC— PIC-PґPiCґPєPë</span>
                        </Link>
                        <div     className='font-[400]     text-[14px]
leading-[15px] my-4 flex cursor-pointer'
                             onClick={handleClickLogOut}>
                            <span>P’PëP№C,Pë</span>
                        </div>
                    </div>
                    <div className="w-full ml-10">
                        <p         className="font-bold         text-
[24px]">P†CЃC,PsCЂC-CЏ P·P°PjPsPIP»PµPSCЊ</p>
                        <div className="w-full border border-black/20
rounded-[3px] mt-3">
                            <div className='tableHeader flex columns-4
justify-between items-center bg-black/20 p-2 font-bold text-center'>
                                <div                    className="w-
full">PќPsPjPµCЂ</div>
                                <div  className="w-full">P”P°C,P°  C-
C‡P°CЃ</div>
                                <div                    className="w-
full">PµPsCЃPëP»P°PSPSCЏ PSP° P·P°PjPsPIP»PµPSPSCЏ</div>
                                <div      className="w-full">PЎCґPjP°
P·P°PjPsPIP»PµPSPSCЏ</div>
```

```
                                </div>
                                <div className='tableBody'>
                                    {data.map((product) => (
                                        <div               key={product.id}
className='tableRow flex columns-4 justify-between p-2 text-center'>
                                            <div             className="w-
full">{product.number}</div>
                                            <div             className="w-
full">{product.date}</div>
                                                <a     href={product.link}
className='w-full'>{product.link}</a>
                                            <div             className="w-
full">{product.amount}</div>
                                        </div>
                                    ))}
                                </div>
                            </div>
                        </div>
                    </div>
                );
            }

            export default CabinetPage;
```

Серверна частина web-додатку

Dockerfile

```
FROM openjdk:17-jdk-alpine
WORKDIR /server
COPY target/etech-1.0.jar /server
```

```
COPY ./src/main/resources/static/images /server/src/main/resources/static/images
ENTRYPOINT ["java", "-jar", "etech-1.0.jar"]
```

Файл: \server\src\main\java\com\etech\ApplicationConfig.txt

```
package com.etech;

import org.springframework.context.annotation.Configuration;
import org.springframework.lang.NonNull;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class ApplicationConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(@NonNull CorsRegistry registry) {
        WebMvcConfigurer.super.addCorsMappings(registry);
        registry.addMapping("/**")
                .allowedOrigins("http://localhost:5173", "https://etechcli.netlify.app")
                .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
                .allowedHeaders("Content-Type", "Authorization")
                .allowCredentials(true)
                .exposedHeaders("Access-Control-Allow-Origin", "Access-Control-Allow-Methods", "Access-Control-Allow-Headers");
    }
}
```

SecurityConfig.java

```
package com.etech;

import com.etech.security.jwt.JwtConfigurer;
import com.etech.security.jwt.JwtTokenProvider;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import
org.springframework.security.authentication.AuthenticationManager;
import
org.springframework.security.config.annotation.web.builders.HttpSecuri
ty;
import
org.springframework.security.config.annotation.web.configuration.WebSe
curityConfigurerAdapter;
import
org.springframework.security.config.http.SessionCreationPolicy;

@Configuration
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final JwtTokenProvider jwtTokenProvider;

    private final static String ADMIN_ENDPOINT = "/admin/**";
    private final static String AUTHENTICATION_ENDPOINT =
"/auth/login";
    private final static String PRODUCTS_ENDPOINT = "/products/**";
    private final static String CATEGORIES_ENDPOINT =
"/categories/**";
    private final static String IMAGES_ENDPOINT = "/images/**";

    private final static String USERS_ENDPOINT = "/users/**";
    private final static String ORDERS_ENDPOINT = "/orders/**";

    public SecurityConfig(JwtTokenProvider jwtTokenProvider) {
```

```
            this.jwtTokenProvider = jwtTokenProvider;
        }

        @Bean
        @Override
        public    AuthenticationManager    authenticationManagerBean()
throws Exception {
            return super.authenticationManagerBean();
        }

        @Override
        protected void configure(HttpSecurity http) throws Exception {
            http
                    .httpBasic().disable()
                    .csrf().disable()

.sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATE
LESS)
                    .and()
                    .authorizeRequests()
                    .antMatchers(AUTHENTICATION_ENDPOINT,
PRODUCTS_ENDPOINT,
                            CATEGORIES_ENDPOINT,      IMAGES_ENDPOINT,
ORDERS_ENDPOINT).permitAll()
                    .antMatchers(USERS_ENDPOINT).hasRole("USER")
                    //.antMatchers().hasRole("MANAGER")
                    .antMatchers(ADMIN_ENDPOINT).hasRole("ADMIN")
                    .anyRequest().authenticated()
                    .and()
                    .apply(new JwtConfigurer(jwtTokenProvider));
        }
    }
```

ServerApplication.java

```
package com.etech;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;

@SpringBootApplication
public class ServerApplication {

    public static void main(String[] args) {
        SpringApplication.run(ServerApplication.class, args);
    }

    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }
}
```

AuthenticationController.java

```
package com.etech.controller;

import com.etech.model.AuthenticationRequest;
import com.etech.model.User;
import com.etech.security.jwt.JwtTokenProvider;
import com.etech.service.UserService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
```

```
    import
org.springframework.security.authentication.AuthenticationManager;
    import
org.springframework.security.authentication.BadCredentialsException;
    import
org.springframework.security.authentication.UsernamePasswordAuthentica
tionToken;
    import org.springframework.security.core.AuthenticationException;
    import
org.springframework.security.core.userdetails.UsernameNotFoundExceptio
n;
    import org.springframework.web.bind.annotation.PostMapping;
    import org.springframework.web.bind.annotation.RequestBody;
    import org.springframework.web.bind.annotation.RequestMapping;
    import org.springframework.web.bind.annotation.RestController;

    import java.util.HashMap;
    import java.util.Map;

    @RestController
    @RequestMapping("/auth/")
    public class AuthenticationController {
        private final AuthenticationManager authenticationManager;

        private final JwtTokenProvider jwtTokenProvider;

        private final UserService userService;

        @Autowired
        public          AuthenticationController(AuthenticationManager
authenticationManager,  JwtTokenProvider  jwtTokenProvider,  UserService
userService) {
            this.authenticationManager = authenticationManager;
            this.jwtTokenProvider = jwtTokenProvider;
            this.userService = userService;
        }
```

```
        @PostMapping("login")
        public ResponseEntity<?> login(@RequestBody
AuthenticationRequest requestDto) {
            try {
                String username = requestDto.getUsername();
                authenticationManager.authenticate(new
UsernamePasswordAuthenticationToken(username,
requestDto.getPassword()));
                User user = userService.findByUsername(username);

                if (user == null) {
                    throw new UsernameNotFoundException("User with
username: " + username + " not found");
                }

                String token = jwtTokenProvider.createToken(username,
user.getPermissions());

                Map<Object, Object> response = new HashMap<>();
                response.put("username", username);
                response.put("lastname", user.getLastname());
                response.put("firstname", user.getFirstname());
                response.put("middlename", user.getMiddlename());
                response.put("email", user.getEmail());
                response.put("token", token);
                response.put("role",
user.getPermissions().getPermission());

                return ResponseEntity.ok(response);
            } catch (AuthenticationException e) {
                throw new BadCredentialsException("Invalid username or
password");
            }
        }
    }
```

CategoryController.java

```
package com.etech.controller;

import com.etech.model.Category;
import com.etech.repository.CategoryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("categories")
public class CategoryController {
    @Autowired
    private CategoryRepository categoryRepository;

    @GetMapping("all")
    public List<Category> getCategoryList(
            @RequestParam(name = "title", required = false) String
title) {
        if (title == null || title.isBlank())
            return
categoryRepository.findAll(Sort.by(Sort.Direction.ASC, "id"));
        else
            return
categoryRepository.findAllByTitleContainsIgnoreCase(title);
    }

    @GetMapping("{id}")
    public Category getCategoryById(
            @PathVariable Long id) {
        return categoryRepository.findById(id).orElse(null);
    }
```

CommentController.java

```
package com.etech.controller;

import com.etech.model.Comment;
import com.etech.repository.CommentRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.List;

@RestController
@RequestMapping("comments")
public class CommentController {
    @Autowired
    private CommentRepository commentRepository;

    @GetMapping("all")
    public List<Comment> getCommentList() {
        return
commentRepository.findAll(Sort.by(Sort.Direction.ASC, "id"));
    }
}
```

ImageController.java

```
package com.etech.controller;

import lombok.Getter;
import lombok.RequiredArgsConstructor;
import lombok.Setter;
```

```
import org.springframework.beans.factory.annotation.Value;
import org.springframework.core.io.Resource;
import org.springframework.core.io.UrlResource;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
import java.util.Objects;

@RestController
@Getter @Setter
@RequiredArgsConstructor
@RequestMapping("/images")
public class ImageController {
    @Value("${image.upload.dir.products}")
    private String productsUploadDir;
    @Value("${image.upload.dir.categories}")
    private String categoriesUploadDir;

    @GetMapping("product/{fileName}")
    public                                  ResponseEntity<Resource>
downloadProductImage(@PathVariable("fileName") String fileName) {
        return downloadImage(fileName, productsUploadDir);
    }

    @GetMapping("category/{fileName}")
```

```
    public                                          ResponseEntity<?>
downloadCategoryImage(@PathVariable("fileName") String fileName) {
        return downloadImage(fileName, categoriesUploadDir);
    }

    @PostMapping("product")
    public                                          ResponseEntity<?>
uploadProductImage(@RequestParam("file") MultipartFile file) {
        return uploadImage(file, productsUploadDir);
    }

    @PostMapping("category")
    public                                          ResponseEntity<?>
uploadCategoryImage(@RequestParam("file") MultipartFile file) {
        return uploadImage(file, categoriesUploadDir);
    }

    private  ResponseEntity<?>  uploadImage(MultipartFile  file,
String uploadDir) {
        try {
            String                    fileName                    =
StringUtils.cleanPath(Objects.requireNonNull(file.getOriginalFilename(
)));

            String contentType = file.getContentType();
            assert contentType != null;
            if  (!contentType.equals(MediaType.IMAGE_PNG_VALUE)  &&
!contentType.equals("image/svg+xml")) {
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("РќРµРґРѕРїСѓСЋ,Ре
РјРёРѕ С,РёРї С„Р°РѕР»Сѓ. Р”РѕРїСѓСЋ,РёРјС– С,С–Р»СЊРєРё PNG С,Р° SVG
С„Р°РѕР»Ре");
            }

            Path filePath = Paths.get(uploadDir, fileName);
            Files.createDirectories(filePath.getParent());
```

```
                Files.copy(file.getInputStream(),           filePath,
StandardCopyOption.REPLACE_EXISTING);

                return ResponseEntity.ok().build();
            } catch (IOException e) {
                return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
            }
        }

        private     ResponseEntity<Resource>     downloadImage(String
fileName, String downloadDir) {
            try {
                Path                    filePath                    =
Paths.get(downloadDir).resolve(fileName);
                Resource resource = new UrlResource(filePath.toUri());

                if (resource.exists()) {
                    String                contentType                =
Files.probeContentType(filePath);
                    if (contentType == null) {
                        contentType = "application/octet-stream";
                    }

                    return ResponseEntity.ok()
                            .header(HttpHeaders.CONTENT_TYPE,
contentType)
                            .body(resource);
                }
                return ResponseEntity.notFound().build();
            } catch (IOException e) {
                return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).build();
            }
        }
    }
```

OrderController.java

```
package com.etech.controller;

import com.etech.model.Order;
import com.etech.model.OrderStatus;
import com.etech.repository.OrderRepository;
import com.etech.repository.OrderStatusRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("orders")
public class OrderController {
    @Autowired
    private OrderRepository orderRepository;
    @Autowired
    private OrderStatusRepository orderStatusRepository;
    private static final Long ORDER_STATUS_ACCEPTED = 1L;

    @GetMapping("all")
    public List<Order> getOrderList() {
        return orderRepository.findAll(Sort.by(Sort.Direction.ASC,
"id"));
    }

    @GetMapping("{id}")
    public ResponseEntity<Order> getOrderById(@PathVariable(name =
"id") Long id) {
        Order result = orderRepository.findById(id).orElse(null);
```

```
            if (result == null)
                return new ResponseEntity<>(HttpStatus.NO_CONTENT);

            return new ResponseEntity<>(result, HttpStatus.OK);
        }

        @PostMapping("/order")
        public ResponseEntity<?> addOrder(@RequestBody Order order) {
            if (order.getFirstname() == null || order.getLastname() ==
null
                    ||     order.getMiddlename()     ==     null     ||
order.getPhone() == null)
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("РќРµРґРѕСЃС‚Р°С‚РѕС
ЧЬРѕ РґР°РЅРёС… РґР»СЏ РѕС„РѕСЂРјР»РµРЅРёСЏ Р·Р°СЏРІРєРё");

            if      (order.getProductList()      ==      null      ||
order.getProductList().size() == 0) {
                return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Р’С–РґСЃСѓС‚РЅС
ЃС‚СЊ С‚РѕРІР°СЂС–РІ Р·Р°СЏРІРєРё");
            }

            OrderStatus                    status                    =
orderStatusRepository.findById(ORDER_STATUS_ACCEPTED).orElse(null);
            if (status == null)
                return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("РЎС‚Р°С‚
СѓСЃ Р·Р°СЏРІРєРё РЅРµ Р·РЅР°Р№РґРµРЅРѕ");

            order.setOrderStatus(status);
            orderRepository.save(order);
            Order                    foundOrder                    =
orderRepository.findFirstByEmailOrderByCreateDateDesc(order.getEmail()
);
```

```
            return ResponseEntity.ok().body(foundOrder);
        }
    }
```

OrderStatusController.java

```
    package com.etech.controller;

    import com.etech.model.OrderStatus;
    import com.etech.repository.OrderStatusRepository;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.data.domain.Sort;
    import org.springframework.web.bind.annotation.GetMapping;
    import org.springframework.web.bind.annotation.RequestMapping;
    import org.springframework.web.bind.annotation.RestController;

    import java.util.List;

    @RestController
    @RequestMapping("order-statuses")
    public class OrderStatusController {
        @Autowired
        private OrderStatusRepository orderStatusRepository;

        @GetMapping("all")
        public List<OrderStatus> getOrderStatusList() {
            return
orderStatusRepository.findAll(Sort.by(Sort.Direction.ASC, "id"));
        }
    }
```

PermissionController.java

```
package com.etech.controller;

public class PermissionController {
}
```

ProductController.java

```
package com.etech.controller;

import com.etech.model.Product;
import com.etech.repository.ProductRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("products")
public class ProductController {

    private final static short DISCOUNT = 30;

    @Autowired
    private ProductRepository productRepository;

    @GetMapping("all")
    public List<Product> getProductList(
            @RequestParam(name = "title", required = false) String
title,
            @RequestParam(name = "producer", required = false)
String producer) {
        if ((title == null || title.isBlank()) &&
```

```
                    (producer == null || producer.isBlank()))
                return
productRepository.findAll(Sort.by(Sort.Direction.ASC, "id"));
            else
                return
productRepository.findAllByTitleContainsIgnoreCaseOrProducerContainsIg
noreCase(title, producer);
        }

        @GetMapping("id={id}")
        public Product getProductById(
                @PathVariable Long id) {
            return productRepository.findById(id).orElse(null);
        }

        @GetMapping("category={category}")
        public List<Product> getProductListByCategoryTitle(
                @PathVariable String category) {
            return
productRepository.findAllByCategory_Title(category);
        }

        @GetMapping("offers")
        public List<Product> getOfferProductsWithImages() {
            return
productRepository.findByDiscountGreaterThanEqual(DISCOUNT);
        }
    }
```

UserController.java

```
    package com.etech.controller;

    import com.etech.model.User;
    import com.etech.repository.UserRepository;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Sort;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("users")
public class UserController {

    @Autowired
    private UserRepository userRepository;

    @GetMapping("all")
    public List<User> getUserList() {
        return  userRepository.findAll(Sort.by(Sort.Direction.ASC,
"id"));
    }
}
```

AuthenticationRequest.java

```
package com.etech.model;

import lombok.Data;

@Data
public class AuthenticationRequest {
    private String username;
    private String password;
}
```

Category .java

```
package com.etech.model;

import lombok.Data;
import lombok.RequiredArgsConstructor;

import javax.persistence.*;
import java.util.Date;
import java.util.List;

@Entity
@RequiredArgsConstructor
@Data
@Table(name = "categories")
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @OneToMany
    @JoinTable(name = "child_category_list",
            joinColumns = @JoinColumn(name = "parent_category_id"),
            inverseJoinColumns = @JoinColumn(name = "child_category_id"))
    private List<Category> childCategories;
    private String title;
    private Date createDate;
}
```

Comment.java

```
package com.etech.model;

import lombok.Data;
```

```
import lombok.RequiredArgsConstructor;

import javax.persistence.*;
import java.util.Date;

@Entity
@RequiredArgsConstructor
@Data
@Table(name = "comments")
public class Comment {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String comment;
    private short rating;
    @ManyToOne
    @JoinColumn(name = "user_id")
    private User user;
    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product product;
    private Date createDate;
}
```

Order.java

```
package com.etech.model;

import lombok.Data;
import lombok.RequiredArgsConstructor;

import javax.persistence.*;
import java.util.Date;
import java.util.List;
```

```
@Entity
@RequiredArgsConstructor
@Data
@Table(name = "orders")
public class Order {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String lastname;
    private String firstname;
    private String middlename;
    private String email;
    private String phone;
    private String comment;
    @ManyToOne
    @JoinColumn(name = "order_status_id")
    private OrderStatus orderStatus;

    @ManyToMany
    @JoinTable(name = "product_list",
            joinColumns = @JoinColumn(name = "order_id"),
            inverseJoinColumns = @JoinColumn(name = "product_id"))
    private List<Product> productList;

    private Date createDate;
}
```

OrderStatus.java

```
package com.etech.model;

import lombok.Data;
import lombok.RequiredArgsConstructor;
```

```
import javax.persistence.*;

@Entity
@RequiredArgsConstructor
@Data
@Table(name = "order_status")
public class OrderStatus {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String title;
}
```

Permissions.java

```
package com.etech.model;

import lombok.Data;
import lombok.RequiredArgsConstructor;

import javax.persistence.*;

@Entity
@RequiredArgsConstructor
@Data
@Table(name = "permissions")
public class Permissions {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private String permission;
}
```

Product.java

```
package com.etech.model;

import lombok.Data;
import lombok.RequiredArgsConstructor;

import javax.persistence.*;
import java.math.BigDecimal;

@Entity
@RequiredArgsConstructor
@Data
@Table(name = "products")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @ManyToOne
    private Category category;
    private String title;
    private BigDecimal price;
    private String producer;
    private short discount;
    private short amount;
    private String description;
    private String imgPath;
}
```

ProductList.java

```
package com.etech.model;

import lombok.Data;
```

```
import lombok.RequiredArgsConstructor;

import javax.persistence.*;

@Entity
@RequiredArgsConstructor
@Data
@Table(name = "product_list")
public class ProductList {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @OneToOne
    @JoinColumn(name = "order_id")
    private Order order;
    @ManyToOne
    @JoinColumn(name = "product_id")
    private Product product;
}
```

User.java

```
package com.etech.model;

import javax.persistence.*;
import lombok.*;

import java.util.Date;

@Entity
@RequiredArgsConstructor
@Data
@Table(name = "users")
public class User {
    @Id
```

```
        @GeneratedValue(strategy = GenerationType.IDENTITY)
        private Long id;
        private String username;
        private String lastname;
        private String firstname;
        private String middlename;
        private String email;
        private String phone;
        private String password;
        @ManyToOne
        @JoinColumn(name = "permission_id")
        private Permissions permissions;
        private Date createDate;
    }
```

CategoryRepository.java

```
    package com.etech.repository;

    import com.etech.model.Category;
    import org.springframework.data.jpa.repository.JpaRepository;
    import org.springframework.data.repository.CrudRepository;
    import
org.springframework.data.repository.PagingAndSortingRepository;
    import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

    import java.util.List;

    @RepositoryRestResource
    public interface CategoryRepository extends
PagingAndSortingRepository<Category, Long>,
            CrudRepository<Category, Long>, JpaRepository<Category,
Long> {
        List<Category> findAllByTitleContainsIgnoreCase(String title);
```

CommentRepository.java

```
package com.etech.repository;

import com.etech.model.Comment;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface CommentRepository extends
PagingAndSortingRepository<Comment, Long>,
        CrudRepository<Comment, Long>, JpaRepository<Comment,
Long> {
}
```

OrderRepository.java

```
package com.etech.repository;

import com.etech.model.Order;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

import java.util.List;


@RepositoryRestResource
```

```
public interface OrderRepository extends
PagingAndSortingRepository<Order, Long>,
        CrudRepository<Order, Long>, JpaRepository<Order, Long> {
    List<Order> findAllByEmailOrderByCreateDateAsc(String email);
    Order findFirstByEmailOrderByCreateDateDesc(String email);
}
```

OrderStatusRepository.java

```
package com.etech.repository;

import com.etech.model.OrderStatus;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface OrderStatusRepository extends
PagingAndSortingRepository<OrderStatus, Long>,
        CrudRepository<OrderStatus, Long>,
JpaRepository<OrderStatus, Long> {
}
```

PermissionsRepository.java

```
package com.etech.repository;

import com.etech.model.OrderStatus;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository;
```

```
public interface PermissionsRepository extends
PagingAndSortingRepository<OrderStatus, Long>,
        CrudRepository<OrderStatus, Long>,
JpaRepository<OrderStatus, Long> {
}
```

ProductRepository.java

```
package com.etech.repository;

import com.etech.model.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

import java.util.List;

@RepositoryRestResource
public interface ProductRepository extends
PagingAndSortingRepository<Product, Long>,
        CrudRepository<Product, Long>, JpaRepository<Product,
Long> {
    List<Product>
findAllByTitleContainsIgnoreCaseOrProducerContainsIgnoreCase(String
title, String producer);
    List<Product> findAllByCategory_Title(String category);

    List<Product> findByDiscountGreaterThanEqual(short discount);
}
```

UserRepository.java

```
package com.etech.repository;

import com.etech.model.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import
org.springframework.data.repository.PagingAndSortingRepository;
import org.springframework.data.repository.query.Param;
import
org.springframework.data.rest.core.annotation.RepositoryRestResource;

@RepositoryRestResource
public interface UserRepository extends
PagingAndSortingRepository<User, Long>,
        CrudRepository<User, Long>, JpaRepository<User, Long> {

    User findByEmail(@Param("email") String email);
    User findByUsername(@Param("username") String username);
}
```

JwtUserDetailsService.java

```
package com.etech.security;

import com.etech.model.User;
import com.etech.security.jwt.JwtUserFactory;
import com.etech.service.UserService;
import org.springframework.security.core.userdetails.UserDetails;
import
org.springframework.security.core.userdetails.UserDetailsService;
import
org.springframework.security.core.userdetails.UsernameNotFoundExceptio
n;
```

```
import org.springframework.stereotype.Service;

@Service
public class JwtUserDetailsService implements UserDetailsService {

    private final UserService userService;

    public JwtUserDetailsService(UserService userService) {
        this.userService = userService;
    }

    @Override
    public UserDetails loadUserByUsername(String username) throws
UsernameNotFoundException {
        User user = userService.findByUsername(username);

        if (user == null) {
            throw    new    UsernameNotFoundException("User    with
username: " + username + " not found.");
        }

        return JwtUserFactory.create(user);
    }
}
```

JwtAuthenticationException.java

```
package com.etech.security.jwt;

import org.springframework.security.core.AuthenticationException;

public       class       JwtAuthenticationException       extends
AuthenticationException {
    public JwtAuthenticationException(String msg, Throwable cause)
{
```

```
        super(msg, cause);
    }

    public JwtAuthenticationException(String msg) {
        super(msg);
    }
}
```

JwtConfigurer.java

```
package com.etech.security.jwt;

import
org.springframework.security.config.annotation.SecurityConfigurerAdapt
er;
import
org.springframework.security.config.annotation.web.builders.HttpSecuri
ty;
import
org.springframework.security.web.DefaultSecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthen
ticationFilter;


public class JwtConfigurer extends
SecurityConfigurerAdapter<DefaultSecurityFilterChain, HttpSecurity> {
    private final JwtTokenProvider jwtTokenProvider;

    public JwtConfigurer(JwtTokenProvider jwtTokenProvider) {
        this.jwtTokenProvider = jwtTokenProvider;
    }

    @Override
    public void configure(HttpSecurity httpSecurity) {
```

```
            JwtTokenFilter jwtTokenFilter = new
JwtTokenFilter(jwtTokenProvider);
            httpSecurity.addFilterBefore(jwtTokenFilter,
UsernamePasswordAuthenticationFilter.class);
        }
    }
```

JwtTokenFilter.java

```
    package com.etech.security.jwt;

    import org.springframework.security.core.Authentication;
    import
org.springframework.security.core.context.SecurityContextHolder;
    import org.springframework.web.filter.GenericFilterBean;

    import javax.servlet.FilterChain;
    import javax.servlet.ServletException;
    import javax.servlet.ServletRequest;
    import javax.servlet.ServletResponse;
    import javax.servlet.http.HttpServletRequest;
    import java.io.IOException;

    public class JwtTokenFilter extends GenericFilterBean {

        private final JwtTokenProvider jwtTokenProvider;

        public JwtTokenFilter(JwtTokenProvider jwtTokenProvider) {
            this.jwtTokenProvider = jwtTokenProvider;
        }

        @Override
        public void doFilter(ServletRequest servletRequest,
ServletResponse servletResponse, FilterChain filterChain) throws
IOException, ServletException {
```

```
            String                          token                          =
jwtTokenProvider.resolveToken((HttpServletRequest) servletRequest);

            if (token != null && jwtTokenProvider.validateToken(token))
{
                Authentication            authentication            =
jwtTokenProvider.getAuthentication(token);

                if (authentication != null) {

SecurityContextHolder.getContext().setAuthentication(authentication);
                }
            }

            filterChain.doFilter(servletRequest, servletResponse);
        }
    }
```

JwtTokenProvider.java

```
    package com.etech.security.jwt;

    import com.etech.model.Permissions;
    import io.jsonwebtoken.*;
    import org.springframework.beans.factory.annotation.Value;
    import
org.springframework.security.authentication.UsernamePasswordAuthentica
tionToken;
    import org.springframework.security.core.Authentication;
    import org.springframework.security.core.userdetails.UserDetails;
    import
org.springframework.security.core.userdetails.UserDetailsService;
    import org.springframework.stereotype.Component;

    import javax.annotation.PostConstruct;
```

```
import javax.servlet.http.HttpServletRequest;
import java.util.Base64;
import java.util.Date;

@Component
public class JwtTokenProvider {
    @Value("${jwt.token.secret}")
    private String secret;
    @Value("${jwt.token.expired}")
    private Long validityMs;
    private final UserDetailsService userDetailsService;

    public JwtTokenProvider(UserDetailsService userDetailsService)
{
        this.userDetailsService = userDetailsService;
    }

    @PostConstruct
    protected void init() {
        secret                                                    =
Base64.getEncoder().encodeToString(secret.getBytes());
    }

    public   String   createToken(String   username,   Permissions
permissions) {
        Claims claims = Jwts.claims().setSubject(username);
        claims.put("permission", getPermissionName(permissions));

        Date now = new Date();
        Date validity = new Date(now.getTime() + validityMs);

        return Jwts.builder()
                .setClaims(claims)
                .setIssuedAt(now)
                .setExpiration(validity)
                .signWith(SignatureAlgorithm.HS256, secret)
```

```
                .compact();
        }

        public Authentication getAuthentication(String token) {
            UserDetails userDetails =
this.userDetailsService.loadUserByUsername(getUsername(token));
            return new
UsernamePasswordAuthenticationToken(userDetails, "",
userDetails.getAuthorities());
        }

        public String getUsername(String token) {
            return
Jwts.parser().setSigningKey(secret).parseClaimsJws(token).getBody().ge
tSubject();
        }

        public String resolveToken(HttpServletRequest req) {
            String bearerToken = req.getHeader("Authorization");
            if (bearerToken != null &&
bearerToken.startsWith("Bearer_")) {
                return bearerToken.substring(7);
            }
            return null;
        }

        public boolean validateToken(String token) {
            try {
                Jws<Claims> claims =
Jwts.parser().setSigningKey(secret).parseClaimsJws(token);
                return !claims.getBody().getExpiration().before(new
Date());
            } catch (JwtException | IllegalArgumentException e) {
                throw new JwtAuthenticationException("JWT token is
expired or invalid");
            }
```

```
        }

        private String getPermissionName(Permissions permissions) {
            return permissions.getPermission();
        }
    }
```

JwtUser.java

```
    package com.etech.security.jwt;

    import org.springframework.security.core.GrantedAuthority;
    import
org.springframework.security.core.authority.SimpleGrantedAuthority;
    import org.springframework.security.core.userdetails.UserDetails;

    import java.util.Collection;
    import java.util.Collections;
    import java.util.Date;

    public class JwtUser implements UserDetails {

        private final Long id;
        private final String username;
        private final String email;
        private final String password;

        private final SimpleGrantedAuthority authorities;
        private final Date createDate;

        public JwtUser(Long id, String username, String email, String
password,
                       SimpleGrantedAuthority    authorities,    Date
createDate) {
            this.id = id;
            this.username = username;
```

```
        this.email = email;
        this.password = password;
        this.authorities = authorities;
        this.createDate = createDate;
    }

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities()
{
        return Collections.singleton(authorities);
    }

    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public String getUsername() {
        return username;
    }

    @Override
    public boolean isAccountNonExpired() {
        return true;
    }

    @Override
    public boolean isAccountNonLocked() {
        return true;
    }

    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
```

```
    @Override
    public boolean isEnabled() {
        return true;
    }

    public Long getId() {
        return id;
    }

    public String getEmail() {
        return email;
    }

    public Date getCreateDate() {
        return createDate;
    }
}
```

JwtUserFactory.java

```
package com.etech.security.jwt;

import com.etech.model.Permissions;
import com.etech.model.User;
import
org.springframework.security.core.authority.SimpleGrantedAuthority;

public final class JwtUserFactory {
    public JwtUserFactory() {
    }

    public static JwtUser create(User user) {
        return new JwtUser(
                user.getId(),
```

```
                    user.getUsername(),
                    user.getEmail(),
                    user.getPassword(),
                    mapToGrantedAuthorities(user.getPermissions()),
                    user.getCreateDate()
            );
        }

        private static SimpleGrantedAuthority
mapToGrantedAuthorities(Permissions permissions) {
            return new
SimpleGrantedAuthority(permissions.getPermission());
        }
    }
```

UserService.java

```
package com.etech.service;

import com.etech.model.User;

import java.util.List;

public interface UserService {
    User register(User user);
    List<User> getAll();
    User findByUsername(String username);
    User findById(Long id);
    void delete(Long Id);
}
```

UserServiceImpl.java

```
package com.etech.service;
```

```
import com.etech.model.User;
import com.etech.repository.UserRepository;
import
org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.stereotype.Service;

import java.util.List;

@Service
public class UserServiceImpl implements UserService {

    private final UserRepository userRepository;
    private final BCryptPasswordEncoder passwordEncoder;

    public     UserServiceImpl(UserRepository     userRepository,
BCryptPasswordEncoder passwordEncoder) {
        this.userRepository = userRepository;
        this.passwordEncoder = passwordEncoder;
    }

    @Override
    public User register(User user) {

user.setPassword(passwordEncoder.encode(user.getPassword()));

        return userRepository.save(user);
    }

    @Override
    public List<User> getAll() {
        return userRepository.findAll();
    }

    @Override
    public User findByUsername(String username) {
        return userRepository.findByUsername(username);
```

```
        }

        @Override
        public User findById(Long id) {
            return userRepository.findById(id).orElse(null);
        }

        @Override
        public void delete(Long id) {
            userRepository.deleteById(id);
        }
    }
```

application.properties

```
    spring.data.rest.base-path=/api
    image.upload.dir.products=src/main/resources/static/images/produc
ts
    image.upload.dir.categories=src/main/resources/static/images/cate
gories

    spring.jpa.database-
platform=org.hibernate.dialect.PostgreSQLDialect
    spring.datasource.driver-class-name=org.postgresql.Driver
    spring.jpa.generate-ddl=false

    spring.datasource.hikari.minimum-idle=3
    spring.main.allow-bean-definition-overriding=true

    jwt.token.secret=jwtsilexshop
    jwt.token.expired=3600000
```