МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Використання React та Node.js для створення віртуального майданчика торгівлі та бартерного обміну**

Виконав: студент 4 курсу, групи 6.1219-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)

В. В. Булигін
(ініціали та прізвище)

Керівник доцент, к.т.н. О. М. Міхайлуца
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Рецензент директор ТОВ «Дісітел»
П.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма ___Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри_____Т.В. Критська_
"_01_" ___березня__ 2023_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Булигіну Володимиру Володимировичу
(прізвище, ім'я, по батькові)

1.Тема роботи _Використання React та Node.js для створення віртуального майданчика торгівлі та бартерного обміну

керівник роботи ___Міхайлуца Олена Миколаївна, доцент, к.т.н._____
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 29.12.2022 № 1893-с

2. Строк подання студентом кваліфікаційної роботи_____14.06.2023_____
3. Вихідні дані бакалаврської роботи
- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми створення веб-додатків;
- створення програмного продукту та його опис;
- дослідження поставленої проблеми та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
_____слайдів презентації

6. Консультанти розділів бакалаврської роботи

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

7. Дата видачі завдання 01.03.2023

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів кваліфікаційної роботи	Примітка
1	Аналіз предметної області	22.04.23	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	25.04.23	виконано
3	Аналіз існуючих методів рішення	26.04-30.04.23	виконано
4	Аналіз сучасних технологій для розробки серверної частини	01.05-04.05.23	виконано
5	Аналіз сучасних технологій для розробки користувацької частини	05.05-08.05.23	виконано
6	Узгодження подальших дій з науковим керівником	10.05.23	виконано
7	Проєктування бази даних	12.05.23	виконано
8	Програмна реалізація серверної частини застосунку	14.05-22.05.23	виконано
9	Представлення отриманих результатів науковому керівнику та узгодження плану подальшого дослідження	25.05.23	виконано
10	Реалізація користувацького інтерфейсу	26.05-03.06.23	виконано
11	Перевірка працездатності додатку	04.06.23	виконано
12	Оформлення звіту	05.06-11.06.23	виконано
13	Оформлення презентації. Отримання рецензій від опонентів.	12.06-15.06.23	виконано

Студент _____ Булигін В.В.
(підпис) (прізвище та ініціали)

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

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціал

АНОТАЦІЯ

Сторінок: 86

Рисунків: 22

Лістингів: 11

Джерел: 16

Булигін В. В. Використання React та Node.js для створення віртуального майданчика торгівлі та бартерного обміну: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник О. М. Міхайлуца. Запоріжжя: ЗНУ, 2023. 86с.

В нашому сучасному контексті з урахуванням наявних умов, добре спроєктований веб-додаток електронної комерції має велику цінність, оскільки він пропонує зручність та ефективність для користувачів, які проявляють інтерес до цієї галузі. Оптимізація алгоритмів та застосування передових підходів та технологій є одними з найбільш поширених та очевидних способів підвищення ефективності та зручності у бізнес-процесах.

Головною метою та завданням дослідження є отримання інформації та аналізу особливостей та трендів розробки веб-додатків у сфері електронної комерції. Встановлено, що використання сучасних підходів та концепцій у таких сервісах не лише вкрай зручне та перспективне, але також є суттєво вагомим напрямом.

Дослідження та порівняльний аналіз існуючих підходів та технологій розробки додатку надали необхідну інформацію і дозволили виявити ключові фактори. За результатами аналізу був розроблений веб-додаток віртуального майданчика для торгівлі та бартерного обміну, з урахуванням отриманих знань та застосуванням сучасних підходів розробки програмного забезпечення та технологій. Цей тип застосунку може задовольняти потреби як користувачів, так і компаній, що прагнуть розпочати діяльність в галузі електронної комерції з метою підвищення продуктивності своєї роботи.

Ключові слова: Електрона комерція, веб-додаток, Node.js , React.

ABSTRACT

Pages: 86

Drawings: 22

Listings: 11

Sources: 16

Bulyhin V. V. Using React and Node.js to create a virtual platform for trade and barter exchange: bachelor's thesis in specialty 121 «Software Engineering» / scientific supervisor O. M. Mikhailutsa. Zaporizhzhia: ZNU, 2023. 86p.

In today's context, a well-designed e-commerce web application is of great value as it offers convenience and efficiency to users who are interested in this industry. Optimising algorithms and applying advanced approaches and technologies are some of the most common and obvious ways to increase efficiency and convenience in business processes.

The main purpose and objective of the study is to obtain information and analyse the features and trends in the development of web applications in the field of e-commerce. It is established that the use of modern approaches and concepts in such services is not only extremely convenient and promising, but also a significant trend.

The research and comparative analysis of existing approaches and technologies for developing the application provided the necessary information and allowed us to identify key factors. Based on the results of the analysis, a web application of a virtual platform for trade and barter exchange was developed, taking into account the knowledge gained and applying modern software development approaches and technologies. This type of application can meet the needs of both users and companies looking to start an e-commerce business to increase their productivity.

Keywords: *E-commerce, web application, Node.js, React*.

ЗМІСТ

ВСТУП

Актуальність теми

Завдяки стрімкому розвитку технологій і їх впровадженню в наше повсякденне життя, перехід більшості видів діяльності в інтернет став необхідністю та невід'ємною частиною сучасного суспільства. Цей розвиток зумовлений великою кількістю факторів, включаючи покращення обчислювальної потужності комп'ютерів, збільшення швидкості Інтернет-з'єднань, зменшення розміру електронних пристроїв та підвищення їх енергоефективності.

Зростаюча кількість інтернет-користувачів у всьому світі є безперечним свідченням того, що доступ до інтернету став масовим явищем. Люди використовують Інтернет для різних цілей, таких як спілкування, пошук інформації, покупки товарів та послуг, розваги, освіта та багато іншого. Це призводить до зростання попиту на якісні й надійні інтернет-послуги.

Таке розширення впливу технологій сприяє переходу компаній та послуг до Інтернету, що є важливим кроком у розвитку бізнесу та досягненні успіху. Інтернет став найбільш доступним та широко використовуваним засобом комунікації, що суттєво впливає на розвиток електронної комерції.

Електронна комерція (e-commerce) — процес купівлі та продажу товарів або послуг через Інтернет, що дозволяє компаніям пропонувати нові продукти та послуги, вдосконалювати існуючі технології та способи продажу, розширювати аудиторію та збільшувати прибутки.

Крім цього, такі сервіси надають безліч можливостей для реклами та просування товарів і послуг. Компанії можуть використовувати електронну комерцію як платформу для ефективного просування своїх пропозицій та досягання більш широкої аудиторії. Що дає можливість максимально використовувати потенціал електронної комерції з метою збільшення продажів та популярності продуктів і послуг. За допомогою електронної комерції, компанії можуть використовувати соціальні медіа для реклами своїх

товарів та послуг, а також співпрацювати з іншими платформами та ресурсами для досягнення більшого кола потенційних клієнтів.

Отже, розвиток електронної комерції та маркетингова орієнтація бізнес-структур робить досить актуальною темою, створення віртуального майданчика торгівлі та бартерного обміну, що буде працювати як веб додаток та надавати необхідний функціонал. Цей додаток буде корисним як для компаній, так і для звичайних користувачів, які бажають активно взаємодіяти у сфері електронної комерції.

Мета дослідження

Мета дослідження полягає у розробці веб-додатку віртуального майданчика торгівлі та бартерного обміну. Що буде ефективно розв'язувати поставлену задачу та надавати необхідний користувачу функціонал.

Завдання дослідження

Дослідити існуючі рішення (аналоги) для електронної комерції в різних галузях бізнесу та виявити їх особливості. Провести детальний аналіз різних підходів та технологій, що використовуються для розробки електронних комерційних систем, та визначити оптимальний варіант для успішної реалізації зручного веб-додатку торгівлі та бартерного обміну.

Об’єкт дослідження

Об’єктом дослідження є процес розробки веб-додатку електронної комерції для торгівлі та обміну товарів між користувачами.

Предмет дослідження

Предметом дослідження є технології створення власного застосунку торгівлі та обміну товарів для задоволення потреб у сфері електронної комерції з метою підвищення ефективності роботи у цій сфері.

Методи дослідження

Теоретичні — аналіз та порівняння технологій створення сервісу електронної комерції. Різновиди класифікації та принципів проєктування веб-додатків торгівлі та бартерного обміну. Порівняльний аналіз існуючих рішень для виділення аспектів підходящих під конкретну ситуацію.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає у тому, що розробка веб-застосунку для торгівлі та бартерного обміну надає різним компаніям та користувача досить зручну у використанні платформу для забезпечення торговельних відносин. Результати дослідження можуть бути використані для розробки віртуального майданчика торгівлі та бартерного обміну, що допоможе розширенню бізнесу та підвищенню економічного розвитку.

Апробація результатів

Результати роботи було представлено на XVI університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2023» [16].

Глосарій

Електрона комерція (*англ. e-commerce*) — процес купівлі та продажу товарів або послуг через Інтернет, яке включає в себе використання електронних платіжних систем, онлайн-маркетплейсів та інших інструментів для проведення торгівлі в мережі.

Віртуальний майданчик — електронна платформа, на якій продавці можуть розміщувати інформацію про свої товари або послуги, а покупці можуть обрати товар за необхідними критеріями.

Конверсія — процес перетворення відвідувачів веб-сайту або потенційних клієнтів на реальних покупців або виконання певної цільової дії.

B2B (або Business-to-Business) — модель бізнесу, в якій комерційні транзакції та взаємодія відбуваються між двома або більше підприємствами,

де одне підприємство надає товари або послуги іншому підприємству, а не безпосередньо кінцевому споживачеві.

B2C (або Business-to-Consumer) — означає комерційні взаємодії між підприємством і кінцевим споживачем, де товари або послуги продаються безпосередньо споживачеві.

Node.js —відкрите середовище виконання JavaScript, яке ґрунтується на двигуні V8, розробленому Google для веб-браузера Chrome та інших, та дозволяє виконувати JavaScript-код на стороні сервера.

JavaScript — високорівнева мова програмування із динамічною типізацією, яка використовується для створення динамічних веб-сторінок та веб-додатків.

TypeScript — мова програмування, розроблена компанією Microsoft, яка надбудовує JavaScript і дозволяє розробникам створювати більш надійні програми, зручні для збереження та розширення, завдяки можливості явно вказувати типи даних.

Nest.js — фреймворк для створення масштабованих та ефективних серверних додатків на Node.js, що базується на концепції модулів та дозволяє організувати код в окремі, автономні блоки, які можуть бути повторно використані в інших додатках.

React.js (або React) — бібліотека JavaScript, розроблена компанією Facebook, що дозволяє розробникам створювати високоякісні, швидкі та масштабовані веб-додатки, використовуючи компонентний підхід для створення користувацьких інтерфейсів.

PostgreSQL (або Postgres) — потужна, об'єктно-реляційна система управління базами даних з відкритим вихідним кодом, яка надає широкий набір функцій для зберігання та обробки структурованих даних, забезпечуючи надійність, масштабованість і гнучкість в роботі з даними.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд літературних джерел

У галузі електронної комерції наявні значні наукові дослідження та джерела, що присвячені огляду та аналізу різних аспектів цієї галузі [1-2]. Зокрема вітчизняні науковці та автори публікації [1] приводять надважливі основні поняття та принципи електронної комерції та Інтернет торгівлі. Відзначається що Електронний бізнес, можна розуміти як здійснення ділової активності через Інтернет. Та що на сьогоднішній день спостерігається зростання цього типу сервісу і поступове перетворення веб-сторінок у повноцінні інтернет-магазини. Це надає бізнесу можливість розширити географію своєї присутності на ринку, швидше реалізувати товари та послуги й отримати додатковий канал для маркетингу і реклами.

Електронна комерція, відома як е-комерція, є одним зі способів здійснення електронного бізнесу. Е-комерція являє собою широкий спектр інтерактивних методів, що використовуються для надання споживачам товарів та послуг. Під е-комерцією розуміють будь-які форми ділових операцій, де сторони взаємодіють за допомогою електронних технологій, а не через фізичний обмін або прямий контакт.

Новітні тренди та перспективи розвитку були розглянуто у праці [2]. Автори підкреслюють що в наслідок пандемії COVID-19, підходи до міжнародного бізнесу радикально змінилися. Світова економіка пройшла глобальну трансформацію, зосереджуючись на цифровізації. Пандемія прискорила використання цифрових технологій компаніями, що змусило їх змінити власні бізнес-моделі для адаптації до нових умов.

Електронна комерція стала надзвичайно важливою для багатьох міжнародних бізнес-суб'єктів і стала єдиним способом збереження або зростання конкурентних позицій на ринку. Модернізація бізнес-процесів за допомогою новітніх інтернет-технологій не тільки створює можливості для

розширення діяльності, але також спонукає до формування нових видів діяльності та суб'єктів бізнесу через створення нового типу інфраструктури.

Як можна побачити на рисунку 1 до 2024 року очікується майже п'ятикратне зростання роздрібних продажів у цій галузі порівняно з 2014 роком. Це свідчить про значне підвищення популярності та прийняття електронної комерції як основного каналу продажу для багатьох бізнесів і споживачів. Тобто цифровізація торгівлі та інших бізнес-процесів призводить до збільшення уваги до електронної комерції при укладанні міжнародних двосторонніх та багатосторонніх угод. Запровадження цифрових рішень та інтернет-технологій відкриває нові можливості для залучення нових клієнтів та розширення географії присутності на ринках, що дозволяє підприємствам ефективно просувати свої товари та послуги у міжнародному масштабі.

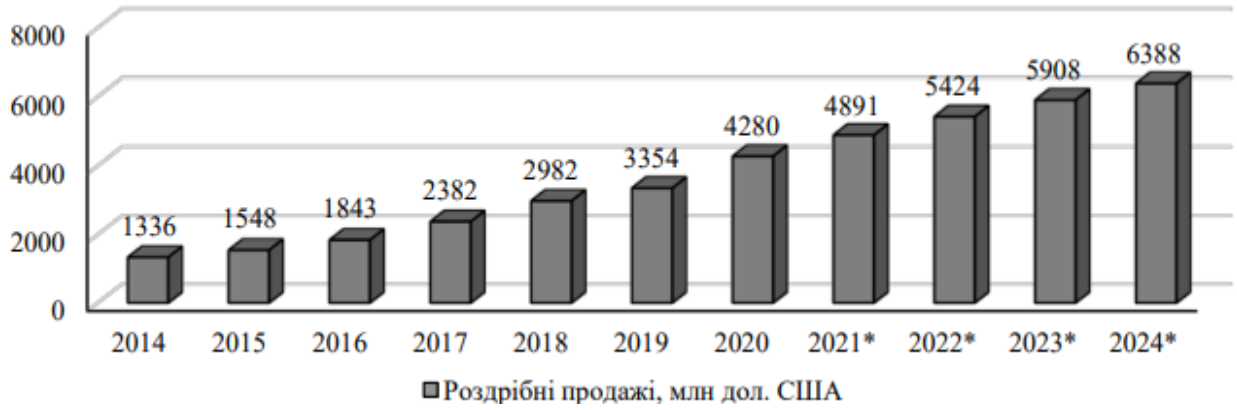

Рис. *1 Роздрібні продажі засобами електронної комерції в глобальному вимірі, млн дол. США*

Усе це свідчить про те, що електронна комерція в міжнародному бізнесі є необхідним і динамічно розвиваючимся напрямком, який впливає на стратегії компаній. Ріст роздрібних продажів в електронній комерції вказує на збільшення популярності онлайн-покупок серед споживачів і важливість присутності бізнесів в цьому сегменті. Компанії пристосовуються до змін, впроваджуючи нові технології, розвиваючи електронні платформи,

модернізуючи маркетингові стратегії та забезпечуючи зручність та безпеку для своїх клієнтів.

1.2 Аналіз програмних продуктів-аналогів

Електрона комерція пройшла довгий шлях який почався в 90-х роках зі з'явленням перших інтернет-магазинів і електронних платіжних систем до сьогодення де з кожним роком її популярність продовжує суттєво зростати. За цей час з'явилося багато сервісів електронної комерції й серед них можна відокремити декілька цікавих сервісів які сприяють розвитку цієї сфери.

OLX

OLX — сайт що являє собою онлайн-маркетплейс де користувачі можуть купувати/продавати або обмінювати товари та послуги між собою. Даний марктеплейс є одним з найбільш популярних в Україні маючи у своєму розпорядженні понад 25 млн користувачів [3].

На сайті доступні різноманітні категорії товарів що дозволяють шукати товари та послуги за необхідною категорією. Також за необхідністю можна звузити коло пошуків за допомогою зручної фільтрації та відсортувати товари за ціною, що дозволить досить зручно та швидко знайти саме той товар який вам потрібен. Ще OLX надає функцію обміну повідомленнями, яка дозволяє користувачам спілкуватися між собою безпосередньо на платформі. Це дозволяє уточнювати деталі, домовлятися про зустрічі або узгоджувати умови продажу чи обміну. має систему оцінок та відгуків, яка допомагає користувачам оцінювати надійність та репутацію інших користувачів. Після успішної угоди користувачі можуть залишати відгуки один про одного, що допомагає іншим користувачам прийняти рішення про покупку. Сайт має систему оцінок та відгуків, яка допомагає користувачам оцінювати надійність та репутацію інших користувачів. Після успішної угоди користувачі можуть залишати відгуки один про одного, що допомагає іншим користувачам прийняти рішення про покупку.

Інтерфейс головної сторінки сайту наведено на відповідному рисунку (див. Рис. 2).

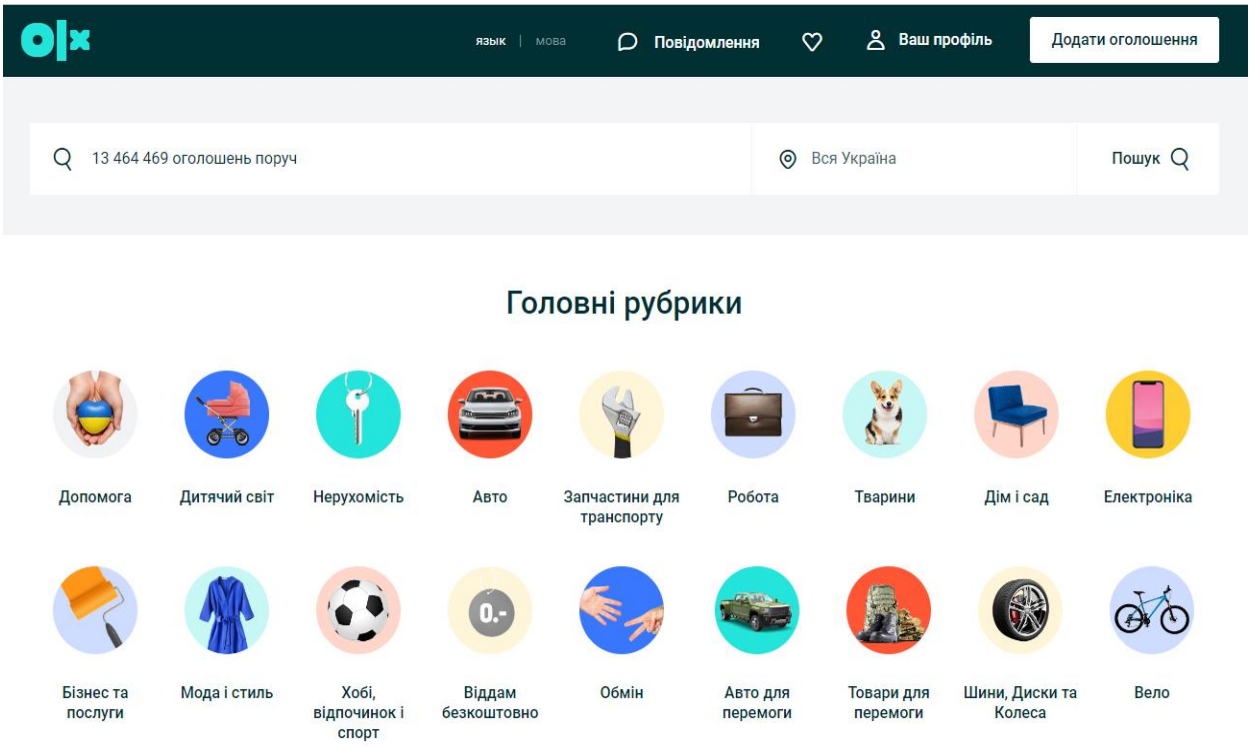

Рис. *2 Інтерфейс головної сторінки сайту OLX*

До переваг цього сайту можна віднести:

1. Інтуїтивно зрозумілий та зручний інтерфейс користувача, який дозволяє легко знайти необхідний товар навіть початківцям.
2. Зручний пошук та фільтрація, які дозволяють користувачам швидко знайти необхідні товари та відсортувати оголошення за багатьма параметрами.
3. Система відгуків користувачів та рейтингу продавців яка дозволяє користувачам залишати відгуки про тих, з ким вони проводили угоди, тим самим допомагаючи іншим користувачам приймати рішення щодо покупки.
4. Зручна вбудована карта що дозволяє подивитися місцезнаходження продавця та товару який він продає.

5. Багато різних категорій товарів. OLX має широкий спектр різних категорій товарів і послуг, від одягу до техніки, що робить платформу цікавим для широкої аудиторії.
6. Користувачі можуть знайти надзвичайно рідкісні товари, які можуть бути недоступні у звичайних магазинах.
7. Вражаючий рівень адаптивності, що дозволяє добре пристосуватися під екрани різних розмірів.

До недоліків цього сайту можна віднести:

1. На сайті можна зіткнутися з надмірною кількістю реклами, що може бути незручним для користувачів.
2. Немає гарантії безпеки платежів, тому користувачі можуть зіткнутися з ризиком шахрайства при купівлі товарів.
3. Платформа не проводить попередню перевірку оголошень, тому можуть з'являтися оголошення з неправдивою інформацією або шахраїв.
4. Час від часу сервіс може працювати нестабільно, що ускладнює роботу користувачів на сайті.

Загалом даний сайт є високопопулярним та значущим ресурсом в нашій країні, який призначений для користувачів, що активно переглядають та шукають товари чи послуги в Інтернеті.

Ebay

Ebay — одна з найбільших інтернет-платформ для онлайн-торгівлі, основний напрямок якої це інтернет-аукціони та створення власних інтернет-магазинів. Ця платформа налічує величезну аудиторію, що складає близько 135 мільйонів покупців по всьому світу, а також має близько 1,7 мільярда активних оголошень, що дозволяє людям з різних країн займатися електронною комерцією та проводити онлайн-транзакції [4].

Ця платформа пропонує користувачу широкий асортимент товарів та вибір із великої кількості категорій на будь-який смак. Користувачі можуть шукати товари за категоріями, регіонами або використовувати фільтри для

зручнішого пошуку. Ebay пропонує безкоштовні або платні опції доставки товарів, а також забезпечує покупцям та продавцям можливість обміну повідомленнями для узгодження деталей оплати та доставки. Також дана платформа надає продавцям набір зручних інструментів для аналізу ринку та просування товарів що допоможе у розвитку власного магазину.

Інтерфейс головної сторінки платформи наведено на відповідному рисунку (див. Рис. 3).

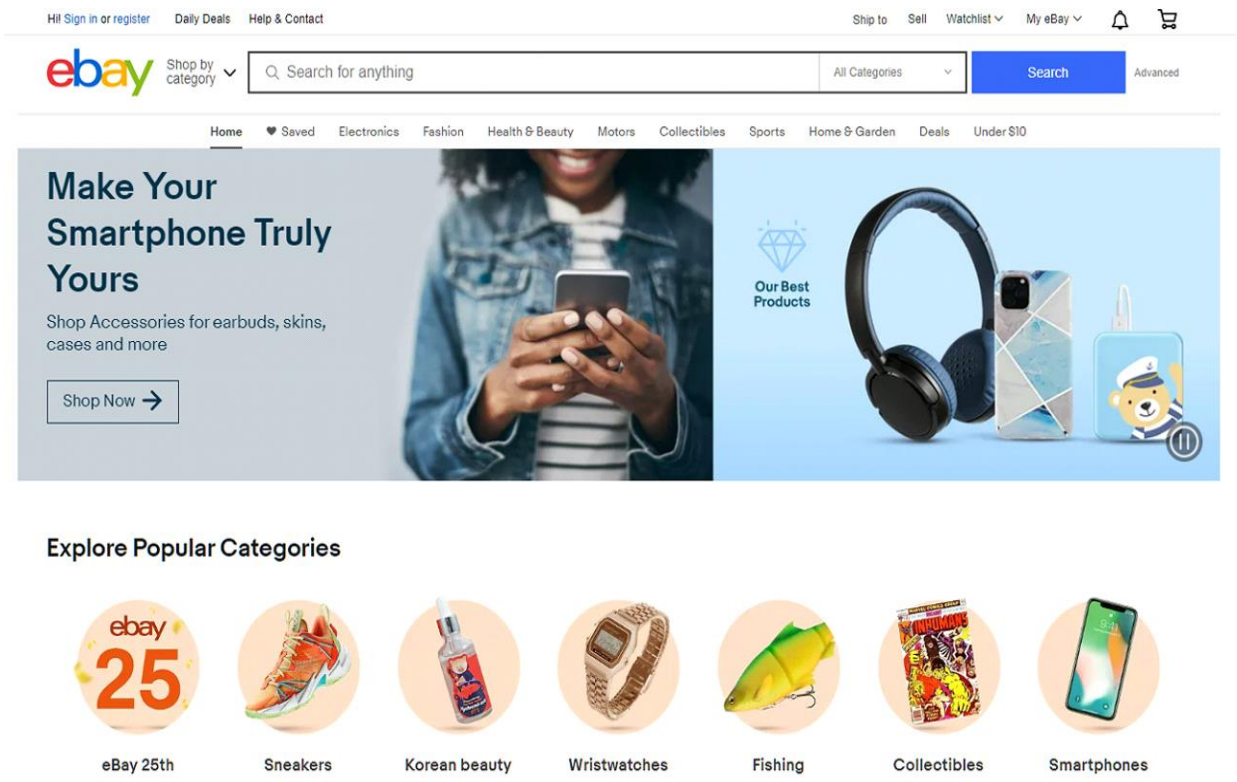

Рис. *3 Головна сторінка сервісу Ebay*

Крім того, на платформі доступний захопливий функціонал аукціонів, приклад якого зображено на рисунку 4. На сторінці аукціону користувачі можуть знайти наступну інформацію:

1. Детальний опис товару, включаючи його характеристики, стан і додаткові деталі. Це дозволяє користувачам ознайомитися з повною інформацією про товар перед зробленням ставки.

2. Найвища ставка, яку користувачі зробили на цей момент. Це дозволяє учасникам аукціону відстежувати поточну ціну товару і зробити свою ставку відповідно.
3. Кількість ставок, які були зроблені на товар. Це дає уявлення про зацікавленість користувачів у лоті та популярність товару серед учасників аукціону.
4. Інформація про час, яка показує, скільки часу залишилося до завершення аукціону. Це допомагає учасникам планувати свої ставки та встигнути придбати товар до закінчення терміну аукціону.

У цьому аукціоні продавець виставляє товар за початковою ціною, а користувачі змагаються, роблячи ставки, щоб зазначити найвищу ціну, яку вони готові заплатити за товар. По закінченні терміну, встановленого продавцем, товар отримує той, хто зробив найвищу ставку.

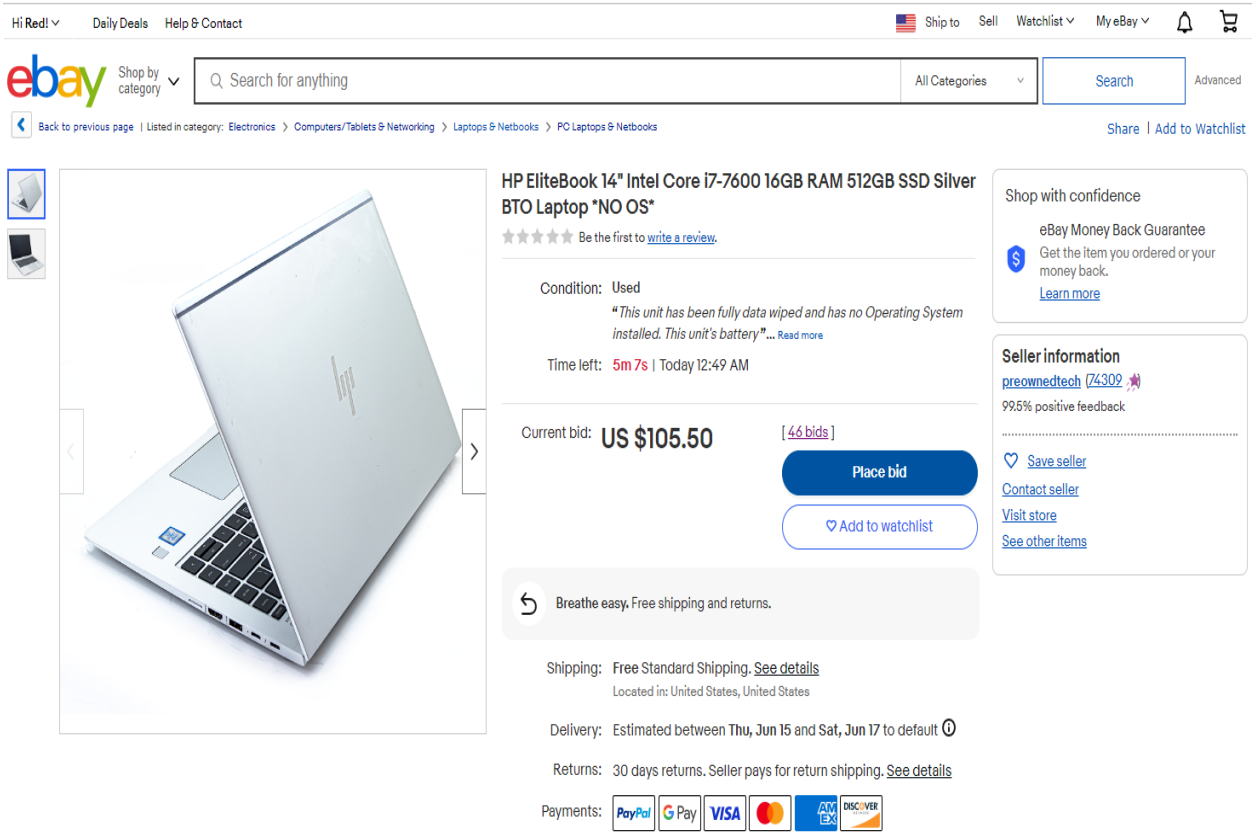

Рис. *4 Сторінка аукціону*

Переваги цієї платформи включають в себе:

1. Платформа має широкий асортимент товарів, включаючи електроніку, одяг, іграшки, автомобілі, меблі, прикраси та багато іншого.
2. Доступні різноманітні способи звязку що дозволяють покупцеві спілкуватися з продавцем.
3. На eBay існує рейтинг продавців, що дозволяє користувачам визначити надійність продавців і зробити потрібний вибір при купівлі товару.
4. Зручну систему аукціонів що дозволяє продавати різні та навіть незвичайні товари за вигідними цінами. Та дозволяє купити товар за достатньо привабливою ціною.
5. Наявність корисного інструментарію, що дозволяє зручно продавати товари та робити аналіз ринку.
6. Технічна підтримка завжди надає гарну допомогу як продавцям, так і покупцям.

До недоліків цієї платформи можна віднести:

1. Досить навантажений та не зрозумілий інтерфейс який може відлякати як покупців, так і продавців.
2. Досить погана адаптивність сайту під маленькі екрани.
3. Платформа накладає значні комісійні збори на продажі, що може призвести до серйозних фінансових витрат для продавців.
4. На eBay може бути недостатньо інформації про товар, що може призвести до того, що покупець не отримує те, що очікував.

В цілому, eBay є однією з найбільших та найвідоміших міжнародних платформ електронної комерції яка дозволяє продавцям та покупцям з різних країн здійснювати торгівлю товарами.

Prom

Prom — це один з найбільших українських B2B (Business-to-Business) та B2C (Business-to-Consumer) онлайн-маркетплейсів, спеціалізований на продажу товарів та послуг як для бізнесу, так і для приватних осіб. Він надає змогу підприємцям самостійно створювати власні інтернет-магазини та

додавати свої товари в загальний каталог. Даний ресурс має величезний асортимент товарів різних категорій та величезну аудиторію приблизно 4.8 мільйонів відвідувачів. На сайті є система рейтингів та відгуків, яка допомагає покупцям швидко та зручно обрати надійного продавця. Для продавців він пропонує можливість швидкого запуску власного онлайн-магазину, залучення нових клієнтів, розширення аудиторії та збільшення продажів. На платформі є різноманітний інструментарій для керування замовленнями, налаштування доставки та оплати, а також інструменти аналітики для відстеження продажу та поведінки покупців [5].

Інтерфейс головної сторінки платформи наведено на відповідному рисунку (див. Рис. 5).

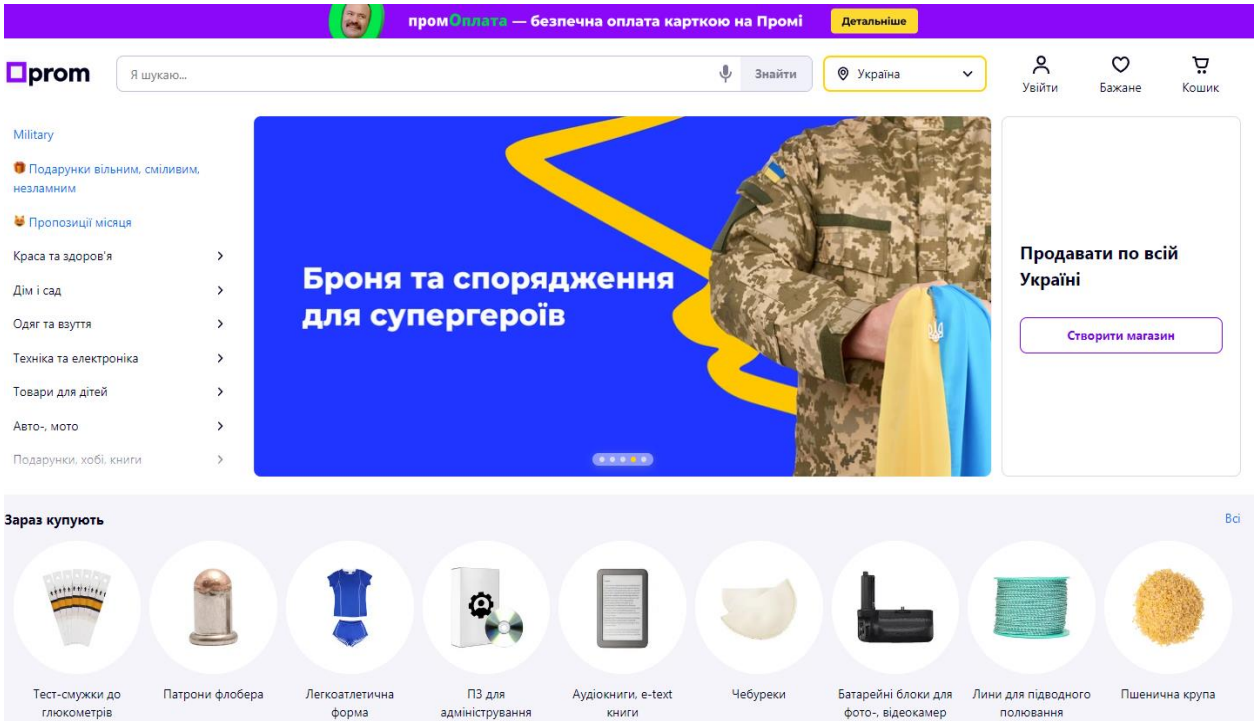

Рис. 5 *Інтерфейс головної сторінки* маркетплейсу *Prom*

Також на Prom є докладна інформація про кожного продавця, яка складається з різних деталей та характеристик. Нижче представлені приклади детальної інформації про кожного продавця, що надають повне уявлення про них (див. Рис. 6):

1. Відсоток позитивних відгуків та рейтинг задоволеності клієнтів

2. Інформація про стаж продавця на маркетплейсі
3. Загальна кількість відгуків
4. Контактна інформація для зв'язку з продавцем
5. Розклад робочих годин продавця
6. Інформація про сервіси доставки та оплати
7. Умови повернення товару та політика гарантії
8. Короткий опис про продавця
9. Перелік усіх оголошень, що належать даному продавцю

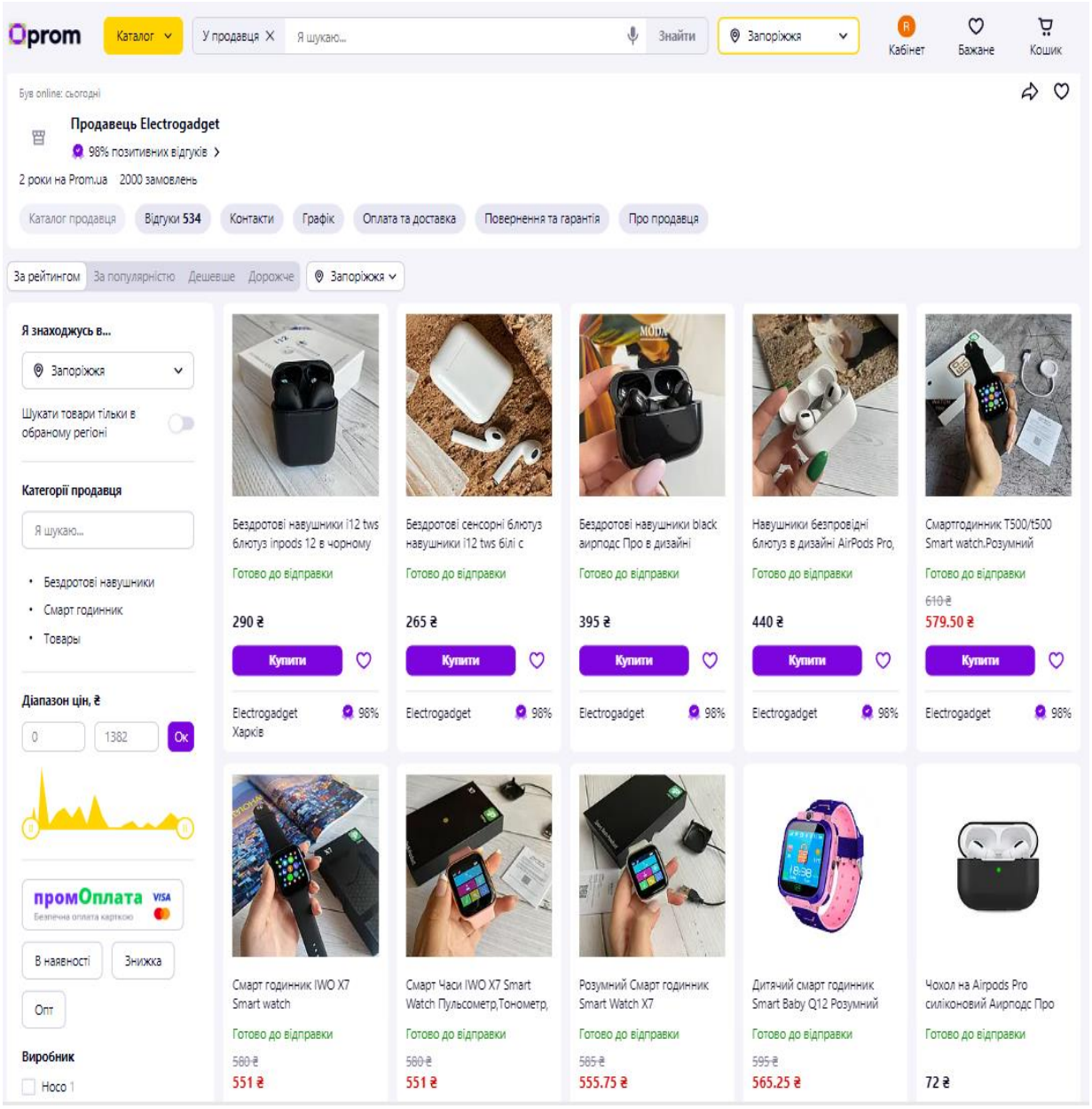

Рис. *6 Сторінка, що містить інформацію про продавця*

Цей маркетплейс має декілька переваг, а саме:

1. Зручний інтерфейс. Prom.ua має простий та інтуїтивно зрозумілий інтерфейс для користувачів, який дозволяє легко знайти та купити потрібний товар.
2. Система рейтингів та відгуків на платформі Prom.ua допомагає покупцям вибрати надійного продавця та уникнути шахрайства.
3. Гарна технічна підтримка продавців яка надає продавцям вчасну допомогу та консультації з питань продажу та управління бізнесом.
4. Зручний інструментарій який дозволяє швидко запустити та управляти інтернет-магазином.

У цьому маркетплейсі також існують певні недоліки, включаючи:

1. В деяких випадках може бути незручно робити оплату на платформі Prom.ua.
2. Платформа має обмеження для деяких продавців, які не можуть продавати певні категорії товарів або працювати на певних умовах.
3. Prom.ua стягує високу комісію з продавців за використання платформи, що може негативно вплинути на прибутки продавців.
4. Prom, як маркетплейс, не забезпечує постійну гарантію на товари, продані продавцями, що може спричинити небезпеку невпевненості для покупців.
5. Prom, як маркетплейс, не забезпечує постійну гарантію на товари, продані продавцями, що може спричинити небезпеку невпевненості для покупців.

1.3 Постановка завдання

Метою кваліфікаційоної роботи є розробка веб-додатку для онлайн торгівлі та бартерного обміну, який буде являти собою сервіс електроної комерції.

Використання сучасних технологій та підходів допоможе полегшити роботу представників різних категорій підприємців та звичайних користувачів у сфері онлайн торгівлі.

Для досягнення покладеної мети необхідно вирішити декілька наступних завдань:

1. Проаналізувати предметну область та визначити необхідні вимоги до віртуального майданчику.
2. Визначити основні сутності, необхідний набір атрибутів та спроектувати схему бази даних.
3. Використовуючи ORM Prisma та СУБД PostgreSQL створити базу даних та міграції для розгортання спроектованої бази даних.
4. Розробити серверну частину застосунку з використанням таких технологій як: Node.js, Nest.js, TypeScript, ORM Prisma.
5. Розробити систему, що має на меті забезпечення безпеки та захисту. Що буде забезпечувати заборону на виконання певних дій у системі без певних прав доступу.
6. Створити інтерфейс користувача з використанням технології React та TypeScript.

2 ДОСЛІДЖЕННЯ ПРОГРАМНИХ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Аналіз сучасних технологій розробки серверної частини

Розробка серверних застосунків стала досить популярною в сучасний час, і це не випадково. Значна потреба у створенні таких застосунків виникає через розширення цифрової сфери та залучення більшої кількості користувачів до онлайн-сервісів. Такі застосунки повинні забезпечувати користувачам максимальну швидкодію та продуктивність під час виконання їх запитів. Це досягається шляхом вибору оптимальних технологій, які забезпечують ефективну обробку даних та оптимізовані алгоритми обробки запитів. Зараз є кілька основних конкурентів серед технологій які дозволяють досягти цієї мети.

ASP.NET

Фреймворк ASP.NET (Active Server Pages.NET) є однією з найпоширеніших технологій для розробки веб-додатків, розроблених компанією Microsoft. ASP.NET використовується для створення динамічних веб-сторінок, веб-служб і веб-додатків. Основною мовою програмування для ASP.NET є C#, але його також можна використовувати з іншими мовами родини .NET, такими як Visual Basic.NET (VB.NET) і F# [6].

ASP.NET базується на технології .NET Framework, що надає готові засоби для виконання різних завдань, пов'язаних з розробкою веб-додатків. Вона пропонує широкий спектр можливостей, включаючи обробку форм, керування станом, автентифікацію та авторизацію користувачів, доступ до баз даних і багато іншого.

Цей фреймворк використовують для розробки різноманітних типів веб-додатків, включаючи корпоративні додатки, соціальні мережі, електронні комерційні сайти, блоги, форуми, системи управління вмістом та інші. Його гнучкість та розширюваність дозволяють розробникам створювати рішення під різноманітні потреби та сфери діяльності.

Серед переваг даної технології можна зазначити:

1. Розширені можливості: ASP.NET надає потужні інструменти для розробки веб-додатків, включаючи інтеграцію з базами даних, кешування, керування сесіями та інші функціональності. Він також підтримує різні моделі програмування, такі як Web Forms і MVC, що дозволяє розробникам вибрати найкращий підхід для свого проєкту.
2. Висока продуктивність: ASP.NET забезпечує високу продуктивність завдяки компіляції коду на етапі розгортання, що сприяє швидкому завантаженню сторінок та ефективній роботі сервера.
3. Інтеграція з екосистемою Microsoft: ASP.NET плідно взаємодіє з іншими продуктами та інструментами Microsoft, такими як Microsoft SQL Server, Azure, Active Directory та інші. Це дозволяє розробникам зручно використовувати ці рішення для створення повноцінних додатків.
4. Безпека: ASP.NET має вбудовану систему безпеки, яка спрощує реалізацію аутентифікації, авторизації та управління ролями. Фреймворк надає захист від різних видів атак, таких як SQL-ін'єкція, міжсайтовий скриптинг (XSS) та інші.

Цій технології притаманні деякі недоліки включаючи:

1. Високі вимоги до серверного обладнання: Запуск ASP.NET додатків вимагає потужного серверного обладнання, що може вплинути на витрати на розгортання та підтримку інфраструктури.
2. Обмежена масштабованість: У деяких сценаріях, особливо високонавантажених додатках, може виникати обмеження масштабованості ASP.NET порівняно з іншими технологіями.

Spring

Spring — високопродуктивний фреймворк для розробки програмного забезпечення на мові Java, який надає значні можливості та зручності для розробників. Він надає комплексний набір інструментів та бібліотек для ефективного створення різноманітних застосунків, від простих веб-сайтів до складних корпоративних додатків [7].

Однією з ключових особливостей Spring є його контейнер інверсії управління (IoC). Цей контейнер дозволяє визначати залежності між компонентами додатка та автоматично керувати їх життєвим циклом. Замість того, щоб ручно створювати об'єкти й встановлювати їх залежності, Spring забезпечує автоматичну ініціалізацію та керування залежностями, що спрощує розробку і підтримку коду.

В основному його використовують для розробки веб-додатків, корпоративних додатків, мікросервісів.

У цього фреймворку є кілька суттєвих переваг:

1. Інверсія управління (IoC): Spring використовує принцип інверсії управління, що дозволяє зменшити залежність між компонентами та забезпечує більшу гнучкість в конфігурації та розширенні додатків.
2. Велика екосистема: Spring має широкий спектр модулів та плагінів, які допомагають у розробці різноманітних функцій, включаючи веб-розробку, роботу з базами даних, безпеку, кешування, розподілені системи та багато іншого.
3. Підтримка корпоративних потреб: Spring надає механізми для управління транзакціями, безпекою, плануванням завдань та іншими корпоративними функціями.
4. Висока безпека: Spring має модуль Spring Security, який дозволяє забезпечувати безпеку веб-додатків шляхом контролю доступу до ресурсів та захисту від різних типів атак.

Однак, цей фреймворк також має свої недоліки:

1. Складність: Spring може бути складним для розробників та потребувати багато часу та зусилля, оскільки він має багато компонентів та концепцій.
2. Комплексність: Spring є потужним фреймворком з великою кількістю можливостей. Проте, ця комплексність може призвести до зайвої складності в додатках, особливо якщо використовують лише деякі його функціональні можливості.

3. Продуктивність: В деяких випадках, через широкий функціонал та обробку IoC, може виникати деяке зниження продуктивності.

Node.js

Технологія Node.js — вільне та відкрите середовище виконання JavaScript, яке побудоване на двигуні JavaScript V8, розробленому Google для використання в браузері Google Chrome. Воно дозволяє виконувати код JavaScript на сервері та створювати масштабовані та ефективні мережеві додатки [8].

Основна ідея Node.js полягає в тому, що він дозволяє розробникам використовувати JavaScript як мову програмування для побудови серверних додатків. Це робить його ідеальним рішенням для розробки веб-додатків, додатків чату, API й багатьох інших типів програмного забезпечення.

У даному середовищі можна виділити декілька переваг, зокрема:

1. Висока продуктивність: Node.js використовує асинхронну модель програмування, що дозволяє обробляти багато запитів одночасно без блокування потоку виконання. Це дозволяє досягти високої продуктивності та швидкості відповіді на запити.
2. Масштабованість: Завдяки необхідності обробки багатьох запитів одночасно, Node.js добре підходить для побудови масштабованих додатків. Він також може бути легко масштабований за допомогою горизонтального масштабування шляхом додавання багатьох серверів.
3. Єдина мова програмування: Використання JavaScript як мови програмування являє собою значну перевагу, оскільки це дозволяє розробникам працювати як на стороні клієнта, так і на стороні сервера, що спрощує розробку і підтримку коду.
4. Широкий вибір модулів та пакетів: Node.js має велику кількість модулів, доступних через npm, що дозволяє розробникам легко використовувати готові рішення для розширення функціональності своїх додатків.

Також варто враховувати деякі недоліки цього середовища, а саме:

1. Однопоточність: Node.js використовує один потік виконання, що означає, що він може мати обмеження в обробці CPU-інтенсивних завдань. Він може стикатися з проблемою блокування при виконанні довгих операцій, які блокують потік виконання.
2. Нестабільність деяких модулів: Оскільки Node.js має велику кількість модулів, які розробляються сторонніми розробниками, існує ризик неповної сумісності, помилок або вразливостей у деяких модулях.

Flask

Фреймворк Flask є одним з найпопулярніших інструментів для розробки веб-додатків у середовищі Python. Він базується на принципах мінімалізму та простоти, що дозволяє розробникам швидко створювати потужні веб-додатки з низьким рівнем складності [9].

Завдяки своїй простоті та гнучкості, Flask може бути використаний для створення різноманітних проектів, від простих статичних сайтів до складних веб-додатків зі складною бізнес-логікою.

Потрібно відзначити кілька переваг цього фреймворку, зокрема:

1. Простота використання: Flask має простий та зрозумілий синтаксис, що полегшує початок роботи з фреймворком та прискорює процес розробки.
2. Гнучкість: Flask надає розробникам велику свободу у виборі компонентів та архітектури свого додатку. Розробники можуть використовувати лише необхідні модулі та розширення, що дозволяє створювати легковажні та ефективні веб-додатки.
3. Розширення: Flask має велику кількість розширень, які розробники можуть використовувати для додавання нової функціональності.

У фреймворку є кілька недоліків, які варто враховувати:

1. Недостатній "вбудований" функціонал: Оскільки Flask має мінімалістичний підхід, в ньому відсутній певний функціонал, який може бути потрібний для деяких проектів.

2. Відсутність стандартизації: Оскільки Flask надає розробникам багато свободи у виборі компонентів та підходів, це може призвести до відсутності стандартизації в проектах.
3. Невелика універсальність: Flask, будучи легковаговим фреймворком, може мати обмежені можливості для деяких специфічних типів додатків, таких як великі торгові платформи або системи управління вмістом.
4. Проведення аналізу технологій розробки серверної частини є важливим етапом у процесі розробки веб-застосунків. При цьому, було враховано різні аспекти, такі як масштабованість, продуктивність, надійність та безпека.

У процесі аналізу були розглянуті різні платформи та мови програмування. Кожна з цих технологій має свої переваги та особливості, і вибір відповідної залежить від конкретних потреб проекту. Тому враховуючи потребу у швидкодії, можливості масштабування та наявності великої кількості модулів та розширень, для розробки сучасних серверних додатків краще обрати Node.js.

Оскільки платформа Node.js стає все більш популярною і саме вона буде оптимальний вибором для розробки багатьох серверних застосунків які потребують продуктивності та легкості масштабування.

2.2 Аналіз сучасних реляційних систем управління базами даних

У сучасному світі використання база даних для збереження, організації та управління великим обсягом структурованої інформації, стало характерною рисою систем які працюють з великим обсягом даних.

Реляційні системи управління базами даних (РСУБД) вважаються невід'ємною складовою сучасної розробки програмного забезпечення. Вони забезпечують потужність та ефективність управління великими обсягами даних, що використовуються у різноманітних додатках та системах.

У сучасних РСУБД реалізовано багато функціональних можливостей для ефективного управління базами даних. Вони дозволяють створювати, змінювати й видаляти таблиці, індексувати дані для швидкого пошуку, контролювати транзакції для забезпечення цілісності даних та здійснювати резервне копіювання для забезпечення безпеки даних. Крім того, РСУБД підтримують розширені можливості, такі як тригери та збережені процедури, які дозволяють виконувати автоматичні дії та зберігати набір інструкцій на стороні сервера.

Одним з найважливіших аспектів РСУБД є їх продуктивність. Швидкодія та ефективність операцій з базою даних визначаються такими факторами, як оптимізація запитів, індексування, кешування, паралельне виконання та оптимізація обсягу даних. Сучасні РСУБД використовують різні техніки та алгоритми для підвищення продуктивності і забезпечення швидкого доступу до даних.

Надійність є ще одним важливим аспектом РСУБД. Вони забезпечують механізми для забезпечення надійності та цілісності даних, такі як транзакційний контроль, відновлення після збоїв та механізми резервного копіювання. Резервне копіювання даних дозволяє відновити базу даних у разі втрати або пошкодження даних.

Реляційні СУБД мають свої особливості та характеристики тому серед них можна виділити декілька типів:

1. Класичні реляційні: Вони використовують класичну реляційну модель даних, засновану на таблицях, стовпцях та рядках. Класичні реляційні СУБД підтримують SQL для запитів та управління базою даних. До прикладів входять Oracle Database, MySQL, Microsoft SQL Server, PostgreSQL.
2. Об'єктно-реляційні СУБД (ОРСУБД): Ці СУБД розширюють класичну реляційну модель даних, дозволяючи зберігати та опрацьовувати об'єкти, такі як зображення, звуки, відео, XML-документи тощо, безпосередньо в базі даних. До прикладів відносяться Oracle Database (з

використанням типу даних "об'єкт"), PostgreSQL (з використанням розширення Postgres Object) та IBM DB2.

3. Розподілені реляційні СУБД: Ці СУБД дозволяють розподіляти базу даних на різні фізичні сервери або вузли, що забезпечує масштабованість та високу доступність. Приклади включають MySQL Cluster, Microsoft SQL Server з режимом Always On та PostgreSQL з розширенням Citus Data.
4. Вбудовані реляційні СУБД: Ці СУБД призначені для вбудованого використання в додатки або пристрої з обмеженими ресурсами. Вони мають легковажний розмір та низькі вимоги до обчислювальних ресурсів. Приклади включають SQLite та HSQLDB.

Серед основних причин використання реляційних СУБД можна виділити наступні:

1. Структуроване збереження даних: Реляційні СУБД надають можливість зберігати дані у вигляді таблиць зі стовпцями та рядками. Це дозволяє організувати дані у логічні сутності та встановлювати зв'язки між ними, що спрощує управління та забезпечує структурованість даних.
2. Запити та аналітика даних: Вони підтримують мову запитів SQL, яка дозволяє виконувати різноманітні операції з даними. За допомогою SQL можна легко вибирати, фільтрувати, сортувати, об'єднувати та агрегувати дані для отримання потрібної інформації. Також підтримують аналітичні функції, що дозволяють проводити розширений аналіз даних.
3. Цілісність та надійність даних: Реляційні СУБД надають механізми для забезпечення цілісності даних, такі як обмеження на поля (наприклад, унікальність, зовнішні ключі) та транзакції. Це допомагає підтримувати правильність та консистентність даних та запобігати некоректним змінам.
4. Безпека даних: Реляційні СУБД надають можливості для забезпечення безпеки даних. Вони дозволяють встановлювати рівні доступу до бази

даних, обмежувати права користувачів та забезпечувати шифрування даних для захисту від несанкціонованого доступу та зламів.

5. Масштабованість: Реляційні СУБД можуть бути масштабовані залежно від потреб організації. Вони можуть працювати з великим обсягом даних та підтримувати багато користувачів одночасно. Також існують можливості для реплікації та розподіленої обробки даних для поліпшення продуктивності та надійності системи.
6. Підтримка резервного копіювання та відновлення даних: Реляційні СУБД зазвичай надають засоби для резервного копіювання та відновлення бази даних. Це дозволяє забезпечити захист даних шляхом створення резервних копій і можливість відновлення даних в разі втрати або пошкодження.

Хоча реляційні системи управління базами даних (РСУБД) мають багато переваг й розділяються на декілька типів, усі вони також мають деякі недоліки, які варто враховувати:

1. Гнучкість: У реляційній моделі даних потрібно визначити схему бази даних заздалегідь, що включає структуру таблиць та відношення між ними. Це може стати проблемою, коли потрібно внести зміни до схеми або додати нові елементи до бази даних.
2. Обробка складних структур даних: РСУБД не завжди ефективно обробляють складні структури даних, такі як вкладені об'єкти, масиви чи географічні дані.
3. Обмежена масштабованість: Деякі РСУБД можуть зіткнутися з обмеженнями масштабованості, особливо при великому обсязі даних або високих навантаженнях.
4. Високі вимоги до апаратного забезпечення: Деякі СУБД можуть вимагати потужного апаратного забезпечення для оптимальної продуктивності та швидкодії.

Проведений аналіз зазначив, що реляційні системи управління базами даних (РСУБД) є широко використовуваними та мають багато переваг, таких

як стандартизація, надійність та потужність. Вони дозволяють зберігати та обробляти дані у вигляді таблиць зі зв'язками, використовуючи мову запитів SQL. Проте, вони також мають свої недоліки, зокрема складність моделювання даних та обмеженість у роботі з неструктурованими даними.

Тому, з огляду на потрібність у цілісності, безпеці та масштабованості, для розробки віртуального майданчика торгівлі та бартерного обміну було прийнято рішення використовувати класичну реляційну СУБД, а саме PostgreSQL.

PostgreSQL є однією з найпопулярніших та широко використовуваних реляційних систем управління базами даних (РСУБД). Ця система відома своєю високою продуктивністю, надійністю та масштабованістю. PostgreSQL надає широкий набір функцій, які включають складні операції з даними, розширені можливості запитів, підтримку географічних даних, роботу з JSON та підтримку відкритих розширень [10]. Однією з основних переваг PostgreSQL є його відкритість. Ця система поширюється під ліцензією PostgreSQL, що дозволяє користувачам використовувати, змінювати та поширювати її безкоштовно. Такий підхід сприяє активному співробітництву в спільноті розробників та користувачів, яка постійно працює над вдосконаленням та розширенням можливостей PostgreSQL.

2.3 Технології для розробки веб-інтерфейсу застосунку

Задача створення якісного та зручного у використанні інтерфейсу є досить важливою, оскільки саме від інтерфейсу залежить значна частина досвіду користувача та успіх онлайн-проекту.

Веб-інтерфейс — обличчя веб-додатку або сайту. Саме він визначає, як користувачі взаємодіють з розробленим продуктом і як вони сприймають його. Якість веб-інтерфейсу має безпосередній вплив на користувацьку задоволеність, залучення, збереження користувачів і конверсію.

Для розробки інтерфейсу користувача зазвичай використовують такі сучасні технології:

1. HTML (HyperText Markup Language): Це основна мова розмітки для створення структури веб-сторінок..
2. CSS (Cascading Style Sheets): CSS використовується для визначення зовнішнього вигляду веб-сторінок.
3. JavaScript: JavaScript є мовою програмування, яка використовується для надання динамічності веб-сторінкам. Ви можете використовувати JavaScript для створення взаємодії з користувачем, валідації форм, анімації, завантаження даних асинхронно (за допомогою AJAX) та багато іншого.
4. UI-бібліотеки: UI-бібліотеки, такі як Bootstrap, Foundation, Material-UI, надають готові компоненти та стилі, що допомагають створити швидкий та привабливий веб-інтерфейс.

Досить важливим питанням є вибір фреймворку для розробки користувацької частини веб-додатку. Існує безліч фреймворків, які можна використовувати для розробки веб-інтерфейсу застосунку. Але серед них можна виділити декілька основних.

React

React — одна з найпопулярніших JavaScript бібліотек для розробки інтерфейсу користувача. Вона була розроблена компанією Facebook та поширюється як відкрите програмне забезпечення. React використовується для побудови ефективних та масштабованих веб-додатків, де компоненти інтерфейсу оновлюються лише при зміні їх стану, що дозволяє ефективно керувати відображенням даних [11].

Зазвичай його використовують для розробки таких типів додатків як:

1. Односторінкові додатки (Single-Page Applications, SPA): React дозволяє створювати потужні SPA, які завантажуються один раз, а далі взаємодіють з користувачем, оновлюючи лише необхідні частини сторінки.

2. Мобільні додатки: За допомогою React Native, який базується на React, можна розробляти нативні мобільні додатки для платформ iOS та Android. Це дозволяє використовувати знання та навички з React для створення мобільних додатків з високою продуктивністю та нативними можливостями.
3. Веб-розширення: React може бути використаний для створення веб-розширень для різних браузерів. Це дозволяє розширити функціональність браузера та створити власні користувацькі інструменти та інтерфейси.

Основними перевагами цієї бібліотеки є:

1. Ефективність: React використовує віртуальний DOM (Virtual DOM), що дозволяє оновлювати тільки необхідні частини сторінки при зміні даних. Це дозволяє уникнути зайвих операцій з маніпулювання реальним DOM і значно підвищує продуктивність веб-додатків при роботі з великою кількістю компонентів.
2. JSX: React використовує JSX — розширення синтаксису JavaScript, що дозволяє використовувати HTML-подібний синтаксис для опису компонентів. Це робить код більш зрозумілим і сприяє легшому створенню та обслуговуванню інтерфейсів.
3. Компонентна архітектура: React побудований на основі компонентів, які можна повторно використовувати, що дозволяє прискорити процес розробки, полегшити супровід коду та забезпечити більшу модульність.
4. Односторонній потік даних: React використовує односторонній потік даних, що робить управління станом додатку простішим і передбачуваним.

Серед недоліків цієї бібліотеки можна виділити:

1. Висока крива навчання: React може бути викликом для початківців. Для розуміння концепцій React, таких як компоненти, стан та пропси, може знадобитись певний час.

2. Висока кількість абстракцій: React використовує багато абстракційних шарів, що можуть викликати деякі труднощі при налагодженні та розумінні роботи додатка.
3. Нестабільність API: React активно розвивається, і нові версії можуть мати зміни у своєму API. Це може вимагати оновлення коду та зусиль для збереження сумісності з останніми версіями React.

В цілому, React є високоефективною бібліотекою JavaScript, яка надає розробникам потужні інструменти для створення складних інтерфейсів користувача. Він підходить для розробки як невеликих, так і великих проєктів, завдяки своїй компонентній архітектурі та великій спільноті розробників, що надає різноманітні ресурси та підтримку. React також відзначається швидким відгуком і високою продуктивністю завдяки використанню віртуального DOM. Крім того, він має розширену екосистему, яка дозволяє розробникам використовувати додаткові бібліотеки та інструменти для поліпшення розробки й розширення функціональності додатка. Усі ці особливості роблять React популярним інструментом у світі розробки веб-додатків.

Angular

Angular — повноцінний веб-фреймворк, розроблений компанією Google, який дозволяє будувати ефективні, масштабовані та декларативні веб-додатки. Він базується на мові програмування TypeScript і використовує компонентний підхід до розробки. В основному на ньому розробляють додатки рівня Enterprise [12].

Основні переваги даного фреймворку:

1. Декларативний підхід: Angular використовує декларативний підхід до опису інтерфейсу користувача. За допомогою шаблонів HTML та директив, розробники можуть описати бажаний стан інтерфейсу, а Angular відповідає за його оновлення при зміні даних.
2. Типізація: Angular використовує TypeScript - строго типізовану версію JavaScript. Це дозволяє виявляти помилки на етапі розробки, поліпшує розуміння коду та забезпечує більшу безпеку та стабільність.

3. Спільнота та документація: Angular має велику та активну спільноту розробників, яка надає підтримку, спільні рішення та розробляє сторонні бібліотеки. Також є офіційна документація, яка містить багато прикладів та пояснень.
4. Модульність: Angular побудований на основі модульної архітектури, що дозволяє розбити додаток на незалежні модулі. Це спрощує розробку, тестування та підтримку додатку.

Недоліки фреймворку Angular:

1. Складність: Angular є досить складним фреймворком, який має крутішу криву навчання порівняно з іншими фреймворками, такими як React або Vue.js
2. Залежність від виробника: Angular розробляється і підтримується командою Google. Це означає, що деякі рішення інтеграції та оновлення можуть бути обмежені в залежності від стратегії компанії.

Хоча Angular має свої особливості та обмеження, його переваги роблять його привабливим вибором для підприємств та великих проектів, де потрібна масштабованість, надійність та підтримка. Angular надає потужні інструменти для масштабування, забезпечує надійність розробки та має активну спільноту підтримки.

Vue.js

Vue.js — це прогресивний JavaScript фреймворк для розробки веб-інтерфейсів, який використовується для створення користувацьких інтерфейсів з високою швидкістю та масштабованістю. Розроблений Еваном Ю, Vue.js здобув значну популярність в розробницькій спільноті завдяки своїй простоті використання та потужному функціоналу [13].

Цей фреймворк пропонує багато позитивних можливостей:

1. Легкість вивчення та використання: Vue.js має просту та лаконічну синтаксичну структуру, що робить його легким для вивчення навіть для початківців.

2. Реактивність: Vue.js пропонує реактивну систему, яка автоматично оновлює компоненти при зміні даних.
3. Компонентний підхід: Vue.js базується на компонентах, що дозволяє розбити веб-інтерфейс на незалежні, повторно використовувані блоки.
4. Гнучкість та розширюваність: Vue.js надає розробникам велику свободу вибору технологій та підходів до розробки.
5. Ефективність: Vue.js володіє високою продуктивністю завдяки своєму легкому розміру та ефективній системі оновлення компонентів.

Серед недоліків використання фреймворку можна виділити:

1. Обмежена кількість розширень: У порівнянні з Angular або React, екосистема Vue має меншу кількість розширень і бібліотек. Це означає, що може бути важче знайти готові рішення для певних функціональних задач.
2. Менша популярність: У порівняні з React або Angular. Vue.js має відносно меншу популярність. кількість розробників та меншу активність спільноти. Це може призвести до меншої кількості ресурсів та матеріалів для навчання та підтримки.
3. Складність для великих проектів: Vue.js своєю чергою, був концепційно спроектований з орієнтацією на розробку невеликих і середніх проектів, тому його складно використовувати для великих проектів.

Незважаючи на меншу популярність у порівнянні з іншими фреймворками, Vue.js привабливий вибір для розробки веб-додатків у проєктах різних розмірів завдяки своїм перевагам, таким як простота використання, гнучкість та продуктивність.

Після ретельного огляду та дослідження всіх особливостей, переваг і недоліків різних технологій для розробки клієнтської частини віртуального майданчика електронної комерції, було прийнято рішення вибрати React.

Обрання ReactJS для розробки віртуального майданчика торгівлі та бартерного обміну зумовлено його компонентною архітектурою, яка сприяє спрощенню процесу розробки та підтримки. За допомогою компонентів можна

створювати елементи інтерфейсу, які можна повторно використовувати, що прискорює розробку та сприяє легкій масштабованості проєкту.

Також він має велику кількість доступних розширень і бібліотек. Ця багатогранна екосистема дозволяє використовувати різноманітні розширення для розширення функціональності та поліпшення розробки.

Крім того, React має досить широку спільноту, що забезпечує доступ до різноманітних ресурсів та підтримки. Це охоплює документацію, навчальні матеріали та інструменти, які сприяють успішній розробці. Ця простора екосистема робить React оптимальним вибором для розробки клієнтської частини додатку.

3 РОЗРОБКА ВЕБ-ДОДАТКУ ВІРТУАЛЬНОГО МАЙДАНЧИКА ТОРГІВЛІ ТА БАРТЕРНОГО ОБМІНУ

3.1 Опис предметної області

Процес розробки сервісів електронної комерції, як правило, охоплює ряд складних завдань, що потребують професійних навичок у розробці програмного забезпечення. Зазвичай такі сервіси розробляються у вигляді веб-застосунків з використанням сучасних технологій програмування та підходів до розробки програмного забезпечення. Хоча базовий набір функціональних можливостей у таких веб-застосунках часто є схожим, він може відрізнятися в залежності від потреб та поставлених цілей. Усі такі сервіси мають достатньо інтуїтивний інтерфейс користувача, який не вимагає глибоких знань для використання, забезпечуючи простоту та зручність взаємодії з ними.

Сервіси електронної комерції надають користувачам широкий спектр функцій. Вони дозволяють здійснювати пошук товарів за різними параметрами, такими як назва, категорія, а також сортувати результати за ціною або датою. Завдяки додатковим фільтрам, наприклад, ціновому діапазону, користувачі можуть налаштувати результати пошуку для знаходження необхідного товару з більшою точністю.

Крім того, на цих платформах доступний функціонал авторизації, який дозволяє користувачам увійти до системи під своїм профілем та отримати розширені можливості. Наприклад, авторизовані користувачі можуть створювати відгуки та оцінки товарів, що сприяють формуванню об'єктивного уявлення про якість товару. Також вони можуть отримувати персоналізовані рекомендації, які допомагають знайти товари, що відповідають їхнім індивідуальним потребам та вподобанням. Ще існують, програми лояльності та знижки які надають додаткові переваги та роблять покупки ще більш вигідними для користувачів.

У електронній комерції, окрім покупців, велику роль відіграють продавці. Вони використовують ці сервіси для розміщення своїх оголошень про товари чи послуги. Оголошення мають свої характеристики та атрибути, такі як назва, опис, фотографії, категорія, адреса та тип оголошення, що може вказувати на намір продажу або обміну. Ці атрибути допомагають продавцям презентувати свої товари чи послуги й привернути увагу потенційних покупців. Чітка та докладна інформація разом з візуальними матеріалами допомагає зрозуміти переваги та характеристики пропонованого продукту та забезпечити якісну взаємодію між продавцем та покупцем.

3.2 Архітектура системи

Веб-додаток віртуального майданчика торгівлі та бартерного обміну розбивається на дві складові частини:

1. Серверна сторона (Back-end): Яка займається обробкою запитів від клієнтської сторони й забезпечує взаємодію з базою даних. Вона відповідає за логіку пошуку, фільтрації, авторизації, аутентифікації, валідації, обробки замовлень та інших бізнес-процесів, пов'язаних з майданчиком торгівлі та бартерним обміном. Серверна сторона взаємодіє з базою даних, де зберігається вся необхідна інформація, така як оголошення, користувачі, транзакції тощо. Вона забезпечує надійне та ефективне управління даними, їх збереження та доступ до них.
2. Клієнтська сторона (Front-end): Що відповідає за взаємодію з користувачем, відправку запитів на сервер, отримання відповідей та представлення інтерфейсу веб-додатку. Включає в себе функціонал авторизації, розміщення оголошень, функцію пошуку, використання фільтрів та інші елементи, що сприяють зручному використанню майданчика торгівлі та бартерного обміну. Клієнтська сторона забезпечує зручну та інтуїтивно зрозумілу взаємодію користувача з системою.

3.3 Вимоги до системи

У розробці будь-якої системи необхідно чітко визначити вимоги, які відображають потреби та очікування користувачів і зацікавлених сторін, а також обмеження, які впливають на проєктування та реалізацію системи. Вимоги до системи є необхідним елементом, оскільки вони встановлюють функціональні та нефункціональні характеристики, які система повинна мати для задоволення потреб користувачів та бізнес-вимог.

Оскільки система розбивається на дві компоненти згідно з її архітектурою, необхідно розробити вимоги як для системи в цілому, так і для кожної окремої компоненти. Це вимагає вироблення функціональних та не функціональних специфікацій для обох частин системи, список яких наведено нижче.

До функціональних вимог можна віднести:

1. Реєстрація та Авторизація користувача:
    a. Користувачі повинні мати можливість створити обліковий запис на майданчику, вводячи необхідну інформацію
    b. Користувачі повинні мати можливість увійти до своїх облікових записів створених у системі.
    c. Система повина забезпечити можливість встановлення нового пароля якщо користувач його забув.
    d. Користувачі повинні мати можливість у будь-який час вийти зі свого облікового запису й увійти до іншого облікового запису.
2. Профіль користувача:
    a. Користувачі мають можливість створювати свій профіль та контактну інформацію.
    b. Система повинна надавати користувачу можливість редагувати інформацію свого профілю.
    c. Користувач повинен мати можливість завантажувати свою фотографію профілю та керувати нею.

3. Блокування користувача:
    a. Адміністратор повинен мати можливість блокувати користувачів, які порушують правила поведінки або надають недостовірну інформацію.
    b. Система повинна забезпечувати можливість перегляду списку заблокованих користувачів для адміністратора.
    c. Користувач, якщо він був заблокований, повинен отримати сповіщення про блокування та пояснення причини блокування.
4. Створення оголошень:
    a. Користувачі повинні мати можливість створювати оголошення про свої товари, які вони бажають продати або обміняти.
    b. Система повинна дозволяти користувачу додавати основні деталі оголошення, такі як назва, опис, фотографії, ціна, категорія тощо.
5. Редагування оголошень:
    a. Користувач повинен мати змогу редагувати кожну деталь свого оголошення таку як назва, опис, фотографії, ціна тощо.
    b. Система повинна забороняти користувачу редагувати чужі оголошення якщо він не має роль адміністратор.
6. Перегляд оголошень:
    a. Користувачам повинна бути надана можливість переглядати список доступних оголошень на майданчику.
    b. Система має надавати основну інформацію про кожне оголошення, таку як назва, категорія, фотографії та короткий опис, щоб забезпечити користувачам швидку оцінку цікавих їм пропозицій.
7. Фільтрація та сортування оголошень:
    a. Система повинна надавати можливість фільтрувати оголошення за різними критеріями, такими як категорія, ціновий діапазон, місцезнаходження тощо.

b. Користувачі повинні мати можливість сортувати список оголошень за різними параметрами, такими як дата розміщення т ціна.

8. Обмін повідомленнями:
    a. Користувачі повинні мати можливість комунікувати один з одним через систему, щоб обговорювати деталі операцій, узгоджувати умови обміну або уточнювати інформацію про товари та послуги.
9. Оцінки та відгуки:
    a. Користувачам повинна бути надана можливість залишати оцінки та відгуки після завершення торгівлі або обміну.
    b. Система повинна забезпечувати зручний інтерфейс, де користувачі можуть виразити свою думку та оцінку щодо взаємодії з іншими користувачами.
    c. Користувачі повинні мати змогу дивитись усі відгуки про інших користувачів.
10. Керування контентом:
    a. Користувач з ролю адміністратора повинен мати можливість отримувати інформацію про усі дані системи.
    b. Адміністратор повинен мати можливість додавати дані до системи.
    c. Система повинна забезпечувати зручний інтерфейс адмін панелі для роботи з даними.
    d. Адміністратор повинен мати можливість редагувати дані.
11. Платежі та транзакції:
    a. Користувачу повинен мати змогу додати кредитну картку.
    b. Користувач повинен мати можливість видалити кредитну карту.
    c. Система повинна надавати змогу зробити оплату товару з обраної кредитної картки.
    d. Система повинна зберігати історію всіх здійснених транзакцій, що містять інформацію про оплати та про товари. Користувачі

повинні мати можливість переглядати свою історію транзакцій для відстеження платежів та ведення обліку.

Наведені функціональні вимоги описують усі ключові можливості, які включає веб-додаток віртуального майданчика торгівлі та бартерного обміну. Ці вимоги враховують всі ключові аспекти та деталі, необхідні для успішної роботи та задоволення потреб користувачів на майданчику.

До нефункціональних вимог можна віднести:

1. Продуктивність:
    a. Система повинна мати високу швидкодію в обробці запитів та відгуках на дії користувачів.
    b. Час завантаження сторінок та відображення результатів пошуку повинен бути мінімальним.
2. Безпека:
    a. Система повинна забезпечувати конфіденційність особистих даних користувачів, таких як інформація про їх пароль та контактні дані.
    b. Механізми аутентифікації та авторизації повинні бути належним чином реалізовані, щоб унеможливити несанкціонованого доступу до системи.
3. Інтерфейс користувача:
    a. Користувацький інтерфейс повинен бути зручним, інтуїтивно зрозумілим та привабливим для користувачів.
    b. Вимоги до дизайну, навігації та взаємодії повинні бути враховані, забезпечуючи зручне використання системи.

Вище наведені основні вимоги до системи визначають ключові характеристики та функціональність, які мають бути вбудовані у програмний продукт. Ці вимоги створюють фундаментальну основу для розробки та реалізації системи, забезпечуючи її коректну роботу.

3.4 Проектування системи веб-додатку

Під час етапу проектування було виконано структурно-функціональне моделювання, яке дозволило детально прослідкувати виконання всіх етапів проектування веб-застосунку. Цей аналітичний підхід також дозволив проаналізувати можливі напрямки удосконалення роботи системи з метою покращення її ефективності та задоволення потреб користувачів.

Для виокремлення основних можливостей системи було розроблено діаграми варіантів використання, які відображають взаємодію акторів (користувач та адміністратор) з системою та описують різні сценарії взаємодії. Діаграми варіантів використання для користувача та адміністратора наведені на рисунках 7 та 8 відповідно.

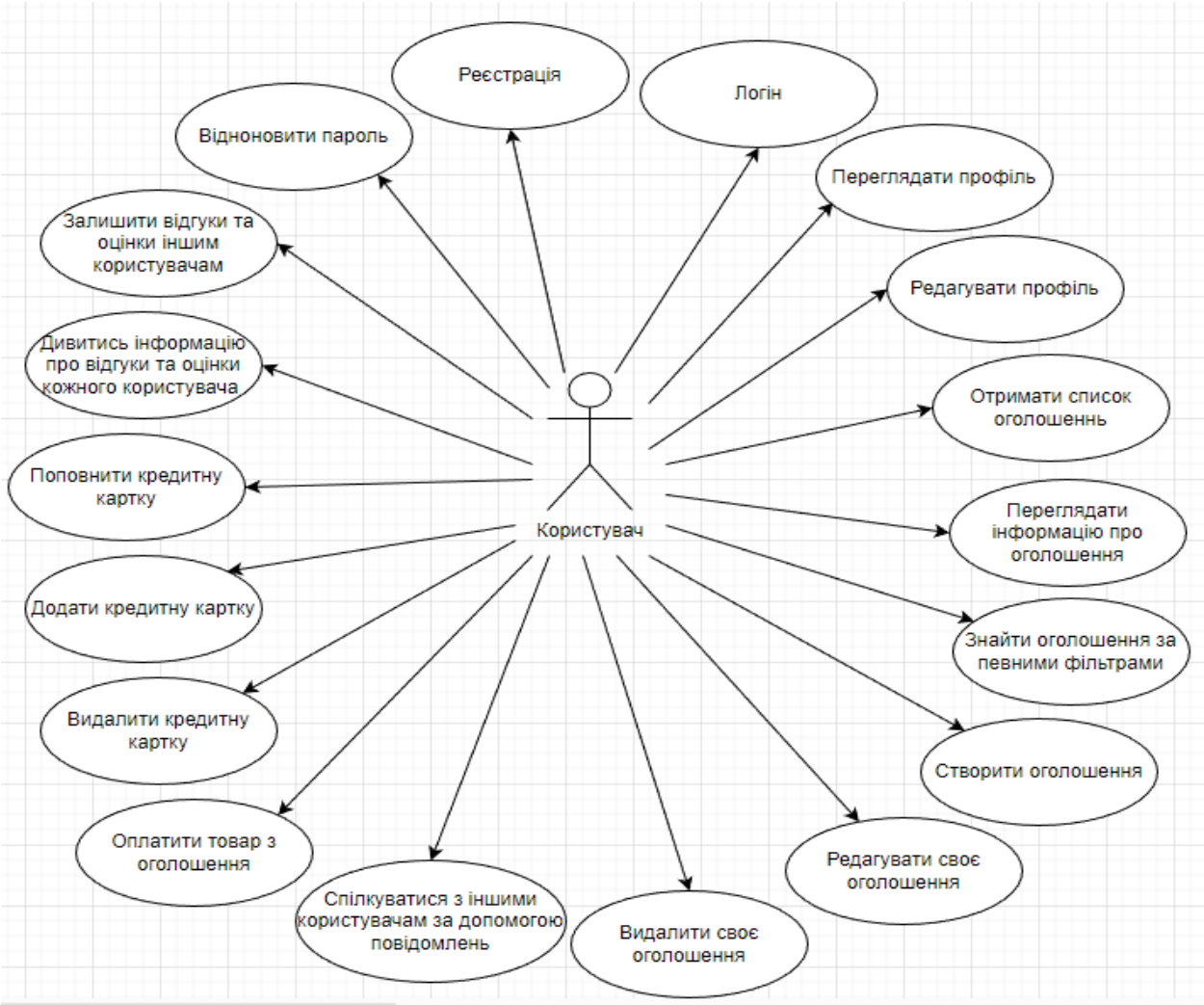

Рис. *7 Діаграма варіантів використання для користувача*

Рис. *8 Діаграма варіантів використання для адміністратора*

База даних є невід'ємною та критично важливою складовою такої системи. Вона виконує роль центрального сховища для збереження, організації та керування великим обсягом даних, що використовуються в системі.

Для створення бази даних було використано ORM Prisma. Завдяки моделі ORM (Object-Relational Mapping) ми можемо розглядати дані бази даних як об'єкти, що взаємодіють з один одним. ORM-система виконує мапінг (відображення) між об'єктно-орієнтованою моделлю даних, яка використовується в програмі, і реляційною моделлю бази даних. Завдяки використанні Prisma усі модель даних були описані у вигляді коду, використовуючи спеціальну мову запитів. Після цього Prisma автоматично

згенерувала необхідний SQL-код для створення таблиць, виконання запитів та здійснення зв'язків між таблицями [14].

Діаграма бази даних, зображена на рисунку 9, надає візуальне відображення структури всіх таблиць і полів, які використовуються для збереження інформації.

Рис. *9 Схема бази даних віртуального майданчика*

У складі бази даних присутні 17 таблиць, які містять сутності, що відповідають їхньому природному зв'язку. Кожна таблиця має унікальний первинний ключ, який ідентифікує кожен запис у таблиці.

1. Сутність «Category» представляє модель категорії оголошень.
    a. Атрибут «id» унікальний ідентифікатор категорії та її первинний ключ.
    b. Атрибут «name» унікальна назва категорії.
2. Сутність «SubCategory» представляє модель підкатегорії оголошень.
    a. Атрибут «id» унікальний ідентифікатор підкатегорії та її первинний ключ.
    b. Атрибут «name» унікальна назва підкатегорії.
    c. Атрибут «categoryId» унікальний ідентифікатор категорії з якою пов’язаний запис.
3. Сутність «CreditCard» представляє модель кредитної картки.
    a. Атрибут «id» є унікальним ідентифікатором кредитної картки та її первинний ключ.
    b. Атрибут «number» унікальний номер кредитної картки.
    c. Атрибут «money» доступна суму грошей на картці. За замовчуванням, цей атрибут має значення 0. Числа представлені з точністю до 2 десяткових знаків та зберігаються в базі даних.
    d. Атрибут «userId» унікальний ідентифікатор користувача з яким пов’язаний запис.
4. Сутність «Ban» представляє модель блокування користувача.
    a. Атрибут «id» унікальний ідентифікатор блокування та його первинний ключ.
    b. Атрибут «userId» містить ідентифікатор користувача, якому належить блокування.
    c. Атрибут «reason» причина блокування користувача.
    d. Атрибут «bannedAt» дата та час, коли було здійснено блокування.

e. Атрибут «unBannedAt» вказує на дату та час, коли було скасовано блокування. Це поле може мати значення null.

5. Сутність «ChatRoom» представляє модель чат-кімнати.
    a. Атрибут «id» є унікальним ідентифікатором чат-кімнати та її первинний ключ.
    b. Атрибут «name» є унікальною назвою чат-кімнати.
6. Сутність «Message» представляє модель повідомлення в чат-кімнаті.
    a. Атрибут «id» є унікальним ідентифікатором повідомлення та його первинний ключ.
    b. Атрибут «text» текст повідомлення.
    c. Атрибут «roomId» унікальний ідентифікатор чат-кімнати, до якої належить повідомлення.
    d. Атрибут «createdById» унікальний ідентифікатор користувача, який створив повідомлення.
    e. Атрибут «createdAt» дата та час створення повідомлення.
7. Сутність «Review» представляє модель відгуку.
    a. Атрибут «id» є унікальним ідентифікатором відгуку та його первинний ключ.
    b. Атрибут «rating» оцінка, яку користувач надав.
    c. Атрибут «comment» текстовий коментар відгуку.
    d. Атрибут «forUserId» унікальний ідентифікатор користувача, до якого відноситься відгук.
    e. Атрибут «createdById» унікальний ідентифікатор користувача, який створив відгук.
    f. Атрибут «createdAt» дата та час створення відгуку.
8. Сутність «Role» представляє модель ролі користувача
    a. Атрибут «id» унікальний ідентифікатор ролі та виступає як її первинний ключ.
    b. Атрибут «name» унікальна назва ролі.

9. Сутність «ContactInfo» представляє модель контактної інформації користувача.
    a. Атрибут «id» унікальний ідентифікатор контактної інформації та виступає як її первинний ключ.
    b. Атрибут «firstname» містить ім'я користувача.
    c. Атрибут «lastname» містить прізвище користувача.
    d. Атрибут «phone» містить номер телефону користувача.
    e. Атрибут «userId» унікальний ідентифікатор користувача, з яким пов'язана ця контактна інформація.
10. Сутність «User» представляє модель користувача.
    a. Атрибут «id» є унікальним ідентифікатором користувача та його первинний ключ.
    b. Атрибут «email» унікальна електронну пошту користувача.
    c. Атрибут «password» пароль користувача.
    d. Атрибут «nickname» псевдонім користувача.
    e. Атрибут «activationLink» зберігає посилання активації облікового запису користувача.
    f. Атрибут «isActivated» вказує на статус активації облікового запису користувача.
    g. Атрибут «avatar» містить посилання на аватар користувача.
    h. Атрибут «roleId» унікальний ідентифікатор ролі користувача, до якої він належить.
    i. Атрибут «createdAt» містить дату і час створення облікового запису користувача.
11. Сутність «Token» представляє модель токена авторизації користувача.
    a. Атрибут «id» унікальний ідентифікатор токена та його первинний ключ.
    b. Атрибут «accessToken» містить токен авторизації.
    c. Атрибут «lastAuthorization» відображає дату та час останньої авторизації з використанням даного токена.

d. Атрибут «userId» є унікальним ідентифікатором користувача, до якого належить токен.

12. Сутність «Region» представляє модель регіону.
    a. Атрибут «id» унікальний ідентифікатор області та виступає як її первинний ключ.
    b. Атрибут «name» містить назву регіону.
13. Сутність «City» представляє модель міста.
    a. Атрибут «id» унікальний ідентифікатор міста та його первинний ключ.
    b. Атрибут «name» містить назву міста.
    c. Атрибут «regionId» унікальний ідентифікатор області, до якої належить місто.
14. Сутність «Image» представляє модель зображення.
    a. Атрибут «id» є унікальним ідентифікатором зображення та виступає як його первинний ключ.
    b. Атрибут «path» містить шлях до файлу зображення.
    c. Атрибут «size» вказує розмір зображення.
    d. Атрибут «adverId» є унікальним ідентифікатором оголошення, до якого належить зображення.
15. Сутність «AdvertType» представляє модель типу оголошення.
    a. Атрибут «id» є унікальним ідентифікатором типу оголошення та виступає як його первинний ключ.
    b. Атрибут «name» унікальна назва типу оголошення.
16. Сутність «Advert» представляє модель оголошення.
    a. Атрибут «id» унікальний ідентифікатор оголошення та його первинний ключ.
    b. Атрибут «title» назва оголошення.
    c. Атрибут «description» опис оголошення.
    d. Атрибут «price» визначає ціну оголошення та має тип «Decimal» з точністю 10,2.

e. Атрибут «advertTypeId» є унікальним ідентифікатором типу оголошення, з яким пов'язане оголошення.
f. Атрибут «createdById» унікальний ідентифікатор користувача, який створив оголошення.
g. Атрибут «subCategoryId» є унікальним ідентифікатором підкатегорії, до якої належить оголошення.
h. Атрибут «locationId» є унікальним ідентифікатором місцезнаходження оголошення.

17. Сутність «Transaction» представляє модель транзакції.
    a. Атрибут «id» унікальний ідентифікатор транзакції та його первинний ключ.
    b. Атрибут «productName» містить назву продукту, що бере участь у транзакції.
    c. Атрибут «price» визначає ціну продукту в рамках транзакції.
    d. Атрибут «amount» кількість продукту.
    e. Атрибут «transactionTypeId» унікальний ідентифікатор типу транзакції, з яким пов'язана дана транзакція.
    f. Атрибут «senderId» унікальний ідентифікатор користувача, який є відправником транзакції.
    g. Атрибут «receiverId» унікальний ідентифікатор користувача, який є отримувачем транзакції.
    h. Атрибут «createdAt» дата та час створення транзакції.

3.5 Програмна реалізація додатку

3.5.1 Програмна реалізація серверної частини

У якості платформи для розробки серверної частини додатку було використано Node.js. Для поліпшення ефективності розробки та забезпечення структурованості, було обрано фреймворк NestJS.

NestJS володіє жорсткою структурою, що дозволяє створювати додатки з організованою архітектурою та зрозумілим потоком даних. Основна концепція такої структури полягає у розбитті коду на окремі модулі та компоненти [15]. У структурі додатку можна виділити декілька таких основних компонентів та їх взаємозв'язків.

Перший компонент це безпека системи що складається з двох складових які є шаром програмного забезпечення, розташованим між вхідними запитами та вихідними відповідями, а саме гуардів та стратегії (див. Рис. 10).

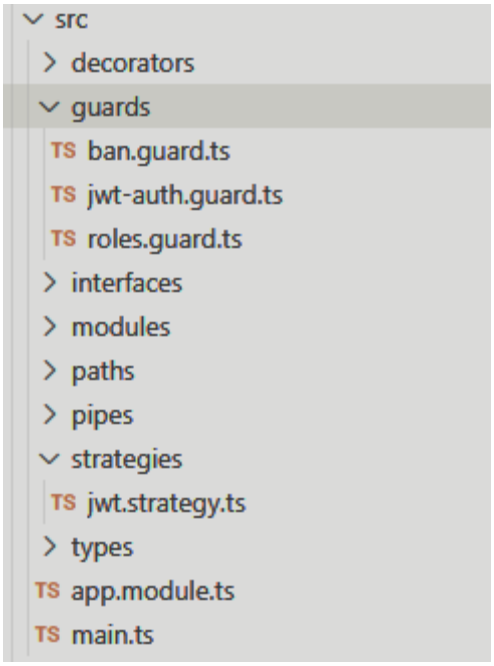

Рис. 10 Файлова структура системи захисту

У папці *strategies* знаходиться клас JwtStrategy який використовується як проміжник який виконує перевірку токену авторизація та передає отримані данні далі (див. Лістинг 1).

Лістинг *1 Процес авторизації користувача за допомогою JWT токену*

```
@Injectable()
export class JwtStrategy extends
PassportStrategy(Strategy) {
```

```
        constructor(private readonly
tokenService:TokenService) {
            super({
                jwtFromRequest:
ExtractJwt.fromAuthHeaderAsBearerToken(),
                ignoreExpiration: false,
                secretOrKey: process.env.JWT_SECRET ||
'jwtsecret'
            });
        }

        async validate(payload:JwtPayload){
            const token = await
this.tokenService.getTokenByUserId(payload.id);
            if(!token){
                throw new UnauthorizedException();
            }
            return {...payload};
        }
    }
```

У папці *guards* зберігаються файли, які містять логіку для захисту маршрутів та ресурсів у застосунках.

Клас — RolesGuard відповідає за перевірку прав доступу користувача за його роллю. У цьому класі знаходиться реалізація перевірки авторизації та відповідності ролі користувача. Якщо користувач авторизований та має необхідну роль тоді він отримає доступ до ресурсу, а інакше буде згенеровано відповідний виняток. Основним метод цього класу є canActive який використовуючи рефлектор отримує масив ролей. Якщо для даного маршруту не визначено жодної ролі, тоді дозвіл на доступ надається безпосередньо. А іншому випадку, коли ролі визначені, перевірка продовжується (див. Лістинг 2).

Лістинг *2 Перевірка прав доступу користувача за його роллю*

```
@Injectable()
export class RolesGuard implements CanActivate{
    constructor(private reflector:Reflector,
                private userService:UserService,
                private roleService:RoleService){}

    async canActivate(context: ExecutionContext) {
        try {
            const roles =
this.reflector.getAllAndOverride<string[]>(ROLES_KEY,
[context.getHandler()]);
            if(!roles)
                return true;
            const req =
context.switchToHttp().getRequest();
            if(!req.user){
                throw new HttpException('No
authorization', 401);
            }
            const user = await
this.userService.getOneByEmail(req.user.email);
            if(!user){
                throw new HttpException('No
authorization', 401);
            }
            const role = await
this.roleService.getOneById(user.roleId);
            if(!role){
                throw new HttpException('No
authorization', 401);
            }
            req.user = user;
```

```
                return roles.includes(role.name);
            } catch (error) {
                if(error instanceof HttpException)
                    throw error;
                throw new HttpException('No access', 403);
            }
        }
    }
```

Клас BanGuard — відповідає за забезпечення безпеки та захисту від заблокованих користувачів. Головним методом цього класу є canActivate який перевіряє авторизацію користувача та отримуючи масив банів перевіряє останній елемент на наявність бану. Якщо користувач не авторизований або має активну заборону, генерується відповідна помилка. Якщо всі перевірки пройдені успішно, доступ до маршруту дозволяється. (див. Лістинг 3).

Лістинг *3 Перевірка бану користувача за його масивом банів*

```
@Injectable()
export class BanGuard implements CanActivate {
    constructor(private userService: UserService) { }

    async canActivate(context: ExecutionContext) {
        try {
            const req =
context.switchToHttp().getRequest();
            if (!req.user) {
                throw new HttpException('No
authorization', 401);
            }
            const user = await
this.userService.getOneById(req.id, FullUserSelect);
            if (!user) {
```

```
                    throw new HttpException('No
authorization', 401)
                }
                if (user.bans.length > 0) {
                    const lastBan =
user.bans[user.bans.length - 1];
                    if (!lastBan.unBannedAt) {
                        throw new HttpException(`You
Banned! By reason: ${lastBan.reason}`, 403);
                    }
                }
                return true;
            } catch (error) {
                if (error instanceof HttpException)
                    throw error;
                throw new HttpException('No access', 403);
            }
        }
    }
```

Другим та головним компонентом додатку є модулі, які відповідають за різні частини функціонала такі як: авторизація, завантаження файлів, відправка пошти, виконання дій з відповідною моделлю у базі даних тощо. Ці модулі забезпечують реалізацію конкретних функцій та дозволяють користувачам взаємодіяти з різними частинами додатку. Вони інкапсулюють логіку та операції, пов'язані з відповідними функціями, та надають необхідні інтерфейси для взаємодії. Особливо, в системі присутні модулі, що керують доменами, тобто обробляють маршрути, спрямовані на конкретний ресурс. На рисунку 11 показано приклад структури одного з таких модулів.

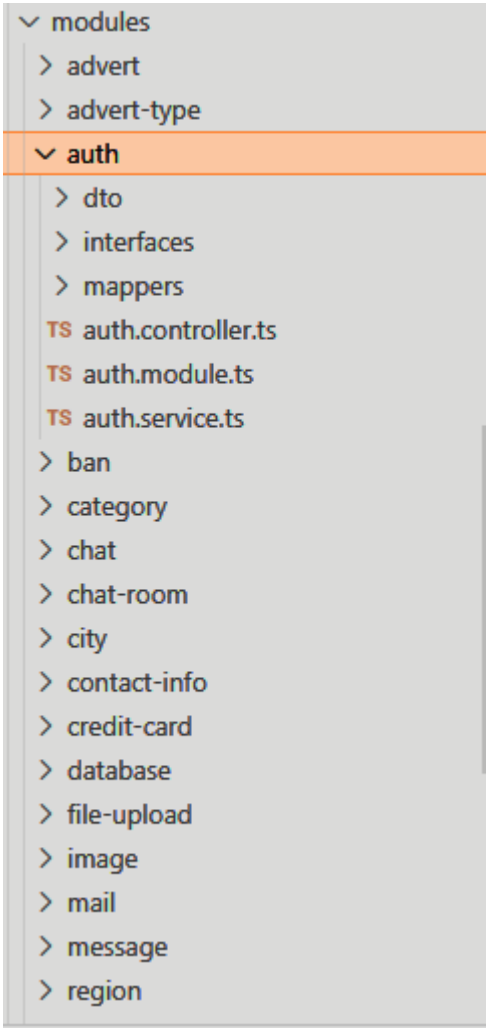

Рис. *11 Файлова структура модулів*

На цьому рисунку зображено модуль який відповідає за домен авторизації. У його структурі можна побачити такі основні складові як головний файл цього модулю, контролер та сервіс. Головний файл auth.module збирає усі складові до купи та організує цілісну структуру. Файл контролеру відповідає за обробку маршрутів, а файл сервісу за бізнес-логіку.

Розглянемо приклад одного з маршрутів, що знаходяться у контролері цього модулю, метод представлений у лістингу 4, відповідає за реєстрацію користувача.

Лістинг *4 Маршрут реєстрації користувача*

```
@Post('/register')
@HttpCode(HttpStatus.CREATED)
```

```
    async register(@Body() dto: RegisterUserDto) {
    dto = registerUserMapper.fromFrontToController(dto);
    const user = await
this.userService.getOneByEmail(dto.email);
    if (user) {
    throw new HttpException('This email is already in
use.', HttpStatus.BAD_REQUEST);
    }
    const role = await
this.roleService.getOneByName(RoleTypes.USER);
    if (!role) {
    throw new NotFoundException('The user role was not
found');
    }
    const hashPassword = await bcryptjs.hash(dto.password,
6);
    const newUser = await this.userService.create({ email:
dto.email, password: hashPassword }, role.id);
    const token = await
this.tokenService.generateToken(newUser);
    if (!token) {
    throw new HttpException('Error creating token.',
HttpStatus.BAD_REQUEST);
    }
    const link = await this.authService.generateLink();
    const linkURL = await
this.authService.createActivationLink(newUser.id, link);
    await this.userService.setActivationLink(newUser.id,
link);
    await this.mailService.sendMail(newUser.email,
'Activation link', `Your activation link: < ${linkURL} >`);
    await this.tokenService.saveToken(newUser.id, token);
```

```
    await this.contactInfoService.create({ userId:
newUser.id, firstname: "", lastname: "" });
    return { message: 'To complete registration you have to
move on the link in your email.' };
    }
```

Цей метод виконує наступні дії: перевіряє валідність вхідних даних, перевіряє чи існує дублювальний запис за електронною адресою. Далі отримується роль "користувач" за допомогою сервісу getOneByName. посилання і зберігає необхідні дані в базі даних. Потім за допомогою бібліотеки bcryptjs хешується пароль користувача. За допомогою сервісу userService та методу create створюється новий користувач з використанням електронної пошти та хешованого пароля, а також ідентифікатора ролі. Генерується токен авторизації для користувача викликом сервісу generateToken. Створюється посилання активації та генерується URL активаційного посилання. Та за допомогою поштового сервісу sendMail відправляється лист з активаційним посиланням на електронну пошту користувача. Повертається повідомлення з інструкціями для завершення реєстрації через посилання, отримане на електронну пошту.

Розглянемо ще один приклад. У контролеру Advert знаходяться методи що оброблюють маршрути по взаємодії з моделлю “Оголошення”. Розглянемо один з таких методів, представлений у лістингу 5, що відповідає за створення оголошення.

Лістинг *5 Метод створення оголошення*

```
@Post('/create')
    @UseGuards(JwtAuthGuard, BanGuard)
    @UseInterceptors(FilesInterceptor('files'))
    @HttpCode(HttpStatus.CREATED)
```

```
        async create(@UploadedFiles() files:
Express.Multer.File[], @Body() dto: CreateAdvertDto, @Req()
req: Express.Request) {
            dto =
createAdvertMapper.fromFrontToController(dto);
            const user = await
this.userService.getOneByEmail(req.user.email);
            if (!user) {
                throw new HttpException(`User with this id
is not exist.`, HttpStatus.BAD_REQUEST);
            }
            if (files && files.length > 8) {
                throw new HttpException('You cannot upload
more than 8 files', HttpStatus.BAD_REQUEST);
            }
            const subCategoryExist = await
this.subCategoryService.getOneById(dto.subCategoryId);
            if (!subCategoryExist) {
                throw new HttpException(`SubCategory with
this id is not exist.`, HttpStatus.BAD_REQUEST);
            }
            const locationExist = await
this.cityService.getOneById(dto.locationId);
            if (!locationExist) {
                throw new HttpException(`Location with this
id is not exist.`, HttpStatus.BAD_REQUEST);
            }
            const advertTypeExist = await
this.adverTypeService.getOneById(dto.advertTypeId);
            if (!advertTypeExist) {
                throw new HttpException(`AdvertType with
this id is not exist.`, HttpStatus.BAD_REQUEST);
            }
```

```
            const newAdvert = await
this.advertService.create({ createdById: user.id, ...dto
});
            if (files) {
                const filesPath = await
this.fileUploadService.uploadFiles(files,
STATIC_FOLDER_PATH + `/${user.id}`);
                const imagesData: CreateImageDto[] = [];
                filesPath.forEach((filePath, index) => {
imagesData.push({ path: path.relative(STATIC_FOLDER_PATH,
filePath), size: files[index].size, advertId: newAdvert.id
}) });
                await
this.imageService.createMany(imagesData);
            }
            return newAdvert;
        }
```

Цей код виконує створення нового оголошення. Він перевіряє різні умови, такі як наявність користувача, обмеження кількості завантажуваних файлів та наявність певних категорій і типів оголошень. Після перевірок створюється нове оголошення та, якщо вказані файли, завантажуються на сервер і зберігаються шляхи до них. На кінцевому етапі повертається створене оголошення.

Розглянемо ще один метод який знаходиться у контролеру ContactInfo та відповідає за маршрут який виконує оновлення контактної інформації користувача (див. Лістинг 6).

Лістинг 6 *Маршрут оновлення контактної інформації користувача*

```
@Patch('/update')
    @UseGuards(JwtAuthGuard)
    @HttpCode(HttpStatus.OK)
```

```
        async updateContactInfo(@Body() dto:
UpdateContactInfoDto, @Req() req: Express.Request) {
            dto =
updateContactInfoMapper.fromFrontToController(dto);
            const reqUser = req.user;
            let user = await
this.userService.getOneByEmail(reqUser.email);
            if (!user) {
                throw new NotFoundException('The user was
not found');
            }
            const contactInfo = await
this.contactInfoService.getOneByUserId(user.id);
            if (!contactInfo) {
                throw new HttpException('Contact info for
this user is not exist', HttpStatus.BAD_REQUEST);
            }
            await
this.contactInfoService.update(contactInfo.id, dto);
            return { message: "Contact info has been
updated successfully" };
        }
```

На лістингу 6 представлено код що відповідає за оновлення контактної інформації користувача. Використовуючи метод HTTP PATCH, він приймає дані користувача в об'єкті dto для оновлення. Автентифікація користувача здійснюється за допомогою JwtAuthGuard. Якщо користувач не знайдений, генерується помилка "The user was not found". Далі перевіряється наявність контактної інформації для користувача, і якщо вона не знайдена, генерується помилка "Contact info for this user is not exist". Потім відбувається оновлення контактної інформації з використанням сервісу contactInfoService. На

кінцевому етапі повертається об'єкт з повідомленням про успішне оновлення контактної інформації.

3.5.2 Програмна реалізація інтерфейсу користувача

Користувацький інтерфейс додатка побудований на основі бібліотеки React та за допомогою TypeScript для типізації коду. Структура користувацької частини була переглянута з метою розбиття монолітного механізму на більш дрібні та розділені компоненти.

Файлова структура проєкту складається з деяких частин що представлено на рисунку 12, а саме:

1. Компоненти — повторно використовувані блоки користувацького інтерфейсу.

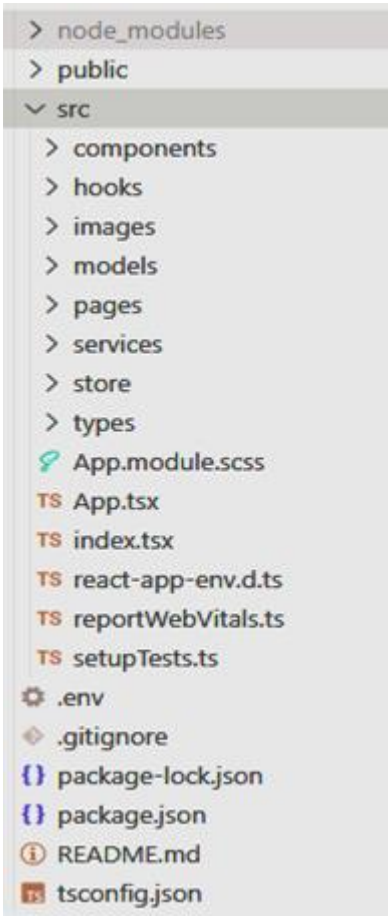

Рис. *12 Загальна файлова структура користувацької частини*

2. Сторінки — компоненти, що складаються з інших компонентів і представляють окремі веб-сторінки.
3. Моделі — структури даних, що використовуються для представлення конкретних сутностей або об'єктів у додатку.
4. Сервіси — компоненти які відповідають за отримання запитів з серверної частини.
5. Сховище (Store) — контейнери для управління станом додатка. Вони зберігають різні дані стану додатку і дозволяє компонентам отримувати доступ до цього стану та підписуватись на його зміни.

У користувацькій частині, звернення до серверного API відіграє важливу роль і має велике значення, тому для реалізації таких звернень було рішено використовувати бібліотеку Axios. Axios надає зручний спосіб виконання HTTP-запитів та дозволяє легко налаштовувати заголовки, передавати параметри запиту, обробляти відповіді сервера та керувати помилками.

На лістингу 7 зображено метод з сервісу AdvertService який виконує запит до API для отримання списку оголошень на основі переданих параметрів. В залежності від отриманих параметрів буде йти звернення по різним маршрутам що дозволяє фільтрувати та отримувати різні списки оголошень.

Лістинг *7 Метод getAdverts*

```
    async getAdverts(dto: GetAdvertsDto): 
Promise<AxiosPromise<IGetAdvertsResponse>> {
            if (dto.categoryId) {
                return 
axiosInstance.get(`/advert/allByCategory/${dto.categoryId}`
, {
                    params: {
                        ...dto
                    }
                });
```

```
            }
            if (dto.subCategoryId) {
                return
axiosInstance.get(`/advert/allBySubCategory/${dto.subCatego
ryId}`, {
                    params: {
                        ...dto
                    }
                });
            }
            return axiosInstance.get('/advert/all', {
                params: {
                    ...dto
                }
            });
        }
```

У лістингу 8 можна побачити реалізацію звернення до сервера з запитом на оновлення балансу кредитної картки з передачею токена авторизації користувача та тілом запиту на оновлення суми.

Лістинг *8 Метод topUpBalance*

```
        async topUpBalance(token: string | null, id:
string, body: TopUpMyBalanceDto) {
            return axiosInstance.patch(`/credit-
card/updateCard/${id}`, body, { headers: { Authorization:
`Bearer ${token}` } });
        },
```

Під час розробки клієнтського застосунку, централізоване сховище даних відіграє важливу роль, і для його реалізації я використовую бібліотеку Redux. Вона полегшує управління складними станами додатка, зберігаючи їх

у вигляді одного сховища та застосовуючи набір певних правил для зміни цього стану.

На лістингу 9 зображено набір правил які в залежності від дії оновлюють стан моделі оголошення.

Лістинг *9 Набір правил для зміну стану Advert*

```
export const advertReducer = (state = initialState,
action: AdvertAction): IAdvertState => {
        switch (action.type) {
            case AdvertActionTypes.SET_ADVERTS:
                return {
                    ...state,
                    currentAdverts: action.payload,
                };
            case AdvertActionTypes.SET_CURRENT_ADVERT:
                return {
                    ...state,
                    currentAdvert:
action.payload.fullAdvert,
                    avgReviewRating:
action.payload.avgReviewRating
                };
            case AdvertActionTypes.UPDATE_ADVERT:
                return {
                    ...state,
                    currentAdvert: action.payload
                };
            case AdvertActionTypes.DELETE_ADVERT:
                return {
                    ...state,
                    currentAdverts: {
```

```
                        adverts:
[...state.currentAdverts.adverts.filter(advert => advert.id
!== action.payload)],
                        total: state.currentAdverts.total
                    }
                };
            default:
                return state;
        }
    }
```

У цьому коді визначена функція advertReducer, яка приймає два параметри: state (початковий стан) та action (дія), і повертає оновлений стан об'єкта. Функція використовує оператор switch для визначення різних типів дій (actions) і відповідних змін стану. Кожний case визначає реакцію на певний тип дії, де відбувається оновлення стану моделі оголошень шляхом заміни поточних оголошень або оголошення на значення, передані через action.payload.

Наприклад:

1. У випадку AdvertActionTypes.SET_ADVERTS, стан оновлюється шляхом заміни поточних оголошень на значення, що передаються через action.payload.
2. У випадку AdvertActionTypes.SET_CURRENT_ADVERT, стан оновлюється шляхом заміни поточного оголошення та середнього рейтингу на значення, які передаються через action.payload.
3. У випадку AdvertActionTypes.UPDATE_ADVERT, стан оновлюється шляхом заміни поточного оголошення на значення, що передається через action.payload.
4. У випадку AdvertActionTypes.DELETE_ADVERT, стан оновлюється шляхом видалення оголошення зі списку поточних оголошень, використовуючи advert.id, яке передається через action.payload.

Якщо ні один з визначених типів дій не збігається, повертається незмінений початковий стан state. Такий підхід дозволив відокремити логіку оновлення стану від інших частин додатка і забезпечити більшу модульність і керованість коду.

Головною перевагою React є створення функціональних компонентів, які є основною одиницею будівництва користувацького інтерфейсу. Розглянемо приклад одного з таких компонентів що наведено нижче (Див. Лістинг 10).

Лістинг *10 Компонент ImageInput*

```
export const ImageInput: FC<IImageInput> = ({
maxImages, onImagesChange, defaultImageUrls = [] }) => {
const [imageUrls, setImageUrls] =
useState<string[]>([]);
const [images, setImages] = useState<File[]>([]);

useEffect(() => {
onImagesChange(images);
}, [imageUrls, images]);

useEffect(() => {
if (defaultImageUrls.length > 0) {
setImageUrls([...defaultImageUrls]);
convertUrlsToFiles(defaultImageUrls)
.then((files) => {
setImages(files || []);
})
.catch((error) => {
console.log(error);
});
}
}, [defaultImageUrls]);
```

```
    const handleChange = async (event:
React.ChangeEvent<HTMLInputElement>) => {
    const files = Array.from(event.target.files || []);
    if (files) {
    const newUrls: string[] = [];
    const newImages: File[] = [];
    try {
    for (const file of files) {
    if (imageUrls.length + newUrls.length >= maxImages) {
    alert(`You cannot upload more than ${maxImages}
files`);
    return;
    }
    const fileSizeInMB = file.size / (1024 * 1024);
    if (fileSizeInMB > 5) {
    alert('File size exceeds the maximum limit of 5MB.');
    return;
    }
    const imageUrl = await readFileAsDataURL(file);
    newUrls.push(imageUrl);
    newImages.push(file);
    }
    } catch (error) {
    console.log(error);
    } finally {
    setImageUrls([...imageUrls, ...newUrls]);
    setImages([...images, ...newImages]);
    }
    }
    event.target.value = '';
    };
```

```
    const deleteImage = (index: number) => {
    const newImageUrls = [...imageUrls];
    newImageUrls.splice(index, 1);
    setImageUrls(newImageUrls);

    const newImages = [...images];
    newImages.splice(index, 1);
    setImages(newImages);
    };

    return (
    <div className={styles.image_input}>
    {imageUrls && imageUrls.map((image, index) => {
    return (
    <div className={styles.image_input_item} style={{ flex:
`0 0 ${100 / maxImages}%` }}>
    <div className={styles.image_input_img_container}>
    <img key={index + Math.random()} src={image}
alt="Preview" />
    </div>
    <div className={styles.image_input_btn} onClick={() =>
deleteImage(index)}>
    <span></span>
    </div>
    </div>
    )
    })}
    {imageUrls && imageUrls.length < maxImages ?
    <input type="file" title="Клікніть щоб додати фото"
onChange={handleChange} accept="image/png, image/jpeg"
style={{ color: 'transparent',
backgroundImage:`url(${inputBackground})` }} multiple />
    : <></>}
```

```
</div>
);
};
```

Цей код представляє компонент ImageInput, який відповідає за перегляд та можливість завантаження нових зображень.

Компонент має декілька станів, які зберігаються за допомогою хука useState. Перший стан це imageUrls що зберігає масив URL-адрес зображень, які вже завантажені або вибрані користувачем, images зберігає масив завантажених файлів зображень. При зміні imageUrls або images спрацьовує useEffect, який викликає функцію onImagesChange з масивом images в якості аргументу.

Другий useEffect викликається при зміні defaultImageUrls і служить для перетворення URL-адрес зображень в файли. Завантажені файли зберігаються у масив images, а URL-адреси в imageUrls.

Функція handleChange викликається при зміні вмісту поля вибору файлу. Вона перевіряє обраний файл на обмеження щодо розміру. Якщо файли відповідають обмеженням, вони додаються до масивів newUrls і newImages, після чого оновлюються значення imageUrls і images.

Функція deleteImage видаляє зображення з масивів imageUrls і images за допомогою індексу. У поверненні компонента відбувається мапінг imageUrls для створення блоків зображень. Кожен блок містить зображення з відповідним URL-адресом та кнопку видалення. Нижче блоків зображень розташовується поле вибору файлу, якщо не досягнуто максимальної кількості зображень. На лістингу 11 наведено код що представляє компонент UserAdvertsPage, він відповідає за відображення сторінки зі списком інформації про усі оголошення користувача.

Лістинг *11 Сторінка оголошень користувача*

```
export const UserAdvertsPage = () => {
```

```
        const { adverts, total } = useTypedSelector(state
=> state.advertReducer.currentAdverts);
        const [searchParams, setSearchParams] =
useSearchParams();
        const [currentPage, setCurrentPage] =
useState<number>(1);
        const { isReady, getUserAdverts } = useAdvert();
        const dispatch = useDispatch();
        const navigate = useNavigate();
        useEffect(() => {
            const pageNumber = searchParams.get('page');
            setCurrentPage(pageNumber && +pageNumber > 0 ?
+pageNumber : 1);
            dispatch(getUserAdverts({ limit: 7, offset:
pageNumber ? (+pageNumber - 1) * 7 : 0 }));
        }, [dispatch, getUserAdverts, searchParams]);
        const handleAdvertClick = (advertId: string) => {
            navigate(`/advert/${advertId}`);
        }
        const handlePaginationClick = (pageNumber: number)
=> {
            const params = new URLSearchParams();
            params.append('page', pageNumber.toString());
            setSearchParams(params);
        }
        if (!isReady) {
            return <Loader />
        }
        return (
            <div className={styles.user_adverts}>
                <div className={styles.user_adverts_body}>
                    <h1>Мої Оголошення</h1>
                    {adverts?.map((advert) => {
```

```
                        return (<div key={advert.id}
className={styles.user_adverts_items} onClick={() =>
handleAdvertClick(advert.id)}>
                            {advert.images &&
advert.images[0] &&
                                <div
className={styles.user_adverts_img_container}>
                                    <img
src={`${process.env.REACT_APP_BACKEND_URL}/${advert.images[
0].path}`} alt={`Not load`}></img>
                                </div>}
                            <div
className={styles.user_adverts_item_info}>
                                <div
className={styles.user_adverts_item_title}>
                                    <h2>{advert.title}</h2>
                                </div>
                                <div
className={styles.user_adverts_item_location}>

<span>{`м.${advert.location.name}`}</span>

<span>{advertPrice(advert)}</span>
                                </div>
                            </div>
                        </div>)
                    })}
                    {total > 7 ? <PaginationListComponent
currentPage={currentPage} limit={7} total={total}
paginate={handlePaginationClick} />
                        : <></>}
                </div>
            </div>
```

```
        )
    }
```

У компоненті за допомогою хука useTypedSelector отримується значення adverts і total зі стану advertReducer.currentAdverts. Ці змінні містять список оголошень користувача та загальну кількість оголошень. Хук useSearchParams використовується для отримання параметрів URL-адреси. Змінна currentPage встановлюється за допомогою хука useState і відображає поточну сторінку оголошень. Хук useAdvert надає методи для отримання оголошень користувача. Хук useDispatch використовується для відправлення дій до Redux store.

Хук useEffect виконується після монтажу компонента і завантажує оголошення користувача з використанням методу getUserAdverts. Також встановлюється поточна сторінка з параметрів URL.

Функція handleAdvertClick викликається при кліку на оголошення і переадресовує користувача на сторінку конкретного оголошення.

Функція handlePaginationClick викликається при кліку на пагінацію та оновлює параметри URL для відображення правильної сторінки оголошень. Якщо оголошення ще не завантажено (isReady === false), компонент повертає компонент Loader для відображення загрузки. У протилежному випадку компонент відображає список оголошень користувача та компонент пагінації (PaginationListComponent), якщо загальна кількість оголошень більше ніж 7.

3.6 Практичне використання додатку

Згідно з вимогами до функціональності веб-додатку, була розроблена клієнтська частина, в якій успішно реалізовані всі вказані функції. Клієнтська частина була розроблена з урахуванням конкретних вимог і специфіки проєкту, забезпечуючи повний набір функцій для задоволення потреб користувачів. Кожна функція була втілена з уважністю до деталей та

ефективності, забезпечуючи зручний та інтуїтивно зрозумілий інтерфейс для користувачів.

Коли користувач відкриває вебсайт, він буде автоматично перенаправлений на головну сторінку (див. Рис. 13).

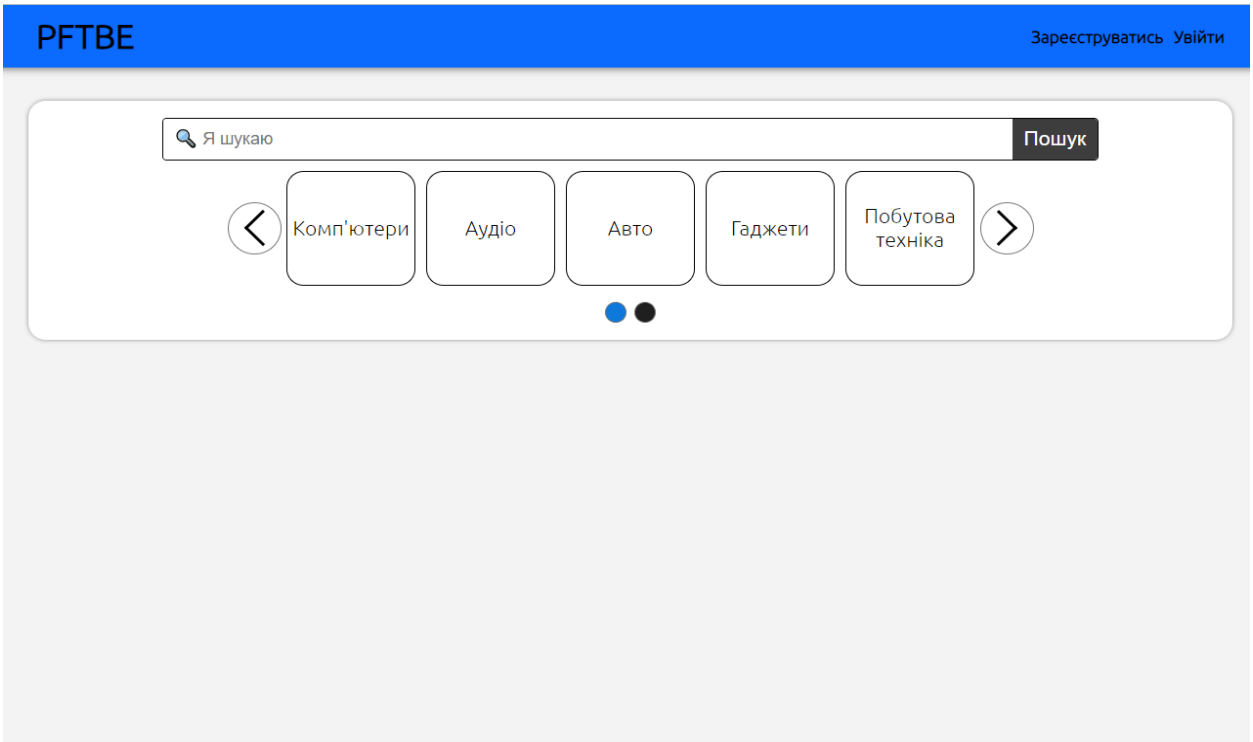

Рис. *13 Головна сторінка веб-додатку*

На головній сторінці доступне поле пошуку та слайдер для вибору категорії. Користувач може вибрати певну категорію, і після цього буде перенаправлений на сторінку зі списком оголошень, які відповідають обраній категорії (див. Рис. 14).

На цій сторінці можна побачити поля фільтрації, підкатегорії та отриманий список оголошень. Якщо натиснути на елемент зі списку оголошень користувача перенаправить на сторінку оголошення.

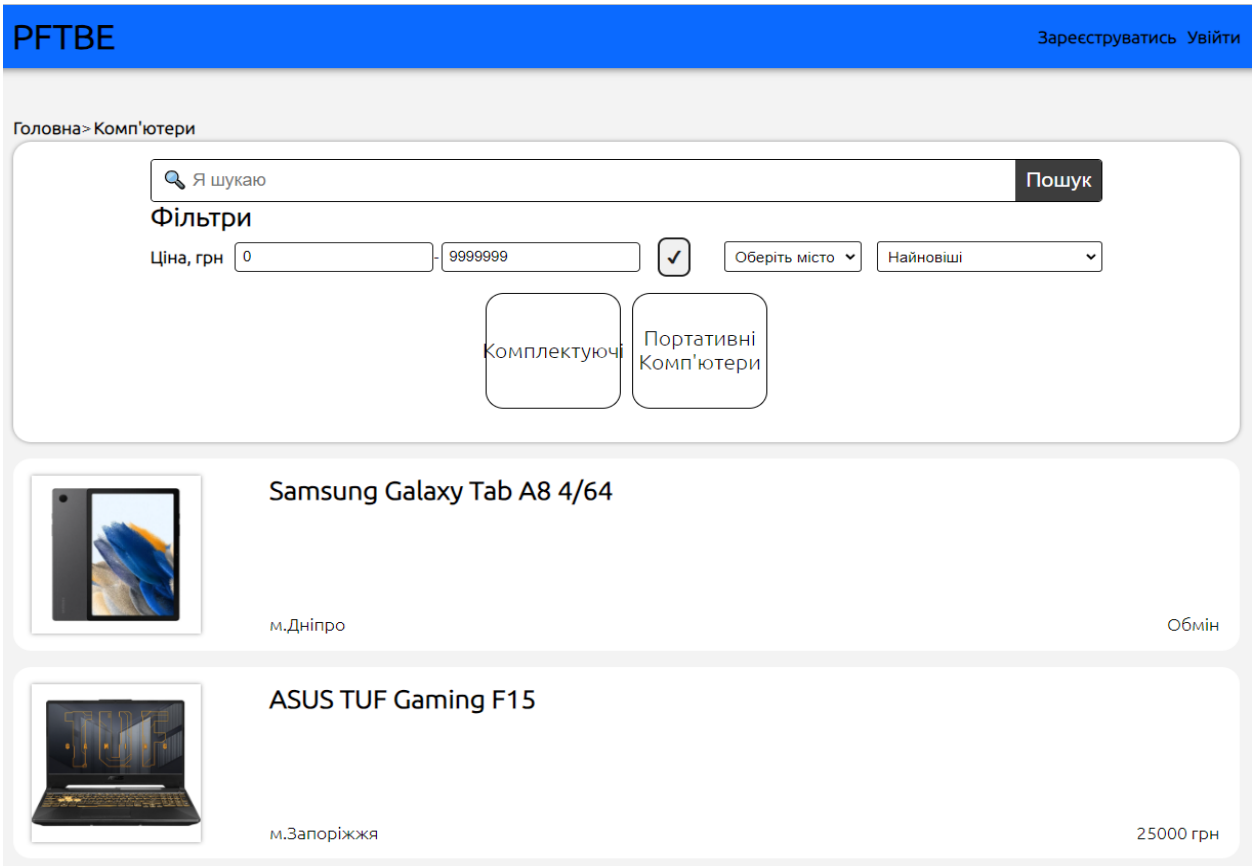

Рис. *14 Сторінка оголошень отриманих за обраною категорією*

На рисунку 15 можна побачити сторінку з повною інформацією про товар з оголошення. Тут користувач може побачити докладну інформацію про товар як розташована по різними блокам.

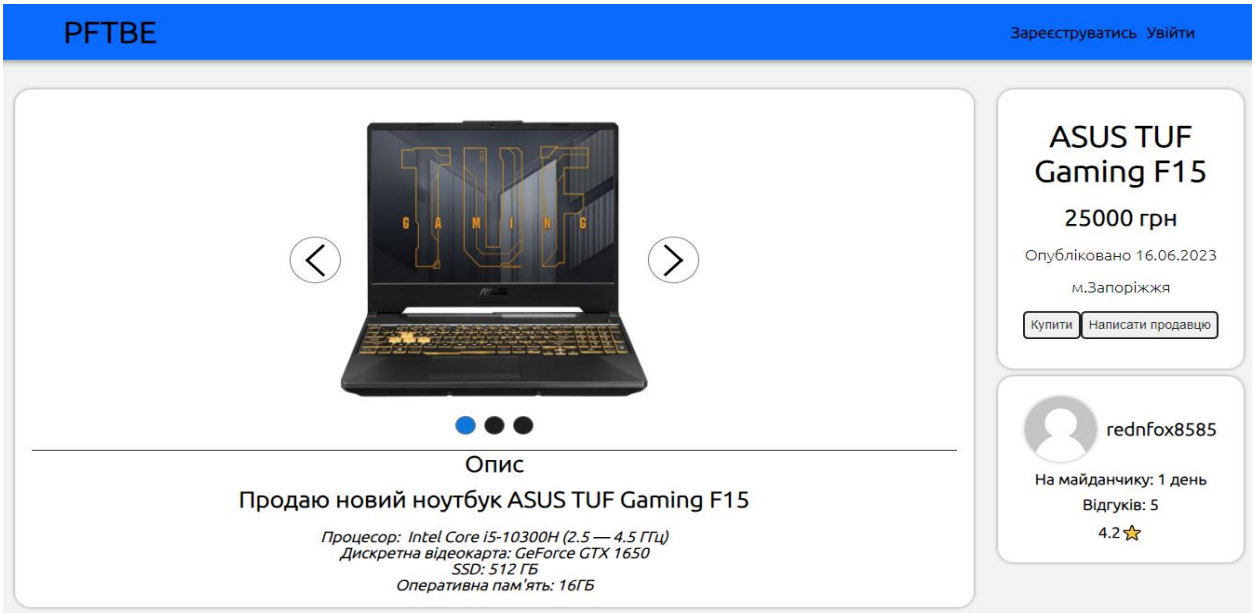

Рис. *15 Сторінка оголошення*

У найбільшому блоці знаходиться слайдер за допомогою якого можна подивитись усі зображення, а нижче знаходиться опис товару.

Праворуч на першому блоці можна побачити назву товару, ціну, дату публікації, місце знаходження та дві кнопки «Купити» та «Написати продавцю».

Другий блок містить відомість про продавця, що включає його аватар, нікнейм, дані про тривалість присутності на платформі, кількість відгуків та загальну оцінку.

Для того, щоб користувач міг створювати подібні оголошення йому потрібно увійти до свого облікового запису або зареєструватися на сайті це можна зробити за допомогою однієї з кнопок що знаходяться у правому верхньому куті. Натиснувши на кнопку зареєструватись користувач потрапить на сторінку реєстрації (див. Рис. 16).

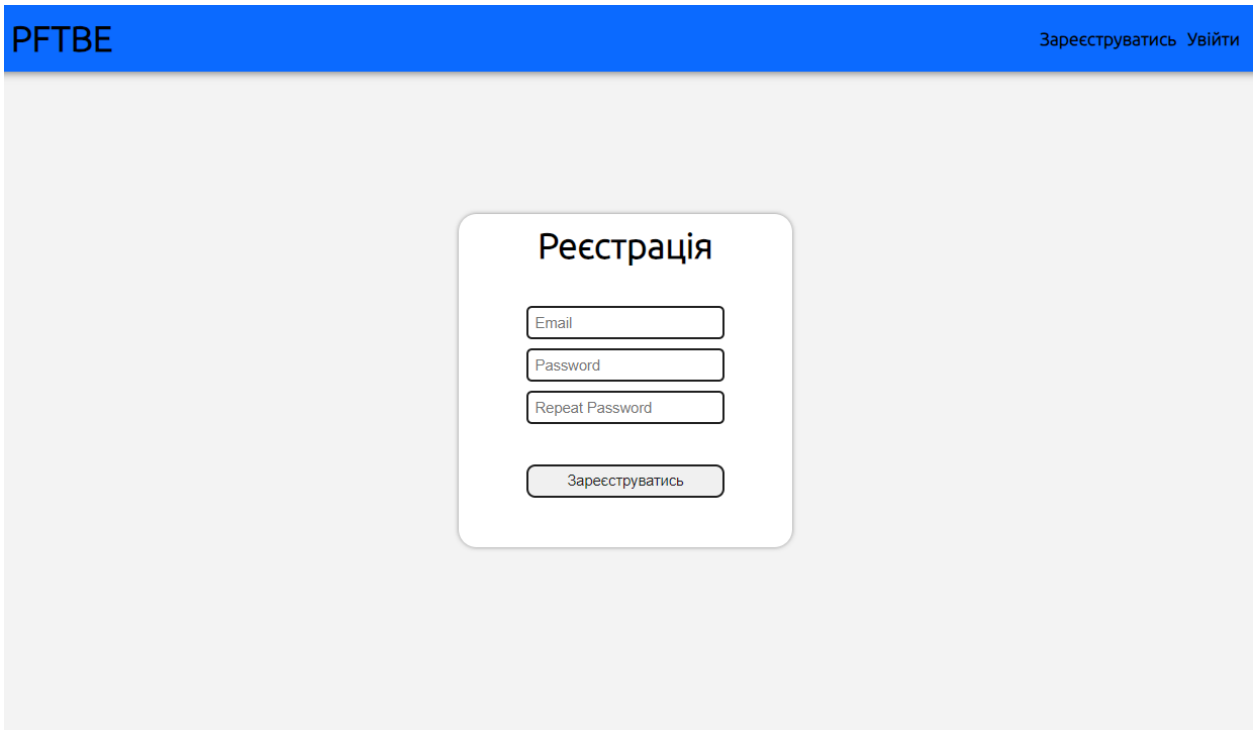

Рис. *16 Сторінка реєстрації*

Ця сторінка містить форму реєстрації, на якій користувачу потрібно ввести унікальну електронну адресу і надійний пароль за його власним

розсудом. Якщо електронна адреса є унікальною і дані введені правильно, користувач натиснувши кнопку «Зареєструватись» отримає повідомлення, в якому буде зазначено, що він має перейти за посиланням, що надіслане на його електронну пошту, для підтвердження реєстрації.

Після того як користувач перейде за посиланням процес реєстрації буде завершено і його перекине на сторінку логіну. На цю сторінку також можливо потрапити натиснувши кнопку «Увійти» що знаходиться у правому верхньому куті вебсайту. На ній буде відображена форма входу де користувачу потрібно ввести свій email та пароль, тобто дані які були використані при реєстрації (див. Рис. 17).

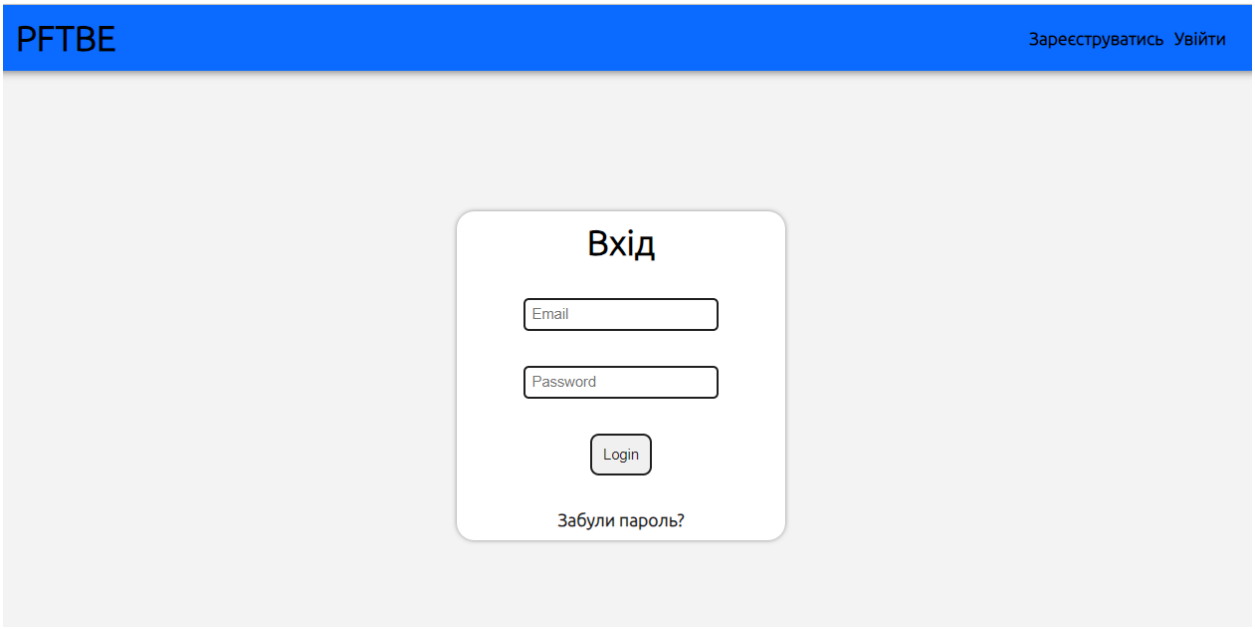

Рис. *17 Сторінка логіну*

Після успішного входу до свого облікового запису. Верхня частина зміниться в ній зникнуть кнопки «Зареєструватись», «Увійти» та з’являться нові які можна побачити на рисунку 18.

Авторизований користувач має більш розширений функціонал. Наприклад натиснувши на кнопку з нікнеймом користувача його буде перенаправлено на сторінку з формою налаштування профілю (див. Рис. 19).

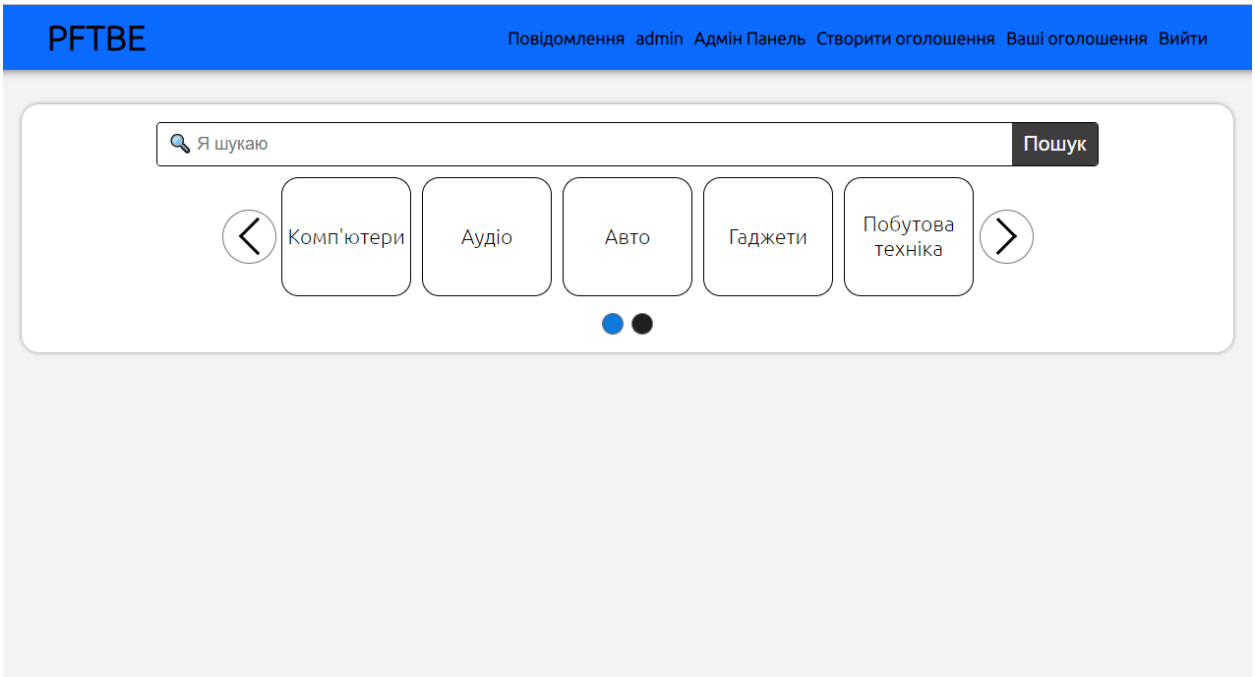

Рис. *18 Головна сторінка авторизованого користувача*

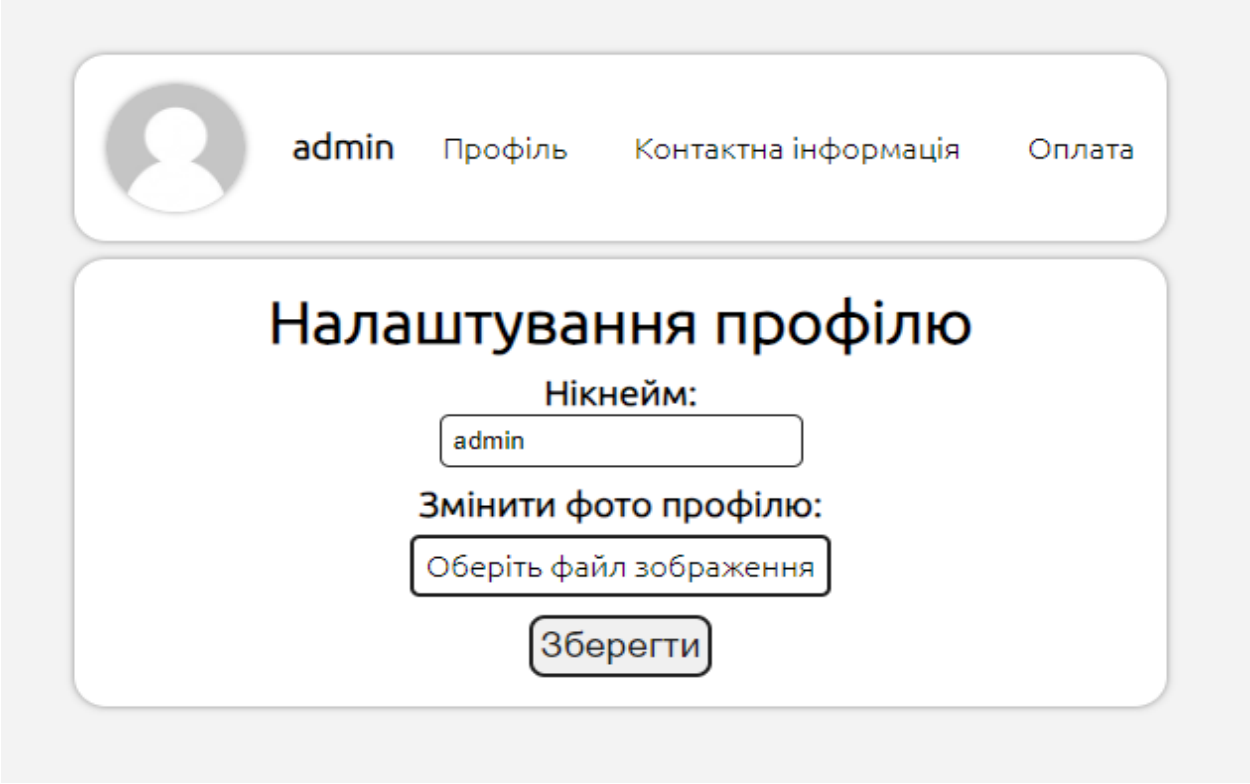

Рис. *19 Форма профілю користувача*

На цій формі з верху можна побачити фото користувача та 3 кнопки «Профіль», «Контактна інформація», «Оплата». Кожна з цих кнопок відкриває форму з відповідним налаштуванням.

У блоці знизу можна побачити саме налаштування профілю з полем для зміни нікнейму користувача, полем завантаження фото профілю та кнопкою збереження змін. Після натискання цієї кнопки будуть відображені зміни, які були внесені.

Якщо користувач бажає створити власне оголошення йому потрібно натиснути на «Створити оголошення» у верхній частині додатка після чого його перенаправить на сторінку створення оголошення (див. Рис. 20).

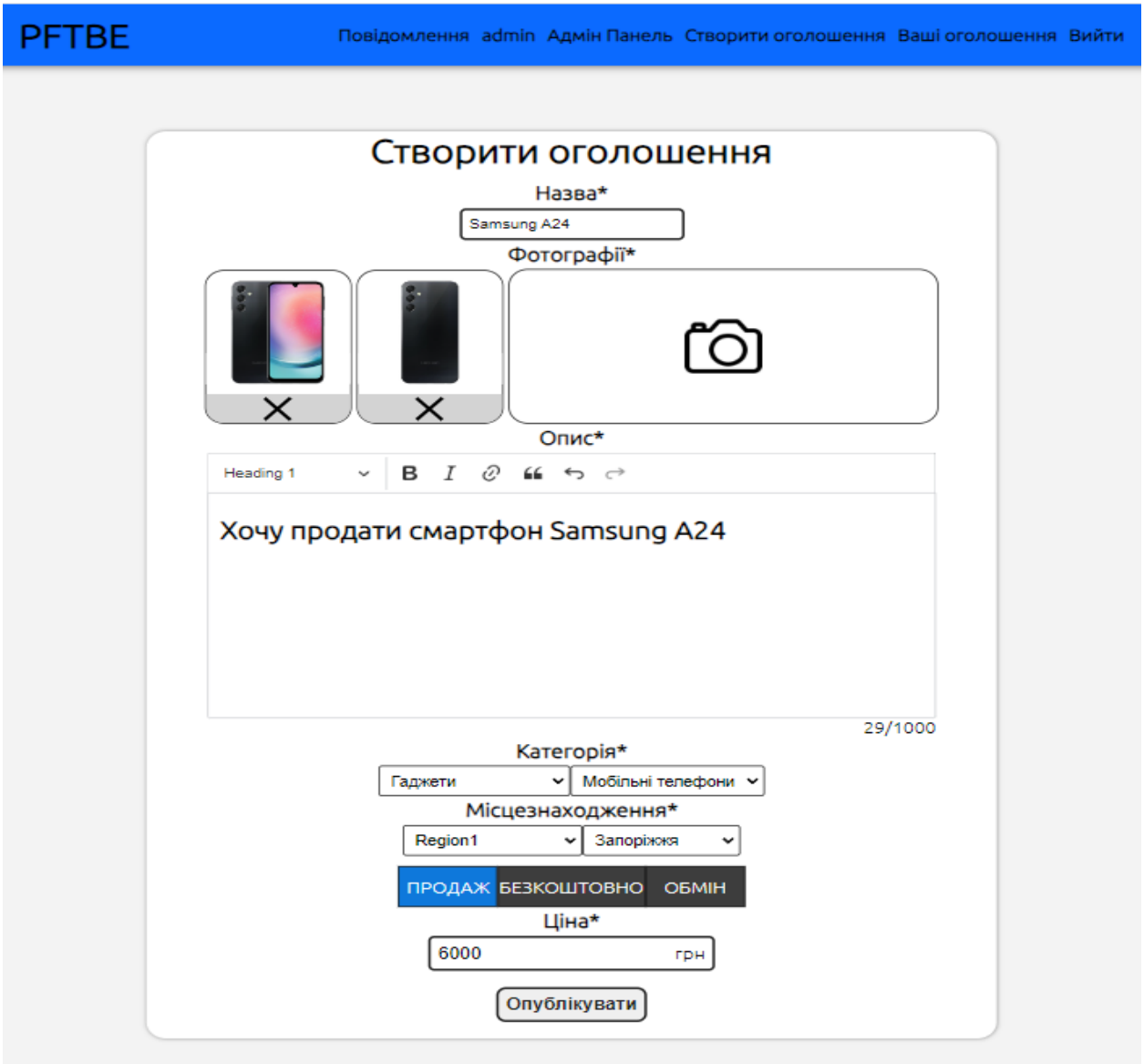

Рис. *20 Сторінка створення оголошення*

На цій сторінці можна побачити форму створення оголошення. Форма має наступні поля: назва оголошення, поле для завантаження одного або

декількох зображень, опис, випадаючий список вибору категорії та підкатегорії, випадаючий список вибору області та міста, кнопки вибору типу оголошення та поле ціни. Заповнивши всі ці поля користувач може створити оголошення.

Після створення оголошення користувача автоматично перенаправить на сторінку з його оголошеннями де першим по черзі буде йти його нове оголошення (див. Рис. 21).

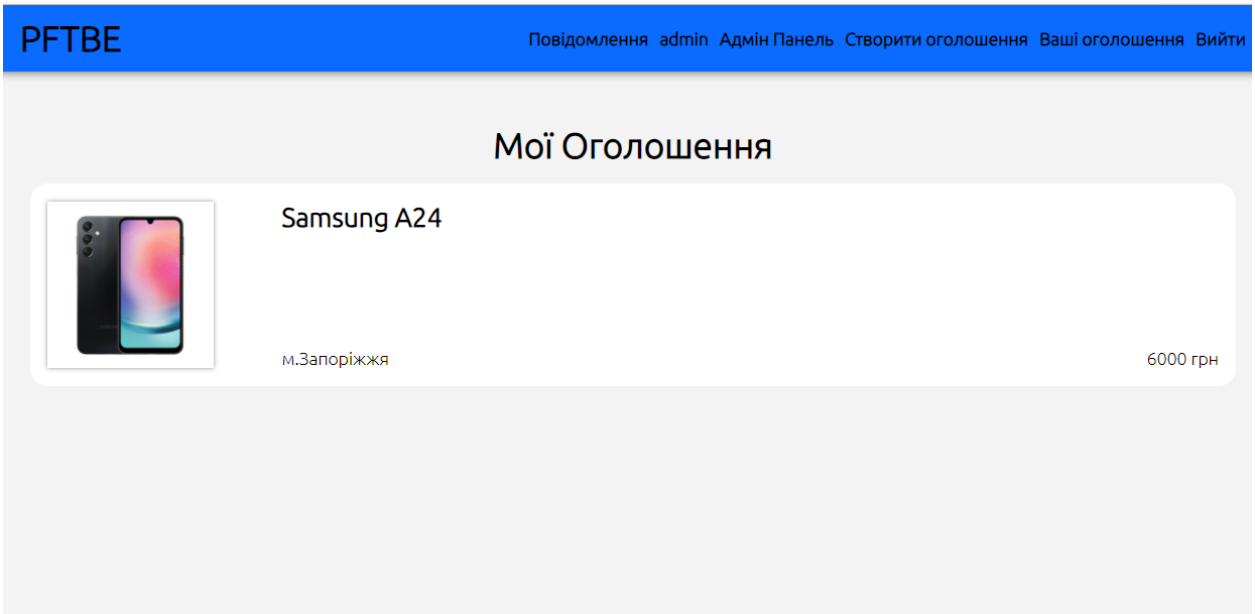

Рис. *21 Сторінка оголошенньь користувача*

Ця сторінка відображає коротку інформацію про всі оголошення, які створив користувач.

Якщо користувач натисне на блок оголошення його перенаправить на сторінку де можна буде побачити детальну інформацію про оголошення та кнопки налаштування (див. Рис. 22).

На цій сторінці можна побачити усю раніше введену інформацію про оголошення користувача, включаючи деталі, опис, фотографії тощо. Крім того, на сторінці розміщені дві кнопки — «Оновити» та «Видалити», які дозволяють користувачу оновити або видалити своє оголошення відповідно.

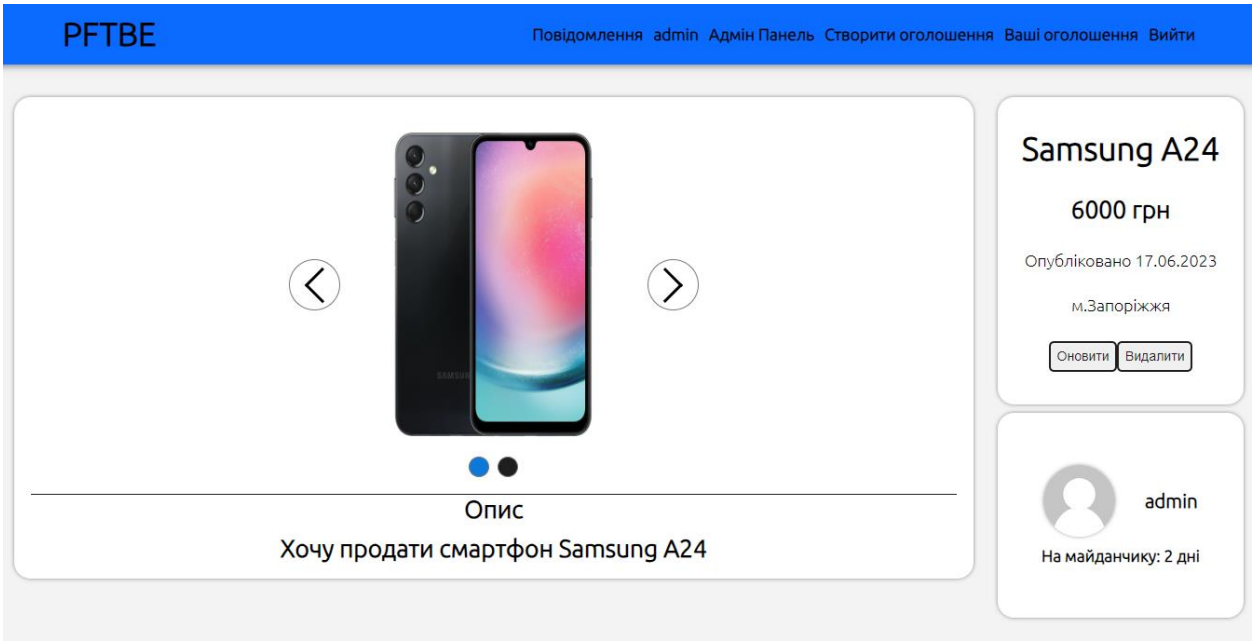

Рис. *22 Сторінка оголошення користувача*

ВИСНОВКИ

1. У процесі виконання кваліфікаційної роботи було проведено дослідження літературних джерел за темою створення веб-додатків електронної комерції та проаналізовано продукти аналоги. Це надало глибоке розуміння теоретичних аспектів та практичних реалій створення веб-додатків електронної комерції.
2. Аналіз доступних інформаційних матеріалів та результатів наукових досліджень підтвердив, що розробники та дослідники активно приділяють увагу IT технологіям, спрямованим на поліпшення процесів електронної комерції, швидку обробку та передачу інформації, ефективну комунікацію з клієнтами та оптимізацію взаємодії з базами даних. Саме тому метою розробки став веб-додаток віртуального майданчика торгівлі та бартерного обміну.
3. Після проведення аналізу поточного стану проблеми, було складено технічне завдання та розроблено план робіт. Перед початком реалізації було проведено проектування архітектури системи, бази даних та визначено варіанти використання. Розроблена система повністю задовольняє всі встановлені функціональні вимоги.
4. Реалізована серверна частина, яка забезпечує надійну та ефективну роботу системи. Вона включає в себе розроблені модулі, сервіси та компоненти, що взаємодіють між собою з метою швидкої обробки даних та виконання запитів.
5. Було розроблено інтерфейс користувача, що дозволяє зручно та інтуїтивно взаємодіяти з системою.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Краус К. М., Краус Н. М., Манжура О. В. Електронна комерція та Інтернет-торгівля : навч.-метод. посібник. Київ : Аграр Медіа Груп, 2021. 454 с.
2. Смолій, Л., & Костюк, В. (2021). НОВІТНІ ТРЕНДИ ТА ПЕРСПЕКТИВИ РОЗВИТКУ ЕЛЕКТРОННОЇ КОМЕРЦІЇ В МІЖНАРОДНОМУ БІЗНЕСІ. Економіка та суспільство, (29). URL: https://doi.org/10.32782/2524-0072/2021-29-43. (дата звернення: 30.04.2023).
3. Про нас OLX : веб-сайт. URL: https: https://blog.olx.ua/pro-nas-uk/ (дата звернення: 25.04.2023).
4. Головна сторінка Ebay : веб-сайт. URL:https://www.ebay.com/ (дата звернення: 28.04.2023).
5. Головна сторінка Prom : веб-сайт. URL: https://zp.prom.ua/ua/ (дата звернення: 04.05.2023).
6. Andrew Lock. ASP.NET Core in Action, Second Edition : Manning Publications, 2021, 832 p.
7. Iuliana Cosmina, Rob Harrop, Chris Schaefer, Clarence Ho. Pro Spring 5: An In-Depth Guide to the Spring Framework and Its Tools Fifth Edition: Apress, 2017. 878 p.
8. Mike Cantelon, Marc Harter, T.J. Holowaychuk, and Nathan Rajlich. Node.js in Action. Second Edition : Manning Publications, 2017. 392 p.
9. Miguel Grinberg. Flask Web Development: Developing Web Applications with Python 2nd Edition. O'Reilly Media, 2018. 312p.
10. Regina Obe (Author), Leo Hsu. PostgreSQL: Up and Running: A Practical Guide to the Advanced Open Source Database 3rd Edition. O'Reilly Media, 2017. 312p.
11. Alex Banks and Eve Porcello. Learning React: Functional Web Development with React and Redux 1st edition. O Reilly, 2017. 350p.

12. Nathan Murray , Felipe Coury , Ari Lerner , Carlos Taborda ng-book: The Complete Guide to Angular 5th Edition. CreateSpace Independent Publishing Platform, 2018. 626p.
13. Andrea Passaglia. Vue.js 2 Cookbook: Build modern, interactive web applications with Vue.js. Packt Publishing, 2017, 454p.
14. Документація до Prisma ORM: веб сайт. URL: https://www.prisma.io/docs. (дата звернення: 06.05.2023).
15. Документація до NestJS : веб сайт. URL: https://docs.nestjs.com. (дата звернення: 06.05.2023).
16. Булигін В.В., студент 4 курсу ІННІ ім. Ю.М. Потебні ЗНУ. Наук. кер.: к.т.н., доц. Міхайлуца О.М. «Використання React та Node.js для створення віртуального майданчика торгівлі та бартерного обміну». Збірник наукових праць студентів, аспірантів і молодих вчених «Молода наука-2023». Запоріжжя : ЗНУ, 2023. Т.5. С. 78-80.