

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

«Інтернет-платформа маркетплейсу "Fox"»

Виконав студент групи ВТп-21-1
ВИГНИЧ Олександр Сергійович

Керівник роботи:
ЧИЖМОТРЯ Олена Геннадіївна

Рецензент:
САВІЦЬКИЙ Роман Святославович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«1» серпня 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ВИГНИЧ Олександр Сергійович**

Керівник роботи: **ЧИЖМОТРЯ Олена Геннадіївна**

Тема роботи: **«Інтернет-платформа маркетплейсу "Fox"»,**

затверджена Наказом закладу вищої освіти від **«1» серпня 2025 р., №270/с**

Вихідні дані для роботи: **Предметом дослідження є технології та методи проєктування і розробки веб-платформи системи маркетплейсу**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. Г.	01.08.2025р.	01.08.2025р.
2	Чижмотря О. Г.	01.08.2025р.	01.08.2025р.
3	Чижмотря О. Г.	01.08.2025р.	01.08.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра присвячена розробці веб-орієнтованої інтернет-платформи маркетплейсу та містить пояснювальну записку.

Пояснювальна записка до кваліфікаційної роботи складається з переліку умовних скорочень, вступу, трьох розділів, висновків, списку використаних джерел та додатків.

Текстова частина оформлена на 54 сторінках друкованого тексту. Список використаних джерел включає 30 найменувань і займає 3 сторінки. У роботі представлено 24 рисунків та 11 таблиць. Загальний обсяг роботи становить 62 сторінок. У випускній кваліфікаційній роботі розглядається задача проєктування та реалізації інформаційної системи маркетплейс-платформи, що забезпечує взаємодію між продавцями та покупцями. Основну увагу приділено архітектурі системи, аналізу аналогічних рішень, вибору технологічного стеку та реалізації функціональних модулів, зокрема: системи аутентифікації та авторизації, керування товарами, обробки замовлень, платіжних операцій, фільтрації контенту, візуалізації аналітики продажів та обробки користувацьких повідомлень.

Запропонована система реалізована за принципами клієнт-серверної архітектури. Клієнтська частина розроблена з використанням ReactJS, серверна — на базі NEXT.js. Для зберігання та обробки даних застосовано СУБД PostgreSQL.

Платформа дозволяє ефективно автоматизувати процеси торгівлі, забезпечує гнучке адміністрування, масштабованість, та адаптацію під потреби цільової аудиторії.

Ключові слова: МАРКЕТПЛЕЙС, ВЕБ-ПЛАТФОРМА, REACT, NEXTJS, БАЗА ДАНИХ, АВТОМАТИЗАЦІЯ ПРОДАЖІВ, РЕКОМЕНДАЦІЇ.

					ІПЗ.КР.Б - 121 - 25 - ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Вигнич О.С.			«Інтернет-платформа маркетплейсу "Fox"»	Літ.	Арк.	Аркушів	
Керівник		Чижморя О.Г.					3	62	
Рецензент		Савіцький Р.С.			Пояснювальна записка	Житомирська політехніка, група ВТп-21-1			
Зав. каф.		Вакалюк Т.А.							

ABSTRACT

The Bachelor's qualification thesis is dedicated to the development of a web-based marketplace platform and includes an explanatory report.

The explanatory report consists of a list of abbreviations, an introduction, three chapters, conclusions, a list of references, and appendices.

The textual part is presented on 54 pages of printed material. The list of references includes 30 sources and occupies 3 pages. The thesis contains 24 figures and 11 tables. The total length of the work is 62 pages. The qualification thesis addresses the design and implementation of an information system for a marketplace platform that facilitates interaction between sellers and buyers. Particular attention is given to the system architecture, analysis of similar solutions, selection of the technology stack, and implementation of functional modules, including: authentication and authorization system, product management, order processing, payment operations, content filtering, sales analytics visualization, and user message handling.

The proposed system is implemented following the principles of client-server architecture. The client side is developed using ReactJS, while the server side is built on the NEXT.js framework. PostgreSQL is used for data storage and processing.

The platform enables efficient automation of trade processes, provides flexible administration, scalability, and adaptability to the needs of the target audience.

Keywords: MARKETPLACE, WEB PLATFORM, REACT, NEXTJS, DATABASE, SALES AUTOMATION, RECOMMENDATIONS.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ	9
1.1. Визначення та постановка задачі	9
1.2. Аналіз аналогів програмного продукту.....	11
1.3. Визначення архітектури програмного забезпечення	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до програмного забезпечення.....	18
Висновки до першого розділу	21
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ МАРКЕТПЛЕЙСУ.....	22
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	22
2.2. Розробка бази даних системи	28
2.3. Проєктування та реалізація алгоритмів роботи системи.....	33
2.4. Реалізація функціоналу платформи	36
Висновки до другого розділу	39
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ	41
3.1. Розгортання платформи	41
3.2. Структура інтерфейсу та користувацька взаємодія	42
3.3. Тестування роботи програмного продукту	48
Висновки до третього розділу	49
ВИСНОВКИ	50
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	52
ДОДАТКИ	55

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

TS – TypeScript.

БД – база даних.

API – Application programming interface

REST API – Representational State Transfer Application Programming Interface.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У пояснювальній записці буде описано процес створення онлайн-платформи маркетплейсу. Ця платформа призначена для електронної комерції, де кожен користувач може заснувати та розвивати власний бренд. Вона забезпечує швидку та надійну обробку великих обсягів даних, а також сприяє зручній взаємодії з товарами та проведенню аналітики.

Актуальність теми. полягає у тому електронна комерція продовжує швидко розвиватися, і все більше підприємців і споживачів переходять до онлайн-покупок. Маркетплейси, як платформи для продажу товарів і послуг, стають все більш популярними завдяки зручності, широкому асортименту. Для малих та середніх підприємств маркетплейси є важливим каналом збуту, оскільки вони можуть використовувати існуючу інфраструктуру платформи для просування та продажу своїх товарів без необхідності розробки власних.

складних технічних рішень.

Завданням є виконання таких задач:

- аналіз існуючих маркетплейсів, визначення унікальності;
- визначення інформаційних потреб предметної області дослідження;
- розробка адаптивного інтерфейсу веб-сайту за допомогою React[17] та
- супутніх бібліотек і фреймворків;
- проєктування бази даних платформи;
- розробка математичної та алгоритмічної моделі функціонування
- системи на основі бази даних;
- написання серверної логіки;

Мета випускної кваліфікаційної роботи є проєктування та розробка веб-орієнтованої інтернет-платформи маркетплейсу, яка забезпечує повноцінну взаємодію між продавцями та покупцями в онлайн-середовищі.

Об'єктом дослідження виступають процеси обробки принципів онлайн торгівлі.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Предметом дослідження є технології та методи розробки інформаційних систем для безпосереднього процесу торгівлі через мережу.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1. Визначення та постановка задачі

Основною метою роботи є розробка інтернет-платформи маркетплейсу, яка дозволить підприємцям автоматизувати процес управління продажами, аналізу ринку та взаємодії між продавцями та покупцями. Платформа забезпечить підприємцям зручний інтерфейс для створення та просування власних онлайн брендів, швидку обробку.

Сучасні підприємці-початківці стикаються з проблемою відсутності ефективного інструменту для масштабування свого онлайн-бізнесу. Багато наявних платформ є складними для інтеграції, мають обмежені аналітичні можливості або не забезпечують належного рівня автоматизації. Це ускладнює підприємцям процес ведення бізнесу, уповільнює прийняття стратегічних рішень та знижує конкурентоспроможність на ринку.

Також існують такі ключові проблеми:

- Відсутність універсальної платформи для всіх потреб малого бізнесу – малий бізнес вимушений використовувати багато різних інструментів для роботи з замовниками.

- Обробка великої кількості даних – треба працювати з даними аналітики для продажів, для товарів, обробляти інформацію по клієнтам, вартість реклама тощо.

- Обмежена автоматизація бізнес-процесів – нестача інтеграції з платіжними системами, службами доставки та CRM-рішеннями змушує підприємців витратити більше часу на операційну діяльність.

Для вирішення зазначених проблем необхідно створити інтернет-платформу маркетплейсу, яка забезпечить зручне та ефективне управління онлайн-продажами. Ця система повинна надавати такі можливості:

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

1. Візуалізація інтуїтивно зрозумілого інтерфейсу для перегляду даних, що допоможе продавцям аналізувати ефективність своїх товарів та маркетингових кампаній.
2. Динамічне оновлення інформації – автоматичне оновлення товарів, замовлень і операцій без необхідності ручного введення даних.
3. Система пошуку – впровадження інструментів швидкого пошуку.
4. Багаторівневий доступ – розмежування прав користувачів, зокрема продавців, покупців та адміністраторів, для ефективного керування платформою.
5. Прогнозування попиту та трендів – використання алгоритмів аналізу історичних продажів і поведінки покупців для прогнозування майбутнього попиту та формування рекомендацій щодо товарного асортименту.

Основні етапи розробки інтернет-платформи маркетплейсу включають:

1. Проєктування інтерфейсу веб-застосунку
 - розробка концепції дизайну інтерфейсу користувача;
 - створення макетів основних сторінок платформи, таких як каталог товарів, кошик, сторінка продавця та адміністративна панель.
2. Розробка архітектури клієнтської частини
 - визначення структури додатку для зручності навігації та взаємодії з користувачем;
 - інтеграція необхідних бібліотек і фреймворків для динамічного оновлення контенту та ефективної взаємодії з сервером.
3. Розробка серверної частини
 - створення API для обробки запитів, керування товарами, замовленнями та користувачами;
 - проєктування бази даних для зберігання інформації про продукти, продавців, транзакції та клієнтів;
4. Розробка основного функціоналу

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

- реалізація механізму реєстрації та авторизації користувачів (як для продавців, так і для покупців);
- створення можливості додавання, редагування та видалення товарів;
- створення можливості додавання та редагування товарів, управління власним брендом;
- розробка системи управління замовленнями;

5. Тестування функціоналу системи

- перевірка функціональності всіх модулів платформи;
- тестування взаємодії між клієнтською та серверною частинами;
- оцінка продуктивності, безпеки даних та стійкості до навантажень.

Результатом роботи буде інтернет-платформа маркетплейсу, що сприятиме централізованому зберігання, попиту та користувальницької поведінки. Завдяки цьому продавці та адміністрація платформи зможуть ефективно контролювати ефективність товарів, керувати замовленнями та передбачати майбутні тренди на ринку.

1.2. Аналіз аналогів програмного продукту

Сайти, що реалізують функціональність інтернет-маркетплейсів, можна умовно поділити на кілька типів. Деякі з них спеціалізуються на окремих категоріях товарів або послуг – наприклад, Etsy[19] орієнтується на вироби ручної роботи та вінтаж, а Booking.com[20] фокусується на бронюванні житла. Інші платформи мають універсальний підхід: вони охоплюють широкий спектр товарів і послуг, як от Amazon, eBay або українська Rozetka.

Також варто виділити так звані самостійні маркетплейси – це сервіси, які повністю контролюються однією компанією (рис. 1.1). Серед таких можна знову ж згадати Amazon чи Etsy.

Щоб краще зрозуміти, як працює спеціалізований маркетплейс, звернемо увагу на приклад «Rozetka» (рис. 1.2). Цей інтернет-магазин став одним із найвідоміших в Україні завдяки широкому вибору товарів(рис. 1.3) за привабливими цінами. Його перевагою є пряма співпраця з виробниками, що

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяє тримати конкурентні ціни. Оформлення замовлення займає мінімум часу, а доставка працює швидко та стабільно. Крім того, клієнти можуть обрати зручний для себе спосіб оплати, що робить процес покупки максимально простим і зрозумілим[2].

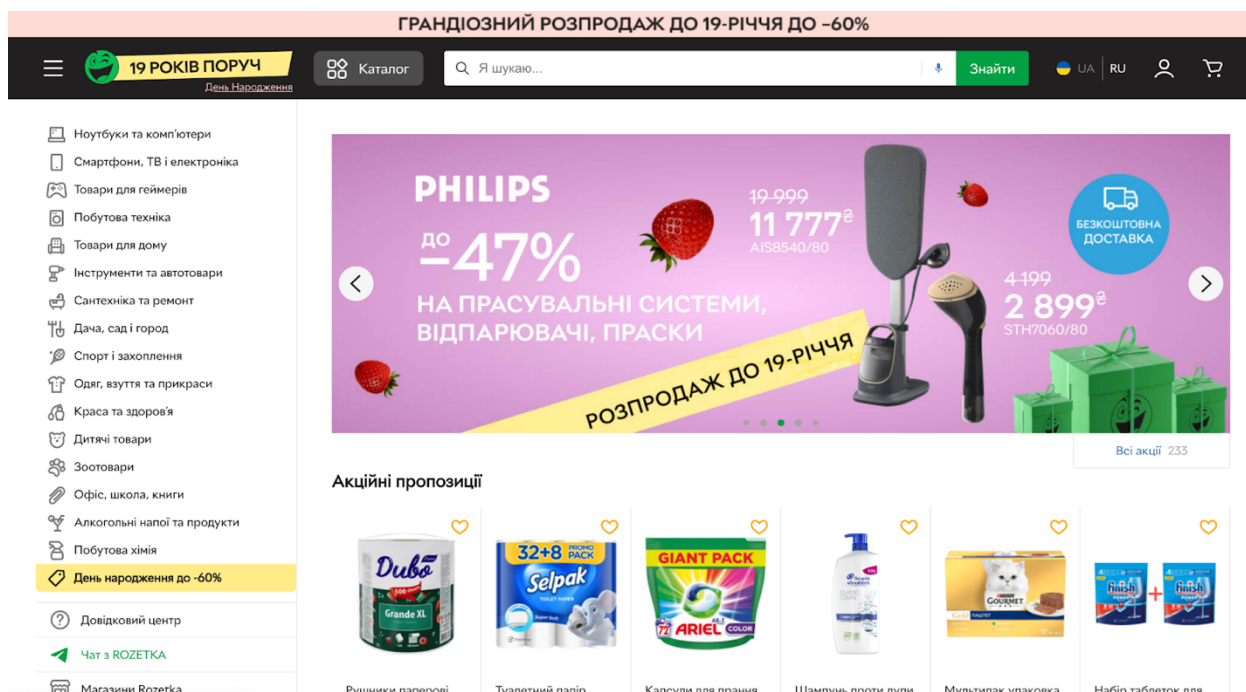


Рис. 1.1. Маркетплейс Rozetka

Сайт має приємну кольорову гаму та загалом виглядає доволі гармонійно завдяки охайному дизайну. Проте в ньому є й недоліки: надмірна кількість деталей у візуальному оформленні може відволікати, відсутня можливість додавати товари в улюблене, а тривалий час очікування відповіді від сервера значно погіршує зручність користування.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						12
Змн.	Арк.	№ докум.	Підпис	Дата		

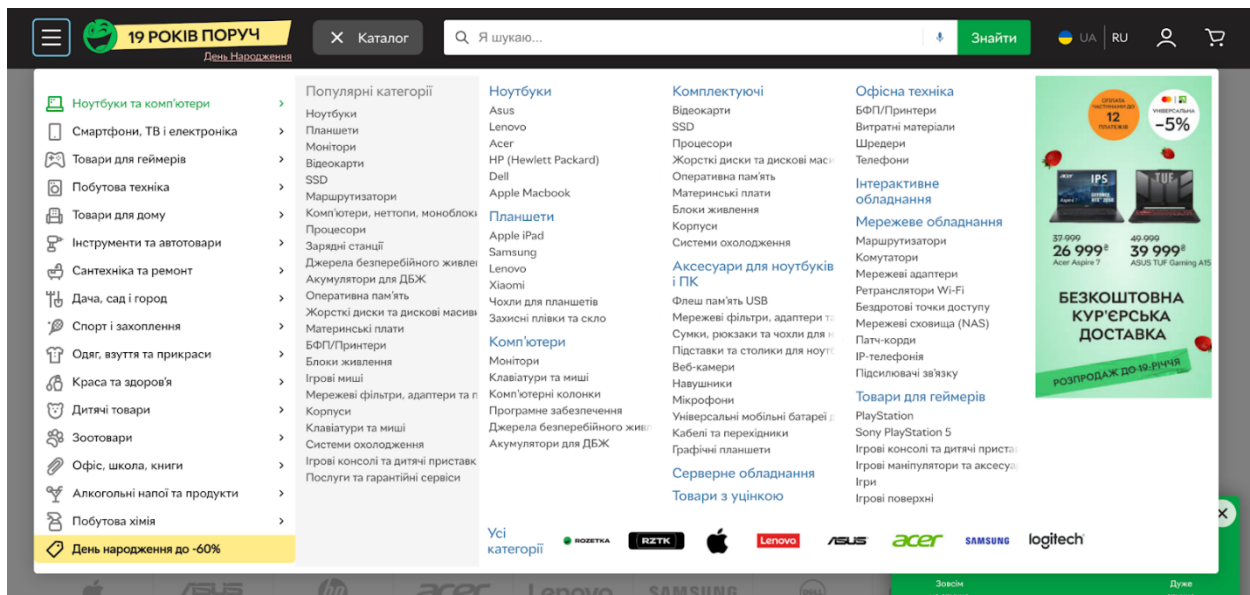


Рис. 1.2. Каталог магазину Rozetka

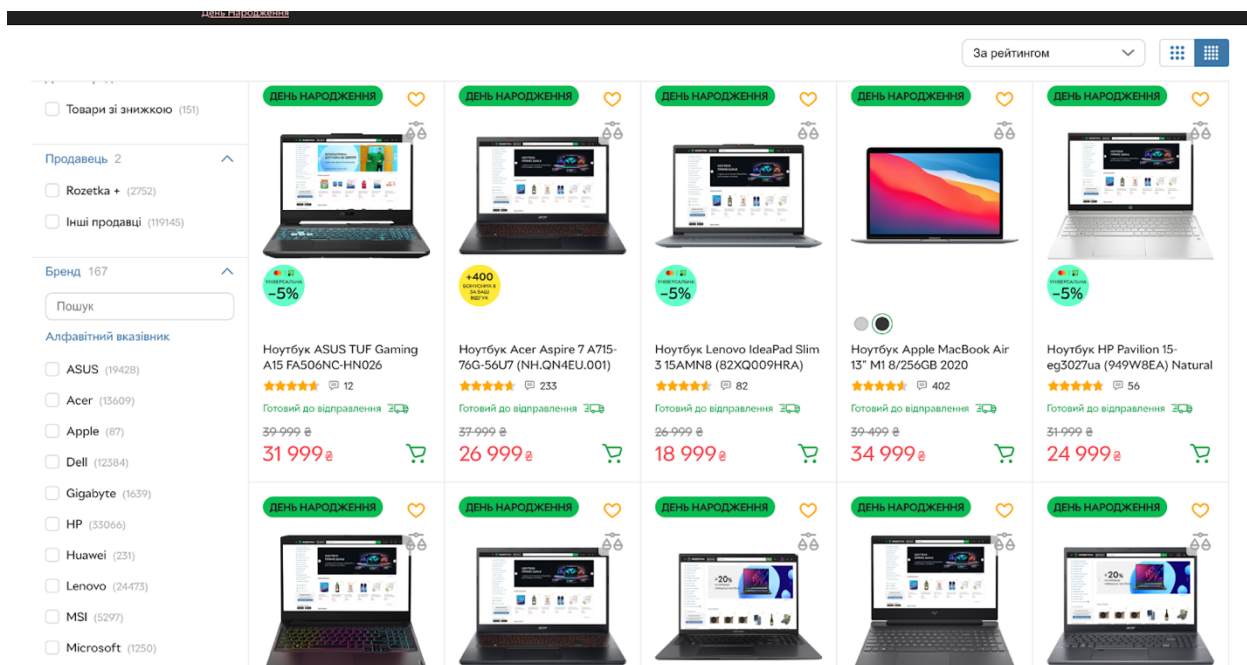


Рис. 1.3. Товари в магазині Сатин

Сайт «Prom» пропонує широкий асортимент товарів та зручний перехід між ролями[1] (рис. 1.4), (рис. 1.5).

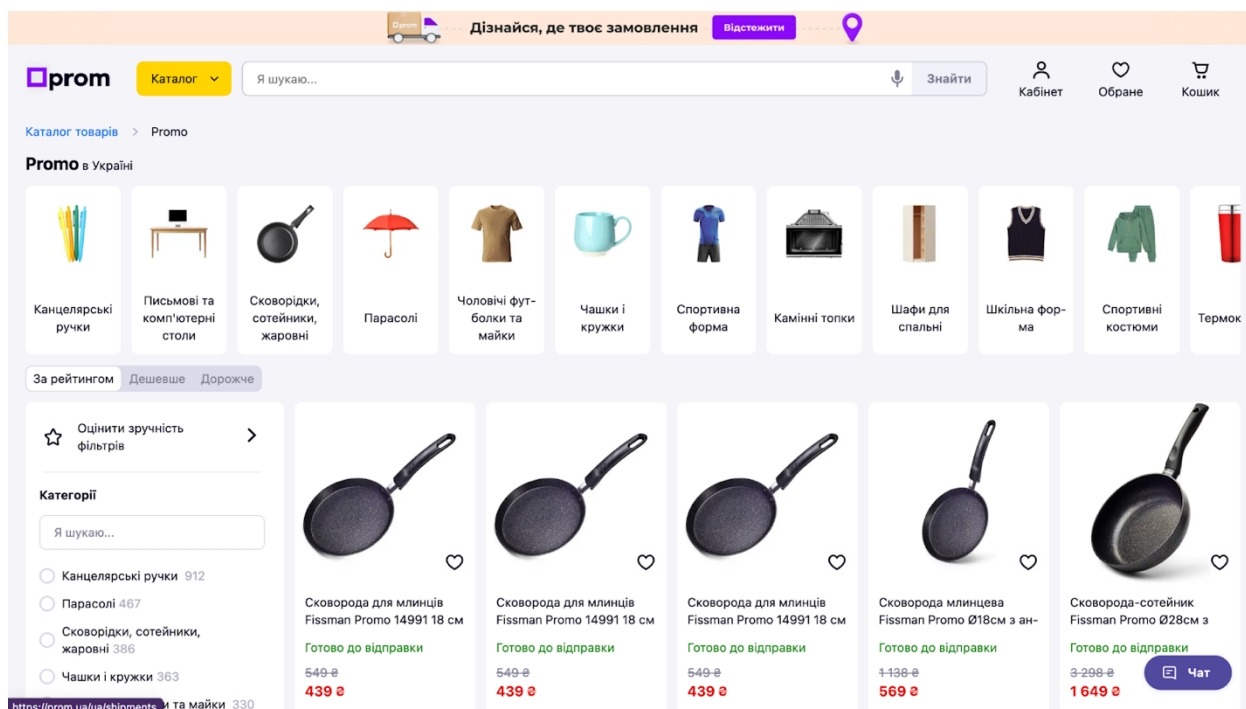


Рис. 1.4. Маркетплейс Prom

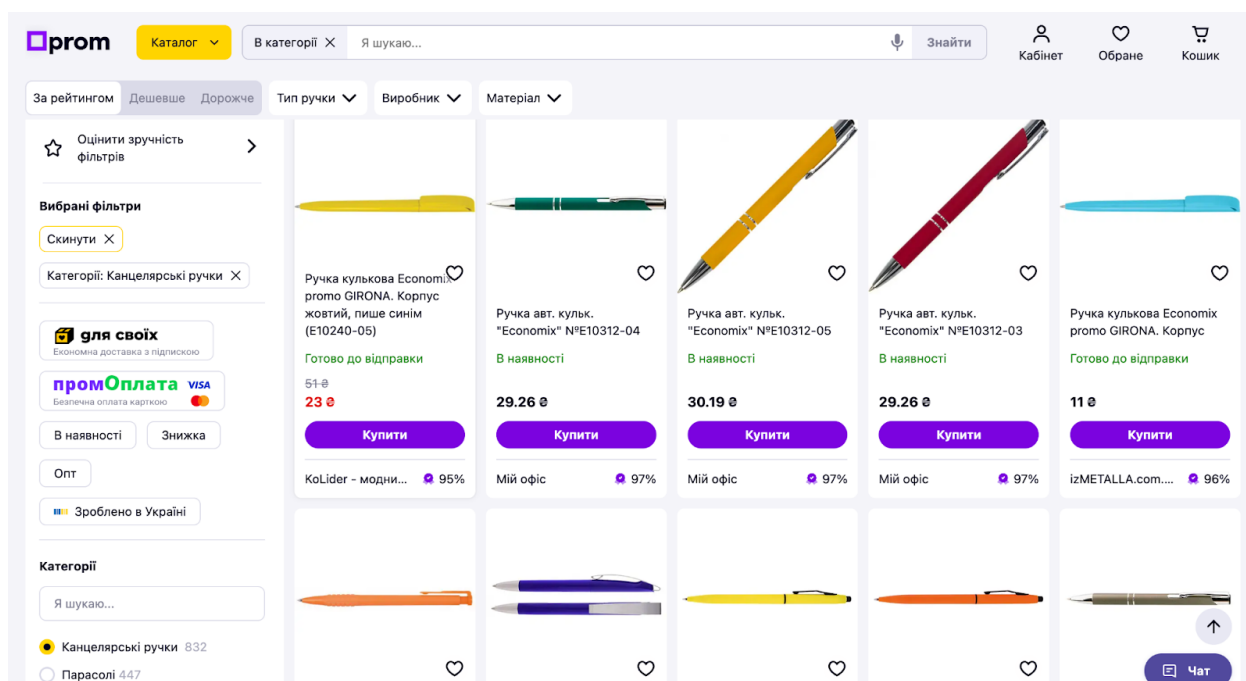


Рис. 1.5. Маркетплейс Prom

Сайт має мінімалістичний дизайн з добре продуманою структурою, завдяки чому користувачеві зручно орієнтуватися між розділами та швидко знаходити потрібну інформацію. Водночас є й суттєві недоліки: занадто великі картки товарів створюють відчуття перевантаженості, відсутність затишної атмосфери знижує привабливість для покупця, а спрощена процедура зміни ролей

призвела до появи великої кількості неякісних товарів, через що ресурс поступово втратив популярність.

amazon.com – це інтернет-магазин, який пропонує великий вибір якісних товарів для дому з акцентом на безпеку та взаємодію з користувачем[3](рис. 1.6).

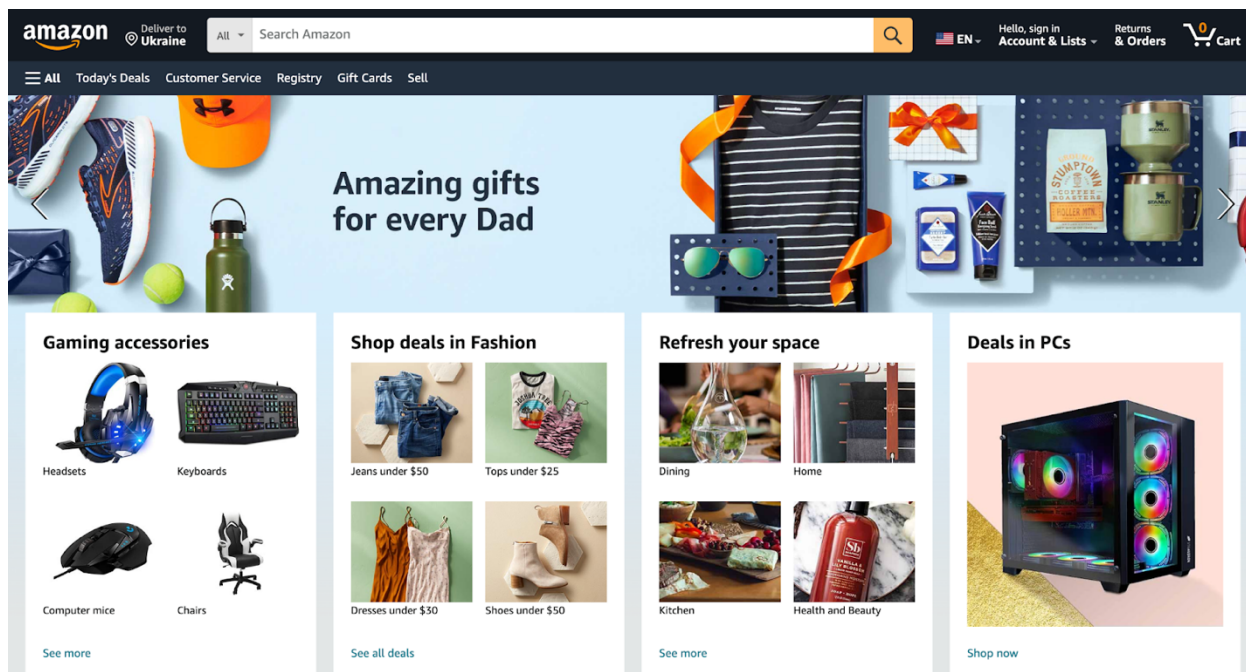


Рис. 1.6. Маркетплейс Amazon

Переваги сайту:

Сайт має мінімалістичний та водночас приємний вигляд, що створює відчуття простоти та легкості під час користування. Додатковою перевагою є можливість зберігати товари в обране, що дозволяє швидко повертатись до вибраних позицій.

Водночас відчутним недоліком є незручне орієнтування, яке ускладнює пошук потрібних розділів чи функцій.

Зважаючи на переваги й недоліки вже існуючих рішень, маркетплейс, що розробляється в межах цієї курсової, має поєднувати найкращі напрацювання. Його ключовими особливостями повинні стати зручний та зрозумілий інтерфейс, привабливий дизайн, проста навігація і можливість швидко масштабувати проєкт. Це створить хороше підґрунтя для конкуренції на сучасному ринку.

Тому логічним рішенням є створення власної платформи – такої, що зможе поєднати переваги аналогів із гнучким керуванням ролями користувачів, який легко адаптується під потреби бізнесу.

1.3. Визначення архітектури програмного забезпечення

Архітектура програмного забезпечення для маркетплейс-платформи, яка використовує стек Next.js та PostgreSQL[4], базується на клієнт-серверному підході. Такий підхід дозволяє гнучко розподіляти функціональність між різними частинами системи, що важливо для подальшого розвитку та масштабування. Основу платформи складають кілька ключових компонентів – клієнтський інтерфейс, серверна логіка, база даних, аналітичний модуль і система сповіщень.

Фронтенд, тобто клієнтська частина, відповідає за зручну взаємодію користувача з платформою. Через веб-інтерфейс можна переглядати товари, оформлювати замовленн. Завдяки використанню Next.js[6] реалізована підтримка серверного рендерингу (SSR) та попереднього рендерингу (SSG) – це дозволяє прискорити завантаження сторінок і покращити видимість у пошукових системах. Зручний інтерфейс у поєднанні з візуалізацією даних допомагає користувачам швидко орієнтуватися у статистиці продажів, зміні попиту й інших важливих тенденціях.

Серверна частина виступає посередником між усіма модулями. Вона отримує та обробляє запити, передає їх до бази даних чи аналітичної системи, і повертає відповідь на клієнтську сторону. Next.js тут також відіграє важливу роль – він не тільки забезпечує швидкий рендеринг, а й має вбудовану маршрутизацію, яка значно спрощує роботу з різними сторінками: товарами, категоріями, профілями тощо. Це дозволяє створити логічну структуру сайту й ефективно організувати всі процеси.

База даних побудована на PostgreSQL – надійній системі, яка добре підходить для зберігання великої кількості даних про товари, користувачів, замовлення й транзакції. Вона підтримує складні запити, забезпечує стабільну продуктивність і дозволяє швидко працювати з великими обсягами інформації. У зв'язці з Next.js[11]

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

ця база даних дає можливість ефективно обробляти запити, зберігаючи стабільність навіть під високим навантаженням. Це критично важливо для надійної роботи маркетплейсу.

Окремий аналітичний модуль у системі відповідає за збір і обробку даних, пов'язаних із діяльністю конкретного продавця. Він дозволяє відстежувати ключові показники – такі як кількість замовлень. Це дає змогу продавцям краще розуміти потреби своєї аудиторії та приймати обґрунтовані рішення.

На прикладі зображено діаграму компонентів системи, яка наочно демонструє, як саме взаємодіють між собою основні модулі платформи (рис. 1.7).

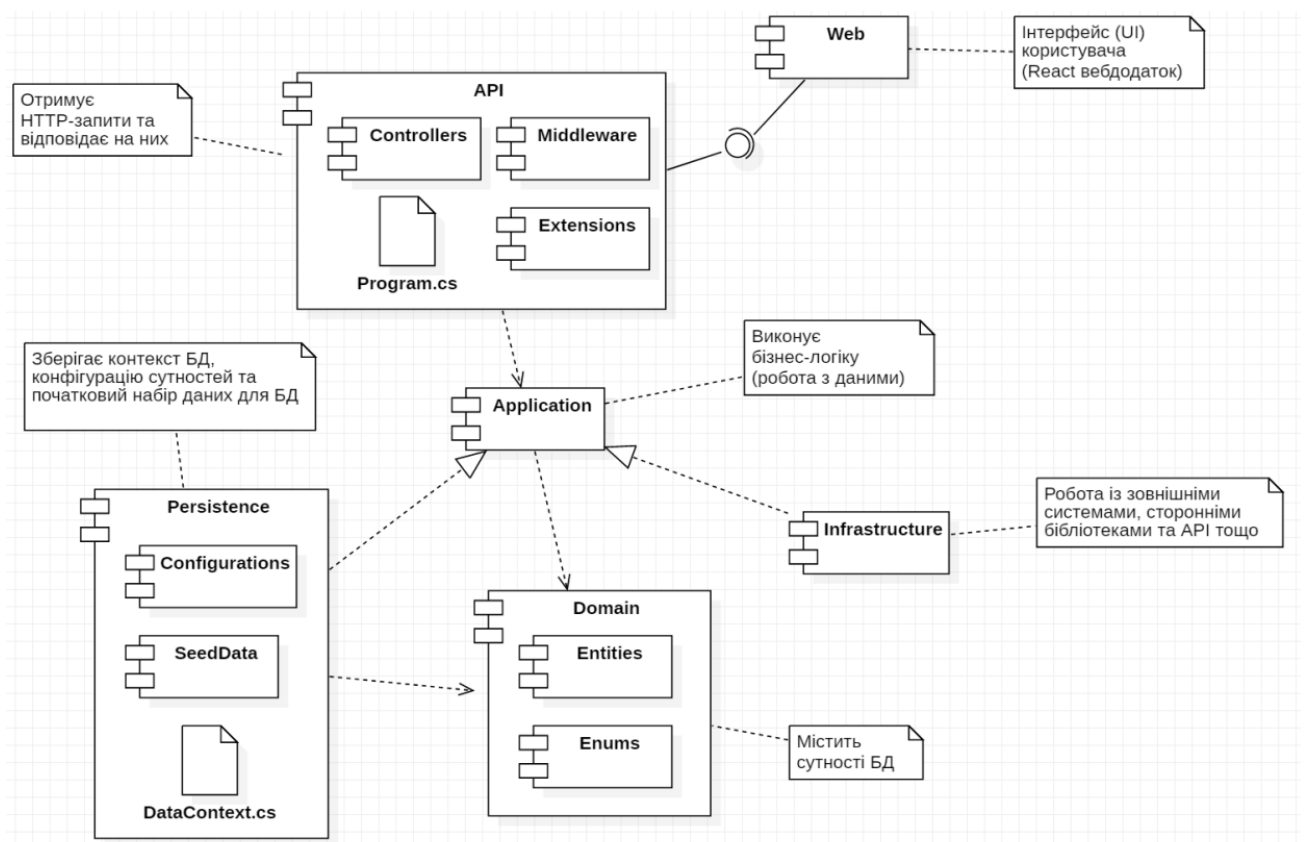


Рис. 1.7. Діаграма компонентів системи.

Такий підхід до архітектури дозволяє досягти високої продуктивності, надійності та гнучкості у розвитку платформи. Завдяки використанню Next.js система швидко обробляє запити й забезпечує стабільний доступ до даних, що позитивно впливає на швидкість завантаження сторінок та загальний досвід користувача.

Добре продумана взаємодія між модулями дає змогу ефективно керувати процесами продажу, відстежувати зміни попиту та вчасно адаптуватися до нових умов ринку. Це особливо важливо для прогнозування тенденцій, зниження ризиків і підтримки стабільної роботи навіть під час пікових навантажень.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до програмного забезпечення

Система інтернет-платформи маркетплейсу була створена з урахуванням конкретних функціональних потреб, які впливали на вибір технологій. У центрі уваги були продуктивність, можливість масштабування, зручність у користуванні та здатність ефективно працювати з великими обсягами даних – як про користувачів, так і про самі товари.

Оскільки платформа передбачає роботу з різними типами користувачів – продавцями, покупцями та адміністраторами – важливо було подбати про безпечну автентифікацію та контроль доступу. Саме тому було обрано Next.js, який має вбудовані засоби для реалізації авторизації та дозволяє гнучко управляти правами на різних рівнях.

Для зберігання інформації про продукти, замовлення, транзакції та профілі користувачів знадобилась потужна й надійна база даних. Після аналізу кількох популярних варіантів – MySQL, MongoDB та PostgreSQL – було обрано останню. PostgreSQL[5] не лише підтримує складні реляційні запити, а й дає можливість працювати з неструктурованими даними завдяки типу JSONB. Це, зокрема, корисно для збереження додаткових характеристик товарів або відгуків.

Щоб зробити інтерфейс зручним і сучасним, було використано Tailwind CSS[7]. Завдяки йому вдалося створити адаптивний, легкий у сприйнятті дизайн з інтерактивними графіками та аналітикою – усе це допомагає користувачам краще орієнтуватися в динаміці продажів і загальних тенденціях на платформі.

Проект розроблявся у середовищі WebStorm – зручно, особливо враховуючи підтримку TypeScript[8] та інтеграцію з основними інструментами. Для перевірки API-запитів застосовувався Postman[16] – він дозволив тестувати взаємодію між

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

клієнтом і сервером ще на етапі розробки, виявляючи потенційні помилки заздалегідь.

Під час вибору бази даних було проведено порівняння можливостей MySQL, PostgreSQL та MongoDB. У результаті PostgreSQL виявилась найбільш збалансованим рішенням – вона поєднує надійність реляційної структури з гнучкістю роботи з неструктурованими даними, що особливо важливо для платформи, яка має рости й розвиватися(табл. 1.1).

Таким чином, вибір кожного інструмента обґрунтований його функціональністю, відповідністю вимогам та широкою підтримкою у розробницькому середовищі. Використання сучасних технологій дозволяє створити продуктивну, безпечну та масштабовану систему.

Таблиця 1.1

Порівняння баз даних

Особливості	MS SQL Server	MongoDB	PostgreSQL
Тип даних	Реляційна база даних	Документ-орієнтована база даних	Реляційна база даних
Мова запитів	Transact-SQL (T-SQL)	MongoDB Query Language (MQL)	Transact-SQL (T-SQL), підтримка JSONB
Схема	Суворі схема даних	Гнучка схема даних	Суворі схема даних
Масштабованість	Вертикальна	Горизонтальна масштабованість	Горизонтальна і вертикальна
Підтримка ACID	Повна підтримка	Зазвичай підтримується в обмеженій формі	Повна підтримка
Реплікація	Підтримується	Підтримується	Підтримується

Продовження таблиці 1.1

Індексування	Широкий набір індексів	Індексування за ключами та текстове індексування	Широкий набір індексів
Масштабованість читань	Висока масштабованість читань	Висока масштабованість читань	Висока масштабованість читань
Масштабованість записів	Обмежена масштабованість записів	Висока масштабованість записів	Обмежена масштабованість записів
Підтримка JSON	Підтримується з MS SQL Server 2016	Вбудована підтримка JSON	Так (JSONB – ефективний тип даних)
Геопросторові дані	Підтримується з допомогою розширення	Вбудована підтримка геопросторових запитів	Підтримується з допомогою розширення
Запити JOIN	Підтримується	Підтримується в обмеженій формі	Підтримується
Застосунки	Підходить для традиційних веб- застосунків	Підходить для складних, масштабованих застосунків	Підходить для традиційних веб- застосунків

Отже, вибір кожної технології у цьому проєкті не був випадковим – усе базувалося на їхніх реальних можливостях, відповідності до поставлених цілей і підтримці в середовищі розробників. Завдяки поєднанню сучасних інструментів вдалося створити платформу, яка не тільки швидка й безпечна, а й добре підготовлена до зростання. Вона дозволяє ефективно працювати з товарами, замовленнями та взаємодією між користувачами – усе для того, щоб платформа залишалась зручною і надійною як для покупців, так і для продавців.

Висновки до першого розділу

Перший розділ роботи присвячений формуванню основної ідеї проєкту та його ролі у світі електронних маркетплейсів. Було детально проаналізовано вже існуючі рішення – різні платформи для онлайн-торгівлі, системи управління товарами та засоби комунікації між продавцями й покупцями. Цей аналіз допоміг виявити сильні й слабкі сторони кожного підходу з огляду на їхню ефективність у створенні маркетплейсів для різних категорій товарів.

На основі цих висновків були сформульовані ключові вимоги до власної платформи. Вона має підтримувати автоматичне додавання товарів, вміти прогнозувати попит, інтегруватися з аналітичними модулями і гарантувати високий рівень безпеки під час транзакцій. Обрано клієнт-серверну архітектуру, яка забезпечує необхідну продуктивність, гнучкість і масштабованість – це дозволить ефективно обробляти велику кількість користувачів і їхні запити.

Окремо приділено увагу вибору програмних засобів і технологій, які допоможуть реалізувати маркетплейс. Використання сучасних інструментів сприятиме створенню зручної системи управління товарами, полегшить взаємодію між користувачами та забезпечить стабільну роботу всієї платформи.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ МАРКЕТПЛЕЙСУ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

У цьому розділі наведено детальний огляд можливих сценаріїв використання інтернет-маркетплейсу, визначено основні ролі користувачів і їхні функціональні потреби. Також описано архітектуру системи, яка має відповідати не лише базовим функціональним вимогам, а й враховувати нефункціональні аспекти — щоб забезпечити надійну і плавну взаємодію між продавцями, покупцями та адміністраторами у цифровому середовищі.

Архітектура системи - система складається з трьох основних логічних модулів. Першим є клієнтський модуль, який забезпечує інтерфейс для покупців і продавців та реалізований на основі технології Next.js. Другим виступає адміністративний модуль, що надає адміністраторам платформи засоби керування та налаштування роботи системи. Третім є аналітичний модуль, призначений для відображення ключових показників, зокрема статистики продажів і активності користувачів.

Основні ролі користувачів - у системі передбачено три основні ролі користувачів. Перша роль — це покупець, який має можливість переглядати товари, додавати їх до кошика, здійснювати замовлення та залишати відгуки. Другою роллю є продавець, що відповідає за додавання й редагування товарів, отримання та обробку замовлень, а також за аналіз власних продажів. Третьою виступає адміністратор, котрий здійснює управління користувачами, перевіряє контент, модерує товари й замовлення та відповідає за налаштування системи загалом.

Мета цього проєкту – створити масштабовану інтернет-платформу для продажу товарів, яка забезпечить комфортну і зрозумілу взаємодію між продавцями та покупцями.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

Щодо функціональності, платформа повинна підтримувати всі основні процеси купівлі-продажу, мати зручний інтерфейс, надсилати сповіщення, дозволяти залишати відгуки, пропонувати ефективну фільтрацію товарів та вести історію замовлень.

Функціональні вимоги. Функціональні можливості системи поділяються залежно від ролей користувачів. Для покупця передбачено можливість переглядати товари, здійснювати пошук, додавати обрані позиції до кошика, оформлювати й переглядати замовлення, а також залишати відгуки щодо придбаних товарів.

Продавець, у свою чергу, має змогу додавати та редагувати товари, зокрема змінювати їх назву, опис, фотографії, ціну та категорію. Крім того, йому надається можливість створювати та редагувати сторінку бренду, переглядати й обробляти замовлення, а також відповідати на відгуки покупців.

Для адміністратора реалізовано функції модерації товарів і контенту, управління користувачами з можливістю призначення ролей, а також налаштування категорій товарів та загальної структури сайту.

Нефункціональні вимоги. Щоб забезпечити зручність використання, інтерфейс системи має бути максимально інтуїтивним і зрозумілим для користувачів із різним досвідом. Йдеться про те, щоб покупці могли розібратися з основними функціями не довше ніж за 10 хвилин, а продавці — за 20. Це дозволить зменшити час на навчання і зробить систему доступною навіть для новачків.

У питаннях надійності передбачається, що система повинна стабільно працювати щонайменше 30 днів поспіль без серйозних збоїв або зупинок. При цьому допускається не більше однієї помилки на кожні 1000 запитів. Такий рівень надійності гарантує впевненість користувачів у працездатності платформи.

Продуктивність системи має дозволяти їй одночасно обслуговувати щонайменше 200 активних користувачів. Це важливо для підтримання швидкої реакції на дії користувачів і уникнення затримок, навіть у години найбільшого навантаження.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Ще однією важливою вимогою є масштабованість. Система має бути здатною до зростання — тобто легко справлятися зі збільшенням кількості товарів, продавців і покупців. При цьому її продуктивність не повинна погіршуватися, а користувацький досвід — змінюватися на гірше.

Системні вимоги

- Браузери: Chrome 90+, Firefox 80+, Edge 85+;
- ОС: Windows 10+, macOS 10.14+, Linux;
- Клієнтська частина: Next.js;
- Серверна частина: NestJS (REST API);
- База даних: PostgreSQL 14+ з підтримкою JSONB;
- Інші технології: TailwindCSS, TypeScript, Prisma ORM;
- Push/email-сповіщення: через API інтеграцію (наприклад, SendGrid або Firebase).

Характеристика об'єкта комп'ютеризації можна описати, як платформу створюється для підтримки онлайн-торгівлі, де одночасно працює багато незалежних продавців, різноманітні категорії товарів і численні покупці(рис. 2.1).

Варіанти використання на показано діаграму варіантів використання, яка відображає основні сценарії роботи: реєстрація, авторизація, додавання товарів,

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

оформлення замовлень та модерація контенту.

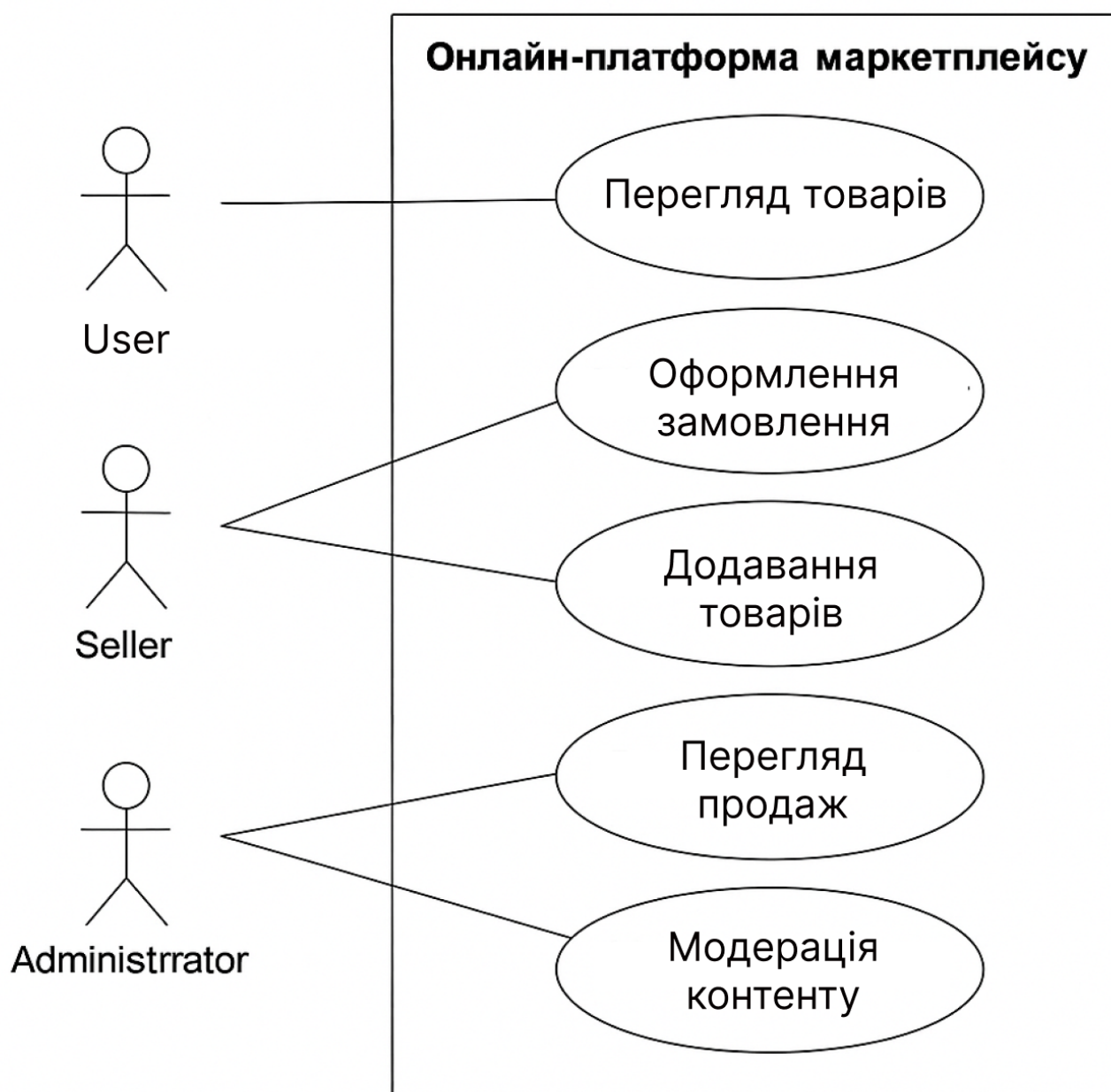


Рис. 2.1. Діаграма варіантів використання

Інтернет-маркетплейс – це складна багаторівнева система, яка поєднує в собі функції класичного онлайн-магазину та можливість залучати сторонніх продавців. Завдяки такій гнучкості платформа має підтримувати різні сценарії взаємодії між трьома основними ролями – покупцем, продавцем і адміністратором.

Кожен користувач працює з платформою по-своєму, а набір функцій залежить від його ролі(табл. 2.1), (табл. 2.2). Щоб наочно показати основні

сценарії, була створена діаграма варіантів використання(табл. 2.3). Нижче наведено їхній опис у вигляді таблиць.

Таблиця 2.1

Опис варіанту використання «Перегляд особистого кабінету продавця»

Назва варіанту використання	Перегляд особистих медичних даних
Основні актори	Продавець
Короткий опис	Особистий кабінет – це основне місце, де продавець взаємодіє з платформою. Тут користувач може стежити за статусом своїх товарів, переглядати історію спілкування з покупцями та відслідковувати ключові бізнес-показники. Такий підхід допомагає приймати обґрунтовані рішення щодо цін, асортименту і маркетингових кампаній..

Таблиця 2.2

Опис варіанту використання «Додавання та редагування товарів»

Назва варіанту використання	Додавання/редагування товарів
Основні актори	Продавець, Адміністратор
Короткий опис	Операція редагування товару — одна з найчастіше використовуваних функцій. Вона дає змогу зручно керувати онлайн-вітриною: змінювати описи, оновлювати фото, актуалізувати ціни, вказувати акційні пропозиції та відображати доступність на складі. У разі виявлення порушень політики платформи адміністратор може внести зміни або приховати картку товару.

Таблиця 2.3

Опис варіанту використання «Додавання та редагування “улюблених товарів”»

Назва варіанту використання	Перегляд аналітики “улюблених товарів”
Основні актори	Користувач
Короткий опис	Цей сценарій надає користувачу інструменти відстеження та маркування окремих продуктів.

Опрацьовані варіанти використання демонструють широкий спектр функцій, які забезпечують стабільну роботу маркетплейсу як для користувачів, так і для управління бізнес-процесами. Взаємодія між акторами і системою ґрунтується на чітко розмежованих повноваженнях – це гарантує контроль, безпеку та можливість масштабування рішення.

Ці сценарії стали основою для подальшого об’єктно-орієнтованого моделювання. Було виділено логічні класи, що відповідають за управління каталогом товарів, збереження історії покупок, адміністрування та обробку запитів.

Діаграма класів (рис. 2.2) привертає до уваги основі сутності системи, їхні властивості та взаємозв’язки. Основним класом є користувач (User), який об’єднує в собі інформацію про користувачів, їх персональні дані, ролі та статус. Клас User також пов’язаний з обліками (Account), сесіями (Session), замовленнями (Order), відгуками (Feedback), кошиком (Cart), списком улюблених (Favorite) та повідомленням адміністратора (AdminMessage). Крім того, користувач може бути власником бренду (Brand), що надає йому можливість створювати та управляти товарами.

Клас Order зберігає дані про замовлення, включно з номером, ціною та датою створення. Він пов’язаний із класом OrderProducts, що описує конкретні товари в замовленні. Самі товари описані класом Product, який має властивості назви, опису, ціни та фото, а також належить до певної категорії (Category) та бренду (Brand).

Клас Feedback дозволяє користувачам залишати відгуки з текстом і рейтингом для конкретних товарів. Класи Cart і Favorite забезпечують збереження товарів у кошику чи списку обраного відповідно.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

Окремо визначені допоміжні класи: VerificationToken для підтвердження пошти, Account для збереження даних сторонніх авторизаційних сервісів, а також Session для керування активними сесіями. Для користувачів передбачені ролі (UserRole) та статуси ролей (RoleStatus), що дозволяє гнучко керувати правами доступу.

Таким чином, діаграма класів описує повну модель предметної області системи, включаючи користувачів, їхню активність, товари та бізнес-логіку взаємодії з платформою.

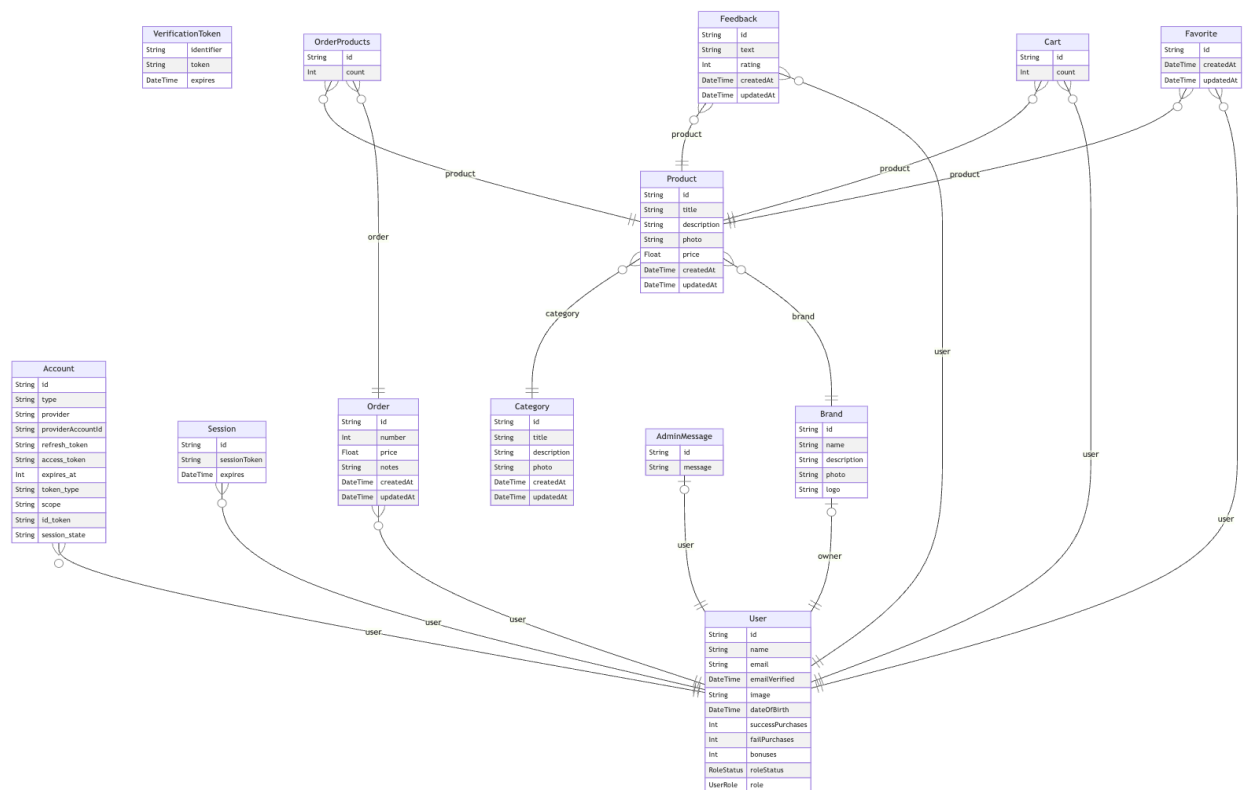


Рис. 2.2. Діаграма класів

2.2. Розробка бази даних системи

У системі маркетплейсу для зберігання даних використовується реляційна СУБД PostgreSQL. Вона відзначається високою надійністю, підтримкою масштабування і здатністю виконувати складні запити, що особливо важливо для реалізації аналітики.

Структура бази даних спроектована з урахуванням принципів нормалізації – чітко розподілені обов’язки між сутностями та передбачені можливості для подальшого розширення й масштабування. Схема включає таблиці для зберігання інформації про користувачів і їх ролі, товари та замовлення.

Нижче наведено опис ключових моделей бази даних, адаптованих до потреб маркетплейсу.

Базовими таблицями бази даних є users, brand, products, orders та roles. В таблиці users зберігається інформація про всіх користувачів платформи, включно з продавцями, покупцями та адміністраторами (табл. 2.4). Таблиця brand містить основні відомості про бренд і виконує роль зв'язку між продавцем та його товарами (табл. 2.9). У таблиці products розміщуються дані про товари, що пропонуються на маркетплейсі, зокрема їхні характеристики та належність до певної категорії (табл. 2.5). Таблиця orders використовується для збереження інформації про замовлення, здійснені користувачами, а також їхній склад і статус (табл. 2.6). Таблиця roles визначає ролі користувачів у системі та регламентує їхні права доступу, що забезпечує розмежування функціональності між адміністраторами, продавцями та покупцями.

Кожна з цих таблиць має чітко визначені поля та зв'язки з іншими таблицями, що дозволяє ефективно управляти даними та забезпечувати цілісність інформації.

Таблиця 2.4

Опис таблиці User

Поле	Тип	Призначення
id	String	Унікальний ідентифікатор користувача
name	String	Ім'я користувача
email	String?	Електронна пошта (унікальна, може бути відсутня)
emailVerified	DateTime?	Дата підтвердження електронної пошти
image	String?	Посилання на зображення профілю
dateOfBirth	DateTime?	Дата народження користувача

Продовження таблиці 2.4

successPurchases	Int?	Кількість успішних покупок
failPurchases	Int?	Кількість невдалих покупок
bonuses	Int?	Накопичені бонуси
roleStatus	RoleStatus	Статус ролі користувача (за замовчуванням APPROVED)
role	UserRole	Роль користувача (за замовчуванням USER)
AdminMessage	AdminMessage?	Пов'язане адміністративне повідомлення (опційно)
brand	Brand?	Пов'язаний бренд (опційно)

Таблиця 2.5

Опис таблиці Product

Поле	Тип	Призначення
id	String	Унікальний ідентифікатор товару
title	String	Назва товару
description	String	Опис товару
price	Decimal	Ціна товару
photo	String?	Посилання на фото товару (опційно)
createdAt	DateTime	Дата створення запису
updatedAt	DateTime	Дата останнього оновлення
categoryId	String	Ідентифікатор категорії товару
brandId	String	Ідентифікатор бренду, якому належить товар

Таблиця 2.6

Опис таблиці Order

Поле	Тип	Призначення
id	String	Унікальний ідентифікатор замовлення
number	Int	Унікальний номер замовлення (автоматично інкрементується)
price	Float	Загальна сума замовлення
notes	String?	Додаткові нотатки (опційно)
userId	String	Ідентифікатор користувача, який зробив замовлення
updatedAt	DateTime	Дата останнього оновлення замовлення
createdAt	DateTime	Дата додавання замовлення

Таблиця 2.7

Опис таблиці OrderProducts

Поле	Тип	Призначення
id	String	Унікальний ідентифікатор позиції
productId	String	Ідентифікатор товару
orderId	String	Ідентифікатор замовлення
count	Int	Кількість одиниць товару у замовленні

Таблиця 2.8

Опис таблиці Category

Поле	Тип	Призначення
id	String	Унікальний ідентифікатор категорії
title	String	Назва категорії (унікальна)
description	String	Опис категорії

Продовження таблиці 2.8

photo	String?	Посилання на зображення категорії (опційно)
createdAt	DateTime	Дата створення запису
updatedAt	DateTime	Дата останнього оновлення

Таблиця 2.9

Опис таблиці Brand

Поле	Тип	Призначення
id	String	Унікальний ідентифікатор
userId	String	Ідентифікатор користувача — власника бренду (унікальний)
name	String	Назва бренду (унікальна)
description	String?	Опис бренду (опційно)
photo	String?	Зображення бренду (опційно)
logo	String?	Логотип бренду (опційно)

Ця структура бази даних забезпечує потрібну гнучкість і масштабованість для ефективної роботи інтернет-платформи маркетплейсу. Вона дає змогу зберігати й обробляти інформацію про користувачів, товари, замовлення, бренди та відгуки.

В розробці маркетплейсу всі дані зберігаються у централізованій реляційній базі PostgreSQL. Вона виступає ядром для обробки, синхронізації інформації про товари, замовлення і користувачів.

Кожен запис у системі - наприклад, публікація нового товару, оплата або створення замовлення - зберігається у відповідних таблицях. Такі записи містять часові позначки (createdAt, updatedAt), зв'язки з іншими сутностями (через зовнішні ключі, як userId, orderId, productId) і статуси, які допомагають контролювати бізнес-логіку.

Для прикладу:

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

Дані про користувачів зберігаються в таблиці User, де фіксуються особисті дані, роль (продавець чи покупець) і час реєстрації.

Товари зберігаються в таблиці Product, прив'язані до продавця (sellerId), із назвою, описом, ціною та залишками на складі.

Деталі замовлення зберігаються в таблиці OrderItem — назва товару, кількість і актуальна ціна.

Відгуки покупців зберігаються в таблиці Feedback, з рейтингом, коментарем і датою публікації.

2.3. Проектування та реалізація алгоритмів роботи системи

Цей проєкт дає змогу користувачам створювати та переглядати оголошення про товари. Він працює як простий маркетплейс, де кожен може додати свій товар і подивитись, що публікують інші.

Платформа побудована на Next.js[9], використовує Prisma[18] для роботи з базою даних і TailwindCSS[29] — для створення зручного та приємного інтерфейсу.

Коли користувач заходить на сайт, його автоматично перекидає на сторінку входу. Авторизація — максимально проста: тільки електронна пошта. Після входу створюється сесія, і користувач переходить у внутрішню частину платформи.

За безпеку відповідає NextAuth.js — він надійно зберігає дані про сесію та стежить за авторизацією.

У верхній частині сторінки розташувався компактний та зручний панель навігації, у якій залишено лише найважливіші елементи. Перехід на головну сторінку дозволяє переглядати всі оголошення, розділ "Адмін панель" відкриває особистий кабінет користувача, у вкладці "Категорії" доступна форма для додавання нового товару, а сторінка "Бренди" містить список користувачів, що відображається лише для певних ролей.

Меню адаптивне — однаково добре працює як на комп'ютері, так і на смартфоні.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дата		

Зареєстровані користувачі можуть додавати товари, заповнивши невелику форму: назва, опис, ціна, категорія тощо. Усе зберігається в базі даних, і нові товари одразу з'являються на головній сторінці.

Окрім основних можливостей, система передбачає низку додаткових функцій. Кожен товар має унікальний ідентифікатор, що дозволяє однозначно визначати його в базі даних. Форма додавання оголошень перевіряє правильність введених даних, запобігаючи наявності порожніх полів чи помилок. Також користувачі мають змогу редагувати або видаляти власні оголошення, що забезпечує зручність у керуванні контентом.

На сторінці “Users” показано список усіх зареєстрованих користувачів. Натискаючи на будь-яке ім'я, можна побачити публічний профіль із товарами, які ця людина додала. Це простий, але зручний спосіб подивитися, хто що продає.

У системі реалізовано два основні типи користувачів: звичайний користувач та адміністратор. Перший має можливість переглядати товари і створювати власні оголошення. Адміністратор, окрім цього, отримує розширені можливості, наприклад, доступ до списку всіх користувачів або керування товарною базою. Хоча наразі адмін-панель реалізована на базовому рівні, у коді вже закладено потенціал для її подальшого розвитку та розширення функціоналу.

З технічного боку проєкт побудовано на Next.js, що використовується одночасно і для фронтенду, і для бекенду, забезпечуючи зручну архітектуру. Для роботи з базою даних використовується Prisma ORM, яка забезпечує доступ до таблиць, зокрема таких як users і products. TailwindCSS застосовується для створення швидкого, адаптивного та візуально впорядкованого інтерфейсу. За реалізацію авторизації та управління сесіями відповідає бібліотека NextAuth.js, яка значно спрощує ці процеси.

Після авторизації користувач потрапляє до особистого кабінету. Тут він бачить основну інформацію про себе – ім'я, електронну пошту, групу (якщо потрібно) та роль у системі.

У цьому розділі доступні дві основні дії:

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

- «Перейти до записів» – відкриває історію дій користувача;
- «Вийти» – дозволяє безпечно завершити сесію.

Такий підхід забезпечує прозорий контроль над персональними даними й робить взаємодію з платформою комфортною.

Щоб кожен мав доступ лише до тих функцій, які йому потрібні, реалізовано чітку матрицю прав(табл. 2.10). У таблиці показано, що можуть робити різні типи користувачів:

- 0 - відсутність доступу
- 1 - перегляд власних даних
- 2 - доступ до даних усіх користувачів

Таблиця 2.10

Таблиця. Матриця доступу до функцій платформи

Сторінка / Функція	Користувач	Продавець	Адміністратор
Перегляд особистого профілю	1	1	0
Перегляд усіх користувачів	0	0	2

Продовження таблиці 2.10

Перегляд історії замовлень	1	1	2
Додавання товарів	0	2	2
Перегляд продажів	0	2	2
Перегляд статистики	1	2	2
Керування об'єктом бренду	0	2	1

Платформа гнучко реагує на дії користувачів, адаптуючи функціональність залежно від їхньої ролі. Такий підхід дозволяє створити чітку й зрозумілу модель взаємодії:

- Покупці бачать лише свої особисті дані та історію замовлень.
- Продавці можуть керувати власними товарами, переглядати статистику та аналітику свого магазину.

– Адміністратори мають повний контроль над платформою – від управління користувачами до роботи з модулями прогнозування й аналітики.

Кожен сценарій – від входу в систему до запуску складних інструментів – реалізовано з урахуванням рівня доступу. Це не лише підвищує безпеку, а й спрощує масштабування платформи: за потреби легко додати нові ролі чи функції(рис. 2.3).

На діаграмі нижче (діаграма активностей) показано загальну логіку взаємодії користувачів із системою: переходи між станами, розгалуження та залежності від ролей.

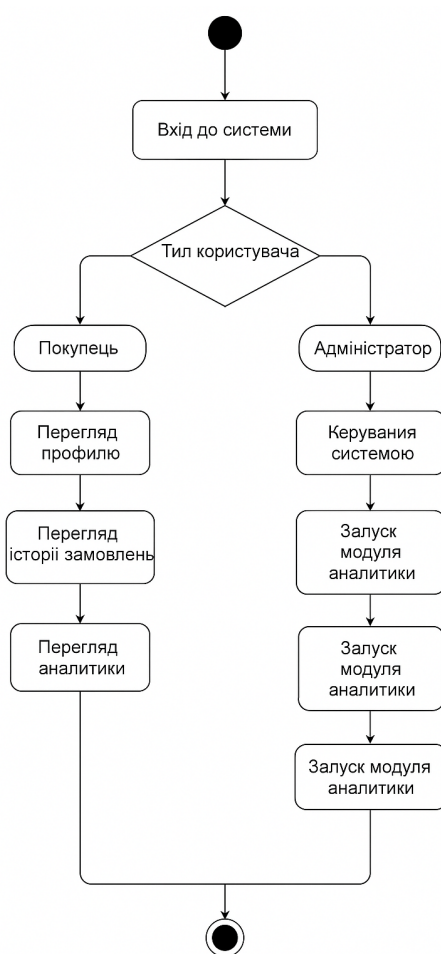


Рис. 2.3. Діаграма активності

2.4. Реалізація функціоналу платформи

Серверна логіка авторизації, валідації та оптимістичного UI. Під час розробки маркетплейсу ми особливо потурбувалися про безпечний і зручний вхід користувачів, якісну перевірку форм і те, щоб інтерфейс реагував миттєво - без

зайвих затримок. Для цього використали сучасні бібліотеки, які добре дружать з Next.js[10] і Prisma.

Авторизація користувачів. За логіку входу відповідає NextAuth.js. Вона підтримує і звичайний вхід через email та пароль, і авторизацію через Google. Це зручно, бо кожен може обрати, що йому до душі - і при цьому залишатися в безпеці. Головний файл, де налаштовуються провайдери і сесії, знаходиться тут: src/pages/api/auth/[...nextauth].ts. Ось як виглядає важливий кусок коду, що відповідає за це:

```
export const authOptions: NextAuthOptions = { Show usages  👤 "Alex"
  adapter: PrismaAdapter(db) as Adapter,
  callbacks: {
    session: ({ session, user } : { session: Session; token: JWT; user: Ada... } ) => {
      session.user.id = user.id;
      // @ts-ignore
      session.user.role = user.role;
      return session;
    },
  },
  providers: [
    GoogleProvider({
      clientId: env.GOOGLE_CLIENT_ID,
      clientSecret: env.GOOGLE_CLIENT_SECRET,
    }),
    GitHubProvider({
      clientId: env.GITHUB_CLIENT_ID,
      clientSecret: env.GITHUB_CLIENT_SECRET,
    }),
  ],
};
```

Така архітектура гарантує, що кожен користувач, який пройшов аутентифікацію, отримує унікальний токен — він потрібен для безпечної взаємодії з сервером. Вхід через Google OAuth працює так: якщо користувача з таким email ще немає в базі, система автоматично його створює з мінімальним набором даних. Це значно спрощує реєстрацію і знімає зайві бар'єри для користувачів.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						37
Змн.	Арк.	№ докум.	Підпис	Дата		

Щоб форми працювали надійно і без помилок, ми взяли класне поєднання - react-hook-form[26] разом із Zod[28] встановлені через prn[24]. Перша відповідає за легку інтеграцію форм із мінімальним навантаженням на продуктивність, а друга - дозволяє описати чіткі правила валідації, які суворо контролюють, що вводить користувач.

Як це виглядає на практиці - у компоненті реєстрації src/components/RegisterForm.tsx. Ось шматок коду, що показує, як ми поєднуємо обидві бібліотеки:

```
type EntityZodErrors<T> = Partial<Record<keyof T, string[] | undefined>>;

export function useValidatedForm<Entity>(insertEntityZodSchema: ZodSchema) { Show usages  "Alex"
  const [errors, setErrors] = useState<EntityZodErrors<Entity> | null>(null);
  const hasErrors : boolean = errors !== null
    && Object.values(errors).some((error : unknown) => error !== undefined);

  const handleChange : (event: FormEvent<HTMLFormElement>) => void = (event: FormEvent<HTMLFormElement>) => {
    const target = event.target as EventTarget;
    if (
      target instanceof HTMLInputElement
      || target instanceof HTMLSelectElement
      || target instanceof HTMLTextAreaElement
    ) {
      if (!(target instanceof HTMLInputElement && target.type === 'submit')) {...}
    }
  };
  return {
    errors, setErrors, handleChange, hasErrors,
  };
}
```

В інтернет-платформі маркетплейсу для підвищення продуктивності взаємодії користувачів із системою впроваджено патерн Optimistic UI, який передбачає миттєве оновлення інтерфейсу при виконанні користувацьких дій без очікування відповіді від сервера. Такий підхід значно покращує користувацький досвід за рахунок усунення затримок, що характерні для мережових запитів.

Оптимістичні оновлення використовуються для операцій, пов'язаних із модифікацією даних, таких як додавання, оновлення або видалення товарів, замовлень, відгуків та інших сутностей. Це дозволяє зробити інтерфейс більш

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						38
Змн.	Арк.	№ докум.	Підпис	Дата		

чутливим і «живим», а також знизити навантаження на сервер шляхом ефективного керування локальним станом.

```
export type TAddOptimistic = (action: OptimisticAction<Favorite>) => void;

export const useOptimisticFavorites : (favorites: ({ product: { id: string; cre... = ( Show usages  ⓘ "Alex"
  favorites: CompleteFavorite[],
  products: Product[],
) => {
  const [optimisticFavorites, addOptimisticFavorite] = useOptimistic(
    favorites,
    (
      currentState: CompleteFavorite[],
      action: OptimisticAction<Favorite>,
    ): CompleteFavorite[] => {
      const { data } = action;

      const optimisticProduct : { id: string; createdAt: Date; updatedAt:... = products.find(
        (product : { id: string; createdAt: Date; updatedAt:... ) => product.id === data.productId,
      )!;

      const optimisticFavorite = {id: 'optimistic'...};

      switch (action.action) {...}
    },
  );
};
```

Висновки до другого розділу

У процесі розробки інтернет-платформи маркетплейсу було визначено три ключові ролі користувачів – покупець, продавець і адміністратор. Вони стали основою для побудови основних сценаріїв використання та заклали логіку системної архітектури. Кожна роль має свій набір можливостей і рівень доступу до функцій платформи.

Відповідно до цих сценаріїв було спроектовано об'єктно-орієнтовану модель, яка охоплює взаємодію між користувачами, товарами, замовленнями та сервісними модулями – оплатою, доставкою, сповіщеннями тощо. Ці взаємозв'язки знаходять відображення у структурі типових сторінок платформи: каталог товарів, сторінка продавця, панель адміністратора, особистий кабінет.

Окрему увагу приділено реляційній базі даних, яка включає таблиці користувачів, товарів, замовлень, категорій, відгуків і повідомлень. Її структура підтримує цілісність даних та ефективність запитів.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дата		

У системі реалізовано низку ключових алгоритмів, серед яких:

- обробка подій покупки;
- валідація транзакцій;
- формування рейтингових списків продавців;
- рекомендаційні механізми;
- автоматичне сповіщення про зміну статусу замовлень.

Уся взаємодія між клієнтською частиною (Next.js) та серверною (NestJS) відбувається через REST API, що забезпечує стабільний обмін даними.

На загальній діаграмі активності зображено, як користувач пересувається між модулями системи – від авторизації до здійснення покупки чи керування товарами. Також на ній видно, як інтерфейс і функції змінюються залежно від ролі.

Таким чином, завдяки аналізу потреб, ролей і сценаріїв, вдалося побудувати архітектуру, яка забезпечує зручну, безпечну та ефективну взаємодію між усіма учасниками торгового процесу.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ПЛАТФОРМОЮ

3.1. Розгортання платформи

Коли розробка в локальному середовищі була завершена, потрібно було зробити платформу доступною онлайн. Для цього використано сервіс Vercel[12]. Він ідеально підходить для Next.js-проектів і дозволяє швидко та без зайвих налаштувань опублікувати сайт.

Було створено обліковий запис на Vercel і підключив свій репозиторій з GitHub[21]. Після цього система сама автоматично підтягує зміни, коли я пушу новий код. Тобто щоразу, як я щось оновлюю в проєкті, сайт збирається заново і оновлюється без моєї участі. Це зручно, бо не потрібно нічого вручну переносити або запускати.

Щоб проєкт міг працювати з базою даних та іншими сервісами, було додано у налаштування Vercel змінні середовища. Наприклад, URL бази даних, API-ключі або інші важливі дані. Це приблизно те саме, що локальний файл .env, тільки в онлайн-інтерфейсі.

Базу даних я використовую PostgreSQL. Вона розміщена окремо, на хмарному сервісі. Для з'єднання з нею я використовую Prisma, який значно полегшує роботу з базою. Після налаштування з'єднання я виконав міграції, щоб створити потрібні таблиці в базі, і на цьому все запрацювало.

Після першого деплою сайт став доступним за посиланням, яке надає Vercel. За бажанням можна підключити власний домен, але в моєму випадку достатньо було стандартного посилання, яке створюється автоматично.

Vercel виявився дуже зручним для такого типу проєкту(рис. 3.1). Він сам слідкує за оновленнями коду, швидко збирає сайт і дозволяє легко керувати середовищем. Тобто технічно все працює стабільно, а можу зосередитись на логіці платформи, а не на налаштуванні серверів.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

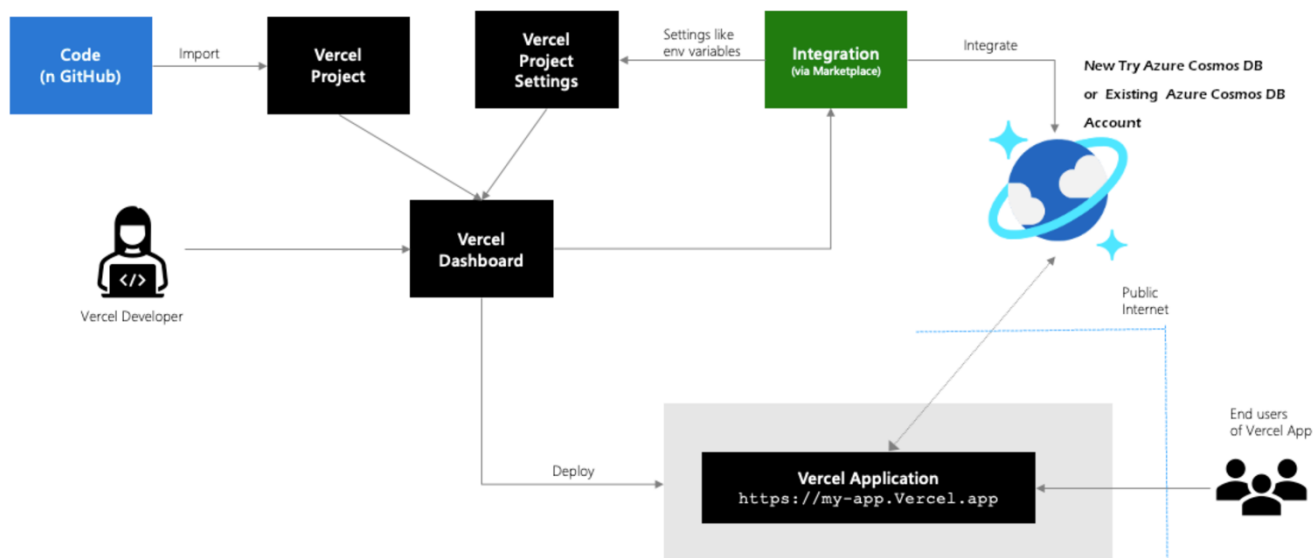


Рис. 3.1. Схема віддалено розгортання та хостингу проєкту

3.2. Структура інтерфейсу та користувацька взаємодія

Коли користувач уперше заходить на сайт і ще не авторизований, він одразу потрапляє на головну сторінку(рис. 3.2). Там показуються банер з коротким описом та продуктова секція з продуктами, які були додані до платформи.

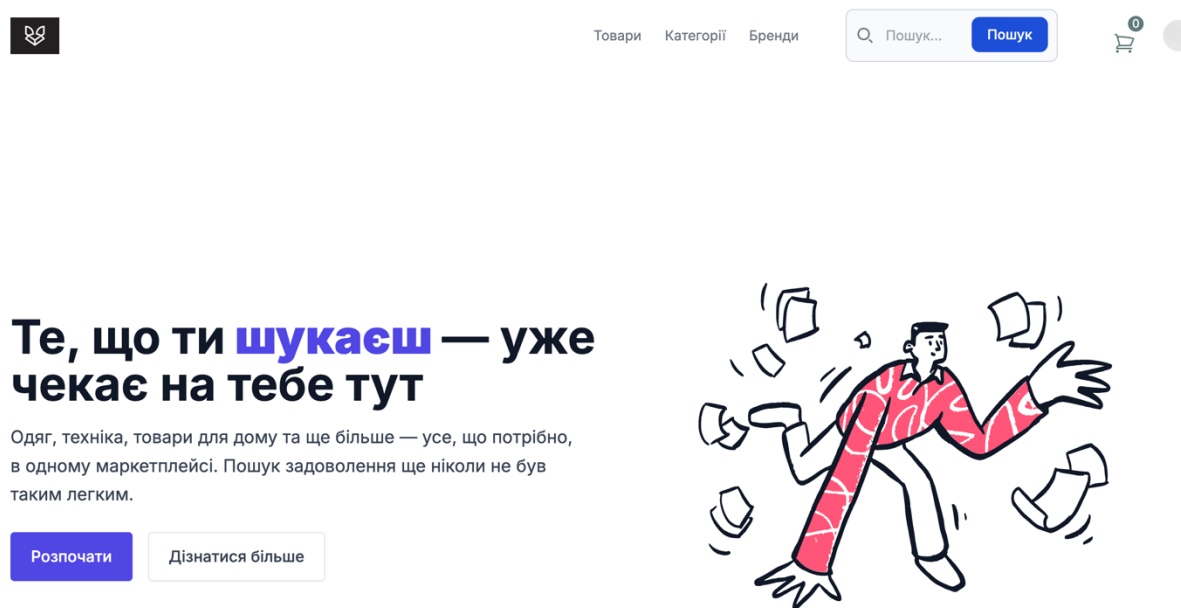


Рис. 3.2. Головна сторінка

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

До моменту авторизації, користувач може взаємодіяти з продуктами, адже хоч і на даний момент прив'язаних до користувача об'єктів корзини та 'улюблене' немає, тому у ся інформація зберігається у Local Storage, також доступна опція пошуку(рис.3.3).

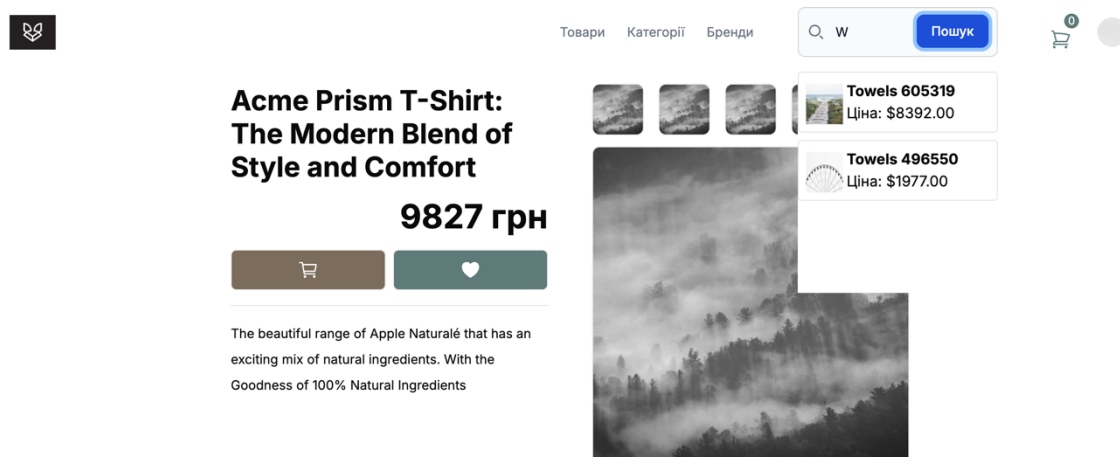


Рис. 3.3. Пошук продуктів

Також доступні для перегляду сторінки брендів та на категорій(рис. 3.4)

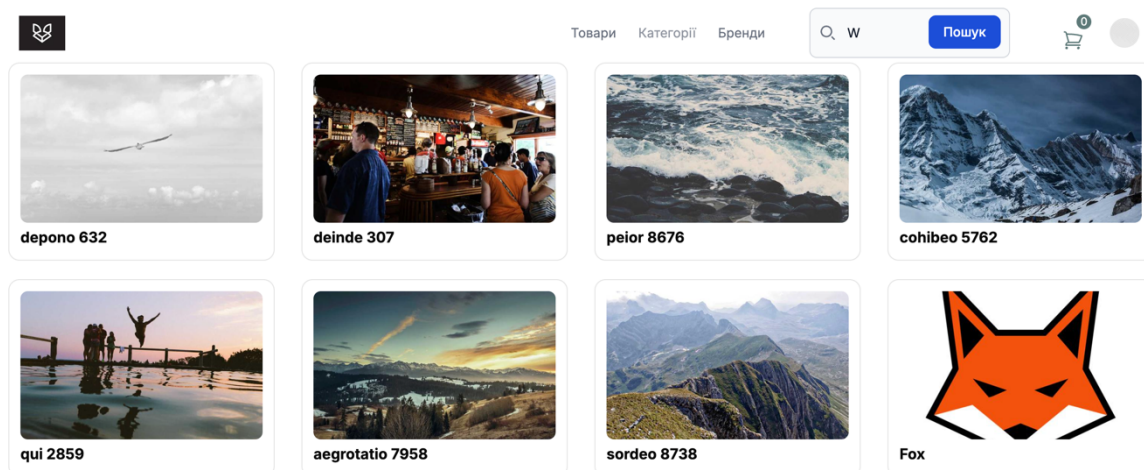


Рис. 3.4. Сторінка категорій

Уся авторизація авторизація відбувається за рахунок NextAuth через сервіси Google[14] та Github[15], що спрощує авторизацію оскільки користувачеві не будк потрібно витратити час на заповнення даних, створення та запам'ятовування

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

паролів(рис. 3.5). Уся необхідна інформація буде отримання від обліком записів сторонніх сервісів, також на необхідністю користувач зможе мануально оновити данні в особистому кабінеті(рис. 3.6).

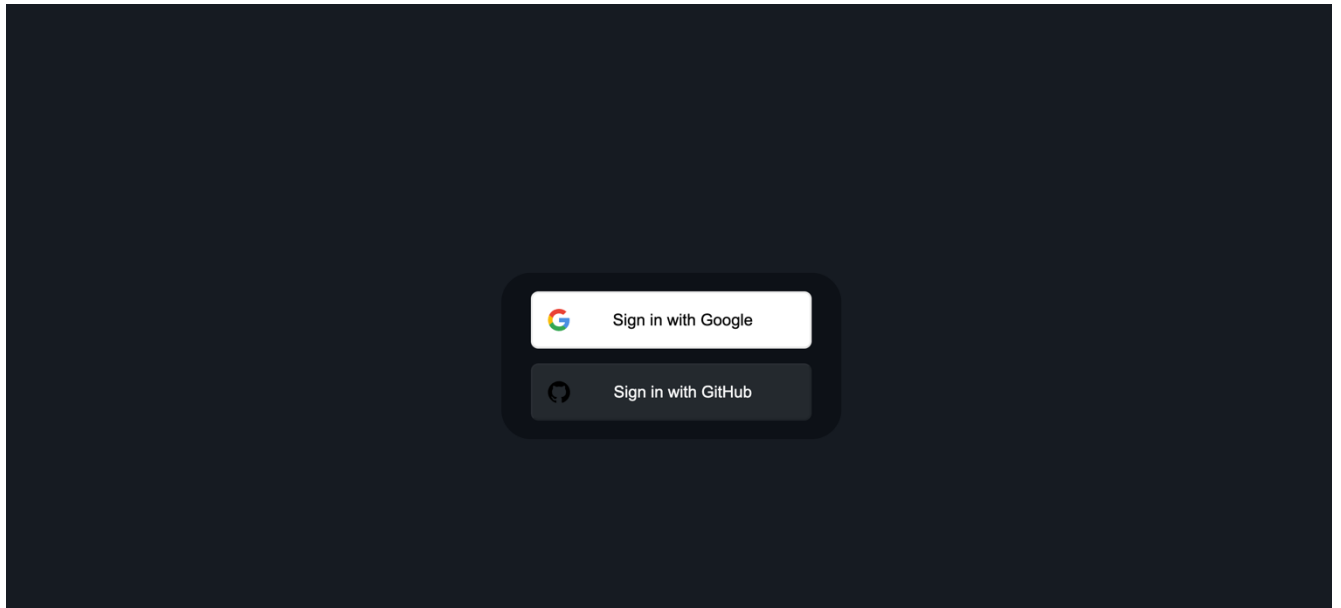


Рис. 3.5. Сторінка авторизації

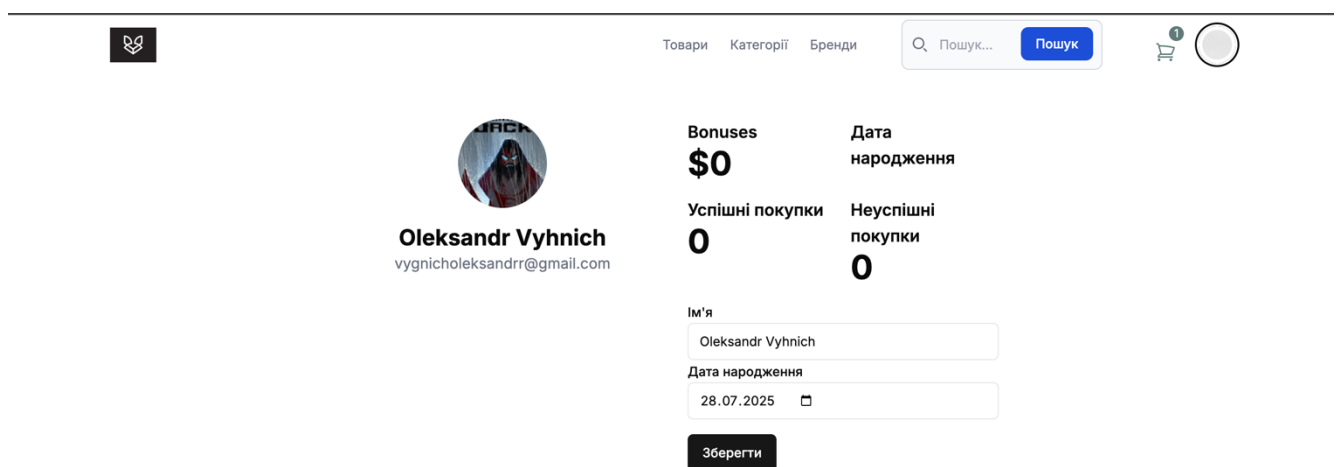
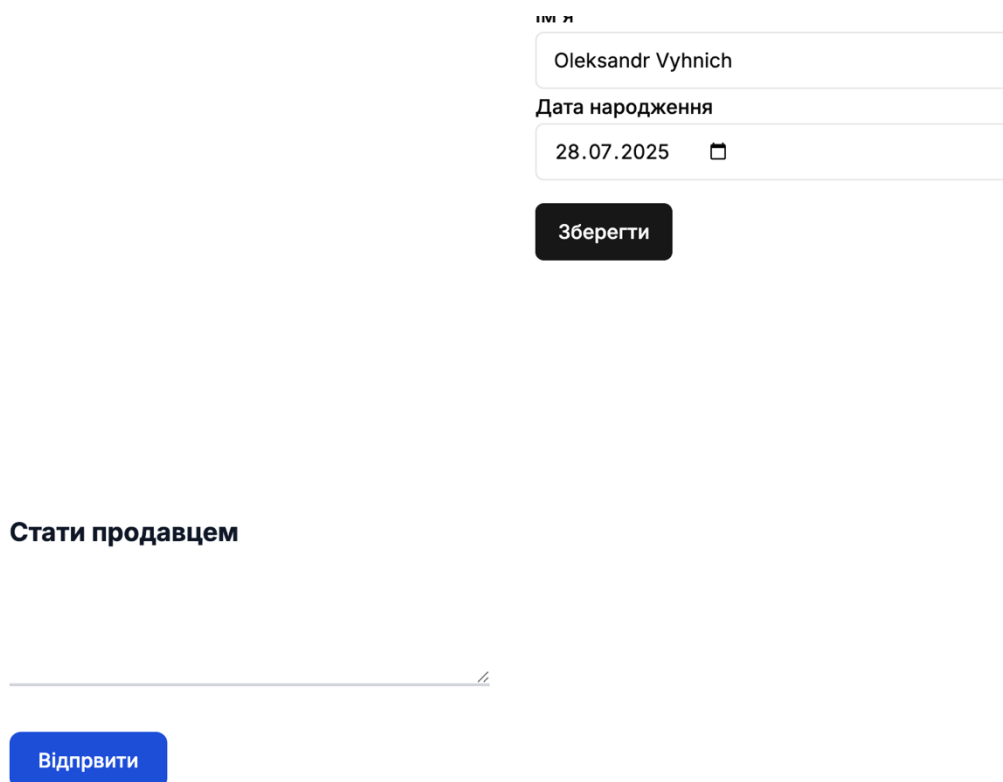


Рис. 3.6. Сторінка профілю

Після авторизації користувач може безпечно поповнювати собі корзину без страху, що продукти кудись зникнуть оскільки далі продукти будуть зберігатися у базі даних з прив'язкою до користувача(рис. 3.7). Той це принцип працює до

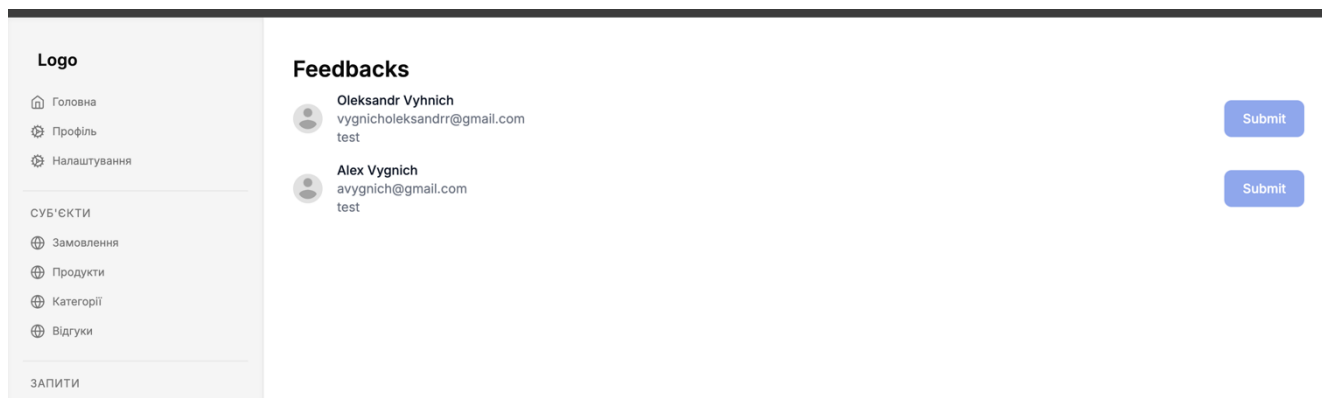
					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

‘улюблених’ поварів. Більш з цікавого функціоналу – це форма “Стати продавцем” вона відправить повідомлення адміністратору, якщо зможе розглянути заявки та прийняти чи відхилити їх(рис. 3.8).



The form is titled "Стати продавцем" (Become a seller). It contains two input fields: the first for the name "Oleksandr Vyhnich" and the second for the date of birth "28.07.2025" with a calendar icon. Below these fields is a dark button labeled "Зберегти" (Save). At the bottom of the form is a blue button labeled "Відправити" (Send).

Рис. 3.7. Форма “Стати продавцем”



The screenshot shows a web interface with a sidebar on the left containing a "Logo" section with links to "Головна" (Home), "Профіль" (Profile), and "Налаштування" (Settings), and a "СВБ'ЄКТИ" (Subjects) section with links to "Замовлення" (Orders), "Продукти" (Products), "Категорії" (Categories), and "Відгуки" (Reviews). The main content area is titled "Feedbacks" and lists two entries: "Oleksandr Vyhnich" with email "vygnicholeksandrr@gmail.com" and "Alex Vygnich" with email "avygnich@gmail.com". Each entry has a "Submit" button to its right.

Рис. 3.8. Сторінка розгляду заявок

На сторінці продукту користувач має змогу переглянути детальну інформацію, про продукт та опції такі, як додати до кошика чи в ‘улюблене’(рис. 3.9).

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		



Acme Prism T-Shirt: The Modern Blend of Style and Comfort

9827 грн

The beautiful range of Apple Naturalé that has an exciting mix of natural ingredients. With the Goodness of 100% Natural Ingredients



Рис. 3.9. Сторінка продукту

Основною сторінкою платформи фактично є сторінка брендів, оскільки тут люди, які вирішили розпочати на нашій платформі комерційну діяльність можуть створювати свій простір(рис. 3.10). Налаштовувати банер для пізнаванності бренду та продавати власну продукцію.

Fox

Вас вітає "Fox studio"



Рис. 3.10. Сторінка бренду

На даному етапі можна глянути панель адміністратора. Фактично у адміністратора є доступ до усіх сутностей платформи, з якими має повний доступ взаємодій, таким чином від може швидко виконувати задачі, по моніторингу та модерації контенту платформи(рис. 3.11). Основною, роботою адміністратора, зрештою, є перегляд запитів на зміну ролі на платформі від користувачів, які наважитись відкрити власний бренд в електронній комерції.

Змн.	Арк.	№ докум.	Підпис	Дата

Logo

Головна

Профіль

Налаштування

СУБ'ЄКТИ

Замовлення

Продукти

Категорії

Відгуки

ЗАПИТИ

Запити на продавця

Feedbacks

Oleksandr Vyhnic

vygnicholeksandrr@gmail.com

test

Alex Vygnich

avygnich@gmail.com

test

Submit

Submit

Рис. 3.11. Сторінка розгляду заявок

Настав час розглянути ту ж сторінку адміністратора, але вже в ролі підприємця(рис. 3.12). Оскільки усі продавці ізольовані одне від одного, то в кожен продавець має лише доступ до даних, які пов'язанні лише з цього бредном

Logo

Головна

Профіль

Налаштування

СУБ'ЄКТИ

Замовлення

Продукти

Категорії

Відгуки

ЗАПИТИ

Запити на продавця

<

Корм для собак12

Edit

```
{
  "id": "cmd277xps0003g1jlb7zz9kv1",
  "title": "Корм для собак12",
  "description": "",
  "photo": "https://darwin.kh.ua/image/cache/catalog/%D0%9A%D0%90%D0%A0%D0%A2%D0%98%D0%9D%D0%9A%D0%98/%D0%9A%D0%BB%D1%83%D0%B1%204%20%D0%9B%D0%B0%D0%BF%D1%8B/%D0%A1%D0%BE%D0%B1%D0%B0%D0%BA%D0%B0/%D0%A1%D1%83%D1%85%D0%BE%D0%B9%20%D0%BA%D0%BE%D1%80%D0%BC/Adult%20Small%20Breeds%20%2C4kg-680x630.jpeg",
  "price": 256.99,
  "createdAt": "2025-07-13T21:40:40.429Z",
  "updatedAt": "2025-07-13T21:40:40.429Z",
  "categoryId": "cmd20vnl000k12689gbgphgw",
  "brandId": "cmd21kx8v000fe8am0uusb3tt"
}
```

Рис. 3.12. Сторінка створення продуктів

Також цікавою особливістю, користування з ролю ‘Seller’ це сторінка з налаштуванням бренду, це від може налаштовувати контент для власної індивідуальності(рис. 3.13)

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						47
Змн.	Арк.	№ докум.	Підпис	Дата		

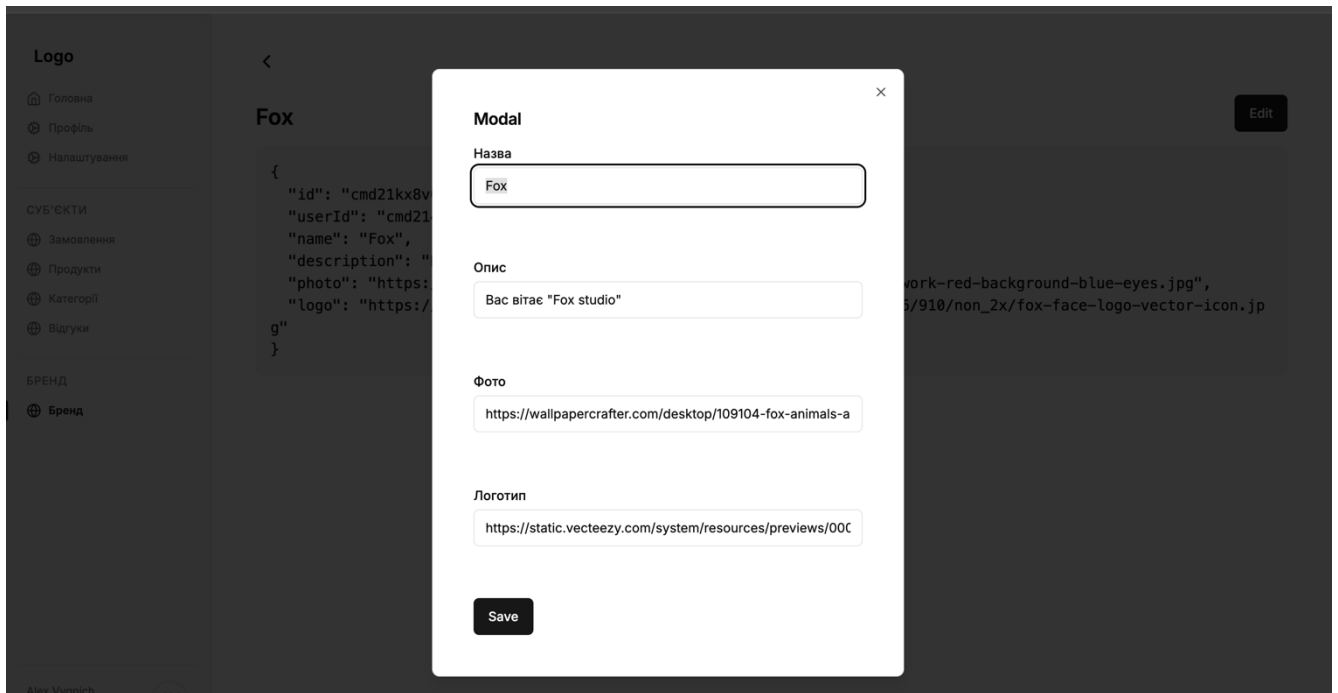


Рис. 3.13. Сторінка налаштування бренду

3.3. Тестування роботи програмного продукту

Щоб переконатися, що платформа працює стабільно і правильно обробляє всі запити та відповіді, я перевіряв її різними способами. Основну увагу приділяв тому, як клієнтська частина взаємодіє з серверною - важливо було, щоб усе працювало без збоїв і давало очікуваний результат.

Для тестування бекенду я використовував Postman. Це зручний інструмент, який дозволяє вручну надсилати запити до потрібних ендпоїнтів і одразу бачити, що саме повертає сервер. Так можна легко перевірити, як система реагує на правильні або неправильні дані, і впевнитися, що все працює так, як задумано.

Щоб не вводити авторизаційні дані вручну під час кожного запиту, я написав невеликий скрипт, який автоматизував цей процес. Він логінується під потрібним користувачем, зберігає токен і автоматично підставляє його в заголовки всіх наступних запитів. Це значно спростило тестування захищених маршрутів.

Під час розробки вдалося виявити й опрацювати типові помилки, які можуть виникати під час роботи з платформою. Головна мета — зробити так, щоб у разі

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

проблем користувач бачив зрозуміле повідомлення про те, що саме пішло не так, а не просто біле вікно чи технічну помилку.

Щоб перевіряти введені дані в усіх формах на сайті (наприклад, при реєстрації, вході або створенні публікацій), я використав бібліотеку Zod[25]. Для кожної форми була створена окрема схема, яка чітко описує, які поля є обов'язковими, що саме можна вводити і які дані вважаються некоректними. У разі помилки ця схема не тільки зупиняє неправильний запит, а й дозволяє зібрати всі повідомлення про помилки, щоб їх можна було зручно показати користувачу.

Висновки до третього розділу

У цьому розділі я описав увесь життєвий цикл платформи - від першого розгортання до фінального тестування функцій. Для того щоб усе працювало стабільно та було готове до зростання навантаження, я використав Docker[22], такж для локального розгортання був використаний Docker Compose[27]. Завдяки цьому вдалося зібрати інфраструктуру, яка легко переноситься та масштабуються за потреби.

Уся система побудована так, щоб клієнтська частина, серверна логіка й сховище чітко взаємодіяли між собою. Структура продумана, щоб уникати хаосу в коді й зробити підтримку простішою.

Що стосується інтерфейсу, то він створений з акцентом на зручність - усе інтуїтивне, від перегляду публікацій до їх створення чи редагування. Користувач може легко зареєструватися, увійти в акаунт і почати взаємодіяти з іншими: залишати вподобання, коментарі або навіть щось купувати.

Я приділив окрему увагу тестуванню. Воно допомогло знайти та виправити критичні помилки ще до того, як вони могли завадити користувачу. В результаті вдалося зібрати систему, яка працює надійно, виглядає сучасно і при цьому залишає приємне враження від взаємодії.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У процесі написання цієї кваліфікаційної роботи вдалося пройти повний шлях створення сучасної вебплатформи-маркетплейсу - від першої ідеї до вже готового та протестованого продукту. Основна мета полягала в тому, щоб дати користувачам простий інструмент для створення, редагування та публікації власних товарів, з акцентом на розвиток персонального бренду. Для цього я використав сучасні інструменти розробки, які дозволили зібрати гнучку, масштабовану та зручну систему.

Перший розділ був присвячений аналізу схожих платформ. Це дало змогу чітко зрозуміти, які функції будуть справді корисними, а які - зайвими. Після порівняння різних технологій я зупинився на стеку Next.js, TypeScript[13], Node.js[23] і PostgreSQL - ці інструменти виявилися оптимальними з точки зору продуктивності, масштабованості та безпеки. Тут же було закладено загальну архітектуру проєкту: як виглядатиме система в цілому, з яких частин складатиметься та як вони взаємодіятимуть між собою.

У другому розділі акцент був зроблений на проєктуванні. Я створив діаграми, які описують типові сценарії користування платформою — від простого перегляду до дій адміністратора. Класова діаграма допомогла спроектувати структуру даних, а діаграми активності дозволили формалізувати основні алгоритми роботи, такі як реєстрація, авторизація, публікація чи редагування товарів. Усе це дозволило побудувати систему, яка легко розширюється і чітко розділяє відповідальність між різними компонентами.

У третьому розділі йдеться вже безпосередньо про реалізацію. Платформа була розгорнута з використанням Docker, що забезпечило стабільну роботу незалежно від середовища. Фронтенд і бекенд зв'язані між собою, працює взаємодія з сховищем. Реалізовано зручний інтерфейс з можливістю реєстрації, авторизації, перегляду й створення продуктів, їх редагування, коментування й навіть покупки - усе, що потрібно для живої взаємодії між користувачами.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрему увагу я приділив тестуванню. Перевірялися як запити на сервер, так і поведінка клієнтської частини. Це дозволило знайти помилки до того, як із ними могли зіткнутися користувачі, і переконатися, що платформа поводить себе стабільно навіть у складних сценаріях.

Загалом, ця робота показала, як можна комплексно підійти до створення повноцінного вебпродукту - не просто «написати сайт», а розробити технічно надійну, продуману з точки зору користувача систему. Платформа повністю відповідає вимогам, які були поставлені на старті, й готова до подальшого розвитку.

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інтернет-магазин Prom [електронний ресурс] – Режим доступу до ресурсу: <https://prom.ua/>
2. Інтернет-магазин Rozetka [електронний ресурс] – Режим доступу до ресурсу: <https://rozetka.com.ua/>
3. Інтернет магазин товарів для дому Amazon [Електронний ресурс] – Режим доступу до ресурсу: <https://www.amazon.com/>
4. Документація PostgreSQL [Електронний ресурс] – Режим доступу до ресурсу: <https://postgrespro.com/docs/postgresql>
5. Документація pgAdmin [Електронний ресурс] – Режим доступу до ресурсу: <https://www.pgadmin.org/docs/>
6. Docs | Next.js [Електронний ресурс]. Режим доступу до ресурсу: <https://nextjs.org/docs>
7. Styling: Tailwind CSS | Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/app/building-your-application/styling/tailwind-->
8. Configuring: TypeScript | Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/pages/building-your-application/configuring/typescript>
9. Routing: API Routes | Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/pages/building-your-application/routing/api-routes>
10. Next.js | Functions [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/app/api-reference/functions>
11. Building Your Application: Data Fetching | Next.js [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/pages/building-your-application/data-fetching>
12. Vercel Deployment [Електронний ресурс] – Режим доступу до ресурсу: <https://vercel.com/docs/deployments/claim-deployments>

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

13. next.js-typescript-starter-kit [Електронний ресурс] – Режим доступу до ресурсу: <https://nextjs.org/docs/app/getting-started/project-structure>
14. Документація для використання Google OAuth [Електронний ресурс] – Режим доступу до ресурсу:
<https://developers.google.com/identity/protocols/oauth2>
15. Документація для використання Git OAuth [Електронний ресурс] – Режим доступу до ресурсу:
<https://docs.github.com/en/authentication>
16. Документація до Postman [Електронний ресурс] – Режим доступу до ресурсу: <https://learning.postman.com/docs/introduction/overview/>
17. React Documentation – Офіційна документація. – [Електронний ресурс]. – Режим доступу: <https://react.dev>
18. Prisma Documentation – Офіційна документація. – [Електронний ресурс]. – Режим доступу: <https://www.prisma.io/docs>
19. Офіційний сайт маркетплейсу Etsy. – [Електронний ресурс]. – Режим доступу: <https://www.etsy.com>
20. Офіційний сайт сервісу Booking.com. – [Електронний ресурс]. – Режим доступу: <https://www.booking.com>
21. Git Documentation – Офіційна документація. – [Електронний ресурс]. – Режим доступу: <https://git-scm.com/doc>
22. Docker Documentation – Офіційна документація. – [Електронний ресурс]. – Режим доступу: <https://docs.docker.com>
23. Node.js Documentation – Офіційна документація. – [Електронний ресурс]. – Режим доступу: <https://nodejs.org/en/docs>
24. NPM Documentation – Офіційна документація. – [Електронний ресурс]. – Режим доступу: <https://docs.npmjs.com>
25. Zod Documentation – [Електронний ресурс]. – Режим доступу: <https://zod.dev>
26. react-hook-form Documentation – [Електронний ресурс]. – Режим доступу: <https://react-hook-form.com/get-started>

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

27. Docker Compose– Офіційна документація. – [Електронний ресурс]. – Режим доступу: <https://docs.docker.com/compose/>
28. Prisma Zod Generator – [Електронний ресурс]. – Режим доступу: <https://omar-dulaimi.github.io/prisma-zod-generator/>
29. Tailwindcss ui-kit. – [Електронний ресурс]. – Режим доступу: <https://tailwindcss.com/plus/ui-kit>
30. Lighthouse Documentation (Google Chrome Developers). – [Електронний ресурс]. – Режим доступу: <https://developer.chrome.com/docs/lighthouse>

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						55
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг файлу файлу utlis/auth/utlis.ts

```

/dist/shared/lib/app-router-context.shared-runtime';
declare module 'next-auth' {
  interface Session {
    user: DefaultSession['user'] & {
      id: string;
    };
  }
}

```

```

export type AuthSession = {
  session: {
    user: {
      id: string;
      name?: string;
      email?: string;
      role?: string;
    };
  } | null;
};

```

```

export const authOptions: NextAuthOptions = {
  adapter: PrismaAdapter(db) as Adapter,
  callbacks: {
    session: ({ session, user }) => {
      session.user.id = user.id;
      // @ts-ignore
      session.user.role = user.role;
      return session;
    },
  },
  providers: [
    GoogleProvider({
      clientId: env.GOOGLE_CLIENT_ID,
      clientSecret: env.GOOGLE_CLIENT_SECRET,
    }),
    GitHubProvider({
      clientId: env.GITHUB_CLIENT_ID,
      clientSecret: env.GITHUB_CLIENT_SECRET,
    }),
  ],
};

```

```

export const getUserAuth = async () => {
  const session = await getServerSession(authOptions);
  return { session } as AuthSession;
};

```

```

export const checkAuth = async () => {
  const { session } = await getUserAuth();

```

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						56
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if (!session) redirect('/api/auth/signin');
if (session?.user.role === 'USER') {
  throw new Error('You need to be an admin');
}
}

export const signOut = async (router: AppRouterInstance) => {
  const response = await fetch('/api/sign-out', {
    method: 'POST',
    redirect: 'manual',
  });

  if (response.status === 0) {

    return router.refresh();
  }
};

export const getIsLoggedIn = async () => {
  const { session } = await getUserAuth();
  return !!session;
};

```

Лістинг коду фійлу `hooks/useValidatedForm.tsx`:

```

'use client';

import { FormEvent, useState } from 'react';
import { ZodSchema } from 'zod';

type EntityZodErrors<T> = Partial<Record<keyof T, string[] | undefined>>;

export function useValidatedForm<Entity>(insertEntityZodSchema: ZodSchema) {
  const [errors, setErrors] = useState<EntityZodErrors<Entity> | null>(null);
  const hasErrors = errors !== null
    && Object.values(errors).some((error) => error !== undefined);

  const handleChange = (event: FormEvent<HTMLFormElement>) => {
    const target = event.target as EventTarget;
    if (
      target instanceof HTMLInputElement
      || target instanceof HTMLSelectElement
      || target instanceof HTMLTextAreaElement
    ) {
      if (!(target instanceof HTMLInputElement && target.type === 'submit')) {
        const field = target.name as keyof Entity;
        const result = insertEntityZodSchema.safeParse({
          [field]: target.value,
        });
        const fieldError = result.success
          ? undefined
          : target.error.flatten().fieldErrors[field];
      }
    }
  };
}

```

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						57
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    setErrors((prev) => ({
      ...prev,
      [field]: fieldError,
    }));
  }
}
};
return {
  errors, setErrors, handleChange, hasErrors,
};
}

```

Лістинг коду з файлу prisma/schema.prisma:

```

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

generator zod {
  provider      = "zod-prisma"
  output        = "./zod"
  relationModel = true
  modelCase     = "camelCase"
  modelSuffix   = "Schema"
  useDecimalJs  = true
  prismaJsonNullability = true
}

datasource db {
  provider = "postgresql"
  url      = env("POSTGRES_PRISMA_URL") // uses connection pooling
  directUrl = env("POSTGRES_URL_NON_POOLING") // uses a direct connection
}

model Account {
  id          String @id @default(cuid())
  userId      String
  type        String
  provider     String
  providerAccountId String
  refresh_token String? @db.Text
  access_token String? @db.Text
  expires_at   Int?
  token_type   String?
  scope        String?
  id_token     String? @db.Text
  session_state String?

  user User @relation(fields: [userId], references: [id], onDelete: Cascade)

  @@unique([provider, providerAccountId])
}

```

```

}

model Session {
  id      String  @id @default(cuid())
  sessionToken String @unique
  userId  String
  expires  DateTime
  user    User    @relation(fields: [userId], references: [id], onDelete: Cascade)
}

enum UserRole {
  ADMIN
  USER
  SELLER
}

enum RoleStatus {
  APPROVED
  REJECTED
  INPROGRESS
}

model User {
  id      String  @id @default(cuid())
  name    String?
  email   String? @unique
  emailVerified DateTime?
  image   String?
  dateOfBirth DateTime?
  successPurchases Int?
  failPurchases  Int?
  bonuses        Int?
  accounts       Account[]
  sessions       Session[]
  orders         Order[]
  feedbacks      Feedback[]
  carts          Cart[]
  favorites      Favorite[]
  roleStatus     RoleStatus @default(APPROVED)
  role           UserRole   @default(USER)
  AdminMessage   AdminMessage?
  brand          Brand?

  @@index([id])
  @@index([email])
}

model VerificationToken {
  identifier String
  token      String @unique
  expires    DateTime

  @@unique([identifier, token])

```

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

model Order {
  id      String      @id @default(cuid())
  number  Int          @unique @default(autoincrement())
  price   Float
  notes   String?
  userId  String
  user    User         @relation(fields: [userId], references: [id], onDelete: Cascade)
  createdAt DateTime   @default(now())
  updatedAt DateTime   @updatedAt
  orderProducts OrderProducts[]

  @@index([id])
  @@index([userId])
}

```

```

model OrderProducts {
  id      String @id @default(cuid())
  product Product @relation(fields: [productId], references: [id])
  productId String
  order   Order  @relation(fields: [orderId], references: [id])
  orderId String
  count   Int
}

```

```

model Product {
  id      String      @id @default(cuid())
  title   String       @unique
  description String?
  photo   String?
  price   Float
  createdAt DateTime   @default(now())
  updatedAt DateTime   @updatedAt
  category Category    @relation(fields: [categoryId], references: [id])
  categoryId String
  feedback Feedback[]
  orderProducts OrderProducts[]
  Cart     Cart[]
  Favorite Favorite[]
  brandId  String
  brand    Brand       @relation(fields: [brandId], references: [id])

  @@index([title])
}

```

```

model Category {
  id      String @id @default(cuid())
  title   String @unique
  description String?
  photo   String?
  products Product[]
  createdAt DateTime @default(now())
}

```

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    updatedAt DateTime @updatedAt

    @@index([title])
}

model Feedback {
  id String @id @default(cuid())
  text String
  rating Int?
  product Product @relation(fields: [productId], references: [id], onDelete: Cascade)
  productId String
  userId String
  user User @relation(fields: [userId], references: [id], onDelete: Cascade)
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([userId])
}

model Cart {
  id String @id @default(cuid())
  product Product @relation(fields: [productId], references: [id], onDelete: Cascade)
  productId String
  count Int
  userId String
  user User @relation(fields: [userId], references: [id], onDelete: Cascade)

  @@index([userId])
}

model Favorite {
  id String @id @default(cuid())
  product Product @relation(fields: [productId], references: [id], onDelete: Cascade)
  productId String
  userId String
  user User @relation(fields: [userId], references: [id], onDelete: Cascade)
  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  @@index([userId])
}

model AdminMessage {
  id String @id @default(cuid())
  userId String @unique
  user User @relation(fields: [userId], references: [id])
  message String
}

model Brand {
  id String @id @default(cuid())
  owner User @relation(fields: [userId], references: [id])
  userId String @unique

```

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

```
Products  Product[]
name      String  @unique
description String?
photo     String?
logo      String?
}
```

					ІПЗ.КР.Б - 121 - 25 - ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		