ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Веб-додаток для онлайн-торгівлі на основі мови програмування Java»**

на здобуття освітнього ступеня бакалавр
за спеціальності 122 Комп'ютерні науки
(код,найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(першого бакалаврського рівня вищої освіти)
(назва)

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

_____ Михайло ЧУМАКОВ
(підпис) *ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр.КНД-41
Михайло ЧУМАКОВ
ім'я, ПРІЗВИЩЕ
Керівник: Сергій СЄРИХ
ім'я, ПРІЗВИЩЕ

Рецензент:
ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерних наук
Ступінь вищої освіти - «Бакалавр»
Спеціальність -122 - комп'ютерні науки

ЗАТВЕРДЖУЮ

Завідувач кафедри
Комп'ютерних наук

_____ В.В. Вишнівський
"____" _____2021 року

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Чумаков Михайло Юрійович

(прізвище, ім'я, по батькові)

1. Тема роботи: «Веб-додаток для онлайн-торгівлі на основі мови програмування Java»

Керівник роботи Сєрих Сергій Олександрович, к.т.н., доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від "24" лютого 2025 р. №56.

2. Строк подання студентом роботи «30» травня 2025 р.

3. Вхідні дані до роботи:

Застосунки для онлайн торгівлі, функціональний інтерфейс, безпечний технологічний стек, розширювана архітектура, науково-технічна література з питань розробки веб-додатків

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1 Аналіз вимог до сучасних комерційних веб-додатків.

4.2 Вибір стеку технологій та проєктування архітектури веб-додатку.

4.3 Розробка власної платформи онлайн-оголошень за моделлю C2C на мові програмування Java

5. Перелік графічного матеріалу

1. Актуальність теми
2. Сучасні функціональні та нефункціональні вимоги до веб-додатків електронної комерції
3. Виділення переваг та недоліків у аналогічних сервісів
4. Вибір мови програмування та інструментів для розробки веб-додат

5. Архітектура додатку
6. ER-діаграма бази даних додатку TradeShop
7. UI/UX дизайн та логіка навігації додатку TradeShop
8. Логіка обробки HTTP запитів (запитів користувача) на серверній частині
9. Висновки
6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Огляд науково-технічної літератури	25.02-01.03.2025	Вик.
2	Аналіз аналогічних платформ та вимог для комерційних веб-додатків	03.03-18.03.2025	Вик.
3	Формування цілей та завдань, аналіз інструментів для розробки	20.03-26.03.2025	Вик.
4	Архітектурне планування веб-додатку	26.03-05.04.2025	Вик.
5	Розробка власного веб-додатку на основі мови програмування Java	06.04-09.05.2025	Вик.
6	Оформлення звітів та висновків	10.05-19.05.2025	Вик.
7	Розробка демонстраційних матеріалів	20.05-30.05.2025	Вик.

Здобувач(ка) вищої освіти _____ Михайло ЧУМАКОВ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Сергій СЄРИХ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина бакалаврської роботи: 55 с., 5 табл., 11 рис., 2 дод., 20 джерел.

КЛЮЧОВІ СЛОВА: BOOTSTRAP, ЕЛЕКТРОННА КОМЕРЦІЯ, FREEMARKER, HTML, JAVA, LOMBOK, MAVEN, MVC, ОНЛАЙН-ТОРГІВЛЯ, REST-API, POSTGRESQL, SPRING BOOT, UI/UX ДИЗАЙН, ВЕБ-ДОДАТОК.

Об’єкт дослідження – архітектура веб-додатків у сфері електронної комерції.

Предмет дослідження – методи розробки веб-додатків для C2C торгівлі.

Мета роботи – розробка веб-додатку онлайн-торгівлі на мові програмування Java з використанням сучасних фреймворків та бібліотек.

Методи дослідження – методи об’єктно орієнтованого програмування, методи аналізу функціональних систем, проєктування архітектури веб-додатків.

Визначено вимоги до комерційних веб-додатків, принципи проєктування та розробки оптимізованого інтерфейсу для реалізації платформи.

Здійснено реалізацію веб-додатку на мові програмування Java з використанням фреймворку Spring Boot та забезпечення взаємодії з базою даних PostgreSQL. Розроблений застосунку враховує потребу в надійному захисті та впровадження системи аутентифікації користувачів

На основі результатів виконаних досліджень розроблено повнофункціональний приклад системи онлайн-торгівлі.

Упровадження розробленого додатку дозволяє автоматизувати процеси популяризації та просування особистих оголошень без додаткових витрат, що може використовуватись як основа для запуску повноцінної платформи C2C моделі.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI (англ. User interface) - зовнішній вигляд сайту.

Бойлер-код (англ. Boilerplate code) – це часто повторюваний код, який не має унікальних змін в різних частинах додатку, але є необхідним для правильної роботи програми.

C2C (англ. consumer-to-consumer) – це модель взаємодії між користувачами, за якою обмін товарами чи послугами відбувається між споживачами, через онлайн-платформи, що виступають посередником для безпеки.

B2C (англ. Business-to-consumer) – Це модель взаємодії, в якій бізнес продає товари чи послуги безпосередньо кінцевим споживачам. Може також реалізовуватись з посередниками (маркетплейсами).

Маркетплейси – Це платформи, в яких продавці можуть розміщувати свої товари чи послуги для продажу кінцевим споживачам.

Cookie – Це невеликі фрагменти текстових даних, які веб-сайти зберігають у браузері, під час взаємодії користувача з ними.

ШІ (Штучний інтелект) – Інтелектуальні машини, які імітують поведінку людини.

ОС (Операційна система) – Основне програмне забезпечення що керує ресурсами комп’ютера.

БД (База даних) – сховище, для зберігання та обробки інформації.

ВСТУП

У сучасному світі торгівля через інтернет ресурси, стає все пріоритетнішою. Онлайн-торгівля вже є невід'ємною частиною життя більшості людей. Завдяки розвитку телекомунікацій інтернет зв'язок став навіть у селах, будь який користувач може звернутися до провайдера і укласти контракт з встановленням необхідного обладнання. В умовах розвинених сервісів з доставки, доставка займає 1-3 дні. У менших населених пунктах, де комерція менш розвинена, набагато легше зайти в браузер на телефоні або комп'ютері, відкрити необхідний сайт і оформити замовлення з доставкою по всій країні. На сьогоднішній день вже є дуже багато ресурсів з вільним доступом, де кожен може обрати для себе ідеальний товар/послугу або створити на зручному сервісі своє оголошення.

Метою цієї дипломної роботи є розробка веб-додатку, що дозволяє користувачам зручно здійснювати пошук товарів, створювати свої оголошення, або переглядати оголошення від інших користувачів та взаємодіяти зі сторінкою через зручний інтерфейс.

В умовах високої конкуренції на ринку важливо створювати зручні, безпечні та функціональні веб-додатки, які забезпечують якісну взаємодію між користувачами. Саме тому будуть проаналізовані сучасні вимоги та аналоги для розробки власного веб-додатку для онлайн-торгівлі. Популярність подібних сервісів показує наскільки є актуальною тема розробки свого веб-додатку.

Розробка такого додатку передбачає глибокі знання у галузі веб-програмування та вимагає поглиблених знань необхідних інструментів для роботи. У межах цієї дипломної роботи створено власний веб-додаток онлайн торгівлі між користувачами, якими можуть бути або юридична фірма, або звичайною людиною. Додаток дозволятиме зручно та швидко проходити реєстрацію та авторизацію, шукати необхідний товар, забезпечує перегляд профілю кожного зареєстрованого користувача та доступ до контактних даних продавця.

Технічна реалізація буде здійснена на мові програмування Java, з використанням фреймворку Spring Boot. Веб-додаток використовує власну базу даних – “PostgreSQL”. Також будуть задіяні інші інструменти для оптимізації коду, наприклад, бібліотека “Lombok” – яка мінімізує кількість шаблонного бойлер-коду, “Spring Security” – для авторизації та реєстрації користувачів.

Таким чином, дипломна робота передбачає аналіз ефективності у використанні сучасних фреймворків, бібліотек у розробці. Будуть досліджені позитивні та негативні сторони готових рішень, з урахуванням яких, складено детальну архітектуру. Робота поєднує теоретичні і практичні навички у сфері веб-програмування, що має велику цінність у сучасних умовах розвитку IT-сфери.

1 АНАЛІЗ ВИМОГ ДЛЯ РОЗРОБКИ КОМЕРЦІЙНОГО ВЕБ-ДОДАТКУ ЗА МОДЕЛЛЮ C2C

1.1 Аналіз сучасного стану електронної комерції та цифрових сервісів

Інтернет є одним із найдинамічніше зростаючих засобів комунікації в історії людства**,** який продовжує проникати в різні сфери життя людства. Відмовитися від використання інтернету економічним сферам з кожним днем стає важче. На державному рівні в Україні вже відцифровують документи, здійснюється повна цифровізація послуг з метою оптимізації документообігу та зменшення навантаження на відповідні структури для боротьби з великою кількістю паперових документів та чергами. При умовах, якщо більшість країн перенесуть всі послуги в інтернет-режим, це знизить попит на виробництво паперу, що призведе до зменшення вирубки лісів. Сьогодні велику частину інформації можна передати з мінімальними витратами.

Електронна комерція є найперспективнішою галуззю навіть в IT сфері, яка по сьогодні розвивається з дуже ефективною швидкістю для вдосконалення умов користування послугами. Через постійний розвиток і вдосконалення інструментів розробки та сприйняття людиною інтернет сервісів спонукає шукати нові вирішення проблем та задач. Конкуренція в інтернет просторі без перерв змушує вдосконалювати веб-додатки, аналізувати відгуки та поведінку користувачів, на це впливає активний сеанс на сторінці, взаємодія з інтерфейсом та інтересами до інших частин сайту. Однією із найважливіших умов досягання позитивних емоцій від користування є – дизайн, від його якості залежить первинне враження про ресурс.

Добре продуманий та реалізований UI робить навігацію по сайту інтуїтивно зрозумілою, не вимагає довго шукати необхідний контент чи функціонал. Усе це підвищує рівень задоволеності користувача, а також підвищує ефективність самого ресурсу. Через інтерфейс формується емоційний зв’язок та загальний імідж

продукту. Середовище подання контенту, кольорова гама та загальний стиль візуального оформлення впливають на ідентичність бренду. У сучасних умовах важливо, щоб UI був адаптивним на різних платформах – від комп'ютерів до телевізорів. Тому UI не лише є технічного або дизайнерською складовою, а й стратегічним елементом, що впливає на функціонування ресурсу.

Станом на сьогодні, більша частина виробників телефонів, комп'ютерів, та веб-ресурсів намагаються створити та впровадити в свої продукти штучний інтелект. Впровадження ШІ в інтернет-ресурси змінює спосіб взаємодії, так і звички споживача які дозволяють отримати більш персональний досвід. Аналізуючи поведінку, інтереси та взаємодію на ресурсі, пропонують індивідуальні послуги та вид товарів, які зможуть зацікавити користувача. Це зменшує інформативне перевантаження та економить час що підвищує задоволення користуванням інтернет ресурсом. Використання чат-ботів та віртуальних асистентів дає змогу оперативно отримати відповідь на поширенні запити користувача незалежно від часу доби. Машини завжди працюють та створюють відповідь в залежності від поведінки користувача, що додатково позитивно впливають на довіру користувача до ресурсу. Таким чином, ШІ змінює технічну сторону систем, та й впливають на користувача, роблячи досвід користування більш комфортним та приємним.

У багатьох країнах, зокрема в Україні, спостерігається зростання онлайн продажів. Поява зручних та швидких сервісів доставки робить процес покупки більш приємним. Вони суттєво розвинули логістику, зробивши її доступною навіть у віддалених населених пунктах.

Також розвиток інтернет-торгівлі суттєво трансформує поведінкові моделі споживачів. Покупці стали більш вимогливими що до доставці, так і до систем електронної комерції. Впровадженням зручного UI, функціоналу, та безпеки персональних даних стали одними із важливіших речей на подібних ресурсах. Якщо раніше користувачі могли закрити очі на повільну роботу сайту, або не функціональний і незручний інтерфейс, з урахуванням конкуренції, розробники

подібних ресурсів повинні: аналізувати відгуки, вдосконалювати функціонал та впроваджувати нові технології.

Онлайн-торгівля подарувала можливість всім спробувати себе в цій сфері. Кожен охочий користувач має змогу започаткувати власну справу, або вийти на новий рівень вже працюючим. Це дозволяє здійснювати продажі за межами регіону проживання, продавати товари та послуги по всій країні, або вийти на міжнародний рівень. Без фізичних точок, достатньо налаштувати сторінку під інтереси в певному регіоні, або країни. Користувач без досвіду у розробці в створенні та програмування веб-додатків, за допомогою існуючих інструментів та ШІ, може зробити свій магазин, купивши рекламу в популярних соціальних мережах або каналів.

Таким чином, онлайн-торгівля перетворилась на важливий елемент економічного розвитку. Її роль у сучасному житті постійно зростає, формуючі нові вимоги до цифрових ресурсів, що, у свою чергу, змінює підхід до споживання інформації та товарів. Це впливає на попит до технологій, змушуючи постійно вдосконалюватися та робити персональні добірки для кожного користувача. Розширення інтернет-зв’язку та акцентування на цифровому світі, сильно підвищує попит в інтернет ресурсах. Саме тому розробка власного веб-додатку онлайн-торгівлі є актуальним і перспективним напрямом діяльності.

1.2 Аналіз вимог до сучасних веб-додатків онлайн-торгівлі

Онлайн-торгівля є один із перспективніших напрямів цифрової економіки. З кожним роком усе більше компаній і приватних осіб створюють особисті веб додатки як основний або додатковий канал продажів. Це обумовлене зростаючим попитом на зручні, доступні та ефективні засоби купівлі й продажу товарів і послуг через інтернет. Водночас, із розвитком технологій та цифрових сервісів зростають і очікування щодо додатків.

Сучасні веб-додатки мають відповідати великому списку сучасних вимог. Визначення цих вимог є обов’язковим етапом при розробці власного програмного

забезпечення, саме вони формують відповідність очікуванням кінцевих користувачів.

Вимоги до сучасних комерційних веб-додатків поділяються на функціональні та нефункціональні, кожна з яких грає важливу роль у процесі розробки додатків.

Функціональні вимоги – це формалізовані положення, які визначають необхідні функції, поведінкові характеристики то варіанти взаємодії програмної системи з користувачем. Ця вимога описує, що саме веб-додаток повинен реалізовувати для забезпечення виконання своєї основної цілі і завдання, є базовим орієнтиром у розробці та тестуванні продукту.

Основні аспекти сучасних функціональних вимог до торгових веб-додатків включають:

1. Взаємодія з користувачем – опис та реалізація логіки сторінки, взаємодія кінцевого користувача з елементами інтерфейсу, включає форми, кнопки, меню, тощо. Важливо передбачити всі можливі взаємодії зі сторінкою для забезпечення високої функціональності.
2. Реєстрація та авторизація користувачів – забезпечення безпечного механізму реєстрації нових користувачів, для ідентифікації, із використанням номеру телефону та електронної адреси.
3. Управління та взаємодія з оголошеннями – можливість створити, передивитися оголошення. Користувачі повинні мати змогу додавати опис, ціну, назву, фото тощо.
4. Пошук – можливість не передивлятися кожний товар окремо, а вписати назву чи вид товару для фільтрації від всього до зацікавленого
5. Особистий кабінет користувача – власна сторінка, яка надає інформацію про користувача, дає змогу переглядати всі оголошення створенні зареєстрованим користувачем, або редагувати особисті дані.
6. Адміністративна панель керування – сторінка яка має обмежений доступ, для керуванням зареєстрованими користувачами.

Для окремих додатків список функціональних вимог буде відрізнятися, він формується на плануванні проєкту: проводиться аналіз ідеї, аналіз чинних аналогів, ознайомлення з відгуками та спілкування з активними споживачами подібних сервісів. Формується список, та в ході розробці він доповнюється чи редагується. Кожна вимога це додаткова складність до системи, яка впливає на час та ціну розробки, або підтримки.

Зі збільшенням кількості функціональних вимог у межах проєкту, істотно збільшуються кількість рядків коду. Внаслідок зростання обсягу програми підвищується складність системи, що, у свою чергу, спонукає збільшенню кількості точок відмови.

Додаткові функціональні вимоги ускладнюють процес тестування продукту. Із збільшенням функціоналу збільшується і кількість сценаріїв для перевірки, що зумовлює ускладнення як автоматизованого, так і ручного тестування. Унаслідок цього темпи розробки знижуються, підвищуються витрати і переноситься дата презентації проєкту.

Підтримка складної системи передбачає наявність великої документації, що впливає на передачу частин проєкту серед команди. Новим учасникам стає складніше адаптуватися до архітектури системи, це впливає на час з ознайомлення та входження у розробку. Крім того, великі проєкти демонструють нижчу гнучкість при переході на нові технологічні рішення, оскільки оновлення пов’язанні з більшими витратами часу та ресурсів.

Частина нових вимог у великих проєктах буде взаємодіяти вже в реалізованими, що може вплинути на всю роботу програми. Поява нових, не передбачених вимог, може потребувати повного або часткового переписування додатку, для коректної інтеграції та взаємодії.

Отже, велика кількість функціональних вимог прямо пов’язана зі складністю їх підтримки, що вимагає додаткового детальнішого планування архітектури.

Нефункціональні вимоги – Сукупність вимог що стосується якості програмного забезпечення, задають умови, обмеження та стандарти які визнають, як система повинна працювати. Вони не стосуються реалізації конкретних функцій,

проте мають критерії, як та в яких умовах система повинна виконувати свої функції.

Основні аспекти сучасних нефункціональних вимог до торгових веб-додатків включають:

1. Безпека – Передбачає захист даних, конфіденційність авторизації тощо. Необхідність застосовувати сучасні методи захисту від атак, шифрування паролі. Також відстежувати підозрілу активність.
2. Зручність використання – Додаток повинен мати інтуїтивно зрозумілий UI/UX інтерфейс. У користувача не повинно виникати труднощів щодо пошуку дії. Інтерфейс має бути логічно структурованим, а дії користувачів – послідовними.
3. Масштабованість – Архітектура передбачає можливість розширення без втрат стабільності.
4. Адаптивність – Передбачає коректне відображення інтерфейсу на всі можливі види пристроїв.

Таким чином, нефункціональні вимоги є важливими для створення працездатного та якісного веб-додатку. Вони не менш важливі, ніж функціональні вимоги, оскільки без їх наявності додаток може втратити конкурентоспроможність.

Аналіз вимог до сучасних веб-додатків онлайн-торгівлі дозволяє сформулювати чітке бачення майбутньої архітектури власного додатку. Реалізація функціональних та нефункціональних вимог, описаних у цьому підпункті, є ключовою умовою для забезпечення зручності, ефективності та безпеки веб-додатку на сучасному ринку цифрових рішень.

1.3 Порівняльний аналіз популярних платформ електронної комерції

Для побудови свого ефективного веб-додатку недостатньо знати лиш спільні вимоги до системи – важливо враховувати та опиратися на аналіз вже реалізованих рішень. Порівняння популярних аналогічних платформ дозволяє визначити сильні та слабкі сторони додатків.

У цьому підпункті буде здійснено порівняльний огляд популярних веб-платформ, які працюють на ринку онлайн-торгівлі. Основна мента аналізу – визначити структуру, функціональні можливості та особливості реалізації цих сервісів, які можуть бути адаптовані або вдосконаленні у власному проєкті.

Такий аналіз дає змогу:

- Відокремити елементи, які стали стандартом у подібних проєктах;
- Уникати повторення поширених помилок;
- Сформувати власну структуру, функціонал майбутнього продукту;

OLX – це платформа для розміщення оголошень, яка є однією із найпопулярніших в Україні серед користувачів, для обміну товарів та послуг. Після проходження процедури реєстрації, користувачі можуть створювати власні оголошення. Основна ідея платформи – забезпечити просту, швидку, то доступну модель C2C-торгівлі, що стала причиною її популярністю.

Ключові функціональні можливості:

- Створення оголошень без проходження довгої реєстрації;
- Сортування оголошень по категоріям;
- Внутрішній чат між користувачами;
- Переглядання профілю користувачів;

На рисунку 1.1, продемонстрована головна сторінка платформи яка пропонує простий і водночас зручний інтерфейс, орієнтований на швидкий вибір зацікавленої категорії або більш детальний пошук оголошення.

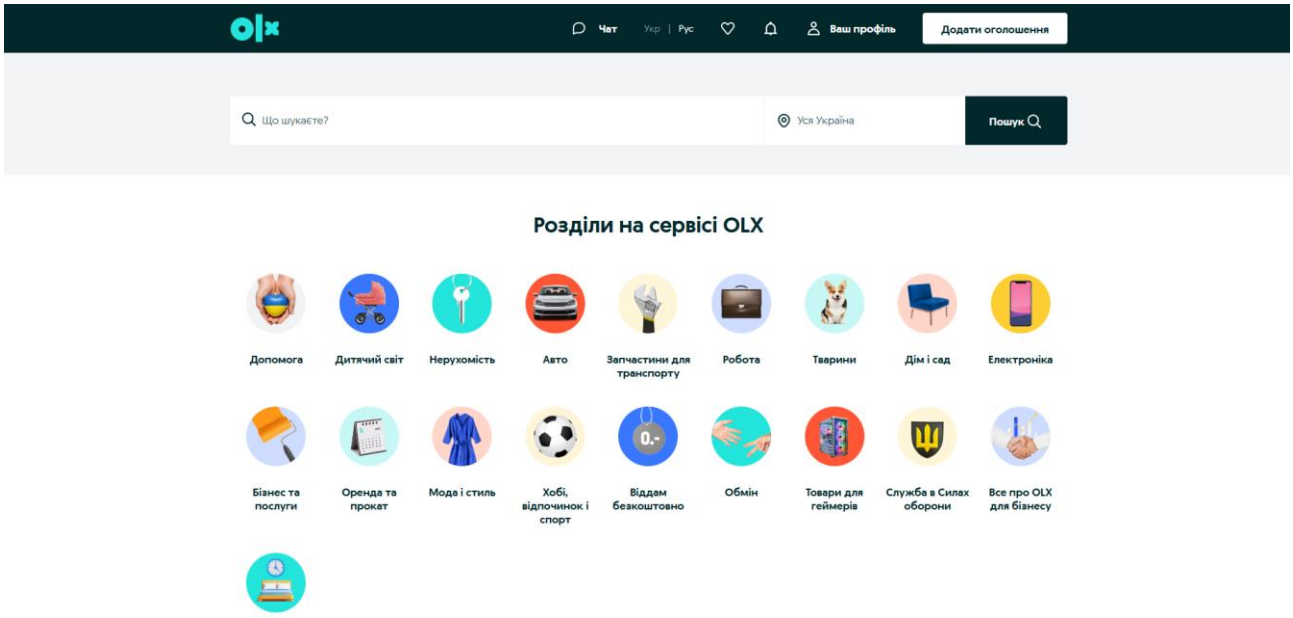


Рисунок 1.1 – Аналіз головної сторінки OLX

Попри візуальну простоту та структурованість, побудова сторінки головної сторінки передбачає першочергово навігацію по категоріям, що займають центральну частину екрану. Живі оголошення розташовані значно нижче та стають доступними користувачу лише після додаткових дій. Запропонована структура є зручною для цілеспрямованого пошуку, однак не враховує поведінкові моделі користувачів, які заходять на платформу без чіткого сформованого запиту.

Згідно з сучасними принципами UX-дизайну, розміщення динамічного та оновлюваного контенту у верхній частині сторінки, доступної для першочергового сприйняття, сприяє залученню користувачів до активної взаємодії з веб ресурсом при першому його відкритті. Це дозволяє реалізувати ефект несподіваного зацікавлення, коли користувач побачивши актуальні пропозиції, починає переглядати більше товарів, навіть не маючи початкового наміру робити покупки. Такий підхід затримує користувача на платформі та залишає позитивний досвід використання, який є важливим для веб-платформ. Також, регулярна поява нових оголошень показує активність на платформі, що викликає довіру та бажання повторного використання.

Окремої уваги заслуговує блок VIP оголошень, які займають частину головної сторінки. Саме вони відображаються при першому відкритті сторінки. Це створює умови, коли звичайні (безкоштовні) оголошення опиняються “в тіні”

платного контенту, що обмежує можливості нових користувачів органічно просувати свої оголошення. Така модель побудови контенту є комерційно привабливою для платформи, однак знижує відкритість системи, що може вплинути на довіру з боку користувачів. Також це впливає на залучення нових користувачів, які очікують рівні умови торгівлі. Демонстрація наведена на рисунку 1.2.

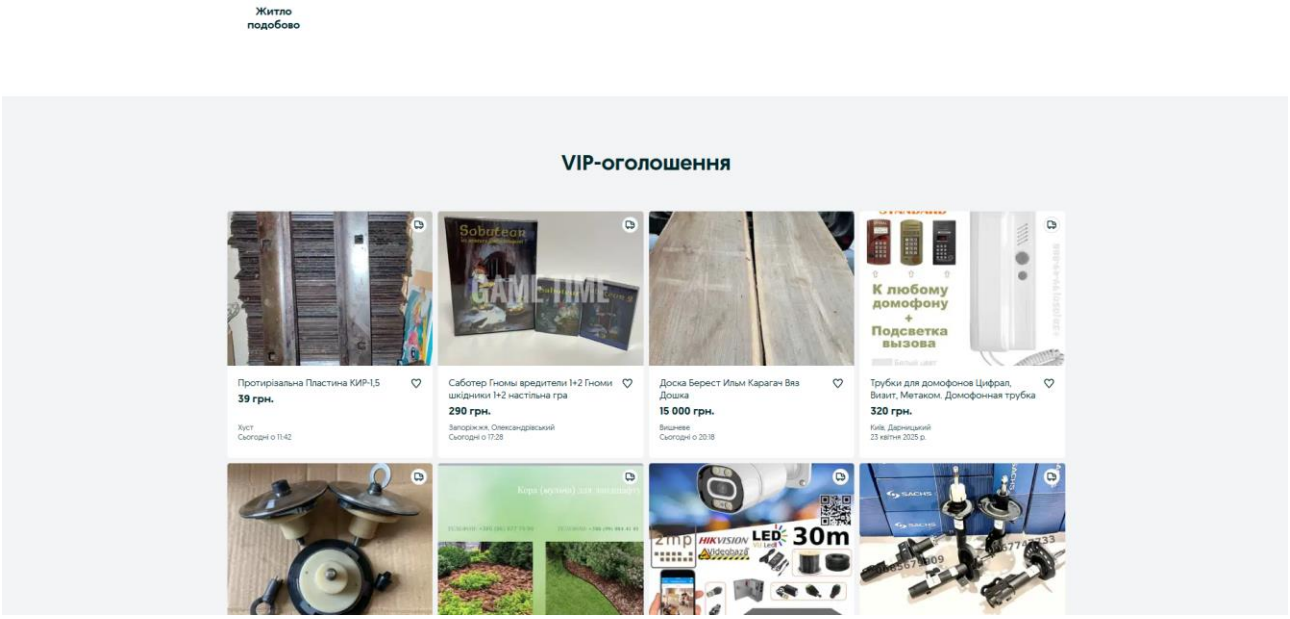


Рисунок 1.2 – Аналіз блоку оголошень сторінки OLX

Ще однією важливою деталлю, є візуальне відображення авторизації користувача, яка потребує вдосконалення. У поточному дизайні OLX статус входу на платформу не позначається чітко — відсутнє будь-яке візуальне відображення, які б вказували, що користувач перебуває в авторизованому стані. Єдиною змінною при авторизації – це поява стрілки біля “Ваш профіль”, яка відкриває меню взаємодії. Іконка або фото профілю, які активно використовуються у сучасних системах, проте тут відсутні.

Така реалізація знижує інтуїтивність навігації та взаємодії. Оптимальним рішенням було б впровадити візуальний індикатор авторизації, наприклад, аватар або кольорове підсвічування елементу, що чітко сигналізує про активний профіль. Демонстрація реалізації зображено на рисунках 1.3 та 1.4.

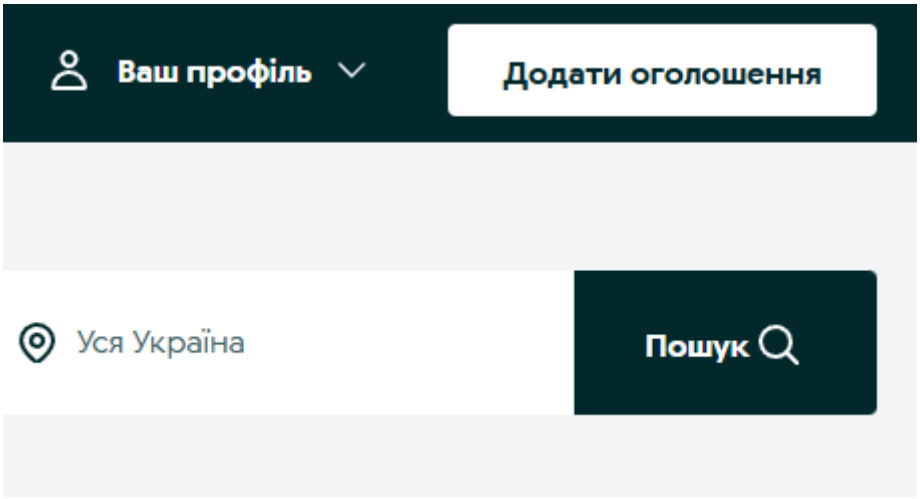


Рисунок 1.3 – Демонстрація авторизованного користувача на платформі OLX

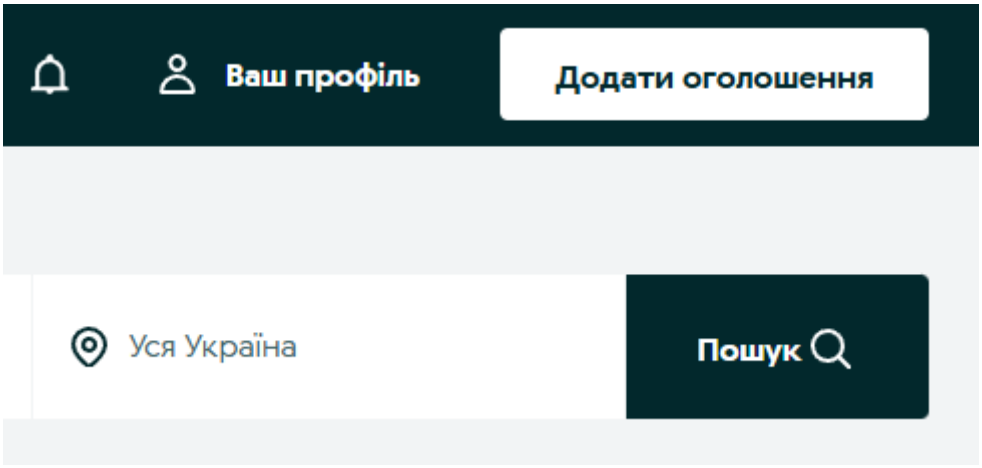


Рисунок 1.4 – Демонстрація не авторизованного користувача на платформі OLX

Сайт дозволяє розміщувати оголошення, без додаткових перевірок, що призводить до появи сумнівних, дублікатів або фейкових оголошень. Це може викликати питання щодо якості адміністрування на платформі.

Але не зважаючи на всі недоліки, вона має низку переваг. Зокрема, OLX, має широку аудиторію, що свідчить про високий рівень довіри до платформи, таким чином приваблює нових користувачів. Крім того, можливо розміщувати оголошення без проходження складної реєстрації.

Переваги платформи:

- Швидке створення оголошень;

- Велика аудиторія;
- Мобільна адаптація додатку;
- Наявність рейтингу продавців;
- Наявність чату;

Недоліки платформи:

- Акцент на категоріях
- Відсутність оголошень у фокусі головної сторінки
- Домінація VIP-оголошень на головній сторінці
- Недостатня модерація контенту

Платформа OLX є добрим прикладом зручного та функціонального веб додатку, орієнтованого на швидку взаємодію між користувачами. Її головні переваги полягають у простоті використання, великій аудиторії та доступності. Проте аналіз інтерфейсу та функціональності виявив обмеження, які можуть негативно вплинути на емоційний стан після використання платформи.

Таким чином, платформа залишається провідною у своєї ідеї, але потребує у змінах UX-механік, якщо прагне отримати максимально позитивний досвід і відгуків від користувачів.

Prom – Також один із найбільших платформ електронної комерції в Україні. Платформа почала працювати у 2008 році як інструмент просування малого та середнього бізнесу. На відмінну від OLX, Prom – реалізує B2C модель торгівлі, даючи можливість компаніям надавати послуги чи реалізовувати продукцію одразу кінцевим споживачам.

Платформа надає можливості: від публікації товару до обробки замовлення, оплати та доставки. Основної аудиторією продажу товарів є офіційні компанії та інтернет магазини, однак доступна і для фізичних осіб.

Ключові функціональні можливості:

- Керування асортиментом, цінами та знижками.
- Рейтинг продавців та відгуки.

- Підтримка онлайн-оплати через спеціальні сервіси.
- Інтеграція з сервісами доставки.

Платформа має досить приємний інтерфейс. Ліворуч розташована вертикальне та мінімалістичне меню категорій, що забезпечує більше місця для іншої важливої інформації. Платформа, на головній сторінці, має всі необхідні кнопки для навігації, вони добре виділенні та не зливаються з іншими елементами інтерфейсу. Посередині встановлений великий рекламний банер, а нижче найпопулярніші категорії, та персональна добірка товарів. Демонстрації інтерфейсу зображені на рисунках 1.5 та 1.6.

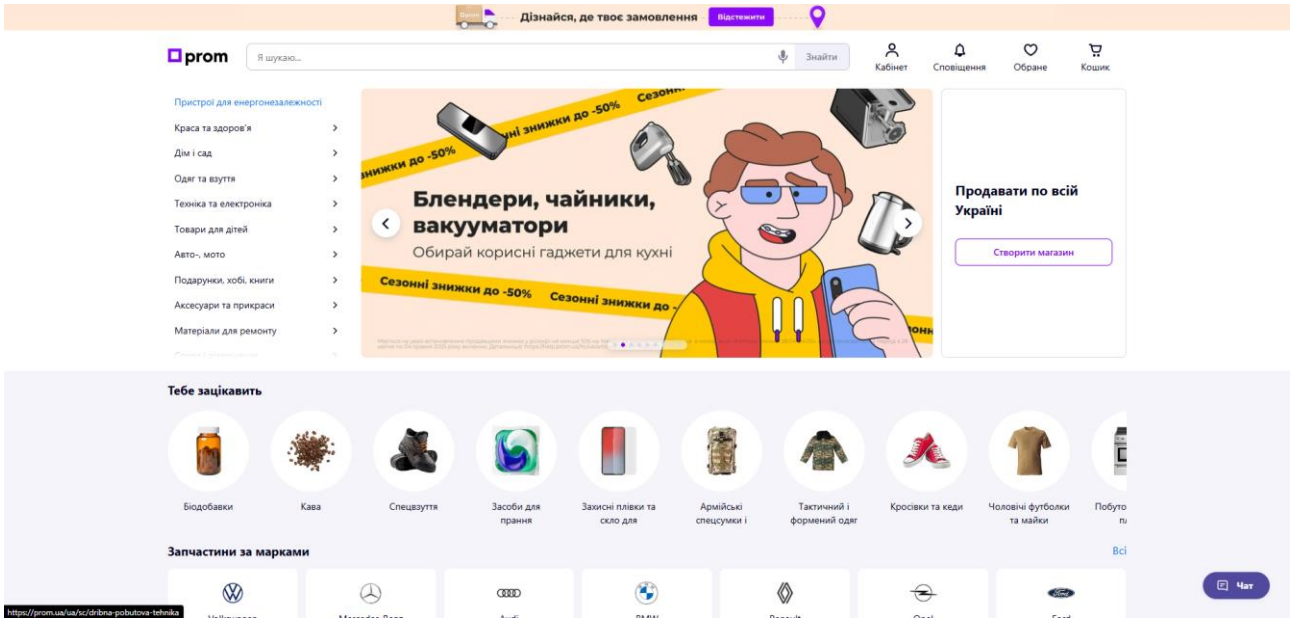


Рисунок 1.5 – Аналіз головної сторінки Prom

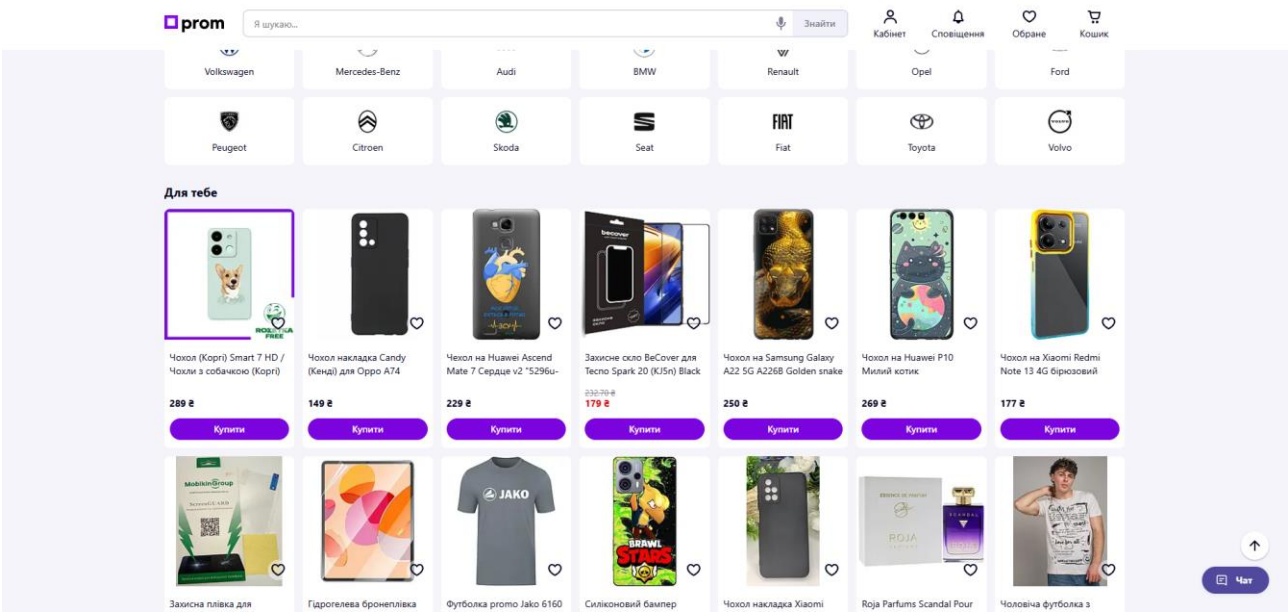


Рисунок 1.6 – Аналіз головної сторінки Prom

Незважаючи на високу функціональність і непогано продуману структуру, інтерфейс платформи може здаватися перевантаженим, особливо для нових користувачів. Велика кількість однотипних блоків, кнопок, рекламних елементів створює відчуття інформаційного перенасичення. Це все впливає на сприйняття контенту та ускладненням інтуїтивної навігації.

PROM також не має доступу до актуального списку товарів при першому відкритті сторінки, що вимагає користувача робити додаткові дії, але на відміну від OLX, ця платформа має розташування рекомендованих товарів під кожного користувача особисто. Сервіс аналізує дії, категорію та вид переглянутих товарів, аналізує Cookie файли, що відкриває можливість робити добірки більш точними до інтересів конкретного користувача. Також сервіс не рекламує “VIP” оголошення товарів, що робить конкуренцію на платформі більш справедливою для компаній та простих продавців, що позитивно впливає на довіру та емоційний стан після використання платформи.

Переваги платформи:

- Індивідуальна добірка товарів.
- Відсутність VIP оголошень.
- Підтримка оплати на сайті.

- Великі функціональні можливості.

Недоліки платформи:

- Перевантажений інтерфейс.
- Складність навігації.
- Комерційна спрямованість.

Було розглянуто особливості сервісу Prom, розглянуто особливості архітектури, функціональності та UI/UX рішень. Prom пропонує розширенні можливості для професійного продажу, але має складніший інтерфейс та підвищений вхідний поріг для новачків.

Результатами порівнянь та аналізу дозволяють сформулювати чітке розуміння що очікує кінцевий користувач від подібних платформ. Ці аспекти будуть враховані під час розробки власного веб-додатку, що поєднає найкращі сторони обох платформ і модернізацію їх проблем.

1.4 Постановка завдання та цілі розробки

У межах цієї дипломної роботи буде створений веб-додаток електронної комерції, на основі C2C моделі, який забезпечує можливість взаємодії користувачів без посередників. Основна мета роботи є створення зручної та функціональної платформи для продажу/обміну товарів, безпосередньо, надавши зв’язок продавця та зацікавленого лиця.

Розробка такого ресурсу відкриє ширший потенціал для неформальної торгівлі в інтернеті.

Після аналізу аналогічних платформ та вимог до інтернет ресурсів, актуальність теми залишається високою по наступним причинам:

1) Аналогічні платформи мають обмеження, як OLX з пріоритетним проплаченими оголошеннями, як зазначено в підпункті 1.3.1. Відсутність чесної конкуренції може створити погані враження користувачів, що сильно впливає на популярність до платформи.

2) Відсутність потреби в оренді приміщення доводить актуальність онлайн платформ та дозволяє скоротити витрати.
3) Швидкість та зручність. Платформи адаптуються під сучасні умови, що сприяє підвищенню залученості користувачів до платформи.
4) Розвинена сфера доставок. Швидка відправка та отримання товарів позитивно впливає на рівень довіри до інтернет ресурсів.
5) Зростання популярності в C2C-моделі торгівлі.
6) Перевантажений інтерфейс аналогічних платформ ускладнює взаємодію з сервісом новими користувачами.
7) Підвищений запит на подібні платформи.

З метою підвищення користувальницького досвіту та зручності користування сервісом, був проведений аналіз потреб та вимог. В межах реалізації були сплановані наступні задачі:

- Створення зручного інтерфейсу, для забезпечення легкої та інтуїтивної навігації користувачів.
- Реалізація меню навігації, де будуть розміщатися поле пошуку, іконка користувача, кнопка для створення товару та виходу з акаунту.
- Створення та видалення оголошень.
- Редагування оголошень.
- Адмін сторінка, для блокування та керування користувачами.
- Розробка та підтримка особистої бази даних.
- Система пошуку товарів.
- Сортування товарів на головній сторінці.
- Процес авторизація та реєстрація користувачів.
- Сторінка особистого профілю, яка передбачає перегляд всіх товарів продавця.
- Редагування особистого профілю.

Важливо підібрати правильні кольорову гамму для дизайну, оскільки кольорова палітра впливає на емоційний сприйняття платформи, формує перше враження про бренд, підвищує уважність до деталей платформи. UI/UX дизайн розроблений з використанням дух відтінків – синього та білого. Поєднання цих кольорів є результативними завдяки естетичній привабливості. Білий колір виступає нейтральним, він сприяє зосередженню та підсилює увагу на ключових елементах інтерфейсу. Синій виступає звичним та спокійним кольором, він асоціюється з надійністю та забезпечує добрий візуальний контраст. Кнопки видалення товару та виходу з акаунту будуть зафарбовані в білий, але виділені червоним. Це характерно для більшості сайтів.

Реалізація особистої сторінки профілю є ключовим елементом для побудови надійної та довіреної платформи за C2C моделлю. Така сторінка забезпечує можливість отримати важливу та повну інформацію про продавця. Це сприяє підвищенню довіри покупців, оскільки дозволяє ознайомитися з продавцем на рівні наданої інформації та прийняти зважене рішення. Також наявність особистої сторінки спрощує навігацію і спрощує процес ознайомлення з усіма товарами конкретної людини. Наявність сторінки особистого профілю сприяє на ефективність управління власними даними, що в результаті позитивно впливає на задоволеність користувача.

Функціонал редагування особистого профілю є невід’ємною частиною побідних веб-додатків. Його наявність забезпечує швидке оновлення персональних даних. Це підвищує рівень довіри, оскільки користувачі отримують можливість самостійно вирішувати як оформлювати власну сторінку. Також це відповідає сучасним вимогам про захист персональної інформації. Оновлення аватару профіля також впливає на загальний користувацький досвід та забезпечує позитивне враження. Завантаження та оновлення власного фото дозволяє ідентифікувати себе в межах платформи, це підвищує рівень залученості. Таким чином, підтримка цього функціоналу є важливою умовою забезпечення безпеки та комфортності використання додатку.

Реалізація адміністративної сторінки є важливим аспектом для повноцінного контролю над системою. Вона забезпечує доступ до головних функцій керування платформою. Її головне призначення – отримати швидку важливу інформацію про всіх користувачів, перегляд статусу акаунту та наявної ролі, зміна ролі, керування статусом облікового запису. При видачі блокування, не відображаються оголошення користувача, але вони продовжують зберігатися в БД, що дає змогу зберегти всю інформацію при помилковому блокуванні. Користувач має змогу особисто видалити оголошення якщо актуальність втрачено. Наявність адміністративної частини системи дозволяє здійснювати модерацію платформи без втручання у базу даних або використання зовнішніх інструментів. Це забезпечує більшу безпеку, гнучкість та спрощує супровід веб-застосунку.

Функціонал авторизації та реєстрації є фундаментальним для будь якого веб-додатку. Реєстрація дозволяє створити унікальний обліковий запис, який є основою для ідентифікації користувачів в системі та забезпечує доступ до індивідуального функціоналу. Авторизація дає гарантію, що доступ до особистої інформації отримує лише її власник, що забезпечує основний рівень безпеки. Тому функціонал авторизації та реєстрації є необхідним для побудови захищеної системи.

Таким чином, перераховані вимоги та дизайнерські рішення орієнтовані на потреби кінцевих користувачів.

2 ПРОЄКТУВАННЯ СИСТЕМИ ТА ВИБІР ТЕХНОЛОГІЙ

2.1 Порівняльний вибір мови програмування

На сьогоднішній день існує значна кількість мов програмування, які використовуються для розробки веб додатків, кожна з яких має свої переварі та недоліки, які ми розглянемо в рамках цього підпункту. Вибір мови програмування залежить від складності проєкту. Будуть аналізуватись вимоги до додатку, формуватися список переваг та недоліків.

Список мов програмування для реалізації веб-додатку:

1. Java.
2. Python
3. Java Script
4. PHP

1) Java – це об’єктно орієнтована мова програмування, яка використовується для створення масштабних веб додатків, яка має сильну надійність. Програми на Java перетворюються в байт-код, це дає змогу додатків написаних на Java працювати на багатьох платформах без необхідності змінювати код. Але через її надійність зростає потреба в оперативній пам’яті, вона її використовує значно більше, що впливає на час запуску програми та збільшенню ресурсів для її роботи. Java має строгий синтаксис, що сильно впливає на поріг входу новачків, які вивчають основи мови.

Мова виділяється наступними перевагами:

- Безпека.
- Незалежність від платформи.
- Масштабованість.
- Багатопоточність.

- Широка спільнота.

Недоліками виділяється:

- Вища складність.
- Ресурсоємність.
- Швидкість розробки.

Незважаючи на недоліки, Java забезпечує стабільне та надійне виконання коду, має підтримку багатопоточності та інтеграцію з сучасними і потужними фреймворками. Також через її надійність та безпеку, більшість великих економічних компанії продовжують використовувати цю мову для роботи своїх систем, незважаючи на підвищену потребу в ресурсах.

2) Python – Це високорівнева мова програмування. Вона має простий синтаксис та активно використовується в веб-розробці. Мова орієнтована на оптимізацію коду та підвищенню легшого сприйняття розробником, за рахунок чого має нижчий поріг входу для новачків. Мова була створена у 1989 році і продовжує активно підтримуватись, продовжують з’являтись нові бібліотеки і фреймворки для розширення сфер використання цієї мови. Вона вже активно використовується у веб-розробці, написання ШІ, аналітиці даних, автоматизації, кібербезпеці.

Мова є універсальним інструментом для багатьох сфер програмування. Потужним інструментом її робить кількість бібліотек та фреймворків, що дає змогу не писати програми з нуля. Має високу кросплатформеність, що дозволяє запускати Python програми на більшості ОС, та навіть у браузерах.

Python має низьку продуктивність, що може призвести до затримок виконання коду та має підвищену потребу у ресурсах системи. Це може бути критичним на слабких чи середніх системах. Мова має погану інтеграцію в мобільній розробці, що значно обмежує її універсальність.

Вона підходить для розробки невеликих проєктів, у той час великі проєкти можуть стикатися з проблемами вдосконалення чи переписування коду. Також через

менш строгу типізацію та правил структурування, це може негативно вплинути на безпеку додатку.

Переваги мови Python:

- Простий і читабельний синтаксис.
- Низький поріг входу.
- Велика кількість бібліотек і фреймворків.
- Широке використання.
- Підтримка ООП та функціонального програмування

Недоліки Python:

- Слабка продуктивність.
- Висока потреба в ресурсах.
- Обмежена багатопоточність.
- Обмежена безпека.

Python розрахований на прості або середні додатки, зі збільшенням навантаженням або кількістю запитів, система може генерувати помилки або збільшувати час виконання кожного з запитів. Але є зручною мовою для вивчення і розуміння основ програмування. Великою перевагою є простий синтаксис, що дозволяє новим розробникам швидше адаптуватися у проєкт, та підвищує читабельність коду.

3) Java Script – це одна з найпопулярніших мов програмування для веб-розробки, створена у 1995 році. Мова підтримує об’єктно орієнтовне, функціональне та подієво-орієнтовне програмування, має динамічні типізацію, та близько взаємодіє з HTML та CSS, за рахунок чого вона стає гнучкішою ніж попередні мови програмування. Має багато фреймворків, які частіше всього використовуються у веб-розробці.

Головною перевагою мови, те що вона виконується безпосередньо в браузері, це дає змогу створювати анімовані інтерфейси без використання компіляторів. Java

Script виконується як на фроненді, так і на бекенді. Це дає змогу використовувати лише одну мову, що сприяє процесу розробки та зменшенню витрат.

Але Java Script не має суворої типізації. Змінна може змінити свій тип при виконанні програми, якщо розробник не врахував це та призвести до масштабного збою. Це ускладнює контроль логіки та підтримку системи, де помилки помічаються виключно при виконанні програми.

Також він має дуже низький рівень безпеки коду, оскільки виконується безпосередньо в браузері, це дозволяє легко отримати код сторінки. Це небезпечно тим що небажана особо може вивчити логіку сторінки, та знайти недоліки якими може зловживати вразливостями системи.

Переваги Java Script:

- Підтримка багатьма сучасними браузерами.
- Має розвинені і дуже популярні бібліотеки.
- Підтримка декілька видів програмування.
- Можливість відтворювати динамічні інтерфейси наживо.
- Широке використання.

Недоліки Java Script:

- Немає суворої типізації.
- Мала продуктивність на серверній частині.
- Відсутність багатопоточності.
- Можливість переглянути код в браузері.

Java Script одне із найпопулярніших рішень для веб-розробки. За рахунок великої аудиторії, вона має багато бібліотек що постійно оновлюються, але одночасно це є і недоліком тому, що це призводить до нестабільності системи. Має слабку продуктивність на серверній частині, та проблеми з безпекою.

4) PHP – Мова програмування орієнтована на розробку серверної частини веб-додатку. Створена у 1994 році і по сьогодні використовується для створення сайтів. Мова є простою для вивчення, що робить поріг входу низьким. Вона також має список бібліотек та фреймворків які досі активно підтримуються. Мова

орієнтована для побудови динамічних веб сайтів, здатна взаємодіяти з базами даних та реалізовувати основну бізнес логіку.

PHP має сильні недоліки в безпеці. Додатки можуть створюватись без врахування сучасних вимог, що дуже сильно сприяє на захищеність системи. Мова має застарілий синтаксис, який без використання сучасних фреймворків може ускладнити підтримку та оновлення проєктів.

Мова залишаться в списку лідерів, продовжує оновлюватись та забезпечується підтримкою бібліотек та фреймворків.

Переваги PHP:

- Простий синтаксис.
- Інтеграція з базами даних.
- Наявність готових рішень.
- Активна підтримка.

Недоліки PHP:

- Менша схильність до великих проєктів.
- Низький рівень безпеки.
- Відсутність суворої типізації.
- Застарілі структура коду.
- Втрачання популярності.

Але PHP залишається ефективним інструментом для створення маленьких проєктів та має низький рівень входу. Проте в порівнянні з конкурентами її популярність значно нижче.

У ході порівняння мов програмування для реалізації веб додатку було обрано мову Java. Незважаючи на підвищену складність та потребу в ресурсах, вона є оптимальним вибором для розробки комерційного веб-додатку. Крім того вона відповідає сучасним вимогам до розробки безпечних, надійних та масштабних проєктів, які були проаналізовані в розділі 1. Крім того, Java має велику екосистему та підтримує більшість сучасних фреймворків для стабільної роботи програми.

Детальне пояснення вибору стеку технологій на Java розглянуто в наступному підрозділі 2.2.

2.2 Опис вибраних технологій

Вибір мови програмування є одним із ключових рішень при розробці власної платформи. У межах цієї дипломної роботи, під час розробки, з урахуванням сучасних потреб та наявний досвід в програмуванні, було обрано перевірений стек технологій. До складу основних інструментів входять:

- Мова програмування Java;
- Фреймфорк Spring (boot, security, web , jpa);
- База даних Postgresql;
- Білдтул Maven;
- Freemarker;
- Bootstrap;
- Lombok;

Враховуючи результати розгляду мов програмування, які були описані в підпункті 2.1, було прийняте рішення використовувати Java для розробки власної платформи. Java є об’єктно-орієнтована мова, що є одним з найголовніших аспектів безпеки у розробці веб-додатків. Крім того, вона має підтримку фреймворків та бібліотек, таких як: Spring Boot, який суттєво спрощує розробку серверної частини. Зокрема – автоматизація процесу ручного написання шаблонного коду, замінюючи на використання анотацій, що суттєво підвищує ефективність та читабельність програми. Завдяки інструментом які працюють з Java, проєкт отримав гнучку архітектуру з високим рівнем оптимізації коду та підтримкою на майбутнє.

Фреймворк Spring Boot був обраний як основа бекенду через його стабільність, та масштабність з активними оновленнями. Він значно спрощує створення Java проєктів шляхом надавання базової конфігурації для більшості

параметрів, що звільняє від ручного налаштування інфраструктури. Фреймворк є одним з найпоширеніших в використанні при створенні веб-додатків на Java, це додатково демонструє його актуальність. Архітектура платформи передбачає RESTful стиль передачі інформації, до котрого ідеально підходить фреймворк Spring Boot.

Також Spring Boot має інші важливі модулі, такі як: Spring Security, Spring JPA, Spring Web. Це забезпечує надійний захист, зручну роботу з БД, та побудови маршрутизації запитів. Для Java веб-додатків – це головний, найперспективніший інструмент у розробці.

Наступним важливим кроком – вибір БД. Насамперед треба забезпечити надійну оброку SQL запитів, підтримку транзакцій і враховувати розширення системи у майбутньому. Було проведено порівняння сучасних і популярних СУБД, які наведені у таблиці 2.1.

Таблиця 2.1 – Порівняння СУБД

Критерій	MySQL	SQLite	PostgreSQL	MongoDB
Тип	Реляційна	Реляційна	Реляційна	NoSQL
Складність налаштування	Низька	Дуже низька	Висока	Низька
Підтримка транзакцій	Так	Так	Так	Часткове
ACID-сумісність	Частково	Частково	Повна	Обмежена
Розширюваність	Середня	Низька	Висока	Висока
Швидкість (прості запити)	Висока	Висока	Висока	Висока
Швидкість (складні запити)	Середня	Низька	Висока	Низька
Масштабованість	Висока	Обмежена	Висока	Дуже висока

PostgreSQL – є найбільш оптимальним вибором для проєкту. Вона має високу продуктивність, швидко виконує складні запити, розширену підтримку транзакцій

та повну ACID-сумісність. Ці переваги особливо корисні для комерційних веб-додатків.

Іншим важливим інструментом для розробки платформи – білдтул (засіб збирання, керування та налаштування проєкту). Для реалізації проєкту було обрано Apache Maven. Maven є також одним із найпоширеніших у розробці Java-додатків. Він виділяється зручним керуванням залежностями через файл “pom.xlm”. Загалом “pom.xml” – основний конфігураційний файл, який описує проєкт, керуванням через цей файл відбувається за допомогою тегів.

Нижче наведено приклад підключення залежностей:

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
</dependencies>
```

Як ми бачимо, спочатку відкривається <dependencies>, після чого одразу розташовується елемент <dependency>, який описує одну бібліотеку. Кожна з яких має параметри, як groupId, artifactId, це дозволяє Maven одразу визначити джерело бібліотеки. Наведені в прикладі параметри є основними, які виконують роль ідентифікатора. Іншим обов’язковим тегом для підключення залежності є <version>. Він визначає версію бібліотеки що підключаються. Не визначивши, Maven автоматичне підтягне найновішу версію. Також є інші необов’язкові параметри.

Це робить роботу з Maven набагато легшою, звільнюючи від ручного завантаження та підключення .jar файлів в IDE.

2.3 Архітектурна структура додатку

Для зберігання даних, необхідних для повноцінної роботи веб-додатку, використовується реляційна БД PostgreSQL. У межах цього проєкту БД виконує функції основного сховища для інформації. Для взаємодії з БД використовується Spring Data JPA, яка значно спрощує роботу, дозволяючи взаємодіяти через Java класи. Для цього проєкту було створено 4 таблиці: users, products, images, users_role. Розбір кожної сутності наведено в таблицях 2.2, 2.3, 2.4, 2.5.

Таблиця 2.2 – Структура сутності Users

Поле	Тип	Not Null	Опис
Id	Bigint	TRUE	Ідентифікатор користувача
Active	Boolean	FALSE	Статус облікового запису
Date_of_registration	Timestamp	FALSE	Дата реєстрації користувача
Email	Varchar	TRUE	Електрона адреса
Name	Varchar	FALSE	Ім’я користувача
Password	Varchar	TRUE	Зашифрований пароль
Phone_number	Varchar	FALSE	Номер телефону
Image_id	Bigint	FALSE	Зовнішній ключ до аватару з таблиці Images

Таблиця 2.3 – Структура сутності Products

Поле	Тип	Not Null	Опис
Id	Bigint	TRUE	Ідентифікатор оголошення

Продовження таблиці 2.3

City	Varchar	FALSE	Місто, знаходження товару
Date_of_creation	Timestamp	FALSE	Дата створення оголошення
Description	Text	FALSE	Опис товару
Preview_image_id	Bigint	FALSE	Ідентифікатор прев'ю картинки з таблиці Images
Price	Integer	FALSE	Ціна товару
Title	Varchar	FALSE	Назва оголошення
User_id	Bigint	FALSE	Зовнішній ключ до творця оголошення з таблиці Users

Таблиця 2.4 – Структура сутності Images

Поле	Тип	Not Null	Опис
Id	Bigint	TRUE	Ідентифікатор зображення
Content_type	Varchar	FALSE	Тип файлу
Image_data	Bytea	FALSE	Зображення
Is_preview_images	Boolean	FALSE	Чи є зображення прев'ю
Name	Varchar	FALSE	Назва зображення
Original_file_name	Varchar	FALSE	Оригінальна назва файлу
Size	Bigint	FALSE	Розмір файлу

Продовження таблиці 2.4

Product_id	Bigint	FALSE	Зовнішній ключ до оголошення з таблиці Products

Таблиця 2.5 – Структура сутності User_role

Поле	Тип	Not Null	Опис
User_id	Bigint	TRUE	Зовнішній ключ до користувача з таблиці Users
Roles	Varchar	FALSE	Роль користувача

Таким чином, розроблена структура БД забезпечує необхідні умови для реалізації ключових функцій платформи. Усі таблиці пов’язані між собою зовнішніми ключами, це дозволяє зберігати зв’язність даних і підтримувати цілісність системи. На рисунку 2.1 зображено ER-діаграму, яка демонструє зв’язки між сутностями.

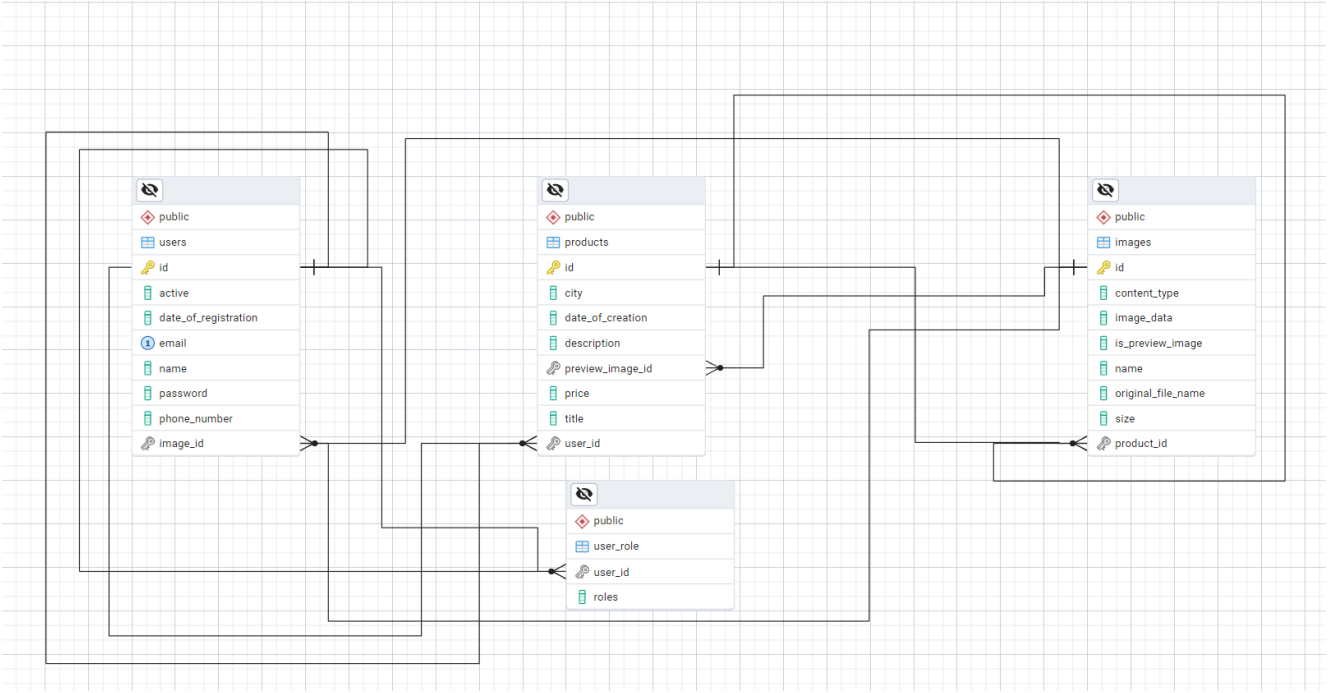


Рисунок 2.1 – ER-діаграма платформи “TradeShop”

Кожна таблиця має зв’язок одна з одною, це відкриває можливість уникнути дублювання полів. Таблиця Users має відношення одразу до трьох: User_role,

Images, Product. В таблиці User_role зберігається id користувача, та має друге поле role, яке зберігає роль користувача підв'язаного до цього id. Таким чином при реєстрації нового користувача, паралельно створюється сутність user_rule, що дає змогу редагувати роль тільки обраного користувача, без необхідності змінювати id ролі.

Таблиця Images зберігає одразу всі фотографії та картинки, маючи поле Is_preview_images одразу ідентифікує картинку показуючи що це прев'ю оголошення. Для забезпечення простоти системи, подібних полів ідентифікаторів більше не використовуються. Для зберігання всіх зображень оголошення, сутність має поле product_id, що підв'язує їх безпосередньо до id таблиці products.

Структура БД оптимізована під розроблені вимоги до веб-додатку онлайн торгівлі на моделі C2C. Забезпечує надійне зберігання інформації та швидке виконання SQL-запитів.

Наступний важливий крок – розробка та проєктування серверної частини. На цьому рівні виконується обробка вхідних даних. Важливо забезпечити надійний захист від спроб маніпулювання з БД через кінцевий інтерфейс. Для розробки було обрано архітектурну модель MVC (Model-View-Controller), яка є з найефективніших структур побудови веб-додатків. Модель чітко розподіляє відповідальності між окремими компонентами системи. У межах даного проєкту акцент розробки буде зроблений саме на цій частині, як було описано в підпункті 1.2 – сучасні вимоги потребують надійний захист обробки інформації на платформах.

Опис моделі:

- Model – Відповідає за бізнес-логіку та роботу з даними БД. Спеціальні класи мають анотації: @Entity, @Table(name = "products"). @Entity позначає, що клас є сутністю, представляю таблицю в БД. @Table(name = "products") вказує ім'я таблиці, за замовчуванням таблиця буде мати назву класу.

- View – Відповідає за відображення даних користувачеві. Формування сторінок виконується за допомогою шаблонізатора Freemarker, а для стилізації фреймворк Bootstrap. Цей компонент активно взаємодіє з контролерами через передачу даних у шаблони.
- Controller – Обробляє запити клієнтів, формує відповіді та передає у відповідному представленні. У проєкті реалізовано чотири контролера: AdminController, ImagesController, ProductController, UserController. Кожен з яких відповідає за свою частину функціоналу.

Повну структуру додатку зображено на рисунку 2.2, 2.3.

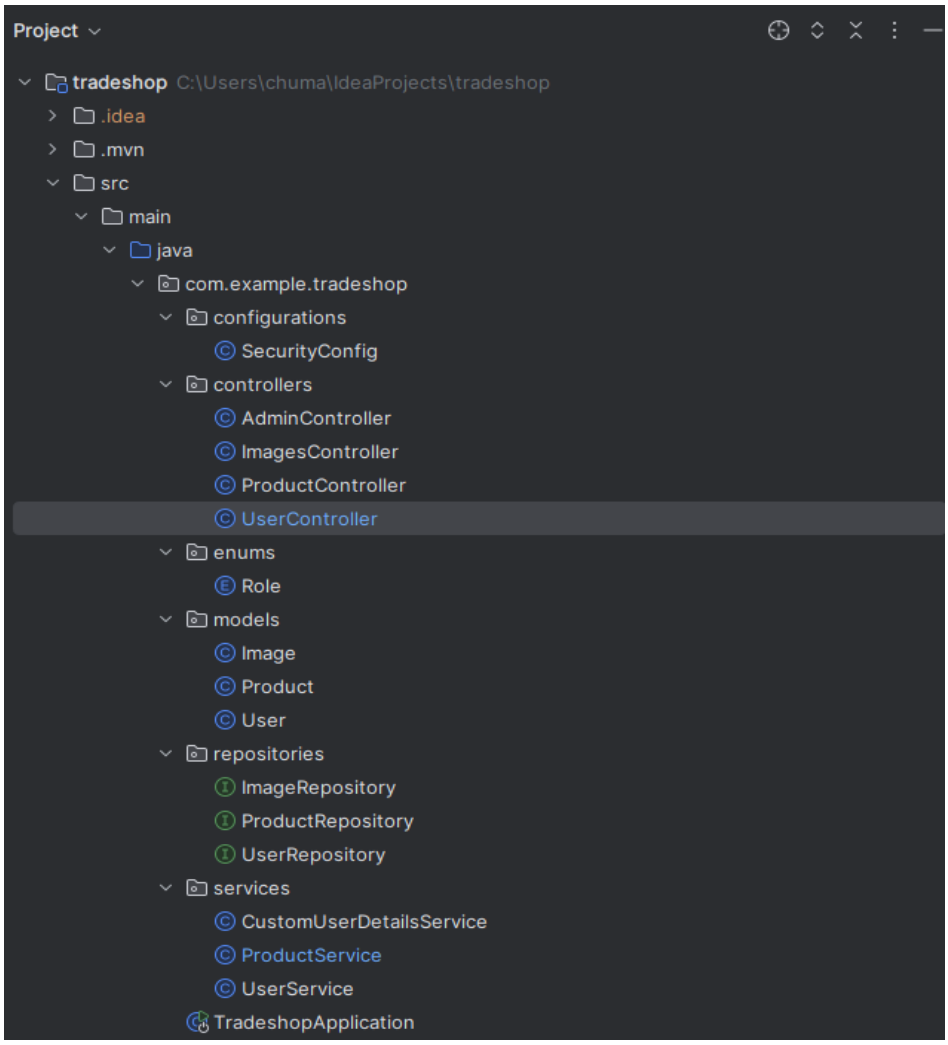


Рисунок 2.2 – Структура веб-додатку “TradeShop”

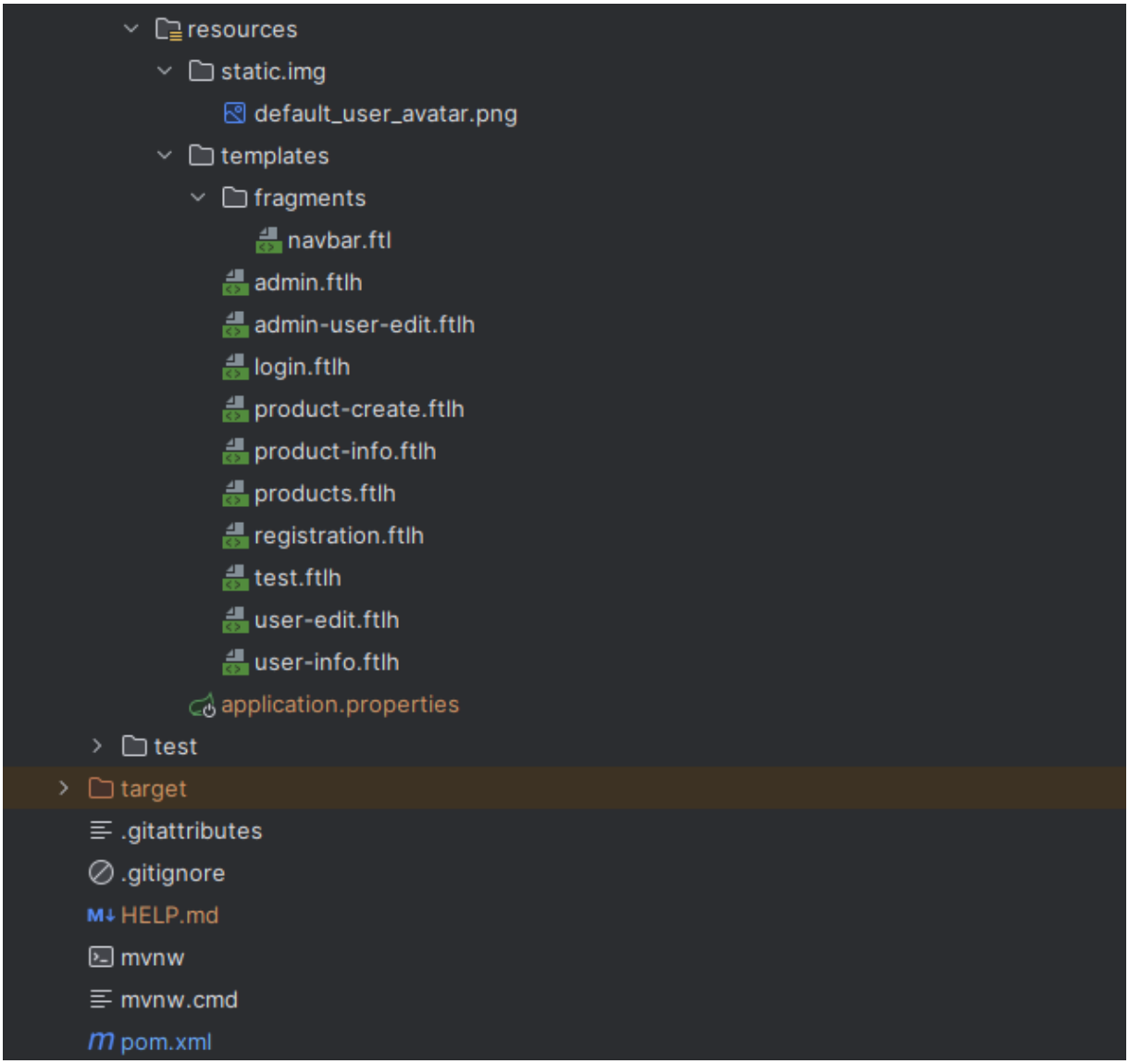


Рисунок 2.3 – Структура веб-додатку “TradeShop”

Таким чином, серверна частина побудована за надійною архітектурною моделлю MVC, кожен компонент знаходиться у відповідній директорії, що забезпечує зручну навігацію. Застосування сучасних інструментів дозволяє ефективно реалізувати обробку запитів, керування даними та забезпечити безпечну взаємодію з користувачами. Така структура відповідає сучасним вимогам та забезпечує стабільну основу для подальшого розвитку платформи.

3 РОЗРОБКА ВЕБ-ДОДАТКУ TradeShop НА МОВІ ПРОГРАМУВАННЯ Java З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ Spring Boot

У цьому розділі розглянуто процес практичної реалізації власного веб-додатку TradeShop, за принципом моделі C2C. Основна мета – створити функціональну платформу, враховуючи сучасні вимоги до захисту даних, яка забезпечує зручний спосіб обміну чи купівлі товарів між кінцевими користувачами.

Розробка платформи охоплює усі етапи створення веб-застосунку: від формування структури БД, до реалізації повнофункціонального інтерфейсу користувача. Веб додаток розроблявся на мові програмування Java, з використанням фреймворку Spring Boot, системи шаблонів Freemarker та реляційної БД PostgreSQL. Для стилізації інтерфейсу використано фреймворк Bootstrap, що дозволяє створювати адаптивні та зручні сторінки.

Архітектура додатку створена з дотримання принципів MVC, що дозволило відокремити різні частини додатку. Це забезпечило гнучкість у розробці та подальшій модернізації платформи.

3.1 Ініціалізація проєкту та підключення БД

Для швидкого створення проєкту, було обрано онлайн-сервіс Spring Initializr, який швидко генерує структуру Spring Boot-застосунку з обраними залежностями. Інструмент забезпечує автоматичне формування директорій та конфігураційного файлу pom.xml. Зображення формування базового проєкту продемонстровано на рисунку 3.1.

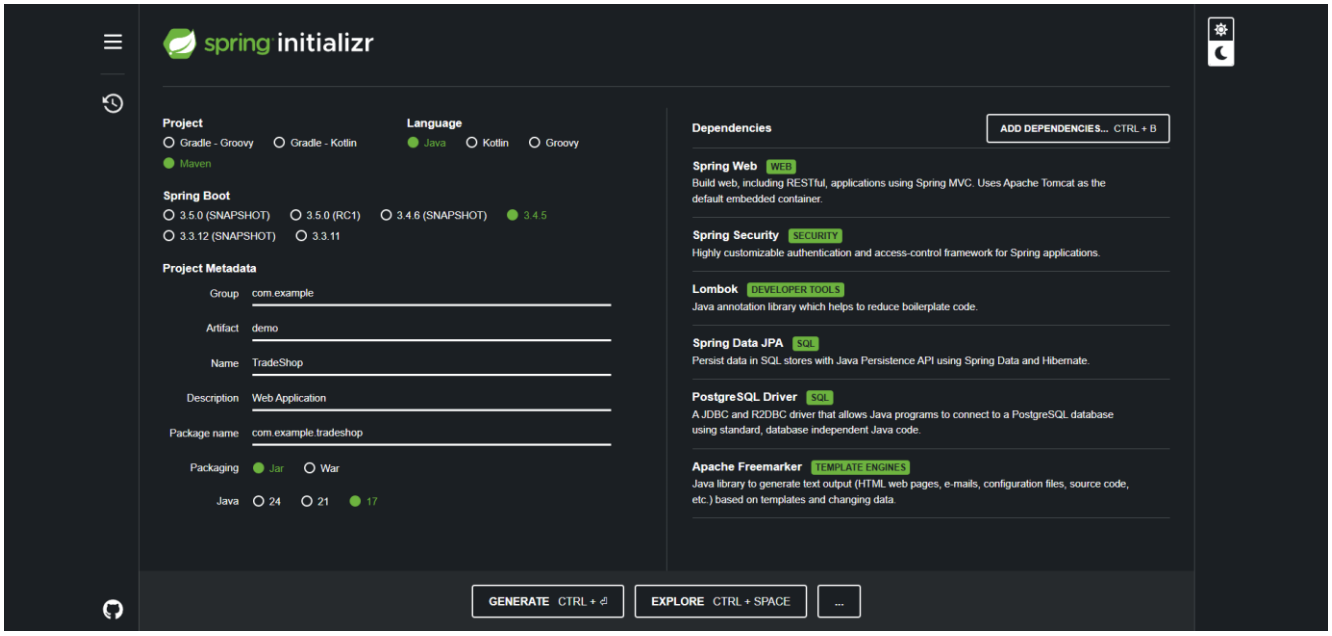


Рисунок 3.1 – Онлайн сервіс Spring initializr для формування Spring додатку

Було обрано 17 версію Java, вона є найпоширенішою для використання у проєктах. Саме ця версія активно використовується у розробці та має підтримку багатьох фреймворків. Для головного фреймворку Spring Boot, найактуальнішою версією є 3.4.5, яку буде замінено на 2.6.7.

При генерації проєкту, було додано низку базових залежностей:

1) Spring Web – має сервер Apache Tomcat, забезпечує підтримку REST-контролерів та обробку HTTP запитів;
2) Spring Security – модуль для впровадження автентифікації, авторизації та захист ресурсів;
3) Lombok – бібліотека, що автоматично генерує бойлер-код за рахунок анотацій;
4) Spring JPA– дозволяє працювати з БД за допомогою ORM (анотацій);
5) PosgreSQL Driver – драйвер для взаємодії з БД;
6) Apache Freemarker – шаблонізатор для формування HTML сторінок з використанням даних з серверу;

Для розробки було обрано професійне середовище розробки intelliJ IDEA, воно є найпотужнішим інструментом для Java розробки. Середовище підтримує повну інтеграцію обраних систем, та має вбудовану систему запуску додатків.

Після відкриття новоствореного проєкту, автоматично генерується конфігураційний файл pom.xml, який використовує булдтьул Maven. Навединим нижче фрагментом є частина стартової конфігурації додатку, згенерованого Spring Initializr та є стандартною частиною подібних додатків.

Базові конфігурації в pom.xml:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.6.7</version>
      <relativePath/>
   </parent>
   <groupId>com.example</groupId>
   <artifactId>tradeshop</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <name>TradeShop</name>
   <description>TradeShop project </description>
   <url/>
```

Ці налаштування є основою для роботи та визначення проєкту створеного за допомогою Maven. Вони регламентують ідентифікацію артефакту, визначають структуру та керують версіями та автоматизують процес компіляції, тестування та запуску проєкту. Без необхідності вміст pom.xml не рекомендується змінювати, особливо базові налаштування які наведені вище.

Для підключення до БД у Spring Boot є спеціальний файл конфігурації application.properties. У ньому задаються основні параметри підключення, вміст файлу відображено на рисунку 3.2.

```
spring.application.name=tradeshop
spring.datasource.url=jdbc:postgresql://localhost:5432/trade_shop
spring.datasource.username=postgres
spring.datasource.password=Password
spring.jpa.hibernate.ddl-auto=update
spring.datasource.hikari.auto-commit=false
spring.jpa.show-sql=true
spring.freemarker.expose-request-attributes=true
```

Рисунок 3.2 – налаштування в application.properties

В application.properties визначається ім’я проєкту, та задається необхідні параметри для підключення до БД. Головними параметрами є: spring.datasource.url, spring.datasource.username, spring.datasource.password.

Параметр spring.datasource.url – визначає порт до БД. Шлях до БД є шаблонним, в якому враховується хост, порт та назва БД. В нашому випадку хост = postgresql, порт = localhost:5432, назва = trade_shop. У підсумку повний URL виглядає так: spring.datasource.url=jdbc:postgresql://localhost:5432/trade_shop

spring.datasource.username – визначає логін облікового запису користувача БД, який було введено при встановленні СУБД.

Останнім важливим параметром є spring.datasource.password – пароль облікового запису БД. Відповідає за захищене з’єднання з СУБД.

Параметр spring.datasource.hikari.auto-commit=false вимикає автоматичну фіксацію транзакцій у пул з’єднань, яке використовується за замовченням у Spring Boot. Таке рішення було прийнято при розробці з урахуванням специфіки роботи з об’ємними даними (фото) в БД.

В postgreSQL при використанні автокоміту, коли передаються важкі дані, може виникнути помилки або процес збереження пройде некоректно, можливе часткове збереження даних або порушення цілісності транзакції. Вимкнення автокоміту дозволяє котролювати момент завершення транзакцій вручну та запобігає виникнення небажаних помилок.

Таким чином, правильне налаштування параметрів у application.properties забезпечує безпечне підключення до БД, що є критично важливим для коректної роботи всі компонентів веб-додатку.

Процес створення БД здійснюється автоматично під час запуску застосунку. Це стає можливим завдяки інтеграції з ORM-фреймворком Spring Data JPA та налаштуванням у application.properties: spring.jpa.hibernate.ddl-auto=update. Цей параметр дає наказ Hibernate оновлювати схеми БД відповідно до класах сутностей. Якщо таблиця відсутня – Hibernate створює її автоматично, при відсутності – таблиця оновлюється з урахуванням змін. Hibernate не може автоматично створити БД, він відповідає тільки за створення та оновлення схем, перед початком її треба створити через pgAdmin.

Кожна таблиця генерується на основі класів, які мають відповідну анотацію: @Entity. Це відкриває можливість не створювати SQL-запити вручну, а дозволяє створити клас з описаною структурою таблиці, помітивши відповідною анотацією. Назви полів, таблиць та ключів визначається через відповідні анотації JPA: @Table, @Column, @Id, @GeneratedValue @ManyToOne, @JoinColumn. Як було описано в пункті 2.3, платформа має 4 класи сутності: Image, Users, Products, Role. Демонстрація структури Entity-класу Image наведено на рисунку 3.3.

```java
@Entity  23 usages  climlock
@Table(name = "images")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Image {
    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;

    @Column(name = "name")
    private String name;

    @Column(name = "originalFileName")
    private String originalFileName;

    @Column(name = "size")
    private Long size;

    @Column(name = "contentType")
    private String contentType;

    @Column(name = "isPreviewImage")
    private boolean isPreviewImage;

    @Lob
    @Column(name = "imageData")
    private byte[] imageData;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "product_id", nullable = true)
    private Product product;

}
```

Рисунок 3.3 – Структура Entity-класу Image в проєкті TradeShop

Таким чином, в додатку TradeShop підключення до БД реалізовано за допомогою Spring Boot з використанням білдтулу Maven. Автоматична генерація таблиць на основі класів-сутностей, позбавила необхідності власноруч виконувати SQL запити. Керування таблицями здійснюється за рахунок відповідних анотацій, що дозволяє створити масштабну архітектуру проєкту.

3.2 Реєстрація та авторизація користувачів

Однією з найголовніших функцій додатку є підтримка механізму реєстрації та авторизації користувачів, для доступу до персоналізованого функціоналу, розроблений за допомогою Spring Security. Для збереження нових та пошуку вже зареєстрованих користувачів створено відповідний клас SecurityConfig. Для коректного використання, новостворений клас успадковує клас WebSecurityConfigurerAdapter, це дозволяє перевизначити низку методів, для подальшого налаштування процесу керування користувачами.

Код методу: метод configure з класу SecurityConfig.

```
@Override
protected void configure(HttpSecurity http) throws Exception {
   http
         .authorizeRequests()
         .antMatchers("/", "/product/**", "/images/**", "/registration", "/user/**",
"/search/**", "/img/**")
         .permitAll()
         .anyRequest().authenticated()
         .and()
         .formLogin()
         .loginPage("/login")
         .permitAll()
         .and()
         .logout()
         .permitAll();
}
```

Метод antMatchers() – надає перелік URL шляхів, які доступні всім користувачам без автентифікації. Виконання цього методу дозволяє корегувати навігацію. На платформі передбачено відвідування наступних сторінок: головна,

авторизації та реєстрації, профіль користувача, сторінка пошуку, та переглядання медіа файлів з БД. Архітектурою платформи передбачено відображення фото чи картинок з БД за URL: /img/, тому вимагає надання доступу користувачам без автентифікації. Для обмежування запитів викликано методи: .anyRequest().authenticated(). В свою чергу .logout().permitAll() дозволяють виходити всім користувачам із системи.

Для прив'язання своєї сторінки авторизації викликається .loginPage("/login"). Проте .formLogin() вказує що авторизація повинна проходити саме через кастомну сторінку.

Також важливо забезпечити надійний захист важливої інформації в БД, зловмисники можуть намагатися передавати SQL запити, для отримання особистих даних користувачів, важливо перед збереженням паролю закодувати його, що реалізовано в наведеному класі, за допомогою BCryptPasswordEncoder.

Для авторизації та реєстрації створено дві окремі форми: login, registration. Форма авторизації обробляється Spring Security. Користувач проходить перевірку на email та пароль. Клас CustomUserDetailsService імплементує UserDetailsService, що дозволяє Spring Security автоматично заходити користувача в базі. Паролі порівнюються вже в зашифрованому виді, що запобігає перехоплення інформації.

Реєстрація відповідно проходить через форму registration, дані з форми передаються до методу CreateUser(User user) в класі UserService. Метод передбачає перевірку унікальності email, та встановлення базової ролі USER_ROLE.

Код методу CreateUser(User user):

```
public boolean CreateUser(User user) {
   String email = user.getEmail();
   if (userRepository.findByEmail(email) != null) {
      return false;
   }
   user.setActive(true);
   user.setPassword(passwordEncoder.encode(user.getPassword()));
```

```
    user.getRoles().add(Role.ROLE_USER);
    log.info("User created: " + email);
    userRepository.save(user);
    return true;
}
```

Приклад сторінки продемонстровано в рисунку 3.4.

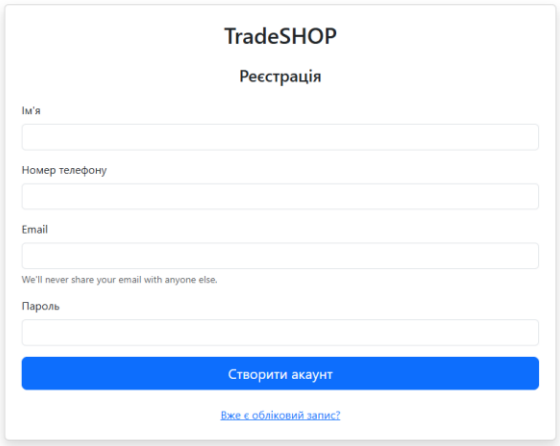


Рисунок 3.4 – сторінка реєстрації додатку TradeShop

Отже, у межах цього підпункту було реалізовано систему автентифікації користувачів. Дані користувача зберігаються у захищеному вигляді, шляхом шифрування паролю за допомогою BCryptPasswordEncoder. HTML форми забезпечують гармонічний вигляд та зручну взаємодію з користувачем.

3.3 Створення та перегляд оголошень

Головною функцією платформи TradeShop є перегляд та створення оголошень для зареєстрованих користувачів. Реалізація цього функціоналу передбачає створення форми додавання товару, його перегляд, прив’язку до користувача та подальше відображення на головній сторінці з обмеженою інформацією, окремою сторінкою та в особистому кабінеті користувачів.

Реалізація передбачає створення відповідних сторінок, класів контролерів та сервіс класів для взаємодії з БД. Демонстрація вигляду карток товарів представлена на рисунках 3.5 та 3.6.

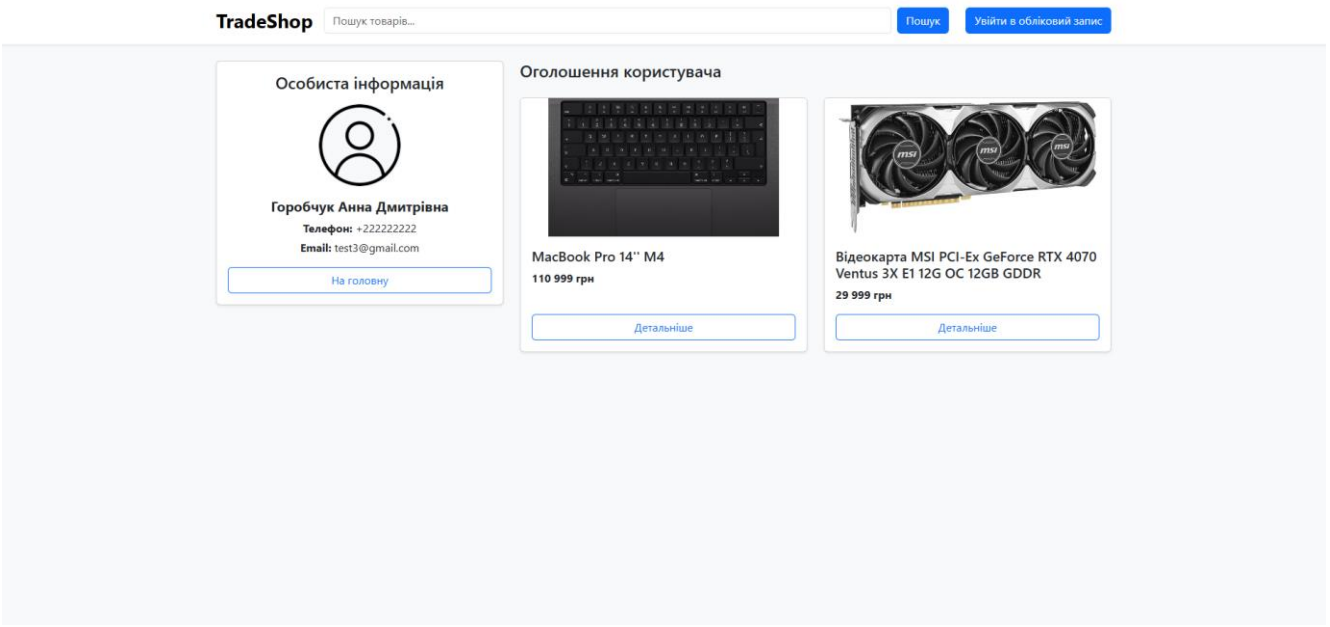


Рисунок 3.5 –вигляд особистого профілю та відображення товарів користувача на платформі TradeShop

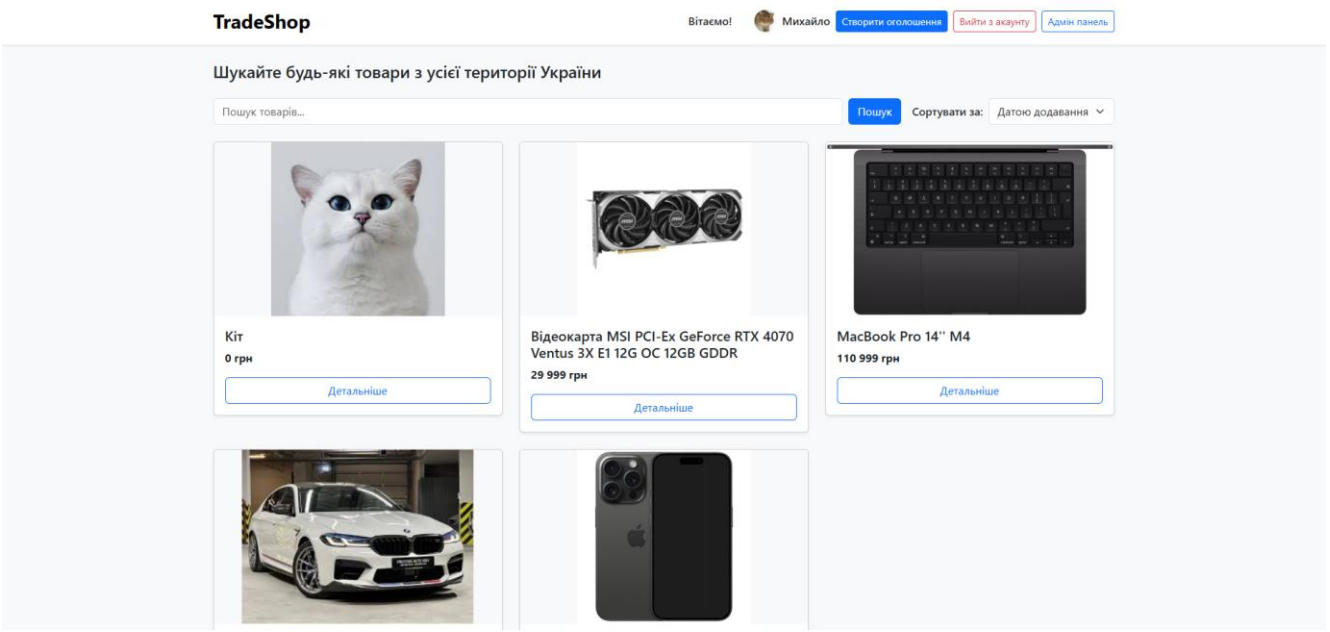


Рисунок 3.6 –вигляд головної сторінки та відображення товарів на платформі TradeShop

Кожне оголошення використовує одразу три сутності: products, users, images. На головній сторінці передбачене сортування товарів за чотирма критеріями: датою

додавання, від найдорожчих, від найдешевших, назвою. За замовчуванням товари сортуються за датою додавання, що відкриває додаткову можливість для новостворених оголошень потрапити одразу в очі користувачам. Це впливає на просування нових товарів і не дає їм залишатися в тіні. Основною інформацією для стислого відображення виступають назва та ціна, таке рішення є оптимальним для забезпечення мінімалістичного дизайну та запобігає перевантаженню інтерфейсу. Якщо користувач зацікавився оголошенням, передбачено перехід на сторінку товару для детальнішого ознайомлення по кнопці “детальніше”, яка при наведенні міняє колір заливки на синій, а шрифт стає білим.

Після переходу на головну сторішку “/”, клас ProductController перехвачує активність, збирає інформацію про статус сортування та передає його в метод getSortedProducts який знаходиться в класі ProductService. Логіка метода передбачає використання switch, враховуючи переданий спосіб сортування, виконує один з методів класу productRepository, а саме: findAllByOrderByPriceAsc(), findAllByOrderByPriceDesc(),findAllByOrderByTitleAsc(),findAllByOrderByDateOf CreationDesc(). Ці методи не реалізовуються вручну, оскільки Spring Data JPA підтримує автоматичне створення SLQ запитів на основі імен методів. Метод findAllByOrderByPriceAsc() автоматично інтерпретується як запит до таблиці products з сортуванням результатів за полем price у зростаючому порядку. Нарешті результат передається в шаблонізатор Freemarker з назвою products. В шаблонізаторі використовується спеціальний тег <#list products as product>, де products це назва атрибуту, результатів сортування, які ми передали з controller класу, а product – один елемент з списку. Шаблонізтор автоматично відображає кожне оголошення з списку в виділене для цього місці. Фреймворк Bootstrap оптимізує карточки оголошень, що дає гармонічний та структурований вигляд.

Код відображення окремого продукту на головній сторінці:

```
<#list products as product>
  <div class="col-md-4 mb-4">
    <div class="card shadow-sm">
```

```
        <img src="images/${product.previewImageId}" class="card-img-top"
alt="${product.title}">
        <div class="card-body">
          <h5 class="card-title">${product.title}</h5>
          <p class="card-text fw-bold">${product.price} грн</p>
          <a href="/product/${product.id}" class="btn btn-outline-primary w-
100">Детальніше</a>
        </div>
      </div>
  </div>
<#else>
  <h5 class="text-center text-muted">Нічого не знайдено...</h5>
</#list>
```

Якщо список оголошень пустий – на сторінці з’явиться надпис “Нічого не знайдено...”, який наголошує про відсутність будь якого товару на платформі.

Після переходу на сторінку користувача, клас UserController перехоплює дію. Пошук оголошень обраного користувача відбувається викликом методу getProducts з класу profileUser, який є об’єктом класу User. profileUser передається з Freemarker сторінки, де з об’єкту товару зчитується творець оголошення.

Для детального ознайомлення з оголошенням створена окрема сторінка product-info, id обраного оголошення передається з попередньої сторінки. В ProductController є метод з анотацією @GetMapping("/product/{id}") який перехоплює перехід, через productService викликається метод getProductById який шукає продукт по обраному id та передає у вигляді атрибуту у шаблонізатор для подальшої обробки. Додатково перевіряється авторизований сеанс користувача з акаунтом оголошення по id, у разі збігу зміна isOwner набуває значення True, який так само передається у Freemarker. У коді сторінки реалізована перевірка на “isOwner=true”, у рази виконання умови з’являється додаткова кнопка “ Видалити товар”.

Код блоку з карткою оголошення в product-info:

```
<div class="price-display">${product.price} грн</div>
   <div class="description-box">
      <p><strong>Опис товару:</strong> ${product.description}</p>
   </div>
</div>
<div class="col-md-6">
   <div class="product-title text-primary" style="font-size: 2rem; font-weight: bold;
margin-bottom: 20px;">${product.title}</div>
   <p><strong>Місто:</strong> ${product.city}</p>
   <p><strong>Продавець:</strong> <a
href="/user/${product.user.id}">${product.user.name}</a></p>
   <button id="show-phone-btn" onclick="showPhone()" class="btn contact-btn w-100
mt-3">📞 Показати номер телефону</button>
   <div id="phone-container" style="display: none;" class="mt-3">
      <div class="phone-number">${product.user.phoneNumber}</div>
   </div>
   <#if isOwner>
      <form action="/product/delete/${product.id}" method="post" class="mt-3">
         <input type="hidden" name="_csrf" value="${_csrf.token}">
         <button type="submit" class="btn btn-outline-danger w-100">🗑 Видалити
товар</button>
      </form>
   </#if>
</div>
```

Можливість створити оголошення доступна лише після авторизації на платформі. На всіх сторінках передбачені для користувачів з роллю user_role реалізована навігаційна панель, де розташовуються назва платформи, яка має гіперпосилання на головні сторінку для зручної навігації, поле пошуку, також якщо

користувач авторизований з’являються: аватар з ім’ям що мають гіперпосилання на особистий профіль, кнопки створення товару та виходу з акаунту. Для користувачів із роллю admin_role додатково відображається кнопка переходу на сторінку адміністратора. Якщо користувач не авторизований, після поля пошуку відображається тільки “увійти в обліковий запис”. Навігаційна панель зображена на рисунках 3.5 та 3.6.

Після заповнення форми вся інформація з полів передається у клас ProductController, в якому створюється новий об’єкт класу product з наданими даними, після чого з productService виконується метод saveProduct, в який передаються необхідні аргументи. У середині цього методу за principal знаходиться id користувача який створював оголошення та підв’язується до новоствореного товару. Приймається список MultipartFile файлів, які є фотографіями оголошення, перше завантажене фото стає прев’ю. Через цикл проходить кожне фото, якщо вони є, де перетворюються з файлу в об’єкт класу image, через метод toImageEntity. toImageEntity – зчинує всі дані які були зазначені в таблиці 2.4. Після отримання всіх перетворених об’єктів викликається product.setImages(images), котрий додає список фотографій до об’єкту товару та готовий товар завантажується в БД через productRepository.save(product).

Демонстрації сторінки створення оголошення зображена на рисунку 3.7.

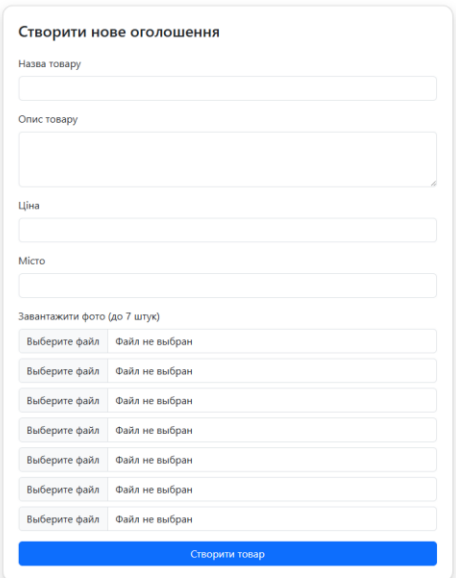


Рисунок 3.7 – сторінка створення оголошення платформи TradeShop

Реалізація функціоналу створення та перегляду оголошень функціоналу забезпечує основну бізнес-логіку додатку. Принцип заснований на шаблоні MVC, відкрив можливість розробити зручні взаємодію між користувачем, серверною логікою та БД. Вся необхідна інформація передається у Freemarker з controller класів у вигляді атрибутів, що дозволяє зручно візуалізувати безпосередньо на HTML сторінці.

3.4 Додаткові можливості платформи

Для реалізації системи адміністрування створено окремий клас AdminController, який відповідає за функціонал управління користувачами. За допомогою анотації @PreAuthorize("hasAuthority('ROLE_ADMIN')") доступ до всіх методів цього контролера обмежений лише користувачам з роллю role_admin.

Демонстрація адміністративної панелі зображено на рисунку 3.8.

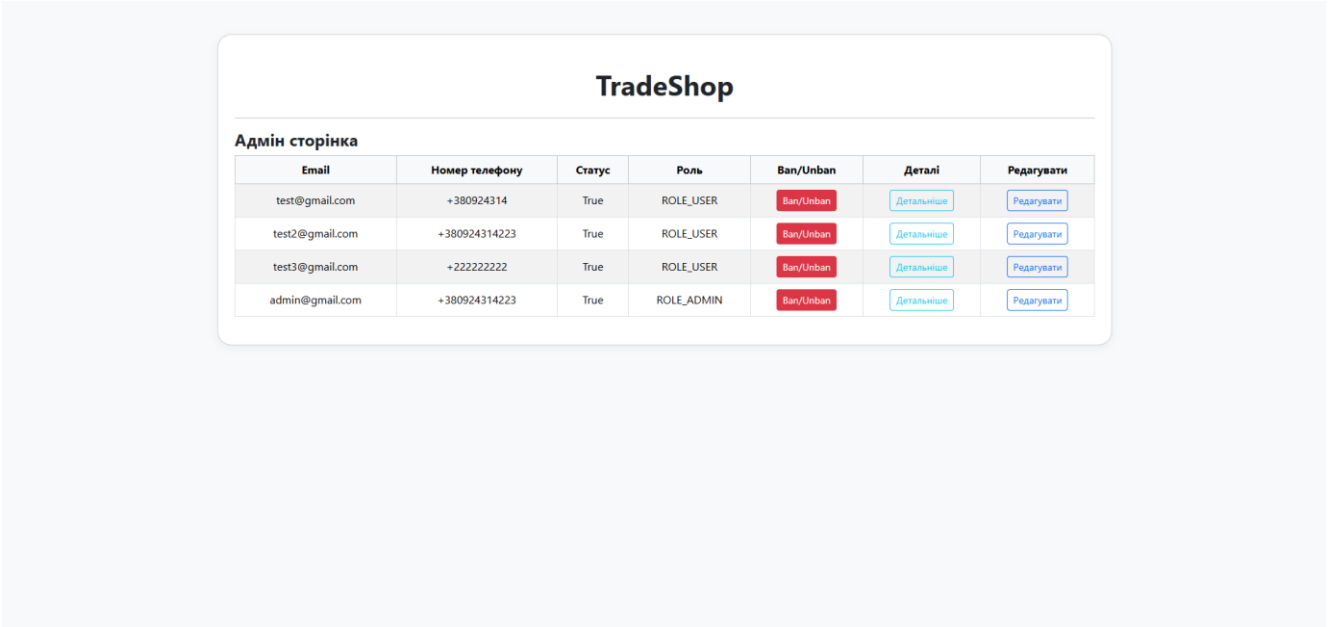


Рисунок 3.8 – адміністративна панель платформи TradeShop.

На сторінці редагування конкретного користувача реалізовано функціонал зміни ролі облікового запису та його повного видалення. За видалення відповідає метод deleteUser з класу userService, він звертається до класу репозиторію таблиці, та стандартний метод deleteById. Метод передбачає повне видалення користувача

по id також всю іншу інформацію пов’язаною з цим обліковим записом. Оскільки між таблицями встановлено відповідні зв’язки, під час видалення користувача автоматично видаляються всі пов’язані з ним об’єкти: оголошення, які він створив, та зображання. Таким чином забезпечується цілісність бази даних без необхідності очищення залежних записів.

Також було реалізовано систему пошуку товару за назвою. Після введення користувачем запиту у відповідне поле, інформація передається в метод searchProducts, що розміщений у класі контролері. Параметр title може містити ключове слово або повну назву товару. Метод getProductByTitle(String title) викликає відповідний метод з репозиторію findByTitle(title), який здійснює пошук записів у базі даних. Для реалізації пошуку товарів було використано модифікований SQL запит, замість стандартного методу Spring Data JPA, помічений анотацією @Query:

```
@Query("SELECT p FROM Product p WHERE p.title LIKE CONCAT('%', :title,
'%')")
List<Product> findByTitle(@Param("title") String title);
```

Використання такого запиту було прийнято враховуючи проблему, що виникала при використанні базового методу: після другої спроби пошуку запит не повертав результати навіть якщо вписати правильну назву наявного товару. Використання LIKE усунуло цю проблему, забезпечивши більш надійніший механізм пошуку. Результати пошуку передаються у шаблон Freemarker, де виводяться у вигляді списку, аналогічної до загальної сторінки. Це забезпечує єдиний стиль відображення то дозволяє користувачам швидко орієнтуватися.

Для кожного шаблону Freemarker які мають розширення “.ftlh” підключено фреймворк Bootstrap. Підключення здійснюється безпосередньо з офіційного DNS-сервера, це дозволяє не зберігати файли стилів і скриптів локально, а завантажувати їх з мережі. Для цього в шаблонах використовуються:

```
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
```

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></scr
ipt>

Перший тег <link> підключає таблицю CSS стилів, що відповідає за зовнішній вигляд елементів змінюючи кольори, розмір шрифтів, оформлення кнопок тощо. Другий тег <script> підключає JavaScript-функціонал, який відповідає за динамічну поведінку компонентів. Згідно з правилами веб-розробки, тег <script> підключається останнім у <body>, це впливає на первинне завантаження сторінки з оформленням, але без анімацій. Деякі веб-проєкти мають надмірно навантажений JavaScript код, що впливає на час завантаження сторінки особливо для пристроїв з низькою продуктивністю. Після підключення бібліотеки, її класи використовуються безпосередньо в HTML сторінках.

При розробці функціоналу навігаційної панелі, її код було винесено в окремий файл, це дозволило не прописувати один й той самий код в кожному фрагменті сайту, де застосовується панель. Навігаційна панель використовує змінні переданих з контролерів, зміні передається не в сам файл панелі, а в шаблон Freemarker де вона використовується. Наступний код показує як саме використовуються змінні в шаблонізаторі та використання Bootstrap:

```
<div class="d-flex align-items-center">
  <#if currentUser.email??>
    <#if searchFild?? && !searchFild>
      <span class="me-3 fw-semibold">Вітаємо!</span>
    </#if>
    <#if currentUser??>
      <div class="d-flex align-items-center ms-3">
        <a href="/user/${currentUser.id}" class="d-flex align-items-center text-
decoration-none">
          <#if currentUserAvatar??>
            <img src="/images/${currentUserAvatar}" alt="Avatar" class="rounded-
```

```
circle me-2" width="32" height="32">
                <#else>
                    <img src="/img/default_user_avatar.png" alt="Default Avatar"
class="rounded-circle me-2" width="32" height="32">
                </#if>
                <span class="fw-semibold text-dark">${currentUser.name}</span>
            </a>
        </div>
    </#if>
    <a href="/product/create/" class="btn btn-primary btn-sm ms-2">Створити
оголошення</a>
    <form action="/logout" method="post" class="d-inline">
        <input type="hidden" name="_csrf" value="${_csrf.token}">
        <button type="submit" class="btn btn-outline-danger btn-sm ms-2">Вийти з
акаунту</button>
    </form>
    <#if currentUser.isAdmin()>
        <a href="/admin" class="btn btn-outline-primary btn-sm ms-2">Адмін
панель</a>
    </#if>
  <#else>
    <a href="/login" class="btn btn-primary">Увійти в обліковий запис</a>
  </#if>
</div>
```

Цей фрагмент демонструє, як за допомогою умови можна керувати процесом відображення навігаційної панелі, в залежності від того, чи авторизований користувач. Проводиться перевірка <#if> за змінною currentUser, якщо вона існує – відображається аватар та ім’ям користувача з додатковими кнопками. Відтворення навігаційної панелі відбувається за допомогою директиви **<#**include

"fragments/navbar.ftl">. Директива вписується в залежності від розташування, враховуючи що панель вище всіх інших елементів, вона прописується на початку тегу <body>. На головній сторінці розташування поля пошуку передбачене не на навігаційної панелі, а нижче, як показано на рисунку 3.6, такий стиль вимагає реалізувати додаткову перевірку в шаблоні “navbar.ftlh” та дописати нову зміну <#assign searchFild=false> на головній сторінці.

Реалізація редагування особистого профіля відбувається в окремому шаблоні user-edit, список інформації для редагування:

- Ім’я;
- Телефон;
- Email;
- Фото профілю;

Для редагування профілю передбачена кнопка, яка розташована одразу після інформації яка описана в картці. Важливою деталлю є, те що кнопка редагування профілю з’являється тільки якщо користувач відкрив сторінку свого особистого профілю. Після переходу на /user/edit/{id} відображається HTML форма. Значення поля автоматично заповнюються поточними даними користувача, які передаються у модель за допомогою контролера. Після заповнення форми метод updateProfile отримує користувача активного сеансу та об’єкт цього користувача з оновленими даними, які передає у метод updateUserProfile. Код методу updateUserProfile(updatedUser, file, principal):

```
User user = userRepository.findById(updatedUser.getId()).orElseThrow();
user.setName(updatedUser.getName());
user.setPhoneNumber(updatedUser.getPhoneNumber());
user.setEmail(updatedUser.getEmail());
if (!file.isEmpty()) {
   try {
      Image image = productService.toImageEntity(file);
      user.setAvatar(image);
```

```
    } catch (IOException e) {
        throw new RuntimeException("Помилка збереження фото", e);
    }
}
userRepository.save(user);
```

Метод отримує новий об’єкт User, але для збереження використовує оригінальний об’єкт з БД, який отримується через userRepository.findById(updatedUser.getId()).orElseThrow(). Далі відбувається оновлення базових полів через методи: setName, setPhoneNumber та setEmail. Також якщо користувач завантажив нове фото, система перетворює його на сутність Image, у разі помилки накидається виключення RuntimeException що запобігає зупинки всій системі. Оновлений користувач зберігаються у БД через userRepository.save(user)

ВИСНОВКИ

В даній дипломній роботі було розроблено власний веб-додаток побудований за моделлю C2C. Проведено дослідження впливу IT на сучасну електронну комерцію, які підходи розробки є оптимальними, також проведено дослідження та аналіз аналогічних платформ, це дало можливість виділити переваги та недоліки подібних додатків. Було проведено ознайомлення з сучасними вимогами до веб-сайтів, з таким підходом до розробки формується детальніший список вимог до системи, що позитивно впливає на досвід користування ресурсом. До реалізації було обрано такі важливі технічні рішення як: розробка зручного інтерфейсу, процес створення та керування оголошеннями, адміністративна панель, процес реєстрації та авторизації користувачів.

В першому розділі було досліджено стан IT структур після чого було зазначено, що сучасні інтереси споживачів більше орієнтовані на інтернет-взаємодію. Враховуючи активне використання швидких сервісів доставки, це додатково показує актуальність теми створення власного веб-додатку. Другим важливим аспектом, є проведений аналіз сучасних функціональних так нефункціональних вимог, які дали визначити ключові технічні рішення. Таким чином було досліджено, що платформа повинна мати відкриту архітектуру до розширення додатку. Це забезпечує більшу гнучкість системи, що впливає на збереження додаткових витрат та конкурентоспроможність.

Також проаналізовано аналогічні платформи, це вплинуло на вибір архітектурних рішень власного проєкту. Були проаналізовані нестандартні рішення в UI/UX дизайну. Головним із нестандартних рішень - просування “VIP” оголошень, яке забезпечує додаткову можливість заробітку, проте негативно впливає на довіру користувачів. Враховуючи подібний ідею побудови платформи, безкоштовні оголошення втрачають конкурентоспроможності, що суперечить рівноправної торгівлі та негативно впливає на досвід користування платформою

Особлива увага виділена на вибір технологій, важливо забезпечити якісно побудовану архітектуру яка відповідає нормам в сучасних потребах в безпеці. Для платформ комерційного напрямку та роботою з особистими даними важливо забезпечити надійний механізм передачі та отримання даних з БД. Враховуючи ці аспекти обрано мову програмування Java, яка є найактуальнішою для поставлених задач. Завдяки використанню моделі MVC структура додатка стає упорядкованою, що позитивно впливає на орієнтацію усередині проєкту та спрощує розширення платформи.

В практичній частині розроблені найважливіші рішення для сучасного веб-додатку, які мають сильний вплив на досвід користувачів. Весь функціонал створений для підтримки гарного враження про платформу та задоволення потреб користувачів. Була продемонстрована практична реалізація особистого профіля користувача, поля пошуку, адмін сторінки, процес створення та редагування оголошень, сортування та візуалізація продуктів.

Дипломна робота виконала всі поставленні цілі та довела актуальність теми створення сласного веб-додатку, також довела актуальність практичного застосування фреймворку Spring Boot, він є найактуальнішим рішенням для Java додатків. Враховуючи Spring Data JPA – завдяки якому достатньо вказати параметри у application.properties, де з’єднання з БД відбувається автоматично та процес керування таблицями відбувається за рахунок анотацій над класами сутностями. Також робота доказала актуальність теми створення власного веб-застосунку та надала практичний приклад реалізації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТА ЛІТЕРАТУРИ

1. Spring Initializr – сервіс для генерації Spring Boot додатків. [Електронний ресурс]. – Режим доступу: https://start.spring.io (дата звернення: 21.03.2025).

2. OLX – платформа онлайн-оголошень [Електронний ресурс]. – Режим доступу: https://www.olx.ua (дата звернення: 6.05.2025).

3. Prom.ua – маркетплейс для електронної комерції [Електронний ресурс]. – Режим доступу: https://prom.ua (дата звернення: 6.05.2025).

4. HTML5 для веб-дизайнерів / Кіт Дж.; пер. з англ. – К.: XYZ, 2012. – 112 с.

5. Мартін Р. Чистий код: створення, аналіз і рефакторинг / пер. з англ. – К.: Діалектика, 2013. – 464 с.

6. Evans C. Spring in Action. – Greenwich: Manning, 2019. – 520 p.

7. Берд Б. Java для чайників. – К.: Діалектика, 2021. – 688 с.

8. Spring Boot Reference Documentation. [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-boot/docs/current/reference/html/ (дата звернення: 13.04.2025).

9. PostgreSQL Documentation. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/ (дата звертання: 11.04.2025).

10. Freemarker Manual. [Електронний ресурс]. – Режим доступу: https://freemarker.apache.org/docs/ (дата звертання: 25.04.2025).

11. Фаулер М. Patterns of Enterprise Application Architecture. — Addison-Wesley, 2003. — 560 p.

12. Блох Дж. Effective Java. — 3rd Edition. — Addison-Wesley, 2018. — 41

13. Bootstrap: офіційна документація [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 25.04.2025).

14. Masse M. REST API Design Rulebook. – O'Reilly Media, 2011. – 144 p.

15. Крамар О. В. Веб-програмування: підручник. – Львів: Львівський національний університет імені Івана Франка, 2020. – 248 с.

16. Раджпут Д. Spring. Усі патерни проєктування / пер. з англ. – Київ: Наш Формат, 2022. – 288 с.

17. Tudose C. Java Persistence with Spring Data and Hibernate. – Apress, 2020. – 576 p.

18. Spilca L. Spring Security in Action. 2nd ed. – Shelter Island: Manning Publications, 2024. – 440 p.

19. Vernon V. Implementing Domain-Driven Design. – Boston: Addison-Wesley, 2013. – 656 p.

20. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. – Boston: Addison-Wesley, 1994. – 395 p.

ДОДАТОК А. Код веб-додатку TradeShop

Клас SecurityConfig

```
package com.example.tradeshop.configurations;

import com.example.tradeshop.services.CustomUserDetailsService;
import lombok.RequiredArgsConstructor;

import org.springframework.context.annotation.Bean;
import
org.springframework.security.config.annotation.authentication.builders.Authentication
ManagerBuilder;
import
org.springframework.security.config.annotation.method.configuration.EnableGlobalMet
hodSecurity;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfig
urerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity
@RequiredArgsConstructor
@EnableGlobalMethodSecurity(prePostEnabled = true)
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    private final CustomUserDetailsService userDetailsService;
```

```
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .authorizeRequests()
                .antMatchers("/", "/product/**", "/images/**", "/registration", "/user/**",
"/test-page/**", "/search/**", "/img/**")
                .permitAll()
                .anyRequest().authenticated()
                .and()
                .formLogin()
                .loginPage("/login")
                .permitAll()
                .and()
                .logout()
                .permitAll();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService)
                .passwordEncoder(passwordEncoder());
    }

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder(8);
    }

}
```

Клас AdminController

```
package com.example.tradeshop.controllers;

import com.example.tradeshop.enums.Role;
import com.example.tradeshop.models.User;
import com.example.tradeshop.services.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;

import java.util.Map;

@Controller
@RequiredArgsConstructor
@PreAuthorize("hasAuthority('ROLE_ADMIN')")
public class AdminController {

    private final UserService userService;
    @GetMapping("/admin")
    public String admin(Model model) {
        model.addAttribute("users", userService.getAllUsers());

        return "admin";
    }
```

```
    @PostMapping("/admin/user/ban/{id}")
    public String userBan(@PathVariable("id") Long id) {
        userService.banUser(id);
        return "redirect:/admin";
    }

    @GetMapping("/admin/user/edit/{user}")
    public String userEdit(@PathVariable("user") User user, Model model) {
        model.addAttribute("user", user);
        model.addAttribute("roles", Role.values());
        return "admin-user-edit";
    }

    @PostMapping("/admin/user/edit")
    public String userEdit(@RequestParam("userId") User user, @RequestParam
Map<String, String> form){

        userService.changeUserRoles(user, form);
        return "redirect:/admin";
    }

    @PostMapping("/admin/user/delete/{id}")
    public String deleteUser(@PathVariable("id") Long id) {
        userService.deleteUser(id);
        return "redirect:/admin";
    }
}
```

Клас ImagesController

```
package com.example.tradeshop.controllers;

import com.example.tradeshop.models.Image;
import com.example.tradeshop.repositories.ImageRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.core.io.InputStreamResource;
import org.springframework.http.MediaType;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RestController;

import java.io.ByteArrayInputStream;



@RestController
@RequiredArgsConstructor
public class ImagesController {
    private final ImageRepository imageRepository;

    @GetMapping("/images/{id}")
    private ResponseEntity<?> getImageById (@PathVariable Long id) {
        Image image = imageRepository.findById(id).orElse(null);
        return ResponseEntity.ok()
                .header("fileName",image.getOriginalFileName())
                .contentType(MediaType.valueOf(image.getContentType()))
                .contentLength(image.getSize())
                .body(new InputStreamResource(new
ByteArrayInputStream(image.getImageData())));
```

```
    }
}
```

Клас ProductController

```
package com.example.tradeshop.controllers;

import com.example.tradeshop.models.Image;
import com.example.tradeshop.models.Product;
import com.example.tradeshop.models.User;
import com.example.tradeshop.services.ProductService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.security.Principal;


@Controller
@RequiredArgsConstructor
public class ProductController {
    private final ProductService productService;

    @GetMapping("/")
    public String products(@RequestParam(value = "sort", required = false, defaultValue
```

```
= "newest") String sort,
                    Model model, Principal principal) {
        Image avatarId = productService.getUserByPrincipal(principal).getAvatar();
        model.addAttribute("products", productService.getSortedProducts(sort));
        model.addAttribute("currentUser", productService.getUserByPrincipal(principal));
        if (avatarId != null) {
            model.addAttribute("currentUserAvatar", avatarId.getId());
        }
        model.addAttribute("sort", sort);
        return "products";

    }

    @GetMapping("/search")
    public String searchProducts(@RequestParam(name = "title") String title, Model
model, Principal principal) {
        Image avatarId = productService.getUserByPrincipal(principal).getAvatar();
        model.addAttribute("products", productService.getProductByTitle(title));
        model.addAttribute("currentUser", productService.getUserByPrincipal(principal));
        if (avatarId != null) {
            model.addAttribute("currentUserAvatar", avatarId.getId());
        }
        return "products";
    }

    @GetMapping("/product/{id}")
    public String productInfo(@PathVariable Long id, Model model, Principal principal)
{
        Product product = productService.getProductById(id);
        User currentUser = productService.getUserByPrincipal(principal);
```

```
        Image avatarId = productService.getUserByPrincipal(principal).getAvatar();
        model.addAttribute("product", product);
        model.addAttribute("images", product.getImages());
        model.addAttribute("currentUser", currentUser);
        if (avatarId != null) {
            model.addAttribute("currentUserAvatar", avatarId.getId());
        }
        boolean isOwner = currentUser != null &&
product.getUser().getId().equals(currentUser.getId());
        model.addAttribute("isOwner", isOwner);

        return "product-info";
    }

    @PostMapping("/product/create-new")
    public String createProduct(@RequestParam("file1") MultipartFile file1,
@RequestParam("file2") MultipartFile file2,
                                @RequestParam("file3") MultipartFile file3,
@RequestParam("file4") MultipartFile file4,
                                @RequestParam("file5") MultipartFile file5,
@RequestParam("file6") MultipartFile file6,
                                @RequestParam("file7") MultipartFile file7, Product product,
Principal principal) throws IOException {
        productService.saveProduct(principal ,product, file1, file2, file3, file4, file5, file6,
file7);
        return "redirect:/";
    }
    @PostMapping("/product/delete/{id}")
    public String deleteProduct(@PathVariable Long id) {
        productService.deleteProduct(id);
```

```
        return "redirect:/";
    }
    @GetMapping("/product/create")
    public String productCreateForm(Model model, Principal principal) {
        Image avatarId = productService.getUserByPrincipal(principal).getAvatar();
        model.addAttribute("currentUser", productService.getUserByPrincipal(principal));
        if (avatarId != null) {
            model.addAttribute("currentUserAvatar", avatarId.getId());
        }
        return "product-create";
    }
}
```

Клас UserController

```
package com.example.tradeshop.controllers;

import com.example.tradeshop.models.Image;
import com.example.tradeshop.models.User;
import com.example.tradeshop.services.ProductService;
import com.example.tradeshop.services.UserService;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.security.Principal;

@Controller
@RequiredArgsConstructor
```

```
public class UserController {

    private final UserService userService;
    private final ProductService productService;

    @GetMapping("/login")
    public String login() {
        return "login";
    }

    @GetMapping("/registration")
    public String registration() {
        return "registration";
    }

    @GetMapping("/user/edit/{id}")
    public String editUser(@PathVariable("id") Long id, Model model, Principal
principal) {
        User user = userService.findUserById(id);
        Image currentUserAvatar = user.getAvatar();
        if (!user.getUsername().equals(principal.getName())) {
            return "redirect:/";
        }
        if (currentUserAvatar != null) {
            model.addAttribute("currentUserAvatar", currentUserAvatar.getId());
        }
        model.addAttribute("currentUser", user);
        return "user-edit";
    }
```

```
    @PostMapping("/user/update")
    public String updateProfile(@ModelAttribute User updatedUser,
                                @RequestParam("file") MultipartFile file, Principal principal) {

        userService.updateUserProfile(updatedUser, file, principal);
        return "redirect:/user/" + updatedUser.getId();
    }

    @PostMapping("/registration")
    public String createUser(User user, Model model) {
        if (!userService.CreateUser(user)) {
            model.addAttribute("errorMassage", "Користувач з таким Email вже
сторенний ");
            return "registration";
        }
        return "redirect:/login";
    }

    @GetMapping("/user/{user}")
    public String userInfo(@PathVariable("user") User profileUser, Model model,
Principal principal) {
        User currentUser = productService.getUserByPrincipal(principal);
        Image avatarId = productService.getUserByPrincipal(principal).getAvatar();
        model.addAttribute("profileUser", profileUser);
        model.addAttribute("products", profileUser.getProducts());
        model.addAttribute("currentUser", currentUser);
        if (avatarId != null) {
            model.addAttribute("currentUserAvatar", avatarId.getId());
        }
        if(profileUser.getAvatar() != null) {
```

```
            model.addAttribute("userAvatar", profileUser.getAvatar().getId());
        }

        boolean isOwner = currentUser != null &&
profileUser.getId().equals(currentUser.getId());
        model.addAttribute("isOwner", isOwner);
        return "user-info";
    }

}
```

Клас Role

```
package com.example.tradeshop.enums;

import org.springframework.security.core.GrantedAuthority;

public enum Role implements GrantedAuthority {
    ROLE_USER, ROLE_ADMIN;

    @Override
    public String getAuthority() {
        return name();
    }
}
```

Клас Image

```
package com.example.tradeshop.enums;

import org.springframework.security.core.GrantedAuthority;
```

```
public enum Role implements GrantedAuthority {
    ROLE_USER, ROLE_ADMIN;

    @Override
    public String getAuthority() {
        return name();
    }
}
```

Клас Product

```
package com.example.tradeshop.models;



import javax.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

@Entity
@Table(name = "products")
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Product {
    @Id
```

```
    @GeneratedValue(strategy = GenerationType.AUTO)
    @Column(name = "id")
    private Long id;

    @Column(name = "title")
    private String title;

    @Column(name = "description", columnDefinition = "text")
    private String description;

    @Column(name = "price")
    private int price;

    @Column(name = "city")
    private String city;


    @OneToMany(cascade = CascadeType.ALL, orphanRemoval = true, fetch =
FetchType.LAZY, mappedBy = "product")
    private List<Image> images = new ArrayList<>();
    private Long previewImageId;
    private LocalDateTime dateOfCreation;

    @ManyToOne(cascade = CascadeType.REFRESH, fetch = FetchType.LAZY)
    @JoinColumn
    private User user;

    @PrePersist
    private void init() {
        dateOfCreation = LocalDateTime.now();
```

```
    }
    public void addImageToProduct(Image image) {
        image.setProduct(this);
        images.add(image);
    }
}
```

Клас User

```
package com.example.tradeshop.models;

import com.example.tradeshop.enums.Role;
import javax.persistence.*;

import lombok.Data;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;

import java.time.LocalDateTime;
import java.util.*;

@Entity
@Table(name = "users")
@Data
public class User implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Long id;

    @Column(name = "email", unique = true, nullable = false)
```

```
    private String email;

    @Column(name = "phone_number")
    private String phoneNumber;

    @Column(name = "name")
    private String name;
    @Column(name = "active")
    private boolean active;

    @OneToOne(cascade = CascadeType.ALL, fetch = FetchType.EAGER)
    @JoinColumn(name = "image_id")
    private Image avatar;

    @Column(name = "password", nullable = false, length = 1000)
    private String password;

    @ElementCollection(targetClass = Role.class, fetch = FetchType.EAGER)
    @CollectionTable(name = "user_role", joinColumns = @JoinColumn(name =
"user_id"))
    @Enumerated(EnumType.STRING)
    private Set<Role> roles = new HashSet<>();

    @OneToMany(cascade = CascadeType.ALL, fetch = FetchType.EAGER, mappedBy
= "user")
    private List<Product> products = new ArrayList<>();

    private LocalDateTime dateOfRegistration;
    @PrePersist
```

```
private void init(){
    dateOfRegistration = LocalDateTime.now();
}

//-----

public boolean isAdmin() {
    return roles.contains(Role.ROLE_ADMIN);
}

@Override
public Collection<? extends GrantedAuthority> getAuthorities() {
    return roles;
}

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

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

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

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

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

Клас ImageRepository

```
package com.example.tradeshop.repositories;


import com.example.tradeshop.models.Image;
import org.springframework.data.jpa.repository.JpaRepository;

public interface ImageRepository extends JpaRepository<Image, Long> {

}
```

Клас ProductRepository

```
package com.example.tradeshop.repositories;


import com.example.tradeshop.models.Product;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
```

```
import org.springframework.data.repository.query.Param;

import java.util.List;

public interface ProductRepository extends JpaRepository<Product, Long> {

    @Query("SELECT p FROM Product p WHERE p.title LIKE CONCAT('%', :title,
'%')")
    List<Product> findByTitle(@Param("title") String title);

    List<Product> findAllByOrderByDateOfCreationDesc();
    List<Product> findAllByOrderByPriceAsc();
    List<Product> findAllByOrderByPriceDesc();
    List<Product> findAllByOrderByTitleAsc();
}
```

Клас UserRepository

```
package com.example.tradeshop.repositories;

import com.example.tradeshop.models.User;
import org.springframework.data.jpa.repository.JpaRepository;

public interface UserRepository extends JpaRepository<User, Long> {

    User findByEmail(String email);
}
```

Клас CustomUserDetailsService

```
package com.example.tradeshop.services;

import com.example.tradeshop.repositories.UserRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Service
@RequiredArgsConstructor
public class CustomUserDetailsService implements UserDetailsService {
    private final UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String email) throws
UsernameNotFoundException {
        return userRepository.findByEmail(email);
    }
}
```

Клас ProductService

```
package com.example.tradeshop.services;

import com.example.tradeshop.models.Image;
import com.example.tradeshop.models.Product;
import com.example.tradeshop.models.User;
import com.example.tradeshop.repositories.ProductRepository;
import com.example.tradeshop.repositories.UserRepository;
import lombok.RequiredArgsConstructor;
```

```
import lombok.extern.slf4j.Slf4j;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.security.Principal;
import java.util.ArrayList;
import java.util.List;

@RequiredArgsConstructor
@Slf4j
@Service
public class ProductService {
    private final ProductRepository productRepository;
    private final UserRepository userRepository;



    public List<Product> getProductByTitle(String title) {
        List<Product> products = productRepository.findByTitle(title);
        if (title.isEmpty()) {
            return productRepository.findAll();
        }
        return products;

    }
    public List<Product> getAllProducts() {
        return productRepository.findAll();
    }

    public void saveProduct(Principal principal, Product product, MultipartFile... files)
```

```
throws IOException {
        product.setUser(getUserByPrincipal(principal));
        List<Image> images = new ArrayList<>();
        for (int i = 0; i < files.length; i++) {
            MultipartFile file = files[i];
            if (!file.isEmpty()) {
                Image image = toImageEntity(file);
                if (i == 0){
                    image.setPreviewImage(true);
                }
                image.setProduct(product);
                images.add(image);
            }
        }

        log.info("Saving new Product. Title: {}; Author: {}", product.getTitle(),
product.getUser().getEmail());
        product.setImages(images);
        Product productDB = productRepository.save(product);
        if (!images.isEmpty()){
            productDB.setPreviewImageId(productDB.getImages().get(0).getId());
            productRepository.save(product);
        }

    }

    public User getUserByPrincipal(Principal principal) {
        if (principal == null) return new User();
        return userRepository.findByEmail(principal.getName());
    }
```

```
public Image toImageEntity(MultipartFile file) throws IOException {
    Image image = new Image();
    image.setName(file.getName());
    image.setOriginalFileName(file.getOriginalFilename());
    image.setContentType(file.getContentType());
    image.setSize(file.getSize());
    image.setImageData(file.getBytes());
    return image;

}

public void deleteProduct(Long id) {
    productRepository.deleteById(id);
}

public Product getProductById(Long id) {
   return productRepository.findById(id).orElse(null);
}

public List<Product> getSortedProducts(String sort) {
    switch (sort) {
        case "cheap":
            return productRepository.findAllByOrderByPriceAsc();
        case "expensive":
            return productRepository.findAllByOrderByPriceDesc();
        case "title":
            return productRepository.findAllByOrderByTitleAsc();
        default:
```

```
            return productRepository.findAllByOrderByDateOfCreationDesc();
        }
    }

}
```

Клас UserService

```
package com.example.tradeshop.services;



import com.example.tradeshop.enums.Role;
import com.example.tradeshop.models.Image;
import com.example.tradeshop.models.User;
import com.example.tradeshop.repositories.UserRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.slf4j.Slf4j;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.stereotype.Service;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.security.Principal;
import java.util.Arrays;
import java.util.List;
import java.util.Map;
import java.util.Set;
import java.util.stream.Collectors;

@Service
@Slf4j
```

```
@RequiredArgsConstructor
public class UserService {
    private final UserRepository userRepository;
    private final PasswordEncoder passwordEncoder;
    private final ProductService productService;

    public boolean CreateUser(User user) {
        String email = user.getEmail();
        if (userRepository.findByEmail(email) != null) {
            return false;
        }
        user.setActive(true);
        user.setPassword(passwordEncoder.encode(user.getPassword()));
        user.getRoles().add(Role.ROLE_USER);
        log.info("User created: " + email);
        userRepository.save(user);
        return true;
    }

    public List<User> getAllUsers() {
        return userRepository.findAll();
    }

    public void banUser(Long id) {
        User user = userRepository.findById(id).orElse(null);

        if (user != null) {
            if (user.isActive()) {
                log.info("User banned: " + user.getEmail());
                user.setActive(false);
```

```
            } else{
                log.info("User unbanned: " + user.getEmail());
                user.setActive(true);
            }

        }
        userRepository.save(user);
    }

    public void changeUserRoles(User user, Map<String, String> form) {
        Set<String> roles = Arrays.stream(Role.values())
                .map(Role::name)
                .collect(Collectors.toSet());
        user.getRoles().clear();
        for (String key : form.keySet()) {
            if (roles.contains(key)) {
                user.getRoles().add(Role.valueOf(key));
            }
        }
        userRepository.save(user);
    }

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

    public void updateUserProfile(User updatedUser, MultipartFile file, Principal
principal) {
        User user = userRepository.findById(updatedUser.getId()).orElseThrow();
```

```
        user.setName(updatedUser.getName());
        user.setPhoneNumber(updatedUser.getPhoneNumber());
        user.setEmail(updatedUser.getEmail());

        if (!file.isEmpty()) {
            try {
                Image image = productService.toImageEntity(file);
                user.setAvatar(image);
            } catch (IOException e) {
                throw new RuntimeException("Помилка збереження фото", e);
            }
        }
        userRepository.save(user);
    }


    public void deleteUser(Long id) {
        userRepository.deleteById(id);
    }
}
```

Клас TradeshopApplication

```
package com.example.tradeshop;

import com.example.tradeshop.repositories.ProductRepository;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class TradeshopApplication {
```

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

}
```

Файл navbar.ftlh

```
<style>
    body {
        background-color: #f8f9fa;
        font-family: 'Segoe UI', sans-serif;
    }
    .navbar-brand {
        font-weight: 700;
        font-size: 28px;
    }
    .card img {
        object-fit: contain;
        width: 100%;
        height: 250px;
        background-color: #f8f9fa;
    }
    .btn-outline-primary:hover {
        background-color: #0d6efd;
        color: white;
    }

</style>
<nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm mb-4">
    <div class="container">
```

```
<a class="navbar-brand" href="/">TradeShop</a>
<#if searchFild?? && searchFild>
    <form class="flex-grow-1 d-flex me-4" action="/search" method="get" style="width: 550px;">
        <input class="form-control me-2 w-100" type="text" name="title" placeholder="Пошук товарів...">
        <button class="btn btn-primary" type="submit">Пошук</button>
    </form>
</#if>
<div class="d-flex align-items-center">
    <#if currentUser.email??>
        <#if searchFild?? && !searchFild>
            <span class="me-3 fw-semibold">Вітаємо!</span>
        </#if>

        <#if currentUser??>
            <div class="d-flex align-items-center ms-3">
                <a href="/user/${currentUser.id}" class="d-flex align-items-center text-decoration-none">
                    <#if currentUserAvatar??>
                        <img src="/images/${currentUserAvatar}" alt="Avatar" class="rounded-circle me-2" width="32" height="32">
                    <#else>
                        <img src="/img/default_user_avatar.png" alt="Default Avatar" class="rounded-circle me-2" width="32" height="32">
                    </#if>
                    <span class="fw-semibold text-dark">${currentUser.name}</span>
                </a>
            </div>
```

```
            </#if>
            <a href="/product/create/" class="btn btn-primary btn-sm ms-2">Створити оголошення</a>
            <form action="/logout" method="post" class="d-inline">
                <input type="hidden" name="_csrf" value="${_csrf.token}">
                <button type="submit" class="btn btn-outline-danger btn-sm ms-2">Вийти з акаунту</button>
            </form>
            <#if currentUser.isAdmin()>
                <a href="/admin" class="btn btn-outline-primary btn-sm ms-2">Адмін панель</a>
            </#if>
        <#else>
            <a href="/login" class="btn btn-primary">Увійти в обліковий запис</a>
        </#if>
    </div>
  </div>
</nav>
```

Файл admin.ftlh

```
<!DOCTYPE html>
<html>
<head>
  <title>TradeShop</title>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH" crossorigin="anonymous">
  <style>
    body {
```

```
            background-color: #f8f9fa;
            font-family: 'Segoe UI', Tahoma, sans-serif;
        }
        .table th, .table td {
            vertical-align: middle;
            text-align: center;
        }
        .card {
            border-radius: 20px;
            box-shadow: 0 4px 12px rgba(0, 0, 0, 0.05);
        }
        h1, h4 {
            font-weight: 700;
        }
    </style>
</head>
<body>
<div class="container my-5">
    <div class="card p-4">
        <div class="text-center mt-4">
            <h1>TradeShop</h1>
        </div>
        <hr>
        <h4>Адмін сторінка</h4>
            <table class="table table-striped table-bordered">
                <thead class="table-light">
                <tr>
                    <th>Email</th>
                    <th>Номер телефону</th>
                    <th>Статус</th>
```

```
                <th>Роль</th>
                <th>Ban/Unban</th>
                <th>Деталі</th>
                <th>Редагувати</th>
            </tr>
            </thead>
            <tbody>
            <#list users as user>
                <tr>
                    <td>${user.email}</td>
                    <td>${user.phoneNumber}</td>
                    <td><#if user.active >True<#else>False</#if></td>
                    <td><#list user.roles as role>${role}<#sep>, </#list></td>
                    <td>
                        <form action="/admin/user/ban/${user.id}" method="post">
                            <input type="hidden" name="_csrf" value="${_csrf.token}">
                            <button type="submit" class="btn btn-danger btn-
sm">Ban/Unban</button>
                        </form>
                    </td>
                    <td><a href="/user/${user.id}" class="btn btn-outline-info btn-
sm">Детальніше</a></td>
                    <td><a href="/admin/user/edit/${user.id}" class="btn btn-outline-
primary btn-sm">Редагувати</a></td>
                </tr>
            <#else>
                <h5 class="text-center text-muted">Користувачі відсутні</h5>
            </#list>
            </tbody>
        </table>
```

```
    </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
</html>
```

Файл admin-user-edit.ftlh

```
<!DOCTYPE html>
<html>
<head>
    <title>TradeShop</title>
</head>
<body>
<h1>TradeShop</h1>
<h3>Редагування користувача ${user.name}</h3>
<form action="/admin/user/edit" method="post">
    <#list roles as role>
        <div>
            <label><input type="checkbox" name="${role}"
${user.roles?seq_contains(role)?string("checked","")}>${role}</label>
        </div>
    </#list>
    <input type="hidden" value="${user.id}" name="userId">
    <input type="hidden" name="_csrf" value="${_csrf.token}">
    <button type="submit">Зберегти</button>
    <br>
```

```
</form>
<form action="/admin/user/delete/${user.id}" method="post">
    <input type="hidden" name="_csrf" value="${_csrf.token}">
    <button type="submit">Видалити акаунт</button>
</form>
</body>
</html>
```

Файл login.ftlh

```
<!DOCTYPE html>
<html>
<head>
    <title>TradeShop</title>
</head>
<body>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEw
IH" crossorigin="anonymous">

<div class="d-flex justify-content-center align-items-center vh-100">
    <div class="card p-4 shadow" style="width: 800px;">
        <h2 class="text-center mb-4">TradeSHOP</h2>
        <h4 class="text-center mb-4">Авторизація</h4>
        <form action="login" method="post">
            <div class="mb-3">
                <label for="exampleInputEmail1" class="form-label">Email address</label>
                <input type="email" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp" name="username">
                <div id="emailHelp" class="form-text">We'll never share your email with
```

```
anyone else.</div>
        </div>
        <div class="mb-3">
            <label for="exampleInputPassword1" class="form-label">Password</label>
            <input type="password" class="form-control" id="exampleInputPassword1"
name="password">
        </div>
        <input type="hidden" name="_csrf" value="${_csrf.token}">
        <button type="submit" class="btn btn-primary btn-lg w-100">Увійти</button>
        <div class="text-center mt-4">
            <a href="registration">Немає облікового запису?</a>
        </div>

    </form>

  </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
</html>
```

Файл product-create.ftlh

```
<!doctype html>
<html lang="uk">
<head>
  <meta charset="utf-8">
```

```
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>TradeShop - Створити оголошення</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
    <style>
        .create-btn {
            background-color: #0d6efd;
            color: white;
            border: 1px solid #0d6efd;
            transition: all 0.2s ease;
        }
        .create-btn:hover {
            transform: scale(1.05);
            box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
        }
        .card {
            border-radius: 16px;
        }
    </style>
</head>
<body>

<nav class="navbar navbar-expand-lg navbar-light bg-white shadow-sm mb-4">
    <div class="container">
        <a class="navbar-brand fw-bold" href="/">TradeShop</a>
        <div>
            <span class="me-3 fw-semibold">Вітаємо, ${currentUser.name}</span>
            <a href="/product/create" class="btn btn-primary btn-sm ms-2">Створити
оголошення</a>
            <form action="/logout" method="post" class="d-inline ms-2">
```

```
            <input type="hidden" name="_csrf" value="${_csrf.token}">
            <button type="submit" class="btn btn-outline-danger btn-sm">Вийти з акаунту</button>
        </form>
      </div>
   </div>
</nav>

<#if currentUser.email??>
   <div class="container">
      <div class="card p-4 shadow mx-auto" style="max-width: 700px;">
         <h4 class="mb-4">Створити нове оголошення</h4>
         <form action="/product/create-new" method="post" enctype="multipart/form-data">
            <div class="mb-3">
               <label class="form-label">Назва товару</label>
               <input type="text" class="form-control" name="title" required>
            </div>
            <div class="mb-3">
               <label class="form-label">Опис товару</label>
               <textarea class="form-control" name="description" rows="3" required></textarea>
            </div>
            <div class="mb-3">
               <label class="form-label">Ціна</label>
               <input type="number" class="form-control" name="price" required>
            </div>
            <div class="mb-3">
               <label class="form-label">Місто</label>
               <input type="text" class="form-control" name="city" required>
```

```
            </div>
            <div class="mb-3">
                <label class="form-label">Завантажити фото (до 7 штук)</label>
                <#list 1..7 as i>
                    <input type="file" class="form-control mb-2" name="file${i}">
                </#list>
            </div>
            <input type="hidden" name="_csrf" value="${_csrf.token}">
            <button type="submit" class="btn create-btn w-100">Створити
товар</button>
        </form>
    </div>
</div>
</#if>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></scr
ipt>
</body>
</html>
```

Файл product-info.ftlh

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>TradeShop - Перегляд товару</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
```

```
<style>
    body {
        background-color: #f8f9fa;
    }
    .product-card {
        max-width: 1500px;
        margin: 30px auto;
        padding: 20px;
        border-radius: 16px;
        background-color: #fff;
        box-shadow: 0 4px 12px rgba(0,0,0,0.05);
        width: 100%;
    }
    .carousel-inner img {
        border-radius: 10px;
        object-fit: contain;
        width: 100%;
        height: auto;
        max-height: 500px;
    }
    .product-title {
        font-size: 1.8rem;
        font-weight: bold;
        margin-top: 15px;
        color: #212529;
    }
    .price-display {
        font-size: 2rem;
        font-weight: bold;
        color: #0d6efd;
```

```
    margin-top: 20px;
}
.description-box {
    border-top: 1px solid #dee2e6;
    margin-top: 15px;
    padding-top: 15px;
}
.contact-btn {
    background-color: #0d6efd;
    color: white;
    transition: all 0.2s ease;
}
.contact-btn:hover {
    transform: scale(1.05);
    box-shadow: 0 4px 12px rgba(0, 0, 0, 0.1);
}
.phone-number {
    background-color: #e9f5ff;
    border: 1px solid #0d6efd;
    color: #0d6efd;
    border-radius: 8px;
    padding: 10px;
    text-align: center;
    font-weight: bold;
    font-size: 1.2rem;
    transition: all 0.3s ease;
}
.container {
    max-width: 1400px;
}
```

```
    </style>
</head>
<body>
<#assign searchFild=true>
<#include "fragments/navbar.ftl">
<div class="container">
    <div class="product-card">
        <h3 class="fw-bold mb-4">Інформація про товар</h3>
        <div class="row">
            <div class="col-md-6">
                <div id="carouselExample" class="carousel slide">
                    <div class="carousel-inner">
                        <#list images as img>
                            <div class="carousel-item <#if img?index == 0>active</#if>">
                                <img src="/images/${img.id}" class="d-block w-100" alt="Фото товару">
                            </div>
                        </#list>
                    </div>
                    <button class="carousel-control-prev" type="button" data-bs-target="#carouselExample" data-bs-slide="prev">
                        <span class="carousel-control-prev-icon" aria-hidden="true"></span>
                        <span class="visually-hidden">Previous</span>
                    </button>
                    <button class="carousel-control-next" type="button" data-bs-target="#carouselExample" data-bs-slide="next">
                        <span class="carousel-control-next-icon" aria-hidden="true"></span>
                        <span class="visually-hidden">Next</span>
                    </button>
                </div>
```

```
            <div class="price-display">${product.price} грн</div>
            <div class="description-box">
                <p><strong>Опис товару:</strong> ${product.description}</p>
            </div>
        </div>
        <div class="col-md-6">
            <div class="product-title text-primary" style="font-size: 2rem; font-weight:
bold; margin-bottom: 20px;">${product.title}</div>
            <p><strong>Місто:</strong> ${product.city}</p>
            <p><strong>Продавець:</strong> <a
href="/user/${product.user.id}">${product.user.name}</a></p>
            <button id="show-phone-btn" onclick="showPhone()" class="btn contact-btn
w-100 mt-3">📞 Показати номер телефону</button>
            <div id="phone-container" style="display: none;" class="mt-3">
                <div class="phone-number">${product.user.phoneNumber}</div>
            </div>
            <#if isOwner>
                <form action="/product/delete/${product.id}" method="post" class="mt-
3">
                    <input type="hidden" name="_csrf" value="${_csrf.token}">
                    <button type="submit" class="btn btn-outline-danger w-100">🗑
Видалити товар</button>
                </form>
            </#if>
        </div>
    </div>
</div>
```

```
<script>
    function showPhone() {
        document.getElementById('show-phone-btn').style.display = 'none';
        document.getElementById('phone-container').style.display = 'block';
    }
</script>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></scr
ipt>
</body>
</html>
```

Файл product.ftlh

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>TradeShop</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">

</head>
<body>
<#assign searchFild=false>
<#include "fragments/navbar.ftl">

<div class="container">
    <h4 class="mb-4">Шукайте будь-які товари з усієї території України</h4>
    <div class="d-flex justify-content-between align-items-center mb-4">
```

```
        <form class="d-flex flex-grow-1 me-3" action="/search" method="get">
            <input class="form-control me-2" type="text" name="title"
placeholder="Пошук товарів...">
            <button class="btn btn-primary" type="submit">Пошук</button>
        </form>
        <div class="d-flex align-items-center">
            <label for="sortSelect" class="me-2 fw-semibold mb-0">Сортувати за:</label>
            <select id="sortSelect" class="form-select w-auto" onchange="location =
this.value;">
                <option value="/?sort=newest" <#if sort?? && sort ==
"newest">selected</#if>>Датою додавання</option>
                <option value="/?sort=expensive" <#if sort?? && sort ==
"expensive">selected</#if>>Від найдорожчих</option>
                <option value="/?sort=cheap" <#if sort?? && sort ==
"cheap">selected</#if>>Від найдешевших</option>
                <option value="/?sort=title" <#if sort?? && sort ==
"title">selected</#if>>Назвою</option>
            </select>
        </div>
    </div>

    <div class="row">
        <#list products as product>
            <div class="col-md-4 mb-4">
                <div class="card shadow-sm">
                    <img src="images/${product.previewImageId}" class="card-img-top"
alt="${product.title}">
                    <div class="card-body">
                        <h5 class="card-title">${product.title}</h5>
                        <p class="card-text fw-bold">${product.price} грн</p>
```

```
                    <a href="/product/${product.id}" class="btn btn-outline-primary w-
100">Детальніше</a>
                </div>
            </div>
        </div>
    <#else>
        <h5 class="text-center text-muted">Нічого не знайдено...</h5>
    </#list>
  </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></scr
ipt>
</body>
</html>
```

Файл registration.ftlh

```
<!DOCTYPE html>
<html>
<head>
  <title>TradeShop</title>
</head>
<body>
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet" integrity="sha384-
QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEw
IH" crossorigin="anonymous">
<div class="d-flex justify-content-center align-items-center vh-100">
  <div class="card p-4 shadow" style="width: 800px;">
```

```
<h2 class="text-center mb-4">TradeSHOP</h2>
<h4 class="text-center mb-4">Реєстрація</h4>
<form action="registration" method="post">
    <div class="mb-3">
        <label for="exampleInputEmail1" class="form-label">Ім'я</label>
        <input type="text" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp" name="name">
    </div>
    <div class="mb-3">
        <label for="exampleInputPassword1" class="form-label">Номер
телефону</label>
        <input type="text" class="form-control" id="exampleInputPassword1"
name="phoneNumber">
    </div>
    <div class="mb-3">
        <label for="exampleInputPassword1" class="form-label">Email</label>
        <input type="email" class="form-control" id="exampleInputEmail1" aria-
describedby="emailHelp" name="email">
        <div id="emailHelp" class="form-text">We'll never share your email with
anyone else.</div>
    </div>
    <div class="mb-3">
        <label for="exampleInputPassword1" class="form-label">Пароль</label>
        <input type="password" class="form-control" id="exampleInputPassword1"
name="password">
    </div>
    <input type="hidden" name="_csrf" value="${_csrf.token}">
    <button type="submit" class="btn btn-primary btn-lg w-100">Створити
акаунт</button>
    <div class="text-center mt-4">
```

```
            <a href="login">Вже є обліковий запис?</a>
            <#if errorMassage??>
               <h2 STYLE="color: red">${errorMassage}</h2>
            </#if>
          </div>

       </form>

   </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"
integrity="sha384-
YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7N6jIeHz"
crossorigin="anonymous"></script>
</body>
</html>
```

Файл user-edit.ftlh

```
<!DOCTYPE html>
<html lang="uk">
<head>
   <meta charset="UTF-8">
   <title>Редагування профілю</title>
   <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
<#assign searchFild=false>
<#include "fragments/navbar.ftl">
```

```
<div class="container mt-5">
    <div class="row justify-content-center">
        <div class="col-md-6">
            <div class="card shadow-sm">
                <div class="card-body">
                    <h4 class="mb-4 text-center">Редагування профілю</h4>
                    <form method="post" action="/user/update" enctype="multipart/form-data">
                        <input type="hidden" name="id" value="${currentUser.id}" />
                        <input type="hidden" name="_csrf" value="${_csrf.token}">
                        <div class="mb-3">
                            <label for="name" class="form-label">Ім'я</label>
                            <input type="text" class="form-control" id="name" name="name" value="${currentUser.name}" required>
                        </div>

                        <div class="mb-3">
                            <label for="phoneNumber" class="form-label">Телефон</label>
                            <input type="text" class="form-control" id="phoneNumber" name="phoneNumber" value="${currentUser.phoneNumber}" required>
                        </div>

                        <div class="mb-3">
                            <label for="email" class="form-label">Email</label>
                            <input type="email" class="form-control" id="email" name="email" value="${currentUser.email}" required>
                        </div>

                        <div class="mb-3">
```

```
                    <label for="avatar" class="form-label">Фото профілю</label>
                    <input type="file" class="form-control" name="file">
                </div>

                <div class="d-flex justify-content-between">
                    <a href="/user/${currentUser.id}" class="btn btn-outline-
secondary">Назад</a>
                    <button type="submit" class="btn btn-primary">Зберегти</button>
                </div>
            </form>
        </div>
    </div>
  </div>
</div>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></scr
ipt>
</body>
</html>
```

Файл user-info.ftlh

```
<!DOCTYPE html>
<html>
<head>
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"
rel="stylesheet">
</head>
<body>
```

```
<#assign searchFild=true>
<#include "fragments/navbar.ftl">

<div class="container mt-4">
    <div class="row">
        <div class="col-md-4">
            <div class="card shadow-sm mb-4">
                <div class="card-body text-center">
                    <h4 class="card-title mb-3">Особиста інформація</h4>

                    <#if userAvatar??>
                        <img src="/images/${userAvatar}" alt="Avatar" class="rounded-circle mb-3 border" style="width: 120px; height: 120px; object-fit: cover; object-position: center;">
                    <#else>
                        <img src="/img/default_user_avatar.png" alt="Default Avatar" class="rounded-circle mb-3 border" style="width: 120px; height: 120px; object-fit: cover; object-position: center;">
                    </#if>

                    <h5 class="fw-bold">${profileUser.name}</h5>
                    <p class="mb-1"><strong>Телефон:</strong> ${profileUser.phoneNumber}</p>
                    <p class="mb-3"><strong>Email:</strong> ${profileUser.email}</p>
                    <#if isOwner>
                        <a href="/user/edit/${currentUser.id}" class="btn btn-primary w-100 mb-2">Редагувати профіль</a>
                    </#if>
                    <a href="/" class="btn btn-outline-primary w-100">На головну</a>
                </div>
```

```
            </div>
        </div>
        <div class="col-md-8">
            <h4 class="mb-4">Оголошення користувача</h4>
            <div class="row">
                <#list products as product>
                    <div class="col-md-6 mb-4">
                        <div class="card shadow-sm h-100">
                            <img src="/images/${product.previewImageId}" class="card-img-top"
alt="${product.title}" style="height:200px; object-fit:cover;">
                            <div class="card-body d-flex flex-column">
                                <h5 class="card-title">${product.title}</h5>
                                <p class="card-text fw-bold">${product.price} грн</p>
                                <a href="/product/${product.id}" class="btn btn-outline-primary
mt-auto">Детальніше</a>
                            </div>
                        </div>
                    </div>
                <#else>
                    <p class="text-muted">Ще немає оголошень</p>
                </#list>
            </div>
        </div>
    </div>
</div>
<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"></scr
ipt>
</body>
```

```
</html>
```

Файл application.properties

```
spring.application.name=tradeshop
spring.datasource.url=jdbc:postgresql://localhost:5432/trade_shop
spring.datasource.username=postgres
spring.datasource.password=Password
spring.jpa.hibernate.ddl-auto=update
spring.datasource.hikari.auto-commit=false
spring.jpa.show-sql=true
spring.freemarker.expose-request-attributes=true
```

Файл pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <parent>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-parent</artifactId>
      <version>2.6.7</version>
      <relativePath/>
   </parent>
   <groupId>com.example</groupId>
   <artifactId>tradeshop</artifactId>
   <version>0.0.1-SNAPSHOT</version>
```

```
<name>TradeShop</name>
<description>TradeShop project </description>
<url/>
<licenses>
  <license/>
</licenses>
<developers>
  <developer/>
</developers>
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>17</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-freemarker</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-mail</artifactId>
```

```
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-security</artifactId>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
    </dependency>
    <dependency>
      <groupId>org.postgresql</groupId>
      <artifactId>postgresql</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.security</groupId>
      <artifactId>spring-security-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

```
    <build>
      <plugins>
        <plugin>
          <groupId>org.springframework.boot</groupId>
          <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
      </plugins>
    </build>

</project>
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ВЕБ-ДОДАТОК ДЛЯ ОНЛАЙН-ТОРГІВЛІ НА ОСНОВІ МОВИ ПРОГРАМУВАННЯ JAVA

Автор дипломної роботи: Чумаков Михайло Юрійович
Науковий керівник: Сєрих Сергій Олександрович

Актуальність теми

Розвиток та популяризація онлайн-торгівлі

- Зростання популярності C2C моделей торгівлі
- Швидка доставка сприяє популяризації торгівлі через інтернет
- Зручність та мінімальні витрати в створенні онлайн-платформи

Мета і завдання

Мета:

- розробка зручного та безпечного веб-додатку за моделлю C2C на мові Програмування Java

Завдання:

- Аналіз електронної комерції та сучасних вимог до веб-додатків
- Вибір стеку технологій для розробки
- Організація Архітектури
- Розробка власного веб-додатку на мові програмування Java з використанням сучасних фреймворків

Сучасні функціональні та нефункціональні вимоги до веб-додатків

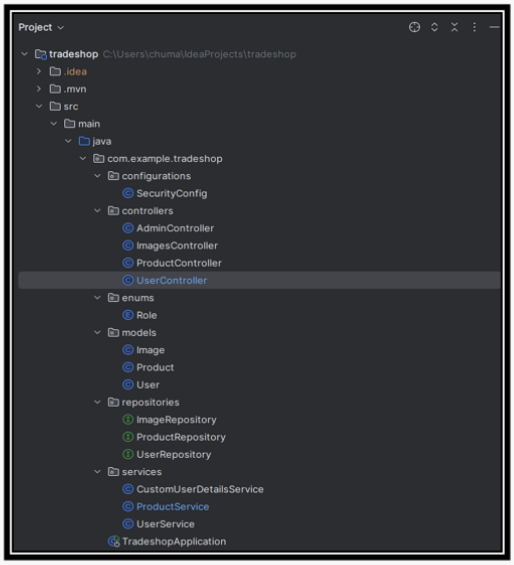
Функціональні вимоги	Не функціональні вимоги
1. Процес авторизації та реєстрації користувачів	1. Надійна безпека персональних даних
2. Створення, редагування оголошень	2. Адаптивність дизайну до різних пристроїв
3. Адміністративна сторінка	3. Масштабованість, розширення системи
4. Особистий кабінет користувача	4. Зручний UI/UX дизайн

Аналоги

Вибір мови програмування, фреймворків та бібліотек

Мова програмування: Java
Фреймворки: Spring Boot, Spring Security, Spring Data JPA, Spring WEB, Bootstrap
СУБД: PostgreSQL
Бібліотеки: Lombok
Засіб керування проєктом: Maven

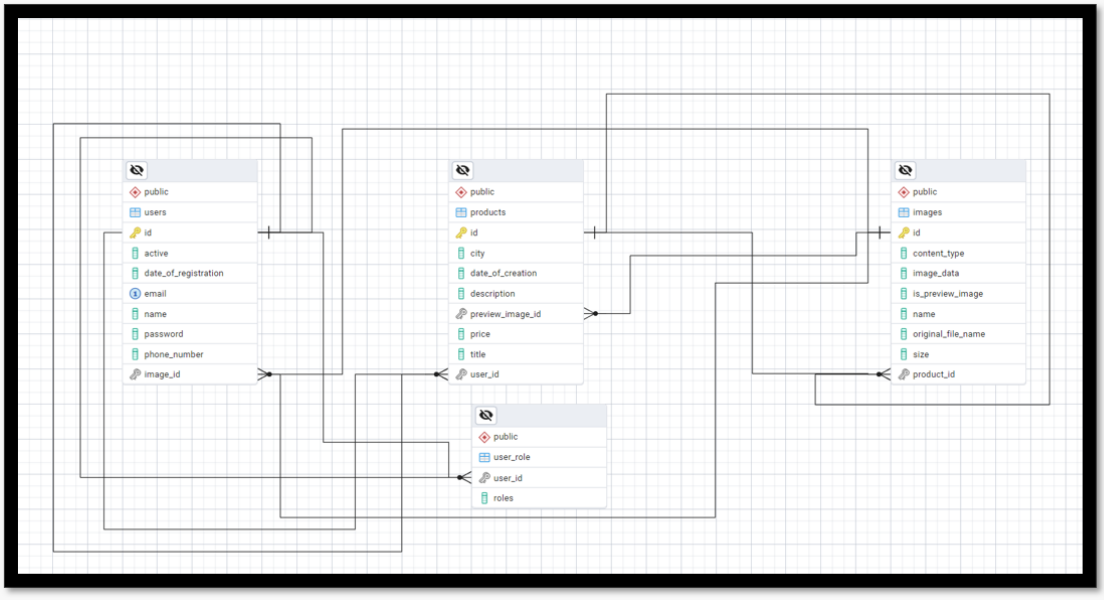
Архітектура додатку

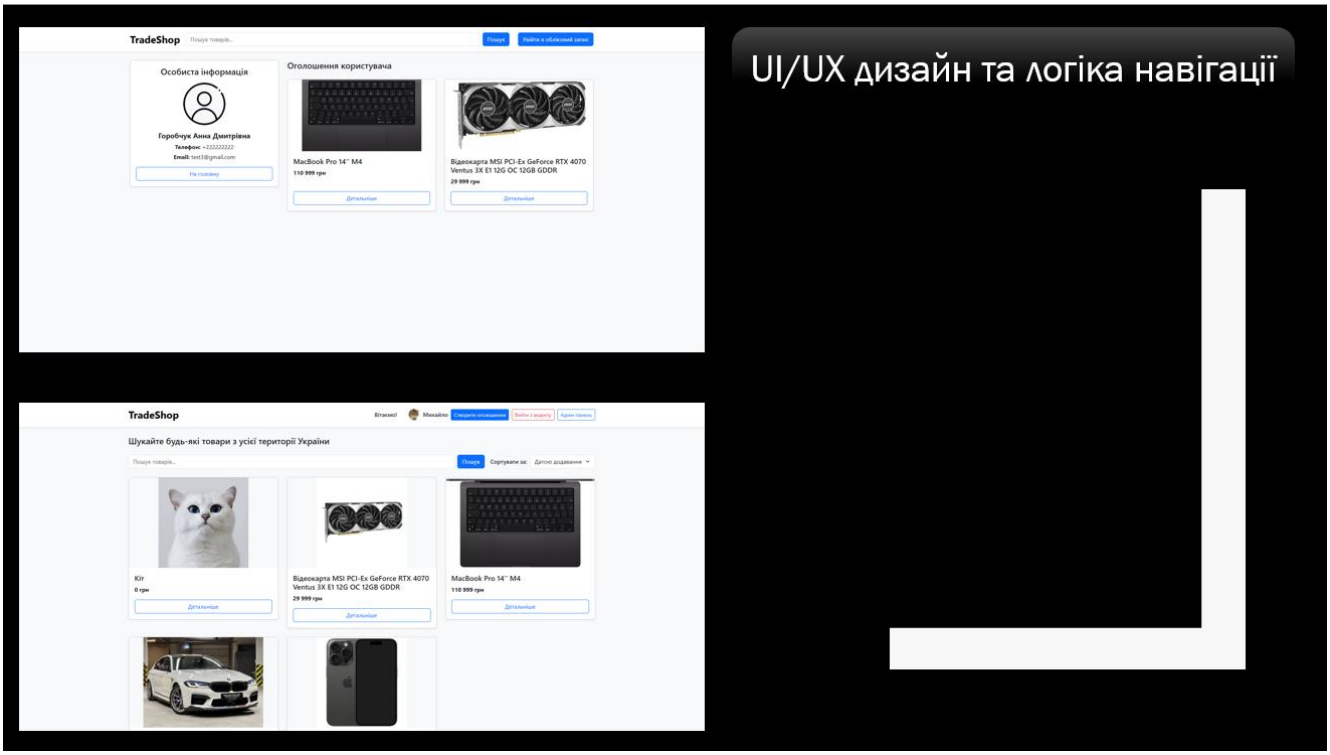


MVC(Model-View-Controller) структура:

- Controllers – Обробка HTTP запитів (запитів користувача)
- Models – Сутності БД
- Repositories – Інтерфейс взаємодії з БД
- Configuration – Налаштування безпеки
- Services – обробка, перевірка та взаємодія з даними

ER-діаграма бази даних





UI/UX дизайн та логіка навігації

Логіка обробки HTTP запитів (запитів користувача)

```
@GetMapping("/")
    public String products(@RequestParam(value =
"sort", required = false, defaultValue = "newest")
String sort,
                           Model model, Principal
principal) {
        Image avatarId =
productService.getUserByPrincipal(principal).getAvat
ar();
        model.addAttribute("p+-roducts",
productService.getSortedProducts(sort));
        model.addAttribute("currentUser",
productService.getUserByPrincipal(principal));
        if (avatarId != null) {
            model.addAttribute("currentUserAvatar",
avatarId.getId());
        }
        model.addAttribute("sort", sort);
        return "products";
    }
```

Висновки

Перспективи розвитку:
- Зростання популярності C2C платформ для торгівлі.
- Архітектура розробленого додатку передбачає майбутнє масштабування.
- Оптимізований UI/UX дизайну для сучасних споживачів
- Можливе впровадження онлайн-платежів та рейтингу продавців.
- Подальша інтеграція з зовнішніми сервісами (доставка, SMS підтвердження)

Задача	Результат
Аналіз електронної комерції та сучасних вимог до веб-додатків	Аналіз підкреслив актуальність теми даної дипломної роботи по розробці власного вед-додатку.
Вибір стеку технологій для розробки	Обраний стек технологій є найактуальніший серед Java додатків, що доказало їх практичне застосування у розробці власного додатку.
Організація Архітектури	Архітектура побудована за MVC моделлю, що відкриває можливість легко модифікувати код та розширити систему без додаткових витрат.
Розробка власного веб-додатку на мові програмування Java з використанням сучасних фреймворків	Отримано універсальне рішення , готове до подальшого розвитку.