ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка платформи для обміну технікою між користувачами»

на здобуття освітнього ступеня бакалавра
зі спеціальності 121 Інженерія програмного забезпечення
освітньо-професійної програми «Інженерія програмного забезпечення»

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

_____ Дмитро ТЕРНІЧЕНКО
(підпис)

Виконав: здобувач вищої освіти групи ПД-43

Дмитро ТЕРНІЧЕНКО

Керівник: Ігор ГАМАНЮК
ст. викладач

Рецензент:

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти Бакалавр

Спеціальність 121 Інженерія програмного забезпечення

Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____ Ірина ЗАМРІЙ
« ____ » _____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Терніченко Дмитру Юрійовичу

1. Тема кваліфікаційної роботи: «Розробка платформи для обміну технікою між користувачами»
керівник кваліфікаційної роботи ст. викладач Ігор ГАМАНЮК,
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «20» лютого 2026 року № 51.

2. Строк подання кваліфікаційної роботи «29» травня 2026 року.

3. Вихідні дані до кваліфікаційної роботи: теоретичні відомості про організацію обміну технікою між користувачами, принципи роботи P2P-платформ, підходи до створення електронних оголошень, фільтрації каталогу пристроїв, формування пропозицій обміну, ведення комунікації між користувачами, керування статусами домовленостей, залишення відгуків і надсилання сповіщень, а також технічна документація щодо використання Java Spring Boot, Spring Security, JWT, Spring Data JPA, PostgreSQL, Liquibase, React, Vite, REST API, Swagger UI та Docker.

4. Зміст розрахунково-пояснювальної записки:

1. Огляд та аналіз предметної галузі обміну технікою між користувачами, розміщення оголошень, пошуку пристроїв, формування пропозицій обміну та комунікації між учасниками.
2. Аналіз існуючих програмних засобів для розміщення та пошуку техніки, зокрема дошок оголошень, маркетплейсів і сервісів для продажу або обміну пристроями.
3. Огляд засобів реалізації web-платформи TechExchange для організації обміну технікою між користувачами.
4. Проєктування архітектури платформи, структури бази даних, доменної моделі, основних модулів системи та взаємодії компонентів.
5. Програмна реалізація та опис функціонування web-платформи TechExchange.
6. Тестування основних функцій платформи та оцінка відповідност

5. Перелік графічного матеріалу:

1. Аналіз аналогів програмного забезпечення для розміщення, пошуку та обміну технікою між користувачами.
2. Вимоги до web-платформи TechExchange.
3. Архітектура програмної реалізації платформи.
4. Структура бази даних платформи TechExchange.
5. Діаграма доменної моделі платформи TechExchange.
6. Діаграма активності роботи користувача на платформі.
7. Діаграма класів або структура основних сутностей і сервісів системи.
8. Діаграма розгортання платформи TechExchange.
9. Екранні форми web-платформи: вхід, реєстрація, головна сторінка, створення оголошення, сторінка пропозицій обміну та профіль користувача.
10. Апробація результатів дослідження.

6. Дата видачі завдання«21» лютого 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Підбір та аналіз науково-технічної літератури за темою обміну технікою між користувачами	21.02–28.02.2026	

2	Аналіз та дослідження існуючих аналогів програмного забезпечення для розміщення, пошуку та обміну технікою	01.03–07.03.2026	
3	Аналіз предметної галузі обміну технікою між користувачами та визначення вимог до web-платформи TechExchange	08.03–15.03.2026	
4	Огляд засобів реалізації web-платформи TechExchange	16.03–22.03.2026	
5	Проєктування web-платформи TechExchange, логічної структури системи, бази даних та UML-діаграм	23.03–05.04.2026	
6	Програмна реалізація серверної та клієнтської частин web-платформи TechExchange	06.04–26.04.2026	
7	Реалізація модулів пропозицій обміну, вбудованого чату, відгуків, сповіщень та адміністративного керування	27.04–03.05.2026	
8	Тестування web-платформи TechExchange та оформлення результатів тестування	04.05–10.05.2026	
9	Оформлення роботи: вступ, висновки, реферат, список джерел та додатки	11.05–16.05.2026	
10	Розробка демонстраційних матеріалів	16.05–17.05.2026	
11	Попередній захист роботи	18.05–22.05.2026	

Здобувач вищої освіти _____ Дмитро ТЕРНІЧЕНКО
(підпис)

Керівник
кваліфікаційної роботи _____ Ігор ГАМАНЮК
(підпис)

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 60 стор., 5 табл., 27 рис., 24 джерел.

Мета роботи – спрощення процесу обміну технікою між користувачами за рахунок розробки web-платформи TechExchange для публікації оголошень, пошуку пристроїв, формування пропозицій обміну, ведення переговорів та відстеження статусу домовленостей.

Об'єкт дослідження – процеси організації обміну технікою між користувачами, розміщення оголошень, пошуку пристроїв, формування пропозицій обміну та комунікації між учасниками.

Предмет дослідження – засоби, методи та технології реалізації web-платформи TechExchange для автоматизації процесу обміну технікою між користувачами, керування оголошеннями, пропозиціями обміну, повідомленнями, відгуками та сповіщеннями.

Короткий зміст роботи: у кваліфікаційній роботі проведено аналіз предметної галузі обміну технікою між користувачами та визначено основні проблеми, пов'язані з розміщенням оголошень на різних платформах, недостатньою структурованістю інформації про пристрої, складністю пошуку потрібної техніки, перенесенням комунікації у зовнішні месенджери та відсутністю централізованого контролю статусу домовленостей. Проаналізовано існуючі програмні рішення, зокрема OLX, eBay, Swappa, Shpock, Facebook Marketplace та Telegram-групи, визначено їх переваги й недоліки [1; 2].

У роботі сформовано функціональні та нефункціональні вимоги до платформи TechExchange, обґрунтовано вибір засобів реалізації клієнтської частини, серверної частини, бази даних, засобів безпеки та розгортання. Для реалізації web-платформи використано Java 17, Spring Boot, Spring Security, Spring Data JPA, PostgreSQL, Liquibase, React, Vite, React Router, Axios, Lucide React, JWT, REST API, Swagger UI, Docker Compose та Git [7; 8; 14-22].

Спроєктовано загальну архітектуру системи, структуру бази даних, діаграму доменної моделі, діаграму активності, діаграму класів, діаграму розгортання та схему взаємодії основних компонентів платформи. Реалізовано основні модулі системи: реєстрацію та авторизацію користувачів, профіль користувача, каталог пристроїв, створення та редагування оголошень, детальну сторінку пристрою, формування пропозицій обміну, керування статусами домовленостей, вбудований чат, відгуки, сповіщення та адміністративний контроль користувачів і оголошень [7; 10; 12; 13].

Особливістю розробленої системи є підтримка повного життєвого циклу пропозиції обміну: від створення заявки до прийняття, резервування пристроїв, завершення обміну та залишення відгуку. На відміну від звичайних дошок оголошень, платформа TechExchange орієнтована саме на P2P-обмін технікою, тому поєднує каталог пристроїв, структуровані оголошення, вибір власного пристрою для обміну, чат у межах конкретної домовленості та систему статусів. Проведено тестування основних функцій платформи, перевірено коректність роботи авторизації, оголошень, пропозицій обміну, повідомлень, відгуків, сповіщень та адміністративних функцій [1; 2].

КЛЮЧОВІ СЛОВА: ОБМІН ТЕХНІКОЮ, TECHEXCHANGE, WEB-ПЛАТФОРМА, P2P-ОБМІН, ОГОЛОШЕННЯ, ПРОПОЗИЦІЯ ОБМІНУ, ВБУДОВАНИЙ ЧАТ, JWT, SPRING BOOT, REACT, POSTGRESQL.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – інтерфейс прикладного програмування, що використовується для взаємодії між програмними компонентами системи.

REST – архітектурний стиль організації обміну даними між клієнтською та серверною частинами web-застосунку.

HTTP – протокол передавання даних у мережі Інтернет.

HTTPS – захищена версія протоколу HTTP, що забезпечує безпечний обмін даними між клієнтом і сервером.

JSON – текстовий формат обміну даними, який використовується для передавання інформації між frontend- і backend-частинами системи.

JavaScript – мова програмування, що використовується для створення інтерактивної поведінки web-сторінок.

React – бібліотека JavaScript для створення компонентного користувацького інтерфейсу.

Vite – інструмент для швидкого створення, запуску та збирання frontend-застосунків.

Axios – бібліотека для виконання HTTP-запитів із клієнтської частини web-застосунку.

React Router – бібліотека для організації маршрутизації між сторінками React-застосунку.

Lucide React – бібліотека іконок для оформлення інтерфейсу користувача.

Java – об’єктно-орієнтована мова програмування, що використовується для реалізації серверної частини застосунку.

Spring – фреймворк для створення серверних Java-застосунків.

Spring Boot – інструмент екосистеми Spring, що спрощує створення та запуск серверних web-застосунків.

Spring Security – модуль Spring для реалізації автентифікації, авторизації та захисту доступу до ресурсів системи.

JWT – формат токена, що використовується для безпечної передачі даних авторизації між клієнтом і сервером.

JPA – специфікація Java для роботи з реляційними базами даних через об’єктну модель.

Hibernate – фреймворк об’єктно-реляційного відображення, який забезпечує зв’язок між Java-класами та таблицями бази даних.

PostgreSQL – об’єктно-реляційна система керування базами даних.

Liquibase – інструмент для керування змінами структури бази даних за допомогою міграцій.

SPA – односторінковий web-застосунок, у якому основна взаємодія з користувачем відбувається без повного перезавантаження сторінки.

IDE – інтегроване середовище розробки програмного забезпечення.

Git – система контролю версій, що використовується для відстеження змін у програмному коді.

Docker – платформа контейнеризації, що використовується для запуску компонентів системи в ізольованому середовищі.

Docker Compose – інструмент для опису та запуску декількох контейнерів як єдиного середовища.

Swagger UI – інструмент для перегляду та тестування REST API через web-інтерфейс.

OpenAPI – специфікація для опису REST API та його endpoint-ів.

P2P – модель взаємодії між користувачами, у якій учасники обмінюються ресурсами напряму без участі магазину або посередника.

CRUD – базовий набір операцій створення, читання, оновлення та видалення даних.

TechExchange – web-платформа для обміну технікою між користувачами.

Device – сутність системи, що описує пристрій або оголошення про техніку.

Exchange Request – пропозиція обміну між користувачами в межах платформи.

Notification – системне сповіщення про події, пов’язані з обміном, повідомленнями або відгуками.

ВСТУП

У сучасних умовах активного розвитку цифрових сервісів усе більшого значення набувають web-платформи, які дозволяють користувачам взаємодіяти між собою безпосередньо. Особливо актуальним це є для сфери обміну технікою, оскільки користувачі часто мають справні пристрої, які вже не використовуються, але можуть бути корисними іншим людям. До таких пристроїв можна віднести смартфони, ноутбуки, планшети, ігрові консолі, аудіотехніку, камери, аксесуари та інші електронні пристрої [4; 10; 12].

На практиці обмін технікою часто здійснюється через універсальні дошки оголошень, соціальні мережі, месенджери або маркетплейси. Проте ці засоби здебільшого орієнтовані на купівлю-продаж товарів, а не на організацію повноцінного процесу обміну. Через це користувачам доводиться вручну описувати умови обміну, самостійно порівнювати пристрої, вести переговори у сторонніх месенджерах і контролювати домовленості без єдиного програмного середовища [1; 2].

Актуальність теми кваліфікаційної роботи зумовлена необхідністю розробки спеціалізованої web-платформи для обміну технікою між користувачами, яка дозволяє централізовано публікувати оголошення, переглядати каталог пристроїв, використовувати фільтрацію, створювати пропозиції обміну, вести переговори через вбудований чат та відстежувати статус домовленостей. Розроблювана платформа TechExchange орієнтована на користувачів, які хочуть не лише продавати або купувати техніку, а й обмінювати пристрої на більш потрібні або актуальні для себе [1; 2; 8].

Особливістю платформи TechExchange є підтримка повного життєвого циклу обміну. Користувач може створити оголошення про власний пристрій, переглянути доступні пропозиції інших учасників, вибрати власний пристрій для обміну, надіслати заявку, вести діалог у межах конкретної домовленості, змінювати статус угоди та залишати відгук після завершення обміну. Завдяки цьому платформа не

обмежується функцією звичайного каталогу, а забезпечує повноцінну взаємодію між користувачами [1; 2].

Мета роботи – спрощення процесу обміну технікою між користувачами за рахунок розробки web-платформи TechExchange для публікації оголошень, пошуку пристроїв, формування пропозицій обміну, ведення переговорів та відстеження статусу домовленостей.

Об’єкт дослідження – процеси організації обміну технікою між користувачами, розміщення оголошень, пошуку пристроїв, формування пропозицій обміну та комунікації між учасниками.

Предмет дослідження – засоби, методи та технології реалізації web-платформи TechExchange для автоматизації процесу обміну технікою між користувачами, керування оголошеннями, пропозиціями обміну, повідомленнями, відгуками та сповіщеннями.

Для досягнення поставленої мети необхідно вирішити такі завдання:

1. Проаналізувати предметну галузь обміну технікою між користувачами та визначити основні проблеми розміщення оголошень, пошуку пристроїв, формування пропозицій обміну й комунікації між учасниками [1; 2].
2. Дослідити існуючі програмні рішення для розміщення та пошуку техніки, порівняти їх функціональні можливості та визначити основні недоліки [1; 2].
3. Сформувати функціональні та нефункціональні вимоги до платформи TechExchange [8; 9].
4. Обґрунтувати вибір технологій для реалізації web-платформи [14-22].
5. Спроєктувати архітектуру платформи, структуру бази даних та основні UML-діаграми системи [7; 10; 12].
6. Реалізувати серверну частину для керування користувачами, оголошеннями, пропозиціями обміну, повідомленнями, відгуками та сповіщеннями.
7. Реалізувати клієнтський web-інтерфейс для авторизації, перегляду каталогу, створення оголошень, оформлення пропозицій обміну, ведення чату та роботи з профілем.

8. Реалізувати адміністративний модуль для контролю користувачів і оголошень.
9. Провести тестування основних функцій платформи та оцінити відповідність розробленого програмного забезпечення поставленим вимогам [7; 8; 11].

Методи дослідження. У бакалаврській роботі використовуються такі методи: аналіз предметної галузі, порівняльний аналіз існуючих програмних засобів для розміщення та пошуку техніки, об’єктно-орієнтоване проєктування, UML-моделювання, проєктування структури реляційної бази даних, розробка клієнт-серверного web-застосунку, реалізація REST API, тестування програмного забезпечення та аналіз результатів роботи системи [6; 7; 10; 11].

Наукова новизна одержаних результатів полягає у розробці структури web-платформи TechExchange, яка поєднує каталог техніки, структуровані оголошення, механізм формування пропозицій обміну, вбудований чат, статуси домовленостей, відгуки та сповіщення. На відміну від універсальних дошок оголошень, запропоноване рішення орієнтоване саме на процес P2P-обміну технікою між користувачами [1; 2; 12].

Практична значущість одержаних результатів полягає в тому, що розроблена web-платформа може бути використана користувачами для зручного розміщення техніки, пошуку пристроїв, створення пропозицій обміну, ведення переговорів і контролю домовленостей. У майбутньому систему можна розширити додаванням модуля геолокації, внутрішньої системи гарантування безпечного обміну, мобільного застосунку, розширеної системи рейтингів, модерації скарг і рекомендацій пристроїв на основі інтересів користувача [1; 2].

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ОБМІНУ ТЕХНІКОЮ МІЖ КОРИСТУВАЧАМИ

1.1 Загальна характеристика процесу обміну технікою між користувачами

У сучасних умовах активного розвитку цифрових сервісів користувачі дедалі частіше взаємодіють між собою через web-платформи, маркетплейси, дошки оголошень, соціальні мережі та месенджери. Одним із поширених напрямів такої взаємодії є обмін технікою між користувачами. До цієї категорії можна віднести смартфони, ноутбуки, планшети, ігрові консолі, комп’ютерні комплектуючі, аудіопристрої, камери, аксесуари та інші електронні пристрої [4; 10; 12].

Обмін технікою між користувачами є різновидом P2P-взаємодії, тобто моделі, за якої користувачі напряму домовляються між собою без участі магазину або централізованого продавця. На відміну від класичної купівлі-продажу, основною метою такого процесу є не обов’язково отримання грошей за товар, а заміна одного пристрою на інший, який краще відповідає актуальним потребам користувача [1; 2].

Наприклад, користувач може мати смартфон, який уже не використовує, але хоче обміняти його на планшет, ігрову консоль або інший пристрій. В іншому випадку користувач може бути зацікавлений в обміні із доплатою або в обміні лише на техніку певної категорії. Саме тому процес обміну є складнішим, ніж звичайне розміщення оголошення про продаж.

Для повноцінної організації обміну необхідно враховувати не лише сам факт наявності пристрою, а й детальні характеристики техніки: категорію, бренд, модель, технічний стан, місто, комплектацію, опис, фото та бажані умови обміну. Також важливими є можливість комунікації між учасниками, фіксація статусу домовленості та подальше оцінювання користувачів після завершення обміну [8; 9].

У загальному вигляді процес обміну технікою включає такі етапи: користувач створює оголошення про власний пристрій, інший користувач

знаходить це оголошення в каталозі, переглядає детальну інформацію, формує пропозицію обміну, після чого сторони ведуть переговори й узгоджують умови. У разі успішного погодження домовленість переходить у наступні статуси, а після завершення обміну користувачі можуть залишити відгук [1; 2].

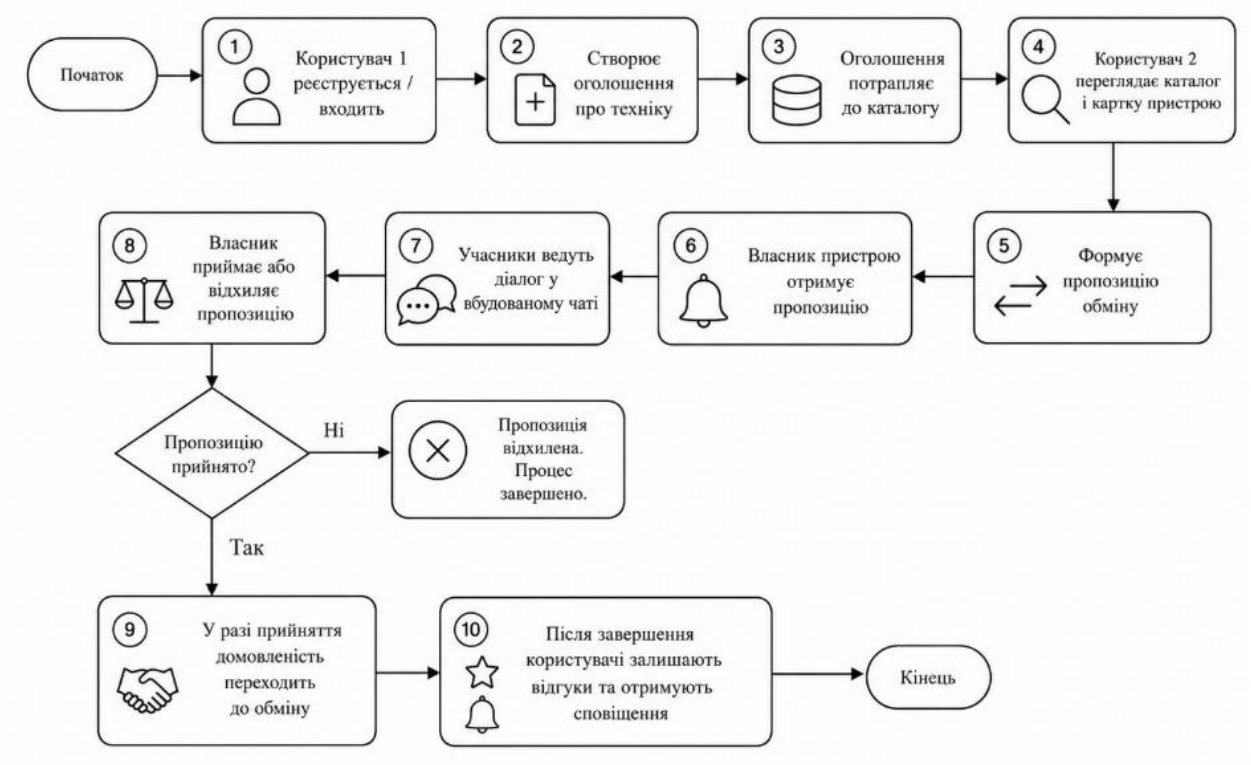


Рис. 1.1 Загальна схема процесу обміну технікою між користувачами

Особливість такої предметної галузі полягає в тому, що система має поєднувати функції каталогу, дошки оголошень, сервісу повідомлень, механізму керування угодами та системи відгуків. Якщо хоча б один із цих елементів відсутній, користувачі змушені переносити частину взаємодії в інші сервіси, що ускладнює контроль домовленостей і підвищує ризик втрати важливої інформації [10; 12].

1.2 Проблеми організації P2P-обміну технікою

На практиці обмін технікою між користувачами часто відбувається через універсальні дошки оголошень, соціальні мережі або месенджери. Такий підхід є зручним на початковому рівні, оскільки не потребує складної реєстрації або

використання спеціалізованої системи. Проте для організованого обміну технікою такі інструменти мають низку обмежень [1; 2].

Першою проблемою є неструктурованість інформації про пристрої. У багатьох сервісах користувачі самостійно описують товар у довільній формі. Через це оголошення можуть містити неповну інформацію: відсутність моделі, стану, міста, комплектації, бажаного варіанту обміну або фото. У результаті іншим користувачам доводиться додатково уточнювати базові дані в особистих повідомленнях [1; 2].

Другою проблемою є складність пошуку потрібного пристрою. Якщо платформа не має достатньо точних фільтрів за категорією, станом, містом або назвою, користувач витрачає більше часу на перегляд нерелевантних оголошень. Це особливо важливо для техніки, оскільки користувач зазвичай шукає не просто будь-який товар, а конкретний тип пристрою з певними характеристиками [1; 2].

Третьою проблемою є відсутність окремого механізму пропозиції обміну. На багатьох майданчиках користувач може лише написати повідомлення автору оголошення, але не може системно оформити пропозицію: вибрати власний пристрій, додати початкове повідомлення, зафіксувати статус заявки та відстежувати подальший перебіг домовленості [1; 2].

Четвертою проблемою є перенесення комунікації в зовнішні канали. Коли користувачі переходять у месенджери, частина інформації про домовленість відділяється від самого оголошення. Це ускладнює подальший контроль, особливо якщо користувач одночасно веде переговори з кількома людьми [1; 2].

П’ятою проблемою є недостатній рівень довіри між учасниками. У P2P-моделі користувачі взаємодіють напряму, тому важливими стають відгуки, рейтинг, історія взаємодії та можливість адміністративного контролю. Без цих механізмів складніше оцінити надійність іншого учасника обміну [1; 2].

Рис. 1.2 Основні проблеми організації обміну технікою через універсальні сервіси

Таким чином, для ефективного вирішення задачі потрібна не просто дошка оголошень, а спеціалізована web-платформа, яка враховує особливості саме обміну технікою. Така система повинна забезпечувати структуровані оголошення, каталог пристроїв, фільтрацію, пропозиції обміну, чат, статуси домовленостей, відгуки, сповіщення та адміністративний контроль [1; 2; 8].

1.3 Аналіз існуючих програмних засобів

Для визначення доцільності розробки платформи TechExchange було розглянуто існуючі програмні засоби, які можуть частково використовуватися для розміщення техніки, пошуку пристроїв, комунікації між користувачами або укладання домовленостей. До таких рішень належать OLX, eBay, Swappa, Shpock, Facebook Marketplace та Telegram-групи [1; 2].

OLX є однією з найбільш поширених дошок оголошень в Україні. Платформа дозволяє створювати оголошення, додавати фото, опис, ціну, місто та категорію товару. Для задачі обміну технікою OLX є корисним завдяки великій аудиторії та наявності категорій електроніки. Проте основна логіка сервісу орієнтована саме на

продаж. Можливість обміну зазвичай зазначається лише в тексті оголошення, а не реалізується як окрема функція [1; 2].

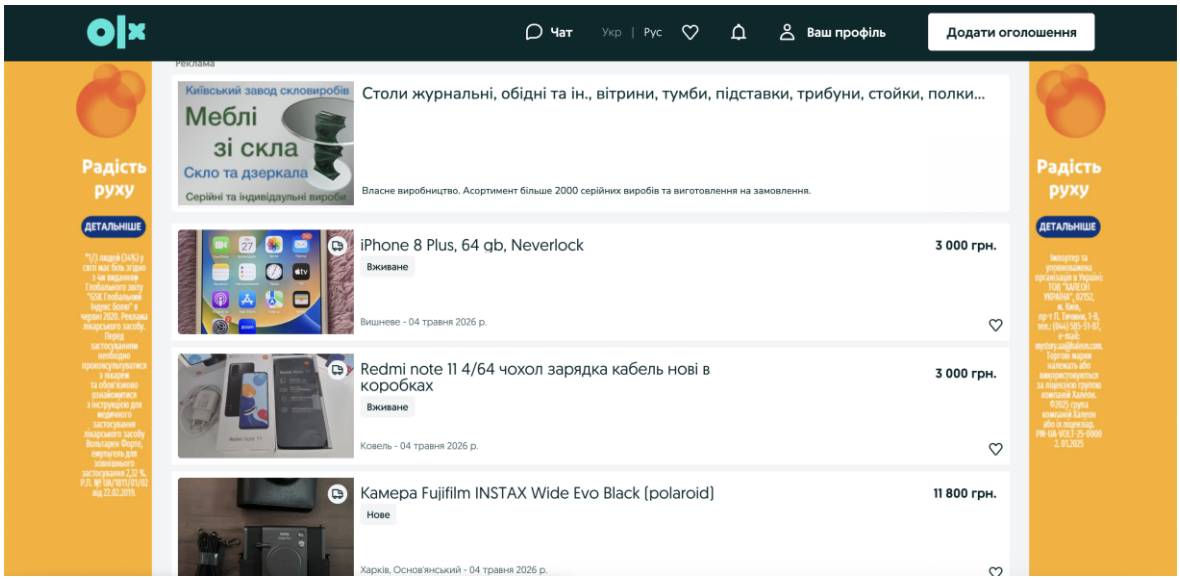


Рис. 1.3 Приклад екранної форми платформи OLX з оголошеннями техніки

eBay є міжнародним маркетплейсом, який має розвинений каталог товарів, рейтинги продавців, фільтри, опис характеристик і систему відгуків. Цей сервіс демонструє зручний підхід до структурованого представлення товарів, однак він також орієнтований переважно на купівлю-продаж або аукціонну модель.

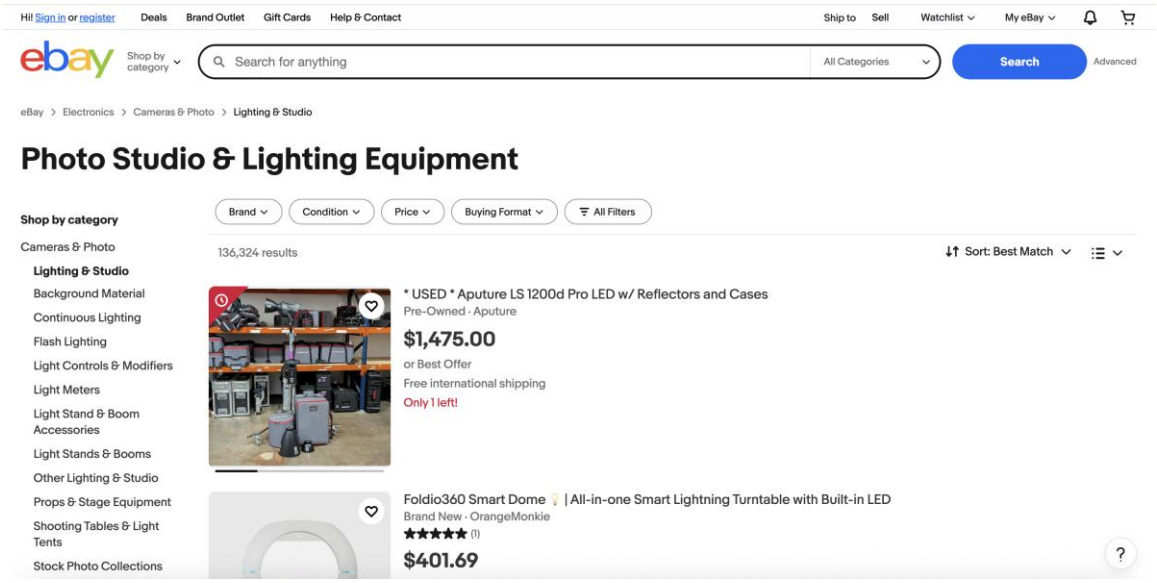


Рис. 1.4 Приклад екранної форми платформи eBay з категорією електроніки

Swappa є спеціалізованою платформою для роботи з уживаною технікою. На відміну від універсальних дошок оголошень, вона краще враховує специфіку електронних пристроїв, оскільки підтримує категорії смартфонів, ноутбуків, планшетів, ігрових консолей та іншої техніки.

Перевагою Swappa є структурованість сторінок пристроїв і орієнтація на електроніку. Водночас сервіс не надає повноцінного механізму обміну одного пристрою на інший [1; 2].

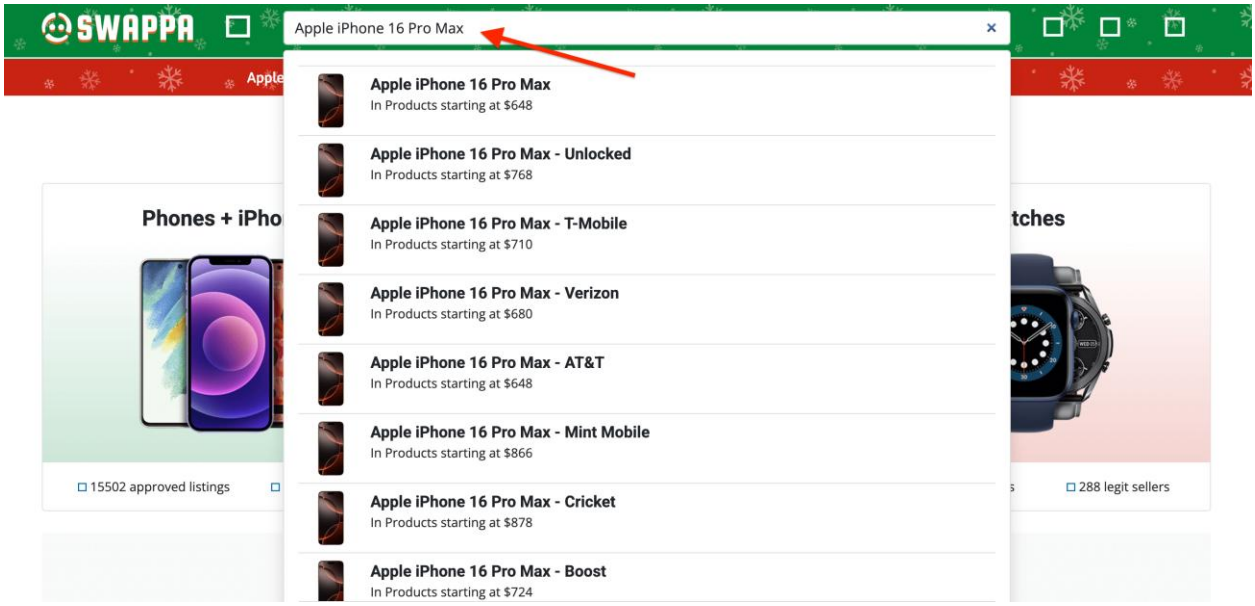


Рис. 1.5 Приклад екранної форми сервісу Swappa з переліком техніки

Shpock є локальним C2C-маркетплейсом, який дозволяє користувачам розміщувати оголошення та взаємодіяти з іншими учасниками. Його перевагою є орієнтація на локальний пошук і простий інтерфейс. Однак платформа не спеціалізується саме на техніці, тому деталізація характеристик пристроїв обмежена. Також відсутній окремий життєвий цикл пропозиції обміну [1; 2].

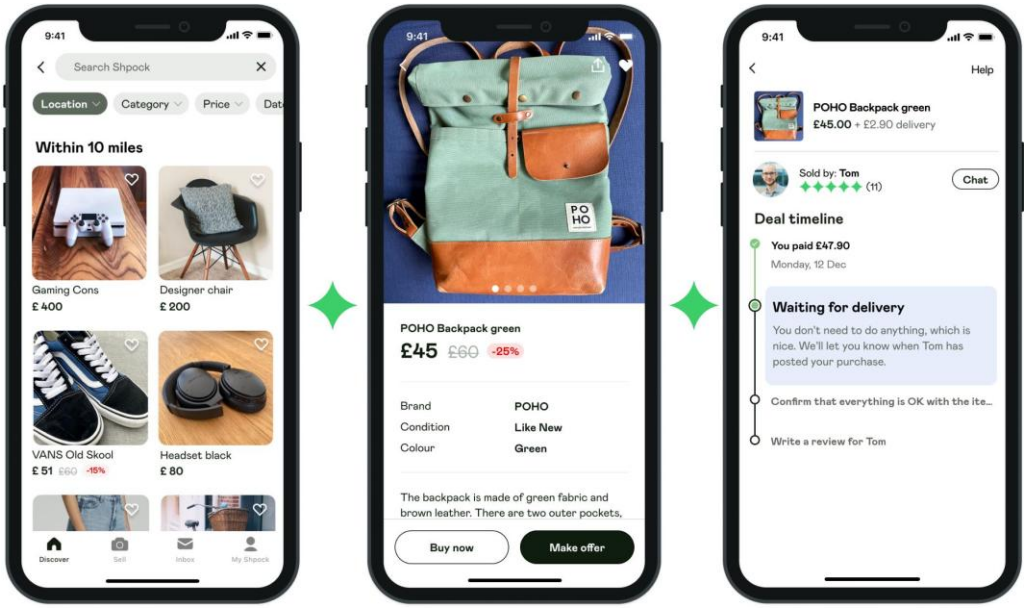


Рис. 1.6 Приклад екранної форми сервісу Shpock

Facebook Marketplace також може використовуватися для розміщення оголошень про техніку. Його перевагою є зв’язок із профілями користувачів та швидка комунікація через Messenger. Проте сервіс не має спеціалізованої форми пропозиції обміну, а умови домовленостей переважно обговорюються вручну [1; 2].

Таблиця 1.1

Порівняльна характеристика існуючих програмних засобів для обміну технікою між користувачами

Показник	**OLX**	**eBay**	**Swappa**	**Shpock**	**TechExchange**
Каталог пристроїв	+	+	+	+	+
Пошук і фільтрація	+	+	+	+	+
Детальна сторінка пристрою	+	+	+	+	+
Структурований опис характеристик	частково	+	+	частково	+
Фото пристрою	+	+	+	+	+

Продовження таблиці 1.1

Порівняльна характеристика існуючих програмних засобів для обміну технікою між користувачами

Показник	**OLX**	**eBay**	**Swappa**	**Shpock**	**TechExchange**
Вибір власного пристрою для обміну	-	-	-	-	+
Вбудований чат по конкретній угоді	частково	частково	+	+	+
Відстеження статусу домовленості	-	-	-	-	+
Відгуки після завершення обміну	частково	+	+	частково	+
Сповіщення про події	+	+	+	+	+
Адміністративний контроль	+	+	+	частково	+
Орієнтація саме на обмін технікою	частково	-	частково	частково	+

Проведений аналіз показує, що існуючі рішення мають окремі корисні функції, однак не забезпечують повного циклу обміну технікою між користувачами. Найчастіше вони виконують роль дошки оголошень або маркетплейсу, де користувачі можуть розмістити товар і почати спілкування, але подальша логіка домовленості залишається поза межами системи.

1.4 Обґрунтування доцільності розробки платформи TechExchange

На основі аналізу предметної галузі та існуючих програмних засобів можна зробити висновок, що розробка спеціалізованої платформи TechExchange є доцільною. Основна ідея системи полягає у створенні єдиного середовища, яке поєднує функції каталогу техніки, дошки оголошень, сервісу пропозицій обміну, вбудованого чату, системи статусів, відгуків і сповіщень.

На відміну від універсальних маркетплейсів, TechExchange орієнтована саме на P2P-обмін технікою. Це означає, що користувач не просто публікує оголошення, а може брати участь у повноцінному процесі обміну: створювати власні оголошення, переглядати пристрої інших користувачів, обирати власний пристрій для обміну, надсилати пропозицію, вести переговори та контролювати статус домовленості [1; 2].

Важливою перевагою майбутньої платформи є структурованість даних. Оголошення про пристрій повинно містити не лише довільний опис, а й конкретні поля: назву, категорію, стан, бренд, модель, місто, бажаний обмін, опис і зображення. Це спрощує пошук техніки та робить каталог більш зрозумілим для користувачів [1; 2].

Ще однією перевагою є підтримка життєвого циклу пропозиції обміну. У системі передбачено статуси домовленості, які дозволяють відстежувати поточний стан угоди: очікування, прийняття, передача пристрою, завершення, відхилення або скасування. Завдяки цьому користувачі бачать не лише факт повідомлення, а й реальний стан взаємодії [1; 2].

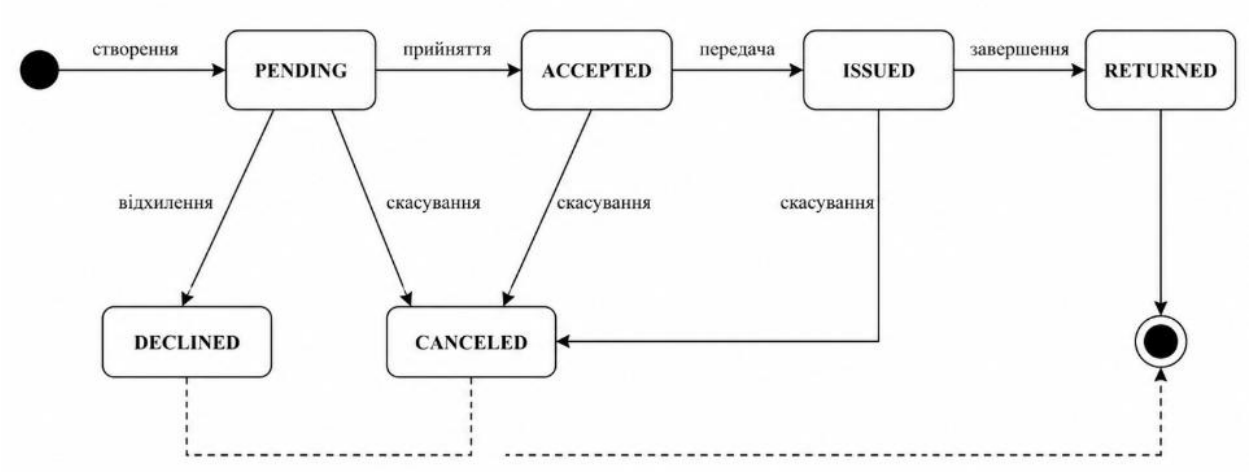


Рис. 1.7 Схема життєвого циклу пропозиції обміну на платформі TechExchange

Вбудований чат у межах конкретної пропозиції дозволяє зберігати комунікацію безпосередньо в системі. Це зменшує залежність від зовнішніх месенджерів і дозволяє пов’язати переговори з конкретною домовленістю. У разі виникнення спірних ситуацій така структура також є зручнішою для аналізу історії взаємодії [1; 2].

Додатково платформа передбачає відгуки після завершення обміну. Це допомагає формувати довіру між користувачами та оцінювати надійність учасників. Адміністративний модуль, у свою чергу, дозволяє контролювати активність користувачів і оголошень, що важливо для підтримання коректної роботи системи [1; 2].

Таким чином, розробка TechExchange дозволяє вирішити ключові проблеми предметної галузі: зменшити розрізненість комунікації, підвищити структурованість оголошень, спростити пошук техніки, надати механізм пропозиції обміну та забезпечити контроль статусу домовленості [1; 2].

1.5 Визначення вимог до програмного забезпечення

На основі аналізу предметної галузі та існуючих аналогів було сформовано основні вимоги до програмного забезпечення. Вимоги поділяються на функціональні та нефункціональні [7; 8].

Функціональні вимоги визначають перелік можливостей, які повинна забезпечувати платформа TechExchange. До основних функціональних вимог належать: [8; 9]

- реєстрація та авторизація користувачів із використанням JWT;
- перегляд поточного користувача та редагування профілю;
- створення, редагування, перегляд і видалення оголошень про техніку;
- додавання до оголошення назви, категорії, стану, бренду, моделі, міста, опису, фото та бажаного варіанту обміну;
- перегляд каталогу пристроїв;
- пошук і фільтрація оголошень за назвою, категорією, станом, містом і належністю до власних оголошень;
- перегляд детальної сторінки пристрою;
- створення пропозиції обміну з можливістю вибору власного пристрою;
- керування статусами домовленості;
- адміністративне керування користувачами й оголошеннями.

Нефункціональні вимоги визначають якісні характеристики системи. Для платформи TechExchange доцільно визначити такі нефункціональні вимоги:

- основні сторінки мають завантажуватися не довше 3 секунд;
- відповідь API для типових запитів має виконуватися до 1 секунди;
- пошук і фільтрація каталогу мають виконуватися до 2 секунд;
- приватні сторінки мають бути доступні лише авторизованим користувачам;
- паролі користувачів повинні зберігатися тільки у хешованому вигляді;
- база даних має забезпечувати зв’язки між користувачами, пристроями, пропозиціями обміну, повідомленнями, відгуками та сповіщеннями;
- система повинна підтримувати не менше 100 одночасних користувачів у тестовому середовищі;

Отже, платформа TechExchange повинна забезпечувати не лише базове розміщення оголошень, а й повний цикл взаємодії між користувачами під час обміну технікою. Саме це відрізняє її від існуючих універсальних рішень і визначає доцільність подальшого проєктування та реалізації системи.

2 ЗАСОБИ РЕАЛІЗАЦІЇ ПЛАТФОРМИ TECHEXCHANGE

2.1 Обґрунтування вибору клієнт-серверної архітектури

Для реалізації платформи TechExchange було обрано клієнт-серверну архітектуру, оскільки система передбачає одночасну роботу з користувацьким інтерфейсом, серверною бізнес-логікою, базою даних, автентифікацією, обміном повідомленнями, сповіщеннями та адміністративними функціями. Такий підхід дозволяє розділити відповідальність між окремими частинами програмного забезпечення та забезпечити більш зручну підтримку системи в майбутньому [12; 13; 18].

Клієнтська частина відповідає за взаємодію з користувачем. Саме через web-інтерфейс користувач реєструється, входить у систему, переглядає каталог пристроїв, створює оголошення, надсилає пропозицію обміну, веде чат, переглядає повідомлення та редагує профіль. Завданням клієнтської частини є зручне представлення даних і передавання запитів до серверної частини.

Серверна частина відповідає за обробку запитів, перевірку прав доступу, виконання бізнес-логіки та взаємодію з базою даних. Наприклад, саме backend перевіряє, чи має користувач право редагувати оголошення, чи може він створити пропозицію обміну, чи допустима зміна статусу домовленості, чи можна залишити відгук після завершення обміну.

База даних використовується для збереження основної інформації платформи: користувачів, оголошень про техніку, пропозицій обміну, повідомлень, відгуків і сповіщень. Винесення даних в окремий рівень дозволяє забезпечити їх цілісність, підтримувати зв’язки між сутностями та виконувати пошук і фільтрацію [12; 15; 16].

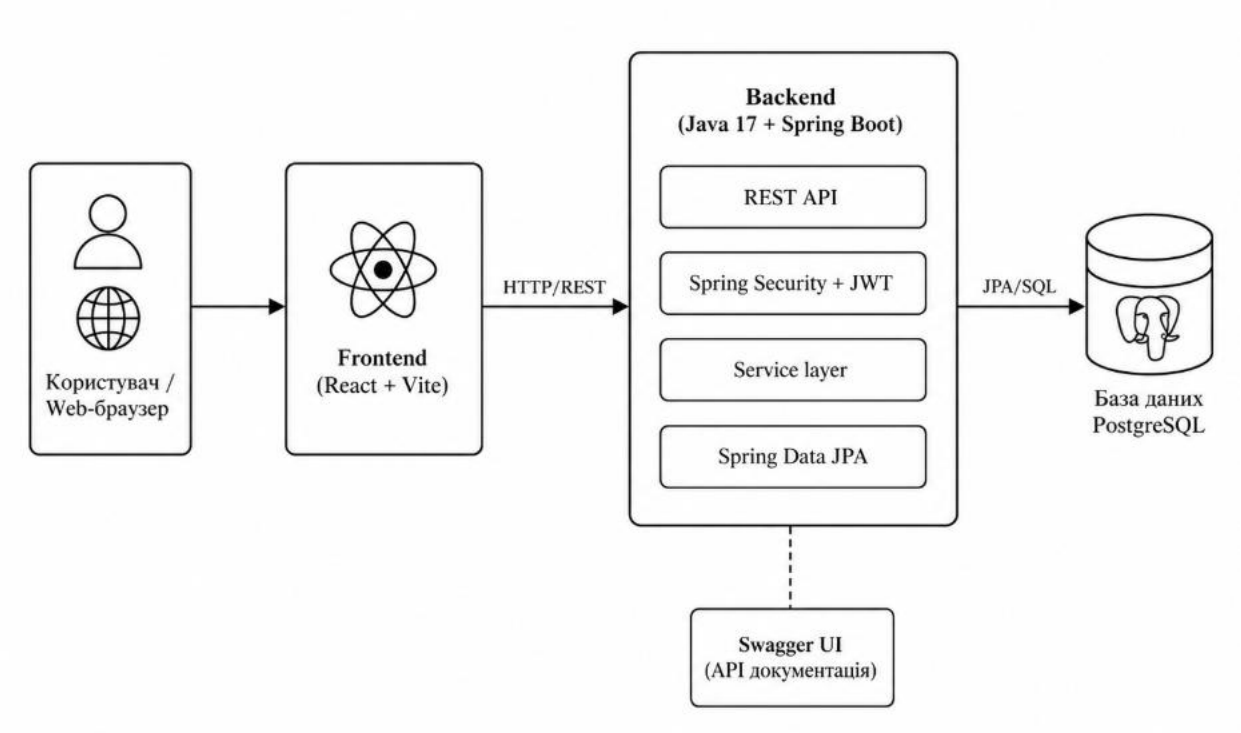


Рис. 2.1 Схема клієнт-серверної архітектури платформи TechExchange

Клієнт-серверна модель є доцільною для TechExchange ще й тому, що система має захищені маршрути та рольову модель доступу. Неавторизовані користувачі можуть перейти лише на сторінки входу та реєстрації, а всі інші функції доступні після перевірки JWT-токена. Окремі адміністративні можливості, такі як керування користувачами та оголошеннями, доступні лише користувачам із роллю ADMIN.

Завдяки такому підходу система стає більш гнучкою. У майбутньому до неї можна додати мобільний застосунок, не змінюючи основну серверну логіку, оскільки frontend і backend взаємодіють через REST API. Також клієнтська та серверна частини можуть розвиватися окремо, що спрощує модернізацію проєкту.

2.2 Засоби реалізації серверної частини

Серверну частину платформи TechExchange реалізовано мовою програмування Java 17 із використанням фреймворку Spring Boot. Java є зручною для побудови серверних застосунків, оскільки підтримує об’єктно-орієнтований

підхід, дозволяє чітко описувати доменні сутності та забезпечує стабільну роботу великих клієнт-серверних систем [14].

У межах проєкту Java використовується для реалізації основної бізнес-логіки платформи. За допомогою класів-сутностей описано користувачів, пристрої, пропозиції обміну, повідомлення, відгуки та сповіщення. Сервісні класи відповідають за обробку основних операцій, а контролери надають REST API для клієнтської частини [15].

Spring Boot використовується для спрощення створення серверного web-застосунку. Він дозволяє швидко налаштувати запуск backend-частини, підключити необхідні модулі, реалізувати REST-контролери, роботу з базою даних, безпеку та документацію API. У проєкті TechExchange Spring Boot є основою серверної частини.

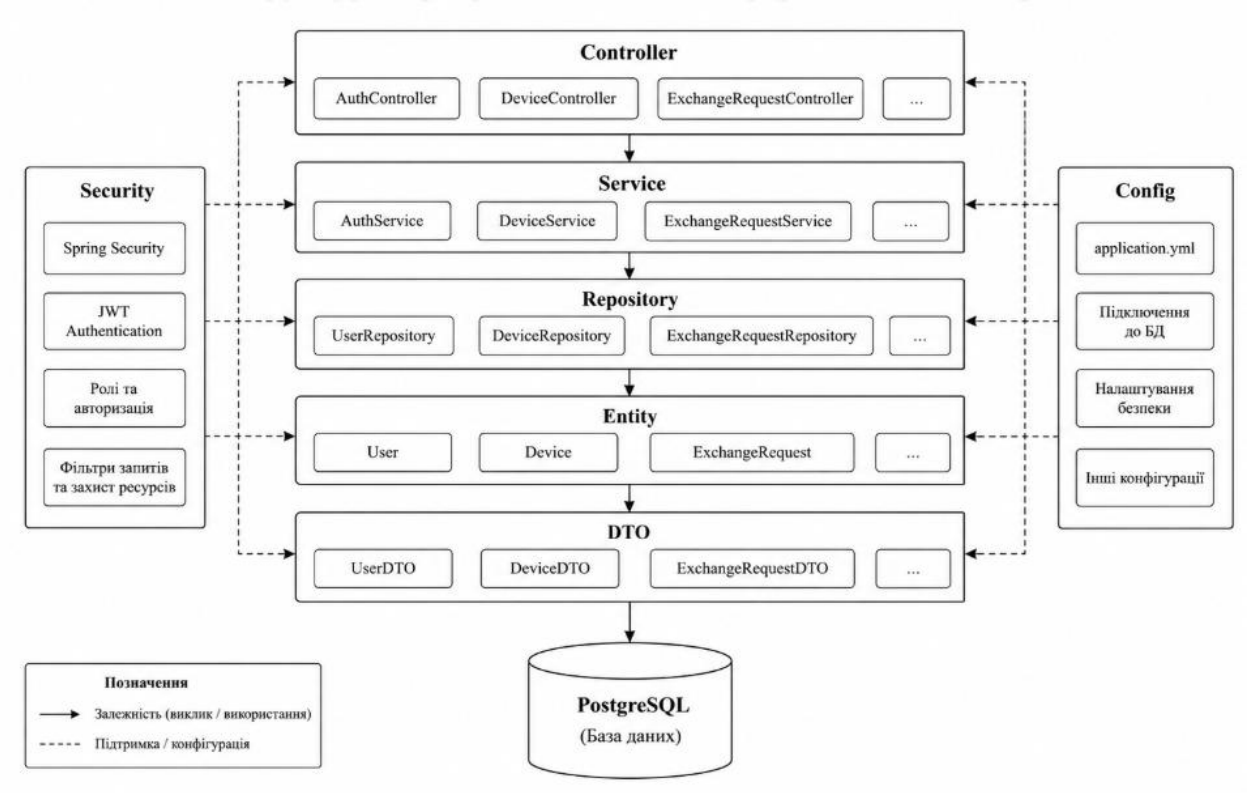


Рис. 2.2 Структура серверної частини платформи TechExchange

Для створення web API використано Spring Web. За допомогою REST-контролерів реалізовано endpoint-и для авторизації, роботи з користувачами,

оголошеннями, пропозиціями обміну, повідомленнями, відгуками, сповіщеннями та адміністративними функціями. Такий підхід дозволяє клієнтській частині звертатися до backend через HTTP-запити та отримувати відповіді у форматі JSON [17].

Для роботи з базою даних використано Spring Data JPA. Цей засіб дозволяє описувати репозиторії для доступу до даних без необхідності вручну писати всі SQL-запити. У проєкті створено репозиторії для основних сутностей: користувачів, пристроїв, пропозицій обміну, повідомлень, відгуків і сповіщень. Через репозиторії сервісний шар отримує, зберігає та оновлює дані [3; 5; 16].

Для реалізації безпеки використано Spring Security. У платформі передбачена реєстрація та авторизація користувачів, а також обмеження доступу до приватних маршрутів. Після входу в систему користувач отримує JWT-токен, який надалі передається в запитах до серверної частини. Backend перевіряє токен і визначає, чи має користувач право виконувати певну дію [3; 5; 16].

Окрему роль відіграє JWT-аутентифікація. Вона дозволяє не зберігати сесію користувача на сервері, а передавати інформацію про авторизованого користувача через токен. Це зручно для REST API, оскільки кожен запит може бути перевірений незалежно [16; 22].

У серверній частині також використовується єдиний підхід до обробки помилок. Якщо користувач намагається виконати заборонену дію, звернутися до неіснуючого ресурсу або передати некоректні дані, система повинна повернути зрозумілу відповідь про помилку. Це підвищує передбачуваність роботи API та спрощує обробку помилок на frontend-частині.

2.3 Засоби реалізації клієнтської частини

Клієнтську частину платформи TechExchange реалізовано з використанням React 18 та Vite. React дозволяє створювати web-інтерфейс на основі компонентів, що є зручним для побудови багатосторінкового інтерфейсу платформи. Кожна частина інтерфейсу може бути реалізована як окремий компонент: форма входу,

форма реєстрації, картка пристрою, каталог, фільтри, модальне вікно створення оголошення, сторінка домовленостей, чат і профіль користувача [20].

Vite використовується як інструмент для запуску та збирання frontend-застосунку. Його перевагою є швидкий старт проєкту, зручна робота під час розробки та оптимізоване збирання клієнтської частини. Для платформи TechExchange це дозволяє швидко оновлювати інтерфейс під час розробки та зручно організовувати структуру frontend-проєкту [20].

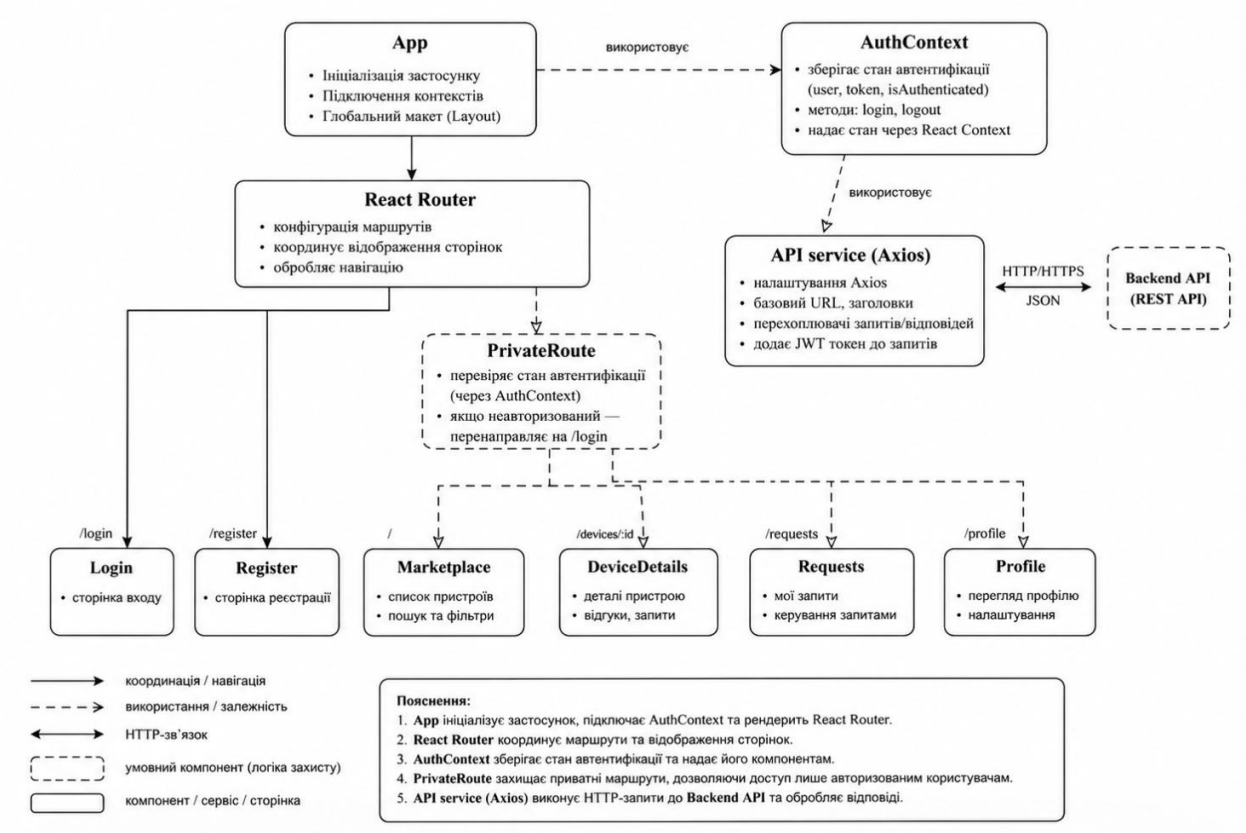


Рис. 2.3 Структура клієнтської частини платформи TechExchange

Для організації переходів між сторінками використано React Router. У проєкті передбачено маршрути для входу, реєстрації, головної сторінки, детальної сторінки пристрою, сторінки пропозицій обміну та профілю користувача. Завдяки маршрутизації користувач може переміщатися між різними частинами платформи без повного перезавантаження сторінки [20].

Особливістю frontend-частини є використання захищених маршрутів. Усі сторінки, крім входу та реєстрації, доступні лише авторизованому користувачу.

Для цього використовується PrivateRoute, який перевіряє наявність JWT-токена та дозволяє або забороняє доступ до сторінки [20].

Для виконання HTTP-запитів до серверної частини використовується Axios. За його допомогою frontend надсилає запити на авторизацію, отримання списку пристроїв, створення оголошення, формування пропозиції обміну, завантаження повідомлень, створення відгуку та інші операції. Axios також дозволяє централізовано додавати JWT-токен до запитів і обробляти помилки відповіді сервера [20].

Для зберігання інформації про поточного користувача та токен використовується AuthContext. Він дозволяє різним компонентам застосунку отримувати доступ до даних авторизації без дублювання логіки. Якщо сервер повертає помилку 401, токен може бути очищений, а користувач — перенаправлений на сторінку входу [20].

Інтерфейс платформи містить такі основні сторінки: Login, Register, Marketplace, DeviceDetails, Requests і Profile. Сторінка Marketplace виконує роль головної сторінки та каталогу пристроїв. На ній користувач може переглядати оголошення, застосовувати фільтри, створювати й редагувати власні оголошення. Сторінка DeviceDetails дозволяє переглянути повну інформацію про пристрій і надіслати пропозицію обміну. Сторінка Requests використовується для роботи з домовленостями, зміни статусів і ведення чату [20].

Для візуального оформлення інтерфейсу використано Lucide React. Ця бібліотека містить набір іконок, які допомагають зробити інтерфейс більш зрозумілим і зручним для користувача. Іконки використовуються для позначення дій, навігації, статусів, профілю та інших елементів.

2.4 Засоби збереження та обробки даних

Для збереження даних платформи TechExchange використано PostgreSQL. Це реляційна система керування базами даних, яка добре підходить для проєктів зі складними зв’язками між сутностями. У даній системі такі зв’язки є важливими,

оскільки користувачі пов’язані з оголошеннями, оголошення — з пропозиціями обміну, пропозиції — з повідомленнями, відгуками та сповіщеннями [13; 18].

У базі даних платформи передбачено основні таблиці: users, devices, exchange_requests, messages, reviews і notifications. Таблиця users зберігає дані про користувачів, їхні ролі та статус активності. Таблиця devices містить інформацію про техніку, яку користувачі публікують для обміну. Таблиця exchange_requests відповідає за домовленості між користувачами. Таблиця messages зберігає повідомлення у межах конкретної домовленості. Таблиця reviews використовується для відгуків після завершення обміну, а notifications — для системних сповіщень [18].

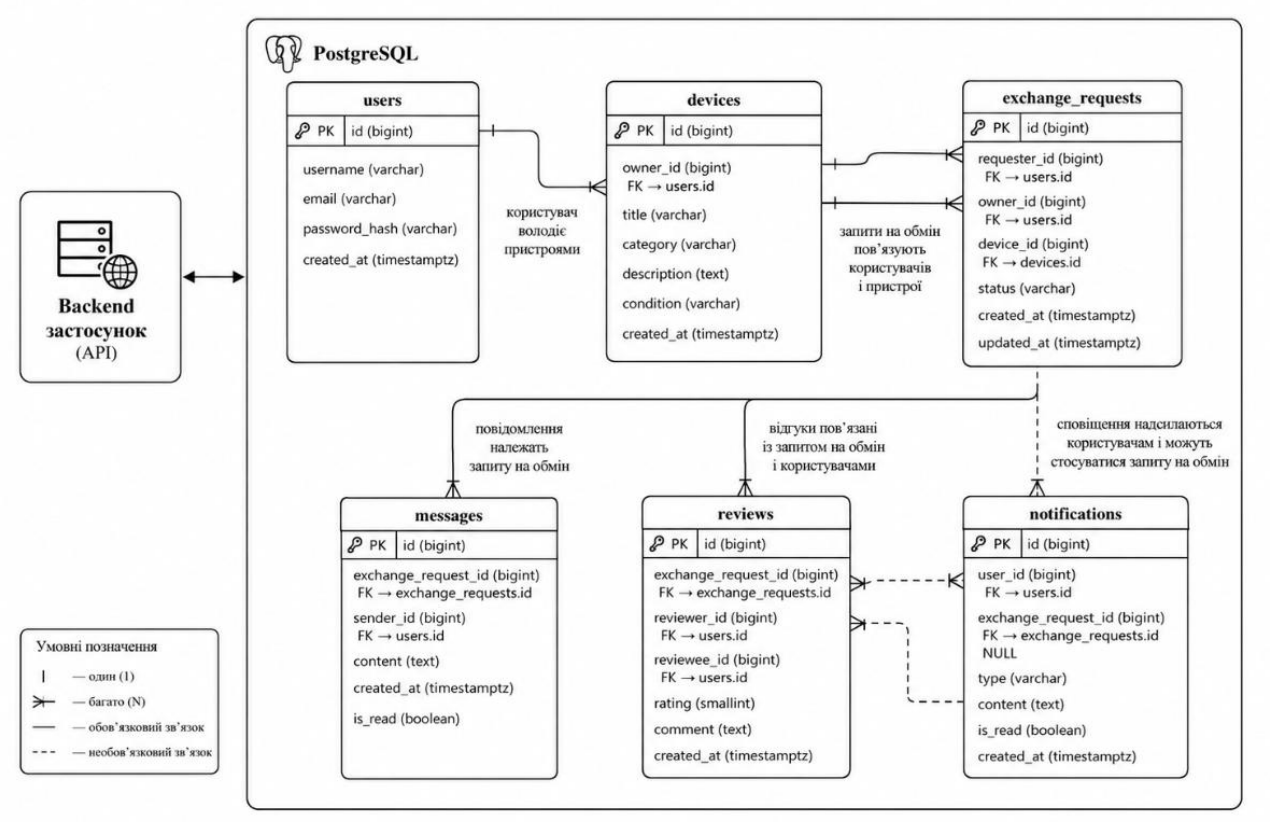


Рис. 2.4 Схема збереження даних платформи TechExchange

Для керування змінами структури бази даних використано Liquibase. Це дозволяє описувати міграції бази даних у вигляді окремих changelog-файлів і послідовно застосовувати їх під час запуску системи. У проєкті міграції використовуються для створення таблиць, зв’язків, обмежень та індексів [19].

Liquibase є важливим для стабільності розробки, оскільки структура бази даних може змінюватися в процесі реалізації системи. Наприклад, до таблиць можуть додаватися нові поля, індекси, обмеження або зв’язки. Використання міграцій дозволяє контролювати ці зміни та відтворювати однакову структуру бази на різних середовищах [19].

Для зв’язку між Java-класами та таблицями бази даних використовується JPA та Hibernate. Сутності User, Device, ExchangeRequest, Message, Review і Notification відповідають основним таблицям бази даних. Поля класів відображаються на поля таблиць, а зв’язки між сутностями описуються через анотації [17; 18].

Важливим аспектом збереження даних є підтримка цілісності. Наприклад, кожен пристрій має власника, кожна пропозиція обміну пов’язана з цільовим пристроєм і користувачами, а кожне повідомлення належить конкретній пропозиції обміну. Завдяки зовнішнім ключам система зберігає логічні зв’язки між даними [13; 18].

Для підвищення швидкості роботи з каталогом і домовленостями використовуються індекси. Зокрема, індекси доцільні для полів, які часто використовуються під час фільтрації або сортування: категорії пристрою, статусу, власника, дати оновлення та зв’язку повідомлень із конкретною пропозицією обміну [18].

2.5 Додаткові інструменти безпеки, документування та розгортання

Окрім основних засобів реалізації серверної та клієнтської частин, у проєкті TechExchange використовуються додаткові інструменти, які забезпечують безпеку, документування API, запуск системи та контроль якості розробки.

Одним із ключових елементів є JWT-захист. Після успішної авторизації користувач отримує токен, який надалі використовується для доступу до захищених endpoint-ів. Такий механізм дозволяє відокремити публічні маршрути від приватних і гарантувати, що дії з оголошеннями, пропозиціями обміну, повідомленнями та профілем виконує саме авторизований користувач.

Рольова модель доступу передбачає дві основні ролі: USER та ADMIN. Звичайний користувач може створювати оголошення, надсилати пропозиції обміну, вести чат, залишати відгуки та переглядати власний профіль. Адміністратор має додаткові права для керування користувачами та оголошеннями, зокрема може змінювати роль, активність користувача або активність оголошення [3; 5; 16].

Для документування REST API використано Swagger UI на основі OpenAPI. Це дозволяє переглядати доступні endpoint-и, методи запитів, параметри, структуру відповідей і тестувати API через web-інтерфейс. Наявність Swagger UI є зручною як для розробника frontend-частини, так і для тестування backend-логіки.

Для розгортання системи використано Docker та Docker Compose. Docker дозволяє запускати компоненти системи в ізольованих контейнерах, а Docker Compose дає змогу описати спільний запуск backend, frontend і PostgreSQL. Такий підхід спрощує запуск проєкту на іншому комп’ютері та зменшує залежність від локальних налаштувань середовища [22].

Для тестування застосовуються JUnit 5, Testcontainers, MockMvc та Spring Security Test. JUnit 5 використовується для написання тестів, Testcontainers дозволяє запускати тестову PostgreSQL-базу в контейнері, MockMvc використовується для перевірки REST API, а Spring Security Test допомагає перевіряти захищені endpoint-и та доступ користувачів [21].

Таким чином, обрані засоби реалізації відповідають задачам розробки платформи TechExchange. Java 17, Spring Boot і Spring Security забезпечують надійну серверну частину, PostgreSQL і Liquibase — стабільне збереження та розвиток структури даних, React і Vite — зручний клієнтський інтерфейс, а Swagger UI, Docker Compose та засоби тестування підвищують якість розробки, документування й розгортання системи [10].

3 ПРОЄКТУВАННЯ ПЛАТФОРМИ TECHEXCHANGE

3.1 Загальна архітектура програмного забезпечення

Платформа TechExchange спроєктована як клієнт-серверний web-застосунок, у якому функції інтерфейсу користувача, серверної логіки та збереження даних розділено між окремими компонентами. Такий підхід дозволяє забезпечити зручну підтримку системи, чітко розмежувати відповідальність між частинами програмного забезпечення та спростити подальше розширення функціоналу.

Загальна архітектура платформи включає три основні рівні: клієнтську частину, серверну частину та базу даних. Клієнтська частина реалізована на React і відповідає за відображення сторінок, форм, каталогу пристроїв, модальних вікон, повідомлень, статусів домовленостей і профілю користувача. Серверна частина реалізована на Java з використанням Spring Boot і відповідає за обробку запитів, перевірку прав доступу, виконання бізнес-логіки та взаємодію з базою даних. База даних PostgreSQL використовується для збереження інформації про користувачів, оголошення, пропозиції обміну, повідомлення, відгуки та сповіщення [12; 13].

Серверна частина побудована за принципом багаторівневої архітектури. У проєкті виділено такі основні шари: [12; 13; 15]

- controller — приймає HTTP-запити від клієнтської частини та повертає відповіді;
- service — містить бізнес-логіку системи;
- repository — забезпечує доступ до бази даних через Spring Data JPA;
- entity — описує основні доменні сутності системи;
- dto — використовується для передачі даних між клієнтом і сервером;
- security — відповідає за JWT-аутентифікацію та перевірку доступу;
- config — містить конфігурацію безпеки, OpenAPI та початкових даних.

Такий поділ дозволяє уникнути змішування логіки контролерів, бізнес-правил і роботи з базою даних. Наприклад, контролер лише приймає запит на

створення оголошення, сервіс перевіряє правила створення та формує об’єкт, а репозиторій виконує збереження в базі даних.

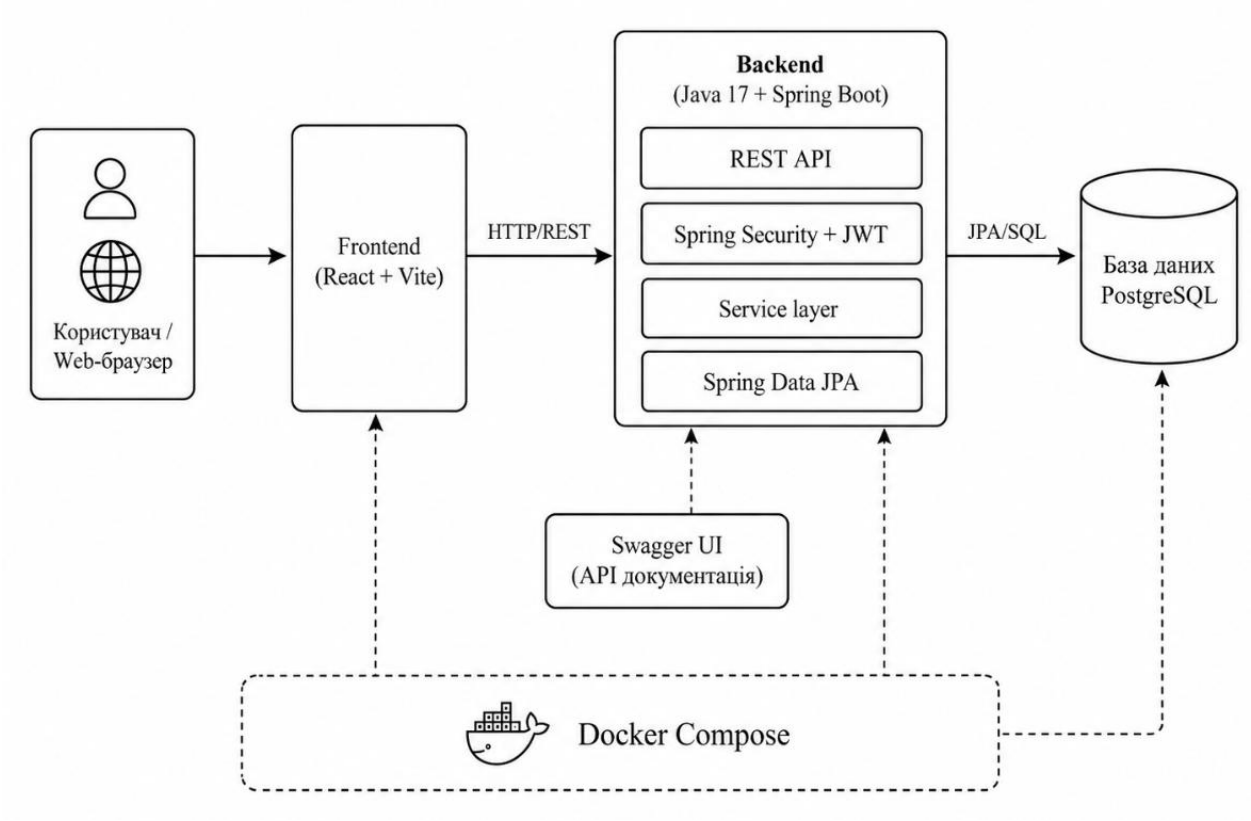


Рис. 3.1 Загальна архітектура платформи TechExchange

Клієнтська частина також має власну логічну структуру. Основними сторінками платформи є сторінка входу, сторінка реєстрації, головна сторінка з каталогом пристроїв, сторінка деталей пристрою, сторінка пропозицій обміну та сторінка профілю. Доступ до приватних сторінок обмежено через механізм PrivateRoute, який перевіряє наявність JWT-токена.

Основна взаємодія між frontend і backend відбувається через REST API. Клієнтська частина надсилає HTTP-запити до серверної частини, а сервер повертає дані у форматі JSON. Наприклад, під час відкриття каталогу frontend надсилає запит на отримання списку пристроїв, backend виконує фільтрацію та повертає перелік оголошень. Під час створення пропозиції обміну frontend передає ідентифікатори пристроїв і повідомлення, а backend перевіряє правила обміну та створює новий запис у базі даних [15; 20].

У системі реалізовано рольову модель доступу. Звичайний користувач має можливість створювати оголошення, переглядати каталог, надсилати пропозиції обміну, вести чат, залишати відгуки та переглядати власний профіль. Адміністратор має додаткові права для керування користувачами й оголошеннями, зокрема може змінювати активність користувачів і блокувати некоректні оголошення [15; 18; 22].

3.2 Проєктування структури бази даних

База даних платформи TechExchange спроєктована з урахуванням основних процесів системи: реєстрації користувачів, публікації оголошень про техніку, створення пропозицій обміну, обміну повідомленнями, формування відгуків і надсилання сповіщень. Для реалізації збереження даних використано реляційну модель, оскільки сутності системи мають чіткі взаємозв’язки між собою.

Основними таблицями бази даних є:

– users;
– devices;
– exchange_requests;
– messages;
– reviews;
– notifications.

Таблиця users зберігає інформацію про користувачів платформи. До її основних полів належать id, email, password_hash, full_name, role, active та created_at. Поле email є унікальним, оскільки воно використовується для авторизації. Поле password_hash зберігає пароль у захищеному хешованому вигляді. Поле role визначає роль користувача в системі, а active дозволяє адміністратору деактивувати обліковий запис.

Таблиця devices використовується для збереження оголошень про техніку. Вона містить інформацію про власника пристрою, назву, категорію, стан, статус, бренд, модель, місто, бажаний варіант обміну, опис, зображення та ознаку

активності. Поле owner_id є зовнішнім ключем на таблицю users і визначає користувача, який створив оголошення.

Таблиця exchange_requests описує пропозиції обміну між користувачами. Вона містить посилання на цільовий пристрій, пристрій, який пропонується в обмін, ініціатора заявки та власника цільового пристрою. Також у таблиці зберігається статус домовленості, початкове повідомлення та дати створення й оновлення.

Таблиця messages використовується для збереження повідомлень у межах конкретної пропозиції обміну. Кожне повідомлення пов’язане з exchange_request та користувачем-відправником. Така структура дозволяє зберігати чат не загально між двома користувачами, а саме в контексті конкретної домовленості.

Таблиця reviews відповідає за відгуки після завершення обміну. Вона містить інформацію про пропозицію обміну, автора відгуку, користувача, якому залишено відгук, оцінку та коментар. Для таблиці передбачено обмеження унікальності, яке не дозволяє одному користувачу залишити більше одного відгуку на одну домовленість.

Таблиця notifications використовується для системних сповіщень. Вона зберігає одержувача, тип події, заголовок, текст сповіщення, ознаку прочитання та дату створення. Сповіщення можуть створюватися при новій заявці, зміні статусу домовленості, новому повідомленні або новому відгуку.

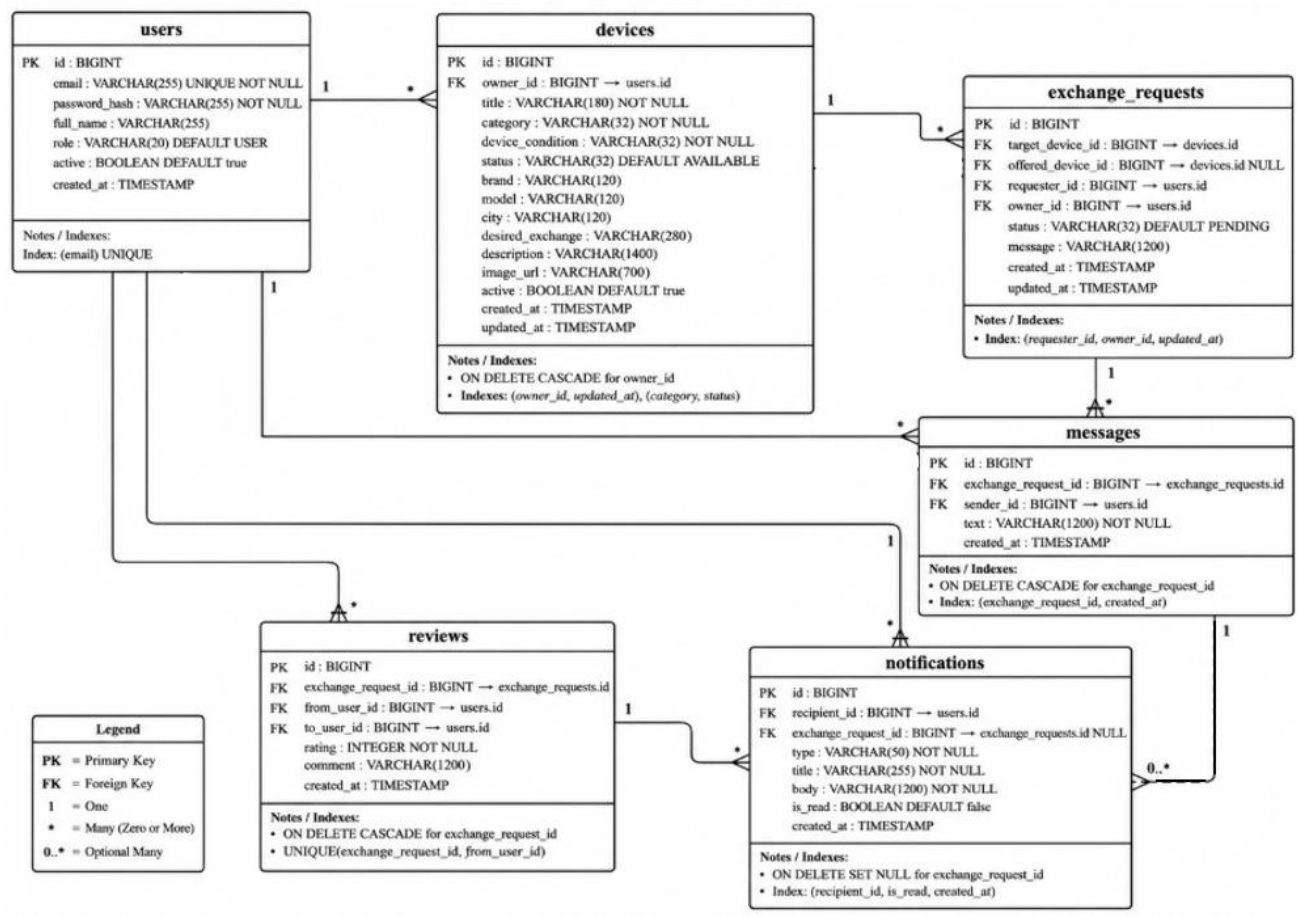


Рис. 3.2 Структура бази даних платформи TechExchange

Для підвищення продуктивності в базі даних передбачено індекси. Зокрема, індекси використовуються для швидкого пошуку пристроїв за власником, категорією та статусом, а також для сортування повідомлень і пропозицій обміну за датою оновлення. Це важливо для роботи каталогу, фільтрації та сторінки домовленостей [14; 18].

Структура бази даних також враховує необхідність логічного видалення окремих об’єктів. Наприклад, оголошення або користувач можуть бути деактивовані через поле active без фізичного видалення з бази. Такий підхід дозволяє зберігати історію взаємодії та уникати порушення зв’язків між таблицями [18].

3.3 Проєктування доменної моделі системи

Доменна модель платформи TechExchange відображає основні об’єкти предметної галузі та зв’язки між ними. Центральними сутностями системи є User, Device, ExchangeRequest, Message, Review та Notification [18].

Сутність User описує користувача платформи. Вона містить ідентифікатор, email, хеш пароля, повне ім’я, роль, статус активності та дату створення. Користувач може створювати оголошення, надсилати пропозиції обміну, бути власником пристрою, надсилати повідомлення, залишати відгуки та отримувати сповіщення [18].

Сутність Device описує пристрій, який користувач публікує для обміну. Вона містить назву, категорію, стан, статус, бренд, модель, місто, бажаний варіант обміну, опис і зображення. Пристрій має власника, а його статус може змінюватися залежно від стану домовленості: доступний, зарезервований або обміняний [18].

Сутність ExchangeRequest є однією з ключових у системі, оскільки вона описує саму пропозицію обміну. Вона пов’язує користувача-ініціатора, власника цільового пристрою, пристрій, на який хочуть обміняти, і пристрій, який пропонують. У деяких випадках offeredDevice може бути відсутнім, якщо користувач надсилає відкриту пропозицію.

Сутність Message описує повідомлення в чаті. Кожне повідомлення пов’язане з конкретною пропозицією обміну та користувачем, який його надіслав. Завдяки цьому всі переговори зберігаються в межах конкретної домовленості [8; 9].

Сутність Review використовується для оцінювання користувачів після завершення обміну. Вона містить автора відгуку, одержувача, оцінку, коментар і посилання на відповідну пропозицію обміну. Це дозволяє формувати рейтинг користувачів і підвищувати довіру між учасниками платформи.

Сутність Notification описує системні сповіщення. Вона містить одержувача, тип події, заголовок, текст, ознаку прочитання та дату створення. Сповіщення допомагають користувачеві не пропускати важливі події, пов’язані з пропозиціями, повідомленнями або відгуками.

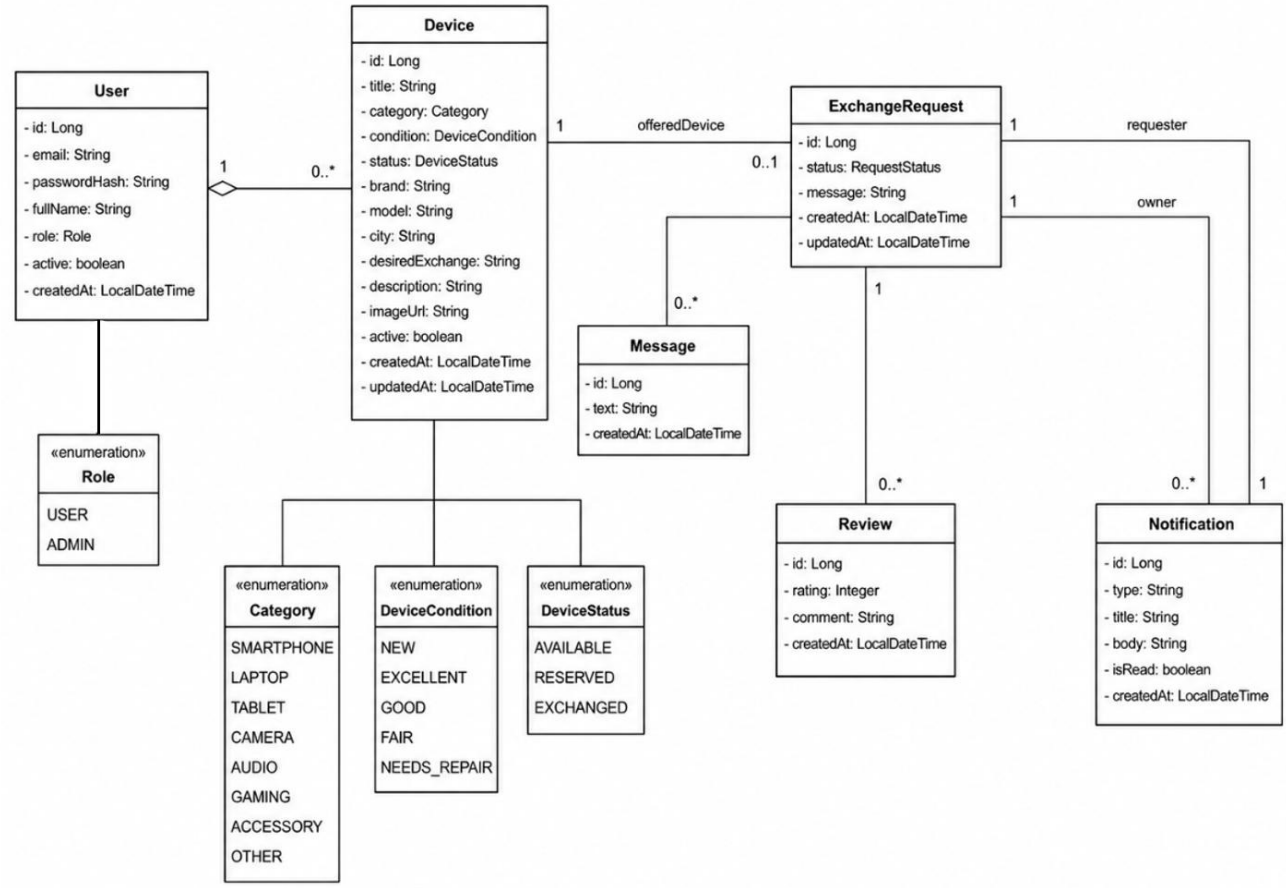


Рис. 3.3 Діаграма доменної моделі платформи TechExchange

У доменній моделі також використовуються переліки значень, які обмежують допустимі стани системи. Наприклад, роль користувача може бути USER або ADMIN. Категорія пристрою може визначати тип техніки: смартфон, ноутбук, планшет, камера, аудіотехніка, ігровий пристрій, аксесуар або інше. Стан пристрою може бути новим, відмінним, добрим, задовільним або таким, що потребує ремонту [1; 2].

Окреме значення має перелік статусів пропозиції обміну. Він визначає життєвий цикл домовленості: PENDING, ACCEPTED, ISSUED, RETURNED, DECLINED або CANCELED. Завдяки цьому система може контролювати допустимі переходи між станами та не дозволяти виконувати нелогічні дії.

3.4 Проєктування логіки роботи користувача на платформі

Логіка роботи користувача на платформі TechExchange починається з авторизації. Якщо користувач ще не має облікового запису, він може перейти на сторінку реєстрації та створити акаунт. Після успішної реєстрації або входу користувач отримує доступ до основних функцій платформи [1; 2].

Після авторизації користувач переходить на головну сторінку, де відображається каталог пристроїв. На цій сторінці він може переглядати оголошення, використовувати пошук і фільтри, відкривати детальну інформацію про пристрій, а також створювати власні оголошення. Каталог є основною точкою входу в процес обміну [1; 2].

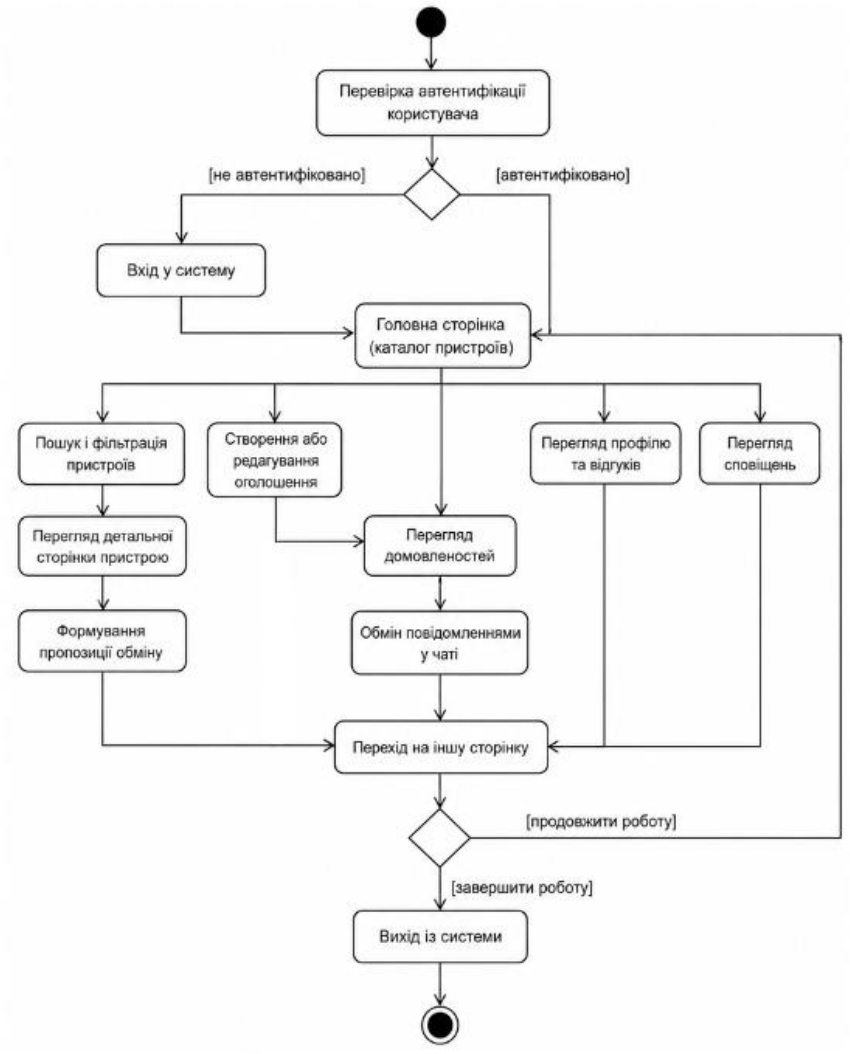


Рис. 3.5 Діаграма активності роботи користувача на платформі TechExchange

Якщо користувач знаходить цікавий пристрій, він відкриває сторінку деталей. На цій сторінці відображається повна інформація про техніку: назва, категорія, стан, бренд, модель, місто, опис, фото та бажаний варіант обміну. Також користувач може створити пропозицію обміну, обравши один зі своїх пристроїв і додавши повідомлення [1; 2].

Після створення пропозиції обміну вона з’являється на сторінці домовленостей. Тут користувач може бачити вхідні й вихідні заявки, поточний статус кожної домовленості, інформацію про пристрої та чат. Залежно від ролі у конкретній домовленості користувач може прийняти, відхилити або скасувати пропозицію.

Життєвий цикл домовленості має чітко визначені статуси. Початковим станом є PENDING. Якщо власник цільового пристрою погоджується на обмін, заявка переходить у статус ACCEPTED. Після передачі пристроїв статус може бути змінено на ISSUED, а після остаточного завершення — на RETURNED. Також пропозиція може бути відхилена або скасована.

Під час зміни статусу система виконує додаткові дії. Наприклад, при прийнятті пропозиції пристрої можуть резервуватися, а конкуруючі заявки автоматично відхилятися. Після завершення обміну користувачі можуть залишити відгуки один одному [14-18].

Окремо передбачено роботу зі сповіщеннями. Користувач отримує повідомлення про створення нової заявки, зміну статусу домовленості, нове повідомлення в чаті або новий відгук. Це дозволяє оперативно реагувати на події без необхідності постійно перевіряти всі розділи платформи.

3.5 Проєктування класів серверної частини

Діаграма класів серверної частини платформи TechExchange відображає основні класи, які забезпечують роботу системи. Для дипломної роботи доцільно показати ключові класи, які представляють сутності, репозиторії та сервіси.

До основних entity-класів належать User, Device, ExchangeRequest, Message, Review і Notification. Вони відповідають таблицям бази даних та описують основні об’єкти предметної галузі. Кожен клас містить поля, які відображають властивості відповідної сутності.

Клас User описує користувача системи. Клас Device відповідає за оголошення про техніку. Клас ExchangeRequest описує пропозицію обміну. Клас Message використовується для повідомлень у чаті. Клас Review зберігає відгуки після завершення обміну. Клас Notification відповідає за системні сповіщення [18].

Репозиторії забезпечують доступ до бази даних. UserRepository використовується для пошуку користувача за email і перевірки існування email. DeviceRepository забезпечує отримання пристроїв, пошук за власником, підрахунок оголошень та статистику. ExchangeRequestRepository використовується для отримання заявок користувача та пошуку заявок за пристроєм і статусом. ReviewRepository дозволяє отримувати відгуки користувача, перевіряти наявність відгуку та розраховувати середній рейтинг [18].

Сервісний шар містить основну бізнес-логіку. UserService відповідає за отримання поточного користувача, оновлення профілю та формування даних профілю. DeviceService реалізує роботу з оголошеннями, каталогом, фільтрацією та статистикою. ExchangeRequestService відповідає за створення заявок, зміну статусів, резервування пристроїв і завершення обміну [18].

Також важливими сервісами є MessageService, ReviewService і NotificationService. MessageService забезпечує створення та отримання повідомлень у межах конкретної домовленості. ReviewService контролює створення відгуків тільки після завершеного обміну. NotificationService створює сповіщення для користувачів при важливих подіях [18].

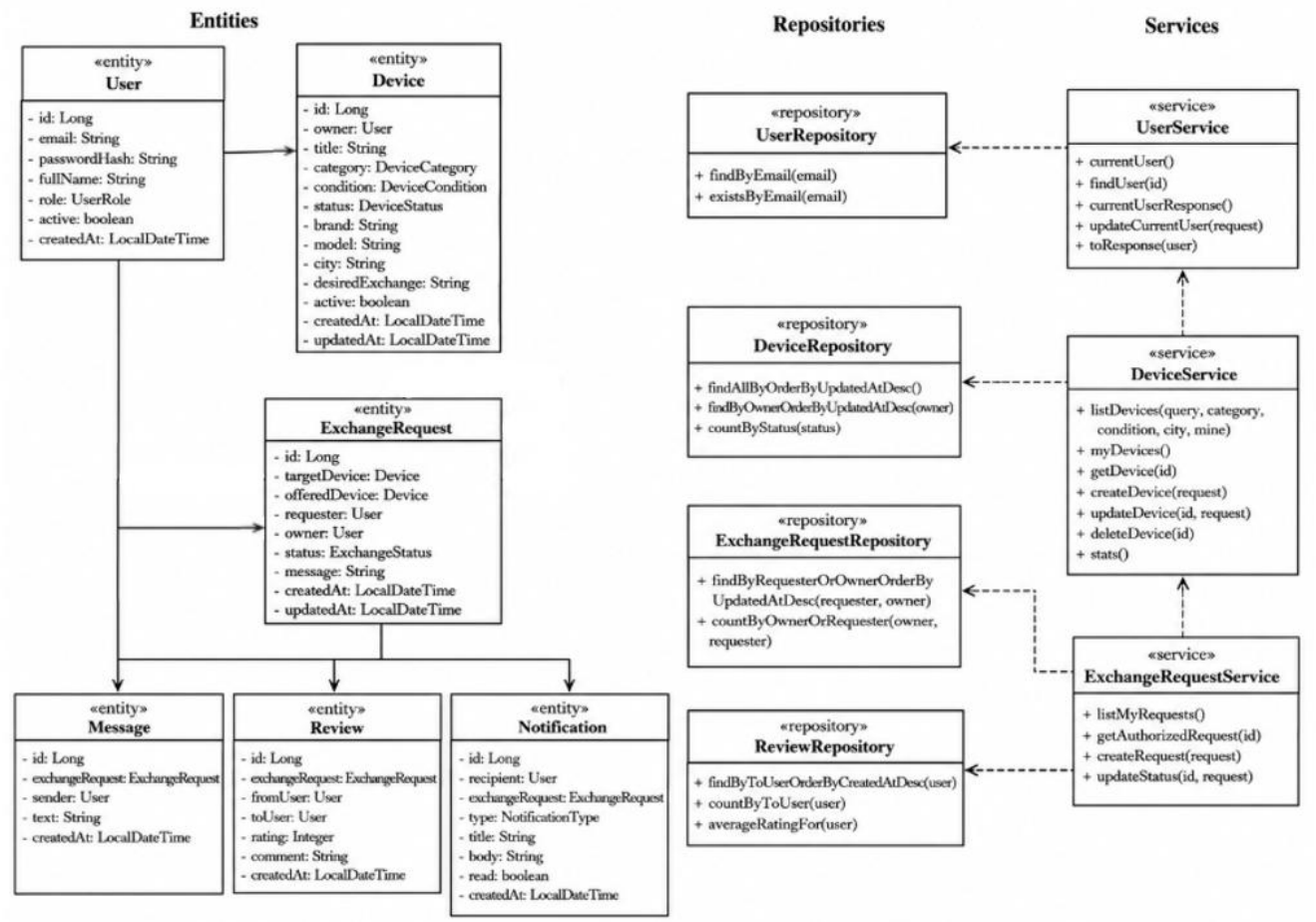


Рис. 3.6 Діаграма класів платформи TechExchange

Діаграма класів показує, що система має чітке розділення відповідальності. Entity-класи описують дані, репозиторії відповідають за доступ до бази, а сервіси реалізують правила роботи платформи. Такий підхід відповідає принципам об’єктно-орієнтованого проєктування та спрощує тестування окремих частин системи.

3.6 Проєктування розгортання системи

Проєктування розгортання платформи TechExchange передбачає визначення середовища, у якому будуть працювати основні компоненти системи. Оскільки платформа реалізована як web-застосунок, користувач взаємодіє з нею через браузер. Клієнтська частина надсилає HTTP-запити до серверної частини, а сервер, у свою чергу, працює з базою даних і повертає відповідь frontend-частині [17; 18].

Основними компонентами розгортання є: [16; 17]

– web-браузер користувача;
– frontend-застосунок React;
– backend-застосунок Spring Boot;
– база даних PostgreSQL;
– середовище контейнеризації Docker;
– Docker Compose для запуску кількох компонентів;
– Swagger UI для перегляду документації API.

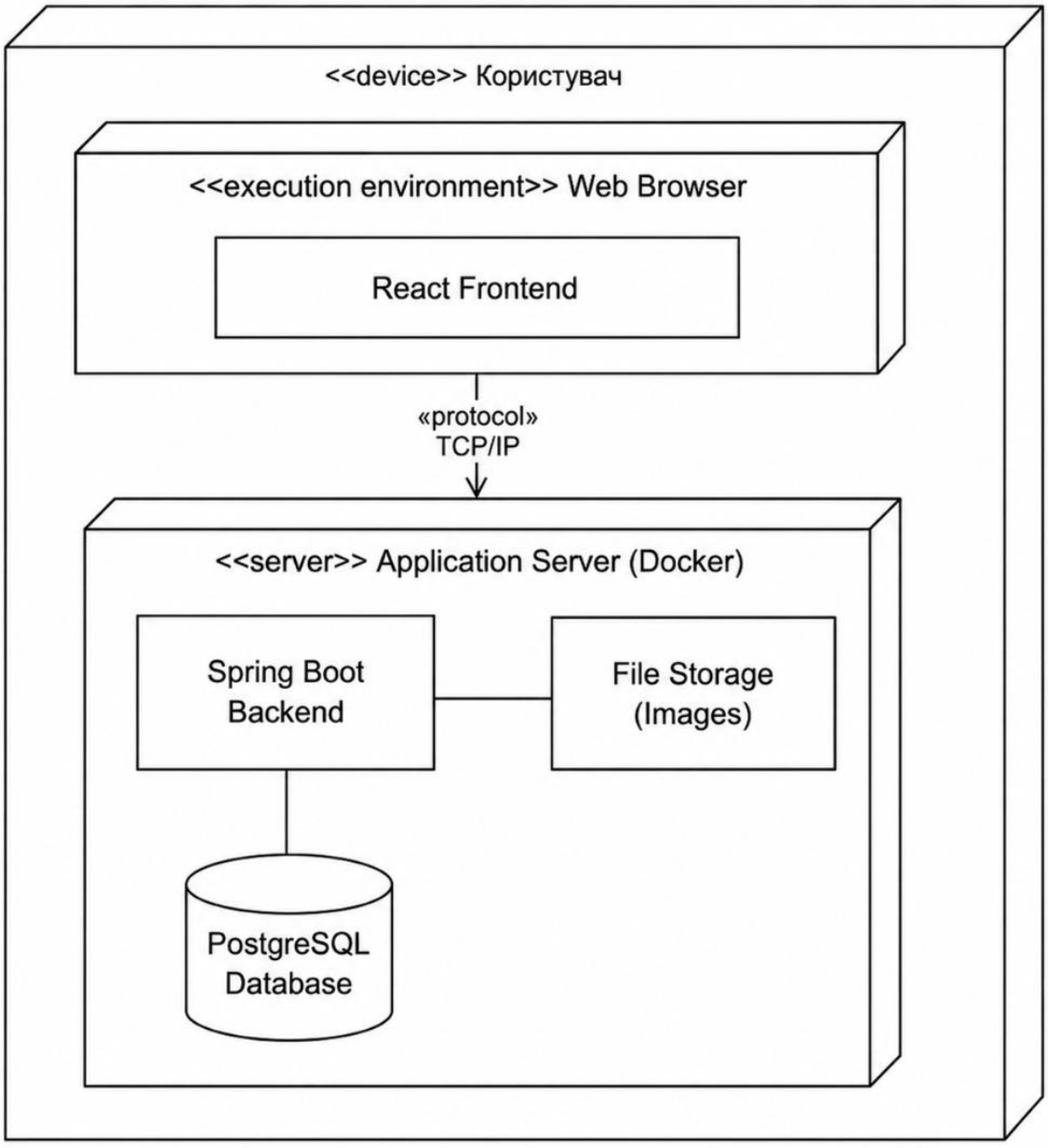


Рис. 3.7 Діаграма розгортання платформи TechExchange

Користувач працює з платформою через web-браузер. У браузері відкривається frontend, який відповідає за відображення сторінок і взаємодію з користувачем. Frontend надсилає запити до backend через REST API. Якщо запит стосується приватного ресурсу, до нього додається JWT-токен.

Backend-застосунок обробляє запити, перевіряє токен, визначає користувача, виконує бізнес-логіку та звертається до бази даних. Наприклад, під час створення оголошення backend перевіряє авторизацію, формує сутність Device, зберігає її в PostgreSQL і повертає відповідь клієнтській частині [15; 20].

PostgreSQL працює як окремий компонент системи та зберігає всі основні дані платформи. Під час запуску backend застосовує міграції Liquibase, які створюють або оновлюють структуру бази даних. Це дозволяє підтримувати актуальну структуру таблиць, зв’язків та індексів.

Docker Compose використовується для запуску системи в узгодженому середовищі. За його допомогою можна підняти базу даних, backend і frontend без ручного налаштування кожного компонента окремо. Це спрощує розгортання системи під час розробки, тестування або демонстрації [15].

Діаграма розгортання показує, що компоненти системи взаємодіють між собою через чітко визначені канали. Frontend звертається до backend, backend працює з базою даних, а користувач отримує результат через браузер. Така структура відповідає вимогам до сучасного web-застосунку та дозволяє в майбутньому масштабувати окремі компоненти системи [20].

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ПЛАТФОРМИ TECHEXCHANGE

4.1 Загальна структура програмного проєкту

Програмна реалізація платформи TechExchange виконана у вигляді клієнт-серверного web-застосунку, що складається із серверної частини, клієнтської частини та бази даних. Така структура відповідає обраній архітектурі системи та дозволяє розділити логіку відображення інтерфейсу, обробку бізнес-процесів і збереження даних [3; 5; 16].

Серверна частина реалізована мовою Java 17 з використанням Spring Boot. Вона містить REST API, бізнес-логіку, механізми автентифікації, роботу з базою даних, обробку помилок і сервісні компоненти. Клієнтська частина реалізована на React і забезпечує взаємодію користувача з платформою через web-інтерфейс. База даних PostgreSQL використовується для зберігання користувачів, оголошень, пропозицій обміну, повідомлень, відгуків і сповіщень.

Структура backend-проєкту побудована за багаторівневим принципом. Основними пакетами є controller, service, repository, entity, dto, security та config. Такий поділ дозволяє логічно розмежувати відповідальність між різними частинами програми.

Пакет controller містить REST-контролери, які приймають запити від клієнтської частини. Пакет service реалізує бізнес-логіку системи. Пакет repository відповідає за доступ до даних через Spring Data JPA. Пакет entity містить класи, що відповідають таблицям бази даних. Пакет dto використовується для передачі даних між frontend і backend. Пакет security містить класи, пов’язані з JWT-аутентифікацією. Пакет config використовується для конфігурації безпеки, OpenAPI та початкових налаштувань [16].

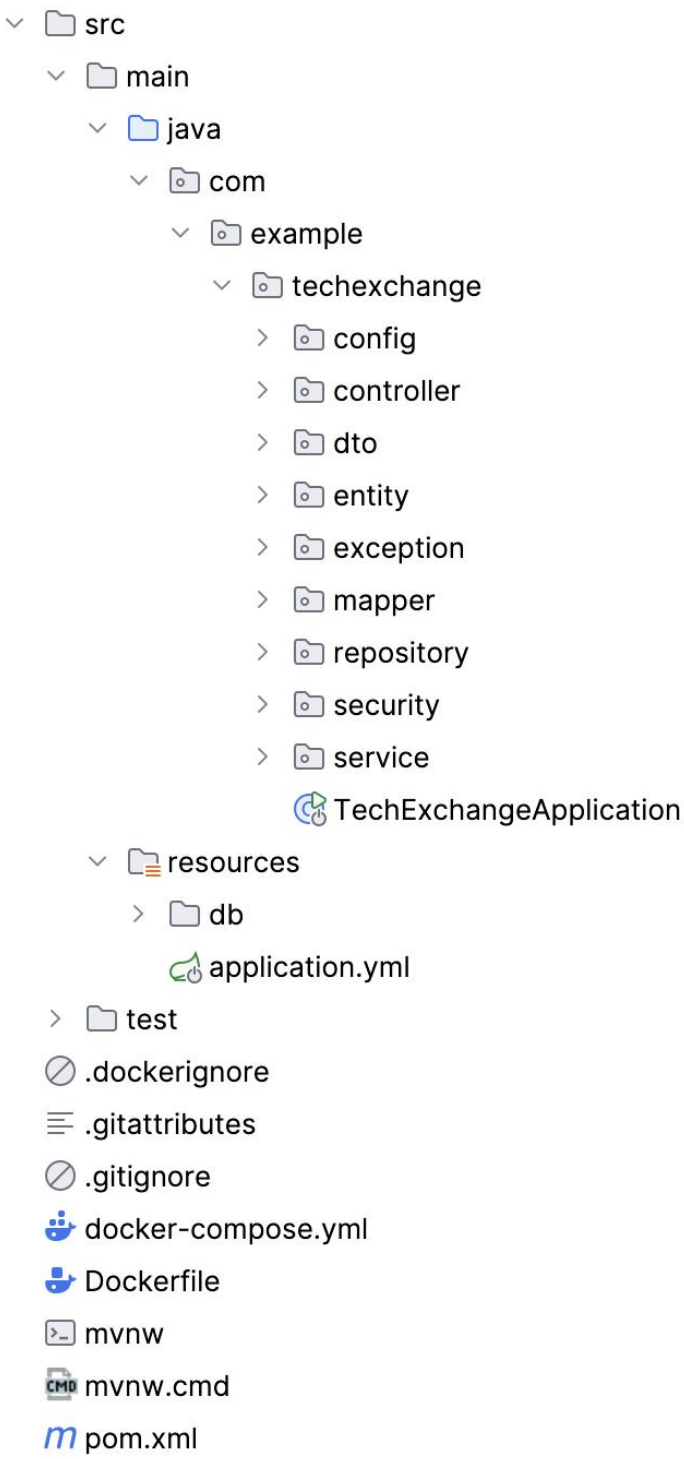


Рис. 4.1 Загальна структура файлів backend-проєкту TechExchange

Клієнтська частина має окрему структуру. Основними елементами frontend-проєкту є сторінки Login, Register, Marketplace, DeviceDetails, Requests і Profile. Також реалізовано контекст автентифікації, захищені маршрути, API-клієнт для запитів до backend та компоненти інтерфейсу [16].

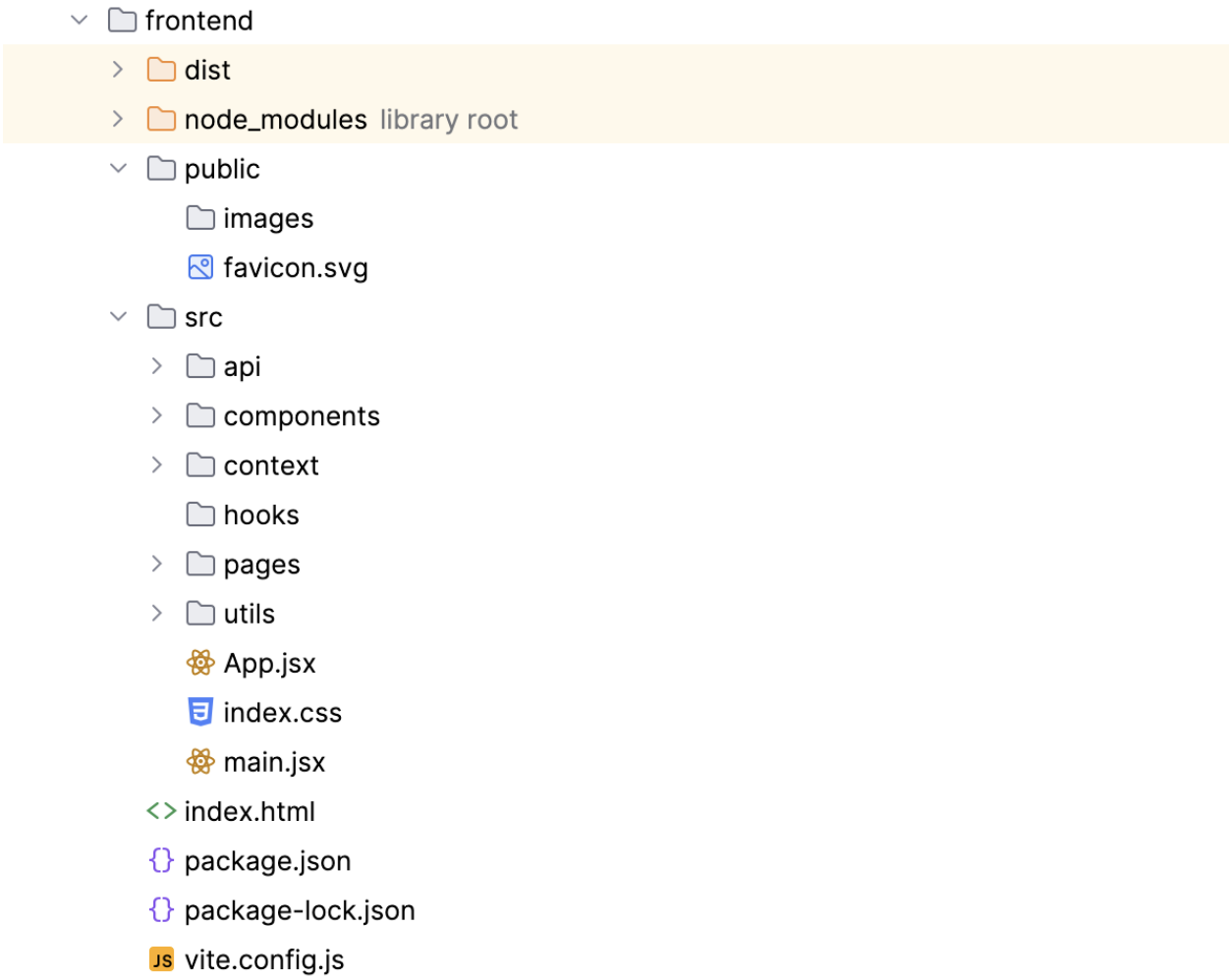


Рис. 4.2 Загальна структура файлів frontend-проєкту TechExchange

Основною точкою входу серверної частини є клас TechExchangeApplication, який запускає Spring Boot-застосунок. Після запуску сервер підключається до бази даних, застосовує міграції Liquibase, налаштовує механізми безпеки та відкриває REST API для клієнтської частини [3; 5; 16].

Загальна структура програмного проєкту дозволяє ізолювати окремі модулі один від одного. Наприклад, зміна логіки роботи з оголошеннями не потребує зміни логіки автентифікації, а зміна інтерфейсу frontend не впливає безпосередньо на структуру бази даних. Це підвищує підтримуваність системи та спрощує її подальший розвиток.

4.2 Реалізація авторизації, реєстрації та JWT-захисту

У системі реалізовано два основні сценарії: реєстрація нового користувача та вхід у систему. Під час реєстрації користувач вводить email, пароль і повне ім’я. Серверна частина перевіряє, чи не використовується такий email іншим користувачем. Якщо email унікальний, пароль хешується, після чого створюється новий запис у таблиці users.

Під час входу користувач передає email і пароль. Система перевіряє облікові дані через механізми Spring Security. Якщо дані коректні, backend генерує JWT-токен і повертає його клієнтській частині. Далі frontend зберігає токен і додає його до захищених запитів у заголовку Authorization.

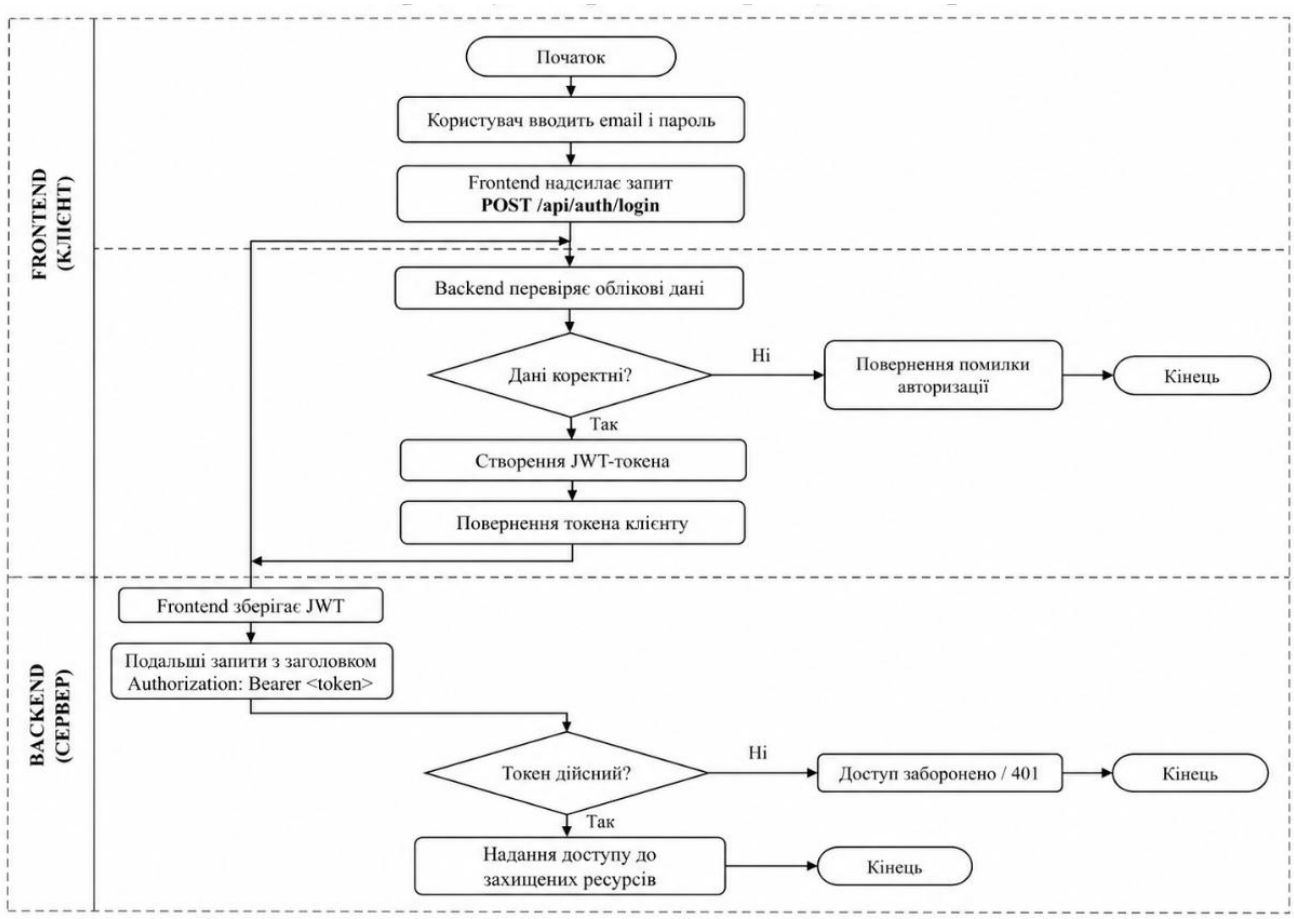


Рис. 4.3 Схема процесу авторизації користувача через JWT

Для реалізації автентифікації використовуються класи AuthController, AuthService, JwtTokenProvider, JwtAuthenticationFilter, UserDetailsServiceImpl і SecurityConfig [15; 17; 18].

AuthController містить endpoint-и для реєстрації та входу. Він приймає запити від клієнтської частини та передає їх до сервісного шару. AuthService виконує основну логіку: перевіряє email, хешує пароль, створює користувача, виконує автентифікацію та формує відповідь із токеном [17; 18].

Клас JwtTokenProvider відповідає за створення, перевірку та розбір JWT-токена. Він генерує токен на основі email користувача, встановлює час дії токена та підписує його секретним ключем. Під час подальших запитів цей клас використовується для перевірки коректності токена.

JwtAuthenticationFilter виконує перевірку токена в кожному запиті. Якщо в заголовку запиту є токен у форматі Bearer, фільтр перевіряє його, отримує email користувача, завантажує дані користувача та встановлює автентифікацію в контекст Spring Security [15; 17].

SecurityConfig визначає правила доступу до endpoint-ів. Публічними залишаються лише маршрути авторизації, реєстрації та Swagger-документації. Усі інші endpoint-и потребують наявності коректного JWT-токена. Адміністративні endpoint-и додатково обмежені роллю ADMIN [17; 18].

На frontend-частині автентифікація реалізована через AuthContext. Він зберігає токен, дані поточного користувача та функції входу й виходу. Для захисту сторінок використовується PrivateRoute, який перевіряє наявність токена перед відкриттям сторінки. Якщо токен відсутній або недійсний, користувач перенаправляється на сторінку входу [17].

4.3 Реалізація модуля оголошень і каталогу пристроїв

Модуль оголошень є одним із центральних у платформі TechExchange, оскільки саме через нього користувачі публікують техніку для обміну та переглядають доступні пристрої інших учасників. У системі реалізовано створення, перегляд, редагування, логічне видалення та фільтрацію оголошень.

Основною сутністю цього модуля є Device. Вона містить інформацію про пристрій: назву, категорію, стан, статус, бренд, модель, місто, бажаний варіант обміну, опис, посилання на зображення, активність, дату створення та дату оновлення. Кожен пристрій пов’язаний із власником, тобто користувачем, який створив оголошення.

На backend-частині за роботу з оголошеннями відповідають DeviceController, DeviceService і DeviceRepository. Контролер приймає запити від frontend-частини, сервіс виконує бізнес-логіку, а репозиторій забезпечує доступ до бази даних.

DeviceController реалізує endpoint-и для отримання списку пристроїв, перегляду власних оголошень, отримання статистики, перегляду конкретного пристрою, створення, оновлення та видалення оголошення. Для видалення використовується не фізичне вилучення запису з бази даних, а soft-delete через поле active=false. Це дозволяє зберігати історію взаємодії та не порушувати зв’язки з іншими сутностями.

DeviceService містить правила роботи з оголошеннями. Під час створення пристрою сервіс визначає поточного користувача як власника, встановлює початковий статус пристрою та зберігає запис у базі даних. Під час редагування перевіряється, що користувач має право змінювати саме своє оголошення. Під час видалення оголошення змінюється його активність.

Особливе значення має фільтрація каталогу. Користувач може шукати пристрої за текстовим запитом, категорією, станом, містом, а також переглядати лише власні оголошення. Така фільтрація дозволяє швидше знаходити потрібну техніку та зменшує кількість нерелевантних результатів [15; 17].

На frontend-частині робота з каталогом реалізована на сторінці Marketplace. Вона містить список пристроїв, блок фільтрів, статистику, кнопку створення оголошення та модальну форму для додавання або редагування пристрою. Картка пристрою містить коротку інформацію: назву, категорію, стан, місто, бажаний обмін і зображення [15; 17].

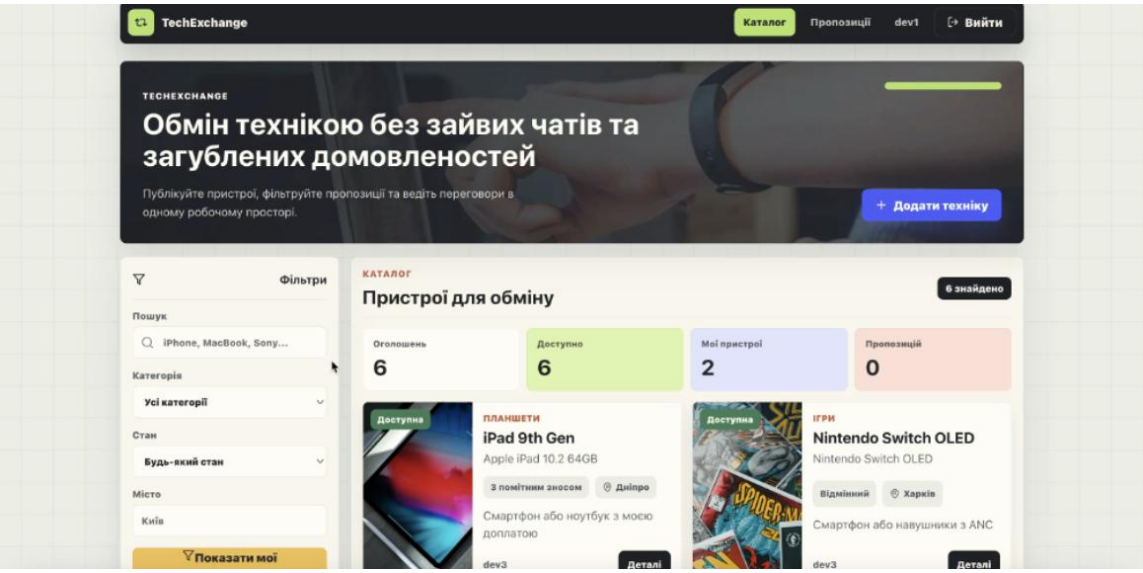


Рис. 4.4 Екранна форма головної сторінки з каталогом пристроїв

Для перегляду повної інформації про пристрій реалізована сторінка DeviceDetails. На ній відображаються всі дані оголошення та доступна форма створення пропозиції обміну. Якщо пристрій належить поточному користувачу, система не дозволяє створити пропозицію обміну на власний пристрій [15; 17].

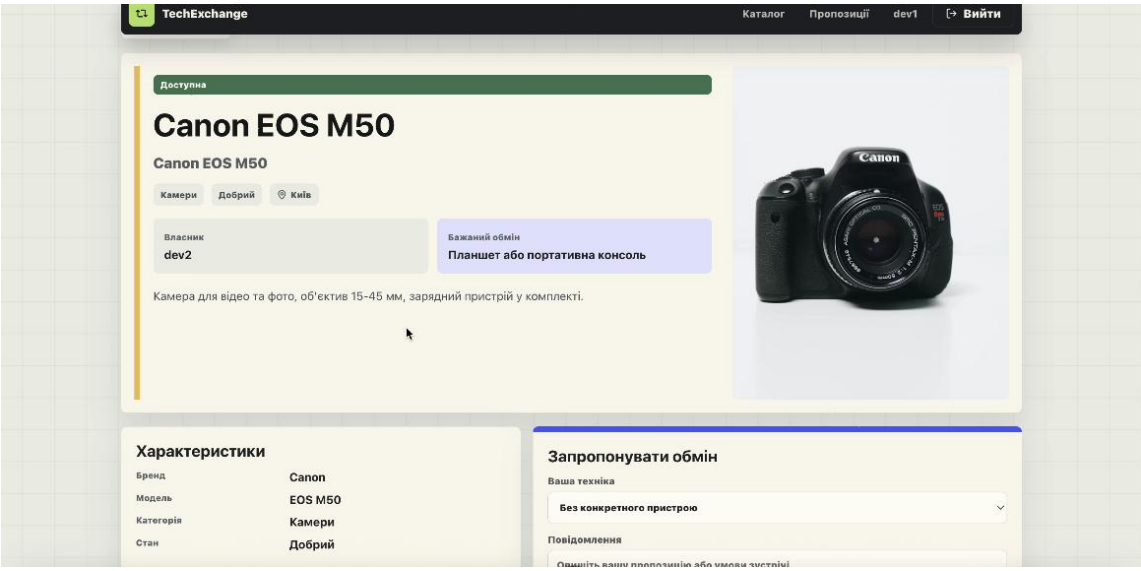


Рис. 4.5 Екранна форма детальної сторінки пристрою

4.4 Реалізація модуля пропозицій обміну та статусів домовленостей

Модуль пропозицій обміну відповідає за основний бізнес-процес платформи TechExchange. Саме він відрізняє систему від звичайної дошки оголошень, оскільки користувач не просто пише повідомлення власнику пристрою, а створює структуровану пропозицію обміну.

Основною сутністю модуля є ExchangeRequest. Вона містить посилання на цільовий пристрій, пристрій, який пропонується в обмін, ініціатора заявки, власника цільового пристрою, статус, початкове повідомлення та часові мітки. Пропозиція може містити offeredDevice або бути відкритою, якщо користувач ще не обрав конкретний пристрій.

На backend-частині роботу модуля забезпечують ExchangeRequestController, ExchangeRequestService і ExchangeRequestRepository. Контролер реалізує endpoint-и для створення пропозиції, отримання списку власних домовленостей і зміни статусу. Сервіс виконує перевірки та бізнес-правила.

Під час створення пропозиції обміну система виконує кілька перевірок. По-перше, користувач не може створити пропозицію на власний пристрій. По-друге, цільовий пристрій має бути активним і доступним. По-третє, якщо користувач обирає власний пристрій для обміну, система перевіряє, що цей пристрій справді належить йому та може брати участь в обміні.

Життєвий цикл пропозиції обміну реалізовано через статуси PENDING, ACCEPTED, ISSUED, RETURNED, DECLINED і CANCELED. Початковим статусом є PENDING. Власник цільового пристрою може прийняти або відхилити пропозицію. Ініціатор може скасувати її. Після прийняття пристрої резервуються, а після завершення обміну отримують статус обміняних.

Під час зміни статусу система контролює допустимість переходів. Наприклад, заявка не може перейти напряму з PENDING у RETURNED, оскільки спочатку вона має бути прийнята. Також користувач не може змінювати статус заявки, якщо він не є учасником відповідної домовленості.

На frontend-частині робота з домовленостями реалізована на сторінці Requests. Вона містить список вхідних і вихідних пропозицій, інформацію про пристрої, поточний статус, доступні дії для користувача та чат. Доступні кнопки залежать від ролі користувача в конкретній домовленості та поточного статусу заявки [15; 16].

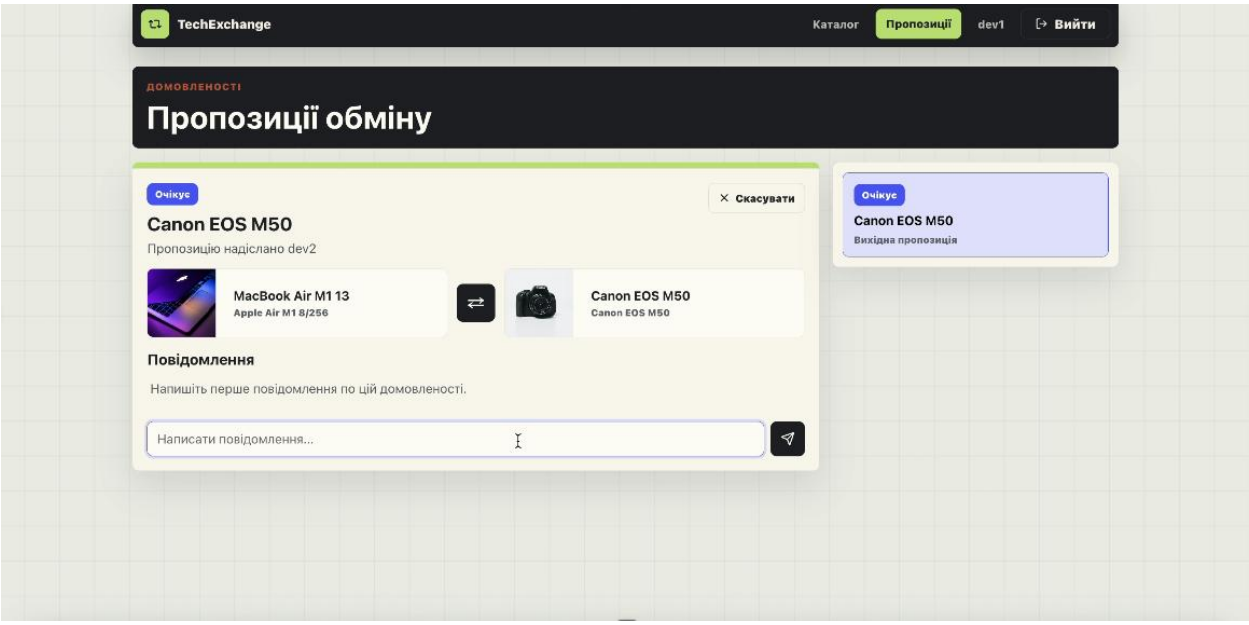


Рис. 4.6 Екранна форма сторінки пропозицій обміну

4.5 Реалізація повідомлень, відгуків, сповіщень та профілю користувача

Для забезпечення повноцінної взаємодії між користувачами в TechExchange реалізовано додаткові модулі: повідомлення, відгуки, сповіщення та профіль користувача. Вони доповнюють основний процес обміну й роблять платформу більш зручною для використання.

Модуль повідомлень реалізує вбудований чат у межах конкретної пропозиції обміну. На відміну від загального чату між користувачами, повідомлення прив’язуються саме до ExchangeRequest. Це дозволяє зберігати історію переговорів у контексті конкретної домовленості [15; 16; 17].

За роботу з повідомленнями відповідають MessageController, MessageService і MessageRepository. Користувач може переглядати повідомлення лише в тих

домовленостях, де він є учасником. Під час створення нового повідомлення система зберігає текст, відправника, час створення та надсилає сповіщення іншому учаснику.

Модуль відгуків використовується для оцінювання користувачів після завершення обміну. Відгук можна залишити лише після того, як домовленість перейшла у фінальний статус завершення. Також у системі реалізовано обмеження: один користувач може залишити лише один відгук у межах однієї домовленості. Це запобігає дублюванню оцінок [20].

ReviewService автоматично визначає, кому саме залишають відгук. Якщо поточний користувач є ініціатором обміну, відгук створюється для власника цільового пристрою, і навпаки. На основі відгуків у профілі користувача розраховується середній рейтинг і кількість отриманих відгуків [20].

Модуль сповіщень забезпечує інформування користувача про важливі події. Сповіщення створюються при появі нової пропозиції обміну, зміні статусу домовленості, новому повідомленні або новому відгуку. Кожне сповіщення має тип, заголовок, текст, ознаку прочитання та дату створення [20].

Профіль користувача реалізовано через UserController і UserService. Користувач може переглянути власні дані, змінити повне ім’я, побачити email, середній рейтинг і кількість відгуків. Також система дозволяє переглядати публічний профіль іншого користувача за ідентифікатором.

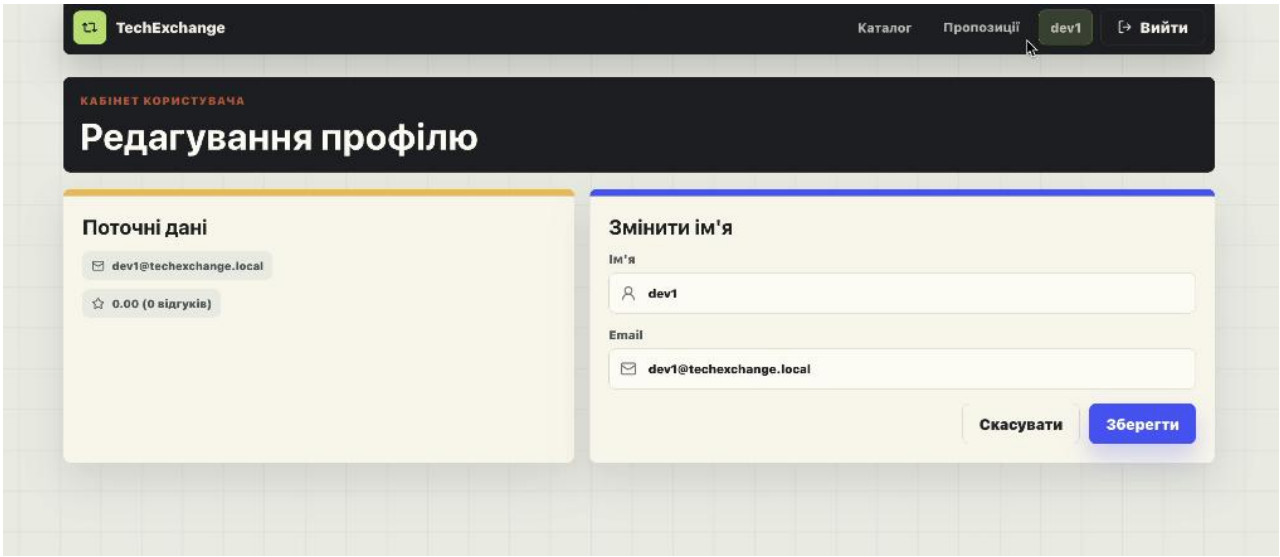


Рис. 4.7 Екранна форма профілю користувача

Завдяки цим модулям платформа забезпечує не лише створення оголошень і заявок, а й повну взаємодію між учасниками: переговори, інформування, завершення угоди та оцінювання користувачів після обміну.

4.6 Реалізація клієнтського web-інтерфейсу та адміністративного модуля

Клієнтський web-інтерфейс платформи TechExchange реалізовано як односторінковий застосунок на React. Інтерфейс побудовано навколо основних сценаріїв користувача: вхід, реєстрація, перегляд каталогу, створення оголошення, перегляд деталей пристрою, створення пропозиції обміну, робота з домовленостями, чат і профіль [20].

Сторінка входу дозволяє користувачу авторизуватися за email і паролем. Після успішного входу користувач переходить на головну сторінку платформи. Сторінка реєстрації дозволяє створити новий обліковий запис [20].

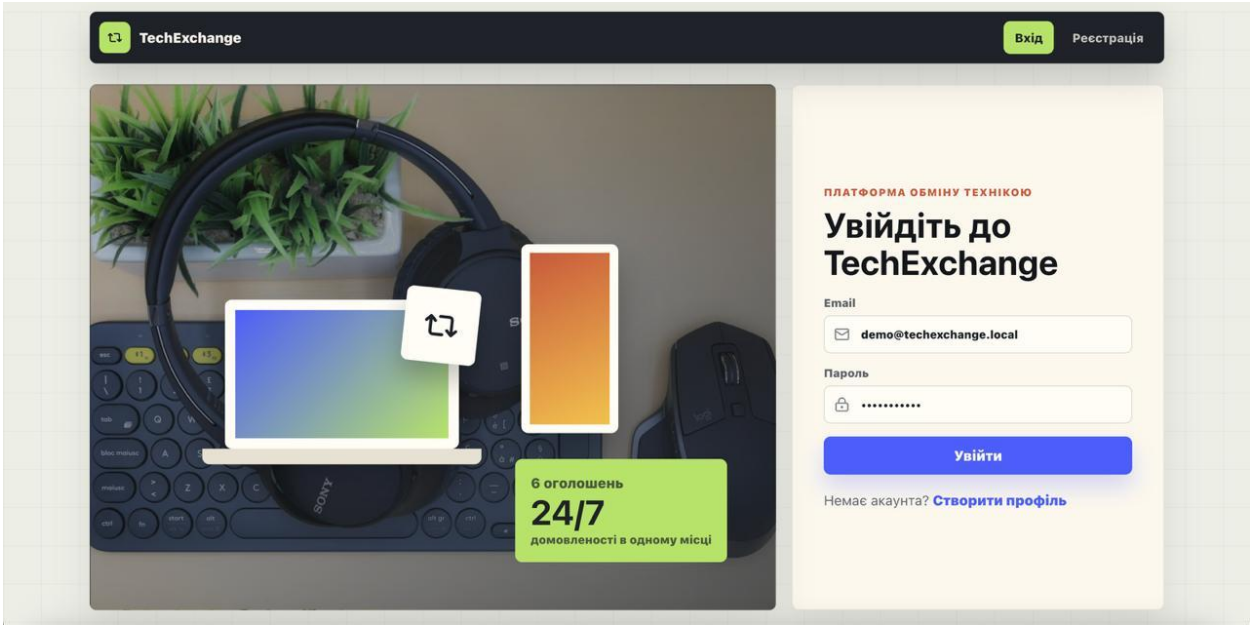


Рис. 4.8 Екранна форма входу до платформи TechExchange

Головна сторінка Marketplace містить каталог пристроїв. Користувач може бачити доступні оголошення, фільтрувати їх, переглядати статистику, створювати

нові оголошення та редагувати власні. Для створення оголошення використовується модальна форма, у якій користувач вводить основні характеристики техніки.

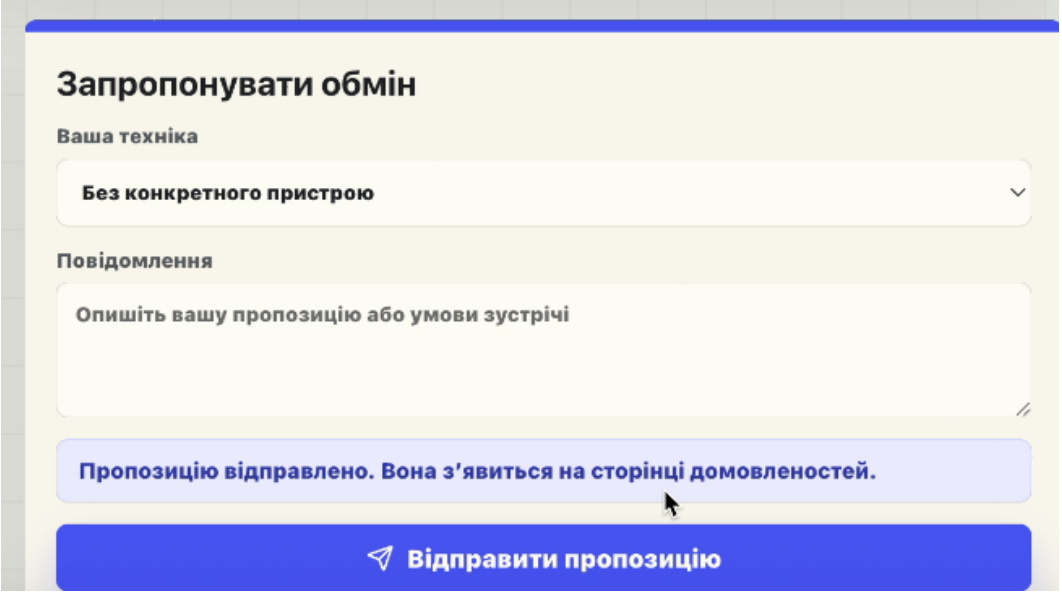


Рис. 4.9 Екранна форма створення оголошення про техніку

Сторінка DeviceDetails призначена для детального перегляду пристрою. На ній користувач бачить повний опис, фото, стан, категорію, місто, бажаний варіант обміну та форму для надсилання пропозиції. У формі користувач може обрати один зі своїх пристроїв і додати повідомлення [20].

Сторінка Requests є однією з найважливіших у системі, оскільки саме тут відбувається керування домовленостями. Вона містить список пропозицій, поточні статуси, доступні дії та чат. Інтерфейс дозволяє користувачу швидко зрозуміти, які пропозиції очікують відповіді, які прийняті, які завершені або скасовані [20].

У клієнтському інтерфейсі передбачено обробку помилок. Якщо користувач виконує некоректну дію, наприклад намагається створити пропозицію на власний пристрій або перейти до захищеної сторінки без авторизації, система відображає відповідне повідомлення або перенаправляє його на сторінку входу [20].

5 ТЕСТУВАННЯ ТА ОЦІНКА РЕЗУЛЬТАТІВ РОЗРОБКИ

5.1 Мета та підхід до тестування платформи

Тестування платформи TechExchange виконувалося з метою перевірки коректності роботи основних функціональних модулів, відповідності системи поставленим вимогам, стабільності взаємодії між клієнтською та серверною частинами, а також правильності обробки користувацьких сценаріїв [7; 8; 10; 11].

Оскільки платформа реалізована як клієнт-серверний web-застосунок, тестування охоплювало декілька рівнів: перевірку серверної бізнес-логіки, тестування REST API, перевірку захищених маршрутів, тестування роботи з базою даних, а також перевірку основних сценаріїв у клієнтському інтерфейсі [7; 10; 11].

Основними об’єктами тестування були:

– реєстрація та авторизація користувачів;
– перевірка JWT-захисту та доступу до приватних маршрутів;
– створення, редагування, перегляд і видалення оголошень;
– пошук і фільтрація каталогу пристроїв;
– створення пропозицій обміну;
– зміна статусів домовленості;
– робота вбудованого чату;
– створення відгуків після завершення обміну;
– формування сповіщень;
– адміністративне керування користувачами та оголошеннями.

Для backend-частини використовувалися unit-тести та інтеграційні тести. Unit-тести застосовувалися для перевірки окремих сервісів, зокрема логіки авторизації та обробки користувацьких даних. Інтеграційні тести використовувалися для перевірки повних сценаріїв роботи API з базою даних [10; 11].

Для інтеграційного тестування застосовано JUnit 5, MockMvc, Spring Security Test та Testcontainers. JUnit 5 використовується як основний фреймворк для

написання тестів. MockMvc дозволяє імітувати HTTP-запити до REST API без запуску окремого web-сервера. Spring Security Test використовується для перевірки доступу до захищених endpoint-ів. Testcontainers дозволяє запускати тестову PostgreSQL-базу даних у контейнері, що робить тестове середовище більш наближеним до реального [7; 10; 11].

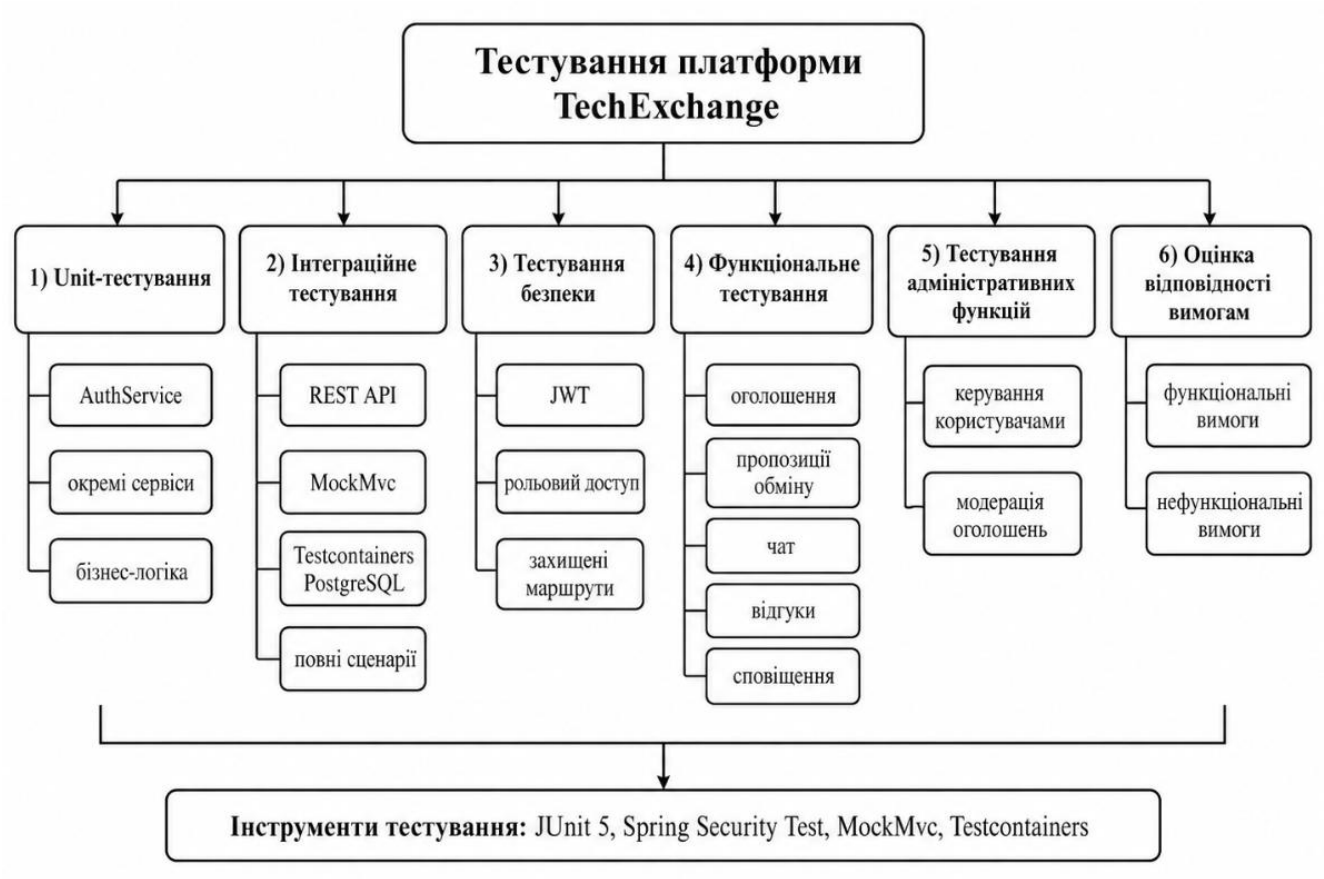


Рис. 5.1 Схема підходу до тестування платформи TechExchange

Під час тестування також враховувалися нефункціональні вимоги. Зокрема перевірялася коректність обробки помилок, обмеження доступу до приватних ресурсів, стабільність роботи при виконанні типових операцій та збереження цілісності даних під час зміни статусів домовленостей [8; 9; 10].

5.2 Тестування автентифікації та захищених маршрутів

Першим етапом тестування була перевірка модуля автентифікації та авторизації користувачів. Цей модуль є критично важливим для платформи,

оскільки більшість функцій доступна лише авторизованим користувачам. Без перевірки користувача не можна створювати оголошення, надсилати пропозиції обміну, вести чат, переглядати домовленості або редагувати профіль [3; 5; 16].

Під час тестування реєстрації перевірялося, чи може новий користувач створити обліковий запис із валідними даними. Система повинна зберігати email, повне ім’я, роль користувача, статус активності та пароль у хешованому вигляді. Також перевірялося, що користувач не може зареєструватися з email, який уже використовується в системі [3; 5; 16].

Під час тестування входу перевірялося, чи повертає система JWT-токен після введення правильного email і пароля. Якщо користувач вводить некоректні облікові дані, система повинна повернути помилку й не надавати доступ до приватних ресурсів [3; 5; 16].

Окремо перевірялася робота захищених маршрутів. Для цього виконувалися запити до приватних endpoint-ів без токена, з некоректним токеном і з валідним токеном. У перших двох випадках система повинна відхиляти запит, а у третьому — дозволяти доступ [3; 5; 16].

Таблиця 5.1

Результати тестування автентифікації та захищених маршрутів

№	**Тестовий сценарій**	**Очікуваний результат**	**Результат**
1	Реєстрація користувача з валідними даними	Користувач створюється, повертається JWT-токен	Виконано
2	Реєстрація з email, який уже існує	Система повертає помилку конфлікту	Виконано
3	Вхід із правильним email і паролем	Система повертає JWT-токен	Виконано
4	Вхід із неправильним паролем	Доступ не надається	Виконано
5	Запит до приватного endpoint-а без токена	Система повертає помилку авторизації	Виконано

Продовження таблиці 5.1

Результати тестування автентифікації та захищених маршрутів

6	Запит до приватного endpoint-а з валідним токеном	Запит виконується успішно	Виконано
7	Запит до адміністративного endpoint-а користувачем USER	Доступ заборонено	Виконано
8	Запит до адміністративного endpoint-а користувачем ADMIN	Доступ дозволено	Виконано

Результати тестування показали, що модуль автентифікації працює коректно. Система не дозволяє неавторизованим користувачам виконувати приватні дії та правильно розмежовує доступ між ролями USER і ADMIN [3; 5; 16].

5.3 Тестування роботи з оголошеннями та каталогом пристроїв

Наступним етапом було тестування модуля оголошень і каталогу пристроїв. Цей модуль забезпечує основну функцію платформи — публікацію техніки для обміну та пошук пристроїв іншими користувачами [17; 18].

Під час тестування створення оголошення перевірялося, чи може авторизований користувач додати пристрій із назвою, категорією, станом, брендом, моделлю, містом, описом, зображенням і бажаним варіантом обміну. Також перевірялося, що створене оголошення автоматично пов’язується з поточним користувачем [17; 18].

Для редагування оголошень перевірялося, що користувач може змінювати лише власні записи. Якщо користувач намагається редагувати чуже оголошення, система повинна заборонити таку дію. Це важливо для захисту даних і коректності роботи платформи [16; 17].

Видалення оголошення реалізовано як логічне приховування через поле active. Тому під час тестування перевірялося, що після видалення оголошення не відображається в загальному каталозі, але його запис не зникає з бази даних

фізично. Такий підхід дозволяє зберігати історію взаємодії та не порушувати зв’язки з пропозиціями обміну [17; 18].

Окремо тестувалася фільтрація каталогу. Було перевірено пошук за назвою, категорією, станом, містом і режимом перегляду власних оголошень. Усі фільтри мають повертати лише ті пристрої, які відповідають заданим параметрам [17; 18].

Таблиця 5.2

Результати тестування модуля оголошень

№	**Тестовий сценарій**	**Очікуваний результат**	**Результат**
1	Створення оголошення авторизованим користувачем	Оголошення створюється та прив’язується до користувача	Виконано
2	Перегляд каталогу пристроїв	Відображаються активні доступні оголошення	Виконано
3	Перегляд детальної сторінки пристрою	Відображається повна інформація про пристрій	Виконано
4	Редагування власного оголошення	Дані оголошення оновлюються	Виконано
5	Редагування чужого оголошення	Доступ заборонено	Виконано
6	Логічне видалення оголошення	Оголошення стає неактивним	Виконано
7	Фільтрація за категорією	Виводяться пристрої відповідної категорії	Виконано
8	Фільтрація за містом	Виводяться пристрої з відповідного міста	Виконано

Результати тестування підтвердили, що модуль оголошень працює відповідно до вимог. Користувач може керувати власними пристроями, переглядати каталог і знаходити потрібну техніку за допомогою фільтрів [17; 18].

5.4 Тестування життєвого циклу пропозиції обміну

Окрему увагу було приділено тестуванню життєвого циклу пропозиції обміну, оскільки цей модуль є ключовою відмінністю TechExchange від звичайних дошок оголошень. Саме через нього користувачі можуть оформлювати не просто повідомлення, а структуровану заявку на обмін [1; 2].

Під час створення пропозиції перевірялося, чи може користувач обрати цільовий пристрій, власний пристрій для обміну та додати початкове повідомлення. Система повинна забороняти створення пропозиції на власний пристрій, оскільки така дія не має практичного сенсу [1; 2].

Після створення заявка отримує статус PENDING. Далі власник цільового пристрою може прийняти або відхилити її. Якщо заявка приймається, вона переходить у статус ACCEPTED, а пристрої резервуються. Також система повинна автоматично відхилити конкуруючі заявки на той самий пристрій, щоб уникнути паралельних домовленостей [1; 2].

Наступним етапом є перевірка переходів у статуси ISSUED і RETURNED. Після завершення обміну пристрої мають отримати статус EXCHANGED. Якщо заявка відхиляється або скасовується, пристрої повинні повернутися до доступного стану [1; 2].

Таблиця 5.3

Результати тестування життєвого циклу пропозиції обміну

№	Тестовий сценарій	Очікуваний результат	Результат
1	Створення пропозиції обміну	Створюється заявка зі статусом PENDING	Виконано
2	Спроба створити заявку на власний пристрій	Система забороняє дію	Виконано
3	Прийняття заявки власником пристрою	Статус змінюється на ACCEPTED	Виконано

Продовження таблиці 5.3

Результати тестування життєвого циклу пропозиції обміну

№	Тестовий сценарій	Очікуваний результат	Результат
4	Резервування пристроїв після ACCEPTED	Пристрої отримують статус RESERVED	Виконано
5	Автоматичне відхилення конкурентних заявок	Конкуруючі заявки переходять у DECLINED	Виконано
6	Завершення обміну	Заявка переходить у RETURNED, пристрої стають EXCHANGED	Виконано
7	Скасування заявки ініціатором	Статус змінюється на CANCELED	Виконано
8	Некоректний перехід статусу	Система повертає помилку	Виконано

Результати тестування показали, що життєвий цикл пропозиції обміну реалізовано коректно. Система контролює допустимі переходи між статусами, не дозволяє користувачам виконувати заборонені дії та забезпечує логічне оновлення стану пристроїв [1; 2].

5.5 Тестування повідомлень, відгуків, сповіщень та адміністративних функцій

Після перевірки основного процесу обміну було протестовано додаткові модулі, які забезпечують повноцінну взаємодію користувачів у системі. До них належать повідомлення, відгуки, сповіщення та адміністративний модуль [1; 2].

Під час тестування повідомлень перевірялося, чи можуть учасники конкретної домовленості надсилати та переглядати повідомлення. Система повинна забороняти доступ до чату користувачам, які не є учасниками відповідної пропозиції обміну. Це забезпечує приватність комунікації [1; 2].

Модуль відгуків тестувався після завершення обміну. Було перевірено, що користувач може залишити відгук лише після фінального статусу домовленості. Також перевірялося обмеження, за яким один користувач може залишити лише один відгук у межах однієї пропозиції обміну. Після створення відгуку система повинна оновлювати середній рейтинг користувача та кількість отриманих відгуків [1; 2].

Модуль сповіщень тестувався на подіях створення заявки, зміни статусу, нового повідомлення та нового відгуку. У кожному випадку система повинна створювати сповіщення для відповідного користувача. Також перевірялося відображення списку сповіщень і можливість позначити їх як прочитані [1; 2].

Адміністративний модуль тестувався окремо. Було перевірено, що тільки користувач із роллю ADMIN має доступ до адміністративних функцій. Адміністратор може переглядати список користувачів, змінювати роль, активувати або деактивувати користувача, а також керувати активністю оголошень [3; 5; 16].

Таблиця 5.4

Результати тестування додаткових модулів платформи

№	**Модуль**	**Тестовий сценарій**	**Очікуваний результат**	**Результат**
1	Повідомлення	Надсилання повідомлення учасником угоди	Повідомлення зберігається	Виконано
2	Повідомлення	Спроба доступу до чужого чату	Доступ заборонено	Виконано
3	Відгуки	Створення відгуку після завершеного обміну	Відгук зберігається	Виконано
4	Відгуки	Повторний відгук на ту саму угоду	Система повертає помилку	Виконано

Продовження таблиці 5.4

Результати тестування додаткових модулів платформи

№	Модуль	Тестовий сценарій	Очікуваний результат	Результат
5	Сповіщення	Створення сповіщення при новому повідомленні	Одержувач отримує сповіщення	Виконано
6	Сповіщення	Позначення сповіщення як прочитаного	Статус is_read змінюється	Виконано
7	Адмін-модуль	Доступ USER до адміністративного API	Доступ заборонено	Виконано
8	Адмін-модуль	Деактивація користувача адміністратором	Користувач стає неактивним	Виконано
9	Адмін-модуль	Деактивація оголошення адміністратором	Оголошення стає неактивним	Виконано

Результати тестування показали, що додаткові модулі працюють коректно й підтримують основний сценарій обміну. Повідомлення забезпечують комунікацію, відгуки формують довіру між користувачами, сповіщення інформують про події, а адміністративний модуль дозволяє контролювати платформу [1; 2].

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було розроблено web-платформу TechExchange, призначену для організації обміну технікою між користувачами. Система дозволяє публікувати оголошення про пристрої, переглядати каталог техніки, використовувати пошук і фільтрацію, створювати пропозиції обміну, вести переговори через вбудований чат, відстежувати статус домовленостей, залишати відгуки після завершення обміну та отримувати сповіщення про важливі події [1; 2].

У межах роботи було виконано аналіз предметної галузі обміну технікою між користувачами. Визначено, що основними проблемами існуючих підходів є неструктурованість оголошень, складність пошуку потрібних пристроїв, відсутність спеціалізованого механізму пропозицій обміну, перенесення комунікації у зовнішні месенджери та неможливість централізовано контролювати статус домовленості [1; 2].

Було проаналізовано існуючі програмні рішення, які можуть використовуватися для розміщення та пошуку техніки, зокрема OLX, eBay, Swappa, Shpock, Facebook Marketplace та Telegram-групи. У результаті порівняння встановлено, що більшість аналогів орієнтована переважно на купівлю-продаж або загальне розміщення оголошень, а не на повний цикл P2P-обміну технікою. Це обґрунтовує доцільність розробки окремої спеціалізованої платформи TechExchange [1; 2].

На основі проведеного аналізу було сформовано функціональні та нефункціональні вимоги до програмного забезпечення. До основних функціональних вимог належать реєстрація та авторизація користувачів, керування оголошеннями, перегляд каталогу пристроїв, пошук і фільтрація, створення пропозицій обміну, зміна статусів домовленостей, вбудований чат, відгуки, сповіщення та адміністративний контроль. До нефункціональних вимог віднесено захист приватних маршрутів, хешування паролів, швидкість відповіді API,

стабільність роботи, використання PostgreSQL, документування API через Swagger UI та можливість запуску системи через Docker Compose [8; 9].

Для реалізації платформи було обґрунтовано вибір сучасного технологічного стеку. Серверну частину реалізовано мовою Java 17 із використанням Spring Boot, Spring Security, Spring Data JPA та JWT. Для збереження даних використано PostgreSQL, а керування змінами структури бази даних здійснюється за допомогою Liquibase. Клієнтську частину реалізовано на React із використанням Vite, React Router, Axios та Lucide React. Для документування API використано Swagger UI, а для розгортання — Docker Compose [14-22].

У процесі проєктування було розроблено загальну архітектуру платформи, структуру бази даних, доменну модель, діаграму активності, діаграму класів і діаграму розгортання. Спроєктована база даних містить таблиці users, devices, exchange_requests, messages, reviews та notifications, які забезпечують збереження основних даних системи та підтримують логічні зв’язки між користувачами, оголошеннями, пропозиціями обміну, повідомленнями, відгуками й сповіщеннями [7; 10; 12; 13].

У межах програмної реалізації було створено серверну частину з REST API для роботи з користувачами, пристроями, пропозиціями обміну, повідомленнями, відгуками, сповіщеннями та адміністративними функціями. Реалізовано JWT-захист, рольову модель доступу USER/ADMIN, перевірку прав користувачів, soft-delete для оголошень, життєвий цикл пропозиції обміну та автоматичне створення сповіщень при важливих подіях [14-19].

Клієнтський web-інтерфейс забезпечує доступ до основних сценаріїв роботи платформи: входу, реєстрації, перегляду каталогу, створення та редагування оголошень, перегляду детальної сторінки пристрою, надсилання пропозиції обміну, роботи зі сторінкою домовленостей, ведення чату та перегляду профілю користувача. Захищені маршрути доступні лише після авторизації [20].

Проведене тестування підтвердило коректність роботи основних модулів платформи. Було перевірено реєстрацію та авторизацію, захищені маршрути, роботу з оголошеннями, фільтрацію каталогу, створення пропозицій обміну, зміну

статусів домовленостей, резервування пристроїв, роботу чату, створення відгуків, сповіщення та адміністративні функції. Результати тестування показали, що розроблена система відповідає поставленим вимогам [7; 10; 11].

Таким чином, у кваліфікаційній роботі було досягнуто поставленої мети — спрощено процес обміну технікою між користувачами за рахунок розробки web-платформи TechExchange. На відміну від універсальних дошок оголошень, розроблена система забезпечує не лише публікацію пристроїв, а й повний цикл взаємодії між користувачами: від створення оголошення та пропозиції обміну до переговорів, зміни статусу домовленості, завершення обміну й залишення відгуку [1; 2].

Робота пройшла апробацію. За її результатами було опубліковано наступні тези доповідей:

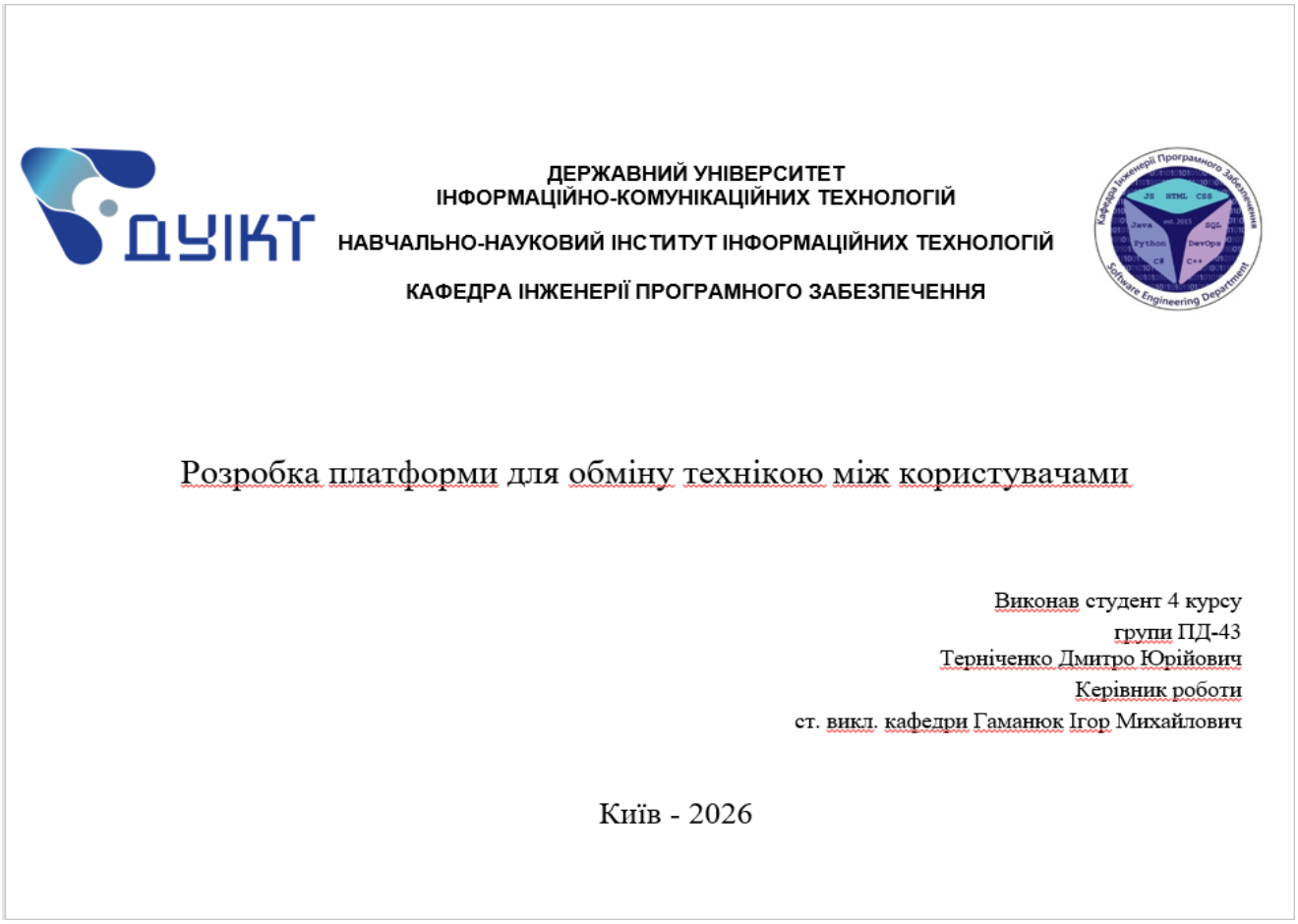
1. Терніченко Д.Ю., Соляник Л.О. Розробка платформи для обміну технікою. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.518-520.
2. Терніченко Д.Ю., Гаманюк І.М. Створення вебсервісу для обміну технікою між користувачами. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. – К.: ДУІКТ, 2026, С.432-434.

ПЕРЕЛІК ПОСИЛАНЬ

1. Терніченко Д. Ю., Соляник Л. О. Розробка платформи для обміну технікою. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. Київ: ДУІКТ, 2025. С. 518–520.
2. Терніченко Д. Ю., Гаманюк І. М. Створення вебсервісу для обміну технікою між користувачами. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 р., Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2026, С.432-434.
3. Про захист персональних даних : Закон України від 01.06.2010 р. № 2297-VI : станом на 01.01.2026 р. URL: https://zakon.rada.gov.ua/laws/show/2297-17
4. Про електронні комунікації : Закон України від 16.12.2020 р. № 1089-IX : станом на 01.01.2026 р. URL: https://zakon.rada.gov.ua/laws/show/1089-20
5. Про основні засади забезпечення кібербезпеки України : Закон України від 05.10.2017 р. № 2163-VIII : станом на 01.01.2026 р. URL: https://zakon.rada.gov.ua/laws/show/2163-19
6. Настанова до зводу знань з управління проєктами. Настанова PMBOK. 7-ме видання та стандарт з управління проєктами / Project Management Institute. Київ: PMI Ukraine, 2021. URL: https://pmiukraine.org/pmbok7
7. ДСТУ ISO/IEC/IEEE 12207:2018. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення. Київ: ДП «УкрНДНЦ», 2018.
8. ДСТУ ISO/IEC 25010:2016. Інженерія систем і програмних засобів. Вимоги до якості та оцінювання систем і програмного забезпечення. Моделі якості системи та програмного забезпечення. Київ: ДП «УкрНДНЦ», 2016.

9. ISO/IEC 25010:2023. Systems and software engineering. Systems and software Quality Requirements and Evaluation (SQuaRE). Product quality model. Geneva: International Organization for Standardization, 2023.
10. Sommerville I. Engineering Software Products: An Introduction to Modern Software Engineering**.** Harlow: Pearson, 2020. 352 p.
11. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach**.** 9th ed. New York: McGraw-Hill Education, 2020. 672 p.
12. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. Sebastopol: O'Reilly Media, 2020. 422 p.
13. Kleppmann M. Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems. Sebastopol: O'Reilly Media, 2017. 616 p.
14. Oracle. Java Platform, Standard Edition 17 Documentation. URL: https://docs.oracle.com/en/java/javase/17/
15. Spring. Spring Boot Reference Documentation. URL: https://docs.spring.io/spring-boot/docs/current/reference/html/
16. Spring. Spring Security Reference Documentation. URL: https://docs.spring.io/spring-security/reference/
17. Spring. Spring Data JPA Reference Documentation. URL: https://docs.spring.io/spring-data/jpa/reference/
18. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/
19. Liquibase. Liquibase Documentation. URL: https://docs.liquibase.com/
20. React. React Documentation. URL: https://react.dev/
21. Docker. Docker Documentation. URL: https://docs.docker.com/
22. OpenAPI Initiative. OpenAPI Specification. URL: https://spec.openapis.org/oas/latest.html

ДОДАТОК А. ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка платформи для обміну технікою між користувачами

Виконав студент 4 курсу
групи ПД-43
Терніченко Дмитро Юрійович
Керівник роботи
ст. викл. кафедри Гаманюк Ігор Михайлович

Київ - 2026

ЗАДАЧІ КВАЛІФІКАЦІЙНОЇ РОБОТИ

1. Провести аналіз предметної галузі обміну технікою між користувачами та визначити основні проблеми розміщення оголошень, пошуку пристроїв, формування пропозицій обміну й комунікації між учасниками.
2. Дослідити існуючі програмні рішення для розміщення та пошуку техніки, порівняти їх функціональні можливості та визначити основні недоліки.
3. Сформувати функціональні та нефункціональні вимоги до платформи TechExchange.
4. Обґрунтувати вибір технологій для реалізації web-платформи.
5. Спроєктувати архітектуру платформи та структуру бази даних.
6. Реалізувати серверну частину для керування користувачами, оголошеннями, пропозиціями обміну, повідомленнями, відгуками та сповіщеннями.
7. Реалізувати клієнтський web-інтерфейс для авторизації, перегляду каталогу, створення оголошень, оформлення пропозицій обміну, ведення чату та роботи з профілем.
8. Провести тестування основних функцій платформи та оцінити відповідність розробленого програмного забезпечення поставленим вимогам.

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: спрощення процесу обміну технікою між користувачами.

Об'єкт дослідження: процес організації обміну технікою між користувачами, розміщення оголошень, пошуку пристроїв, формування пропозицій обміну та комунікації між учасниками.

Предмет дослідження: програмна платформа TechExchange для публікації оголошень про техніку, створення пропозицій обміну, ведення переговорів через вбудований чат, відстеження статусу домовленостей, отримання сповіщень та залишення відгуків після завершення обміну.

3

АНАЛІЗ АНАЛОГІВ

Показник	OLX	eBay	Swappa	Shpock	TechExchange
Каталог пристроїв	+	+	+	+	+
Пошук і фільтрація	+	+	+	+	+
Детальна сторінка пристрою	+	+	+	+	+
Структурований опис характеристик	базові поля	+	+	базові поля	+
Фото пристрою	+	+	+	+	+
Формування пропозиції обміну	-	-	-	-	+
Вибір власного пристрою для обміну	-	-	-	-	+
Вбудований чат по конкретній угоді	загальні повідомлення	загальні повідомлення	+	+	+
Відстеження статусу домовленості	-	-	-	-	+
Відгуки після завершення обміну	рейтинг продавця	+	+	рейтинг профілю	+
Сповіщення про події	+	+	+	+	+
Адміністративний контроль	+	+	+	базова модерація	+
Орієнтація на обмін технікою	продаж із можливістю обміну	-	продаж техніки	локальні оголошення	+

4

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціональні вимоги

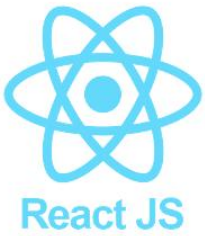
1. Авторизація та реєстрація, перевірка користувача через JWT.
2. Керування оголошеннями.
3. Перегляд доступної техніки з короткою.
4. Пошук і фільтрація оголошеня.
5. Детальна сторінка пристрою.
6. Пропозиції обміну.
7. Керування домовленостями.
8. Вбудований чат.
9. Можливість залишити рейтинг і коментар після завершення обміну.
10. Керування користувачами та оголошеннями з боку адміністратора.

Нефункціональні вимоги

1. Завантаження сторінок — до 3 с.
2. Відповідь API — до 1 с для типових запитів.
3. Пошук і фільтрація каталогу — до 2 с.
4. Захист доступу — усі приватні маршрути доступні лише після авторизації.
5. Зберігання паролів — тільки у хешованому вигляді.
6. База даних — PostgreSQL з підтримкою індексів для пошуку та зв'язків між сутностями.
7. Масштабованість — підтримка не менше 100 одночасних користувачів у тестовому середовищі.
8. Документування API — доступ через Swagger UI.
9. Розгортання — запуск системи через Docker Compose.

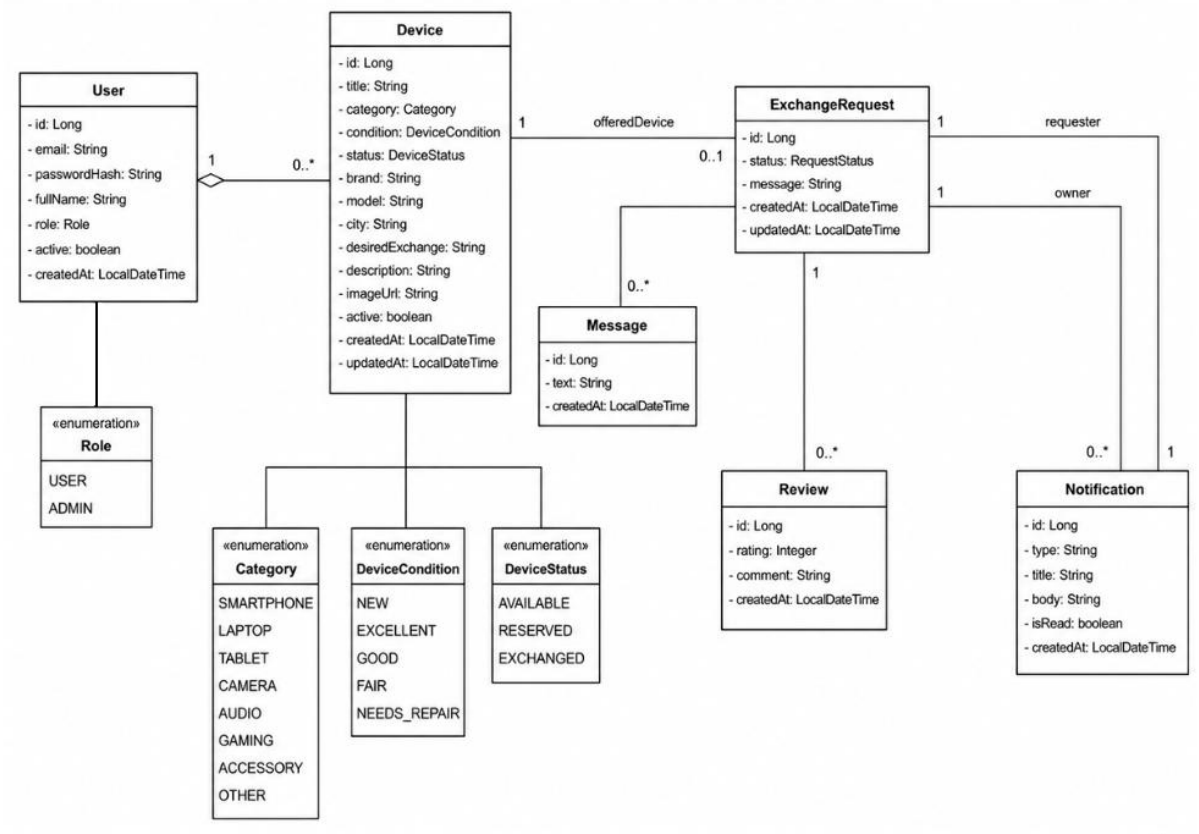
5

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ



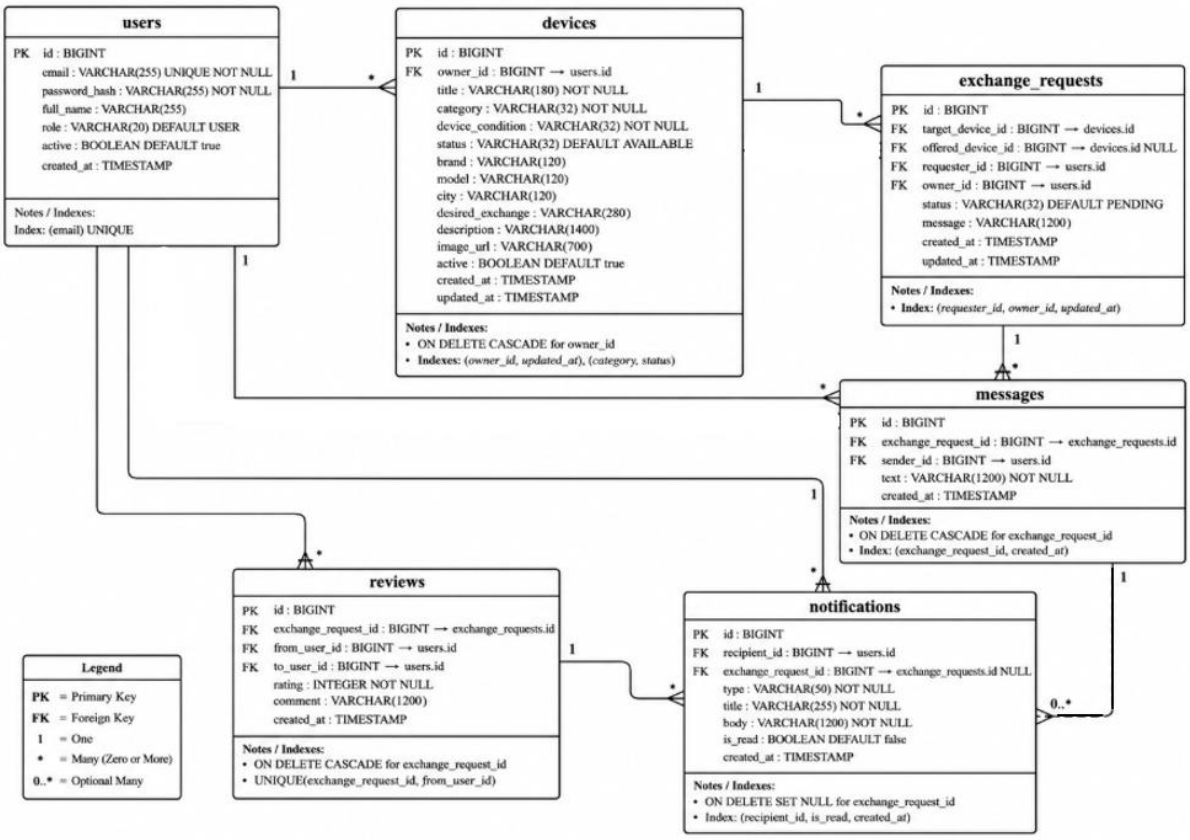
6

ДІАГРАМА ДОМЕННОЇ МОДЕЛІ



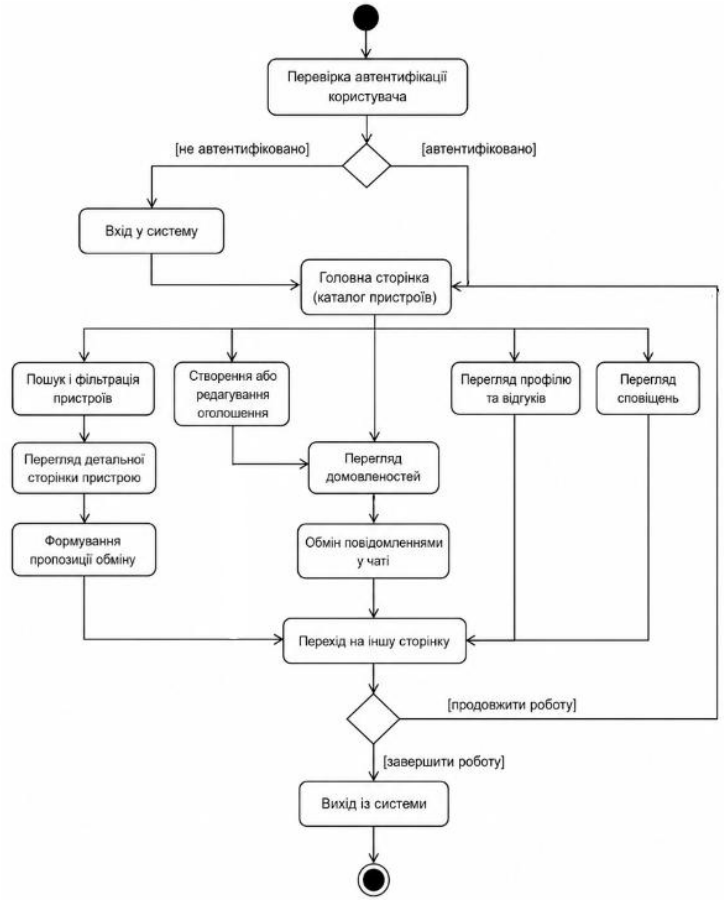
7

СТРУКТУРА БАЗИ ДАНИХ



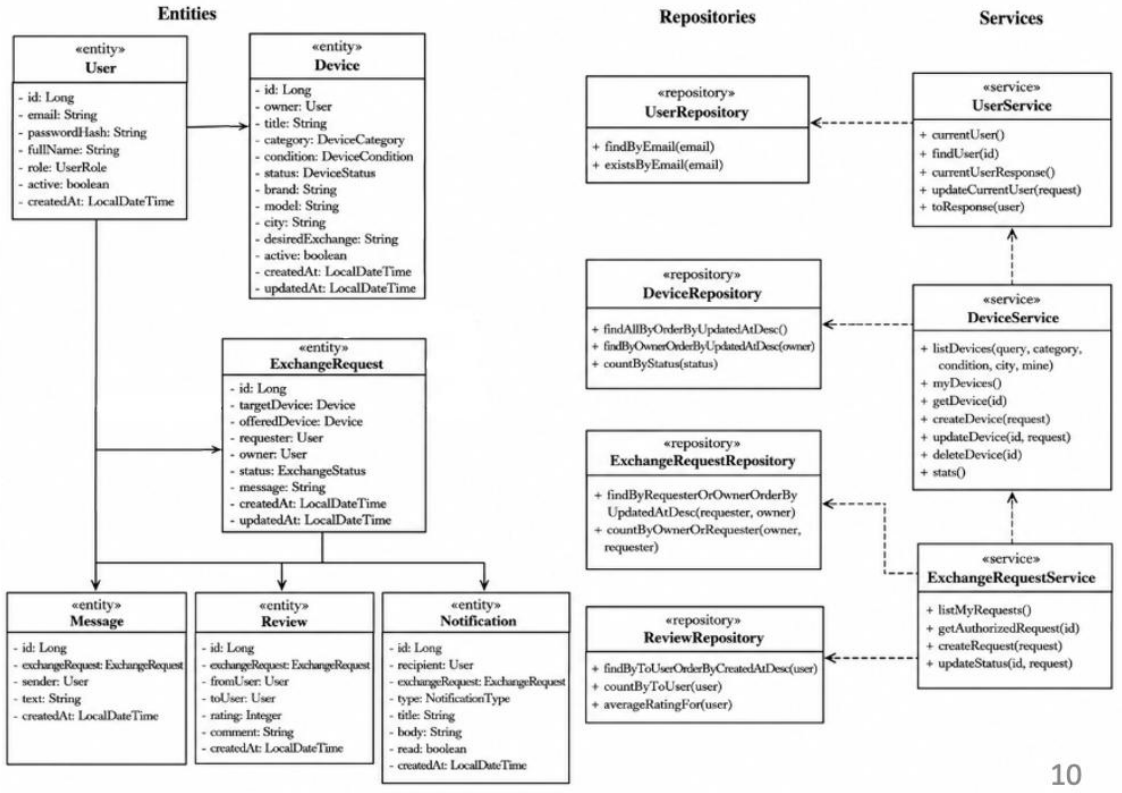
8

ДІАГРАМА АКТИВНОСТІ НА ПЛАТФОРМІ



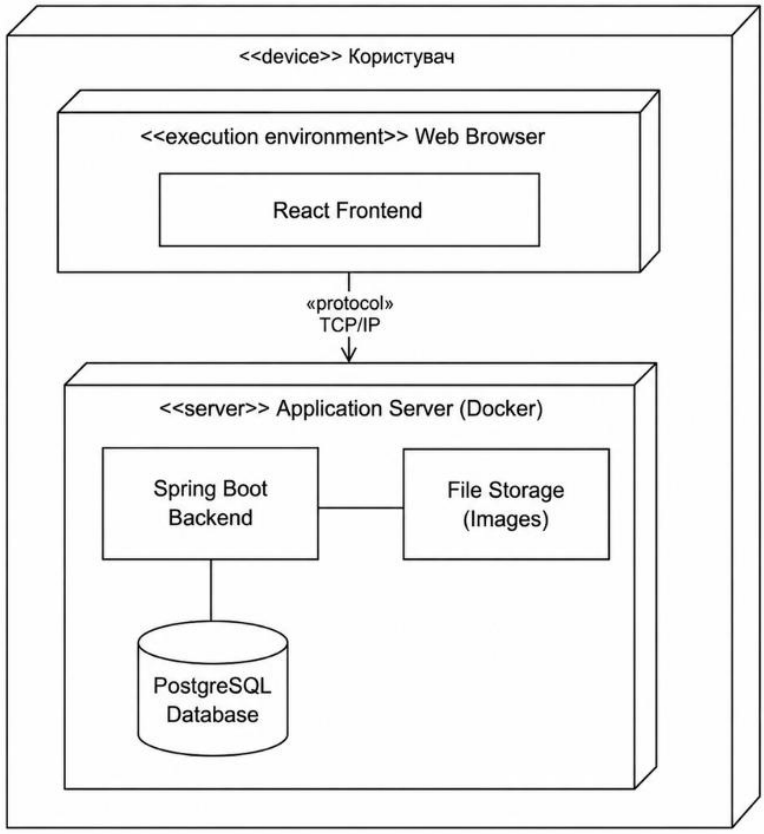
9

ДІАГРАМА КЛАСІВ



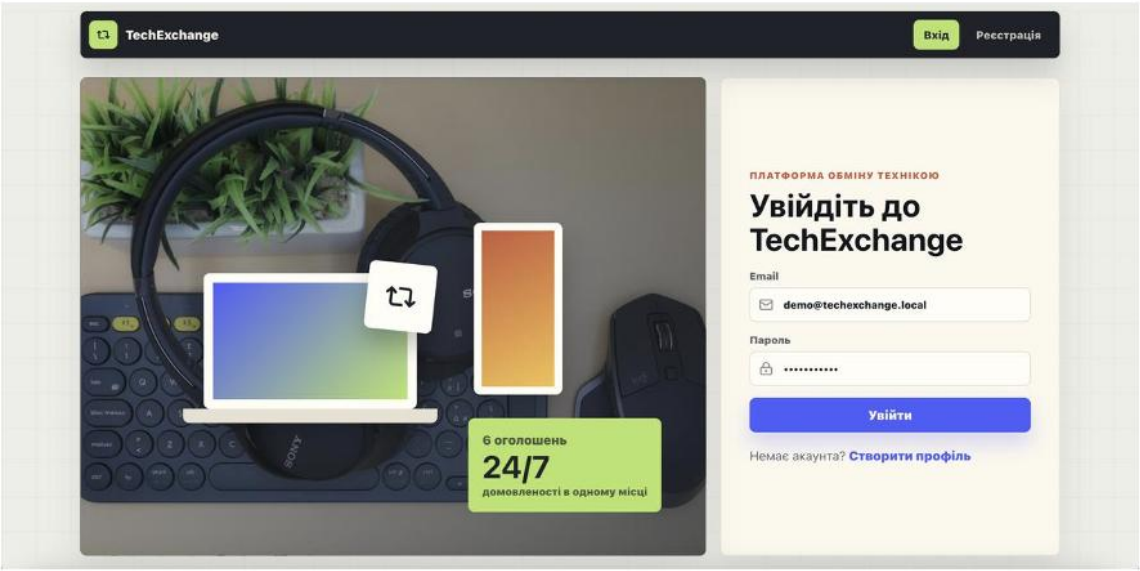
10

ДІАГРАМА РОЗГОРТАННЯ



11

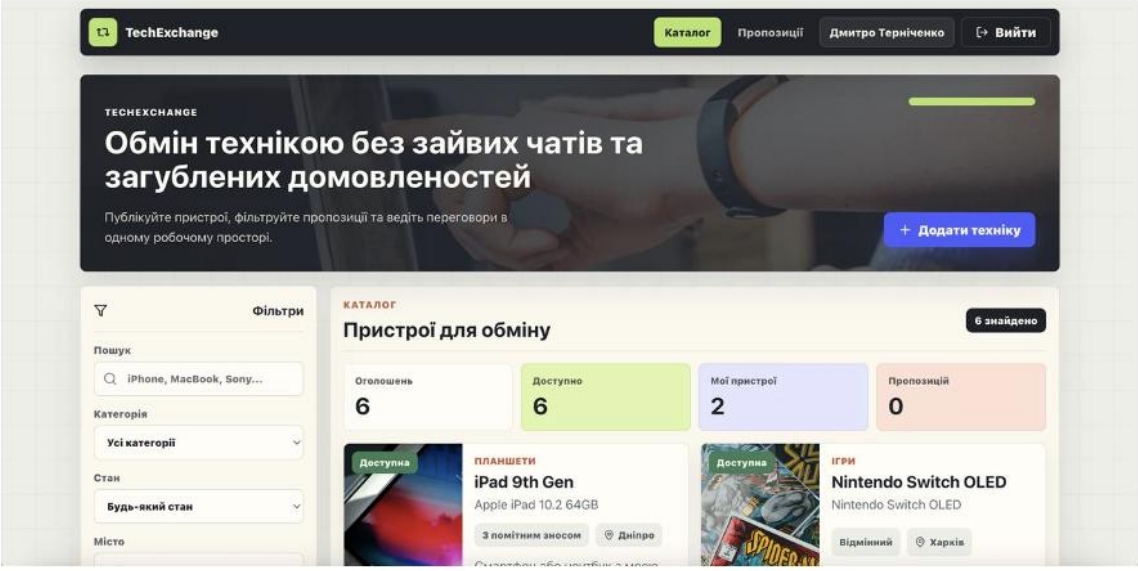
ЕКРАННІ ФОРМИ



ВХІД

12

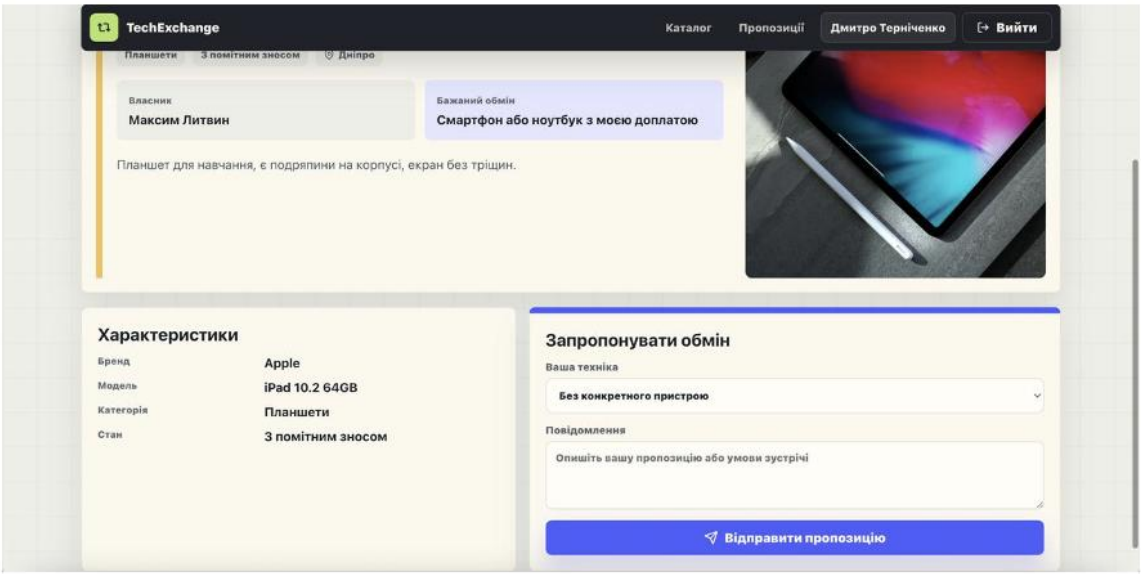
ЕКРАННІ ФОРМИ



ГОЛОВНА СТОРІНКА

13

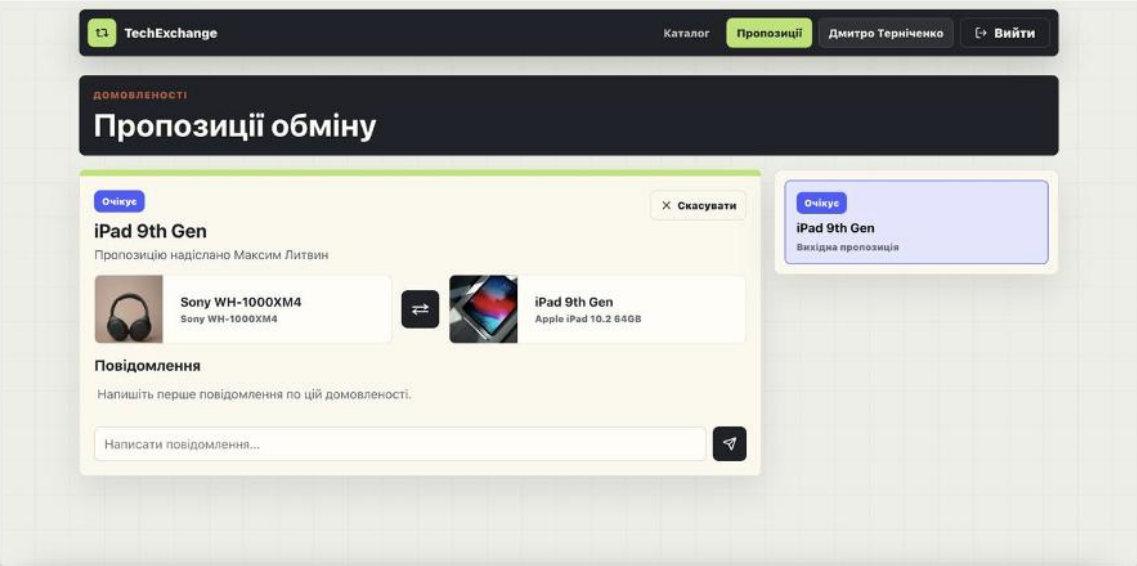
ЕКРАННІ ФОРМИ



СТОРІНКА СТВОРЕННЯ ОГОЛОШЕННЯ

14

ЕКРАННІ ФОРМИ



СТОРІНКА ПРОПОЗИЦІЙ ОБМІНУ

15

РЕЗУЛЬТАТИ ТЕСТУВАННЯ

Напрям тестування	Що перевірено	Результат перевірки
Авторизація	вхід, реєстрація, JWT-токен	токен створюється, приватні сторінки відкриваються лише після входу
Рольовий доступ	права USER та ADMIN	USER не має доступу до admin API, ADMIN може керувати користувачами
Оголошення	CRUD пристроїв	користувач створює, редагує та деактивує лише власні оголошення
Каталог	пошук і фільтри	пристрої коректно відбираються за назвою, категорією, станом і містом
Пропозиції обміну	статуси PENDING → ACCEPTED → ISSUED → RETURNED	система блокує неправильні переходи та резервує пристрої після прийняття
Чат	повідомлення в межах угоди	доступ до чату мають тільки учасники конкретної домовленості
Відгуки	створення оцінки після обміну	відгук дозволений лише після завершення угоди, повторний відгук блокується
Сповіщення	події заявки, повідомлення, статусу	користувач отримує notification після ключових дій
Адмін-модуль	керування користувачами й оголошеннями	адміністратор може деактивувати акаунт або приховати оголошення

17

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

Тези доповідей:

1. Терніченко Д.Ю., Соляник Л.О. Розробка платформи для обміну технікою. Матеріали II Всеукраїнської науково-технічної конференції «Виклики та рішення в програмній інженерії», 26 листопада 2025 року, Київ, Державний університет інформаційно-комунікаційних технологій. Збірник тез. – К.: ДУІКТ, 2025, С.518-520.

2. Терніченко Д.Ю., Гаманюк І.М. Створення вебсервісу для обміну технікою між користувачами. Матеріали VII Всеукраїнської науково-технічної конференції «Застосування програмного забезпечення в інформаційно-комунікаційних технологіях», 23 квітня 2026 року, Київ, ДУІКТ. Збірник тез. – К.: ДУІКТ, 2026, С.432-434.

18

ВИСНОВКИ

1. Проведено аналіз предметної галузі обміну технікою між користувачами та визначено основні проблеми розміщення оголошень, пошуку пристроїв, формування пропозицій обміну й комунікації між учасниками.
2. Досліджено існуючі програмні рішення для розміщення та пошуку техніки, зокрема OLX, Facebook Marketplace, eBay, Swappa, Shpock і Telegram-групи.
3. Визначено їхні переваги та недоліки, серед яких відсутність повноцінного механізму обміну, недостатній контроль статусу домовленостей і слабка структурованість пропозицій.
4. Сформовано функціональні та нефункціональні вимоги до платформи TechExchange, зокрема керування оголошеннями, каталог пристроїв, фільтрація, пропозиції обміну, вбудований чат, відгуки та сповіщення.
5. Обґрунтовано вибір технологій реалізації: Java 17 і Spring Boot — для серверної логіки та REST API, Spring Security JWT — для захисту доступу, PostgreSQL і Liquibase — для збереження даних та міграцій БД, React і Vite — для web-інтерфейсу, Swagger UI — для документації API, Docker Compose — для розгортання системи.
6. Спроєктовано архітектуру TechExchange відповідно до вимог: розділено frontend, backend і PostgreSQL, визначено структуру БД, доменну модель, UML-діаграми та схему розгортання.
7. Реалізовано основні модулі згідно з функціональними вимогами: авторизацію, каталог пристроїв, CRUD оголошень, пропозиції обміну, чат, відгуки, сповіщення та адміністрування та проведено тестування відповідності системи вимогам.

19

ДОДАТОК Б. ЛІСТИНГИ ПРОГРАМНИХ МОДУЛІВ

Листинг 1. Головний клас застосунку

```
package com.example.techexchange;

import org.springframework.boot.SpringApplication;
import
org.springframework.boot.autoconfigure.SpringBootApp
lication;

@SpringBootApplication
public class TechExchangeApplication {
   public static void main(String[] args) {

SpringApplication.run(TechExchangeApplication.class,
args);
   }
}
```

Листинг 2. Сутність користувача

```
@Entity
@Table(name = "users")
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor @Builder
public class User {
   @Id
   @GeneratedValue(strategy =
GenerationType.IDENTITY)
   private Long id;

   @Column(nullable = false, unique = true, length =
255)
   private String email;

   @Column(name = "password_hash", nullable = false)
   private String passwordHash;

   @Column(name = "full_name", length = 255)
   private String fullName;

   @Enumerated(EnumType.STRING)
   @Column(nullable = false, length = 20)
   @Builder.Default
   private UserRole role = UserRole.USER;

   @Column(nullable = false)
   @Builder.Default
   private boolean active = true;

   @CreationTimestamp
   @Column(name = "created_at", updatable = false)
   private LocalDateTime createdAt;
}
```

Листинг 3. Сутність оголошення техніки

```
@Entity
@Table(name = "devices")
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor @Builder
public class Device {
   @Id
   @GeneratedValue(strategy =
GenerationType.IDENTITY)
   private Long id;

   @ManyToOne(fetch = FetchType.LAZY, optional =
false)
   @JoinColumn(name = "owner_id", nullable = false)
   private User owner;

   @Column(nullable = false, length = 180)
   private String title;

   @Enumerated(EnumType.STRING)
   @Column(nullable = false, length = 32)
   private DeviceCategory category;

   @Enumerated(EnumType.STRING)
   @Column(name = "device_condition", nullable =
false, length = 32)
   private DeviceCondition condition;

   @Enumerated(EnumType.STRING)
   @Column(nullable = false, length = 32)
   @Builder.Default
   private DeviceStatus status =
DeviceStatus.AVAILABLE;

   @Column(nullable = false)
   @Builder.Default
   private boolean active = true;

   private String brand;
   private String model;
   private String city;

   @Column(name = "desired_exchange", length = 280)
   private String desiredExchange;

   @Column(length = 1400)
   private String description;

   @Column(name = "image_url", length = 700)
   private String imageUrl;

   @CreationTimestamp
   private LocalDateTime createdAt;

   @UpdateTimestamp
   private LocalDateTime updatedAt;
}
```

Листинг 4. Сутність пропозиції обміну

```
@Entity
@Table(name = "exchange_requests")
@Getter @Setter
@NoArgsConstructor @AllArgsConstructor @Builder
public class ExchangeRequest {
   @Id
```

```
    @GeneratedValue(strategy =
GenerationType.IDENTITY)
    private Long id;

    @ManyToOne(fetch = FetchType.LAZY, optional =
false)
    @JoinColumn(name = "target_device_id", nullable =
false)
    private Device targetDevice;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "offered_device_id")
    private Device offeredDevice;

    @ManyToOne(fetch = FetchType.LAZY, optional =
false)
    @JoinColumn(name = "requester_id", nullable =
false)
    private User requester;

    @ManyToOne(fetch = FetchType.LAZY, optional =
false)
    @JoinColumn(name = "owner_id", nullable = false)
    private User owner;

    @Enumerated(EnumType.STRING)
    @Column(nullable = false, length = 32)
    @Builder.Default
    private ExchangeStatus status =
ExchangeStatus.PENDING;

    @Column(length = 1200)
    private String message;

    @CreationTimestamp
    private LocalDateTime createdAt;

    @UpdateTimestamp
    private LocalDateTime updatedAt;
}
```

Листинг 5. Контролер оголошень

```
@RestController
@RequestMapping("/api/devices")
@RequiredArgsConstructor
@Tag(name = "Техніка")
public class DeviceController {
    private final DeviceService deviceService;

    @GetMapping
    public List<DeviceResponse> listDevices(
            @RequestParam(required = false) String query,
            @RequestParam(required = false)
DeviceCategory category,
            @RequestParam(required = false)
DeviceCondition condition,
            @RequestParam(required = false) String city,
            @RequestParam(required = false) Boolean mine)
{
        return deviceService.listDevices(query, category,
condition, city, mine);
    }

    @GetMapping("/my")
    public List<DeviceResponse> myDevices() {
        return deviceService.myDevices();
    }

    @GetMapping("/stats")
    public MarketplaceStatsResponse stats() {
        return deviceService.stats();
    }

    @GetMapping("/{id}")
    public DeviceResponse getDevice(@PathVariable
Long id) {
        return deviceService.getDevice(id);
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public DeviceResponse createDevice(@Valid
@RequestBody DeviceRequest request) {
        return deviceService.createDevice(request);
    }

    @PutMapping("/{id}")
    public DeviceResponse updateDevice(@PathVariable
Long id,
                                       @Valid @RequestBody
DeviceRequest request) {
        return deviceService.updateDevice(id, request);
    }

    @DeleteMapping("/{id}")
    @ResponseStatus(HttpStatus.NO_CONTENT)
    public void deleteDevice(@PathVariable Long id) {
        deviceService.deleteDevice(id);
    }
}
```

Листинг 6. Сервіс роботи з оголошеннями

```
@Service
@RequiredArgsConstructor
public class DeviceService {
    private final DeviceRepository deviceRepository;
    private final ExchangeRequestRepository
exchangeRequestRepository;
    private final UserService userService;

    @Transactional(readOnly = true)
    public List<DeviceResponse> listDevices(String
query, DeviceCategory category,
                                            DeviceCondition condition,
String city, Boolean mine) {
        User current = userService.currentUser();

        return
deviceRepository.findAllByOrderByUpdatedAtDesc().st
ream()
                .filter(device -> Boolean.TRUE.equals(mine)
                        ?
device.getOwner().getId().equals(current.getId())
                        : device.isActive())
                .filter(device -> category == null ||
device.getCategory() == category)
```

```
            .filter(device -> condition == null ||
device.getCondition() == condition)
            .filter(device -> city == null ||
device.getCity().toLowerCase().contains(city.toLowerCa
se()))
            .map(device -> toResponse(device, current))
            .toList();
    }

    @Transactional
    public DeviceResponse createDevice(DeviceRequest
request) {
        User current = userService.currentUser();

        Device device = Device.builder()
            .owner(current)
            .title(request.getTitle())
            .category(request.getCategory())
            .condition(request.getCondition())
            .status(request.getStatus() == null ?
DeviceStatus.AVAILABLE : request.getStatus())
            .brand(request.getBrand())
            .model(request.getModel())
            .city(request.getCity())

.desiredExchange(request.getDesiredExchange())
            .description(request.getDescription())
            .imageUrl(request.getImageUrl())
            .active(true)
            .build();

        return toResponse(deviceRepository.save(device),
current);
    }

    @Transactional
    public void deleteDevice(Long id) {
        User current = userService.currentUser();
        Device device = findDevice(id);

        if
(!device.getOwner().getId().equals(current.getId())) {
            throw new AccessDeniedException("Можна
змінювати лише власні оголошення");
        }

        device.setActive(false);
        device.setStatus(DeviceStatus.RESERVED);
    }

    public Device findDevice(Long id) {
        return deviceRepository.findById(id)
            .orElseThrow(() -> new
ResourceNotFoundException("Оголошення не
знайдено"));
    }
}
```

Листинг 7. Контролер пропозицій обміну

```
@RestController
@RequestMapping("/api/exchange-requests")
@RequiredArgsConstructor
@Tag(name = "Пропозиції обміну")
public class ExchangeRequestController {
    private final ExchangeRequestService
exchangeRequestService;

    @GetMapping
    public List<ExchangeRequestResponse>
listMyRequests() {
        return exchangeRequestService.listMyRequests();
    }

    @PostMapping
    @ResponseStatus(HttpStatus.CREATED)
    public ExchangeRequestResponse createRequest(
            @Valid @RequestBody
ExchangeRequestCreateRequest request) {
        return
exchangeRequestService.createRequest(request);
    }

    @PutMapping("/{id}/status")
    public ExchangeRequestResponse updateStatus(
            @PathVariable Long id,
            @Valid @RequestBody
ExchangeStatusUpdateRequest request) {
        return exchangeRequestService.updateStatus(id,
request);
    }
}
```

Листинг 8. Контролер повідомлень

```
@RestController
@RequestMapping("/api/messages")
@RequiredArgsConstructor
@Tag(name = "Повідомлення")
public class MessageController {
    private final MessageService messageService;

    @GetMapping("/request/{requestId}")
    public List<MessageResponse>
listMessages(@PathVariable Long requestId) {
        return messageService.listMessages(requestId);
    }

    @PostMapping("/request/{requestId}")
    @ResponseStatus(HttpStatus.CREATED)
    public MessageResponse
createMessage(@PathVariable Long requestId,
                                @Valid @RequestBody
MessageRequest request) {
        return messageService.createMessage(requestId,
request);
    }
}
```

Листинг 9. Налаштування безпеки

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
@RequiredArgsConstructor
public class SecurityConfig {
    private final JwtAuthenticationFilter jwtAuthFilter;
    private final UserDetailsServiceImpl
userDetailsService;
```

```
    @Bean
    public SecurityFilterChain filterChain(HttpSecurity
http) throws Exception {
        http
                .csrf(AbstractHttpConfigurer::disable)
                .cors(cors ->
cors.configurationSource(corsConfigurationSource()))
                .sessionManagement(s ->
s.sessionCreationPolicy(SessionCreationPolicy.STATEL
ESS))
                .authorizeHttpRequests(auth -> auth
                        .requestMatchers(
                                "/api/auth/**",
                                "/api-docs/**",
                                "/swagger-ui/**",
                                "/swagger-ui.html"
                        ).permitAll()
                        .anyRequest().authenticated()
                )

.authenticationProvider(authenticationProvider())
                .addFilterBefore(jwtAuthFilter,
UsernamePasswordAuthenticationFilter.class);

        return http.build();
    }

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