МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА СЕРВІСУ ПРЕЗЕНТАЦІЇ ТОВАРІВ З ВИКОРИСТАННЯМ VUE.JS»**

Виконала: студентка 2 курсу, групи 8.1214-з-дн
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
М.О. Слабеняк
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, доцент, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Слабеняк Марині Олексіївні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка сервісу презентації товарів з використанням Vue.js

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 28 » лютого 2025 року № 327-с

2. Строк подання студентом роботи 30.01.2026 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Проєктування сервісу.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 14.03.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	28.03.2025	
2.	Збір вихідних даних.	18.04.2025	
3.	Обробка методичних та теоретичних	16.06.2025	
	джерел.		
4.	Розробка першого та другого розділу.	15.09.2025	
5.	Розробка третього розділу.	29.12.2025	
6.	Оформлення та нормоконтроль		
	кваліфікаційної роботи магістра.	26.01.2026	
7.	Захист кваліфікаційної роботи.	13.02.2026	

Студент _____ М.О. Слабеняк
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка сервісу презентації товарів з використанням Vue.js»: 97 с., 25 рис., 2 табл., 21 джерело, 3 додатки.

E-COMMERCE, FASTAPI, JWT, PINIA, POSTGRESQL, PYTHON, SQLALCHEMY, TAILWINDCSS, VEEVALIDATE, VUE.JS 3.

Об’єкт дослідження – інформаційні системи електронної комерції.

Предмет дослідження – методи та засоби розробки універсальних, масштабованих платформ для електронної комерції на основі FastAPI, Vue.js 3, PostgreSQL і мови програмування Python.

Мета роботи: розробка сучасної e-Commerce платформи для інтернет-торгівлі з повним циклом замовлень, адміністративною панеллю і адаптивним дизайном.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

При розробці інтернет-магазину був проведений аналіз предметної області, обрано та спроєктовано архітектуру системи за допомогою UML у вигляді наборів діаграм варіантів використання, послідовностей, компонентів, розгортання та діяльності. Реалізовано повнофункціональний інтернет-магазин з використанням сучасного стеку технологій FastAPI, Vue.js 3 та PostgreSQL.

В результаті роботи було повністю реалізовано основний функціонал інтернет-магазину, здійснено повний цикл розробки та впровадження універсальної вебплатформи – сайту для презентації товарів реального виробника м’яких меблів власного виробництва для тварин, з можливостями зв’язку з виробником для замовлення за конкретними параметрами (розмір, колір та інше), з використанням Vue.js.

SUMMARY

Master’s qualifying paper “Development of a Product Presentation Service Using Vue.js”: 97 pages, 25 figures, 2 tables, 21 references, 3 supplements.

E-COMMERCE, FASTAPI, JWT, PINIA, POSTGRESQL, PYTHON, SQLALCHEMY, TAILWINDCSS, VEEVALIDATE, VUE.JS 3.

The object of research is e-commerce information systems.

The subject of research is methods and tools for developing universal, scalable e-commerce platforms based on FastAPI, Vue.js 3, PostgreSQL, and the Python programming language.

The objective of the work is to develop a modern e-commerce platform for online retail featuring a full order cycle, an administrative panel, and a responsive design.

The research methods include methods of software requirements gathering and analysis, as well as methods of software modeling, design, construction, and testing.

During the development of the online store, a domain analysis was conducted, and the system architecture was designed using UML in the form of use case, sequence, component, deployment, and activity diagrams. A fully functional online store was implemented using a modern technology stack of FastAPI, Vue.js 3, and PostgreSQL.

As a result of the work, the core functionality of the online store was fully implemented, and a complete cycle of development and deployment of a universal web platform was carried out. The project resulted in a website for the presentation of products from a real manufacturer of custom-made soft furniture for pets, providing the ability to contact the manufacturer to order products with specific parameters (size, color, etc.) using Vue.js.

ЗМІСТ

Завдання на кваліфікаційну роботу ……..2
Реферат 4
Summary 5
Вступ..... 8
1 Технічне завдання на розробку сервісу презентації товарів..... 11
1.1 Терміни та визначення..... 11
1.2 Функціональні вимоги..... 14
1.2.1 Призначення і цілі створення системи 15
1.2.2 Загальні функціональні можливості системи 16
1.3 Нефункціональні вимоги..... 17
1.3.1 Інтерфейс користувача 17
1.3.2 Підтримка браузерів 17
1.3.3 Вимоги до продуктивності..... 18
1.3.4 Вимоги до безпеки 18
1.4 Опис предметної області 18
1.5 Опис платформи для продажу/презентації товарів 21
1.6 Порівняння з подібними системами..... 22
1.6.1 Огляд ринку електронної комерції в Україні..... 22
1.6.2 E-commerce платформи: огляд найпопулярніших рішень..... 30
1.6.3 Ознаки успішного інтернет-магазину..... 34
2 Проєктування сервісу презентації товарів..... 37
2.1 Обґрунтування вибору інструментів для розробки..... 37
2.1.1 Опис стеку технологій для Backend..... 37
2.1.2 Опис стеку технологій для Frontend 42
2.1.3 Архітектура проєкту..... 46
2.1.4 Опис бази даних..... 50
2.2 Розробка структурних і поведінкових діаграм 54

3 Реалізація та тестування 61
3.1 Реалізація користувацького інтерфейсу 61
3.2 Опис завершених компонент проєкту 64
3.3 Опис функцій платформи та їхньої завершеності 67
3.4 Тестування функціоналу системи 69
Висновки 72
Перелік посилань 73
Додаток А Лістинг коду для основного макету платформи 76
Додаток Б Лістинг коду для адміністративного макету платформи 80
Додаток В API Examples & Testing scripts 82

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій електронна комерція стала невід'ємною складовою глобальної економіки. Все більше підприємств і приватних осіб переходять до онлайн-формату торгівлі, що зумовлює зростаючу потребу у створенні зручних, надійних та масштабованих вебрішень. Розробка інтернет-магазину є одним із найбільш затребуваних напрямів у сфері вебпрограмування.

Особливої актуальності ця тема набуває в контексті використання сучасних технологій, таких як Vue.js 3, який підтримує сучасний Composition API. Набір офіційних правил ESLint-правил забезпечує: перевірку структури компонентів; дотримання рекомендацій коду Vue; уникнення помилок у SFC (Single File Components); коректність використання props, emits, reactivity API. Vue/recommended допомагає тримати компоненти чистими, логічно розділеними та без помилок. Python, своєю чергою, є однією з найпопулярніших мов програмування у світі, завдяки своїй простоті, універсальності та широкому набору бібліотек. Застосований стек технологій відкриває широкі можливості для інтеграції інтерфейсів, баз даних, систем електронних платежів і засобів аналітики.

Отже, розробка інтернет-магазину з використанням Vue.js 3 і Python є **актуальною** як з технічної, так і з економічної точки зору. Такий проєкт дозволяє не лише поглибити практичні навички розробки вебдодатків, а й відповідає реальним потребам бізнесу в цифровізації та автоматизації торговельної діяльності.

Мета – розробка сучасної e-Commerce платформи для інтернет-торгівлі з повним циклом замовлень, адміністративною панеллю і адаптивним дизайном.

Досягнення поставленої мети передбачає реалізацію наступних **завдань**:

- проаналізувати вимоги для розробки сервісу презентації товарів;
- дослідити предметну область розробки і здійснити порівняльний аналіз платформ для продажу/презентації товарів;
- обґрунтувати вибір інструментів для розробки;
- розробити структурні і поведінкові діаграми для розроблюваного сервісу;
- реалізувати користувацький інтерфейс і описати усі компоненти системи;
- протестувати функціонал системи.

Об’єкт дослідження – інформаційні системи електронної комерції.

Предмет дослідження – методи та засоби розробки універсальних, масштабованих платформ для електронної комерції на основі FastAPI, Vue.js 3, PostgreSQL і мови програмування Python.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

Основні положення і результати кваліфікаційної роботи доповідались, обговорювалися та знайшли схвалення на Шістнадцятій Всеукраїнській, Двадцять третій регіональній науковій конференції молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя, 24-25 квітня 2025 р.), за результатами яких опубліковано тези «Ефективність використання vue.js для створення компонентно-орієнтованих вебдодатків: досвід розробки сервісу презентації товарів».

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку посилань та трьох додатків. Загальний обсяг роботи становить 97 сторінок. Додатки викладено на 22 сторінках. У переліку посилань 21 використане джерело.

Перший розділ присвячений складанню технічного завдання на розробку сервісу презентації товарів: окреслені терміни та визначення, визначені функціональні та нефункціональні вимоги, описані предметна

область та характеристики платформи для продажу/презентації товарів та здійснено порівняння розроблюваної платформи з існуючими подібними системами. У другому розділі наведено обґрунтування вибору інструментів для розробки, проілюстровано структурні та поведінкові діаграми. В третьому розділі описано реалізацію користувацького інтерфейсу, ключових компонентів проєкту та протестовано функціонал системи. В додатках наведено програмну реалізацію сервісу презентації товарів.

1 ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ СЕРВІСУ ПРЕЗЕНТАЦІЇ ТОВАРІВ

1.1 Терміни та визначення

Назва проєкту – повнофункціональний сервіс презентації товарів на сучасній платформі для інтернет-торгівлі PetShop.

Сервіси презентації товарів – це інструменти та процеси, які допомагають продавцям демонструвати та переконувати клієнтів у цінності товару чи послуги.

Презентація «очима» спеціаліста з продажу – повний пакет продажів, який надається потенційному клієнту з метою спонукати його здійснити купівлю продукту чи послуги [1].

Сервіси презентації товарів, які можна використовувати як інтернет-магазини, охоплюють як маркетплейси (наприклад, OLX, Prom, Rozetka), так і конструктори сайтів (наприклад, Shop-Express), що дозволяють створити власний онлайн-магазин. Маркетплейси надають готову платформу з аудиторією, а конструктори дають можливість побудувати власний унікальний бренд.

Основна мета інтернет-магазинів – забезпечити зручність та швидкість покупок, надання широкого вибору товарів, забезпечення безпеки оплати та доставки, а також задоволення користувачів шляхом надання якісного сервісу та підтримки клієнтів, тобто створити зручну та доступну платформу для електронних покупок.

Особливості функціонування інтернет-магазинів зумовлені специфікою електронної комерції та сучасними інформаційно-комунікаційними технологіями [2]. Насамперед, ключовою характеристикою є онлайн-доступність, яка забезпечує можливість здійснення покупок незалежно від просторових та часових обмежень. Користувачі мають змогу отримувати доступ до торговельної платформи з будь-якого місця за наявності

підключення до мережі Інтернет, що істотно підвищує зручність і гнучкість споживчої поведінки.

Важливою особливістю інтернет-магазинів є широкий асортимент товарів і послуг, який, на відміну від традиційних торговельних точок, не обмежується фізичними площами. Це дозволяє задовольняти різноманітні потреби споживачів, у тому числі щодо спеціалізованих, нішевих або ексклюзивних товарів. Така варіативність сприяє підвищенню конкурентоспроможності суб’єктів електронної торгівлі.

Не менш значущим аспектом є зручність і оперативність здійснення покупок. Функціональні можливості інтернет-магазинів дають змогу швидко здійснювати пошук товарів, порівнювати їх характеристики, ціни та умови придбання, оформлювати замовлення й проводити оплату з мінімальними часовими витратами. Автоматизація процесів замовлення та оплати сприяє підвищенню ефективності обслуговування клієнтів і зменшенню бар’єрів на шляху до завершення покупки.

Окрему увагу слід приділити комфортності та безпеці споживчого досвіду. Можливість здійснювати покупки без фізичного відвідування магазину знижує психологічний тиск на споживача та мінімізує незручності, пов’язані з чергами чи обмеженим режимом роботи. Крім того, сучасні інтернет-магазини впроваджують механізми захисту персональних даних і фінансової інформації клієнтів, що є важливою передумовою формування довіри до електронної комерції.

Суттєвою перевагою інтернет-магазинів є різноманітність способів оплати та доставки, що дозволяє адаптувати сервіс до індивідуальних потреб споживачів. Використання банківських карток, електронних платіжних систем, цифрових гаманців, а також різних форматів доставки (кур’єрська служба, поштові оператори, пункти самовивозу) підвищує рівень зручності та доступності електронної торгівлі.

Важливу роль у формуванні споживчої довіри відіграють механізми зворотного зв’язку та системи відгуків. Можливість ознайомлення з оцінками

інших покупців і коментарями щодо якості товарів та обслуговування сприяє прийняттю більш обґрунтованих рішень. Додатково інтернет-магазини забезпечують комунікацію з клієнтами через онлайн-чати, електронну пошту або телефонну підтримку, що підвищує рівень сервісу.

Переваги інтернет-магазинів як складової сучасної торговельної системи полягають, зокрема, у глобальному характері доступу, що дозволяє здійснювати продажі на міжнародних ринках без необхідності фізичної присутності. Це розширює потенційну аудиторію споживачів та створює передумови для масштабування бізнесу.

Крім того, інтернет-магазини забезпечують економію витрат, пов’язану зі зменшенням потреби в оренді торговельних площ та утриманні значного персоналу. Це дозволяє формувати більш конкурентоспроможну цінову політику, а також активно застосовувати інструменти стимулювання попиту, зокрема знижки, акційні пропозиції та програми лояльності.

Ще однією важливою перевагою є простота порівняння товарів, яка досягається завдяки доступності великого обсягу інформації про характеристики, ціни та споживчі властивості продукції. Аналітичний підхід до вибору товарів сприяє підвищенню задоволеності клієнтів та зниженню ймовірності помилкових покупок.

Комплекс взаємопов’язаних аспектів, пов’язаних із їх функціонуванням та управлінням інтернет-магазинів, охоплює процес розробки та структурної побудови інтернет-магазину, включаючи вибір платформи, архітектуру, функціональне наповнення, дизайн та користувацький інтерфейс. Окремим напрямом є маркетинг і просування, що передбачає застосування SEO-оптимізації, контент-маркетингу, соціальних мереж, цифрової реклами та інших інструментів залучення трафіку. Також аналізуються операційні процеси електронної комерції, зокрема управління асортиментом, складським обліком, замовленнями, оплатою, доставкою та післяпродажним обслуговуванням. Важливе місце займають питання аналітики та оптимізації, які включають збір і аналіз даних про поведінку користувачів, рівень

конверсії, ефективність маркетингових кампаній та фінансові показники діяльності. Не менш актуальними є аспекти інформаційної безпеки та захисту даних, що охоплюють протидію шахрайству, забезпечення конфіденційності та надійності електронних транзакцій.

Узагальнюючи, слід зазначити, що ключовим завданням інтернет-магазину є досягнення високого рівня конверсії, тобто трансформації відвідувачів у реальних покупців. Для цього необхідне поєднання якісного товарного наповнення з інтуїтивно зрозумілим інтерфейсом, зручними механізмами пошуку, замовлення та оплати, що в сукупності забезпечує ефективність функціонування електронної торговельної платформи.

1.2 Функціональні вимоги

Формування функціональних вимог – один з ключових етапів проєктування будь-якої інформаційної системи, що дозволяє чітко визначити сферу її застосування та пріоритети розвитку [3]. В рамках даної кваліфікаційної роботи функціонал платформи поділяється на дві основні групи: функції для покупців і функції для продавців, а також роль адміністратора, яка забезпечує підтримку та нагляд за роботою системи.

Основні функції для покупців:

- реєстрація та авторизація користувача;
- перегляд каталогу товарів у зручному форматі (розділ за категоріями, пошук, фільтрація);
- перегляд детальної інформації про товар;
- додавання товару в кошик, формування та оформлення замовлення;
- можливість залишати відгук чи оцінку щодо товару;
- перегляд статусу своїх замовлень.

Основні функції для продавців:

- реєстрація та верифікація акаунта продавця;

– додавання, редагування та видалення власних товарів, керування категоріями;
– перегляд переліку та статусів замовлень, обробка замовлень (підтвердження, відміна, доставка);
– управління зворотним зв’язком (відгуки покупців, повідомлення);
– персональна сторінка продавця із статистикою продажів та налаштуваннями профілю.

Функції адміністратора:

– модерування товарів і категорій;
– управління користувачами (блокування/розблокування, перегляд статистики);
– формування аналітичних звітів;
– відстеження активності та безпеки системи.

Вимоги до платформи також передбачають:

– адаптивний інтерфейс для мобільних пристроїв;
– високу швидкодію та стійкість до збоїв;
– прозорість і зручність навігації;
– надійний захист персональних даних і платіжної інформації.

Задоволення зазначених вимог стане запорукою конкурентоспроможності створеної платформи на ринку електронної комерції.

1.2.1 Призначення і цілі створення системи

Метою проєкту є створити сучасний, швидкий і зручний вебсервіс для демонстрації товарів (каталогу), який забезпечить:

– привабливу візуальну презентацію товарів;
– зручну навігацію, пошук та фільтрацію;
– швидкий доступ з мобільних пристроїв;
– комфортну інтеграцію з backend;

– підготовленість до подальшого розвитку (PWA, SSR, e-commerce розширення).

Функціональне призначення системи – реалізувати можливість вибору товарів через інтернет-магазин.

Експлуатаційне призначення системи – інтернет-магазин може експлуатуватися адміністратором і користувачами на різних рівнях доступу.

Мета створення системи – створення сучасної повнофункціональної e-Commerce платформи для інтернет-торгівлі з повним циклом замовлень, адміністративною панеллю та адаптивним дизайном.

1.2.2 Загальні функціональні можливості системи

Функціонал користувача має включати:

– аутентифікацію: реєстрацію, вхід, профіль з JWT;
– каталог: товари з фільтрацією, пошуком і пагінацією;
– кошик: додавання товарів, розрахунок вартості;
– замовлення: повний цикл від оформлення до відстеження;
– відгуки: система рейтингів і коментарів.

Адміністративний функціонал має надавати такі можливості:

– товари: CRUD операції з товарами і категоріями;
– замовлення: управління статусами і обробка;
– користувачі: перегляд та управління акаунтами;
– аналітика: статистика продажів та активності.

Технічні особливості мають включати:

– адаптивний дизайн для всіх пристроїв;
– Real-time валідацію форм з VeeValidate;
– State management з Pinia;
– REST API з автодокументацією;
– JWT авторизацію із захищеними роутами.

1.3 Нефункціональні вимоги

Основними нефункціональними вимогами до системи є наступні:

– надійність: fallback для відсутнього зображення; обробка помилок API;
– масштабованість: можливість підключення до великих каталогів товарів;
– тестування: unit-тести компонентів;
– документація: README, інструкція запуску, API contract (OpenAPI spec).

1.3.1 Інтерфейс користувача

Інтернет-платформа повинна відображати коректно інтерфейс користувача на будь-якому пристрої.

UX / UI вимоги:

– сучасний, чистий, повністю адаптивний дизайн (Mobile First), фокус на фото товару;
– сучасний інтерфейс з TailwindCSS;
– інтерактивні елементи (модальні вікна, дропдауни);
– повідомлення (vue-toastification);
– стани завантаження та помилок;
– відповідність Accessibility.

1.3.2 Підтримка браузерів

Інтернет-платформа повинна бути сумісною з:

– браузерами Chrome 90+, Firefox 88+, Safari 14+, Edge 90+;

– мобільними системами iOS 12+, Android 8+;
– роздільною здатністю: 320px – 1920px+.

1.3.3 Вимоги до продуктивності

Повнофункціональна e-commerce платформа, побудована на сучасному стеку технологій, має забезпечувати швидкодію і зручність для усіх користувачів шляхом використання:

– Lazy loading компонентів;
– кешування API запитів;
– оптимізованих зображень;
– мініфікації та стиснення;
– Tree shaking в Vite.

1.3.4 Вимоги до безпеки

Безпека забезпечується наступним чином:

– JWT автентифікація з refresh токенами з обмеженим терміном дії;
– SQL injection захист через SQLAlchemy ORM;
– захищені маршрути (користувач/адміністратор);
– Input валідація форм на клієнті і сервері;
– CORS налаштування для продакшену;
– Bcrypt хешування паролів.

1.4 Опис предметної області

Предметна область для даного проєкту охоплює електронну комерцію та онлайн-презентацію товарів, включно з бізнес-процесами, пов’язаними з

показом, пошуком, порівнянням і вибором продукції. Розглянемо ключові аспекти предметної області.

Каталог товарів – це основа платформи. Каталог містить: категорії та підкатегорії (структурна ієрархія), товари та їх властивості, опис, фото, характеристики, документація, ціну, знижки та промо-параметри, наявність на складі, фільтри та атрибути (колір, габарити, бренд, тип).

Каталог – це динамічна сутність, яка регулярно оновлюється залежно від асортименту підприємства.

Моделі даних товарів. Кожен товар має набір параметрів: назва, унікальний ідентифікатор, короткий і повний опис, галерея зображень, технічні характеристики, ціна (актуальна/стара), рейтинг (за потреби), наявність, теги / атрибути. Усі ці дані в предметній області взаємопов’язані й повинні мати стандартизовану структуру.

Користувачі системи. У предметній області виділяють кілька типів користувачів:

– покупець/гість – переглядає товари, використовує фільтри, читає описи;
– зареєстрований користувач (опціонально) – отримує доступ до розширених функцій: вибране, історія переглядів;
– адміністратор – додає нові товари, редагує характеристики, змінює статуси;
– маркетолог/контент-менеджер – працює з описами, банерами, категоріями.

Бізнес-процеси предметної області. Основні бізнес-процеси, які взаємодіють із предметною областю:

– додавання нового товару;
– оновлення інформації про товар;
– видалення або приховування товару;
– категоризація та управління товарами в каталозі;
– пошук та фільтрація;
– робота з медіа-матеріалами (фото/відео);

– підготовка інформаційних сторінок (опис категорій, промо-блоки);

– презентація товару користувачеві.

Інформаційні потоки предметної області: потік даних з бекенд-сервера до клієнта (каталог, дані товарів); потік запитів на фільтри, пошук, сортування; потоки оновлень від адміністратора в CMS/API; потоки мультимедійних файлів (завантаження фото товарів).

Предметна область платформи – це електронний каталог товарів з гнучкою структурою, ієрархією, атрибутами та медіа-контентом.

Предметна область дослідження охоплює сукупність теоретичних і практичних аспектів, пов’язаних із функціонуванням та управлінням інтернет-магазином як суб’єктом електронної комерції.

Розглянемо основні тематичні напрями, які виокремлюються у межах дослідження.

По-перше, розглядаються питання розробки та структурної побудови інтернет-магазину, зокрема аналізуються його архітектура, функціональні можливості, дизайн і користувацький інтерфейс. Особливу увагу приділено вибору програмної платформи, процесам розгортання, налаштування та адаптації інтернет-магазину відповідно до вимог бізнесу й очікувань споживачів.

По-друге, досліджується маркетингова складова діяльності інтернет-магазину, що включає формування та реалізацію стратегій просування в цифровому середовищі. Аналізуються інструменти SEO-оптимізації, маркетинг у соціальних мережах, контент-маркетинг, онлайн-реклама та інші методи залучення цільового трафіку і підвищення впізнаваності торговельної платформи.

По-третє, значну увагу приділено операційним процесам електронної комерції, зокрема досліджуються бізнес-моделі інтернет-торгівлі, управління асортиментом товарів, організація складського обліку, процеси оформлення та обробки замовлень, функціонування платіжних систем і логістичних сервісів, а також питання післяпродажного обслуговування і роботи з поверненнями.

По-четверте, у роботі аналізуються аналітичні інструменти та методи оптимізації діяльності інтернет-магазину, які передбачають збір і обробку даних про відвідуваність вебресурсу, поведінку користувачів, рівень конверсії та інші показники ефективності. Розглядаються сучасні засоби вебаналітики, що дозволяють підвищувати результативність управлінських рішень та ефективність функціонування торговельної платформи.

По-п’яте, досліджуються питання інформаційної безпеки та захисту даних, зокрема безпека електронних транзакцій, захист персональної інформації споживачів, протидія шахрайству та кіберзлочинності. Окрема увага приділяється застосуванню технологій шифрування та інших інструментів забезпечення надійності й конфіденційності інформаційних процесів в інтернет-магазині.

1.5 Опис платформи для продажу/презентації товарів

Платформа – це вебдодаток, створений на Vue.js, який виконує функції вітрини (presentation layer) та каталогу.

Платформа призначена для забезпечення:

– привабливої подачі товарів клієнтам;
– швидкого доступу до асортименту;
– простої навігації й фільтрації;
– адаптації під мобільні пристрої;
– інтеграції з backend API та CRM.

Це повнофункціональний інтернет-магазин з головним фокусом на презентації товару, побудований на сучасному стеку технологій.

Основними функціями платформи є:

– відображення каталогу товарів;
– мультимедійна презентація (фото/відео);
– попередній перегляд (quick view);

– сортування за параметрами;
– повнотекстовий або атрибутний пошук;
– фільтрація товарів;
– можливість переходу між категоріями;
– перегляд історії переглядів/вибране (опціонально);
– адаптивність під мобільні екрани.

Переваги використання Vue.js для платформи:

– швидкий рендеринг і мінімізація затримок;
– компонентний підхід → повторне використання UI модулів;
– легка інтеграція з API та headless CMS;
– хороша підтримка ком’юніті та плагінів;
– проста масштабованість;
– легке впровадження SSR через Nuxt.js для SEO.

Платформа – це сучасний Vue.js-додаток, який забезпечує високоякісну презентацію товарів, зручну взаємодію користувачів із каталогом і створений як повноцінний e-commerce продукт.

1.6 Порівняння з подібними системами

1.6.1 Огляд ринку електронної комерції в Україні

Український ринок електронної комерції демонструє стале та динамічне зростання протягом останнього десятиліття, що зумовлено підвищенням рівня проникнення мережі Інтернет, широким використанням смартфонів, а також зростанням цифрової грамотності населення [4; 10-11]. За оцінками міжнародних аналітичних агентств, рівень проникнення Інтернету в Україні у 2023 році перевищив 80%, що створило сприятливі передумови для розвитку онлайн-торгівлі [4]. У 2024 році подальше зростання сектору електронної комерції формується під впливом як глобальних цифрових трендів, так і

специфічних внутрішніх чинників, пов’язаних із воєнно-економічною ситуацією в країні.

Станом на 2023 рік обсяг ринку електронної комерції в Україні оцінюється на рівні понад 6 млрд дол. США, а середньорічний темп зростання (CAGR) у 2020–2023 роках становив близько 20% [4]. За прогнозами аналітиків, у 2024 році загальний обсяг ринку може перевищити 7 млрд дол. США [4]. Основними факторами зростання є підвищення довіри споживачів до онлайн-покупок, розвиток цифрової та платіжної інфраструктури, а також удосконалення логістичних сервісів. Незважаючи на повномасштабну війну з російською федерацією, цифровий сектор економіки України демонструє відносну стійкість, а електронна комерція відіграє важливу роль у підтримці споживчого попиту та економічного відновлення.

Розглянемо основні тренди розвитку українського ринку e-commerce останніх років [4].

Зростання мобільної комерції (m-commerce). Мобільна комерція є одним із ключових драйверів розвитку українського ринку електронної комерції. За підсумками 2023 року понад 60% онлайн-покупок в Україні було здійснено з використанням мобільних пристроїв. Очікується, що у 2024 році цей показник продовжить зростати завдяки розширенню доступу до мобільного Інтернету та поширенню доступних смартфонів. Згідно з даними Statista, рівень проникнення мобільного Інтернету в Україні збільшиться з 71% у 2023 році до приблизно 75% у 2024 році (див. рис. 1.1) [4]. У відповідь на ці тенденції ритейлери активно інвестують у розвиток мобільних застосунків та оптимізацію вебплатформ для мобільних користувачів.

Розвиток соціальної комерції. Соціальні мережі поступово трансформуються в повноцінні канали онлайн-продажів. Платформи Instagram, Facebook і TikTok широко використовуються українськими компаніями для прямої взаємодії з цільовою аудиторією, особливо серед молодих споживачів. У 2024 році очікується подальше зростання ролі соціальної комерції, зокрема за рахунок розвитку інфлюенс-маркетингу та

інтерактивних форматів продажів. Українські підприємці дедалі активніше впроваджують механізми продажу через прямі трансляції (live commerce), що вже довели свою ефективність на азійських ринках, особливо в сегментах моди, косметики та товарів для дому.

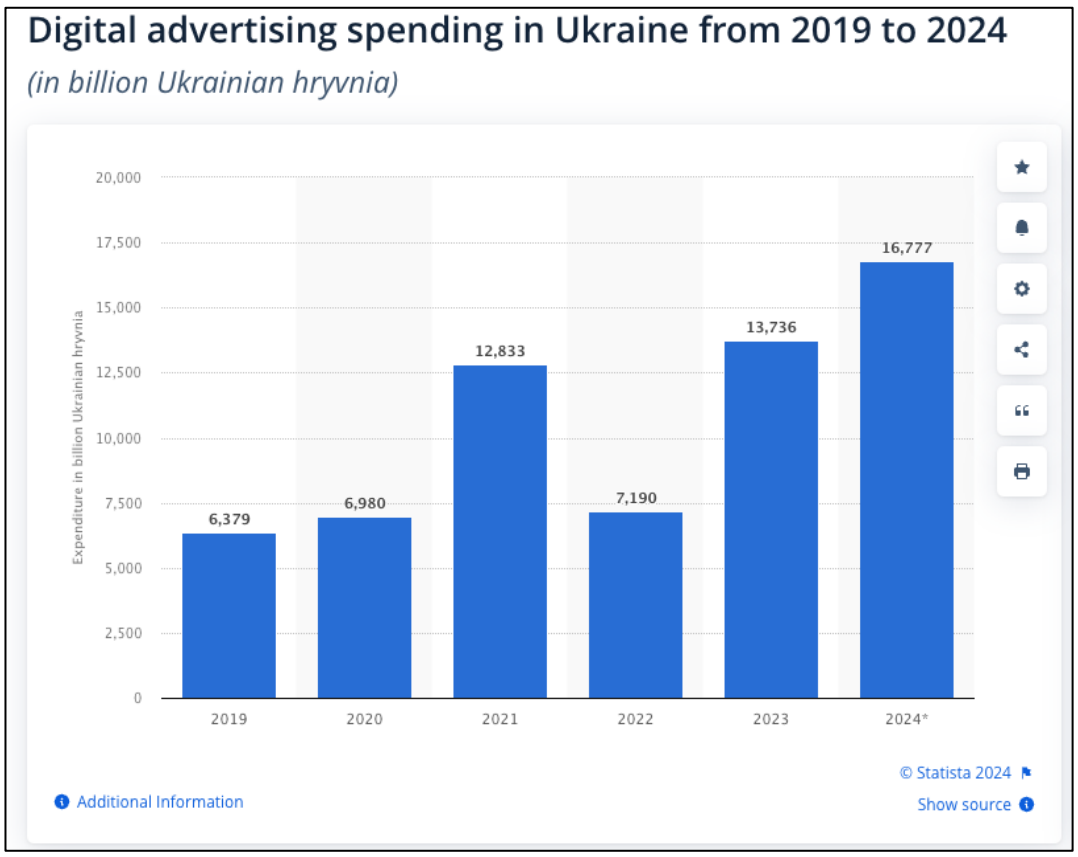


Рисунок 1.1 – Проникнення мобільного інтернету в Україні [4]

Розширення платіжних рішень та інтеграція з фінтехом. Розвиток цифрових платіжних інструментів є важливим чинником зростання електронної комерції. У 2023 році понад 80% онлайн-транзакцій в Україні здійснювалися безготівковими способами, включаючи платіжні картки, мобільні банківські застосунки та електронні гаманці (Apple Pay, Google Pay) [4]. Провідні фінтех-платформи, зокрема Privat24 і Monobank, забезпечують швидкі та зручні платіжні рішення як для фізичних осіб, так і для бізнесу. Значного поширення набувають моделі «Buy Now, Pay Later» (BNPL), які стимулюють попит у сегментах електроніки, побутової техніки та fashion-товарів.

Удосконалення логістичних і доставкових рішень. Логістика традиційно залишається одним із найбільш проблемних елементів українського e-commerce, особливо в умовах воєнних дій. Водночас у 2024 році спостерігається активна оптимізація логістичних процесів. Компанії «Нова Пошта» та «Укрпошта» суттєво розширили мережу відділень і пунктів видачі, у тому числі в регіонах, що зазнали негативного впливу війни. Популярності набувають сервіси доставки в день замовлення, а також автоматизовані пункти видачі, що відповідає зростаючому попиту споживачів на швидкість і зручність.

ТОП продуктів, які користуються попитом в 2024 в українському e-Commerce, представлено на рисунку 1.2 [4].



Рисунок 1.2 – ТОП-10 продуктів, які користуються попитом в 2024 в українському eCommerce [4]

Військові дії та регулярні повітряні тривоги суттєво впливають на споживчу поведінку. У містах, розташованих поблизу зон бойових дій, спостерігається зростання частки онлайн-покупок, що пов'язано з обмеженим

доступом до традиційних торговельних центрів та підвищеними ризиками офлайн-шопінгу.

Попри позитивну динаміку, український ринок електронної комерції у 2024 році стикається з низкою суттєвих проблем. Серед них – питання кібербезпеки та захисту персональних даних, вплив воєнних факторів на логістику й енергопостачання, регуляторна невизначеність та зростаюча конкуренція. Частка маркетплейсів у структурі e-commerce вже перевищує 60% і продовжує зростати, що ускладнює вихід на ринок для малих і середніх онлайн-проєктів. Крупні бренди створюють власні маркетплейси (див. рис. 1.3) [4], що створює конкуренцію з іншими eCommerce проєктами та забирає в них частину ринку.

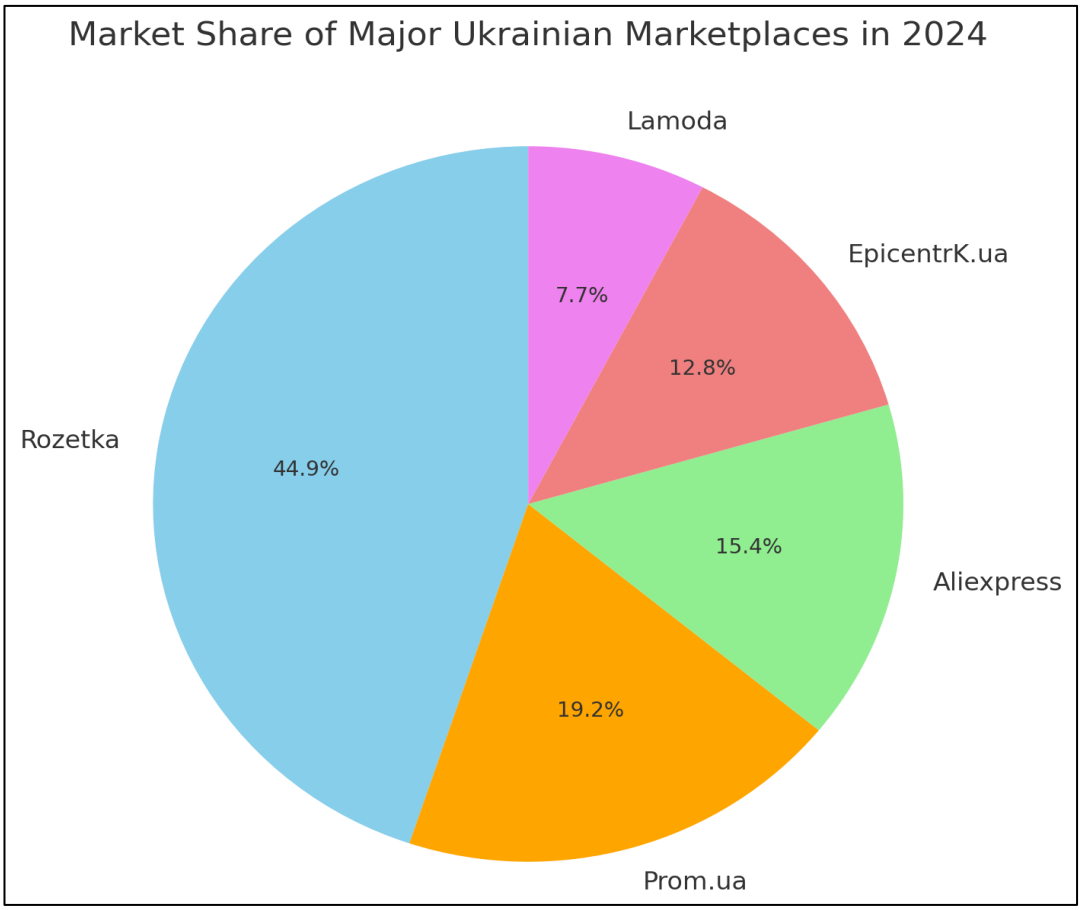


Рисунок 1.3 – Основні українські маркетплейси [4]

Додатковим фактором трансформації ринку може стати вихід Monobank із власним маркетплейсом, інтегрованим у фінансову екосистему банку.

Очікується, що завдяки доступу до понад 7 млн активних користувачів та використанню фінансових інструментів (BNPL, кешбеки, програми лояльності) новий гравець здатен зайняти до 10-12% ринку протягом перших років функціонування [4]. Це посилить конкурентний тиск на таких лідерів, як Rozetka та Prom.ua, і стимулюватиме подальші інновації у сфері сервісу та фінансових технологій.

Отже, український ринок електронної комерції характеризується високими темпами зростання та значним потенціалом подальшого розвитку. Прогнозований середньорічний темп зростання на рівні 15-20% до 2025 року свідчить про перспективність галузі [4]. Водночас не всі компанії здатні ефективно монетизувати зростання ринку, що обумовлює необхідність стратегічного управління, інноваційних бізнес-моделей та адаптації до зовнішніх викликів. У довгостроковій перспективі електронна комерція залишатиметься одним із ключових чинників цифрової трансформації та економічного відновлення України.

Вихід на ринок електронної комерції нового великого гравця у вигляді маркетплейсу Monobank потенційно здатен суттєво змінити конкурентну структуру галузі та спричинити перерозподіл ринкових часток між існуючими учасниками. З урахуванням масштабної клієнтської бази банку, високого рівня довіри до бренду та наявності розвиненої фінансово-цифрової екосистеми, очікується, що до кінця 2025 року новий маркетплейс може акумулювати близько 10-12% загального обсягу ринку електронної комерції [4]. Важливою конкурентною перевагою Monobank є інтеграція фінансових інструментів безпосередньо у процес купівлі, що дозволяє зменшити транзакційні витрати споживачів і підвищити рівень конверсії.

Активізація фінансово-орієнтованих маркетплейсів стимулюватиме інших гравців ринку до впровадження додаткових фінансових сервісів, удосконалення платіжної інфраструктури та підвищення якості клієнтського обслуговування. У результаті український ринок електронної комерції у 2025 році характеризуватиметься зростанням рівня конкуренції, підвищенням

технологічної насиченості та переходом від цінової конкуренції до конкуренції бізнес-моделей і сервісних рішень. Маркетплейси та онлайн-платформи, що зможуть інтегрувати фінансові, логістичні та аналітичні сервіси в єдину цифрову екосистему, матимуть суттєві переваги у боротьбі за споживача.

У середньостроковій перспективі український ринок електронної комерції зберігатиме позитивну динаміку розвитку з прогнозованим середньорічним темпом зростання (CAGR) на рівні 15-20% до 2025 року [4]. Основними драйверами зростання залишатимуться мобільна комерція, розвиток платіжних технологій, розширення логістичної інфраструктури та цифровізація споживчої поведінки. Водночас суб'єкти ринку змушені адаптуватися до зовнішніх викликів, зокрема загроз кібербезпеці, регуляторної невизначеності та наслідків воєнних дій, що вимагає підвищеної гнучкості управлінських рішень.

У міру поглиблення цифрової трансформації економіки електронна комерція продовжуватиме відігравати системоутворюючу роль у процесах економічного відновлення та розвитку підприємництва. Разом із тим зростання ринку не гарантує автоматичного отримання прибутку всіма його учасниками. Практика показує, що значна частина підприємців започатковує інтернет-магазини без чіткого розуміння принципів функціонування цифрової бізнес-моделі та її взаємозв'язку з основними операційними процесами компанії.

Електронна комерція не зводиться до технічного запуску вебсайту або створення сторінки в соціальних мережах. Вона передбачає перехід бізнесу до нової операційної логіки, у межах якої процеси обліку, продажів, маркетингових комунікацій і управління клієнтською базою функціонують як єдина інтегрована система. За таких умов, можна виділити ключові характеристики сучасної електронної комерції [4-5].

По-перше, діджитал-активність компанії слід сприймати не як статичну даність, а як динамічну систему безперервного менеджменту клієнтського

досвіду. Критичним є не сам факт наявності онлайн-каналу, а швидкість і якість обробки запитів, зручність платіжних рішень, ефективність контролю доставки та формування повторних продажів. Часткова цифровізація окремих функцій без інтеграції з іншими процесами призводить до зростання операційних витрат і зниження ефективності.

По-друге, визначальну роль у прийнятті управлінських рішень відіграє аналітика. У 2025 році інтуїтивні підходи поступово витісняються даними, що генеруються аналітичними системами та CRM-платформами. Інструменти вебаналітики дозволяють ідентифікувати поведінкові патерни споживачів, оцінювати ефективність маркетингових каналів і виявляти точки втрати прибутку, що формує основу для обґрунтованого масштабування бізнесу.

По-третє, вартість входу на ринок електронної комерції суттєво знизився, однак низький фінансовий поріг не компенсує відсутність системності. Успіх онлайн-проєкту визначається не вибором платформи, а рівнем інтеграції ключових процесів: автоматизацією платежів, прозорістю логістики та структурованістю клієнтської бази. Крах більшості онлайн-проєктів спричинений не низькою зацікавленістю аудиторії, а хаосом у внутрішніх бізнес-процесах та відсутністю системного підходу до розвитку інтернет-платформи

По-четверте, малий та середній бізнес у сфері електронної комерції часто володіє конкурентними перевагами порівняно з великими компаніями завдяки більшій гнучкості управління, швидкості прийняття рішень та можливості персоналізованої комунікації з клієнтами. У регіональних сегментах ринку саме ці чинники часто визначають рівень лояльності споживачів.

По-п’яте, ключовим фактором успіху залишається рівень довіри та безпеки клієнта. Прозорість бізнесу, чітка політика повернення товарів, достовірна інформація про компанію та захист персональних даних формують основу споживчої довіри, яка у багатьох випадках має вищу значущість, ніж інструменти цінової конкуренції.

З огляду на це, перехід до електронної комерції слід розглядати не як окреме маркетингове рішення, а як комплексну організаційну трансформацію бізнесу. Підприємства, що сприймають онлайн-канал як інструмент системного управління, аналітики та роботи з клієнтами, здатні закріпитися на ринку та забезпечити довгострокову конкурентоспроможність. Натомість компанії, які обмежуються використанням електронної комерції лише як додаткового каналу збуту, ризикують не досягти сталих фінансових результатів.

1.6.2 E-commerce платформи: огляд найпопулярніших рішень

Оскільки цифровізація економіки набирає обертів, електронна комерція стала фундаментом бізнесу, адже сучасним споживачам значно швидше та зручніше оформлювати замовлення в онлайн-форматі. Функціонування інтернет-магазину вже не обмежується наявністю онлайн-вітрини товарів, а передбачає використання комплексних цифрових рішень для управління продажами, клієнтськими відносинами, логістикою та аналітикою. За таких умов вибір відповідної e-commerce платформи набуває стратегічного значення, оскільки саме вона визначає технічні можливості масштабування бізнесу, рівень автоматизації процесів та якість взаємодії зі споживачами.

На сучасному ринку представлені різні програмні рішення для створення та управління інтернет-магазинами, що відрізняються функціональними можливостями, моделями ліцензування та рівнем технічної підтримки. Аналіз найбільш поширених платформ дозволяє виокремити їх ключові характеристики, переваги та обмеження, що узагальнено у таблиці 1.1 [6].

При цьому вибір конкретного рішення має здійснюватися з урахуванням специфіки бізнес-моделі, масштабів діяльності та фінансових можливостей підприємства.

Таблиця 1.1 – Найпопулярніші e-commerce платформи [6]

№	E-commerce платформа	Переваги	Недоліки
1.	Shopify	– інтуїтивно зрозумілий інтерфейс; – широкий вибір шаблонів і додатків; – вбудовані інструменти маркетингу та аналітики; – високий рівень безпеки.	– високі комісії за транзакції; – обмеження налаштувань; – обмежений вибір шлюзів оплати.
2.	WooCommerce	– безкоштовне використання; – гнучкість у налаштуванні; – широкий вибір плагінів і розширень; – велика спільнота підтримки.	– вимагає технічних навичок для налаштування; – необхідність встановлювати окремо плагіни для розширеного функціоналу; – обмежений вбудований функціонал.
3.	Magento	– налаштування та гнучкість; – розширені можливості управління каталогом; – багатоканальні функції продажу; – висока масштабованість.	– складна у використанні для початківців; – вимагає значних технічних ресурсів; – висока вартість використання.
4.	PrestaShop	– безкоштовне використання; – широкий вибір модулів і шаблонів; – мультимовна підтримка; – велика спільнота підтримки.	– вимагає технічних навичок для налаштування; – необхідність встановлювати окремо модулі для розширеного функціоналу; – обмежене SEO-оптимізація.
5.	BigCommerce	– простота використання; – розширений функціонал; – вбудовані маркетингові інструменти; – висока масштабованість.	– вищі тарифи, ніж у деяких конкурентів; – більш обмежені можливості налаштування; – обмежена підтримка сторонніх розширень.

Серед додаткових аспектів, які потребують поглибленого аналізу при виборі e-commerce платформи, доцільно виділити ще деякі [6], розглянемо їх детальніше. По-перше, *ціноутворення*. Вартість використання платформ електронної комерції може варіюватися від безкоштовних або умовно

безкоштовних рішень з базовим функціоналом до професійних систем, що передбачають щомісячну оплату в розмірі сотень або навіть тисяч доларів. Ціна залежить від рівня автоматизації, кількості інтеграцій, можливостей кастомізації та наявності технічної підтримки.

По-друге, *функціональність платформи*. Ключовими є можливості управління товарним каталогом, інтеграція з платіжними шлюзами, системами доставки, CRM-рішеннями та інструментами аналітики. Платформа повинна забезпечувати стабільну роботу, обробку замовлень у реальному часі та масштабування без втрати продуктивності.

По-третє, *можливості налаштування та кастомізації*. Важливим критерієм вибору є здатність платформи адаптувати дизайн, структуру та функціонал інтернет-магазину відповідно до бренд-стратегії підприємства та потреб цільової аудиторії. Обмежені можливості налаштування можуть стримувати розвиток бізнесу на подальших етапах.

По-четверте, *рівень технічної підтримки*. Наявність оперативної та професійної підтримки (через онлайн-чат, електронну пошту або телефон) є критичною для безперервного функціонування інтернет-магазину, особливо на етапах запуску та масштабування.

Таким чином, вибір e-commerce платформи є одним із ключових управлінських рішень, що безпосередньо впливає на ефективність функціонування інтернет-магазину. Системна оцінка бізнес-потреб, технічних можливостей та фінансових обмежень дозволяє прийняти обґрунтоване рішення і сформувати надійну технологічну основу для досягнення стратегічних цілей у сфері електронної комерції.

У зазначених умовах особливої актуальності набуває пошук і впровадження технічних рішень для розробки мобільних додатків у сфері e-commerce, оскільки мобільна комерція стає домінуючим каналом взаємодії зі споживачами. Розробка мобільного додатка для електронної торгівлі передбачає комплексний підхід, що охоплює кілька взаємопов’язаних етапів [7].

На етапі планування та визначення обсягу робіт здійснюється ідентифікація цільової аудиторії з урахуванням її демографічних характеристик, потреб і поведінкових особливостей. Паралельно формулюються цілі мобільного додатка, зокрема збільшення обсягів продажів, підвищення рівня лояльності клієнтів або оптимізація сервісного обслуговування. Важливим є також проведення конкурентного аналізу та розробка карти користувацького досвіду (UX), що відображає ключові точки взаємодії користувача з додатком.

Етап розробки передбачає вибір платформи (iOS, Android або кросплатформені рішення), створення інтуїтивно зрозумілого інтерфейсу та інтеграцію мобільного додатка з базовою e-commerce платформою для забезпечення синхронізації даних. Функціональне наповнення додатка має включати пошук товарів, оформлення замовлень, платіжні модулі та використання специфічних можливостей мобільних пристроїв, таких як push-сповіщення, геолокація та персоналізовані пропозиції.

На етапі тестування та розгортання проводиться оцінка зручності використання, перевірка функціональної стабільності та сумісності додатка з різними пристроями. Після запуску важливим є постійний моніторинг роботи системи, збір зворотного зв’язку від користувачів і поступове вдосконалення функціоналу.

Мобільні додатки для електронної комерції забезпечують низку суттєвих переваг, серед яких персоналізація покупок, покращення користувацького досвіду, підвищення коефіцієнтів конверсії та зміцнення лояльності клієнтів через програми заохочення. Крім того, вони дозволяють накопичувати та аналізувати дані про поведінку споживачів, що є основою для прийняття обґрунтованих управлінських рішень.

Отже, використання системного та поетапного підходу до вибору e-commerce платформи та розробки мобільних додатків є передумовою підвищення ефективності електронної комерції, зростання обсягів продажів і формування стійкої конкурентної позиції підприємства в цифровому середовищі.

1.6.3 Ознаки успішного інтернет-магазину

Основні характеристики та функціональні завдання інтернет-магазину й традиційного магазину за своєю сутністю є подібними, оскільки обидві форми торгівлі спрямовані на задоволення споживчого попиту та отримання прибутку. Водночас інтернет-магазин має низку специфічних ознак, які принципово відрізняють його від класичної торговельної моделі та формують його конкурентні переваги в умовах цифрової економіки.

До ключових характеристик інтернет-магазинів належать:

– можливість представлення практично необмеженого асортименту товарів незалежно від фізичних обмежень торговельної площі;
– оперативне та масштабне інформування користувачів про товари, акції та новинки;
– персоналізація взаємодії з клієнтами на основі аналізу їх поведінки та історії покупок;
– широкі можливості проведення маркетингових досліджень без прямого контакту зі споживачем.

Ефективність функціонування інтернет-магазину значною мірою залежить від якості його вебсайту, зручності користування та відповідності сучасним вимогам електронної комерції. У наукових і прикладних дослідженнях виокремлюють низку ознак, притаманних успішним та конкурентоспроможним інтернет-магазинам.

Висока швидкість завантаження вебресурсу. Швидкість завантаження сторінок є критичним фактором користувацького досвіду. Повільна робота сайту негативно впливає на поведінку відвідувачів і призводить до зростання показника відмов. Дослідження свідчать, що значна частка користувачів залишає сайт, якщо час його завантаження перевищує кілька секунд. Крім того, швидкість сторінки є одним із факторів ранжування в пошукових системах, зокрема для мобільного пошуку. Забезпечення швидкої роботи сайту досягається за рахунок оптимізації програмного коду, зображень та використання адаптивного дизайну.

Адаптивний вебдизайн. Адаптивний дизайн забезпечує коректне відображення та зручну взаємодію користувача з сайтом незалежно від типу пристрою. В умовах зростання частки мобільного трафіку мобільна версія сайту набула пріоритетного значення для пошукових систем. Це зумовлює необхідність створення інтернет-магазинів, орієнтованих на комфортне користування зі смартфонів і планшетів.

Конверсійно-орієнтований дизайн. Дизайн інтернет-магазину має не лише естетичну, а й функціональну складову, оскільки безпосередньо впливає на рішення користувача щодо здійснення покупки. Ефективний дизайн передбачає логічну структуру, чітку навігацію, зрозумілу категоризацію товарів та акцентування уваги на ключових елементах процесу купівлі.

Важливим елементом конверсійного дизайну є інформаційна сторінка «Про компанію», яка формує довіру до бренду. Вона має містити стислий і чіткий опис діяльності магазину, його конкурентні переваги, досягнення, а також контактну інформацію та соціальні докази якості (відгуки, партнерства, нагороди).

SEO-оптимізація вебресурсу. Пошукова оптимізація є ключовим інструментом залучення органічного трафіку. Технічна оптимізація сайту, коректне використання ключових слів у назвах і описах товарів, а також структурованість контенту сприяють підвищенню позицій інтернет-магазину в пошуковій видачі та зростанню його відвідуваності.

Використання якісного візуального контенту. Візуальна інформація відіграє вирішальну роль у формуванні першого враження про товар. Якісні фотографії з високою роздільною здатністю, можливість перегляду товару з різних ракурсів, функції масштабування та 3D-огляду значно підвищують ймовірність здійснення покупки.

Різноманітність і зручність платіжних інструментів. Надання користувачам можливості обирати зручний спосіб оплати є важливим чинником підвищення конверсії. Інтеграція різних платіжних систем та збереження платіжних даних для повторних покупок сприяють формуванню позитивного користувацького досвіду.

Наявність експертного контенту та блогу. Блог інтернет-магазину виконує функцію інструменту контент-маркетингу та підвищення авторитету ресурсу. Корисні, аналітичні та SEO-оптимізовані матеріали дозволяють залучати додатковий трафік, формувати довіру до бренду та краще розуміти потреби цільової аудиторії.

Формування довіри та лояльності клієнтів. Для нових інтернет-магазинів особливо важливим є вибудовування довгострокових відносин із клієнтами. Цьому сприяють якісна онлайн-підтримка, активна присутність у соціальних мережах, партнерство з відомими брендами та використання відгуків реальних покупців як інструменту соціального доказу.

Використання інструментів вебаналітики. Аналітика є необхідною умовою ефективного управління інтернет-магазином. Застосування спеціалізованих сервісів, зокрема Google Analytics, дозволяє відстежувати ключові показники ефективності, аналізувати поведінку користувачів, рівень конверсії та виявляти проблемні місця в процесі продажу [8].

2 ПРОЄКТУВАННЯ СЕРВІСУ ПРЕЗЕНТАЦІЇ ТОВАРІВ

2.1 Обґрунтування вибору інструментів для розробки

Вибір технологічного стеку для реалізації сучасної платформи електронної комерції є надзвичайно відповідальним етапом, оскільки саме від нього залежить простота і швидкість розробки, масштабованість, безпечність та витрати на підтримку проєкту у майбутньому. Нижче наведено обґрунтування вибору основних інструментів для реалізації даної роботи.

2.1.1 Опис стеку технологій для Backend

Стек технологій для бекенду включає: FastAPI + PostgreSQL + SQLAlchemy + JWT-аутентифікацію.

FastAPI – високопродуктивний вебфреймворк для Python, який використовується для створення REST API-сервісів [9].

Головні переваги (див. рис. 2.1) [10]:

– висока продуктивність: FastAPI побудований поверх ASGI (Asynchronous Server Gateway Interface), використовує Starlette та Pydantic, швидкість роботи наближена до Node.js та Go;
– автоматична генерація документації: Swagger UI та ReDoc створюються автоматично на основі типів Python – це спрощує тестування API й інтеграцію з фронтендом;
– типізація та валідація: всі дані перевіряються через Pydantic, що підвищує стабільність системи та мінімізує помилки.

FastAPI використовують в проєкті для:

– створення REST-ендпоінтів для каталогу товарів, користувачів та адмін-панелі;

– реалізації аутентифікації та авторизації (через JWT);
– обробки запитів клієнтів від Vue.js фронтенду;
– асинхронної роботи з базою даних [10].

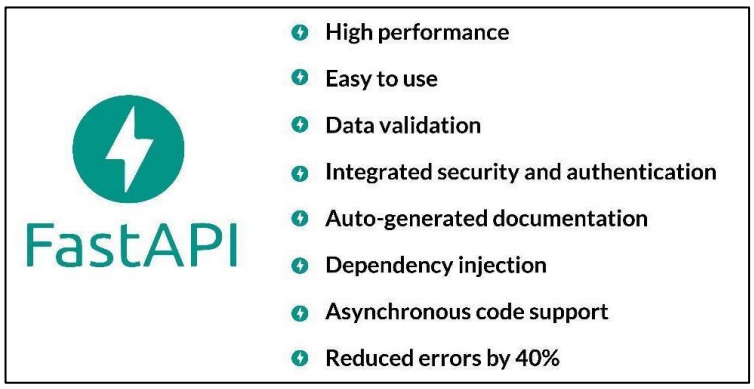


Рисунок 2.1 – Головні переваги FastAPI [10]

PostgreSQL – основна реляційна база даних. PostgreSQL – потужна, надійна й масштабована SQL-база, що оптимально підходить для складних бізнес-систем.

Переваги PostgreSQL (див. рис. 2.2) [11]:

– надійність і транзакційність: підтримка ACID гарантує цілісність даних;
– потужний SQL і розширення: повнотекстовий пошук (FTS), JSONB-тип для комбінованих структур даних, індекси (GIN, BTREE);
– масштабованість: підходить для роботи з великими каталогами товарів [11].

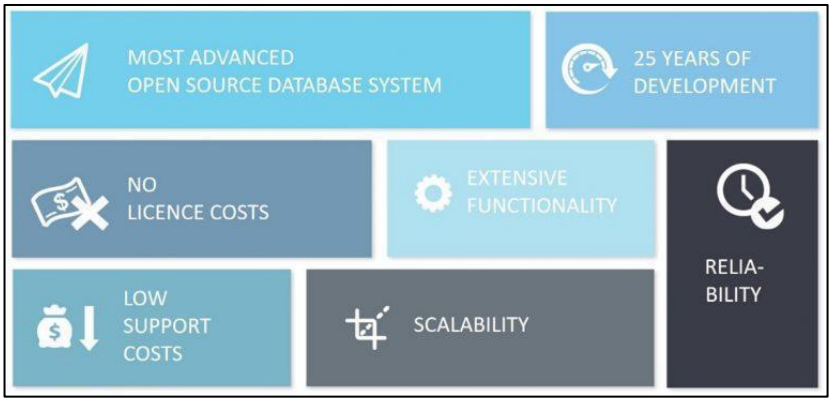


Рисунок 2.2 – Переваги PostgreSQL [11]

В розроблюваному проєкті PostgreSQL використовується для:

– зберігання даних про товари, категорії, користувачів;
– збереження прав доступу;
– фіксації логів операцій;
– виконання складних запитів для пошуку й фільтрації.

SQLAlchemy – ORM та запити до БД. SQLAlchemy – найпопулярніший ORM у світі Python. У поєднанні з FastAPI найчастіше застосовують SQLAlchemy 2.0 з асинхронним драйвером (aiopg/asyncpg).

Переваги:

– ORM + core SQL: можна працювати як на рівні ORM-моделей (класи Python), так і формувати складні SQL-запити на «core» SQLAlchemy;
– міграції (через Alembic): легко змінювати структуру таблиць, виконувати оновлення без втрати даних;
– інтеграція з FastAPI: зручно створювати залежності (dependency injection) для сесій БД.

ORM SQLAlchemy (див. рис. 2.3) використовується для абстрагування роботи з SQL-запитами і створення моделей даних у вигляді об’єктів Python [12].

SQLAlchemy ORM (Object-Relational Mapping) – це частина бібліотеки SQLAlchemy, що дозволяє взаємодіяти з базами даних, використовуючи об’єктно-орієнтований підхід, замість написання SQL-запитів безпосередньо. Він перетворює об’єкти Python на записи в базі даних та навпаки, дозволяючи працювати з даними як з об’єктами мови Python.

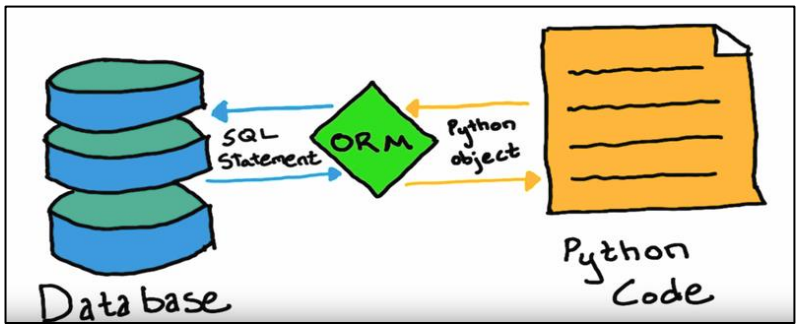


Рисунок 2.3 – ORM SQLAlchemy [12]

Детальніше SQLAlchemy ORM можна охарактеризувати таким чином:

– ORM як прошарок: SQLAlchemy ORM виступає як прошарок між вашим Python-кодом і базою даних, абстрагуючи SQL-запити;
– об’єктно-орієнтований підхід: ви визначаєте схему бази даних, використовуючи класи Python, і SQLAlchemy ORM автоматично створює таблиці та забезпечує взаємодію з ними через об’єкти, які ви створюєте;
– спрощення роботи з базами даних: замість написання складних SQL-запитів, ви працюєте з об’єктами, що значно спрощує розробку та підтримку коду, особливо для складних проєктів;
– підтримка різних діалектів SQL: SQLAlchemy підтримує багато популярних баз даних, таких як MySQL, PostgreSQL, SQLite тощо, дозволяючи перемикатися між ними без зміни коду;
– Data Mapper Pattern: SQLAlchemy використовує «Data Mapper» паттерн, що відрізняється від «Active Record» паттерна, що використовується в деяких інших ORM, і забезпечує більш гнучку взаємодію з базою даних;
– SQLAlchemy Core: ORM побудований поверх SQLAlchemy Core, який забезпечує низькорівневий доступ до бази даних.

У підсумку, SQLAlchemy ORM дозволяє працювати з базами даних більш зручним та ефективним способом, використовуючи об’єктно-орієнтований підхід, що спрощує розробку та підтримку програмного забезпечення [12].

Використання ORM у проєкті, що розробляється, необхідно для:

– формування моделей товарів, користувачів, категорій;
– забезпечення безпечного доступу до БД;
– уникнення повтору SQL-коду;
– прозорого запису/оновлення/видалення сутностей.

JWT – аутентифікація та авторизація. JWT (JSON Web Token) – стандарт токенів для захищеної авторизації в вебдодатках (див. рис. 2.4) [13].

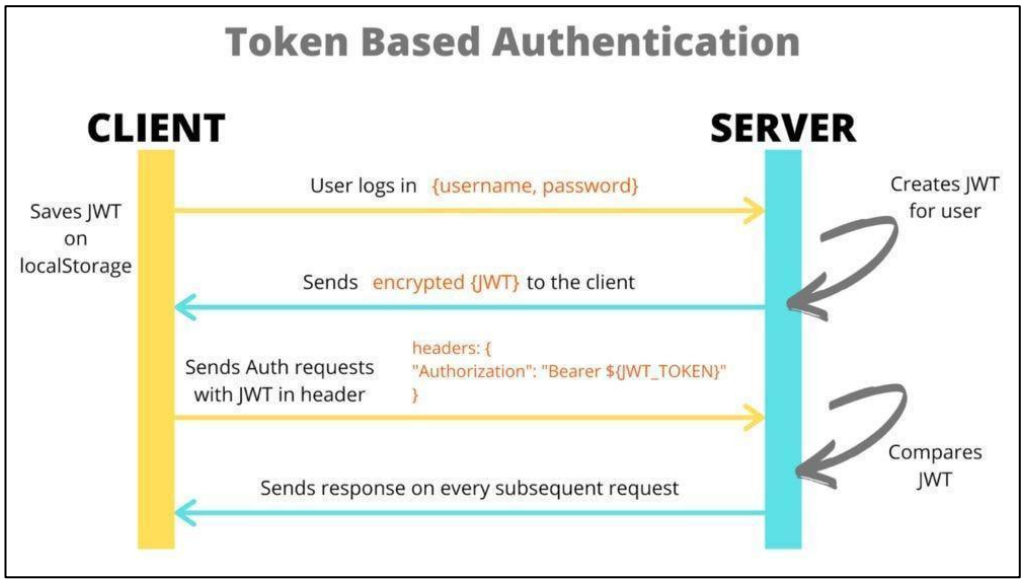


Рисунок 2.4 – Логіка JWT (JSON Web Token) [13]

Переваги JWT:

– безсерверність: токен перевіряється без збереження стану на сервері – значить масштабування спрощується;
– швидка перевірка: токен підписано секретом або RSA-ключем – FastAPI перевіряє його за мілісекунди.

В JWT містяться: user_id, роль (адмін/користувач), час життя токена, додаткові claims.

Використання JWT у платформі необхідне для:

– авторизації адміністратора, який керує товарами;
– захисту закритих API-методів (додавання/видалення товарів, редагування даних);
– авторизації користувача для розширених можливостей сайту (вибране, історія, адмін-панель) [13].

Розглянемо, як працює весь стек разом:

1) користувач (Front – Vue.js) → FastAPI (Backend): Vue.js надсилає запит до API (REST); FastAPI приймає запит, валідує, перевіряє токен (якщо потрібна авторизація);

2) FastAPI → SQLAlchemy → PostgreSQL: FastAPI отримує об’єкт сесії БД, SQLAlchemy створює запит, PostgreSQL повертає результат;

3) FastAPI → Vue.js: відповідає JSON-даними у форматі REST;

4) JWT: зберігається у браузері (localStorage або cookie, краще – HttpOnly cookie), додається в заголовок Authorization при кожному запиті, сервер перевіряє підпис і приймає/відхиляє запит.

Перевагами саме цього стеку, який був обраний, є:

– висока продуктивність (FastAPI + async DB);
– надійність і консистентність даних (PostgreSQL);
– гнучкість моделювання (SQLAlchemy ORM);
– безпечна авторизація (JWT);
– масштабованість (можна розширити до великих e-commerce проєктів);
– чітка структура API → легко інтегрується з Vue.js.

2.1.2 Опис стеку технологій для Frontend

Стек технологій фронтенду для проєкту включає: Vue.js 3 + TailwindCSS + Pinia + VeeValidate.

Vue.js 3 – сучасний JavaScript-фреймворк для побудови SPA. Vue.js 3 – прогресивний фреймворк для створення високопродуктивних інтерфейсів. У даному проєкті він використовується як основа клієнтської частини (див. рис. 2.5) [14].

Переваги Vue.js 3:

– композиційний API: дозволяє більш ефективно організувати логіку, повторно використовувати функції та спрощує масштабування фронтенду;
– реактивність нового покоління: система реактивності вбудована на Proxies – працює швидше та передбачуваніше, ніж у Vue 2;
– компонентна архітектура: увесь інтерфейс складається з невеликих компонентів – це полегшує розробку, тестування й підтримку;

– висока продуктивність: оптимізація рендерингу та оновлення DOM дозволяє працювати навіть із великими каталогами товарів.

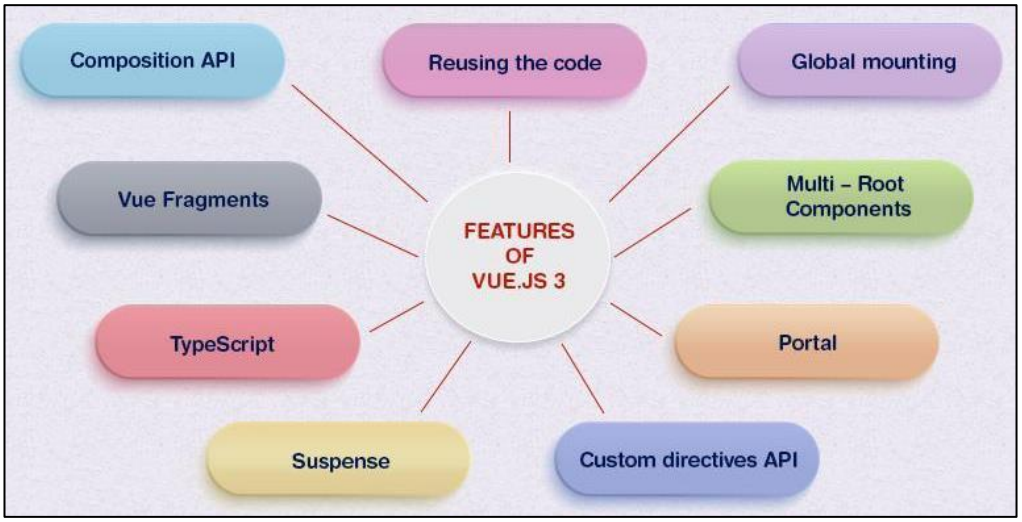


Рисунок 2.5 – Характеристика Vue.js 3 [14]

Використання Vue.js 3 у проєкті є обов'язковим, обумовлено його перевагами і передбачає: реалізацію SPA для презентації товарів; створення сторінок каталогу, товарів, фільтрів, пошуку; побудову модальних вікон, каруселей, галереї тощо.

TailwindCSS – утилітарний CSS-фреймворк. TailwindCSS – це сучасний CSS-фреймворк, який базується на утилітарних класах (див. рис. 2.6) [15].

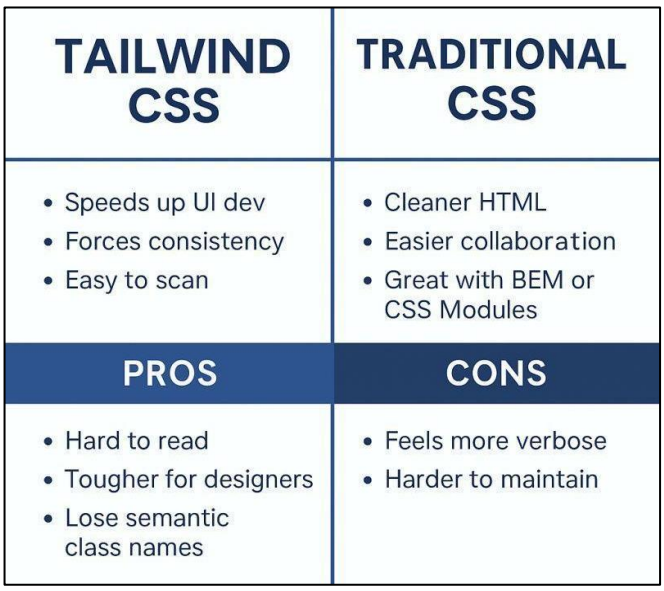


Рисунок 2.6 – Переваги TailwindCSS порівняно з традиційною CSS [15]

Переваги TailwindCSS:

– швидка розробка: замість написання стилів вручну – використання готових CSS-класів (p-4, flex, grid, rounded-xl);
– гнучкість дизайну: не нав'язує стилі (як Bootstrap) – кожен елемент можна оформити індивідуально;
– адаптивність: вбудована система breakpoints для створення mobile-first інтерфейсу;
– оптимізація: Tailwind очищає CSS на етапі білду: залишаються тільки ті класи, які реально використані на сторінці.

CSS-фреймворк використовується у проєкті для:

– побудови адаптивної сітки каталогу;
– стилізації карток товарів, форм, фільтрів;
– створення сучасного, мінімалістичного UI без ручного CSS [16].

Pinia – менеджер стану для Vue.js 3. Pinia – це офіційна бібліотека для керування глобальним станом у Vue.js 3 (наступник Vuex).

Переваги Pinia (див. рис. 2.7) [17]:

– простота та зрозумілість API: код читається легко, не потребує зайвих обгорток чи мутацій;
– реактивність і підтримка TypeScript: підтримує сильну типізацію, що знижує кількість помилок;
– глобальні сторінки стану: легко керувати авторизацією, кошиком (якщо додати e-commerce), фільтрами каталогу, списком товарів, користувацькими налаштуваннями [17].

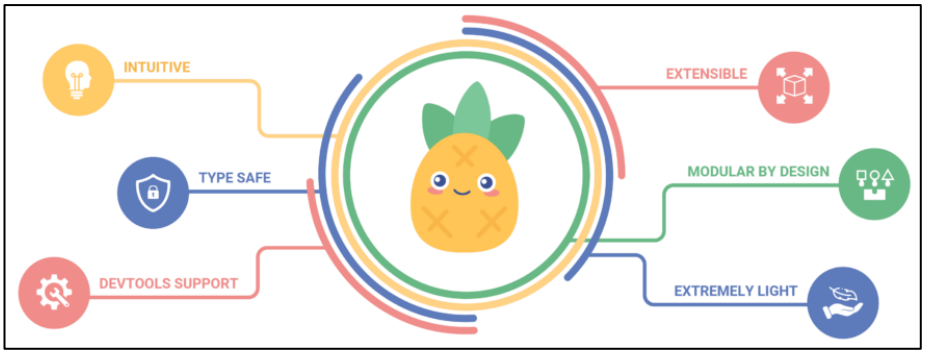


Рисунок 2.7 – Переваги Pinia [17]

Pinia використовується у проєкті для:

– збереження обраних товарів;
– збереження даних фільтрів і пошуку;
– керування станом авторизованого користувача;
– кешування каталогу для швидкого повторного відображення.

VeeValidate – валідація форм. VeeValidate – популярна бібліотека для валідації форм у Vue (див. рис. 2.8) [18].

Переваги:

– гнучка схема валідації: підтримує регулярні вирази, yup-схеми, кастомні правила;
– реактивність: статус полів оновлюється миттєво: valid / invalid / dirty;
– простота інтеграції: форми легко створювати через <Form>, <Field>, <ErrorMessage>.

VeeValidate використовується в проєкті для:

– валідації форми запиту/зворотного зв’язку;
– валідації форми авторизації/реєстрації;
– контролю правильності введених даних при фільтрації товарів;
– запобігання некоректним запитам до сервера.

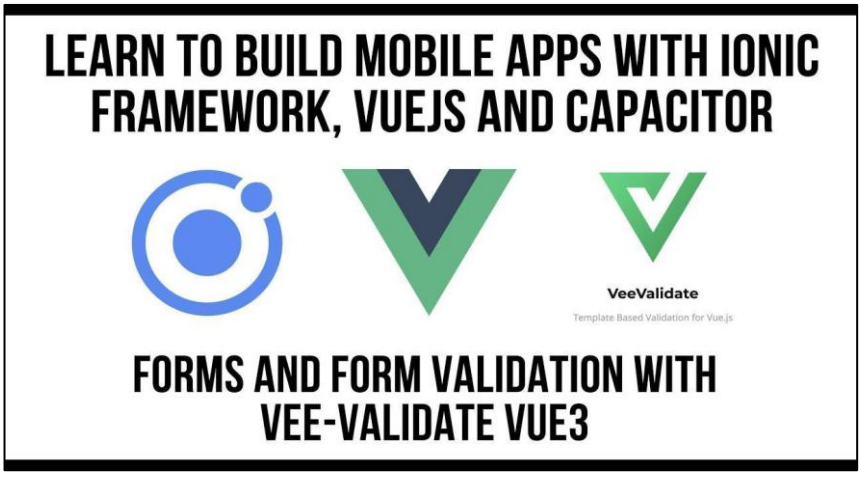


Рисунок 2.8 – VeeValidate [18]

Механізм роботи фронтенд-стеку разом можна представити таким чином:

1) Vue.js 3 відповідає за структуру та логіку компонентів (каталог, картка товару, пошук);

2) TailwindCSS відповідає за зовнішній вигляд компонентів, створюючи стильний та адаптивний інтерфейс;

3) Pinia керує глобальним станом (авторизація, фільтри, кошик, категорії);

4) VeeValidate відповідає за правильність введених користувачем даних у формах.

Отже, обрана комбінація технологій дозволяє досягти: високу продуктивність; чистий і сучасний UI; простий у підтримці код; надійність і структурованість логіки; повністю адаптивний інтерфейс; готовність до масштабування (додавання e-commerce функцій).

2.1.3 Архітектура проєкту

Архітектура на основі REST API та SPA (Single Page Application) є сучасним і широко використовуваним підходом у веброзробці, що забезпечує швидку взаємодію, масштабованість і простоту подальшого розвитку системи. Вона складається з двох логічних частин: Backend, який надає REST API, та Frontend, який функціонує як SPA.

REST API – архітектурний стиль для побудови серверної частини. REST (Representational State Transfer) – це архітектурний стиль взаємодії між клієнтом та сервером, заснований на принципах простоти, масштабованості й незалежності (див. рис. 2.9) [19].

Основні принципи REST:

– клієнт-серверна модель: фронтенд і бекенд незалежні один від одного – це дозволяє розвивати, масштабувати та оновлювати кожну частину окремо;

– безстановість (stateless): сервер не зберігає інформацію про стан

клієнта між запитами – кожен запит містить всю необхідну інформацію;

– єдиний інтерфейс: взаємодія здійснюється через стандартні HTTP-методи:
    а) GET – отримання даних;
    б) POST – створення;
    в) PUT/PATCH – оновлення;
    г) DELETE – видалення;
– ресурсна структура: кожен елемент системи представляється окремим ресурсом:
    а) /products;
    б) /products/{id};
    в) /categories;
    г) /users/{id};
– формат обміну даними: на практиці – JSON.

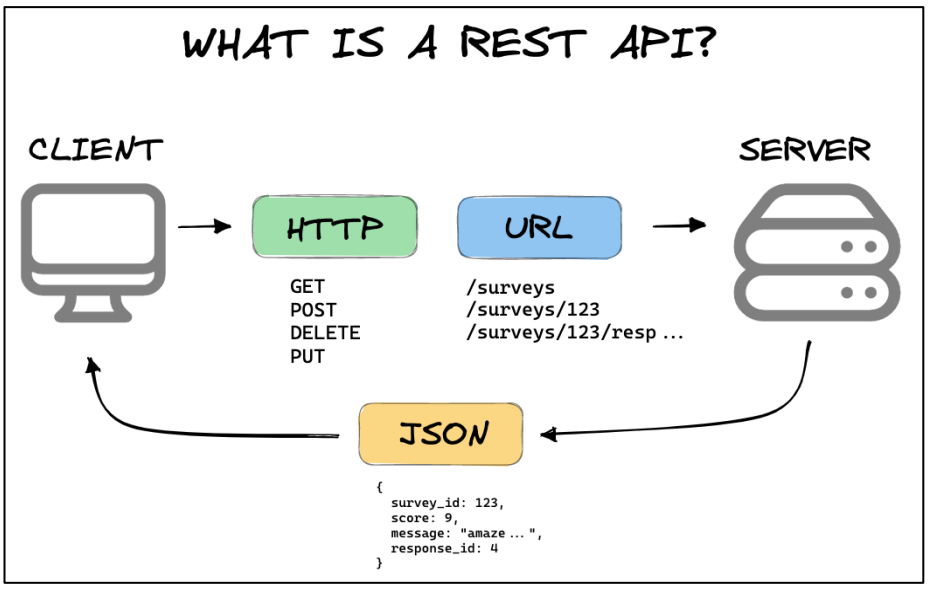


Рисунок 2.9 – Сутність Representational State Transfer [19]

Переваги REST API в проєкті:

– повна незалежність фронтенду та бекенду;
– легка інтеграція з мобільними додатками, сторонніми сервісами;

– можливість масштабування бекенду окремо;
– чіткі контракти між сервісами;
– спрощений контроль прав доступу.

У даному проєкті REST API використовується для:

– отримання списку товарів;
– отримання категорій;
– пошуку і фільтрації;
– авторизації через JWT;
– обробки запитів адміністратора (додавання/редагування товарів).

SPA (Single Page Application) – архітектура клієнтської частини. SPA – це вебзастосунок, який працює в межах однієї HTML-сторінки, а контент оновлюється динамічно за допомогою JavaScript (рис. 2.10) [20].

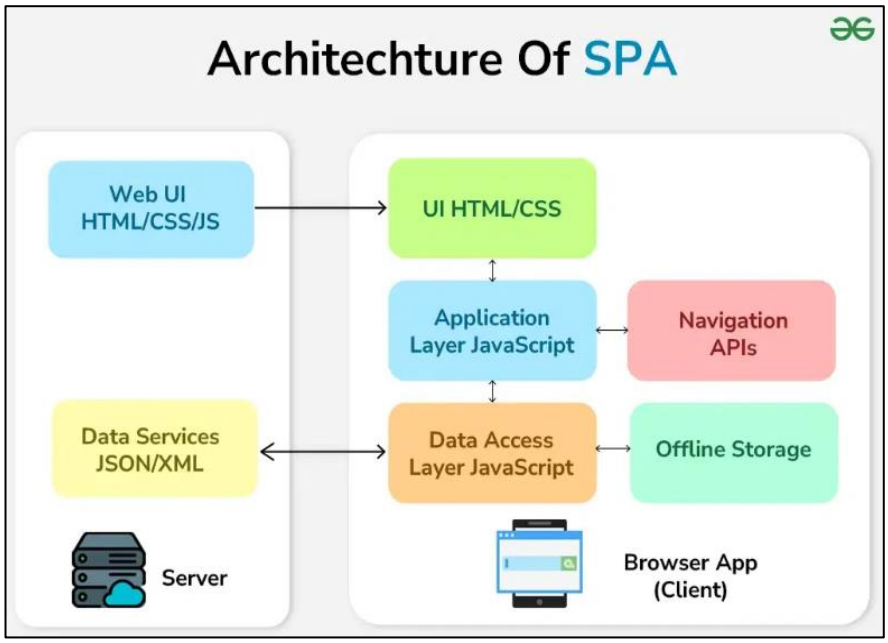


Рисунок 2.10 – Архітектура SPA [20]

Основні характеристики SPA:

– інтерфейс без перезавантаження сторінки: всі переходи виконуються на стороні браузера, що робить роботу додатку швидкою і плавною;
– функціональність, схожа на desktop-додатки: додаток реагує миттєво, забезпечує інтерактивність, анімації, динамічні списки;

– чітке розділення логіки та відображення: UI реалізується на JS-фреймворку (у проєкті – Vue.js 3), який: керує станом, формує віртуальний DOM, оновлює інтерфейс ефективно;
– завантаження даних через REST API: SPA не містить серверної логіки – вона делегована API [20].

Переваги SPA в проєкті:

– дуже швидка робота інтерфейсу;
– зручність перегляду каталогу, фільтрів, пошуку;
– можливість перетворити додаток у PWA;
– ефективне кешування даних;
– чудовий користувацький досвід (UX);
– можливість використання модулів (Pinia, VeeValidate, Tailwind).

У проєкті SPA забезпечує:

– відображення каталогу;
– перегляд карток товару;
– пошук та фільтрацію;
– роботу з обраними товарами;
– форму зворотного зв’язку;
– авторизацію користувача.

Взаємодія **REST API + SPA**. Архітектура виглядає так:

1) SPA (Vue.js): відображає інтерфейс, керує станом (Pinia), валідує форми (VeeValidate), надсилає запити до API;
2) REST API (FastAPI): обробляє запити, перевіряє JWT токени, виконує CRUD операції, працює з SQLAlchemy, взаємодіє з PostgreSQL;
3) база даних (PostgreSQL): зберігає товари, категорії, користувачів.

SPA (Vue.js) → REST API (FastAPI) → PostgreSQL

← JSON responses ←

Переваги архітектури REST API + SPA:

– висока швидкість інтерфейсу: оновлюється лише частина сторінки, а не весь документ;

– гнучкість і масштабованість: можна окремо масштабувати бекенд і фронтенд;
– легкість розширення: пізніше можна додати мобільний додаток, який буде працювати з тим самим REST API;
– незалежність команд розробки: frontend і backend можуть розвиватися паралельно;
– підтримка мікросервісної архітектури: REST API легко розбити на модулі або мікросервіси;
– універсальність: API можна використовувати для інтеграції з CRM, ERP, іншими сайтами, маркетинговими сервісами.

Отже, архітектура REST API + SPA є оптимальним вибором для створення сучасного вебсервісу презентації товарів. Вона забезпечує швидку взаємодію з користувачем, чіткий розподіл обов'язків між фронтендом і бекендом, масштабованість, безпеку й гнучкість, можливість подальшого розвитку до повного e-commerce рішення.

2.1.4 Опис бази даних

Розглянемо базу даних для проєкту, яка включає використання PostgreSQL як основної СУБД та Alembic як інструменту для міграцій.

PostgreSQL – основна реляційна база даних. PostgreSQL – це високопродуктивна, надійна та функціональна система керування реляційними базами даних (СУБД), яка підтримує складні SQL-запити, транзакційність та розширення. Вибір PostgreSQL у проєкті обумовлений її стабільністю, відкритим кодом, високою продуктивністю та широкою підтримкою у професійному середовищі.

Особливості PostgreSQL можна подати через такі характеристики:

– ACID-транзакційність: забезпечує цілісність і надійність даних навіть у разі збоїв;

– потужний SQL-функціонал: підтримує складні запити, тригери, процедури, функції збереження, CTE (WITH-вирази);
– типи даних розширеної складності: PostgreSQL дозволяє використовувати JSON / JSONB, масиви, ENUM, геодані, розширення (наприклад, pg_trgm для пошуку);
– повнотекстовий пошук: зручно використовувати для пошуку товарів у каталозі;
– масштабованість та індексація: підтримує BTREE, GIN, GiST, HASH індекси – це робить роботу з великими обсягами товарів, категорій та користувачів ефективною.

У проєкті база даних PostgreSQL використовується для зберігання всіх структурованих даних, зокрема: інформації про товари; даних про категорії; інформації про користувачів; ролей користувачів (адміністратор, редактор, клієнт); даних для пошуку та фільтрації; логів операцій або історії змін.

PostgreSQL є центральним сховищем системи, яке забезпечує: надійність, швидкість вибірки, консистентність даних.

Alembic – система керування міграціями. Alembic – це інструмент для управління схемою бази даних (міграціями), який інтегрується з ORM SQLAlchemy.

Міграції – це процес створення, зміни та оновлення структури таблиць у базі даних у контрольований спосіб. Переваги міграцій представлено на рисунку 2.11 [22].

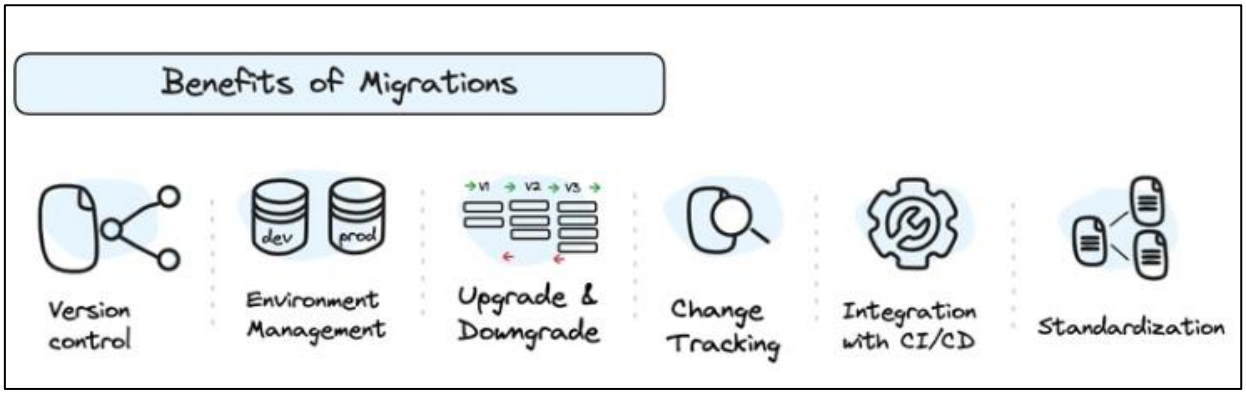


Рисунок 2.11 – Переваги міграції [22]

Основні функції Alembic (див. рис. 2.12) [22]:

– автоматичне створення міграцій: на основі змін у моделях SQLAlchemy;
– відстеження версій БД: міграції зберігаються у окремих скриптах, що дозволяє повернутися до будь-якої попередньої версії;
– командне розгортання: команда розробників може синхронізувати структуру БД на різних середовищах: dev, staging, production;
– безпечне оновлення: Alembic генерує SQL-команди створення таблиць, зміни стовпців, видалення, індексації, створення зв’язків.

Саме функції Alembic обумовили його використання в проєкті для:

– створення таблиць для товарів, користувачів, категорій;
– додавання або редагування колонок без втрати існуючих даних;
– оновлення схеми під час зміни бізнес-вимог;
– підтримка історії змін БД;
– контроль цілісності структури в командній роботі.

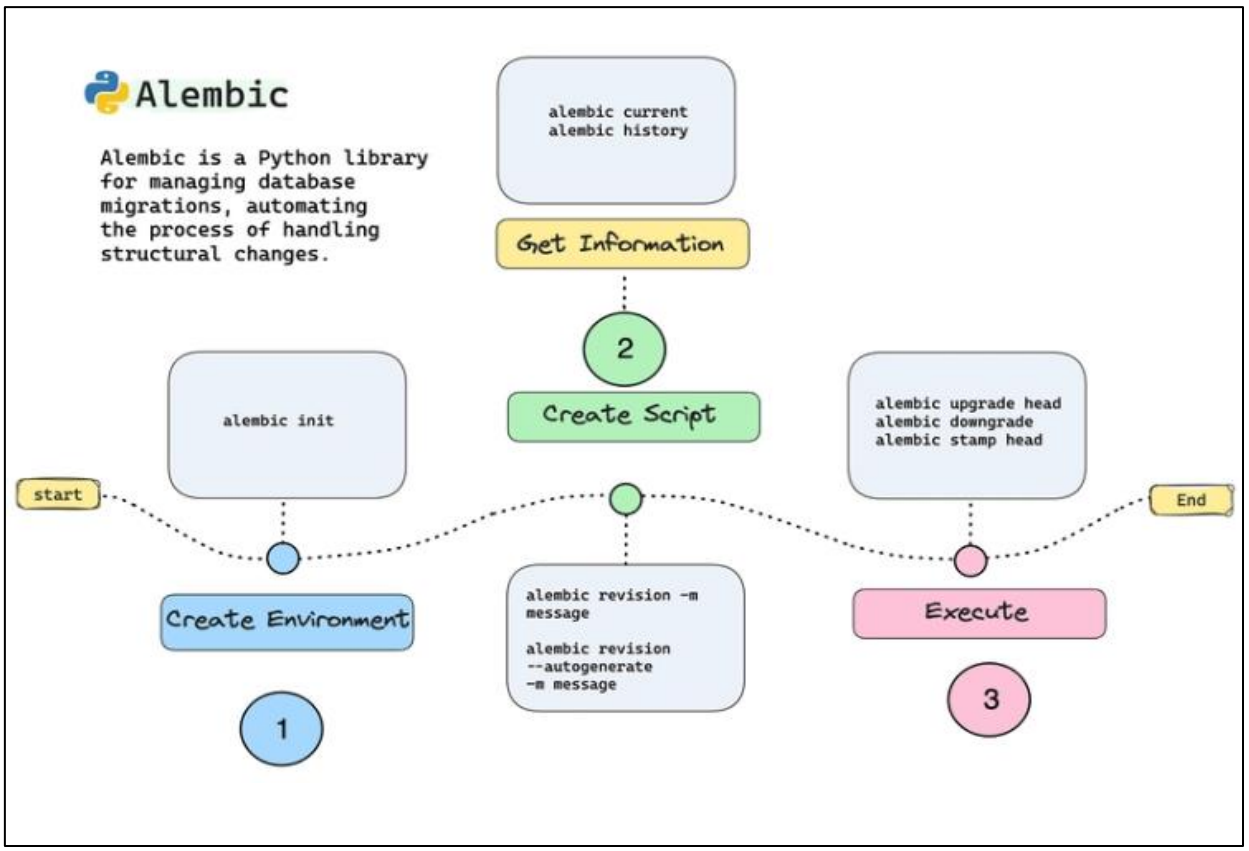


Рисунок 2.12 – Alembic [22]

PostgreSQL + Alembic разом працюють наступним чином:

1) розробник створює або змінює модель у SQLAlchemy, наприклад, додає поле price до таблиці products;

2) викликає команду міграції alembic revision --autogenerate -m «Add price to products»;

3) генерується файл міграції – у ньому описані SQL-команди: створити/змінити колонку, оновити індекси, змінити тип даних;

4) застосування міграції: alembic upgrade head – усі зміни вносяться в PostgreSQL;

5) сервер API працює з оновленою схемою без збоїв.

Ключовими перевагами використання PostgreSQL + Alembic у проєкті є:

– надійність даних: PostgreSQL забезпечує ACID-транзакції;
– масштабованість: індексування та оптимізація запитів підтримують великі каталоги;
– контрольований розвиток: Alembic дозволяє безпечно оновлювати структуру БД;
– зменшення помилок: автоматична генерація міграцій мінімізує ручні SQL-помилки;
– прозора командна робота: кожен член команди має однакову структуру БД;
– готовність до розширення: надалі можна легко додати таблиці кошик, замовлення, логування подій, історію цін.

Отже, використання PostgreSQL як основної СУБД та Alembic для керування міграціями створює надійну, гнучку й масштабовану базу даних, яка: повністю відповідає вимогам сучасних вебплатформ; легко підтримується та оновлюється; гарантує стабільність і цілісність даних; дозволяє ефективно працювати з великими наборами даних. Цей стек ідеально підходить для системи презентації товарів, де важливі: структурованість, безпека, продуктивність, довготривала підтримка.

2.2 Розробка структурних і поведінкових діаграм

Для візуалізації логіки та структури вебплатформи застосовуються діаграми UML. Структурна UML-діаграма класів (Class Diagram) наведена на рисунках 2.13 та 2.14. Саме діаграми UML забезпечують наочне представлення компонентів, зв’язків між ними та можливі сценарії взаємодії користувачів із системою.

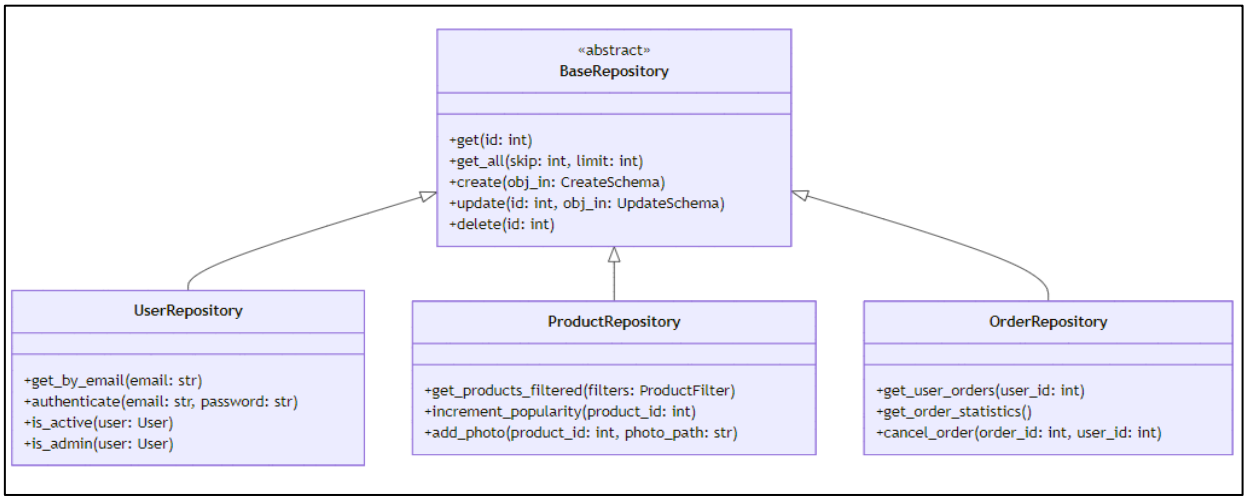


Рисунок 2.13 – UML-діаграма класів: Repository Pattern (CRUD шар)

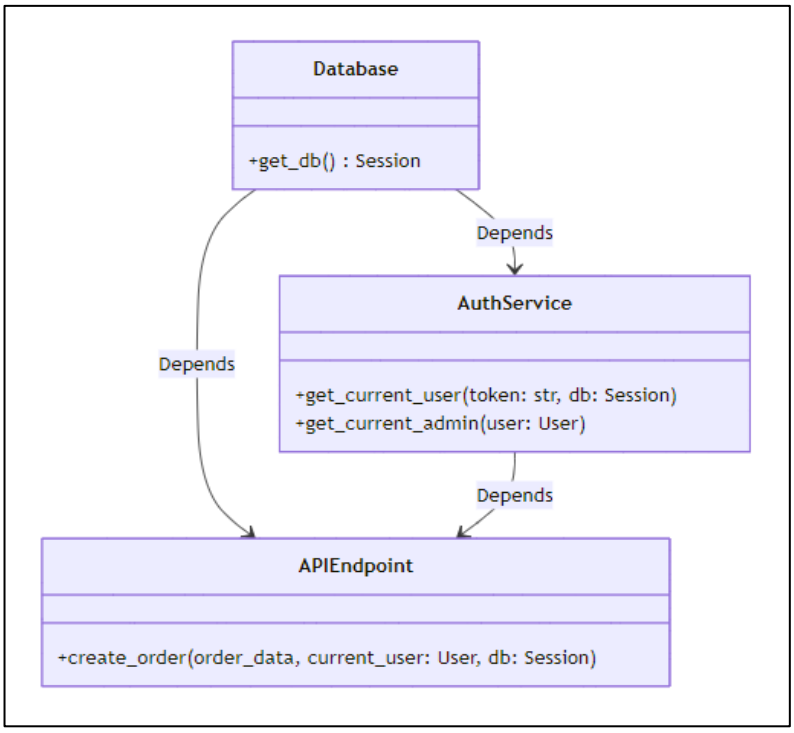


Рисунок 2.14 – UML-діаграма класів: Dependency Injection Pattern (FastAPI)

State Diagram (діаграма станів) – це тип UML-діаграми, який описує поведінку об’єкта або системи через послідовність станів та переходів між ними. Вона показує, як об’єкт реагує на події, змінює свої стани та переходить з одного стану в інший. State Diagram – це UML-інструмент для опису того, як система змінює стани під впливом подій, що допомагає: візуалізувати поведінку, уникати логічних помилок, спростити розробку, полегшити тестування, описати бізнес-регламенти. Це одна з найважливіших поведінкових діаграм у UML.

Діаграма життєвого циклу замовлень наведена на рисунку 2.15.

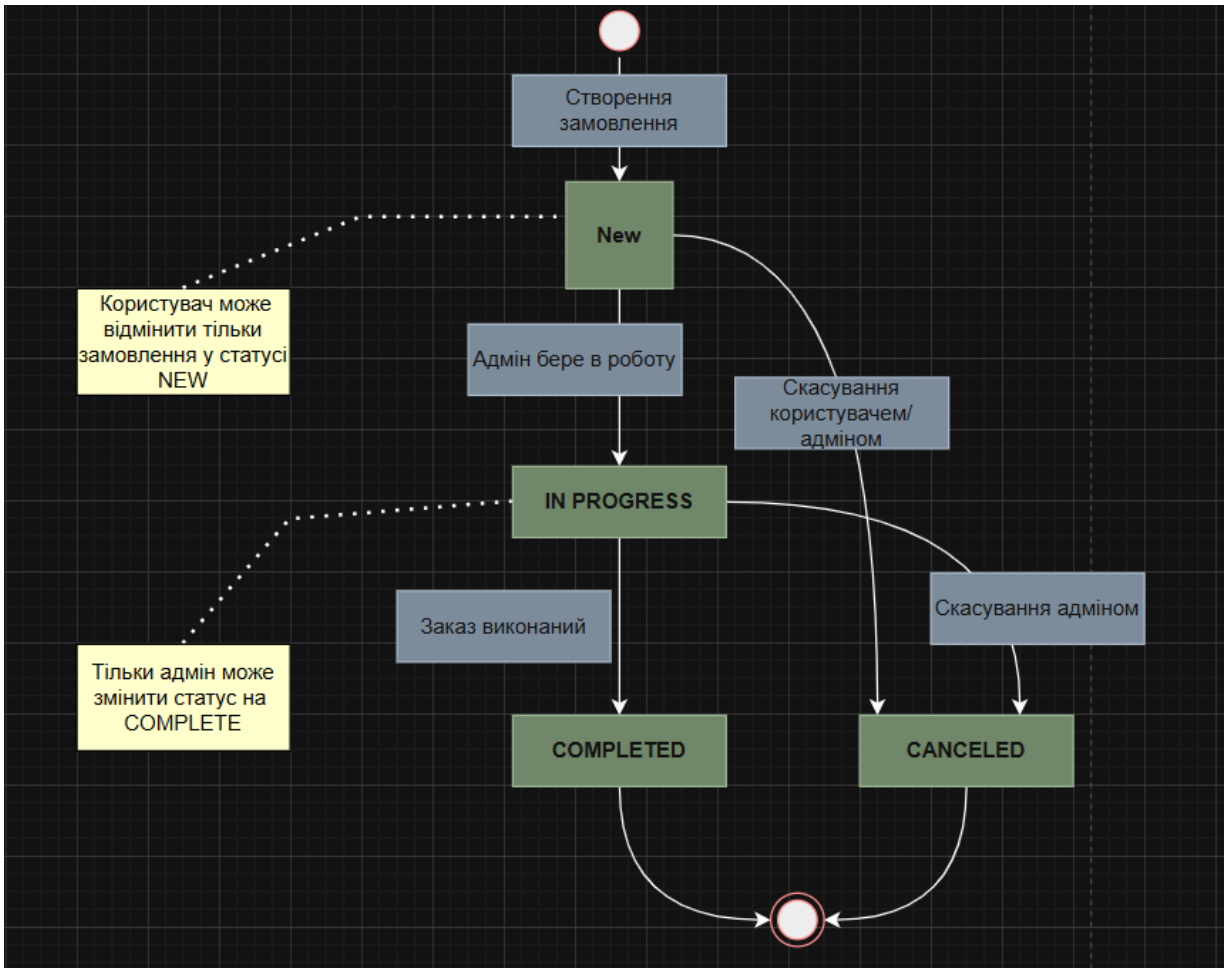
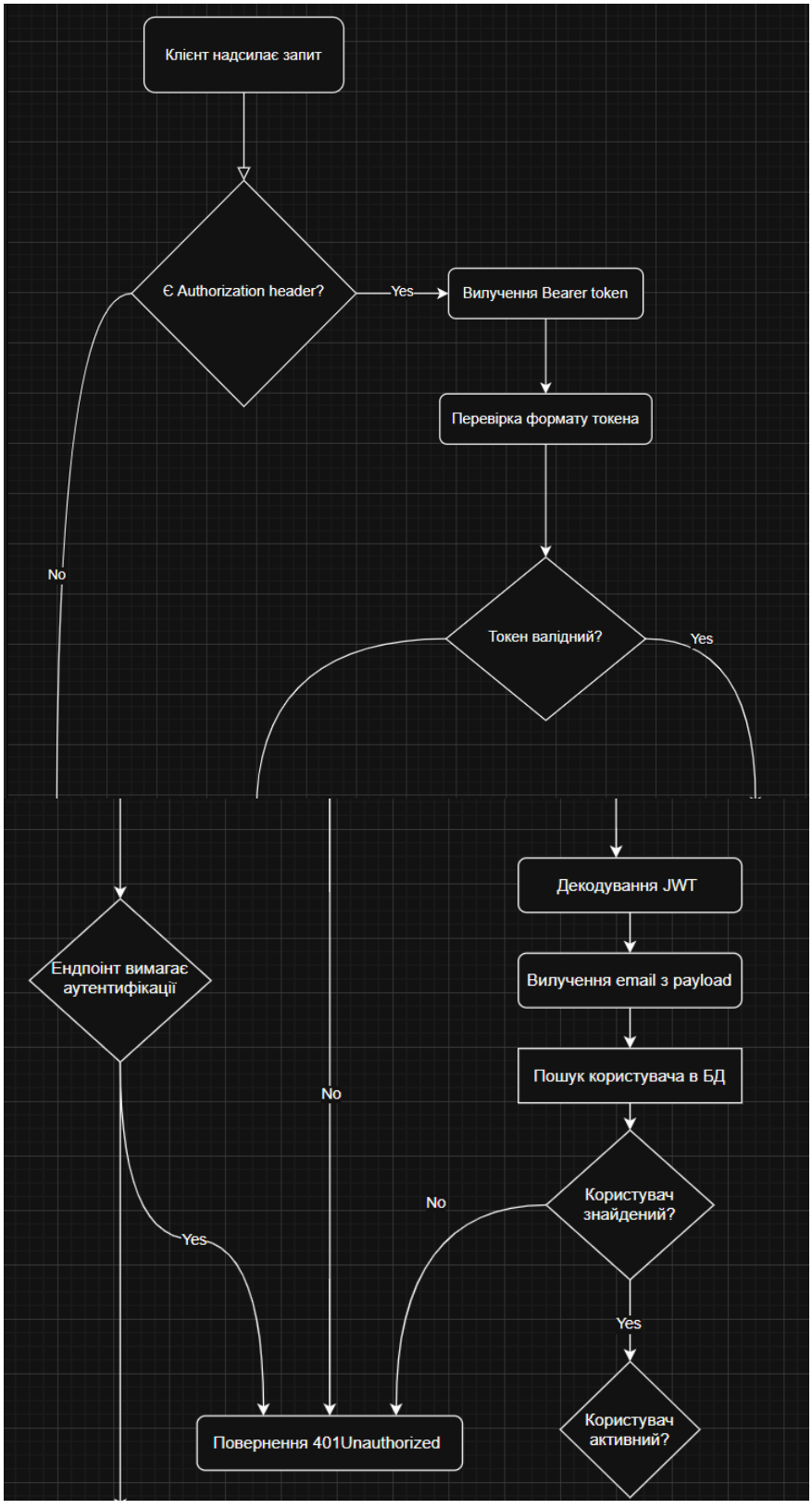


Рисунок 2.15 – State Diagram життєвого циклу замовлень

Для графічного представлення алгоритму, бізнес-процесу або послідовності дій у вигляді блоків та стрілок створюється Flowchart diagram (блок-схема), кожен блок якої відповідає певній операції або рішенню, а

стрілки показують порядок переходу між етапами. Flowchart – це один із найстаріших та найпоширеніших інструментів моделювання процесів.

Логіка процесу функціонування розроблюваної платформи простим візуальним способом наведена на Flowchart diagram на рисунку 2.16.



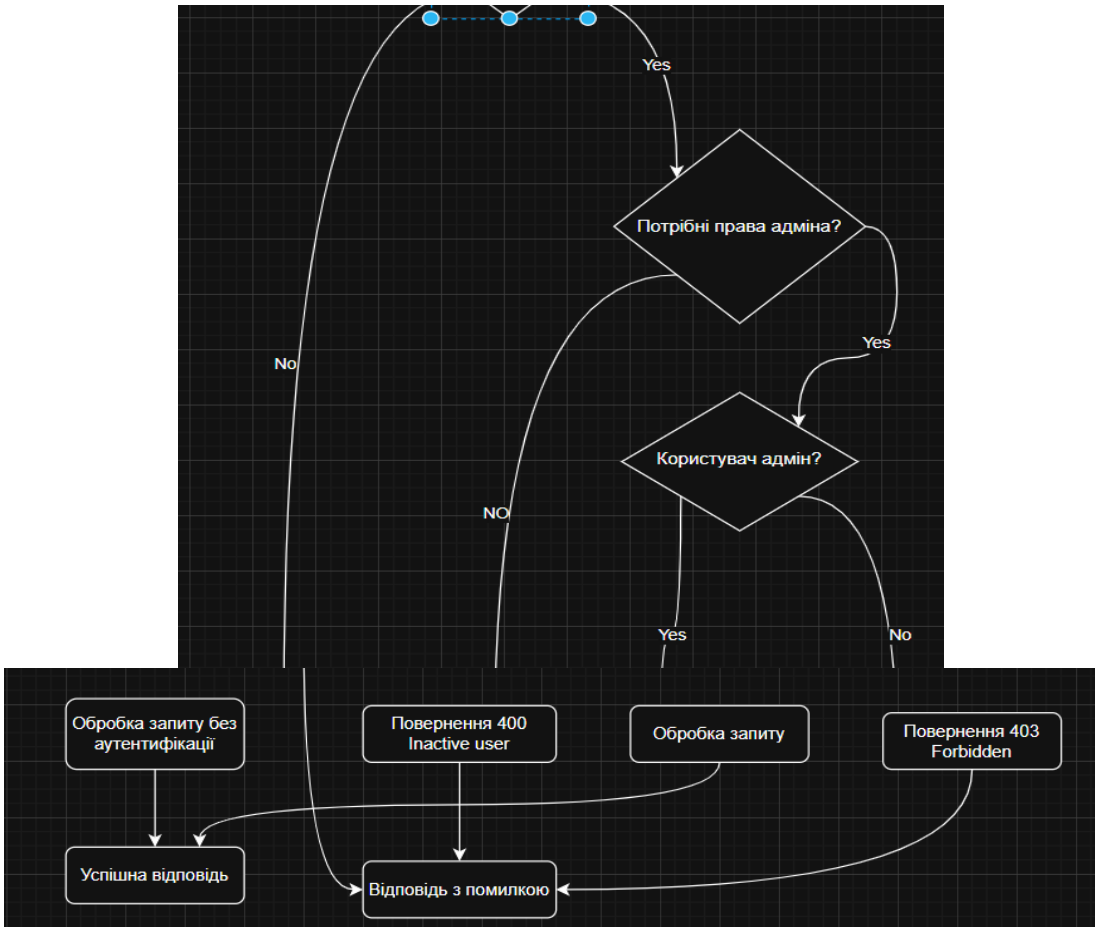


Рисунок 2.16 – Flowchart diagram

ERD (Entity-Relationship Diagram) – це діаграма сутність-зв’язок, яка відображає структуру даних у системі, взаємозв’язки між сутностями та логіку організації бази даних. ERD є однією з ключових діаграм у проєктуванні інформаційних систем та баз даних. ERD-схема – це ключовий документ проєктної документації, що визначає логічну архітектуру бази даних. Вона забезпечує розуміння структури системи, полегшує технічну реалізацію, покращує комунікацію і допомагає уникнути помилок на ранніх етапах.

ERD є фундаментом для: проєктування БД, побудови API, реалізації сервісів, тестування, подальшого масштабування системи. RD дозволяє наочно показати: які сутності існують у системі; які атрибути належать кожній сутності; як сутності пов’язані між собою. Оскільки REST API працює з ресурсами, які відповідають сутностям БД, ERD допомагає: визначити ендпоінти; логіку CRUD операцій; структуру JSON-відповідей; авторизацію та ролі. ERD для розроблюваного проєкту наведена на рисунку 2.17.

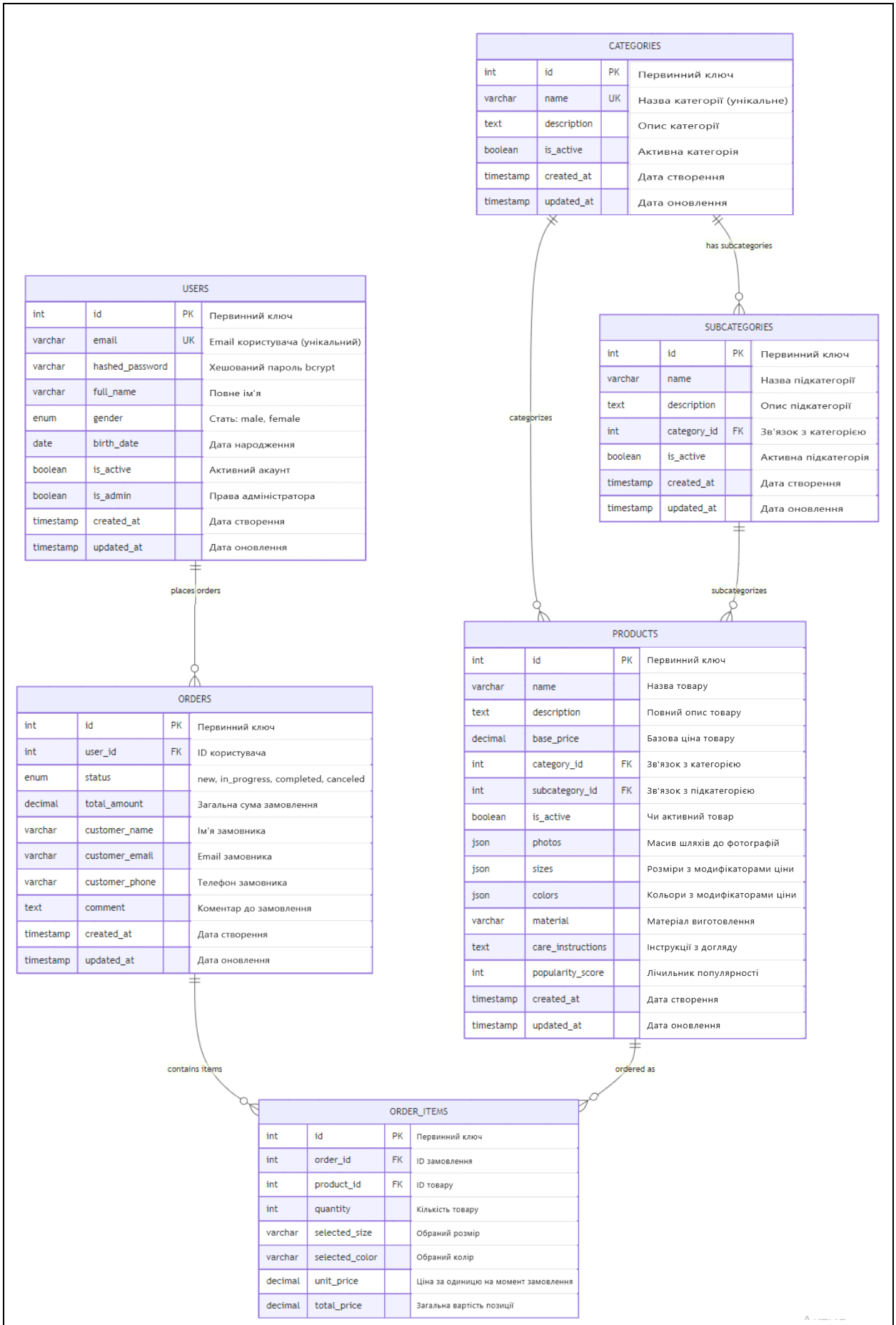


Рисунок 2.17 – Детальна ERD-схема проєкту

Sequence Diagram (діаграма послідовності) – це одна з ключових поведінкових UML-діаграм.

Sequence Diagram – це графічне представлення сценарію, яке демонструє: порядок виконання дій у часі, учасників (акторів або об’єктів), повідомлення (messages), які вони надсилають одне одному, життєву лінію кожного об’єкта (lifeline), активації (activation bars). Це фактично «стрічка часу», по якій рухається логіка виконання.

Для розкриття алгоритмів обробки ключових бізнес-процесів платформи, зокрема процесу оформлення замовлення, доцільно представити діаграму послідовності (Sequence Diagram). Sequence Diagram для процесу «Сценарій створення замовлення» наведена на рисунку 2.18.

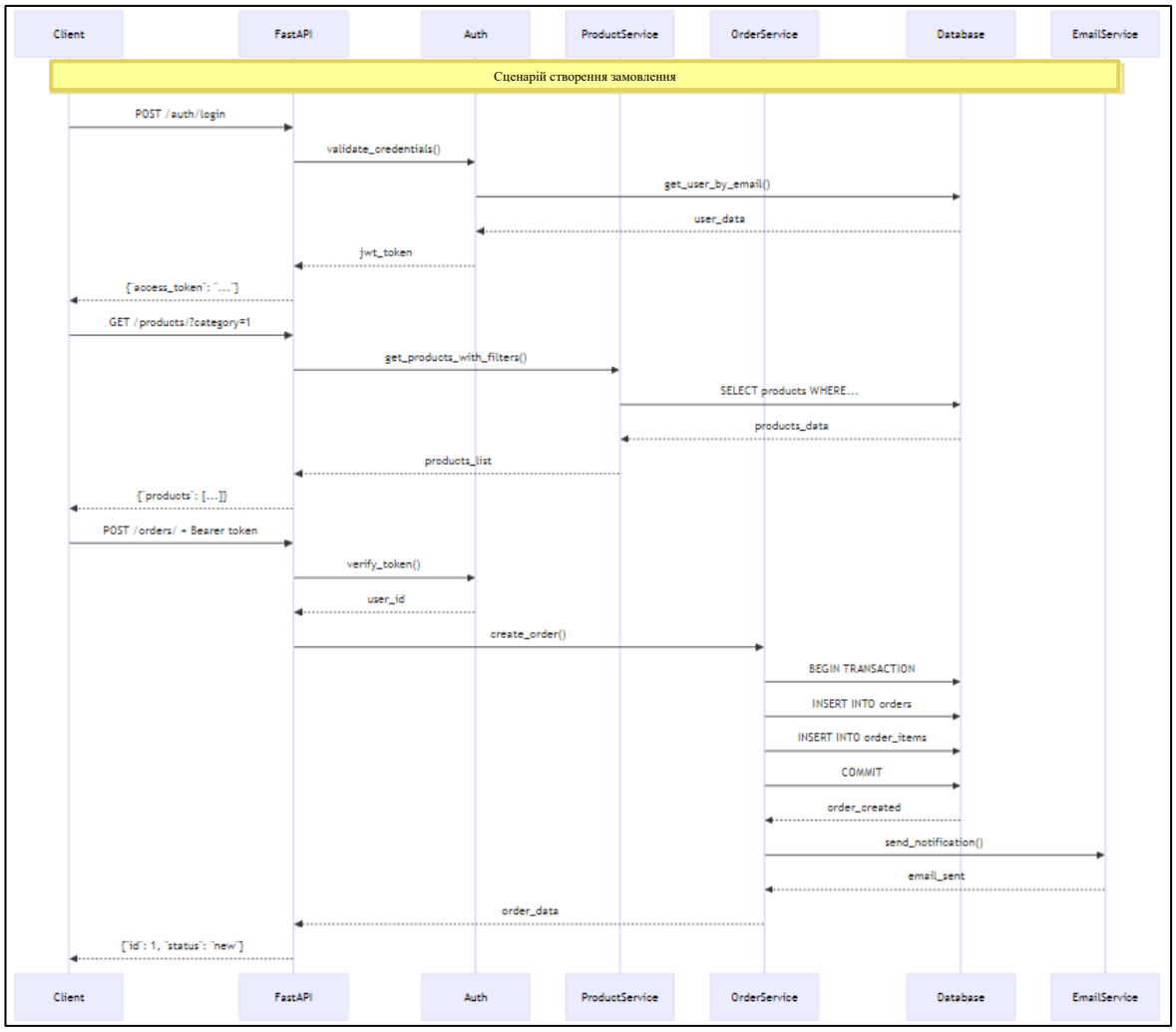


Рисунок 2.18 – Sequence Diagram для процесу «Сценарій створення замовлення»

Таким чином, у рамках проєктного розділу було визначено й обґрунтовано вибір програмних і апаратних засобів для реалізації сучасної, безпечної й масштабованої платформи продажу товарів через Інтернет. Надана структура та архітектура системи відповідають передовим світовим практикам створення web-e-commerce рішень і забезпечують можливість швидкого розвитку й подальшої кастомізації функціоналу.

Описані логічна структура даних, функціональні вимоги, базові сценарії використання платформи та представлено базові діаграми, що ілюструють взаємодію основних компонентів системи. Запропонований підхід закладає надійний фундамент для подальшої безпосередньої реалізації та тестування платформи.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Реалізація користувацького інтерфейсу

Одним із ключових факторів успіху сучасної e-commerce-платформи є зручність, інтуїтивність та естетична привабливість користувацького інтерфейсу. Високий рівень UX/UI-дизайну дозволяє не лише залучати нових користувачів, а й ефективно утримувати наявних клієнтів, мінімізуючи когнітивне навантаження та підвищуючи рівень конверсії.

В роботі розроблено сайт для презентації продукції реального виробника м’яких меблів для тварин «Romi Pet Comfort». Основна концепція інтерфейсу базується на забезпеченні прямого зв’язку між клієнтом та виробником, що дозволяє реалізувати можливість індивідуального замовлення за конкретними параметрами: розміром, типом тканини та кольоровою гамою тощо.

Візуальною основою для створення інтернет-магазину став наявний контент сторінки бренду в мережі Instagram: https://www.instagram.com/romi_petcomfort?igsh=MXNwcjdjM2pndDlhNA.

Прототипи основних сторінок платформи наведені на рисунках 3.1 – 3.4.

Для входу в акаунт необхідно попередньо зареєструватися. Форма реєстрації містить стандартні поля: повне ім’я, електронна адреса, пароль, стать та дата народження (див. рис. 3.1).

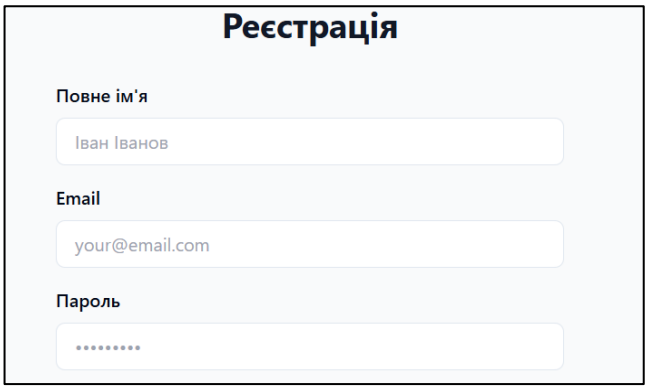


Рисунок 3.1 – Форма реєстрації для входу в акаунт

Для зареєстрованих користувачів форма входу виглядає так, як представлено на рисунку 3.2.

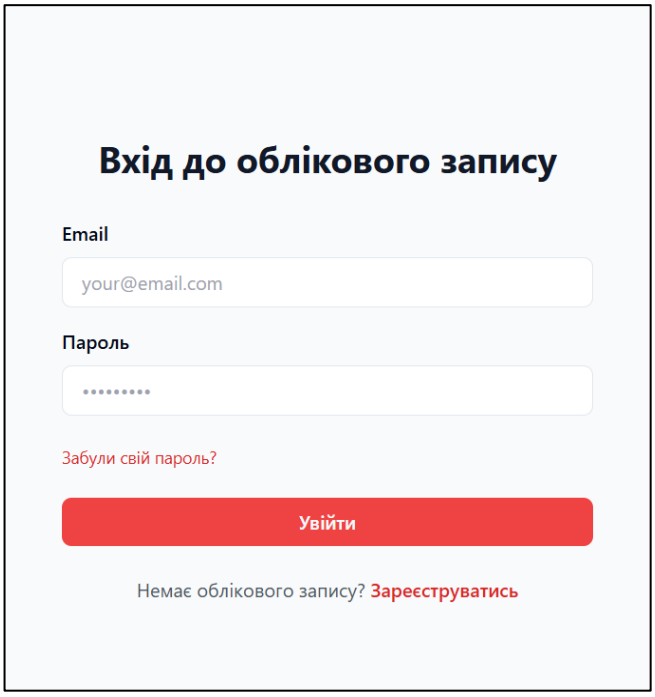


Рисунок 3.2 – Форма входу в акаунт

На рисунку 3.3 представлено головну сторінку розробленого інтернет-магазину «Romi Pet Comfort». Інтерфейс спроєктовано за принципами мінімалізму та фокусування уваги користувача на ключових цільових діях.

Основними елементами сторінки є:

– верхня навігаційна панель (Header): містить логотип бренду, посилання на основні розділи («Головна», «Товари»), кошик із лічильником обраних позицій та меню авторизації користувача;
– Hero-секція (головний банер): яскравий візуальний блок із закликом до дії (CTA-кнопка «Дивитись товари»), що презентує місію бренду та стимулює початок перегляду каталогу;
– блок популярних категорій: реалізований у вигляді сітки інтерактивних карток («Лежаки», «Іграшки», «Бокси», «Одяг для собак» тощо) – кожна картка має лаконічний опис, що допомагає користувачу швидко зорієнтуватися в асортименті.

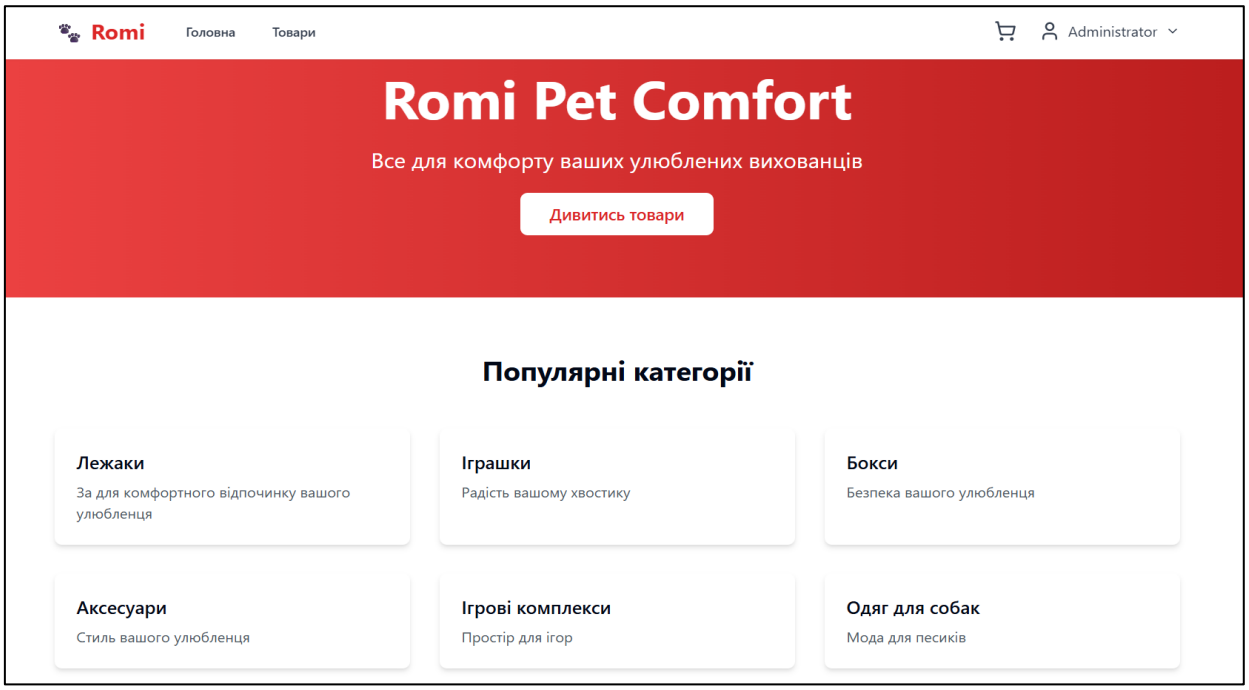


Рисунок 3.3 – Головна сторінка інтернет-магазину

На рисунку 3.4 представлено інтерфейс аналітичного дашборду адміністратора, який є ключовим інструментом для моніторингу бізнес-показників платформи «Romi Pet Comfort». Дашборд забезпечує візуалізацію критично важливих даних у реальному часі.

Функціональні блоки дашборду включають:

– верхня панель статистичних індикаторів: складається з чотирьох карток, що відображають кількісні показники замовлень за їх станом: «Усього замовлень», «Нові», «В обробці» та «Виконані» – це дозволяє менеджеру миттєво оцінити поточну операційну завантаженість;
– фінансовий блок (Виторг): відображає сумарний дохід магазину, значення середнього чека та загальну кількість зареєстрованих користувачів – ці метрики є фундаментальними для оцінки комерційної ефективності проєкту;
– блок статусів замовлень: деталізований список, що дублює інформацію про стани замовлень (включаючи «Скасовано»), забезпечуючи швидкий перехід до аналізу причин відмов або затримок;

– бічна панель навігації (Sidebar): забезпечує швидкий доступ до основних модулів управління системою: «Товари», «Категорії», «Замовлення» та «Користувачі».

Також передбачено кнопку швидкого повернення до публічної частини сайту.

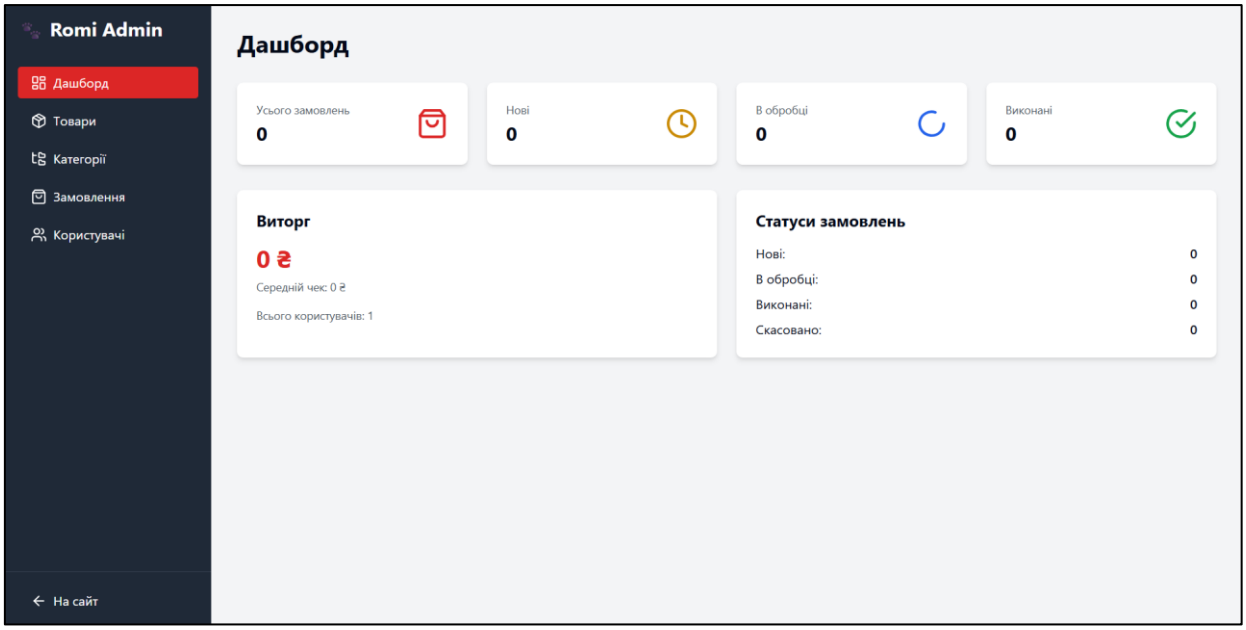


Рисунок 3.4 – Дашборд інтернет-магазину

Лістинги коду для основного макету платформи та адміністративного макету наведені відповідно в додатках А і Б.

3.2 Опис завершених компонент проєкту

Результатом етапу розробки стала цілісна програмна екосистема, що складається з взаємопов’язаних серверних та клієнтських модулів. Нижче наведено детальний опис архітектурних складових та функціональних компонентів проєкту, що забезпечують стабільну роботу платформи.

Серверна частина (**Backend**) – реалізована у повному обсязі.

Архітектура та документація:

– детальний README.md з API документацією;
– ARCHITECTURE.md з UML діаграмами;
– API_EXAMPLES.md з прикладами використання;
– PROJECT_STATUS.md зі статусом розробки.

Моделі даних:

– User (користувачі);
– Category (категорії);
– Product (товари);
– Cart & CartItem (кошик);
– Order & OrderItem (замовлення).

API ендпоінти:

– аутентифікація (/auth/*);
– управління товарами (/products/*);
– управління категоріями (/categories/*);
– кошик покупок (/cart/*);
– замовлення (/orders/*);
– адміністративні функції (/admin/*).

Клієнтська частина (**Frontend**) – перебуває на фінальній стадії розробки (ступінь готовності 95%), що включає інтеграцію базових сервісів та інтерфейсних рішень.

Інфраструктура:

1) налаштування проєкту:
   – Vite + Vue 3 конфігурація;
   – TailwindCSS інтеграція;
   – ESLint + Prettier налаштування;
   – структура папок і файлів;
2) маршрутизація:
   – Vue Router 4 із захищеними маршрутами;
   – навігаційні гарди;
   – метадані сторінок;

- 404 сторінка;

3) управління станом:
   - Pinia stores (auth, products, cart);
   - Persistence в localStorage;
   - глобальний стан помилок;
4) API інтеграція:
   - Axios клієнт з автоматичним оновленням токенів;
   - обробка помилок;
   - типізовані методи API.

Компоненти та сторінки:

1) Layout компоненти:
   - AppHeader з навігацією та пошуком;
   - AppFooter з контактами та посиланнями;
   - адаптивне меню для мобільних;
2) продуктові компоненти:
   - ProductCard для карток товарів;
   - ProductListItem для списку;
   - Pagination компонент;
3) основні сторінки:
   - Home – головна з банерами та категоріями;
   - Products – каталог з пошуком та фільтрами;
   - Cart – кошик з управлінням товарами;
   - Profile – профіль користувача з редагуванням;
4) аутентифікація:
   - Login – сторінка входу з валідацією;
   - Register – реєстрація з перевіркою пароля;
   - захищені роути;
5) адміністративна панель:
   - AdminLayout – макет адмінки;
   - Dashboard – дашборд зі статистикою;

- Products – управління товарами;
- Categories – управління категоріями;
- Orders – управління замовленнями;
- Users – управління користувачами;
- ProductForm – створення/редагування товарів;

6) допоміжні сторінки:
   - About – інформація про компанію;
   - Contact – контактна форма;
   - NotFound – 404 сторінка.

3.3 Опис функцій платформи та їхньої завершеності

Для систематизації результатів розробки було проведено оцінку функціонального наповнення системи за критеріями готовності до експлуатації. Розподіл функцій на категорії дозволяє чітко розмежувати ядро системи (MVP), яке вже функціонує, та додаткові модулі, що заплановані для майбутніх ітерацій розробки.

До переліку повністю реалізованих функцій належать:

- система реєстрації та багатофакторної авторизації користувачів;
- модуль візуалізації та навігації за каталогом товарів;
- механізми пошуку та динамічної фільтрації асортименту;
- повний цикл взаємодії з кошиком покупок (додавання, видалення, коригування);
- кабінет користувача з персональними налаштуваннями;
- розширена адміністративна панель для керування контентом;
- повний набір CRUD-операцій для всіх системних сутностей;
- кросбраузерна адаптивна верстка інтерфейсу.

Функціональні модулі, що перебувають на стадії прототипування або потребують доопрацювання:

– оформлення замовлення (ступінь готовності – 5%): на даному етапі підготовлено базовий макет сторінки, що потребує подальшої інтеграції з API платіжних систем та логістичних сервісів;
– детальна сторінка продукту (ступінь готовності – 10%): реалізовано каркас сторінки, який у майбутньому буде доповнений блоками відгуків, рейтингових систем та розширених специфікацій;
– історія замовлень (ступінь готовності – 15%): створено інтерфейсну заглушку, яка потребує підключення до відповідних ендпоінтів бази даних для динамічного відображення архіву операцій.

Перспективний план розвитку (Roadmap), узгоджений із замовником, передбачає впровадження таких можливостей:

– система відгуків та рейтингів;
– інструментарій для персоналізації (Wishlist та порівняння товарів);
– маркетингові інструменти (промокоди, системи знижок);
– сервіси автоматизації (Email-сповіщення та мультивалютність/багатомовність);
– інтеграція з платіжними шлюзами та сервісами доставки;
– темна тема.

Поточний стан готовності платформи до експлуатації та перелік пріоритетних напрямків для подальшої розробки зведено у таблиці 3.1.

Таблиця 3.1 – Статус готовності та перспективи розвитку платформи

Категорія готовності	**Функціональний модуль / Компонент**	**Поточний статус та опис**
Готові до експлуатації	Каталог товарів	Повністю функціональний, підтримує відображення актуального асортименту.
	Кошик покупок	Забезпечує повний цикл операцій з вибору та підготовки товарів до замовлення.
	Система автентифікації	Повністю готова до використання в продакшн-середовищі.
	Адміністративна панель	Реалізовано повний функціонал для керування контентом та користувачами.
	Адаптивний дизайн	Коректно працює на всіх типах пристроїв (десктоп, планшет, смартфон).

Категорія готовності	Функціональний модуль / Компонент	Поточний статус та опис
Потребують реалізації	Процес оплати	Необхідна інтеграція з зовнішніми платіжними шлюзами (payment gateways).
	Email-сервіс	Впровадження автоматичних повідомлень про статус замовлення.
	Система доставки	Додавання алгоритмів розрахунку вартості та термінів логістики.
	Просунуті фільтри	Розширення можливостей пошуку за специфічними характеристиками товарів.
	SEO-оптимізація	Налаштування метатегів, генерація sitemap та оптимізація під пошукові системи.

Загалом, реалізований функціонал формує надійний базис для запуску платформи у форматі MVP (Minimum Viable Product). Це дозволяє вже зараз демонструвати продукт замовнику та використовувати його як основу для подальшого нарощування сервісних можливостей.

3.4 Тестування функціоналу системи

Тестування є важливим етапом розробки і впровадження вебплатформи, оскільки воно дозволяє виявити помилки на ранніх стадіях, забезпечити коректність виконання бізнес-логіки, перевірити якість реалізації функціоналу та відповідність системи вимогам, визначеним у проєктному розділі.

Розглянемо основні види проведеного тестування.

Модульне тестування: передбачає перевірку окремих функцій, методів, модулів серверної частини (наприклад, моделі Users, Products, Orders, Reviews тощо). Для цього використовуються бібліотеки unittest, pytest як стандарт у середовищі Python.

Інтеграційне тестування: перевіряється взаємодія між компонентами системи, зокрема – обмін між сервером, базою даних і клієнтським інтерфейсом. Наприклад: створення замовлення користувачем → запис у таблиці Orders → зміна статусу продавцем.

Системне тестування: охоплює повну перевірку платформи з огляду на всі заявлені функції: реєстрації, логін, замовлення, робота з кошиком, фільтрація, перегляд статистики, відгуки, адмін-модуль.

Тестування безпеки: перевіряється реалізація механізмів захисту даних (CSRF, XSS), хешування паролів, розмежування прав доступу, рольова політика.

Тестування продуктивності: стрес-тести моделюють навантаження, роботу з великою кількістю користувачів чи товарів, швидкість відповіді серверу, відмовостійкість системи.

Інструменти для тестування розробленого сервісу включають наступні:

– Pytest – модульне та інтеграційне тестування логіки, моделей;
– Postman – тестування API-запитів, перевірка обробки даних сервером;
– Selenium – автоматизоване енд-ту-енд тестування користувацьких сценаріїв через браузер;
– Coverage.py – аналіз покриття коду тестами.

Детальніше ознайомитися з лістингами програмних скриптів та прикладами тестування функцій можна у Додатку В.

Результати тестування показали наступне: усі основні функціональні модулі, інтерфейсні сценарії, ключові бізнес-процеси платформи пройшли тестування відповідно до вимог. Виявлені незначні недоліки та помилки були усунені в процесі доопрацювання. Проведене тестування підтвердило працездатність та високий рівень надійності системи при базовому та напівінтенсивному навантаженні.

Таким чином, проєкт готовий до використання та демонстрації. Основний функціонал інтернет-магазину реалізований повністю. Система готова для:

– демонстрації замовнику;
– деплоя в продакшен (з базовим функціоналом);
– подальшої розробки додаткових функцій;
– використання в якості основи для інших e-commerce проєктів.

Таким чином, в роботі було розглянуто та реалізовано всі основні етапи створення сучасної вебплатформи для електронної комерції – від проєктування інтерфейсу до впровадження серверної логіки, організації взаємодії з базою даних, забезпечення захисту й тестування функціоналу системи. Отримані результати засвідчують, що обраний стек технологій дозволяє ефективно реалізовувати гнучкі, захищені та масштабовані рішення з урахуванням вимог предметної області.

Комплексне тестування довело відповідність роботи платформи споживчим вимогам, підтвердило надійність механізмів збереження й обробки даних, що формує передумови для впровадження та подальшого розвитку даної інформаційної системи у сфері електронної комерції.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було здійснено повний цикл розробки та впровадження універсальної вебплатформи – сайту для презентації товарів реального виробника, що шиє диванчики для котів і собак та продає їх, з можливостями зв’язку з виробником для замовлення за конкретними параметрами (розмір, колір та інше), з використанням Vue.js.

Для досягнення мети було проведено аналіз предметної області та ринку, обґрунтовано вибір сучасного технологічного стеку і засобів, спроєктовано ефективну архітектуру, структуровано базу даних, реалізовано користувацький інтерфейс, серверну логіку, а також заходи із безпеки платформи.

У рамках проєктного розділу було визначено й обґрунтовано вибір програмних і апаратних засобів для реалізації сучасної, безпечної й масштабованої платформи продажу товарів через Інтернет. Надана структура та архітектура системи відповідають передовим світовим практикам створення web-e-commerce рішень і забезпечують можливість швидкого розвитку й подальшої кастомізації функціоналу.

Описані логічна структура даних, функціональні вимоги, базові сценарії використання платформи та представлено базові діаграми, що ілюструють взаємодію основних компонентів системи. Запропонований підхід закладає надійний фундамент для подальшої безпосередньої реалізації та тестування платформи.

У ході розробки і тестування показано, що створена система відповідає сучасним стандартам, забезпечує зручний функціонал для різних категорій користувачів, гнучко масштабується під потреби ринку та може бути платформою для подальшого розвитку сервісів електронної комерції.

ПЕРЕЛІК ПОСИЛАНЬ

1. Що таке торгова презентація? *WebPR digital agency*. URL : https://blog.webpr.ua/uk/ prosuvannya/stvoryuyemo-prezentacziyu-tovaru-abo-poslugy/ (дата звернення: 12.08.2025).

2. Ознаки ефективного інтернет-магазину. *Letda*. URL : https://letda.com/internet-magazyn-yakyj-prodaye-yakyj-vin/ (дата звернення: 12.08.2025).

3. Docker Documentation. URL : https://docs.docker.com/ (дата звернення: 12.08.2025).

4. Тренди та виклики українського ринку eCommerce у 2024 році. URL : https://uaateam.agency/blog/trendy-ta-vyklyky-ukrainskogo-rynku-ecommerce/ (дата звернення: 16.08.2025).

5. Електронна комерція 2025: практичний перехід бізнесу в онлайн. URL : https://surl.li/kfexea (дата звернення: 30.10.2025).

6. E-commerce платформи: огляд найпопулярніших рішень. URL : https://optoviki.org.ua/e-commerce-platformy-oglyad-najpopulyarnishyh-rishen/ (дата звернення: 28.10.2025).

7. Розробка мобільних додатків для e-commerce. URL : https://optoviki.org.ua/rozrobka-mobilnyh-dodatkiv-dlya-e-commerce/ (дата звернення: 22.08.2025).

8. Плескач В. Л., Гужва І. Ю. Електронна комерція : підручник. Київ : Знання, 2007. 535 с.

9. FastAPI documentation. 2018. URL: https://fastapi.tiangolo.com (дата звернення: 22.08.2025).

10. FastAPI Python Development Company. *QuintaGroup*. URL : https://quintagroup.com/services/python/fastapi/ (дата звернення: 22.08.2025).

11. PostgreSQL Documentation. URL : https://www.postgresql.org/docs/ (дата звернення: 12.10.2025).

12. SQLAlchemy Documentation. URL : https://docs.sqlalchemy.org/ (дата звернення: 12.10.2025).

13. Thanveer M. Understanding JSON Web Tokens (JWT): A Secure Approach to Web Authentication. 2024. *DEV Community*. URL : https://dev.to/thanvs/understanding-json-web-tokens-jwt-the-key-to-secure-authentication-3e79 (дата звернення: 12.10.2025).

14. What are the new features in Vue.js 3? *NextBrain*. URL : https://www.nextbraintech.com/blog/what-are-the-new-features-in-vue-js-3 (дата звернення: 12.10.2025).

15. Tailwind CSS vs Traditional CSS – What's Better? *Linkedin*. URL : https://www.linkedin.com/posts/vikramps98_teamtailwind-teamcss-webdevelopment-activity-7330464622925320192-SrLe/ (дата звернення: 22.10.2025).

16. Guler K. Tailwind CSS vs Traditional CSS. *Medium*. 2025. URL: https://medium.com/@raelsei/tailwind-css-vs-traditional-css-why-ill-never-go-back-2b35bff7ae0d (дата звернення: 22.10.2025).

17. Pinia vs Vuex: Farewell Vuex, Hello Pinia! How to Get Started? *Software Planet Group*. URL : https://softwareplanetgroup.co.uk/piniavsvuex/ (дата звернення: 22.10.2025).

18. Managing Form Validation with Custom Form Components And Vee-Validate. *Ionic Forums*. URL : https://forum.ionicframework.com/t/managing-form-validation-with-custom-form-components-and-vee-validate/206829 (дата звернення: 23.10.2025).

19. REST API Basics – 4 Things you Need to Know. URL : https://mannhowie.com/rest-api (дата звернення: 23.10.2025).

20. What are Single Page Apps? *GeeksforGeeks*. URL : https://www.geeksforgeeks.org/reactjs/what-are-single-page-apps/ (дата звернення: 23.10.2025).

21. Mendez Escobar R. E. Simplify Database Migrations using Python with Alembic. 2024. *Medium*. URL :

https://medium.com/@romina.elena.mendez.escobar/simplify-database-migrations-using-python-with-alembic-f8aa3df36449 (дата звернення: 23.10.2025).

ДОДАТОК А

Лістинг коду для основного макету платформи

```
<template>
  <div class="min-h-screen flex flex-col">
    <header class="bg-white shadow-sm sticky top-0 z-50">
      <nav class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8">
        <div class="flex justify-between items-center h-16">
          <div class="flex items-center space-x-8">
            <router-link to="/" class="flex items-center space-x-2">
              <span class="text-2xl font-bold text-primary-600">🐾 Romi</span>
            </router-link>
            <div class="hidden md:flex space-x-4">
              <router-link to="/" class="text-gray-700 hover:text-primary-600 px-3 py-2 rounded-md
text-sm font-medium">
                Головна
              </router-link>
              <router-link to="/products" class="text-gray-700 hover:text-primary-600 px-3 py-2
rounded-md text-sm font-medium">
                Товари
              </router-link>
            </div>
          </div>

          <div class="flex items-center space-x-4">
            <router-link to="/cart" class="relative p-2 text-gray-700 hover:text-primary-600">
              <ShoppingCart :size="24" />
              <span v-if="cartStore.totalItems > 0" class="absolute -top-1 -right-1 bg-primary-600
text-white text-xs rounded-full h-5 w-5 flex items-center justify-center">
                {{ cartStore.totalItems }}
              </span>
            </router-link>

            <template v-if="authStore.isAuthenticated">
              <div class="relative" ref="userMenuRef">
                <button @click="showUserMenu = !showUserMenu" class="flex items-center space-x-
2 text-gray-700 hover:text-primary-600">
                  <User :size="24" />
                  <span class="hidden md:inline">{{ authStore.user?.full_name }}</span>
                  <ChevronDown :size="16" />
                </button>
                <div v-if="showUserMenu" class="absolute right-0 mt-2 w-48 bg-white rounded-md
```

```
shadow-lg py-1 z-50">
              <router-link to="/profile" class="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100" @click="showUserMenu = false">
                Профіль
              </router-link>
              <router-link to="/orders" class="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100" @click="showUserMenu = false">
                Мої замовлення
              </router-link>
              <router-link v-if="authStore.isAdmin" to="/admin" class="block px-4 py-2 text-sm text-gray-700 hover:bg-gray-100" @click="showUserMenu = false">
                Адмін-панель
              </router-link>
              <button @click="handleLogout" class="block w-full text-left px-4 py-2 text-sm text-gray-700 hover:bg-gray-100">
                Вийти
              </button>
            </div>
          </div>
        </template>
        <template v-else>
          <router-link to="/login" class="btn btn-outline">
            Увійти
          </router-link>
        </template>
      </div>
    </div>
  </nav>
</header>

<main class="flex-1">
  <router-view />
</main>

<footer class="bg-gray-800 text-white mt-12">
  <div class="max-w-7xl mx-auto px-4 sm:px-6 lg:px-8 py-8">
    <div class="grid grid-cols-1 md:grid-cols-3 gap-8">
      <div>
        <h3 class="text-lg font-semibold mb-4">Romi Pet Comfort</h3>
        <p class="text-gray-400">Товари для ваших улюблених пітомцев</p>
      </div>
      <div>
        <h3 class="text-lg font-semibold mb-4">Навігація</h3>
        <ul class="space-y-2">
          <li><router-link to="/" class="text-gray-400 hover:text-white">Головна</router-link></li>
```

```
            <li><router-link to="/products" class="text-gray-400 hover:text-
white">Товари</router-link></li>
          </ul>
        </div>
        <div>
          <h3 class="text-lg font-semibold mb-4">Контакти</h3>
          <p class="text-gray-400">Email: info@romipet.com</p>
          <p class="text-gray-400">Телефон: +380 (67) 123-45-67</p>
        </div>
      </div>
      <div class="border-t border-gray-700 mt-8 pt-8 text-center text-gray-400">
        <p>&copy; 2025 Romi Pet Comfort. Всі права захищені.</p>
      </div>
    </div>
  </footer>
 </div>
</template>

<script setup lang="ts">
import { ref, onMounted, onUnmounted } from 'vue'
import { useRouter } from 'vue-router'
import { ShoppingCart, User, ChevronDown } from 'lucide-vue-next'
import { useAuthStore } from '@/stores/auth'
import { useCartStore } from '@/stores/cart'

const router = useRouter()
const authStore = useAuthStore()
const cartStore = useCartStore()
const showUserMenu = ref(false)
const userMenuRef = ref<HTMLElement | null>(null)

const handleLogout = () => {
 authStore.logout()
 showUserMenu.value = false
 router.push('/')
}

const handleClickOutside = (event: MouseEvent) => {
 if (userMenuRef.value && !userMenuRef.value.contains(event.target as Node)) {
  showUserMenu.value = false
 }
}

onMounted(() => {
 document.addEventListener('click', handleClickOutside)
})
```

```
onUnmounted(() => {
  document.removeEventListener(‘click’, handleClickOutside)
})
</script>
```

ДОДАТОК Б

Лістинг коду для адміністративного макету платформи

```
<template>
  <div class="bg-gray-100">
    <div class="flex min-h-screen">
      <aside class="flex w-64 flex-col bg-gray-800 text-white">
        <div class="p-4">
          <router-link to="/" class="flex items-center space-x-2 mb-8">
            <span class="text-2xl font-bold">🐾 Romi Admin</span>
          </router-link>
          <nav class="space-y-2">
            <router-link to="/admin" exact-active-class="bg-primary-600" class="flex items-center space-x-2 px-4 py-2 rounded hover:bg-gray-700">
              <LayoutDashboard :size="20" />
              <span>Дашборд</span>
            </router-link>
            <router-link to="/admin/products" active-class="bg-primary-600" class="flex items-center space-x-2 px-4 py-2 rounded hover:bg-gray-700">
              <Package :size="20" />
              <span>Товари</span>
            </router-link>
            <router-link to="/admin/categories" active-class="bg-primary-600" class="flex items-center space-x-2 px-4 py-2 rounded hover:bg-gray-700">
              <FolderTree :size="20" />
              <span>Категорії</span>
            </router-link>
            <router-link to="/admin/orders" active-class="bg-primary-600" class="flex items-center space-x-2 px-4 py-2 rounded hover:bg-gray-700">
              <ShoppingBag :size="20" />
              <span>Замовлення</span>
            </router-link>
            <router-link to="/admin/users" active-class="bg-primary-600" class="flex items-center space-x-2 px-4 py-2 rounded hover:bg-gray-700">
              <Users :size="20" />
              <span>Користувачі</span>
            </router-link>
          </nav>
        </div>
        <div class="mt-auto w-full p-4 border-t border-gray-700">
          <button @click="handleBackToSite" class="flex w-full items-center space-x-2 rounded px-4 py-2 hover:bg-gray-700">
            <ArrowLeft :size="20" />
```

```
          <span>На сайт</span>
        </button>
      </div>
    </aside>

    <main class="flex-1 p-8">
      <router-view />
    </main>
  </div>
 </div>
</template>

<script setup lang="ts">
import { useRouter } from 'vue-router'
import { LayoutDashboard, Package, FolderTree, ShoppingBag, Users, ArrowLeft } from 'lucide-vue-next'

const router = useRouter()

const handleBackToSite = () => {
 router.push('/')
}
</script>
```

ДОДАТОК В

API Examples & Testing scripts

1. Скрипт створення тестових даних

```python
# scripts/create_test_data.py
import asyncio
import aiohttp
import json

API_BASE = "http://localhost:8000"

async def create_test_data():
    async with aiohttp.ClientSession() as session:
        # 1. Реєстрація адміна
        admin_data = {
            "email": "admin@romipetcomfort.com",
            "password": "admin123",
            "full_name": "Адміністратор"
        }

        async with session.post(f"{API_BASE}/auth/register", json=admin_data) as resp:
            admin = await resp.json()
            print(f"Адміна створено: {admin['email']}")

        # 2. Авторизація адміна
        login_data = {
            "username": "admin@romipetcomfort.com",
            "password": "admin123"
        }

        async with session.post(f"{API_BASE}/auth/login", data=login_data) as resp:
            tokens = await resp.json()
            token = tokens['access_token']

        headers = {"Authorization": f"Bearer {token}"}

        # 3. Створення категорії
        categories = [
            {"name": "Іграшки", "description": "Радість вашому хвостику"},
            {"name": "Ігрові комплекси", "description": "Простір для ігор"},
            {"name": "Лежаки", "description": "За для комфортного відпочинку вашого
```

```
улюбленця"}
    ]

    created_categories = []
    for cat_data in categories:
        async with session.post(f"{API_BASE}/categories/",
                                json=cat_data, headers=headers) as resp:
            category = await resp.json()
            created_categories.append(category)
            print(f"Категорія створена: {category['name']}")

    # 4. Створення підкатегорії
    subcategories = [
        {"name": "Маленькі", "description": "Для кошенят", "category_id":
created_categories[0]['id']},
        {"name": "Великі", "description": "Для великих кішок", "category_id":
created_categories[0]['id']},
        {"name": "Міні", "description": "Для маленьких порід", "category_id":
created_categories[1]['id']},
        {"name": "Максі", "description": "Для великих порід", "category_id":
created_categories[1]['id']}
    ]

    for subcat_data in subcategories:
        async with session.post(f"{API_BASE}/categories/subcategories/",
                                json=subcat_data, headers=headers) as resp:
            subcategory = await resp.json()
            print(f"Підкатегорія створена: {subcategory['name']}")

    # 5. Створення товарів
    products = [
        {
            "name": "М’який диван 'Затишок'",
            "description": "Неймовірно м’який диван",
            "base_price": 2500.0,
            "category_id": created_categories[0]['id'],
            "material": "Велюр, ортопедична піна",
            "care_instructions": "Хімчистка або прання при 30°C",
            "sizes": [
                {"name": "S", "price_modifier": 0},
                {"name": "M", "price_modifier": 500},
                {"name": "L", "price_modifier": 1000}
            ],
            "colors": [
                {"name": "Сірий", "price_modifier": 0},
                {"name": "Коричневий", "price_modifier": 200},
```

```
                {"name": "Бежевий", "price_modifier": 150}
            ]
        },
        {
            "name": "Диван 'Аристократ'",
            "description": "Елегантний диван зі штучної шкіри",
            "base_price": 4000.0,
            "category_id": created_categories[1]['id'],
            "material": "Екокожа, холлофайбер",
            "care_instructions": "Протирання вологою тканиною",
            "sizes": [
                {"name": "M", "price_modifier": 0},
                {"name": "L", "price_modifier": 800},
                {"name": "XL", "price_modifier": 1500}
            ],
            "colors": [
                {"name": "Чорний", "price_modifier": 0},
                {"name": "Коричневий", "price_modifier": 300}
            ]
        }
    ]

    for prod_data in products:
        async with session.post(f"{API_BASE}/products/",
                                json=prod_data, headers=headers) as resp:
            product = await resp.json()
            print(f"Товар створено: {product['name']}")

if __name__ == "__main__":
    asyncio.run(create_test_data())
```

2. Скрипт для тестування навантаження

```python
# scripts/load_test.py
import asyncio
import aiohttp
import time
import random

API_BASE = "http://localhost:8000"
CONCURRENT_REQUESTS = 10
TOTAL_REQUESTS = 100

async def test_endpoint(session, endpoint, method="GET", data=None):
```

```
    """Тестування одного ендпоінта"""
    try:
        start_time = time.time()
        if method == "GET":
            async with session.get(f"{API_BASE}{endpoint}") as resp:
                status = resp.status
                response_time = time.time() - start_time
                return {"status": status, "time": response_time, "endpoint": endpoint}
        elif method == "POST":
            async with session.post(f"{API_BASE}{endpoint}", json=data) as resp:
                status = resp.status
                response_time = time.time() - start_time
                return {"status": status, "time": response_time, "endpoint": endpoint}
    except Exception as e:
        return {"status": 0, "time": 0, "endpoint": endpoint, "error": str(e)}

async def load_test():
    """Запуск навантажувального тестування"""
    endpoints = [
        "/",
        "/health",
        "/products/",
        "/categories/",
        "/products/1"
    ]

    results = []

    async with aiohttp.ClientSession() as session:
        tasks = []

        for _ in range(TOTAL_REQUESTS):
            endpoint = random.choice(endpoints)
            task = test_endpoint(session, endpoint)
            tasks.append(task)

        # Виконання запитів з обмеженням паралельності
        semaphore = asyncio.Semaphore(CONCURRENT_REQUESTS)

        async def bounded_task(task):
            async with semaphore:
                return await task

        bounded_tasks = [bounded_task(task) for task in tasks]
        results = await asyncio.gather(*bounded_tasks)
```

```
    # Аналіз результатів
    successful = [r for r in results if r['status'] == 200]
    failed = [r for r in results if r['status'] != 200]

    avg_response_time = sum(r['time'] for r in successful) / len(successful) if successful else 0

    print(f"Результати навантажувального тестування:")
    print(f"Всього запитів: {TOTAL_REQUESTS}")
    print(f"Успішних: {len(successful)}")
    print(f"Неуспішних: {len(failed)}")
    print(f"Середній час відповіді: {avg_response_time:.3f}s")
    print(f"Паралельність: {CONCURRENT_REQUESTS}")

if __name__ == "__main__":
    asyncio.run(load_test())
```

3. Повний приклад роботи з API

```python
# scripts/api_example.py
import requests
import json

class RomiPetComfortAPI:
    def __init__(self, base_url="http://localhost:8000"):
        self.base_url = base_url
        self.session = requests.Session()
        self.token = None

    def register(self, email, password, full_name, gender=None, birth_date=None):
        """Реєстрація користувача"""
        data = {
            "email": email,
            "password": password,
            "full_name": full_name
        }
        if gender:
            data["gender"] = gender
        if birth_date:
            data["birth_date"] = birth_date

        response = self.session.post(f"{self.base_url}/auth/register", json=data)
        return response.json()

    def login(self, email, password):
```

```
        """Авторизація"""
        data = {
            "username": email,
            "password": password
        }
        response = self.session.post(f"{self.base_url}/auth/login", data=data)
        if response.status_code == 200:
            tokens = response.json()
            self.token = tokens['access_token']
            self.session.headers.update({"Authorization": f"Bearer {self.token}"})
        return response.json()

    def get_products(self, page=1, size=20, category_id=None, search=None,
sort_by="created_at"):
        """Отримання переліку товарів"""
        params = {
            "page": page,
            "size": size,
            "sort_by": sort_by
        }
        if category_id:
            params["category_id"] = category_id
        if search:
            params["search"] = search

        response = self.session.get(f"{self.base_url}/products/", params=params)
        return response.json()

    def get_product(self, product_id):
        """Отримання товару по ID"""
        response = self.session.get(f"{self.base_url}/products/{product_id}")
        return response.json()

    def create_order(self, customer_name, customer_email, customer_phone, items,
comment=None):
        """Створення замовлення"""
        data = {
            "customer_name": customer_name,
            "customer_email": customer_email,
            "customer_phone": customer_phone,
            "items": items
        }
        if comment:
            data["comment"] = comment

        response = self.session.post(f"{self.base_url}/orders/", json=data)
```

```
    return response.json()

def get_my_orders(self, page=1, size=20):
    """Отримання своїх замовлень"""
    params = {"page": page, "size": size}
    response = self.session.get(f"{self.base_url}/orders/me", params=params)
    return response.json()

def cancel_order(self, order_id):
    """Скасування замовлення"""
    response = self.session.patch(f"{self.base_url}/orders/{order_id}/cancel")
    return response.json()

def confirm_email(self, token):
    """Підтвердження email"""
    data = {"token": token}
    response = self.session.post(f"{self.base_url}/auth/confirm-email", json=data)
    return response.json()

def resend_confirmation(self, email):
    """Повторне надсилання листа з підтвердженням"""
    data = {"email": email}
    response = self.session.post(f"{self.base_url}/auth/resend-confirmation", json=data)
    return response.json()

def forgot_password(self, email):
    """Запит скидання пароля"""
    data = {"email": email}
    response = self.session.post(f"{self.base_url}/auth/forgot-password", json=data)
    return response.json()

def reset_password(self, token, new_password):
    """Скидання пароля"""
    data = {
        "token": token,
        "new_password": new_password
    }
    response = self.session.post(f"{self.base_url}/auth/reset-password", json=data)
    return response.json()

def change_password(self, current_password, new_password):
    """Зміна пароля (потребує авторизації)"""
    data = {
        "current_password": current_password,
        "new_password": new_password
    }
```

```
        response = self.session.put(f"{self.base_url}/auth/change-password", json=data)
        return response.json()

# Приклад використання
def main():
    api = RomiPetComfortAPI()

    # 1. Реєстрація
    print("1. Реєстрація користувача...")
    user = api.register(
        email="test@example.com",
        password="password123",
        full_name="Тест Тестів",
        gender="male"
    )
    print(f"Користувача створено: {user['email']}")

    # 2. Авторизація
    print("\n2. Авторизація...")
    tokens = api.login("test@example.com", "password123")
    print(f"Токен отримано: {tokens['access_token'][:20]}...")

    # 3. Отримання товарів
    print("\n3. Отримання товарів...")
    products = api.get_products(page=1, size=5)
    print(f"Знайдено товарів: {products['total']}")
    for product in products['products']:
        print(f"  - {product['name']}: {product['base_price']} руб.")

    # 4. Створення замовлення
    if products['products']:
        print("\n4. Створення замовлення...")
        product = products['products'][0]

        order_items = [
            {
                "product_id": product['id'],
                "quantity": 1,
                "selected_size": "M" if product['sizes'] else None,
                "selected_color": "Сірий" if product['colors'] else None
            }
        ]

        order = api.create_order(
            customer_name="Тест Тестів",
            customer_email="test@example.com",
```

```
            customer_phone="+3-800-123-45-67",
            items=order_items,
            comment="Тестове замовлення"
        )
        print(f"Замовлення створено: #{order['id']} на суму {order['total_amount']} руб.")

        # 5. Отримання своїх замовлень
        print("\n5. Мої замовлення...")
        my_orders = api.get_my_orders()
        for order in my_orders['orders']:
            print(f"  Замовленя #{order['id']}: {order['status']} - {order['total_amount']} грн.")

    # Демонстрація email функцій
    print("\n=== Email функції ===")

    # 6. Повторне надсилання підтвердження email
    print("\n6. Повторне надсилання підтвердження...")
    try:
        resend_result = api.resend_confirmation("test@example.com")
        print(f"Результат: {resend_result.get('message', 'Лист відправлено')}")
    except Exception as e:
        print(f"Помилка відправки: {e}")

    # 7. Запит скидання пароля
    print("\n7. Запит скидання пароля...")
    try:
        forgot_result = api.forgot_password("test@example.com")
        print(f"Результат: {forgot_result.get('message', 'Інструкції надіслані на email')}")
    except Exception as e:
        print(f"Помилка: {e}")

    # 8. Зміна пароля (для авторизованого користувача)
    print("\n8. Зміна пароля...")
    try:
        change_result = api.change_password("password123", "newpassword123")
        print(f"Результат: {change_result.get('message', 'Пароль змінено')}")
    except Exception as e:
        print(f"Помилка: {e}")

    print("\n=== Демонстрація завершена ===")
    print("Примітка: Для підтвердження email та скидання пароля потрібні токени з листів")

if __name__ == "__main__":
    main()
```

4. Тестування Email функції

```python
# scripts/email_test.py
import requests
import time

class EmailTestClient:
    def __init__(self, base_url="http://localhost:8000"):
        self.base_url = base_url
        self.session = requests.Session()

    def test_email_confirmation_flow(self, email="test@example.com", password="password123"):
        """Повний тест процесу підтвердження email"""
        print("Тестування процесу підтвердження email")
        print("=" * 50)

        # 1. Реєстрація користувача
        print("1. Реєстрація користувача...")
        register_data = {
            "email": email,
            "password": password,
            "full_name": "Test User"
        }

        try:
            response = self.session.post(f"{self.base_url}/auth/register", json=register_data)
            if response.status_code == 201:
                print(f"Користувач зареєстрований: {email}")
                user = response.json()
                print(f"Email підтверджений: {user.get('email_confirmed', False)}")
            else:
                print(f"Помилка реєстрації: {response.json()}")
                return
        except Exception as e:
            print(f"Помилка: {e}")
            return

        # 2. Спроба входу без підтвердження email
        print("\n2. Спроба входу без підтвердження email...")
        login_data = {
            "username": email,
            "password": password
        }
```

```
    try:
        response = self.session.post(f"{self.base_url}/auth/login", data=login_data)
        if response.status_code == 200:
            print("Вхід дозволений (email не потребує підтвердження)")
        else:
            print(f"Вхід заблоковано: {response.json()}")
    except Exception as e:
        print(f"Помилка: {e}")

    # 3. Повторне надсилання підтвердження
    print("\n3. Повторне надсилання листа з підтвердженням...")
    try:
        response = self.session.post(f"{self.base_url}/auth/resend-confirmation",
                                     json={"email": email})
        if response.status_code == 200:
            print("Лист надіслано повторно")
            result = response.json()
            print(f"Повідомлення: {result.get('message', 'Немає повідомлення')}")
        else:
            print(f"Помилка відправки: {response.json()}")
    except Exception as e:
        print(f"Помилка: {e}")

    # 4. Симуляція підтвердження email
    print("\n4. Симуляція підтвердження email...")
    print("   ℹ Насправді потрібен токен з листа")
    print("   ℹ Приклад: POST /auth/confirm-email {'token': 'email_token_here'}")

    return True

def test_password_reset_flow(self, email="test@example.com"):
    """Тест процесу скидання пароля"""
    print("\n Тестування процесу скидання пароля")
    print("=" * 50)

    # 1. Запит скидання пароля
    print("1. Запит скидання пароля...")
    try:
        response = self.session.post(f"{self.base_url}/auth/forgot-password",
                                     json={"email": email})
        if response.status_code == 200:
            print("Запит відправлено")
            result = response.json()
                print(f"Повідомлення: {result.get('message', 'Інструкції надіслані')}")
        else:
            print(f"Помилка: {response.json()}")
```

```
        except Exception as e:
            print(f"Помилка: {e}")

        # 2. Симуляція скидання пароля
        print("\n2. Симуляція скидання пароля...")
        print("   ℹ Насправді потрібен токен з листа")
        print("   ℹ Приклад: POST /auth/reset-password {'token': 'reset_token', 'new_password': 'new123'}")

        return True

    def test_password_change(self, email="test@example.com", password="password123"):
        """Тест зміни пароля авторизованим користувачем"""
        print("\n Тестування зміни пароля")
        print("=" * 50)

        # 1. Авторизація
        print("1. Авторизація...")
        login_data = {
            "username": email,
            "password": password
        }

        try:
            response = self.session.post(f"{self.base_url}/auth/login", data=login_data)
            if response.status_code == 200:
                tokens = response.json()
                token = tokens['access_token']
                self.session.headers.update({"Authorization": f"Bearer {token}"})
                print("Авторизація успішна")
            else:
                print(f"Помилка авторизації: {response.json()}")
                return
        except Exception as e:
            print(f"Помилка: {e}")
            return

        # 2. Зміна пароля
        print("\n2. Зміна пароля...")
        change_data = {
            "current_password": password,
            "new_password": "newpassword123"
        }

        try:
            response = self.session.put(f"{self.base_url}/auth/change-password",
```

```
                                json=change_data)
            if response.status_code == 200:
                print("Пароль змінено")
                result = response.json()
                print(f"Повідомлення: {result.get('message', 'Пароль успішно змінено')}")
            else:
                print(f" Помилка зміни: {response.json()}")
        except Exception as e:
            print(f"Помилка: {e}")

        return True

def main():
    """Запуск всіх email тестів"""
    client = EmailTestClient()

    # Тестуємо з унікальним email
    test_email = f"test_{int(time.time())}@example.com"

    print(f"Тестування з email: {test_email}")
    print("\n")

    # Запуск тестів
    client.test_email_confirmation_flow(test_email)
    client.test_password_reset_flow(test_email)
    client.test_password_change(test_email)

    print("\n" + "=" * 60)
    print("Всі тести завершено")
    print("\nПримітка:")
    print("- Для повного тестування потрібне налаштування SMTP")
    print("- Токени email та скидання пароля приходять у листах")
    print("- У продакшені увімкніть перевірку підтвердженого email")

if __name__ == "__main__":
    main()
```

5. Утиліта для роботи с Email токенами

```python
# scripts/email_tokens.py
import secrets
import string
from datetime import datetime, timedelta
from sqlalchemy.orm import Session
```

```
from app.db.session import SessionLocal
from app.crud.user import get_user_by_email
from app.core.security import create_access_token

def generate_email_token(length=32):
    """Генерація токена для email"""
    alphabet = string.ascii_letters + string.digits
    return ''.join(secrets.choice(alphabet) for _ in range(length))

def simulate_email_confirmation(email: str):
    """Симуляція підтвердження email (для розробки)"""
    db = SessionLocal()
    try:
        user = get_user_by_email(db, email)
        if not user:
            print(f"Користувача {email} не знайдено")
            return

        # Генеруємо токен
        token = generate_email_token()

        # Зберігаємо токен
        user.email_confirmation_token = token
        user.email_confirmation_sent_at = datetime.utcnow()
        db.commit()

        print(f"Email токен для {email}: {token}")
        print(f"Посилання для підтвердження:")
        print(f"http://localhost:3000/confirm-email?token={token}")

        return token

    except Exception as e:
        print(f"Помилка: {e}")
        db.rollback()
    finally:
        db.close()

def simulate_password_reset(email: str):
    """Симуляція скидання пароля"""
    db = SessionLocal()
    try:
        user = get_user_by_email(db, email)
        if not user:
            print(f"Користувача {email} не знайдено")
            return
```

```
        # Генеруємо токен
        token = generate_email_token()

        #Зберігаємо токен
        user.password_reset_token = token
        user.password_reset_sent_at = datetime.utcnow()
        db.commit()

        print(f"Токен скидання пароля для {email}: {token}")
        print(f"Посилання для скидання:")
        print(f"http://localhost:3000/reset-password?token={token}")

        return token

    except Exception as e:
        print(f"Помилка: {e}")
        db.rollback()
    finally:
        db.close()

def confirm_email_manually(email: str):
    """Ручне підтвердження email"""
    db = SessionLocal()
    try:
        user = get_user_by_email(db, email)
        if not user:
            print(f"Користувача {email} не знайдено")
            return

        user.email_confirmed = True
        user.email_confirmation_token = None
        user.email_confirmation_sent_at = None
        db.commit()

        print(f"Email {email}
    except Exception as e:
        print(f"Помилка: {e}")
        db.rollback()
    finally:
        db.close()

def main():
    """Інтерактивна утиліта для роботи з email"""
    print("Утиліта для роботи с Email токенами")
    print("=" * 40)
```

```
    while True:
        print("\nВиберіть дію:")
        print("1. Згенерувати токен підтвердження email")
        print("2. Згенерувати токен скидання пароля")
        print("3. Підтвердити email вручну")
        print("4. Вихід")

        choice = input("\nВаш вибір (1-4): ").strip()

        if choice == "1":
            email = input("Ввести email: ").strip()
            simulate_email_confirmation(email)

        elif choice == "2":
            email = input("Ввести email: ").strip()
            simulate_password_reset(email)

        elif choice == "3":
            email = input("Ввести email: ").strip()
            confirm_email_manually(email)

        elif choice == "4":
            break

        else:
            print("Невірно введено")

if __name__ == "__main__":
    main()
```