Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп’ютерних наук

Допущено до захисту
Завідувач кафедри комп’ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
«01» червня 2026 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «бакалавр»
спеціальність 122 Комп’ютерні науки
освітня програма 122.00.01 Інформатика

Тема: МОДУЛЬНА СИСТЕМА ВЕБ-СЕРВІСУ З МОЖЛИВІСТЮ РОЗШИРЕННЯ ПІД БІЗНЕС-МОДЕЛІ

Виконав
студент групи Інб-1-22-4.0д
Баранник Богдан Костянтинович

_____
(підпис)
Науковий керівник
кандидат педагогічних наук, доцент
Вікторія Вембер

_____
(підпис)

Київ - 2026

Київський столичний університет імені Бориса Грінченка
Факультет інформаційних технологій та математики
Кафедра комп'ютерних наук

«Затверджую»
Завідувач кафедри комп'ютерних наук
доктор технічних наук, професор
Андрій БОНДАРЧУК
«31» жовтня 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА
«МОДУЛЬНА СИСТЕМА ВЕБ-СЕРВІСУ З МОЖЛИВІСТЮ РОЗШИРЕННЯ ПІД БІЗНЕС-МОДЕЛІ»

Виконавець - студент групи ІНб-1-22-4.0д,
спеціальності 122 Комп'ютерні наук,
освітньої програми 122.00.01 Інформатика
Баранник Богдан Костянтинович

1. Вихідні дані: Нормативно-правові акти України у сфері електронної комерції, захисту персональних даних та інформаційної безпеки, наукові роботи та публікації за напрямом розробки веб-застосунків, програмної інженерії та інформаційних систем, сучасні технології веб-розробки, документація програмних платформ і бібліотек.

2. Основні завдання: Проаналізувати предметну область та існуючі програмні рішення у сфері маркетплейсів для визначення функціональних вимог до системи та обґрунтування її модульної архітектури. Спроєктувати архітектуру вебсервісу, структуру бази даних та моделі доступу користувачів із обґрунтуванням вибору технологічного стеку та організації монорепозиторію. Розробити програмну систему (MVP), що включає реалізацію серверної та клієнтської частин, функціоналу управління оголошеннями, адміністративної панелі та інтеграцію платіжної системи. Забезпечити безпеку даних, провести тестування функціональних можливостей розробленого сервісу та надати економічне обґрунтування доцільності розробки.

3. Пояснювальна записка: Обсяг - до 50 сторінок формату А4 комп'ютерного набору з дотриманням вимог стандарту і методичних рекомендацій кафедри.

4. Графічні матеріали: Презентація результатів розробки програмної системи, діаграми архітектури програмного забезпечення, UML-діаграми взаємодії компонентів системи, схема структури бази даних та інтерфейсів користувача.

5. Додатки: не передбачено.

6. Строк подання роботи на кафедру: « » травня 2026 р.

Науковий керівник
к.п.н., доцент
_____ Вембер В.П.
«31» жовтня 2025 р.

Виконавець:
_____ Баранник Б.К.
«31» жовтня 2025 р.

Календарний план

№	Етап виконання роботи	Термін виконання	Примітка
1	Затвердження теми кваліфікаційної роботи бакалавра	31.10.25	виконано
2	Аналіз проблеми, вивчення аналогів, формування цілей і завдань	01.11.25 - 11.01.26	виконано
3	Аналіз предметної області веб-маркетплейсів, існуючих програмних рішень, бізнес-процесів електронної комерції та обґрунтування вибору модульної архітектури веб-сервісу	26.01.26 - 15.03.26	виконано
4	Проєктування загальної архітектури програмної системи, структури бази даних, REST API, моделей взаємодії користувачів, ролей доступу та механізмів авторизації і безпеки	16.03.26 - 31.03.26	виконано
5	Розробка клієнтської та серверної частин веб-сервісу, реалізація функціоналу управління оголошеннями, замовленнями, адміністративної панелі та інтеграція платіжної системи	01.04.26 - 15.04.26	виконано
6	Тестування основних функціональних сценаріїв роботи системи, перевірка механізмів безпеки, контролю доступу, роботи адміністративної панелі та коректності взаємодії компонентів веб-сервісу	16.04.26 - 30.04.26	виконано
7	Впровадження розробленого програмного продукту, фінальне налаштування взаємодії з базою даних, хмарною інфраструктурою та підготовка системи до дослідної експлуатації	01.05.26 - 15.05.26	виконано
8	Підготовка пояснювальної записки, графічних матеріалів, діаграм, додатків та оформлення результатів кваліфікаційної роботи	25.05.26 - 12.06.26	виконано
9	Захист кваліфікаційної роботи	17.06.26	виконано

Студент Баранник Б.К.

Керівник роботи к.п.н. Вембер В.П.

РЕФЕРАТ

Кваліфікаційна робота: 69 с., 5 рис., 35 посилання.

Актуальність: стрімкий розвиток електронної комерції та цифрових сервісів зумовлює необхідність створення гнучких, масштабованих і безпечних веб-платформ для взаємодії користувачів. Сучасні маркетплейси повинні забезпечувати зручний механізм розміщення оголошень, обробки замовлень, управління користувачами та інтеграції платіжних систем. Використання модульної архітектури дозволяє підвищити розширюваність системи, спростити її супровід і забезпечити ефективне масштабування при зростанні кількості користувачів та функціональності.

Об'єкт - це веб-сервіс електронного маркетплейсу як інформаційна система для розміщення оголошень та управління замовленнями.

Предмет - процес проєктування та реалізації модульної програмної системи маркетплейсу з використанням сучасних веб-технологій, засобів автентифікації, управління доступом та інтеграції платіжних сервісів.

Мета - розробити MVP (Minimum Viable Product) версію веб-сервісу маркетплейсу на основі модульної архітектури, яка забезпечить можливість створення та управління оголошеннями, обробки замовлень, адміністрування системи та інтеграції платіжної системи з перспективою подальшого масштабування та розширення функціоналу.

Основні результати:

1. Проведено аналіз існуючих систем електронних маркетплейсів та визначено їх основні функціональні можливості.
2. Обґрунтовано використання модульної архітектури для забезпечення гнучкості та масштабованості системи.
3. Розроблено MVP версію веб-сервісу маркетплейсу з використанням сучасних технологій веб-розробки.

4. Реалізовано механізми автентифікації користувачів, управління ролями доступу, систему створення оголошень та замовлень, адміністративну панель модерації та інтеграцію платіжної системи.

Практичне значення дослідження полягає у створенні прототипу веб-платформи маркетплейсу, який може бути використаний як основа для подальшої розробки комерційної системи електронної торгівлі або інформаційного сервісу.

Ключові слова: маркетплейс, веб-сервіс, модульна архітектура, REST API, Supabase, авторизація, адміністрування, платіжна система, інформаційна система.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Скорочення	**Розшифрування**
API	Application Programming Interface - прикладний програмний інтерфейс
BaaS	Backend as a Service - модель надання backend-інфраструктури як сервісу
CRUD	Create, Read, Update, Delete - базові операції роботи з даними
CSS	Cascading Style Sheets - каскадні таблиці стилів
ER	Entity-Relationship - модель сутність-зв’язок
HTTPS	HyperText Transfer Protocol Secure - захищений протокол передачі даних
JWT	JSON Web Token - токен для автентифікації та авторизації
MVP	Minimum Viable Product - мінімально життєздатний продукт
ORM	Object-Relational Mapping - технологія взаємодії об’єктів із базою даних
PostgreSQL	реляційна система керування базами даних
React	JavaScript-бібліотека для створення користувацьких інтерфейсів
REST API	Representational State Transfer API - архітектурний стиль взаємодії веб-сервісів
RLS	Row Level Security - механізм контролю доступу на рівні рядків таблиці
Supabase	Backend as a Service платформа на базі PostgreSQL
Vite	інструмент збірки frontend-застосунків

Зміст

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується активним впровадженням веб-сервісів, орієнтованих на електронну комерцію, автоматизацію бізнес-процесів та надання цифрових послуг широкому колу користувачів. Одним із найбільш поширених класів таких систем є веб-маркетплейси, які забезпечують взаємодію між продавцями та покупцями, управління товарами або послугами, обробку замовлень та фінансові розрахунки.

Актуальною проблемою при розробці веб-маркетплейсів є забезпечення масштабованості, розширюваності та безпеки програмних систем. Багато існуючих рішень створюються як вузькоспеціалізовані продукти, що ускладнює їх подальший розвиток, адаптацію під нові бізнес-моделі та інтеграцію додаткового функціоналу. Зростання кількості користувачів, ролей, типів контенту та бізнес-процесів вимагає застосування модульних архітектурних підходів, а також реалізації надійних механізмів контролю доступу до даних.

Особливу роль у сучасних веб-системах відіграє безпека даних, зокрема розмежування доступу до інформації між різними категоріями користувачів. Традиційні підходи, що покладаються виключно на перевірки на рівні серверної логіки, не завжди забезпечують достатній рівень захисту. У зв’язку з цим перспективним є використання механізмів безпеки на рівні бази даних, таких як Row Level Security (RLS), які дозволяють обмежувати доступ до записів залежно від ролі та контексту користувача.

У даній кваліфікаційній роботі розглядається розробка модульної системи веб-сервісу з можливістю розширення під різні бізнес-моделі на прикладі маркетплейсу з адміністративною панеллю. Система орієнтована на повний цикл взаємодії користувачів: від реєстрації та створення оголошень до оформлення замовлень, модерації контенту та інтеграції платіжних механізмів. При цьому особлива увага приділяється архітектурі проєкту, безпеці даних та можливості подальшого масштабування.

Метою кваліфікаційної роботи є проєктування та реалізація безпечної модульної архітектури веб-сервісу типу маркетплейс, яка забезпечує масштабованість, розширюваність та безпеку даних на рівні бази даних.

Для досягнення поставленої мети у роботі необхідно вирішити такі завдання:

1. Проаналізувати предметну область і існуючі рішення у сфері маркетплейсів для визначення функціональних вимог та обґрунтування архітектури системи.

2. Спроєктувати безпечну архітектуру вебсервісу, структуру бази даних і моделі доступу користувачів із обґрунтуванням вибору технологічного стеку.

3. Розробити MVP програмної системи, що включає серверну та клієнтську частини, функціонал управління оголошеннями, адміністративну панель та інтеграцію платіжної системи.

4. Забезпечити захист даних, провести тестування системи та надати економічне обґрунтування доцільності розробки.

Об’єктом дослідження є процеси розробки та експлуатації веб-орієнтованих інформаційних систем електронної комерції.

Предметом дослідження є методи та засоби проєктування модульної архітектури веб-сервісів, механізми розмежування доступу до даних та реалізація маркетплейсу з адміністративною панеллю.

У роботі використано такі методи дослідження: аналіз предметної області та існуючих програмних рішень; системний аналіз архітектур веб-систем; проєктування баз даних та моделей доступу; об’єктно-орієнтований та компонентний підхід до розробки програмного забезпечення; практичне програмування з використанням сучасних веб-технологій; тестування та аналіз результатів роботи програмної системи.

Практичне значення отриманих результатів полягає у можливості використання розробленої модульної архітектури як основи для створення та розвитку веб-маркетплейсів або інших бізнес-орієнтованих веб-сервісів.

Реалізований підхід до безпеки на рівні бази даних може бути застосований у комерційних проєктах для підвищення надійності та захисту даних користувачів.

Результати роботи апробовано шляхом публікації тез доповіді на конференції Інформаційні технології - 2026: зб. тез XII Всеукраїнської науково-практичної конференції молодих учених, 14 трав. 2025 р., м. Київ / Київський столичний університет імені Бориса Грінченка. с. 178-179 [31].

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз веб-маркетплейсів як класу інформаційних систем

Веб-маркетплейси є одним із найпоширеніших різновидів веб-орієнтованих інформаційних систем у сфері електронної комерції. Основною їх особливістю є забезпечення централізованої платформи для взаємодії між різними групами користувачів, зокрема продавцями, покупцями та адміністраторами системи. На відміну від класичних інтернет-магазинів, маркетплейси не обмежуються одним постачальником товарів або послуг, а підтримують багатосторонню модель взаємодії.

З функціональної точки зору веб-маркетплейс поєднує у собі такі ключові можливості:

- реєстрацію та автентифікацію користувачів;
- управління профілями користувачів;
- створення, редагування та публікацію оголошень;
- класифікацію товарів або послуг за категоріями;
- пошук та перегляд доступного контенту;
- оформлення замовлень та взаємодію між продавцем і покупцем;
- контроль і модерацію контенту з боку адміністратора.

Зазначені можливості формують складну програмну систему, яка повинна забезпечувати надійність, безпеку та зручність використання при значній кількості користувачів і постійних змінах у бізнес-вимогах.

Залежно від призначення, маркетплейси можуть бути орієнтовані на продаж фізичних товарів, цифрових продуктів, послуг або поєднувати декілька типів пропозицій. Такі платформи активно використовуються у сферах електронної торгівлі, оренди, фрилансу, доставки, логістики та сервісних послуг. Універсальність цього підходу обумовлює необхідність створення гнучких та розширюваних архітектур.

З погляду архітектури, веб-маркетплейс зазвичай складається з трьох основних компонентів:

- клієнтської частини (frontend), що забезпечує взаємодію користувача з системою через веб-інтерфейс;
- серверної частини (backend), яка реалізує бізнес-логіку, обробку запитів та інтеграцію із зовнішніми сервісами;
- системи зберігання даних, що відповідає за збереження інформації про користувачів, оголошення, замовлення та інші сутності.

Важливою характеристикою маркетплейсів є наявність різних ролей користувачів, кожна з яких має власний набір прав доступу. Типовими ролями є звичайний користувач, який може виступати покупцем або продавцем, та адміністратор, що відповідає за управління платформою, модерацію контенту і забезпечення дотримання правил користування сервісом. Така рольова модель значно ускладнює питання безпеки та контролю доступу до даних.

Сучасні веб-маркетплейси повинні відповідати низці нефункціональних вимог, серед яких ключовими є масштабованість, продуктивність і безпека. Масштабованість забезпечує можливість розвитку системи без необхідності її повної переробки, зокрема шляхом додавання нових бізнес-моделей, ролей або функціональних модулів. Продуктивність визначає здатність системи обробляти велику кількість одночасних запитів без зниження якості обслуговування. Безпека, у свою чергу, гарантує захист персональних даних користувачів і фінансової інформації.

Таким чином, веб-маркетплейс можна розглядати як складну багатокомпонентну інформаційну систему, розробка якої потребує комплексного підходу до архітектури, проєктування бази даних, реалізації механізмів безпеки та організації бізнес-процесів. Саме ці аспекти зумовлюють актуальність дослідження та розробки модульної системи веб-сервісу, здатної адаптуватися до змін бізнес-вимог та забезпечувати високий рівень захисту даних.

1.2 Бізнес-процеси маркетплейсів та вимоги до їх автоматизації

Функціонування веб-маркетплейсу базується на сукупності взаємопов’язаних бізнес-процесів, які забезпечують повний цикл взаємодії між користувачами платформи. Бізнес-процеси визначають логіку роботи системи, послідовність дій користувачів та правила обробки даних. Від коректності їх реалізації залежить ефективність роботи маркетплейсу, зручність користування та можливість подальшого розвитку системи.

До основних бізнес-процесів маркетплейсу належать:

- реєстрація та автентифікація користувачів;
- створення та управління оголошеннями;
- публікація і перегляд товарів або послуг;
- оформлення та обробка замовлень;
- взаємодія між покупцем і продавцем;
- модерація контенту та управління користувачами з боку адміністратора;
- обробка платежів і фіксація фінансових операцій.

Кожен із зазначених бізнес-процесів повинен бути автоматизований з урахуванням ролей користувачів та їх прав доступу. Наприклад, процес створення оголошення доступний лише авторизованим користувачам, тоді як зміна статусу оголошення або блокування користувача належить виключно до повноважень адміністратора.

Одним із ключових бізнес-процесів маркетплейсу є створення та публікація оголошення. Даний процес включає введення основної інформації про товар або послугу, вибір категорії, завантаження зображень, зазначення контактних даних та встановлення початкового статусу оголошення. Автоматизація цього процесу повинна забезпечувати перевірку коректності введених даних, обмеження кількості зображень, а також контроль доступу для заблокованих користувачів.

Процес оформлення замовлення є наступним важливим етапом життєвого циклу маркетплейсу. Він передбачає вибір оголошення, визначення кількості товару або параметрів послуги, введення даних для доставки та формування замовлення в системі. На цьому етапі система повинна забезпечувати цілісність даних, коректний розрахунок вартості та збереження інформації про замовлення для подальшої обробки.

Важливим елементом бізнес-логіки є взаємодія між покупцем і продавцем, яка в межах даної системи реалізується через статуси замовлень, перегляд історії продажів та замовлень, а також контактні дані, зазначені в оголошеннях. Автоматизація цих процесів дозволяє зменшити кількість помилок і спрощує контроль виконання замовлень.

Окрему роль відіграє адміністративний бізнес-процес, що включає управління категоріями, модерацію оголошень, блокування користувачів та аудит дій. Наявність адміністративної панелі дозволяє централізовано контролювати роботу маркетплейсу, забезпечувати дотримання правил користування та оперативно реагувати на порушення. Автоматизація адміністративних функцій є необхідною умовою масштабування системи, оскільки зростання кількості користувачів унеможливлює ручне управління контентом.

Для ефективної автоматизації бізнес-процесів маркетплейсу система повинна відповідати таким вимогам:

- чітке розмежування ролей і прав доступу;
- збереження цілісності та актуальності даних;
- можливість розширення бізнес-логіки без значної перебудови системи;
- підтримка журналювання та аудиту дій;
- інтеграція з зовнішніми сервісами, зокрема платіжними системами.

Таким чином, бізнес-процеси маркетплейсу формують основу вимог до програмної системи та визначають необхідність використання модульного підходу до її реалізації. Автоматизація цих процесів дозволяє не лише підвищити

ефективність роботи платформи, але й створює передумови для її подальшого розвитку та адаптації до нових бізнес-моделей.

1.3 Проблеми масштабованості та розширюваності веб-сервісів

Розробка веб-сервісів, зокрема маркетплейсів, на початкових етапах зазвичай орієнтована на мінімально необхідний функціонал, що дозволяє швидко запустити продукт та перевірити його життєздатність. Однак із розвитком системи та зростанням кількості користувачів виникають проблеми, пов’язані з масштабованістю та розширюваністю програмного забезпечення.

Однією з основних проблем є ускладнення архітектури при додаванні нового функціоналу. У разі відсутності чіткого поділу системи на логічні модулі зміни в одній частині коду можуть призводити до небажаних наслідків в інших компонентах. Це ускладнює підтримку проєкту, підвищує ризик появи помилок та збільшує витрати часу на розробку.

Ще однією поширеною проблемою є жорстка прив’язка бізнес-логіки до конкретної моделі використання. Багато веб-маркетплейсів створюються з урахуванням однієї бізнес-моделі, наприклад продажу товарів, що ускладнює подальше розширення функціоналу для підтримки інших сценаріїв, таких як оренда, надання послуг або підписка. Відсутність модульності змушує розробників вносити значні зміни в існуючий код, що негативно впливає на стабільність системи.

Масштабованість веб-сервісів також тісно пов’язана з керуванням даними та доступом до них. Зі збільшенням обсягу даних і кількості одночасних користувачів зростає навантаження на базу даних та серверну частину. Якщо система не спроєктована з урахуванням можливості масштабування, це може призвести до зниження продуктивності та погіршення якості обслуговування користувачів.

Особливу складність становить розмежування доступу до даних у багатокористувацьких системах. У маркетплейсах різні користувачі мають доступ лише до власних даних, тоді як адміністратори повинні мати можливість

переглядати та змінювати інформацію в межах усієї системи. Реалізація таких обмежень виключно на рівні прикладної логіки може бути недостатньо надійною та створює додаткові ризики безпеки.

Крім того, важливою проблемою є адаптація системи до змін у зовнішньому середовищі, зокрема інтеграція нових сервісів, платіжних систем або каналів взаємодії з користувачами. У випадку відсутності чітко визначених точок інтеграції такі зміни можуть вимагати значної переробки існуючого коду.

Зазначені проблеми свідчать про необхідність використання архітектурних підходів, що забезпечують модульність, ізольованість компонентів та чіткий розподіл відповідальностей. Модульна архітектура дозволяє розвивати окремі частини системи незалежно, спрощує тестування та супровід, а також створює передумови для масштабування веб-сервісу без критичних змін у його основі.

Таким чином, питання масштабованості та розширюваності є ключовими при проєктуванні сучасних веб-маркетплейсів. Їх урахування на етапі розробки архітектури дозволяє створити гнучку та надійну систему, здатну адаптуватися до змін бізнес-вимог і зростання навантаження.

1.4 Аналіз існуючих програмних рішень та підходів до реалізації

На сучасному етапі розвитку веб-технологій існує значна кількість підходів до розробки інформаційних систем типу маркетплейс. Вибір конкретного підходу залежить від масштабів проєкту, вимог до продуктивності, безпеки та можливостей подальшого розвитку. Аналіз існуючих рішень дозволяє визначити їх переваги та недоліки, а також обґрунтувати доцільність вибору архітектури для розроблюваної системи.

Одним із найбільш поширених підходів є монолітна архітектура, при якій уся бізнес-логіка, користувацький інтерфейс та доступ до даних реалізуються в межах одного програмного застосунку. Основною перевагою такого підходу є простота початкової розробки та розгортання. Проте зі зростанням функціональності монолітні системи стають складними для підтримки, оскільки будь-які зміни потребують перекомпіляції та повторного розгортання всього

застосунку. Крім того, масштабування окремих компонентів у межах моноліту є обмеженим.

Іншим підходом є мікросервісна архітектура, яка передбачає розбиття системи на незалежні сервіси, кожен з яких відповідає за окрему бізнес-функцію. Мікросервіси забезпечують високу гнучкість, незалежне масштабування та можливість використання різних технологій. Водночас цей підхід ускладнює інфраструктуру, вимагає додаткових витрат на оркестрацію сервісів, моніторинг та забезпечення безпеки взаємодії між ними. Для проєктів середнього масштабу, зокрема бакалаврських кваліфікаційних робіт, використання мікросервісів часто є надмірним.

Альтернативним підходом є модульний моноліт, який поєднує переваги монолітної та мікросервісної архітектур. У межах цього підходу система реалізується як єдиний застосунок, проте внутрішньо поділяється на ізольовані модулі з чітко визначеними зонами відповідальності. Такий підхід спрощує розгортання та підтримку, водночас забезпечуючи можливість поступового розвитку та рефакторингу окремих частин системи.

Окремої уваги заслуговує використання Backend as a Service (BaaS) платформ, які надають готову інфраструктуру для роботи з базою даних, автентифікацією, зберіганням файлів та безпекою. Застосування BaaS дозволяє значно скоротити час розробки та зосередитися на бізнес-логіці застосунку. Одним із сучасних рішень цього класу є Supabase, який поєднує реляційну базу даних PostgreSQL з розвиненими механізмами автентифікації та контролю доступу.

Використання Supabase відкриває можливість реалізації безпеки не лише на рівні прикладної логіки, але й безпосередньо на рівні бази даних завдяки механізму Row Level Security. Це дозволяє зменшити ризики витоку даних та підвищити надійність системи, особливо в багатокористувацьких середовищах, характерних для маркетплейсів.

Таким чином, аналіз існуючих підходів до реалізації веб-маркетплейсів показує, що для розробки модульної, безпечної та розширюваної системи

доцільним є використання модульного архітектурного підходу у поєднанні з сучасними BaaS-рішеннями. Такий підхід дозволяє досягти балансу між гнучкістю, складністю реалізації та можливостями подальшого розвитку системи.

1.5 Обґрунтування вибору модульної архітектури веб-сервісу

З урахуванням проаналізованих особливостей предметної області, бізнес-процесів маркетплейсів та існуючих архітектурних підходів, доцільним є використання модульної архітектури веб-сервісу. Такий підхід дозволяє забезпечити гнучкість, масштабованість та зручність подальшого розвитку програмної системи без необхідності її повної перебудови.

Модульна архітектура передбачає поділ системи на окремі логічні модулі, кожен з яких відповідає за певну функціональну область. У межах маркетплейсу такими модулями можуть бути управління користувачами, робота з оголошеннями, обробка замовлень, адміністративний функціонал, інтеграція зовнішніх сервісів та інші компоненти. Кожен модуль розробляється з чітко визначеним інтерфейсом взаємодії, що зменшує зв’язність між компонентами системи.

Основною перевагою модульної архітектури є можливість незалежного розвитку окремих частин системи. Додавання нових функцій або бізнес-моделей не потребує змін у всій системі, а обмежується розширенням відповідних модулів. Це особливо важливо для маркетплейсів, які можуть еволюціонувати від простої платформи оголошень до повноцінного сервісу з підтримкою різних типів взаємодії між користувачами.

Ще одним важливим аспектом є спрощення тестування та супроводу програмного забезпечення. Модульна структура дозволяє локалізувати помилки в межах окремих компонентів, що зменшує ризик негативного впливу на інші частини системи. Крім того, така архітектура полегшує впровадження змін і покращень без порушення стабільності роботи веб-сервісу.

У межах даної кваліфікаційної роботи модульний підхід реалізується з використанням монорепозиторію, що дозволяє об’єднати frontend- та backend-частини проєкту в єдину структуру з централізованим управлінням залежностями. Такий підхід забезпечує узгодженість розробки, спрощує обмін кодом між модулями та зменшує витрати на підтримку проєкту.

Важливою складовою обґрунтування вибору архітектури є також питання безпеки. Модульна архітектура у поєднанні з використанням платформи Supabase дозволяє реалізувати контроль доступу до даних не лише на рівні серверної логіки, але й безпосередньо на рівні бази даних. Застосування механізму Row Level Security забезпечує додатковий рівень захисту інформації та знижує ризик несанкціонованого доступу.

Таким чином, вибір модульної архітектури веб-сервісу є обґрунтованим з точки зору вимог до масштабованості, безпеки та підтримуваності системи. Застосування даного підходу створює основу для реалізації повнофункціонального маркетплейсу та забезпечує можливість його подальшого розширення під інші бізнес-моделі.

Висновки до розділу 1

У першому розділі розглянуто предметну область веб-маркетплейсів та визначено їх місце серед сучасних інформаційних систем електронної комерції. Було встановлено, що маркетплейс не можна розглядати лише як сайт для розміщення товарів або послуг, оскільки така система поєднує значну кількість взаємопов’язаних процесів: реєстрацію користувачів, створення оголошень, перегляд пропозицій, оформлення замовлень, модерацію контенту, адміністрування та, за потреби, взаємодію з платіжними сервісами.

Окрему увагу приділено бізнес-процесам маркетплейсу. Було визначено, що основними процесами є створення та публікація оголошень, перегляд товарів або послуг, оформлення замовлень, управління користувачами та модерація контенту. Їх автоматизація є необхідною умовою стабільної роботи системи, оскільки ручне виконання таких дій при збільшенні кількості користувачів стає

неефективним. Автоматизація також зменшує ризик помилок, покращує контроль над даними та робить систему більш зручною для користувачів.

У процесі аналізу було визначено проблеми, які виникають під час розвитку веб-сервісів. Найбільш суттєвими з них є ускладнення структури проєкту, зростання кількості залежностей між компонентами, складність додавання нових функцій і потреба у масштабуванні. Якщо на етапі проєктування не передбачити можливість розширення системи, подальші зміни можуть потребувати значної переробки програмного коду. Це особливо важливо для маркетплейсів, які з часом можуть змінювати бізнес-модель, додавати нові ролі користувачів або інтегрувати додаткові сервіси.

Також було розглянуто основні підходи до побудови веб-систем. Монолітна архітектура є зручною для швидкого старту, проте з часом може ускладнювати підтримку та масштабування. Мікросервісна архітектура, навпаки, забезпечує високу гнучкість, але потребує складнішої інфраструктури та є надмірною для проєктів невеликого або середнього масштабу. Найбільш доцільним варіантом для розроблюваної системи визначено модульний підхід, який дозволяє зберегти відносну простоту реалізації та водночас забезпечити можливість подальшого розвитку окремих частин системи.

Отже, результати першого розділу підтвердили актуальність розробки модульного веб-сервісу типу маркетплейс. Було встановлено, що така система має бути не лише функціональною, а й гнучкою, безпечною та придатною до подальшого розширення. Проведений аналіз став основою для визначення архітектури програмного продукту, вибору технологічного стеку, проєктування бази даних і формування моделі доступу користувачів.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА АРХІТЕКТУРА ВЕБ СЕРВІСУ

2.1 Загальна характеристика та структура програмної системи

Розроблювана програмна система є веб-орієнтованим маркетплейсом з адміністративною панеллю, призначеним для публікації оголошень, оформлення замовлень та управління взаємодією між користувачами платформи. Система спроєктована з урахуванням вимог до модульності, масштабованості та безпеки, що дозволяє використовувати її як основу для реалізації різних бізнес-моделей у сфері електронної комерції.

Загалом програмна система реалізує повний цикл роботи маркетплейсу та включає такі основні функціональні компоненти:

- користувацький веб-інтерфейс;
- серверну частину з реалізацією бізнес-логіки;
- систему зберігання та обробки даних;
- механізми автентифікації та авторизації;
- адміністративний модуль управління контентом і користувачами.

Взаємодія між компонентами системи організована за принципом клієнт-серверної архітектури. Користувацька частина здійснює обмін даними з серверною частиною через HTTP-запити до REST API, тоді як серверна частина відповідає за обробку запитів, перевірку прав доступу та взаємодію з базою даних і зовнішніми сервісами.

Архітектура системи побудована на основі **монорепозиторію**, що дозволяє зберігати frontend- та backend-частини проєкту в єдиній структурі. Такий підхід забезпечує узгодженість розробки, спрощує управління залежностями та полегшує супровід програмного забезпечення. Монорепозиторій містить окремі застосунки для клієнтської та серверної частин, а також спільні конфігураційні та допоміжні файли.

Структура проєкту має такий вигляд:

- каталог apps/web, що містить клієнтський веб-застосунок, реалізований із використанням бібліотеки React;
- каталог apps/api, який містить серверний застосунок на платформі Node.js з використанням фреймворку Express;
- каталог supabase, що включає SQL-скрипти для створення структури бази даних, політик безпеки та початкового наповнення даними;
- конфігураційні файли монорепозиторію та документацію проєкту.

Користувачами системи є зареєстровані та незареєстровані відвідувачі платформи, а також адміністратори. Незареєстровані користувачі мають можливість переглядати активні оголошення, тоді як зареєстровані користувачі можуть створювати власні оголошення, оформлювати замовлення та переглядати історію своїх операцій. Адміністратори системи володіють розширеними правами доступу, що дозволяють керувати категоріями, модерацією оголошень та користувачами.

Важливою характеристикою системи є реалізація багаторівневої моделі безпеки. Контроль доступу до функціональних можливостей здійснюється як на рівні клієнтського інтерфейсу та серверної логіки, так і на рівні бази даних. Такий підхід забезпечує додатковий захист інформації та мінімізує ризик несанкціонованого доступу до даних.

Таким чином, розроблювана програмна система являє собою модульний веб-сервіс з чітко визначеною структурою та розподілом відповідальностей між компонентами. Обрана архітектура створює передумови для подальшого розвитку системи, спрощує її супровід та дозволяє адаптувати рішення під нові бізнес-вимоги.

2.2 Обґрунтування вибору технологічного стеку

Вибір технологічного стеку є одним із ключових етапів проєктування веб-сервісу, оскільки він безпосередньо впливає на продуктивність, масштабованість, безпеку та зручність подальшого розвитку програмної системи. Для реалізації розроблюваного маркетплейсу було обрано сучасний

стек веб-технологій, який поєднує гнучкість, надійність і широку підтримку спільнотою розробників.

Для реалізації користувацького інтерфейсу використано бібліотеку React, яка є однією з найбільш поширених технологій для створення односторінкових веб-застосунків. Основною перевагою React є компонентний підхід, що дозволяє розбивати інтерфейс на незалежні багаторазово використовувані елементи. Це спрощує підтримку коду та сприяє реалізації модульної архітектури на рівні клієнтської частини.

У якості інструменту збірки frontend-застосунку обрано Vite, який забезпечує швидкий старт розробки та ефективну збірку проєкту. Використання Vite дозволяє зменшити час компіляції та підвищити продуктивність процесу розробки, що є важливим фактором при роботі з масштабними інтерфейсами.

Для підвищення надійності та типобезпеки коду використано TypeScript, який є надбудовою над мовою JavaScript. TypeScript дозволяє виявляти значну частину помилок ще на етапі розробки, покращує читабельність коду та спрощує його супровід у великих проєктах. Використання єдиного типізованого підходу як у frontend-, так і в backend-частині забезпечує узгодженість програмної логіки.

Для стилізації користувацького інтерфейсу застосовано Tailwind CSS, який реалізує утилітарний підхід до побудови дизайну. Використання Tailwind CSS дозволяє швидко створювати адаптивні інтерфейси без необхідності написання великої кількості кастомних CSS-стилів. Це сприяє збереженню єдиного візуального стилю та спрощує підтримку інтерфейсу.

Маршрутизація в клієнтській частині реалізована за допомогою React Router, що дозволяє організувати навігацію між сторінками без перезавантаження браузера. Для роботи з асинхронними даними та керування станом запитів використано TanStack Query, який забезпечує кешування, повторне виконання запитів та синхронізацію стану інтерфейсу з сервером.

Серверна частина системи реалізована на платформі Node.js з використанням фреймворку Express. Такий вибір обумовлений високою продуктивністю Node.js при обробці великої кількості паралельних запитів, а

також можливістю використання єдиної мови програмування для всієї системи. Express надає мінімалістичний та гнучкий інструментарій для створення REST API та реалізації бізнес-логіки.

Для зберігання даних, автентифікації користувачів і роботи з файлами використано платформу Supabase, яка є сучасним рішенням класу Backend as a Service. Supabase базується на реляційній базі даних PostgreSQL та надає розвинені механізми безпеки, зокрема підтримку Row Level Security. Використання Supabase дозволяє зменшити складність серверної частини та підвищити надійність роботи з даними.

Таким чином, обраний технологічний стек забезпечує реалізацію вимог до модульності, масштабованості та безпеки веб-сервісу. Поєднання сучасних frontend- і backend-технологій створює міцну основу для розробки маркетплейсу та його подальшого розвитку під інші бізнес-моделі.

2.3 Архітектура системи та взаємодія компонентів

Архітектура розроблюваного веб-сервісу побудована з урахуванням принципів модульності, чіткого розподілу відповідальностей та мінімізації зв’язності між окремими компонентами системи. Такий підхід забезпечує стабільність роботи застосунку, спрощує його супровід та створює умови для подальшого масштабування.

Загальна архітектура системи реалізована за клієнт-серверною моделлю, у межах якої користувацький веб-інтерфейс взаємодіє із серверною частиною через стандартизовані HTTP-запити. Серверна частина, у свою чергу, забезпечує обробку запитів, виконання бізнес-логіки та доступ до бази даних і зовнішніх сервісів.

Користувацька частина системи виконує такі основні функції:

- відображення даних у зручному для користувача вигляді;
- збір і валідація введених даних;
- ініціювання запитів до серверної частини;
- обробку відповідей сервера та відображення результатів.

Frontend-застосунок не містить бізнес-логіки, пов’язаної з обмеженнями доступу до даних, а виконує лише роль інтерфейсу взаємодії. Це дозволяє уникнути дублювання логіки та зменшує ризики порушення безпеки.

Серверна частина системи є центральним елементом архітектури та відповідає за:

- реалізацію REST API;
- перевірку прав доступу користувачів;
- обробку бізнес-правил маркетплейсу;
- взаємодію з Supabase та іншими зовнішніми сервісами;
- централізовану обробку помилок.

Взаємодія з базою даних здійснюється через Supabase, який виступає не лише як сховище даних, але й як компонент системи безпеки. Завдяки використанню механізму Row Level Security контроль доступу до даних реалізується безпосередньо на рівні бази даних, що дозволяє зменшити залежність безпеки від прикладної логіки серверної частини.

Окрему роль в архітектурі системи відіграє модуль автентифікації, реалізований на основі Supabase Auth. Процес автентифікації відбувається із використанням токенів доступу, які передаються між клієнтом і сервером та використовуються для ідентифікації користувача при виконанні запитів. Серверна частина використовує ці дані для визначення ролі користувача та перевірки дозволених дій.

Адміністративна панель реалізована як частина клієнтського застосунку, проте доступ до її функціоналу обмежений роллю адміністратора. Запити, що надходять з адміністративної частини, обробляються тими самими серверними ендпоінтами, але з додатковими перевірками прав доступу та розширеними можливостями управління даними.

Інтеграція з платіжною системою LiqPay реалізується через серверну частину, що дозволяє уникнути прямої взаємодії клієнта з платіжним сервісом. Такий підхід підвищує рівень безпеки та дозволяє контролювати обробку платіжних подій у централізованому вигляді.

Таким чином, архітектура системи забезпечує чітке розмежування обов’язків між компонентами, знижує ризики порушення безпеки та створює основу для подальшого розвитку веб-сервісу. Взаємодія компонентів реалізована у спосіб, що відповідає сучасним вимогам до проєктування веб-застосунків і дозволяє підтримувати високий рівень надійності та масштабованості.

На рис. 2.1 нижче показано основні компоненти системи та канали їх взаємодії. Клієнтський веб-застосунок (Web UI) звертається до серверної частини через REST API по HTTPS, що забезпечує централізовану обробку бізнес-логіки та перевірок доступу. Сервер взаємодіє з Supabase (PostgreSQL) для виконання операцій із даними, використовує Supabase Auth для роботи з сесіями та ролями, а також Supabase Storage для завантаження і отримання зображень оголошень. Інтеграція LiqPay реалізована через сервер, що дозволяє не розкривати секретні параметри клієнту та контролювати платіжний флоу.

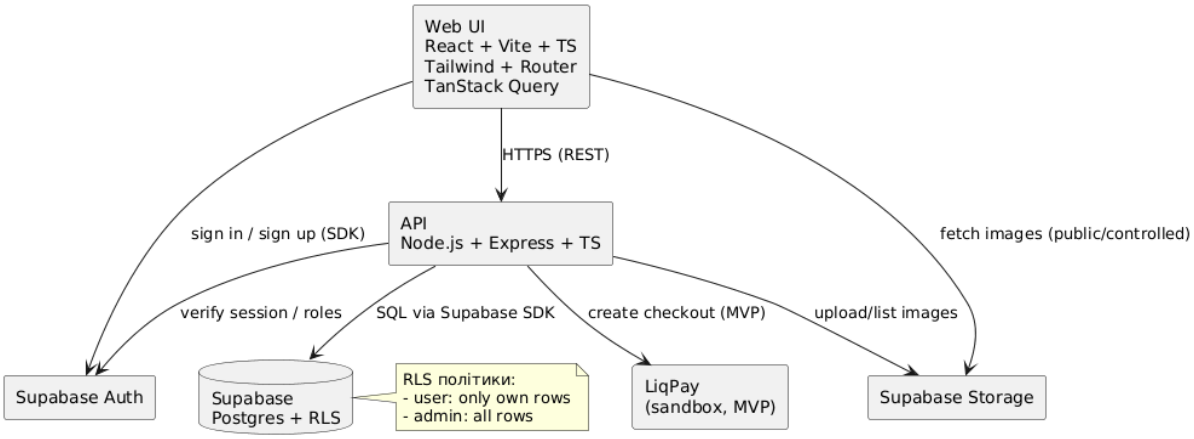

Рисунок 2.1. Архітектура веб-сервісу та взаємодія його компонентів

2.4 Проєктування монорепозиторію та організація модулів

Для організації структури програмної системи у даній кваліфікаційній роботі використано підхід монорепозиторію, який передбачає зберігання всіх складових проєкту в межах одного репозиторію з єдиною системою керування залежностями. Такий підхід є доцільним для модульних веб-систем, у яких

frontend- та backend-частини тісно пов’язані між собою та розвиваються паралельно.

Основною перевагою монорепозиторію є централізоване управління кодовою базою, що дозволяє уникнути дублювання конфігурацій, спростити обмін типами даних та узгодити версії залежностей між різними частинами системи. Це особливо важливо при використанні TypeScript, оскільки забезпечується єдність типів даних у клієнтській і серверній частинах.

У межах проєкту монорепозиторій організований за логічним принципом поділу на застосунки та допоміжні модулі. Основні компоненти структури включають:

- каталог apps/web, що містить клієнтський веб-застосунок;
- каталог apps/api, що містить серверну частину з реалізацією REST API;
- каталог supabase, у якому зосереджено SQL-скрипти для створення структури бази даних, політик безпеки та початкового наповнення;
- конфігураційні файли керування залежностями та робочим простором.

Клієнтський застосунок у каталозі apps/web структуровано за модульним принципом, що передбачає поділ на компоненти інтерфейсу, сторінки, сервіси для роботи з API та допоміжні утиліти. Такий підхід дозволяє локалізувати зміни та полегшує підтримку інтерфейсу при розширенні функціональності системи.

Серверна частина, розміщена у каталозі apps/api, організована відповідно до принципів розділення відповідальностей. Основні логічні модулі включають обробку HTTP-запитів, реалізацію бізнес-логіки, валідацію даних, взаємодію з Supabase та інтеграцію зовнішніх сервісів. Кожен модуль виконує чітко визначену функцію, що сприяє зниженню зв’язності між компонентами.

Для управління залежностями в монорепозиторії використано pnpm workspaces, що дозволяє оптимізувати використання дискового простору та прискорити процес встановлення залежностей. Використання єдиного

менеджера пакетів забезпечує узгодженість версій бібліотек та спрощує налаштування середовища розробки.

Окрему увагу приділено організації спільних конфігураційних файлів, таких як змінні середовища, правила форматування коду та документація. Наявність файлу .env.example дозволяє стандартизувати налаштування середовища та зменшити ризик помилок під час розгортання системи.

Таким чином, використання монорепозиторію в поєднанні з модульною організацією коду дозволяє створити впорядковану та масштабовану структуру програмної системи. Такий підхід спрощує командну розробку, полегшує тестування та створює основу для подальшого розширення веб-сервісу без суттєвих архітектурних змін.

На рис. 2.2 відображено організацію репозиторію у вигляді окремих застосунків apps/web та apps/api, а також каталогу supabase, який містить SQL-скрипти для створення схеми, політик безпеки та початкового наповнення даними. Така структура забезпечує ізоляцію відповідальностей між клієнтською та серверною частинами, спрощує керування залежностями через pnpm-workspace.yaml і забезпечує зручність розгортання та супроводу проєкту.

2.5 Проєктування бізнес-процесів маркетплейсу

Проєктування бізнес-процесів є важливим етапом створення веб-сервісу, оскільки саме вони визначають логіку взаємодії користувачів із системою та послідовність виконання основних операцій. Для розроблюваного маркетплейсу бізнес-процеси сформовано з урахуванням ролей користувачів, функціональних вимог та обмежень безпеки.

Основними учасниками бізнес-процесів у системі є:

- незареєстрований користувач;
- зареєстрований користувач (покупець або продавець);
- адміністратор системи.

Кожна з ролей має власний набір дозволених дій, що визначає доступ до відповідних бізнес-процесів.

Базовим бізнес-процесом маркетплейсу є перегляд оголошень. Незареєстровані користувачі мають можливість переглядати перелік активних оголошень, фільтрувати їх за категоріями та переходити на сторінку окремого оголошення. Даний процес не потребує автентифікації, що дозволяє залучити широку аудиторію користувачів.

Процес реєстрації та автентифікації користувачів забезпечує ідентифікацію користувача в системі та відкриває доступ до розширеного функціоналу. Після успішної автентифікації користувач може створювати власні оголошення, оформлювати замовлення та переглядати історію взаємодії з маркетплейсом. Реалізація даного процесу ґрунтується на використанні Supabase Auth, що забезпечує безпечне зберігання облікових даних.

Одним із ключових бізнес-процесів є створення та публікація оголошення. Користувач вводить основні дані про товар або послугу, обирає категорію, додає зображення та контактну інформацію. Після збереження оголошення воно набуває статусу активного та стає доступним для перегляду іншими користувачами. Для заблокованих користувачів даний процес недоступний, що забезпечує дотримання правил користування сервісом.

Наступним важливим етапом є оформлення замовлення. Покупець обирає оголошення, зазначає кількість товару або параметри послуги, а також вводить адресу та контактний телефон для доставки. На основі цих даних у системі формується замовлення з відповідним статусом, що зберігається в базі даних та використовується для подальшої обробки.

Процес оплати замовлення у межах даної роботи реалізовано у форматі мінімально життєздатного продукту (MVP) з використанням платіжної системи LiqPay у sandbox-режимі. Це дозволяє продемонструвати інтеграцію платіжного сервісу без здійснення реальних фінансових операцій. Після ініціації оплати система фіксує відповідний статус замовлення.

Окрему групу бізнес-процесів становлять адміністративні процеси, що включають управління категоріями, модерацію оголошень, блокування користувачів та перегляд журналу аудиту. Адміністратор має можливість

змінювати статус оголошень, що дозволяє контролювати якість контенту та дотримання правил користування платформою.

Таким чином, проєктування бізнес-процесів маркетплейсу дозволяє формалізувати основні сценарії використання системи та визначити вимоги до реалізації програмної логіки. Чіткий опис бізнес-процесів є основою для подальшого проєктування структури бази даних і реалізації функціональних модулів веб-сервісу.

Нижче на рисунку 2.2 подано основні варіанти використання для трьох ролей: гість, користувач і адміністратор. Гість має доступ до перегляду оголошень та сторінки окремого товару, а також може виконати реєстрацію або вхід. Авторизований користувач додатково отримує можливість створювати та керувати власними оголошеннями, оформлювати замовлення, переглядати розділи “My orders” і “My sales”, а також ініціювати оплату в режимі MVP. Адміністратор має доступ до CRUD операцій для категорій, модерації оголошень, блокування користувачів і перегляду журналу аудиту.

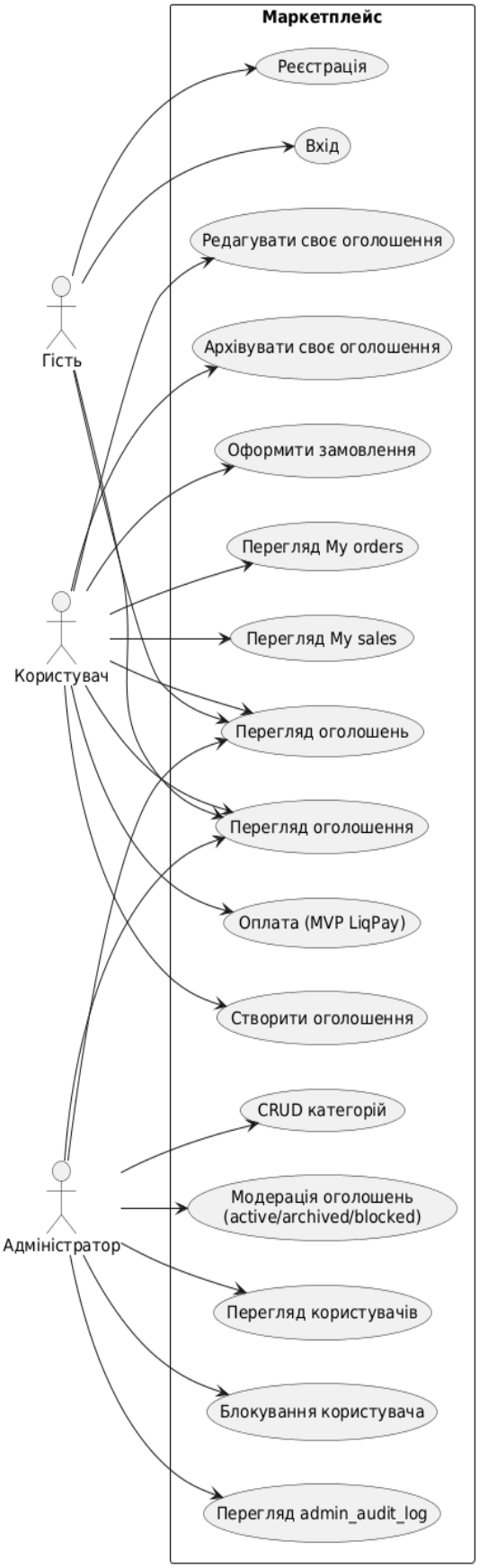

Рисунок 2.2. Діаграма варіантів використання бізнес-процесів маркетплейсу

2.6 Проєктування структури бази даних

Проєктування структури бази даних є одним із ключових етапів створення веб-сервісу, оскільки саме база даних забезпечує зберігання, цілісність та доступність інформації, що використовується в усіх бізнес-процесах маркетплейсу. Коректно спроєктована структура даних дозволяє спростити реалізацію серверної логіки, підвищити продуктивність системи та забезпечити безпеку доступу до інформації.

У межах даної кваліфікаційної роботи для зберігання даних використовується реляційна база даних PostgreSQL, що надається платформою Supabase. Реляційна модель даних обрана з огляду на необхідність збереження чітких зв’язків між сутностями, забезпечення цілісності даних та підтримки складних запитів.

Під час проєктування структури бази даних було виділено основні сутності предметної області маркетплейсу, які відображають ключові бізнес-процеси системи. До таких сутностей належать:

- користувачі платформи;
- категорії оголошень;
- оголошення товарів або послуг;
- замовлення;
- дії адміністратора.

Центральною сутністю системи є користувач, інформація про якого зберігається у таблиці profiles. Дана таблиця містить додаткові дані, пов’язані з користувачем, зокрема роль у системі, статус блокування та контактну інформацію. Таблиця profiles логічно пов’язана з механізмом автентифікації Supabase та використовується для реалізації рольової моделі доступу.

Для класифікації оголошень у системі використовується таблиця categories, яка зберігає перелік доступних категорій та їх унікальні ідентифікатори. Такий підхід дозволяє централізовано керувати категоріями та спрощує фільтрацію оголошень у користувацькому інтерфейсі.

Таблиця listings призначена для зберігання інформації про оголошення, створені користувачами. Вона містить посилання на власника оголошення, категорію, основні характеристики товару або послуги, контактні дані та статус публікації. Наявність статусів дозволяє реалізувати механізм модерації контенту та керування життєвим циклом оголошень.

Для обробки покупок у системі використовується таблиця orders, яка зберігає інформацію про замовлення, створені покупцями. Кожне замовлення пов’язане з конкретним оголошенням і користувачем, що дозволяє відстежувати історію покупок і продажів. Крім того, таблиця містить дані для доставки та статус замовлення, що використовується у бізнес-логіці системи.

Окремою сутністю є таблиця admin_audit_log, яка призначена для фіксації дій адміністраторів. Наявність журналу аудиту дозволяє підвищити прозорість управління системою, а також спрощує аналіз змін і розв’язання спірних ситуацій.

Між основними таблицями бази даних встановлено зв’язки типу «один-до-багатьох», що забезпечує логічну цілісність даних. Для забезпечення коректності збереження інформації використовуються первинні та зовнішні ключі, а також обмеження на рівні бази даних.

Таким чином, спроєктована структура бази даних відображає основні сутності предметної області та підтримує всі ключові бізнес-процеси маркетплейсу. Вона створює основу для реалізації механізмів безпеки та контролю доступу, що розглядаються у наступному підрозділі.

На рис. 2.3 показано основні таблиці системи: profiles, categories, listings, orders та admin_audit_log. Таблиця listings пов’язана з profiles через owner_id та з categories через category_id, що відображає приналежність оголошення користувачеві та категорії. Таблиця orders містить посилання на покупця (buyer_id) і відповідне оголошення (listing_id), що дозволяє відстежувати історію покупок і продажів. Таблиця admin_audit_log фіксує дії адміністратора через зв’язок admin_id з profiles, забезпечуючи аудит управлінських операцій.

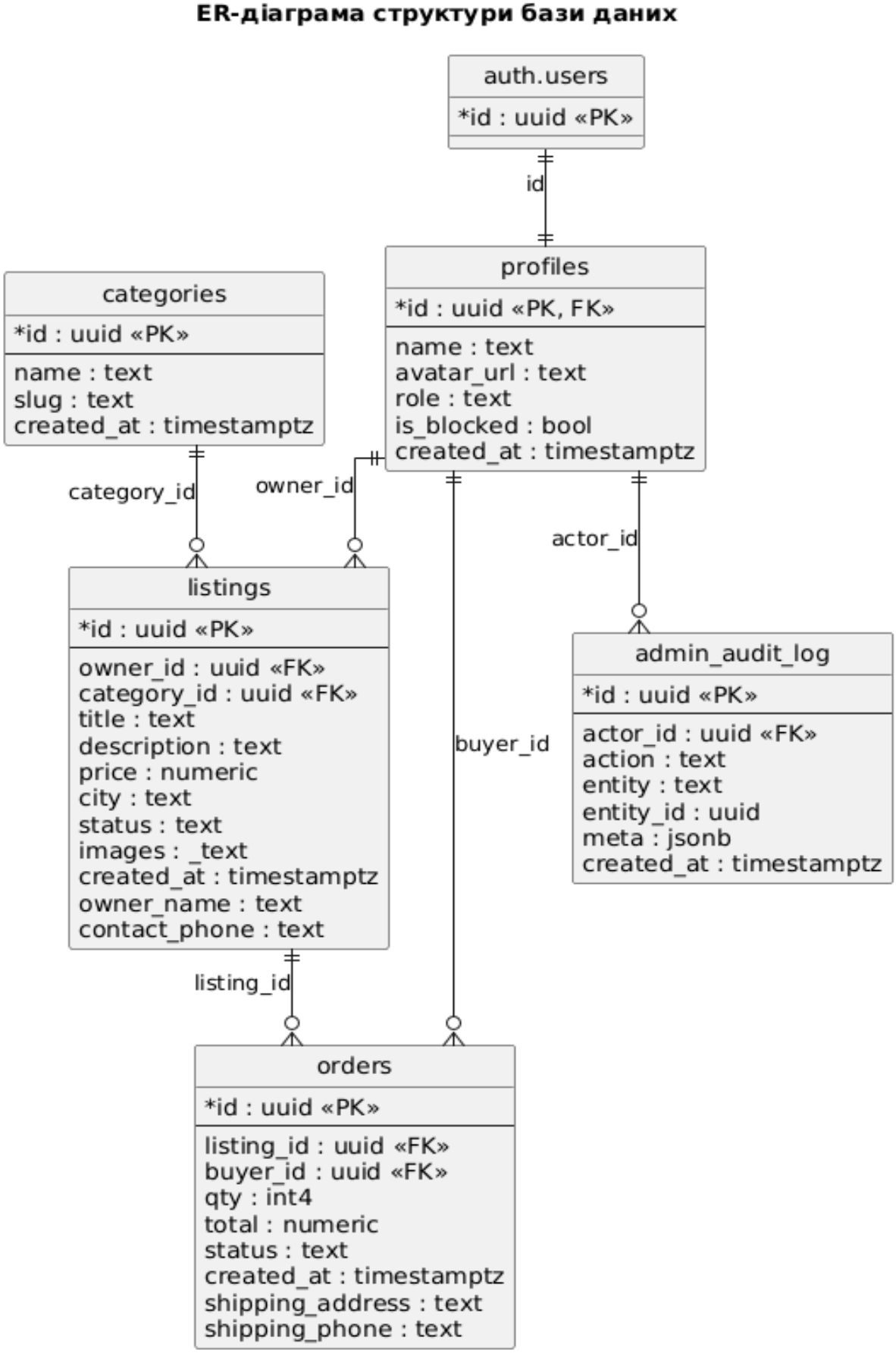

Рисунок 2.3. ER-діаграма структури бази даних

2.7 Проєктування ролей користувачів та моделей доступу

Одним із ключових аспектів проєктування веб-маркетплейсу є визначення ролей користувачів та моделей доступу до функціональних можливостей і даних системи. У багатокористувацьких інформаційних системах коректна реалізація контролю доступу є необхідною умовою забезпечення безпеки, цілісності даних та стабільної роботи сервісу.

У розроблюваній системі передбачено дві основні ролі користувачів:

- звичайний користувач (user);
- адміністратор (admin).

Звичайний користувач може виконувати дії, пов’язані з переглядом оголошень, створенням власних оголошень, оформленням замовлень та переглядом історії своїх покупок і продажів. При цьому користувач має доступ виключно до власних даних, що відповідає вимогам конфіденційності та захисту персональної інформації.

Адміністратор системи володіє розширеними правами доступу, які дозволяють керувати всіма основними сутностями маркетплейсу. До повноважень адміністратора належать створення та редагування категорій, зміна статусів оголошень, блокування користувачів, а також перегляд і аналіз журналу аудиту дій. Наявність окремої адміністративної ролі дозволяє централізовано контролювати роботу платформи та забезпечувати дотримання правил користування сервісом.

Окремо в системі реалізовано механізм блокування користувачів, який є важливим елементом моделі доступу. Заблокований користувач втрачає можливість створювати оголошення та взаємодіяти з ключовими бізнес-процесами маркетплейсу, проте може зберігати доступ до перегляду загальнодоступної інформації. Такий підхід дозволяє мінімізувати негативний вплив порушників правил без повного видалення їх облікових записів.

Модель доступу до даних у системі побудована за принципом мінімально необхідних прав, відповідно до якого кожен користувач має доступ лише до тієї

інформації, яка потрібна для виконання його функцій. Даний принцип реалізується як на рівні прикладної логіки серверної частини, так і на рівні бази даних.

Важливою особливістю розроблюваної системи є використання механізму Row Level Security, який дозволяє визначати правила доступу до окремих рядків таблиць залежно від ролі та ідентифікатора користувача. Такий підхід зменшує залежність безпеки від серверної логіки та забезпечує додатковий рівень захисту даних навіть у випадку помилок у прикладному коді.

Роль користувача та інформація про його статус зберігаються у таблиці profiles і використовуються під час виконання запитів до бази даних. Серверна частина системи передає контекст користувача до Supabase, що дозволяє застосовувати відповідні політики доступу автоматично.

Таким чином, проєктування ролей користувачів та моделей доступу забезпечує чітке розмежування прав у системі, підвищує рівень безпеки та створює основу для реалізації складних бізнес-сценаріїв у маркетплейсі. Запропонована модель доступу відповідає сучасним вимогам до веб-сервісів та є придатною для подальшого масштабування системи.

Висновки до розділу 2

У другому розділі було виконано проєктування програмної системи та визначено основні архітектурні рішення для реалізації веб-сервісу маркетплейсу. На цьому етапі сформовано загальну структуру майбутнього застосунку, визначено його основні компоненти, способи їх взаємодії та технології, які доцільно використати для розробки.

Розроблювану систему спроєктовано як веб-орієнтований маркетплейс з адміністративною панеллю. Вона має забезпечувати перегляд оголошень, створення користувацького контенту, оформлення замовлень, управління категоріями, модерацію та контроль доступу. Для реалізації цих можливостей було обрано клієнт-серверну архітектуру, у якій frontend відповідає за взаємодію

з користувачем, backend — за обробку бізнес-логіки, а база даних — за зберігання та захист інформації.

Важливим результатом проєктування стало визначення технологічного стеку. Для клієнтської частини обрано React, оскільки ця бібліотека дає змогу будувати інтерфейс із незалежних компонентів і спрощує повторне використання коду. Vite використовується як інструмент збірки, що забезпечує швидку розробку та оптимізацію застосунку. TypeScript підвищує надійність програмного коду завдяки типізації, а Tailwind CSS дозволяє створювати адаптивний інтерфейс без надмірного ускладнення стилів.

Серверна частина системи проєктується на основі Node.js та Express. Такий вибір є доцільним, оскільки ці технології дозволяють створити гнучкий REST API, організувати обробку HTTP-запитів і реалізувати основну бізнес-логіку системи. Використання JavaScript/TypeScript як на клієнтській, так і на серверній частині спрощує розробку та підтримку проєкту.

Для зберігання даних і реалізації частини backend-функціональності обрано Supabase. Це рішення є важливим для даної роботи, оскільки воно поєднує PostgreSQL, засоби автентифікації, зберігання файлів і механізми контролю доступу. Особливе значення має підтримка Row Level Security, яка дозволяє обмежувати доступ до записів безпосередньо на рівні бази даних. Для маркетплейсу це є важливою перевагою, адже користувачі повинні працювати переважно з власними даними, тоді як адміністратор має мати розширені права.

У розділі також було спроєктовано організацію монорепозиторію. Такий підхід дозволяє зберігати frontend, backend і SQL-скрипти в єдиній структурі, що спрощує керування залежностями, налаштування середовища розробки та супровід проєкту. При цьому кожна частина системи має власну зону відповідальності, що відповідає принципам модульної архітектури.

Окремо було визначено бізнес-процеси системи, структуру бази даних і модель доступу користувачів. До основних сутностей належать користувачі, профілі, категорії, оголошення, замовлення та журнал адміністративних дій. Така структура дозволяє відобразити ключові процеси маркетплейсу та

забезпечити логічні зв’язки між даними. Рольова модель доступу передбачає поділ користувачів на звичайних користувачів і адміністраторів, що є необхідним для безпечної роботи системи.

Отже, у другому розділі було сформовано технічну основу програмної системи. Обрана архітектура, технологічний стек, структура монорепозиторію, база даних і модель доступу відповідають меті роботи та створюють умови для реалізації модульного, безпечного й розширюваного веб-сервісу.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Реалізація користувацького інтерфейсу веб-застосунку

Користувацький інтерфейс розроблюваного веб-сервісу реалізовано у вигляді односторінкового веб-застосунку, основним призначенням якого є забезпечення зручної та інтуїтивно зрозумілої взаємодії користувачів із функціональними можливостями маркетплейсу. Під час реалізації інтерфейсу основна увага приділялася модульності, читабельності коду та відповідності сучасним вимогам до веб-дизайну.

Frontend-частина системи реалізована з використанням бібліотеки React, що дозволило застосувати компонентний підхід до побудови інтерфейсу. Кожен логічний елемент інтерфейсу, зокрема навігаційна панель, картка оголошення, форма створення оголошення або список замовлень, реалізований у вигляді окремого компонента. Такий підхід спрощує повторне використання елементів інтерфейсу та полегшує супровід коду.

Збірка та запуск клієнтського застосунку здійснюються за допомогою інструмента Vite, який забезпечує швидкий старт середовища розробки та ефективну оптимізацію під час створення production-збірки. Це позитивно впливає на продуктивність інтерфейсу та зменшує час завантаження сторінок для кінцевого користувача.

Для стилізації інтерфейсу використано Tailwind CSS, який дозволяє формувати дизайн безпосередньо у розмітці компонентів за допомогою утилітарних класів. Застосування даного підходу дозволило реалізувати єдиний візуальний стиль, адаптивну верстку та забезпечити коректне відображення інтерфейсу на різних типах пристроїв.

Навігація між сторінками веб-застосунку реалізована з використанням React Router. У межах застосунку передбачено такі основні маршрути: головна сторінка з переліком оголошень, сторінка окремого оголошення, сторінки

керування оголошеннями та замовленнями користувача, сторінки авторизації та реєстрації, а також адміністративна панель. Перехід між маршрутами відбувається без перезавантаження сторінки, що забезпечує плавну взаємодію з інтерфейсом.

Для роботи з даними та обробки асинхронних запитів до серверної частини використано бібліотеку TanStack Query. Вона дозволяє централізовано керувати станами запитів, кешувати результати та автоматично оновлювати дані після виконання мутацій. Це значно спрощує реалізацію інтерфейсу та зменшує кількість допоміжного коду, пов’язаного з керуванням станом.

Особливу увагу приділено реалізації форм введення даних, зокрема форм створення оголошень та оформлення замовлень. Усі введені користувачем дані проходять первинну валідацію на клієнтській стороні, що дозволяє зменшити кількість помилкових запитів до сервера та підвищити зручність користування системою.

Інтерфейс веб-застосунку повністю українізований, що відповідає вимогам локалізації та забезпечує зручність використання для цільової аудиторії. Повідомлення про помилки, статуси операцій та елементи навігації відображаються українською мовою.

Таким чином, реалізація користувацького інтерфейсу веб-застосунку забезпечує зручну, зрозумілу та безпечну взаємодію користувачів із маркетплейсом. Обрані технології та підходи дозволяють підтримувати модульність інтерфейсу та створюють умови для подальшого розширення функціональних можливостей системи.

На рис. 3.1 відображено типовий сценарій оплати: користувач ініціює оплату з інтерфейсу, після чого клієнтський застосунок надсилає запит до API. Сервер виконує перевірку замовлення та доступу до нього, звертається до LiqPay для формування параметрів checkout і повертає ці параметри клієнту. Далі клієнт відкриває платіжну форму LiqPay. У межах MVP підтвердження операції може виконуватися спрощено, без повної обробки callback-подій у sandbox-середовищі.

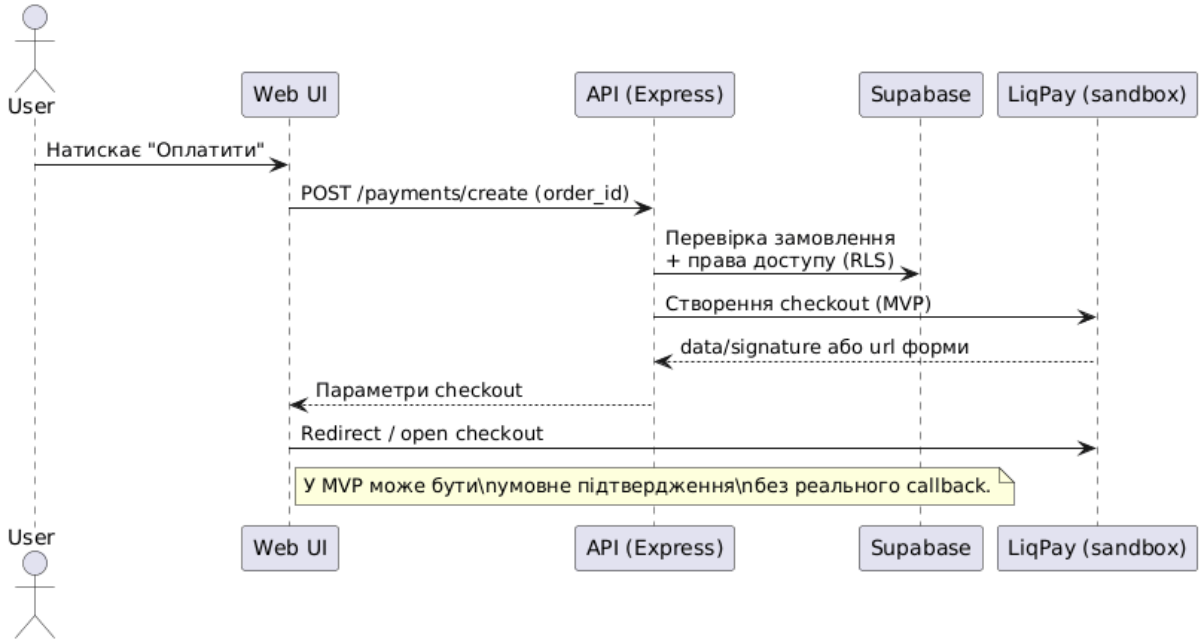

Рисунок 3.1. Діаграма послідовностей процесу оплати (MVP)

3.2 Реалізація механізмів автентифікації та авторизації

Механізми автентифікації та авторизації є критично важливими складовими веб-маркетплейсу, оскільки вони забезпечують ідентифікацію користувачів, контроль доступу до функціональних можливостей системи та захист персональних даних. У розроблюваному веб-сервісі ці механізми реалізовані з використанням платформи Supabase Auth, що дозволяє застосувати сучасні та надійні підходи до управління обліковими записами користувачів.

Автентифікація користувачів здійснюється на основі облікових даних, які вводяться під час реєстрації або входу в систему. Supabase Auth відповідає за безпечне зберігання паролів, їх хешування та перевірку під час входу користувача. У результаті успішної автентифікації користувач отримує токен доступу, який використовується для подальшої взаємодії з серверною частиною та базою даних.

Процес реєстрації користувача передбачає створення нового облікового запису в системі автентифікації Supabase. Після цього автоматично створюється відповідний запис у таблиці profiles, де зберігається додаткова інформація про користувача, зокрема його роль у системі та статус блокування. Такий підхід

дозволяє розділити облікові дані та бізнес-інформацію користувача, що підвищує гнучкість і безпеку системи.

Авторизація користувачів реалізована на основі рольової моделі доступу, у межах якої кожному користувачеві призначається роль user або admin. Роль користувача визначає перелік доступних йому дій як на рівні інтерфейсу, так і на рівні серверної логіки. Наприклад, доступ до адміністративної панелі надається виключно користувачам з роллю адміністратора.

На клієнтській стороні реалізовано механізми умовного відображення інтерфейсу залежно від ролі користувача. Це дозволяє приховувати недоступні елементи навігації та зменшує ризик помилкових дій. Водночас такі обмеження не розглядаються як основний механізм безпеки, а виконують допоміжну роль для покращення користувацького досвіду.

Серверна частина системи виконує перевірку токенів доступу та використовує інформацію про користувача для визначення дозволених операцій. Для взаємодії з Supabase із підвищеними правами доступу використовується Service Role Key, який зберігається виключно на сервері та недоступний клієнтській частині. Це дозволяє виконувати адміністративні операції без ризику компрометації секретних даних.

Особливу увагу приділено реалізації механізму блокування користувачів. У разі встановлення відповідного прапорця в таблиці profiles заблокований користувач втрачає можливість створювати оголошення, оформлювати замовлення та взаємодіяти з ключовими бізнес-процесами системи. Перевірка статусу блокування здійснюється як на рівні серверної логіки, так і через політики доступу бази даних.

Таким чином, реалізовані механізми автентифікації та авторизації забезпечують надійну ідентифікацію користувачів, чітке розмежування прав доступу та захист даних у системі. Застосування Supabase Auth у поєднанні з рольовою моделлю доступу створює міцну основу для безпечної роботи маркетплейсу.

3.3 Реалізація серверної частини та REST API

Серверна частина розроблюваного веб-сервісу виконує ключову роль у реалізації бізнес-логіки маркетплейсу, забезпеченні взаємодії з базою даних та інтеграції зовнішніх сервісів. Backend реалізовано на платформі Node.js з використанням фреймворку Express, що дозволяє створити гнучкий та продуктивний REST API.

Основним призначенням серверної частини є обробка HTTP-запитів від клієнтського застосунку, перевірка прав доступу користувачів, виконання бізнес-правил та повернення коректних відповідей. Сервер виступає проміжною ланкою між користувацьким інтерфейсом і сервісами Supabase, що дозволяє централізувати логіку та підвищити рівень безпеки системи.

Структура серверного застосунку організована за модульним принципом. Основні функціональні блоки включають:

- маршрутизацію HTTP-запитів;
- контролери для обробки бізнес-логіки;
- сервіси для взаємодії з Supabase;
- модулі валідації вхідних даних;
- централізовану обробку помилок.

Маршрутизація в Express реалізована з використанням окремих роутерів для різних сутностей системи, таких як оголошення, замовлення, категорії та користувачі. Кожен маршрут відповідає конкретній операції та використовує відповідний HTTP-метод, що відповідає принципам REST-архітектури.

Для забезпечення надійності роботи API реалізовано перевірку вхідних даних, що дозволяє виявляти некоректні або неповні запити ще на початковому етапі їх обробки. У разі виникнення помилок сервер повертає стандартизовані повідомлення з відповідними HTTP-статусами, що спрощує обробку помилок на клієнтській стороні.

Серверна частина взаємодіє з Supabase для виконання операцій з базою даних, автентифікації користувачів та роботи з файлами. Для виконання

операцій, що потребують підвищених прав доступу, використовується Service Role Key, який зберігається у змінних середовища серверного застосунку. Доступ до цього ключа обмежений лише серверною частиною, що унеможливлює його використання з боку клієнта.

Особливу увагу приділено реалізації бізнес-логіки, пов’язаної зі створенням і зміною статусів оголошень та замовлень. Сервер контролює коректність переходів між статусами, перевіряє роль користувача та забезпечує виконання правил системи незалежно від дій клієнтського інтерфейсу.

Таким чином, реалізація серверної частини та REST API забезпечує централізоване управління бізнес-логікою, підвищує безпеку веб-сервісу та створює надійну основу для взаємодії клієнтського застосунку з базою даних і зовнішніми сервісами.

3.4 Реалізація роботи з базою даних та Supabase

Робота з базою даних у розроблюваному веб-сервісі реалізована з використанням платформи Supabase, яка надає доступ до реляційної бази даних PostgreSQL та додаткові сервіси для автентифікації і зберігання файлів. Такий підхід дозволяє поєднати переваги класичної реляційної моделі даних із сучасними механізмами безпеки та спрощує реалізацію серверної логіки.

Основні операції з базою даних виконуються через серверну частину застосунку. Сервер виступає посередником між клієнтським інтерфейсом і Supabase, що дозволяє централізовано контролювати доступ до даних та застосовувати додаткові перевірки бізнес-логіки. Для виконання запитів використовується офіційний клієнт Supabase, який забезпечує типобезпечну взаємодію з базою даних.

Структура бази даних відповідає проєктуванню, наведеному у попередньому розділі, та включає таблиці profiles, categories, listings, orders і admin_audit_log. Для кожної таблиці визначено первинні ключі, зовнішні ключі та обмеження цілісності, що дозволяє уникнути збереження некоректних або суперечливих даних.

Під час реалізації операцій створення, оновлення та видалення записів особливу увагу приділено перевірці прав доступу користувачів. Наприклад, під час створення оголошення сервер перевіряє, чи є користувач авторизованим і чи не перебуває він у заблокованому стані. Аналогічно, під час роботи з замовленнями перевіряється відповідність користувача ролі покупця або продавця.

Ключовою особливістю реалізації доступу до даних є використання механізму Row Level Security. Для кожної таблиці в базі даних визначено політики доступу, які обмежують можливість перегляду та зміни даних залежно від ідентифікатора користувача та його ролі. Наприклад, звичайний користувач має доступ лише до власних оголошень і замовлень, тоді як адміністратор може переглядати та змінювати всі записи.

Застосування Row Level Security дозволяє перенести значну частину логіки контролю доступу безпосередньо на рівень бази даних. Це знижує ризик витоку інформації у разі помилок у серверному коді та підвищує загальний рівень безпеки системи. Серверна частина у такому випадку передає контекст користувача до Supabase, а база даних самостійно визначає, які записи доступні для конкретного запиту.

Для роботи з файлами, зокрема зображеннями оголошень, використовується Supabase Storage. Зображення завантажуються у спеціально створений bucket, а доступ до них обмежується відповідними правилами безпеки. У базі даних зберігаються лише посилання на файли, що дозволяє зменшити обсяг збережених даних та спростити їх обробку.

Таким чином, реалізація роботи з базою даних та Supabase забезпечує надійне зберігання інформації, чіткий контроль доступу та підтримку всіх основних бізнес-процесів маркетплейсу. Використання Row Level Security у поєднанні з серверною логікою створює багаторівневу модель безпеки, яка відповідає сучасним вимогам до веб-сервісів.

3.5 Реалізація системи оголошень та замовлень

Система оголошень та замовлень є центральним функціональним ядром розроблюваного маркетплейсу, оскільки саме вона забезпечує основну взаємодію між продавцями та покупцями. Реалізація даного модуля ґрунтується на бізнес-процесах, визначених на етапі проєктування, та тісно пов’язана з механізмами безпеки і контролю доступу.

Функціональність оголошень реалізована на основі таблиці listings, яка зберігає інформацію про товари або послуги, створені користувачами. Процес створення оголошення доступний лише авторизованим користувачам із роллю user, які не перебувають у заблокованому стані. Під час створення оголошення користувач вводить назву, опис, ціну, обирає категорію та додає зображення. Серверна частина перевіряє коректність даних і зберігає відповідний запис у базі даних.

Кожне оголошення має статус, який визначає його поточний стан у системі. Передбачено такі основні статуси: активне, архівоване та заблоковане. Статус активного оголошення дозволяє його відображення у загальному переліку та доступність для оформлення замовлень. Архівовані оголошення залишаються доступними лише власнику, тоді як заблоковані оголошення приховуються від користувачів і можуть бути змінені лише адміністратором.

Процес перегляду оголошень доступний усім користувачам платформи, включно з незареєстрованими. Для цього реалізовано відповідний API-ендпоінт, який повертає перелік лише активних оголошень. Завдяки використанню Row Level Security додатково гарантується, що користувачі не мають доступу до прихованих або заблокованих записів.

Система замовлень реалізована на основі таблиці orders, яка зберігає інформацію про покупки, здійснені користувачами. Процес оформлення замовлення починається з вибору оголошення та введення даних для доставки, зокрема адреси та контактного телефону. Після цього сервер формує новий запис замовлення та зберігає його у базі даних з відповідним початковим статусом.

Кожне замовлення пов’язане з конкретним покупцем і оголошенням, що дозволяє відстежувати історію покупок та продажів. Покупець має доступ лише до власних замовлень, тоді як продавець може переглядати замовлення, пов’язані з його оголошеннями. Такий розподіл доступу реалізується через політики Row Level Security та перевірки на рівні серверної логіки.

Зміна статусів замовлень відбувається відповідно до бізнес-правил системи. Наприклад, після створення замовлення воно може переходити у стан оплати або підтвердження. Контроль коректності переходів між статусами здійснюється серверною частиною, що запобігає некоректним діям з боку користувачів.

Таким чином, реалізація системи оголошень та замовлень забезпечує повний цикл взаємодії між продавцями та покупцями в межах маркетплейсу. Поєднання серверної логіки та механізмів безпеки на рівні бази даних дозволяє гарантувати цілісність даних і коректність виконання бізнес-процесів.

3.6 Реалізація адміністративної панелі та модерації

Адміністративна панель є невід’ємною складовою розроблюваного маркетплейсу, оскільки забезпечує централізоване управління контентом, користувачами та основними довідковими даними системи. Реалізація адміністративного функціоналу дозволяє підтримувати порядок на платформі, контролювати дотримання правил користування та оперативно реагувати на порушення.

Адміністративна панель реалізована як частина клієнтського веб-застосунку, проте доступ до неї обмежений виключно користувачами з роллю admin. На рівні інтерфейсу адміністративні сторінки приховані для звичайних користувачів, а на рівні серверної логіки та бази даних реалізовано додаткові перевірки прав доступу.

Однією з основних функцій адміністративної панелі є управління категоріями оголошень. Адміністратор має можливість створювати, редагувати та видаляти категорії, які використовуються для класифікації оголошень у

маркетплейсі. Усі операції з категоріями виконуються через відповідні API-ендпоінти та супроводжуються перевіркою ролі користувача.

Важливим елементом адміністративного функціоналу є модерація оголошень. Адміністратор має можливість переглядати всі оголошення незалежно від їх статусу та змінювати їхній стан. Зокрема, оголошення можуть бути переведені у статус активних, архівованих або заблокованих. Блокування оголошень використовується у випадках порушення правил платформи або наявності некоректного контенту.

Крім того, адміністративна панель надає можливість управління користувачами. Адміністратор може переглядати перелік зареєстрованих користувачів та змінювати їхній статус, зокрема виконувати блокування облікових записів. Заблоковані користувачі автоматично втрачають доступ до створення оголошень і оформлення замовлень, що забезпечується як перевірками на серверній стороні, так і політиками Row Level Security.

Для підвищення прозорості управління системою реалізовано журнал аудиту дій адміністратора, який зберігається у таблиці admin_audit_log. У журналі фіксуються основні дії адміністратора, такі як зміна статусів оголошень, блокування користувачів та редагування категорій. Наявність журналу аудиту дозволяє відстежувати історію змін та спрощує аналіз можливих конфліктних ситуацій.

Усі адміністративні операції виконуються через серверну частину з використанням Service Role Key, що дозволяє обмежити доступ до критичних функцій і запобігти несанкціонованим змінам даних. Такий підхід забезпечує високий рівень безпеки та відповідає принципу мінімально необхідних прав.

Таким чином, реалізація адміністративної панелі та механізмів модерації дозволяє ефективно керувати роботою маркетплейсу, забезпечувати якість контенту та підтримувати безпечне середовище для користувачів. Адміністративний функціонал є важливим елементом масштабованості системи та її готовності до реального використання.

3.7 Інтеграція платіжної системи LiqPay (MVP)

Інтеграція платіжної системи є важливим елементом функціональності маркетплейсу, оскільки забезпечує можливість здійснення фінансових операцій між користувачами. У межах даної кваліфікаційної роботи інтеграцію платіжної системи реалізовано у форматі мінімально життєздатного продукту (MVP) з використанням сервісу LiqPay у тестовому (sandbox) режимі.

Вибір LiqPay обумовлений його поширеністю на території України, наявністю документації та підтримкою інтеграції з веб-застосунками. Використання sandbox-режиму дозволяє продемонструвати повний цикл взаємодії з платіжною системою без здійснення реальних фінансових операцій, що відповідає вимогам навчального проєкту.

Інтеграція платіжного сервісу реалізована через серверну частину застосунку. Такий підхід дозволяє уникнути прямої взаємодії клієнтського інтерфейсу з платіжною системою та зменшує ризик компрометації конфіденційних даних. Сервер формує платіжний запит на основі інформації про замовлення та передає його до LiqPay для подальшої обробки.

Процес оплати починається з ініціації платіжної операції користувачем зі сторінки замовлення. Серверна частина перевіряє коректність даних замовлення, після чого створює платіжну сесію та повертає клієнтській частині необхідні параметри для переходу до платіжної форми LiqPay. У межах MVP підтвердження оплати обробляється умовно, без реального зарахування коштів.

Після завершення платіжного процесу система отримує відповідний результат, на основі якого оновлюється статус замовлення в базі даних. Контроль зміни статусу здійснюється серверною частиною, що запобігає несанкціонованим змінам з боку клієнта. Таким чином, реалізується базовий сценарій взаємодії з платіжною системою.

Реалізація інтеграції у форматі MVP дозволяє зосередитися на архітектурних та функціональних аспектах платіжного процесу, не ускладнюючи систему повною фінансовою логікою. У подальшому дана

інтеграція може бути розширена для підтримки реальних платежів, обробки callback-запитів та автоматичного підтвердження транзакцій.

Таким чином, інтеграція платіжної системи LiqPay у тестовому режимі демонструє можливість розширення маркетплейсу фінансовими функціями та завершує реалізацію повного циклу взаємодії користувачів у межах розроблюваної системи

Висновки до розділу 3

У третьому розділі було описано практичну реалізацію програмної системи маркетплейсу. На основі рішень, прийнятих під час проєктування, було створено основні модулі веб-сервісу: користувацький інтерфейс, серверну частину, механізми автентифікації та авторизації, роботу з базою даних, систему оголошень і замовлень, адміністративну панель та інтеграцію з платіжною системою.

Клієнтську частину реалізовано у вигляді веб-застосунку на основі React. Інтерфейс побудовано з окремих компонентів, що дозволяє зробити код більш зрозумілим і зручним для подальшої підтримки. У системі реалізовано сторінки для перегляду оголошень, створення нових записів, оформлення замовлень, роботи з профілем користувача та виконання адміністративних дій. Використання TypeScript зменшує ризик помилок у коді, а Tailwind CSS забезпечує єдиний стиль інтерфейсу та його адаптивність.

Механізми автентифікації та авторизації реалізовано за допомогою Supabase Auth. Користувач може зареєструватися, увійти до системи та отримати доступ до функцій відповідно до своєї ролі. Після створення облікового запису формується профіль користувача, який використовується для подальшої роботи з оголошеннями, замовленнями та правами доступу. Такий підхід дозволяє розділити публічний функціонал системи та можливості, доступні лише авторизованим користувачам.

Серверна частина реалізована на основі Node.js та Express. Вона забезпечує обробку запитів від клієнтської частини, взаємодію з Supabase, перевірку прав

доступу та виконання бізнес-логіки. REST API виступає проміжним рівнем між користувацьким інтерфейсом і базою даних. Це дозволяє централізовано контролювати обробку даних і зменшити залежність клієнтської частини від внутрішньої структури зберігання інформації.

Робота з базою даних організована через Supabase на основі PostgreSQL. У системі використовуються таблиці для зберігання профілів користувачів, категорій, оголошень, замовлень та адміністративних дій. Така структура відповідає основним бізнес-процесам маркетплейсу та дозволяє зберігати дані у впорядкованому вигляді. Важливо, що база даних не лише зберігає інформацію, а й бере участь у забезпеченні безпеки завдяки механізмам контролю доступу.

Одним із головних результатів реалізації стала система оголошень і замовлень. Користувачі можуть створювати оголошення, додавати до них необхідну інформацію, переглядати доступні пропозиції та оформлювати замовлення. Замовлення пов’язує покупця, продавця та конкретне оголошення, що дозволяє відтворити основний сценарій роботи маркетплейсу. Наявність статусів для оголошень і замовлень дає змогу контролювати їхній поточний стан у системі.

Адміністративна панель реалізована для управління основними елементами системи. Адміністратор може працювати з категоріями, користувачами, оголошеннями та виконувати модерацію контенту. Також передбачено фіксацію адміністративних дій, що є важливим для контролю змін у системі та підвищення прозорості її роботи.

Інтеграцію з платіжною системою LiqPay реалізовано на рівні MVP. Взаємодія з платіжним сервісом відбувається через серверну частину, що є більш безпечним підходом, оскільки конфіденційні параметри не передаються безпосередньо у клієнтський код. Це дозволяє продемонструвати можливість підключення фінансових операцій до системи та залишає простір для подальшого розширення платіжного функціоналу.

Отже, у третьому розділі було реалізовано основну функціональність веб-сервісу. Створений MVP підтверджує працездатність обраної архітектури та

технологічного стеку. Система забезпечує базові сценарії маркетплейсу, підтримує роботу з користувачами, оголошеннями, замовленнями, адміністративними функціями та може бути використана як основа для подальшого розвитку.

РОЗДІЛ 4
ЗАБЕЗПЕЧЕННЯ БЕЗПЕКИ, ТЕСТУВАННЯ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Забезпечення безпеки веб-сервісу

Безпека веб-сервісу є одним із ключових факторів, що визначають надійність та придатність програмної системи до практичного використання. Для маркетплейсів питання безпеки набуває особливої актуальності у зв’язку з обробкою персональних даних користувачів, інформації про замовлення та фінансові операції. Тому під час розробки даної системи питання безпеки розглядалися як на етапі проєктування, так і на етапі реалізації.

Модель безпеки розроблюваного веб-сервісу базується на принципі багаторівневого захисту, відповідно до якого контроль доступу та захист даних реалізуються на кількох рівнях. Такий підхід дозволяє знизити ризики несанкціонованого доступу навіть у разі помилок або вразливостей в окремих компонентах системи.

Першим рівнем безпеки є клієнтський рівень, на якому здійснюється обмеження доступу до функціональних можливостей інтерфейсу залежно від ролі користувача. Наприклад, елементи адміністративної панелі не відображаються для звичайних користувачів. Хоча даний рівень не є основним механізмом захисту, він підвищує зручність користування та зменшує кількість помилкових дій.

Другим рівнем безпеки є серверна частина, яка відповідає за перевірку автентичності користувача, його ролі та статусу перед виконанням будь-яких критичних операцій. Серверна логіка забезпечує контроль коректності вхідних даних, перевірку дозволів на виконання операцій та централізовану обробку помилок. Це дозволяє запобігти виконанню несанкціонованих дій навіть у разі модифікації клієнтських запитів.

Найважливішим рівнем безпеки у розроблюваній системі є рівень бази даних, на якому реалізовано механізм Row Level Security. Політики доступу

визначають, які записи таблиць можуть бути прочитані або змінені конкретним користувачем. Такий підхід гарантує, що навіть у разі прямого доступу до бази даних користувач не зможе отримати інформацію, на яку він не має прав.

Окрему увагу приділено захисту конфігураційних даних і секретів. Усі чутливі параметри, зокрема ключі доступу до Supabase та платіжної системи, зберігаються у змінних середовища та не включаються до клієнтського коду. Service Role Key використовується виключно на серверній стороні, що унеможливлює його витік через клієнтський інтерфейс.

Для зменшення ризиків, пов’язаних із завантаженням файлів, реалізовано обмеження на типи та кількість зображень, які можуть бути додані до оголошень. Зберігання файлів здійснюється у Supabase Storage з відповідними правилами доступу, що запобігає несанкціонованому використанню ресурсів.

Таким чином, система безпеки розроблюваного веб-сервісу побудована на поєднанні клієнтських, серверних і базоданих механізмів захисту. Реалізований підхід забезпечує високий рівень захисту даних, відповідає сучасним вимогам до веб-застосунків та створює основу для подальшого розширення функціональних можливостей системи без зниження рівня безпеки.

4.2 Реалізація Row Level Security та контролю доступу

Однією з ключових особливостей розроблюваного веб-сервісу є реалізація контролю доступу до даних безпосередньо на рівні бази даних із використанням механізму Row Level Security (RLS). Даний підхід дозволяє забезпечити високий рівень безпеки багатокористувацької системи та зменшити залежність захисту даних від прикладної логіки серверної частини.

Row Level Security у PostgreSQL дозволяє визначати політики доступу, які обмежують можливість виконання операцій читання, вставки, оновлення та видалення для окремих рядків таблиці залежно від контексту користувача. У межах даної системи RLS використовується для всіх основних таблиць бази даних, що забезпечує єдину модель доступу до інформації.

Для реалізації RLS у системі використовується механізм автентифікації Supabase, який передає ідентифікатор поточного користувача до контексту виконання запитів. Це дозволяє будувати політики доступу на основі унікального ідентифікатора користувача та його ролі, збереженої у таблиці profiles.

Для таблиці profiles реалізовано політики, які дозволяють користувачеві переглядати та редагувати лише власний профіль. Доступ до профілів інших користувачів обмежено, за винятком користувачів із роллю адміністратора. Такий підхід гарантує конфіденційність персональних даних і запобігає несанкціонованому доступу до інформації.

У таблиці listings політики Row Level Security визначають, що звичайний користувач може створювати оголошення від власного імені та переглядати лише активні оголошення інших користувачів. Редагування або видалення оголошень дозволяється лише їх власникам або адміністраторам системи. Заблоковані користувачі позбавляються можливості створювати нові оголошення, що реалізується через відповідні умови в RLS-політиках.

Для таблиці orders політики доступу обмежують можливість перегляду та зміни замовлень лише для користувачів, які безпосередньо пов’язані з відповідним замовленням, зокрема покупців і продавців. Адміністратори мають доступ до всіх замовлень для здійснення контролю та аналізу роботи системи.

Окрему увагу приділено таблиці admin_audit_log, доступ до якої обмежено виключно користувачами з роллю адміністратора. Запис дій у журнал аудиту здійснюється автоматично під час виконання адміністративних операцій, що дозволяє відстежувати історію змін та підвищує прозорість управління системою.

Реалізація контролю доступу на рівні бази даних дозволяє уникнути дублювання перевірок у серверній логіці та мінімізує ризики витоку даних у разі помилок у коді. Навіть за умови прямого виконання SQL-запитів користувач зможе отримати доступ лише до тих даних, на які він має відповідні права.

Таким чином, використання Row Level Security у поєднанні з рольовою моделлю доступу забезпечує надійний та масштабований механізм захисту даних у веб-сервісі. Реалізований підхід відповідає сучасним вимогам до безпеки веб-застосунків і є однією з ключових переваг розробленої системи.

4.3 Захист даних та конфігураційних параметрів системи

Захист даних та конфігураційних параметрів є важливою складовою загальної моделі безпеки веб-сервісу, оскільки витік конфіденційної інформації або секретних ключів може призвести до повної компрометації системи. У межах розроблюваного маркетплейсу даному аспекту приділено окрему увагу як на етапі проєктування, так і під час реалізації.

Персональні дані користувачів, зокрема контактна інформація, адреси доставки та історія замовлень, зберігаються у реляційній базі даних PostgreSQL. Доступ до цих даних обмежений за допомогою політик Row Level Security, що забезпечує їхню конфіденційність і цілісність. Кожен користувач має доступ виключно до власних даних, тоді як адміністратори можуть переглядати інформацію в межах виконання службових обов’язків.

Окрему увагу приділено захисту конфігураційних параметрів та секретних ключів. Усі чутливі дані, такі як ключі доступу до Supabase, Service Role Key та параметри інтеграції платіжної системи, зберігаються у змінних середовища. Для цього використовується файл .env, який не включається до репозиторію проєкту. Натомість у репозиторії зберігається файл .env.example, що містить перелік необхідних змінних без їх фактичних значень.

Service Role Key використовується виключно у серверній частині застосунку та недоступний клієнтському інтерфейсу. Це дозволяє виконувати адміністративні операції з підвищеними правами без ризику компрометації ключа через браузер або клієнтський код. Клієнтська частина взаємодіє з Supabase лише з використанням публічних ключів, що мають обмежені права доступу.

Для захисту даних під час передачі між клієнтом і сервером використовується протокол HTTPS, який забезпечує шифрування трафіку та захист від перехоплення інформації. Це є особливо важливим при передачі токенів автентифікації та даних замовлень.

Зберігання файлів, зокрема зображень оголошень, здійснюється у Supabase Storage. Для кожного bucket налаштовано правила доступу, які обмежують можливість завантаження та перегляду файлів залежно від ролі користувача. Крім того, реалізовано перевірки типів файлів і обмеження кількості зображень, що зменшує ризики зловживань та перевантаження сховища.

Таким чином, захист даних та конфігураційних параметрів у розроблюваній системі реалізовано з урахуванням принципів мінімально необхідних прав, ізоляції секретів та використання сучасних механізмів шифрування. Застосовані підходи забезпечують надійність і безпеку веб-сервісу та відповідають сучасним вимогам до розробки веб-застосунків.

4.4 Тестування функціональних можливостей системи

Тестування є необхідним етапом розробки програмної системи, оскільки дозволяє перевірити коректність реалізації функціональних вимог, виявити помилки та оцінити стабільність роботи веб-сервісу. У межах даної кваліфікаційної роботи тестування виконувалося з метою перевірки працездатності основних модулів маркетплейсу та відповідності реалізованого функціоналу поставленим задачам.

Тестування системи проводилося у вигляді функціонального та інтеграційного тестування, що є доцільним для веб-застосунків з клієнт-серверною архітектурою. Основна увага приділялася перевірці ключових бізнес-процесів, а також механізмів безпеки та контролю доступу.

Під час тестування було перевірено роботу механізмів реєстрації та автентифікації користувачів. Тестові сценарії включали створення нового облікового запису, виконання входу в систему, завершення сесії та повторну

автентифікацію. Окремо перевірялася коректність створення записів у таблиці profiles після реєстрації користувача.

Наступним етапом тестування була перевірка функціональності системи оголошень. Було протестовано створення, редагування та перегляд оголошень, а також коректність відображення лише активних оголошень для незареєстрованих користувачів. Особливу увагу приділено перевірці обмежень для заблокованих користувачів, які не повинні мати можливості створювати нові оголошення.

У межах тестування системи замовлень перевірено сценарії оформлення замовлення, збереження даних доставки та відображення замовлень у відповідних розділах користувацького інтерфейсу. Було перевірено, що покупець має доступ лише до власних замовлень, а продавець - лише до замовлень, пов’язаних із його оголошеннями.

Окремо проведено тестування адміністративного функціоналу, зокрема управління категоріями, модерації оголошень та блокування користувачів. Перевірено, що доступ до адміністративної панелі мають лише користувачі з роллю адміністратора, а всі адміністративні дії коректно фіксуються у журналі аудиту.

Важливим аспектом тестування стала перевірка механізмів безпеки, зокрема політик Row Level Security. Було перевірено, що прямі запити до бази даних не дозволяють отримати доступ до даних інших користувачів, навіть у разі спроби виконання несанкціонованих операцій. Це підтвердило коректність реалізації контролю доступу на рівні бази даних.

Таким чином, результати тестування підтверджують коректність реалізації основних функціональних можливостей веб-сервісу та відповідність системи визначеним вимогам. Проведене тестування дозволяє зробити висновок про готовність розробленого програмного рішення до демонстрації та подальшого використання.

4.5 Економічне обґрунтування розробки програмного продукту

Економічне обґрунтування кваліфікаційної роботи дозволяє оцінити доцільність розробки програмного продукту з точки зору витрат ресурсів та потенційної практичної користі. У межах даної роботи економічна оцінка проводиться в узагальненому вигляді, що відповідає навчальному характеру проєкту.

Розробка програмної системи здійснювалася з використанням відкритих і безкоштовних технологій, зокрема React, Node.js, TypeScript, PostgreSQL та Supabase. Це дозволило суттєво знизити витрати на ліцензійне програмне забезпечення та зробити проєкт економічно доцільним для використання малими та середніми підприємствами.

Основними витратами під час створення програмного продукту є:

- витрати часу розробника на аналіз, проєктування та реалізацію системи;
- витрати на хостинг серверної та клієнтської частин;
- витрати на зберігання даних та файлів.

Для розгортання системи використано хмарні сервіси Vercel, Render та Supabase, які надають безкоштовні або умовно безкоштовні тарифні плани. Це дозволяє експлуатувати систему з мінімальними фінансовими витратами на початковому етапі та масштабувати її у разі зростання навантаження.

Модульна архітектура веб-сервісу забезпечує можливість повторного використання програмного рішення для реалізації інших бізнес-моделей, що знижує вартість розробки нових продуктів на його основі. Крім того, централізована реалізація безпеки на рівні бази даних зменшує витрати на підтримку та знижує ризики фінансових втрат унаслідок витоку даних.

Таким чином, розроблений програмний продукт є економічно обґрунтованим, оскільки поєднує низькі витрати на реалізацію з можливістю подальшого практичного використання та масштабування.

4.6 Огляд використаних протоколів безпеки

У розробленій системі з модульною архітектурою безпека забезпечується на декількох рівнях: під час передавання даних між клієнтом і сервером, під час автентифікації користувачів, під час доступу до ресурсів бази даних, а також під час інтеграції з платіжним сервісом. Оскільки застосунок є маркетплейсом, особлива увага приділяється захисту персональних даних користувачів, контролю доступу до оголошень і замовлень, а також перевірці платіжних операцій.

Основними протоколами та механізмами безпеки, що використовуються у проєкті, є HTTPS, JWT Bearer Authentication, CORS, Row Level Security у PostgreSQL/Supabase, а також механізм цифрового підпису LiqPay.

HTTPS

Для захищеної взаємодії між клієнтською частиною, сервером, Supabase та платіжною системою передбачається використання протоколу HTTPS. HTTPS є розширенням HTTP, яке працює поверх TLS і забезпечує шифрування даних під час передавання мережею.

Завдяки HTTPS захищаються такі дані:

- email та пароль під час входу або реєстрації;
- JWT-токени користувачів;
- дані оголошень і замовлень;
- контактна інформація користувачів;
- платіжні дані, що передаються до LiqPay.

У локальному середовищі застосунок може працювати через http://localhost, однак у production-розгортанні використання HTTPS є обов’язковим. Без HTTPS зловмисник потенційно може перехопити access token або інші чутливі дані.

JWT Bearer Authentication

Для автентифікації користувачів використовується Supabase Auth. Після успішного входу користувач отримує session, яка містить JWT access token. Цей токен frontend додає до запитів на backend у заголовку:

Authorization: Bearer <access_token>

На сервері middleware перевіряє токен через Supabase Auth. Якщо токен відсутній, недійсний або користувач заблокований, сервер повертає помилку 401 Unauthorized або 403 Forbidden.

Такий підхід дозволяє:

- не зберігати паролі у власній базі застосунку;
- ідентифікувати користувача в кожному запиті;
- обмежувати доступ до захищених маршрутів;
- перевіряти роль користувача, наприклад user або admin;
- блокувати користувачів без видалення їхніх облікових записів.

JWT використовується не тільки для перевірки на backend, а й для запитів до Supabase від імені конкретного користувача. Це дозволяє застосовувати політики доступу на рівні бази даних.

CORS

Для обмеження доступу до API використовується механізм CORS. Backend налаштований так, щоб приймати запити лише з дозволеного frontend-домену, який задається змінною середовища WEB_ORIGIN.

Це зменшує ризик того, що сторонній сайт зможе напряму виконувати запити до API від імені користувача через браузер. CORS не є повноцінною заміною автентифікації, але є важливим додатковим рівнем захисту для web-застосунків.

Row Level Security

Одним із ключових механізмів безпеки є Row Level Security у Supabase/PostgreSQL. RLS дозволяє задавати правила доступу не тільки на рівні API, а безпосередньо на рівні рядків таблиць бази даних.

У проєкті RLS увімкнено для основних таблиць:

- profiles;

- categories;
- listings;
- orders;
- admin_audit_log.

Наприклад, користувач може переглядати тільки власний профіль, власні оголошення або замовлення, пов’язані з ним. Адміністратор має ширший доступ, але він визначається через роль у таблиці profiles.

Для оголошень діє правило: публічно доступні лише активні оголошення, власник яких не заблокований. Заблоковані або архівні оголошення не відображаються звичайним користувачам.

Цей механізм важливий тим, що навіть у випадку помилки на рівні frontend або backend база даних додатково перевіряє, чи має користувач право на доступ до конкретного запису.

Контроль ролей

У системі реалізовано рольову модель доступу. Користувач може мати роль user або admin.

Звичайний користувач може:

- створювати оголошення;
- редагувати власні оголошення;
- створювати замовлення;
- переглядати власні замовлення;
- переглядати продажі за своїми оголошеннями.

Адміністратор може:

- керувати категоріями;
- переглядати користувачів;
- блокувати або розблоковувати користувачів;
- змінювати статуси оголошень;
- переглядати адміністративні дані.

Доступ до адміністративних маршрутів перевіряється як на frontend, так і на backend. Frontend приховує сторінки адміністратора від звичайних

користувачів, але основний захист виконується саме на сервері через middleware requireAdmin.

Захист платіжної інтеграції LiqPay

Для оплати замовлень використовується LiqPay. Backend формує платіжний payload, який містить суму, валюту, опис платежу, ідентифікатор замовлення, URL повернення та URL серверного callback.

Для захисту платежу використовується цифровий підпис. Дані платежу кодуються у Base64, після чого формується підпис за допомогою приватного ключа LiqPay:

base64(sha1(private_key + data + private_key))

Frontend отримує від backend тільки готові поля data і signature, після чого перенаправляє користувача на сторінку LiqPay Checkout. Приватний ключ LiqPay зберігається тільки на backend і не передається у браузер.

Після завершення платежу LiqPay надсилає callback на сервер. Backend повторно обчислює підпис і порівнює його з отриманим. Якщо підпис не збігається, запит відхиляється. Це дозволяє перевірити, що callback справді надійшов від LiqPay, а не був підроблений сторонньою особою.

Валідація вхідних даних

Для перевірки вхідних даних на backend використовується бібліотека Zod. Вона дозволяє перевіряти структуру та типи даних перед виконанням операцій у базі.

Перевіряються такі дані:

- email і пароль;
- назва оголошення;
- ціна;
- UUID категорії або замовлення;
- кількість товарів;
- URL зображень;
- контактний телефон;
- статус оголошення.

Це зменшує ризик некоректних запитів, помилок у базі даних та частково захищає систему від спроб передати неочікувані або шкідливі дані.

Зберігання секретних ключів

У проєкті використовуються змінні середовища для зберігання чутливих даних:

- SUPABASE_SERVICE_ROLE_KEY;
- SUPABASE_ANON_KEY;
- LIQPAY_PUBLIC_KEY;
- LIQPAY_PRIVATE_KEY;
- WEB_ORIGIN;
- API_BASE_URL.

Особливо важливим є SUPABASE_SERVICE_ROLE_KEY, оскільки він має розширені права доступу до Supabase і може обходити RLS-політики. Тому він використовується тільки на backend і не повинен потрапляти у frontend-код або публічний репозиторій.

У системі використовується багаторівнева модель безпеки. Передавання даних має виконуватися через HTTPS, користувачі автентифікуються за допомогою JWT-токенів, доступ до API контролюється middleware та ролями, а доступ до записів бази даних додатково обмежується RLS-політиками. Платіжна інтеграція з LiqPay захищена механізмом цифрового підпису, що дозволяє перевіряти справжність платіжних callback-запитів.

Таким чином, безпека проєкту не залежить лише від одного механізму, а реалізована комплексно: на рівні клієнта, сервера, бази даних і зовнішньої платіжної системи.

Висновки до розділу 4

У четвертому розділі було розглянуто питання безпеки, тестування та економічної доцільності розробки веб-сервісу. Цей етап є важливим, оскільки після реалізації основної функціональності необхідно оцінити, наскільки

система є захищеною, працездатною та придатною для подальшого використання або розвитку.

Безпека системи розглядається на кількох рівнях. На рівні клієнтської частини обмежується доступ до окремих сторінок і функцій залежно від ролі користувача. На рівні серверної частини перевіряється автентичність користувача, його роль і право виконувати певну дію. Додатковий рівень захисту реалізовано на рівні бази даних за допомогою Row Level Security. Такий підхід є більш надійним, ніж використання лише перевірок у програмному коді, оскільки правила доступу застосовуються безпосередньо до даних.

Механізм Row Level Security є одним із ключових елементів захисту системи. Завдяки йому користувач може працювати лише з тими записами, до яких має право доступу. Наприклад, звичайний користувач може переглядати власні замовлення та керувати власними оголошеннями, тоді як адміністратор має ширші можливості для модерації й управління системою. Це дозволяє зменшити ризик несанкціонованого доступу до даних і підвищити рівень довіри до системи.

Також було враховано питання захисту конфігураційних параметрів. Секретні ключі, параметри доступу до Supabase та дані платіжної системи не повинні зберігатися у відкритому вигляді в репозиторії або клієнтському коді. Для цього використовуються змінні середовища, а в кодовій базі може зберігатися лише приклад конфігураційного файлу без реальних значень. Такий підхід є обов’язковим для безпечної розробки веб-застосунків, особливо якщо система взаємодіє з платіжними сервісами або обробляє дані користувачів.

У розділі було проведено тестування основних функціональних можливостей системи. Перевірялися сценарії реєстрації та входу користувача, створення профілю, перегляд і створення оголошень, оформлення замовлень, робота адміністративної панелі та обмеження доступу для різних ролей. Результати тестування підтвердили, що реалізовані модулі виконують свої функції відповідно до поставлених вимог. Також було перевірено, що користувачі не можуть виконувати дії, які не відповідають їхнім правам доступу.

Економічне обґрунтування показало, що розробка такого програмного продукту є доцільною з погляду витрат і подальшого використання. Обраний технологічний стек складається переважно з відкритих або безкоштовних інструментів, що дозволяє зменшити витрати на ліцензійне програмне забезпечення. Використання Supabase, React, Node.js, TypeScript та інших сучасних технологій дає можливість швидко створити MVP і в подальшому розвивати його залежно від потреб бізнесу.

Окремо було розглянуто протоколи та механізми безпеки, які використовуються в системі. До них належать HTTPS для захищеної передачі даних, JWT Bearer Authentication для автентифікації користувачів, CORS для контролю доступу між ресурсами, Row Level Security для захисту даних на рівні бази, а також цифровий підпис LiqPay для перевірки платіжних операцій. Їх поєднання дозволяє забезпечити базовий рівень захисту, необхідний для веб-сервісу електронної комерції.

Отже, у четвертому розділі було підтверджено, що розроблена система не лише реалізує основний функціонал маркетплейсу, а й має необхідні механізми безпеки, проходить базове функціональне тестування та є економічно обґрунтованою. Реалізований веб-сервіс може бути використаний як основа для подальшого розвитку, розширення функціональності та адаптації під різні бізнес-моделі.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було спроєктовано та реалізовано модульну систему веб-сервісу з можливістю розширення під різні бізнес-моделі. Розроблений програмний продукт являє собою повнофункціональний маркетплейс з адміністративною панеллю та підтримкою основних безпекових протоколів та бізнес-процесів електронної комерції.

Відповідно до поставленої мети та визначених завдань отримано такі результати:

1. У межах першого завдання було здійснено аналіз нормативно-правових актів України у сфері електронної комерції, захисту персональних даних та інформаційної безпеки, а також розглянуто сучасні наукові підходи та технології розробки вебзастосунків. Це дозволило сформувати вимоги до програмної системи та визначити основні напрями її реалізації.

2. У межах другого завдання проведено аналіз предметної області вебмаркетплейсів та існуючих програмних рішень, визначено ключові бізнес-процеси та функціональні вимоги до системи. На основі проведеного аналізу спроєктовано архітектуру вебсервісу, структуру бази даних і моделі доступу користувачів, а також обґрунтовано вибір технологічного стеку та організацію монорепозиторію.

3. У межах третього завдання розроблено програмну систему у форматі мінімально життєздатного продукту (MVP), що включає клієнтську та серверну частини, REST API, систему управління оголошеннями та замовленнями, адміністративну панель з функціями модерації контенту та журналом аудиту, а також інтеграцію платіжної системи.

4. У межах четвертого завдання забезпечено захист даних користувачів шляхом застосування сучасних механізмів контролю доступу, зокрема технології Row Level Security на рівні бази даних. Проведене тестування підтвердило коректність роботи основних функціональних модулів системи та її відповідність поставленим вимогам.

Отримані результати дають підстави зробити наступні висновки, що розроблений веб-сервіс може бути використаний як основа для подальшого розвитку та адаптації під інші бізнес-моделі, а також має практичну цінність для використання в реальних проєктах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. Видавництво “ДП «УкрНДНЦ»”, 2016 - 26 с.

2. ISO 5807:2016. Обробляння інформації. Символи та угоди щодо документації стосовно даних, програм та системних блоксхем. Видавництво “ДП «УкрНДНЦ»”, 2017 - 34 с.

3. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні вимоги та правила складання. Видавництво “ДП «УкрНДНЦ»”, 2016 - 16 с.

4. Sommerville I. Software Engineering 10th ed. Edition. Видавництво “Pearson Education”, 2016 - 816 с.

5. Bass L., Clements P., Kazman R. Software Architecture in Practice 3rd ed. Edition. Видавництво “Addison-Wesley”, 2013 - 624 с.

6. Richards M., Ford N. Fundamentals of Software Architecture. Видавництво “O’Reilly Media”, 2020 - 448 с.

7. Newman S. Building Microservices. Видавництво “O’Reilly Media”, 2015 - 280 с.

8. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : doctoral dissertation. Видавництво “University of California, Irvine”, 2000 - 180 с.

9. React Documentation [Електронний ресурс]. - режим доступу: https://react.dev (дата звернення: 31.03.2026).

10. Vite Documentation [Електронний ресурс]. - режим доступу: https://vitejs.dev (дата звернення: 11.03.2026).

11. TypeScript Documentation [Електронний ресурс]. - режим доступу: https://www.typescriptlang.org (дата звернення: 31.03.2026).

12. Node.js Documentation [Електронний ресурс]. - режим доступу: https://nodejs.org (дата звернення: 11.03.2026).

13. Express.js Documentation [Електронний ресурс]. - режим доступу: https://expressjs.com (дата звернення: 29.03.2026).

14. PostgreSQL Documentation [Електронний ресурс]. - режим доступу: https://www.postgresql.org/docs (дата звернення: 31.03.2026).

15. Supabase Documentation [Електронний ресурс]. - режим доступу: https://supabase.com/docs (дата звернення: 17.03.2026).

16. PostgreSQL Row Level Security [Електронний ресурс]. - режим доступу: https://www.postgresql.org/docs/current/ddl-rowsecurity.html (дата звернення: 28.01.2026).

17. OWASP Foundation. OWASP Top 10 Web Application Security Risks [Електронний ресурс]. - режим доступу: https://owasp.org/www-project-top-ten (дата звернення: 28.01.2026).

18. Tailwind CSS Documentation [Електронний ресурс]. - режим доступу: https://tailwindcss.com/docs (дата звернення: 28.02.2026).

19. TanStack Query Documentation [Електронний ресурс]. - режим доступу: https://tanstack.com/query (дата звернення: 01.02.2026).

20. LiqPay API Documentation [Електронний ресурс]. - режим доступу: https://www.liqpay.ua/documentation (дата звернення: 23.03.2026).

21. D'Adamo I., González-Sánchez R., Medina-Salgado M. S., Settembre-Blundo D. E-Commerce Calls for Cyber-Security and Sustainability. Sustainability. 2021. Vol. 13, No. 12. Article 6752. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/2071-1050/13/12/6752 (дата звернення: 23.05.2026).

22. Liu X., Ahmad S. F., Anser M. K., Ke J., Irshad M., Ul-Haq J., Abbas S. Cyber security threats: A never-ending challenge for e-commerce. Frontiers in Psychology. 2022. Vol. 13. Article 927398. [Електронний ресурс]. - режим доступу: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2022.927398/full (дата звернення: 23.05.2026).

23. Shahid J., Hameed I., Javed I. T., Qureshi K. N., Crespi N. A Comparative Study of Web Application Security Parameters: Current Trends and Future Directions. Applied Sciences. 2022. Vol. 12, No. 8. Article 4077. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/2076-3417/12/8/4077 (дата звернення: 23.05.2026).

24. Althunayyan M., Saxena N., Khorsandroo S. Evaluation of Black-Box Web Application Security Scanners in Detecting Injection Vulnerabilities. Electronics. 2022. Vol. 11, No. 13. Article 2049. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/2079-9292/11/13/2049 (дата звернення: 23.05.2026).

25. Aydos M., Aldan Ç., Coşkun E., Soydan A. Security testing of web applications: A systematic mapping of the literature. Journal of King Saud University - Computer and Information Sciences. 2022. Vol. 34, No. 9. C. 6775-6792. [Електронний ресурс]. - режим доступу: https://www.sciencedirect.com/science/article/pii/S131915782100269X (дата звернення: 23.05.2026).

26. Saeed S., Almuhaideb A. M., Kumar N., Islam A. K. M. N. A Customer-Centric View of E-Commerce Security and Privacy. Applied Sciences. 2023. Vol. 13, No. 2. Article 1020. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/2076-3417/13/2/1020 (дата звернення: 23.05.2026).

27. Munshi A., Yasser A., Halim Z., Aljohani N. R. An Electronic Commerce Big Data Analytics Architecture and Platform. Applied Sciences. 2023. Vol. 13, No. 19. Article 10962. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/2076-3417/13/19/10962 (дата звернення: 23.05.2026).

28. Morić Z., Bajić B., Radošević D. Protection of Personal Data in the Context of E-Commerce. Digital. 2024. Vol. 4, No. 3. C. 715-728. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/2624-800X/4/3/34 (дата звернення: 23.05.2026).

29. Sandu A., Cotfas L. A., Delcea C., Chiriță N. E-Commerce Meets Emerging Technologies: An Overview of Recent Trends and Future Directions. Journal of Theoretical and Applied Electronic Commerce Research. 2025. Vol. 20, No.

4. Article 320. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/0718-1876/20/4/320 (дата звернення: 23.05.2026).

30. Papastamoulou P., Vasilopoulos A., Kavoura A. Artificial Intelligence in E-Commerce: A Comparative Study of Digital Commerce Platforms. Systems. 2025. Vol. 13, No. 9. Article 746. [Електронний ресурс]. - режим доступу: https://www.mdpi.com/2079-8954/13/9/746 (дата звернення: 23.05.2026).

31. Бондарчук, А., Жебка, В., Корецька, В., & Шилкіна, А. (2024). Порівняльна характеристика web-орієнтованих інструментів автоматизації освітнього процесу в умовах цифрової трансформації. Публічно-управлінські та цифрові практики, (1), 13-21.

32. Abramov, V., Astafieva, M., Boiko, M., Bodnenko, D., Bushma, A., Vember, V., ... & Yaskevych, V. (2021). Theoretical and practical aspects of the use of mathematical methods and information technology in education and science.

33. Liudmyla Ilich, Oksana Hlushak, Svetlana Semenyaka, Yaroslav Shestack «Modeling of Structural Changes in the Employment as the Direction of Economic Security Risk Management» // Cybersecurity Providing in Information and Telecommunication Systems, 2023, 3421. p p. 46-55.

34. Баранник Б. К., Вембер В. П. Модульна система веб-сервісу з можливістю розширення під бізнес-моделі. Збірник матеріалів Всеукраїнської конференції молодих учених “Інформаційні технології”. 2026. № 13. С. 60-64. [Електронний ресурс]. - режим доступу: https://zcit.kubg.edu.ua/index.php/journal/article/view/72 (дата звернення: 23.05.2026).

35. Машкіна, І. В., Абрамов, В. О., Носенко, Т. І., Іваніченко, Є. В., & Яскевич, В. О. (2024). Навчально-методичний посібник до виконання і захисту кваліфікаційної роботи бакалавра спеціальностей 122 Комп’ютерні науки для здобувачів вищої освіти денної форми навчання.

Додаток А

Структура монорепозиторію.