

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Інтернет-магазин з продажу оргтехніки»

Виконав студент групи ПЗ-22-3
БАРАНІВСЬКИЙ Назар Михайлович

Керівник роботи:
БАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЛЕВКІВСЬКИЙ Віталій Леонідович

Житомир – 2026

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **БАРАНІВСЬКИЙ Назар Михайлович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: **«Інтернет-магазин з продажу оргтехніки»**,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., №92/с**

Вихідні дані для роботи: **об'єктом дослідження є інтернет-магазин оргтехніки**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	01.03.2026р.	01.03.2026р
2	Вакалюк Т.А.	01.03.2026р	01.03.2026р
3	Вакалюк Т.А.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з веб-платформи «Інтернет-магазин з продажу оргтехніки» та пояснювальної записки. Пояснювальна записка містить 65 сторінок, 24 ілюстрації, 15 таблиць, 5 додатків та 31 використане джерело.

Метою роботи є створення інтернет-магазину оргтехніки зі зручним інтерфейсом, зручного вибору товарів, інтеграцією способів оплати та доставки, а також системою керування магазином. Інтернет-магазин надає користувачам можливість пошуку товарів за категоріями, ціною, технічними характеристиками, а також швидкого оформлення замовлення.

У реалізації проєкту використано фреймворк FastAPI для серверної логіки, React – клієнтської частини, PostgreSQL – зберігання даних, API Нової пошти – для доставки, платіжну систему LiqPay – онлайн-оплати. Проєкт розроблявся у середовищі Visual Studio Code, для локального розгортання використовується 2 сервери – Vite для фронтенду та Uvicorn для бекенду. Взаємодія між фронтендом і бекендом відбувається через REST API.

КЛЮЧОВІ СЛОВА: ІНТЕРНЕТ-МАГАЗИН, ОРГТЕХНІКА, REACT, FASTAPI, POSTGRESQL.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазину з продажу оргтехніки	Літ.	Арк.	Аркушів
Розроб.		Баранівський Н.М.						
Керівник		Вакалюк Т.А.					3	87
Рецензент		Левківський В.Л.				Житомирська політехніка, група ІПЗ-22-3		
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

This bachelor's thesis consists of a web platform titled “Online Store for Office Equipment” and an explanatory note. The explanatory note contains 65 pages, 24 illustrations, 15 tables, 5 appendices, and 31 references.

The goal of the project is to create an office equipment online store with a user-friendly interface, convenient product selection, integrated payment and delivery methods, and a user management system. The online store allows users to search for products by category, price, and technical specifications, as well as to quickly place an order.

The project implementation uses the FastAPI framework for server-side logic, React for the client-side, PostgreSQL for data storage, the Nova Post API for delivery, and the LiqPay payment system for online payments. The project was developed in the Visual Studio Code environment; two servers are used for local deployment — Vite for the frontend and Uvicorn for the backend. Communication between the frontend and backend occurs via REST API.

KEYWORDS: ONLINE STORE, OFFICE EQUIPMENT, REACT, FASTAPI, POSTGRESQL.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ.....	9
1.1. Постановка задачі.....	9
1.2. Аналіз аналогів програмного продукту.....	11
1.3. Вибір архітектури веб орієнтованої системи	14
1.4. Вибір інструментальних засобів та вимоги до системи	17
Висновки до першого розділу.....	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ОРГТЕХНІКИ.....	23
2.1. Визначення варіантів використання інтернет магазину оргтехніки.....	23
2.2. Розробка бази даних системи	28
2.3. Проєктування та реалізація алгоритмів роботи системи	36
2.4. Реалізація функціоналу інтернет-магазину оргтехніки.....	40
Висновки до другого розділу	43
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ ОРГТЕХНІКИ.....	45
3.1. Порядок встановлення та налаштування параметрів системи	45
3.2. Структура інтерфейсу інтернет-магазину оргтехніки.....	48
3.3. Тестування роботи інтернет-магазину оргтехніки	55
Висновки до третього розділу.....	59
ВИСНОВКИ	61
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	62
ДОДАТКИ.....	65

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База Даних

СКБД – Система Керування Базами Даних

API – Application Programming Interface

ASGI – Asynchronous Server Gateway Interface

HTTP – Hypertext Transfer Protocol

JWT – JSON Web Token

JSON – JavaScript Object Notation

ORM – Object-Relational Mapping

REST – REpresentational State Transfer

SPA – Single Page Application

URL – Uniform Resource Locator

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Розвиток електронної комерції є одним із найважливіших напрямів застосування інформаційних технологій у сучасній торгівлі та автоматизації бізнес-процесів. Інтернет-магазини дозволяють підприємствам ефективно представляти товари в електронному каталозі, забезпечувати покупцям цілодобовий доступ до актуальної інформації про продукцію, вести облік продажів та автоматизувати обробку замовлень. Для кінцевого споживача такі системи створюють можливість зручно й швидко знаходити потрібну продукцію, порівнювати ціни та оформлювати купівлю без відвідування фізичних точок продажу.

Особливо актуальним є використання веб орієнтованих систем у сфері дистрибуції оргтехніки: принтерів, сканерів, багатофункціональних пристроїв, комп'ютерної периферії та витратних матеріалів. Оскільки ці товари мають складні технічні характеристики та специфічні параметри сумісності, для них необхідна побудова якісної інформаційної системи із чітким структуруванням категорій, надійним пошуком та зрозумілим процесом взаємодії. Актуальність теми випускної кваліфікаційної роботи бакалавра зумовлена потребою у створенні сучасного веб застосунку, який повністю автоматизує цикл роздрібної онлайн-торгівлі офісним обладнанням, включаючи інтеграцію платіжного сервісу та служби доставки.

Метою випускної кваліфікаційної роботи є розробка веб орієнтованої системи інтернет-магазину оргтехніки з гнучким керуванням контентом, автоматизованою системою замовлень та безпечним розмежуванням прав доступу користувачів.

Для досягнення поставленої мети необхідно виконати такі завдання:

- провести аналіз предметної області та дослідити функціональні можливості наявних аналогів на ринку;
- обґрунтувати вибір архітектури системи, обрати інструментальні засоби розробки;

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

- спроектувати реляційну структуру бази даних для надійного зберігання інформації;
- програмно реалізувати серверний API системи та динамічний клієнтський інтерфейс;
- інтегрувати зовнішні сервіси для забезпечення онлайн-оплати та отримання актуальних логістичних даних для доставки;
- виконати тестування розробленого програмного продукту.

Об’єктом дослідження є процес автоматизації торгівлі оргтехнікою засобами веб орієнтованих інформаційних систем.

Предметом дослідження є методи, моделі та технології розробки інтернет-магазину з використанням SPA архітектури на базі React, асинхронного фреймворку FastAPI, реляційної бази даних PostgreSQL та інтеграційних інтерфейсів API сторонніх сервісів.

Під час виконання роботи було використано такі методи дослідження: системний аналіз – для вивчення предметної області та визначення вимог до системи; об’єктно-орієнтоване проєктування – для розробки логічної структури додатка; моделювання баз даних – для побудови схем реляційних таблиць; методи веб програмування – для безпосередньої реалізації клієнтської та серверної частин; а також методи функціонального тестування – для перевірки коректності роботи програми.

Практичне значення одержаних результатів полягає у створенні працездатного інтернет-магазину оргтехніки малого або середнього масштабу. Розроблений програмний продукт забезпечує повний цикл електронної комерції: від перегляду каталогу й пошуку товарів до безпечної авторизації, ведення кошика, оформлення замовлення з вибором відділення Нової Пошти та здійснення онлайн-оплати через LiqPay. Наявність вбудованої адміністративної панелі дозволяє адміністратору керувати товарами, категоріями, користувачами та замовленнями, що робить систему готовим рішенням для практичного комерційного використання з можливістю подальшого масштабування.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1. Постановка задачі

Розроблення інтернет-магазину оргтехніки передбачає створення веб-орієнтованої системи, яка автоматизує основні процеси онлайн-продажу офісного обладнання, комп'ютерної периферії та супутніх товарів. Такий програмний продукт має забезпечувати зручну взаємодію покупця з каталогом, підтримувати роботу з кошиком і замовленнями, надавати засоби онлайн-оплати та доставки, а також містити адміністративну частину для керування магазином.

Предметна область продажу оргтехніки має низку особливостей, які необхідно врахувати під час проєктування системи. До асортименту магазину можуть входити принтери, сканери, багатофункціональні пристрої, витратні матеріали та інші товари для офісної роботи. Кожна група продукції має власні технічні характеристики, ціновий діапазон, артикул, наявність на складі та набір параметрів, важливих для покупця. Через це каталог повинен мати чітку структуру, підтримувати поділ товарів за категоріями та давати змогу швидко переходити до потрібної групи товарів.

Для роботи з каталогом потрібно передбачити пошук і фільтрацію. Пошук має виконуватися за назвою товару та артикулом, оскільки покупці часто шукають як конкретну модель, так і позицію за внутрішнім або виробничим кодом. Фільтрація повинна враховувати ціну, наявність і технічні характеристики, які залежать від категорії. Наприклад, для принтерів можуть бути важливими тип друку, формат паперу або підтримка кольорового друку, а для моніторів — діагональ, роздільна здатність і тип матриці. Такий підхід дає змогу не перевантажувати користувача однаковими параметрами для всіх товарів і водночас забезпечує точніший підбір потрібного товару.

Сторінка товару має надавати повну інформацію про обрану позицію. На ній необхідно відображати зображення, назву, артикул, вартість, залишок на складі, опис і набір характеристик. З цієї сторінки покупець повинен мати можливість додати товар до кошика та перейти до подальшого оформлення

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

замовлення. Наявність детального подання продукції є важливою, оскільки користувач приймає рішення не лише за ціною, а й за технічними параметрами та сумісністю з власними потребами.

Кошик повинен виконувати роль проміжного етапу між вибором товарів і створенням замовлення. Там користувач переглядає перелік доданих товарів, за потреби, змінює кількість товару, видаляє непотрібні та бачить підсумкову вартість. Після перевірки вмісту кошика користувач переходить до оформлення замовлення. Така послідовність дій робить процес купівлі зрозумілим і дозволяє уникнути випадкового придбання зайвих товарів.

Під час оформлення замовлення користувач повинен вказати контактні дані, обрати спосіб оплати та заповнити інформацію для доставки. У системі реалізовано два способи оплати: готівкою та онлайн через платіжний сервіс LiqPay. У разі вибору онлайн-оплати покупець переходить на сторінку платіжного сервісу, після чого повертається до відповідної сторінки інтернет-магазину залежно від результату операції. Якщо оплата пройшла успішно, користувач отримує повідомлення про успішне замовлення. Якщо оплата не відбулася, система повинна повідомити про помилку та надати можливість повторити спробу або повернутися до головної сторінки.

У системі необхідно передбачити два типи користувачів: покупця та адміністратора. Покупець працює з каталогом, фільтрами, пошуком, кошиком, оформленням замовлень і особистим кабінетом. Адміністратор отримує доступ до панелі керування, через яку може працювати з категоріями, товарами, користувачами та замовленнями. Такий розподіл ролей дозволяє відокремити звичайні дії покупця від операцій, що впливають на наповнення магазину та обробку замовлень.

Для зберігання інформації необхідно розробити реляційну базу даних. У ній мають бути передбачені таблиці для користувачів, категорій, товарів, зображень, кошиків, елементів кошика, замовлень та елементів замовлення. Зв'язки між цими сутностями повинні відображати логіку роботи інтернет-магазину. Товар належить до певної категорії, кошик містить вибрані позиції,

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

замовлення складається з окремих товарів, а користувач може мати історію покупок.

Під час реалізації системи потрібно врахувати вимоги до безпеки. Паролі користувачів мають зберігатися у захищеному вигляді, а доступ до приватних і адміністративних дій повинен надаватися лише після авторизації. Для цього необхідно використовувати ролі, захищені маршрути та перевірку прав доступу.

У результаті виконання поставлених задач має бути отриманий інтернет-магазин оргтехніки, який охоплює основні процеси електронної торгівлі.

1.2. Аналіз аналогів програмного продукту

Перед початком розробки інтернет-магазину оргтехніки, слід провести аналіз наявних рішень на ринку. Це дозволить визначити їх сильні та слабкі сторони, та перейняти найкращі рішення у власну розробку. В якості аналогічних продуктів були обрані наступні доступні рішення на ринку: Rozetka - найбільший онлайн-ритейлер в Україні, Click - інтернет-магазин комп'ютерної, побутової та електронної техніки та RPC – інтернет-магазин техніки та електроніки [1, 2, 3].

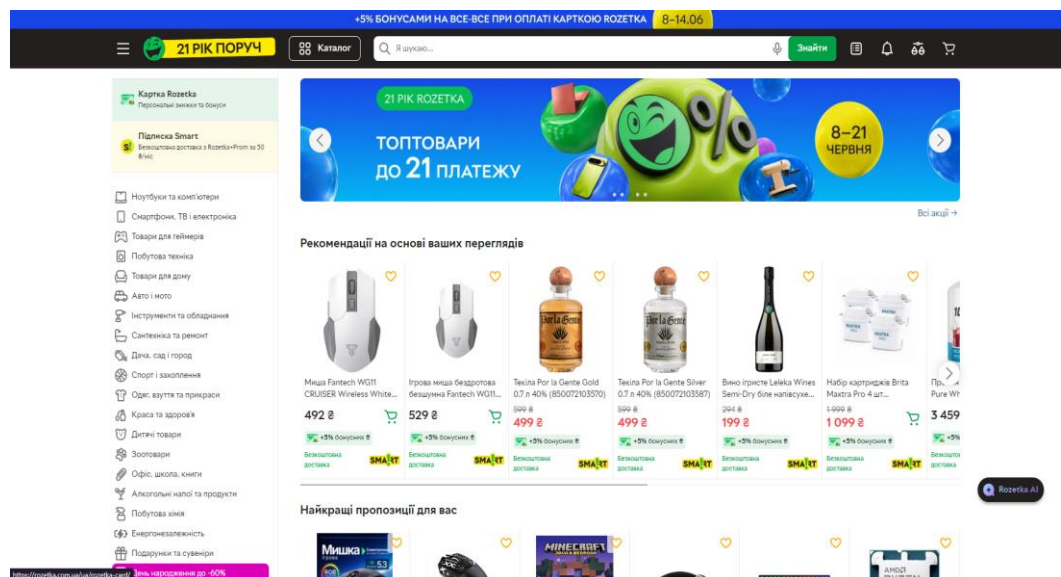


Рис. 1.1. Головна сторінка Rozetka

Rozetka має найбільший асортимент і розвинену інфраструктуру, однак її універсальність створює більш складну структуру для покупця, який шукає саме офісне обладнання. Через велику кількість різних категорій, рекламних блоків і додаткових сервісів інтерфейс виглядає перевантаженим порівняно зі спеціалізованими магазинами.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

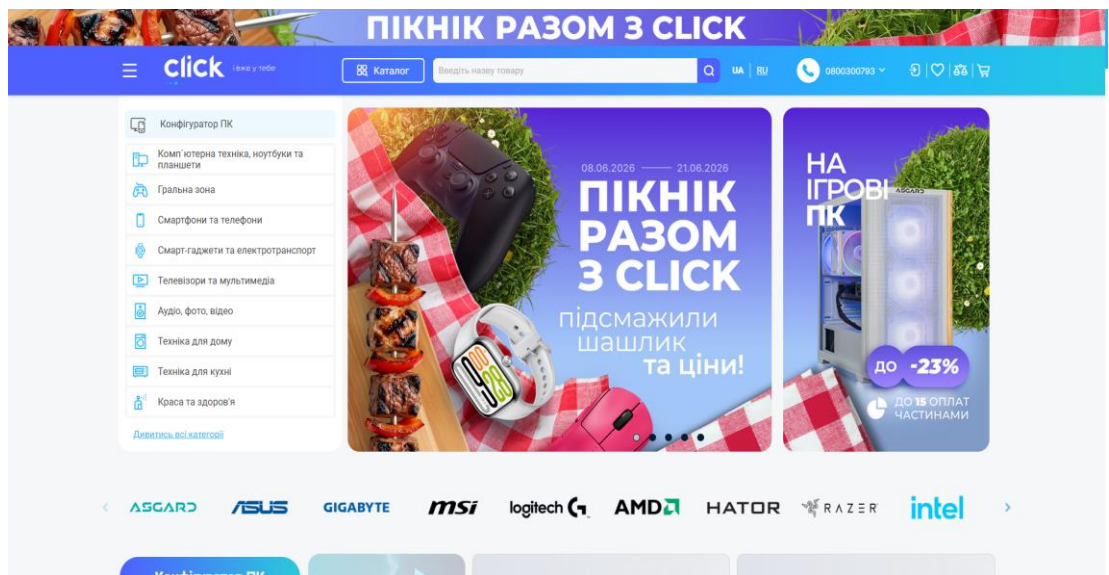


Рис. 1.2. Головна сторінка Click

Click має виражену технічну спрямованість і містить окремі категорії офісної техніки, БФП, принтерів, сканерів, шредерів, ламіновальних матеріалів та аксесуарів до друкувальної техніки. Його структура більш наближена до тематики інтернет-магазину оргтехніки, ніж у великого маркетплейсу. Водночас Click охоплює не тільки оргтехніку, а й комп'ютерну техніку, смартфони, електроніку, ігрові товари, тощо, тому спеціалізація саме на офісному обладнанні є частковою.

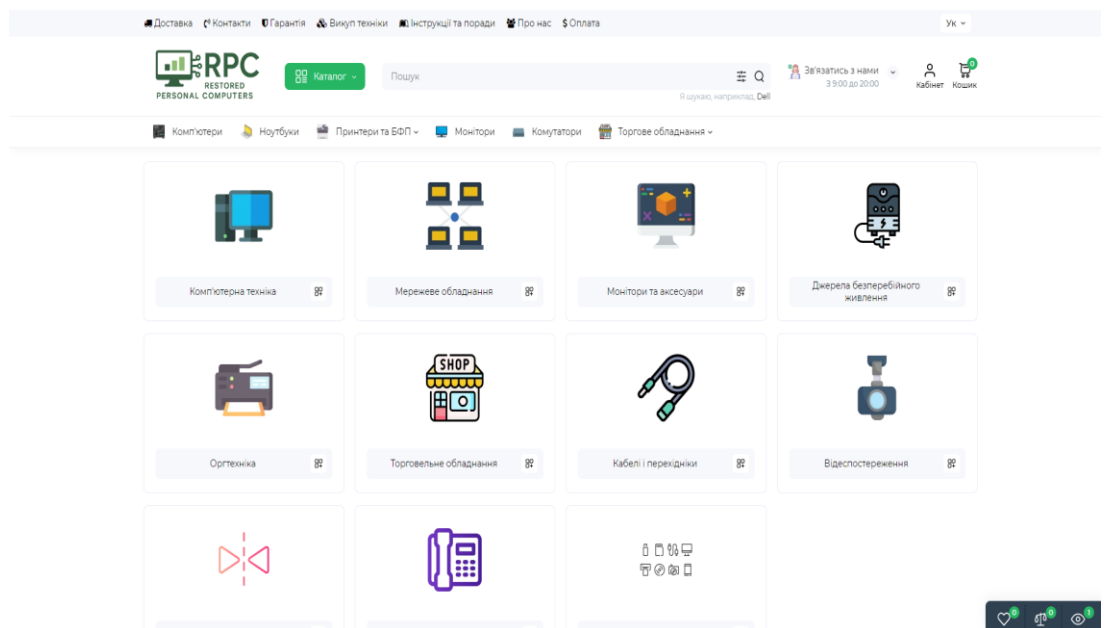


Рис. 1.3. Головна сторінка RPC

RPC найбільше відповідає тематиці розробки, оскільки має окремий акцент на комп'ютерній техніці, оргтехніці, мережевому обладнанні, принтерах,

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

БФП, сканерах, картриджах, тонерах, витратних матеріалах і запчастинах для друкувальної техніки. Його інтерфейс є простішим і менш перевантаженим, що є перевагою для користувача, який шукає конкретну технічну продукцію. Разом із цим магазин поступається великим платформам за масштабом додаткових сервісів і широтою асортименту.

Проаналізувавши аналогічні рішення, складемо наступну таблицю порівняння:

Таблиця 1.1

Таблиця порівняння аналогів

Характеристика	Rozetka	Click	RPC
Простота інтерфейсу	-	+-	+
Пошук і фільтрація	+	+	+-
Інтеграція з доставкою	+	+	+
Асортимент оргтехніки	+	+	+-
Відображення к-ті наявного товару	-	-	-
Онлайн-оплата	+	+	+

Rozetka має широкий асортимент, розвинений пошук, фільтрацію, інтеграцію з доставкою та онлайн-оплату, проте через масштаб маркетплейсу його інтерфейс є більш перевантаженим для користувача, який шукає саме офісне обладнання. Click має збалансовану структуру, підтримує пошук, фільтрацію, доставку й онлайн-оплату, а також містить достатній асортимент техніки, однак спеціалізація на оргтехніці в нього є частковою. RPC вирізняється простішим інтерфейсом і кращою відповідністю тематиці оргтехніки, але поступається більшим платформам за масштабом асортименту та відсутністю

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

фільтрації. Окремим недоліком усіх розглянутих аналогів є те, що вони переважно показують наявність товару, не вказуючи доступну кількість на момент перегляду.

Отже, після аналізу та порівняння доступних рішень, під час розробки власного інтернет-магазину доцільно врахувати переваги розглянутих інтернет-магазинів: просту та зручну структуру інтерфейсу, пошук і фільтрацію, інтеграцію з сервісом доставки для отримання актуальних логістичних даних, онлайн-оплату, достатній асортимент оргтехніки та відображення доступної кількості товару для покупки.

1.3. Вибір архітектури веб орієнтованої системи

Вибір архітектури є одним із ключових та найбільш відповідальних етапів розроблення інтернет-магазину оргтехніки, оскільки саме архітектурна концепція визначає спосіб взаємодії між автономними частинами програмного продукту, логіку обробки запитів, організацію збереження великих масивів даних і можливості подальшого еволюційного розвитку платформи. Для сучасної веб орієнтованої системи електронної торгівлі архітектурне рішення має гарантувати стабільну роботу каталогу при паралельних запитах користувачів, високу швидкість обробки, захищений доступ до особистих кабінетів, коректне оформлення замовлень, безперебійну інтеграцію з платіжним шлюзом та отримання актуальних логістичних даних. Користувацька частина повністю відповідає за візуалізацію, тоді як серверна частина бере на себе функції обробки HTTP-запитів, верифікації прав доступу, виконання бізнес-логіки та комунікації із зовнішніми сервісами. База даних при цьому забезпечує цілісність, ізолюваність і надійне збереження всієї інформації про зареєстрованих користувачів, категорії та товари з їхніми технічними характеристиками. Через таку багатофункціональність архітектура повинна забезпечувати суворий поділ відповідальності між цими компонентами, унеможлиблюючи ситуації, коли зміни в інтерфейсному дизайні руйнують серверні алгоритми.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

Під час етапу проектування веб застосунку були детально розглянуті та проаналізовані різні архітектурні підходи, серед яких монолітна, мікросервісна, серверлес та класична клієнт-серверна моделі. Монолітна архітектура передбачає об'єднання графічного інтерфейсу, бізнес-логіки та механізмів безпосереднього доступу до бази даних в єдиному програмному коді та середовищі розгортання, що є зручним виключно для швидкого старту невеликих проєктів, але суттєво ускладнює незалежний розвиток і розгортання фронтенду та бекенду при збільшенні функціоналу та зростанні навантаження. Мікросервісна архітектура, навпаки, базується на поділі системи на десятки незалежних сервісів, які комунікують між собою через мережеві протоколи, що є високоефективним для величезних міжнародних маркетплейсів під критичним навантаженням, проте для інтернет-магазину оргтехніки вона створює надмірну складність у налаштуванні міжсервісної комунікації, моніторингу помилок та розгортанні інфраструктури, вимагаючи великих обчислювальних ресурсів. Серверлес-архітектура, яка орієнтована на короткочасне виконання окремих функцій у хмарі лише безпосередньо під час запиту, є найменш передбачуваною для інтернет-магазину через специфіку електронної комерції, яка потребує постійного стабільного підключення до бази даних, безперервного контролю за користувацькими сесіями та миттєвої синхронізації залишків на складі.

Враховуючи особливості розроблюваного інтернет-магазину оргтехніки, а також необхідність забезпечення високої швидкості роботи інтерфейсу та безпеки даних, було обрано трирівневу клієнт-серверну архітектуру. Вона забезпечує чіткий логічний та фізичний розподіл веб системи на три незалежні та автономні рівні, до яких відносяться клієнтська і серверна частини, та реляційна база даних. Такий підхід дозволяє створити сучасний динамічний веб інтерфейс, який не перезавантажує сторінки при кожній дії користувача, організувати безпечну та стандартизовану взаємодію через REST API, а також оптимізувати, модифікувати та масштабувати будь-яку з частин системи в майбутньому без необхідності повної переробки всієї програми.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		15

Клієнтська частина системи реалізується як окремий frontend-застосунок за принципом Single Page Application за допомогою сучасної бібліотеки React та інструменту збирання Vite [4, 5]. Цей рівень виконується безпосередньо в браузері кінцевого користувача, що мінімізує навантаження на сервер і забезпечує миттєву реакцію інтерфейсу на дії користувача. Клієнтський застосунок лише збирає введені користувачем дані, валідує їх на первинному рівні та надсилає асинхронні мережеві запити до сервера, не виконуючи при цьому критичну бізнес-логіку та не зберігаючи у відкритому вигляді конфіденційні відомості чи паролі, що підвищує загальний рівень безпеки системи.

Серверна частина, побудована на базі високопродуктивного асинхронного фреймворку FastAPI, який виконує обробку даних та координацію всіх процесів інтернет-магазину [6]. Вона безперервно приймає вхідні HTTP-запити від клієнта, здійснює ретельну перевірку їхньої коректності та безпеки, реалізує алгоритми автентифікації, звертається до бази даних і повертає структурований результат у форматі JSON за принципами REST API [7, 8]. Завдяки такій повній ізоляції фронтенд і бекенд можна розробляти, тестувати та розгортати абсолютно незалежно один від одного, оскільки будь-які експерименти зі стилями сторінок чи дизайном кнопок на клієнтському рівні не потребують переробки серверної логіки, якщо зберігається єдиний формат обміну даними. Рівень зберігання даних повністю відокремлено від бізнес-логіки та розміщено у реляційній базі даних. Серверна частина звертається до бази даних через спроектовані об'єктні моделі, що повністю виключає можливість прямого доступу клієнта до таблиць, захищає систему від SQL-ін'єкцій та суттєво підвищує захищеність комерційної інформації.

Важливою частиною обраної архітектури є механізм авторизації та розмежування прав доступу на основі використання JWT токенів [9]. Після успішного введення логіна та пароля, які перевіряються на сервері, користувач отримує підписаний цифровим ключем токен, який згодом автоматично додається до заголовків усіх захищених HTTP-запитів. Це дозволяє серверу

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

миттєво ідентифікувати особу без повторного запиту пароля, а адміністративні функції системи, такі як додавання нових товарів, додатково обмежуються на рівні бекенду суворого перевіркою ролі користувача, що робить неможливим несанкціонований доступ до панелі керування. Окрему увагу при проєктуванні архітектури приділено обробці гостей сесій, завдяки чому неавторизовані користувачі можуть вільно наповнювати кошик товарами, а під час подальшої реєстрації або входу система автоматично об'єднує кошики.

Обрана трирівнева клієнт-серверна архітектура суттєво спрощує процедури тестування, відлагодження та супроводу системи, оскільки інтерфейсні компоненти React, серверні маршрути FastAPI та реляційну структуру бази даних можна перевіряти, оптимізувати та покривати тестами як абсолютно автономні програмні модулі. Крім того, наявність чітко задокументованого REST API робить розроблений додаток надзвичайно гнучким для подальшого розвитку.

1.4. Вибір інструментальних засобів та вимоги до системи

Для реалізації інтернет-магазину було обрано сучасний набір інструментальних засобів, який повністю відповідає клієнт-серверній архітектурі системи. Веб застосунок розподіляється на незалежні клієнтську частину, сервер та базу даних, тому технології добиралися з урахуванням швидкості розроблення інтерфейсу, стабільної обробки запитів, захищеної авторизації, надійної інтеграції платіжного сервісу та автоматичного отримання логістичної інформації. Обраний стек повністю підтримує стабільний локальний запуск під час розробки, а також забезпечує можливість розгортання проєкту на віддаленому хостингу у майбутньому.

Клієнтську частину інтернет-магазину реалізовано за допомогою компонентної бібліотеки React та сучасного швидкого інструменту збирання Vite. Компонентний підхід React дозволяє декомпонувати інтерфейс на окремі, ізольовані блоки коду, придатні для багаторазового використання, що значно спрощує підтримку й розширення сайту. Для створення легкого, простого та зручного дизайну, застосовуються CSS-стилі, що дозволяє повністю відмовитися

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

від сторонніх важких CSS-фреймворків, уникнути конфліктів глобальних стилів та полегшити підсумкову продуктову збірку проєкту [10]. Взаємодія фронтенду із сервером відбувається асинхронно через вбудований у браузері інтерфейс Fetch API, який забезпечує передачу форм, керування товарами в кошику та виконання адміністративних операцій у форматі обміну даними JSON без підключення додаткових сторонніх бібліотек.

Серверна частина побудована на мові програмування Python із використанням асинхронного веб фреймворку FastAPI [11]. За допомогою FastAPI реалізовано всі маршрути REST API, які обробляють запити користувачів, а також автоматично генерують інтерактивну документацію в стандартах OpenAPI та Swagger для тестування функцій. Для захисту сервера від некоректної або неповної інформації та автоматичної перевірки типів даних у вхідних формах застосовується бібліотека Pydantic, яка перетворює JSON-структури у повноцінні об'єкти Python [12, 13, 14]. Безпека облікових записів та захист персональних даних реалізовані за допомогою бібліотеки bcrypt, яка забезпечує надійне одностороннє хешування паролів із додаванням випадкової солі перед їх безпосереднім збереженням у базу даних, що повністю виключає можливість відновлення пароля навіть у разі отримання зловмисниками доступу до таблиць [15].

Для надійного збереження всієї інформації про товари, категорії, користувачів та замовлення використовується реляційна база даних PostgreSQL, структура якої підтримує цілісність зв'язків, транзакційність та виключає дублювання записів [16, 17]. Взаємодія серверу з таблицями бази даних відбувається через інструмент ORM SQLAlchemy, що дозволяє працювати з даними на рівні об'єктів Python, а зміни у структурі схеми даних фіксуються й послідовно застосовуються через систему міграцій Alembic [18, 19, 20, 21, 22]. Управління всіма залежностями backend-частини, створення ізольованого віртуального середовища та швидкий запуск сервера здійснюється за допомогою інструменту uv, який відрізняється високою швидкістю виконання операцій [23]. Додатково під час розробки в середовищі Visual Studio Code застосовуються

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

форматувальник коду та лінтер Ruff для підтримки єдиного стилю написання програм відповідно до стандартів PEP 8 [24, 25].

Окремою перевагою обраного технологічного стеку є його відповідність модульному принципу розробки. Серверна частина може бути поділена на маршрути, сервіси, схеми, моделі та допоміжні модулі, що спрощує підтримку програмного коду та дозволяє логічно відокремити різні частини функціоналу. Наприклад, маршрути авторизації відповідають лише за реєстрацію, вхід і отримання даних поточного користувача, маршрути товарів – за роботу з каталогом, а маршрути замовлень – за оформлення замовлення і оплату. Такий поділ робить структуру бекенд частини зрозумілою та зменшує ризик помилок під час подальшого розширення системи. Клієнтська частина також організована за модульним принципом. Окремі сторінки та компоненти відповідають за головну сторінку, каталог, картку товару, кошик, оформлення замовлення, особистий кабінет і адміністративну панель. Це дозволяє повторно використовувати спільні елементи інтерфейсу, наприклад шапку сайту, картку товару, модальні вікна, кнопки, форми та повідомлення про результат виконання дії.

Авторизація користувачів та надійне розмежування прав доступу між користувачем і адміністратором реалізовані на основі механізму JWT-токенів, які видаються сервером після успішного входу та перевіряються при кожному зверненні до захищених маршрутів або адміністративної панелі. Для проведення безпечної онлайн-оплати без прямої обробки платежів, на сайті підключено платіжний шлюз LiqPay, а для вибору актуальних міст та відділень під час заповнення логістичних даних використовується API Нової Пошти, що мінімізує помилки ручного введення адреси та суттєво прискорює подальшу відправку замовлення [26, 27].

Для локального запуску системи під час розробки та перевірки модулів достатньо системи з 2-ядерним процесором із частотою від 1,8 ГГц, 4 ГБ оперативної пам'яті та 5 ГБ вільного місця на накопичувачі під керуванням операційної системи Windows. Для розгортання системи на хостингу або для

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

комфортної локальної роботи рекомендується використовувати комп'ютер із 4-ядерним процесором із частотою від 2,4 ГГц, 8 ГБ оперативної пам'яті та від 10 ГБ вільного місця на диску.

Важливим аспектом безпеки розгортання є захист конфігураційних даних та чутливих змінних середовища. Усі приватні ключі LiqPay, параметри підключення до бази даних, секрети для формування JWT-токенів та ключ API Нової Пошти повністю ізолюються у локальних змінних середовища і не потрапляють до репозиторію. Створений програмний продукт також спроектовано так, щоб стабільно реагувати на будь-які виняткові ситуації, включаючи порожній кошик, відсутність потрібного товару, помилки авторизації або тимчасову недоступність зовнішнього API. У всіх подібних випадках система не припиняє свою роботу критичним збоєм сервера, а генерує та виводить користувачеві повідомлення.

Таким чином, обраний інструментальний стек розробки, що поєднує бібліотеку React, інструмент Vite, асинхронний фреймворк FastAPI, систему керування базою даних PostgreSQL та зовнішні API, повністю задовольняє всі сформовані функціональні, технічні та безпекові вимоги до системи торгівлі оргтехнікою. Чітке розділення відповідальності між клієнтською та серверною частинами забезпечує високу швидкість завантаження сторінок, захищеність фінансових операцій та створює надійну основу для подальшого масштабування та розширення функціоналу інтернет-магазину.

Висновки до першого розділу

У першому розділі випускної кваліфікаційної роботи бакалавра було проведено аналіз предметної області та сформовано теоретичний фундамент для розробки веб орієнтованої системи інтернет-магазину оргтехніки. Під час аналізу ринку роздрібної онлайн-торгівлі офісним обладнанням, комп'ютерною периферією та витратними матеріалами було виявлено специфічні особливості таких товарів, що вимагають побудови чіткої інформаційної структури. Установлено, що для успішної взаємодії користувача з платформою критично важливими є наявність деталізованих карток техніки з параметрами сумісності,

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

реалізація надійного пошуку за назвою й кодом артикулу, а також впровадження динамічної фільтрації залежно від обраної категорії товарів.

Було визначено функціональні можливості, які мають бути реалізовані в системі. До них належать реєстрація та авторизація користувачів, робота з категоріями й товарами, перегляд зображень і характеристик, формування кошика, створення замовлення, вибір способу оплати, інтеграція з LiqPay для онлайн-оплати та використання сервісу Нової Пошти для отримання логістичних даних. Окремо визначено роль адміністратора, який через панель керування працює з категоріями, товарами, користувачами та замовленнями.

Під час аналізу аналогів розглянуто інтернет-магазини Rozetka, Click та RPC. Їх порівняння дозволило визначити типовий набір можливостей, характерний для веб систем електронної торгівлі оргтехнікою. Установлено, що базовими елементами таких рішень є каталог товарів, фільтрація, пошук, детальна сторінка товару, кошик, оформлення замовлення, особистий кабінет та адміністративна панель. Водночас аналіз показав, що власна розробка має орієнтуватися на простішу структуру спеціалізованого магазину, зберігаючи функції, необхідні для повного циклу купівлі товару.

У межах вибору архітектури розглянуто кілька підходів до побудови веб орієнтованих систем: монолітну, клієнт-серверну, мікросервісну та серверлес-архітектуру. Для розроблюваного інтернет-магазину обґрунтовано використання клієнт-серверної моделі. Такий підхід забезпечує чіткий розподіл відповідальності між компонентами, спрощує підтримку системи, дає змогу організувати взаємодію через REST API та створює основу для подальшого розширення функціоналу.

Також у першому розділі обґрунтовано вибір інструментальних засобів. Для клієнтської частини обрано React і Vite, для серверної - Python і FastAPI, для зберігання даних - PostgreSQL. Робота з базою даних організовується через SQLAlchemy, зміни її структури керуються за допомогою Alembic, а перевірка даних виконується засобами Pydantic. Для авторизації застосовується JWT, для HTTP-запитів на клієнтському рівні використовується Fetch API, а керування

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

залежностями backend-частини виконується через *uv*. Визначено вимоги до локального запуску та розгортання системи на хостингу.

Отже, у першому розділі сформовано теоретичну та технічну основу для подальшого проєктування інтернет-магазину оргтехніки.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ОРГТЕХНІКИ

2.1. Визначення варіантів використання інтернет магазину оргтехніки

Проектування інтернет-магазину оргтехніки доцільно розпочати з визначення варіантів використання системи, оскільки саме вони відображають основні сценарії взаємодії користувачів із програмним продуктом. На цьому етапі встановлюється, які дії може виконувати кожен учасник системи, які обмеження накладаються на доступ до окремих функцій і які модулі необхідно реалізувати для забезпечення повного циклу роботи інтернет-магазину. Для розроблюваної системи основними акторами є гість, зареєстрований користувач та адміністратор.

Гість є користувачем сайту, який ще не зареєструвався або не виконав вхід в обліковий запис. Йому доступні наступні функції інтернет-магазину: перегляд головної сторінки, категорій, сторінок товарів, пошук, застосування фільтрів і додавання товарів до кошика. Можливість формування кошика без авторизації робить взаємодію із сайтом зручнішою, оскільки покупець може спочатку обрати потрібні товари, але для оформлення замовлення, гостю все ж таки потрібно буде зареєструватись або увійти у свій обліковий запис.

Зареєстрований користувач має ширший набір можливостей. Після реєстрації або входу в обліковий запис він може подивитись свої дані в обліковому записі, та за потреби, змінити їх, дивитись історію замовлень, оформлювати замовлення.

Адміністратор відповідає за підтримання актуальності даних інтернет-магазину. Йому доступна панель адміністратора, через яку він може керувати інтернет-магазином, а саме категоріями, товарами, користувачами та замовленнями. Адміністратор може додавати, редагувати та видаляти категорії і товари, змінювати роль користувачів, деактивувати облікові записи, змінювати статус замовлень.

Основні варіанти використання представлені в таблицях 2.1 – 2.5.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Таблиця 2.1

Таблиця варіанту використання «Реєстрація/вхід»

Основні актори	Гість, зареєстрований користувач, адміністратор
Короткий опис	Користувачі можуть увійти/зареєструватися, щоб продовжити роботу зі сайтом, наприклад, для оформлення замовлення. Адміністратор входить до свого облікового запису, щоб керувати інтернет-магазином

Для того, щоб користувач мав повний доступ до функціоналу інтернет-магазину (перегляд облікового запису, історію замовлень, оформлення замовлення), йому необхідно виконати вхід або зареєструватися у системі. Адміністратору для отримання доступу до адмін-панелі, необхідно увійти до облікового запису з роллю адміна.

Таблиця 2.2

Таблиця варіанту використання «Керування обліковим записом»

Основні актори	Зареєстрований користувач
Короткий опис	Зареєстрований користувач може переглядати свій обліковий запис, та, за потреби, редагувати дані. Також може дивитись свою історію замовлень

Виконавши вхід або зареєструвавшись у системі, користувач бачить сторінку свого облікового запису, де він може переглянути свої контактні дані, заповнені під час реєстрації, та, за бажанням, редагувати їх. Користувачу також доступна вкладка з історіями замовлень, де він може переглянути список своїх оформлених замовлень за весь час.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

Таблиця 2.3

Таблиця варіанту використання «Перегляд, пошук та фільтрація товарів»

Основні актори	Гість, зареєстрований користувач
Короткий опис	Користувачі можуть переглядати каталог товарів, а для знаходження потрібного товару можуть скористатись функціоналом пошуку та фільтрації товарів

Користувачі, як незареєстровані, так і зареєстровані, можуть переглядати каталог товарів, а для знаходження потрібного товару, можуть скористатись пошуком та фільтрами. У полі пошуку можна здійснити пошук потрібного товару за його назвою або артикулом. Якщо користувачу потрібен товар з потрібними йому параметрами, він може скористатися фільтрами.

Таблиця 2.4

Таблиця варіанту використання «Оформлення замовлення»

Основні актори	Зареєстрований користувач
Короткий опис	Після того, як користувач заповнив кошик необхідними йому товарами, він переходить до сторінки оформлення замовлення, де заповнює необхідні дані для виконання замовлення

Користувач, вибравши та додавши до кошика необхідні йому товари, переходить до сторінки оформлення замовлення, де він заповнює форму з необхідними даними: контактні дані, які підставляються автоматично, дані доставки (обирає місто та відділення) та спосіб оплати (готівкою при отриманні або онлайн через LiqPay).

Таблиця варіанту використання «Керування інтернет-магазином»

Основні актори	Адміністратор
Короткий опис	Адміністратор має можливість керування інтернет-магазином через адмін-панель, звідки він може додавати, редагувати та видаляти категорії і товари, керувати обліковими записами користувачів та змінювати статус замовлень

Адміністратор, після входу у свій обліковий запис має доступ до адмін-панелі, звідки може керувати магазином: категоріями, товарами, користувачами та замовленнями.

Після визначення варіантів використання побудуємо діаграму класів, яка відображає основні сутності системи, їхні атрибути та зв'язки. Діаграма класів наведена на рисунку 2.1.

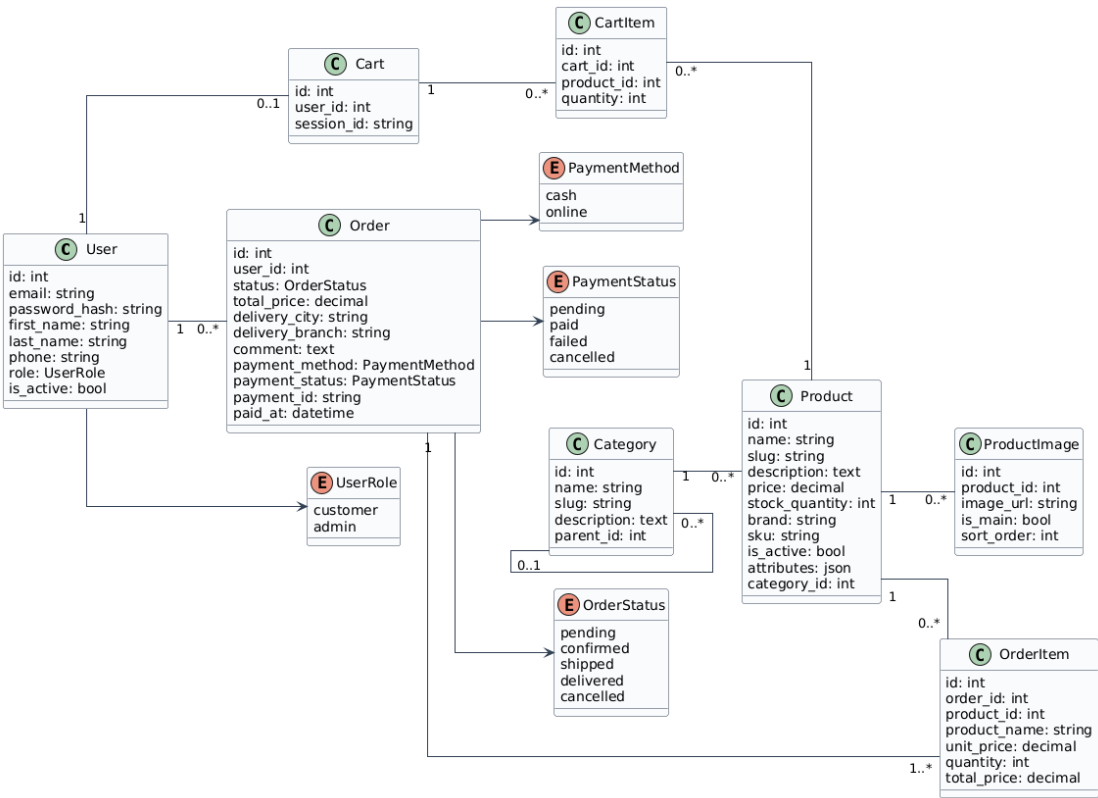


Рис. 2.1. Діаграма класів інтернет-магазину оргтехніки

Клас User описує користувача системи. Він містить ідентифікатор користувача, електронну пошту, хеш пароля, ім'я, прізвище, номер телефону, роль та стан активності облікового запису. Через поле role користувач може бути звичайним покупцем або адміністратором. Користувач може мати один кошик та оформлювати багато замовлень.

Клас UserRole є переліком можливих ролей користувача. У системі передбачено дві ролі: customer – звичайний зареєстрований користувач, та admin - адміністратор, який має доступ до адмін-панелі й може керувати товарами, категоріями, користувачами та замовленнями.

Клас Cart описує кошик покупця. Він містить ідентифікатор кошика, ідентифікатор користувача та session_id, який використовується для роботи з кошиком гостя. Це дозволяє додавати товари до кошика навіть без авторизації. Після входу або реєстрації гостьовий кошик може бути об'єднаний із кошиком зареєстрованого користувача.

Клас CartItem описує окрему позицію в кошику. Він містить ідентифікатор позиції, посилання на кошик, посилання на товар та кількість одиниць товару. Один кошик може містити багато позицій, а кожна позиція пов'язана з конкретним товаром.

Клас Category описує категорію товарів. Він містить ідентифікатор, назву, slug, опис та parent_id, який використовується для побудови ієрархії категорій. Завдяки цьому категорії можуть мати підкатегорії. Одна категорія може містити багато товарів.

Клас Product описує товар інтернет-магазину. Він містить ідентифікатор, назву, slug, опис, ціну, кількість, бренд, артикул, стан активності, характеристики у форматі JSON та посилання на категорію. Поле attributes використовується для зберігання характеристик товару, які можуть відрізнятися залежно від категорії, наприклад тип друку, формат паперу, швидкість друку або інші параметри оргтехніки.

Клас ProductImage описує зображення товару. Він містить ідентифікатор, посилання на товар, URL зображення, ознаку головного зображення та порядок

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

сортування. Один товар може мати декілька зображень, серед яких одне може бути позначене як основне.

Клас Order описує замовлення користувача. Він містить ідентифікатор замовлення, посилання на користувача, статус замовлення, загальну суму, місто, відділення, коментар, спосіб оплати, статус платежу, ідентифікатор платежу та дату успішної оплати.

Клас OrderItem описує окрему позицію в замовленні. Він містить ідентифікатор позиції, посилання на замовлення, посилання на товар, назву товару, ціну за одиницю, кількість та загальну вартість позиції. Збереження назви товару й ціни в OrderItem дозволяє зберегти історію замовлення навіть у разі подальшої зміни ціни або назви товару в каталозі.

Клас OrderStatus є переліком можливих статусів замовлення. У системі використовуються такі статуси: pending – замовлення очікує обробки, confirmed – замовлення підтверджено, shipped – замовлення відправлено, delivered – замовлення доставлено, cancelled – замовлення скасовано.

Клас PaymentMethod описує доступні способи оплати. У системі передбачено два способи оплати: cash – оплата готівкою при отриманні, та online – онлайн-оплата через платіжний сервіс LiqPay.

Клас PaymentStatus описує стан платежу. Він може набувати значень pending, paid, failed або cancelled. Статус pending означає, що платіж очікує обробки, paid – оплату успішно виконано, failed – платіж не пройшов, cancelled – оплату або замовлення було скасовано.

Таким чином, діаграма класів демонструє логічну структуру системи та взаємозв'язки між її основними сутностями. Користувач взаємодіє з каталогом товарів, додає товари до кошика та оформлює замовлення.

2.2. Розробка бази даних системи

База даних інтернет-магазину оргтехніки призначена для зберігання інформації про користувачів, категорії, кошики, та замовлення. Вона є центральним елементом серверної частини, оскільки саме через неї забезпечується збереження стану системи, доступ до актуальних товарних даних,

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

формування історії покупок і робота адміністративної панелі. Для реалізації бази даних використано СКБД PostgreSQL, що дає змогу працювати з реляційними зв'язками, обмеженнями цілісності, числовими типами для вартості товарів і форматом JSONB для зберігання технічних характеристик товару.

Під час проєктування бази даних було враховано основні процеси інтернет-магазину: реєстрацію користувачів, перегляд каталогу, фільтрацію товарів, додавання позицій до кошика, оформлення замовлення та адміністративне керування. Структура побудована так, щоб кожна сутність відповідала окремому об'єкту предметної області, а зв'язки між таблицями відображали логіку роботи системи.

Схема БД інтернет-магазину наведена у додатку А.

Опис таблиць БД інтернет-магазину наведено у таблицях 2.6 – 2.14.

Таблиця 2.6

Опис таблиці users

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор користувача
email	character varying(255)	Email користувача
password_hash	character varying(255)	Хешований пароль користувача
first_name	character varying(100)	Ім'я користувача
last_name	character varying(100)	Прізвище користувача
phone	character varying(20)	Мобільний телефон користувача
role	userrole	Роль користувача
is_active	boolean	Статус активності облікового запису користувача

created_at	timestamp with time zone	Дата і час створення облікового запису
updated_at	timestamp with time zone	Дата і час оновлення облікового запису

Таблиця users призначена для зберігання інформації про користувачів системи. Вона містить унікальний ідентифікатор, електронну пошту, хеш пароля, ім'я, прізвище, номер телефону, роль, статус активності облікового запису, а також дату створення та оновлення облікового запису. Поле role визначає права користувача в системі: звичайний покупець або адміністратор. Таблиця users пов'язана з таблицями carts та orders, оскільки користувач може мати кошик і оформлювати замовлення.

Таблиця 2.7

Опис таблиці categories

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор категорії
name	character varying(100)	Назва категорії
slug	character varying(100)	Унікальна людино-зрозуміла URL
parent_id	integer	Посилання на батьківську категорію
description	text	Опис категорії
created_at	timestamp with time zone	Дата і час створення категорії
updated_at	timestamp with time zone	Дата і час оновлення категорії

Таблиця categories використовується для зберігання категорій товарів. Вона містить ідентифікатор категорії, назву, унікальний slug, опис, дату створення та оновлення. Поле parent_id дозволяє реалізувати ієрархічну структуру категорій, тобто створювати підкатегорії. Одна категорія може містити багато товарів, що відображено зв'язком із таблицею products.

Таблиця 2.8

Опис таблиці products

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор товару
name	character varying(255)	Назва товару
slug	character varying(255)	Унікальна людино-зрозуміла URL
description	text	Опис товару
price	numeric(10,2)	Ціна товару
stock_quantity	integer	Кількість товару
brand	character varying(100)	Бренд товару
sku	character varying(100)	Артикул товару
is_active	boolean	Статус активності товару
attributes	jsonb	Список технічних характеристик товару
category_id	integer	Посилання на категорію
created_at	timestamp with time zone	Дата і час створення товару
updated_at	timestamp with time zone	Дата і час оновлення товару

Таблиця products зберігає основну інформацію про товари інтернет-магазину. До неї входять назва товару, slug, опис, ціна, кількість на складі, бренд,

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

артикул, ознака активності, додаткові характеристики у форматі jsonb, а також посилання на категорію через поле category_id. Використання поля attributes типу jsonb дозволяє зберігати характеристики, які можуть відрізнятися залежно від категорії товару, наприклад тип друку, формат паперу, швидкість друку або інші параметри оргтехніки.

Таблиця 2.9

Опис таблиці product_images

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор фото
product_id	integer	Посилання на товар
image_url	character varying(500)	Посилання на фото
is_main	boolean	Чи є фото головним
sort_order	integer	Порядок сортування
created_at	timestamp with time zone	Дата і час створення фото
updated_at	timestamp with time zone	Дата і час оновлення фото

Таблиця product_images призначена для зберігання зображень товарів. Вона містить ідентифікатор зображення, посилання на товар, URL зображення, ознаку головного зображення та порядок сортування. Один товар може мати декілька зображень, що дає змогу відображати товар у каталозі та на сторінці детального перегляду.

Таблиця 2.10

Опис таблиці carts

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор кошика

user_id	integer	Посилання на користувача
session_id	character varying(36)	Посилання на гостя
created_at	timestamp with time zone	Дата і час створення кошика
updated_at	timestamp with time zone	Дата і час оновлення кошика

Таблиця carts відповідає за зберігання кошиків покупців. Вона містить ідентифікатор кошика, посилання на користувача та session_id. Поле user_id використовується для кошика зареєстрованого користувача, а session_id – для кошика гостя. Завдяки цьому система дозволяє додавати товари до кошика навіть без авторизації, а після входу або реєстрації користувача гостьовий кошик може бути об'єднаний із кошиком акаунта.

Таблиця 2.11

Опис таблиці cart_items

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор позиції кошика
cart_id	integer	Посилання на кошик
product_id	integer	Посилання на товар
quantity	integer	Кількість одиниць товару

Таблиця cart_items зберігає позиції кошика. Кожен запис містить ідентифікатор позиції, посилання на кошик, посилання на товар та кількість одиниць товару. Таблиця є проміжною між carts і products, оскільки один кошик може містити багато товарів, а один товар може бути доданий до різних кошиків.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

Опис таблиці orders

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор замовлення
user_id	integer	Посилання на користувача
status	orderstatus	Статус замовлення
total_price	numeric(10,2)	Загальна сума замовлення
delivery_city	character varying(100)	Місто для доставки замовлення
comment	text	Коментар до замовлення
payment_method	paymentmethod	Спосіб оплати замовлення
created_at	timestamp with time zone	Дата і час створення замовлення
updated_at	timestamp with time zone	Дата і час оновлення замовлення
delivery_branch	character varying(255)	Відділення для доставки замовлення
payment_status	paymentstatus	Статус оплати замовлення
payment_reference	character varying(100)	Референс оплати замовлення
payment_error	text	Повідомлення про помилку

delivery_city	character varying(100)	Місто для доставки замовлення
comment	text	Коментар до замовлення

Таблиця orders використовується для зберігання оформлених замовлень. Вона містить ідентифікатор замовлення, посилання на користувача, статус замовлення, загальну суму, місто доставки, відділення доставки, коментар до замовлення, спосіб оплати, статус платежу, платіжний ідентифікатор або референс, інформацію про помилку оплати, а також дату створення й оновлення запису. Статус замовлення дозволяє адміністратору відстежувати етап виконання замовлення, а статус платежу використовується для контролю результату онлайн-оплати.

Таблиця 2.13

Опис таблиці order_items

Назва поля	Тип даних	Призначення
id	serial	Унікальний ідентифікатор позиції
order_id	integer	Посилання на замовлення
product_id	integer	Посилання на товар
product_name	character varying(255)	Назва товару
unit_price	numeric(10,2)	Ціна за одиницю товару
quantity	integer	Кількість товару

Таблиця order_items зберігає товари, що входять до конкретного замовлення. Вона містить ідентифікатор позиції, посилання на замовлення, посилання на товар, назву товару, ціну за одиницю та кількість. Збереження назви товару й ціни безпосередньо в позиції замовлення є важливим, оскільки

дозволяє зберегти історію покупки навіть у разі подальшої зміни ціни, назви або характеристик товару в каталозі.

Таблиця 2.14

Опис таблиці alembic_version

Назва поля	Тип даних	Призначення
version_num	character_varying(32)	Унікальний ідентифікатор версії БД

Таблиця alembic_version є службовою таблицею, яка використовується інструментом Alembic для контролю версій міграцій бази даних. Вона потрібна для коректного оновлення структури БД під час розробки.

2.3. Проєктування та реалізація алгоритмів роботи системи

Після визначення варіантів використання, побудови структури та розроблення бази даних необхідно описати узагальнені алгоритми роботи інтернет-магазину. Алгоритми відображають послідовність дій, які виконуються під час взаємодії користувача із системою, а також показують порядок обміну даними між клієнтом, сервером, базою даних і зовнішніми сервісами. Для розроблюваного інтернет-магазину важливо розглянути наступні алгоритми: роботи з кошиком, реєстрації і входу, оформлення замовлення та адміністративного керування.

Загальний принцип роботи інтернет-магазину полягає в виконанні дії в інтерфейсі, після чого клієнтська частина формує запит до серверної частини. Сервер отримує запит, перевіряє його коректність і визначає, яку операцію потрібно виконати. Якщо дія пов'язана з отриманням даних, сервер звертається до бази даних і повертає потрібну інформацію. Якщо дія змінює стан системи, наприклад створює нового користувача, додає товар до кошика або формує замовлення, сервер виконує перевірки, записує або оновлює дані в базу даних.

Алгоритм використання системи гостем починається з відкриття головної сторінки інтернет-магазину. Гість може переглядати каталог товарів, переходити між категоріями, відкривати сторінки окремих товарів, ознайомлюватися з

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		36

описом, ціною, артикулом, наявністю та технічними характеристиками. Для швидкого пошуку потрібної продукції він може скористатися пошуком за назвою або артикулом, а також застосувати фільтрацію за ціною, наявністю та характеристиками відповідної категорії. Після вибору товару гість має змогу додати його до кошика, де тимчасово зберігаються обрані позиції. Завершити оформлення замовлення без авторизації він не може, тому для переходу до покупки система пропонує виконати вхід або пройти реєстрацію.

Алгоритм використання системи зареєстрованим користувачем передбачає повний цикл оформлення покупки. Після авторизації користувач може працювати з каталогом, пошуком, фільтрами та кошиком, змінювати кількість товарів або видаляти непотрібні позиції. Під час оформлення замовлення він перевіряє склад кошика, вказує контактні дані, обирає місто та відділення Нової Пошти, за потреби додає коментар та вибирає спосіб оплати. Якщо обрано оплату готівкою, замовлення створюється без переходу до платіжного сервісу. Якщо вибрано онлайн-оплату, система формує платіж через LiqPay, після чого користувач переходить на сторінку оплати та після завершення операції повертається до інтернет-магазину. В особистому кабінеті зареєстрований користувач може переглядати власні дані та історію оформлених замовлень.

Алгоритм використання системи адміністратором починається з авторизації та перевірки ролі користувача. Після входу адміністратор отримує доступ до адміністративної панелі, звідки керує інтернет-магазином. Він може створювати, редагувати та видаляти категорії, додавати нові товари, змінювати їх назву, артикул, опис, бренд, ціну, кількість, зображення та технічні характеристики. Також адміністратор переглядає список зареєстрованих користувачів і працює із замовленнями: відкриває їх деталі, аналізує склад покупки, спосіб оплати, статус платежу, місто й відділення доставки, після чого за потреби змінює статус замовлення відповідно до етапу обробки. Усі адміністративні дії виконуються через захищені маршрути системи та доступні лише користувачу з роллю адміністратора.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

Одним із основних алгоритмів системи є алгоритм реєстрації та авторизації користувача. Реєстрація починається із заповнення форми, у якій користувач вказує електронну пошту, пароль, ім'я, прізвище та номер телефону. Після надсилання форми клієнтська частина передає дані на сервер. Сервер перевіряє їх, контролює наявність обов'язкових полів і перевіряє, чи не використовується така електронна пошта іншим обліковим записом. Якщо дані коректні, пароль хешується, а в таблиці users створюється новий запис. Авторизація виконується шляхом перевірки електронної пошти та пароля. Після успішної перевірки сервер формує JWT-токен, який надалі використовується для доступу до захищених маршрутів.

Алгоритм роботи з кошиком починається з натискання кнопки додавання товару. Клієнт передає на сервер ідентифікатор товару та кількість. Сервер перевіряє, чи існує товар, чи є він активним і чи достатньо одиниць на складі. Після цього визначається, до якого кошика потрібно додати позицію. Якщо користувач авторизований, кошик пов'язується з його обліковим записом через user_id. Якщо дію виконує гість, використовується session_id, що дозволяє тимчасово зберігати товари до моменту авторизації. Якщо товар уже є в кошику, збільшується його кількість. Якщо товар додається вперше, створюється новий запис у таблиці cart_items. Під час перегляду кошика користувач бачить перелік доданих товарів, їх кількість, ціну та підсумкову суму. Він може змінювати кількість кожної позиції або видаляти товари. Кожна така дія супроводжується запитом до backend, де перевіряється коректність нової кількості та оновлюються записи в базі даних. Якщо кількість товару зменшується до нуля або користувач натискає кнопку видалення, відповідний запис видаляється з кошика.

Оформлення замовлення є одним із центральних алгоритмів системи. Воно починається після того, як зареєстрований користувач переходить із кошика до сторінки оформлення. Спочатку система перевіряє авторизацію та наявність товарів у кошику. Якщо користувач не авторизований або кошик порожній, оформлення не продовжується. Після успішної перевірки користувач заповнює контактні дані, обирає місто й відділення Нової Пошти, зазначає спосіб оплати

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

та за потреби додає коментар. Після підтвердження замовлення frontend надсилає відповідні дані на backend.

На серверній частині під час оформлення замовлення виконується повторна перевірка кошика, наявності товарів і коректності переданих даних. Це необхідно, оскільки інформація на клієнтській частині може бути застарілою або зміненою користувачем. Сервер обчислює загальну суму, створює запис у таблиці orders, а для кожного товару створює окремий запис у таблиці order_items. У позиціях замовлення зберігається назва товару, ціна на момент покупки та кількість. Це дозволяє зберегти історію замовлення навіть у разі подальшої зміни ціни або назви товару.

Окремим етапом оформлення є вибір доставки через сервіс Нової Пошти. Користувач обирає місто, після чого система отримує список доступних відділень. Щоб не виконувати зайві запити, на клієнтській частині використовується кешування отриманих даних. Якщо користувач повторно відкриває список міст або відділень, фронтенд може використати вже завантажену інформацію. Це покращує швидкість роботи та зменшує кількість звернень до зовнішнього сервісу. Обране місто та відділення зберігаються в таблиці orders разом з іншими даними замовлення.

Онлайн-оплата через LiqPay використовується в тому випадку, якщо користувач обирає відповідний спосіб оплати. Після створення або підготовки замовлення сервер формує платіжні дані, які містять суму, опис платежу, ідентифікатор замовлення та службові параметри. Користувач переходить до платіжної сторінки LiqPay і виконує оплату. Після завершення операції LiqPay надсилає callback-запит до серверу. Цей callback обробляється на сервері, перевіряє отримані дані та оновлює статус платежу в замовленні. Якщо платіж успішний, замовлення позначається як оплачене. Якщо операція завершилася помилкою або була скасована, у замовленні фіксується відповідний статус.

Алгоритм адміністративного керування охоплює роботу з категоріями, товарами, користувачами та замовленнями. Після авторизації сервер перевіряє JWT-токен і роль користувача. Якщо користувач має роль адміністратора, йому

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

відкривається доступ до панелі керування. Під час роботи з категоріями адміністратор може створювати нові записи, редагувати назви й описи, а також змінювати структуру каталогу. Під час роботи з товарами він може додавати нові позиції, змінювати ціну, залишок, артикул, бренд, опис, зображення та технічні характеристики. Під час обробки замовлень адміністратор переглядає список створених покупок, відкриває детальну інформацію про кожне замовлення, перевіряє склад, суму, спосіб оплати, статус платежу, місто та відділення доставки. За потреби він змінює статус замовлення відповідно до етапу його виконання.

2.4. Реалізація функціоналу інтернет-магазину оргтехніки

Тепер, коли було виконано проєктування сценаріїв використання інтернет-магазину, побудовано діаграму класів та розроблено базу даних, можна перейти до реалізації функціоналу інтернет-магазину.

Функціонал реєстрації та авторизації користувачів реалізовано на основі взаємодії форми на клієнтській частині з відповідними маршрутами серверного API. Під час реєстрації користувач заповнює форму, у якій зазначає email, пароль, ім'я, прізвище та номер телефону. Після відправлення форми дані передаються на сервер, де проходять перевірку, після чого пароль хешується, а інформація про користувача зберігається в таблиці users. Такий підхід дозволяє не зберігати пароль у відкритому вигляді та підвищує безпеку роботи з обліковими записами. Під час авторизації користувач вводить email та пароль. Сервер перевіряє наявність користувача в базі даних, порівнює введений пароль із хешованим значенням і в разі успішного входу формує JWT-токен. Цей токен використовується для подальшого доступу до захищених маршрутів, зокрема до особистого кабінету, оформлення замовлення та адміністративної панелі. На клієнтській частині стан авторизації зберігається в AuthProvider, що дозволяє різним сторінкам визначати, чи виконав користувач вхід до системи, а також яку роль він має. Завдяки цьому інтерфейс може змінюватися залежно від стану користувача: для гостя відображаються одні елементи, для зареєстрованого покупця – інші, а для адміністратора відкривається доступ до адмін-панелі.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

Каталог товарів реалізовано через отримання списку категорій і товарів із сервера. Сервер надає API-маршрути для перегляду всіх категорій, отримання списку товарів, перегляду товару та отримання товарів певної категорії. На клієнтській частині ці дані відображаються у вигляді каталогу з картками товарів. Картка містить основну інформацію, потрібну для швидкого ознайомлення з товаром: назву, артикул, ціну, доступну кількість, зображення та кнопку додавання до кошика.

Зв'язок товарів із категоріями реалізовано через поле `category_id`, яке пов'язує товар із відповідною категорією в базі даних. Це дозволяє формувати структурований каталог і відображати продукцію за групами. Такий підхід є важливим, оскільки асортимент може включати принтери, сканери, багатофункціональні пристрої, монітори, периферію, витратні матеріали та інші товари. Поділ на категорії полегшує навігацію та допомагає користувачу швидше знайти потрібний товар.

Пошук і фільтрація товарів реалізовані переважно на клієнтській частині після отримання товарів із серверу. Сервер повертає товари за вибраною категорією, а клієнт застосовує до отриманого списку пошуковий запит і фільтри. Користувач може шукати товар за назвою або артикулом, що зручно для випадків, коли покупець знає конкретну модель або код товару. Крім пошуку, реалізовано фільтрацію за ціною, брендом, наявністю та додатковими характеристиками. Це дає змогу звужити список товарів і залишити в каталозі лише ті позиції, які відповідають потребам покупця.

Фільтри формуються на основі поля `attributes`, у якому зберігаються технічні характеристики товару у форматі JSON. Завдяки цьому для різних категорій можуть відображатися різні набори фільтрів. Така реалізація робить фільтрацію гнучкішою та дозволяє не створювати однаковий набір характеристик для всіх товарів, незалежно від їх призначення.

Сторінка окремого товару реалізована для детального перегляду інформації про обрану позицію. Після переходу з каталогу користувач бачить повні дані про товар: зображення, назву, артикул, бренд, ціну, доступну

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

кількість, опис і технічні характеристики. На цій сторінці також передбачено можливість вибору кількості товару та додавання його до кошика.

Функціонал кошика передбачає додавання товарів, зміну кількості, видалення окремих позицій і повне очищення. На сервері для цього використовуються таблиці `carts` і `cart_items`, а на frontend стан кошика зберігається в `CartProvider`. Це дозволяє різним компонентам інтерфейсу отримувати актуальну інформацію про кількість товарів у кошику. Наприклад, після додавання товару з каталогу або сторінки товару кількість у шапці сайту оновлюється без перезавантаження сторінки. Це робить роботу з кошиком більш зручною.

Оформлення замовлення реалізовано як окрему сторінку, на яку користувач переходить із кошика. На цій сторінці відображаються контактні дані користувача, які автоматично підтягуються з бази даних: електронна пошта, ім'я, прізвище та номер телефону. Це зменшує кількість ручного введення та пришвидшує процес оформлення. Користувач також обирає місто й відділення доставки, за потреби зазначає коментар та вибирає спосіб оплати. Після підтвердження замовлення дані передаються на серверну частину, де створюється запис у таблиці `orders` і формуються позиції замовлення в `order_items`.

Особистий кабінет доступний для зареєстрованих користувачів і призначений для перегляду власної інформації та історії замовлень. У кабінеті відображаються контактні дані користувача, а також історія замовлень. Історія замовлень завантажується з серверу через відповідний API-маршрут і містить номер замовлення, дату створення, статус, суму, спосіб оплати, місто та відділення доставки, а також перелік товарів у замовленні. Завдяки цьому покупець може переглядати власні попередні покупки та контролювати стан замовлень.

Адміністративна панель реалізована для користувачів із роллю адміністратора. Доступ до неї обмежений через перевірку авторизації та ролі, тому звичайний покупець не може виконувати адміністративні дії. Панель

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		42

містить розділи для керування категоріями, товарами, користувачами та замовленнями. Такий поділ дозволяє адміністратору швидко переходити до потрібної групи даних і виконувати дії, пов'язані з підтриманням актуальності магазину. У розділі керування категоріями адміністратор може створювати нові категорії, редагувати наявні та видаляти неактуальні. Під час роботи з товарами адміністратор може додавати нові позиції, редагувати назву, артикул, опис, бренд, ціну, кількість, категорію, зображення та характеристики. Розділ замовлень дозволяє переглядати список оформлених замовлень і переглядати детальну інформацію. Адміністратор бачить склад замовлення, контактні дані покупця, спосіб оплати, статус платежу, місто та відділення доставки. За потреби він може змінити статус замовлення відповідно до етапу обробки. Керування користувачами реалізовано для перегляду зареєстрованих облікових записів. Адміністратор може переглядати інформацію про користувачів, їхні контактні дані, роль та стан облікового запису. За потреби він може змінювати роль або стан облікового запису. Це потрібно для розмежування доступу, наприклад для надання адміністративних прав.

Для зручності роботи в адмін-панелі створення, перегляд і редагування даних виконуються через модальні вікна. Такий підхід дозволяє не переходити на окремі сторінки для кожної операції та пришвидшує роботу з даними.

Висновки до другого розділу

У другому розділі виконано проєктування інтернет-магазину оргтехніки. Було визначено основні варіанти використання системи для гостя, зареєстрованого користувача та адміністратора. Описані сценарії показують, як користувач може переглядати каталог, виконувати пошук і фільтрацію товарів, додавати їх до кошика, а після авторизації оформлювати замовлення. Для адміністратора передбачено роботу з категоріями, товарами, користувачами й замовленнями через панель керування.

Також розглянуто структуру програмного комплексу. Діаграма класів відображає основні сутності інтернет-магазину: користувачі, категорії, товари, кошики та замовлення. Описано призначення кожного класу, його основні

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

атрибути та зв'язки з іншими частинами системи. Така структура дозволяє показати логіку взаємодії між товарним каталогом, кошиком, процесом оформлення замовлення та адміністративним керуванням.

У межах проєктування було розроблено реляційну схему PostgreSQL. Було описано таблиці, а також пояснено їхнє призначення й основні зв'язки. Окрему увагу приділено зберіганню технічних характеристик товарів у полі JSONB, що дає змогу використовувати різний набір фільтрів для різних категорій оргтехніки. Структура бази даних забезпечує збереження інформації про користувачів, товари, кошики, замовлення, оплату та доставку.

Описано узагальнені алгоритми роботи системи для кожного типу користувача. Для гостя визначено порядок перегляду каталогу, пошуку, фільтрації та додавання товарів до кошика. Для зареєстрованого користувача розглянуто повний процес оформлення покупки, а для адміністратора – алгоритм роботи з адмін-панелю для керування інтернет-магазином.

Отже, у другому розділі сформовано проєктну основу інтернет-магазину оргтехніки. Визначені варіанти використання, діаграма класів, структура бази даних і алгоритми роботи системи створюють підґрунтя для реалізації основного функціоналу програмного комплексу та подальшого опису інтерфейсу, порядку встановлення й тестування системи.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		44

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ ОРГТЕХНІКИ

3.1. Порядок встановлення та налаштування параметрів системи

Оскільки розроблена система має клієнт-серверну архітектуру, процес розгортання охоплює налаштування клієнтської і серверної частин, бази даних, змінних середовища та зовнішніх сервісів. Коректне встановлення є важливим етапом, оскільки від нього залежить працездатність основних функцій інтернет-магазину: перегляду каталогу, реєстрації та авторизації, роботи з кошиком, оформлення замовлень, онлайн-оплати та даних доставки.

Архітектура інтернет-магазину оргтехніки складається з кількох взаємопов'язаних компонентів. Клієнтська частина реалізована на React і відповідає за відображення інтерфейсу користувача. Серверна частина реалізована на FastAPI та виконує обробку запитів, перевірку даних, авторизацію, роботу з бізнес-логікою, звернення до бази даних і взаємодію із зовнішніми сервісами. Для зберігання даних використовується PostgreSQL, у якій розміщуються таблиці користувачів, категорій, товарів, кошиків та замовлень.

Перед початком встановлення необхідно підготувати програмне середовище. Мають бути встановлені Python, Node.js, PostgreSQL, Git, uv для серверної частини та npm для клієнтської частини. Python використовується для запуску сервера, Node.js – для клієнта, PostgreSQL – для зберігання даних, а Git – для завантаження проєкту з репозиторію.

Першим етапом встановлення є завантаження компонентів програмного комплексу з git-репозиторію. Після клонування репозиторію користувач отримує структуру проєкту, яка містить окремі директорії для серверної та клієнтської частин. У серверній частині розміщено файли FastAPI-застосунку, моделі бази даних, схеми, маршрути, сервіси, міграції Alembic та службові файли конфігурації. У клієнтській частині містяться React-компоненти, сторінки інтерфейсу, файли стилів, контексти авторизації та кошика, а також конфігурація

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

Vite. Такий поділ дозволяє окремо встановлювати залежності та запускати кожну частину системи.

Після завантаження проєкту потрібно встановити залежності. Для серверної частини це виконують у відповідній директорії backend. Менеджер uv створює або використовує віртуальне середовище та встановлює необхідні Python-бібліотеки, серед яких FastAPI, SQLAlchemy, Alembic, Pydantic та інші пакети, потрібні для роботи серверної логіки. Для клієнтської частини встановлення залежностей виконується через npm, який завантажує React, Vite, бібліотеки маршрутизації, допоміжні пакети для роботи із запитами та інші модулі, необхідні для побудови інтерфейсу. Після встановлення залежностей обидві частини системи можуть бути запущені.

Наступним важливим етапом є налаштування бази даних PostgreSQL. Перед запуском бекенд частини необхідно створити базу даних. Це можна зробити через pgAdmin 4 або за допомогою командного рядка PSQL. Після створення бази даних потрібно визначити користувача, пароль, назву бази, адресу сервера та порт підключення. Ці дані надалі вказуються у конфігураційному файлі серверної частини.

Підключення серверної частини до бази даних забезпечується за допомогою драйвера asynсpg, який дозволяє виконувати асинхронні запити до бази даних, що відповідає загальній асинхронній моделі роботи FastAPI [28].

Окрему увагу під час налаштування потрібно приділити конфігураційним файлам .env. У серверній частині вказуються параметри підключення до бази даних, секретний ключ для генерації та підпису JWT-токенів, алгоритм підпису, час дії токена, список дозволених адрес для CORS-політики, ключ API Нової Пошти, а також параметри LiqPay [29].

У клієнтській частині також створюється файл .env, у якому вказуються параметри, потрібні для її роботи. Основним параметром є адреса серверної частини, до якої надсилатимуться HTTP-запити. Через цю адресу клієнтська частина звертається до API сервера для отримання товарів, виконання

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		46

авторизації, роботи з кошиком, оформлення замовлення, отримання даних користувача та взаємодії з адміністративними маршрутами.

Після налаштування змінних середовища потрібно ініціалізувати структуру бази даних. Для цього використовується Alembic, який керує міграціями. Міграції дозволяють створити необхідні таблиці, типи даних і зв'язки між ними відповідно до моделей, визначених у серверній частині. Застосування міграцій є зручним, оскільки структура бази даних формується автоматично та може оновлюватися під час розвитку проєкту.

За потреби можна виконати наповнення бази тестовими даними, виконавши скрипт `seed.py`. Наявність тестових даних спрощує перевірку працездатності інтернет-магазину, оскільки після запуску `frontend` користувач одразу може побачити товари в каталозі, виконати пошук, застосувати фільтри та додати позиції до кошика. Також це корисно під час демонстрації програмного комплексу, оскільки система виглядає заповненою та придатною для перевірки основних сценаріїв.

Запуск серверної частини виконується введенням команди `uv run fastapi`, яка ініціює роботу FastAPI-застосунку через Uvicorn [30]. Після успішного запуску сервер починає приймати HTTP-запити на визначеній адресі та порту. Також стає доступною автоматично згенерована документація API, за допомогою якої можна перевірити маршрути.

Запуск клієнтської частини виконується введенням команди `npm run`. Ця команда запускає сервер Vite, який збирає React та відкриває його за локальною адресою. Після запуску користувач може перейти до головної сторінки інтернет-магазину через браузер.

Для можливості вибору міста та відділення для доставки необхідно налаштувати інтеграцію з API Нової Пошти. У серверному середовищі вказується посилання на API, після чого система може отримувати дані про міста та відділення. Під час оформлення замовлення користувач обирає місто, а система завантажує відповідний список відділень. Для оптимізації взаємодії з цим сервісом на клієнтській частині використовується кешування отриманих

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

даних. Це дозволяє повторно використовувати вже завантажені міста або відділення без зайвих запитів, що покращує швидкодію форми оформлення замовлення.

Для роботи онлайн-оплати потрібно налаштувати параметри LiqPay. У файлі .env серверної частини вказуються публічний і приватний ключі, які використовуються для формування платіжних даних. Під час вибору онлайн-оплати бекенд створює платіжні параметри, після чого користувач переходить до платіжної сторінки. Після завершення операції LiqPay надсилає callback на серверну частину, де результат платежу перевіряється та записується в замовлення, але оскільки локальний сервер зазвичай недоступний ззовні, платіжний сервіс не може напряду надіслати запит. Для вирішення цієї проблеми використовується утиліта ngrok [31]. Вона надає та перенаправляє з тимчасової публічної адреси на локальний сервер.

3.2. Структура інтерфейсу інтернет-магазину оргтехніки

Взаємодія користувача з інтернет-магазином починається з головної сторінки, яка представлена на рисунку 3.1. У верхній частині сторінки розташовані логотип з назвою магазину, поле пошуку, кнопки «Увійти» та «Кошик». Така структура дозволяє користувачу одразу перейти до основних дій: пошуку товару, авторизації або перегляду кошика. Зліва розташовані категорії, які користувач може вибрати, після чого в каталозі відображаються товари відповідної групи. Також користувач може скористатися пошуком за назвою товару або артикулом, а також фільтрами, які розташовані праворуч від каталогу товарів. У самому каталозі представлені картки товарів, на яких відображаються фото, назва, артикул, ціна та доступна кількість для покупки. Такий формат дає змогу швидко оцінити основну інформацію про товар без переходу на окрему сторінку. При натисканні на картку товару користувач переходить до сторінки вибраного товару, де може переглянути детальну інформацію про нього, зокрема опис та технічні характеристики. На цій сторінці користувач може уточнити деталі товару, перевірити його наявність і прийняти рішення щодо додавання товару до кошика.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

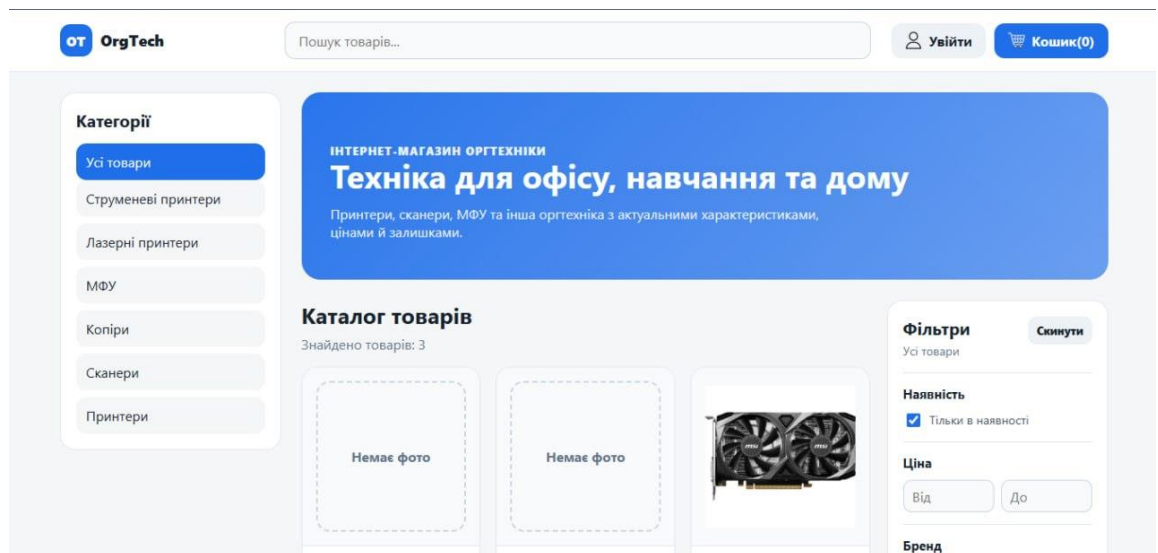


Рис. 3.1. Головна сторінка інтернет-магазину

Після натискання кнопки «Увійти» на шапці сайту, користувача перекидає на сторінку авторизації. На цій сторінці користувач може обрати дію, вибравши між вкладками «Вхід» та «Реєстрація». В залежності від обраної вкладки, користувач може увійти до існуючого облікового запису або створити новий, зареєструвавшись у системі.

Обравши вкладку «Вхід» (рис. 3.2) користувачу пропонується ввести email та пароль у відповідні поля. Після того, як користувач заповнив необхідні поля, перевіряється правильність формату вводу поля email, та чи достатньо символів для паролю ввів користувач. Далі перевіряється наявність email`у в базі даних інтернет-магазину та чи правильний пароль було введено.

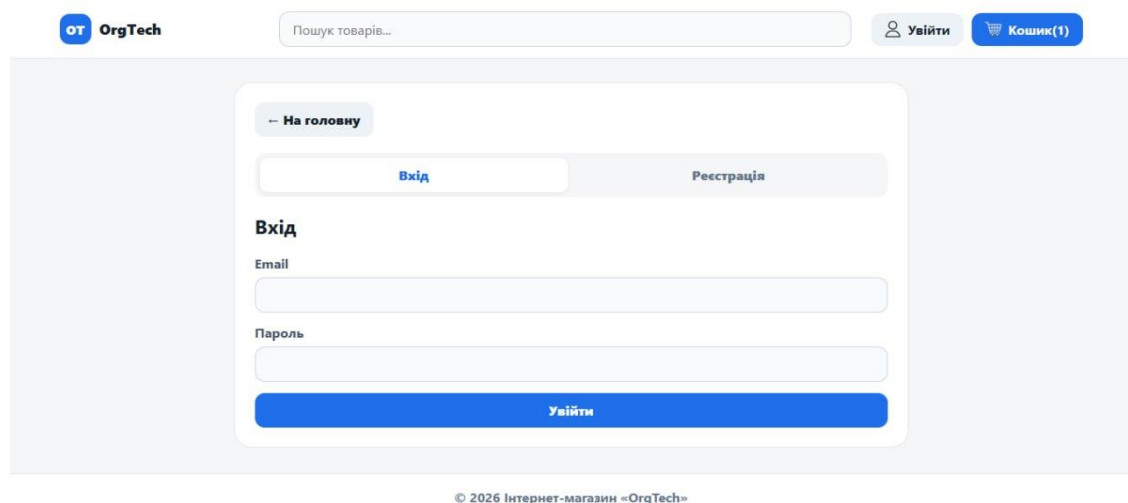


Рис. 3.2. Вхід у систему

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

Обравши вкладку «Реєстрація» (рис. 3.3) користувачу пропонується ввести email, пароль, ім'я, прізвище, та мобільний телефон у відповідні поля. Аналогічно вкладці «Вхід» відбувається перевірка формату введення email`у та на достатню кількість введених символів для пароля. Це дозволяє зменшити кількість помилок під час створення облікового запису та забезпечує коректне передавання даних на серверну частину системи.

Рис. 3.3. Реєстрація нового користувача

Після входу/реєстрації, користувача перекидає до сторінки облікового запису (рис. 3.4), де він може продивитись свої контактні дані, які були заповнені під час реєстрації.

Рис. 3.4. Сторінка облікового запису

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

За потреби, користувач може редагувати свої дані, натиснувши відповідну кнопку, після чого відкривається модальне вікно з можливістю редагувати ім'я, прізвище та номер телефону (рис. 3.5). Також є можливість вийти з облікового запису, натиснувши відповідну кнопку.

Рис. 3.5. Модальне вікно редагування контактних даних

Коли користувач обрав потрібні товари, він може перейти до кошика, натиснувши відповідну кнопку. Після натискання, його перекидує до сторінки кошика (рис. 3.6), де він бачить сформований список обраних товарів. Користувач має можливість змінити потрібну кількість одиниці товару або видалити товар із кошику. З правої сторони списку відображається вікно з підсумком, де описано загальну кількість товарів у кошику і загальну суму. Далі відображаються дві кнопки з можливістю оформити замовлення або повністю очистити кошик.

Рис. 3.6. Сторінка кошику

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Після натискання кнопки «Оформити замовлення» на сторінці кошику, користувача перекидує до сторінки оформлення замовлення (рис. 3.7), де йому потрібно заповнити форму для оформлення замовлення. Спочатку користувач заповнює свої контактні дані, які автоматично підтягуються із бази даних, які він заповнював під час реєстрації, що значно спрощує та прискорює процес оформлення замовлення. Далі користувачу необхідно заповнити поля місто та відділення. У полі міста користувачу необхідно ввести, як мінімум, 2 символи, перед тим, як йому висвітиться список доступних міст, після чого йому відкривається можливість заповнити поле відділення, яке показує список доступних відділень, в залежності від обраного міста. Після користувачу пропонується обрати спосіб оплати: готівкою при отриманні або онлайн, через платіжний сервіс LiqPay. Перед підтвердженням замовлення, користувач може, за бажанням, залишити коментар до замовлення. Після заповнення всіх необхідних полів, користувач натискає кнопку «Підтвердити замовлення».

Рис. 3.7. Сторінка оформлення замовлення

При вході у систему з роллю адміністратора, на верху сторінки з'являється кнопка для переходу до панелі адміністратора. Після натискання на кнопку адміністратора перекидує до сторінки адміністративної панелі (рис. 3.8), через яку він може дивитись статистику магазину: кількість активних категорій, товарів, загальну кількість замовлень, скільки з них очікують та кількість користувачів.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

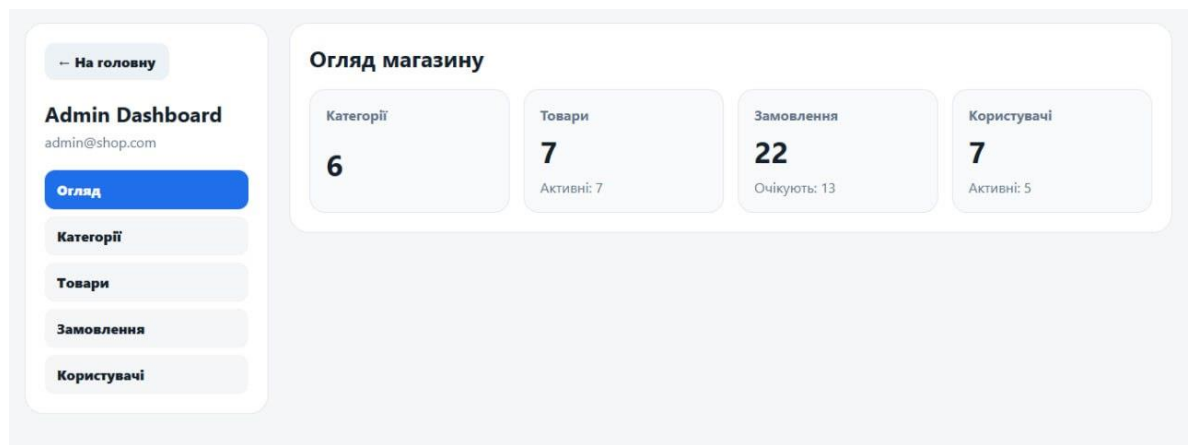


Рис. 3.8. Вкладка «Огляд» адмін-панелі

У вкладці «Категорії» (рис. 3.9), адміністратору доступні наступні операції для управління категоріями: переглядати поточний список категорій, додавати нові, редагувати або видаляти наявні категорії для підтримки актуальності каталогу.

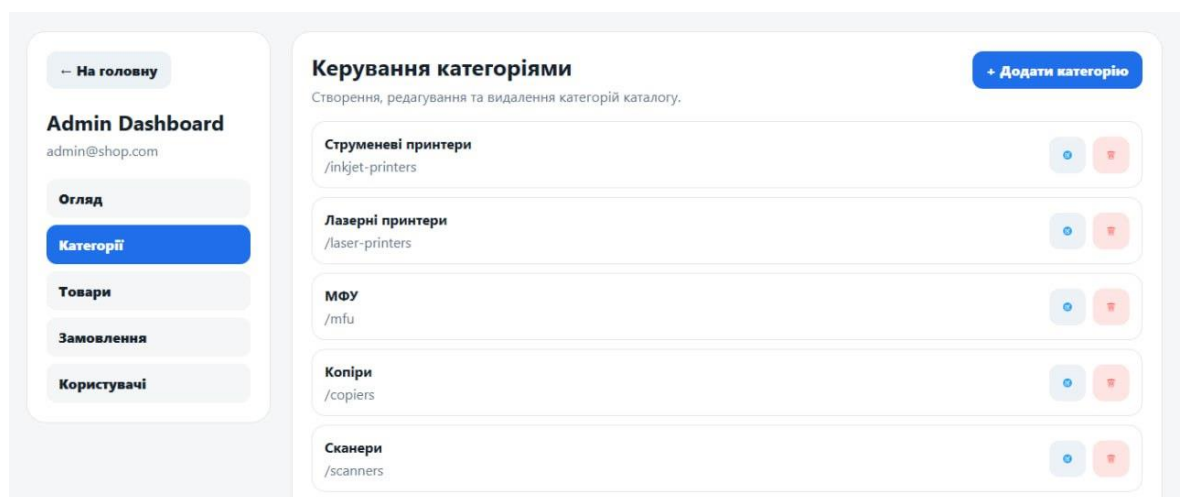


Рис. 3.9. Вкладка «Категорії»

У вкладці «Товари» (рис. 3.10), адміністратор може керувати асортиментом інтернет-магазину. Тут він може додавати новий товар до каталогу, прив'язавши його до відповідної категорії, редагувати наявні, наприклад змінити назву, категорію або технічні характеристики, та видалити з каталогу.

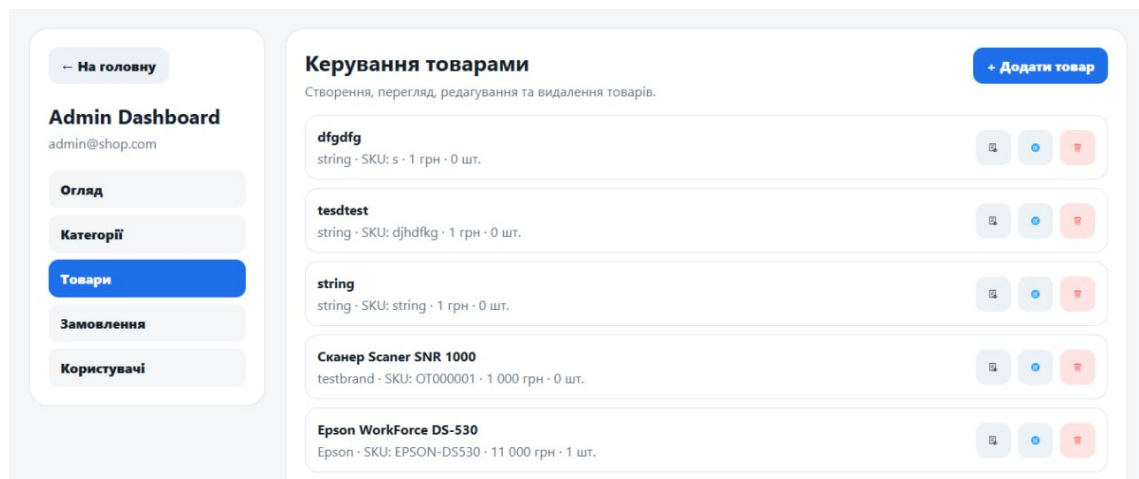


Рис. 3.10. Вкладка «Товари»

У вкладці «Замовлення» (рис. 3.11), адміністратор має можливість продивитись деталі кожного оформленого замовлення: склад кошика, дані покупця, спосіб оплати, місто та адреса відділення, на яке потрібно відправити замовлення. Також адміністратор може змінювати статус поточного замовлення.

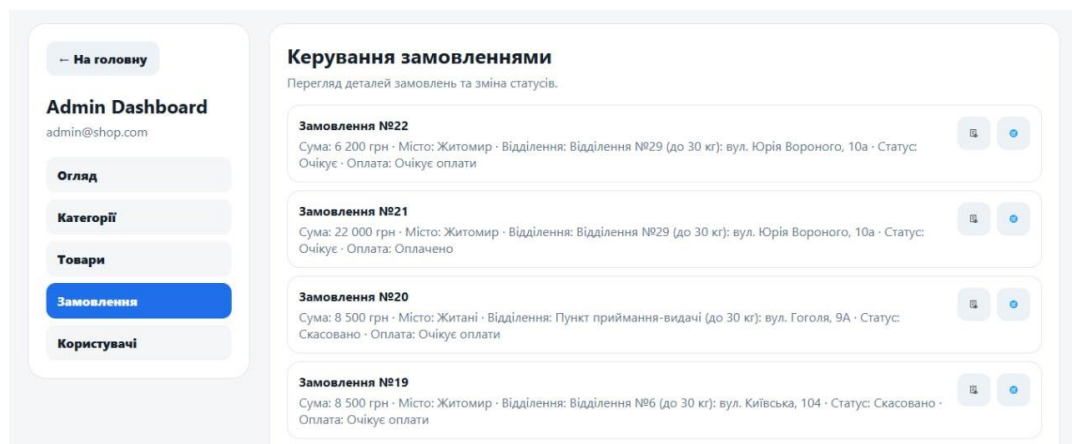


Рис. 3.11. Вкладка «Замовлення»

У вкладці «Користувачі» (рис. 3.12), адміністратор може продивлятися список усіх користувачів, оглянути детальну інформацію про обліковий запис, міняти роль, та деактивувати або повторно активувати обліковий запис користувача.

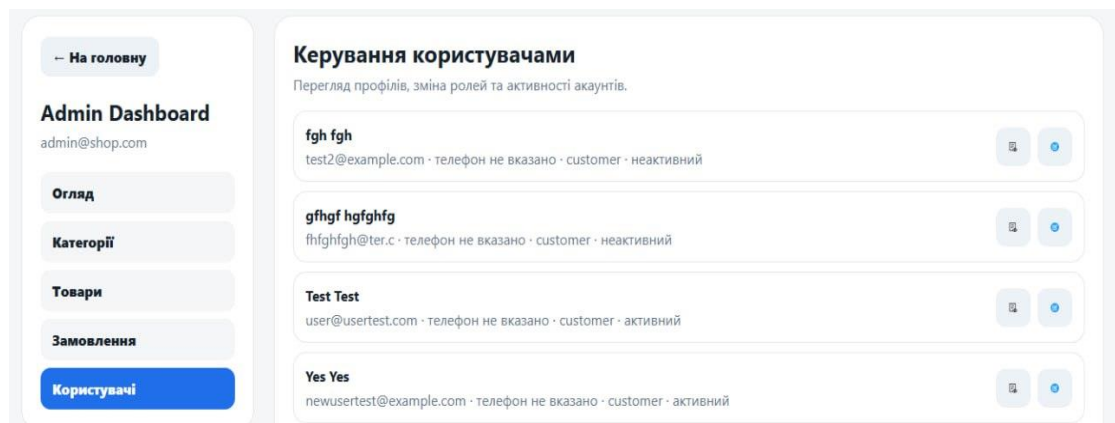


Рис. 3.12. Вкладка «Користувачі»

3.3. Тестування роботи інтернет-магазину оргтехніки

Тестування інтернет-магазину було проведено шляхом взаємодії користувача з системою, від входу/реєстрації до оформлення та онлайн-оплати замовлення. Воно потрібне для впевненості, що функціонал інтернет-магазину працює і готовий до запуску.

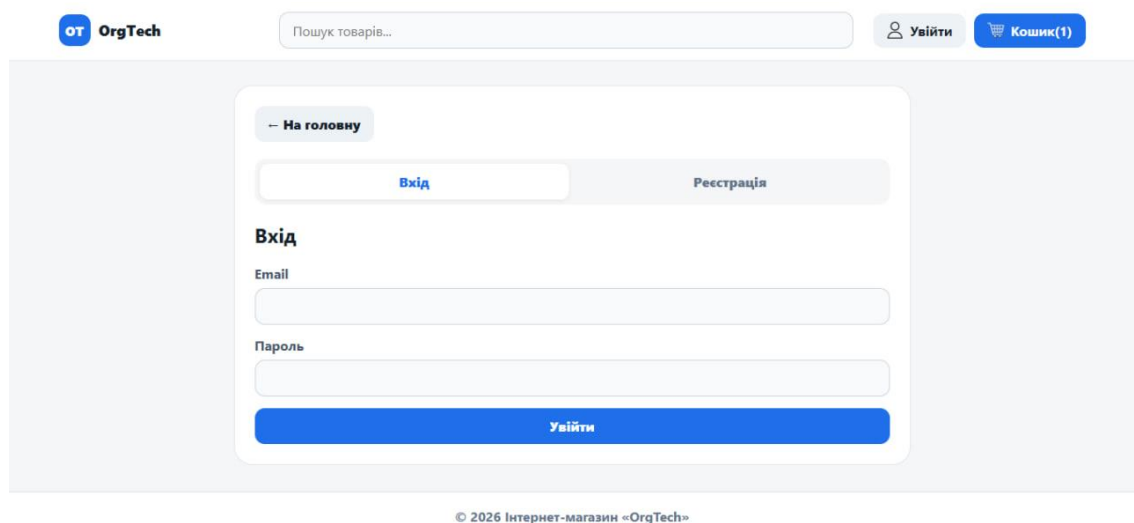


Рис. 3.13. Сторінка входу

На сторінці входу (рис. 3.13) для авторизації у вже існуючий обліковий запис користувач вводить свої email і пароль. Якщо користувач ще не зареєстрований, він може перейти до вкладки «Реєстрація» та заповнити форму реєстрації.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

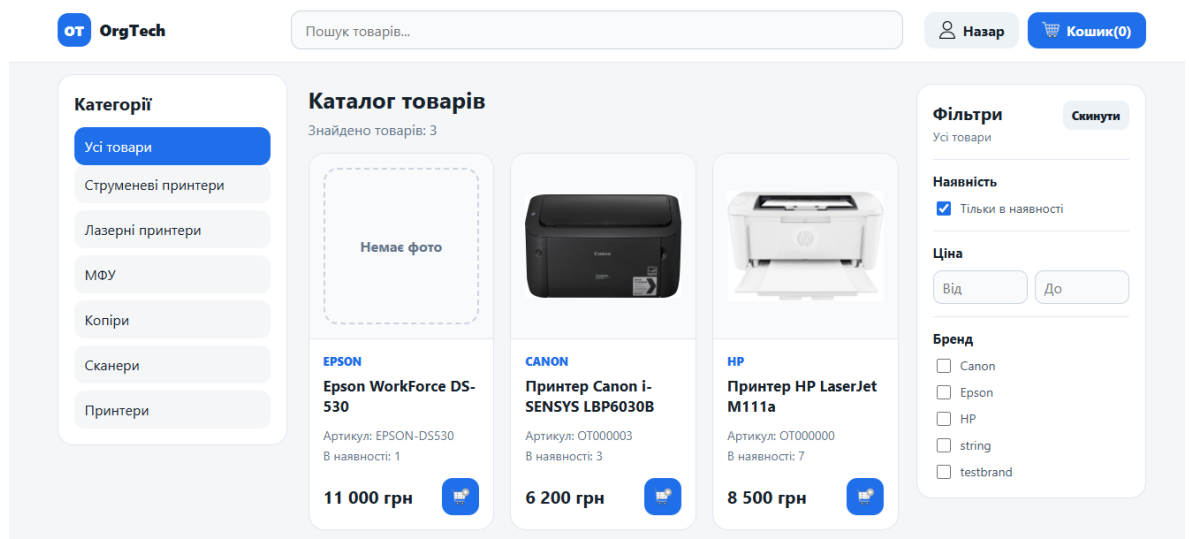


Рис. 3.14. Сторінка каталогу товарів

Після входу, користувач бачить сторінку облікового запису з його контактними даними. Щоб повернутися до каталогу (рис. 3.14), він може натиснути кнопку «На головну», або на логотип інтернет-магазину.

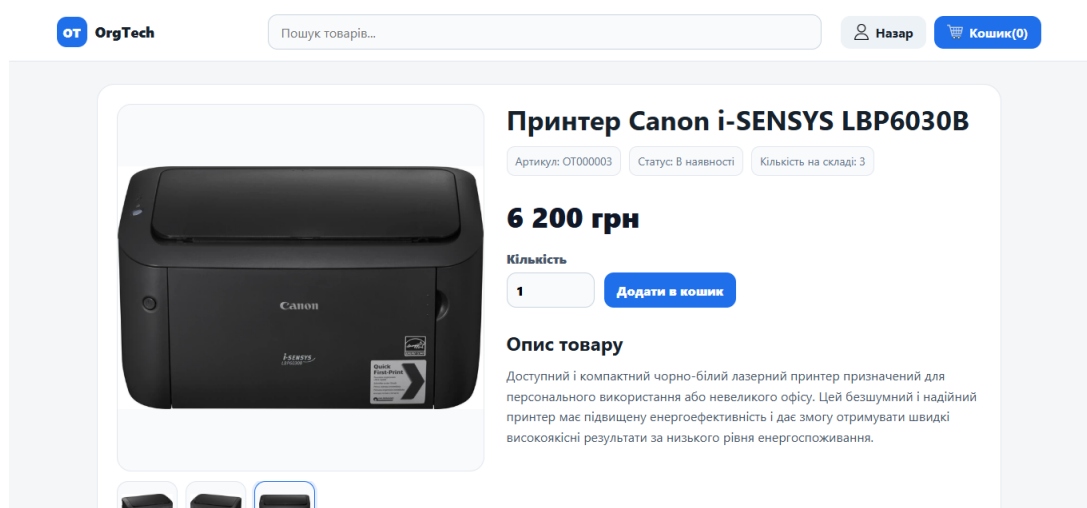


Рис. 3.15. Сторінка товару

Ознайомившись з каталогом, користувач може натиснути на картку товару, після чого відкривається сторінка з вибраним товаром (рис. 3.15), де користувач може детально ознайомитись з інформацією про відповідний товар, та додати його у кошик, з можливістю зазначити необхідну кількість.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

Рис. 3.16. Сторінка кошику

Після того, як користувач додав необхідні йому товари до кошика, він переходить до відповідної сторінки (рис. 3.16), де може змінити йому необхідну кількість одиниці товару, видалити товар з кошику, або взагалі повністю його очистити, натиснувши на кнопку, яка знаходиться під кнопкою «Оформити замовлення».

Рис. 3.17. Сторінка оформлення замовлення

Після натискання кнопки «Оформити замовлення» на сторінці кошика, користувача переводить до відповідної сторінки (рис. 3.17), де він заповнює форму для оформлення замовлення. Після заповнення необхідних полів, в залежності від вибраного способу оплати, користувачу виводиться повідомлення про успішне оформлення замовлення, якщо способом платежу було обрано готівку, або його переводить до сторінки платежу на LiqPay, де необхідно оплатити товар, і в залежності від статусу оплати, виводиться відповідна сторінка.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Номер картки

visa 4242 4242 4242 4242

Термін дії 05 / 30

CVV2 / CVC2

Ім'я власника картки*

Назар

Прізвище власника картки*

Баранівський

Email для отримання квитанції

example@domain.com

Натискаючи на кнопку «Оплатити», ви підтверджуєте що ознайомлені з переліком інформації про послугу та надаєте згоду з умовами [публічного договору](#)

Оплатити 6200.00 UAH

[Скасувати оплату](#)

Рис. 3.18. Оплата товару на LiqPay

Так як при оформленні замовлення було обрано онлайн-оплату, користувача перекидує до сторінки LiqPay (рис. 3.18), де він заповнює необхідні поля для проведення платежу, після чого натискає кнопку «Оплатити».

Дякуємо!

Платіж успішно відправлено

ID 2873553694

Дата 16.06.2026 10:15:10

Сума 6200.00 UAH

Завантажити квитанцію

Повернутися на сайт

Рис. 3.19. Повідомлення про успішний платіж на LiqPay

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		58

Якщо платіж пройшов успішно, користувачу виводиться відповідне повідомлення (рис. 3.19), йому пропонується завантажити квитанцію або повернутися на сайт.

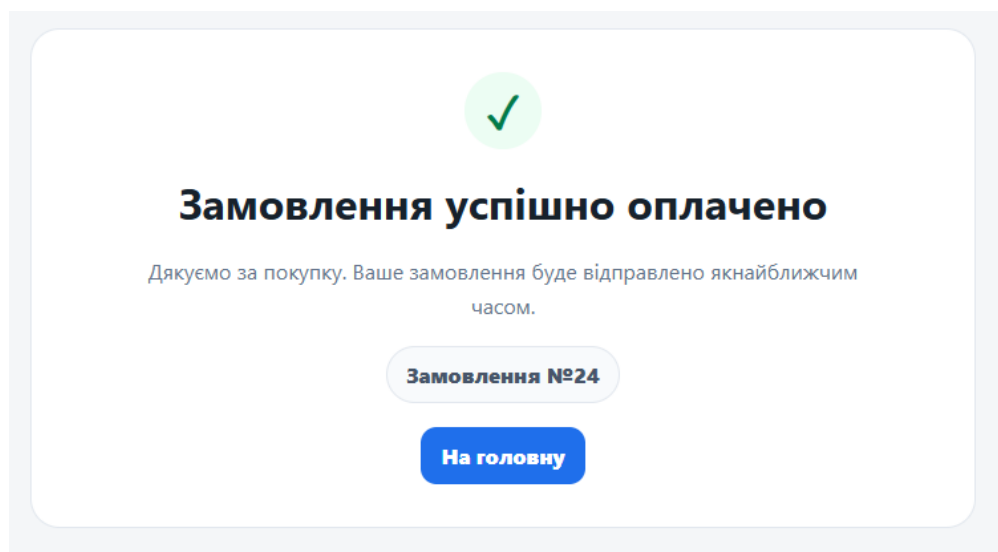


Рис. 3.20. Повідомлення про успішне оформлення замовлення на сайті

Після повернення до сайту, користувачу виводиться повідомлення про успішне оформлення замовлення (рис. 3.20), яке вже може бути оброблене та підтверджене адміністратором.

Висновки до третього розділу

У третьому розділі було розглянуто інтерфейс та порядок роботи з інтернет-магазином оргтехніки. Описано процес встановлення й налаштування інтернет-магазину оргтехніки, зокрема підготовку серверної та клієнтської частин, налаштування підключення до бази даних PostgreSQL, використання змінних середовища, запуск бекенду та фронтенду. Також було зазначено параметри, необхідні для коректної роботи авторизації, онлайн-оплати через LiqPay та API Нової Пошти.

У межах опису інтерфейсу було розглянуто основні сторінки інтернет-магазину: головну сторінку, сторінку окремого товару, кошик, сторінку оформлення замовлення, сторінку успішної оплати, особистий кабінет користувача та адміністративну панель. Було показано логіку взаємодії користувача із системою: перегляд товарів, пошук за назвою або артикулом, фільтрацію за ціною, наявністю та характеристиками, додавання товарів до

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

кошика, вибір способу оплати й доставки. Окремо описано можливості адміністратора, який через панель керування може працювати з категоріями, товарами, користувачами та замовленнями.

Також у розділі було розглянуто тестування роботи інтернет-магазину оргтехніки: реєстрацію та вхід, перегляд каталогу, додавання товарів до кошика, оформлення замовлення, та онлайн-оплату через LiqPay.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи бакалавра було розроблено інтернет-магазин оргтехніки. Було проведено аналіз аналогічних рішень: Rozetka, Click та RPC, визначено ключові вимоги до системи: зручний інтерфейс, зручна навігація, пошук і фільтрація товарів, інтеграція з платіжною системою LiqPay, отримання логістичної інформації через API Нової Пошти, а також наявність адміністративної панелі для керування інтернет-магазином.

Було спроектовано об'єктно-орієнтовану архітектуру проєкту, створено реляційну базу даних PostgreSQL для зберігання інформації про товари, категорії, замовлення, кошики та користувачів. Реалізовано функціонал входу/реєстрації, кабінет користувача з особистими даними та історією замовлень, кошик, пошук та фільтрація товарів. Окремо впроваджено панель адміністратора з можливістю керування товарами, категоріями, користувачами та замовленнями.

Було описано процес встановлення й налаштування інтернет-магазину оргтехніки, зокрема підготовку серверної та клієнтської частин, налаштування підключення до бази даних PostgreSQL, використання змінних середовища, запуск бекенду та фронтенду. Також було зазначено параметри, необхідні для коректної роботи авторизації, онлайн-оплати через LiqPay та API Нової Пошти. Інтерфейс інтернет-магазину є зрозумілим, а реалізований функціонал відповідає вимогам електронної комерції.

Практична цінність проєкту полягає в створенні функціонального та зручного інтернет-магазину оргтехніки. У перспективі на майбутнє передбачено можливість розширення функціоналу: залишення відгуків, система лояльності, інтеграція з іншими службами доставки, тощо.

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Інтернет-магазин ROZETKA™: офіційний сайт онлайн-гіпермаркету Розетка в Україні. URL: <https://rozetka.com.ua/> (дата звернення: 10.04.2026)
2. Click.ua – інтернет-магазин комп'ютерної техніки, електроніки та гаджетів в Україні. URL: <https://click.ua/> (дата звернення: 10.04.2026)
3. Комп'ютери, Оргтехніка, Мережеве та Торгове Обладнання – RPC. URL: <https://rpc.com.ua/> (дата звернення: 10.04.2026)
4. Quick Start – React. URL: <https://react.dev/learn> (дата звернення: 10.05.2026)
5. Getting Started | Vite. URL: <https://vite.dev/guide/> (дата звернення: 10.05.2026)
6. FastAPI – FastAPI. URL: <https://fastapi.tiangolo.com/> (дата звернення: 13.04.2026)
7. Concurrency and async / await – FastAPI. URL: <https://fastapi.tiangolo.com/async/> (дата звернення: 13.04.2026)
8. Security – FastAPI. URL: <https://fastapi.tiangolo.com/tutorial/security/> (дата звернення: 13.04.2026)
9. JSON Web Token Introduction – jwt.io. URL: <https://www.jwt.io/introduction#what-is-json-web-token> (дата звернення: 16.04.2026)
10. CSS: Cascading Style Sheets | MDN. URL: <https://developer.mozilla.org/en-US/docs/Web/CSS> (дата звернення: 10.05.2026)
11. Welcome to Python.org. URL: <https://www.python.org/> (дата звернення: 12.04.2026)
12. Welcome to Pydantic | Pydantic Docs. URL: <https://pydantic.dev/docs/validation/latest/get-started/> (дата звернення: 14.04.2026)
13. Models | Pydantic Docs. URL: <https://pydantic.dev/docs/validation/latest/concepts/models/> (дата звернення: 14.04.2026)

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

14. Fields | Pydantic Docs. URL: <https://pydantic.dev/docs/validation/latest/concepts/fields/> (дата звернення: 14.04.2026)
15. GitHub – pyca/bcrypt: Modern(-ish) password hashing for your software and your servers. URL: <https://github.com/pyca/bcrypt/> (дата звернення: 15.04.2026)
16. PostgreSQL: The world’s most advanced open source database. URL: <https://www.postgresql.org/> (дата звернення: 15.04.2026)
17. PostgreSQL: Documentation: 18: PostgreSQL 18.4 Documentation. URL: <https://www.postgresql.org/docs/18/index.html> (дата звернення: 15.04.2026)
18. SQLAlchemy Documentation – SQLAlchemy 2.0 Documentation. URL: <https://docs.sqlalchemy.org/en/20/> (дата звернення: 15.04.2026)
19. ORM Quick Start – SQLAlchemy 2.0 Documentation. URL: <https://docs.sqlalchemy.org/en/20/orm/quickstart.html> (дата звернення: 15.04.2026)
20. SQLAlchemy Core – SQLAlchemy 2.0 Documentation. URL: <https://docs.sqlalchemy.org/en/20/core/index.html> (дата звернення: 15.04.2026)
21. PostgreSQL – SQLAlchemy 2.0 Documentation. URL: <https://docs.sqlalchemy.org/en/20/dialects/postgresql.html> (дата звернення: 15.04.2026)
22. Welcome to Alembic’s documentation! – Alembic Documentation. URL: <https://alembic.sqlalchemy.org/en/latest/index.html> (дата звернення: 15.04.2026)
23. uv. URL: <https://docs.astral.sh/uv/> (дата звернення: 13.04.2026)
24. Visual Studio Code – The open source AI code editor | Your home for multi-agent development. URL: <https://code.visualstudio.com/> (дата звернення: 12.04.2026)
25. Documentation for Visual Studio Code. URL: <https://code.visualstudio.com/docs> (дата звернення: 12.04.2026)
26. Огляд методів | API Docs | NovaPost. URL: <https://api-portal.novapost.com/metodi-1/metodi> (дата звернення: 20.05.2026)
27. LiqPay – APIs. URL: <https://www.liqpay.ua/uk/doc/api> (дата звернення: 21.05.2026)

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

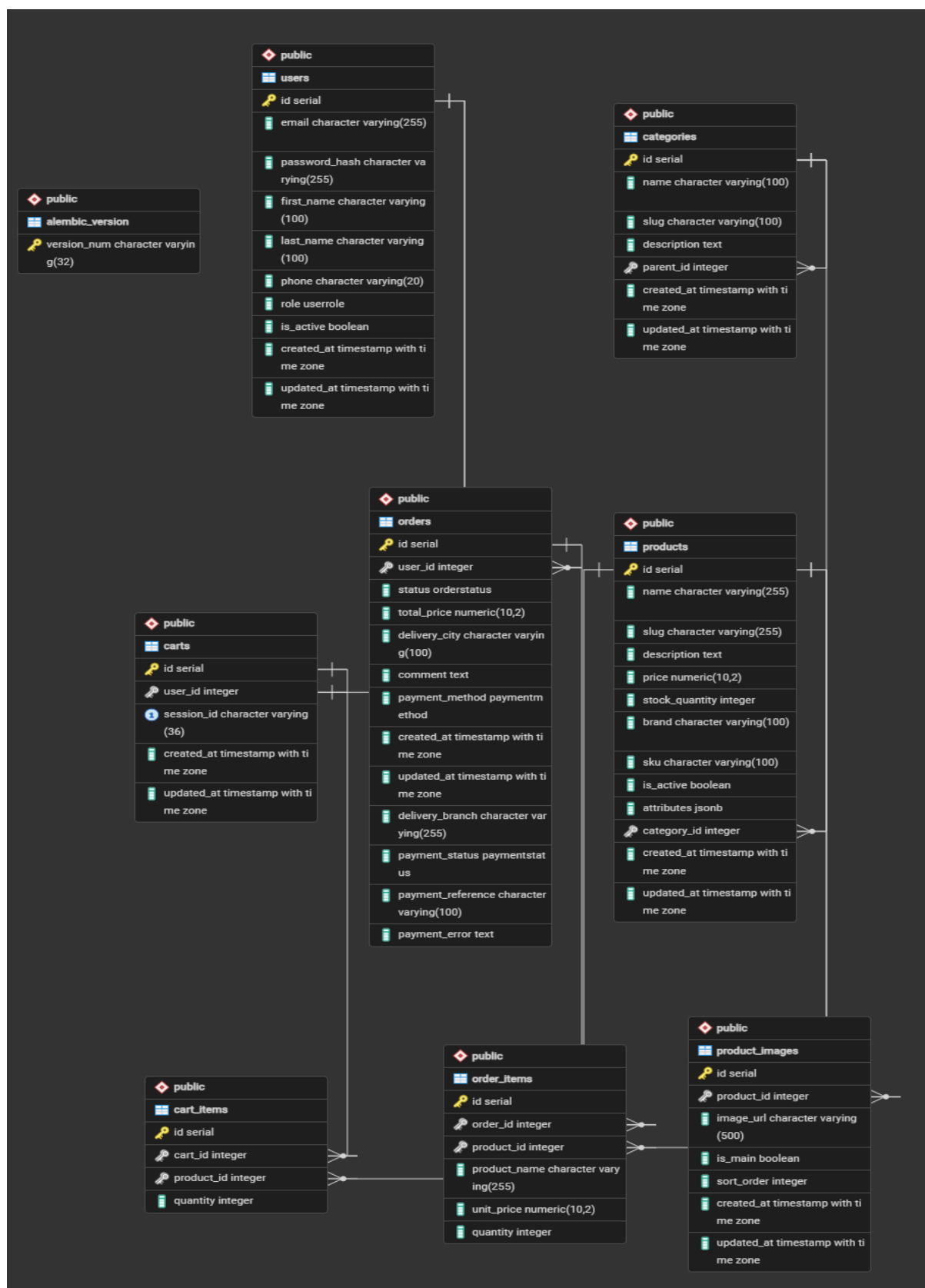
28. asyncpg – asyncpg Documentation. URL: <https://magicstack.github.io/asyncpg/current/index.html> (дата звернення: 15.04.2026)
29. CORS (Cross-Origin Resource Sharing) – FastAPI. URL: <https://fastapi.tiangolo.com/tutorial/cors/> (дата звернення: 10.05.2026)
30. Index – Uvicorn. URL: <https://uvicorn.dev/> (дата звернення: 13.04.2026)
31. Share Localhost Overview – ngrok documentation. URL: <https://ngrok.com/docs/guides/share-localhost/overview> (дата звернення: 22.05.2026)

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

ДОДАТКИ

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Схема БД інтернет-магазину



Бізнес-логіка кошика

```

from uuid import uuid4

from sqlalchemy.ext.asyncio import AsyncSession

from app.exceptions import InsufficientStockError, NotFoundError
from app.models.cart import Cart
from app.models.cart_item import CartItem
from app.repositories.carts import CartItemRepository, CartRepository
from app.repositories.products import ProductRepository
from app.schemas.cart import CartItemCreate, CartItemUpdate

class CartService:
    def __init__(self, db: AsyncSession):
        self.db = db
        self.cart_repository = CartRepository(db)
        self.item_repository = CartItemRepository(db)
        self.product_repository = ProductRepository(db)

    async def get_or_create_cart(
        self,
        user_id: int | None,
        session_id: str | None,
    ) -> tuple[Cart, str | None]:
        if user_id is not None:
            cart = await self.cart_repository.get_by_user_id(user_id)

            if cart:

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				67
Змн.	Арк.	№ докум.	Підпис	Дата		

```
return cart, None
```

```
cart = Cart(user_id=user_id)
```

```
created_cart = await self.cart_repository.create(cart)
```

```
await self.db.commit()
```

```
cart = await self.cart_repository.get_by_id(created_cart.id)
```

```
return cart, None
```

```
if session_id:
```

```
    cart = await self.cart_repository.get_by_session_id(session_id)
```

```
    if cart:
```

```
        return cart, None
```

```
new_session_id = str(uuid4())
```

```
cart = Cart(session_id=new_session_id)
```

```
created_cart = await self.cart_repository.create(cart)
```

```
await self.db.commit()
```

```
cart = await self.cart_repository.get_by_id(created_cart.id)
```

```
return cart, new_session_id
```

```
async def add_item(
```

```
    self,
```

```
    user_id: int | None,
```

```
    session_id: str | None,
```

```
    payload: CartItemCreate,
```

```
) -> tuple[Cart, str | None]:
```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

cart, new_session_id = await self.get_or_create_cart(user_id, session_id)

product = await self.product_repository.get_by_id(payload.product_id)

if not product or not product.is_active:
    raise NotFoundError("Товар")

existing_item = await self.item_repository.get_by_cart_and_product(
    cart.id,
    payload.product_id,
)

current_quantity = existing_item.quantity if existing_item else 0
new_quantity = current_quantity + payload.quantity

if new_quantity > product.stock_quantity:
    raise InsufficientStockError(product.name, product.stock_quantity)

if existing_item:
    await self.item_repository.update_quantity(existing_item,
new_quantity)
else:
    item = CartItem(
        cart_id=cart.id,
        product_id=payload.product_id,
        quantity=payload.quantity,
    )
    await self.item_repository.create(item)

await self.db.commit()

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

        updated_cart = await self.cart_repository.get_by_id(cart.id)
        return updated_cart, new_session_id

    async def update_item(
        self,
        user_id: int | None,
        session_id: str | None,
        item_id: int,
        payload: CartItemUpdate,
    ) -> Cart:
        cart, _ = await self.get_or_create_cart(user_id, session_id)

        item = await self.item_repository.get_by_id(item_id)

        if not item or item.cart_id != cart.id:
            raise NotFoundError("Позицію кошика")

        if payload.quantity > item.product.stock_quantity:
            raise InsufficientStockError(item.product.name,
            item.product.stock_quantity)

        await self.item_repository.update_quantity(item, payload.quantity)
        await self.db.commit()

        updated_cart = await self.cart_repository.get_by_id(cart.id)
        return updated_cart

    async def remove_item(
        self,

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        user_id: int | None,
        session_id: str | None,
        item_id: int,
    ) -> Cart:
        cart, _ = await self.get_or_create_cart(user_id, session_id)

        item = await self.item_repository.get_by_id(item_id)

        if not item or item.cart_id != cart.id:
            raise NotFoundError("Позицію кошика")

        await self.item_repository.delete(item)
        await self.db.commit()

        updated_cart = await self.cart_repository.get_by_id(cart.id)
        return updated_cart

    async def clear_cart(
        self,
        user_id: int | None,
        session_id: str | None,
    ) -> Cart:
        cart, _ = await self.get_or_create_cart(user_id, session_id)

        for item in list(cart.items):
            await self.item_repository.delete(item)

        await self.db.commit()

        updated_cart = await self.cart_repository.get_by_id(cart.id)

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

```
return updated_cart
```

```
async def merge_guest_cart_into_user_cart(
```

```
    self,
```

```
    user_id: int,
```

```
    session_id: str | None,
```

```
) -> None:
```

```
    if not session_id:
```

```
        return
```

```
    guest_cart = await self.cart_repository.get_by_session_id(session_id)
```

```
    if not guest_cart or not guest_cart.items:
```

```
        return
```

```
    user_cart = await self.cart_repository.get_by_user_id(user_id)
```

```
    if not user_cart:
```

```
        user_cart = Cart(user_id=user_id)
```

```
        created_user_cart = await self.cart_repository.create(user_cart)
```

```
        await self.db.flush()
```

```
        user_cart = await self.cart_repository.get_by_id(created_user_cart.id)
```

```
    for guest_item in list(guest_cart.items):
```

```
        product = guest_item.product
```

```
        if not product or not product.is_active or product.stock_quantity <= 0:
```

```
            continue
```

```
    existing_item = await self.item_repository.get_by_cart_and_product(
```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        user_cart.id,
        product.id,
    )

    if existing_item:
        merged_quantity = existing_item.quantity + guest_item.quantity
        final_quantity = min(merged_quantity, product.stock_quantity)

    if final_quantity > 0:
        await self.item_repository.update_quantity(
            existing_item, final_quantity
        )
    else:
        final_quantity = min(guest_item.quantity, product.stock_quantity)

    if final_quantity > 0:
        new_item = CartItem(
            cart_id=user_cart.id,
            product_id=product.id,
            quantity=final_quantity,
        )
        await self.item_repository.create(new_item)

    await self.cart_repository.delete(guest_cart)
    await self.db.commit()

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

Бізнес-логіка замовлень

```

from decimal import Decimal

from sqlalchemy.ext.asyncio import AsyncSession

from app.core.constants import ORDER_STATUS_TRANSITIONS
from app.exceptions import InsufficientStockError, NotFoundError,
ValidationError

from app.models.enums import PaymentMethod, PaymentStatus
from app.models.order import Order
from app.models.order_item import OrderItem
from app.repositories.carts import CartItemRepository, CartRepository
from app.repositories.orders import OrderItemRepository, OrderRepository
from app.repositories.products import ProductRepository
from app.schemas.order import OrderCreate, OrderStatusUpdate


class OrderService:
    def __init__(self, db: AsyncSession):
        self.db = db
        self.order_repository = OrderRepository(db)
        self.order_item_repository = OrderItemRepository(db)
        self.cart_repository = CartRepository(db)
        self.cart_item_repository = CartItemRepository(db)
        self.product_repository = ProductRepository(db)

    async def list_orders(
        self,
        skip: int = 0,

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        limit: int = 100,
        user_id: int | None = None,
    ) -> list[Order]:
        return await self.order_repository.list(
            skip=skip,
            limit=limit,
            user_id=user_id,
        )

    async def get_order(self, order_id: int, user_id: int | None = None) -> Order:
        order = await self.order_repository.get_by_id(order_id)

        if not order:
            raise NotFoundError("Замовлення")

        if user_id is not None and order.user_id != user_id:
            raise NotFoundError("Замовлення")

        return order

    async def create_order_from_cart(
        self,
        user_id: int,
        payload: OrderCreate,
    ) -> Order:
        cart = await self.cart_repository.get_by_user_id(user_id)

        if not cart or not cart.items:
            raise ValidationError("Кошик порожній")

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```

total_price = Decimal("0.00")

for item in cart.items:
    product = item.product

    if not product or not product.is_active:
        raise NotFoundError("Товар")

    if item.quantity > product.stock_quantity:
        raise InsufficientStockError(product.name, product.stock_quantity)

    total_price += product.price * item.quantity

order = Order(
    user_id=user_id,
    total_price=total_price,
    delivery_city=payload.delivery_city,
    delivery_branch=payload.delivery_branch,
    comment=payload.comment,
    payment_method=payload.payment_method,
    payment_status=PaymentStatus.PENDING,
)

created_order = await self.order_repository.create(order)

for item in cart.items:
    product = item.product

    order_item = OrderItem(
        order_id=created_order.id,

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

        product_id=product.id,
        product_name=product.name,
        unit_price=product.price,
        quantity=item.quantity,
    )

```

```

    await self.order_item_repository.create(order_item)

```

```

    product.stock_quantity -= item.quantity
    await self.product_repository.update(
        product,
        {"stock_quantity": product.stock_quantity},
    )

```

```

    await self.cart_item_repository.delete(item)

```

```

    await self.db.commit()

```

```

    order = await self.order_repository.get_by_id(created_order.id)

```

```

    return order

```

```

async def get_online_payment_order(self, order_id: int, user_id: int) ->

```

Order:

```

    order = await self.get_order(order_id, user_id=user_id)

```

```

    if order.payment_method != PaymentMethod.ONLINE:

```

```

        raise ValidationError("Для цього замовлення обрано оплату
готівкою")

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

if order.payment_status == PaymentStatus.PAID:
    raise ValidationError("Це замовлення вже оплачено")

return order

async def apply_liqpay_result(self, payload: dict) -> Order:
    from app.services.liqpay import LiqPayService

    liqpay = LiqPayService()
    order_id = liqpay.parse_order_id(payload.get("order_id"))

    if not order_id:
        raise ValidationError("Некоректний номер замовлення LiqPay")

    order = await self.order_repository.get_by_id(order_id)

    if not order:
        raise NotFoundError("Замовлення")

    status = payload.get("status")
    payment_reference = str(
        payload.get("payment_id") or payload.get("liqpay_order_id") or ""
    ) or None
    payment_error = payload.get("err_description") or
payload.get("err_code")

    if status in {"sandbox", "success", "wait_accept", "wait_compensation"}:
        payment_status = PaymentStatus.PAID
        payment_error = None
    elif status in {"failure", "error"}:

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		78

```

        payment_status = PaymentStatus.FAILED
    elif status == "reversed":
        payment_status = PaymentStatus.CANCELLED
    else:
        payment_status = PaymentStatus.PENDING

    updated_order = await self.order_repository.update_payment(
        order,
        payment_status,
        payment_reference=payment_reference,
        payment_error=payment_error,
    )
    await self.db.commit()

    order = await self.order_repository.get_by_id(updated_order.id)

    return order

async def update_order_status(
    self,
    order_id: int,
    payload: OrderStatusUpdate,
) -> Order:
    order = await self.order_repository.get_by_id(order_id)

    if not order:
        raise NotFoundError("Замовлення")

    allowed_statuses = ORDER_STATUS_TRANSITIONS.get(order.status,

```

[1]

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		79

```
if payload.status not in allowed_statuses:
```

```
    raise ValidationError(
```

```
        f"Перехід зі статусу '{order.status.value}' "
```

```
        f"у статус '{payload.status.value}' заборонений"
```

```
    )
```

```
    updated_order = await self.order_repository.update_status(order,  
payload.status)
```

```
    await self.db.commit()
```

```
    order = await self.order_repository.get_by_id(updated_order.id)
```

```
    return order
```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		80

Бізнес-логіка товарів

```

from sqlalchemy.ext.asyncio import AsyncSession

from app.exceptions import AlreadyExistsError, NotFoundError,
ValidationError

from app.models.product import Product
from app.repositories.categories import CategoryRepository
from app.repositories.products import ProductRepository
from app.schemas.product import ProductCreate, ProductUpdate


class ProductService:

    def __init__(self, db: AsyncSession):
        self.db = db

        self.repository = ProductRepository(db)
        self.category_repository = CategoryRepository(db)

    async def list_products(
        self,
        skip: int = 0,
        limit: int = 100,
        category_id: int | None = None,
        only_active: bool = False,
    ) -> list[Product]:
        if category_id is not None:
            category = await self.category_repository.get_by_id(category_id)

            if not category:
                raise NotFoundError("Категорію")

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		81

```

return await self.repository.list(
    skip=skip,
    limit=limit,
    category_id=category_id,
    only_active=only_active,
)

```

```

async def get_product(self, product_id: int) -> Product:
    product = await self.repository.get_by_id(product_id)

    if not product:
        raise NotFoundError("Товар")

```

```

return product

```

```

async def get_product_by_slug(self, slug: str) -> Product:
    product = await self.repository.get_by_slug(slug)

    if not product:
        raise NotFoundError("Товар")

```

```

return product

```

```

async def create_product(self, payload: ProductCreate) -> Product:
    category = await self.category_repository.get_by_id(payload.category_id)

    if not category:
        raise ValidationError("Категорію товару не знайдено")

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		82

```

existing_slug = await self.repository.get_by_slug(payload.slug)

if existing_slug:
    raise AlreadyExistsError("Товар з таким slug")

if payload.sku:
    existing_sku = await self.repository.get_by_sku(payload.sku)

    if existing_sku:
        raise AlreadyExistsError("Товар з таким SKU")

product = Product(**payload.model_dump())

created_product = await self.repository.create(product)
await self.db.commit()

return await self.get_product(created_product.id)

async def update_product(
    self,
    product_id: int,
    payload: ProductUpdate,
) -> Product:
    product = await self.repository.get_by_id(product_id)

    if not product:
        raise NotFoundError("Товар")

    update_data = payload.model_dump(exclude_unset=True)

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

if "category_id" in update_data and update_data["category_id"] is not None:

```
category = await self.category_repository.get_by_id(
    update_data["category_id"]
)
```

if not category:

```
raise ValidationError("Категорію товару не знайдено")
```

if "slug" in update_data:

```
existing_slug = await self.repository.get_by_slug(update_data["slug"])
```

if existing_slug and existing_slug.id != product.id:

```
raise AlreadyExistsError("Товар з таким slug")
```

if "sku" in update_data and update_data["sku"]:

```
existing_sku = await self.repository.get_by_sku(update_data["sku"])
```

if existing_sku and existing_sku.id != product.id:

```
raise AlreadyExistsError("Товар з таким SKU")
```

```
await self.repository.update(product, update_data)
```

```
await self.db.commit()
```

```
return await self.get_product(product_id)
```

async def delete_product(self, product_id: int) -> None:

```
product = await self.repository.get_by_id(product_id)
```

if not product:

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

raise NotFoundError("Товар")

await self.repository.delete(product)

await self.db.commit()

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

Бізнес логіка користувачів

```

from sqlalchemy.ext.asyncio import AsyncSession

from app.exceptions import NotFoundError
from app.models.user import User
from app.repositories.users import UserRepository
from app.schemas.user import UserUpdate, UserAdminUpdate

class UserService:
    def __init__(self, db: AsyncSession):
        self.db = db
        self.user_repository = UserRepository(db)

    async def list_users(self, skip: int = 0, limit: int = 100) -> list[User]:
        return await self.user_repository.list(skip=skip, limit=limit)

    async def get_user(self, user_id: int) -> User:
        user = await self.user_repository.get_by_id(user_id)

        if not user:
            raise NotFoundError("Користувача")

        return user

    async def update_user(self, user_id: int, payload: UserUpdate) -> User:
        user = await self.user_repository.get_by_id(user_id)

        if not user:

```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

```
raise NotFoundError("Користувача")
```

```
update_data = payload.model_dump(exclude_unset=True)
```

```
updated_user = await self.user_repository.update(user, update_data)
```

```
await self.db.commit()
```

```
await self.db.refresh(updated_user)
```

```
return updated_user
```

```
async def admin_update_user(self, user_id: int, payload: UserAdminUpdate)
```

-> User:

```
user = await self.user_repository.get_by_id(user_id)
```

```
if not user:
```

```
raise NotFoundError("Користувача")
```

```
update_data = payload.model_dump(exclude_unset=True)
```

```
updated_user = await self.user_repository.update(user, update_data)
```

```
await self.db.commit()
```

```
await self.db.refresh(updated_user)
```

```
return updated_user
```

		Баранівський Н.М.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Вакалюк Т.А.				
Змн.	Арк.	№ докум.	Підпис	Дата		87