

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 музичних інструментів»

Виконав студент групи ПЗ-21-5

ЛЕБІГА Олексій Сергійович

Керівник роботи:

КОНЦИДАЙЛО Андрій Михайлович

Рецензент:

ВЛАСЕНКО Олег Васильович

Житомир – 2025

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

«26» лютого 2025р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЛЕБІГА Олексій Сергійович**

Керівник роботи: **КОНЦИДАЙЛО Андрій Михайлович**

Тема роботи: **«Інтернет-магазин музичних інструментів»,**

затверджена наказом закладу вищої освіти від **«14» лютого 2025р., №61/с**

Вихідні дані для роботи:

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Концидайло А.М.	27.02.2025	27.02.2025
2	Концидайло А.М.	27.02.2025	27.02.2025
3	Концидайло А.М.	27.02.2025	27.02.2025

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «бакалавр» присвячена розробці інтернет-магазину музичних інструментів. Робота складається з розробленого програмного продукту та пояснювальної записки. Пояснювальна записка містить вступ, три розділи, висновки, список використаних джерел та додатки.

Загальний обсяг роботи становить 81 сторінок. Текстова частина пояснювальної записки містить 63 сторінок, 37 ілюстрацій та 17 таблиць.

Метою роботи є створення сучасного, функціонального та зручного інтернет-магазину, який дозволяє користувачам знаходити, порівнювати та купувати музичні інструменти й аксесуари, а адміністраторам ефективно керувати товарною базою, обробляти замовлення та взаємодіяти з клієнтами.

У процесі роботи було проведено аналіз предметної області та існуючих рішень на ринку, на основі чого були сформульовані основні завдання для створення системи. До них належать проектування архітектури вебзастосунку, розробка модулів для користувачів (покупців) і адміністраторів, реалізація каталогу товарів з можливістю пошуку, створення системи кошика, оформлення замовлень, а також реалізація особистого кабінету користувача з історією покупок.

Для розробки було обрано клієнт-серверну архітектуру. Серверна частина реалізована на базі середовища виконання Node.js з використанням вебфреймворку Express.js, що забезпечує швидку та асинхронну обробку запитів. Для зберігання інформації про товари, користувачів та замовлення застосовано нереляційну документо-орієнтовану базу даних MongoDB.

Ключові слова: ІНТЕРНЕТ-МАГАЗИН, МУЗИЧНІ ІНСТРУМЕНТИ, NODE.JS, EXPRESS.JS, MONGODB, JWT, HANDLEBARS.

					ІПЗ.КР.Б-121.25 - ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Лебіга О.С.			Інтернет-магазин музичних інструментів	Лім.	Арк.	Аркушів
Керівник		Концидайло А.М.					3	63
						Житомирська Політехніка, група ІПЗ-21-5		
Рецензент		Власенко О.В.						
Зав. кафедри		Вакалюк Т.А.						

ABSTRACT

The Bachelor's qualification work is dedicated to the development of an online store for musical instruments. The work consists of a developed software product and an explanatory note. The explanatory note includes an introduction, three chapters, conclusions, a list of references, and appendices.

The total volume of the work is 81 pages. The text part of the explanatory note contains 63 pages, 37 illustrations, and 17 tables.

The aim of the work is to create a modern, functional, and user-friendly online store that allows users to find, compare, and purchase musical instruments and accessories, and allows administrators to effectively manage the product database, process orders, and interact with customers.

In the course of the work, an analysis of the subject area and existing market solutions was conducted, which formed the basis for defining the main tasks for the system's creation. These include designing the web application architecture, developing modules for users (customers) and administrators, implementing a product catalog with advanced filtering and search capabilities, creating a shopping cart system, order placement functionality, and implementing a user account with purchase history.

A client-server architecture was chosen for the development. The server-side is implemented based on the Node.js runtime environment using the Express.js web framework, which ensures fast and asynchronous request processing. For storing information about products, users, and orders, the non-relational document-oriented database MongoDB was used.

Keywords: ONLINE STORE, MUSICAL INSTRUMENTS, NODE.JS, EXPRESS.JS, MONGODB,JWT, HANDLEBARS.

		Лебіга О.С.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

Зміст

РЕФЕРАТ	3
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В РОЗРОБЦІ САЙТУ ДЛЯ МУЗИЧНИХ ІНСТРУМЕНТІВ	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	16
Висновки до першого розділу.....	19
РОЗДІЛ 2. ПРОЕКТУВАННЯ САЙТУ МУЗИЧНИХ ІНСТРУМЕНТІВ	20
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	20
2.2. Розробка системи збереження даних	25
2.3. Проектування та реалізація алгоритмів роботи системи.....	31
2.4. Реалізація функціоналу	35
Висновки до другого розділу	39
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	40
3.1 Порядок встановлення та налаштування системи.....	40
3.3 Порядок встановлення та налаштування системи.....	57
Висновки до третього розділу.....	59
ВИСНОВКИ.....	60
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	61
ДОДАТКИ.....	64

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface

JWT - JSON Web Token

MVC - Model-View-Controller

БД - база даних

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми зумовлена стрімким переходом до цифрового суспільства та постійним збільшенням популярності музичного мистецтва, а відтак зростає потреба у якісних музичних інструментах та обладнанні. Дедалі більше людей, від новачків-аматорів до професійних музикантів, активно практикуються у музиці, що створює запит на зручні та надійні платформи для придбання необхідної продукції. Інтернет-магазини в цій ніші стають ключовими гравцями, адже вони долають географічні бар'єри та пропонують доступ до набагато ширшого вибору товарів, ніж звичайні магазини. Також вони дозволяють покупцям ретельно досліджувати характеристики товарів, порівнювати різні моделі та ознайомлюватися з відгуками інших користувачів, що є вкрай важливим при виборі музичного інструмента. Отже, створення сучасного, функціонального та орієнтованого на клієнта інтернет-магазину є надзвичайно важливим завданням, яке відповідає як потребам сучасного ринку електронної комерції, так і потребам музичного співтовариства.

Метою випускної кваліфікаційної роботи є розробка функціональної, зручної та безпечної веб-платформи для продажу музичних інструментів, яка забезпечить зручну та ефективну взаємодію користувачів з каталогом товарів, дозволить здійснювати пошук товарів, додавати їх у кошик, оформлювати замовлення та оплачувати покупки. Крім того, платформа має на меті сприяти автоматизації внутрішніх процесів інтернет-магазину, зокрема управління товарним асортиментом, обробки замовлень та збору відгуків.

Для реалізації поставленої мети потрібно виконати наступні завдання:

1. Провести аналіз існуючих аналогів подібних платформ.
2. Здійснити вибір засобів проектування.
3. Розробити архітектуру платформи (сайту), а саме розробити серверну частину та клієнтську частину.
4. Провести тестування сайту для виявлення та усунення помилок, а також оцінити якість створеного програмного продукту.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Об'єкт дослідження це процес створення, розгортання та впровадження веб-орієнтованого комерційного додатку, призначеного для онлайн-торгівлі у сфері музичних інструментів. До об'єкта дослідження входить розробка інтерфейсів взаємодії, структурування даних, а також інтеграція із зовнішніми сервісами, такими як платіжні системи та служби доставки, для покращення користувацького досвіду.

Предмет дослідження це методи, технології та інструменти проектування, розробки та тестування сучасних веб-платформ комерційного призначення (інтернет-магазинів). Особлива увага приділяється підходам до реалізації клієнт-серверної архітектури, адаптивного дизайну, систем управління кошиком та онлайн-оплати, засобам автентифікації, а також забезпеченню зручності користування та безпеки персональних даних користувачів.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ЗАСТОСУВАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В РОЗРОБЦІ САЙТУ ДЛЯ МУЗИЧНИХ ІНСТРУМЕНТІВ

1.1. Постановка задачі

Основною метою даної кваліфікаційної роботи є розробка багатофункціонального вебсайту інтернет-магазину музичних товарів, що базується на сучасних підходах та технологіях веброзробки. Передбачається створення гнучкої платформи, яка з одного боку забезпечить адміністратору магазину прості та ефективні інструменти для оновлення інформації, можливість легкого розширення функціоналу та подальшого вдосконалення проєкту. З іншого боку, для кінцевих користувачів цей вебресурс повинен стати надійним та зручним провідником у світ музики, гарантуючи легкий доступ до актуального асортименту товарів, вичерпної інформації про продукцію, а також інтуїтивно зрозумілий інтерфейс для здійснення покупок та оформлення замовлень.

На сьогоднішній день ринок інтернет-торгівлі музичними товарами є надзвичайно насиченим та конкурентним як в Україні, так і за її межами. Існуючі онлайн-платформи зазвичай надають користувачам досить стандартний набір можливостей, що включає докладні описи товарів, їхні технічні характеристики, фотографії, відомості про наявність, ціни та умови доставки. Яскравим прикладом успішної реалізації є Thomann, один з найбільших інтернет-магазинів музичних товарів у Європі, сайт якого вирізняється зручною навігацією, потужними фільтрами за категоріями, системою відгуків покупців та багатомовною підтримкою [1]. Подібні функціональні можливості реалізовано на інших популярних світових ресурсах, таких як Sweetwater у США [2] чи Gear4music у Великій Британії [3], а також у великій кількості українських онлайн-магазинів, зокрема Музикант [4], Soundmag [5] або Jam [6]. Усі ці сайти дозволяють користувачам не тільки обрати необхідний інструмент чи обладнання, але й замовити доставку, отримати кваліфіковану онлайн-консультацію, ознайомитися з рекомендаціями та прочитати огляди.

Враховуючи високу конкуренцію та розвиненість ринку, для того щоб стояти на рівні з провідними європейськими магазинами, розроблюваний

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

продукт повинен бути чимось більшим, ніж просто каталогом товарів. Він має стати повноцінною інтерактивною платформою, що забезпечує максимально комфортний процес вибору, надає зручні інструменти для порівняння товарів за характеристиками та гарантує безпеку покупки, щоб користувач відчував спокій та впевненість на кожному етапі, не виходячи з дому.

Сам процес створення такої багатофункціональної платформи є комплексним та складається з кількох послідовних етапів, кожен з яких відіграє критичну роль у досягненні кінцевого результату. Все починається з фундаментального етапу візуального оформлення, тобто з детальної розробки дизайну користувацького інтерфейсу та досвіду [7]. На цій початковій стадії формується не просто зовнішній вигляд, а загальна концепція та унікальний візуальний стиль майбутнього сайту, що включає вибір кольорової палітри, шрифтів та графічних елементів. Після цього ретельно проєктується та промальовується вигляд кожної окремої сторінки, щоб забезпечити логічну, інтуїтивно зрозумілу та гармонійну взаємодію користувача з платформою. Коли візуальна частина готова та затверджена, закладається технічний фундамент проєкту. На етапі архітектурного проєктування продумується структура як клієнтської частини, з якою безпосередньо контактує користувач у своєму браузері, так і серверної частини, яка виступає ядром системи та відповідає за всю внутрішню логіку, обробку даних та взаємодію з базою даних. Далі настає черга безпосередньої програмної реалізації ключового функціоналу, що по суті є серцем усього застосунку. На цьому етапі створюється надійна віддалена база даних, де буде безпечно зберігатися вся необхідна інформація про товари, користувачів та їхні замовлення. Потім відбувається процес інтеграції, коли серверна та клієнтська частини зв'язуються між собою для обміну даними, і програмується вся запланована бекенд-логіка. Завершальним і вкрай важливим кроком усього процесу розробки є всебічне тестування готового додатку. Цей етап забезпечення якості включає як функціональне тестування для ретельної перевірки працездатності всіх механізмів, кнопок та форм, так і тестування

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

графічного інтерфейсу, щоб переконатися, що все виглядає та працює коректно на різних пристроях та у різних браузерах.

Кінцевим результатом виконання всього цього плану дій є створення унікального та повноцінного вебсайту. Він є не просто торговельним майданчиком, а цифровим простором, на якому користувачі зможуть не лише придбати необхідний музичний інструмент. Вони також зможуть знайти для себе натхнення, отримати корисну інформацію та обрати той творчий напрямок, у якому вони зможуть надалі вдосконалювати свої музичні навички та розвивати таланти.

1.2. Аналіз аналогів програмного продукту

Створення успішного та конкурентоспроможного інтернет-магазину музичних інструментів є неможливим без попереднього глибокого дослідження ринку. Український сегмент електронної комерції у цій сфері вже є досить насиченим, адже існує значна кількість сайтів схожої тематики. Саме тому ретельний аналіз діяльності конкурентів стає фундаментальним та обов'язковим етапом, що передує розробці. Таке дослідження дозволяє об'єктивно оцінити сильні та слабкі сторони наявних платформ, виявити успішні ринкові практики, які варто перейняти, та зрозуміти поширені помилки, яких слід уникнути. Це в свою чергу дає можливість сформуванню унікальної торговельної пропозиції та розробити продукт, що буде вигідно вирізнятися на тлі існуючих рішень.

Першим об'єктом для аналізу є сайт Muzline (рис. 1.1). Компанія Muzline.UA спеціалізується на продажу та обслуговуванні музичних інструментів, а також звукового та світлового обладнання. Асортимент компанії включає товари для виконавців різного рівня та професійне обладнання для студій і сцени. Бізнес-модель компанії поєднує роботу інтернет-магазину та фізичного магазину. Це надає клієнтам можливість ознайомитися з товарами на сайті, після чого відвідати фізичний магазин для тестування інструментів та отримання консультації від фахівців.[8]

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

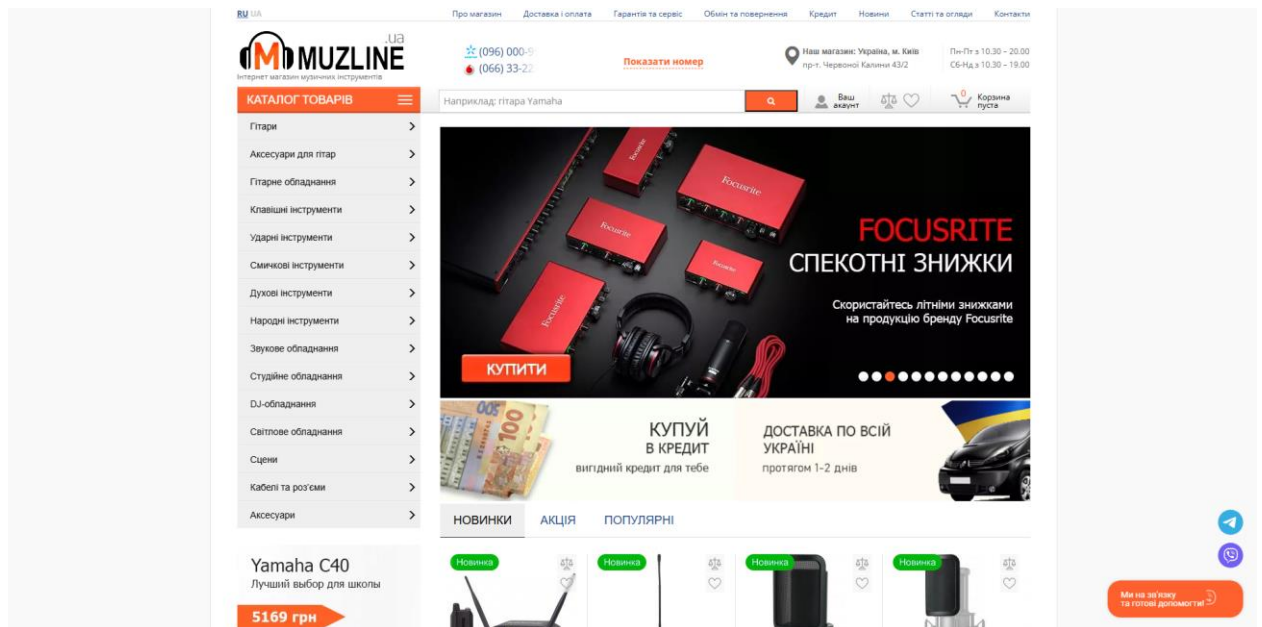


Рис. 1.1. MUZLINE

Його перевагою є наявність професійного блогу з оглядами, статтями та новинами музичної індустрії, що приваблює не лише покупців, а й широке коло зацікавлених музикантів, підвищуючи довіру до бренду. Сайт має приємний, добре структурований дизайн, а картки товарів часто містять якісні фотографії та вичерпні описи.

Це створює образ експертної та надійної платформи. Водночас, асортимент товарів у деяких категоріях може поступатися таким гігантам, як JAM, а система фільтрації, хоч і зручна, не завжди пропонує такий самий рівень деталізації для відбору специфічних моделей. Окрім того, програма лояльності та акційні пропозиції менш агресивні, що може знижувати мотивацію до імпульсивних покупок.

JAM (рис. 1.2), яка представляє себе як великий музичний комплекс на ринку України. Згідно з наданою інформацією, їхній фізичний простір займає три поверхи та вміщує тисячі різноманітних музичних інструментів. Ключовий акцент у роботі компанії зроблено на наданні клієнтам можливості безпосередньої взаємодії з продукцією. Кожен інструмент, представлений у комплексі, доступний для того, щоб його можна було оглянути, протестувати його звучання та фізично оцінити перед покупкою.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк. 12
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 1.2. JAM

Головною перевагою виступає надзвичайно глибокий та різноманітний асортимент, який охоплює все від гітар для початківців до професійного студійного та світлового обладнання. Платформа надає користувачам потужну і розгалужену систему фільтрації за безліччю технічних параметрів, що значно полегшує пошук потрібної моделі. Додаткову цінність створює інформативний контент, оскільки багато товарів супроводжуються відеооглядами, а програма лояльності заохочує до повторних покупок. Проте, попри ці переваги, сайт має і суттєві недоліки.

Його дизайн, хоч і функціональний, виглядає візуально застарілим на фоні сучасних тенденцій в електронній комерції. Інтерфейс, особливо на головній сторінці, часто здається перевантаженим великою кількістю банерів та акційних пропозицій, що може ускладнювати навігацію для нових відвідувачів. Також мобільна версія сайту не завжди забезпечує комфортний досвід взаємодії.

Музикант (рис. 1.3). Згідно з наведеним джерелом, дана платформа використовує різноманітні маркетингові інструменти для залучення клієнтів, до яких належать спеціальні комплекти продукції, святкові акції та сезонні знижки. Для постійних клієнтів у магазині діє система лояльності. Асортимент сайту орієнтований на широку аудиторію та включає як інструменти для початківців, так і обладнання для професійного використання. Крім того, для взаємодії з

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						13

клієнтами та надання консультацій на сайті реалізовано функцію онлайн-чату з консультантом.[4]

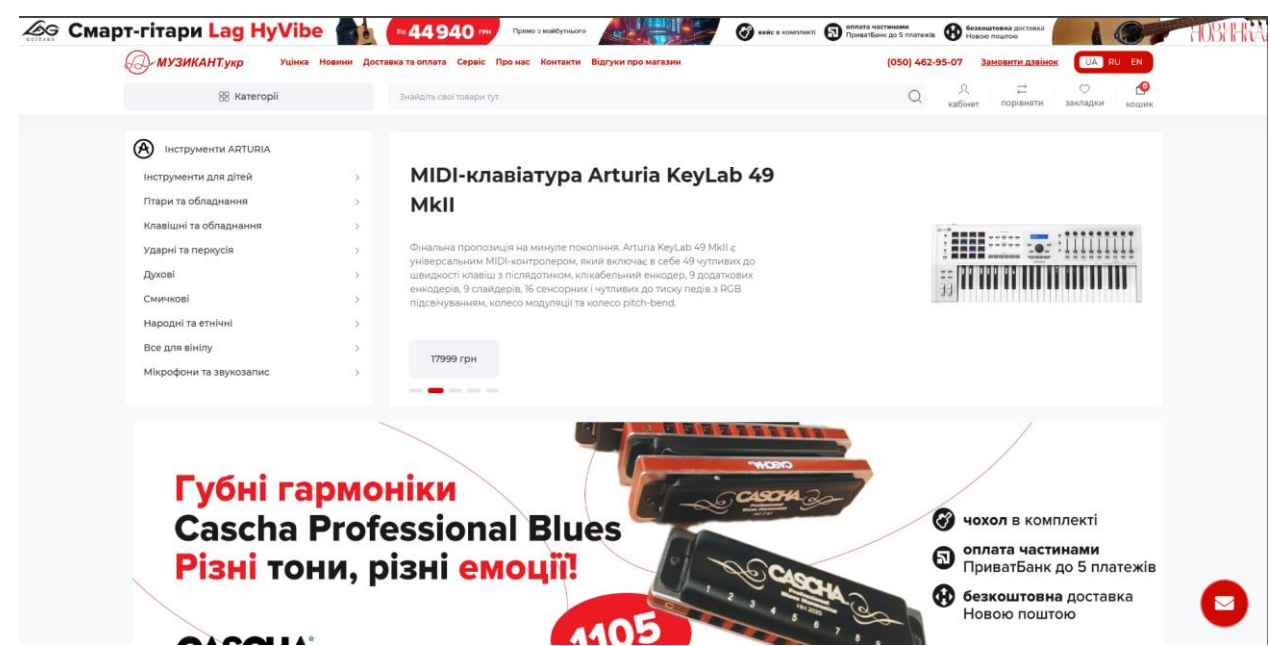


Рис. 1.3. МУЗИКАНТ

Його беззаперечною перевагою є чистий, мінімалістичний та візуально привабливий дизайн, що створює приємний користувацький досвід. Компанія робить особливий акцент на якісному клієнтському сервісі, пропонуючи доступні онлайн-консультації через чат, та представляє товари за допомогою якісних фотографій.

Однак існують і певні недоліки. Система фільтрації на сайті є менш деталізованою, ніж у старших конкурентів, що може ускладнювати вибір специфічних товарів. Асортимент, хоч і постійно зростає, у деяких нішевих категоріях представлений не так широко. Крім того, відсутність прозорої бонусної програми для постійних клієнтів може зменшувати їхню мотивацію до повторних покупок саме на цій платформі.

PRO Audio (рис. 1.4), яка спеціалізується на звуковому та світловому обладнанні. Згідно з їхніми даними, вони є першим шоу-румом музичних інструментів у Рівненській області. Основна діяльність компанії зосереджена навколо цієї фізичної локації, на базі якої регулярно проводяться різноманітні заходи. До таких заходів належать показові майстер-класи, демонстрації нових продуктів від виробників, а також творчі музичні вечори.[9]

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

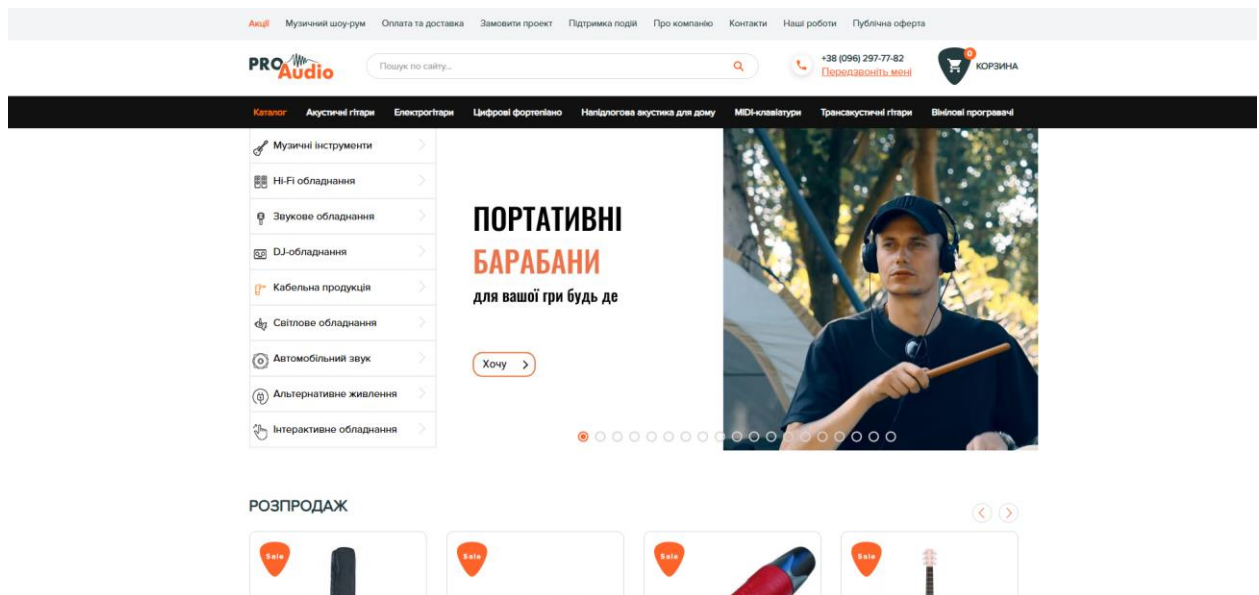


Рис. 1.4. ProAudio

Головним плюсом магазину є його експертність він пропонує глибокий асортимент у своїй категорії, а картки товарів містять вичерпні технічні характеристики, орієнтовані на фахівців. Консультації тут надають професіонали, що є великою цінністю для цільової аудиторії. Разом з тим, ця вузька спеціалізація є і головним недоліком для широкого ринку. Інтерфейс сайту може здаватися складним та недружнім для початківців чи музикантів-аматорів.

В асортименті практично відсутні класичні музичні інструменти, такі як гітари, піаніно чи духові, що робить магазин непридатним для більшості музикантів. Це інструмент для професіоналів, а не універсальна платформа.

Наведена спеціальна таблиця для порівняння магазинів (табл. 1.1)

Таблиця 1.1

Порівняння усіх магазинів

Функціонал	Muzline	Jam	Музикант.укр	PRO Audio	NoteUA
Детальна фільтрація	+	+	+/-	++	+/-
Список бажаного	+	+	+	+	-
Порівняння товарів	+	+	-	-	-
Відгуки клієнтів	+	+	+	-	+
Відео/аудіо огляди	+/-	+	+/-	+/-	+
Блог / Статті	+	+	-	+	+

Онлайн-чат	+	+	+	+	-
Бонусна система	+	+	-	+/-	-
Наявність в магазинах	+	+	+	-	+
Налаштування інструменту	-	-	-	-	+

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Вибір технологічного стеку є фундаментальним етапом у життєвому циклі розробки будь-якого програмного продукту. Для проєктування інтернет-магазину музичних інструментів було ухвалено рішення спиратися на сучасні, перевірені часом та індустрією технології [10]. Головними критеріями при виборі стали гнучкість, висока продуктивність, масштабованість та наявність великої спільноти розробників, що гарантує стабільну підтримку та розвиток інструментарію.

Фундаментом для серверної частини, відомої як бекенд, було обрано Node.js [11], що є сучасним та потужним асинхронним середовищем виконання коду на мові JavaScript [12]. В його основі лежить високопродуктивний рушій V8, розроблений компанією Google, що забезпечує надзвичайно швидку обробку операцій. Таке архітектурне рішення обумовлене кількома ключовими перевагами, що є критично важливими для розробки ефективного інтернет-магазину. По-перше, неблокуюча та керована подіями архітектура Node.js дозволяє серверу обробляти величезну кількість одночасних з'єднань від користувачів з мінімальними накладними витратами на системні ресурси [13]. У контексті інтернет-магазину, де в будь-який момент часу можуть перебувати сотні користувачів, які переглядають товари, додають їх у кошик та роблять замовлення, така висока продуктивність та здатність ефективно працювати з операціями вводу-виводу є вирішальною. По-друге, використання JavaScript як на клієнтській, так і на серверній стороні створює єдину та гармонійну екосистему для розробки [14]. Це уніфікує весь процес, дозволяючи команді

працювати з однією мовою програмування, що не тільки значно спрощує та прискорює створення нових функцій, але й зменшує ймовірність виникнення помилок [15]. Для додаткового обґрунтування цього вибору було проведено порівняльний аналіз (табл. 1.2) Node.js з іншими популярними серверними платформами для розробки.

Таблиця 1.2

Порівняння інструментів реалізації програмного продукту

Параметр	Node.js	PHP	Python (Django/Flask)
Продуктивність	Висока	Середня	Середня
Масштабованість	Дуже висока	Висока	Висока
Екосистема	Дуже велика (NPM)	Дуже велика (Composer)	Велика (Pip)
Основне застосування	API,	Вебсайти, CMS	Аналіз даних, веб, AI

Для структурування серверної логіки на базі Node.js було використано фреймворк Express.js [16]. Він зарекомендував себе як мінімалістичний, гнучкий та швидкий інструмент для створення вебдодатків та API [17]. Завдяки своїй простоті, Express.js не нав'язує розробнику суворої архітектури, надаючи свободу в організації коду. Його потужна система маршрутизації дозволяє легко створювати зрозумілі та логічні кінцеві точки API, а механізм проміжного програмного забезпечення (middleware) є ідеальним інструментом для реалізації таких завдань, як логування запитів, обробка помилок та, що найважливіше, автентифікація користувачів.

Що стосується зберігання даних, вибір було зроблено на користь MongoDB [18] провідної нереляційної, документно-орієнтованої системи управління базами даних. На відміну від традиційних реляційних баз даних, MongoDB зберігає дані у гнучких, JSON-подібних документах [19]. Такий підхід ідеально пасує для проєктів, де структура даних може змінюватися або бути різноманітною. Наприклад, в каталозі музичних інструментів різні товари (гітари, синтезатори, ударні установки) можуть мати унікальні набори характеристик, і гнучка схема MongoDB дозволяє легко це реалізувати. Крім

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						17

того, ця база даних відома своєю горизонтальною масштабованістю, що дає змогу системі легко рости разом зі збільшенням обсягу даних та навантаження [20]. У таблиці представлено порівняння обраної СУБД MongoDB з її основними реляційними аналогами (табл. 1.2).

Таблиця 1.2

Порівняння різних баз даних

Параметр	MongoDB (NoSQL)	PostgreSQL (SQL)	MySQL (SQL)
Модель даних	Документно-орієнтована	Об'єктно-реляційна	Реляційна
Гнучкість схеми	Дуже висока	Низька	Низька
Підтримка транзакцій	+/-	+ (Повна підтримка ACID)	+ (Повна підтримка ACID)
Масштабованість	Горизонтальна	Вертикальна	Вертикальна
Типовий сценарій	Динамічні проєкти, Big Data	Складні системи, аналітика	Вебсайти, класичні додатки

Для візуалізації даних та створення динамічного користувацького інтерфейсу було застосовано шаблонізатор Handlebars [21]. Цей інструмент дозволяє створювати HTML-шаблони та вбудовувати в них дані, що надходять із сервера, безпосередньо перед відправкою сторінки користувачу. Вибір Handlebars обумовлений його надзвичайною простотою, високою швидкістю роботи та безшовною інтеграцією з фреймворком Express.js. Такий підхід дозволяє швидко й ефективно генерувати динамічний контент, наприклад, оновлювати список товарів у каталозі, відображати вміст кошика чи персоналізовані дані в профілі користувача, мінімізуючи при цьому навантаження на клієнтську частину.

Важливою складовою проєкту є його API (Application Programming Interface) програмний інтерфейс, що виступає посередником для обміну даними між клієнтською та серверною частинами. Саме через API веббраузер користувача надсилає запити на отримання товарів чи оформлення замовлення, а сервер надає відповідь.

Для того, щоб забезпечити безпеку цього обміну даними та захистити облікові записи користувачів, у проєкті було реалізовано механізм

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						18

автентифікації на основі JWT (JSON Web Token) [22]. Це сучасний та надійний стандарт для створення токенів доступу. Коли користувач успішно входить у систему, сервер генерує унікальний зашифрований JWT-токен, що містить інформацію про користувача, і відправляє його на клієнтську сторону. Надалі, при кожному запиті до захищених ресурсів API, клієнт автоматично додає цей токен до заголовка. Сервер, у свою чергу, перевіряє валідність токена перед тим, як надати доступ до даних, що унеможливорює несанкціонований доступ до чужих замовлень чи персональної інформації.

Висновки до першого розділу.

У процесі цього дослідження особливу увагу було приділено вивченню аналогічних і схожих за концепцією продуктів. Були детально проаналізовані та порівняні наявні на ринку інтернет-магазини, що дало змогу виявити їхні сильні та слабкі сторони, запозичити успішні функціональні рішення та уникнути поширених помилок в архітектурі та дизайні. На основі отриманих даних був сформований фінальний перелік функціональних та нефункціональних вимог для сайту, що охоплює все від структури каталогу та системи фільтрації до вимог щодо безпеки та продуктивності. Саме цей чітко визначений набір критеріїв і став фундаментом для подальшого вибору основних технологій та засобів реалізації, що дозволило обґрунтовано сформулювати технологічний стек, який найкраще відповідає поставленим задачам розробки.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

РОЗДІЛ 2. ПРОЕКТУВАННЯ САЙТУ МУЗИЧНИХ ІНСТРУМЕНТІВ

2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи

Для проєктування архітектури та функціоналу інтернет-магазину було визначено два ключові типи користувачів, кожен з яких має власні повноваження та сценарії взаємодії з системою. На діаграмі представлені дві головні ролі користувач та адміністратор, а також відповідні варіанти використання, доступні для кожного з них (рис . 2.1).

Користувач є основним актором системи, який переважно виступає у ролі покупця. Для нього реалізовано повний цикл взаємодії з інтернет-магазином, що починається з перегляду товарів та пошуку за ключовими словами і закінчується оформленням замовлення. Нерідко користувач починає знайомство з платформою як незареєстрований відвідувач. У такому випадку йому доступні базові можливості, зокрема перегляд товарів у каталозі та пошук. Для доступу до розширених функцій необхідно пройти реєстрацію або авторизацію.

Після входу в систему користувач отримує доступ до персоналізованих опцій. Він може створювати і редагувати власний профіль, вказувати контактну інформацію, зберігати адреси доставки. Основний функціонал, пов'язаний із здійсненням покупки, передбачає можливість додавання товарів до кошика та подальше оформлення замовлення з підтвердженням обраних позицій та введенням даних для доставки. Важливою функцією для користувача є можливість налаштування власного інструменту. Такий модуль дозволяє за допомогою конфігуратора обрати індивідуальні параметри, як-от тип звукознімачів, вид деревини чи колір для гітари. Це додає персоналізації до процесу купівлі та робить досвід клієнта більш унікальним і захопливим.

Адміністратор системи є користувачем з розширеними правами доступу, який відповідає за повне управління контентом сайту та підтримку його операційної діяльності. Його функції можна умовно розділити на два ключові напрямки, а саме управління каталогом та обробка замовлень.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

Перший напрямок, управління каталогом, включає в себе роботу з категоріями та товарами. Адміністратор має змогу створювати, редагувати та видаляти категорії і підкатегорії, формуючи таким чином зрозумілу і зручну логіку для навігації по сайту, що значно покращує взаємодію користувача з платформою. У межах керування товарами адміністратор здійснює додавання нових позицій до відповідних категорій, заповнюючи усі необхідні характеристики, включаючи назву, опис, вартість і зображення. Окрім того, він має змогу редагувати кількість товарів, змінювати їхні статуси, наприклад "в наявності" чи "немає в наявності", а також видаляти позиції, які вже не є актуальними. Другим важливим напрямком є управління замовленнями. Цей функціонал дозволяє адміністратору переглядати всі покупки, зроблені клієнтами, відстежувати їхні статуси та вчасно обробляти для відправки. Таким чином, адміністратор забезпечує не лише наповнення сайту контентом, але й виконання ключових бізнес-процесів інтернет-магазину..

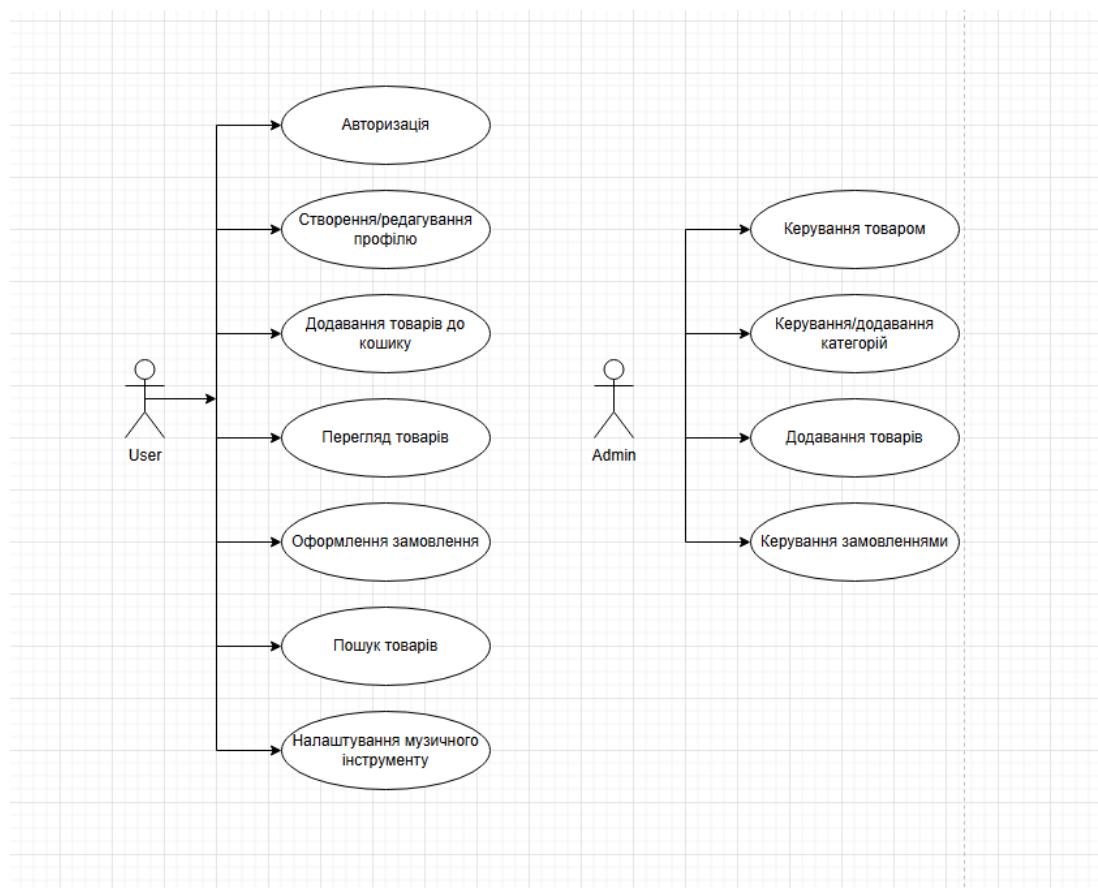


Рис. 2.1. Діаграма використання

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.7.

Таблиця 2.1

Опис прецеденту “Додавання/редагування/видалення інформації про категорію”

Назва варіанту використання	Додавання/редагування/видалення інформації про категорію/підкатегорію
Основні актори	Адмін
Короткий опис	Додавання, редагування та видалення інформації про категорію, а саме, фото, опис, назва

Таблиця 2.2

Опис прецеденту “Додавання/видалення товару”

Назва варіанту використання	Додавання/видалення товару
Основні актори	Адмін
Короткий опис	Додавання та видалення та зміну кількості товарів сайту через меню адміністрації

Таблиця 2.3

Опис прецеденту “Перегляд сторінок”

Назва варіанту використання	Перегляд сторінок
Основні актори	Звичайний користувач
Короткий опис	Можливість переглядати вміст всіх сторінок клієнтської частини сайту

Таблиця 2.4

Опис прецеденту “Вибір товару”

Назва варіанту використання	Вибір товару
Основні актори	Звичайний користувач
Короткий опис	Можливість обрати будь який товар з сайту

Таблиця 2.5

Опис прецеденту “Оплата занятия через LiqPay”

Назва варіанту використання	Оплата занятия через LiqPay
Основні актори	Звичайний користувач
Короткий опис	Можливість оплатити заняття через сервіс LiqPay

Таблиця 2.6

Опис прецеденту “Реєстрація/Вхід”

Назва варіанту використання	Реєстрація/Вхід
Основні актори	Звичайний користувач, Адмін
Короткий опис	Можливість авторизуватись на сайті як користувачу так і адміну

Таблиця 2.7

Опис прецеденту “Налаштування інструменту”

Назва варіанту використання	Налаштування інструменту
Основні актори	Звичайний користувач
Короткий опис	Можливість за допомогою онлайн тюнера налаштувати струни свого інструменту

Основною метою створення сайту інтернет-магазину музичних інструментів є формування багатофункціональної онлайн-платформи, орієнтованої на клієнта, що слугитиме комфортним простором для задоволення потреб широкого кола музикантів. Проєкт розроблений з метою надання можливості кожному охочому, незалежно від рівня його підготовки, швидко та зручно знаходити і купувати необхідні товари для музичної творчості. Це передбачає, що інтерфейс і структура сайту мають бути однаково зрозумілими як для початківців, що тільки починають свій шлях у музиці, так і для досвідчених професіоналів, котрі шукають спеціалізоване обладнання.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

Для досягнення цієї мети пріоритет надається гарантуванню доступу до розширеного та добре організованого асортименту продукції. Каталог має включати різноманітні категорії від класичних інструментів, таких як гітари та клавішні, до сучасного студійного обладнання та аксесуарів. Не менш важливою є презентація вичерпної інформації про кожен товар. Вона охоплює не тільки докладні технічні параметри та описи, а й якісні фотографії з різних ракурсів, а в ідеалі відеоогляди та аудіоприклади звучання, що дає змогу клієнтові зробити максимально обґрунтований вибір.

Завершальним етапом взаємодії з платформою є простий і прозорий процес оформлення замовлення. Система кошика, реєстрації та оплати спроектована таким чином, щоб мінімізувати кількість етапів та не викликати складнощів у користувача. Треба забезпечити інтуїтивно зрозумілий процес від додавання товару в кошик до успішного завершення покупки, що разом формує позитивний клієнтський досвід і спонукає користувача повертатися до магазину.

Для успішної реалізації проєкту було сформульовано детальний набір вимог, що охоплюють як потреби користувачів, так і технічні аспекти системи. Було визначено дві основні ролі внутрішніх користувачів. Для кінцевого клієнта передбачено інтуїтивно зрозумілий шлях він може вільно переглядати сторінки сайту та каталоги товарів, додавати вподобані позиції до кошика та, врешті-решт, безпечно оплачувати своє замовлення за допомогою інтегрованого платіжного сервісу LiqPay. З іншого боку, адміністратор отримує повний набір інструментів для управління контентом він може редагувати інформацію на статичних сторінках, додавати та змінювати контактні дані, а найголовніше керувати каталогом товарів, додаючи нові продукти та переглядаючи список замовлень.

Для реалізації цих можливостей було визначено ключові функціональні вимоги. Система повинна забезпечувати надійну реєстрацію та вхід користувачів, при цьому безпека паролів гарантується за допомогою їх хешування. Для кожного авторизованого користувача створюється унікальний JWT-токен, що забезпечує захищений доступ до особистих даних, а можливість оплати реалізована через сервіс LiqPay.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрім функціоналу, велику увагу було приділено нефункціональним вимогам. Інтерфейс спроектовано таким чином, щоб бути максимально інтуїтивним, не вимагаючи від користувача додаткової підготовки, а час на його освоєння не перевищує кількох хвилин. Швидкодія системи також є пріоритетом час відгуку на дії користувача не повинен перевищувати трьох секунд, а на завантаження платіжної форми п'ятнадцяти. Що стосується надійності, вебсайт має бути доступний для користувачів цілодобово, а його продуктивність повинна забезпечувати одночасну роботу великої кількості відвідувачів без втрати швидкості.

Нарешті, було визначено системні вимоги для коректної роботи на стороні клієнта достатньо пристрою з 2 Гб оперативної пам'яті, процесором від 1.4 GHz та доступом до мережі інтернет. Серверна ж частина вимагає підтримки технології NodeJS для свого функціонування.

2.2. Розробка системи збереження даних

Вибір системи управління базами даних є ключовим архітектурним рішенням, що безпосередньо впливає на гнучкість, продуктивність та масштабованість усього проекту. У рамках розробки даного інтернет-магазину для зберігання всіх даних було обрано MongoDB. Це сучасна та провідна нереляційна документо-орієнтована система, яка дозволяє зберігати інформацію не в звичних таблицях зі строгими рядками та стовпцями, як у класичних базах даних на кшталт MySQL, а у вигляді гнучких документів формату BSON, що дуже схожі на звичайні записи в форматі JSON. Такий підхід є надзвичайно зручним для сучасних вебзастосунків, оскільки він усуває необхідність заздалегідь створювати складну та жорстку структуру даних і дозволяє легко її змінювати та розширювати в майбутньому.

У спроектованій базі даних зберігається вся необхідна для функціонування магазину інформація. Вона включає дані про користувачів та адміністраторів, їхні замовлення, повний перелік товарів, тимчасові кошики покупок, а також відгуки, залишені клієнтами. Вся ця інформація логічно організована у спеціальних колекціях. Колекція в MongoDB є аналогом таблиці в реляційних

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

базах і слугує для групування однотипних документів. Наприклад, уся інформація про товари зберігається в колекції products, дані про зареєстрованих користувачів та адміністраторів знаходяться в колекції users, а всі замовлення потрапляють до колекції orders, що забезпечує зрозумілу організацію даних.

Розглянемо таблиці на яких зображена дані (таблиці 2.8- 2.13).

Таблиця 2.8

Опис таблиці users

Поле	Тип	Призначення
id	objectId	Ідентифікатор
name	string	Ім'я
email	string	Електронна пошта
Password	string	Пароль
role	string	Роль

Таблиця або колекція users (табл. 2.8) є одним із центральних елементів системи збереження даних, оскільки вона відповідає за управління ідентифікацією, автентифікацією та авторизацією всіх користувачів платформи. Кожен запис у цій таблиці представляє одного унікального користувача та містить його персональний ідентифікатор, ім'я, що використовується для персоналізації взаємодії з сайтом, та його електронну пошту, яка слугує як унікальний логін для входу в систему. Особливу увагу приділено безпеці, тому поле для пароля зберігає не сам пароль у відкритому вигляді, а його криптографічний хеш, отриманий за допомогою надійного алгоритму SHA-256. [24] Це є ключовим заходом безпеки, що гарантує неможливість відновлення оригінального пароля навіть у випадку несанкціонованого доступу до бази даних. Крім того, для розмежування прав доступу в системі використовується поле role, яке визначає рівень повноважень кожного користувача, дозволяючи розрізняти звичайних клієнтів та адміністраторів.

Таблиця 2.9

Опис таблиці products

Поле	Тип	Призначення
id	objectId	Ідентифікатор
name	string	Назва товару
description	string	Опис

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Продовження таблиці 2.9

image	string	Url посилання на картинку товару
price	int32	Ціна
additionalImage	Array(string)	Додаткове фото
categoryId	string	Категорія
inStock	boolean	В наявності чи ні
quantity	string	Кількість
specifications	string	Характеристики
createdAt	date	Коли створено
updatedAt	date	Коли змінено

Ключовою сутністю в базі даних будь-якого інтернет-магазину є таблиця або колекція, що відповідає за товари (табл. 2.9). У даному проєкті цю роль виконує колекція `products`, яка зберігає всю вичерпну інформацію про кожну окрему позицію, доступну для продажу. Кожен запис у цій таблиці має унікальний ідентифікатор `id`, що дозволяє однозначно ідентифікувати товар у системі. Таблиця містить основні комерційні дані, такі як назва товару, його детальний опис та актуальна ціна. Для візуальної презентації передбачено поле для основного зображення товару, яке є посиланням на відповідний файл, а також окреме поле для зберігання масиву додаткових фотографій, що дозволяє створити повноцінну галерею для кращого ознайомлення покупця з продуктом.

Кожен товар обов'язково пов'язаний з певною категорією через поле `categoryId`, що забезпечує логічну структуру каталогу. Для управління складом використовуються два важливих поля, а саме `inStock`, яке вказує на наявність товару, та `quantity`, що зберігає точну кількість одиниць на складі. Для надання покупцям повної технічної інформації існує поле `specifications`, де зберігаються всі ключові характеристики продукту. Нарешті, для відстеження життєвого циклу запису в таблиці передбачені поля `createdAt` та `updatedAt`, які автоматично фіксують час створення та останнього оновлення інформації про товар.

Таблиця 2.10

Опис таблиці `cards`

Поле	Тип	Призначення
<code>id</code>	<code>objectId</code>	Ідентифікатор
<code>user</code>	<code>objectId</code>	Юзер

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовження таблиці 2.10

items	array	Список товарів
createdAt	date	Коли створено
updatedAt	date	Коли змінено

Таблиця, що в системі має назву cards, відіграє одну з ключових ролей у процесі взаємодії клієнта з магазином, оскільки вона реалізує функціонал віртуального кошика для покупок (табл. 2.10). Кожен документ у цій колекції є унікальним кошиком, що нерозривно пов'язаний з конкретним користувачем через поле user. Це дозволяє зберігати вміст кошика навіть після того, як користувач залишає сайт, забезпечуючи так звану персистентність даних і можливість продовжити покупки з будь-якого пристрою. Основний вміст кошика зберігається у полі items, що є масивом товарів, обраних покупцем. Кожен елемент цього масиву зазвичай містить посилання на конкретний продукт та інформацію про замовлену кількість, що дозволяє гнучко керувати складом майбутнього замовлення. Крім того, для аналітики та управління життєвим циклом кошика, таблиця фіксує точні дати його створення та останнього оновлення через поля createdAt та updatedAt. Ця інформація може використовуватися, наприклад, для нагадувань про покинуті кошики або для автоматичного очищення застарілих даних.

Таблиця 2.11

Опис таблиці categories

Поле	Тип	Призначення
id	objectId	Ідентифікатор
name	string	Ім'я
slug	string	Переводить текст у нижній регістр
description	string	Опис
image	string	Зображення
isActive	boolean	Чи активна
createdAt	date	Коли змінено
updatedAt	date	Коли змінено
parentId	Null(objectId)	Чи є батьківська категорія

Таблиця categories відіграє фундаментальну роль в організації всього асортименту інтернет-магазину, оскільки вона відповідає за створення логічної

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

та інтуїтивно зрозумілої структури каталогу (табл. 2.11). Кожен запис у цій колекції представляє окрему категорію товарів та містить її назву, детальний опис та головне зображення, які використовуються для візуального представлення розділу на сайті. Важливим полем є slug, що автоматично генерує унікальне посилання з назви категорії для використання в URL-адресах, що є критично важливим для пошукової оптимізації та створення зрозумілих для людини посилань. Для адміністративного контролю передбачено поле isActive, яке дозволяє тимчасово приховати або показати категорію на сайті, не видаляючи її. Ключовою особливістю цієї таблиці є поле parentId, яке дає можливість створювати необмежену ієрархію та вкладеність, пов'язуючи одну категорію з іншою як з батьківською. Саме цей механізм дозволяє будувати складну багаторівневу структуру каталогу, наприклад, створюючи підкатегорії, що значно спрощує навігацію для користувачів.

Таблиця 2.12

Опис таблиці orders

Поле	Тип	Призначення
id	objectId	Ідентифікатор
user	objectId	Ідентифікатор
items	array	Товари
totalAmount	int32	Кількість товарів
status	string	Статус замовлення
paymentStatus	string	Статус оплати
paymentData	null	Дата оплати
shippingAddress	object(string)	Адреса доставки
notes	string	Коментар
createdAt	int32	Коли створений

Колекція orders (табл. 2.12) є ключовим компонентом бази даних з точки зору бізнес-логіки, оскільки вона зберігає всю вичерпну інформацію про завершені покупки, зроблені користувачами. Кожен документ у цій колекції представляє собою окреме замовлення з унікальним ідентифікатором. Він обов'язково містить посилання на конкретного користувача, що його оформив, а також на масив придбаних товарів, де зафіксовано їхню кількість та ціну на момент покупки. Для організації доставки в таблиці передбачено поле, що

зберігає повну адресу доставки, вказану клієнтом. Фінансова частина замовлення відображена у полях загальної суми покупки та статусу оплати, а також у полі для збереження деталей платіжної транзакції. Найважливішим інструментом для адміністратора є поле статусу виконання замовлення, яке дозволяє відстежувати його життєвий цикл від моменту створення до відправки та отримання клієнтом. Крім того, у таблиці є можливість зберігати будь-які коментарі, залишені покупцем під час оформлення замовлення.

Таблиця 2.13

Опис таблиці reviews

Поле	Тип	Призначення
id	objectId	Ідентифікатор
productId	objectId	Ідентифікатор
userId	objectId	Ідентифікатор
rating	int32	Оцінка
comment	string	Коментар
createdAt	date	Коли створено

Ця таблиця використовується для зберігання відгуків, які покупці залишають до товарів (табл. 2.13). Кожен відгук містить посилання як на конкретний продукт, так і на користувача, що його написав. Окрім текстового коментаря, в таблиці зберігається числова оцінка (рейтинг), виставлена товару, та дата створення самого відгуку.

Для наочного уявлення структури даних, які використовуються у системі, зображено узагальнену концептуальну модель бази даних інтернет-магазину (рис 2.2). Ця діаграма відображає основні сутності, що є фундаментом для всієї бізнес-логіки, а також логічні зв'язки між ними, які забезпечують цілісність та коректну роботу платформи. Центральними сутностями моделі є Користувач (User), де зберігається інформація про зареєстрованих клієнтів та їхні ролі, та Товар (Product), що містить повний набір даних про кожну позицію в каталозі, включаючи назву, опис, ціну та кількість на складі. Для логічної організації асортименту використовується сутність Категорія (Category), з якою безпосередньо пов'язаний кожен товар. Процес покупки представлений двома ключовими сутностями. Перша це Кошик (Card), що є тимчасовим сховищем,

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						30

яке пов'язує конкретного користувача з переліком обраних ним товарів. Друга це Замовлення (Order), що фіксує вже завершену транзакцію і містить інформацію про покупця, склад замовлення, його статус та адресу доставки. Крім того, для реалізації системи зворотного зв'язку передбачена сутність Відгук (Review), яка дозволяє користувачам залишати оцінки та коментарі до конкретних товарів. Усі ці сутності взаємопов'язані, створюючи єдину інформаційну структуру, що дозволяє ефективно керувати даними на всіх етапах взаємодії клієнта з магазином.

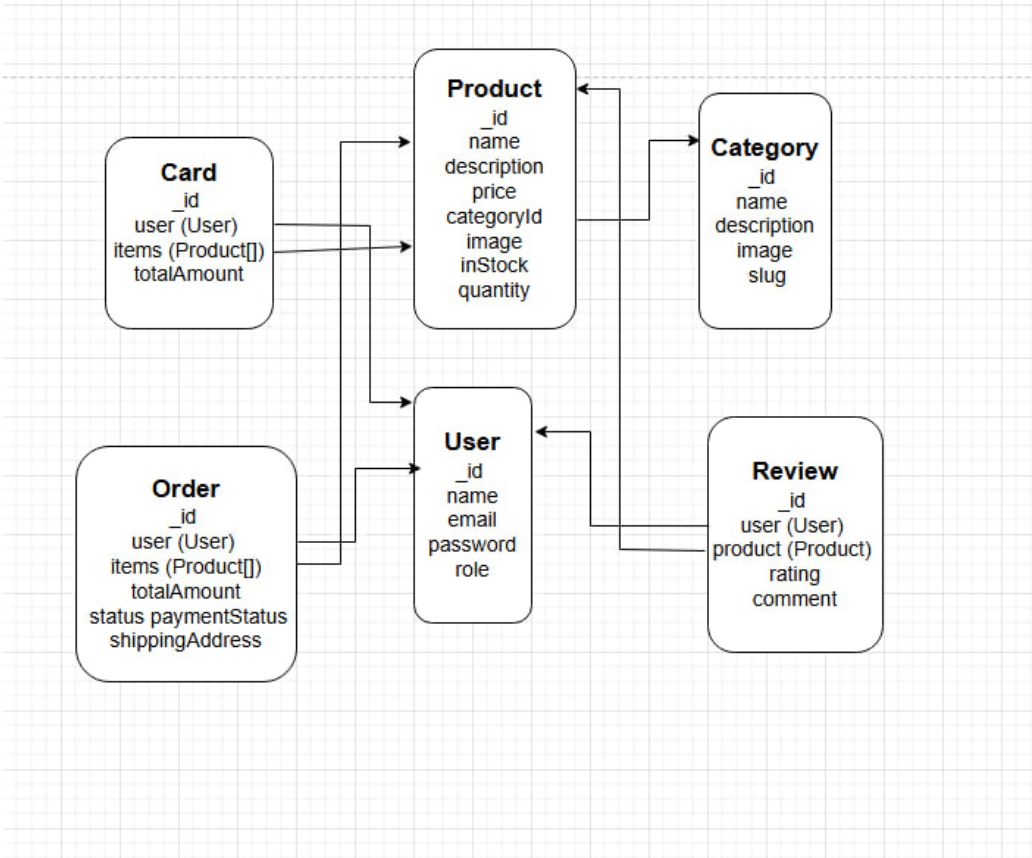


Рис. 2.2. Структура бази даних

2.3. Проектування та реалізація алгоритмів роботи системи

Весь алгоритм роботи для звичайного користувача в системі інтернет-магазину спроектовано як інтуїтивно зрозумілий та послідовний шлях, що веде клієнта від знайомства з асортиментом до успішної покупки. Для наочної демонстрації цієї логіки взаємодії було розроблено відповідну діаграму активності (рис. 2.3). Подорож користувача зазвичай починається з головної сторінки, яка виступає не просто відправною точкою, а цифровою вітриною та навігаційним центром усього сайту. Саме тут відвідувач може одразу

ознайомитися з актуальними пропозиціями, побачити популярні товари та отримати доступ до всіх ключових розділів, таких як сторінка з контактною інформацією, особистий профіль або кошик. Бажаючи ознайомитися з асортиментом, користувач зазвичай обирає одну з представлених категорій товарів. Після цього система плавно перенаправляє його на відповідну сторінку каталогу, де у зручному форматі відображаються всі товари, що належать до цієї категорії, з можливістю переглянути їхні короткі характеристики для швидкого порівняння. Якщо якийсь конкретний продукт привертає увагу покупця, він може перейти на його персональну сторінку для більш глибокого та детального вивчення. На цій сторінці йому надається вся вичерпна інформація, включаючи галерею якісних фотографій, повний опис, технічні параметри та відгуки інших клієнтів. Коли користувач остаточно приймає рішення про покупку, він натискає на кнопку додавання товару до кошика, що є першим кроком у процесі замовлення. Зі сторінки кошика, де можна ще раз перевірити та відредагувати свій вибір, він переходить безпосередньо до оформлення замовлення. Під час оформлення замовлення він має вказати свої данні, вибрати адресу доставки що дуже зручно бо вона реалізована за допомогою сервісу NovaPoshta [23]. Фінальним та найважливішим етапом є оплата, яку клієнт може зручно та абсолютно безпечно здійснити за допомогою інтегрованого та надійного платіжного сервісу LiqPay [26], після успішного завершення якої система фіксує коректний платіж після якого завершує цикл покупки та додає замовлення в його профіль якщо воно успішне.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		32

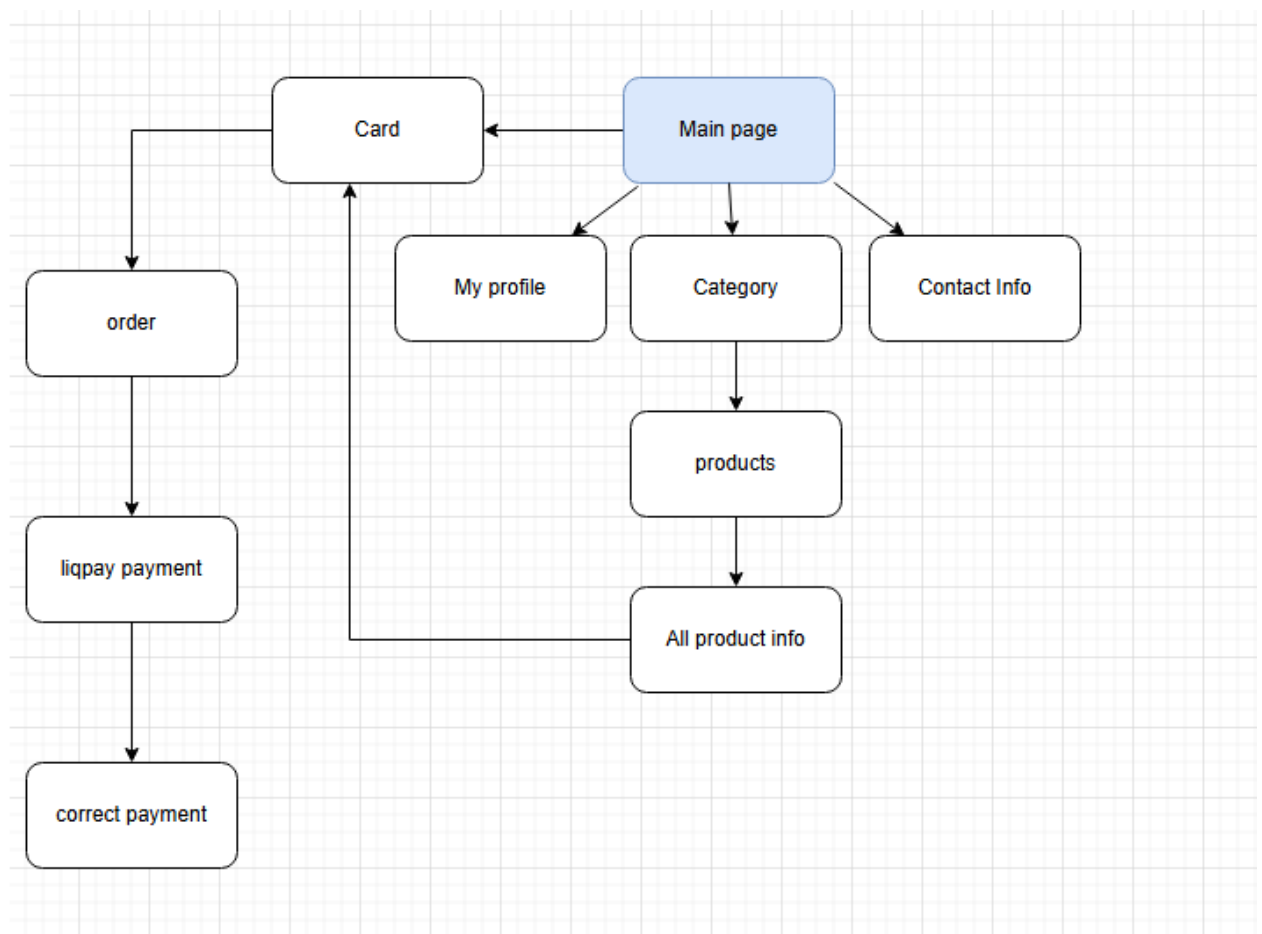


Рис. 2.3. Діаграма активності користувача

Адміністративна панель спроектована як централізований та інтуїтивно зрозумілий робочий простір, що надає уповноваженому користувачу набір ключових інструментів для повного контролю над контентом та операційною діяльністю інтернет-магазину. Вона є ядром системи управління, звідки доступні всі основні модулі, що дозволяють ефективно підтримувати актуальність інформації та обробляти замовлення. Основний функціонал зосереджений навколо управління життєвим циклом товарів, категорій та замовлень.

Адміністратор має можливість здійснювати повне управління товарами, що включає легке додавання нових продуктів до каталогу через спеціальну форму, де заповнюються всі необхідні дані, такі як назва, детальний опис, характеристика, ціна та зображення. Так само він може в будь-який момент редагувати інформацію про вже існуючі товари, оновлюючи їхні статуси, наприклад, наявність на складі, та змінюючи кількість. Окрім цього, доступна функція видалення товарів, які більше не є актуальними. Для структурування

каталогу передбачено модуль управління категоріями, де адміністратор може створювати, редагувати та видаляти категорії і підкатегорії. Не менш важливою є функція управління замовленнями, яка дозволяє переглядати всі нові покупки, відстежувати їхні статуси та вчасно обробляти їх для відправки клієнтам. Для наочної демонстрації взаємозв'язку цих функціональних блоків була створена відповідна діаграма активності (рис.2.4).

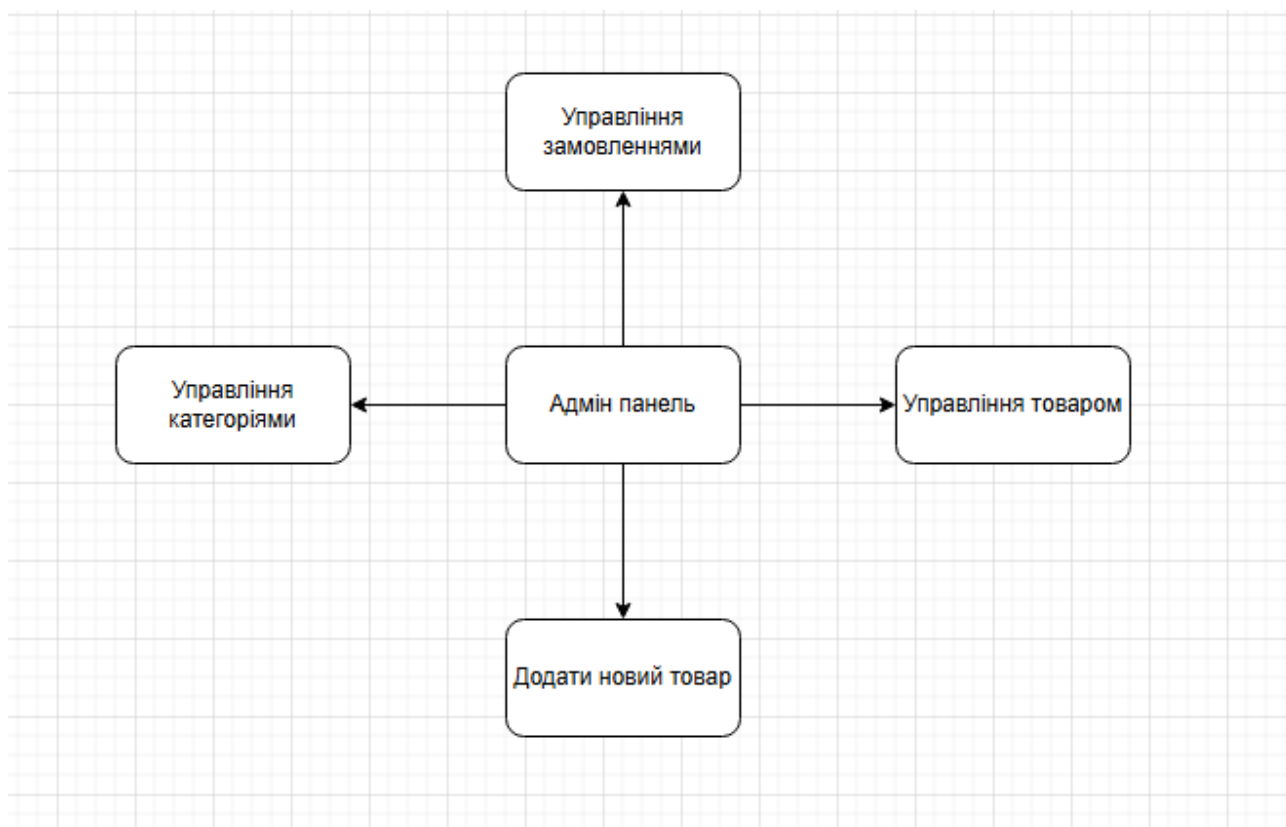


Рис. 2.4. Діаграма активності адміна

В основі архітектури розробленого інтернет-магазину лежить принцип чіткого розмежування прав доступу, що реалізовано через дві ключові ролі, а саме звичайного Користувача та Адміністратора. Такий підхід дозволяє створити з одного боку зручний та безпечний простір для клієнтів, а з іншого надати потужні інструменти для управління бізнесом.

Роль Користувача охоплює весь клієнтський шлях, починаючи від першого знайомства з асортиментом і закінчуючи успішною покупкою. Йому доступні всі необхідні функції для пошуку та вибору товарів, включаючи перегляд каталогу, використання фільтрів та ознайомлення з детальними характеристиками продуктів. Після простої процедури авторизації користувач отримує доступ до

персоналізованих можливостей, таких як управління особистим профілем, а також може переходити до процесу покупки, що включає роботу з кошиком та оформлення замовлення через інтегровані платіжні сервіси.

Роль Адміністратора, натомість, є центром управління всією платформою. Його повноваження умовно діляться на дві основні сфери. Перша це управління контентом, що включає повний контроль над каталогом, а саме створення та редагування категорій і безпосередньо самих товарів. Друга сфера це операційна діяльність, яка полягає в управлінні замовленнями, що дозволяє відстежувати та обробляти покупки клієнтів. Таким чином, забезпечується повний контроль як над вітриною магазину, так і над його внутрішніми бізнес-процесами.

2.4. Реалізація функціоналу

В основі серверної частини розробленого інтернет-магазину лежить перевірений та ефективний стек технологій, а саме середовище виконання Node.js та вебфреймворк Express.js. Такий вибір дозволив побудувати швидкий та масштабований бекенд, здатний завдяки своїй асинхронній природі ефективно обробляти велику кількість одночасних запитів від користувачів. Архітектура застосунку була спроектована за класичним та добре структурованим принципом MVC [27]. Діаграму роботи цього патерну можна побачити на (рис.2.5).

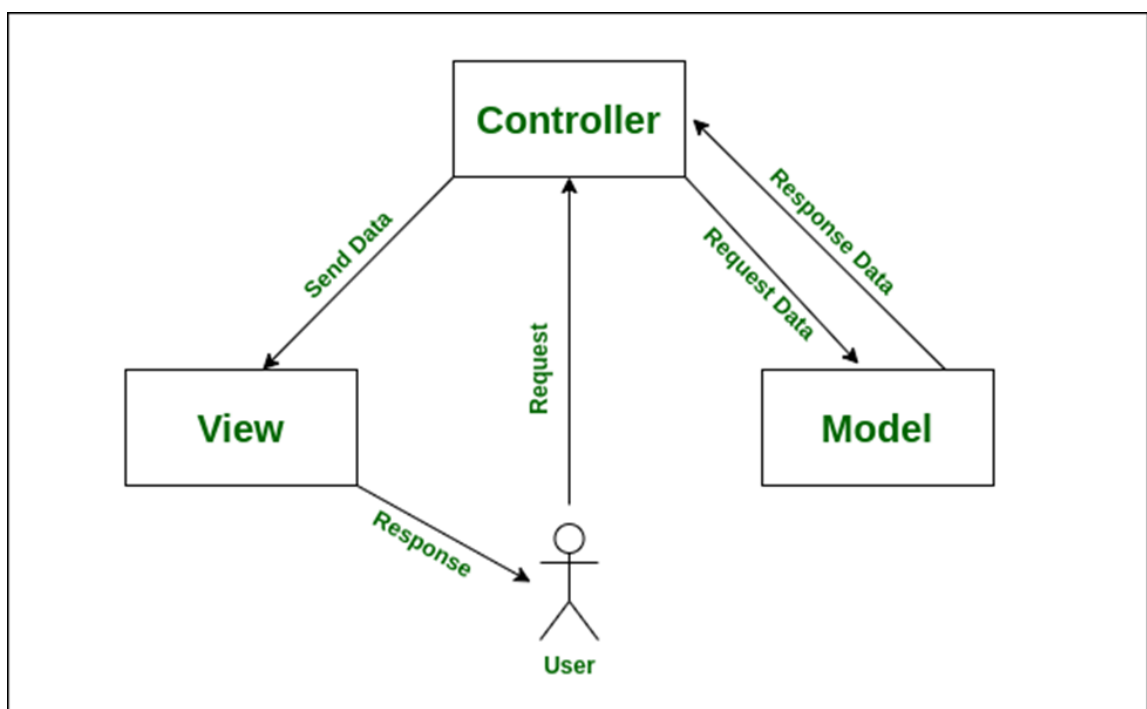


Рис. 2.5 Робота MVC патерну

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Концидайло А.М.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

Цей підхід забезпечив чіткий поділ відповідальності між компонентами системи, де моделі, реалізовані через схеми Mongoose [28], відповідають за структуру даних, контролери інкапсулюють у собі всю бізнес-логіку, обробляючи дії користувачів, а за візуальне представлення інформації відповідають динамічні відображення. Останні генеруються за допомогою серверного шаблонізатора Handlebars, який повністю підтримує українську мову, що гарантує коректне виведення всього текстового контенту.

Для зберігання всієї інформації було обрано хмарну документо-орієнтовану базу даних MongoDB Atlas. Це сучасне рішення гарантує не тільки надійне збереження даних завдяки механізмам резервного копіювання, але й високу швидкість доступу та легку горизонтальну масштабованість у майбутньому, що є критично важливим для комерційного проєкту. Взаємодія з базою даних у коді відбувається через бібліотеку Mongoose, яка слугує зручним абстракційним шаром між додатком та базою, надаючи інструменти для валідації даних та роботи зі строго типізованими схемами.

Особливу увагу в проєкті було приділено системі безпеки та автентифікації користувачів. Для їхньої ідентифікації використовуються сучасні JWT-токени, які після успішного входу в систему зберігаються в HTTP-only cookies [29]. Такий метод зберігання значно підвищує безпеку, оскільки унеможливорює доступ до токенів через JavaScript на стороні клієнта і захищає від XSS-атак. Для розмежування прав доступу між звичайними покупцями та адміністраторами було впроваджено спеціальний програмний компонент middleware. Він виступає в ролі сторожа, який перевіряє повноваження користувача перед наданням доступу до закритих адміністративних розділів. Паролі користувачів ніколи не зберігаються у відкритому вигляді, натомість вони проходять процедуру незворотного хешування за криптографічним алгоритмом SHA-256, що унеможливорює їх розшифровку.

Управління каталогом товарів було спроектовано з урахуванням сучасних вимог до юзабіліті та пошукової оптимізації. Для категорій автоматично генеруються так звані slug, тобто зручні для читання та оптимізовані для

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

пошукових систем URL-адреси, з повною підтримкою української мови. Реалізована ієрархічна структура дозволяє створювати не лише основні категорії, а й необмежену кількість вкладених підкатегорій, що забезпечує логічну організацію асортименту. Адміністративна панель надає повний набір функцій CRUD для управління цими категоріями через комфортний інтерфейс з модальними вікнами, які асинхронно взаємодіють із сервером без необхідності перезавантаження сторінки.

Поряд з управлінням категоріями, ключовою функцією адміністративної панелі є повне управління життєвим циклом товарів, що реалізовано через набір операцій CRUD. Адміністратор може легко створювати нові товарні позиції, заповнюючи спеціальну форму, де вказуються назва, детальний опис, ціна, кількість на складі та завантажуються зображення. Існуючі товари можна в будь-який момент редагувати, оновлюючи будь-яку інформацію, від вартості та опису до статусу наявності. Також передбачена можливість видалення товарів, які більше не є актуальними. Весь асортимент товарів для зручності представлений у табличному вигляді, що дозволяє швидко знаходити потрібні позиції та здійснювати над ними необхідні маніпуляції.

Система кошика покупок також спроектована для максимальної зручності клієнта. Для авторизованих користувачів стан кошика зберігається безпосередньо в базі даних MongoDB, що дозволяє їм продовжити покупки з будь-якого іншого пристрою або після тривалої перерви, зберігаючи обрані товари. Для гостей сайту дані кошика тимчасово зберігаються у серверній сесії. Функціонал кошика дозволяє легко додавати чи видаляти товари, змінювати їхню кількість та миттєво бачити автоматично перераховану загальну вартість. Процес оформлення замовлення було максимально автоматизовано та спрощено. Завдяки інтеграції з офіційним API Нової Пошти, користувачі можуть зручно обрати потрібне місто та відділення доставки прямо зі списку, що мінімізує ризик помилок при введенні адреси. Оплата замовлень відбувається через надійний та популярний в Україні платіжний сервіс LiqPay, що гарантує безпеку фінансових транзакцій. Перед створенням замовлення система додатково проводить

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

валідацію всіх введених даних та перевіряє актуальну наявність товарів на складі.

Візуальна частина сайту розроблена з використанням сучасного UI-фреймворку Bootstrap 5 [30] та доповнена власними каскадними таблицями стилів CSS, що разом забезпечує привабливий зовнішній вигляд та повну адаптивність інтерфейсу на всіх типах пристроїв, від смартфонів до настільних комп'ютерів. Для підвищення довіри та соціального доказу реалізовано систему відгуків, де користувачі можуть ділитися своїми враженнями про товари. Для покращення користувацького досвіду на сайті працює динамічна фільтрація та сортування товарів за різними параметрами, що дозволяє швидко знаходити потрібні позиції.

Загальна безпека системи підтримується шляхом ретельної валідації всіх вхідних даних на сервері, використання вбудованих механізмів захисту від поширених CSRF-атак [31] та застосування безпечних заголовків HTTP [32].

Однією з унікальних та практично значущих функцій розробленого інтернет-магазину є вбудований онлайн-тюнер для налаштування музичних інструментів, який працює безпосередньо у веббраузері користувача. Інтерфейс цього інструменту спроектовано з акцентом на мінімалізм та інтуїтивну зрозумілість. Центральним елементом є великий півкруглий індикатор з окремими сегментами для візуального відображення точності налаштування, а в його центрі відображається назва ноти, що звучить у даний момент. Щоб розпочати процес налаштування, користувачеві достатньо один раз клікнути на область індикатора, після чого браузер надішле стандартний запит на отримання дозволу на використання мікрофона, що є необхідним для аналізу звуку інструмента.

Технічна реалізація визначення висоти тону базується на сучасній технології машинного навчання. Для цього використовується JavaScript-бібліотека ml5.js, яка з високою точністю аналізує звуковий потік у реальному часі та визначає частоту звуку. Коли користувач грає ноту на своєму інструменті, тюнер миттєво розпізнає її та відображає назву українською мовою, наприклад

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

"До", "Ре" або "Мі", що робить інструмент максимально зручним для української аудиторії.

Для того, щоб налаштувати свій інструмент, музикант просто грає ноту на одній зі струн та, орієнтуючись на візуальні підказки тюнера, підтягує або послаблює кілочок доти, доки не побачить зелений індикатор. Щоб зупинити роботу тюнера та звільнити доступ до мікрофона, достатньо ще раз клікнути на індикатор. Весь процес є простим, швидким та оптимізованим для плавної роботи без затримок, що робить цей інструмент корисним як для початківців, так і для досвідчених музикантів

Висновки до другого розділу

В результаті виконаної роботи було успішно спроектовано та реалізовано повнофункціональну веб-платформу інтернет-магазину. Створено надійну серверну архітектуру на базі Node.js та Express.js за принципом MVC, а також гнучку систему зберігання даних з використанням хмарної бази даних MongoDB Atlas. Було впроваджено весь необхідний для електронної комерції функціонал, включаючи захищену систему автентифікації, повне управління каталогом через адміністративну панель, а також зручний цикл покупки з інтеграцією сервісів LiqPay та Нова Пошта. Окрім стандартних можливостей, було розроблено унікальний інструмент у вигляді онлайн-тюнера на основі технології машинного навчання. Особливу увагу було приділено безпеці, адаптивності інтерфейсу та загальній зручності користування, що в сукупності створює готовий до експлуатації програмний продукт, який відповідає всім поставленим вимогам.

Під час програмної реалізації було збудовано серверну частину на основі технологій Node.js та Express.js. Впроваджено механізм надійної аутентифікації, з використанням JWT-токенів та хешування паролів. Розроблено базовий бізнес-функціонал. До нього увійшли повне керування каталогом та товарів, система кошика з підтримкою сесій. Результатом роботи, описаної в даному розділі, є створений програмний прототип інтернет-магазину з реалізованими основними функціями, повністю готовий до етапу фінального тестування.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1 Порядок встановлення та налаштування системи

У даному підрозділі розглядається архітектура розгортання розробленого інтернет-магазину, а також логічна взаємодія між його ключовими компонентами у робочому середовищі. Платформа спроектована за принципом трирівневої архітектури, що включає рівень представлення, рівень застосунку та рівень даних, і не вимагає від кінцевого користувача встановлення жодного спеціалізованого програмного забезпечення. Взаємодія з системою відбувається через стандартний веббраузер, встановлений на пристрої користувача, що забезпечує максимальну доступність та легкість використання платформи. Серверна частина, яка є ядром системи і розроблена на технології NodeJS, розгортається на віддаленому хмарному хостингу, що виступає центральним вузлом для обробки всієї бізнес-логіки. Вона приймає та обробляє запити від клієнтських браузерів, а також координує роботу з іншими елементами архітектури. Для зберігання всієї інформації використовується хмарна база даних MongoDB Atlas, яка функціонує як окремий, незалежний сервіс, а зв'язок між сервером та базою даних відбувається через захищений мережевий протокол. Крім того, для реалізації платіжного функціоналу та опцій доставки система інтегрується із зовнішніми сервісами, надсилаючи безпечні HTTPS запити до офіційних API LiqPay та Нової Пошти. Така розподілена архітектура гарантує високу надійність, безпеку та масштабованість, дозволяючи кожному компоненту виконувати свою чітко визначену роль.

Діаграма розгортання є важливим візуальним представленням, що демонструє конфігурацію оброблювальних вузлів та компонентів системи під час її функціонування. Вона дозволяє наочно відобразити фізичне розміщення всіх програмних елементів у робочому середовищі, а також простежити взаємозв'язки між ними. Таким чином представлено саме таку діаграму для розробленого інтернет-магазину музичних інструментів (рис 3.1). В основі системи лежить класична та надійна клієнт-серверна архітектура, що передбачає

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

чіткий поділ на фронтенд, з яким взаємодіє користувач, бекенд, що містить основну логіку, окрему базу даних та інтегровані зовнішні сервіси.

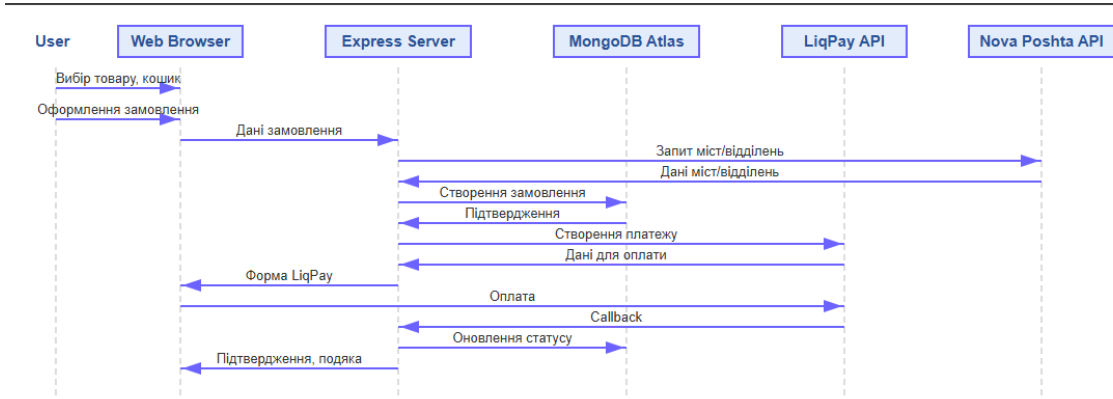


Рис. 3.1 Діаграма послідовності оформлення замовлення

Процес взаємодії з системою починається з клієнтської частини, яку становить звичайний веб-браузер на пристрої користувача. Важливо, що для роботи з магазином не потрібно встановлювати будь-яке додаткове програмне забезпечення, адже інтерфейс сайту, реалізований за допомогою шаблонізатора Handlebars та JavaScript, повністю відображається у вікні браузера. Користувач взаємодіє з цим інтерфейсом, надсилаючи HTTP-запити до серверної частини. Серверна логіка, побудована на платформі Node.js з використанням фреймворку Express.js, виступає центральним мозком усієї системи. Вона відповідає за обробку цих запитів, забезпечує маршрутизацію, проводить авторизацію користувачів, а також керує всіма основними операціями, такими як перегляд товарів, робота з кошиком та оформлення замовлень. Уся бізнес-логіка виконується на рівні серверного застосунку, який отримує HTTP-запити, активує відповідні контролери, перевіряє наявність товарів у базі даних та взаємодіє з MongoDB Atlas через зручний ODM Mongoose. Сам сервер бази даних MongoDB Atlas працює як окремий хмарний сервіс, де надійно зберігається вся інформація про користувачів, товари та замовлення, а обмін даними з ним відбувається через захищене мережеве з'єднання. Для реалізації фінансових операцій система інтегрується з платіжним шлюзом LiqPay. Сервер надсилає до нього безпечні HTTPS-запити для створення й підтвердження транзакцій, а також обробляє

спеціальні callback-повідомлення, які надходять від LiqPay після зміни статусу платежу. Для організації доставки використовується API Нової Пошти, з яким сервер також взаємодіє через захищені HTTP-запити, щоб отримувати актуальні списки міст та відділень і надавати можливість відстеження відправлень.

3.2 Структура інтерфейсу та порядок взаємодії з інтернет-магазином

Інтерфейс інтернет-магазину спроектовано з головною метою забезпечити максимальну зручність та інтуїтивну зрозумілість для користувача на кожному етапі його взаємодії з платформою. Головна сторінка виступає цифровою вітриною, яка зустрічає чітко структурованим меню категорій, що дозволяє одразу перейти до потрібного розділу, та. Унікальною особливістю є інтегрований онлайн-тюнер для налаштування інструментів, що додає практичної цінності для музикантів. При переході до каталогу товарів користувач отримує потужний інструмент для пошуку у вигляді системи можливості сортування продукції за ціною чи назвою. Сторінка конкретного товару є центром ухвалення рішення, тому вона містить вичерпну інформацію, включаючи галерею якісних зображень, детальний опис, технічні характеристики, а також блок з відгуками інших покупців. Процес покупки починається з додавання товару в кошик та переходу на сторінку оформлення замовлення, де користувач проходить кілька простих кроків, а саме заповнення контактних даних, автоматизований вибір відділення Нової Пошти та перехід до безпечної сторінки оплати через сервіс LiqPay. Для зареєстрованих користувачів передбачено особистий кабінет, де можна відстежити історію замовлень та керувати своїми даними. Адміністратор, у свою чергу, отримує доступ до окремої панелі управління, яка є повноцінним робочим інструментом для контролю над усім контентом та замовленнями сайту.

Після відкриття сайту користувач потрапляє на головну сторінку інтернет-магазину, це вказано на (рис. 3.2-3.4), де представлено зручний хедер з навігацією по сайту, всі контактні дані з магазином, пошук.

Під фото з текстом «Музична крамниця NoteUA» розташоване зручно структуроване меню категорій, що дає змогу в один клік перейти до потрібного

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

розділу. Окремою перевагою є інтегрований онлайн-тюнер для налаштування музичних інструментів, який доступний прямо з головної сторінки та робить платформу корисною не лише для покупок, а й для повсякденної практики музикантів.

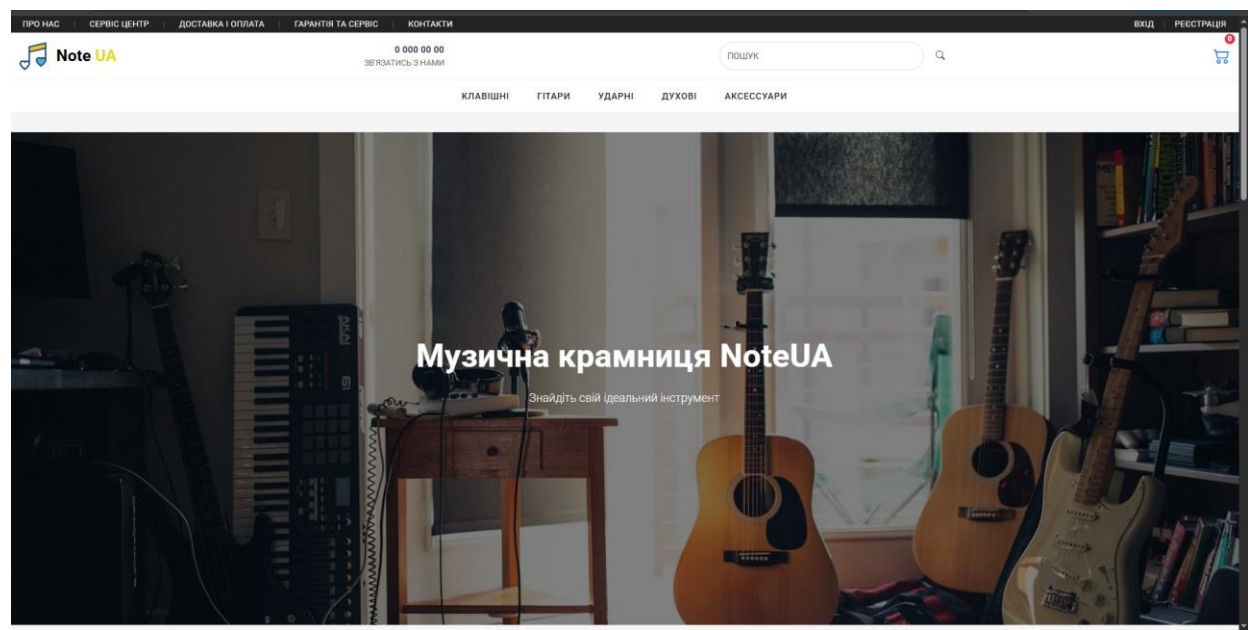


Рис. 3.2 Головна сторінка

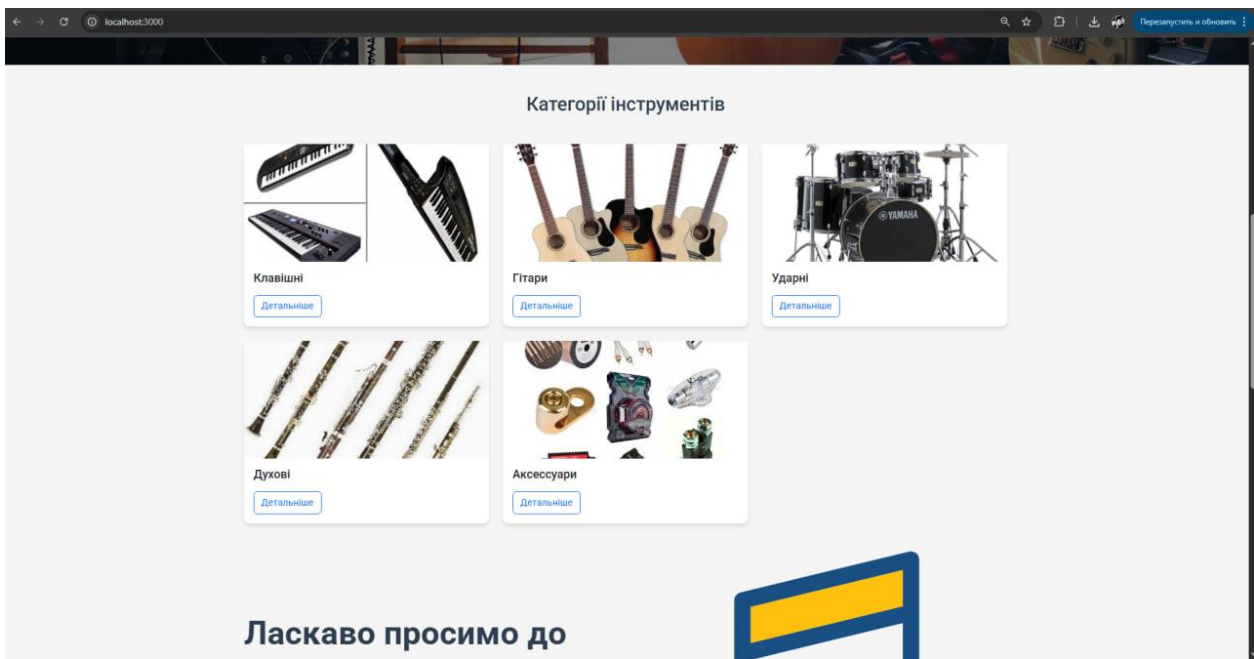


Рис. 3.3 Головна сторінка(Категорії інструментів)

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

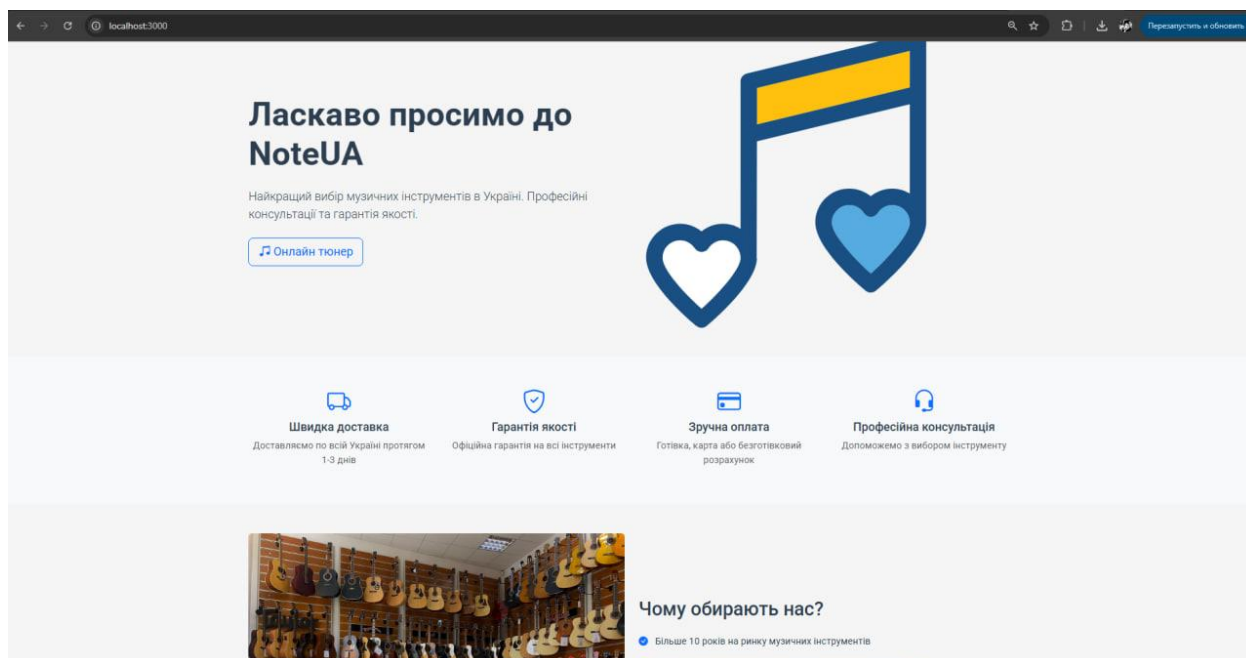


Рис. 3.4 Головна сторінка(Онлайн тюнер для налаштування музичного інструменту)

Для того щоб використовувати майже усі функції інтернет-магазину користувачу потрібно авторизуватись, натиснувши кнопку «Вхід» у верхній частині сайту (рис 3.5.). Після цього відкривається сторінка авторизації, на якій користувач вводить свої облікові дані (електронну пошту та пароль) для доступу до персонального кабінету.

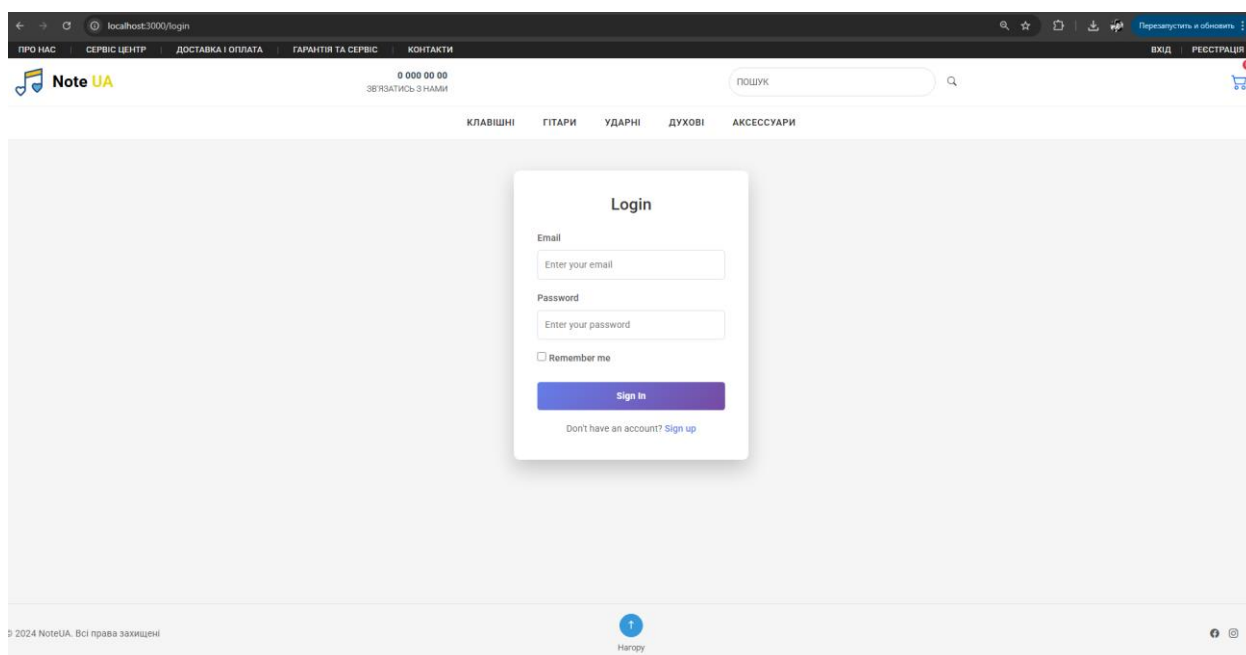


Рис. 3.5 Сторінка входу

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докum.	Підпис	Дата		44

Якщо користувач ще не зареєстрований, він може натиснути на кнопку «Реєстрація» у верхньому правому куті сторінки. Далі відкривається форма для реєстрації, в якій необхідно вказати ім'я, адресу електронної пошти та пароль. Якщо обліковий запис вже є, можна скористатися кнопкою «Увійти» для авторизації. Після цього його перенаправить на головну сторінку. Інтерфейс сторінки реєстрації зображено на (рис. 3.6.)

Рис. 3.6 Сторінка реєстрації

Процес реєстрації або входу в систему реалізовано через просту та зрозумілу форму. Після авторизації відкривається доступ до особистого кабінету, де можна переглядати історію замовлень, змінювати особисті дані та керувати налаштуваннями профілю. Це продемонстровано на (рис 3.7-3.8).

Рис. 3.7 Сторінка «Профіль» в особистому кабінеті користувача

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

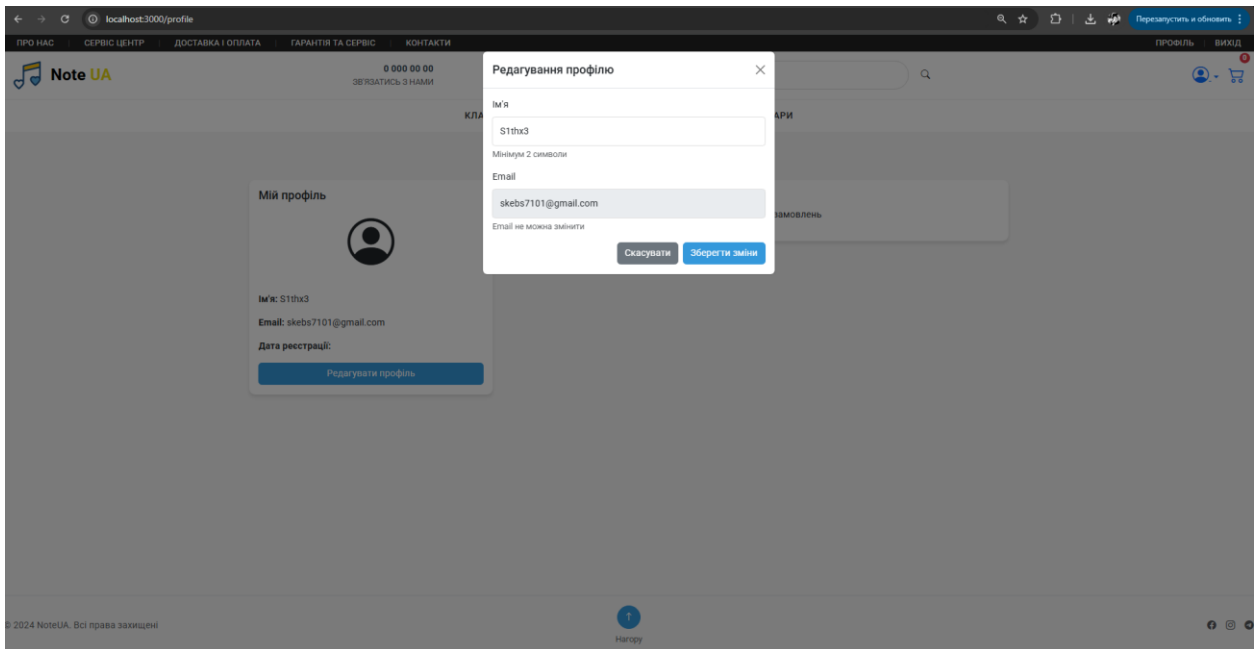


Рис. 3.8 Вкладка «Редагувати профіль» в особистому кабінеті користувача.

Звичайний авторизований користувач на сайті інтернет-магазину музичних інструментів отримує більше можливостей порівняно з неавторизованим відвідувачем. Після входу в систему користувач може оформлювати замовлення, оскільки його контактні дані автоматично підтягуються у форму, що значно спрощує процес покупки. Крім того, авторизований користувач може залишати відгуки до придбаних товарів, оцінювати їхню якість і ділитися власним досвідом з іншими покупцями.

Переходячи до категорій, користувач отримує у своє розпорядження зручну систему сортування, яка дозволяє впорядкувати товари за ціною, назвою, що значно спрощує пошук потрібного інструменту серед великої кількості позицій. Також відображаються усі товари з наявної категорії, якщо користувач захоче знайти саме той музичний інструмент який йому потрібний він натискає на підкатегорію яка так само присутня на сторінці з товарами, в ній окремо будуть товари які належать до данної підкатегорії. Це продемонстровано на нижче зображених (рис.3.9-3.10).

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						46

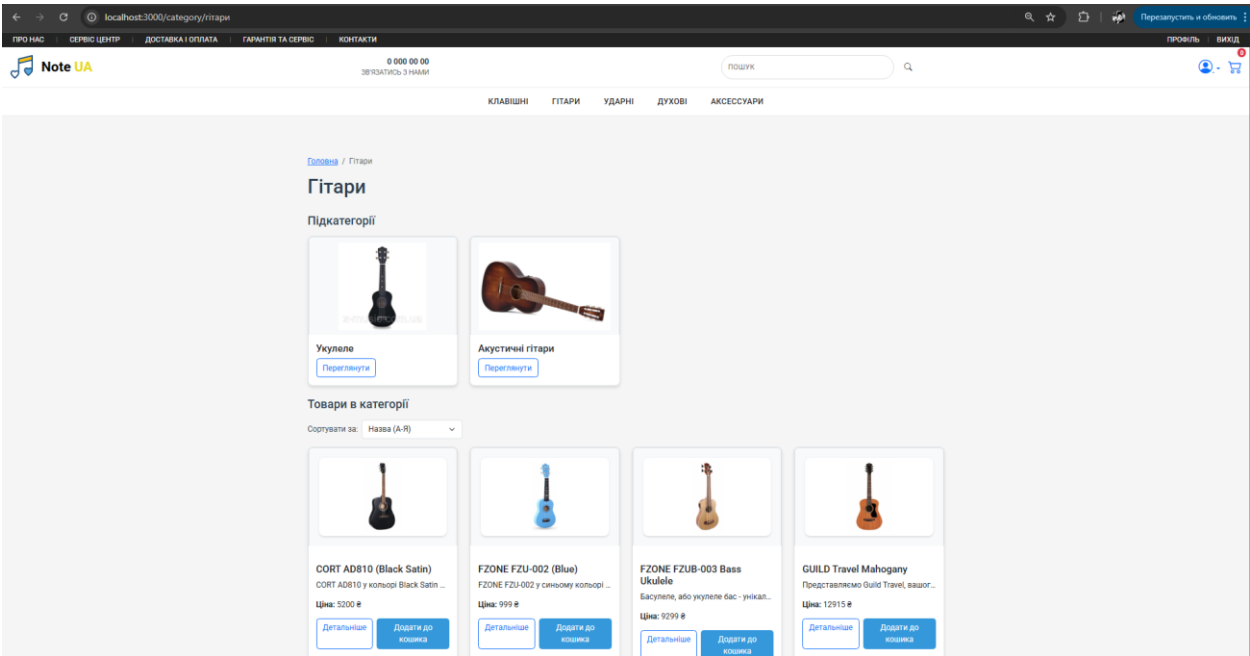


Рис. 3.9 Сторінка категорій з товарами

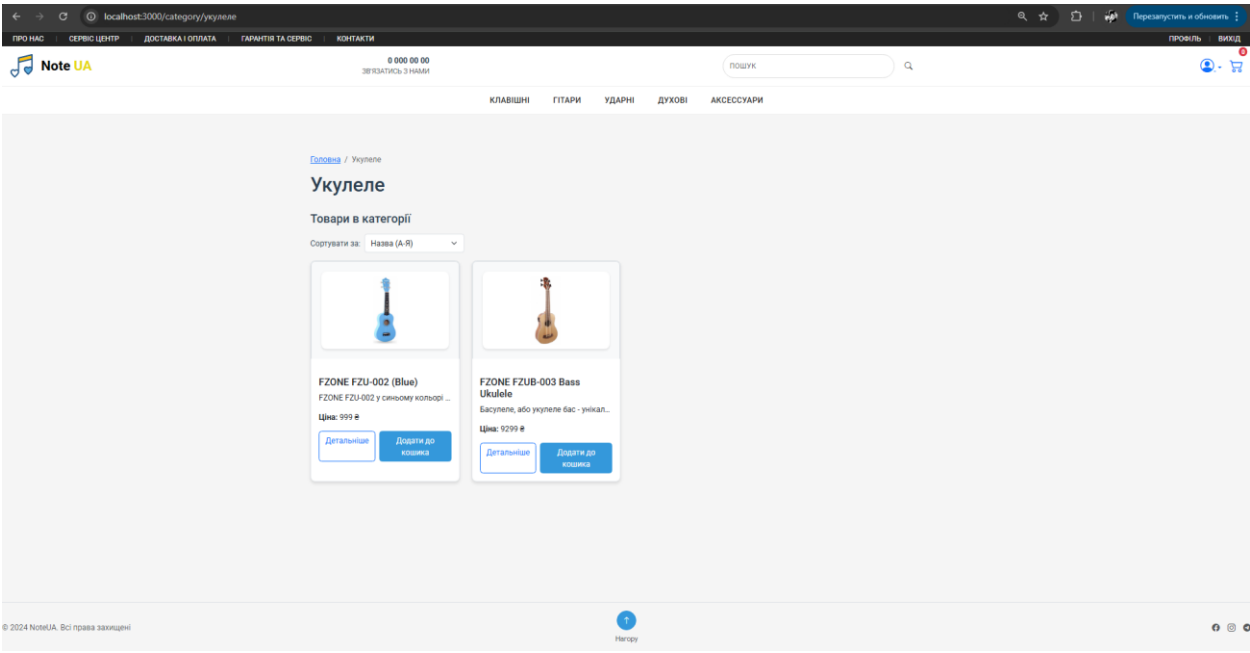


Рис. 3.10 Сторінка підкатегорій

Сторінка окремого товару яка продемонстрована на (рис 3.11) є одним із найважливіших елементів усього інтернет-магазину, адже саме тут відвідувач ухвалює фінальне рішення про покупку. Центральне місце на сторінці займає візуальна презентація продукту, реалізована у вигляді великої галереї якісних фотографій. Користувачеві надається можливість переглядати інструмент з різних ракурсів та у збільшеному вигляді, що дозволяє детально роздивитися текстуру матеріалів, якість збірки та дрібні елементи дизайну, компенсуючи

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		47

відсутність фізичного контакту з товаром. Поруч з візуальним блоком розміщується вичерпна інформаційна частина. Вона включає детальний опис, який розповідає про особливості звучання та переваги інструмента, а також окремий блок з точними технічними характеристиками для досвідчених покупців. Ключовим елементом, що спонукає до дії, є яскраво виражена кнопка «Додати в кошик».

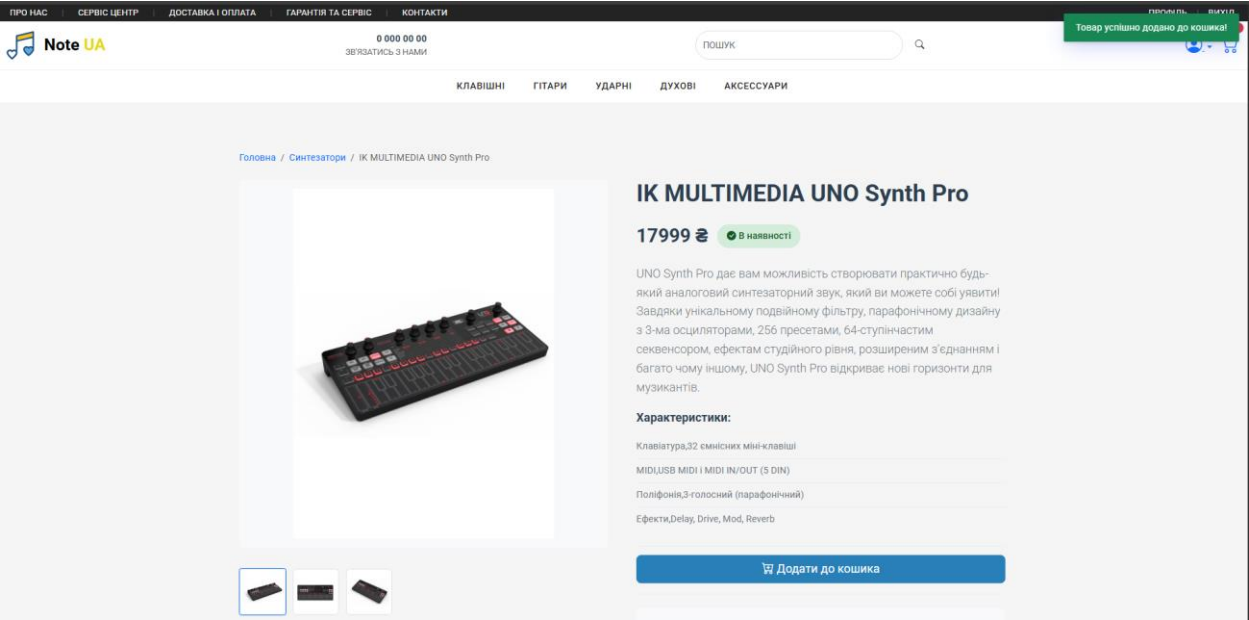


Рис. 3.11 Сторінка товару(флеш повідомлення що товар додано)

Нижче розташований блок із відгуками інших покупців(рис. 3.12), де можна ознайомитися з реальним досвідом використання товару, а за бажанням залишити власний коментар.

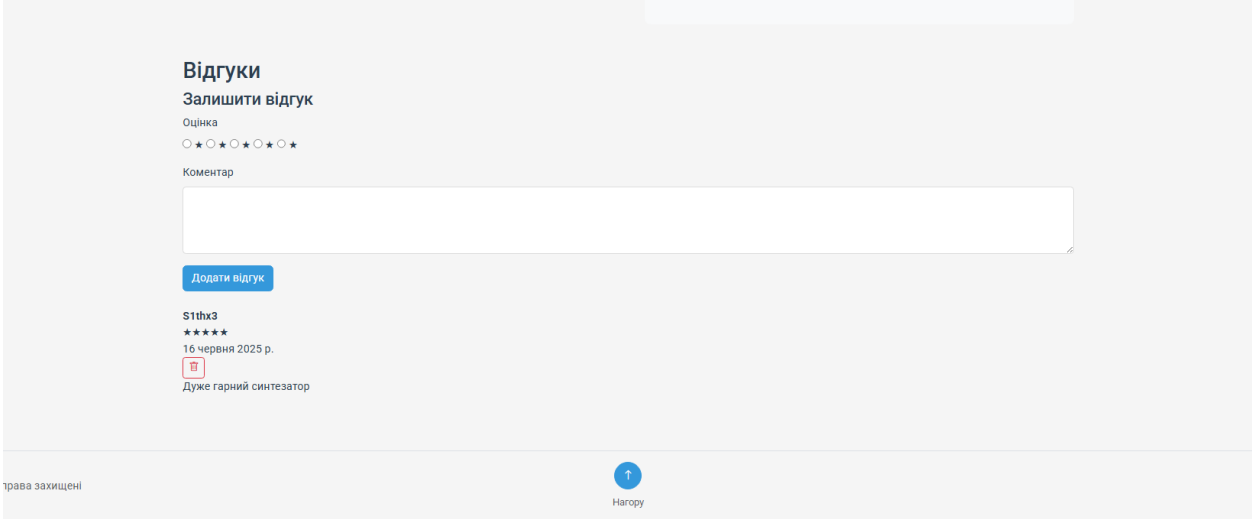


Рис. 3.12 Відгуки

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		48

Коли користувачу потрібно знайти конкретний товар за його назвою, йому достатньо ввести ключові слова у це поле(рис. 3.13). Система миттєво обробляє запит, здійснюючи пошук збігів у назвах усіх наявних у каталозі продуктів. Після цього користувача автоматично перенаправляє на сторінку результатів, де всі знайдені товари представлені у зручному форматі. Кожен знайдений продукт відображається з фотографією, назвою та ціною, що дозволяє швидко оцінити релевантність результатів та перейти на сторінку товару для подальшого ознайомлення.

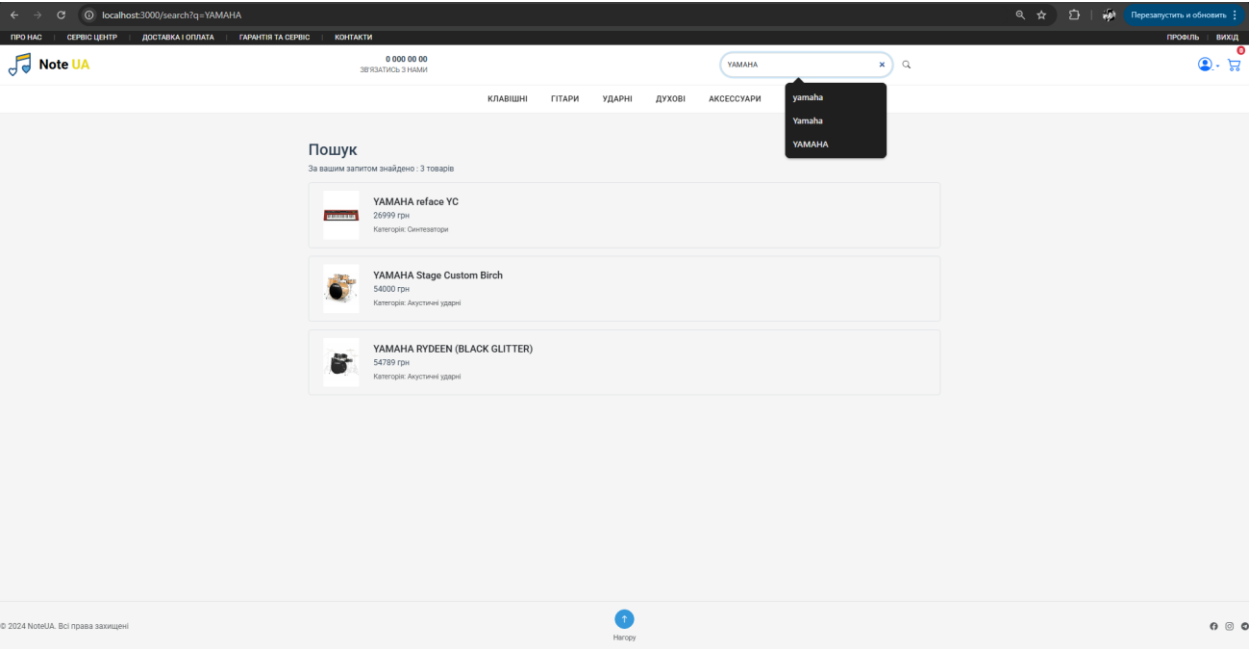


Рис. 3.13 Сторінка пошуку товарів

Сторінка кошика (рис. 3.14) важливий перехідний пункт між рішенням купити певні товари та завершенням процесу замовлення. На цій сторінці, у формі чіткого переліку, представлені усі товари, які відвідувач вже додав до кошика. Для кожної позиції показано основну інформацію, а саме її назву, зображення та вартість. Клієнт має повний контроль над вмістом кошика, адже легко може скоригувати кількість одиниць кожного товару або повністю вилучити зайві позиції зі списку. Знизу під переліком товарів завжди міститься блок із загальною сумою замовлення, яка змінюється при кожному оновленні. Тут також розташовано головну кнопку для переходу до оформлення покупки.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		49

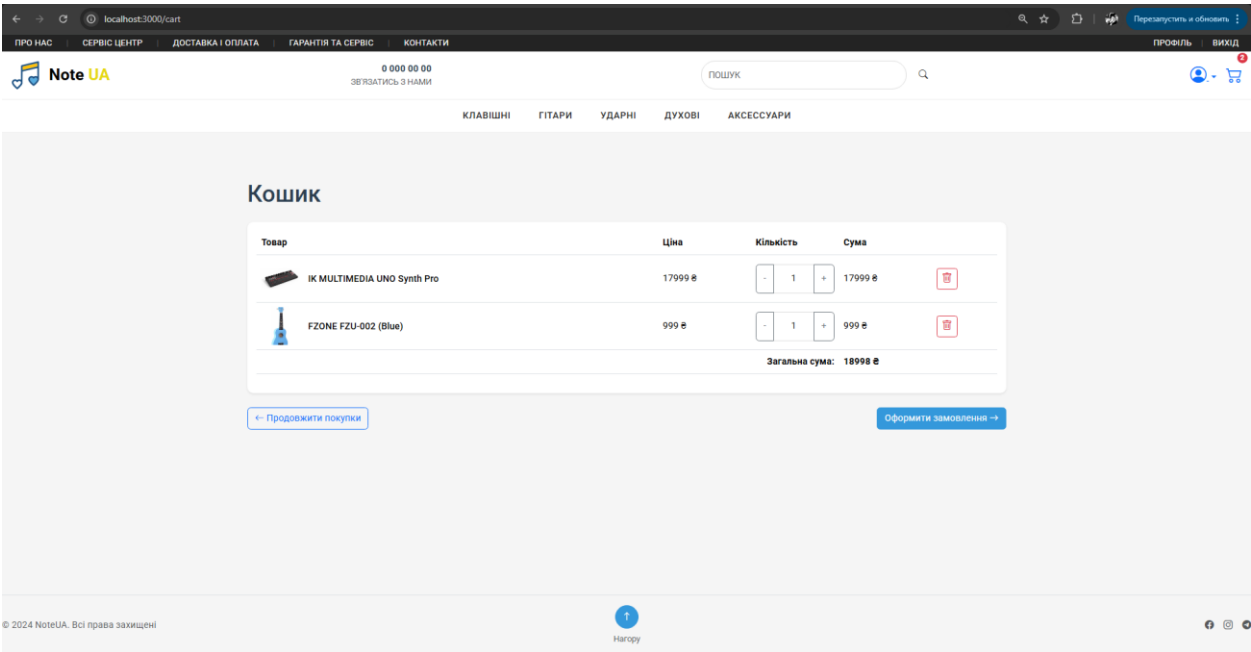


Рис. 3.14 Сторінка корзини

На сторінці оформлення замовлення (рис.3.15) користувач заповнює контактні дані, обирає місто та відділення Нової Пошти через інтегровану систему підказок, а також вибирає спосіб оплати наприклад, онлайн через LiqPay. Після підтвердження замовлення система перенаправляє користувача на сторінку подяки або, у разі онлайн-оплати, на захищену платіжну сторінку.

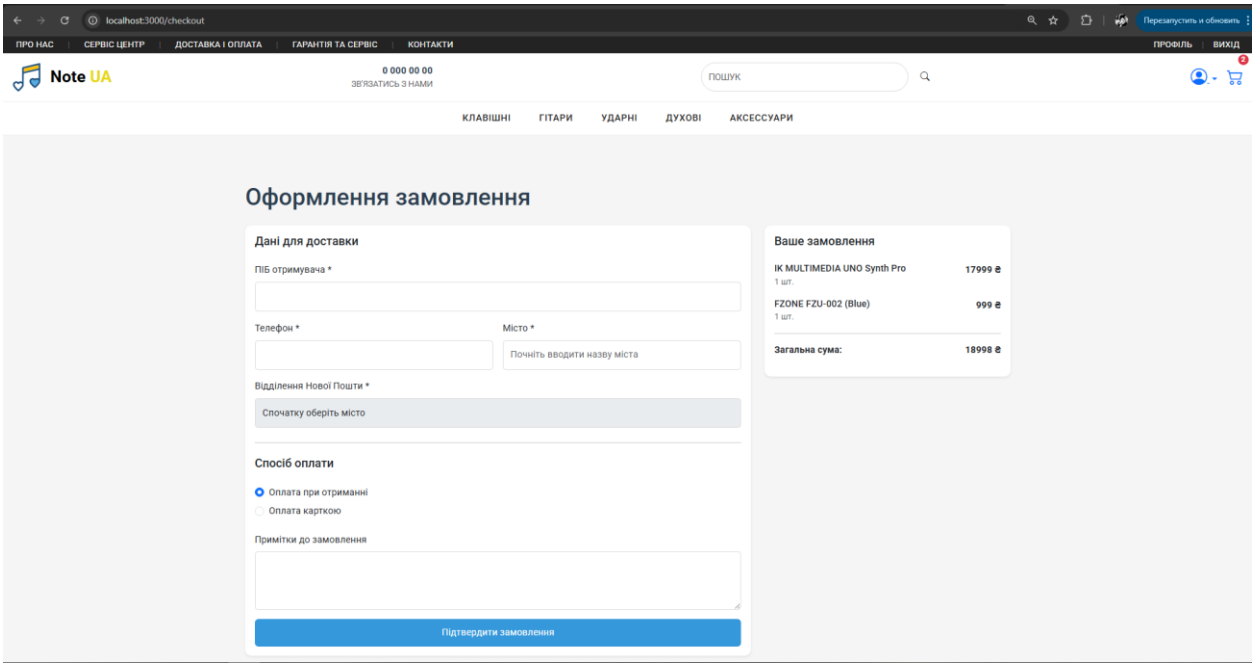


Рис. 3.15 Сторінка оформлення замовлення

Вибір усіх доступних відділень або поштоматів у вказаному місті (рис.3.16). які відображаються у вигляді списку.

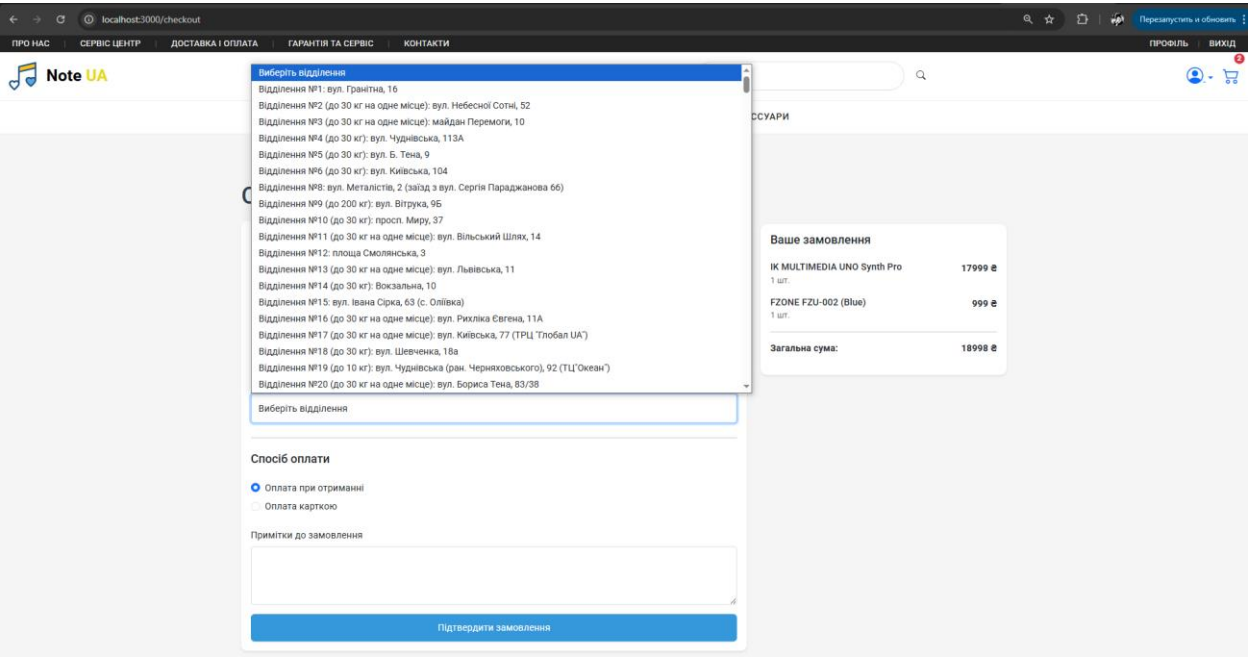


Рис. 3.16 Інтегрована система Нової пошти

Якщо підтвердити замовлення та вибрати оплата карткою то з’явиться форма від liqpay і попросить данні для оплати, після успішної оплати, користувача направить на свій профіль або на головну.Це наведено на (рис. 3.17-3.19).

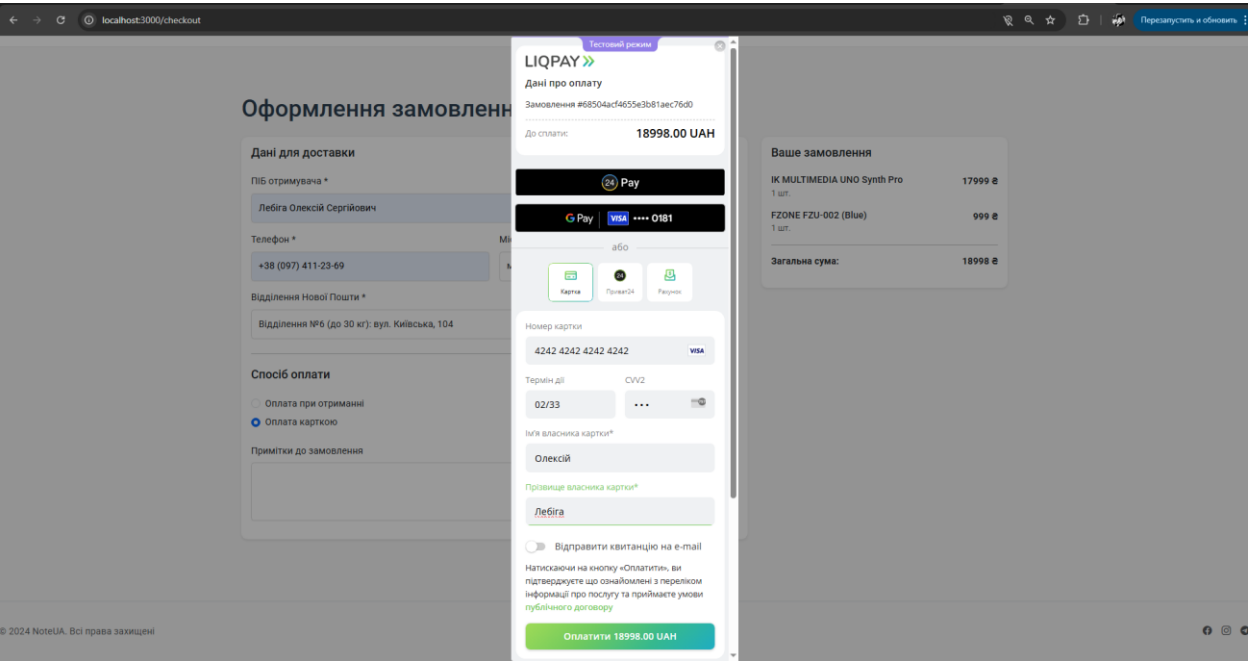


Рис. 3.17 Оплата Liqpay

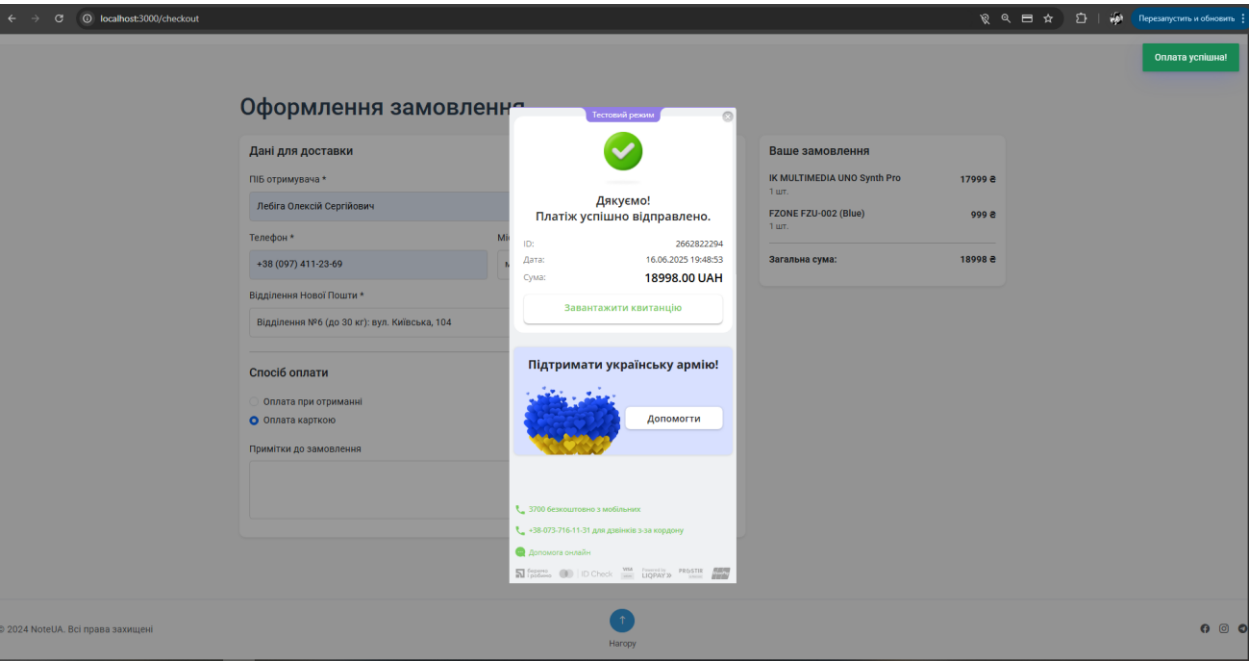


Рис. 3.18 Оплата Liqpay(успішна)

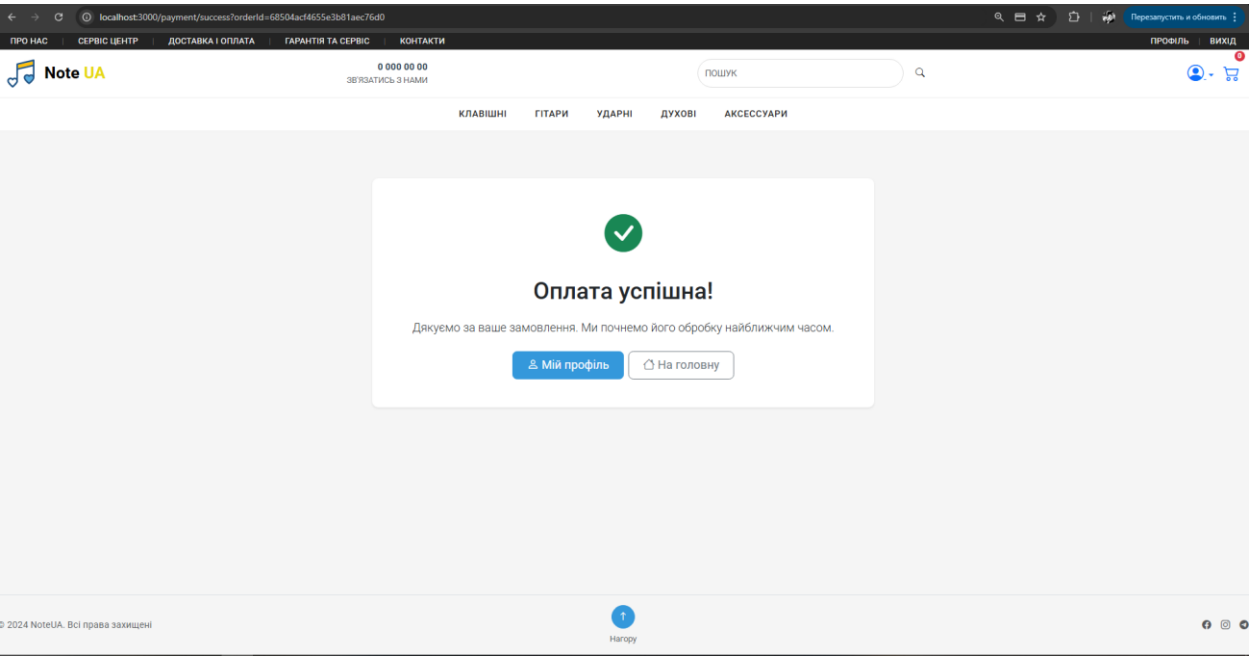


Рис. 3.19 Сторінка після оплати з вибором переходу у профіль або на головну

На головній сторінці є особлива функція сайту це онлайн-тюнер для налаштування музичного інструменту яка використовує мікрофон для виявлення звуків(рис. 3.20).

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

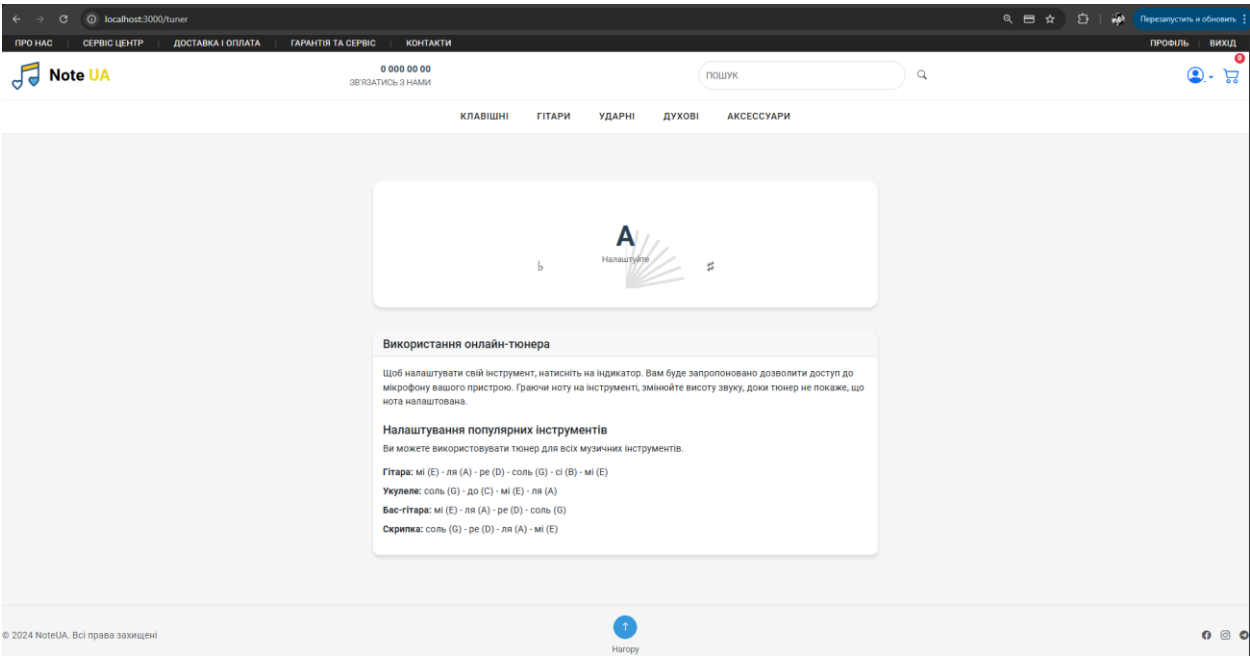


Рис. 3.20 Онлайн-тюнер

За допомогою нього можна налаштувати музичний інструмент а саме Гітара, Укулеле, Бас-гітара, Скрипка.Ньому є індикатори які підсвічуються кольорами та з’являється назви нот (рис. 3.21) що інформують користувача коли інструмент налаштований, а коли ні.

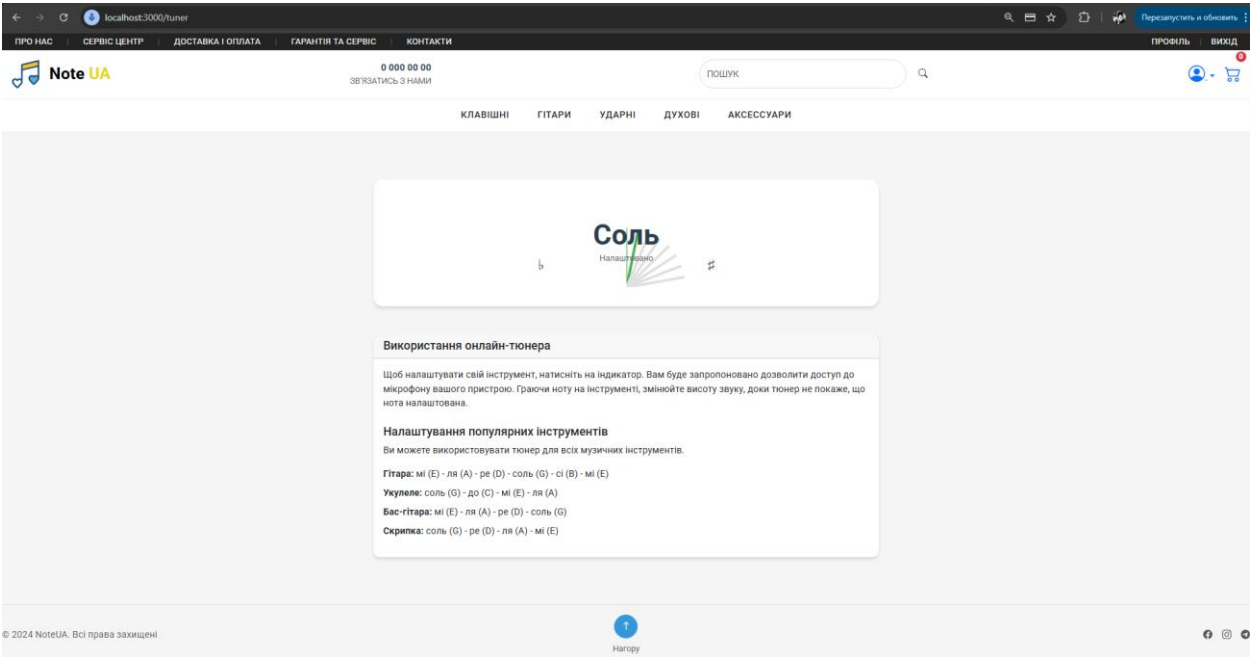


Рис. 3.21 Онлайн-тюнер (Музичним інструментом була Укулеле перша нота)

Інтерфейс адмін-панелі (рис.3.22) розроблений таким чином, щоб забезпечити максимально ефективне керування всіма ключовими аспектами роботи магазину. На головній сторінці панелі адміністратор бачить зведену статистику щодо кількості замовлень, товарів та користувачів, а також швидкі посилання на основні розділи.

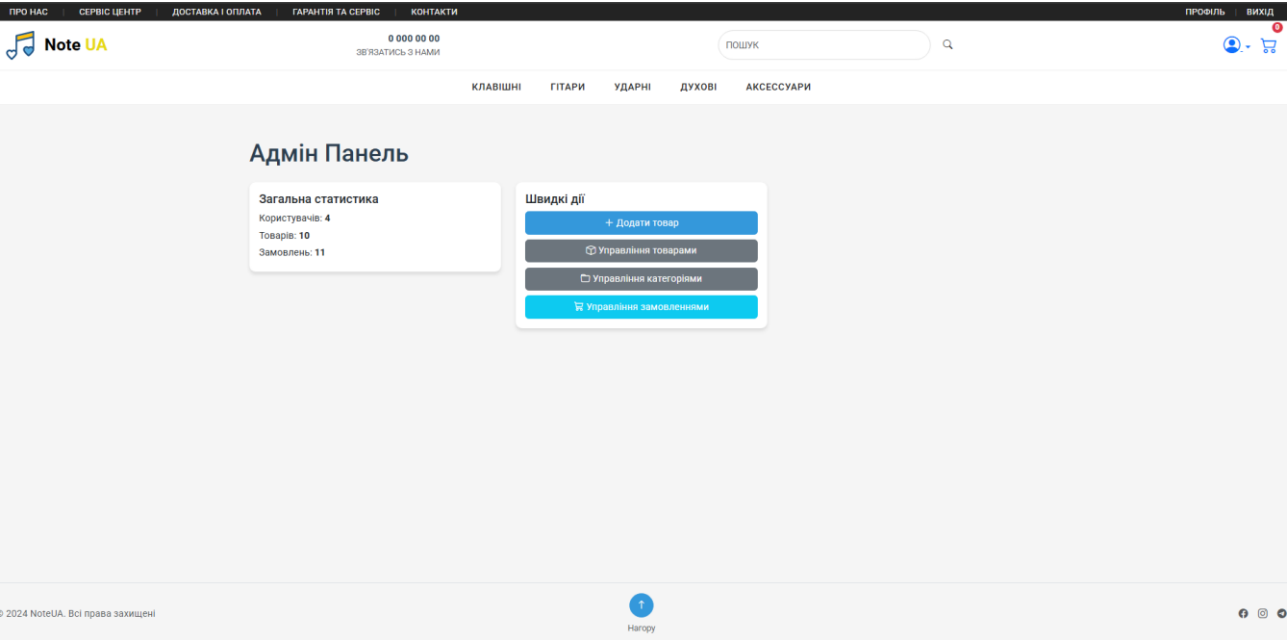


Рис. 3.22 Адмін панель

Після натискання кнопки додати товар (рис.3.23) відкривається вікно в якому можна створити новий товар.

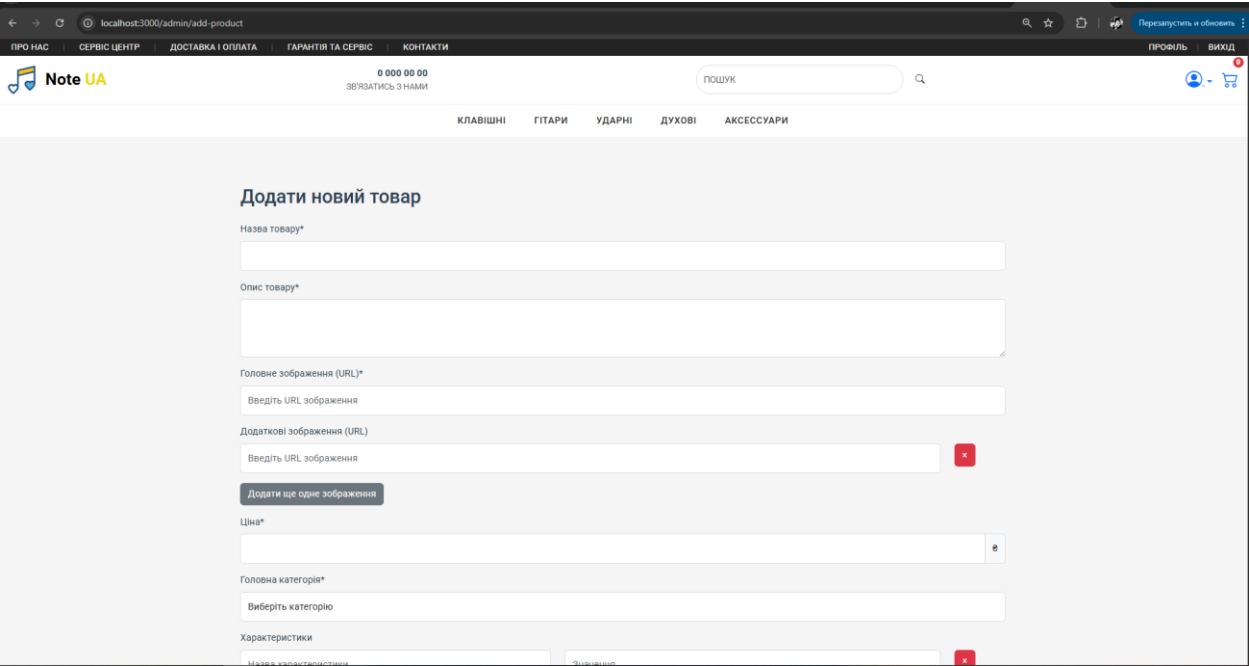


Рис. 3.23 Сторінка додавання товару

Наступна швидка дія у адмін панелі це управління товарами (рис.3.24) є можливість використовувати CRUD операції.

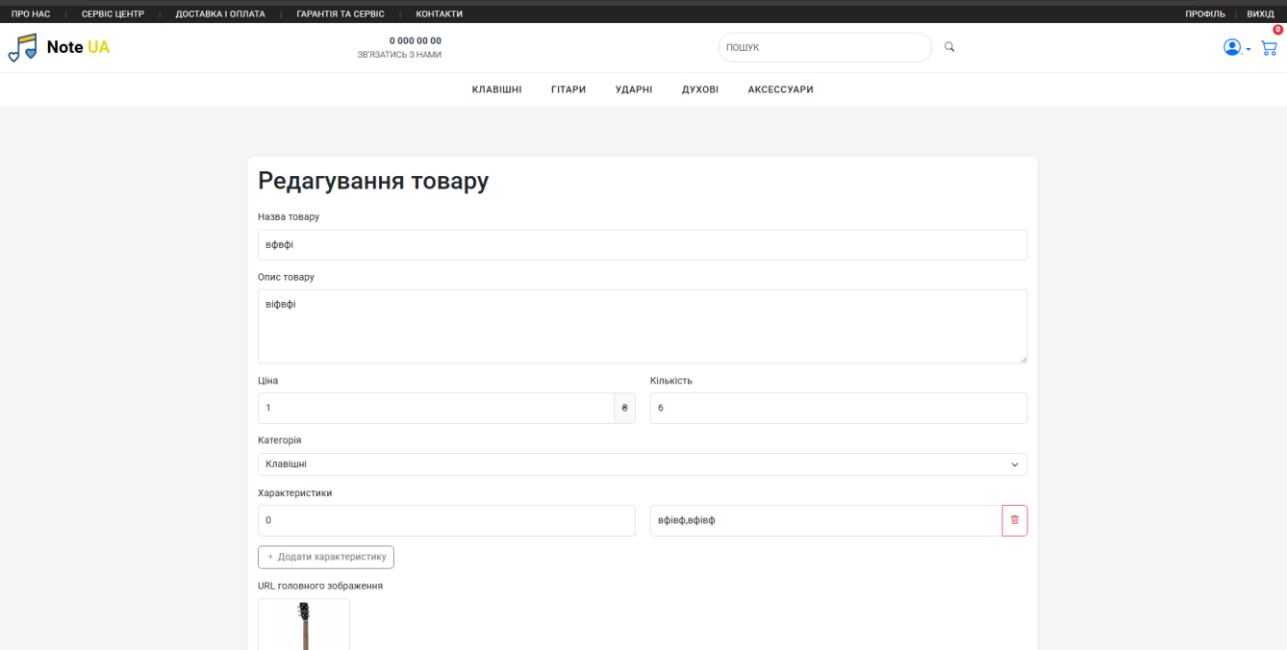


Рис. 3.24 Сторінка редагування товарами.

Також є управління категоріями в якій є можливість редагувати та створювати як категорії так і підкатегорії.Це наведено у (рис. 3.25-3.27).

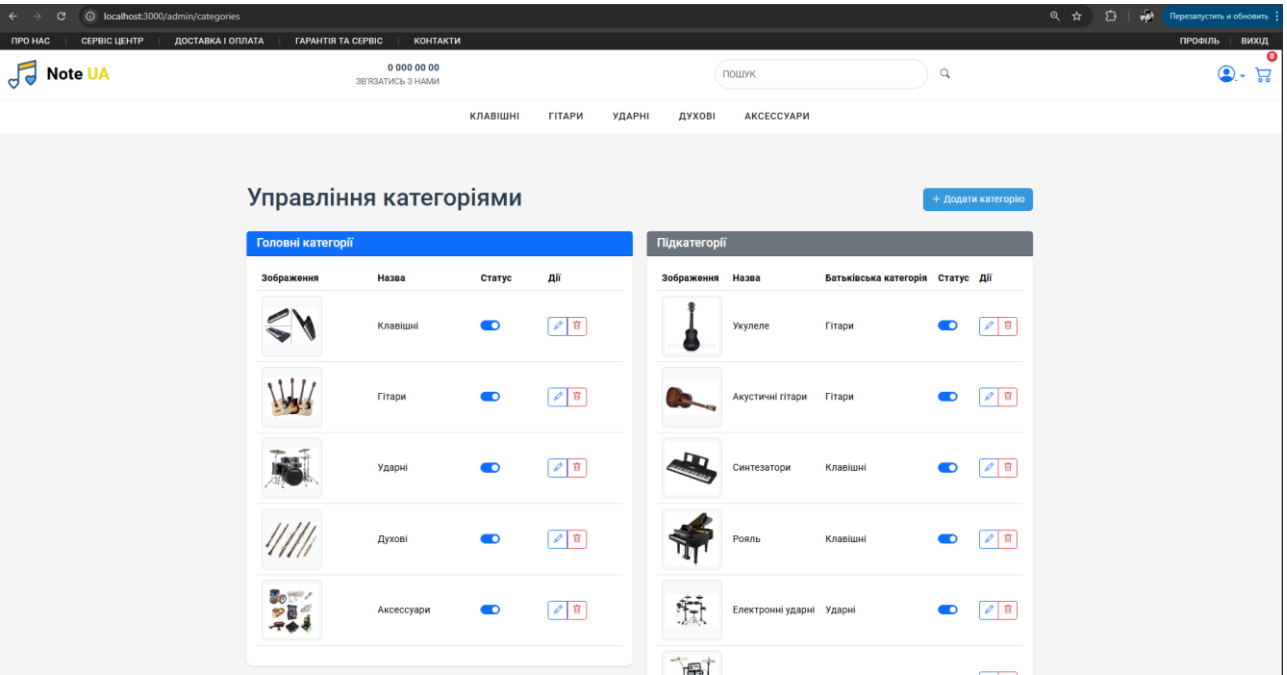


Рис. 3.25 Сторінка редагування категоріями.

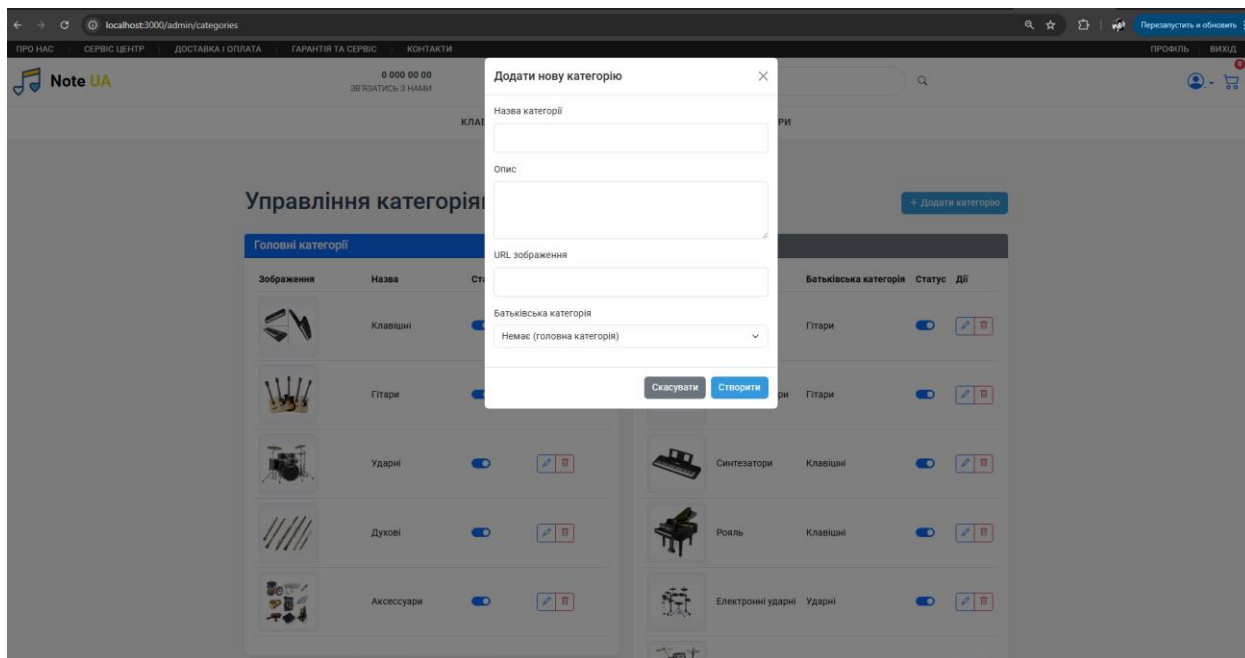


Рис. 3.26 Створення категорії.

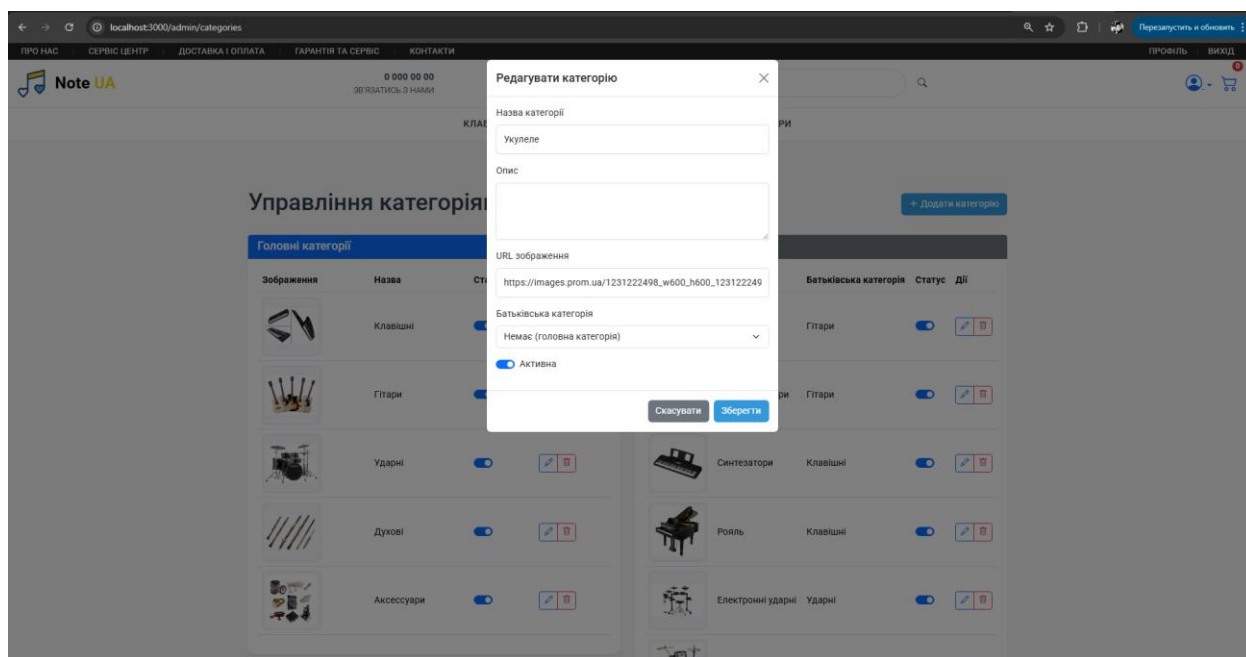


Рис. 3.27 Редагування наявної категорії.

Останньою дією є управління замовленнями (рис. 3.28), на ній відображена таблиця з переліком усіх покупок які виконали користувачі на інших аккаунтах, де для кожного замовлення вказані усі данні. Адміністратор має можливість змінити статус замовлення та подивитись усе детальніше.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				56
Змн.	Арк.	№ докум.	Підпис	Дата		

ПРО НАС

СЕРВІС ЦЕНТР

ДОСТАВКА І ОПЛАТА

ГАРАНТІЯ ТА СЕРВІС

КОНТАКТИ

0 000 00 00

ЗВ'ЯЗАТИСЬ З НАМИ

ПОШУК

ПРОФІЛЬ

ВИХІД

КЛАВІШНИ

ГІТАРИ

УДАРНИ

ДУХОВІ

АКСЕСУАРИ

Управління замовленнями

ID замовлення	Клієнт	Сума	Статус	Статус оплати	Дата	Дії
6852fa843279e1bb5ca138fd	S1thx3 skebs7101@gmail.com	999 грн	Підтверджено	Оплатено	18 червня 2025 р.	Деталі
6851f0b3a229938a70d33241	S1thx3 skebs7101@gmail.com	1 грн	Підтверджено	Оплатено	18 червня 2025 р.	Деталі
6851ec382f6b6aeeed8455ee8	S1thx3 skebs7101@gmail.com	1 грн	Відправлено	Оплатено	18 червня 2025 р.	Деталі
6851ec022f6b6aeeed8455e88	S1thx3 skebs7101@gmail.com	53.96 грн	В обробці	Оплатено	18 червня 2025 р.	Деталі
6851ebaf2f6b6aeeed8455e60	S1thx3 skebs7101@gmail.com	53.96 грн	В обробці	Оплатено	18 червня 2025 р.	Деталі
6851ea010851c01c4a88ef72	S1thx3 skebs7101@gmail.com	53.96 грн	В обробці	Оплатено	18 червня 2025 р.	Деталі
6851e9db0851c01c4a88ef4b	S1thx3 skebs7101@gmail.com	53.96 грн	В обробці	Оплатено	18 червня 2025 р.	Деталі
6851e93f0851c01c4a88ee7f	S1thx3 skebs7101@gmail.com	5201 грн	Підтверджено	Оплатено	18 червня 2025 р.	Деталі
6851e8509f73e6a22eeeb41	S1thx3 lebigasem13@gmail.com	1 грн	Підтверджено	Оплатено	18 червня 2025 р.	Деталі
6851e7cfff609aeb0312aa1	S1thx3 skebs7101@gmail.com	1 грн	Скасовано	Оплатено	18 червня 2025 р.	Деталі

1

2

Наступна

Рис. 3.28 Перегляд замовлень.

3.3 Порядок встановлення та налаштування системи

Після завершення фази основної розробки програмного продукту настав завершальний та один із найвідповідальніших етапів, а саме всебічне тестування системи (табл. 3.1). Цей процес є обов'язковою та невід'ємною складовою створення якісного і надійного програмного забезпечення, що гарантує відповідність кінцевого продукту поставленим вимогам. Головною метою цього етапу була комплексна перевірка та валідація розробленого функціоналу, контроль коректної роботи усіх елементів системи на кожному рівні, а також своєчасне виявлення та подальше усунення будь-яких потенційних дефектів чи помилок у бізнес-логіці. Процес випробувань охоплював декілька ключових напрямків для забезпечення повноцінного аналізу. Було проведено функціональне тестування, що полягало у покроковій перевірці кожного сценарію взаємодії для підтвердження того, що всі функції працюють згідно з очікуваннями. Також було виконано тестування користувацького інтерфейсу для перевірки його зручності, інтуїтивної зрозумілості та візуальної коректності. Крім того, проводилось тестування на сумісність для гарантії стабільної та однакової роботи сайту у різних популярних веббраузерах та на пристроях з різними розмірами екранів, включаючи настільні комп'ютери, планшети та смартфони.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

Детальній перевірці підлягали всі основні сценарії взаємодії з сайтом, починаючи з процесу реєстрації нового користувача та його подальшої авторизації, і завершуючи повним циклом здійснення покупки. Цей цикл включав у себе коректне додавання продукту до кошика, покрокове оформлення замовлення, правильність валідації введених даних та успішне проходження оплати через інтегрований платіжний сервіс LiqPay з подальшим підтвердженням транзакції. Окрему увагу було приділено випробуванню адміністративної панелі, де перевірялася стабільність та коректність роботи всіх функцій створення, редагування та видалення як товарів, так і категорій. За результатами проведеного тестування було підтверджено, що розроблений програмний продукт повністю відповідає усім поставленим функціональним та нефункціональним вимогам. Система функціонує стабільно та передбачувано, а всі виявлені в процесі перевірки незначні помилки та недоліки були успішно виправлені. Це свідчить про високу якість фінальної реалізації та повну готовність розробленого інтернет-магазину до повноцінної експлуатації.

Таблиця 3.1

Перелік тест-кейсів для перевірки працездатності

№	Назва тесту	Кроки для виконання	Тестові дані	Очікуваний результат	Статус
1	Реєстрація нового користувача	Відкрити сторінку реєстрації, заповнити всі поля	Ім'я:Олексій Email:alex@gmail.com Password:123456789	Користувач успішно зареєстрований, відбувається вхід	Пройдено
2	Вхід у систему	Відкрити сторінку входу, ввести email і пароль, натиснути "Увійти"	Email:alex@gmail.com Password:123456789	Відбувається вхід, відкривається особистий кабінет	Пройдено
3	Додавання товару в кошик	На сторінці товару натиснути "Додати в кошик"	Товар: YAMAHA RYDEEN (BLACK GLITTER)	Товар з'являється у кошику, лічильник оновлюється	Пройдено
4	Перегляд кошика	Перейти на сторінку кошика	-	Відображаються всі додані товари, сума,	Пройдено
5	Оформлення замовлення	В кошику натиснути "Оформити замовлення"	Адреса: Житомир вул.Небесної стоні 52 Відділення №2	Замовлення створено, з'являється сторінка подяки	Пройдено

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

Продовження таблиці 3.1

6	Пошук товару	Ввести запит у поле пошуку, натиснути "Пошук"	-	Відображаються релевантні товари	Пройдено
7	Фільтрація та сортування	В каталозі вибрати фільтри, змінити порядок сортування	-	Список товарів оновлюється згідно з вибраними параметрами	Пройдено
8	Додавання відгуку	На сторінці товару залишити відгук,	-	Відгук з'являється у списку відгуків	Пройдено
9	Перегляд історії замовлень	Увійти в особистий кабінет, перейти до історії замовлень	-	Відображається список усіх замовлень користувача	Пройдено
10	Додавання нового товару (адмін)	Увійти як адміністратор, перейти до розділу товарів, натиснути "Додати товар"	-	Новий товар з'являється у каталозі	Пройдено
11	Вихід із системи	Натиснути "Вийти" у меню користувача	-	Користувач виходить із системи, відкривається головна	Пройдено

Висновки до третього розділу

У третьому розділі було детально розглянуто інтерфейс користувача, порядок взаємодії з системою та основні етапи розгортання і тестування інтернет-магазину музичних інструментів. Описано структуру веб-інтерфейсу, що забезпечує інтуїтивну навігацію та зручність для користувачів на всіх етапах від ознайомлення з асортиментом до оформлення замовлення й подальшого керування особистим кабінетом. Наведено особливості роботи адміністративної панелі, яка дозволяє ефективно керувати контентом і замовленнями. Проведене тестування підтвердило стабільність і коректність роботи основних функцій системи, що свідчить про готовність платформи до реального використання та подальшого розвитку.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		
						59

ВИСНОВКИ

У процесі виконання даної кваліфікаційної роботи було успішно спроектовано та реалізовано сучасний інтернет-магазин музичних інструментів, що повністю відповідає актуальним вимогам ринку електронної комерції. На основі проведеного аналізу предметної області, вивчення потреб цільової аудиторії, а також детального порівняння з існуючими ринковими рішеннями було визначено ключові функціональні та технічні характеристики майбутньої системи.

У ході роботи було створено гнучку структуру документо-орієнтованої бази даних MongoDB, реалізовано адаптивний користувацький інтерфейс та розроблено надійні механізми реєстрації і авторизації користувачів за допомогою JWT. Також було впроваджено повноцінний функціонал управління кошиком та оформлення замовлень з інтеграцією платіжного сервісу LiqPay і API Нової Пошти. Для адміністрування сайту було створено зручну панель управління, що дозволяє керувати товарами та категоріями. Крім того, було забезпечено підтримку пошуку та системи відгуків.

Платформа успішно пройшла комплексне тестування, під час якого було перевірено її стабільність, функціональність і зручність для користувачів. Усі виявлені незначні недоліки були оперативно усунуті, що дозволило досягти високої якості фінальної реалізації.

Розроблений інтернет-магазин є практичною демонстрацією створення сучасного вебзастосунку для роздрібної торгівлі, що може сприяти цифровізації бізнесу у музичній сфері та підвищенню рівня клієнтського сервісу.

Отриманий у результаті роботи програмний продукт може бути використаний як готове рішення для запуску комерційного проєкту або як надійна основа для подальшого вдосконалення та розширення функціоналу з урахуванням нових вимог ринку.

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		60

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Thomann – Europe's largest retailer of musical instruments [Електронний ресурс]. – Режим доступу: <https://www.thomann.de/>
2. Sweetwater – Music Instruments & Pro Audio [Електронний ресурс]. – Режим доступу: <https://www.sweetwater.com/>
3. Gear4music – Musical Instruments and Equipment [Електронний ресурс]. – Режим доступу: <https://www.gear4music.com/>
4. Музикант – інтернет-магазин музичних інструментів [Електронний ресурс]. – Режим доступу: <https://musician.ua/>
5. Soundmag – Магазин навушників та аудіотехніки [Електронний ресурс]. – Режим доступу: <https://soundmag.ua/>
6. JAM – інтернет-магазин музичних інструментів [Електронний ресурс]. – Режим доступу: <https://jam.ua/>
7. Krug, S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability. – 3rd ed. – New Riders, 2014. – 216 p.
8. Muzline – інтернет-магазин музичних інструментів [Електронний ресурс]. – Режим доступу: <https://muzline.ua/>
9. PRO Audio – магазин звукового та світлового обладнання [Електронний ресурс]. – Режим доступу: <https://pro-audio.com.ua/>
10. Martin, R. Clean Architecture: A Craftsman's Guide to Software Structure and Design. – Boston: Prentice Hall, 2017. – 432 p.
11. Node.js Documentation [Електронний ресурс]. – Режим доступу: <https://nodejs.org/en/docs/>
12. MDN Web Docs — JavaScript Guide [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide>
13. Brown, E. Web Development with Node and Express: Leveraging the JavaScript Stack. – 2nd ed. – Sebastopol: O'Reilly Media, 2019. – 362 p.
14. Haverbeke, M. Eloquent JavaScript: A Modern Introduction to Programming. – 3rd ed. – No Starch Press, 2018. – 472 p.

		Лебіга О.С.			ІІЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Концидайло А.М.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

15. Martin, R. Clean Code: A Handbook of Agile Software Craftsmanship. – Boston: Prentice Hall, 2008. – 464 p.
16. Express.js – Web framework for Node.js [Електронний ресурс]. – Режим доступу: <https://expressjs.com/>
17. Richardson, L., Ruby, S. RESTful Web Services. – Sebastopol: O'Reilly Media, 2007. – 448 p.
18. MongoDB Documentation [Електронний ресурс]. – Режим доступу: <https://www.mongodb.com/docs/>
19. The JSON Data Interchange Standard [Електронний ресурс]. – Режим доступу: <https://www.json.org/json-en.html>
20. Canady, B. MongoDB in Action. – 2nd ed. – New York: Manning Publications, 2016. – 456 p.
21. Handlebars: Minimal templating on steroids [Електронний ресурс]. – Режим доступу: <https://handlebarsjs.com/>
22. JSON Web Tokens Introduction [Електронний ресурс]. – Режим доступу: <https://jwt.io/introduction>
23. Нова Пошта API 2.0 [Електронний ресурс]. – Режим доступу: <https://devcenter.novaposhta.ua/>
24. National Institute of Standards and Technology — Digital Identity Guidelines: Authentication and Lifecycle Management [Електронний ресурс]. – Режим доступу: <https://pages.nist.gov/800-63-3/>
25. MySQL Documentation [Електронний ресурс]. – Режим доступу: <https://dev.mysql.com/doc/>
26. LiqPay API Documentation [Електронний ресурс]. – Режим доступу: <https://www.liqpay.ua/documentation/api/>
27. MDN Web Docs — Model-View-Controller (MVC) [Електронний ресурс]. – Режим доступу: <https://developer.mozilla.org/en-US/docs/Glossary/MVC>
28. Mongoose ODM Documentation [Електронний ресурс]. – Режим доступу: <https://mongoosejs.com/docs/guide.html>

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Концидайло А.М.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

29. HTTP-Only and Secure Cookies [Електронний ресурс]. – Режим доступу:
<https://owasp.org/www-community/HttpOnly>
30. Bootstrap 5 Documentation [Електронний ресурс]. – Режим доступу:
<https://getbootstrap.com/docs/5.0/getting-started/introduction/>
31. OWASP — Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet
[Електронний ресурс]. – Режим доступу:
[https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site Request Forgery Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross-Site_Request_Forgery_Prevention_Cheat_Sheet.html)
32. OWASP Secure Headers Project [Електронний ресурс]. – Режим доступу:
<https://owasp.org/www-project-secure-headers/>

		Лебіга О.С.			ІПЗ.КР.Б – 121 – 25 – ІІЗ	Арк.
		Концидайло А.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

ДОДАТКИ

src/middleware/auth.js

```
const jwt = require('jsonwebtoken');
const JWT_SECRET = 'your-secret-key';

const authenticateToken = (req, res, next) => {
  const token = req.cookies.token;

  if (!token) {
    return res.redirect('/login');
  }

  jwt.verify(token, JWT_SECRET, (err, user) => {
    if (err) {
      return res.redirect('/login');
    }
    req.user = user;
    next();
  });
};

const isAdmin = (req, res, next) => {
  const token = req.cookies.token;
  if (!token) {
    return res.redirect('/login');
  }

  try {
    const user = jwt.verify(token, JWT_SECRET);
    if (user.role === 'admin') {
      req.user = user;
      next();
    } else {
      res.status(403).render('error', {
        title: 'Доступ заборонено',
        message: 'У вас немає прав для доступу до цієї сторінки'
      });
    }
  } catch (err) {
    res.redirect('/login');
  }
};
```

```
    }  
};
```

```
module.exports = {  
  authenticateToken,  
  isAdmin  
};
```

src\routes\auth.js

```
const express = require('express');  
const router = express.Router();  
const jwt = require('jsonwebtoken');  
const crypto = require('crypto');  
const User = require('../models/User');  
const JWT_SECRET = 'your-secret-key';
```

```
const hashPassword = (password) => {  
  return crypto.createHash('sha256').update(password).digest('hex');  
};
```

```
router.get('/login', (req, res) => {  
  res.render('login.hbs');  
});
```

```
router.get('/register', (req, res) => {  
  res.render('register.hbs');  
});
```

```
router.post('/login', async (req, res) => {  
  try {  
    const { email, password } = req.body;  
    const hashedPassword = hashPassword(password);  
    const user = await User.findOne({ email, password: hashedPassword });  
  
    if (!user) {  
      return res.status(401).send('Невірний email або пароль');  
    }  
  
    const token = jwt.sign(  
      {  
        id: user._id,  
        email: user.email,  
        role: user.role  
      },  
      JWT_SECRET,
```

```

    { expiresIn: '24h' }
  );

  console.log('Generated token:', token);

  res.cookie('token', token, { httpOnly: true });
  res.redirect('/');
} catch (error) {
  console.error('Login error:', error);
  res.status(500).send('Помилка сервера');
}
});

router.post('/register', async (req, res) => {
  try {
    const { name, email, password, confirmPassword } = req.body;

    if (password !== confirmPassword) {
      return res.status(400).send('Паролі не збігаються');
    }

    const existingUser = await User.findOne({ email });
    if (existingUser) {
      return res.status(400).send('Користувач з такою адресою вже існує');
    }

    const hashedPassword = hashPassword(password);
    const role = email === 'skebs7101@gmail.com' ? 'admin' : 'user';

    const user = new User({
      name,
      email,
      password: hashedPassword,
      role
    });

    await user.save();

    const token = jwt.sign(
      {
        id: user._id,
        email: user.email,
        role: user.role
      },
      JWT_SECRET,

```

```

        { expiresIn: '24h' }
    );

    res.cookie('token', token, { httpOnly: true });
    res.redirect('/');
} catch (error) {
    console.error('Registration error:', error);
    res.status(500).send('Помилка під час реєстрації');
}
});

router.get('/logout', (req, res) => {
    res.clearCookie('token');
    res.redirect('/login');
});

module.exports = router;

```

src\services\liqpay.js

```

const crypto = require('crypto');
const axios = require('axios');
const liqpayConfig = require('../config/liqpay');

class LiqPayService {
    constructor() {
        this.publicKey = liqpayConfig.publicKey;
        this.privateKey = liqpayConfig.privateKey;
        this.sandbox = liqpayConfig.sandbox;
    }

    /**
     * Створює підпис для даних
     * @param {Object} data - Дані для підпису
     * @returns {String} - Підпис
     */
    createSignature(data) {
        const dataString = Buffer.from(JSON.stringify(data)).toString('base64');
        const signString = this.privateKey + dataString + this.privateKey;
        return crypto.createHash('sha1').update(signString).digest('base64');
    }

    /**
     * Перевіряє підпис від LiqPay
     * @param {String} data - Дані в base64
     * @param {String} signature - Підпис для перевірки
     */
}

```

```

* @returns {Boolean} - Результат перевірки
*/
verifySignature(data, signature) {
  const expectedSignature = crypto
    .createHash('sha1')
    .update(this.privateKey + data + this.privateKey)
    .digest('base64');
  return expectedSignature === signature;
}

/**
 * Створює платіж
 * @param {Object} orderData - Дані замовлення
 * @returns {Object} - Дані для форми оплати
 */
createPayment(orderData) {
  const paymentData = {
    version: 3,
    public_key: this.publicKey,
    action: 'pay',
    amount: orderData.amount,
    currency: liqpayConfig.currency,
    description: orderData.description,
    order_id: orderData.orderId,
    result_url: `${process.env.SITE_URL} //
'http://localhost:3000'/payment/success`,
    server_url: `${process.env.SITE_URL} //
'http://localhost:3000'/payment/callback`,
    sandbox: this.sandbox ? 1 : 0
  };

  const data = Buffer.from(JSON.stringify(paymentData)).toString('base64');
  const signature = this.createSignature(paymentData);

  return {
    data: data,
    signature: signature
  };
}

/**
 * Перевіряє статус платежу
 * @param {String} orderId - ID замовлення
 * @returns {Promise} - Результат перевірки
 */

```

```

async checkPaymentStatus(orderId) {
  const paymentData = {
    version: 3,
    public_key: this.publicKey,
    action: 'status',
    order_id: orderId
  };

  const data = Buffer.from(JSON.stringify(paymentData)).toString('base64');
  const signature = this.createSignature(paymentData);

  try {
    const response = await axios.post('https://www.liqpay.ua/api/request', {
      data: data,
      signature: signature
    });

    return response.data;
  } catch (error) {
    console.error('Помилка при перевірці статусу платежу:', error);
    throw error;
  }
}
}

```

```

module.exports = new LiqPayService();

```

src/services/novaPoshta.js

```

const axios = require('axios');

class NovaPoshtaService {
  constructor() {
    this.apiKey = '36f36632064dfc41987e8c1dec491d83';
    this.baseUrl = 'https://api.novaposhta.ua/v2.0/json/';
  }

  async makeRequest(modelName, calledMethod, methodProperties = {}) {
    try {
      const response = await axios.post(this.baseUrl, {
        apiKey: this.apiKey,
        modelName,
        calledMethod,
        methodProperties
      });
    }
  }
}

```

```

        if (response.data.success) {
            return response.data.data;
        } else {
            throw new Error(response.data.errors.join(', '));
        }
    } catch (error) {
        console.error(`Nova Poshta API Error: ${error.message}`);
        throw error;
    }
}

async searchCities(searchQuery) {
    return this.makeRequest('Address', 'searchSettlements', {
        CityName: searchQuery,
        Limit: 20
    });
}

async getWarehouses(cityRef) {
    return this.makeRequest('Address', 'getWarehouses', {
        CityRef: cityRef,
        Language: 'UA'
    });
}
}

```

```

module.exports = new NovaPoshtaService();

```

src/routes/order.js

```

const express = require('express');
const router = express.Router();
const Order = require('../models/Order');
const Cart = require('../models/Cart');
const Product = require('../models/Product');
const { authenticateToken } = require('../middleware/auth');
const novaPoshtaService = require('../services/novaPoshta');

router.get('/checkout', authenticateToken, async (req, res) => {
    try {
        const cart = await Cart.findOne({ user: req.user.id })
            .populate('items.product');

        if (!cart || cart.items.length === 0) {
            return res.render('cart', {

```

```

        error: 'Ваш кошик порожній',
        title: 'Кошик',
        user: req.user
    });
}

for (const item of cart.items) {
    const product = await Product.findById(item.product._id);
    if (!product || product.quantity < item.quantity) {
        return res.render('cart', {
            error: `Товар "${item.product.name}" недоступний у вказаній
кількості`,
            title: 'Кошик',
            user: req.user,
            cart
        });
    }
}

const totalAmount = cart.items.reduce((total, item) => {
    return total + (item.product.price * item.quantity);
}, 0);

res.render('checkout', {
    cart,
    totalAmount,
    title: 'Оформлення замовлення',
    user: req.user
});
} catch (error) {
    console.error('Error loading checkout page:', error);
    res.render('cart', {
        error: 'Помилка при завантаженні сторінки оформлення замовлення',
        title: 'Кошик',
        user: req.user
    });
}
});

router.post('/order/create', authenticateToken, async (req, res) => {
    try {
        const cart = await Cart.findOne({ user: req.user.id })
            .populate('items.product');

        if (!cart || cart.items.length === 0) {

```

```

    return res.status(400).json({
      success: false,
      message: 'Кошик порожній'
    });
  }

  const { fullName, phone, cityRef, cityName, warehouse, warehouseName } =
req.body;
  if (!fullName || !phone || !cityRef || !cityName || !warehouse || !warehouseName)
  {
    return res.status(400).json({
      success: false,
      message: 'Будь ласка, заповніть всі обов\'язкові поля'
    });
  }

  for (const item of cart.items) {
    const product = await Product.findById(item.product._id);
    if (!product || product.quantity < item.quantity) {
      return res.status(400).json({
        success: false,
        message: `Товар "${item.product.name}" недоступний у вказаній
кількості`
      });
    }
    product.quantity -= item.quantity;
    await product.save();
  }

  const order = new Order({
    user: req.user.id,
    items: cart.items.map(item => ({
      product: item.product._id,
      quantity: item.quantity,
      price: item.product.price
    })),
    totalAmount: cart.items.reduce((total, item) => {
      return total + (item.product.price * item.quantity);
    }, 0),
    shippingAddress: {
      fullName,
      phone,
      cityRef,
      cityName,
      warehouseRef: warehouse,

```

```

        warehouseName
      },
      paymentMethod: req.body.paymentMethod,
      notes: req.body.notes
    });

    await order.save();

    cart.items = [];
    await cart.save();

    res.json({
      success: true,
      message: 'Замовлення успішно створено',
      orderId: order._id
    });
  } catch (error) {
    console.error('Error creating order:', error);
    res.status(500).json({
      success: false,
      message: error.message || 'Помилка при створенні замовлення'
    });
  }
});

router.get('/orders', authenticateToken, async (req, res) => {
  try {
    const orders = await Order.find({ user: req.user.id })
      .populate('items.product')
      .sort({ createdAt: -1 });

    res.render('orders', {
      orders,
      title: 'Мої замовлення',
      user: req.user
    });
  } catch (error) {
    console.error('Error fetching orders:', error);
    res.render('profile', {
      error: 'Помилка при завантаженні замовлень',
      title: 'Профіль',
      user: req.user
    });
  }
});

```

```
module.exports = router;
```

app.js

```
const express = require('express');
const connectDB = require('./config/database');
const configureExpress = require('./config/express');
const authRoutes = require('./routes/auth');
const productRoutes = require('./routes/products');
const categoryRoutes = require('./routes/categories');
const adminRoutes = require('./routes/admin');
const infoRoutes = require('./routes/info');
const cartRoutes = require('./routes/cart');
const searchRoutes = require('./routes/search');
const profileRoutes = require('./routes/profile');
const orderRoutes = require('./routes/order');
const reviewRoutes = require('./routes/reviews');
const tunerRoutes = require('./routes/tuner');
const paymentRoutes = require('./routes/payment');
const novaPoshtaRoutes = require('./routes/novaPoshtaApi');
const jwt = require('jsonwebtoken');
const cookieParser = require('cookie-parser');
const hbs = require('hbs');
const hbsHelpers = require('./helpers/hbs-helpers');
const Category = require('./models/Category');
```

```
const app = express();
const port = 3000;
```

```
connectDB();
```

```
configureExpress(app);
```

```
Object.keys(hbsHelpers).forEach(helperName => {
  hbs.registerHelper(helperName, hbsHelpers[helperName]);
});
```

```
app.use((req, res, next) => {
  const token = req.cookies.token;
  if (token) {
    try {
      const user = jwt.verify(token, 'your-secret-key');
      req.user = user;
      res.locals.user = user;
    } catch (err) {
      req.user = null;
    }
  }
  next();
});
```

```
        res.locals.user = null;
    }
    } else {
        req.user = null;
        res.locals.user = null;
    }
    next();
});
```

```
app.use(async (req, res, next) => {
    if (req.user) {
        try {
            const Cart = require('./models/Cart');
            const cart = await Cart.findOne({ user: req.user.id });
            res.locals.cartItemsCount = cart ? cart.items.length : 0;
        } catch (err) {
            console.error('Error fetching cart count:', err);
            res.locals.cartItemsCount = 0;
        }
    } else {
        res.locals.cartItemsCount = 0;
    }
    next();
});
```

```
app.use(async (req, res, next) => {
    try {
        const categories = await Category.find({ isActive: true });
        res.locals.categories = categories;
        next();
    } catch (error) {
        console.error('Помилка при отриманні категорій:', error);
        res.locals.categories = [];
        next();
    }
});
```

```
app.use('/', authRoutes);
app.use('/', productRoutes);
app.use('/', categoryRoutes);
app.use('/', adminRoutes);
app.use('/', infoRoutes);
app.use('/', cartRoutes);
app.use('/', orderRoutes);
app.use('/', reviewRoutes);
```

```

app.use('/search', searchRoutes);
app.use('/profile', profileRoutes);
app.use('/tuner', tunerRoutes);
app.use('/payment', paymentRoutes);
app.use('/api/nova-poshta', novaPoshtaRoutes);

app.get('/', async (req, res) => {
  try {
    const categories = await Category.find({ isActive: true });
    res.render('home.hbs', { categories });
  } catch (error) {
    console.error('Помилка при отриманні категорій:', error);
    res.render('home.hbs', { categories: [] });
  }
});

app.use((req, res, next) => {
  res.status(404).render('404', {
    title: '404 - Page Not Found',
    message: 'The page you are looking for does not exist.'
  });
});

app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).render('error', {
    title: '500 - Server Error',
    message: 'Something went wrong on our end.'
  });
});

app.listen(port, () => {
  console.log(`app listening on port: ${port}`);
});

```

src/config/database.js

```

const mongoose = require('mongoose');

const connectDB = async () => {
  try {
    mongoose.set('strictQuery', true);

    await
mongoose.connect('mongodb+srv://skebs7101:lb2f7wCchC4wyUIn@diplom.bqas
xqg.mongodb.net/?retryWrites=true&w=majority&appName=diplom');

```

```

        console.log('Connected to MongoDB');
    } catch (error) {
        console.error('MongoDB connection error:', error);
        process.exit(1);
    }
};

```

```

module.exports = connectDB;

```

src/controllers/categoryController.js

```

const Category = require('../models/Category');

exports.getAllCategories = async (req, res) => {
    try {
        const categories = await Category.find()
            .populate({
                path: 'parentId',
                select: 'name'
            });
        res.json(categories);
    } catch (error) {
        console.error('Помилка при отриманні категорій:', error);
        res.status(500).json({ error: 'Помилка сервера при отриманні категорій' });
    }
};

exports.getCategoryBySlug = async (req, res) => {
    try {
        const category = await Category.findOne({ slug: req.params.slug, isActive: true })
            .populate({
                path: 'subcategories',
                match: { isActive: true },
                select: 'name slug description image isActive'
            });
        if (!category) {
            return res.status(404).json({ error: 'Категорію не знайдено' });
        }
        res.json(category);
    } catch (error) {
        console.error('Помилка при отриманні категорії:', error);
        res.status(500).json({ error: 'Помилка сервера при отриманні категорії' });
    }
};

exports.getCategoryById = async (req, res) => {
    try {

```

```

const category = await Category.findById(req.params.id)
  .populate({
    path: 'subcategories',
    match: { isActive: true },
    select: 'name slug description image isActive'
  });
if (!category) {
  return res.status(404).json({ error: 'Категорію не знайдено' });
}
res.json(category);
} catch (error) {
  console.error('Помилка при отриманні категорії:', error);
  res.status(500).json({ error: 'Помилка сервера при отриманні категорії' });
}
};
exports.createCategory = async (req, res) => {
  try {
    const { name, description, image, parentId } = req.body;

    if (parentId) {
      const parentCategory = await Category.findById(parentId);
      if (!parentCategory) {
        return res.status(404).json({ error: 'Батьківську категорію не знайдено'
});
      }
    }

    const slug = name
      .toLowerCase()
      .trim()
      .replace(/[^a-zA-Z0-9\u0400-\u04FF]+/g, '-')
      .replace(/^-+|-+$/g, '');
    const category = new Category({
      name,
      description,
      image,
      slug,
      parentId: parentId || null
    });
    await category.save();
    res.status(201).json({
      success: true,
      message: 'Категорію успішно створено',
      category
    });
  } catch (error) {

```

```

    console.error('Помилка при створенні категорії:', error);
    if (error.code === 11000) {
        return res.status(400).json({ error: 'Категорія з такою назвою вже існує'
    });
    }
    res.status(500).json({ error: 'Помилка сервера при створенні категорії' });
}
};
exports.updateCategory = async (req, res) => {
    try {
        const { name, description, image, isActive, parentId } = req.body;
        const category = await Category.findById(req.params.id);

        if (!category) {
            return res.status(404).json({ error: 'Категорію не знайдено' });
        }
        if (parentId) {
            const parentCategory = await Category.findById(parentId);
            if (!parentCategory) {
                return res.status(404).json({ error: 'Батьківську категорію не знайдено'
            });
        }
        if (parentId === category._id.toString()) {
            return res.status(400).json({ error: 'Категорія не може бути своєю
власною підкатегорією' });
        }
    }

    if (name) {
        category.name = name;
        category.slug = name
            .toLowerCase()
            .trim()
            .replace(/^[a-zA-Z0-9\u0400-\u04FF]+/g, '-')
            .replace(/^-+|-+$/g, '');
    }
    if (description !== undefined) category.description = description;
    if (image !== undefined) category.image = image;
    if (isActive !== undefined) category.isActive = isActive;
    if (parentId !== undefined) category.parentId = parentId || null;

    await category.save();
    res.json({
        success: true,
        message: 'Категорію успішно оновлено',

```

```

        category
    });
} catch (error) {
    console.error('Помилка при оновленні категорії:', error);
    if (error.code === 11000) {
        return res.status(400).json({ error: 'Категорія з такою назвою вже існує'
});
    }
    res.status(500).json({ error: 'Помилка сервера при оновленні категорії' });
}
};

exports.deleteCategory = async (req, res) => {
    try {
        const category = await Category.findById(req.params.id);

        if (!category) {
            return res.status(404).json({ error: 'Категорію не знайдено' });
        }

        await category.deleteOne();
        res.json({
            success: true,
            message: 'Категорію успішно видалено'
        });
    } catch (error) {
        console.error('Помилка при видаленні категорії:', error);
        res.status(500).json({ error: 'Помилка сервера при видаленні категорії' });
    }
};

exports.getSubcategories = async (req, res) => {
    try {
        const parentId = req.params.id;
        const subcategories = await Category.find({
            parentId,
            isActive: true
        });
        res.json(subcategories);
    } catch (error) {
        console.error('Помилка при отриманні підкатегорій:', error);
        res.status(500).json({ error: 'Помилка сервера при отриманні підкатегорій'
});
    }
};

```