ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Вебсайт інтернет-магазину музичних інструментів
Назва теми

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»
Шифр і найменування

Спеціальність 121 «Інженерія програмного забезпечення»
Код і найменування

Освітня програма «Інженерія програмного забезпечення»
Найменування

Шифр КвРІПЗ.2201110.01.15.ПЗ

Виконав здобувач IV курсу група ІПЗ-22-1 Андрій СТЕФАНОВСЬКИЙ
Шифр Підпис Ім'я, ПРІЗВИЩЕ

Керівник канд. техн. наук, доцент Юрій ФОРКУН
Науковий ступінь, учене звання Підпис Ім'я, ПРІЗВИЩЕ

Нормоконтролер кант. техн. наук, доцент Оксана ЯШИНА
Підпис Ім'я, ПРІЗВИЩЕ

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення Леонід БЕДРАТЮК
Підпис Ім'я, ПРІЗВИЩЕ

9 червня 2026
Дата

Хмельницький, 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Інженерії програмного забезпечення

Рівень вищої освіти Перший (бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
02.01 2026 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кутю Владиславу Олександровичу
Прізвище, ім'я, по батькові студента

1. Тема роботи Вебсайт інтернет-магазину музичних інструментів

Керівник проекту Форкун Юрій Вікторович к. техн. наук, доцент
Прізвище, ім'я, по батькові науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2026 р. №0-КП

2. Строк подання студентом проекту на кафедру 15.05.2026 р.

3. Вихідні дані до роботи: Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Аналіз предметної області та постановка задачі

Проєктування програмного забезпечення

Програмна реалізація та тестування програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)
Три креслення у форматі А3:
1. Діаграма класів
2. DFD-діаграма
3. Діаграма процесів

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Яшина М. О., доцент	05.05.26	07.05.26
Антиплагіат	Форкун Ю. В., доцент	05.05.26	05.06.26

7. Дата видачі завдання « 02 » січня 2026 р.

Календарний графік

виконання кваліфікаційної роботи студентом гр. ІПЗ-22-1 за спеціальністю 121 «Інженерія програмного забезпечення»

№ з/п	Назва етапу виконання	Дата початку-завершення етапу	Примітка
1	Ознайомлення з тематикою дипломного проєктування, визначення та узгодження індивідуальних тем кваліфікаційних робіт (КвР)	01.12 - 31.12.2025	
2	Збір матеріалу за темою КвР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	01.01 - 20.02.2026	
3	Проєктування програмного забезпечення	21.02 - 20.03.2026	
4	Програмна реалізація з використанням відповідних засобів розробки. Тестування ПЗ	21.03 - 30.04.2026	
5	Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КвР згідно вимог	01.05 - 25.05.2026	
6	Попередній захист КвР	15.05.2026	
7	перевірка КвР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 - 30.06.2026	
8	Здача КвР на кафедру; підготовка КвР для розміщення у репозиторій ХНУ; підготовка до захисту та захист КвР	з 01.06.2026	

Студент _____ Підпис Стефановський Андрій Ім'я, ПРІЗВИЩЕ

Керівник проєкту (роботи) _____ Підпис Юрій ФОРКУН Ім'я, ПРІЗВИЩЕ

ВІДОМІСТЬ ДОКУМЕТІВ

№ рядка		Формат	Позначення документа	Найменювання документа	К-ть аркушів	№ екз.	Примітка
				Текстові документи			
1		A4	КвРІПЗ.2201110.01.15.ПЗ	Пояснювальна записка	67		
2		A4		Завдання на кваліфіка-	1		
				ційну роботу			
3		A4		Анотація	1		
				Графічні частина			
4		A3	КвРІПЗ.2201110.01.15.ПЗ	UML-діаграма варіантів використання	1		
5		A3	КвРІПЗ.2201110.01.15.ПЗ	ER-діаграм бази даних	1		
6		A3	КвРІПЗ.2201110.01.15.ПЗ	Діаграма стркутури вебсайту інтернет магазину	1		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.2201110.01.15.ПЗ		
Розроб.		Стефановський А.А		28.05	Вебсайт інтернет-магазину з продажу музичних інструментів	Літ.	Арк. 5 / Акрушів 55
Перевір.		Форкун Ю.В.		28.05			
Реценз.						ХНУ. ІПЗ-22-1	
Н. Контр.		Яшина О. М.		25.05			
Затверд.		Бедратюк Л. П.		25.05			

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Вебсайт інтернет-магазину з продажу музичних інструментів».

Автор роботи:Стефановський Андрій Анатолійович.

Керівник роботи: Форкун Юрій Вікторович

Пояснювальна записка: 74 с., 43 рис., 1 табл., 3 дод., 40 джерел.

Графічна частина: 3 креслення.

Мета кваліфікаційної роботи полягає у розробці вебзастосунку для продажу музичних інструментів із можливістю перегляду товарів, роботи з кошиком та оформлення замовлень.

У роботі проведено аналіз предметної області, визначено вимоги до системи та розроблено структуру вебзастосунку і базу даних. Для реалізації проєкту використано технології HTML, CSS, JavaScript, Node.js, Express.js та MySQL.

У результаті створено вебзастосунок інтернет-магазину музичних інструментів, який забезпечує перегляд каталогу, авторизацію користувачів, роботу з кошиком та оформлення замовлень. Проведене тестування підтвердило коректну роботу основних функцій системи.

28.05

Дата

Підпис

ЗМІСТ

Вступ.....7
1 Дослідження предметної області та постановка задачі.....9
1.1 Аналіз предметної області, її стурктурних та функціональних особливостей.....9
1.2 Аналіз останніх публікацій, досліджень та існуючих рішень предметної області.....
14
1.3 Аналіз вимог до програмного забезпечення.....18
1.4 Висновки. Постановка задачі.....23
2. Проєктування структури та компонентів програмного забезпечення.....25
2.1 Проєктування архітектури програмного забезпечення.....25
2.2 Проектування модулів.....28
2.3 Проєктування інтерфейсу користувача.....32
2.4 Аналіз технологій і методів реалізації ПЗ.....34
2.5 Проєктування користувацького інтерфейсу.....37
3. Програмна реалізація та тестування програмного забезпечення.....40
3.1 Програмна реалізація модулів.....40
3.2 Тестування програмного забезпечення.....47
3.3 Висновки по реалізації та тестуванні ПЗ.....52
ВИСНОВКИ.....57
Перелік посилань.....60
Додаток А.....65
Додаток Б.....71
Додаток В.....96

					КвРІПЗ.2201110.01.15.ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Стефановський А.А.		25.05	Вебсайт інтернет-магазину з продажу музичних інструментів	Літ.	Арк.	Акрушів
Перевір.		Форкун.Ю.В.		25.05			5	55
Реценз.						ХНУ. ІПЗ-22-1		
Н. Контр.		Яшина О. М.		25.05	Пояснювальна записка			
Затверд.		Бедратюк Л. П.		25.05				

ВСТУП

Сучасний етап розвитку інформаційних технологій характеризується стрімким зростанням ролі мережі Інтернет у всіх сферах д іяльності людини. Однією з найбільш динамічних галузей є електронна комерція, яка дозволяє здійснювати купівлю та продаж товарів і послуг у онлайн-режимі. Використання вебтехнологій у сфері торгівлі сприяє спрощенню процесу взаємодії між продавцем і покупцем, розширенню ринку збуту та підвищенню ефективності ведення бізнесу.

Інтернет-магазини стали невід'ємною частиною сучасної економіки. Вони забезпечують користувачам можливість швидко знаходити необхідні товари, порівнювати їх характеристики та здійснювати покупки без необхідності фізичного відвідування магазину. Особливо актуальним це є в умовах зростання популярності мобільних пристроїв та розвитку цифрових сервісів.

Окрему нішу в електронній комерції займають інтернет-магазини музичних інструментів. Такі системи мають певні особливості, пов'язані зі специфікою товарів. Музичні інструменти характеризуються великою кількістю технічних параметрів, що вимагає надання детальної інформації про продукцію. Крім того, користувач не має можливості безпосередньо ознайомитися з товаром перед покупкою, тому важливу роль відіграє якість опису, зображення та зручність інтерфейсу.

Актуальність даної роботи полягає у необхідності створення сучасного вебзастосунку інтернет-магазину музичних інструментів, який забезпечить зручну взаємодію користувача із системою, швидкий доступ до інформації про товари та можливість оформлення замовлення в онлайн-режимі.

Метою кваліфікаційної роботи є розробка вебсайту інтернет-магазину музичних інструментів, який забезпечує перегляд каталогу товарів, пошук і фільтрацію, додавання товарів до кошика та оформлення замовлення.

Для досягнення поставленої мети необхідно вирішити такі задачі:

– провести аналіз предметної області електронної комерції та інтернет-магазинів;

– Для досягнення поставленої мети необхідно вирішити такі задачі:

– провести аналіз предметної області електронної комерції та визначити особливості функціонування інтернет-магазинів музичних інструментів;

– дослідити існуючі програмні рішення у сфері онлайн-торгівлі та визначити їх переваги і недоліки;

–сформулювати функціональні та нефункціональні вимоги до вебзастосунку;

– розробити архітектуру програмного забезпечення та визначити основні компоненти системи;

– спроєктувати структуру бази даних для зберігання інформації про товари, користувачів і замовлення;

– реалізувати клієнтську та серверну частини вебсайту інтернет-магазину;

– забезпечити взаємодію між клієнтською частиною, сервером та базою даних;

– провести тестування розробленого програмного забезпечення та перевірити його відповідність поставленим вимогам.

Об’єктом дослідження є процес функціонування інтернет-магазину музичних інструментів.

Предметом дослідження є методи та засоби розробки вебзастосунків для електронної комерції.

Практичне значення отриманих результатів полягає у створенні вебсайту, який може бути використаний як основа для подальшого розвитку інтернет-магазину музичних інструментів та впровадження його у реальне середовище

1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної області, її структурних та функціональних особливостей

Сфера електронної комерції є одним із ключових напрямів розвитку сучасних інформаційних технологій та цифрової економіки. Вона охоплює процеси купівлі та продажу товарів і послуг через мережу Інтернет, що дозволяє значно спростити взаємодію між продавцем і покупцем, а також автоматизувати значну частину бізнес-процесів. Завдяки розвитку вебтехнологій інтернет-магазини стали невід’ємною частиною сучасної торгівлі та одним із найбільш ефективних способів реалізації продукції.

За останні роки онлайн-магазини отримали значне поширення серед користувачів завдяки зручності використання, швидкому доступу до товарів та можливості здійснювати покупки без необхідності фізичного відвідування магазину. Особливо актуальним це стало в умовах активного розвитку мобільних технологій, оскільки значна частина користувачів здійснює покупки саме за допомогою смартфонів або планшетів.

Інтернет-магазини музичних інструментів займають окрему нішу в електронній комерції. Це пов’язано з тим, що дана категорія товарів має специфічні характеристики, які не завжди можна оцінити лише за зовнішнім виглядом. При виборі музичного інструмента користувачі звертають увагу на бренд, технічні характеристики, матеріал виготовлення, тип інструмента, рівень професійності та інші параметри. Наприклад, для гітар важливими характеристиками є тип корпусу, матеріал, кількість струн та тип звукознімача, а для клавішних інструментів — кількість клавіш, тип механіки та функціональні можливості.

У зв’язку з цим вебсайти даної категорії повинні надавати максимально детальну інформацію про товари, включаючи характеристики, якісні зображення,

описи, відеоогляди та відгуки користувачів. Це дозволяє частково компенсувати відсутність фізичного контакту з товаром перед покупкою та допомагає користувачу зробити правильний вибір.

Для кращого розуміння процесу розвитку систем електронної комерції доцільно розглянути основні етапи еволюції інтернет-магазинів. З розвитком вебтехнологій змінювались не лише підходи до створення сайтів, а й функціональні можливості систем онлайн-продажу. Від простих статичних вебсторінок інтернет-магазини поступово перейшли до багатофункціональних платформ із підтримкою інтерактивної взаємодії, систем пошуку, онлайн-оплати та персоналізованих рекомендацій.

Рисунок 1.1 – Етапи розвитку інтернет-магазинів

Як показано на рисунку 1.1, розвиток інтернет-магазинів супроводжувався ускладненням структури систем та збільшенням кількості функціональних можливостей. Сучасні вебзастосунки вже не є простими каталогами товарів, а являють собою комплексні інформаційні системи, які забезпечують повний цикл взаємодії між користувачем і магазином.

Сучасний інтернет-магазин музичних інструментів являє собою складну інформаційну систему, яка складається з декількох взаємопов’язаних компонентів. Основними з них є користувацький інтерфейс, каталог товарів, система фільтрації та пошуку, кошик покупця, модуль оформлення замовлення, система авторизації користувачів, а також адміністративна частина для управління товарами та замовленнями. Схему взаємодії основних модулів системи наведено на рисунку 1.2.

Рисунок 1.2 – Загальна структура інтернет-магазину

Важливою складовою таких систем є зручність взаємодії користувача з інтерфейсом. Якщо навігація по сайту є складною або нелогічною, це може негативно вплинути на досвід користувача та знизити ймовірність здійснення покупки. Саме тому при розробці вебзастосунку необхідно забезпечити просту та зрозумілу структуру сторінок, швидкий доступ до основних функцій та адаптивний дизайн.

Окрему роль відіграє система пошуку та фільтрації товарів. У випадку інтернет-магазину музичних інструментів це є особливо важливим, оскільки кількість товарних позицій може бути значною, а їх характеристики —

різноманітними. Тому користувач повинен мати можливість швидко знаходити потрібний інструмент за заданими параметрами, такими як бренд, тип інструмента, цінова категорія або технічні характеристики.

Крім того, сучасні вебсайти повинні бути адаптивними, тобто коректно відображатися на різних пристроях, включаючи персональні комп'ютери, планшети та смартфони. Це є важливим фактором, оскільки значна частина користувачів здійснює покупки саме з мобільних пристроїв.

Для підвищення конкурентоспроможності вебзастосунку доцільним є впровадження додаткових інтелектуальних функцій. Однією з таких можливостей може бути система рекомендацій музичних інструментів залежно від рівня підготовки користувача. Наприклад, початківцям система може пропонувати базові моделі інструментів із нижчою вартістю та спрощеними характеристиками, тоді як досвідченим музикантам професійне обладнання.

Для покращення взаємодії користувача із системою у вебзастосунку може бути реалізований механізм рекомендацій музичних інструментів залежно від рівня підготовки користувача. Під час реєстрації або налаштування профілю користувач може вказати власний рівень володіння музичними інструментами (початковий, середній або професійний). На основі цієї інформації система здатна формувати персоналізовані рекомендації товарів.

Крім того, перспективною функцією є реалізація системи порівняння товарів, що дозволить одночасно переглядати характеристики декількох музичних інструментів. Це значно спрощує процес вибору продукції та покращує взаємодію користувача із системою.

Також сучасні вебсайти можуть використовувати блоки персоналізованих рекомендацій, які формуються на основі переглянутих товарів або популярних категорій. Це дозволяє підвищити зацікавленість користувача та ефективність продажів.

Не менш важливим аспектом є безпека системи. Оскільки вебзастосунок працює з персональними даними користувачів, необхідно забезпечити їх захист та

безпечну передачу інформації між клієнтом і сервером. Для цього можуть використовуватись сучасні методи шифрування даних та системи авторизації користувачів.

Таким чином, предметна область інтернет-магазину музичних інструментів включає не лише процеси онлайн-продажу, а й комплекс питань, пов’язаних із зручністю користувача, структурою даних, організацією каталогу, безпекою інформації та ефективною взаємодією між усіма компонентами системи. Створення сучасного вебзастосунку для продажу музичних інструментів є актуальним завданням, оскільки дозволяє забезпечити ефективну взаємодію між магазином та клієнтом, автоматизувати процес продажу та покращити якість обслуговування користувачів.

Розвиток електронної комерції призвів до появи великої кількості готових програмних рішень та платформ для створення інтернет-магазинів. Сучасні вебзастосунки відрізняються між собою архітектурою, функціональними можливостями, технологіями реалізації та рівнем адаптації до потреб користувачів.

У зв’язку з цим доцільним є проведення аналізу існуючих програмних рішеньта сучасних підходів до розробки інтернет-магазинів. Це дозволяє визначитиосновні переваги та недоліки існуючих систем, а також сформувати вимоги домайбутнього програмного забезпечення.

Слід зазначити, що розвиток інтернет-магазинів безпосередньо пов’язаний ізрозвитком вебтехнологій та збільшенням кількості користувачів мережі Інтернет. З кожним роком вебзастосунки стають більш функціональними, швидкими таорієнтованими на потреби користувачів. Це призводить до постійного вдосконалення підходів до створення систем електронної комерції.

Крім стандартного функціоналу сучасні інтернет-магазини активно використовують різноманітні додаткові сервіси, зокрема системи рекомендацій персоналізований контент, інтеграцію з платіжними системами та соціальними

мережами. Такі можливості дозволяють покращити взаємодію користувача із системою та підвищити ефективність онлайн-продажів.

1.2 Аналіз останнії публікацій, досліджень та існуюючих рішень предметної області

На сучасному етапі розвитку інформаційних технологій електронна комерція займає провідне місце серед способів реалізації товарів та послуг. Інтернет-магазини стали основним інструментом ведення онлайн-бізнесу, що зумовлює появу великої кількості програмних рішень для їх створення та підтримки.

Існуючі рішення у сфері розробки інтернет-магазинів можна умовно класифікувати за кількома ознаками: способом реалізації, рівнем складності, гнучкістю налаштування та способом розгортання. Найбільш поширеним є поділ на готові платформи та індивідуально розроблені вебзастосунки.

До готових платформ належать системи, що надають користувачу можливість створити інтернет-магазин із використанням вже готових інструментів. До таких рішень можна віднести Shopify, WooCommerce, OpenCart, Magento та інші. Вони забезпечують базовий функціонал, необхідний для роботи магазину: управління товарами, кошик, оформлення замовлення, інтеграцію з платіжними системами та службами доставки.

Основною перевагою таких платформ є швидкість впровадження. Користувач може створити інтернет-магазин за відносно короткий час без необхідності глибокого вивчення програмування. Крім того, більшість платформ мають велику кількість готових шаблонів дизайну та додаткових модулів, що дозволяє розширювати функціональність системи.

Разом з тим, готові платформи мають і певні обмеження. Зокрема, вони не завжди дозволяють реалізувати специфічні вимоги до системи, що особливо важливо для вузькоспеціалізованих інтернет-магазинів, таких як магазини

музичних інструментів. Обмеження можуть стосуватися як логіки роботи системи, так і структури бази даних або інтерфейсу користувача.

Ще одним недоліком є залежність від сторонньої платформи. Оновлення системи, зміни в політиці використання або технічні обмеження можуть впливати на роботу інтернет-магазину. Крім того, використання великої кількості плагінів може негативно позначатися на продуктивності системи.

Альтернативним підходом є індивідуальна розробка вебсайту. У цьому випадку система створюється з урахуванням конкретних вимог, що дозволяє повністю контролювати її функціональність, структуру та зовнішній вигляд. Такий підхід є більш складним, але водночас більш гнучким.

Індивідуальна розробка дозволяє оптимізувати систему під конкретні задачі, зменшити надлишковий функціонал та забезпечити кращу продуктивність. Крім того, розробник має можливість самостійно визначати архітектуру системи та використовувати сучасні технології для її реалізації.

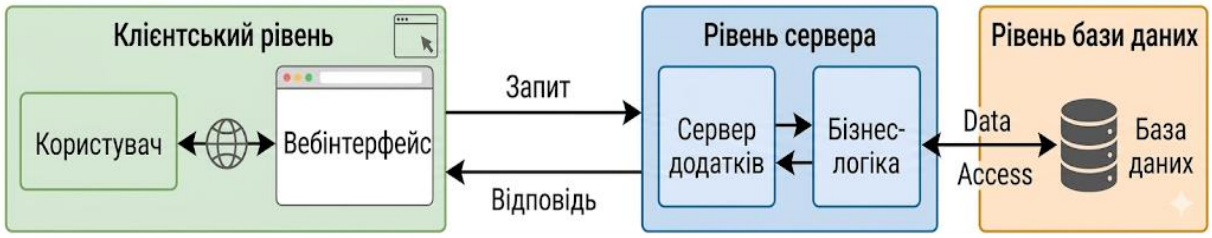


Рисунок 1.3 = Типова архітектура інтернет магазину

Залежно від підходу до побудови системи, архітектура інтернет-магазину може бути як монолітною, так і розподіленою. У монолітних системах усі компоненти об’єднані в одному застосунку, що спрощує розробку, але ускладнює масштабування. У більш складних рішеннях можуть використовуватися мікросервісні підходи, що дозволяють розділити систему на окремі модулі.

Особливістю інтернет-магазинів музичних інструментів є підвищені вимоги до структури каталогу товарів. Такі товари мають велику кількість характеристик, які необхідно зберігати та відображати. Наприклад, для гітар це може бути тип

корпусу, кількість струн, матеріал, виробник; для клавішних — кількість клавіш, тип механіки, функціональні можливості.

У зв’язку з цим важливу роль відіграє правильна організація бази даних. Вона повинна забезпечувати швидкий доступ до інформації, підтримувати фільтрацію та сортування товарів, а також дозволяти легко розширювати структуру при додаванні нових категорій.

Крім того, важливим елементом є система пошуку, яка дозволяє користувачеві швидко знайти необхідний товар. Ефективна реалізація пошуку повинна враховувати можливі помилки введення, варіанти написання назв та інші особливості.

Окрему увагу слід приділити користувацькому інтерфейсу. Для інтернет-магазину музичних інструментів важливо, щоб інтерфейс був зрозумілим як для початківців, так і для досвідчених музикантів. Це означає, що інформація повинна бути структурована та легко доступна.

Також сучасні інтернет-магазини активно використовують додаткові функції, такі як:

- рекомендації товарів на основі переглядів;
- система відгуків і рейтингів;
- порівняння товарів;
- інтеграція з соціальними мережами.

Ці функції дозволяють підвищити зручність користування сайтом та збільшити ймовірність здійснення покупки.

Важливим фактором є і продуктивність системи. При великій кількості товарів та користувачів необхідно забезпечити стабільну роботу сайту та швидке завантаження сторінок. Це досягається за рахунок оптимізації запитів до бази даних, кешування та використання сучасних технологій розробки.

Таким чином, аналіз існуючих рішень показує, що універсального підходу до створення інтернет-магазину не існує. Кожен із варіантів має свої переваги та недоліки, а вибір конкретного рішення залежить від поставлених задач.

З урахуванням особливостей предметної області, доцільним є створення індивідуального вебсайту інтернет-магазину музичних інструментів, який буде враховувати специфіку товарів, забезпечувати зручність користування та відповідати сучасним вимогам до вебзастосунків.

Для кращого розуміння особливостей сучасних інтернет-магазинів музичних інструментів доцільно розглянути приклади існуючих вебресурсів, які використовуються для онлайн-продажу музичного обладнання.

Одним із популярних рішень є вебсайт Muzline.ua, який спеціалізується напродажі музичних інструментів та аксесуарів. Система має зручний каталог товарів, поділ продукції на категорії та систему пошуку. Користувач може переглядати характеристики товарів, зображення та опис продукції.

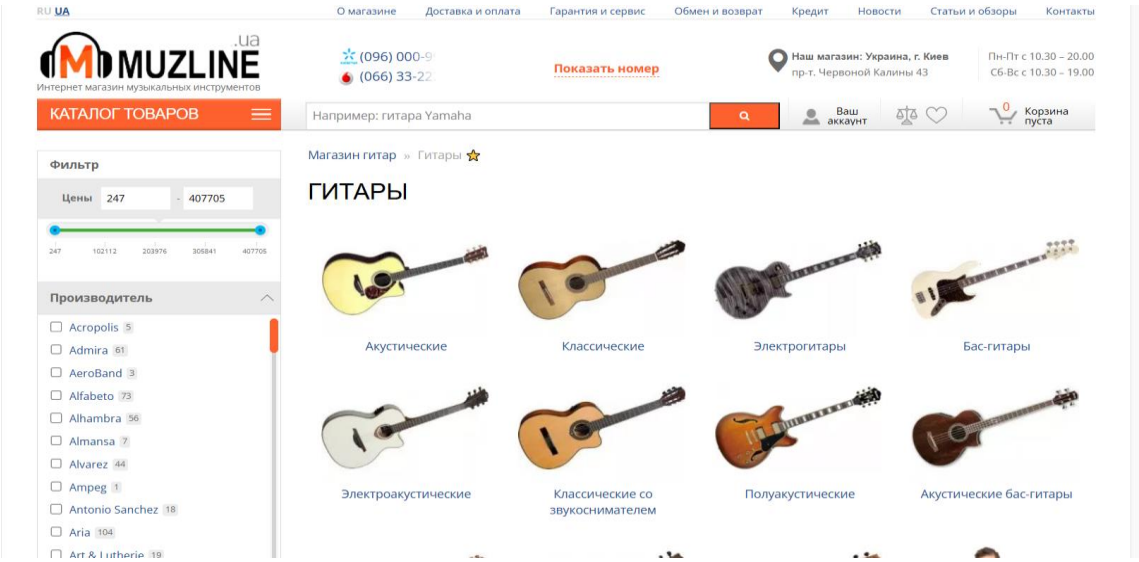


Рисунок 1.4 – Інтерфейс сайту Muzline.ua

Ще одним прикладом є вебсайт JAM, який також орієнтований на продаж музичних інструментів та професійного обладнання. Особливістю даного вебресурсу є сучасний дизайн, адаптивний інтерфейс та підтримка системи фільтрації товарів за різними параметрами.

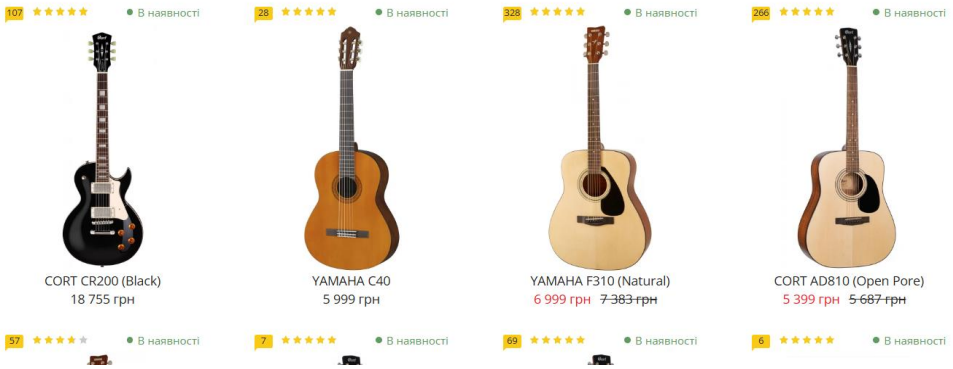


Рисунок 1.5 – Каталог музичних інструментів сайту JAM

Також доцільно розглянути розділ музичних інструментів інтернет-магазину Rozetka. Даний ресурс забезпечує швидкий пошук товарів, зручну навігацію та можливість оформлення онлайн-замовлення. Крім того, користувач має доступ до відгуків, рейтингу товарів та додаткової інформації про продукцію.

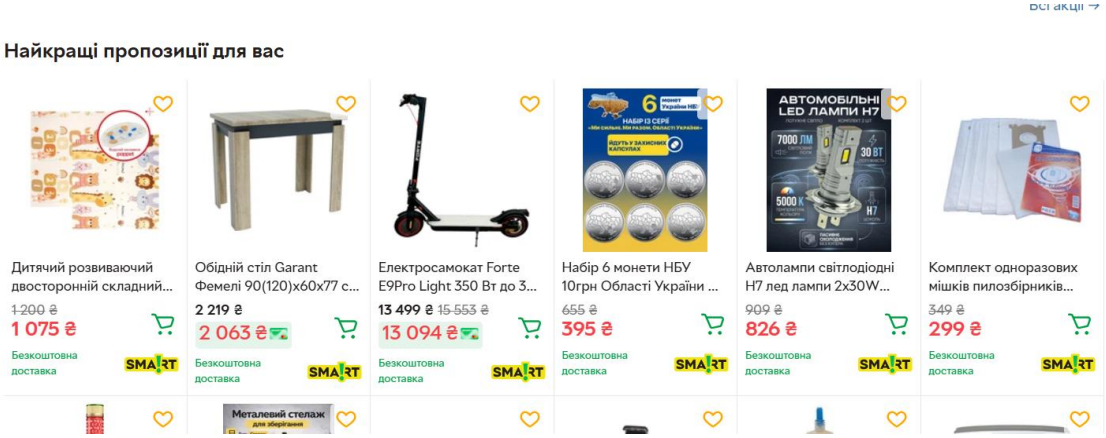


Рисунок 1.6 – Приклад структури товарів Rozetka

Проведений аналіз існуючих вебресурсів дозволив визначити основні функціональні можливості, які доцільно реалізувати у власному вебзастосунку. До них належать система пошуку та фільтрації товарів, адаптивний інтерфейс, каталог продукції, кошик користувача та система оформлення замовлення.

1.3 Аналіз вимог до програмного забезпення

Після проведеного аналізу предметної області та існуючих рішень необхідно сформулювати вимоги до програмного забезпечення інтернет-магазину музичних інструментів. Чітке визначення вимог є важливим етапом розробки, оскільки воно дозволяє визначити основні функції системи, її структуру та особливості реалізації.

Вимоги до програмного забезпечення доцільно поділити на функціональні та нефункціональні.

До функціональних вимог належать можливості, які система повинна забезпечувати для користувача. Насамперед користувач повинен мати змогу переглядати каталог товарів, який містить різні категорії музичних інструментів, такі як гітари, клавішні інструменти, ударні установки та аксесуари. Каталог повинен бути структурований таким чином, щоб користувач міг легко орієнтуватися у великій кількості товарів.

Важливою функцією є пошук товарів за ключовими словами. Це дозволяє значно скоротити час на знаходження необхідного інструмента. Додатково повинна бути реалізована можливість фільтрації товарів за різними параметрами, зокрема за ціною, брендом, типом інструмента або іншими характеристиками.

Користувач також повинен мати доступ до детальної інформації про кожен товар. Така інформація включає опис, технічні характеристики, зображення та, за можливості, відгуки інших покупців. Це є особливо важливим для інтернет-магазинів музичних інструментів, оскільки користувач не має можливості фізично оцінити товар перед покупкою.

Наступною важливою функцією є можливість додавання товарів до кошика. Кошик дозволяє користувачу сформувати список обраних товарів перед оформленням замовлення. Після цього система повинна забезпечувати процес оформлення замовлення, який включає введення контактних даних та підтвердження покупки.

Крім того, доцільно реалізувати можливість реєстрації та авторизації користувачів. Це дозволяє зберігати історію замовлень, спрощує процес повторних покупок та підвищує зручність використання системи.

Основні сценарії взаємодії користувача із системою можуть бути представлені у вигляді діаграми варіантів використання.

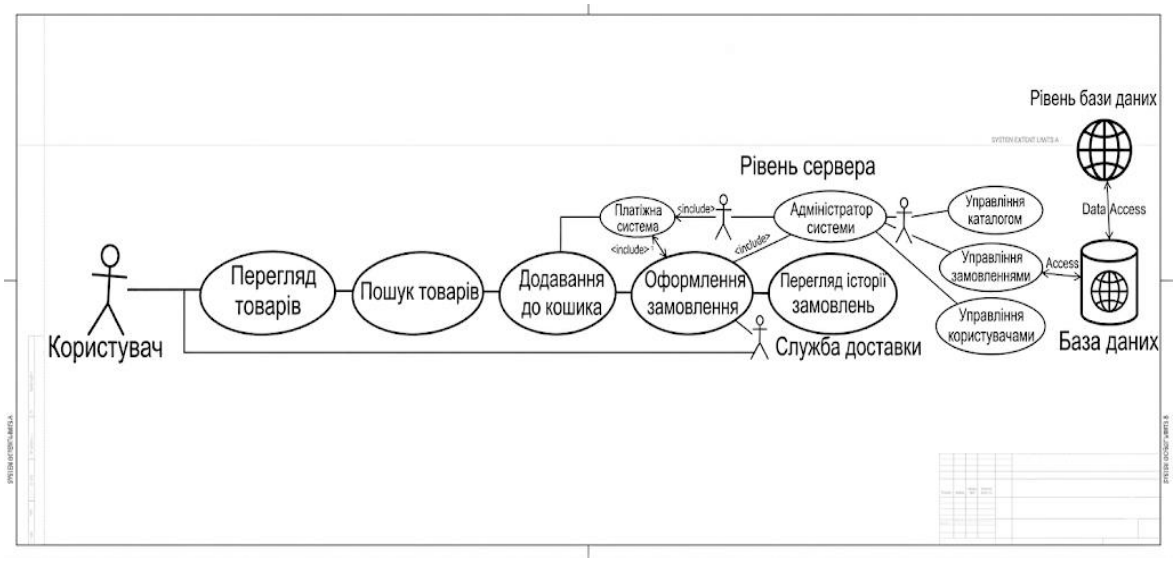


Рисунок 1.7 – Діаграма варіантів використання системи

Нефункціональні вимоги визначають якісні характеристики системи. Однією з найважливіших є продуктивність. Система повинна забезпечувати швидке завантаження сторінок навіть при великій кількості товарів у каталозі. Це безпосередньо впливає на зручність користування та загальне враження від сайту.

Не менш важливою є зручність користування. Інтерфейс повинен бути інтуїтивно зрозумілим, логічно структурованим та не вимагати від користувача додаткових зусиль для виконання основних дій. Навігація по сайту повинна бути простою та зрозумілою.

Система також повинна бути адаптивною, тобто коректно відображатися на різних типах пристроїв, включаючи персональні комп’ютери, планшети та мобільні телефони. Це є важливим, оскільки значна частина користувачів здійснює покупки саме з мобільних пристроїв.

До нефункціональних вимог також належить надійність системи. Вебсайт повинен працювати стабільно, без збоїв та помилок, навіть при значному навантаженні. Крім того, необхідно забезпечити належний рівень безпеки, зокрема захист персональних даних користувачів.

Важливою характеристикою є масштабованість. Система повинна мати можливість розширення функціоналу у майбутньому без необхідності повної переробки. Це дозволяє адаптувати інтернет-магазин до зростання кількості товарів та користувачів.

Окрему увагу слід приділити вимогам до структури даних. Система повинна забезпечувати ефективне зберігання інформації про товари, включаючи їх характеристики, категорії та інші параметри. При цьому база даних повинна підтримувати швидкий пошук, фільтрацію та сортування інформації.

Інтерфейс користувача повинен бути не лише функціональним, але й привабливим з візуальної точки зору. Це сприяє підвищенню довіри до сайту та покращує загальний досвід взаємодії.

Крім користувацької частини, система повинна мати адміністративну панель. Вона забезпечує можливість управління товарами, категоріями та замовленнями. Адміністратор повинен мати змогу додавати нові товари, редагувати існуючі, а також переглядати інформацію про оформлені замовлення.

Загальну взаємодію компонентів системи можна представити у вигляді узагальненої схеми на рисунку 1.8.

Рисунок 1.8 – Загальна модель взаємодії системи

Представлена модель демонструє основні етапи взаємодії користувача із вебзастосунком. Користувач надсилає запит через інтерфейс системи, після чого

інформація передається на сервер для обробки. Сервер виконує необхідні операції та звертається до бази даних для отримання або збереження інформації.

База даних забезпечує централізоване зберігання інформації про музичні інструменти, користувачів та замовлення. Після виконання необхідних операцій результати повертаються на сервер, а потім відображаються користувачу через вебінтерфейс.

Запропонована модель взаємодії дозволяє забезпечити швидкий обмін даними між компонентами системи та підтримує коректну роботу всіх функцій інтернет-магазину музичних інструментів. Такий підхід широко використовується під час створення сучасних вебзастосунків завдяки своїй надійності та можливості подальшого масштабування.

Таким чином, сформульовані вимоги визначають основні характеристики майбутнього програмного забезпечення та створюють основу для подальшого проєктування інтернет-магазину музичних інструментів.

Результати аналізу предметної області та існуючих програмних рішень дозволяють сформувати основні вимоги до майбутнього вебзастосунку.На основіотриманої інформації визначаються ключові функції системи, структура взаємодії між користувачем і вебсайтом, а також основні компоненти програмного забезпечення.

Для кращого розуміння логіки роботи системи та взаємодії користувачів із вебзастосунком доцільно використовувати UML-діаграми. Вони дозволяють графічно представити основні сценарії використання системи, структуру взаємодії між окремими елементами та основні функціональні можливості програмного продукту. Використання UML-діаграм є важливим етапом проєктування програмного забезпечення, оскільки вони допомагають спростити процес аналізу системи, виявити взаємозв’язки між компонентами та покращити розуміння загальної структури вебзастосунку.

На основі сформованих вимог та проведеного аналізу предметної області було розроблено UML-діаграму варіантів використання системи, яка відображає основні можливості користувача та адміністратора вебзастосунку.

Крім опису функціональних можливостей системи, важливим етапом проєктування є визначення ролей користувачів та їх взаємодії із вебзастосунком. У системі інтернет-магазину музичних інструментів можна виділити декілька основних категорій користувачів: звичайний користувач, зареєстрований клієнт та адміністратор системи.

Користувач вебзастосунку має можливість переглядати каталог товарів, здійснювати пошук музичних інструментів та оформлювати замовлення. У свою чергу адміністратор системи відповідає за керування товарами та оновлення інформації на вебсайті.

1.4 Висновки. Постановка задачі

У даному розділі було проведено аналіз предметної області, що стосується розробки інтернет-магазинів, зокрема систем для продажу музичних інструментів. Розглянуто основні особливості функціонування таких вебзастосунків, а також визначено ключові фактори, які впливають на ефективність їх роботи та зручність використання.

У ході дослідження було встановлено, що інтернет-магазини є складними інформаційними системами, які поєднують у собі функції представлення товарів, взаємодії з користувачем та обробки замовлень. Особливістю магазинів музичних інструментів є необхідність надання детальної інформації про товари, оскільки користувач не має можливості безпосередньо ознайомитися з продукцією перед покупкою.

Також було проаналізовано існуючі рішення у сфері електронної комерції. Визначено, що використання готових платформ дозволяє швидко створити

інтернет-магазин, однак має обмеження у гнучкості та можливостях адаптації. У той же час індивідуальна розробка забезпечує більшу свободу у реалізації функціоналу, але потребує більше ресурсів та часу.

На основі проведеного аналізу були сформульовані вимоги до програмного забезпечення, які враховують як функціональні можливості системи, так і її якісні характеристики. Визначено, що майбутній вебсайт повинен забезпечувати зручний доступ до каталогу товарів, ефективну систему пошуку та фільтрації, можливість оформлення замовлення, а також стабільну та швидку роботу.

Отримані результати дозволяють зробити висновок про доцільність розробки власного вебсайту інтернет-магазину музичних інструментів, який буде враховувати специфіку даної предметної області та забезпечувати необхідний рівень функціональності.

Виходячи з цього, у межах дипломної роботи необхідно вирішити ряд основних задач. Передусім слід дослідити принципи побудови інтернет-магазинів та визначити їх основні структурні елементи. Далі необхідно сформувати архітектуру вебзастосунку, яка забезпечить ефективну взаємодію між усіма компонентами системи. Важливим етапом є реалізація програмної частини, що включає розробку інтерфейсу користувача, серверної логіки та системи зберігання даних.

Крім того, необхідно провести тестування розробленого програмного продукту з метою перевірки його працездатності та відповідності поставленим вимогам. Це дозволить оцінити ефективність створеної системи та визначити можливі напрями її подальшого вдосконалення.

Таким чином, результати першого розділу створюють теоретичну основу для подальшого проєктування та реалізації вебсайту інтернет-магазину музичних інструментів, що буде розглянуто у наступних розділах роботи.

2. ПРОЕКТУВАННЯ СТРУКТУРИ ТА КОМПОНЕНТІВ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Проектування архітектури програмного забезпечення

Після визначення основних вимог до програмного забезпечення наступним етапом є проєктування архітектури вебзастосунку інтернет-магазину музичних інструментів. Архітектура системи визначає структуру взаємодії між основними компонентами програмного забезпечення та забезпечує ефективну реалізацію функціональних можливостей вебзастосунку.

Інтернет-магазин музичних інструментів є багатокомпонентною системою, яка включає клієнтську частину, серверну частину та систему зберігання даних. Кожен із цих компонентів виконує власні функції, але разом вони формують єдину систему для роботи користувача із вебзастосунком.

Клієнтська частина відповідає за взаємодію користувача з інтерфейсом системи. Саме frontend забезпечує відображення сторінок вебзастосунку, каталогу музичних інструментів, кошика покупок, сторінок авторизації та інших елементів системи. Користувач взаємодіє із сайтом через браузер, використовуючи графічний інтерфейс вебзастосунку.

Серверна частина відповідає за обробку запитів користувача, авторизацію користувачів та реалізацію бізнес-логіки системи. Сервер отримує запити від клієнтської частини, виконує необхідну обробку інформації та повертає результат користувачу.

Для реалізації архітектури системи було використано клієнт-серверний підхід, який є одним із найбільш популярних рішень у сучасній веброзробці. Даний підхід дозволяє розділити систему на окремі компоненти, що значно спрощує підтримку програмного забезпечення та подальше розширення функціоналу.

На рисунку 2.1 представлено загальну архітектуру вебзастосунку. Користувач взаємодіє із системою через браузер, використовуючи клієнтську частину вебзастосунку. Серверна частина виконує обробку запитів та забезпечує взаємодію між користувачем і системою.

Рисунок 2.1 – Архітектура вебсайту інтернет-магазину

Для реалізації серверної частини було використано платформу Node.js та фреймворк Express.js. Node.js дозволяє створювати швидкі та масштабовані вебзастосунки, які можуть ефективно працювати з великою кількістю одночасних запитів користувачів.

Однією з переваг Node.js є використання JavaScript як на серверній, так і на клієнтській стороні. Це дозволяє спростити процес розробки та забезпечити кращу взаємодію між окремими компонентами системи.

Використання Node.js для реалізації серверної частини вебзастосунку дозволяє забезпечити високу швидкість обробки запитів та стабільну роботу системи. Завдяки асинхронному принципу роботи сервер може одночасно обробляти велику кількість запитів користувачів без значного зниження продуктивності.

Особливо важливо це для інтернет-магазину музичних інструментів, оскільки система повинна забезпечувати стабільну роботу каталогу товарів,

пошуку, кошика та оформлення замовлень навіть при одночасній роботі декількох користувачів.

Фреймворк Express.js використовується для реалізації маршрутизації та обробки HTTP-запитів. Завдяки використанню Express.js серверна частина системи має більш зрозумілу структуру та спрощує процес розробки програмного забезпечення.

Під час проєктування архітектури особлива увага приділялася розподілу функцій між окремими компонентами системи. Клієнтська частина відповідає за відображення інтерфейсу користувача та взаємодію із вебсторінками, тоді як серверна частина забезпечує обробку даних та виконання основних функцій системи.

Для передачі інформації між frontend та backend частинами системи використовується API. Клієнтська частина надсилає HTTP-запити на сервер, після чого сервер виконує необхідну обробку даних та повертає результат у форматі JSON.

Наприклад, під час відкриття каталогу музичних інструментів система надсилає запит до сервера для отримання інформації про товари. Після отримання відповіді frontend відображає список музичних інструментів користувачу.

Аналогічний принцип використовується під час авторизації користувачів, додавання товарів до кошика та оформлення замовлення.

Важливою перевагою клієнт-серверної архітектури є можливість незалежного розвитку окремих компонентів системи. Це означає, що інтерфейс користувача або серверна частина можуть змінюватися незалежно один від одного без повної переробки вебзастосунку.

Під час проєктування архітектури також було використано принцип модульності. Це означає, що система складається з окремих функціональних компонентів, кожен із яких виконує власні функції.

До основних компонентів системи належать:

– модуль каталогу товарів;

– модуль авторизації;

– модуль кошика;

– модуль оформлення замовлень;

– адміністративний модуль.

Модуль каталогу відповідає за перегляд музичних інструментів та навігацію між категоріями товарів. Модуль авторизації забезпечує перевірку облікових даних користувача та вхід у систему.

Модуль кошика використовується для формування списку товарів перед оформленням замовлення. Модуль оформлення замовлення забезпечує створення покупки та передачу інформації на сервер.

Адміністративний модуль використовується для керування товарами, категоріями та іншими елементами вебзастосунку.

Крім того, під час проєктування архітектури було враховано можливість подальшого розширення функціоналу вебзастосунку. У майбутньому система може бути доповнена онлайн-оплатою, системою рекомендацій, відгуками користувачів та інтеграцією із сервісами доставки.

Таким чином, спроєктована архітектура вебзастосунку забезпечує стабільну роботу системи, зручну взаємодію між компонентами та можливість подальшого вдосконалення програмного забезпечення.

2.2 Проектування модулів

Під час розробки вебзастосунку інтернет-магазину музичних інструментів було виконано проєктування основних функціональних модулів системи. Поділ програмного забезпечення на окремі модулі дозволяє спростити структуру вебзастосунку, підвищити зручність підтримки програмного забезпечення та забезпечити можливість подальшого розширення функціоналу.

Кожен модуль системи виконує окремі функції та взаємодіє з іншими компонентами через визначені механізми. Такий підхід дозволяє більш ефективно

організувати структуру вебзастосунку та забезпечити стабільну роботу всієї системи.

До основних модулів вебзастосунку належать:

– модуль каталогу товарів;

– модуль авторизації та реєстрації;

– модуль кошика;

– модуль оформлення замовлень;

– адміністративний модуль.

Одним із головних модулів системи є модуль каталогу товарів. Його основним призначенням є відображення музичних інструментів та забезпечення зручної навігації між категоріями товарів.

У каталозі користувач може переглядати:

– гітари;

– клавішні інструменти;

– ударні установки;

– духові інструменти;

– музичні аксесуари.

Для кожного товару система відображає назву, фотографію, ціну та короткий опис. Після переходу на сторінку товару користувач отримує більш детальну інформацію про музичний інструмент, зокрема характеристики, опис та фотографії товару.

Важливою функцією модуля каталогу є можливість пошуку товарів за категоріями. Це дозволяє значно спростити навігацію по сайту та швидко знаходити необхідні музичні інструменти.

На рисунку 2.2 представлено діаграму структури взаємодії основних модулів вебзастосунку. Модулі системи взаємодіють між собою та забезпечують стабільну роботу всіх функцій інтернет-магазину.

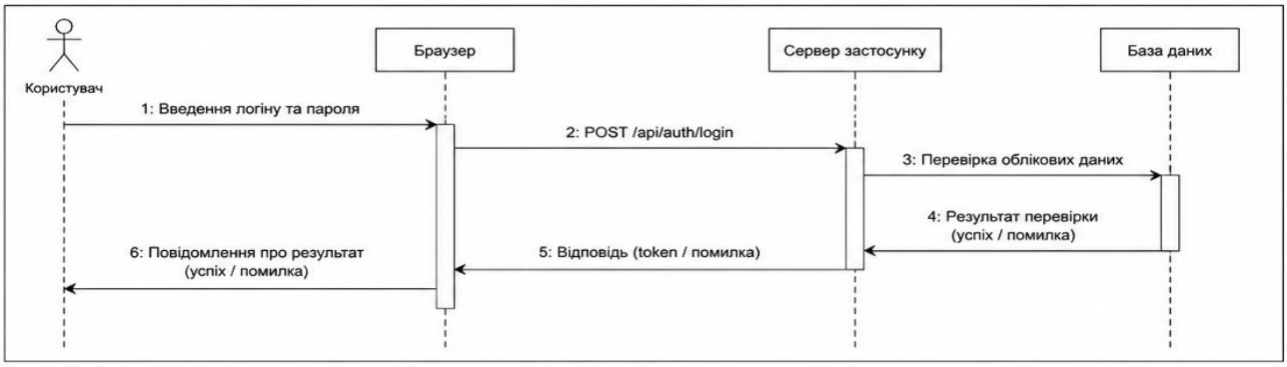


Рисунок 2.2 – Діаграма структури вебсайту інтернет-магазину

Наступним важливим компонентом системи є модуль авторизації та реєстрації користувачів. Даний модуль забезпечує створення облікових записів, перевірку введених даних та авторизацію користувачів у системі.

Під час реєстрації користувач вводить:

– логін;

– електронну пошту;

– пароль.

Після успішної реєстрації дані користувача зберігаються у системі та використовуються для подальшої авторизації. Під час входу система перевіряє правильність введених даних та надає користувачу доступ до функцій вебзастосунку.

Модуль авторизації дозволяє забезпечити персоналізацію роботи із системою та захист інформації користувачів. Крім того, авторизований користувач отримує доступ до кошика та можливості оформлення замовлення.

Одним із найважливіших компонентів вебзастосунку є модуль кошика. Даний модуль дозволяє користувачу формувати список обраних товарів перед оформленням замовлення.

Користувач має можливість:

– додавати музичні інструменти до кошика;

– змінювати кількість товарів;

– видаляти товари з кошика;

– переглядати загальну вартість покупки.

Під час роботи кошика система автоматично виконує підрахунок вартості замовлення залежно від кількості товарів та їх ціни.

Наявність кошика є важливою складовою будь-якого сучасного інтернет-магазину, оскільки дозволяє користувачу зручно працювати із товарами перед оформленням покупки.

Після завершення формування кошика користувач переходить до модуля оформлення замовлення. Даний модуль забезпечує створення покупки та передачу інформації до серверної частини системи.

Під час оформлення замовлення користувач вводить контактні дані та підтверджує покупку. Після цього система формує нове замовлення та зберігає інформацію про покупку.

Важливою особливістю модуля оформлення замовлення є можливість обробки декількох товарів у межах одного замовлення. Це дозволяє користувачу оформлювати комплексні покупки без необхідності створення окремого замовлення для кожного товару.

Окремим компонентом вебзастосунку є адміністративний модуль. Даний модуль призначений для керування товарами, категоріями та замовленнями системи.

Адміністратор має можливість:

– додавати нові товари;

– редагувати інформацію про музичні інструменти;

– видаляти товари;

– змінювати категорії;

– переглядати замовлення користувачів.

Наявність адміністративної панелі дозволяє спростити підтримку вебзастосунку та забезпечує можливість швидкого оновлення інформації на сайті.

Під час проєктування модулів було враховано можливість подальшого розширення функціоналу системи. У майбутньому вебзастосунок може бути доповнений новими функціями, наприклад системою рекомендацій музичних інструментів, онлайн-оплатою або відгуками користувачів.

Таким чином, проєктування модулів дозволило створити зрозумілу структуру вебзастосунку, забезпечити розподіл функцій між окремими компонентами системи та підвищити зручність підтримки програмного забезпечення.

2.3 Проєктування інтерфейсу користувача

Одним із найважливіших етапів розробки вебзастосунку інтернет-магазину музичних інструментів є проєктування бази даних. Саме база даних забезпечує збереження інформації про товари, користувачів, категорії та замовлення, а також підтримує стабільну роботу всіх основних функцій системи.

Для реалізації системи було використано систему керування базами даних MySQL. Дане рішення дозволяє ефективно працювати з великим обсягом інформації та забезпечує надійне збереження даних.

Під час проєктування бази даних особлива увага приділялася правильній організації зв’язків між таблицями. Це дозволяє уникнути дублювання інформації та забезпечує цілісність даних у системі.

База даних вебзастосунку містить декілька основних таблиць:

– users;

– products;

– categories;

– orders;

– order_items.

Таблиця users використовується для зберігання інформації про користувачів системи. У ній містяться логін, електронна пошта, пароль та інші дані, необхідні для авторизації користувача.

Таблиця products містить інформацію про музичні інструменти, зокрема назву товару, опис, ціну, фотографії та характеристики.

Для зручності навігації товари розподілені за окремими категоріями. Інформація про категорії музичних інструментів зберігається у таблиці categories.

Таблиця orders використовується для зберігання загальної інформації про замовлення користувача. У ній містяться дані про дату замовлення, користувача та загальну вартість покупки.

Таблиця order_items використовується для збереження товарів, які входять у конкретне замовлення. Такий підхід дозволяє одному замовленню містити декілька музичних інструментів одночасно.

На рисунку 2.3 представлено взаємозв'язки між таблицями бази даних. Усі таблиці пов'язані між собою за допомогою зовнішніх ключів, що забезпечує правильну організацію даних та стабільну роботу системи.

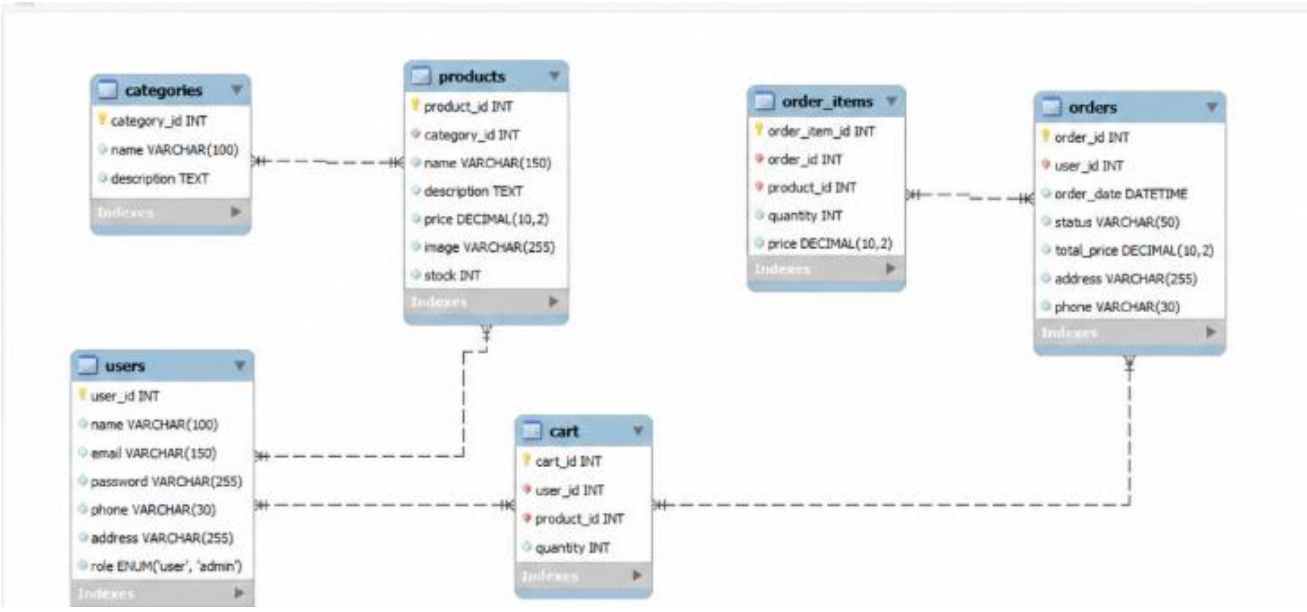


Рисунок 2.3 – ER-діаграма бази даних

Під час проєктування бази даних також було враховано можливість подальшого розширення функціоналу вебзастосунку. У майбутньому структура бази даних може бути доповнена новими таблицями для реалізації системи відгуків, онлайн-оплати або рекомендацій музичних інструментів.

Використання реляційної бази даних дозволяє забезпечити швидку обробку інформації та ефективну взаємодію між компонентами системи. Крім того, правильна організація структури бази даних спрощує підтримку програмного забезпечення та подальшу модернізацію вебзастосунку.

Таким чином, спроєктована база даних забезпечує надійне збереження інформації, підтримує роботу основних функцій інтернет-магазину музичних інструментів та дозволяє ефективно організувати взаємодію між компонентами системи.

.

2.4 Аналіз технологій і методів реалізації ПЗ

Під час розробки вебзастосунку інтернет-магазину музичних інструментів було проведено аналіз сучасних технологій вебпрограмування для вибору оптимальних засобів реалізації системи.

Основними критеріями вибору технологій були:

– швидкість роботи вебзастосунку;

– стабільність системи;

– зручність розробки;

– можливість подальшого розширення функціоналу;

– підтримка роботи з базами даних;

– зручність створення користувацького інтерфейсу.

Для реалізації серверної частини вебзастосунку було обрано платформу Node.js. Дане середовище дозволяє створювати сучасні вебзастосунки та забезпечує високу швидкість обробки запитів.

Однією з головних переваг Node.js є використання асинхронного принципу роботи. Завдяки цьому сервер може одночасно обробляти велику кількість запитів користувачів без значного зниження продуктивності.

Використання Node.js є особливо актуальним для інтернет-магазину музичних інструментів, оскільки система повинна забезпечувати стабільну роботу каталогу товарів, пошуку, кошика та оформлення замовлень.

Для реалізації серверної логіки було використано фреймворк Express.js. Express.js спрощує створення серверної частини вебзастосунку та забезпечує реалізацію маршрутизації й обробки HTTP-запитів.

Перевагою Express.js є проста структура проєкту та зручна організація серверного коду. Це дозволяє спростити підтримку програмного забезпечення та прискорює процес розробки.

Для створення клієнтської частини вебзастосунку було використано HTML, CSS та JavaScript.

HTML використовується для створення структури вебсторінок та розміщення основних елементів інтерфейсу користувача.

CSS застосовується для оформлення сторінок вебзастосунку, налаштування кольорової схеми, розташування елементів та створення сучасного дизайну системи.

JavaScript використовується для реалізації динамічної взаємодії користувача із вебзастосунком. За допомогою JavaScript реалізується робота кошика, взаємодія з API, обробка подій та оновлення інформації без перезавантаження сторінки.

Кожна з обраних технологій виконує власну функцію та забезпечує коректну роботу окремих компонентів системи. Їх спільне використання дозволяє реалізувати повноцінний вебзастосунок для продажу музичних інструментів із можливістю перегляду товарів, реєстрації користувачів, роботи з кошиком та оформлення замовлень.

Використання сучасних вебтехнологій забезпечує високу швидкість роботи системи, зручність її підтримки та можливість подальшого розширення

функціональних можливостей. Крім того, обрані технології мають широку підтримку серед розробників та активно використовуються під час створення сучасних вебзастосунків.

Схему взаємодії основних технологій, використаних під час розробки вебзастосунку, наведено на рисунку 2.4.

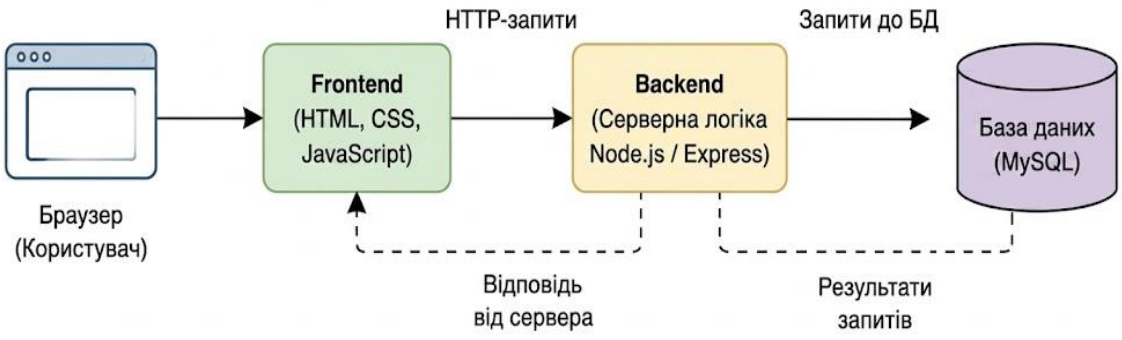


Рисунок 2.4 – Взаємодія технологій у вебзастосунку

Клієнтська частина взаємодіє із сервером через API, а серверна частина забезпечує роботу з базою даних.

Для зберігання інформації було використано систему керування базами даних MySQL. Дана система забезпечує надійне збереження інформації про користувачів, товари та замовлення.

Перевагою MySQL є підтримка реляційної структури баз даних та можливість організації зв’язків між таблицями. Це дозволяє забезпечити правильну структуру даних та ефективну взаємодію між компонентами системи.

Для розробки програмного забезпечення використовувалось середовище Visual Studio Code. Даний редактор коду забезпечує зручну роботу з JavaScript, Node.js та іншими технологіями веброзробки.

Під час розробки також використовувалась система контролю версій, що дозволяє спростити процес внесення змін до програмного коду та підтримку проєкту.

Під час вибору технологій особлива увага приділялася можливості подальшого масштабування вебзастосунку. Обрані технології дозволяють доповнювати систему новими функціями без необхідності повної переробки програмного забезпечення.

Таким чином, використання Node.js, Express.js, HTML, CSS, JavaScript та MySQL дозволило створити сучасний вебзастосунок для продажу музичних інструментів із зручним інтерфейсом, стабільною роботою та можливістю подальшого розвитку системи.

2.5 Проєктування користувацького інтерфейсу

Інтерфейс користувача є важливою складовою вебзастосунку інтернет-магазину музичних інструментів, оскільки саме через нього користувач взаємодіє із системою. Від якості інтерфейсу залежить зручність використання вебзастосунку та загальне враження від роботи із сайтом.

Під час проєктування інтерфейсу основна увага приділялася простоті навігації, сучасному дизайну та швидкому доступу до основних функцій системи. Інтерфейс повинен бути зрозумілим як для початківців, так і для професійних музикантів.

Головна сторінка вебзастосунку повинна містити навігаційне меню, яке забезпечує доступ до основних розділів сайту. Також на головній сторінці доцільно розміщувати популярні музичні інструменти, новинки та інформацію про магазин.

Для зручності користувача товари поділені на окремі категорії:

– гітари;

– клавішні інструменти;

– ударні установки;

– духові інструменти;

– музичні аксесуари.

Поділ товарів за категоріями дозволяє значно спростити пошук необхідного музичного інструмента та покращує навігацію по вебзастосунку.

Поділ товарів за категоріями дозволяє значно спростити пошук необхідного музичного інструмента та покращує навігацію по вебзастосунку.

Сторінка каталогу повинна містити список товарів із можливістю пошуку та фільтрації. Це дозволяє користувачу швидко знаходити необхідний товар серед великої кількості позицій.

Сторінка товару повинна містити детальну інформацію про музичний інструмент, зокрема:

– фотографії;

– опис;

– характеристики;

– ціну товару.

Також на сторінці товару повинна бути реалізована можливість додавання музичного інструмента до кошика.

Сторінка кошика повинна дозволяти користувачу переглядати обрані товари, змінювати їх кількість та переходити до оформлення замовлення.

Важливою частиною інтерфейсу є форма оформлення замовлення, яка повинна бути максимально простою та зрозумілою. Чим менше дій необхідно виконати користувачу для завершення покупки, тим більш зручним є вебзастосунок.

Під час проєктування інтерфейсу особлива увага приділялася зручності користування та швидкому доступу до основних функцій системи. Користувач має можливість швидко переходити між розділами сайту, переглядати доступні категорії товарів та отримувати детальну інформацію про музичні інструменти.

Важливою особливістю інтерфейсу є використання єдиного стилю оформлення для всіх сторінок вебзастосунку. Це забезпечує цілісність дизайну та покращує сприйняття інформації користувачем.

Також під час проєктування було враховано необхідність адаптації вебзастосунку до різних типів пристроїв. Завдяки адаптивному дизайну сайт коректно відображається як на персональних комп’ютерах, так і на планшетах та смартфонах.

Розроблений інтерфейс забезпечує зручну взаємодію користувача із системою та дозволяє ефективно виконувати основні операції, пов’язані з пошуком, вибором та придбанням музичних інструментів.

На рисунку 2.5 зображено основні елементи інтерфейсу користувача, які забезпечують зручну навігацію по вебзастосунку. Головна сторінка містить меню навігації, пошуковий рядок, каталог музичних інструментів та інформаційні блоки з популярними товарами.

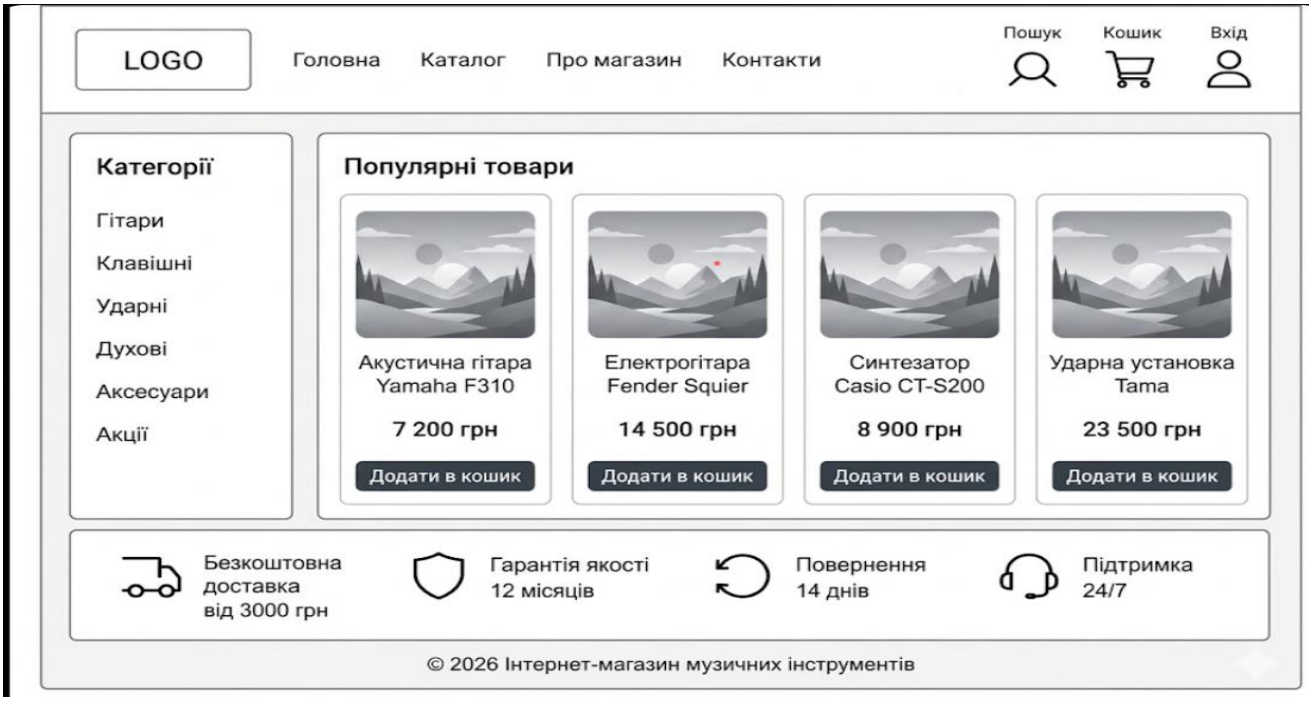


Рисунок 2.5 – Макет головної сторінки вебсайту

Розроблений інтерфейс забезпечує швидкий доступ до каталогу музичних інструментів, кошика та інших функцій системи. Особливу увагу приділено зручності навігації та візуальному оформленню сторінок.

Використання єдиного стилю оформлення сприяє покращенню користувацького досвіду та забезпечує цілісність дизайну вебзастосунку.

Завдяки адаптивному дизайну користувачі можуть комфортно працювати із сайтом як на персональних комп'ютерах, так і на мобільних пристроях.

При розробці інтерфейсу також важливо враховувати адаптивність. Сайт повинен коректно відображатися на різних пристроях, що забезпечує зручність користування незалежно від типу пристрою.

Таким чином, правильно спроєктований інтерфейс дозволяє підвищити ефективність роботи системи та покращити взаємодію з користувачем

Крім основних сторінок вебзастосунку, під час проєктування інтерфейсу було враховано необхідність забезпечення швидкої взаємодії користувача із системою. Для цього основні елементи управління розташовані у зрозумілих та доступних місцях інтерфейсу.

Особлива увага приділялася візуальному оформленню карток товарів. Для кожного музичного інструмента відображаються фотографія, назва, короткий опис та ціна. Це дозволяє користувачу швидко ознайомитися з основною інформацією про товар без необхідності переходу на окрему сторінку.

Під час розробки інтерфейсу також було враховано використання сучасних принципів UX/UI дизайну. Зокрема, інтерфейс повинен забезпечувати мінімальну кількість дій для виконання основних операцій, таких як пошук музичних інструментів, додавання товару до кошика та оформлення замовлення.

Важливою особливістю вебзастосунку є використання єдиного стилю оформлення для всіх сторінок системи. Це дозволяє забезпечити цілісність дизайну та покращує сприйняття інформації користувачем.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Програмна реалізація модулів

Клієнтська частина вебсайту інтернет-магазину музичних інструментів забезпечує взаємодію користувача із системою та реалізує основний функціонал інтерфейсу. Вона є важливим компонентом програмного продукту, оскільки саме через неї користувач отримує доступ до інформації про товари та може виконувати основні дії, пов’язані з оформленням замовлення.

У процесі розробки клієнтської частини було використано сучасні вебтехнології, зокрема HTML, CSS та JavaScript. Мова розмітки HTML застосовується для створення структури сторінок. Вона дозволяє визначити розташування елементів інтерфейсу, таких як заголовки, текстові блоки, зображення, кнопки та форми. Завдяки цьому формується логічна структура сторінки, яка є зрозумілою для користувача.

Оформлення вебсайту реалізовано за допомогою CSS. Використання стилів дозволило створити єдиний дизайн для всіх сторінок, забезпечити правильне розташування елементів та покращити візуальне сприйняття інформації. Особлива увага була приділена адаптивності, що дозволяє вебсайту коректно відображатися на різних типах пристроїв, включаючи мобільні телефони та планшети.

Для забезпечення інтерактивності використано мову програмування JavaScript. З її допомогою реалізовано обробку дій користувача, таких як натискання кнопок, введення даних у форми та взаємодія з кошиком. Також JavaScript використовується для динамічного оновлення вмісту сторінок без необхідності їх перезавантаження.

У процесі розробки було організовано структуру клієнтської частини, яка включає окремі сторінки для реалізації різного функціоналу. Основною сторінкою є `index.html`, яка виконує роль головної сторінки вебсайту. Для перегляду товарів

використовується сторінка `catalog.html`, яка містить список продукції та інструменти для пошуку і фільтрації.

Сторінка `product.html` призначена для відображення детальної інформації про окремий товар. Вона містить опис, характеристики та зображення музичного інструмента. Також на цій сторінці реалізовано можливість додавання товару до кошика.

Для роботи з обраними товарами використовується сторінка `cart.html`, яка дозволяє переглядати список доданих товарів, змінювати їх кількість або видаляти. Оформлення замовлення здійснюється на сторінці `checkout.html`, де користувач вводить свої контактні дані та підтверджує покупку.

Загальний вигляд головної сторінки вебзастосунку наведено на рисунку 3.1.

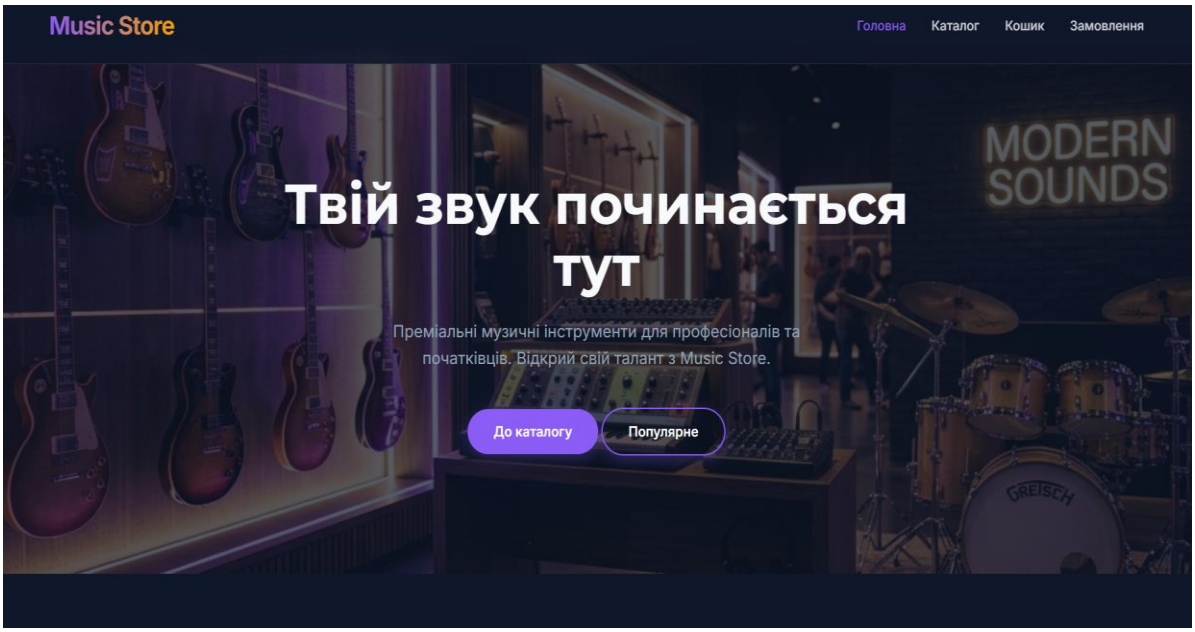


Рисунок 3.1 – Головна сторінка вебсайту

Сторінка каталогу забезпечує перегляд асортименту товарів та дозволяє використовувати пошук і фільтрацію для швидкого знаходження потрібного інструмента. Каталог є одним із ключових елементів вебзастосунку, оскільки саме через нього користувач ознайомлюється з доступним асортиментом музичних

інструментів. Для підвищення зручності навігації товари згруповані за категоріями, що дозволяє швидко перейти до потрібного типу інструментів.

Кожен товар у каталозі представлений у вигляді окремої картки, яка містить основну інформацію про музичний інструмент, його фотографію та вартість. Такий підхід дозволяє користувачу швидко порівнювати товари та приймати рішення щодо покупки.

Для більш детального ознайомлення з товаром передбачено окрему сторінку, на якій відображаються технічні характеристики, опис та додаткові відомості про музичний інструмент. Це дає змогу користувачу отримати всю необхідну інформацію перед оформленням замовлення.

Загальний вигляд сторінки каталогу вебзастосунку наведено на рисунку 3.2.

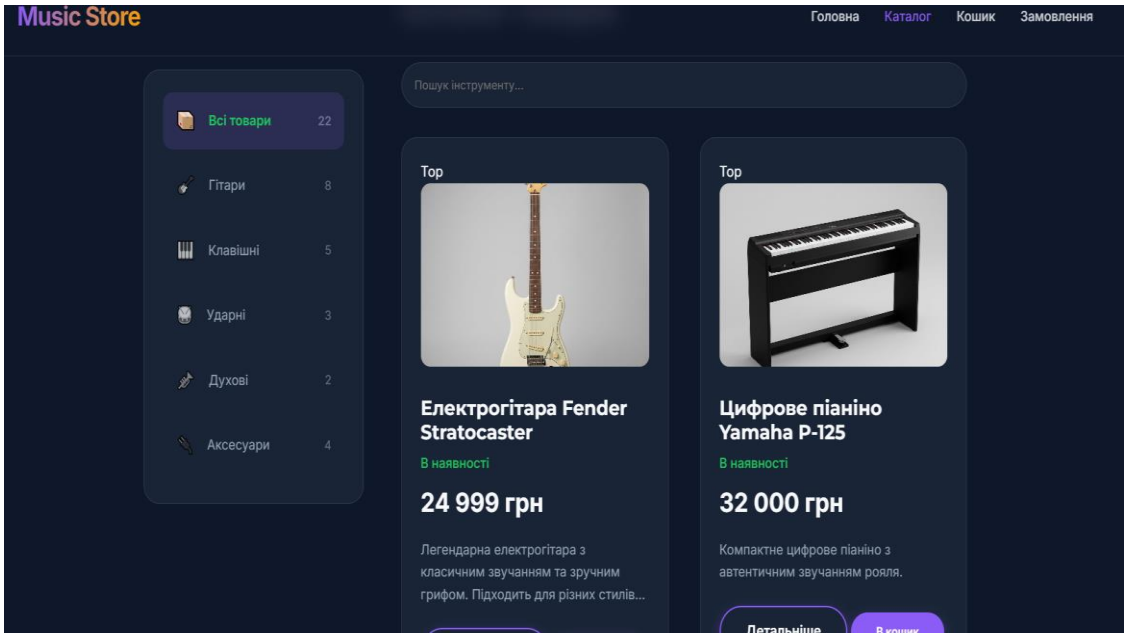


Рисунок 3.2 – Сторінка каталогу товарів

Представлена сторінка забезпечує зручний доступ до асортименту товарів та реалізує основні функції пошуку і навігації по каталогу музичних інструментів. Це значно спрощує процес вибору товарів та покращує взаємодію користувача із вебзастосунком.

Загальний вигляд сторінки товару наведено на рисунку 3.3.

Рисунок 3.3 – Сторінка товару

Кошик дозволяє користувачу переглядати обрані товари перед оформленням замовлення та змінювати їх параметри.

Загальний вигляд сторінки кошика вебзастосунку наведено на рисунку 3.4

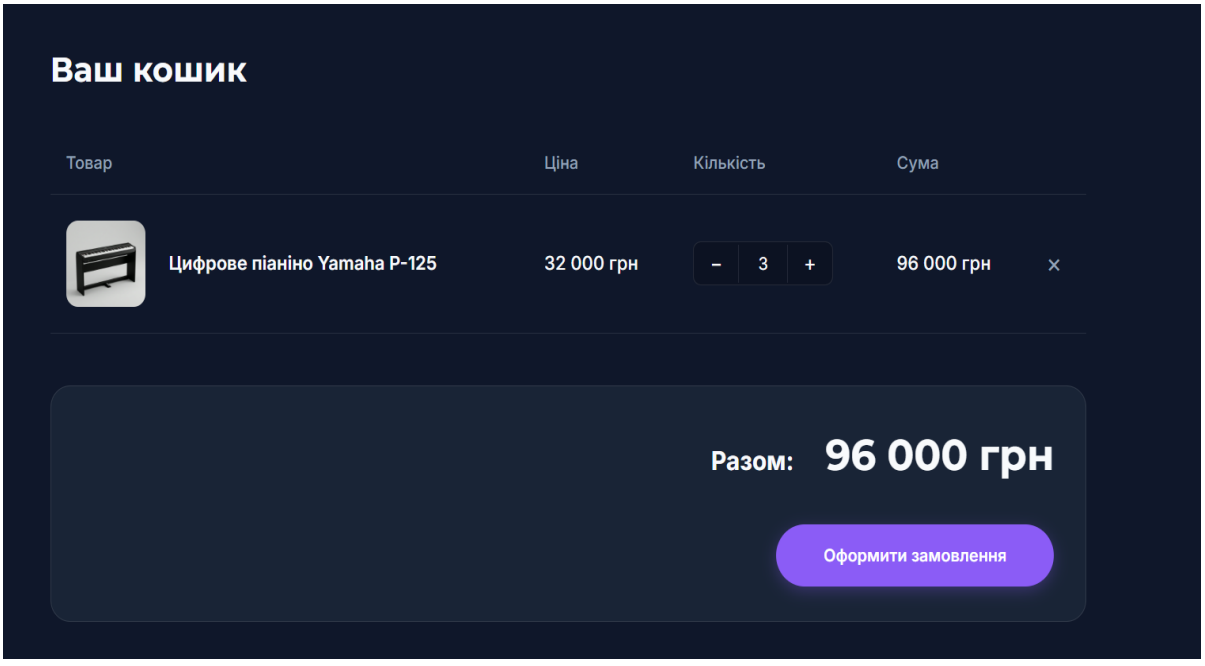


Рисунок 3.4 – Сторінка кошика

Сторінка кошика є важливим елементом вебзастосунку, оскільки забезпечує формування замовлення перед його оформленням. Користувач має можливість переглядати обрані музичні інструменти, змінювати їх кількість та контролювати загальну вартість покупки.

Кошик автоматично виконує перерахунок суми замовлення після зміни кількості товарів або видалення окремих позицій. Це дозволяє користувачу оперативно оцінювати вартість майбутньої покупки та приймати рішення щодо подальшого оформлення замовлення.

Також на сторінці кошика реалізовано можливість переходу до оформлення замовлення. Після натискання відповідної кнопки користувач переходить до форми введення контактних даних та підтвердження покупки.

Представлений інтерфейс забезпечує просту та зрозумілу взаємодію користувача із системою та відповідає основним вимогам сучасних інтернет-магазинів.

HTML є базовою технологією для створення структури вебсторінок та розміщення елементів інтерфейсу користувача. За допомогою HTML формуються блоки навігації, картки товарів, форми авторизації, сторінки каталогу та інші складові вебзастосунку.

У розробленому інтернет-магазині музичних інструментів HTML використовується для створення структури всіх основних сторінок системи. Кожен елемент інтерфейсу має власне призначення та забезпечує взаємодію користувача з функціоналом вебзастосунку.

Особлива увага приділялася правильній організації структури документа та використанню семантичних елементів HTML. Це дозволяє покращити читабельність програмного коду та спрощує його подальшу підтримку.

Для відображення інформації про музичні інструменти використовуються окремі блоки, які містять назву товару, зображення, ціну та короткий опис. Така структура забезпечує зручне відображення інформації та покращує користувацький досвід.

Крім того, HTML використовується для створення форм введення даних під час авторизації користувачів та оформлення замовлень. Завдяки цьому забезпечується коректна передача інформації між клієнтською та серверною частинами системи.

Важливою перевагою HTML є його сумісність із сучасними вебтехнологіями, такими як CSS та JavaScript. Це дозволяє створювати інтерактивні вебзастосунки із сучасним дизайном та широкими функціональними можливостями.

Для створення структури сторінок використовується HTML. Приклад фрагмента HTML-коду наведено на рисунку 3.5.

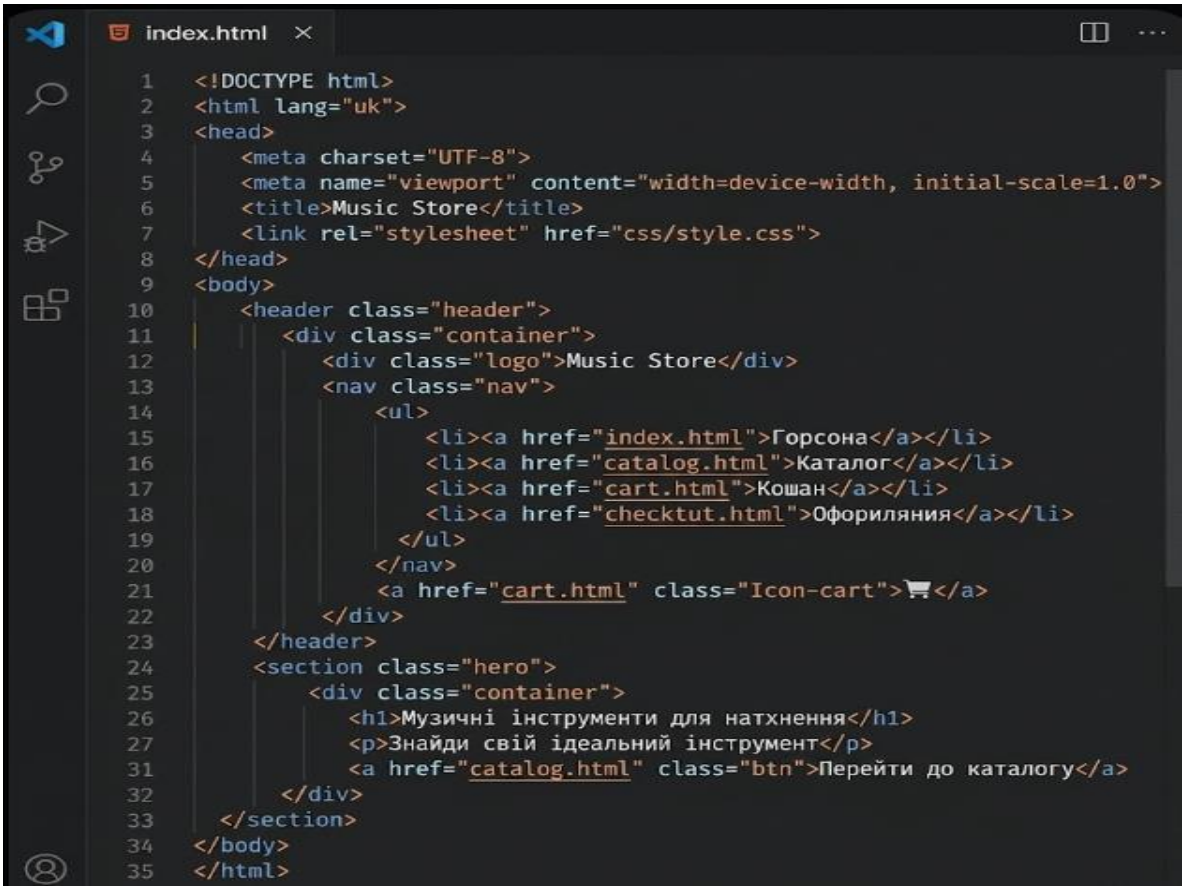


Рисунок 3.5 – Фрагмент HTML-коду сторінки

Оформлення інтерфейсу здійснюється за допомогою CSS. Приклад стилізації наведено на рисунку 3.6.

Рисунок 3.6 – Фрагмент CSS-коду

JavaScript використовується для реалізації основної логіки взаємодії користувача із вебзастосунком. За допомогою цієї технології забезпечується динамічне оновлення інформації на сторінках без необхідності їх повного перезавантаження.

Однією з важливих функцій є робота кошика покупця. Під час натискання кнопки додавання товару до кошика виконується відповідний програмний код, який передає інформацію про обраний музичний інструмент та оновлює дані користувача.

Крім того, JavaScript використовується для перевірки введених даних, обробки подій інтерфейсу та взаємодії із серверною частиною вебзастосунку. Це дозволяє забезпечити швидку та зручну роботу системи.

Використання клієнтських сценаріїв позитивно впливає на продуктивність вебзастосунку та покращує користувацький досвід під час перегляду каталогу музичних інструментів і оформлення замовлень.

Реалізація функції додавання товарів до кошика є однією з ключових складових вебзастосунку інтернет-магазину музичних інструментів. Саме цей механізм забезпечує можливість формування майбутнього замовлення та подальшого переходу до оформлення покупки.

Під час натискання кнопки додавання товару до кошика виконується JavaScript-код, який формує відповідний запит до серверної частини вебзастосунку. Сервер отримує дані про обраний товар, виконує їх обробку та зберігає інформацію у відповідних структурах даних.

Після успішного виконання операції інтерфейс користувача автоматично оновлюється та відображає актуальну кількість товарів у кошику. Такий підхід забезпечує швидку взаємодію між користувачем та системою без необхідності перезавантаження сторінки.

Важливою перевагою використання JavaScript є можливість виконання перевірки даних на стороні клієнта. Це дозволяє зменшити навантаження на сервер та підвищити швидкість роботи вебзастосунку.

Крім того, реалізований механізм забезпечує коректну взаємодію з API серверної частини, що дозволяє підтримувати актуальний стан кошика та синхронізувати дані між клієнтом і сервером.

Завдяки використанню сучасних можливостей JavaScript вдалося реалізувати зручний та швидкий механізм роботи кошика, який відповідає вимогам сучасних інтернет-магазинів та забезпечує комфортну взаємодію користувача із вебзастосунком.

Фрагмент JavaScript-коду, який реалізує додавання товарів до кошика, наведено на рисунку 3.7.

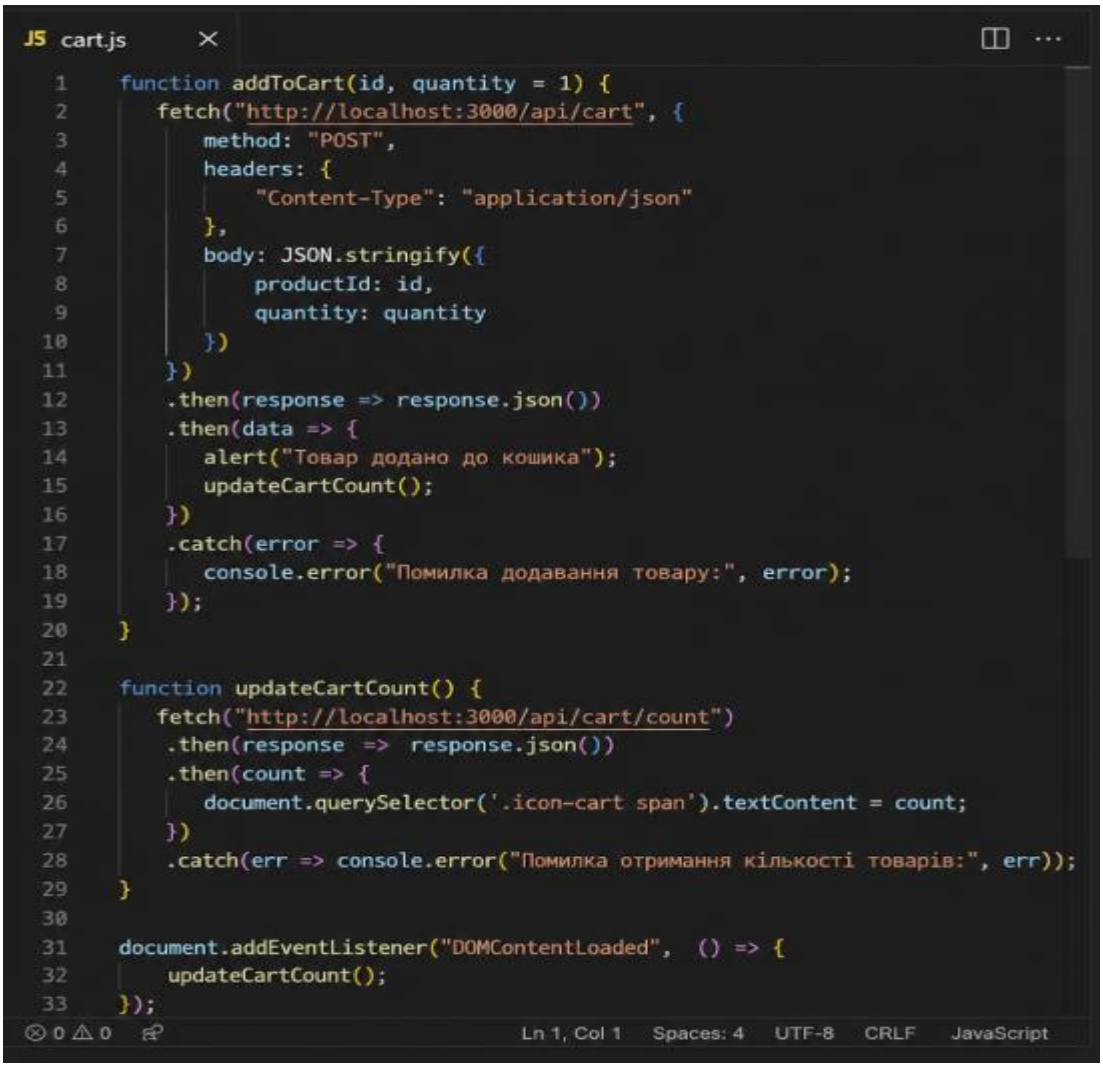


Рисунок 3.7 – JavaScript-код роботи кошика

Для отримання даних із серверної частини використовується функція fetch, яка дозволяє виконувати HTTP-запити.

Функція fetch використовується для обміну даними між клієнтською та серверною частинами вебзастосунку. За її допомогою клієнт може отримувати інформацію про музичні інструменти, категорії товарів, вміст кошика та інші дані, необхідні для роботи системи.

Схему реалізації HTTP-запиту за допомогою функції fetch наведено на рисунку 3.8.

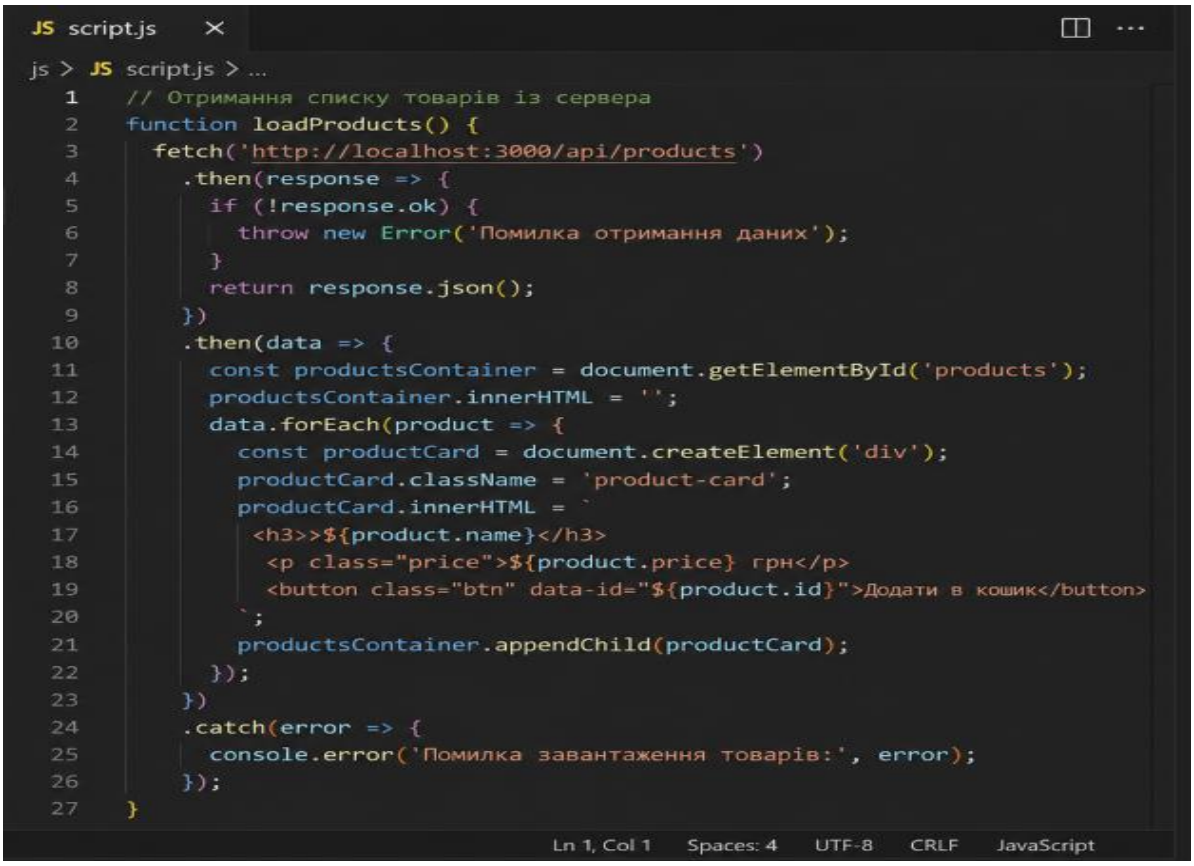


Рисунок 3.8 – Отримання даних із сервера

Після отримання відповіді від сервера дані обробляються на стороні клієнта та відображаються у відповідних елементах інтерфейсу. Такий підхід дозволяє оновлювати інформацію без повного перезавантаження сторінки та забезпечує більш комфортну роботу користувача із вебзастосунком.

Використання JavaScript у поєднанні з технологією fetch забезпечує швидку взаємодію між компонентами системи та дозволяє реалізувати сучасний

динамічний інтерфейс інтернет-магазину музичних інструментів. Завдяки цьому користувач може переглядати каталог товарів, додавати музичні інструменти до кошика та оформлювати замовлення в режимі реального часу.

Таким чином, використання JavaScript та сучасних засобів взаємодії із сервером дозволяє реалізувати основний функціонал вебзастосунку та забезпечити стабільну роботу системи під час виконання основних операцій користувача.

Передача даних на сервер, наприклад при оформленні замовлення, реалізується за допомогою POST-запиту, що наведено на рисунку 3.9.

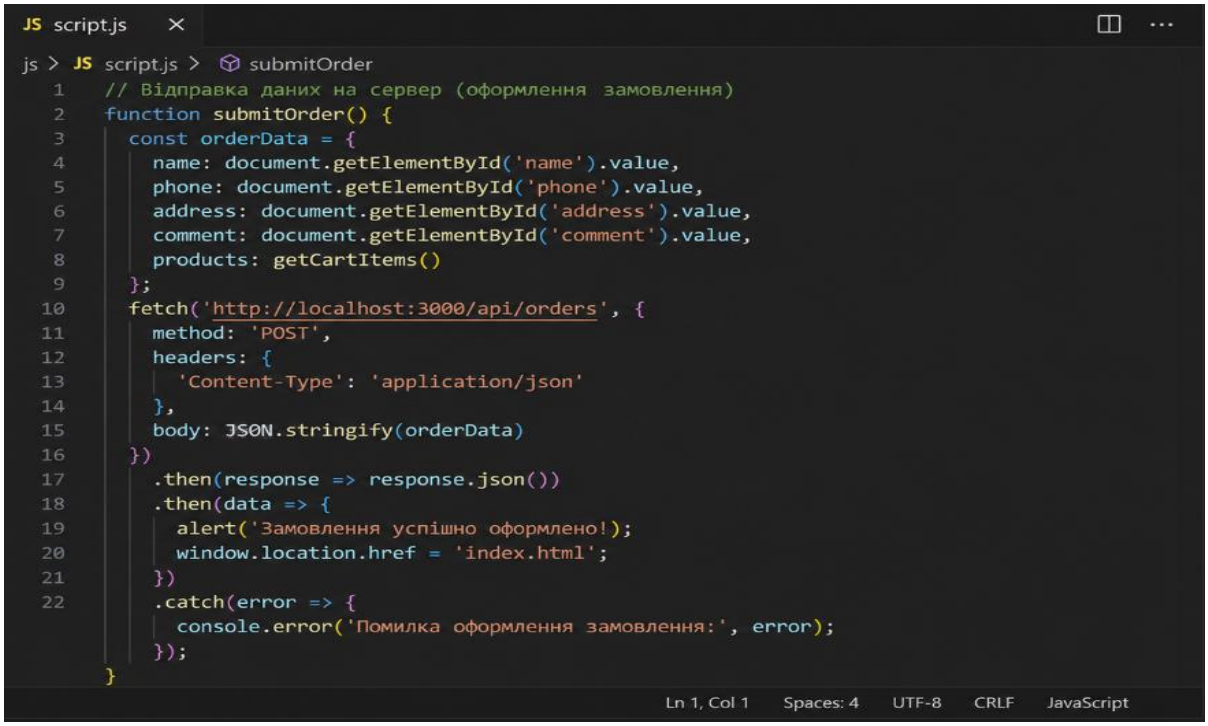


Рисунок 3.9 – Відправка даних на сервер

Технологія async/await значно спрощує роботу з асинхронним кодом та робить його більш зрозумілим для розробника. На відміну від використання великої кількості вкладених викликів або ланцюжків Promise, даний підхід дозволяє записувати асинхронні операції у вигляді звичайної послідовності команд.

Використання асинхронного програмування є особливо важливим під час розробки вебзастосунків, які активно взаємодіють із сервером та базою даних. У процесі роботи інтернет-магазину музичних інструментів виконується значна кількість запитів для отримання інформації про товари, категорії, користувачів та замовлення.

Під час завантаження каталогу система надсилає запит до серверної частини для отримання списку музичних інструментів. Після отримання відповіді дані автоматично відображаються користувачу без необхідності оновлення сторінки. Такий підхід покращує швидкодію вебзастосунку та забезпечує більш комфортну взаємодію із системою.

Асинхронний підхід також використовується під час авторизації користувачів, додавання товарів до кошика та оформлення замовлень. Це дозволяє виконувати обробку даних у фоновому режимі та не блокувати роботу інтерфейсу користувача.

Крім того, використання async/await сприяє підвищенню надійності програмного коду, оскільки спрощує обробку помилок та покращує структуру програмних модулів. Завдяки цьому код стає більш зрозумілим та легшим для подальшого супроводу. Використання конструкції async/await дозволяє спростити реалізацію асинхронних операцій та зробити програмний код більш зрозумілим для розробника. Завдяки цьому покращується читабельність коду та спрощується його подальша підтримка.

Даний підхід активно використовується під час взаємодії клієнтської частини вебзастосунку із сервером та базою даних. Це дозволяє забезпечити швидке виконання запитів і коректну обробку отриманих результатів без блокування роботи інтерфейсу користувача.

Схему використання конструкції async/await під час взаємодії вебзастосунку із серверною частиною наведено на рисунку 3.10.

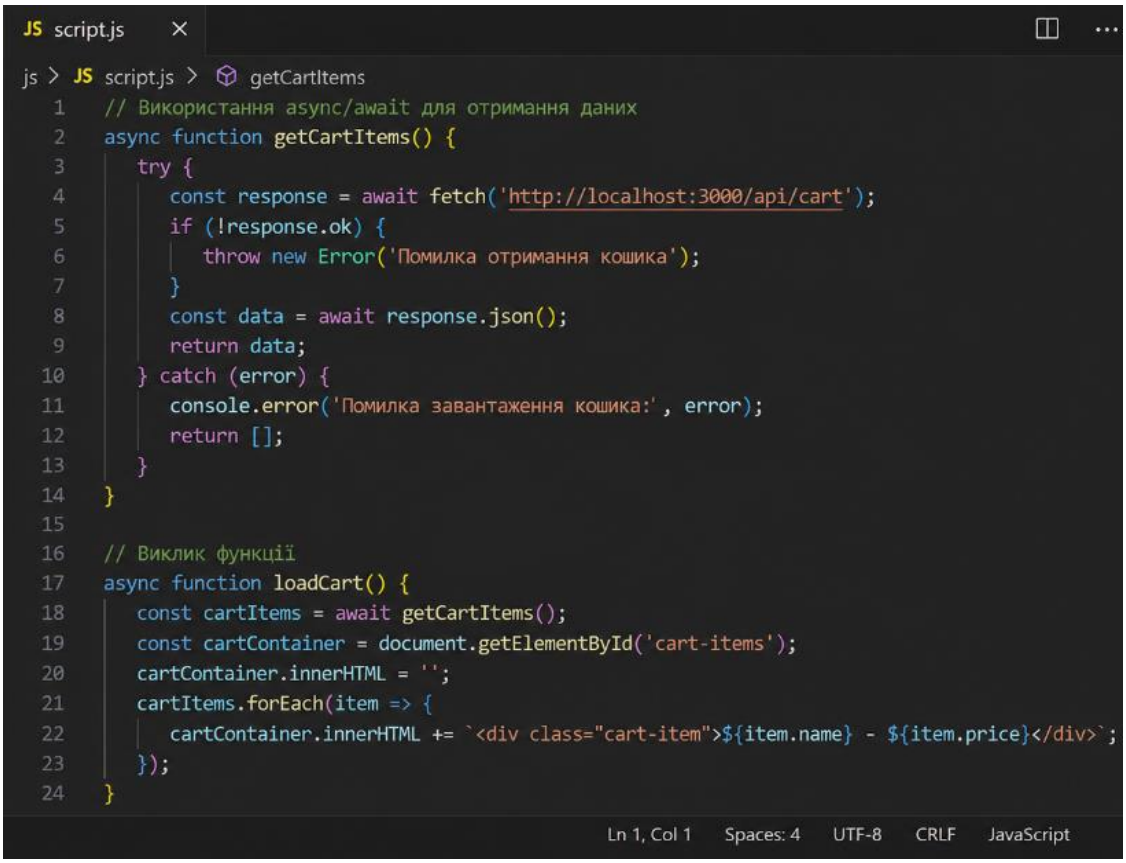


Рисунок 3.10 – Асинхронна обробка даних

Після отримання відповіді від сервера виконується обробка отриманих даних та їх відображення в інтерфейсі користувача. У випадку виникнення помилок система забезпечує їх коректну обробку та виведення відповідних повідомлень.

Використання async/await позитивно впливає на читабельність програмного коду та спрощує його подальшу підтримку. Це особливо важливо для вебзастосунків із великою кількістю запитів до сервера та взаємодією з базою даних.

Завдяки використанню сучасних можливостей JavaScript було реалізовано стабільну взаємодію між клієнтською та серверною частинами системи. Це дозволяє забезпечити коректну роботу каталогу музичних інструментів, кошика покупця, системи авторизації та оформлення замовлень.

Таким чином, використання асинхронного програмування дозволяє підвищити продуктивність вебзастосунку та забезпечити комфортну роботу користувачів із системою незалежно від кількості виконуваних операцій.

У результаті виконаної роботи було реалізовано клієнтську частину вебсайту, яка забезпечує зручну взаємодію користувача із системою, швидкий доступ до інформації про товари та можливість оформлення замовлення. Реалізований інтерфейс є зрозумілим, адаптивним та придатним для подальшого розширення функціоналу.

3.2 Тестування програмного забезпечення

Серверна частина вебсайту інтернет-магазину музичних інструментів забезпечує обробку запитів користувачів, виконання бізнес-логіки та взаємодію з базою даних. Вона є основою функціонування всієї системи, оскільки саме на сервері відбувається обробка даних та формування відповідей для клієнтської частини.

У процесі розробки серверної частини було реалізовано програмний модуль, який приймає HTTP-запити, обробляє їх та повертає відповіді у форматі JSON. Це дозволяє організувати ефективну взаємодію між клієнтською та серверною частинами та забезпечує швидку передачу даних.

Архітектура серверної частини побудована за модульним принципом, що передбачає розділення функціоналу на окремі компоненти. До основних елементів відносяться маршрути, обробники запитів, підключення до бази даних та механізми обробки помилок. Такий підхід дозволяє спростити структуру проєкту та полегшує його подальше розширення.

Однією з основних функцій серверної частини є обробка запитів на отримання інформації про товари. Для цього реалізовано маршрут, який обробляє GET-запити та повертає список товарів із бази даних. Отримані дані передаються

у форматі JSON, що дозволяє клієнтській частині легко їх обробляти та відображати.

Приклад реалізації маршруту отримання списку товарів наведено на рисунку 3.11.

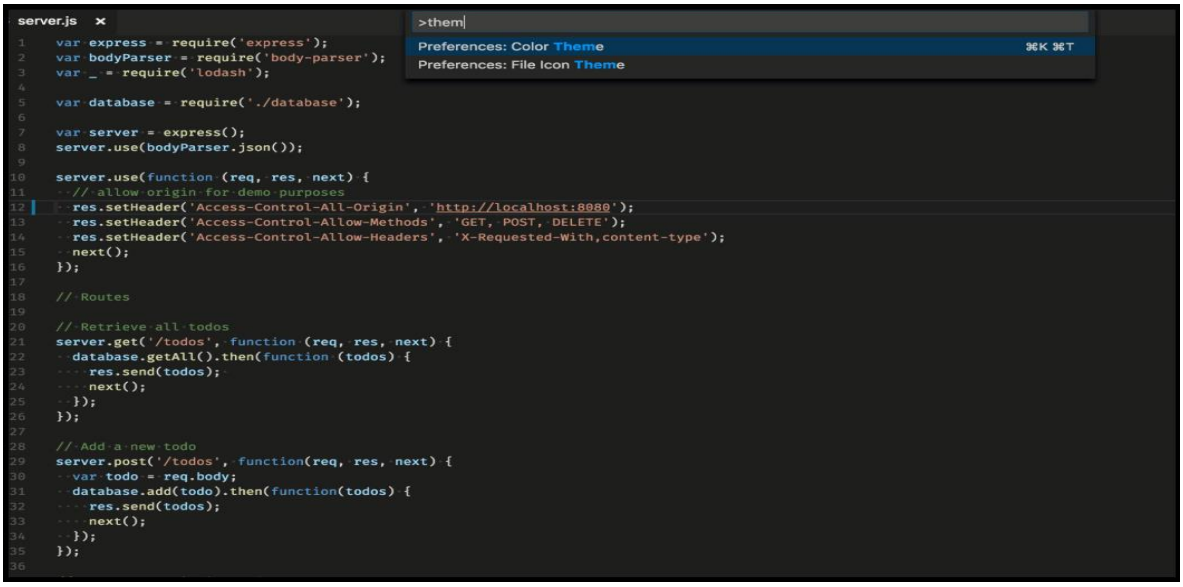


Рисунок 3.11 – Реалізація маршруту отримання товарів (GET /api/products)

Окрім цього, реалізовано можливість отримання інформації про конкретний товар за його ідентифікатором. Такий підхід дозволяє передавати лише необхідні дані та підвищує ефективність роботи системи.

Отримання інформації про окремий товар є важливою функцією інтернет-магазину музичних інструментів. Після вибору музичного інструмента користувач переходить на сторінку товару, де відображається детальна інформація про нього.

Під час виконання запиту сервер звертається до бази даних та отримує інформацію лише про той товар, який був запитаний користувачем. Це дозволяє зменшити обсяг переданих даних та підвищити швидкодію вебзастосунку.

Отримані дані містять назву музичного інструмента, його опис, вартість, категорію та інші характеристики, необхідні для ознайомлення користувача з товаром перед покупкою.

Крім операцій отримання даних, серверна частина також забезпечує обробку запитів на створення замовлень, авторизацію користувачів та роботу з кошиком. Для цього використовуються окремі маршрути API, які забезпечують взаємодію між клієнтською та серверною частинами системи.

Використання REST API дозволяє забезпечити чітке розділення логіки між frontend та backend частинами вебзастосунку. Такий підхід спрощує підтримку програмного забезпечення та дозволяє розширювати функціональні можливості системи в майбутньому. Отримання детальної інформації про музичний інструмент є важливою функцією вебзастосунку, оскільки саме на основі цих даних користувач приймає рішення щодо покупки товару. Для реалізації даної можливості було створено окремий серверний маршрут, який дозволяє отримувати інформацію про конкретний товар за його унікальним ідентифікатором.

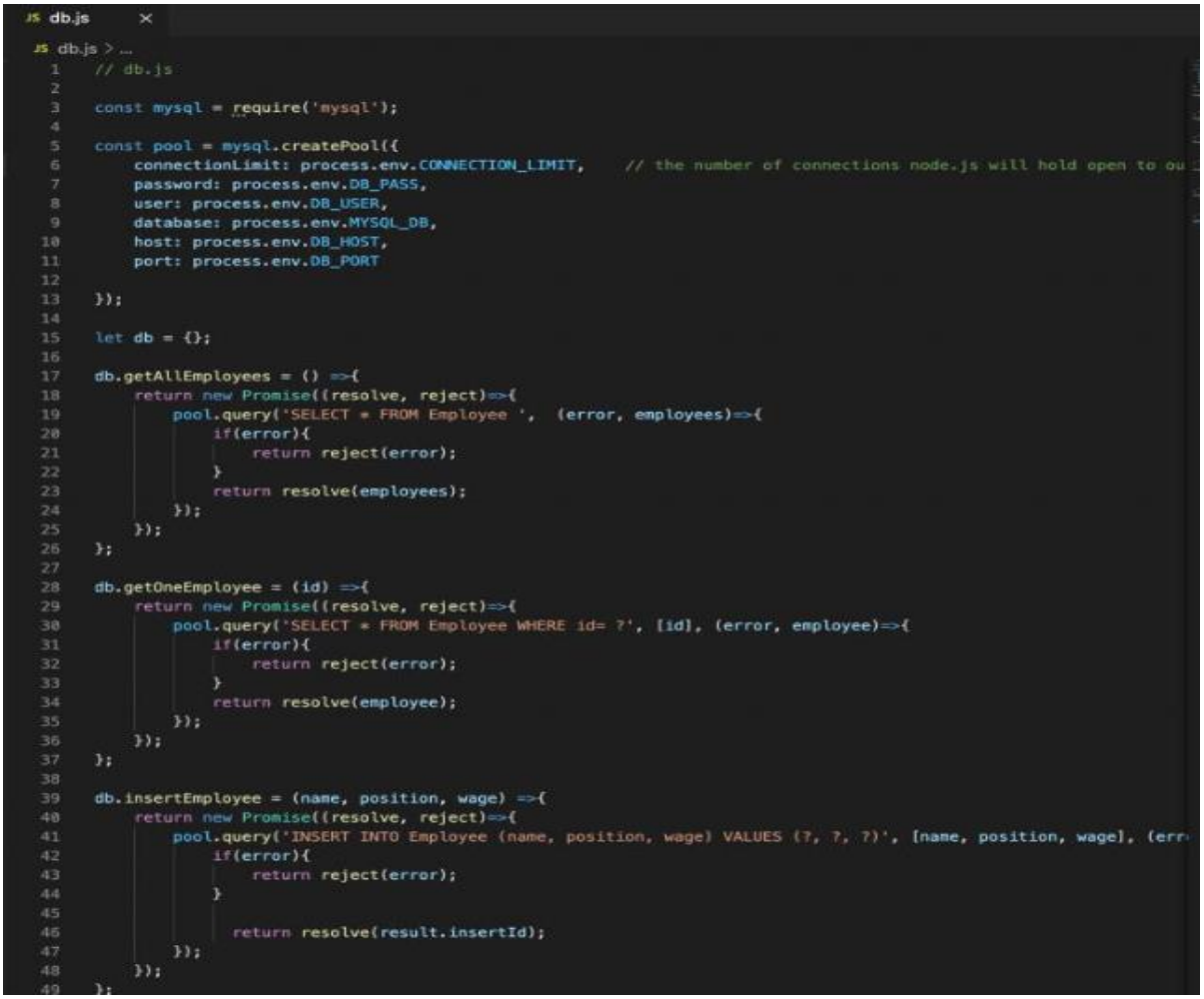
Після надходження запиту сервер виконує звернення до бази даних та формує відповідь, яка містить усю необхідну інформацію про вибраний музичний інструмент. До таких даних належать назва товару, його опис, вартість, категорія, характеристики та посилання на зображення.

Використання окремого маршруту для отримання детальної інформації дозволяє зменшити обсяг передаваних даних та підвищити продуктивність системи. Клієнтська частина отримує лише ту інформацію, яка необхідна для відображення сторінки конкретного товару.

Крім того, такий підхід спрощує підтримку програмного забезпечення та дозволяє легко розширювати функціональні можливості вебзастосунку в майбутньому. Наприклад, за необхідності можна додати нові характеристики

музичних інструментів або реалізувати систему відгуків користувачів без суттєвих змін існуючої архітектури системи.

Приклад реалізації маршруту отримання інформації про конкретний товар наведено на рисунку 3.12.



```
// db.js

const mysql = require('mysql');

const pool = mysql.createPool({
    connectionLimit: process.env.CONNECTION_LIMIT,    // the number of connections node.js will hold open to ou
    password: process.env.DB_PASS,
    user: process.env.DB_USER,
    database: process.env.MYSQL_DB,
    host: process.env.DB_HOST,
    port: process.env.DB_PORT

});

let db = {};

db.getAllEmployees = () =>{
    return new Promise((resolve, reject)=>{
        pool.query('SELECT * FROM Employee ',  (error, employees)=>{
            if(error){
                return reject(error);
            }
            return resolve(employees);
        });
    });
};

db.getOneEmployee = (id) =>{
    return new Promise((resolve, reject)=>{
        pool.query('SELECT * FROM Employee WHERE id= ?', [id], (error, employee)=>{
            if(error){
                return reject(error);
            }
            return resolve(employee);
        });
    });
};

db.insertEmployee = (name, position, wage) =>{
    return new Promise((resolve, reject)=>{
        pool.query('INSERT INTO Employee (name, position, wage) VALUES (?, ?, ?)', [name, position, wage], (err
            if(error){
                return reject(error);
            }

              return resolve(result.insertId);
        });
    });
};
```

Рисунок 3.12 – Отримання даних одного товару (GET /api/products/:id)

Для забезпечення роботи кошика створено окремий маршрут, який обробляє POST-запити. Користувач додає товар, після чого відповідна інформація передається на сервер, де вона обробляється та зберігається. Це дозволяє формувати список обраних товарів і використовувати його під час оформлення замовлення.

У випадку, якщо товар вже присутній у кошику користувача, система автоматично збільшує його кількість. Такий підхід дозволяє уникнути дублювання записів та забезпечує коректне зберігання інформації про обрані музичні інструменти.

Реалізацію маршруту додавання товару до кошика наведено на рисунку 3.13.

```
// Додавання товару до кошика
const express = require('express');
const router = express.Router();
const db = require('../db');

router.post('/', (req, res) => {
    const { user_id, product_id, quantity } = req.body;
    const sql = `INSERT INTO cart (user_id, product_id, quantity)
         VALUES (?, ?, ?)`
         ON DUPLICATE KEY UPDATE quantity = quantity + ?`;
   db.query(sql, [user_id, product_id, quantity, quantity], (err, result) => {
      if (err) {
         return res.status(500).json({ message: 'Помилка сервера' });
      }
      res.json({ message: 'Товар додано до кошика' });
    });
});
module.exports = router;
```

Рисунок 3.13 – Додавання товару до кошика (POST /api/cart)

Для завершення процесу покупки реалізовано механізм оформлення замовлення. Користувач вводить свої дані, які передаються на сервер. Сервер перевіряє їх коректність, обробляє та зберігає у базі даних. Після цього клієнту повертається підтвердження про успішне оформлення замовлення.

Процес оформлення замовлення є завершальним етапом роботи користувача з інтернет-магазином музичних інструментів. На цьому етапі система виконує перевірку отриманих даних та забезпечує їх коректне збереження у базі даних.

Після успішного створення замовлення інформація про покупку записується до відповідних таблиць бази даних. Зокрема, зберігаються відомості про

користувача, дату оформлення замовлення, перелік обраних музичних інструментів та загальну вартість покупки.

Використання окремих маршрутів API для роботи із замовленнями дозволяє забезпечити чіткий розподіл функціональних обов'язків між компонентами системи. Це позитивно впливає на підтримку програмного забезпечення та спрощує його подальший розвиток.

Крім того, реалізований механізм оформлення замовлень забезпечує можливість подальшого розширення функціоналу вебзастосунку. У майбутньому система може бути доповнена інтеграцією з платіжними сервісами, службами доставки та механізмами автоматичного інформування користувачів про статус замовлення.

Приклад реалізації маршруту оформлення замовлення наведено на рисунку 3.14.

```
// Оформлення замовлення
const express = require('express');
const router = express.Router();
const db = require('../db');

router.post('/', (req, res) => {
    const { user_id, name, phone, address, comment, total } = req.body;
    const sql = `INSERT INTO orders (user_id, name, phone, address,
                 comment, total, created_at)
                 VALUES (?, ?, ?, ?, ?, ?, NOW())`;
    db.query(sql, [user_id, name, phone, address, comment, total], (err, result) => {
      if (err) {
        return res.status(500).json({ message: 'Помилка сервера' });
      }
      res.json({ message: 'Замовлення успішно оформлено', order_id: result.insertId });
    });
});
module.exports = router;
```

Рисунок 3.14 – Обробка замовлення (POST /api/orders)

Серверна частина також здійснює взаємодію з базою даних. Для цього використовується окремий модуль підключення, який відповідає за встановлення з'єднання та виконання запитів. Завдяки цьому всі операції з даними виконуються централізовано, що спрощує їх обробку та підвищує надійність системи.

Приклад реалізації підключення до бази даних наведено на рисунку 3.15.

db.js > ...

```
1  // Підключення до бази даних MySQL
2  const mysql = require('mysql2');
3
4  const db = mysql.createConnection({
5      host: 'localhost',
6      user: 'root',
7      password: 'password',
8      database: 'music_store'
9  });
10 db.connect((err) => {
11     if (err) {
12         console.error('Помилка підключення до бази даних:', err.stack);
13     } else {
14         console.log('Підключено до бази даних MySQL');
15     }
16 });
17 module.exports = db;
```

Рисунок 3.15 – Приклад підключення до бази даних

У процесі роботи сервер може стикатися з різними помилками, тому було реалізовано механізм їх обробки. У разі виникнення помилки сервер формує відповідне повідомлення та передає його клієнту, що дозволяє уникнути збоїв у роботі системи.

Обробка помилок є важливою складовою серверної частини вебзастосунку. Наявність механізмів перевірки та обробки виняткових ситуацій дозволяє підвищити надійність роботи системи та забезпечити коректну взаємодію між її компонентами.

Під час виконання запитів можуть виникати помилки підключення до бази даних, некоректні дані від користувача або збої під час виконання окремих

операцій. Для кожної такої ситуації передбачено відповідні механізми обробки та інформування користувача.

Особлива увага приділяється перевірці результатів виконання SQL-запитів та контролю коректності отриманих даних. Це дозволяє уникнути втрати інформації та забезпечити стабільну роботу вебзастосунку навіть у випадку виникнення непередбачуваних ситуацій.

Крім того, реалізовані механізми логування помилок, що дозволяють швидко виявляти та усувати проблеми під час супроводу програмного забезпечення. Завдяки цьому спрощується процес тестування та подальшої підтримки системи.

Приклад реалізації механізму обробки помилок у серверній частині вебзастосунку наведено на рисунку 3.16.

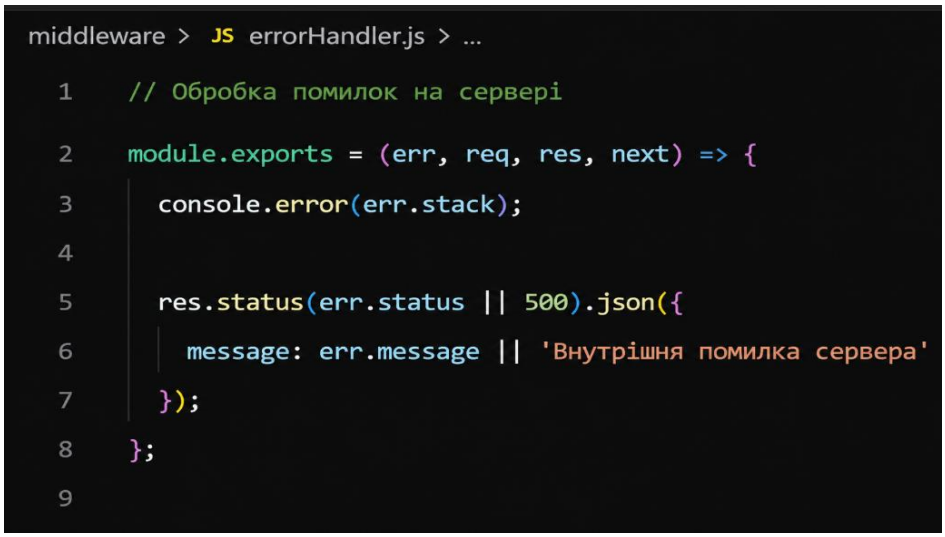


Рисунок 3.16 – Обробка помилок на сервері

Усі маршрути підключаються до основного файлу сервера, який відповідає за запуск програми. У цьому файлі також налаштовуються допоміжні компоненти, такі як обробка JSON-запитів та підтримка крос-доменних запитів. Після запуску сервер починає приймати запити та обробляти їх відповідно до визначених маршрутів.

Використання централізованої системи маршрутизації дозволяє логічно організувати структуру серверної частини вебзастосунку. Кожен маршрут відповідає за виконання окремої функції, що спрощує розробку, тестування та подальшу підтримку програмного забезпечення.

Такий підхід дозволяє централізовано керувати роботою серверної частини та забезпечує коректну взаємодію між усіма компонентами вебзастосунку для продажу музичних інструментів. Приклад запуску сервера та підключення маршрутів наведено на рисунку 3.17.

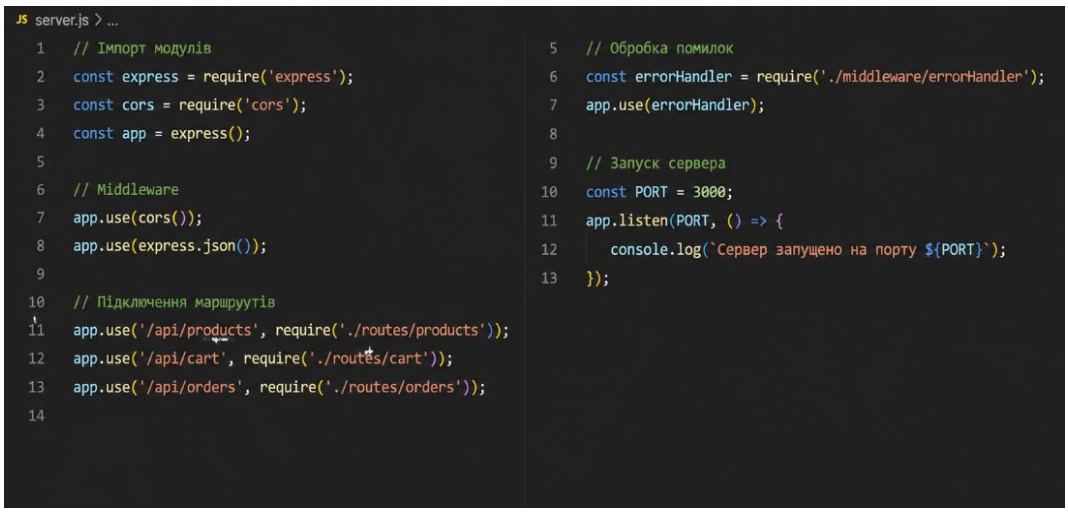


Рисунок 3.17 – Запуск сервера та підключення маршрутів

Під час розробки також враховано базові принципи безпеки. Зокрема, реалізовано перевірку вхідних даних перед їх обробкою, що дозволяє уникнути помилок та забезпечити стабільність роботи системи.

У результаті виконаної роботи було створено серверну частину вебсайту, яка забезпечує обробку запитів, взаємодію з базою даних та реалізацію основних функцій інтернет-магазину. Реалізована система є гнучкою, надійною та придатною для подальшого розширення функціоналу.

3.3 Висновки по реалізації та тестуванні ПЗ

Після завершення розробки клієнтської та серверної частин вебсайту було проведено тестування системи з метою перевірки її працездатності, стабільності та відповідності поставленим вимогам. Тестування є важливим етапом розробки програмного забезпечення, оскільки дозволяє виявити можливі помилки та забезпечити коректну роботу всіх функцій.

У процесі тестування було перевірено основні функціональні можливості вебсайту. Зокрема, здійснювалася перевірка роботи головної сторінки, каталогу товарів, сторінки окремого товару, кошика та механізму оформлення замовлення. Усі сторінки коректно відображаються у браузері та забезпечують зручну навігацію для користувача

Для відображення каталогу музичних інструментів на клієнтській стороні було реалізовано серверний маршрут отримання списку товарів. Після надходження запиту сервер виконує звернення до бази даних та повертає інформацію про доступні товари у форматі JSON.

Даний маршрут використовується під час завантаження каталогу та забезпечує передачу інформації про музичні інструменти від серверної частини до інтерфейсу користувача. Отримані дані використовуються для формування карток товарів та їх подальшого відображення на сторінках вебзастосунку.

Особлива увага приділялася перевірці взаємодії клієнтської та серверної частин. Було протестовано виконання HTTP-запитів до серверу, отримання даних у форматі JSON та їх відображення у вебінтерфейсі. Результати тестування показали, що передача даних відбувається коректно, а відповіді серверу обробляються без помилок. Завдяки використанню окремого маршруту забезпечується зручна організація серверної логіки та спрощується подальший розвиток програмного забезпечення.

Приклад реалізації маршруту отримання списку товарів наведено на рисунку 3.18.

Рисунок 3.18 – Отримання даних з сервера у браузері (Network / JSON)

Було перевірено роботу функції додавання товарів до кошика. Користувач має можливість додавати товари, змінювати їх кількість та видаляти. Дані коректно передаються на сервер і зберігаються.

Під час тестування особлива увага приділялася коректності обробки даних на всіх етапах роботи вебзастосунку. Було перевірено правильність взаємодії між клієнтською частиною, сервером та базою даних під час виконання основних операцій користувача.

Крім додавання товарів до кошика, було протестовано зміну кількості товарів, видалення окремих позицій та автоматичний перерахунок загальної вартості замовлення. Усі операції виконувалися коректно та відображали актуальний стан кошика.

Також було проведено перевірку процесу оформлення замовлення. Під час тестування встановлено, що інформація про замовлення успішно передається на сервер, обробляється та зберігається у базі даних MySQL без втрати даних.

Окремо виконувалося тестування роботи API-запитів за допомогою інструментів розробника браузера. Було підтверджено коректне отримання відповідей сервера та передачу даних у форматі JSON.

Результати тестування підтвердили стабільну роботу вебзастосунку, коректне функціонування основних модулів системи та відповідність реалізованого програмного забезпечення поставленим вимогам.

Результати тестування підтвердили коректну роботу даного функціоналу. Після надсилання запиту сервер успішно обробляє отримані дані та повертає відповідь про результат виконання операції.

Приклад перевірки роботи функції додавання товару до кошика наведено на рисунку 3.19.

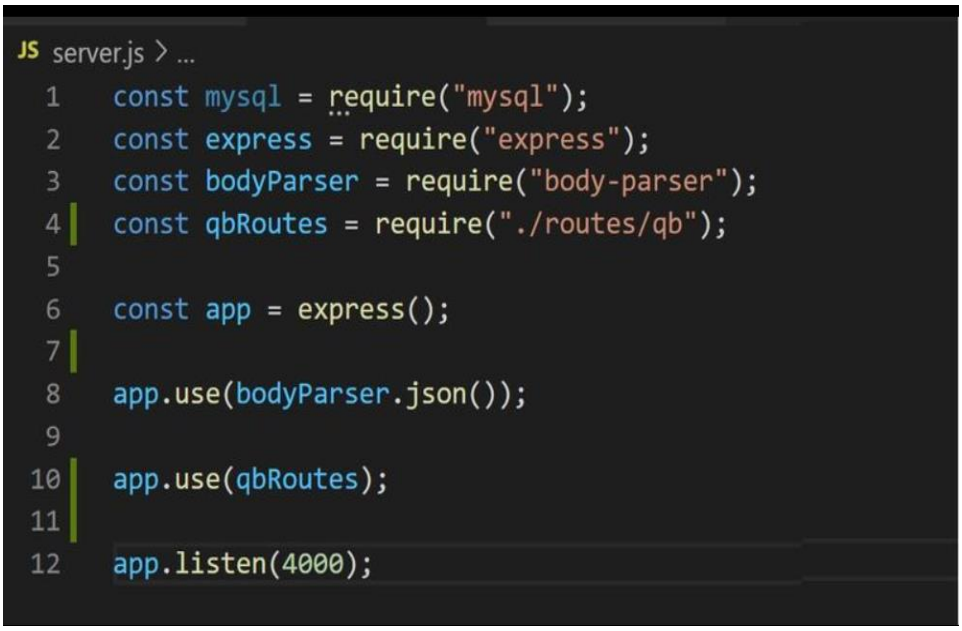


Рисунок 3.19 – Додавання товару до кошика

Окремо було протестовано процес оформлення замовлення. Після введення користувачем необхідних даних вони передаються на сервер, обробляються та зберігаються у базі даних. У разі успішного виконання операції користувач отримує відповідне повідомлення.

Приклад реалізації маршруту додавання товару до кошика наведено на рисунку 3.20.

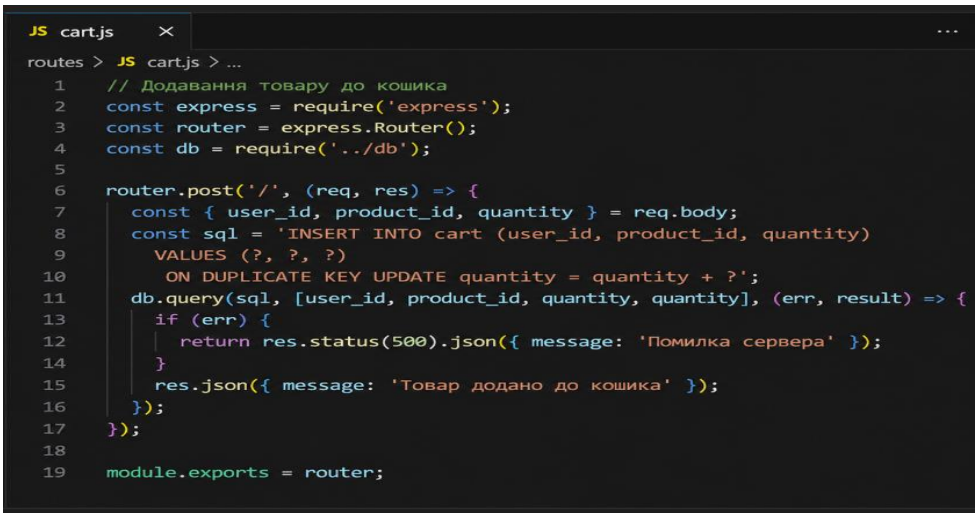


Рисунок 3.20 – Оформлення замовлення

Також було проведено тестування серверної частини з використанням спеціальних інструментів для перевірки API. Було перевірено роботу основних маршрутів, таких як отримання списку товарів, отримання інформації про товар, додавання до кошика та оформлення замовлення.

Під час тестування серверної частини особлива увага приділялася перевірці коректності обробки HTTP-запитів та взаємодії з базою даних. Для кожного маршруту API було перевірено правильність отримання, обробки та повернення даних клієнту.

Також виконувалося тестування механізмів перевірки вхідних даних. Це дозволило переконатися у тому, що сервер коректно обробляє як правильні, так і помилкові запити користувачів та запобігає виникненню критичних помилок під час роботи системи.

Окремо було перевірено процес створення замовлень та збереження інформації у базі даних. Під час тестування підтверджено, що всі дані про

замовлення, користувачів та товари успішно записуються до відповідних таблиць бази даних.

Крім функціонального тестування було проведено перевірку коректності відображення інформації в інтерфейсі користувача. Усі сторінки вебзастосунку відображаються без помилок та забезпечують коректну роботу основних функцій системи.

За результатами проведеного тестування критичних помилок виявлено не було. Виявлені незначні недоліки були усунені на етапі налагодження програмного забезпечення.

Таким чином, результати тестування підтвердили працездатність вебзастосунку для продажу музичних інструментів, стабільність його роботи та відповідність поставленим вимогам до програмного забезпечення.

Приклад тестування API-запитів наведено на рисунку 3.21

db.js > ...

```
// Підключення до бази даних MySQL
const mysql = require('mysql2');

const db = mysql.createConnection({
    host: 'localhost',
    user: 'root',
    password: '',
    database: 'music_store'
});

db.connect(err => {
    if (err) {
        console.error('Помилка підключення до бази даних:', err.stack);
        return;
    }
    console.log('Підключено до бази даних MySQL');
});

module.exports = db;
```

Рисунок 3.21 – Тестування API (Postman або аналог)

Під час тестування також перевірялася обробка помилок. У разі некоректного запиту або виникнення помилки сервер повертає відповідне повідомлення, що дозволяє клієнтській частині правильно реагувати на ситуацію.

Під час розробки вебзастосунку особлива увага приділялася забезпеченню стабільної роботи серверної частини. Для цього було реалізовано спеціальний механізм обробки помилок, який дозволяє своєчасно виявляти проблеми та повертати користувачу зрозумілі повідомлення про помилки.

Механізм обробки помилок використовується під час виконання всіх основних операцій, пов’язаних із отриманням даних, роботою з кошиком, авторизацією користувачів та оформленням замовлень. Це дозволяє уникнути аварійного завершення роботи сервера та підвищує надійність функціонування системи.

У випадку виникнення помилки інформація про неї фіксується у журналі сервера, після чого клієнт отримує відповідне повідомлення із зазначенням причини помилки. Такий підхід значно спрощує процес налагодження та супроводу програмного забезпечення.

Приклад реалізації механізму обробки помилок наведено на рисунку 3.22.

middleware > JS errorHandler.js > ...

```
1  // Обробка помилок на сервері
2
3  module.exports = (err, req, res, next) => {
4    console.error(err.stack);
5
6    res.status(err.status || 500).json({
7      message: err.message || 'Внутрішня помилка сервера'
8    });
9  };
```

Рисунок 3.22 – Обробка помилок при запиті

Крім функціонального тестування, було проведено перевірку швидкодії системи. Вебсайт швидко завантажується, а обробка запитів відбувається без помітних затримок. Це свідчить про ефективність реалізованої архітектури.

У процесі тестування значних помилок не виявлено, а всі основні функції працюють відповідно до поставлених вимог. Система демонструє стабільну роботу та може бути використана для подальшого розвитку і вдосконалення.

Таким чином, результати тестування підтверджують коректність роботи розробленого вебсайту інтернет-магазину та його готовність до використання.

ВИСНОВКИ

У результаті виконання дипломної роботи було розглянуто процес розробки вебсайту інтернет-магазину з продажу музичних інструментів. У ході роботи було проведено аналіз предметної області, досліджено існуючі рішення та визначено основні вимоги до програмного забезпечення. Це дозволило сформувати чітке уявлення про функціональні можливості системи та підходи до її реалізації.

На першому етапі було проаналізовано сучасний стан розвитку електронної комерції та особливості інтернет-магазинів. Встановлено, що використання вебтехнологій у сфері торгівлі дозволяє значно спростити процес купівлі товарів, розширити аудиторію клієнтів та підвищити ефективність ведення бізнесу. Особливу увагу було приділено інтернет-магазинам музичних інструментів, які мають свою специфіку, пов’язану з великою кількістю різновидів товарів та необхідністю надання детальної інформації про них.

У процесі аналізу було встановлено, що існуючі рішення мають як переваги, так і недоліки. Готові платформи дозволяють швидко створити інтернет-магазин, проте мають обмеження у функціоналі та налаштуванні. Власна розробка, навпаки, дає можливість створити систему, яка повністю відповідає поставленим вимогам, але потребує більше часу та ресурсів. На основі цього було прийнято рішення про розробку власного вебзастосунку.

У другому розділі було розглянуто архітектуру системи та обрано технології для її реалізації. Для створення клієнтської частини було використано HTML, CSS та JavaScript, що забезпечує зручний та адаптивний інтерфейс користувача. Серверна частина реалізована з використанням Node.js та фреймворку Express, що дозволяє ефективно обробляти запити та організувати взаємодію з базою даних.

Особливу увагу було приділено організації структури проєкту та розділенню функціоналу на окремі модулі. Такий підхід дозволив спростити процес розробки та забезпечити зручність підтримки коду. Було реалізовано систему маршрутів,

яка відповідає за обробку запитів до серверу, а також механізми роботи з базою даних.

У третьому розділі було виконано безпосередню реалізацію вебсайту інтернет-магазину. Було розроблено клієнтську частину, яка забезпечує взаємодію користувача з системою, включаючи перегляд товарів, пошук, додавання до кошика та оформлення замовлення. Інтерфейс було зроблено максимально простим та зрозумілим, що дозволяє користувачам швидко орієнтуватися у функціоналі сайту.

Серверна частина забезпечує обробку запитів, виконання бізнес-логіки та взаємодію з базою даних. Було реалізовано основні маршрути для роботи з товарами, кошиком та замовленнями. Використання API дозволяє організувати ефективний обмін даними між клієнтом та сервером.

Важливим етапом роботи стало тестування розробленого програмного забезпечення. Було перевірено працездатність усіх основних функцій системи, включаючи отримання даних з серверу, додавання товарів до кошика та оформлення замовлення. Результати тестування показали, що система працює стабільно та відповідає поставленим вимогам.

У процесі виконання роботи було досягнуто поставленої мети — розроблено вебсайт інтернет-магазину, який дозволяє користувачам переглядати асортимент товарів, отримувати інформацію про них та здійснювати покупки. Реалізована система є зручною у використанні, має зрозумілий інтерфейс та забезпечує коректну обробку даних.

До основних результатів роботи можна віднести:

– проведення аналізу предметної області та існуючих рішень;

– визначення вимог до програмного забезпечення;

– розробку архітектури системи;

– реалізацію клієнтської та серверної частин вебсайту;

– створення бази даних та організацію роботи з нею;

– тестування системи та перевірку її працездатності.

Отримані результати можуть бути використані для подальшого розвитку системи. Зокрема, можливе розширення функціоналу, додавання нових можливостей, таких як система авторизації користувачів, інтеграція платіжних систем, покращення дизайну та оптимізація продуктивності.

Таким чином, виконана дипломна робота має практичну цінність та може бути використана як основа для створення повноцінного інтернет-магазину. Розроблена система відповідає сучасним вимогам до вебзастосунків та демонструє можливість ефективного використання сучасних технологій у сфері електронної комерції.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 29.03.2026).

2. Express.js Documentation. URL: https://expressjs.com/ (дата звернення: 02.04.2026).

3. MySQL 8.0 Reference Manual. URL: https://dev.mysql.com/doc/ (дата звернення: 31.03.2026).

4. React Documentation. URL: https://react.dev/ (дата звернення: 05.04.2026).

5. TypeScript Documentation. URL: https://www.typescriptlang.org/docs/ (дата звернення: 08.04.2026).

6. MDN Web Docs. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript (дата звернення: 30.03.2026).

7. Bootstrap Documentation. URL: https://getbootstrap.com/docs/ (дата звернення: 12.04.2026).

8. REST API Tutorial. URL: https://restfulapi.net/ (дата звернення: 04.04.2026).

9. Visual Studio Code Documentation. URL: https://code.visualstudio.com/docs (дата звернення: 09.04.2026).

10. GitHub Docs. URL: https://docs.github.com/ (дата звернення: 14.04.2026).

11. Figma Documentation. URL: https://help.figma.com/ (дата звернення: 01.04.2026).

12. UML Resource Page. URL: https://www.uml.org/ (дата звернення: 16.04.2026).

13. W3Schools JavaScript Tutorial. URL: https://www.w3schools.com/js/ (дата звернення: 06.04.2026).

14. W3Schools CSS Tutorial. URL: https://www.w3schools.com/css/ (дата звернення: 18.04.2026).

15. HTML Living Standard. URL: https://html.spec.whatwg.org/ (дата звернення: 07.04.2026).

16. CSS Reference. URL: https://developer.mozilla.org/en-US/docs/Web/CSS (дата звернення: 11.04.2026).

17. JSON Official Website. URL: https://www.json.org/json-en.html (дата звернення: 13.04.2026).

18. JWT Introduction. URL: https://jwt.io/introduction (дата звернення: 03.04.2026).

19. Axios HTTP Client Documentation. URL: https://axios-http.com/docs/intro (дата звернення: 20.04.2026).

20. npm Documentation. URL: https://docs.npmjs.com/ (дата звернення: 10.04.2026).

21. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1995. 395 p.

22. Martin R. Clean Code: A Handbook of Agile Software Craftsmanship. Boston : Prentice Hall, 2008. 464 p.

23. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley, 2002. 533 p.

24. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2015. 816.

25. Pressman R. Software Engineering: A Practitioner’s Approach. New York : McGraw-Hill, 2014. 976 p.

26. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. New York : McGraw-Hill, 2019. 1376 p.

27. Date C. An Introduction to Database Systems. Boston : Pearson, 2003. 1024 p.

28. Sebesta R. Concepts of Programming Languages. Boston : Pearson, 2016. 792 p.

29. Freeman E., Robson E. Head First Design Patterns. Sebastopol : O’Reilly Media, 2021. 694 p.

30. Nixon R. Learning PHP, MySQL & JavaScript. Sebastopol : O’Reilly Media, 2018. 820 p.

31. Flanagan D. JavaScript: The Definitive Guide. Sebastopol : O'Reilly Media, 2020. 704 p.

32. Duckett J. HTML and CSS: Design and Build Websites. Indianapolis : Wiley, 2014. 490 p.

33. Duckett J. JavaScript and JQuery: Interactive Front-End Web Development. Indianapolis : Wiley, 2014. 640 p.

34. Krug S. Don't Make Me Think. Indianapolis : New Riders, 2014. 216 p.

35. Evans E. Domain-Driven Design. Boston : Addison-Wesley, 2003. 560 p.

36. Oracle JavaScript Documentation. URL: https://docs.oracle.com/javase/8/docs/technotes/guides/javascript/ (дата звернення: 17.04.2026).

37. OpenJS Foundation. Node.js Best Practices. URL: https://openjsf.org/ (дата звернення: 15.04.2026).

38. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 21.04.2026).

39. Linux Documentation Project. URL: https://tldp.org/ (дата звернення: 19.04.2026).

40. OWASP Foundation. Web Application Security Testing. URL: https://owasp.org/ (дата звернення: 22.04.2026).

Додаток А

(Обов’язковий)

Технічне завдання

Введення:

Робота виконується в межах розробки веб-застосунку «Music store», який призначений для організації онлайн-продажу музичних інструментів та супутнього обладнання. Актуальність теми обумовлена активним розвитком електронної комерції та зростанням попиту на дистанційні способи придбання товарів.

---

1 Підстава для розробки

Підставою для розробки даного вебзастосунку є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри програмного забезпечення.

Найменування розробки:

«Вебзастосунок для продажу музичних інструментів»

---

2 Призначення розробки

2.1 Функціональне призначення

Функціональне призначення вебзастосунку полягає у наданні користувачам можливості перегляду, вибору та придбання музичних інструментів і обладнання через мережу Інтернет.

Система забезпечує:

- доступ до каталогу товарів;
- перегляд детальної інформації про товари;
- додавання товарів до кошика;
- оформлення замовлення;
- взаємодію між клієнтами та представниками магазину.

Таким чином, вебзастосунок виступає як посередник між покупцем та продавцем, забезпечуючи зручний процес здійснення покупок у онлайн-режимі.

2.2 Експлуатаційне призначення

Вебзастосунок призначений для використання широким колом користувачів без спеціальної підготовки.

Доступ до системи здійснюється через веббраузер на таких пристроях:

- персональні комп’ютери;
- ноутбуки;
- планшети;
- смартфони.
- Підтримувані операційні системи:
- Windows;
- macOS;
- Android;
- iOS.

3 Вимоги до програмного продукту

3.1 Вимоги до функціональних характеристик

Система повинна забезпечувати такі можливості:

– перегляд каталогу товарів та детальної інформації про кожен товар;

– реєстрацію користувачів, а також вхід та вихід із системи;

– додавання товарів до кошика;

– оформлення замовлення;

– взаємодію з працівниками магазину у разі виникнення питань або проблем.

3.2 Вимоги до надійності

Вебзастосунок повинен забезпечувати безпечне зберігання та передачу даних користувачів.

Основні вимоги:

- передача персональних даних та платіжної інформації у зашифрованому вигляді;
- захист від несанкціонованого доступу;
- стабільна робота системи без критичних збоїв.
- У разі виникнення помилок система повинна:
- відображати зрозуміле повідомлення про помилку;
- забезпечувати коректне перенаправлення користувача на робочу сторінку.

3.3 Вимоги до складу та параметрів технічних засобів та програмної сумісності

Мінімальні вимоги для ПК:

- процесор 32 (x86) або 64 (x64);
- не менше 4 ГБ оперативної пам’яті;
- не менше 4 ГБ вільного місця на диску;
- відеокарта з підтримкою DirectX 10;
- операційна система Windows 7 або новіша;
- доступ до мережі Інтернет.
- Вимоги для мобільних пристроїв:
- Android 4.0 або новіше;
- iOS 10 або новіше;
- не менше 2 ГБ оперативної пам’яті.

3.4 Вимоги до транспортування та зберігання

Програмний продукт може розповсюджуватися:

- через хмарні сервіси;
- за допомогою фізичних носіїв інформації.
- Документація надається у двох форматах:
- електронному;

- паперовому.

Умови експлуатації відповідають стандартним умовам використання технічних засобів.

3.5 Спеціальні вимоги

Спеціальні вимоги до програмного забезпечення відсутні.

---

4 Вимоги до програмної документації

Замовнику повинні бути надані:

- опис програмного продукту;
- технічне завдання;
- інструкція користувача;
- програмне забезпечення у робочому стані.

---

5 Техніко-економічне обґрунтування

Використання розробленого програмного продукту дозволяє досягти наступного ефекту:

– зменшення навантаження на працівників магазину;

– розширення аудиторії клієнтів за рахунок онлайн-доступу;

– можливість масштабування бізнесу;

– спрощення процесу ведення документації;

– оптимізація процесів оформлення та оплати замовлень.

---

6 Стадії розробки

Стадія та період	Етап	Зміст
Технічне завдання, січень	Обґрунтування необхідності розробки	Вибір теми ДП, коротка характеристика ПЗ, вимоги до ПЗ, стадії розробки ПЗ

Ескізний проект, січень-лютий	Створення ескізного проекту	Початкова розробка структури системи, вибір середовища та технологій
Технічний проект, лютий-березень	Створення технічного проекту	Уточнення структури та вибір остаточних технологій розробки, проектування дизайну ПЗ
Робочий проект, квітень	Створення робочого проекту	Програмна реалізація ПЗ, створення застосунку, що відповідає усім нормам
Розробка документації, травень	Створення документації	Створення супровідної документації для проекту, розробка ілюстративних матеріалів
Тестування, травень,	Проведення тестування	Проведення тестування модулів та системи цілком, виправлення виявлених невідповідностей ПЗ
Впровадження, червень	Отримання документів від курівників	Підготовка документації, необхідної для затвердження роботи: відгуки, рецензії.
Захист, червень	Розробка документації	Підготовка до захисту та захист КвР

7 Порядок контролю та приймання

Контроль і приймання програмного продукту здійснюється користувачами вебзастосунку та керівниками проєкту.

Перевірці підлягає:

- функціональність системи;
- відповідність програмного забезпечення поставленим вимогам;
- наявність та повнота документації.

Додаток Б

(Обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

Код класу `ProductController`

```
import {
  Controller,
  Get,
  Param,
  ParseIntPipe
} from '@nestjs/common'
import { ProductService } from './product.service'

@Controller('products')
export class ProductController {
  constructor(private readonly productService: ProductService) {}

  @Get()
  async getAllProducts() {
    return this.productService.getAllProducts()
  }

  @Get(':id')
  async getProductById(@Param('id', ParseIntPipe) id: number) {
    return this.productService.getProductById(id)
  }
}
```

Код класу `ProductService`

```
import { Injectable, NotFoundException } from '@nestjs/common'

@Injectable()
export class ProductService {
  private products = [
    {
      id: 1,
```

```
      name: 'Акустична гітара Yamaha F310',
      category: 'Гітари',
      description: 'Акустична гітара для початківців та любителів.',
      price: 7200
    },
    {
      id: 2,
      name: 'Електрогітара Fender Squier',
      category: 'Гітари',
      description: 'Електрогітара для репетицій та концертів.',
      price: 14500
    },
    {
      id: 3,
      name: 'Синтезатор Casio CT-S200',
      category: 'Клавішні',
      description: 'Компактний синтезатор з набором тембрів.',
      price: 8900
    },
    {
      id: 4,
      name: 'Ударна установка Tama',
      category: 'Ударні',
      description: 'Ударна установка для студії та репетицій.',
      price: 23500
    }
  ]

async getAllProducts() {
    return this.products
  }

  async getProductById(id: number) {
    const product = this.products.find(item => item.id === id)

    if (!product) {
      throw new NotFoundException('Товар не знайдено')
```

```
    }

    return product
  }
}
```

Код класу `ProductModule`

```
import { Module } from '@nestjs/common'
import { ProductController } from './product.controller'
import { ProductService } from './product.service'

@Module({
  controllers: [ProductController],
  providers: [ProductService]
})
export class ProductModule {}
```

Код класу `CartController`

```
import {
  Body,
  Controller,
  Delete,
  Get,
  Param,
  ParseIntPipe,
  Post
} from '@nestjs/common'
import { CartService } from './cart.service'
import { AddToCartDto } from './dto/add-to-cart.dto'

@Controller('cart')
export class CartController {
  constructor(private readonly cartService: CartService) {}

  @Post()
  async addToCart(@Body() dto: AddToCartDto) {
    return this.cartService.addToCart(dto)
  }
```

```
  @Get(':userId')
  async getCart(@Param('userId', ParseIntPipe) userId: number) {
    return this.cartService.getCart(userId)
  }

  @Delete(':id')
  async removeFromCart(@Param('id', ParseIntPipe) id: number) {
    return this.cartService.removeFromCart(id)
  }
}
```

Код класу `CartService`

```
import { Injectable, NotFoundException } from '@nestjs/common'
import { AddToCartDto } from './dto/add-to-cart.dto'

@Injectable()
export class CartService {
  private cart = []

  async addToCart(dto: AddToCartDto) {
    const existingItem = this.cart.find(
      item => item.userId === dto.userId && item.productId === dto.productId
    )

    if (existingItem) {
      existingItem.quantity += dto.quantity
    } else {
      this.cart.push({
        id: this.cart.length + 1,
        userId: dto.userId,
        productId: dto.productId,
        quantity: dto.quantity
      })
    }

    return {
      message: 'Товар додано до кошика'
    }
```

```
  }

  async getCart(userId: number) {
    return this.cart.filter(item => item.userId === userId)
  }

  async removeFromCart(id: number) {
    const index = this.cart.findIndex(item => item.id === id)

    if (index === -1) {
      throw new NotFoundException('Товар у кошику не знайдено')
    }

    this.cart.splice(index, 1)

    return {
      message: 'Товар видалено з кошика'
    }
  }
}
```

Код класу `AddToCartDto`

```
export class AddToCartDto {
  userId: number
  productId: number
  quantity: number
}
```

Код класу `CartModule`

```
import { Module } from '@nestjs/common'
import { CartController } from './cart.controller'
import { CartService } from './cart.service'

@Module({
  controllers: [CartController],
  providers: [CartService]
})
export class CartModule {}
```

Код класу `OrderController`

```
import {
  Body,
  Controller,
  Get,
  Param,
  ParseIntPipe,
  Post
} from '@nestjs/common'
import { OrderService } from './order.service'
import { CreateOrderDto } from './dto/create-order.dto'

@Controller('orders')
export class OrderController {
  constructor(private readonly orderService: OrderService) {}

  @Post()
  async createOrder(@Body() dto: CreateOrderDto) {
    return this.orderService.createOrder(dto)
  }

  @Get(':userId')
  async getUserOrders(@Param('userId', ParseIntPipe) userId: number) {
    return this.orderService.getUserOrders(userId)
  }
}
```

Код класу `OrderService`

```
import { Injectable } from '@nestjs/common'
import { CreateOrderDto } from './dto/create-order.dto'

@Injectable()
export class OrderService {
  private orders = []

  async createOrder(dto: CreateOrderDto) {
    const order = {
      id: this.orders.length + 1,
```

```
      userId: dto.userId,
      customerName: dto.customerName,
      phone: dto.phone,
      address: dto.address,
      total: dto.total,
      status: 'Нове замовлення',
      createdAt: new Date()
    }

    this.orders.push(order)

    return {
      message: 'Замовлення успішно оформлено',
      order
    }
  }

  async getUserOrders(userId: number) {
    return this.orders.filter(order => order.userId === userId)
  }
}
```

Код класу `CreateOrderDto`

```
export class CreateOrderDto {
  userId: number
  customerName: string
  phone: string
  address: string
  total: number
}
```

Код класу `OrderModule`

```
import { Module } from '@nestjs/common'
import { OrderController } from './order.controller'
import { OrderService } from './order.service'

@Module({
  controllers: [OrderController],
  providers: [OrderService]
```

```
})
export class OrderModule {}
```

Код класу `UserController`

```
import {
  Body,
  Controller,
  Get,
  Param,
  ParseIntPipe,
  Post
} from '@nestjs/common'
import { UserService } from './user.service'
import { CreateUserDto } from './dto/create-user.dto'
import { LoginUserDto } from './dto/login-user.dto'

@Controller('users')
export class UserController {
  constructor(private readonly userService: UserService) {}

  @Post('register')
  async register(@Body() dto: CreateUserDto) {
    return this.userService.register(dto)
  }

  @Post('login')
  async login(@Body() dto: LoginUserDto) {
    return this.userService.login(dto)
  }

  @Get(':id')
  async getUserById(@Param('id', ParseIntPipe) id: number) {
    return this.userService.getUserById(id)
  }
}
```

Код класу `UserService`

```
import { Injectable, NotFoundException, UnauthorizedException } from
'@nestjs/common'
```

```
import { CreateUserDto } from './dto/create-user.dto'
import { LoginUserDto } from './dto/login-user.dto'

@Injectable()
export class UserService {
  private users = []

  async register(dto: CreateUserDto) {
    const user = {
      id: this.users.length + 1,
      name: dto.name,
      email: dto.email,
      password: dto.password
    }

    this.users.push(user)

    return {
      message: 'Користувача успішно зареєстровано',
      user
    }
  }

  async login(dto: LoginUserDto) {
    const user = this.users.find(item => item.email === dto.email)

    if (!user || user.password !== dto.password) {
      throw new UnauthorizedException('Невірний email або пароль')
    }

    return {
      message: 'Вхід виконано успішно',
      user
    }
  }

  async getUserById(id: number) {
    const user = this.users.find(item => item.id === id)
```

```
    if (!user) {
      throw new NotFoundException('Користувача не знайдено')
    }

    return user
  }
}
```

Код класу `CreateUserDto`

```
export class CreateUserDto {
  name: string
  email: string
  password: string
}
```

Код класу `LoginUserDto`

```
export class LoginUserDto {
  email: string
  password: string
}
```

Код класу `UserModule`

```
import { Module } from '@nestjs/common'
import { UserController } from './user.controller'
import { UserService } from './user.service'

@Module({
  controllers: [UserController],
  providers: [UserService]
})
export class UserModule {}
```

Код класу `AppModule`

```
import { Module } from '@nestjs/common'
import { ProductModule } from './product/product.module'
import { CartModule } from './cart/cart.module'
import { OrderModule } from './order/order.module'
import { UserModule } from './user/user.module'
```

```
@Module({
  imports: [
    ProductModule,
    CartModule,
    OrderModule,
    UserModule
  ],
  controllers: [],
  providers: []
})
export class AppModule {}
```

Код файлу main.ts

```
import { NestFactory } from '@nestjs/core'
import { ValidationPipe } from '@nestjs/common'
import { AppModule } from './app.module'

async function bootstrap() {
  const app = await NestFactory.create(AppModule)

  app.enableCors()

  app.useGlobalPipes(
    new ValidationPipe({
      whitelist: true,
      forbidNonWhitelisted: true,
      transform: true
    })
  )

  await app.listen(3000)

  console.log('Сервер запущено на http://localhost:3000')
}

bootstrap()
```

Код файлу package.json

```
{
```

```
  "name": "emusics",
  "version": "1.0.0",
  "description": "Веб-застосунок інтернет-магазину музичних інструментів",
  "scripts": {
    "start": "nest start",
    "start:dev": "nest start --watch"
  },
  "dependencies": {
    "@nestjs/common": "^10.0.0",
    "@nestjs/core": "^10.0.0",
    "@nestjs/platform-express": "^10.0.0",
    "reflect-metadata": "^0.1.13",
    "rxjs": "^7.8.1"
  },
  "devDependencies": {
    "@nestjs/cli": "^10.0.0",
     "@nestjs/schematics": "^10.0.0",
    "@nestjs/testing": "^10.0.0",
    "typescript": "^5.0.0"
  }
}
```

Код файлу index.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>MusicStore</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

<header class="header">
  <h1>MusicStore</h1>

  <nav class="nav">
    <a href="index.html">Головна</a>
    <a href="catalog.html">Каталог</a>
```

```
    <a href="cart.html">Кошик</a>
  </nav>
</header>

<section class="hero">
  <h2>Інтернет-магазин музичних інструментів</h2>
  <p>Гітари, клавішні інструменти, ударні установки та аксесуари для музикантів.</p>
  <a href="catalog.html" class="button">Перейти до каталогу</a>
</section>

<section class="section">
  <h2>Популярні категорії</h2>

  <div class="categories">
    <div class="category-card">Гітари</div>
    <div class="category-card">Клавішні</div>
    <div class="category-card">Ударні</div>
    <div class="category-card">Аксесуари</div>
  </div>
</section>

<footer class="footer">
  <p>© 2026 EMusics. Усі права захищені.</p>
</footer>

</body>
</html>
```

Код файлу catalog.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Каталог товарів</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>
```

```
<header class="header">
  <h1>MusicStore</h1>

  <nav class="nav">
    <a href="index.html">Головна</a>
    <a href="catalog.html">Каталог</a>
    <a href="cart.html">Кошик</a>
  </nav>
</header>

<main class="section">
  <h2>Каталог музичних інструментів</h2>

  <input
    type="text"
    id="searchInput"
    class="search"
    placeholder="Пошук товару..."
  >

  <div id="productList" class="products"></div>
</main>

<script src="catalog.js"></script>
</body>
</html>
```

Код файлу cart.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Кошик</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

<header class="header">
```

```
  <h1>EMusics</h1>

  <nav class="nav">
    <a href="index.html">Головна</a>
    <a href="catalog.html">Каталог</a>
    <a href="cart.html">Кошик</a>
  </nav>
</header>

<main class="section">
  <h2>Кошик користувача</h2>

  <div id="cartList" class="cart-list"></div>

  <div class="order-form">
    <h3>Оформлення замовлення</h3>

    <input type="text" id="customerName" placeholder="Ваше ім’я">
    <input type="text" id="phone" placeholder="Телефон">
    <input type="text" id="address" placeholder="Адреса доставки">

    <button onclick="createOrder()" class="button">Оформити замовлення</button>
  </div>
</main>

<script src="cart.js"></script>
</body>
</html>
```

Код файлу style.css

```
* {
  box-sizing: border-box;
}

body {
  margin: 0;
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
```

```
  color: #222;
}

.header {
  background-color: #1f1f1f;
  color: white;
  padding: 18px 40px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

.header h1 {
  margin: 0;
}

.nav a {
  color: white;
  text-decoration: none;
  margin-left: 20px;
  font-size: 16px;
}

.nav a:hover {
  text-decoration: underline;
}

.hero {
  background: linear-gradient(120deg, #242424, #555);
  color: white;
  text-align: center;
  padding: 90px 20px;
}

.hero h2 {
  font-size: 38px;
  margin-bottom: 15px;
}
```

```
.hero p {
  font-size: 18px;
  margin-bottom: 30px;
}

.button {
  display: inline-block;
  background-color: #222;
  color: white;
  padding: 12px 22px;
  border: none;
  border-radius: 5px;
  text-decoration: none;
  cursor: pointer;
}

.button:hover {
  background-color: #444;
}

.section {
  max-width: 1100px;
  margin: 30px auto;
  background-color: white;
  padding: 25px;
  border-radius: 8px;
}

.categories {
  display: grid;
  grid-template-columns: repeat(4, 1fr);
  gap: 20px;
}

.category-card {
  background-color: #eeeeee;
  padding: 30px;
```

```
  text-align: center;
  border-radius: 8px;
  font-weight: bold;
}

.search {
  width: 100%;
  padding: 12px;
  margin: 15px 0 25px 0;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.products {
  display: grid;
  grid-template-columns: repeat(auto-fit, minmax(230px, 1fr));
  gap: 20px;
}

.product-card {
  background-color: #fafafa;
  border: 1px solid #ddd;
  padding: 18px;
  border-radius: 8px;
}

.product-card h3 {
  margin-top: 0;
}

.price {
  font-weight: bold;
  color: #111;
  margin: 10px 0;
}

.cart-list {
  margin-bottom: 25px;
```

```
}

.cart-item {
  background-color: #fafafa;
  border: 1px solid #ddd;
  padding: 15px;
  border-radius: 6px;
  margin-bottom: 12px;
}

.order-form {
  margin-top: 25px;
}

.order-form input {
  width: 100%;
  padding: 12px;
  margin-bottom: 12px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

.footer {
  text-align: center;
  padding: 20px;
  background-color: #1f1f1f;
  color: white;
  margin-top: 40px;
}

@media (max-width: 768px) {
  .header {
    flex-direction: column;
    gap: 15px;
  }

  .categories {
    grid-template-columns: repeat(2, 1fr);
```

```
  }

  .hero h2 {
    font-size: 28px;
  }
}
```

Код файлу catalog.js

```
const products = [
  {
    id: 1,
    name: 'Акустична гітара Yamaha F310',
    category: 'Гітари',
    description: 'Акустична гітара для початківців та любителів.',
    price: 7200
  },
  {
    id: 2,
    name: 'Електрогітара Fender Squier',
    category: 'Гітари',
    description: 'Електрогітара для репетицій та концертів.',
    price: 14500
  },
  {
    id: 3,
    name: 'Синтезатор Casio CT-S200',
    category: 'Клавішні',
    description: 'Компактний синтезатор з набором тембрів.',
    price: 8900
  },
  {
    id: 4,
    name: 'Ударна установка Tama',
    category: 'Ударні',
    description: 'Ударна установка для студії та репетицій.',
    price: 23500
  },
  {
```

```
    id: 5,
    name: 'Мікрофон Shure SM58',
    category: 'Аксесуари',
    description: 'Динамічний мікрофон для вокалу.',
    price: 4800
  }
]

const productList = document.getElementById('productList')
const searchInput = document.getElementById('searchInput')

function renderProducts(items) {
  productList.innerHTML = ''

  items.forEach(product => {
    const card = document.createElement('div')
    card.className = 'product-card'

    card.innerHTML = `
      <h3>${product.name}</h3>
      <p>${product.category}</p>
      <p>${product.description}</p>
      <p class="price">${product.price} грн</p>
      <button class="button" onclick="addToCart(${product.id})">
        Додати до кошика
      </button>
    `

    productList.appendChild(card)
  })
}

function addToCart(productId) {
  const product = products.find(item => item.id === productId)
  let cart = JSON.parse(localStorage.getItem('cart')) || []

  const existingProduct = cart.find(item => item.id === productId)
```

```
  if (existingProduct) {
    existingProduct.quantity += 1
  } else {
    cart.push({
      ...product,
      quantity: 1
    })
  }

  localStorage.setItem('cart', JSON.stringify(cart))
  alert('Товар додано до кошика')
}

searchInput.addEventListener('input', () => {
  const value = searchInput.value.toLowerCase()

  const filteredProducts = products.filter(product =>
    product.name.toLowerCase().includes(value) ||
    product.category.toLowerCase().includes(value)
  )

  renderProducts(filteredProducts)
})

renderProducts(products)
```

Код файлу cart.js

```
const cartList = document.getElementById('cartList')

function renderCart() {
  const cart = JSON.parse(localStorage.getItem('cart')) || []

  cartList.innerHTML = ''

  if (cart.length === 0) {
    cartList.innerHTML = '<p>Кошик порожній</p>'
    return
  }
```

```
  cart.forEach(item => {
    const div = document.createElement('div')
    div.className = 'cart-item'

    div.innerHTML = `
      <h3>${item.name}</h3>
      <p>Ціна: ${item.price} грн</p>
      <p>Кількість: ${item.quantity}</p>
      <p>Сума: ${item.price * item.quantity} грн</p>
      <button class="button" onclick="removeFromCart(${item.id})">
        Видалити
      </button>
    `

    cartList.appendChild(div)
  })
}

function removeFromCart(productId) {
  let cart = JSON.parse(localStorage.getItem('cart')) || []

  cart = cart.filter(item => item.id !== productId)

  localStorage.setItem('cart', JSON.stringify(cart))
  renderCart()
}

function createOrder() {
  const cart = JSON.parse(localStorage.getItem('cart')) || []

  const customerName = document.getElementById('customerName').value
  const phone = document.getElementById('phone').value
  const address = document.getElementById('address').value

  if (!customerName || !phone || !address) {
    alert('Заповніть усі поля')
    return
```

```
  }

  if (cart.length === 0) {
    alert('Кошик порожній')
    return
  }

  localStorage.removeItem('cart')

  alert('Замовлення успішно оформлено')

  window.location.href = 'index.html'
}

renderCart()
```

Код файлу product.html

```
<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <title>Товар | MusicStore</title>
  <link rel="stylesheet" href="style.css">
</head>
<body>

<header class="header">
  <h1>MusicStore</h1>

  <nav class="nav">
    <a href="index.html">Головна</a>
    <a href="catalog.html">Каталог</a>
    <a href="cart.html">Кошик</a>
  </nav>
</header>

<main class="section">
  <div id="productDetails" class="product-details"></div>
```

```
</main>

<script src="product.js"></script>
</body>
</html>
```

Код файлу product.js

```
const products = [
  {
    id: 1,
    name: 'Акустична гітара Yamaha F310',
    category: 'Гітари',
    description: 'Акустична гітара для початківців та любителів. Має зручний корпус, якісне звучання та підходить для навчання.',
    price: 7200
  },
  {
    id: 2,
    name: 'Електрогітара Fender Squier',
    category: 'Гітари',
    description: 'Електрогітара для репетицій, студійного запису та концертних виступів.',
    price: 14500
  },
  {
    id: 3,
    name: 'Синтезатор Casio CT-S200',
    category: 'Клавішні',
    description: 'Компактний синтезатор з набором тембрів та зручним керуванням.',
    price: 8900
  },
  {
    id: 4,
    name: 'Ударна установка Tama',
    category: 'Ударні',
    description: 'Ударна установка для студійної роботи, репетицій та живих виступів.',
    price: 23500
```

```
  },
  {
    id: 5,
    name: 'Мікрофон Shure SM58',
    category: 'Аксесуари',
    description: 'Динамічний мікрофон для вокалу з якісною передачею звуку.',
    price: 4800
  }
]

const productDetails = document.getElementById('productDetails')
const params = new URLSearchParams(window.location.search)
const productId = Number(params.get('id'))

const product = products.find(item => item.id === productId)

if (!product) {
  productDetails.innerHTML = '<p>Товар не знайдено</p>'
} else {
  productDetails.innerHTML = `
    <h2>${product.name}</h2>
    <p><strong>Категорія:</strong> ${product.category}</p>
    <p>${product.description}</p>
    <p class="price">${product.price} грн</p>
    <button class="button" onclick="addToCart(${product.id})">
      Додати до кошика
    </button>
  `
}

function addToCart(productId) {
  const product = products.find(item => item.id === productId)
  let cart = JSON.parse(localStorage.getItem('cart')) || []

  const existingProduct = cart.find(item => item.id === productId)

  if (existingProduct) {
    existingProduct.quantity += 1
```

```
  } else {
    cart.push({
      ...product,
      quantity: 1
    })
  }

  localStorage.setItem('cart', JSON.stringify(cart))
  alert('Товар додано до кошика')
}
```

ДОДАТОК В

(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Рисунок В.1 – слайд «Титульна сторінка»

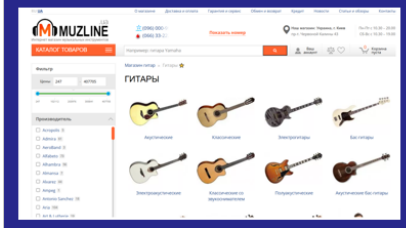
Рисунок В.2 – слайд «Актуальність кваліфікаційної роботи»

Рисунок В.3 – слайд «Мета і завдання»

Рисунок В.4 – слайд «Аналіз предметної області»

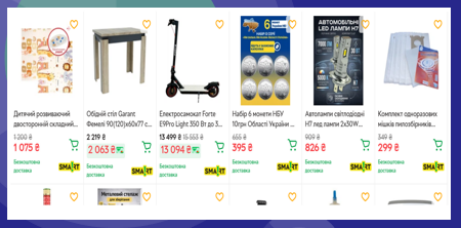
АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Під час аналізу предметної області було розглянуто сучасні інтернет-магазини музичних інструментів, які мають схожий функціонал із розроблюваним вебзастосунком.



Muzline
- зручний каталог товарів;
- система пошуку та фільтрації;
- детальний опис продукції.

JAM
- сучасний інтерфейс;
- підтримка кошика та авторизації;
- адаптивний дизайн сайту.



Rozetka
- швидкий пошук товарів;
- зручна навігація;
- можливість онлайн-замовлення.

Рисунок В.5 – слайд «Аналіз існуючих рішень»

ФУНКЦІОНАЛЬНІ І НЕФУНКЦІОНАЛЬНІ РІШЕННЯ

Функціональні:

1. Неавторизовані користувачі:
- Перегляд каталогу музичних інструментів
- Пошук товарів за категоріями
- Перегляд характеристик товарів

2. Авторизовані користувачі:
- Додавання товарів до кошика
- Оформлення замовлення
- Перегляд власних замовлень

3. Адміністратор:
- Додавання та редагування товарів
- Управління категоріями
- Перегляд замовлень користувачів

Нефункціональні:

- Зручний та сучасний інтерфейс
- Швидка робота вебзастосунку
- Безпечна авторизація користувачів
- Надійне збереження даних у MySQL
- Адаптивність для різних пристроїв

Рисунок В.6 – слайд «Функціональні, нефункціональні рішення»

Рисунок В.7 – слайд «Проектування модулів і даних»4

Рисунок В.8 – слайд «Аналіз та вибір технологій»

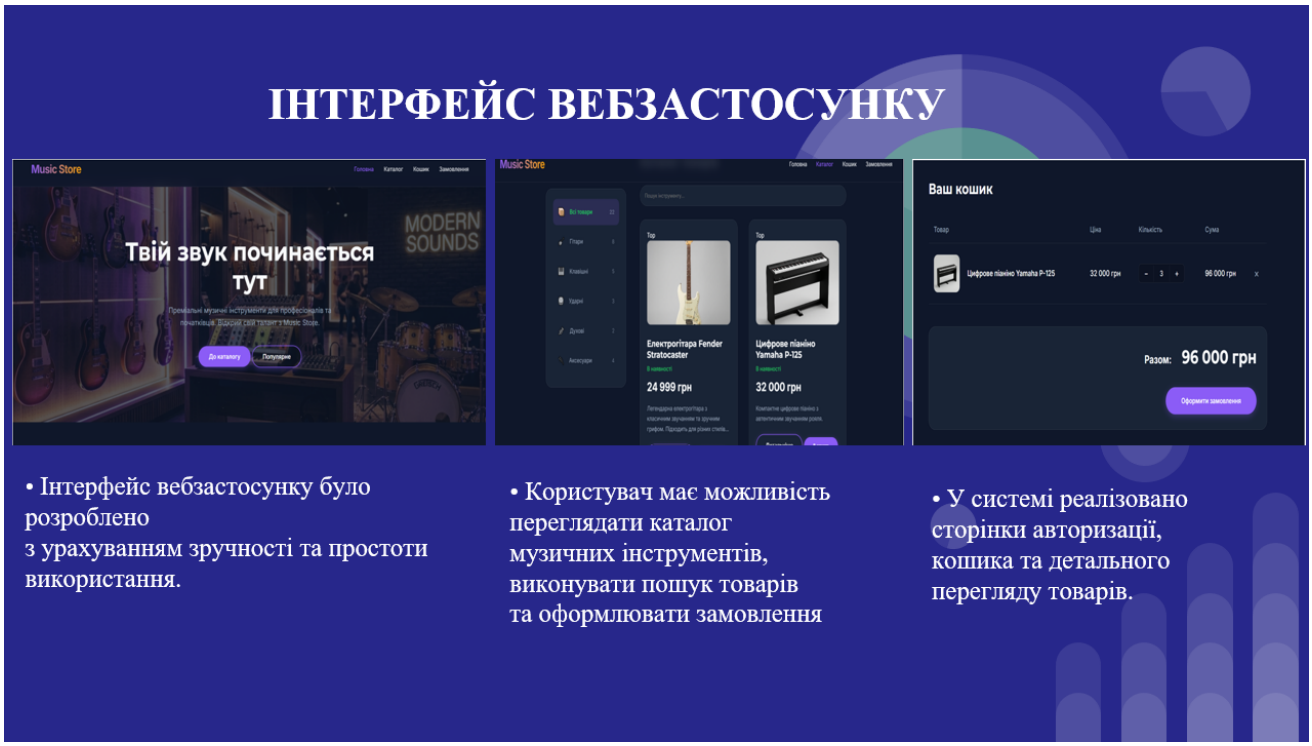


Рисунок В.9 – слайд «Інтерфейс вебзастосунку»

Рисунок В.10 – слайд «UML-Діаграма варіантів використання»

Рисунок В.11 – слайд «Загальна модель взаємодії »

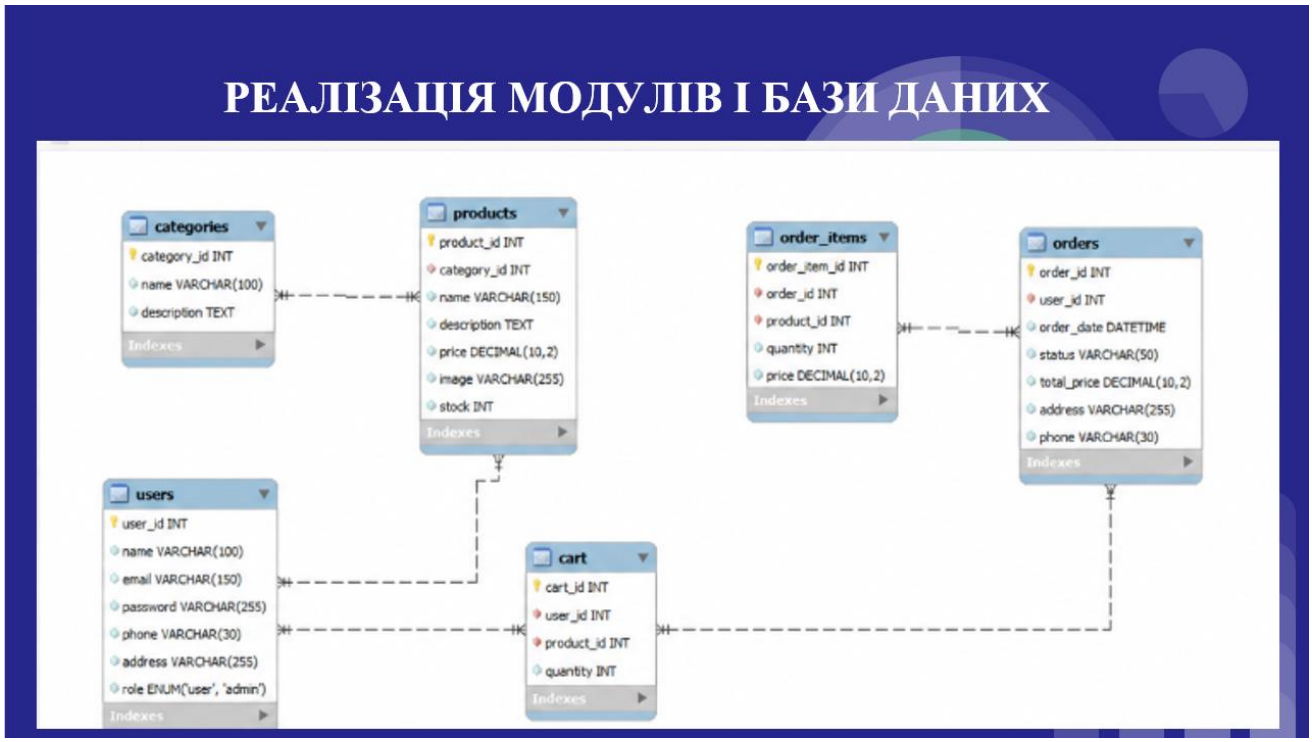


Рисунок В.12 – слайд «Реалізація модюлів і бази даних »

ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

• Після завершення розробки вебзастосунку було проведено тестування основних функцій інтернет-магазину музичних інструментів.

• Тестування виконувалося на основі реальних сценаріїв використання сайту звичайним користувачем та адміністратором системи.

• Було перевірено:
– роботу реєстрації та авторизації користувачів;
– коректність відображення каталогу музичних інструментів;
– пошук товарів за категоріями;
– перегляд сторінки товару;
– додавання товарів до кошика;
– зміну кількості товарів у кошику;
– оформлення замовлення;
– взаємодію вебзастосунку з базою даних MySQL.

• Під час тестування було виявлено та виправлено окремі помилки інтерфейсу та взаємодії між клієнтською і серверною частинами.

• Результати тестування підтвердили стабільну роботу вебзастосунку, зручність використання системи та відповідність поставленим вимогам.

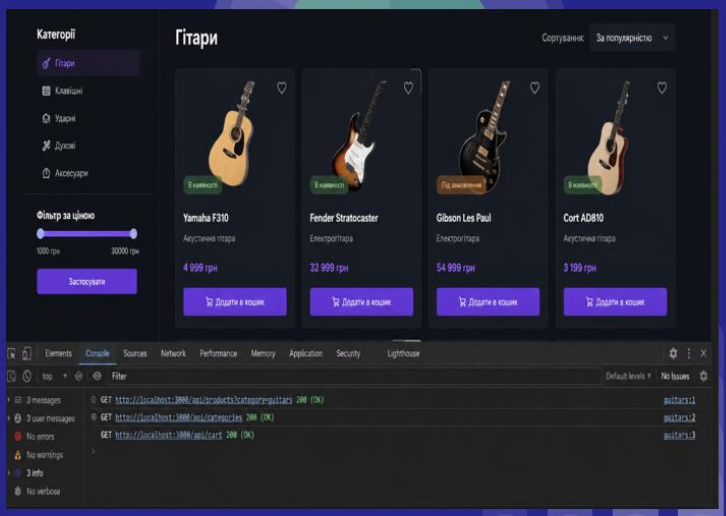


Рисунок В.13 – слайд «Тестування програмного забезпечення»

ВИСНОВКИ

Таблиця порівняння:

Аналіз предметної області та існуючих аналогів	Виконано
Визначення функціональних вимог	Виконано
Розробка структури та архітектури системи	Виконано
Створення бази даних	Виконано
Реалізація клієнтської та серверної частини	Виконано
Реєстрація та авторизація користувачів	Виконано
Тестування системи	Виконано

Підсумковий коментар:

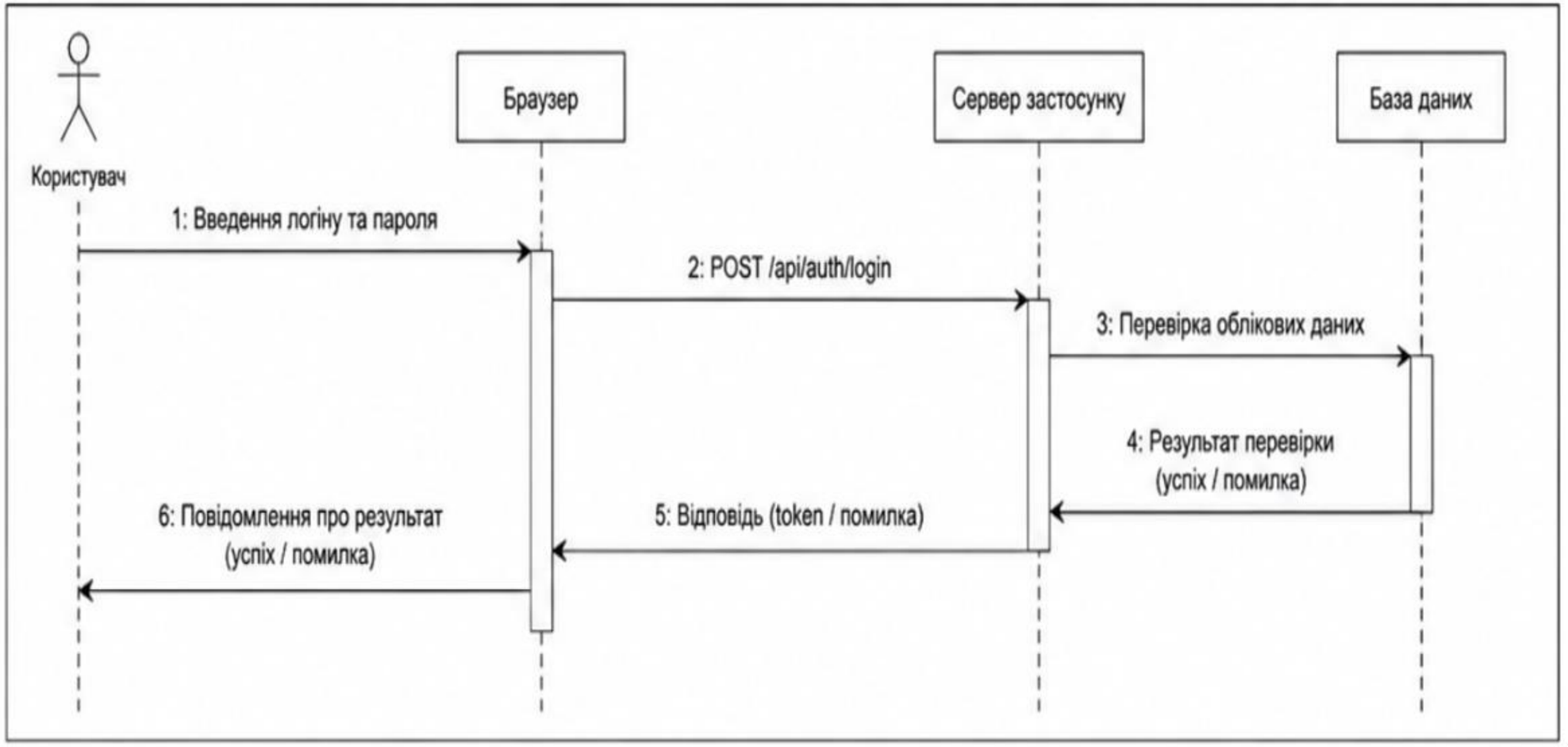
У результаті виконання кваліфікаційної роботи було створено вебзастосунок для продажу музичних інструментів, який забезпечує зручний перегляд товарів, роботу з кошиком, оформлення замовлень та взаємодію з базою даних.

ф

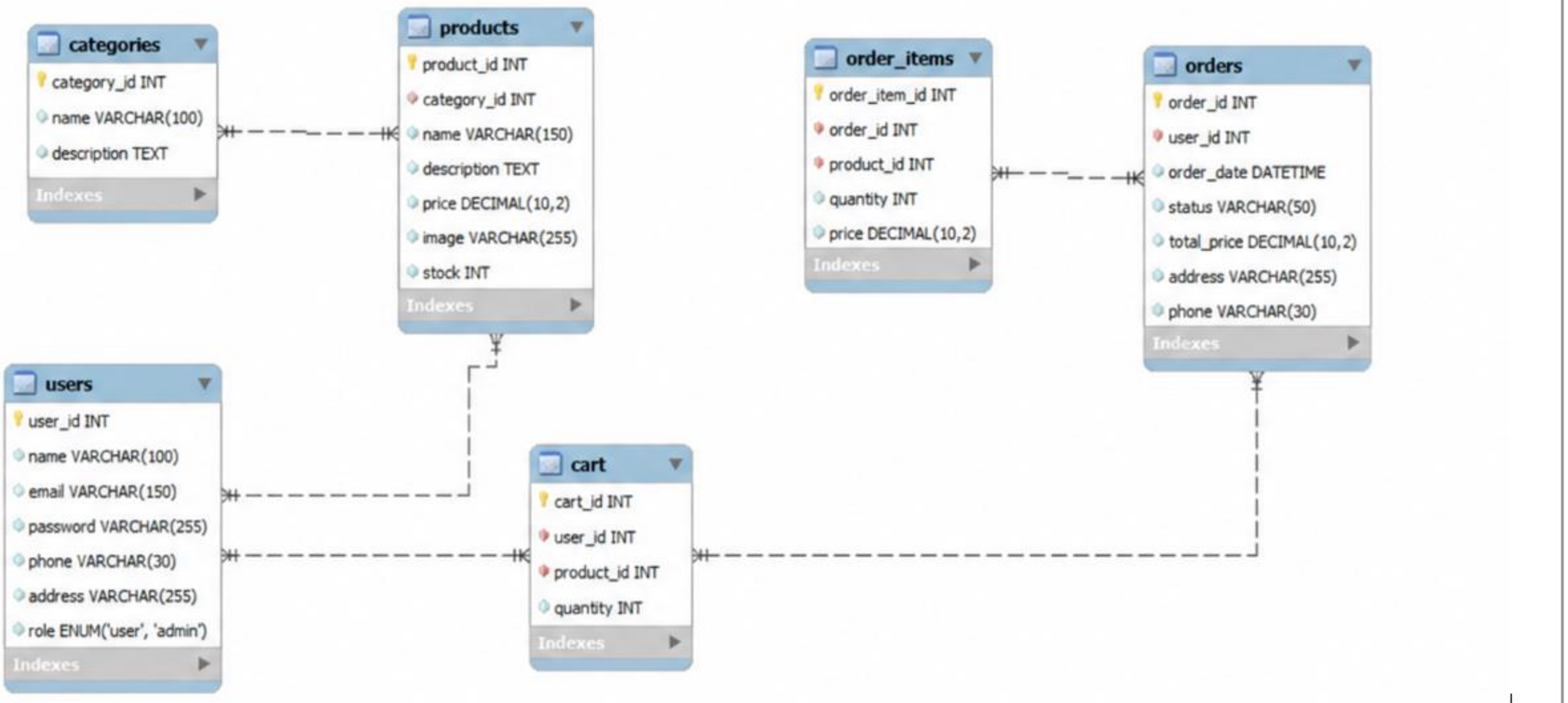
Рисунок В.14 – слайд «Висновки»

Рисунок В.15 – слайд «Дякуюза увагу»

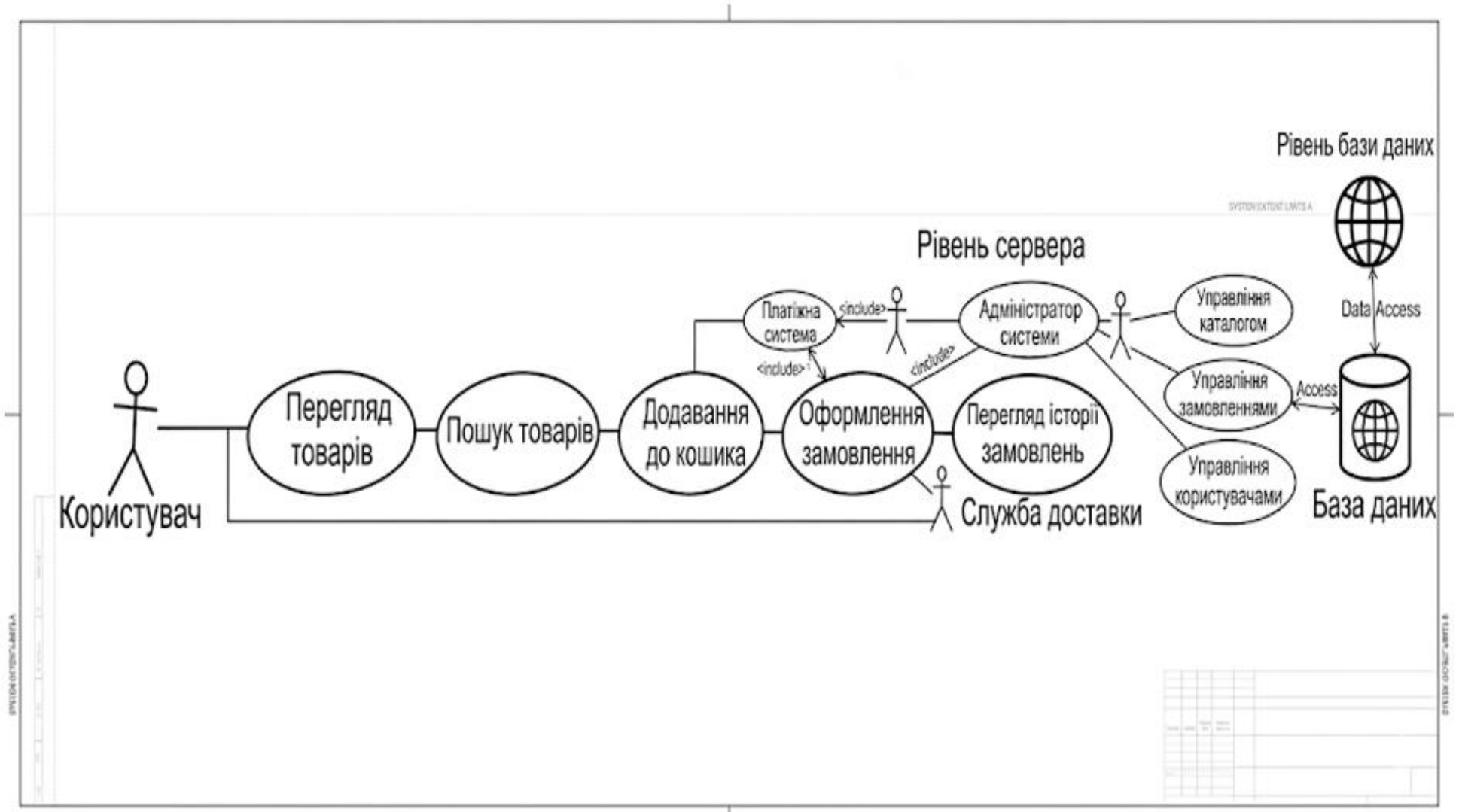
ГРАФІЧНА ЧАСТИНА



					КвРІПЗ.2201110.01.15ПЗ		
					Вебсайт інтернет магазину з продажу музичних інструментів	Літера	Маса	Масштаб
Зм.	Арк.	№ докум.	Підпис	Дата				
Розробив		Стефановський			Діаграма структури вебсайту інтернет-магазину			
Керівник		Форкун Ю.В.				Арк.3	Аркушів 3	
Консульт.								
Н. Контр.		Яшина О.М.				ХНУ, ІПЗ-22-1		
Зав. каф.		Бедратюк Л. П.						



					КвРІПЗ.2201110.01.15 ПЗ		
Зм.	Арк.	№ докум.	Підпис	Дата	Вебсайт інтернет магазину з продажу музичних інструментів ER-Діаграма бази даних	Літера	Маса / Масштаб
Розробив		Стефановський					
Керівник		Форкун Ю.В.				Арк 2	Аркушів 3
Консульт.							
Н. Контр.		Яшина О.М			ХНУ, ІПЗ-22-1		
Зав. каф.		Бекарюк Л. П.					



					КвРІПЗ.2201110.01.15.ПЗ	
Зм.	Арк.	№ докум.	Підпис	Дата	Вебсайт інтернет магазину з продажу музичних інструментів	Літера / Маса / Масштаб
Розробив		Стефановський				
Керівник		Федорак Ю.В.			UML-діаграма варіантів використання	Арк. 1 / Аркушів 3
Консульт.						
Н. Контр.		Яшина О.М.				ХНУ, ІПЗ-22-1
Зав. каф.		Бакалова Л.П.				

СУПРОВІДНІ ДОКУМЕНТИ

Завідувачу кафедри інженерії програмного
забезпечення проф. Леоніду БЕДРАТЮКУ
здобувача вищої освіти
Стефановського Андрія Анатолійовича
факультет ІТ, IVкурс, група ІПЗ-22-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності в Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії Хмельницького національного університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-обчислювального комплексу StrikePlagiarism та/або програмно-технічного засобу AntiPlagiarism і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення текстових збігів у роботах.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

28.05
дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Стефановський Андрій Анатолійович

Тема Вебсайт інтернет-магазину з продажу музичних інструментів

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3; кількість сторінок записки 74;
1 У кваліфікаційній роботі досліджено та проаналізовано предметну область інтернет-магазинів музичних інструментів, визначено функціональні та нефункціональні вимоги до вебзастосунку. Проведено аналіз існуючих аналогів, розглянуто їх переваги та недоліки, а також обґрунтовано актуальність розробки нового програмного забезпечення. Виконано проєктування структури вебзастосунку, бази даних та інтерфейсу користувача. Для реалізації проєкту використано сучасні технології веброзробки. Також проведено тестування програмного забезпечення, результати якого підтвердили коректність роботи системи та її готовність до експлуатації.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі обґрунтовано актуальність теми, визначено мету, об'єкт, предмет та завдання кваліфікаційної роботи. У першому розділі проведено аналіз предметної області продажу музичних інструментів через мережу Інтернет, розглянуто існуючі програмні рішення та сформовано вимоги до вебзастосунку. У другому розділі виконано проєктування системи, розроблено структуру бази даних, архітектуру вебзастосунку та макети інтерфейсу користувача. У третьому розділі описано програмну реалізацію клієнтської та серверної частин системи, роботу з базою даних, API-маршрути, кошик покупця та механізм оформлення замовлень. Також проведено тестування вебзастосунку та підтверджено коректність його роботи.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки електронна комерція продовжує активно розвиватися, а продаж музичних інструментів через Інтернет потребує зручних та функціональних програмних рішень. Позитивною стороною роботи є комплексний підхід до створення вебзастосунку, який включає реалізацію каталогу товарів, системи авторизації користувачів, кошика покупця, оформлення замовлень та бази даних.У роботі використано сучасні технології веброзробки, а також проведено тестування, яке підтвердило працездатність основних функцій.

5. Негативні сторони роботиУ роботі основну увагу приділено реалізації базових функцій інтернет-магазину музичних інструментів. Подальший розвиток проєкту може передбачати впровадження системи онлайн-оплати, відгуків і рейтингів товарів, персональних рекомендацій для користувачів та інтеграції зі службами доставки. Також можливе розширення адміністративної панелі та додавання додаткових засобів аналітики продажів.

6. Оцінка графічного оформлення та пояснювальної записки.Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та представлено у вигляді UML-діаграм, схем бази даних, макетів інтерфейсу та результатів тестування вебзастосунку. Пояснювальна записка оформлена відповідно до встановлених вимог та повністю відображає зміст виконаної роботи.

7. Відгук про кваліфікаційну роботу в цілому.Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал викладено логічно, послідовно та зрозуміло. У роботі розглянуто всі основні етапи створення вебзастосунку: аналіз предметної області, проєктування, програмна реалізація та тестування. Результатом виконання роботи є працездатний вебзастосунок для продажу музичних інструментів, який має практичне значення та може бути використаний як основа для створення повноцінного інтернет-магазину.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗНТ Гладчук Є.Т., д.т.н., проф. каф. КІС

"____" _____ 2026_ р. _____

(підпис)

(підпис)

Протокол аналізу звіту подібності науковим керівником

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Андрій СТЕФАНОВСЬКИЙ

Співавтор:

Назва: Вебсайт інтернет-магазину з продажу музичних інструментів Назва теми

Науковий керівник: канд. техн. наук, доцент Юрій ФОРКУН

Підрозділ: Кафедра інженерії програмного забезпечення

Коефіцієнт подібності 1:10.14%

Коефіцієнт подібності 2:1.06%

Мікропробіли: 21

Заміна букв: 0

Інтервали: 0

Білі знаки: 20

Дата створення звіту: 2026-06-03 13:37:10.0

Після аналізу Звіту подібності констатую наступне:

☑ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата 03.06.26 експерт (Форкун Ю.В.)

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатами звіту/звітів перевірки роботи, продукованими програмно-технічним засобом (ами), на наявність текстових збігів.
Назва кваліфікаційної роботи: «Вебсайт інтернет-магазину з продажу музичних інструментів»
Автор: Стефановський Андрій Анатолійович
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Спеціальність: 121 – Інженерія програмного забезпечення
Науковий керівник: Форкун Юрій Вікторович, кандидат технічних наук, доцент
Після аналізу звіту/звітів зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається до захисту.	**відповідає**
2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована.	
3	Виявлені запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Виявлені запозичення частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнуті. Робота може бути допущена до захисту після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системою перевірки на плагіат виявлено схожість з окремими документами переважно в частині загальновживаних обов'язкових словосполучень у стандартних бланках, структурі змісту, назвах розділів і підрозділів, рамках форм, а також у назвах та URL-адресах публікацій переліку джерел посилання;

2) запозичення, виявлені у тексті роботи, мають фрагментарний характер і не впливають на самостійність виконання кваліфікаційної роботи.

Максимальний обсяг запозичень, визначений системою Anti-Plagiarism, складає 2.0% з одного джерела. Загальна сумарна подібність у базі даних складає 3% за символами та 10% за лексемами.

Крім того, за результатами додаткового аналізу коефіцієнт подібності 1 становить 10.14%, коефіцієнт подібності 2 – 1.06%. Заміни букв, маніпуляцій з інтервалами та зайвих білих знаків не виявлено. Виявлені мікропробіли у кількості 21 мають технічний характер і не свідчать про навмисне спотворення тексту. З урахуванням наведених обґрунтувань, результати перевірки відповідають характеру теми і свідчать на користь самостійності виконання кваліфікаційної роботи.

Дата 9.06.26

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Юрій ФОРКУН