МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

Виконав: студент групи 3157ст

_____Жулай В.Ю.

(підпис, ПІБ)

Керівник роботи:

_____Доцент кафедри ПЗАС, к.т.н.

(посада, науковий ступень вчене звання)

_____Пухалевич А.В.

(підпис, ПІБ)

Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту: Жулай Владиславу Юрійовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи:розробка вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

Керівник роботи доцент кафедри ПЗАС, к.т.н. Пухалевич Андрій Володимирович
Затверджені наказом ректора №257-уч від «17» березня 2023 р.

2. Термін подання роботи: 29.05.2023 р.

3. Вихідні дані по роботі:_____

_____

4. Перелік питань, що належать до розробки (найменування розділів)

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1. Титульний аркуш; 2. Актуальність теми; 3. Мета розробки; 4. Фугкції для користувача та адміністратора; 5.Аналіз вимог. Діаграма варіантів використання; 6. Розробка архітектури; 7. Ескізний проект. Діаграми діяльності(діаграма діяльності для варіанту використання «Оформити замовлення»); 8. Ескізний проект. Діаграми діяльності(діаграма діяльності для варіанту використання «Редагувати інформацію про товар»); 9. Ескізний проект. Діаграми діяльності(діаграма діяльності для варіанту використання «Автентифікація»); 10. Ескізний проект. Концептуальна модель даних; 11. Технічний проект. Діаграма класів; 12. Технічний проект. Діаграми послідовності(Діаграма послідовності для прецеденту «Додати товар»); 13. Технічний проект. Діаграми послідовності(Діаграма послідовності для прецеденту «Додати товар до кошика»); 14. Технічний проект. Діаграми послідовності(Діаграма послідовності для прецеденту «Пошук товару»); 15 - 19.Робочий проект; 20. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 20.03.2023 р.

КАЛЕНДАРНИЙ ПЛАН

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	24.03.2023	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	31.03.2023	ПП*
3.	Аналіз вимог до ПЗ	07.04.2023	ПП*
4.	Розробка архітектури ПЗ	21.04.2023	
5.	Розробка проекту ПЗ	05.05.2023	
6.	Реалізація та тестування ПЗ	12.05.2023	
7.	Підготовка розділу Результати розробки ПЗ	17.05.2023	
8.	Підготовка розділу з охорони праці	19.05.2023	ПП*
9.	Підготовка розділу ВИСНОВКИ	24.05.2023	
10.	Оформлення списку використаних джерел та додатків	26.05.2023	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	29.05.2023	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	02.06.2023	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	05.06.2023	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	12.06.2023	

* - за результатами переддипломної практики (ПП), яка була з 01.02.2023 до 19.03.2023 р.

Студент _____ Жулай В.Ю.
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А.В.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебсервісу для магазину з продажу музичних інструментів «MusicBrand».

Об’єктом кваліфікаційної роботи є процес розробки вебсервісу для магазину з продажу музичних інструментів «MusicBrand». Проведено аналіз практичної задачі та проаналізовано існуючі аналоги та методи розробки вебсервісів. Було розроблено проект вебсервісу, а також реалізовано вебсервіс для магазину з продажу музичних інструментів «MusicBrand».

Розробку вебсервісу було виконано за допомогою середовища розробки Intelijj IDEA, мови програмування Java та JavaScript, використано СУБД MySQL.

Кваліфікаційна робота викладена на 120 сторінках друкованого тексту, містить 35 рисунків, 12 таблиць, список використаних джерел із 11 найменувань та 5 додатків.

Кваліфікаційна робота виконана українською мовою.

ABSTRACT

Qualification work is devoted to solving the practical software engineering problem characterized by complexity and uncertainty, namely the development of a web service for a musical instrument store "MusicBrand".

The object of the qualification work is the process of developing a web service for a musical instrument store "MusicBrand". An analysis of the subject area was carried out and existing analogues and methods of web service development were analyzed. A web service project was developed, and a web service for a musical instrument store "MusicBrand" was implemented.

The web service was developed using the Intelijj IDEA development environment, the Java and JavaScript programming languages, and the MySQL DBMS was used.

Qualification work is laid out on 120 pages of printed text, contains 35 figures, 12 tables, a list of used sources of 11 items and 5 appendices.

Qualifying work was written in Ukrainian.

ЗМІСТ

ВСТУП 8

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND» 10

1.1 Аналіз практичної задачі інженерії програмного забезпечення 10

1.2 Опис існуючих технологій розробки 12

1.3 Аналіз існуючих аналогів та обґрунтування необхідності розробки .. 17

1.4 Постановка задачі на розробку вебсервісу для магазину з продажу музичних інструментів 24

2 РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND» 27

2.1 Аналіз вимог до вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 27

2.1.1 Побудова моделі варіантів використання вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 27

2.1.2 Специфікації варіантів використання вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 30

2.2 Розробка архітектури вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 40

2.3 Проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 43

2.3.1 Ескізний проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 44

2.3.2 Технічний проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 51

2.3.3 Робочий проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 64

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND» 73

4 ОХОРОНА ПРАЦІ 75

2.1 Основні положення 75

2.2 Основні законодавчі та нормативно правові акти про охорону праці в Україні 75

2.3 Аналіз небезпечних і шкідливих факторів при роботі з комп’ютером 77

2.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів при роботі з комп’ютером 79

ВИСНОВКИ 81

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 82

ДОДАТОК А – Технічне завдання на розробку вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 83

ДОДАТОК Б – Програма та методика випробувань вебсервісу для магазину з

продажу музичних інструментів «MusicBrand»..... 89

ДОДАТОК В – Інструкція користувача вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 92

ДОДАТОК Г – Текст вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 97

ДОДАТОК Д – Опис вебсервісу для магазину з продажу музичних інструментів «MusicBrand» 117

ВСТУП

У сучасному світі електронна комерція є невід'ємною частиною життя людей, і вебсервіси стають все більш популярними як засіб купівлі та продажу товарів та послуг. З кожним роком все більше людей воліють робити покупки онлайн замість того, щоб витрачати час і сили на відвідування фізичних магазинів. Крім того, швидкий темп життя та ритм сучасного суспільства змушують людей шукати нові способи економії часу, такі як онлайн-шопінг. У цьому контексті розробка вебсервісу для магазину продажу музичних інструментів є актуальною темою для дослідження та розробки.

Розробка вебсервісу для магазину продажу музичних інструментів є важливим завданням, оскільки сучасний ринок вимагає від компаній виходити в онлайн-сферу та пропонувати клієнтам зручний та швидкий спосіб придбання товарів. Створення функціонуючого вебсервісу дозволить магазину розширювати свій бізнес та залучати нових клієнтів, а також збільшувати виручку та покращувати задоволеність наявної клієнтської бази.

Наразі доступні готові рішення вебсервісів, але є досить вагомі причини, чому використання таких рішень може не бути оптимальним. Більшість таких програмних продуктів в цій галузі мають високу вартість, а також в більшості випадків реалізовують функціонал, який потрібний тільки для конкретних підприємств. Безкоштовні аналоги часто не відповідають сучасним вимогам і можуть бути невдало спроектовані для оптимізації.

Таким чином, успішна розробка вебсервісу для магазину продажу музичних інструментів «MusicBrand» дозволить збільшити конкурентоспроможність компанії та покращити зручність покупки для клієнтів.

Мета роботи полягає у розв’язанні практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та

невизначеністю умов, а саме у створенні вебсервісу, який дозволить клієнтам купувати музичні інструменти онлайн.

Для досягнення мети роботи потрібно вирішити наступні завдання:

- провести дослідження предметної області;

- провести аналіз існуючих аналогів вебсервісів продажу музичних інструментів;

- провести аналіз методів розробки вебсервісів;

- створити проект програмного забезпечення;

- провести тестування програмного забезпечення;

- ознайомитися з питаннями охорони праці при розробці ПЗ та роботі з комп’ютером.

Об’єктом роботи є розробка вебсервісу для магазину з продажу музичних інструментів «MusicBrand».

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ПРОГРАМНОЇ ІНЖЕНЕРІЇ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Музика є невід'ємною частиною життя людей з давніх-давен. Музичні інструменти використовуються для самовираження, творчості та розваги. Ринок музичних інструментів постійно зростає і багато людей шукають можливості купити якісний інструмент. У той же час, у багатьох музикантів і музикантів-початківців немає часу на відвідування магазинів музичних інструментів та вивчення всього асортименту. Вебсервіс, що дозволяє придбати музичний інструмент онлайн, може вирішити цю проблему та задовольнити потреби клієнтів.

У цій кваліфікаційній роботі розглядається розробка вебсервісу для магазину з продажу музичних інструментів «MusicBrand». Ця задача відноситься до практичної задачі програмної інженерії, що характеризується комплексністю та невизначеністю умов. Це завдання є актуальним, тому що в даний час все більше людей воліють купувати товари через інтернет, у тому числі музичні інструменти.

Магазин із продажу музичних інструментів працює за принципом роздрібної торгівлі. Магазин займає невелику територію, на якій розташовані вітрини з інструментами та музичними аксесуарами.

Клієнти можуть зайти до магазину та подивитися на товари, а також протестувати інструменти у спеціальній зоні для прослуховування. Якщо клієнту сподобається інструмент, він може придбати його на місці, а якщо ні – попросити консультацію у продавця.

У цій роботі передбачається створення функціонального та зручного у використанні вебсервісу, який дозволить клієнтам магазину швидко та зручно підібрати та придбати необхідний їм музичний інструмент.

Одним із ключових аспектів розробки вебсервісу є забезпечення безпеки даних клієнтів та збереження конфіденційності інформації, яку вони надають при купівлі товарів. Для цього необхідно реалізувати механізми автентифікації та авторизації.

Інший важливий аспект розробки вебсервісу для магазину з продажу музичних інструментів «MusicBrand» – забезпечення його масштабованості. Магазин музичних інструментів може швидко зростати і залучати все більше клієнтів, тому необхідно переконатися, що вебсервіс зможе підтримувати велику кількість користувачів та операцій без втрати продуктивності та якості обслуговування.

Таким чином, дана кваліфікаційна робота спрямована на розв’язання практичної задачі програмної інженерії, що характеризується комплексністю та невизначеністю умов, а саме у розробці вебсервісу для магазину продажу музичних інструментів з урахуванням особливостей даної предметної галузі. Розробка даного вебсервісу повинна забезпечити клієнтам магазину зручність та безпеку при купівлі музичних інструментів в інтернеті, а також забезпечити ефективне керування обліком товарів.

Необхідно в процесі розробки створити програмний код, який реалізовуватиме функціональність вебсервісу, а саме код, котрий реалізовуватиме пошук та вибір музичних інструментів, додавання та видалення товарів з кошика, оформлення замовлень тощо.

Після написання коду необхідно здійснити тестування вебсервісу. Тестування дозволяє виявити помилки та недоліки у роботі системи, та переконатися у її коректній роботі у різних сценаріях використання. У цій роботі необхідно провести функціональне тестування, інтеграційне тестування та тестування продуктивності.

При розробці вебсервісу для магазину з продажу музичних інструментів необхідно враховувати специфічні вимоги цієї галузі. Наприклад, магазин може мати великий асортимент товарів, різні категорії товарів, а також різні параметри для кожного товару, такі як бренд, тип, розмір тощо.

У зв'язку з цим вебсервіс повинен забезпечувати можливість зручного та швидкого пошуку товарів, фільтрації за різними параметрами, можливість вибору відповідних послуг. Вебсервіс також повинен забезпечувати зручний інтерфейс для клієнтів, який дозволить їм швидко знаходити необхідну інформацію та здійснювати покупки.

В цілому, розробка вебсервісу для магазину з продажу музичних інструментів «MusicBrand» є складним та багатогранним завданням, яке потребує використання різних методів та інструментів інженерії програмного забезпечення. Кваліфікаційна робота, присвячена цьому завданню, передбачає собою розробку вебсервісу, котрий відповідатиме вимогам магазину та забезпечуватиме зручність та безпеку для його клієнтів.

1.2 Опис існуючих технологій розробки вебсервісів

Існує безліч рішень для створення і організації роботи вебсервісів, нижче приведені найпопулярніші з них:

- Розробка з використанням CMS;
- Платформи SaaS;
- Розробка з використанням фреймворків.

Розробка з використанням CMS

CMS – це система керування вмістом, яка надає інструменти для створення, керування та публікації контенту на вебсайті без необхідності програмування.

Переваги використання CMS для створення вебсервісів:

- Зручність використання. CMS надає простий та інтуїтивно зрозумілий інтерфейс для керування вмістом на вебсайті, включаючи додавання, редагування та видалення вмісту.
- . Це дозволяє користувачам без технічних знань керувати сайтом без необхідності звертатися до розробників.
- Гнучкість. CMS дозволяє створювати вебсервіси різної складності, починаючи від простих блогів та закінчуючи складними корпоративними сайтами.
- Швидкість розробки. CMS надає готові шаблони та компоненти, які спрощують процес створення сторінок та дозволяють скоротити час розробки.
- Розширюваність. CMS надає можливість встановлення плагінів та додаткових модулів, що дозволяє розширювати функціональність вебсервісу без необхідності зміни вихідного коду.

Недоліки використання CMS для створення вебсервісів:

- Обмеження. CMS надає готові шаблони та компоненти, які можуть обмежувати гнучкість та оригінальність вебсайту. Крім того, обмежений доступ до вихідного коду може створювати проблеми за необхідності виконання специфічних завдань.
- Безпека: CMS, як і будь-яке інше програмне забезпечення, може мати вразливість у безпеці, що може призвести до порушення безпеки сайту.
- Необхідність оновлень. CMS потрібно регулярно оновлювати, щоб виправляти помилки та вразливості у безпеці. Це може вимагати додаткових витрат на обслуговування сайту.

SaaS-платформи

SaaS (Software as a Service) – це модель надання програмного забезпечення, при якій програми зберігаються та обробляються в хмарній інфраструктурі, а користувачі можуть отримати доступ до них через інтернет.

SaaS-платформи – це хмарні сервіси, які надають інструменти для створення, розгортання та керування вебсервісами. Вони пропонують широкий набір функцій та інструментів, включаючи бази даних, різні інструменти для аналітики та моніторингу, а також можливість керування та масштабування вебсервісу.

Переваги використання SaaS-платформ:

- Економія часу та грошей. SaaS-платформи можуть скоротити час, необхідний для створення та розгортання вебсервісу, а також позбавити необхідності купівлі та встановлення обладнання.
- Гнучкість та масштабованість. SaaS-платформи зазвичай надають гнучкі можливості налаштування та масштабування програм залежно від потреб бізнесу.
- Безпека та надійність. SaaS-платформи пропонують високий рівень безпеки та захисту даних, а також забезпечують високу доступність програми.

Недоліки використання SaaS-платформ:

- Обмеження в налаштуванні. Використання SaaS-платформ може обмежити можливості налаштування вебсервісу, оскільки багато налаштувань керуються провайдером сервісу.
- Обмежений контроль над обладнанням. Власник програми не має повного контролю над обладнанням, на котрому зберігається та обробляється його програма.
- Залежність від провайдера. Власник програми стає залежним від провайдера SaaS-сервісу, який може змінити свою політику або припинити надавати послуги у будь-який момент.

Тим не менш, SaaS-платформи все ще є популярним та зручним способом створення та управління вебсервісами, особливо для малих та середніх бізнесів.

Розробка з використанням фреймворків

Фреймворк – це сукупність програмних бібліотек, інструментів та стандартів, призначених для розробки. Він надає розробнику готову архітектуру та набір інструментів, які дозволяють швидше та зручніше створювати, тестувати та розгортати вебсервіси.

Фреймворк може включати такі компоненти:

- Засоби розробки. Фреймворк може включати засоби для написання коду, такі як текстові редактори, інтегровані середовища розробки та інші інструменти, які дозволяють швидко і зручно створювати код для вебсервісів.
- Архітектуру. Фреймворк визначає архітектуру вебсервісу та його компоненти, такі як моделі даних, контролери та уявлення. Це дозволяє розробникам швидко створювати нові компоненти та перевикористовувати вже існуючі.
- Бібліотеки. Фреймворк може включати набір готових бібліотек, які дозволяють вирішувати різні завдання при розробці вебсервісу, наприклад, обробку запитів, аутентифікацію, авторизацію, валідацію даних і т.д.
- Засоби тестування. Фреймворк може включати засоби для автоматичного тестування вебсервісів, що дозволяє розробникам швидко перевіряти роботу створеного коду і швидко виявляти помилки і проблеми.

Найбільш популярними фреймворками для розробки вебсервісів є наступні Flask, Django, Ruby on Rails, Express та інші. Кожен з них має свої особливості та переваги, які дозволяють вибрати найбільш підходящий для конкретного завдання.

Переваги використання фреймворків:

- Прискорення розробки. Фреймворк надає розробникам готову архітектуру та набір інструментів, які дозволяють швидше та зручніше створювати вебсервіси. Це скорочує час розробки та дозволяє швидше вивести продукт на ринок.

- Зниження витрат. Використання фреймворку дозволяє розробникам перевикористовувати готовий код та компоненти, що дозволяє знизити витрати на розробку та тестування програми.
- Велика стабільність та надійність. Фреймворк має добре налагоджену архітектуру та бібліотеки, які дозволяють створювати стабільні та надійні вебсервіси. Крім того, фреймворк часто містить безліч інструментів для тестування та налагодження, що дозволяє виявляти та виправляти помилки на ранніх етапах розробки.
- Легкість підтримки. Використання фреймворку робить програму більш легко підтримуваною. Розробники можуть швидко освоїтися з архітектурою та компонентами програми, і в разі потреби швидко внести зміни до коду.
- Велика спільнота розробників. Популярні фреймворки мають велику спільноту розробників, які активно обмінюються досвідом та допомагають один одному у вирішенні проблем. Це дозволяє швидко знаходити відповіді на запитання та отримувати рекомендації від досвідченіших колег.
- Гнучкість. Фреймворк дозволяє розробникам створювати програми, які можуть легко масштабуватися та адаптуватися під різні завдання та умови.
- Стандартизація. Фреймворки надають стандартизований підхід до розробки вебсервісів, що дозволяє підвищити якість та сумісність програм.

Недоліки використання фреймворків:

- Складність. Використання фреймворків може бути складним для розробників-початківців, які не мають досвіду роботи з певним фреймворком або мовою програмування. У такому разі розробка може зайняти більше часу і вимагати додаткових зусиль для вивчення документації та навчання.
- Обмеженість можливостей. Незважаючи на те, що фреймворки надають безліч інструментів і компонентів, деякі проекти можуть вимагати

реалізації функцій або алгоритмів, яких немає у фреймворку. У такому разі розробники повинні шукати додаткові рішення та інструменти.

- Ускладнення налагодження. У деяких випадках використання фреймворків може ускладнити налагодження програми, так як більша частина коду буде написана за допомогою інструментів та компонентів фреймворку. Це може ускладнити виявлення та виправлення помилок та проблем у додатку.

1.3 Аналіз існуючих аналогів та обґрунтування необхідності розробки

muzmarket.com.ua – це вебсервіс, через котрий можна придбати музичні інструменти та відповідне обладнання. Дана сторінка належить найближчому конкуренту, а саме «MusicFactor» [2]. Даний вебсервіс має простий інтерфейс та зрозумілі розділи сторінки. Проте рішення, котре використала компанія, має ряд значних недоліків. Головна сторінка сервісу представлена на рисунку 1.1. На сторінці фактично відсутня асоціація з самою компанією, серед згадок назви можна спостерігати назву «MusicMarket», яка фактично не відповідає дійсній назві компанії.

Також на головній сторінці можна помітити те, що інформація оформлена українською мовою вперемішку з російською.

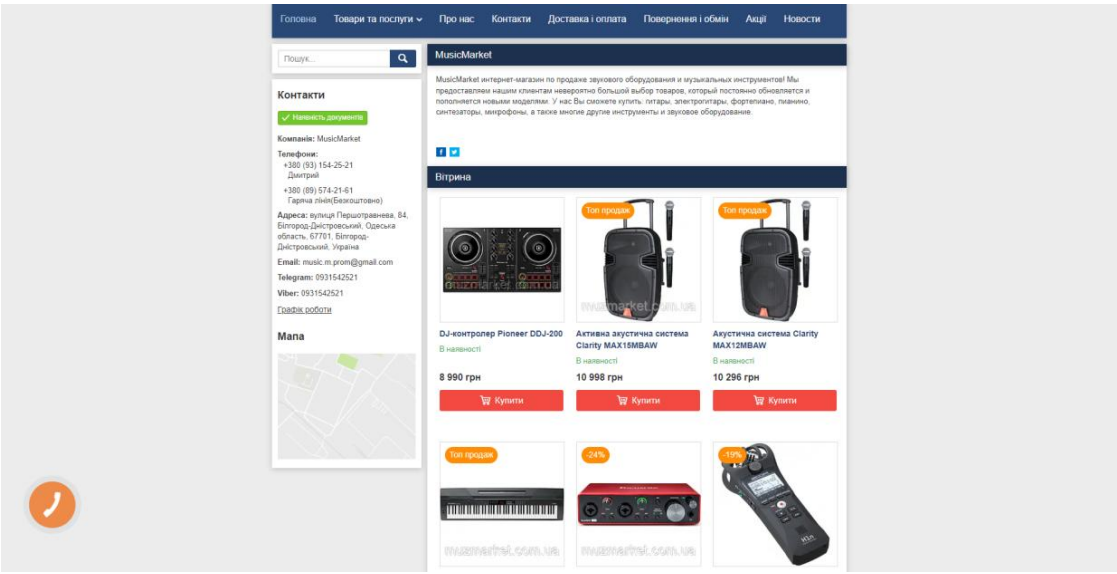


Рисунок 1.1 – Головна сторінка вебсервісу магазину продажу музичних інструментів «MusicFactor»

На сторінці «Товари та послуги», котра представлена на рисунку 1.2, зовсім відсутня фільтрації по категоріям, користувачу доступна лише фільтрація по ціні та виробнику.

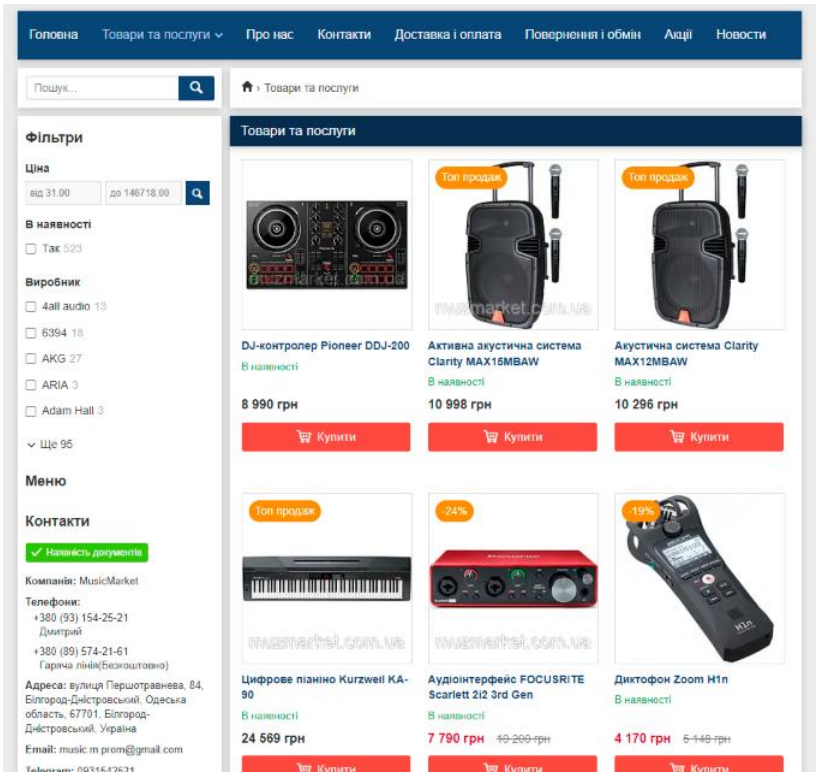


Рисунок 1.2 – Сторінка «Товари та послуги» магазину продажу музичних інструментів «MusicFactor»

Суттєвим мінусом є відсутність реєстрації та авторизації для користувачів, проте є можливість оформити замовлення завдяки раніше зареєстрованому аккаунту на платформі Prom.ua.

Також можна відмітити наявність помилки при наведенні на пункт «Товари та послуги» котра перешкоджає подальшій навігації по сервісу.

До мінусів можна також віднести те, що сервіс був створений за допомогою платформи Prom.ua, адже використовуючи дану платформу для запуску сервісу, компанія просто губиться на фоні величезної кількості таких же магазинів, котрі відрізняються лише кольором оформлення сторінки.

4Club.com.ua – це один з найпопулярніших вебсервісів, що спеціалізується на продажі музичних інструментів та відповідного обладнання в Україні [3]. Даний вебсервіс має простий інтерфейс та зрозумілі розділи сторінки. Більшість недоліків, котрі були виявлені в попередньому прикладі, тут відсутні. Нижче, на рисунку 1.3 представлено головну сторінку вебсервісу.

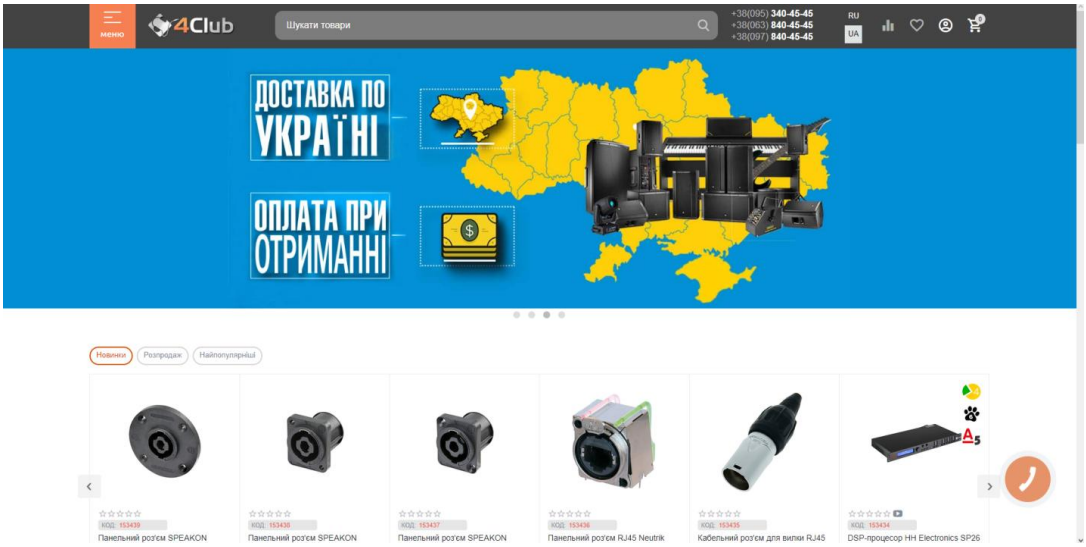


Рисунок 1.3 – Головна сторінка вебсервісу магазину продажу музичних інструментів «4Club»

Проте, незважаючи на приємне оформлення веб-сторінок, навігація переускладнена, що можна спостерігати на рисунку 1.4 та на рисунку 1.5, адже для перегляду каталогу доступних товарів потрібно спочатку відкрити випадаюче меню, після обрати категорію. Після того, як користувач обрав категорію, його перенаправить на сторінку підкатегорій, лише після вибору підкатегорії користувач отримує можливість ознайомлення з асортиментом.

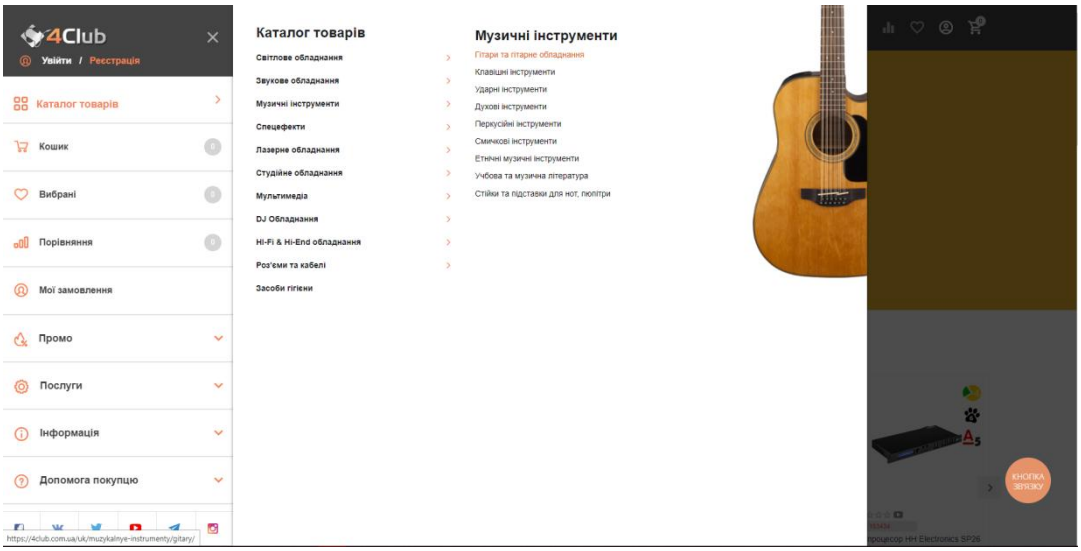


Рисунок 1.4 – Випадаюче меню вебсервісу магазину продажу музичних інструментів «4Club»

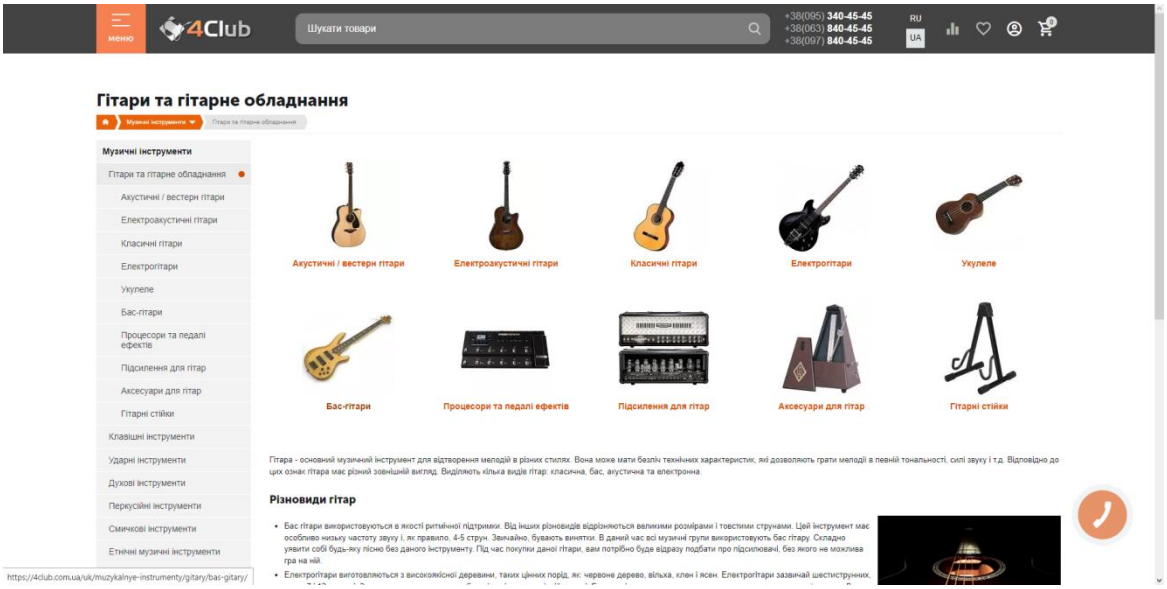


Рисунок 1.5 – Навігація по каталогу вебсервісу магазину продажу музичних інструментів «4Club»

Серед вагомих переваг є те, що в сервісі інтегрована оплата за допомогою картки, та є можливість придбати товар в розстрочку.

Також, в ході аналізу було виявлено те, що сервіс був створений за допомогою платформи horoshop.ua. Дане рішення призводить до того, що використовуючи дану платформу для запуску сервісу, компанія просто губиться поміж інших.

SoundMaster.ua – вебсервіс, який спеціалізується на продажу музичних інструментів та обладнання. Належить музичному магазину «SoundMaster». Музичний магазин «SoundMaster» був відкритий одним із перших серед інтернет-магазинів музичного обладнання в Україні, і з того часу являється одними з лідерів на українському ринку [4].

Головна сторінка вебсервісу представлена на рисунку 1.6.

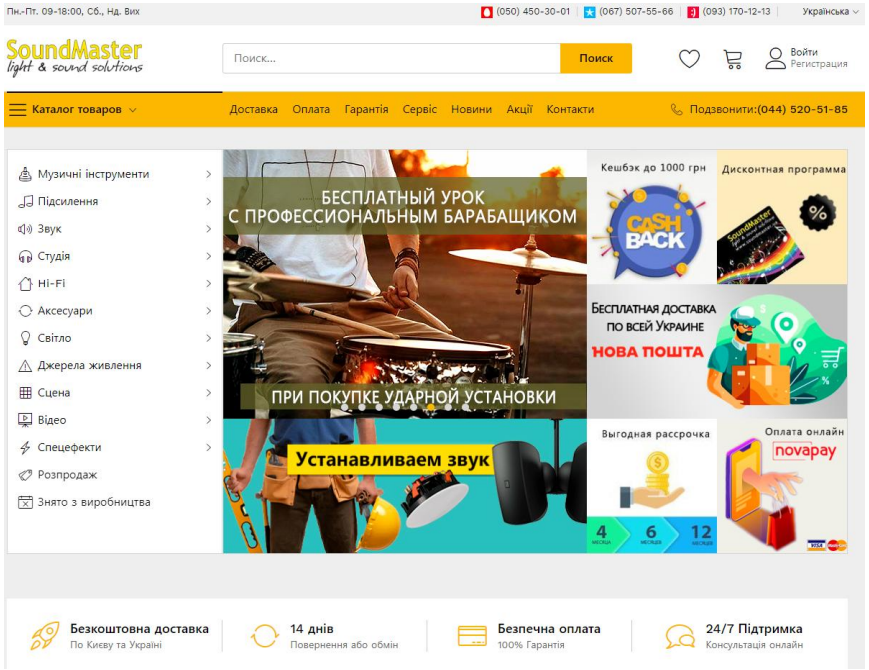


Рисунок 1.6 – Головна сторінка вебсервісу магазину продажу музичних інструментів «SoundMaster»

На відміну від раніше розглянутого вебсервісу, в даному рішенні навігація по каталогу реалізована більш зручним чином та дозволяє

переглядати каталог доступних товарів без конкретизації сегменту товару. Скріншот сторінки навігації по каталогу представлено на рисунку 1.7

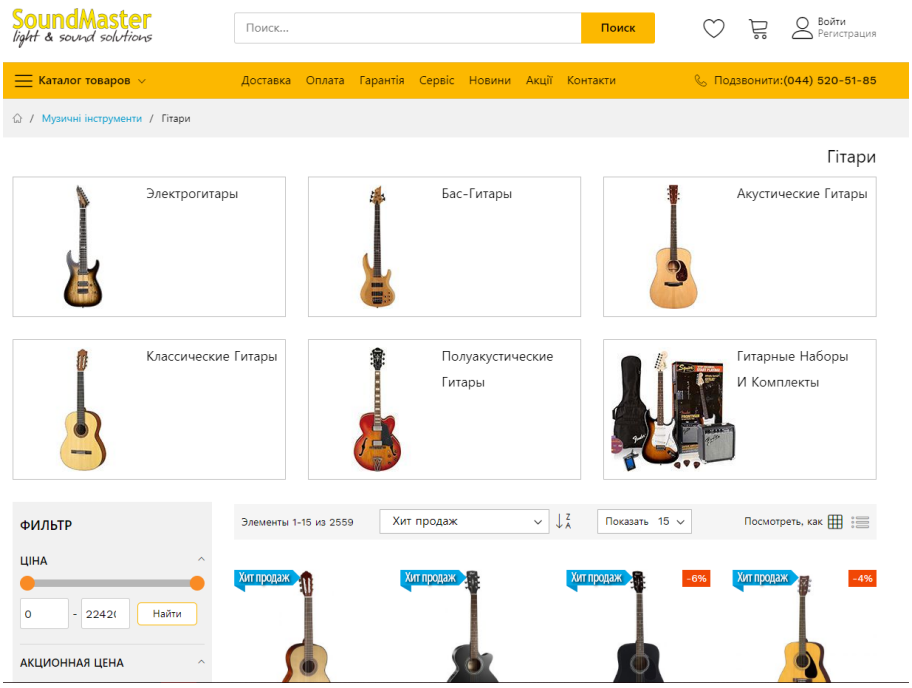


Рисунок 1.7 – Навігація по каталогу вебсервісу магазину продажу музичних інструментів «SoundMaster»

Магазин «SoundMaster» запустив вебсервіс не використовуючи платформ, котрими скористались раніше описані компанії, що вигідно виокремлює компанію «SoundMaster» від конкурентів.

Нажаль, не зважаючи на зручність та індивідуальність підходу, на сторінках присутні блоки, в котрих локалізація не відповідає обраній.

Незважаючи на те, що рішення, обране компанією «SoundMaster», виявилось найбільш вдалим, використання даного рішення для запуску вебсервісу магазину з продажу музичних інструментів «MusicBrand» неможливе, адже дане рішення не розповсюджується ні на безкоштовній основі, ні на комерційній.

Виходячи з вище перелічених недоліків можна зробити висновок, що створення власного вебсервісу для магазину з продажу музичних

інструментів «MusicBrand» окрім вирішення описаних вище проблем, надасть наступні переваги:

- Індивідуальність. Розробка власного рішення дозволяє врахувати всі індивідуальні вимоги та особливості бізнесу. Готові рішення не завжди можуть відповідати всім вимогам, які необхідні для вирішення конкретних завдань бізнесу. Свій власний продукт дозволить повністю задовольнити всі потреби та вимоги бізнесу. Готові рішення зазвичай мають обмежений функціонал, який не завжди може відповідати всім потребам і вимогам компанії. З іншого боку, власне рішення може бути розроблене з урахуванням конкретних потреб бізнесу та мати функціонал, який повністю відповідає вимогам компанії.
- Гнучкість та масштабованість. Готові рішення можуть бути обмежені своїми можливостями і не завжди можуть бути легко розширені, щоб задовольняти потреби бізнесу. Власне рішення може бути легко масштабовано та змінено відповідно до нових вимог бізнесу.
- Підтримка та оновлення. Готові рішення можуть бути неактуальними та вимагати регулярних оновлень. При розробці власного рішення можна забезпечити підтримку та оновлення відповідно до потреб бізнесу.
- Витрати. Розробка власного рішення може дозволити знизити витрати у довгостроковій перспективі. Готові рішення можуть бути дорогими, особливо, якщо потрібно платити за використання або купівлю ліцензії на продукт. Власне рішення може бути більш економічним у довгостроковій перспективі.
- Контроль. Розробка свого рішення дає більший контроль над вебсервісом. Можна контролювати кожен аспект розробки та забезпечувати високу якість продукту.

1.4 Постановка задачі на розробку вебсервісу для магазину з продажу музичних інструментів

Виконавши аналіз предметної області та проаналізувавши недоліки існуючих аналогів, сформулюємо постановку задачі на розробку вебсервісу для магазину з продажу музичних інструментів «MusicBrand».

Розроблюваний вебсервіс має відповідати наступним вимогам:

1) Вимоги до функціональних характеристик:

Функції користувача:

• Реєстрація – реєстрація призначається для підвищення швидкості виконання повторних покупок на сайті. Користувач вводить свої дані, такі як: ПІБ, номер телефону, адреса електронної пошти, пароль. Після закінчення реєстрації отримує відповідне повідомлення. Додаткова інформація про тип та адресу доставки збережеться під час першої покупки.

• Авторизація – вхід до особистого кабінету здійснюється введенням логіну (номер телефону/email) та пароля.

• Переглянути товар – натисканням плитки з представленим товаром клієнт потрапляє на «односторінок». Тут відображається опис та характеристики товару (ціна, вага, тип, габарити тощо). На цій сторінці передбачена можливість оформлення замовлення/додавання в кошик.

• Оформити замовлення – якщо користувач авторизований, введення необхідних для оформлення замовлення (ПІБ, номер телефону, тип доставки, адреса, тип оплати) не потребуються. Вони автоматично з’являються в формі оформлення замовлення з можливістю коригування. Інакше, якщо користувач не авторизований, відображається сторінка, на якій клієнт вводить дані необхідні для оформлення замовлення. На сторінці оформлення замовлення відображається перелік товарів, які будуть входити в замовлення (ціна, кількість, назва, зображення).

• Помістити товар у кошик – під час перегляду товару доступна можливість натисканням додавання товару до списку для подальшого оформлення замовлення кількох товарів одночасно.

Функції адміністратора:

• Авторизація – вхід до особистого кабінету здійснюється введенням логіну та пароля.

• Управління каталогом товарів – адміністратор може додавати, редагувати та видаляти товари в каталозі.

• Управління замовленнями – адміністратор може переглядати, обробляти та відстежувати замовлення, що надходять до магазину.

Значення властивостей та характеристик за їх відсутності у сутності не виводяться.

У всіх областях, де використовується таблично-стрічкове представлення товару, застосовується стандартне каталожне подання товару, якщо не вказано інше.

Форми авторизації/реєстрації. Усі спливаючі форми введення виконуються в єдиному стилі.

2) Вимоги до організації вхідних та вихідних даних

Дані про товари зберігаються у базі даних. СУБД забезпечує розмежування прав доступу до даних – надає клієнту права на читання, а адміністратору – на читання та запис.

3) Вимоги до складу та параметрів технічних засобів

Сервер повинен відповідати наступному складу технічних засобів з мінімальними параметрами:

- Операційна система: Linux або Windows Server
- Процесор: 64-бітний двох або більше ядерний процесор з тактовою частотою не менше 2 ГГц
- Оперативна пам'ять: не менше 4 ГБ
- Диск: не менше 100 ГБ

Для роботи клієнтської частини необхідні наявність браузера з підтримкою HTML-5/CSS-3 та доступ до виходу у мережу Internet.

Більш докладно постановка задачі сформульована в технічному завданні, яке наведене в Додатку А.

2 РОЗРОБКА ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

2.1 Аналіз вимог до вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

У цьому підрозділі проведено формалізацію вимог до вебсервісу для магазину з продажу музичних інструментів «MusicBrand» шляхом побудови діаграми варіантів використання та приведення до них специфікацій.

2.1.1 Побудова моделі варіантів використання вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

В ході аналізу предметної галузі та формуванні вимог було виявлено два актори, котрі взаємодіють з вебсервісом, а саме клієнт та адміністратор.

Вебсервіс орієнтований на клієнтів магазину з продажу музичних інструментів, надалі їх названо «Користувачем». Особа відповідальна за заповнення інформацією системи, редагування та видалення, а також несе відповідальність за працездатність сервера, названо «Адміністратором». Вони являються акторами системи. Короткий опис акторів наведено в таблиці.

Таблиця 2.1 – Виявлені актори

Актор	Короткий опис
Користувач	Особа, котра переглядає каталог товарів, оформлює замовлення
Адміністратор	Особа, котра редагує інформацію про товари та обробляє оформленні замовлення

Виявлені варіанти використання вебсервісу магазину з продажу музичних інструментів «MusicBrand» зведено до таблиці 2.2.

Таблиця 2.2 – Виявлені варіанти використання вебсервісу магазину з продажу музичних інструментів «MusicBrand»

Актор	Назва	Короткий опис
1	2	3
Користувач, Адміністратор	Автентифікація	Даний варіант використання дозволяє користувачеві та адміністратору пройти автентифікацію в системі
Адміністратор	Додати товар	Варіант використання дозволяє адміністратору додавати товар до системи
Адміністратор	Редагувати інформацію про товар	Варіант використання, котрий дозволяє адміністратору редагувати інформацію про будь-який товар в системі
Адміністратор	Видалити товар	Варіант використання дозволяє адміністратору видаляти будь-який товар з системи
Адміністратор	Обробити замовлення	Варіант використання, котрий дозволяє адміністратору змінити статус виконання замовлення
Користувач	Реєстрація	Даний варіант використання дозволяє користувачеві пройти реєстрацію

Продовження таблиці 2.2

1	2	3
Користувач, Адміністратор	Пошук товару	Варіант використання, котрий дозволяє користувачеві виконувати пошук товарів за заданими критеріями
Користувач, Адміністратор	Перегляд товару	Варіант використання дозволяє користувачеві переглянути сторінку з детальним описом товару
Користувач	Додати товар до кошика	Варіант використання, котрий дозволяє користувачеві додавати товар до кошика
Користувач	Видалити товар з кошика	Варіант використання, котрий дозволяє користувачеві видалити товар з кошика
Користувач	Оформити замовлення	Варіант використання, котрий дозволяє користувачеві оформити замовлення на основі обраних раніше в кошику товарів

Варіанти використання грають ключову роль в управлінні процесом розробки. Під час аналізу та проектування, вони дозволяють зрозуміти, як очікувані результати користувача впливають на архітектуру системи та взаємодію компонентів, щоб забезпечити необхідну функціональність [5]. Рисунок 2.1 демонструє діаграму варіантів використання вебсервісу магазину з продажу музичних інструментів «MusicBrand», яка ілюструє цей підхід.

Рисунок 2.1 – Діаграма варіантів використання

2.1.2 Специфікації варіантів використання вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

Специфікації варіантів використання вебсервісу представлені в таблицях 2.3 – 2.12:

Таблиця 2.3 – Специфікація варіанту використання «Автентифікація»

Назва	Характеристика
1	2
Опис прецеденту	Автентифікація

Продовження таблиці 2.3

1	2
Дійові особи прецеденту	Користувач, адміністратор
Передумови	Користувач або адміністратор мають бути зареєстровуваними у системі
Потік подій	Прецедент починається, коли актор переходить до веб-сторінки вебсервісу
Базовий потік	1. Користувач або адміністратор натискає кнопку «Увійти». 2. Система переадресує його на сторінку авторизації. 3. Користувач або адміністратор вводить свій логін та пароль. 4. Система перевіряє ці дані на збіг з даними у базі даних та автентифікує користувача.
Альтернативні потоки	1. Користувач або адміністратор вводить свої дані. 2. Система не знаходить в базі даних такої комбінації логіну та паролю. 3. Система відображає повідомлення про неправильний логін або пароль.
Постумови	Якщо користувач чи адміністратор були успішно автентифіковані, то їх ім’я має відобразитися у спеціально відведеному для цього місці.

Таблиця 2.4 – Специфікація варіанту використання «Додати товар»

Назва	Характеристика
1	2
Опис прецеденту	Додавання нових товарів до системи
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор повинен увійти в систему.
Базовий потік	1. Адміністратор відкриває адмін-панель. 2. Адміністратор натискає кнопку «Додати товар». 3. Адміністратор вводить данні про товар(категорія, виробник, опис і т.д.) 4. Адміністратор натискає кнопку «Зберегти». 5. Система зберігає товар в базі даних.
Альтернативні потоки	1. Адміністратор відкриває адмін-панель. 2. Адміністратор натискає кнопку «Додати товар». 3. Адміністратор вводить данні про товар(категорія, виробник, опис і т.д.). 4. Адміністратор заповнив не всі поля та натиснув кнопку «Зберегти». 5. Система відображає повідомлення про необхідність заповнити всі обов'язкові поля.
Постумови	Додані продукти будуть збережені в базі даних та відповідне повідомлення відобразиться системою.

Таблиця 2.5 – Специфікація варіанту використання «Редагувати інформацію про товар»

Назва	Характеристика
1	2
Опис прецеденту	Варіант використання, котрий дозволяє адміністратору редагувати інформацію про будь-який товар в системі
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор повинен увійти в систему.
Базовий потік	1. Адміністратор відкриває адмін-панель. 2. Адміністратор натискає кнопку «Товар». 3. Адміністратор натискає кнопку «Редагувати» навпроти товару зі списку. 4. Адміністратор редагує данні про товар(категорія, виробник, опис і т.д.) 5. Адміністратор натискає кнопку «Зберегти». 6. Система зберігає товар в базі даних.
Альтернативні потоки	1. Адміністратор відкриває адмін-панель. 2. Адміністратор натискає кнопку «Товар». 3. Адміністратор натискає кнопку «Редагувати» навпроти товару зі списку. 4. Адміністратор редагує данні про товар(категорія, виробник, опис і т.д.) 6. Адміністратор заповнив не всі поля та натиснув кнопку «Зберегти». 7. Система відображає повідомлення про необхідність заповнити всі обов'язкові поля.

Продовження таблиці 2.5

1	2
Постумови	Відредаговані товари будуть збережені в базі даних та відповідне повідомлення відобразиться системою.

Таблиця 2.6 – Специфікація варіанту використання «Обробити замовлення»

Назва	Характеристика
1	2
Опис прецеденту	Адміністратор змінює статус виконання замовлення
Дійові особи прецеденту	Адміністратор
Передумови	Адміністратор повинен увійти в систему
Базовий потік	1. Адміністратор відкриває діалогове вікно «Замовлення». 2. Адміністратор натискає кнопку «Змінити статус замовлення». 3. Адміністратор натискає кнопку «Зберегти». 4. Система зберігає зміни товар в базу даних.
Альтернативні потоки	Відсутні
Постумови	Зміна статусу замовлення зберігається в базі даних та відповідне повідомлення відобразиться системою.

Таблиця 2.7 – Специфікація варіанту використання «Реєстрація»

Назва	Характеристика
1	2
Опис прецеденту	Реєстрація

Продовження таблиці 2.7 – Сценарій використання «Реєстрація»

1	2
Дійові особи прецеденту	Користувач
Передумови	Відсутні
Потік подій	Прецедент починається, коли актор переходить до веб-сторінки вебсервісу
Базовий потік	1. Користувач натискає кнопку «Увійти». 2. Система переадресує його на сторінку авторизації. 3. Користувач натискає кнопку «Не маєте профіля? Зареєструватися » 4. Користувач вводить необхідні для реєстрації данні. 5. Користувач натискає кнопку «Зареєструватися». 6. Система перевіряє ці дані на збіг з даними у базі даних , реєструє та автентифікує користувача.
Альтернативні потоки	1. Користувач натискає кнопку «Увійти». 2. Система переадресує його на сторінку авторизації. 3. Користувач натискає кнопку «Не маєте профіля? Зареєструватися » 4. Користувач вводить данні. 4.а. Користувач заповнив не всі поля. 4.б. Користувач ввів некоректні данні. 5. Користувач натискає кнопку «Зареєструватися».

Продовження таблиці 2.7 – Сценарій використання «Реєстрація»

1	2
	6. Система виводить повідомлення про помилку 6.а. Не заповненні обов’язкові поля. 6.б. Помилка валідації.
Постумови	Якщо користувач був успішно зареєстрований, то їх ім’я має відобразитися у спеціально відведеному для цього місці.

Таблиця 2.8 – Специфікація варіанту використання «Перегляд товару»

Назва	Характеристика
1	2
Опис прецеденту	Актор переглядає сторінку товару
Дійові особи прецеденту	Користувач, адміністратор
Передумови	Прецедент починається, коли актор переходить до веб-сторінки магазину
Базовий потік	1. Актор натискає на блок товару. 2. Система перенаправляє користувача на сторінку товару.
Альтернативні потоки	Відсутні
Постумови	На екрані відображається інформація про обраний товар.

Таблиця 2.9 – Специфікація варіанту використання «Пошук товару»

Назва	Характеристика
1	2
Опис прецеденту	Актор виконує пошук за певними критеріями

Продовження таблиці 2.9 – Специфікація варіанту використання «Пошук товару»

1	2
Дійові особи прецеденту	Користувач
Передумови	Прецедент починається, коли актор переходить до веб-сторінки магазину
Базовий потік	1. Користувач налаштовує фільтр пошуку у відповідному блоці. 2. Система виконує пошук товарів по заданим критеріям. 3. Система виводить на екран товари, котрі відповідають заданим критеріям
Альтернативні потоки	Відсутні
Постумови	На екрані відображаються товари, котрі відповідають заданим критеріям

Таблиця 2.10 – Специфікація варіанту використання «Додати товар до кошика»

Назва	Характеристика
1	2
Опис прецеденту	Додати товар до кошика
Дійові особи прецеденту	Користувач
Передумови	1. Користувач виконує операцію перегляду або пошуку. 2. Користувач повинен увійти в систему.

Продовження таблиці 2.10 – Специфікація варіанту використання «Додати товар до кошика»

1	2
Потік подій	Прецедент починається, коли клієнт переходить до веб-сторінки магазину
Базовий потік	1. Користувач обирає один або кілька продуктів зі списку. 2. Користувач натискає кнопку «Купити». 3. Система додає цей товар до кошика 4. Система відображає повідомлення про успішне додавання товару до кошика
Альтернативні потоки	Відсутні
Постумови	Система зберігає вміст кошика у базі данних.

Таблиця 2.11 – Специфікація варіанту використання «Видалити товар з кошика»

Назва	Характеристика
1	2
Опис прецеденту	Видалити товар з кошика
Дійові особи прецеденту	Користувач
Передумови	Користувач повинен увійти в систему.
Потік подій	Прецедент починається, коли клієнт переходить до веб-сторінки магазину
Базовий потік	1. Користувач обирає один або кілька продуктів зі списку. 2. Користувач натискає кнопку «Видалити з кошика».

Продовження таблиці 2.11 – Специфікація варіанту використання «Видалити товар з кошика»

1	2
	3. Система видаляє цей товар з кошика 4. Система відображає повідомлення про успішне видалення товару з кошика
Альтернативні потоки	Відсутні
Постумови	Система зберігає вміст кошика у базі данних.

Таблиця 2.12 – Специфікація варіанту використання «Оформити замовлення»

Назва	Характеристика
1	2
Опис прецеденту	Користувач у вікні корзини натискає кнопку «Оформити замовлення» і система зберігає інформацію про замовлення.
Дійові особи прецеденту	Користувач
Передумови	Користувач повинен увійти в систему.
Потік подій	Прецедент починається, коли клієнт переходить до веб-сторінки магазину
Базовий потік	1. Користувач відкриває сторінку корзини. 2. Користувач натискає кнопку «Оформити замовлення». 3. Система відображає вікно оформлення замовлення. 4. Користувач вводить контактну інформацію. 5. Система зберігає інформацію про замовлення.

Продовження таблиці 2.12 – Сценарій використання «Оформити замовлення»

1	2
	6. Система відображає повідомлення про успішне оформлення замовлення
Альтернативні потоки	1. Користувач відкриває сторінку корзини. 2. Користувач натискає кнопку «Оформити замовлення». 3. Система відображає вікно оформлення замовлення. 4. Користувач вводить контактну інформацію. 5. Виникає помилка валідації даних. 6. Система відображає повідомлення про помилку, котра виникла.
Постумови	Система зберігає данні замовлення у базі даних.

2.2 Розробка архітектури вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

Для вирішення завдання з розробки вебсервісу для магазину з продажу музичних інструментів «MusicBrand» обрано архітектуру MVC.

Архітектура MVC обрана тому, що вона надає модульну структуру, яка допомагає краще організувати та керувати кодом програми. Вона також дозволяє легко розділяти різні рівні програми, такі як уявлення, логіка та дані.

У контексті вебсервісу для магазину з продажу музичних інструментів, архітектура MVC допомагає відокремити логіку бізнесу, таку як управління кошиком покупок або обробка замовлень, від представлення, такого як веб-сторінки, які відображають інформацію про продукти магазину.

Контролери (Controller) обробляють запити, що входять, і направляють їх на відповідні сервіси (Service), які містять бізнес-логіку програми. Сервіси (Service) використовують модель (Model), щоб отримувати та обробляти дані, та репозиторії (Repository), щоб зберігати та отримувати дані з бази даних [6].

Завдяки такій структурі, можна легко масштабувати та модифікувати додаток, не торкаючись інших його компонентів. Це також спрощує тестування та забезпечує кращу читаність коду, що може прискорити процес розробки.

Якщо підвести підсумок, то переваги наступні:

Поділ відповідальності

MVC-архітектура дозволяє розділити додаток на три компоненти з різними обов'язками. Це дозволяє спростити розробку та підтримку програми, а також покращити її масштабованість. Сутності (entity) є моделлю, контролери відповідають за управління потоками даних, а послуги містять бізнес-логіку. Це дозволяє краще структурувати програму та зменшити зв'язність між компонентами.

Найкраща керованість:

MVC-архітектура дозволяє краще контролювати потік даних у додатку. Контролери є центром управління і можуть опрацьовувати різні типи запитів. Модель не залежить від уявлення, що дозволяє її легше тестувати та перевикористовувати.

Гнучкіша архітектура:

MVC-архітектура дозволяє гнучко змінювати компоненти програми, не торкаючись інших частин. Наприклад, можна легко змінити уявлення (view) без впливу на модель (model) чи контролери (controller).

Нижче, на рисунку 2.2 представлено діаграму компонентів в загальному випадку:

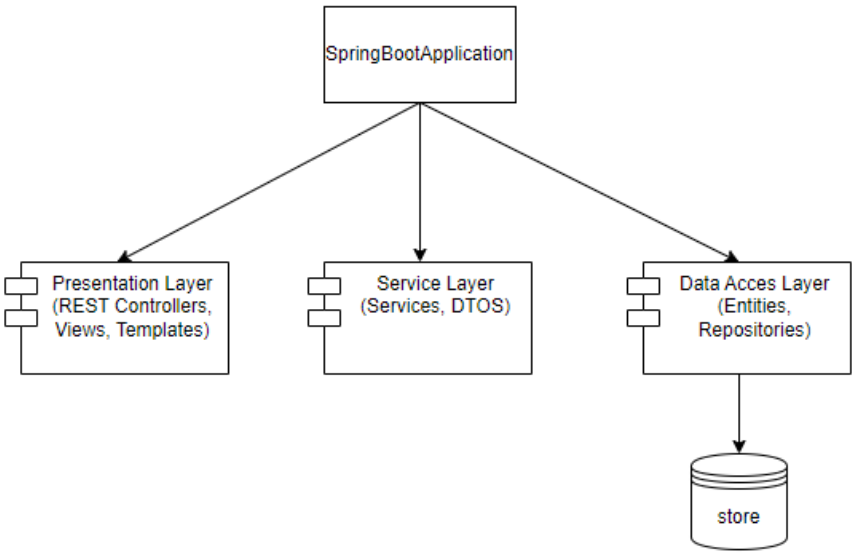


Рисунок 2.2 – Діаграма компонентів

Ланцюжок взаємодії компонентів представлено на рисунку 2.3 та виглядає наступним чином:

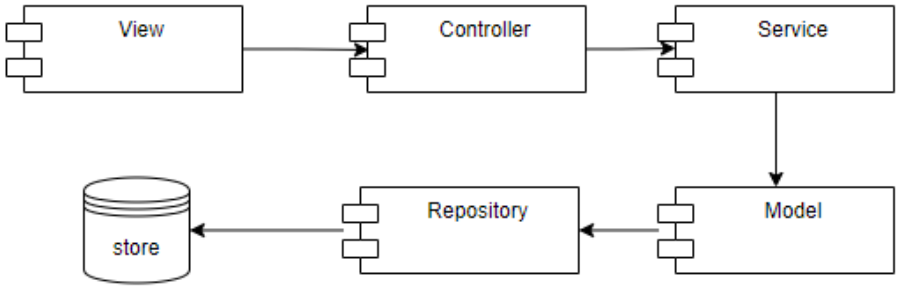


Рисунок 2.3 – Діаграма компонентів

Фронтенд (View) представлений HTML/CSS, що обробляється браузером клієнта. Контролер (Controller) відповідає за обробку запитів від клієнта та направляє їх у відповідний сервіс (Service). Сервіс (Service) містить бізнес-логіку програми та використовує модель (Model) для отримання та обробки даних. Модель (Model) представляє сутності програми і взаємодіє з репозиторіями (Repository), щоб отримувати чи зберігати дані у базі даних. Репозиторії (Repository) надають інтерфейс взаємодії з базою даних.

Для представлення того, як система буде розгорнута фізично, доцільно використовувати діаграму розміщення. Діаграма розміщення (deployment diagram) є однією з діаграм уніфікованої мови моделювання (UML), яка використовується для відображення фізичної архітектури програмного забезпечення. Вона дозволяє показати, які програмні та апаратні засоби використовуються в системі, а також як вони взаємодіють між собою.

За допомогою діаграми розміщення можна відобразити фізичні компоненти системи, такі як сервери, комп'ютери, додатки, бази даних, мережі тощо, а також з'єднання між ними. Діаграма показує, як ці компоненти фізично розташовані та зв'язані між собою.

Діаграма наведена на рисунку 2.4.

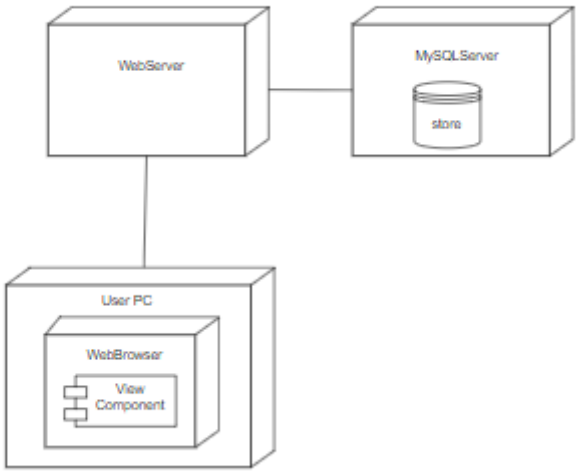


Рисунок 2.4 – Діаграма розміщення

2.3 Проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

В ході проектування вебсервісу для магазину з продажу музичних інструментів було розроблено ескізний, технічний та робочий проекти.

2.3.1 Ескізний проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

В ході розробки ескізного проекту вебсервісу для магазину з продажу музичних інструментів було побудовано діаграми діяльності, котрі формалізують сценарії варіантів використання. Також було побудовано інформаційну модель, котра репрезентована за допомогою концептуальної моделі даних згідно методології IDEF1X. Окрім цього, в ході розробки ескізного проекту, було створено проект користувальницького інтерфейсу.

Сценарії основних варіантів використання найкраще репрезентують діаграми діяльності.

Діаграма діяльності - це одна з діаграм уніфікованої мови моделювання (UML), яка використовується для відображення поведінки системи або процесу [7]. Ця діаграма дозволяє показати послідовність дій, які система або актор можуть виконувати, а також умови, які потрібно виконати для того, щоб дії були виконані.

Кожна дія на діаграмі має вхідні та вихідні потоки, що показують переходи між кроками виконання дії. Крім того, можуть бути використані умови, які показують, за яких обставин дія може бути виконана або пропущена.

Діаграма діяльності для варіанту використання «Оформити замовлення» представлена на рисунку 2.5.

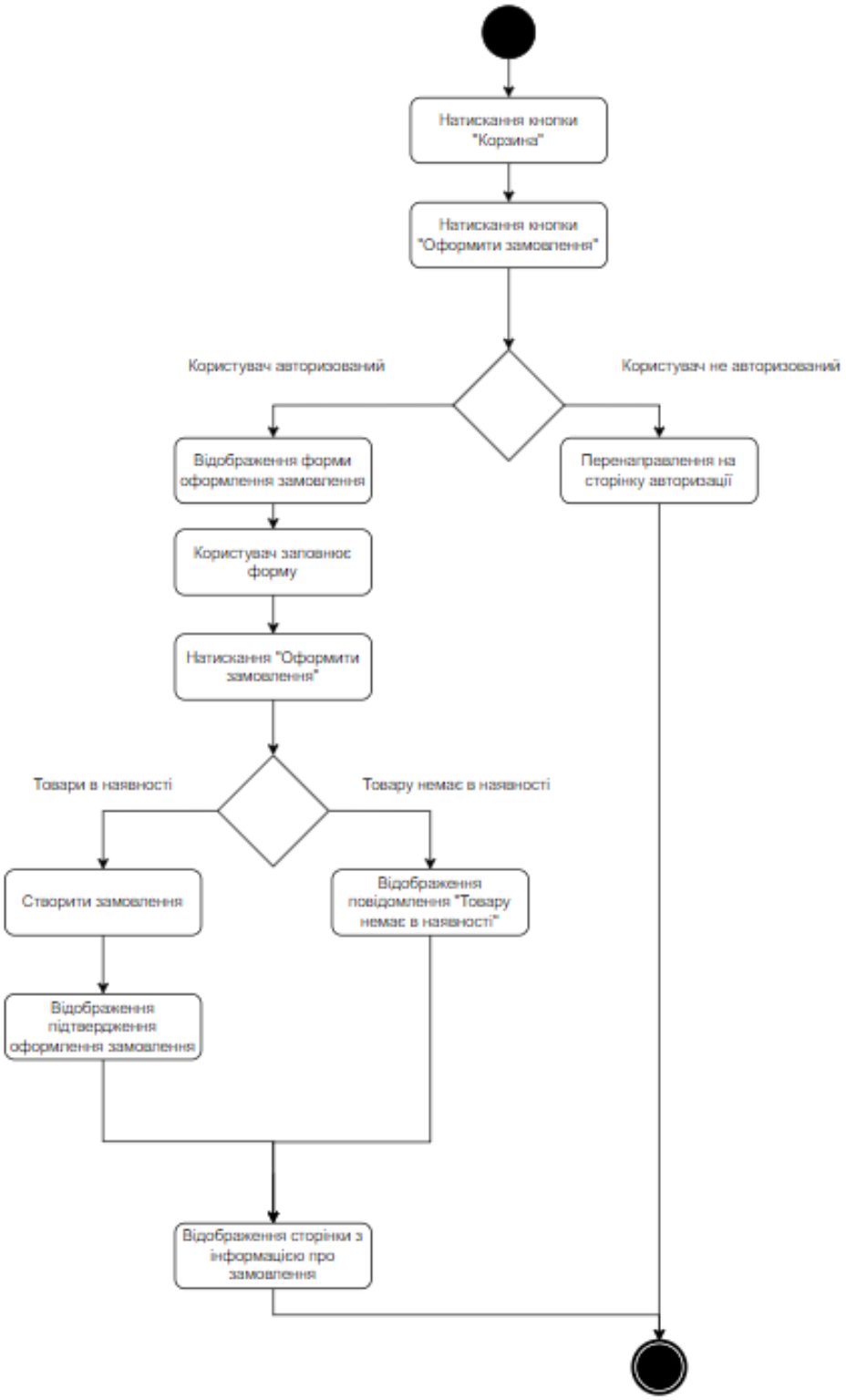


Рисунок 2.5 – Діаграма діяльності для варіанту використання «Оформити замовлення»

Діаграма діяльності для варіанту використання «Редагувати інформацію про товар» представлена на рисунку 2.6.

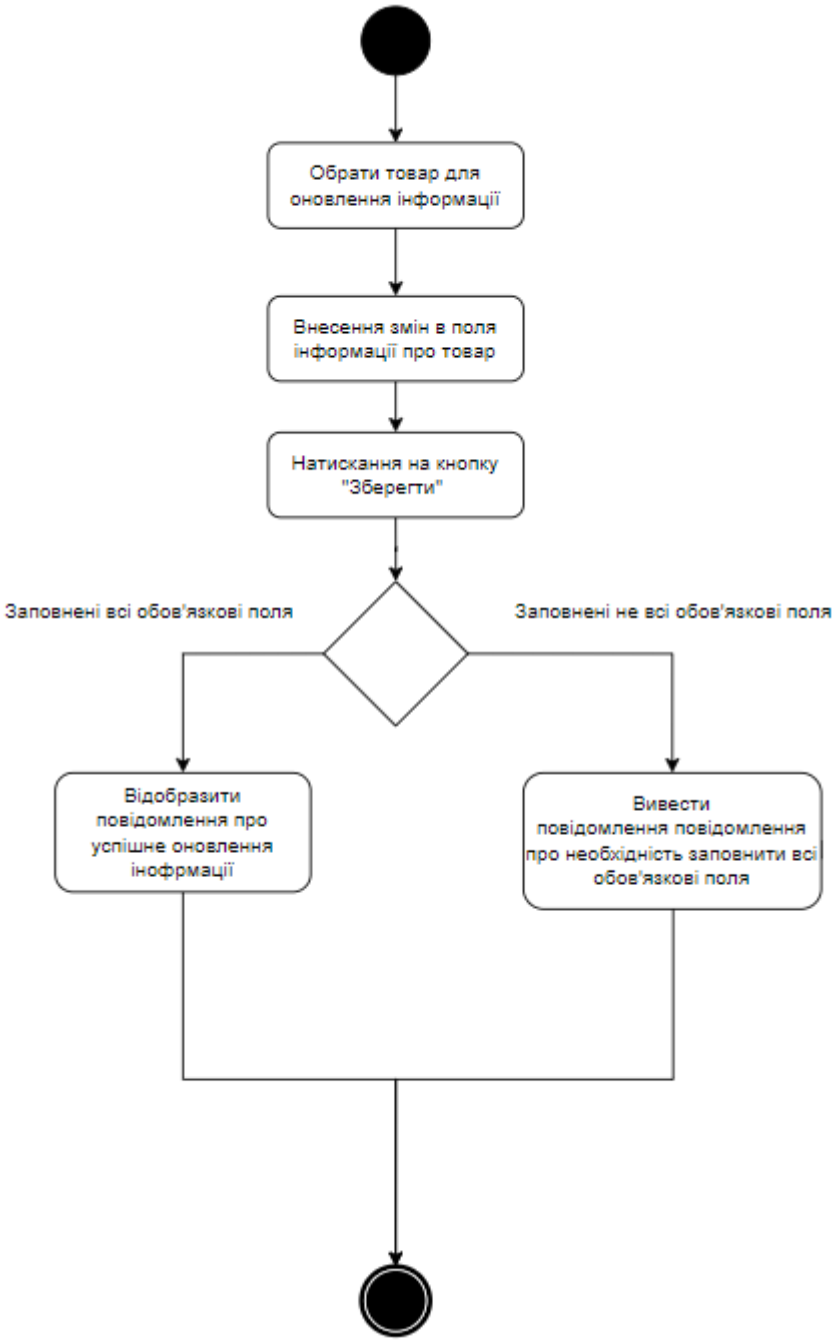


Рисунок 2.6 – Діаграма діяльності для варіанту використання «Редагувати інформацію про товар»

Діаграму діяльності для варіанту використання «Автентифікація» представлено на рисунку 2.7.

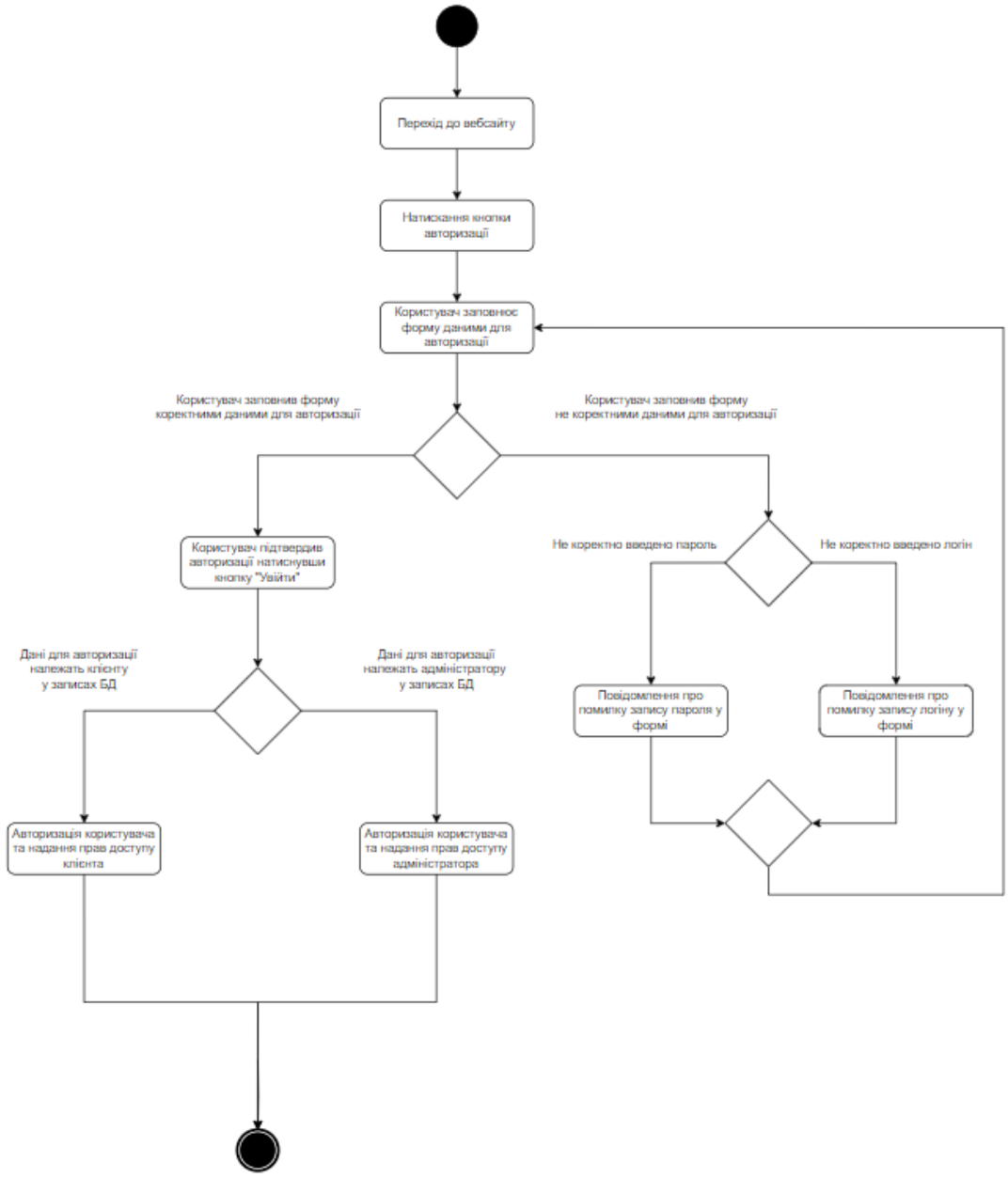


Рисунок 2.7 – Діаграма діяльності для варіанту використання «Автентифікація»

Нижче на рисунку 2.8 представлено концептуальну модель даних, котра представляє собою абстрактне подання інформації, котру міститиме база даних, та описує основні сутності, їх атрибути та зв’язки між ними.

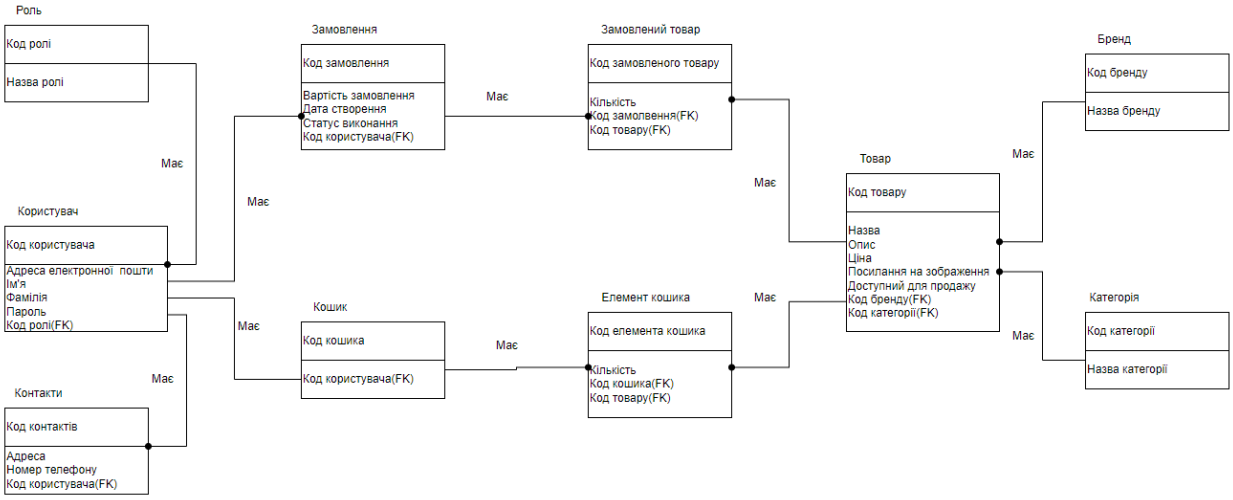


Рисунок 2.8 – Концептуальна модель даних

Наступним етапом розробки ескізного проекту вебсервісу для магазину з продажу музичних інструментів «MusicBrand» являється розробка інтерфейсу. Нижче, на рисунках 2.9 – 2.15, представлені ескізи інтерфейсу.

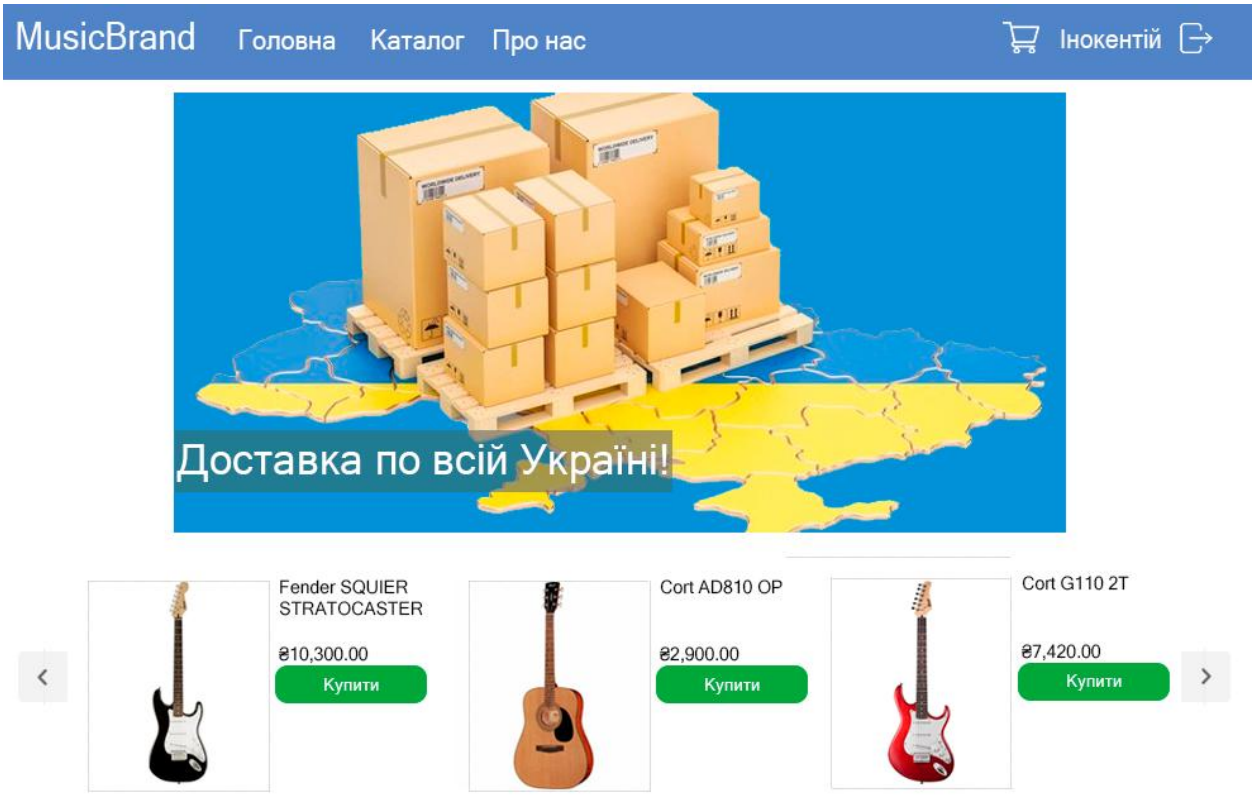


Рисунок 2.9 – Ескіз головної сторінки

Рисунок 2.10 – Ескіз сторінки товару

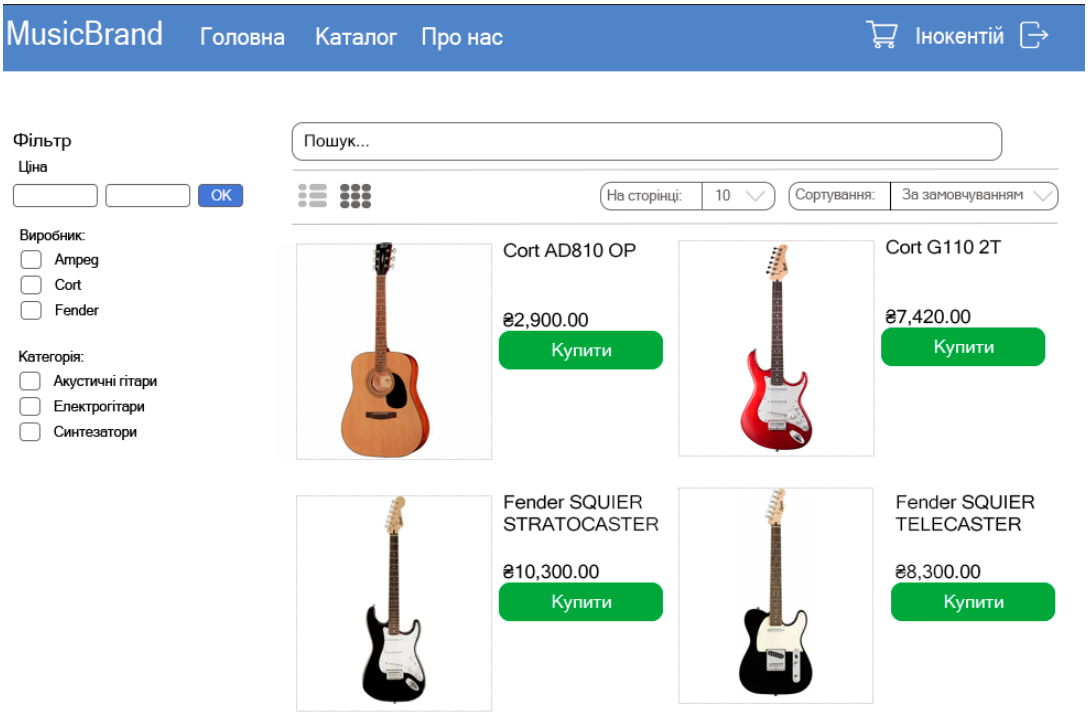


Рисунок 2.11 – Ескіз сторінки каталогу

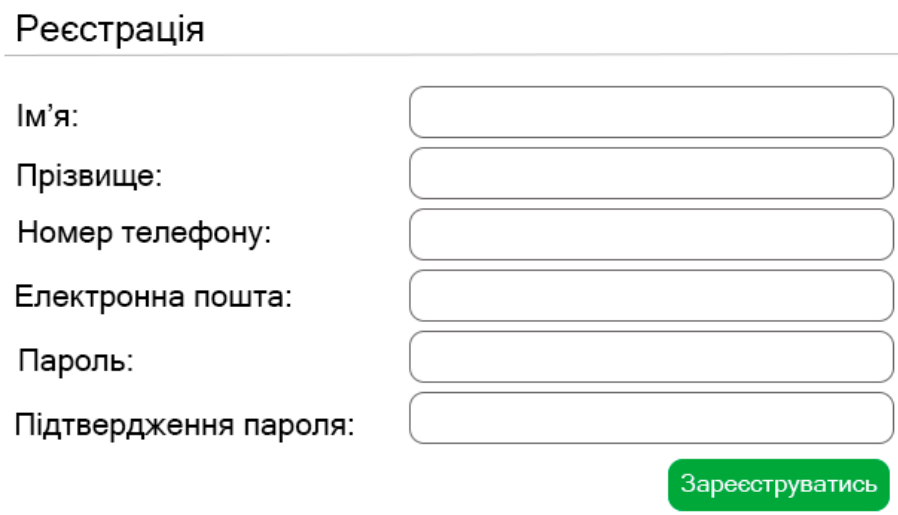


Рисунок 2.12 – Ескіз сторінки реєстрації

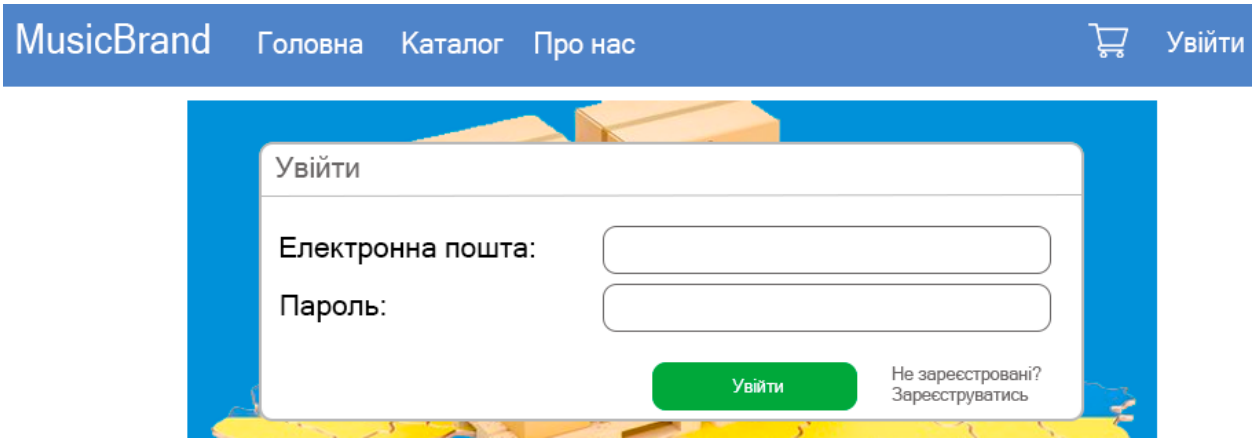


Рисунок 2.13 – Ескіз сторінки авторизації

Рисунок 2.14 – Ескіз панелі адміністрування

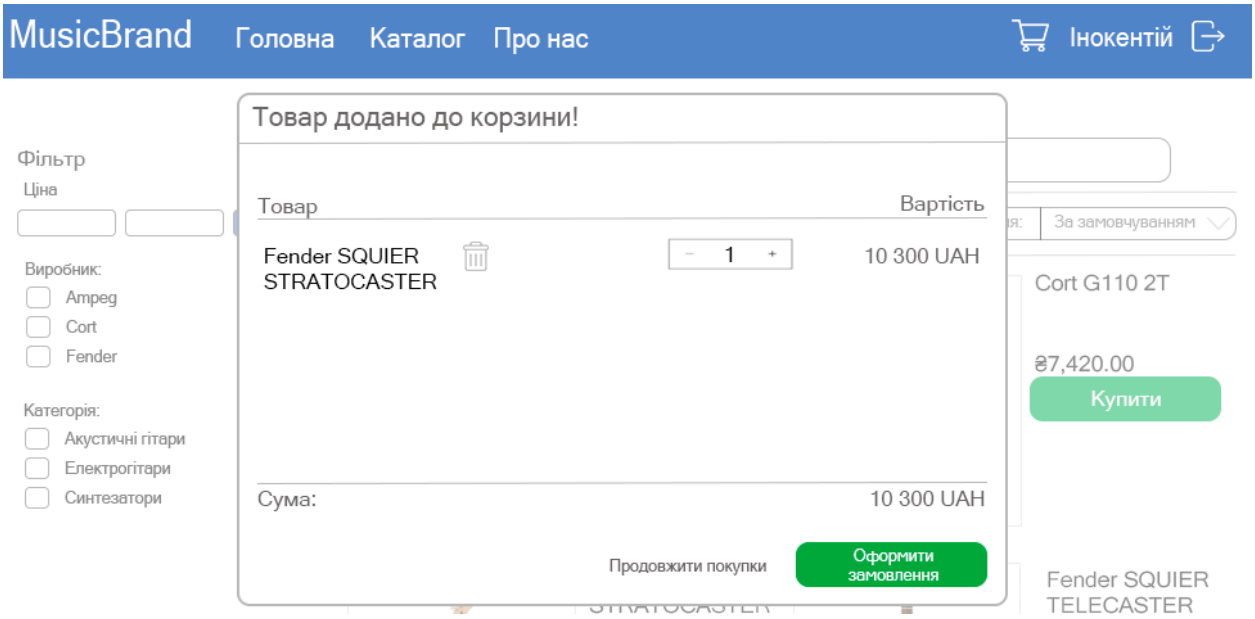


Рисунок 2.15 – Ескіз сторінки кошика

2.3.2 Технічний проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

Провівши аналіз ескізного проекту, було розроблено технічний проект вебсервісу для магазину з продажу музичних інструментів «MusicBrand». Для цього було побудовано статичну та динамічну моделі котрі репрезентують роботу системи. Статичну модель представлено у вигляді діаграми класів, а динамічну у вигляді діаграм послідовності.

Статична модель розроблюваного сервісу у вигляді діаграми класів представлено на рисунку 2.16:

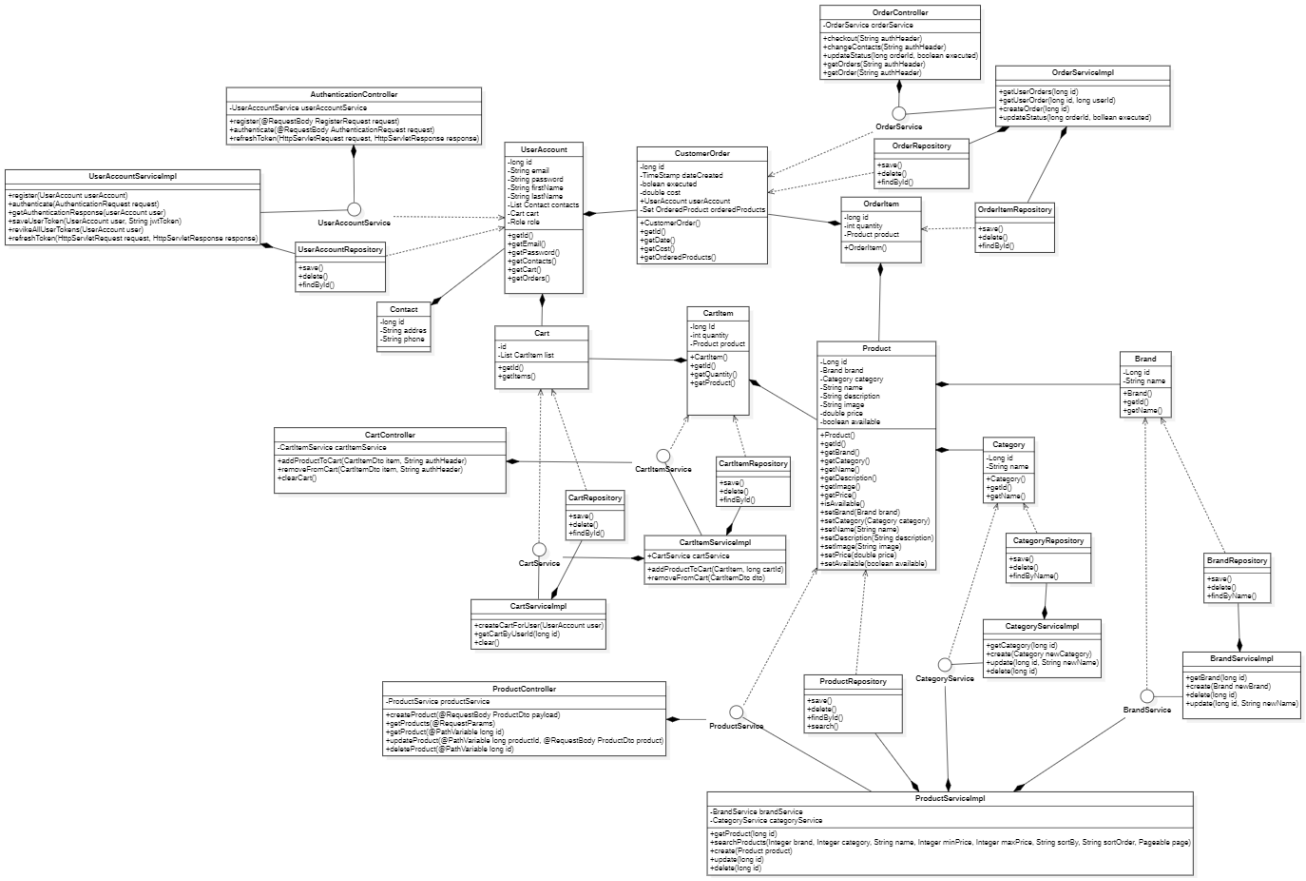


Рисунок 2.16 Діаграма класів

Специфікації класів вебсервісу для магазину продажу музичних інструментів «MusicBrand» приведено нижче:

Клас Product

Назва: Product

Короткий опис: Клас є моделлю товару, яка представляє таблицю в базі даних.

Атрибути:

- id – атрибут, котрий зберігає в собі id товару;
- brand – атрибут, котрий зберігає в собі екземпляр класу пов’язаного з цим товаром бренду;
- category – призначений для зберігання екземпляру класу пов’язаної з цим товаром категорії;

- name – атрибут, котрий призначений для зберігання назви даного товару;
- description – атрибут, котрий призначений для зберігання опису даного товару;
- image – атрибут, котрий призначений для зберігання шляху до зображення, котре пов’язане з даним товаром;
- price – атрибут, котрий призначений для зберігання вартості даного товару;
- available – атрибут, котрий призначений для зберігання інформації стосовно наявності даного товару.

Методи:

- Методи з префіксом "get" призначені для отримання інформації чи значень атрибутів чи властивостей.
- Методи з префіксом "set" призначені для встановлення значень атрибутів чи властивостей системи чи об'єктів.

Клас Category

Назва: Category

Короткий опис: Клас є моделлю категорії товару, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id даної категорії;
- name – призначений для зберігання назви даної категорії.

Методи:

- Методи з префіксом "get" призначені для отримання інформації чи значень атрибутів чи властивостей.
- Методи з префіксом "set" призначені для встановлення значень атрибутів чи властивостей системи чи об'єктів.

Клас Brand

Назва: Brand

Короткий опис: Клас є моделлю бренду товару, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id даного бренду;
- name – призначений для зберігання назви даного бренду.

Методи:

- Методи з префіксом "get" призначені для отримання інформації чи значень атрибутів чи властивостей.
- Методи з префіксом "set" призначені для встановлення значень атрибутів чи властивостей системи чи об'єктів.

Клас CategoryServiceImpl

Назва: CategoryServiceImpl

Короткий опис: Клас призначений для обробки даних сутності Category, яка представляє таблицю в базі даних.

Методи:

- getCategory() – отримати категорію;
- create() – створити категорію;
- update() – оновити інформацію про категорію;
- delete() – видалити категорію.

Клас BrandServiceImpl

Назва: BrandServiceImpl

Короткий опис: Клас призначений для обробки даних сутності Brand, яка представляє таблицю в базі даних.

Методи:

- getBrand() – отримати бренд;
- create() – створити бренд;

- update() – оновити інформацію про бренд;
- delete() – видалити бренд.

Клас ProductServiceImpl

Назва: ProductServiceImpl

Короткий опис: Клас призначений для обробки даних сутності Product, яка представляє таблицю в базі даних.

Атрибути:

- brandService – інтерфейс для взаємодії з сутністю Brand
- categoryService – інтерфейс для взаємодії з сутністю Category

Методи:

- getProduct() – отримати товар;
- searchProducts() – отримати список товарів за поданими параметрами;
- create() – створити товар;
- update() – оновити інформацію про товар;
- delete() – видалити товар.

Клас ProductController

Назва: ProductController

Короткий опис: Клас представляє собою контролер призначений для обробки HTTP-запитів та повернення даних у форматі JSON.

Атрибути:

- productService – інтерфейс для взаємодії з сутністю Product

Методи:

- getProduct() – отримати товар;
- getProducts() – отримати список товарів за поданими параметрами;
- createProduct() – створити товар;
- updateProduct() – оновити інформацію про товар;
- deleteProduct() – видалити товар.

Клас Cart

Назва: Cart

Короткий опис: Клас є моделлю кошика користувача, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id даної корзини;
- list – призначений для зберігання вмісту корзини.

Методи:

- Методи з префіксом "get" призначені для отримання інформації чи значень певних атрибутів чи властивостей.
- Методи з префіксом "set" призначені для встановлення значень атрибутів чи властивостей системи чи об'єктів.

Клас CartItem

Назва: CartItem

Короткий опис: Клас є моделлю елемента кошика користувача, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id даного елемента;
- quantity – призначений для зберігання інформації, стосовно кількості товару;
- product – атрибут, котрий зберігає в собі екземпляр товару.

Методи:

- Методи з префіксом "get" призначені для отримання інформації чи значень певних атрибутів чи властивостей.
- Методи з префіксом "set" призначені для встановлення значень атрибутів чи властивостей системи чи об'єктів.

Клас CartServiceImpl

Назва: CartServiceImpl

Короткий опис: Клас призначений для обробки даних сутності Cart, яка представляє таблицю в базі даних.

Методи:

- createCartForUser() – створити екземпляр кошику для користувача;
- getCartByUserId() – отримати екземпляр кошику користувача по id;
- clear() – очистити кошик.

Клас CartItemServiceImpl

Назва: CartItemServiceImpl

Короткий опис: Клас призначений для обробки даних сутності Cart, яка представляє таблицю в базі даних.

Атрибути:

- cartService – інтерфейс для взаємодії з сутністю Cart

Методи:

- addProductToCart() – додати екземпляр до кошику;
- removeFromCart() – видалити екземпляр з кошику.

Клас CartController

Назва: CartController

Короткий опис: Клас представляє собою контролер призначений для обробки HTTP-запитів та повернення даних у форматі JSON.

Атрибути:

- cartItemService – інтерфейс для взаємодії з сутністю CartItem.

Методи:

- addProductToCart() – додати товар до корзини користувача;
- removeFromCart() – видалити товар з корзини користувача;
- clearCart() – очистити корзину користувача.

Клас UserAccount

Назва: UserAccount

Короткий опис: Клас є моделлю користувача, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id даного користувача;
- email – призначений для зберігання адресу електронної пошти користувача;
- password – призначений для зберігання паролю користувача;
- firstName – призначений для зберігання імені користувача;
- lastName – призначений для зберігання фамілії користувача;
- contacts – атрибут призначений для зберігання списку екземплярів класу Contact даного користувача;
- cart – призначений для зберігання екземпляру класу Cart, котрий приналежить користувачу;
- role – призначений для зберігання екземпляру класу пов’язаної з цим користувачем ролі.

Методи:

- Методи з префіксом "get" призначені для отримання інформації чи значень певних атрибутів чи властивостей.
- Методи з префіксом "set" призначені для встановлення значень атрибутів чи властивостей системи чи об'єктів.

Клас Contact

Назва: Contact

Короткий опис: Клас є моделлю контактів(необхідних для відправки замовлення даних) користувача, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id;
- addres – призначений для адреси для відправки;

- phone – призначений для зберігання номеру телефона користувача.

Методи:

- Методи з префіксом "get" призначені для отримання інформації чи значень певних атрибутів чи властивостей.
- Методи з префіксом "set" призначені для встановлення значень атрибутів чи властивостей системи чи об'єктів.

Клас UserAccountServiceImpl

Назва: UserAccountServiceImpl

Короткий опис: Клас призначений для обробки даних сутності UserAccount, яка представляє таблицю в базі даних.

Методи:

- register() – зареєструвати користувача в системі;
- authenticate() – аутентифікувати користувача;
- getAuthenticationResponse() – генерує JWT-токен та токен оновлення на основі переданого об'єкта UserAccount;
- saveUserToken() – зберігає новий токен доступу для переданого користувача;
- revokeAllUserTokens() – відкликає всі токени доступу користувача;
- refreshToken() – оновлює токен доступу.

Клас AuthenticationController

Назва: AuthenticationController

Короткий опис: Клас представляє собою контролер призначений для обробки HTTP-запитів та повернення даних у форматі JSON.

Атрибути:

- userAccountService – інтерфейс для взаємодії з сутністю UserAccount.

Методи:

- register() – зареєструвати користувача в системі;
- authenticate() – аутентифікувати користувача;

- refreshToken() – оновити токен доступу.

Клас CustomerOrder

Назва: CustomerOrder

Короткий опис: Клас є моделлю замовлень користувача, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id замовлення;
- dateCreated – призначений для зберігання дати створення замовлення;
- executed – призначений для зберігання статусу обробки замовлення;
- cost – призначений для зберігання вартості замовлення;
- userAccount – призначений для зберігання екземпляру класу UserAccount, котрий відноситься до даного замовлення;
- orderItems – призначений для зберігання списку екземплярів товарів, котрі мають відношення до даного замовлення.

Клас OrderItem

Назва: OrderItem

Короткий опис: Клас є моделлю товарів котрі замовив користувач, яка представляє таблицю в базі даних.

Атрибути:

- id – призначений для зберігання id екземпляру;
- quantity – призначений для зберігання кількості товару;
- product – призначений для зберігання екземпляру класу товару, котрий замовили.

Клас OrderServiceImpl

Назва: OrderServiceImpl

Короткий опис: Клас призначений для обробки даних сутності CustomerOrder, яка представляє таблицю в базі даних.

Методи:

- getUserOrders() – отримати список замовлень, котрі створив користувач по id користувача;
- getUserOrder() – отримати екземпляр замовлення користувача, по id користувача та id замовлення;
- createOrder() – створити замовлення;
- updateStatus() – оновити статус замовлення;

Клас OrderController

Назва: OrderController

Короткий опис: Клас представляє собою контролер призначений для обробки HTTP-запитів та повернення даних у форматі JSON.

Атрибути:

- orderService – інтерфейс для взаємодії з сутністю CustomerOrder.

Методи:

- checkout() – оформлення замовлення;
- changeContacts() – змінити дані для відправки замовлення;
- updateStatus() – оновити статус замовлення;
- getOrders() – отримати список замовлень;
- getOrder() – отримати замовлення.

Наступним кроком розробки технічного проекту програмного забезпечення є побудова динамічної моделі розроблюваного сервісу. На основі розроблених раніше моделей, а саме на основі моделі варіантів використання і діаграми класів, була побудована динамічна модель. На рисунках 2.17 – 2.19 Приведені діаграми послідовності основних варіантів використання, котрі реалізують дану модель.

Діаграма послідовності для прецеденту «Пошук товару» представлена на рисунку 2.17.

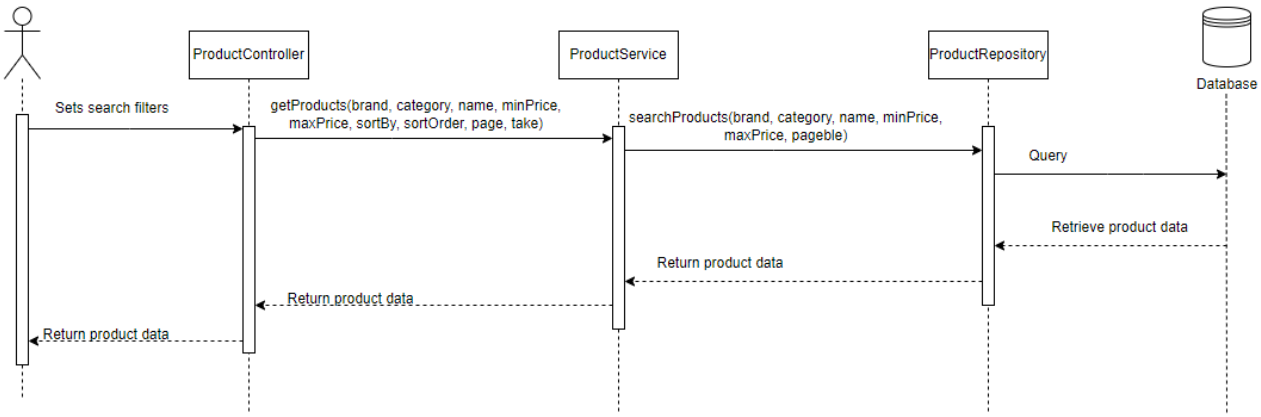


Рисунок 2.17 – Діаграма послідовності для прецеденту «Пошук товару»

Діаграма послідовності для прецеденту «Додати товар до кошика» представлена на рисунку 2.18.

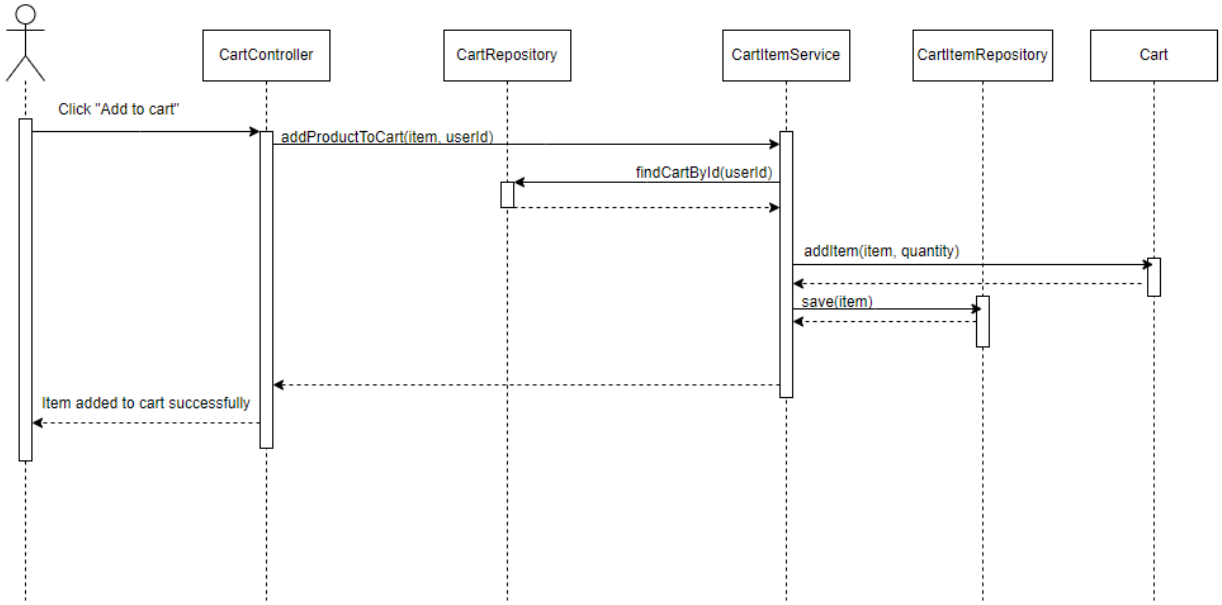


Рисунок 2.18 – Діаграма послідовності для прецеденту «Додати товар до кошика»

Діаграма послідовності для прецеденту «Додати товар», до котрої має доступ адміністратор, представлено на рисунку 2.19.

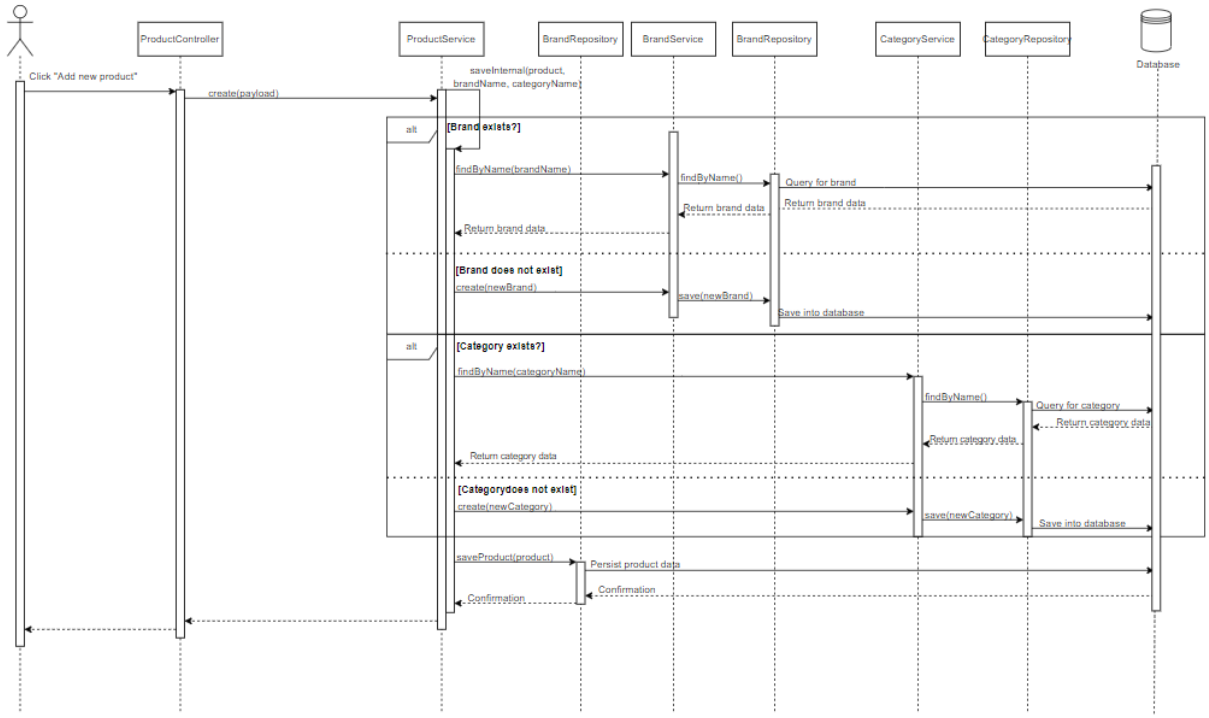


Рисунок 2.19 – Діаграма послідовності для прецеденту «Додати товар»

На основі концептуальної моделі даних створено логічну модель даних вебсервісу для магазину з продажу музичних інструментів «MusicBrand». Дана модель відображає структуру бази даних незалежно від використовуваної СУБД.

Модель представлено на рисунку 2.20:

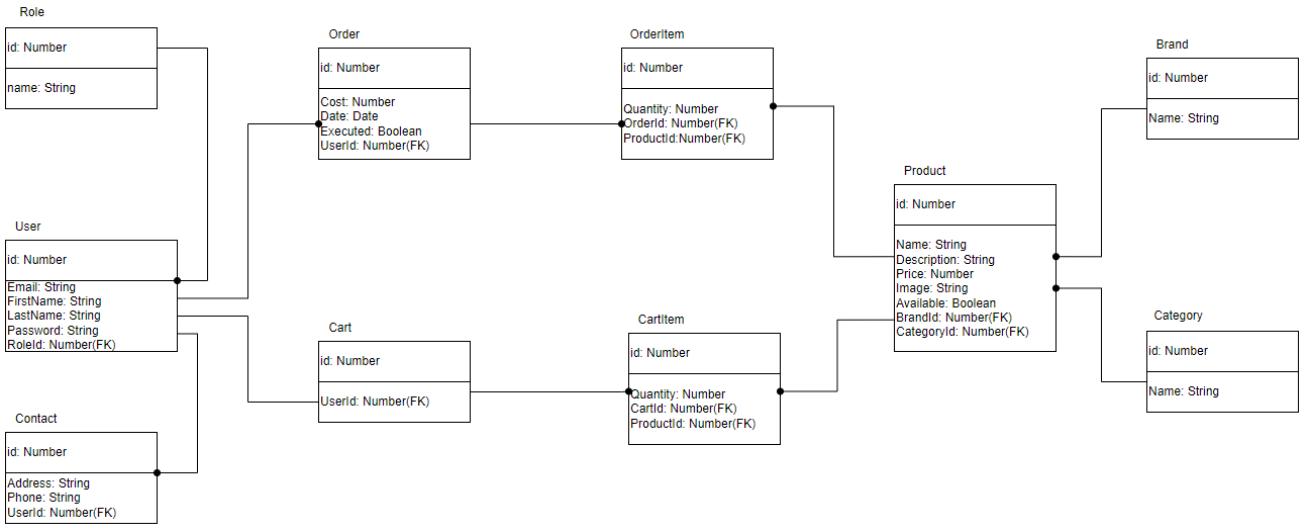


Рисунок 2.20 – Логічна модель даних

2.3.3 Розробка робочого проекту вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

Обґрунтування вибору мови програмування та засобів розробки

Для вирішення завдання розробки вебсервісу для магазину з продажу музичних інструментів «MusciBrand» доцільно використовувати об’єктно-орієнтовану методологію. Однією з найбільш потужних мов, підтримуючих об’єктно-орієнтований підхід, відносять мову програмування Java.

Java є однією з найпопулярніших мов програмування у світі. Ця мова програмування добре підходить для розробки складних веб-застосунків і масштабованих вебсервісів. При розробці вебсервісу для магазину з продажу музичних інструментів, використання мови Java та фреймворку Spring надає безліч переваг, які забезпечать ефективність та стійкість програми.

Завдяки фреймворку Spring можливо створити додаток, який може бути легко розширено. Завдяки модульній архітектурі Spring та його можливостям інтеграції з багатьма іншими технологіями, такими як Hibernate, JPA та JDBC, програму можна швидко та легко масштабувати [8]. Крім того, Spring забезпечує підтримку RESTful-сервісів, що полегшує інтеграцію з іншими програмами та сервісами.

Окрім цього, фреймворк Spring має інструменти безпеки, які допомагають захистити програму від зовнішніх загроз. Це дозволяє сфокусуватися на створенні функціональності, не відволікаючись на забезпечення безпеки. Spring Security забезпечує потужний механізм автентифікації та авторизації, а також підтримує механізми шифрування та захисту від атак CSRF та XSS [9].

Реалізація вебсервісу для магазину з продажу музичних інструментів

Нижче приведено фрагменти коду реалізованого функціоналу додавання товару до системи.

Фрагмент коду з класу ProductController:

```
@PostMapping
@PreAuthorize("hasRole('ADMIN')")
public Product createProduct(@RequestBody ProductDto payload)
{
    return productService.create(payload);
}
```

Метод @PostMapping є анотацією, що використовується фреймворці Spring, для позначення методу контролера, який відповідає на HTTP POST-запити. У цьому конкретному прикладі, метод createProduct обробляє POST-запити, які надходять на шлях, який співпадає з URL-адресою цього методу.

@PreAuthorize("hasRole('ADMIN')") – це ще одна анотація, яка використовується у Spring Security для встановлення прав доступу до методу. У цьому випадку, метод createProduct може бути викликаний лише користувачами, які мають роль "ADMIN".

Метод createProduct отримує об'єкт ProductDto як параметр методу, який передається у тілі POST-запиту (@RequestBody).

У методі createProduct викликається productService.create(payload), де productService є сервісом, який відповідає за обробку бізнес-логіки пов'язаної з продуктами. Метод create цього сервісу створює новий продукт на основі отриманих даних payload та зберігає його в системі.

Нижче приведено фрагмент коду з класу ProductService:

```
@Transactional
@Override
public Product create(ProductDto dto) {
    Product product = Product.builder()
            .image(dto.image)
            .description(dto.description)
            .name(dto.name)
            .available(dto.available)
            .price(dto.price)
            .build();
```

```
    return saveInternal(product, dto.brand, dto.category);
}
private Product saveInternal(Product product, String brandName,
String categoryName) {

    Optional<Brand> brand = brandService.findByName(brandName);;
   if (brand.isEmpty())
   {
        Brand newBrand = new Brand(brandName);
        brandService.create(newBrand);
        brand = Optional.of(newBrand);
    }
    product.setBrand(brand);
    Optional <Category> category =
categoryService.findByName(categoryName);
if(category.isEmpty())
    {
        Category newCategory = new Category(categoryName);
        categoryService.create(newCategory);
        category = Optional.of(newCategory);
    }
    product.setCategory(category);
    productRepository.save(product);
    return
productRepository.findById(Long.valueOf(product.getId())).get();
}
```

В процесі виконання данного фрагменту створюється новий об'єкт Product за допомогою шаблону "Builder". Значення для полів об'єкта Product отримуються з ProductDto.

Викликається метод saveInternal, який приймає створений об'єкт Product, а також імена бренду та категорії з ProductDto.

В методі saveInternal перевіряється, чи існує бренд з вказаним ім'ям. Встановлюється бренд для об'єкта Product якщо такий запис вже існує в базі.

Якщо бренд не знайдений, створюється новий об'єкт Brand з вказаним ім'ям за допомогою сервісного методу create об'єкта brandService.

Виконується пошук категорії за її ім'ям за допомогою сервісного методу findByName об'єкта categoryService. Якщо категорія не знайдена, створюється новий об'єкт Category з вказаним ім'ям за допомогою сервісного методу create об'єкта categoryService. В протилежному випадку, використовується знайдена категорія. Встановлюється категорія для об'єкта Product.

Зберігається об'єкт Product в репозиторії за допомогою методу save об'єкта productRepository.

Більш детально з кодом реалізованого вебсервісу для магазину з продажу музичних інструментів «MusicBrand» можна ознайомитись у Додатку Г.

Тестування вебсервісу для магазину з продажу музичних інструментів

Проведені тести виконані за принципом «чорної скриньки». Іноді даний метод називають тестуванням на основі специфікації. Метою кожного кейсу є перевірка на помилки в роботі вебсервісу.

Тест 1 – Перевірка можливості автентифікації користувача

Вхідними даними є:

Сценарій:

Для виконання автентифікації користувача, йому потрібно на головній сторінці сайту, у правій верхній частині сторінки, натиснути кнопку «Увійти». У формі котра відобразилась після натискання користувачу потрібно ввести дані для входу до свого профілю. Натиснувши кнопку «Увійти» очікуваним результатом у випадку коректно введених є те, що система перенаправляє користувача на головну сторінку і його ім’я відображається в хедері сторінки. Процес та результат автентифікації представлено на рисунках 2.21 – 2.22.

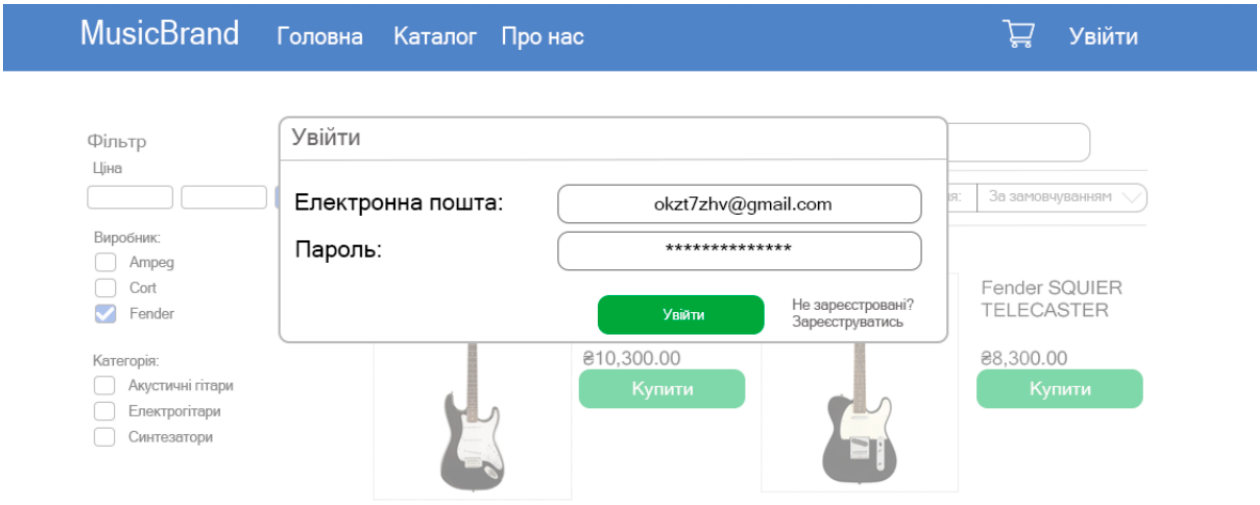
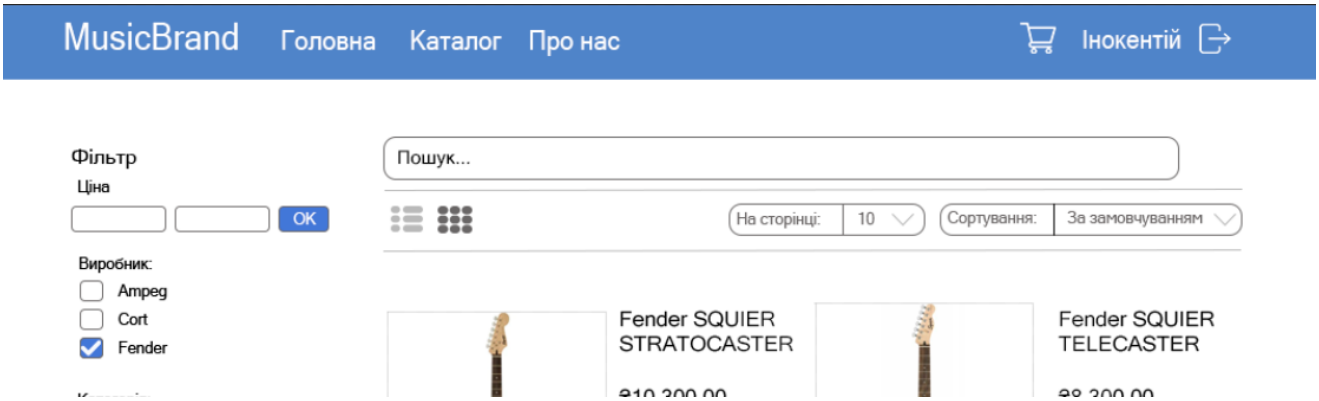


Рисунок 2.21 – Введення даних користувачем



2.22 – Результат тесту автентифікації

Тест 2 – Перевірка неможливості автентифікації за неіснуючою комбінацією логіна та паролю

Сценарій:

Користувач на головній сторінці сайту, у правій верхній частині сторінки, натискає кнопку «Увійти». У формі котра відобразилась після натискання користувач вводить дані для входу неіснуючого профілю. Натиснувши кнопку «Увійти» очікуваним результатом є виведення повідомлення про невірні логін та пароль.

Результат перевірки представлено на рисунку 2.23:

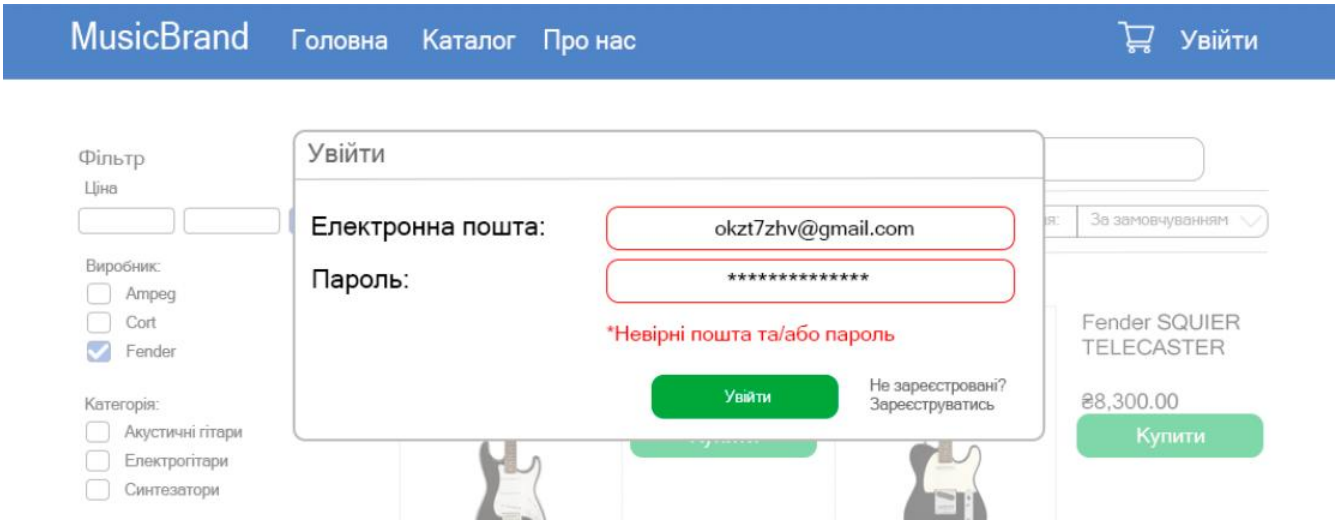


Рисунок 2.23 – Відображення повідомлення про неправильну комбінацію логіну та паролю

Тест 3 – Перевірка можливості додати товар до кошика

Сценарій:

Користувач попередньо авторизований. Користувач на сторінці товару натискає кнопку «Купити», після цього система додає товар до кошика та виводить повідомлення про успішне додання товару до кошика з його вмістом.

Результат перевірки представлено на рисунку 2.24:

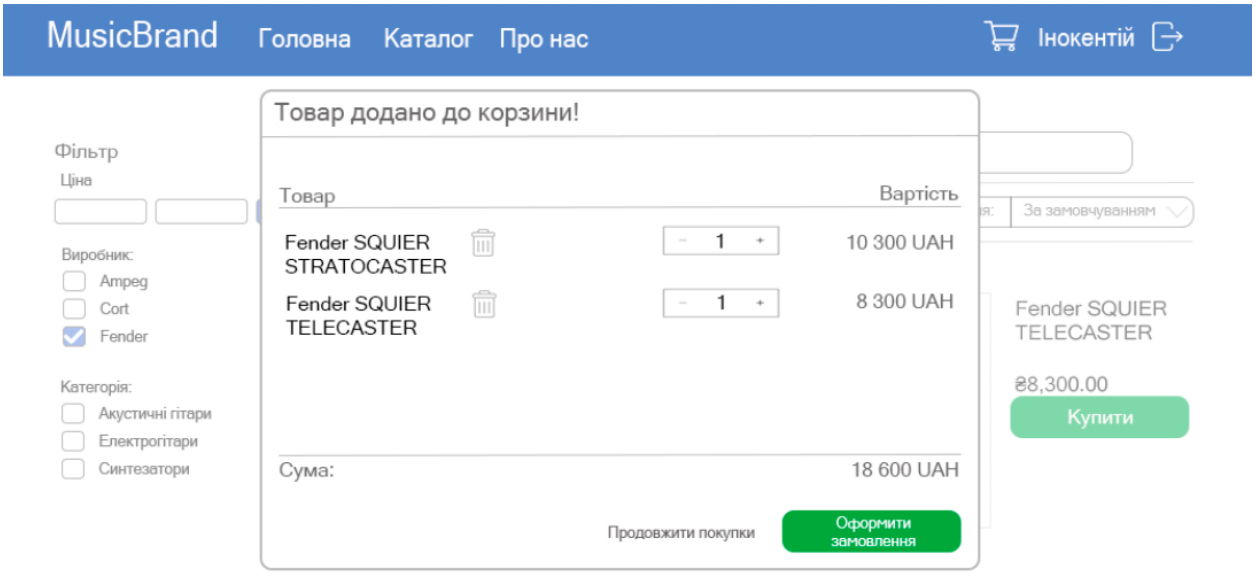


Рисунок 2.24 – Повідомлення про додавання товару до кошика з вмістом кошика

Тест 4 – Перевірка можливість видалити товар з кошика

Сценарій:

Користувач попередньо авторизований та додав товар до корзини. У вікні корзини відображаються товари котрі користувач попередньо додав до корзини, це представлено на рисунку 2.25. Користувач натискає кнопку «Видалити з кошика» навпроти назви необхідного товару. Система видаляє обраний товар з кошика та прибирає його зі списку товарів. Результат перевірки представлено на рисунку 2.26.

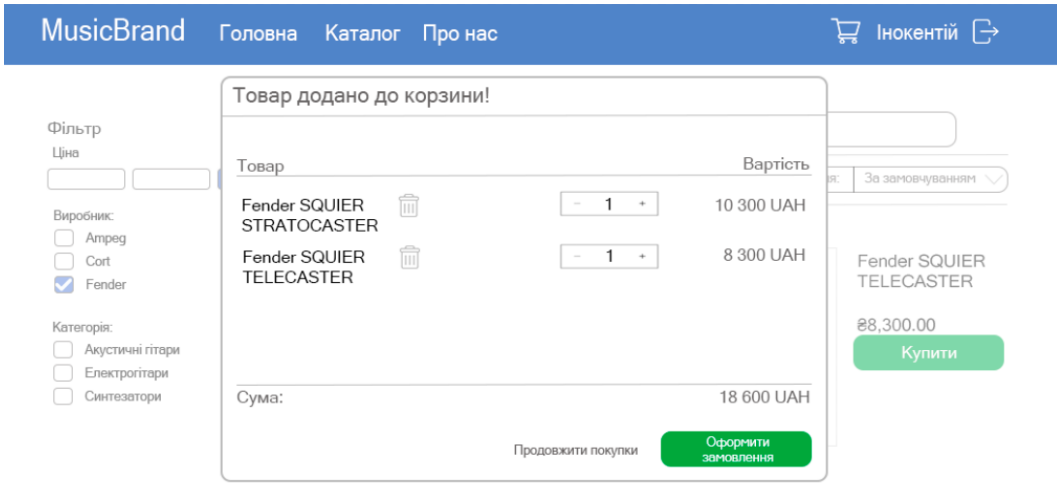


Рисунок 2.25 – Вміст кошика користувача

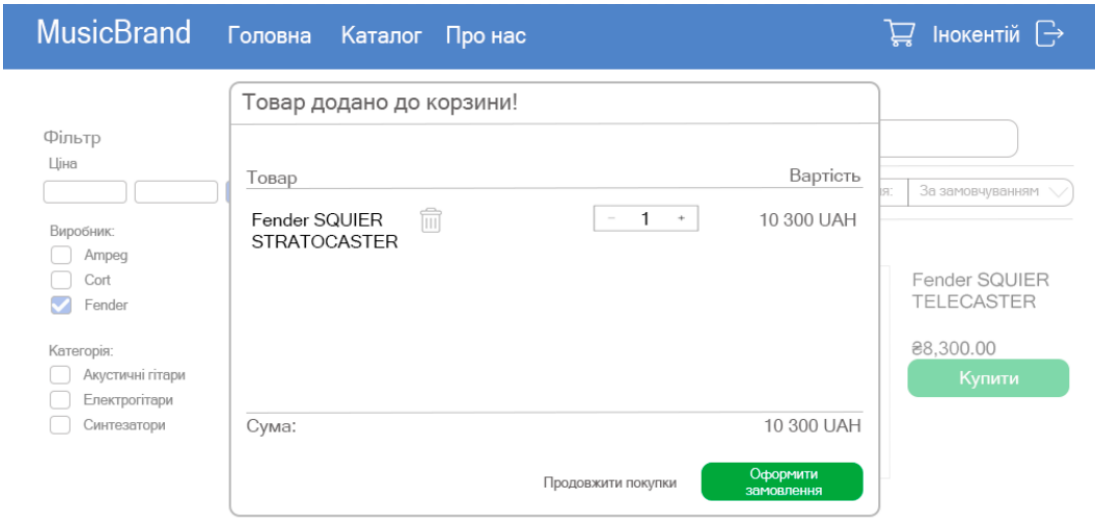


Рисунок 2.26 – Результат перевірки видалення товару з кошика

Тест 5 – Перевірка роботи фільтра пошуку товарів

Сценарій:

Користувач переходить на сторінку «Каталог». На цій сторінці користувачу відображається невідфільтрований асортимент товарів, що представлено на рисунку 2.27. Задаючи параметри фільтру у відповідному блоці, на сторінці відображаються лише товари, котрі відповідають параметрам фільтру. Результат перевірки роботи фільтра представлено на рисунку 2.28.

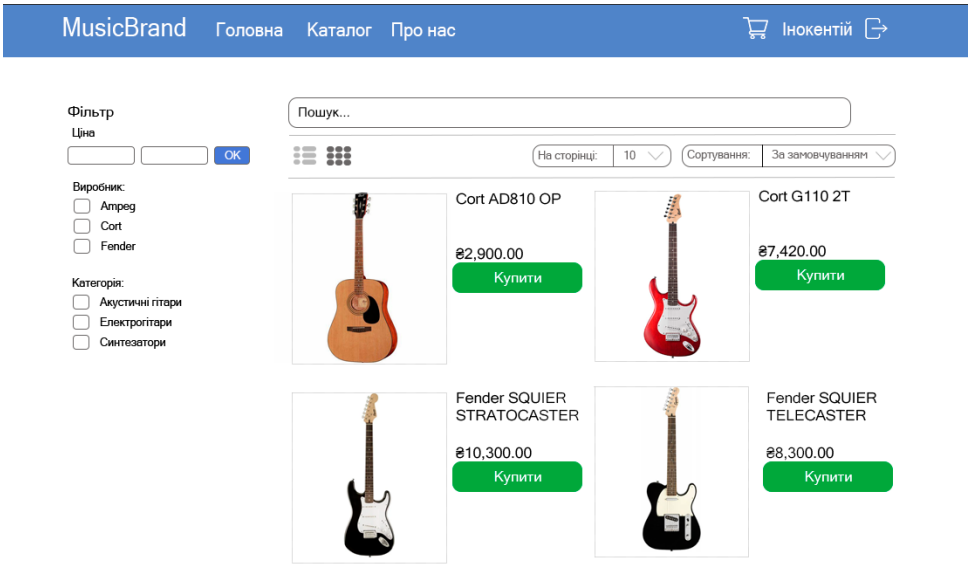


Рисунок 2.27 – Невідфільтрований асортимент товарів

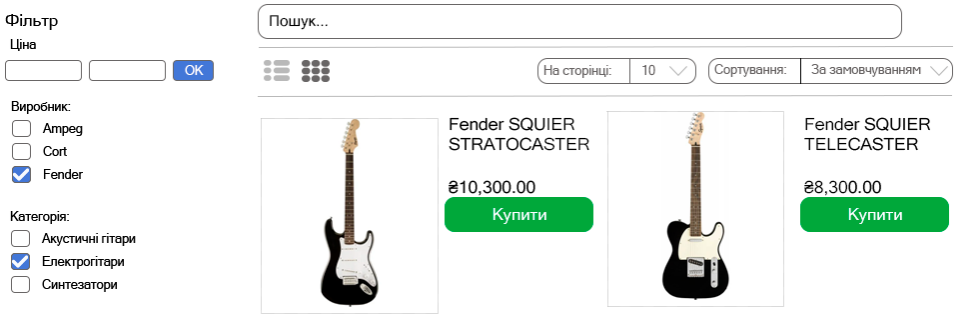


Рисунок 2.28 – Результат перевірки роботи фільтру

Випробування вебсервісу для магазину з продажу музичних інструментів «MusicBrand»

Наступним етапом розробки робочого проекту було випробування вебсервісу для магазину з продажу музичних інструментів «MusicBrand», згідно з програмою та методикою випробувань, яка наведена в Додатку Б. Під час проведення випробувань недоліків у функціонуванні системи не виявлено.

Було виконано випробування наступних варіантів використання:

- Автентифікація: виконано перехід на сторінку вебсервісу та відкрито форму авторизації. Після введення всіх необхідних даних, система провела перевірку введеної інформації на коректність. Після успішної перевірки, користувача було автентифіковано, що відповідає очікуваному результату.
- Реєстрація: виконано перехід на сторінку реєстрації. Після введення всіх необхідних даних, система провела перевірку введеної інформації на коректність. Після успішної перевірки, користувача було зареєстровано та автентифіковано, що відповідає очікуваному результату.
- Додати товар до кошика: виконано перехід на сторінку товару. Після натискання кнопки «Купити», користувачеві відобразилося відповідне повідомлення про додавання товару до кошику, що відповідає очікуваному результату.
- Видалити товар з кошика: відкрито вікно кошика. Після натискання кнопки «Видалити», товар зник зі списку кошика, що відповідає очікуваному результату.
- Пошук товарів: виконано перехід на сторінку каталогу та задано параметри фільтру у відповідному блоці сторінки. Система знайшла в базі товари, котрі відповідають встановленим параметрам фільтру, та вивела список товарів, що відповідає очікуваному результату.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

У результаті виконання кваліфікаційної роботи було вирішено практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблено вебсервіс для магазину з продажу музичних інструментів «MusicBrand».

Вебсервіс повністю відповідає вимогам до функціональних характеристик, вимогам до організації вхідних і вихідних даних та іншим вимогам, зазначеним у постановці задачі.

Вебсервіс для магазину з продажу музичних інструментів «MusicBrand» було розроблено з використанням сучасних веб-технологій.

Серверна частина реалізована на мові програмування Java з використанням фреймворку Spring. Вона відповідає за обробку запитів користувача, взаємодію з базою даних і виконання логіки бізнес-процесів. Для зберігання даних використовується реляційна база даних MySQL. Клієнтська частина сервісу написана з використанням HTML, CSS і JavaScript.

Розроблене програмне забезпечення реалізує наступні основні функції:

- Реєстрація;
- Авторизація;
- Додавання товарів до системи;
- Редагування інформації про товари в системі;
- Видалення товарів з системи;
- Додавання товарів в кошик;
- Видалення товарів з кошику;
- Оформлення замовлення;
- Оброблення замовлення.

Інструкція користувача наведена в Додатку В. Лістинг коду наведено у Додатку Д.

Під час розробки програмного забезпечення було проведене його тестування та випробування (згідно програми та методики випробувань, наведеної в Додатку Б), у ході яких була підтверджена повна працездатність програмного продукту.

Вихідний код розробленого вебсервісу для магазину з продажу музичних інструментів «MusicBrand» займає близько 424 КБ дискового простору. Сервісний шар розробленого додатку складається з 2462 рядків коду. Байт-код, котрий виконується на віртуальній машині JVM, займає приблизно 11 МБ дискового простору.

4 ОХОРОНА ПРАЦІ

4.1 Основні положення

Охорона праці є комплексом правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження здоров'я і працездатності працівників під час праці.

Безпека праці, як сфера практичної діяльності, спрямована на створення умов праці, які є безпечними і нешкідливими для працівників. Вона набуває все більшого значення на сучасному етапі розвитку виробництва.

Охорона праці має соціальне значення, яке проявляється у зростанні продуктивності праці, збереженні трудових ресурсів і збільшенні сукупного національного продукту.

Збереження трудових ресурсів і підвищення професійної активності працівників досягається шляхом поліпшення стану здоров'я і подовження середньої тривалості життя за допомогою поліпшення умов праці. Це також супроводжується високою трудовою активністю і збільшенням виробничого стажу. Підвищення професійного рівня також відбувається через зростання кваліфікації і майстерності.

4.2 Основні законодавчі та нормативно правові акти про охорону праці в Україні

До загальних законів, що встановлюють основні принципи охорони праці, в Україні входять Конституція України, Кодекс законів про працю та Закон України "Про охорону праці". Ці закони тісно пов'язані з іншими законами, такими як "Про охорону здоров'я", "Про пожежну безпеку", "Про

використання ядерної енергії та радіаційний захист", "Про забезпечення санітарного благополуччя населення", "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності", "Про підприємства в Україні", "Про оплату праці", "Про колективні договори та угоди", "Про зайнятість населення" та інші.

Основні закони про охорону праці в Україні включають:

- Стаття 43 Конституції України, яка гарантує право на працю і право на належні, безпечні і здорові умови праці.
- Стаття 45 Конституції, яка гарантує право всіх працюючих на щотижневий відпочинок, щорічну оплачувану відпустку, а також встановлення скороченого робочого дня для окремих професій і виробництв, або скорочену тривалість роботи в нічний час.
- Інші статті, що встановлюють право громадян на соціальний захист, включаючи забезпечення у разі повної, часткової або тимчасової втрати працездатності (стаття 46), охорону здоров'я, медичну допомогу та медичне страхування (стаття 49), право на інформацію про свої права та обов'язки (стаття 57) та інші загальні права громадян, включаючи право на охорону праці.
- Закон України "Про забезпечення санітарного та епідемічного благополуччя населення" встановлює потребу у гігієнічній регламентації небезпечних і шкідливих факторів фізичного, хімічного та біологічного характеру, які присутні у середовищі, де живе людина, а також вимоги щодо їх державної реєстрації, проектування, будівництва, розробки, виготовлення та використання нових засобів виробництва і технологій, а також гігієнічні вимоги до повітря у населених пунктах, виробничих і інших приміщеннях, а також вимоги щодо радіаційної безпеки та інше.
- Закон України "Про охорону здоров'я" регулює суспільні відносини у галузі охорони здоров'я з метою забезпечення гармонійного розвитку фізичних і духовних сил, високої працездатності та тривалого

активного життя громадян, а також усунення факторів, що шкодять їхньому здоров'ю.

- Закон України "Про загальнообов'язкове державне соціальне страхування від нещасного випадку на виробництві та професійного захворювання, які спричинили втрату працездатності".
- Закон "Про підприємства в Україні" визначає, що підприємство зобов'язане забезпечити своїм працівникам безпечні та нешкідливі умови праці і несе відповідальність за шкоду, заподіяну їхньому здоров'ю, у встановленому законом порядку.
- Закон України "Про колективні договори та угоди" передбачає, що колективний договір встановлює взаємні зобов'язання сторін щодо охорони праці, а угоди на державному, галузевому та регіональному рівнях регулюють основні принципи і норми реалізації соціально-економічної політики, зокрема щодо умов охорони праці.

Спеціальні законодавчі акти в галузі охорони праці включають:

- Державні стандарти України з питань безпеки праці, зокрема ДСТУ 2053-92 "Засоби індивідуального захисту органів дихання ізолюючі".
- Нормативно-правові акти з охорони праці.
- Локальні нормативні акти, що діють в межах підприємства.

4.3 Аналіз небезпечних і шкідливих факторів при роботі з комп’ютером

З розвитком сучасних технологій і все більшим використанням комп'ютерів у різних сферах діяльності, аналіз небезпечних і шкідливих факторів, пов'язаних з роботою з комп'ютером, стає дедалі важливішим завданням. Продовжений вплив цих факторів може вплинути на здоров'я та добробут користувачів. У цьому підрозділі розглянуто основні небезпечні і шкідливі фактори, їх вплив на людей та методи аналізу для виявлення та управління цими ризиками.

- Вплив електромагнітного та електростатичного випромінювання: Електромагнітне випромінювання, що виникає від комп'ютерів, моніторів, клавіатур і інших пристроїв, може мати негативний вплив на здоров'я людини. Продовжений вплив може призводити до таких проблем, як перенапруження очей, головні болі, порушення сну та загальне відчуття дискомфорту. Електростатичне випромінювання може призвести до статичного заряду на корпусі пристрою та створення неприємних електростатичних розрядів. Для зменшення впливу цих факторів слід забезпечити правильну заземленість пристроїв та використовувати екрани, що захищають від електромагнітного випромінювання.
- Вплив недостатньої освітленості робочого місця: Недостатня освітленість на робочому місці може призвести до зримого напруження, втоми очей, головних болів та погіршення зору. Недостатня освітленість також може спричинити помилки при роботі з комп'ютером. Для запобігання цим проблемам необхідно забезпечити належне освітлення робочого місця, використовуючи природне та штучне освітлення, а також правильно розташувати робочу зону відносно джерел світла.
- Вплив електричного струму: При неправильному підключенні або пошкодженому обладнанні може виникнути ризик виникнення електричних уражень. Високий електричний струм може призвести до опіків, спалень і навіть смерті. Щоб уникнути цих небезпек, необхідно забезпечити належне заземлення, перевіряти електричні кабелі на пошкодження та дотримуватися правил безпеки при підключенні та використанні обладнання.
- Вплив шуму: Робота з комп'ютером може бути пов'язана з шумом від вентиляторів, жорстких дисків та інших компонентів комп'ютера. Тривалий вплив шуму може призвести до стресу, погіршення концентрації, втоми та порушення сну. Для зменшення впливу шуму слід використовувати акустично ізольовані кабінети, забезпечити належне охолодження комп'ютера та використовувати шумозахисні матеріали.

• Вплив несприятливих умов мікроклімату: Недостатня вентиляція, перепади температури та вологості можуть впливати на комфорт та здоров'я користувачів комп'ютерів. Зайве нагрівання або охолодження може призвести до перегріву або переохолодження організму, спричинити дискомфорт, сухість шкіри та очей. Для зменшення впливу несприятливих умов мікроклімату слід забезпечити належну вентиляцію та контролювати температуру та вологість у приміщенні.

4.4 Розробка заходів щодо зменшення впливу небезпечних і шкідливих факторів при роботі з комп’ютером

Заходи щодо зменшення впливу небезпечних і шкідливих факторів при роботі з комп'ютером включають наступні дії:

1) Електромагнітне та електростатичне випромінювання:

- Забезпечення належної заземленості пристроїв та систем електроживлення.
- Використання екранів або фільтрів, що зменшують рівень електромагнітного випромінювання.
- Розташування робочих місць на відстані від джерел електромагнітних полів.
- Забезпечення правильного розташування та налаштування моніторів та інших пристроїв, що зменшує їхнє випромінювання.

2) Недостатня освітленість робочого місця:

- Використання штучного освітлення, такого як настільні лампи або світильники, для додаткового підсвічування робочої зони.
- Правильне розташування робочого місця відносно джерел природного світла.
- Використання ергономічних світильників з регульованим освітленням, що дозволяє налаштовувати яскравість та колірну температуру світла.

3) Електричний струм:

- Регулярна перевірка електричних кабелів та обладнання на пошкодження.
- Забезпечення належної заземленості та використання захисних пристроїв, таких як розетки з автоматичним відключенням електропостачання при перевищенні струму.
- Користування електричними пристроями, які відповідають стандартам безпеки та мають сертифікати відповідності.

4) Шум:

- Використання шумозахисних матеріалів для зменшення розсіювання шуму.
- Використання навушників або навушників з активним шумозаглушенням для зменшення впливу шуму на слух.
- Встановлення комп'ютерних систем охолодження, які працюють тихо.

5) Несприятливі умови мікроклімату:

- Забезпечення належної вентиляції та циркуляції повітря в приміщенні.
- Контроль температури та вологості повітря, щоб забезпечити комфортні умови роботи.
- Використання зволожувачів повітря в приміщенні з низькою вологістю.

Ці заходи допоможуть зменшити негативний вплив небезпечних і шкідливих факторів при роботі з комп'ютером та створити безпечне та комфортне робоче середовище. Проте важливо пам'ятати, що дотримання правильної пози та регулярні перерви для відпочинку також є важливими аспектами забезпечення здоров'я при роботі з комп'ютером.

ВИСНОВКИ

Мета даної кваліфікаційної роботи полягала в розв’язанні практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме у розробці вебсервісу, який дозволить клієнтам купувати музичні інструменти онлайн.

Мета кваліфікаційної роботи була досягнута. Для досягнення поставленої мети були вирішені наступні завдання:

- проведено дослідження предметної області;
- проведено аналіз існуючих аналогів вебсервісів продажу музичних інструментів;
- проведено аналіз методів розробки вебсервісів;
- створено проект програмного забезпечення;
- проведено тестування програмного забезпечення;
- ознайомлено з питаннями охорони праці при розробці ПЗ.

У розробленому вебсервісі реалізовані наступні функції:

- реєстрація;
- автентифікація;
- додавання товарів до системи;
- видалення товарів з системи;
- редагування інформації про товар в системі;
- додавання товарів до кошика;
- видалення товарів з кошика;
- оформлення замовлення;
- оброблення замовлення.

Крім того, важливо зазначити, що даний вебсервіс має можливість для подальшого розширення та доповнення функціоналом для вирішення додаткових задач, що робить створену систему досить гнучкою та в певній мірі універсальною.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Опис платформ для створення вебсервісів. URL: https://mercurio-cms.com/ru/blog/yak-obrati-platformu-dlya-e-commerce-cms-saas-chi-freymvork (дата звернення 05.03.2023)

2. MusicMarket – магазин продажу музичних інструментів та звукового обладнання. URL: https://muzmarket.com.ua/ua/ (дата звернення 06.03.2023)

3. 4Club – інтернет-магазин продажу музичних інструментів. URL: https://4club.com.ua (дата звернення 06.03.2023)

4. SoundMaster – продаж музичних інструментів онлайн. URL: https://soundmaster.ua (дата звернення 06.03.2023)

5. Dr. S. Suriya B.E. "UML Diagramming: A Case Study Approach." Видавництво: Auerbach Publications, 2022. 416с.

6. David Shirey. "21st Century RPG: /Free, ILE, and MVC: Free, ILE, and MVC." Видавництво: MC Press, 2017. 360с.

7. Martina Seidl, Marion Scholz. "UML @ Classroom: An Introduction to Object-Oriented Modeling." Видавництво: Springer, 2015. 218с.

8. Catalin Tudose. "Java Persistence with Spring Data and Hibernate Expanded Edition." Видавництво: Manning, 2023. 661с.

9. Laurentiu Spilca. "Spring Security in Action." Видавництво: Manning, 2020. 560с.

10. Про охорону праці. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 12.03.2023)

11. Про затвердження Вимог щодо безпеки та захисту здоров’я працівників. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#n14 (дата звернення: 12.03.2023)

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

Вступ

Повна назва програмного продукту: вебсервіс для магазину продажу музичних інструментів «Music Brand»

Сфера застосування: інтернет торгівля.

1. Підстави для розробки

Підставою для розробки даного програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 «Інженерія програмного забезпечення» в Національному університеті кораблебудування ім. адмірала Макарова.

2. Призначення розробки

2.1 Функціональне призначення

Для відвідувача сайту програма надає наступні можливості: перегляду товарів за категоріями з можливістю отримання детальної інформації про товар на відповідній сторінці та оформлення замовлення товару.

Для адміністратора програма дозволяє додавати/редагувати/видаляти інформацію про товари в каталозі магазину. Також передбачається можливість прибрати банер товару з каталогу у разі відсутності певного товару.

2.2 Експлуатаційне призначення

Програма має експлуатуватися на персональній ЕОМ користувача через браузер. З правами адміністратора програма запускається на комп'ютері адміністратора аналогічним чином.

3. Вимоги до програми або програмного виробу

3.1 Вимоги до функціональних характеристик

3.1.1 Вимоги до складу виконуваних функцій

Функції користувача:

• Реєстрація – реєстрація призначається для підвищення швидкості виконання повторних покупок на сайті. Користувач вводить свої дані, такі як: ПІБ, номер телефону, адреса електронної пошти, пароль. Після закінчення реєстрації отримує відповідне повідомлення. Додаткова інформація про тип та адресу доставки збережеться під час першої покупки.

• Авторизація – вхід до особистого кабінету здійснюється введенням логіну (номер телефону/email) та пароля.

• Переглянути товар – натисканням плитки з представленим товаром клієнт потрапляє на «односторінок». Тут відображається опис та характеристики товару (ціна, вага, тип, габарити тощо). На цій сторінці передбачена можливість оформлення замовлення/додавання в кошик.

• Оформити замовлення – якщо користувач авторизований, введення необхідних для оформлення замовлення (ПІБ, номер телефону, тип доставки, адреса, тип оплати) не потребуються. Вони автоматично з’являються в формі оформлення замовлення з можливістю коригування. Інакше, якщо користувач не авторизований, відображається сторінка, на якій клієнт вводить дані необхідні для оформлення замовлення. На сторінці оформлення замовлення відображається перелік товарів, які будуть входити в замовлення (ціна, кількість, назва, зображення).

• Помістити товар у кошик – під час перегляду товару доступна можливість натисканням додавання товару до списку для подальшого оформлення замовлення кількох товарів одночасно.

Функції адміністратора:

• Авторизація – вхід до особистого кабінету здійснюється введенням логіну та пароля.

• Управління каталогом товарів – адміністратор може додавати, редагувати та видаляти товари в каталозі.

• Управління замовленнями – адміністратор може переглядати, обробляти та відстежувати замовлення, що надходять до магазину.

Значення властивостей та характеристик за їх відсутності у сутності не виводяться.

У всіх областях, де використовується таблично-стрічкове представлення товару, застосовується – стандартне каталожне подання товару якщо не вказано інше.

Форми авторизації/реєстрації. Усі спливаючі форми введення виконуються в єдиному стилі.

3.1.2 Вимоги до організації вхідних та вихідних

Дані про товари зберігаються у базі даних. СУБД забезпечує розмежування прав доступу до даних – надає клієнту права на читання, а оператору – на читання та запис.

3.1.3 Вимоги до часових характеристик

Вимог до часових характеристик не висувається.

3.2 Вимоги до надійності

3.2.1 Вимоги до забезпечення надійного (сталого) функціонування програми

Надійне (стійке) функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких подано нижче:

- захист від некоректних дій користувача(під час розпізнавання);

3.2.2 Час відновлення після відмови

Вимоги до часу відновлення після відмови не пред’являються.

3.2.3 Відмови через некоректні дії оператора

Відмова можлива при наступних умовах:

- внаслідок закриття програми під час розпізнавання, в цьому випадку користувачеві показується додаткове вікно, чи впевнений він що хоче закрити програму;

- якщо розмір файлу перевищує розмір оперативної пам’яті, то з’являється вікно про недостачу оперативної пам’яті для розпізнавання файлу.

3.3 Умови експлуатації

3.3.1 Кліматичні умови експлуатації

Кліматичні умови експлуатації, при яких повинні забезпечуватися задані характеристики, повинні відповідати вимогам, що пред'являються до технічних засобів в частині умов їх експлуатації.

3.3.2 Вимоги до чисельності та кваліфікації користувачів

Користувач повинен мати базові навички роботи з комп’ютером.

3.4 Вимоги до складу і параметрів технічних засобів

Сервер повинен відповідати наступному складу технічних засобів з мінімальними параметрами:

- Операційна система: Linux або Windows Server
- Процесор: 64-бітний двох або більше ядерний процесор з тактовою частотою не менше 2 ГГц
- Оперативна пам'ять: не менше 4 ГБ
- Диск: не менше 100 ГБ

Для роботи клієнтської частини необхідні наявність браузера з підтримкою HTML-5/CSS-3 та доступ до виходу у мережу Internet.

3.5 Вимоги до маркування та упаковки

3.5.1 Вимоги до маркування

Програма поставляється вже встановленою на спеціальному обладнанні та готовою до використання.

3.5.2 Вимоги до упаковки

Упаковка програмного виробу повинна здійснюватися в пакувальну тару підприємства-виробника.

3.6 Спеціальні вимоги

Програма повинна забезпечувати взаємодію з користувачем за допомогою графічного інтерфейсу користувача, розробленого відповідно до рекомендацій компанії-виробника операційної системи.

4. Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

– технічне завдання;

– програму та методику випробувань;

– керівництво користувача;

– опис програми;

– код програми.

5. Техніко-економічні показники

У даній кваліфікаційній роботі техніко-економічні показники не розраховуються.

6. Етапи роботи

Етапи роботи представлені у таблиці А.1

Таблиця А.1 – Етапи роботи

Номер	Назва етапів роботи	Терміни виконання
1.	Аналіз практичної задачі інженерії ПЗ	31.03.2023
2.	Аналіз вимог до ПЗ	07.04.2023
3.	Розробка архітектури ПЗ	21.04.2023
4.	Розробка ескізного проекту ПЗ	05.05.2023
5.	Розробка технічного проекту ПЗ	10.05.2023
6.	Розробка робочого проекту	12.05.2023
7.	Реалізація ПЗ	17.05.2023
8.	Тестування ПЗ	21.05.2023
9.	Підготовка розділу Результати розробки ПЗ	23.05.2023

випробування

7. Порядок контролю та приймання

Для контролю приймання має бути надано опис програми, а також програму і методику випробувань.

Якщо програма не пройшла випробування, виконавець зобов’язаний виправити помилки і недоліки в строк, не більш ніж за 1 тиждень від дня випробування.

ДОДАТОК Б – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

1. Об’єкт випробування

Об’єктом випробувань є вебсервіс для магазину з продажу музичних інструментів «MusicBrand».

2. Мета випробувань

Метою проведення випробувань є перевірка працездатності вебсервісу, перевірка відповідності характеристик розробленого вебсервісу та вимог викладених в технічному завданні.

3. Вимоги до програми

При проведенні випробувань функціональних характеристик, вебсервіс для магазину з продажу музичних інструментів підлягає перевірці на відповідність вимогам, викладеним у пункті «Вимоги до функціональних характеристик» технічного завдання.

4. Вимоги до програмної документації

До складу програмної документації повинні входити наступні документи:

- Технічне завдання;
- Програма та методика випробувань;
- Інструкція користувача;
- Текст програми;
- Опис програми.

5. Засоби і порядок випробувань

5.1 Склад технічних засобів

До складу технічних засобів, які використовувалися під час випробувань:

- Процессор: 2 ядра, 2.0 ГГц;
- Оперативна пам’ять: 8 ГБ;
- Жорсткий диск: 128 ГБ;
- Інтернет з’єднання зі швидкістю 10 Мбіт/с.

5.2 Склад програмних засобів

До складу програмних засобів, які використовувались при проведенні випробувань:

- Операційна система Windows 10;
- Браузер Chrome версії 102.

5.3 Порядок проведення випробувань

Порядок проведення випробувань:

- Перевірка вимог до функціональних характеристик;
- Виконання тестів в довільному порядку;
- Аналіз отриманих результатів тестування;
- Відповідність програмного продукту вимогам.

Після виявлення недоліків у функціонуванні системи, створюється список цих проблем, який обговорюється з розробником для визначення терміну їх виправлення. Після цього замовник проводить повторне тестування, охоплюючи всі аспекти системи, можливо навіть застосовуючи нові або додаткові тести.

6. Методи випробувань

Методом випробувань для тестування розробленого програмного забезпечення було обрано метод чорної скриньки. Цей метод дозволяє перевіряти програму без прямого доступу до коду.

Перелік випробувань основних функціональних можливостей:

- перевірка можливості реєстрації користувача;
- перевірка можливості автентифікації користувача;
- перевірка можливості додати товар до кошика;
- перевірка можливості видалити товар з кошика;
- перевірка роботи фільтра пошуку товарів.

В процесі випробувань необхідно перевірити правильність виконання визначених функціональних можливостей.

ДОДАТОК В – ІНСТРУКЦІЯ КОРИСТУВАЧА ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

Для роботи з вебсервісом для магазину з продажу музичних інструментів «MusicBrand» потрібно перейти до вебсайту за допомогою браузера з підтримкою HTML-5/CSS-3, після чого з’явиться головна сторінка вебсервісу.

Головна сторінка наведена на рисунку В.1.

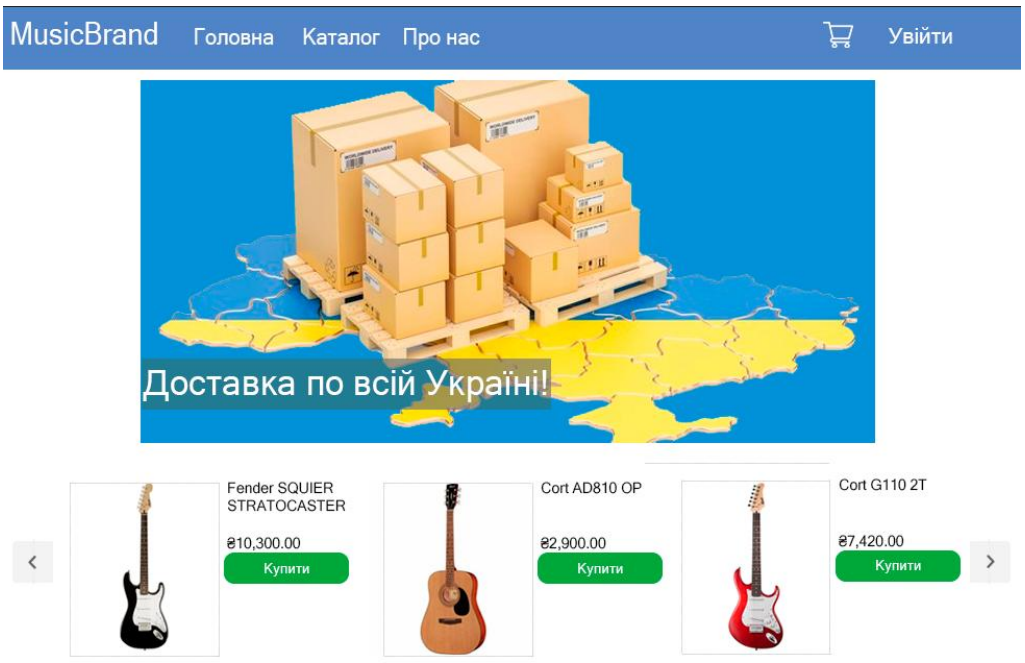


Рисунок В.1 – Головна сторінка

Для отримання доступу до всіх функцій сайту, користувач повинен пройти процес автентифікації, якщо він вже зареєстрований, або зареєструватися в разі відсутності облікового запису. Для проходження автентифікацій необхідно на головній сторінці натиснути кнопку «Увійти», що відкриє вікно авторизації, де користувач повинен ввести адресу пошти та пароль, потім натиснути кнопку «Увійти». Даний крок наведено на рисунках В.2 – В.3.

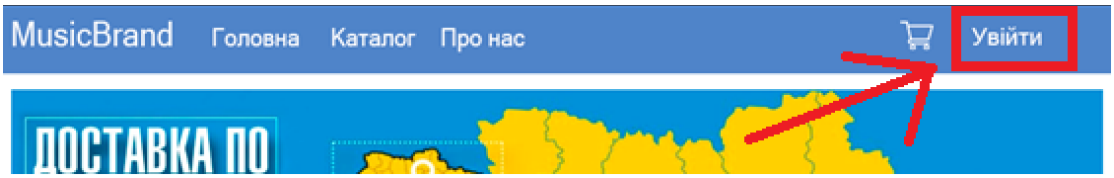


Рисунок В.2 – Кнопка «Увійти»

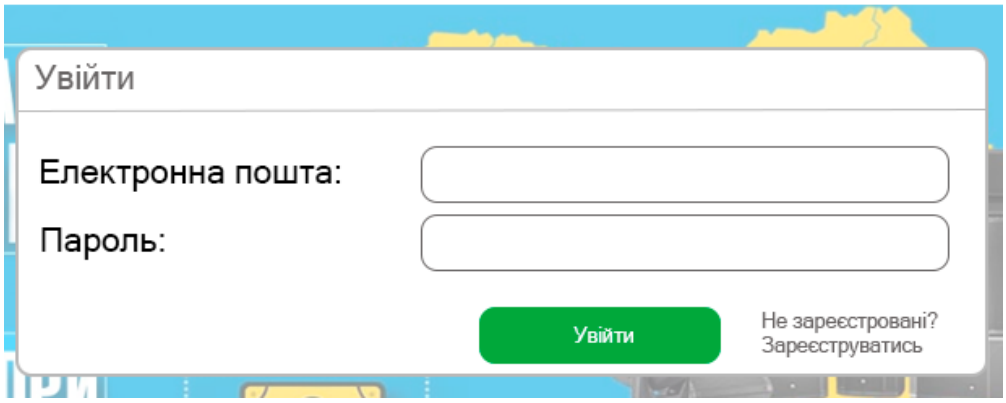


Рисунок В.3 – Вікно автентифікації

Якщо користувач не був зареєстрований, то в вікні автентифікації він має натиснути кнопку «Не зареєстровані? Зареєструватись». На сторінці, що відкриється, користувач має ввести всі необхідні дані та натиснути кнопку «Зареєструватись».

На рисунках В.4 – В.5 представлено процес реєстрації.

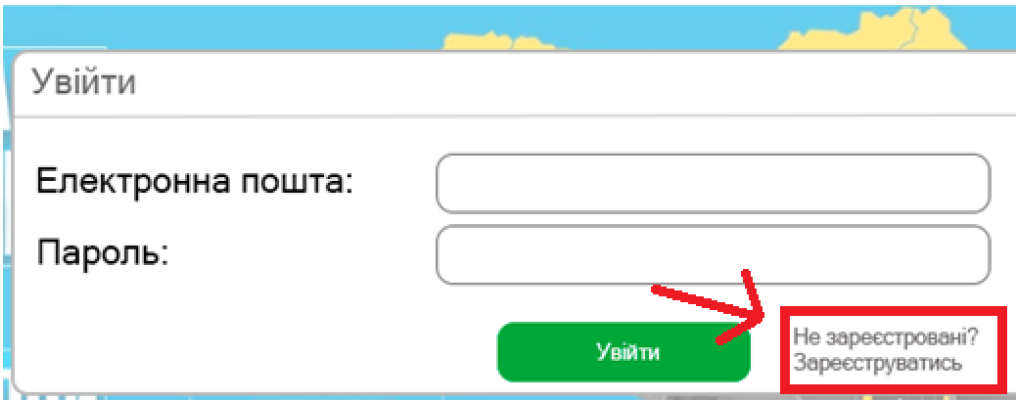


Рисунок В.4 – Кнопка реєстрації у вікні автентифікації

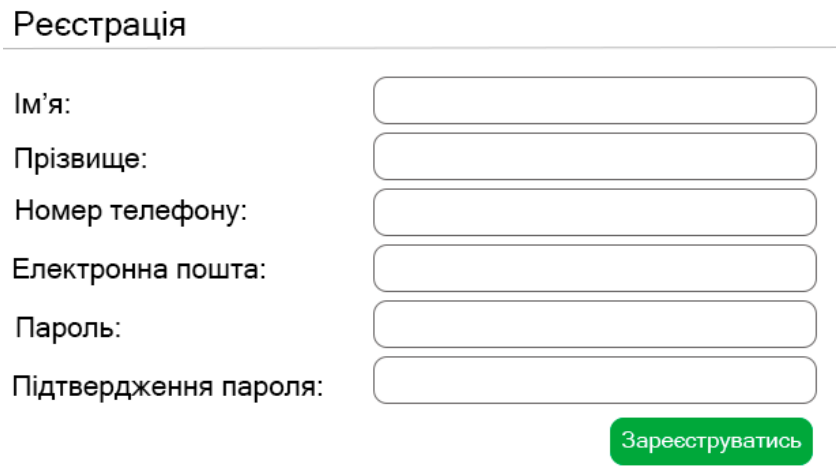


Рисунок В.5 – Сторінка реєстрації

Користувач має можливість додати товар до корзини. Для цього йому попередньо необхідно авторизуватися. Для того, щоб додати товар до корзини, потрібно натиснути на кнопку «Купити» навпроти відповідного товару на сторінці «Каталог» або, натисканням на назву товару, перейти на сторінку товару та натиснути «Купити». Після того, як товар був доданий до корзини, програма сповістить користувача про це, відображаючи відповідне повідомлення з вмістом корзини. Даний крок наведено на рисунках В.6 – В.7.

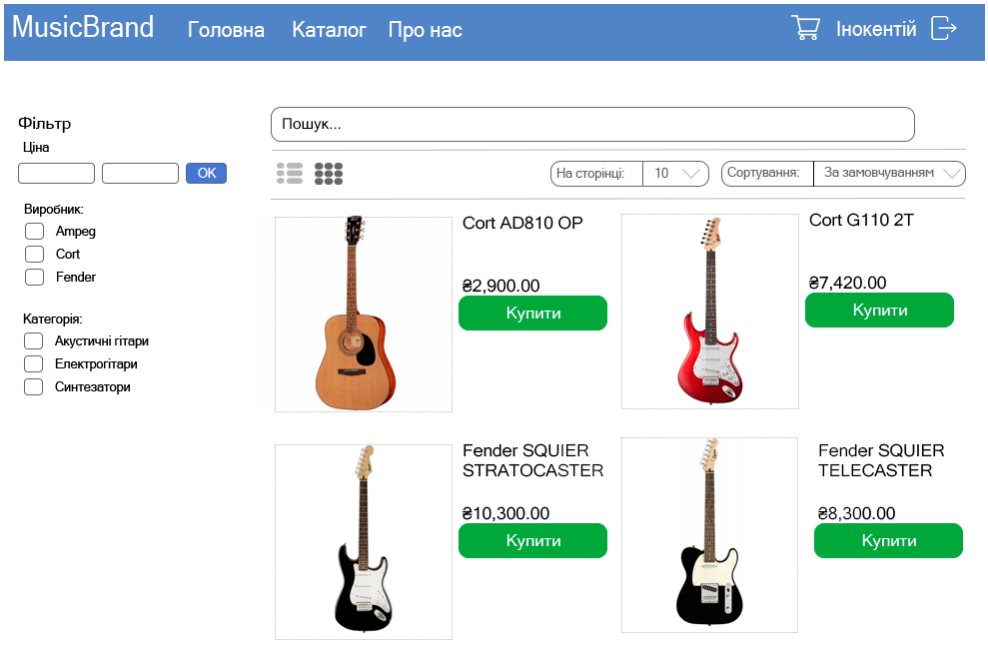


Рисунок В.6 – Сторінка «Каталог»

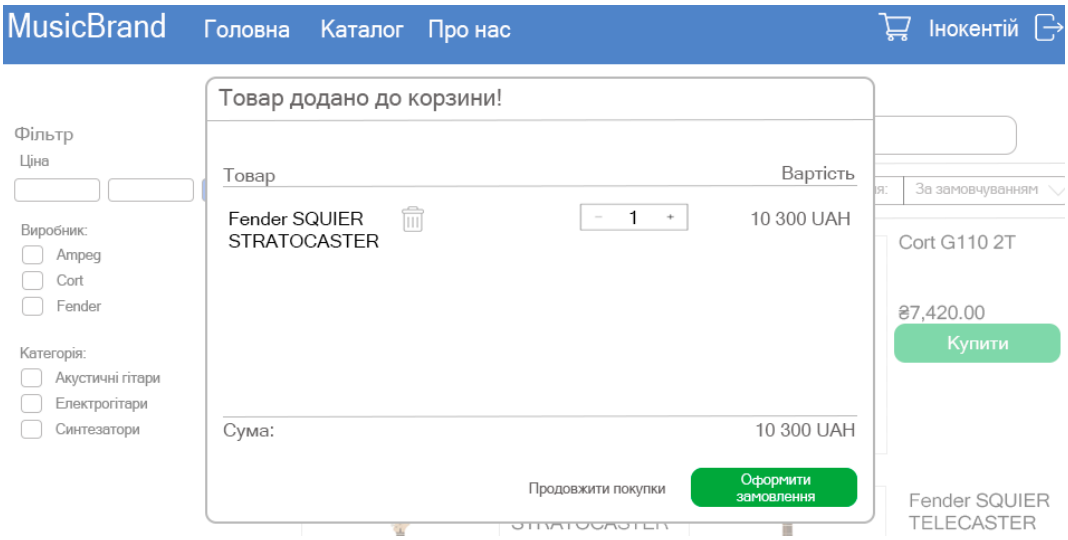


Рисунок В.7 – Повідомлення про успішне додавання товару до кошика

Після цього, користувач має можливість редагувати вміст кошика відповідними елементами управління або перейти до оформлення замовлення натиснувши кнопку «Оформити замовлення». На сторінці, що відкрилася, потрібно заповнити всі обов'язкові поля та натиснути кнопку «Оформити замовлення».

На рисунку В.8 представлено процес оформлення замовлення.

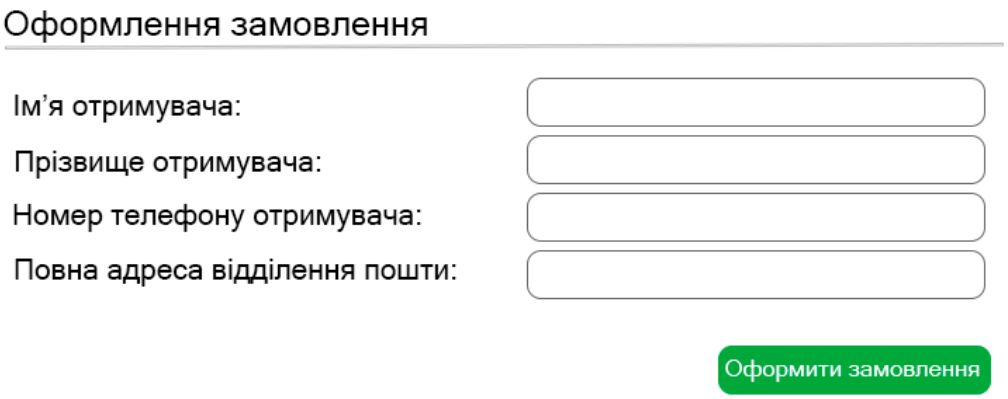


Рисунок В.8 – Сторінка оформлення замовлення

Якщо користувач увійшов в систему як адміністратор, то програма його відразу перенаправляє на сторінку адміністрування. На цій сторінці адміністратору надаються розширені можливості, такі як управління каталогом товарів та обробка замовлень. В адміністраторському режимі

адміністратор може додавати, редагувати та видаляти товари з каталогу. Крім того, адміністратор має доступ до замовлень, що надійшли, і може їх обробляти, а саме встановлювати статус замовлення. Завдяки цим можливостям адміністратор може ефективно управляти всіма аспектами системи та забезпечувати безперебійну роботу і задоволення потреб користувачів.

На рисунку В.9 представлено сторінку адміністрування.

Рисунок В.9 – Сторінка адміністрування

ДОДАТОК Г – ТЕКСТ ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

/*Product.java*/

```
package com.example.demo.models;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Optional;
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "products", schema = "storфe", catalog = "")
public class Product {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", insertable = false, updatable = false,
nullable = false)
    private Long id;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "brand_id", nullable = false)
    private Brand brand;
    @ManyToOne(fetch = FetchType.EAGER)
    @JoinColumn(name = "category_id", nullable = false)
    private Category category;
    @Column(name = "name", nullable = false)
    private String name;
    @Column(name = "description")
    private String description;
    @Column(name = "image")
```

```
    private String image;
    @Column(name = "price", nullable = false)
    private int price;
    @Column(name = "available", nullable = false)
    private boolean available;
    public boolean isAvailable() {
        return available;
    }
    public void setAvailable(boolean available) {
        this.available = available;
    }
    public void setBrand(Optional<Brand> brand) {
        this.brand = brand.get();
    }
    public void setCategory(Category category) {
        this.category = category;
    }
}
```

/*Category.java*/

```
package com.example.demo.models;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Objects;
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "categories", schema = "store", catalog = "")
public class Category {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
```

```
    @Id
    @Column(name = "id", nullable = false)
    private Integer id;
    @Basic
    @Column(name = "name", nullable = false, unique = true)
    private String name;
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        Category that = (Category) o;
        return id == that.id && Objects.equals(name, that.name);
    }
    @Override
    public int hashCode() {
        return Objects.hash(id, name);
    }
}
```

/*Brand.java*/

```
package com.example.demo.models;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import java.util.Objects;
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "brands", schema = "store", catalog = "")
public class Brand {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id")
    private Integer id;
    @Basic
    @Column(name = "name", nullable = false, unique = true)
    private String name;
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        Brand brands = (Brand) o;
        return id == brands.id && Objects.equals(name,
brands.name);
    }
    @Override
    public int hashCode() {
        return Objects.hash(id, name);
    }
}
```

/*UserAccount.java*/

```
package com.example.demo.models;
import com.example.demo.security.token.Token;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.springframework.security.core.GrantedAuthority;
import
org.springframework.security.core.authority.AuthorityUtils;
import
org.springframework.security.core.userdetails.UserDetails;
```

```
import java.util.*;
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "user_accounts", schema = "store", catalog = "")
public class UserAccount implements UserDetails {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "id", insertable = false, updatable = false,
nullable = false)
    private Long id;
    @Basic
    @Column(name = "firstName", nullable = false, length = 255)
    private String firstName;
    @Basic
    @Column(name = "lastName", nullable = false, length = 255)
    private String lastName;
    @Basic
    @Column(name = "email", nullable = false, length = 255)
    private String email;
    @Basic
    @Column(name = "password", nullable = false, length = 1000)
    private String password;
    @OneToOne(mappedBy = "userAccount", cascade =
CascadeType.ALL)
    private Cart cart;
    @OneToMany(mappedBy = "userAccountByIdUser")
    private Collection<Contact> contactsById;
    @OneToMany(mappedBy = "userAccount", cascade =
CascadeType.ALL, orphanRemoval = true)
    private List<CustomerOrder> orders = new ArrayList<>();
    @Enumerated(EnumType.STRING)
```

```
    private Role role;
    @OneToMany(mappedBy = "user", cascade = CascadeType.ALL)
    private List<Token> tokens;
    public UserAccount(String firstName, String lastName, String
email, String password) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.email = email;
        this.password = password;
    }
    public Long getId() {
        return id;
    }
    public void setId(Long id) {
        this.id = id;
    }
    public String getFirstName() {
        return firstName;
    }
    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }
    public String getLastName() {
        return lastName;
    }
    public void setLastName(String lastName) {
        this.lastName = lastName;
    }
    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }
```

```
    @Override
    public Collection<? extends GrantedAuthority>
getAuthorities() {
        if (this.role.name().equals(new String("USER")))
        {
            return
AuthorityUtils.createAuthorityList("ROLE_USER");
        }
        else
        {
            return
AuthorityUtils.createAuthorityList("ROLE_ADMIN");
        }
    }
    @Override
    public String getPassword() {
        return password;
    }
    @Override
    public String getUsername() {
        return email;
    }
    @Override
    public boolean isAccountNonExpired() {
        return true;
    }
    @Override
    public boolean isAccountNonLocked() {
        return true;
    }
    @Override
    public boolean isCredentialsNonExpired() {
        return true;
    }
    @Override
```

```
    public boolean isEnabled() {
        return true;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return
false;
        UserAccount that = (UserAccount) o;
        return id == that.id && Objects.equals(firstName,
that.firstName) && Objects.equals(lastName, that.lastName) &&
Objects.equals(email, that.email) && Objects.equals(password,
that.password);
    }
    @Override
    public int hashCode() {
        return Objects.hash(id, firstName, lastName, email,
password);
    }

    public Cart getCart() {
        return cart;
    }
    public void setCart(Cart cartById) {
        this.cart = cartById;
    }
}
```

/*Cart.java*/

```
package com.example.demo.models;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
```

```
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
import org.hibernate.annotations.GenericGenerator;
import java.util.*;
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "carts", schema = "store", catalog = "")
public class Cart {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    @OneToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "user_id", referencedColumnName = "id")
    private UserAccount userAccount;
    @OneToMany(mappedBy = "cart", cascade = CascadeType.ALL,
orphanRemoval = true)
    private List<CartItem> items = new ArrayList<>();
    public Cart(UserAccount user) {
        this.userAccount = user;
    }
}
```

/*CartItem.java*/

```
package com.example.demo.models;
import com.example.demo.dto.CartItemDto;
import jakarta.persistence.*;
import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;
@Data
```

```
@Builder
@NoArgsConstructor
@AllArgsConstructor
@Entity
@Table(name = "cart_items", schema = "store", catalog = "")
public class CartItem {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;
    private int quantity;
    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "cart_id", nullable = false)
    private Cart cart;

    @ManyToOne(fetch = FetchType.LAZY)
    @JoinColumn(name = "product_id", nullable = false)
    private Product product;

    public Long getId() {
        return id;
    }
    public Product getProduct() {
        return product;
    }
    public void setProduct(CartItemDto product) {
        this.product = new Product();
        this.product.setId(product.productId);
    }
    public Cart getCart() {
        return cart;
    }
    public void setCart(Cart cart) {
        this.cart = cart;
    }
    public int getQuantity() {
```

```
        return quantity;
    }
    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }
    public void setId(Long cartItemId) {
        this.id = cartItemId;
    }
}
```

/*BrandRepository.java*/

```
package com.example.demo.repositories;
import com.example.demo.models.Brand;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.repository.CrudRepository;
import org.springframework.stereotype.Repository;
import java.util.Optional;
@Repository
public interface BrandRepository extends
CrudRepository<Brand,Long>, JpaRepository<Brand,Long> {
    public Optional<Brand> findBrandByName(String name);
    public void  deleteById(Long brandId);
}
```

/*ProductRepository.java*/

```
package com.example.demo.repositories;
import com.example.demo.models.Product;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.CrudRepository;
import org.springframework.data.repository.query.Param;
import org.springframework.stereotype.Repository;
@Repository
```

```
public interface ProductRepository extends
CrudRepository<Product,Long>, JpaRepository<Product, Long> {
    @Query("SELECT p FROM Product  p WHERE (:brand IS NULL OR
p.brand.id = :brand) AND (:category IS NULL OR p.category.id =
:category) AND (:name IS NULL OR p.name LIKE %:name%) AND
(:minPrice IS NULL OR p.price >= :minPrice) AND (:maxPrice IS
NULL OR p.price <= :maxPrice)")
    Page<Product> searchProducts(@Param("brand") Integer brand,
@Param("category") Integer category, @Param("name") String name,
@Param("minPrice") Integer minPrice, @Param("maxPrice") Integer
maxPrice, Pageable pageable);
    public Product findProductById(Long id);
}
```

/*ProductService.java*/

```
package com.example.demo.service.impl;
import com.example.demo.dto.ProductDto;
import com.example.demo.models.Brand;
import com.example.demo.models.Category;
import com.example.demo.models.Product;
import com.example.demo.repositories.ProductRepository;
import com.example.demo.service.BrandService;
import com.example.demo.service.CategoryService;
import com.example.demo.service.ProductService;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;
import java.util.Optional;
@Service
public class ProductServiceImpl implements ProductService {
    private final ProductRepository productRepository;
```

```
    private final BrandService brandService;
    private CategoryService categoryService;
    public ProductServiceImpl(ProductRepository
productRepository, BrandService brandService, CategoryService
categoryService) {
        this.productRepository = productRepository;
        this.brandService = brandService;
        this.categoryService = categoryService;
    }
    @Transactional
    @Override
    public Product getProduct(long productId){
        return productRepository.findById((long)
productId).get();
    }
    @Transactional(readOnly = true)
    public Optional<Product> findById(long productId) {
        return productRepository.findById(productId);
    }
    @Transactional
    @Override
    public Product create(ProductDto dto) {
        Product product = Product.builder()
                .image(dto.image)
                .description(dto.description)
                .name(dto.name)
                .available(dto.available)
                .price(dto.price)
                .build();
        return saveInternal(product, dto.brand, dto.category);
    }
    @Transactional
    @Override
    public void update(long productId, ProductDto product)  {
        Product original = getProduct(productId);
```

```
            original.setName(product.name);
            original.setAvailable(product.available);
            original.setPrice(product.price);
            original.setDescription(product.description);
            original.setImage(product.image);
            saveInternal(original, product.brand, product.category);
        }
        @Transactional
        @Override
        public void delete(long product) {
            productRepository.deleteById(product);
        }
        private Product saveInternal(Product product, String
brandName, String categoryName) {
            Optional<Brand> brand =
brandService.findByName(brandName);;
           if (brand.isEmpty())
           {
                System.out.println("бренда нима, создаём новый");
                Brand newBrand = new Brand(brandName);
                brandService.create(newBrand);
                brand = Optional.of(newBrand);
            }
            product.setBrand(brand);
            Category category = null;
            try {
                category = categoryService.findByName(categoryName);
            }
            catch (Exception e)
            {
                category = new Category(categoryName);
                categoryService.create(category);
            }
            product.setCategory(category);
            productRepository.save(product);
```

```
        return
productRepository.findById(Long.valueOf(product.getId())).get();
    }
    @Transactional(readOnly = true)
    @Override
    public List<Product> getProducts(Integer brand, Integer
category, String name, Integer minPrice, Integer maxPrice,String
sortBy, String sortOrder, Pageable page)  {
        Pageable pageable = PageRequest.of(page.getPageNumber(),
page.getPageSize(),
Sort.by(Sort.Direction.fromString(sortOrder), sortBy));
        Page<Product> productsPage =
productRepository.searchProducts(brand, category, name,
minPrice, maxPrice, pageable);
        return productsPage.getContent();
    }
}
```

/*ProductController.java*/

```
package com.example.demo.controllers;
import com.example.demo.dto.ProductDto;
import com.example.demo.models.Product;
import com.example.demo.responses.ResponseList;
import com.example.demo.responses.ResponseSingle;
import com.example.demo.service.ProductService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.web.bind.annotation.*;
import java.util.List;
import java.util.Optional;
```

```
@RestController
@RequestMapping("/products")
public class ProductController {
    @Autowired
    ProductService productService;
    @PostMapping
    @PreAuthorize("hasRole('ADMIN')")
    public Product createProduct(@RequestBody ProductDto
payload)
    {
        return productService.create(payload);
    }
    @GetMapping
    public ResponseEntity<List<Product>> getProducts(
            @RequestParam(required = false) Integer brand,
            @RequestParam(required = false) Integer category,
            @RequestParam(required = false) String name,
            @RequestParam(required = false) Integer minPrice,
            @RequestParam(required = false) Integer maxPrice,
            @RequestParam(defaultValue = "price") String sortBy,
            @RequestParam(defaultValue = "asc") String
sortOrder,
            @RequestParam(defaultValue = "0") int page,
            @RequestParam(defaultValue = "10") int take) {
        Pageable pageable = PageRequest.of(page, take,
Sort.Direction.fromString(sortOrder), sortBy);
        List<Product> products =
productService.getProducts(brand, category, name, minPrice,
maxPrice, sortBy,sortOrder, pageable);
        return ResponseEntity.ok(products);
    }
    @GetMapping("/{id}")
    public ResponseEntity<ResponseSingle<Product>>
getProduct(@PathVariable("id") long id) throws Exception {
        try
```

```
        {
            Product product = productService.getProduct(id);
            return new
ResponseEntity<ResponseSingle<Product>>(new
ResponseSingle<Product>(product, null), HttpStatus.OK);
        }
        catch (Exception e)
        {
            return new
ResponseEntity<ResponseSingle<Product>>(new
ResponseSingle<Product>(null, "product null"),
HttpStatus.NOT_FOUND);
        }
    }
    @PutMapping("/{id}")
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity updateProduct(@PathVariable("id") long
productId, @RequestBody ProductDto product) {
        try {
            productService.update(productId, product);
            return ResponseEntity.ok().build();
        } catch (Exception e) {
            return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body(e.g
etMessage());
        }
    }
    @DeleteMapping("/{id}")
    @PreAuthorize("hasRole('ADMIN')")
    public ResponseEntity<Void>
deleteProduct(@PathVariable("id") long id) {
        productService.delete(id);
        return ResponseEntity.noContent().build();
    }
}
```

/*CartController.java*/

```
package com.example.demo.controllers;
import com.example.demo.dto.CartItemDto;
import com.example.demo.security.config.JwtService;
import com.example.demo.service.CartItemService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import org.springframework.web.bind.annotation.*;
@RestController
@RequestMapping("/cart")
public class CartController {
    @Autowired
    private CartItemService cartItemService;
    private final JwtService jwtService;
    @Autowired
    public CartController(JwtService jwtService) {
        this.jwtService = jwtService;
    }
    @PostMapping
    public void addProductToCart(@RequestBody CartItemDto item,
@RequestHeader("Authorization") String authHeader) {

        cartItemService.addProductToCart(item,
jwtService.extractClaim(token, claims -> claims.get("id",
Integer.class)), jwtService.extractClaim(token, claims ->
claims.get("cart_id", Integer.class)));
    }
    @DeleteMapping
    public ResponseEntity<Void>
deleteProductFromCart(@RequestBody CartItemDto item) {
```

```
        cartItemService.removeCartItem(item);
        return ResponseEntity.noContent().build();
    }
}
```

/*CartController.java*/

```
package com.example.demo.controllers;
import com.example.demo.dto.UserDto;
import com.example.demo.exception.UserAlreadyExistsException;
import com.example.demo.models.UserAccount;
import com.example.demo.service.UserAccountService;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.crypto.bcrypt.BCrypt;
import org.springframework.web.bind.annotation.*;
import java.util.Optional;
@RestController
@RequestMapping("/users")
public class UserController {
    @Autowired
    UserAccountService userService;
    @PostMapping("/register")
    public ResponseEntity<?> registerUser(@RequestBody UserDto
userRegistrationRequest) {
        try {
            userService.registerUser(new
UserAccount(userRegistrationRequest.firstName,userRegistrationRe
quest.email, userRegistrationRequest.lastName,
userRegistrationRequest.password));
            return ResponseEntity.ok().build();
        } catch (UserAlreadyExistsException e) {
            return
```

```
ResponseEntity.status(HttpStatus.CONFLICT).body(e.getMessage());
        }
    }
    @PostMapping("/login")
    public ResponseEntity<?> loginUser(@RequestBody UserDto
userLoginRequest, HttpServletRequest request) {
        String email = userLoginRequest.email;
        String password = userLoginRequest.password;
        Optional<UserAccount> user =
userService.findByEmail(email);
        if (user == null || !BCrypt.checkpw(password,
user.get().getPassword())) {
            return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).build();
        }
        HttpSession session = request.getSession(true);
        session.setAttribute("user", user);
        session.setMaxInactiveInterval(30 * 60);
        return ResponseEntity.ok().build();
    }
}
```

ДОДАТОК Д – ОПИС ВЕБСЕРВІСУ ДЛЯ МАГАЗИНУ З ПРОДАЖУ МУЗИЧНИХ ІНСТРУМЕНТІВ «MUSICBRAND»

1. Загальні відомості

1.1 Позначення і найменування програми

Найменування: Вебсервіс для магазину з продажу музичних інструментів «MusicBrand».

Позначення: «MusicBrand».

1.2 Програмне забезпечення, необхідне для функціонування програми

Програмне забезпечення розділено на серверну частину та клієнтську. Для повноцінної роботи програмного продукту нижче вказано технічні характеристики як для серверу так і клієнтської частини.

1.2.1 Серверна частина

Сервер повинен відповідати наступному складу технічних засобів з мінімальними параметрами:

- Операційна система: Linux або Windows Server
- Процесор: 64-бітний двох або більше ядерний процесор з тактовою частотою не менше 2 ГГц
- Оперативна пам'ять: не менше 4 ГБ
- Диск: не менше 100 ГБ

1.2.2 Клієнтська частина

Для роботи клієнтської частини необхідні наявність браузера з підтримкою HTML-5/CSS-3 та доступ до виходу у мережу Internet.

1.3 Мови програмування

Для розробки вебсервісу для магазину з продажу музичних інструментів «MusicBrand» було використано фреймворк Java Spring, реалізований на мові програмування Java, та СУБД MySql.

Інтегроване середовище розробки що було використано під час розробки – IntelliJ IDEA 2021.

2. Функціональне призначення

2.1 Класи розв’язуваних завдань та призначення програми

Вебсервіс для магазину з продажу музичних інструментів «MusicBrand» повинно забезпечувати виконання нижче перерахованих функцій:

- Реєстрація;
- Аутентифікація;
- Додавання товарів до системи;
- Видалення товарів з системи;
- Редагування інформації про товари в системі;
- Додавання товару до кошика;
- Видалення товару з кошика;
- Оформлення замовлення.

2.2 Функціональні обмеження

Одним з найголовніших функціональних обмежень розробленого вебсервісу для магазину з продажу музичних інструментів «MusicBrand» є необхідність підключення до мережі інтернету.

3 Опис логічної структури

3.1 Використовуванні методи

При розробці вебсервісу для магазину з продажу музичних інструментів «MusicBrand» та проведення його аналізу було використано об’єктно-орієнтовану методологію та об’єктно-орієнтовану мову програмування Java.

3.2 Алгоритм програми

Програмне забезпечення веб-сайту реалізовано з використанням Java Spring - фреймворку для розробки веб-додатків з архітектурою Model-View-Controller (MVC). Веб-сайт розділено на клієнтську та серверну частини, причому клієнтом виступає веб-браузер, а сервером - комп'ютер або група комп'ютерів, на яких розташовані код програми та база даних.

Нижче наведено опис алгоритму взаємодії між клієнтом і сервером, а також між сервером і базою даних у контексті Java Spring:

1) Користувач відкриває веб-сайт у своєму веб-браузері.
2) Браузер надсилає запит на сервер.
3) Веб-сервер отримує цей запит і передає його до контролера Spring MVC.
4) Контролер обробляє запит, виконуючи відповідні дії та взаємодіючи з необхідними сервісами.
5) Якщо потрібно, контролер взаємодіє з базою даних, використовуючи Spring Data JPA, відправляючи SQL-запити та отримуючи результати операцій.
6) Програмне забезпечення формує відповідь на запит, створюючи модель даних.
7) Веб-сервер повертає відповідь до браузера через HTTP-протокол.
8) Браузер отримує відповідь та відображає її користувачу.

3.3 Використовувана методологія

Вебсервіс для магазину з продажу музичних інструментів «MusicBrand» розроблено на мові програмування Java, яка є об’єктно-

орієнтованою. В основу розробки вебсервісу покладено об’єктно-орієнтовану методологію.

4. Використовувані технічні засоби

Для роботи вебсервісу необхідний наступний склад технічних засобів з мінімальними параметрами:

- Процесор: 64-бітний двох або більше ядерний процесор з тактовою частотою не менше 2 ГГц
- Оперативна пам'ять: не менше 4 ГБ
- Жорсткий диск: не менше 100 ГБ

Швидкість роботи вебсервісу для магазину з продажу музичних інструментів «MusicBrand» на конкретному комп’ютері залежить також від характеристик окремих його комплектуючих(процесора, оперативної пам’яті та ін.).

5. Виклик і завантаження

Після розгортання вебсервісу на хостингу веб-сервера відбувається запуск програмного забезпечення. Його активація відбувається шляхом переходу на посилання веб-сайту.

Вхідною точкою програмного забезпечення є функція main().

6. Вхідні і вихідні дані

Введення інформації здійснюється за допомогою клавіатури. Вхідні дані повинні вводитися у спеціальні поля на сайті.

Вихідні дані мають відображатися на сайті у відведених для них полях.