Державний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Клієнт-серверний додаток «Muzline» з реалізації студійного обладнання»

Студентки 4 курсу, 7 групи,
спеціальності 121 «Інженерія програмного забезпечення»
освітньої програми «Інженерія програмного забезпечення» _____ Головченко Віолетти Дмитрівни
підпис студента

Науковий керівник
старший викладач кафедри інженерії програмного забезпечення та кібербезпеки _____ Гнатченко Дмитро Дмитрович
підпис керівника

Гарант освітньої програми
кандидат технічних наук,
доцент кафедри інженерії програмного забезпечення та кібербезпеки _____ Рзаєва Світлана Леонідівна
підпис гаранта

КИЇВ – 2023

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
«14» листопада 2022 р.

Завдання
на випускний кваліфікаційний проєкт студентові

Головченко Віолетті Дмитрівні
(прізвище, ім'я, по батькові)

1. Тема випускного кваліфікаційного проєкту «Клієнт-серверний додаток «Muzline» з реалізації студійного обладнання»

Затверджена наказом ректора від «6» грудня 2022 р. № 3288

2. Строк здачі студентом закінченого проєкту 5 червня 2023

3. Цільова установка та вихідні дані до проєкту

Мета проєкту є розробка зручного та ефективного клієнт-серверного додатку, що забезпечить користувачам простий та зрозумілий інтерфейс для керування студійним обладнанням з будь-якого місця з доступом до мережі Інтернет.

Об'єкт дослідження є музична індустрія та робота зі студійним обладнанням.

Предмет дослідження є проектування та розробка функціональності додатку, який дозволить користувачам зручно та ефективно керувати роботою студійного обладнання через мережу Інтернет.

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТА ДОСЛІДЖЕННЯ І ОПИС ЙОГО ОСНОВНИХ ПАРАМЕТРІВ ТА ХАРАКТЕРИСТИК

1.1. Аналіз поточного стану реалізації студійного обладнання

1.2. Розробка та впровадження клієнт-серверного додатку з реалізації студійного обладнання

1.3. Технічне завдання

1.4. Висновок до розділу 1

РОЗДІЛ 2. ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ З РЕЛІЗАЦІЇ СТУДІЙНОГО ОБЛАДНАННЯ

2.1. Чинники, що визначають хід і результати роботи клієнт-серверного додатку

2.2. Інструменти розробки клієнт-серверного додатку з реалізації студійного обладнання

2.3. Висновок до розділу 2

РОЗДІЛ 3. РОЗРОБКА ТА ВПРОВАДЖЕННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ З РЕЛІЗАЦІЇ СТУДІЙНОГО ОБЛАДНАННЯ

3.1. Функціональні характеристики розробленого клієнт-серверного додатку «Muzline» з реалізації студійного обладнання

3.2. Особливості впровадження клієнт-серверного додатку «Muzline» з реалізації студійного обладнання

3.3. Висновок до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проєкту

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2022*	*21.09.2022*
2.	*Розробка та затвердження завдання на проєкт*	*14.11.2022*	*14.11.2022*
3.	*Вступ та перелік літературних джерел*	*23.12.2022*	*23.12.2022*
4.	*Розділ 1. Аналіз предмета дослідження і опис його основних параметрів та характеристик*	*27.01.2023*	*27.01.2023*
5.	*Розділ 2. Проєктування клієнт-серверного додатку з релізації студійного обладнання*	*03.03.2023*	*03.03.2023*
6.	*Розділ 3. Розробка та впровадження клієнт-серверного додатку з релізації студійного обладнання*	*14.04.2023*	*14.04.2023*
7.	*Висновки*	*28.04.2023*	*28.04.2023*
8.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*17.05.2023*	*17.05.2023*
9.	*Підготовка автореферату та презентації доповіді*	*26.05.2023*	*26.05.2023*
10.	*Попередній захист випускного кваліфікаційного проєкту*	*29.05.2023 – 02.06.2023*	
11.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*05.06.2023*	*05.06.2023*
12.	*Здача прошитого випускного кваліфікаційного проєкту на кафедру*	*05.06.2023*	*05.06.2023*
13.	*Публічний захист випускного кваліфікаційного проєкту*		

7. Дата видачі завдання «14» листопада 2022 р.

8. Науковий керівник випускного кваліфікаційного проєкту Гнатченко Д.Д.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Рзаєва С.Л.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Головченко В.Д.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

Науковий керівник випускного кваліфікаційного проєкту

_____

(підпис, дата)

Відмітка про попередній захист _____

(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента _____Головченко В.Д.____

(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____ Рзаєва С.Л.

(прізвище, ініціали, підпис)

Завідувач кафедри _____ Криворучко О. В.

(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Відповідно до мети дослідження робота присвячена розробці та реалізації клієнт-серверного додатку для студійного обладнання в музичній індустрії. Метою роботи було створення зручного та ефективного інструменту для музикантів та звукорежисерів для керування студійним обладнанням та обміну аудіо-даними.

В результаті порівняльного аналізу аналогічних рішень визначено основні функціональні вимоги до додатку.

Додаток "Muzline" було реалізовано з використанням сучасних технологій та мов програмування, що дозволило досягти високої продуктивності та надійності системи. Проведено серію тестувань, які підтвердили функціональність та якість додатку.

Готовий програмний комплекс Muzline було успішно протестовано. Отриманий клієнт-серверний додаток "Muzline" представляє собою ефективне рішення для музикантів та звукорежисерів, що дозволяє зручно та швидко керувати студійним обладнанням та обмінюватися аудіо-даними. Результати роботи можуть бути використані в музичній індустрії для поліпшення процесу створення та редагування музики.

Ключові слова: клієнт-серверний додаток, Muzline, студійне обладнання, музична індустрія, функціональні вимоги, архітектура, користувачі, інтерфейс, аудіо-дані, серверна частина, безпека даних, продуктивність, надійність, тестування, звукорежисери, музиканти.

ABSTRACT

According to the purpose of the research, the work is devoted to the development and implementation of a client-server application for studio equipment in the music industry. The goal of the work was to create a convenient and effective tool for musicians and sound engineers to control studio equipment and exchange audio data.

As a result of a comparative analysis of similar solutions, the main functional requirements for the application were determined.

The "Muzline" application was implemented using modern technologies and programming languages, which made it possible to achieve high performance and reliability of the system. A series of tests were conducted that confirmed the functionality and quality of the application.

The finished Muzline software package was successfully tested. The resulting client-server application "Muzline" is an effective solution for musicians and sound engineers, which allows you to conveniently and quickly manage studio equipment and exchange audio data. The results of the work can be used in the music industry to improve the process of creating and editing music.

Keywords: client-server application, Muzline, studio equipment, music industry, functional requirements, architecture, users, interface, audio data, backend, data security, performance, reliability, testing, sound engineers, musicians.

ЗМІСТ

ВСТУП **3**

РОЗДІЛ 1 АНАЛІЗ ПРЕДМЕТА ДОСЛІДЖЕННЯ І ОПИС ЙОГО ОСНОВНИХ ПАРАМЕТРІВ ТА ХАРАКТЕРИСТИК **5**

1.1. Аналіз поточного стану реалізації студійного обладнання 5

1.2. Розробка та впровадження клієнт-серверного додатку з реалізації студійного обладнання 10

1.3. Технічне завдання 17

1.4. Висновки до розділу 1 26

РОЗДІЛ 2 ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ З РЕЛІЗАЦІЇ СТУДІЙНОГО ОБЛАДНАННЯ **28**

2.1. Чинники, що визначають хід і результати роботи клієнт-серверного додатку 28

2.2. Інструменти розробки клієнт-серверного додатку з реалізації студійного обладнання 37

2.3. Висновки до розділу 2 44

РОЗДІЛ 3 РОЗРОБКА ТА ВПРОВАДЖЕННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ З РЕЛІЗАЦІЇ СТУДІЙНОГО ОБЛАДНАННЯ **46**

3.1. Функціональні характеристики розробленого клієнт-серверного додатку «Muzline» з реалізації студійного обладнання 46

3.2. Особливості впровадження клієнт-серверного додатку «Muzline» з реалізації студійного обладнання 56

3.3. Висновок до розділу 3 57

ВИСНОВКИ ТА ПРОПОЗИЦІЇ **59**

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ **61**

ДОДАТКИ……

					ДТЕУ 121 07-09.БР		
Зм.	*Аркуш*	*№ докум.*	*Підпис*	*Дата*			
Зав. каф.		Криворучко О.В.		23.12.22	Клієнт-серверний додаток «Muzline» з реалізації студійного обладнання	*Стадія*	*Аркуш*	*Аркушів*
Керівник		Гнатченко Д.Д.		23.12.22		*З*	*2*	*62*
Гарант		Рзаєва С.Л.		23.12.22		*Факультет інформаційних технологій 4 курс, 7 група*		
Розробив		Головченко В.Д.		23.12.22	*Зміст*			

ВСТУП

Клієнт-серверні додатки є важливою складовою сучасних інформаційних технологій, що забезпечують зручний доступ до різноманітних сервісів та даних через мережу Інтернет. У контексті музичної індустрії, студійне обладнання відіграє ключову роль у процесі створення якісної та професійної музики.

Об’єктом дослідження є музична індустрія та робота зі студійним обладнанням. Предметом дослідження є проектування та розробка функціональності додатку, що дозволить користувачам зручно та ефективно керувати роботою студійного обладнання через мережу Інтернет.

Основною метою проекту є розробка зручного та ефективного клієнт-серверного додатку, що забезпечить користувачам простий та зрозумілий інтерфейс для керування студійним обладнанням з будь-якого місця з доступом до мережі Інтернет.

Основними завданнями проекту є: аналіз вимог до додатку, проектування архітектури системи, розробка функціональності додатку, тестування та підтримка додатку.

Практична значущість проекту полягає в покращенні ефективності роботи зі студійним обладнанням, забезпеченні доступу до обладнання з будь-якої точки світу та зручному керуванні його функціоналом.

Тема проекту є досить актуальною та добре вивченою в науковому та практичному аспекті. Існують різноманітні клієнт-серверні додатки для управління різними пристроями та обладнання, але зазначений додаток має специфічну спрямованість на управління студійним обладнанням, що робить його унікальним у своєму роді. Додаток може бути корисним як професійним

музикантам та звукорежисерам, так і початківцям у цій галузі, які хочуть покращити якість своєї музики та зробити процес створення більш ефективним та зручним.

Дослідження та розробка клієнт-серверного додатку "Muzline" має великий потенціал для практичного використання, а також для подальшого розвитку і вдосконалення у майбутньому. Задача розробки додатку передбачає вирішення складних технічних завдань, що знайомить розробника з новими технологіями та методами, що можуть знадобитися у майбутньому.

Отже, розробка клієнт-серверного додатку "Muzline" для управління студійним обладнанням має великий потенціал для покращення якості та ефективності створення музики, а також може стати першим кроком на шляху до подальшого розвитку та вдосконалення інформаційних технологій в галузі музики та звукозапису.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТА ДОСЛІДЖЕННЯ І ОПИС ЙОГО ОСНОВНИХ ПАРАМЕТРІВ ТА ХАРАКТЕРИСТИК

1.1. Аналіз поточного стану реалізації студійного обладнання

Так як мова йде про студійне обладнання, то поточний стан реалізації цього обладнання залежить від багатьох факторів, таких як ринкові тенденції, технічні можливості, попит на ринку та конкуренція. Однак, оглядаючи загальну картину, можна зробити декілька спостережень [5]:

1. Розвиток цифрової технології. Сучасні студійні обладнання стають все більш компактними та зручними у використанні завдяки використанню цифрових технологій. Це забезпечує збільшення продуктивності та ефективності в роботі.

Так, розвиток цифрової технології суттєво впливає на розвиток студійного обладнання. Основною перевагою цифрової технології є зменшення розміру та ваги обладнання.

Однією з головних переваг цифрової технології є можливість зберігання та обробки великих обсягів даних на комп'ютерах та інших цифрових пристроях. Це забезпечує значну економію простору та зберігання інформації на протязі довгого часу. Крім того, цифрова технологія забезпечує більшу точність та якість обробки даних, що є важливим для професіоналів у галузі зйомки та обробки відео та аудіоматеріалів.

З використанням цифрової технології, сучасні студійні обладнання стають все більш компактними та зручними у використанні. Зокрема, цифрові

камери та мікрофони мають більш невеликий розмір та вагу, що дозволяє професіоналам використовувати їх у різних умовах зйомки, включаючи зйомки в руху та важкодоступних місцях.

Крім того, цифрова технологія дозволяє забезпечити більшу автоматизацію процесів обробки та монтажу відео та аудіо, що дозволяє значно зменшити час та зусилля, необхідні для створення професійних відео та аудіоматеріалів.

Отже, розвиток цифрової технології суттєво впливає на розвиток студійного обладнання, забезпечуючи більшу продуктивність, ефективність

2. Зростання популярності онлайн-стрімінгу. За останні роки онлайн-стрімінг став дуже популярним, що призвело до збільшення попиту на студійне обладнання для відеозйомки та живого стрімінгу. Це пов'язано з тим, що відео є однією з основних форматів контенту в онлайн-середовищі, а онлайн-стрімінг дозволяє відтворювати цей контент в режимі реального часу.

За останні роки онлайн-стрімінг став дуже популярним, зокрема завдяки зростанню швидкості Інтернет-з'єднання та поширенню мобільних пристроїв з підтримкою стрімінгу.

У зв'язку з цим, багато виробників студійного обладнання випускають спеціальні моделі для стрімінгу та відеозйомки, які мають додаткові функції для підтримки цих процесів. Наприклад, це можуть бути спеціальні камери з підтримкою відеострімінгу, мікрофони зі зменшеною чутливістю до шумів, а також інші елементи обладнання, які забезпечують високу якість зйомки та звуку для онлайн-стрімінгу.

Також варто зазначити, що студійне обладнання стає все більш доступним для широкої аудиторії, що дозволяє користувачам самостійно створювати якісний відеоконтент для онлайн-стрімінгу. Це може бути важливим для відеоблогерів, викладачів, музикантів та інших категорій

користувачів, які хочуть поділитися своїми ідеями та творчістю в онлайн режимі.

3. Конкуренція на ринку. Ринок студійного обладнання є досить конкурентним, з багатьма виробниками, які пропонують різні продукти та послуги для відеозйомки, монтажу та постпродакшну, що призводить до зниження цін на деякі продукти та зростання якості інших. Ця конкуренція має позитивний вплив на розвиток технологій та підвищення якості обладнання, а також зниження цін на деякі продукти.

З одного боку, конкуренція спонукає виробників до того, щоб постійно вдосконалювати свої продукти, забезпечувати нові функції та можливості, що забезпечує зростання якості студійного обладнання. Наприклад, виробники постійно розробляють нові камери, які мають більшу роздільну здатність, вищу чутливість та динамічний діапазон, що дозволяє знімати високоякісне відео.

З іншого боку, конкуренція також призводить до зниження цін на деякі продукти, особливо на ті, що вже давно присутні на ринку. Це дозволяє більшій кількості людей отримувати доступ до необхідного обладнання та інструментів для створення відео контенту.

Проте, в той же час, конкуренція може бути також негативним фактором, оскільки вона змушує виробників знижувати ціни на свої продукти, що може призвести до зниження якості виробів, щоб зменшити витрати на виробництво. Також конкуренція може викликати складнощі для менших виробників, які не мають достатнього обсягу продажів та ресурсів, щоб бути конкурентоспроможними на ринку.

4. Розширення функціональності. Сучасні студійні обладнання мають все більше функцій, які забезпечують більш глибоку обробку та редагування аудіо- та відеоматеріалів.

Розширення функціональності студійного обладнання стало можливим завдяки розвитку технологій, таких як штучний інтелект, машинне навчання та обробка даних. Нові можливості дозволяють не тільки виконувати базові завдання, такі як запис та збереження аудіо- та відеоматеріалів, але й додавати до них багато інших функцій.

Наприклад, сучасні програмні продукти для редагування відео дозволяють додавати різноманітні спецефекти, включаючи 3D-графіку та анімацію. Крім того, вони мають вбудовані інструменти для корекції кольору та освітлення, що дозволяє покращити якість відео та забезпечити його більш професійний вигляд.

У сфері аудіозапису та обробки сучасні студійні обладнання також мають багато функцій, таких як фільтри шуму та ефекти, що дозволяють додавати до звукових доріжок різноманітні звукові елементи та покращувати якість звуку.

Розширення функціональності студійного обладнання дозволяє професіоналам з різних галузей використовувати його для виконання різних завдань, що забезпечує більшу гнучкість та ефективність в роботі.

5. Збільшення розширення та якості відео. За останні кілька років зросла важливість якості та розширення відео, що призвело до збільшення попиту на студійне обладнання, яке забезпечує зйомку та обробку відео в високій якості.

Збільшення розширення та якості відео є однією з найбільш важливих тенденцій в сучасному студійному обладнанні. Зростання популярності відео-контенту на різних платформах, таких як YouTube, Instagram, TikTok тощо, призвело до зростання вимог щодо якості та розширення відео.

Сучасні студійні камери мають можливість зйомки в 4K та 8K розширенні, що забезпечує високу якість та деталізацію зображення. Також з'явилися спеціальні камери для зйомки відео з високою швидкістю

збільшення кадрів на секунду, що дає можливість отримувати більш якісні відео з кращою динамікою [7].

Для обробки відео високої якості створюються спеціальні програмні засоби та обладнання. Сучасні програмні пакети дозволяють не тільки редагувати та монтажувати відео, але й забезпечують можливість корекції кольору, зменшення шуму та виправлення недоліків зображення. Для роботи з великими файлами відео створюються спеціальні обчислювальні системи, які забезпечують швидке оброблення великого обсягу відеоданих.

Отже, загалом, студійне обладнання розвивається досить швидко і забезпечує все більше можливостей для професійної роботи в галузі зйомки та обробки аудіо- та відеоматеріалів. Це, з одного боку, забезпечує професіоналам більші можливості для творчої роботи та розвитку, з іншого - створює певний виклик у забезпеченні сумісності між різними пристроями та програмним забезпеченням.

Однак, на сьогоднішній день, не всі професіонали мають доступ до новітнього студійного обладнання через високі вартість та складність використання. Крім того, низька якість звуку або відео може бути результатом недостатньої підготовки та знань професіоналів у використанні обладнання.

Для того, щоб розвивати сучасну галузь студійного обладнання, необхідно надавати належну увагу освіті та навчанню професіоналів, створювати доступніші та економічніші варіанти обладнання, а також забезпечувати більшу сумісність між різними програмними та апаратними засобами.

У цілому, розвиток студійного обладнання залежить від розвитку технологій та відповідного ринкового попиту. Якщо розвиток продовжиться на поточному рівні, можна очікувати появу нових технологій та виробів, які забезпечать більшу ефективність та продуктивність в галузі студійної роботи.

1.2. Розробка та впровадження клієнт-серверного додатку з реалізації студійного обладнання

Розробка та впровадження клієнт-серверного додатку для студійного обладнання потребує наявності досвіду у програмуванні та розумінні принципів роботи мережевих протоколів.

Основні етапи розробки та впровадження такого додатку наведені на рис. 1.1 [1].

Розглянемо дані етапи більш детально:

1. Визначення вимог до додатку: у вас має бути чітко сформульоване, що вимагається від додатку, який має бути розроблений.

Визначення вимог до додатку є одним з найважливіших етапів в процесі розробки. Цей етап визначає, які функції має надавати додаток, яка повинна бути його продуктивність, який має бути рівень безпеки, який має бути інтерфейс та інші вимоги, які повинні бути виконані.

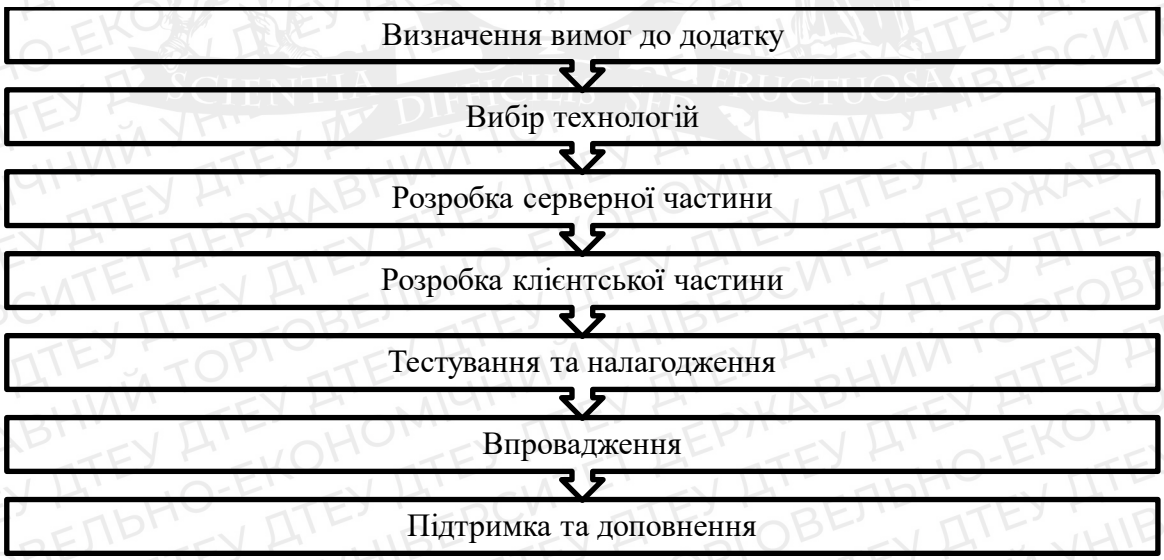

Рис. 1.1. Основні етапи розробки та впровадження клієнт-серверного додатку з реалізації студійного обладнання

Джерело: побудовано автором

Для визначення вимог до додатку можна проводити діалог з потенційними користувачами або експертами в області студійного обладнання. Важливо розібратися, які є різні функції студійного обладнання, і які з них є необхідними для користувачів. Наприклад, це можуть бути функції запису та відтворення аудіо, регулювання гучності, настройка рівня частот, ефектів та інших параметрів звукового сигналу, робота з різними форматами файлів тощо.

Також важливо врахувати, на яких пристроях має працювати додаток, яка повинна бути швидкість його роботи та який рівень безпеки має бути забезпечений. Наприклад, якщо додаток має працювати на мобільних пристроях, то він повинен бути оптимізований для роботи з обмеженими ресурсами.

В результаті визначення вимог до додатку повинна бути сформована специфікація вимог, яка містить детальний опис того, які функції та вимоги повинні бути виконані в додатку. Цей документ служить основою для наступних етапів розробки та допомагає забезпечити виконання всіх необхідних вимог.

2. Вибір технологій: ви повинні визначити, які технології ви будете використовувати для розробки додатку. Наприклад, мови програмування, бази даних, мережеві протоколи тощо.

Вибір технологій є одним з найважливіших етапів в процесі розробки клієнт-серверного додатку для студійного обладнання. Правильний вибір технологій дозволяє забезпечити ефективну та швидку розробку, покращити функціональність та надійність додатку, а також забезпечити легкість його підтримки та масштабування.

У випадку клієнт-серверного додатку, потрібно вибрати технології для розробки як клієнтської, так і серверної частин. Для розробки клієнтської частини можна використовувати мови програмування, такі як Java, C++, C#,

Python, JavaScript тощо. Важливо враховувати особливості студійного обладнання та потреби користувачів у додатку, щоб визначити, яка мова програмування буде найбільш ефективною.

Для розробки серверної частини можна використовувати бази даних, такі як MySQL, PostgreSQL, MongoDB, а також веб-сервери, такі як Apache, Nginx. Важливо врахувати особливості додатку та його потреби у масштабуванні та надійності, щоб визначити, які технології будуть найбільш підходящими.

Для забезпечення комунікації між клієнтською та серверною частинами можна використовувати різні мережеві протоколи, такі як HTTP, WebSocket, TCP/IP, UDP тощо. Важливо враховувати потреби додатку у швидкості та безпеці передачі даних, щоб визначити, який мережевий протокол буде найбільш ефективним.

3. Розробка серверної частини: серверна частина додатку має забезпечувати обробку запитів від клієнтів, взаємодію з базою даних та іншими додатками. Ви можете розробити серверну частину додатку з використанням фреймворків, бібліотек та інших інструментів.

Розробка серверної частини додатку передбачає створення програмного забезпечення, яке буде працювати на сервері та забезпечувати обробку запитів від клієнтів. Для розробки серверної частини можна використовувати різні технології, такі як фреймворки, бібліотеки, мови програмування тощо.

Один із популярних фреймворків для розробки серверної частини - це Django. Django надає зручний інтерфейс для роботи з базами даних, автоматичну генерацію адміністративних панелей, підтримку міжнародизації та інші корисні можливості.

Іншим популярним фреймворком є Flask, який простіший за Django, але дозволяє розробляти додатки швидше та більш гнучко. Flask також має

підтримку різних баз даних, тестування коду, захист від атак, підтримку мережевих протоколів і т.д.

Також можна використовувати мови програмування, які мають підтримку для створення серверної частини додатків, такі як Java, C#, Ruby, Python, PHP, Node.js і т.д. Ці мови програмування мають свої фреймворки та бібліотеки, які можна використовувати для розробки додатку.

Крім того, для розробки серверної частини можна використовувати різноманітні бази даних, наприклад MySQL, PostgreSQL, MongoDB та інші. Вибір бази даних залежить від вимог до додатку, обсягу даних, які потрібно зберігати, та інших факторів.

4. Розробка клієнтської частини: клієнтська частина додатку має забезпечувати взаємодію користувача з серверною частиною. Наприклад, користувачі мають мати можливість надсилати запити до сервера, отримувати відповіді, переглядати інформацію тощо. Клієнтська частина може бути розроблена з використанням фреймворків, бібліотек та інших інструментів.

Розробка клієнтської частини додатку є важливим етапом у створенні клієнт-серверної системи. Для забезпечення взаємодії користувачів з серверною частиною можуть бути використані різні технології, включаючи веб-технології, десктопні застосунки, мобільні додатки та інші.

Зазвичай розробка клієнтської частини додатку включає в себе розробку інтерфейсу користувача, який дозволяє взаємодіяти з сервером за допомогою графічного інтерфейсу. Для розробки інтерфейсу можна використовувати різні технології, включаючи HTML, CSS, JavaScript, а також фреймворки для розробки інтерфейсу, такі як React, Angular, Vue та інші.

Крім того, клієнтська частина додатку може включати в себе логіку обробки даних на стороні клієнта, таку як перевірка даних, перед відправкою

на сервер, або обробка даних, отриманих від сервера перед їх відображенням користувачу.

У процесі розробки клієнтської частини додатку також необхідно забезпечити зручний інтерфейс для взаємодії користувачів з серверною частиною, включаючи надання можливості вводити запити до сервера, отримувати та відображати результати запитів, а також надання можливості користувачам працювати з інформацією, збереженою на сервері.

Таким чином, розробка клієнтської частини додатку включає в себе багато елементів, від розробки інтерфейсу користувача до логіки обробки даних на стороні клієнта.

5. Тестування та налагодження. Ви можете використовувати різні методи тестування, такі як модульне тестування, інтеграційне тестування, функціональне тестування тощо. Також варто забезпечити налагодження додатку, щоб виправити будь-які помилки, які були виявлені під час тестування.

Після розробки додатку важливо провести його тестування та налагодження, щоб переконатись у його працездатності та відповідності вимогам.

Модульне тестування дозволяє перевірити окремі модулі додатку на відповідність вимогам та виявити можливі помилки. Інтеграційне тестування полягає у перевірці взаємодії між різними модулями та підсистемами додатку. Функціональне тестування дозволяє перевірити функціональність додатку згідно з вимогами.

Також важливо виконати налагодження додатку. Під час налагодження необхідно виявляти та виправляти можливі помилки та недоліки, щоб додаток працював бездоганно [9].

Для тестування та налагодження додатку можна використовувати різні інструменти, наприклад, автоматизовані тестові фреймворки, дебагери та

інші програмні засоби. Також можна проводити тестування вручну, наприклад, за допомогою тестових сценаріїв.

Після завершення тестування та налагодження необхідно перевірити, чи відповідає додаток вимогам та чи готовий він до впровадження.

6. Впровадження: коли додаток пройшов тестування та налагодження, він може бути впроваджений. Це може включати установку серверного та клієнтського ПЗ на відповідні комп'ютери, налаштування мережевих параметрів та інші необхідні дії.

При впровадженні клієнт-серверного додатку з реалізацією студійного обладнання, додаток повинен бути встановлений на всі комп'ютери, які беруть участь у взаємодії з системою.

Перш за все, необхідно підготувати всі необхідні сертифікати та ключі безпеки, щоб забезпечити захист передачі даних між сервером та клієнтами. Потім необхідно встановити серверну частину додатку на відповідні сервери. Відповідно до вимог до додатку, мають бути налаштовані параметри сервера, такі як рівень безпеки, швидкість обробки даних, забезпечення резервного копіювання та відновлення даних тощо.

Далі, необхідно встановити клієнтську частину додатку на кожен комп'ютер, з якого будуть здійснюватись запити до сервера. Для зручності користувачів можна розмістити відповідні ярлики на робочому столі або в меню програм. Важливо відповідно налаштувати клієнтську частину додатку, щоб забезпечити правильне з'єднання з сервером, автентифікацію користувачів та надання доступу до необхідних функцій додатку.

Крім того, перед впровадженням додатку слід провести інструктаж користувачів щодо правильного використання програмного забезпечення та дотримання правил безпеки в роботі з системою. Також необхідно забезпечити можливість швидкого вирішення технічних проблем, що

виникають під час використання додатку, за допомогою надання контактних даних служби підтримки.

7. Підтримка та доповнення: після впровадження додатку можуть з'являтися нові вимоги або проблеми, які потрібно буде вирішувати. Важливо забезпечити підтримку та доповнення додатку, яке може включати усунення помилок, вдосконалення функціоналу, адаптацію до нових вимог тощо.

Підтримка та доповнення додатку - це процес, який має за мету забезпечити продовження роботи додатку відповідно до вимог та потреб користувачів. Цей процес може включати різні дії, зокрема [10]:

– Виявлення та усунення помилок: Коли користувачі повідомляють про помилки в додатку, важливо їх виявити та усунути. Це може включати аналіз логів, відновлення стану системи до попередньої версії, зміну коду, щоб уникнути подібних помилок у майбутньому та інші дії.

– Доповнення функціоналу: З часом можуть з'являтися нові вимоги та потреби користувачів, які не передбачалися на етапі розробки. Важливо аналізувати ці потреби та додавати новий функціонал у вигляді оновлень або патчів.

– Адаптація до змін: Зміни в інфраструктурі, зміна зовнішніх сервісів та інші зміни можуть впливати на роботу додатку. Підтримка та доповнення додатку повинна включати адаптацію до цих змін.

– Підтримка безпеки: Із зміною умов роботи додатку можуть з'являтися нові загрози безпеці. Підтримка та доповнення додатку повинні включати заходи забезпечення безпеки, такі як оновлення бібліотек, усунення вразливостей тощо.

– Підтримка різних платформ: Якщо додаток працює на різних платформах, наприклад, на різних операційних системах або пристроях, важливо забезпечити його підтримку на цих платформах та вирішувати проблеми, які виникають на кожній з них.

– Забезпечення безпеки: при розробці додатку необхідно забезпечити високий рівень безпеки для захисту конфіденційної інформації користувачів та даних, що обробляються додатком. Для цього можуть використовуватися різні методи та інструменти, такі як шифрування даних, захист від атак типу SQL injection, розробка безпечного протоколу автентифікації тощо.

– Оптимізація продуктивності: при використанні додатком великої кількості користувачів можуть виникнути проблеми з продуктивністю, яка може бути пов'язана з обробкою великих обсягів даних, розширенням функціоналу тощо. Для підвищення продуктивності додатку можуть використовуватися різні методи та інструменти, такі як кешування даних, оптимізація запитів до бази даних, розробка складних алгоритмів та технік паралельного обчислення.

– Моніторинг та аналітика: для підтримки та вдосконалення додатку важливо забезпечити його моніторинг та аналітику. Це може включати збір статистичних даних про використання додатку, моніторинг продуктивності та роботи серверів, аналіз поведінки користувачів та їх вимог до функціоналу тощо. Ці дані можуть використовуватися для вдосконалення додатку та підвищення задоволення користувачів його використанням.

Розробка та впровадження клієнт-серверного додатку для студійного обладнання може бути складним та часовим процесом. Проте, якщо ви маєте необхідний досвід та знання, а також використовуєте ефективні інструменти, то ви зможете розробити потрібний додаток, який відповідатиме всім вимогам та задовольнить потреби користувачів.

1.3. Технічне завдання

1. Загальні відомості

1.1. Найменування системи

1.1.1. Повне найменування системи - Muzline

1.1.2. Скорочене найменування системи - Muzline

1.2. Планові терміни початку та закінчення робіт

Початок робіт - 01.01.2023

Закінчення робіт - 01.05.2023

1.3. Порядок оформлення і пред'явлення результатів робіт

Результати робіт будуть оформлені у вигляді програмного коду, документації з поясненням алгоритмів, принципів роботи та інструкцій щодо встановлення, налаштування та користування додатком. Результати будуть пред'явлені замовнику в електронному вигляді, а також у паперовому вигляді за запитом замовника.

1.4. Головний бенефіціар та потенційні користувачі системи

Головним бенефіціаром системи є студія, що володіє обладнанням та замовник, який замовив розробку додатку. Потенційні користувачі системи - працівники студії, що мають права доступу до додатку та можуть змінювати налаштування обладнання, зберігати налаштування та отримувати статистику роботи обладнання.

2. Мета та призначення створення системи

2.1. Призначення системи

Клієнт-серверний додаток призначений для управління студійним обладнанням, зберігання налаштувань та отримання статистики роботи обладнання.

2.2. Мета створення системи

Метою створення системи є поліпшення ефективності та продуктивності роботи студії, забезпечення максимальної точності та якості виконання роботи обладнання, зменшення часу на встановлення та налаштування обладнання, зберігання налаштувань та отримання статистики роботи обладнання. Додаток повинен бути зручним та простим у

користуванні, містити всі необхідні функції та опції для ефективного управління студійним обладнанням.

3. Вимоги до системи

3.1. Вимоги до системи в цілому

3.1.1. Вимоги до структури та функціонування системи, перелік підсистем

3.1.1.1. Вимоги до способів і засобів інформаційного обміну між компонентами системи

Компоненти системи повинні забезпечувати безперебійний обмін даними між клієнтами та сервером. Для цього використовуються протоколи забезпечення безпеки та стандарти взаємодії, такі як HTTP, HTTPS, TCP/IP. Для передачі даних можуть використовуватися формати JSON або XML.

3.1.1.2. Вимоги до режимів функціонування системи

Система повинна працювати у режимі 24/7, без перебоїв та забезпечувати швидку відповідь на запити користувачів. При виникненні помилок чи неполадок система повинна автоматично відновлюватися до робочого стану. Також, система повинна забезпечувати безпеку даних та можливість резервного копіювання і відновлення даних.

3.1.1.3. Вимоги до діагностування системи

Система повинна мати механізми діагностування та логування помилок та виникнення неполадок. Це дозволить оперативно виявляти та усувати проблеми, що виникають у роботі системи. Для цього можуть використовуватися спеціальні програмні засоби моніторингу та аналізу даних.

3.1.1.4. Вимоги до режимів управління системою

Система повинна мати зручний та простий інтерфейс управління, який дозволить оперативно налаштовувати та контролювати роботу студійного обладнання. Крім того, повинні бути передбачені механізми для забезпечення

доступу до системи та управління правами користувачів. Всі зміни, внесені до системи, повинні бути відображені у відповідних логах та журналах.

3.1.2. Показники призначення

3.1.2.1. Параметри, що характеризують ступінь відповідності системи призначенням:

– максимальна кількість користувачів, що можуть одночасно підключатися до системи;

– максимальний час відповіді системи на запит користувача;

– мінімальна швидкість передачі даних між клієнтами та сервером;

– ступінь точності вимірювань, що здійснюються студійним обладнанням;

– ступінь зручності та простоти використання системи для користувачів з різним рівнем технічної підготовки.

3.1.2.2. Вимоги до пристосовності системи до змін

– можливість додавання нових функцій та модулів до системи без необхідності її повного перепроектування;

– можливість модифікації налаштувань системи в залежності від потреб користувачів;

– можливість інтеграції системи з іншими студійним обладнанням та програмним забезпеченням.

3.1.2.3. Вимоги до збереження працездатності системи в різних ймовірних умовах

– можливість працювати в різних операційних системах та на різних пристроях;

– можливість автоматичного відновлення системи після аварійного вимкнення або збоїв в роботі;

– мінімальний час, необхідний для відновлення роботи системи після збою або відключення від мережі Інтернет.

3.1.3. Вимоги до надійності

3.1.3.1. Склад показників надійності до системи в цілому

– Частота відмов системи (MTBF - mean time between failures);

– Час відновлення системи після відмови (MTTR - mean time to repair);

– Ймовірність відмови за вказаний період (reliability);

– Ступінь забезпечення можливості збереження та відновлення даних в разі відмови системи (fault tolerance);

– Ступінь забезпечення захисту від несанкціонованого доступу до даних (security);

– Ступінь забезпечення захисту від збоїв у роботі мережі (network fault tolerance).

3.1.3.2. Вимоги до надійності технічних засобів і програмного забезпечення

– Надійність програмного забезпечення повинна бути високою і забезпечувати нормальну роботу системи без відмов протягом тривалого часу;

– Надійність технічних засобів повинна бути також високою, забезпечувати стійкість до зовнішніх впливів і забезпечувати нормальну роботу системи протягом тривалого часу.

3.1.3.3. Вимоги до методів оцінки і контролю показників надійності на різних стадіях створення системи

– Проведення аналізу вимог до надійності та встановлення параметрів надійності системи в цілому і її складових;

– Проведення тестування на різних етапах створення системи з метою виявлення можливих відмов та помилок;

– Використання методів математичної статистики для оцінки ймовірності відмови та розрахунку середнього часу між відмовами (MTBF);

– Перевірка надійності системи в реальних умовах експлуатації.

3.1.4. Вимоги до ергономіки та технічної естетики

Додаток повинен мати зручний та логічно побудований інтерфейс для користувача, що дозволяє з легкістю виконувати всі необхідні функції та мінімізує його зусилля та час на операції. Колірна гамма та фонт повинні бути зручними для сприйняття та не викликати візуального дискомфорту або неприємних відчуттів у користувача. Елементи інтерфейсу повинні мати відповідні розміри та відстані між ними, щоб користувач легко міг здійснювати взаємодію з програмою.

3.1.5. Вимоги до експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи

Додаток повинен мати простий та зрозумілий інтерфейс для розуміння користувачем процесів обслуговування та ремонту. Для зберігання даних додаток повинен мати можливість резервного копіювання і відновлення даних, що дозволяє зберігати дані в безпечному місці. Компоненти системи повинні мати можливість заміни та обслуговування без необхідності звертатися до фахівців, що дозволяє знизити витрати на обслуговування. Додаток повинен працювати на різних платформах та оперативних системах і мати можливість оновлення програмного забезпечення з мінімальними зусиллями від користувача.

3.1.6 Вимоги до захисту інформації від несанкціонованого доступу

3.1.6.1. Вимоги до інформаційної безпеки

Система повинна забезпечувати захист інформації, яка обробляється та зберігається, від несанкціонованого доступу, зміни та видалення. Система повинна мати механізми аутентифікації та авторизації користувачів з різними рівнями доступу. Система повинна забезпечувати конфіденційність, цілісність та доступність даних відповідно до вимог законодавства та політиків безпеки інформації компанії. Система повинна забезпечувати

моніторинг захисту інформації, виявлення та реагування на можливі інциденти зі злому безпеки.

3.1.6.2. Вимоги до антивірусного захисту

Система повинна мати антивірусне програмне забезпечення, яке забезпечує захист від шкідливих програм та компонентів. Антивірусне програмне забезпечення повинне оновлюватися регулярно та автоматично, для забезпечення найвищого рівня захисту. Система повинна мати механізми контролю цілісності програмного забезпечення та його складових.

3.1.6.3. Розмежування відповідальності ролей при доступі

Система повинна мати механізми розмежування доступу до інформації в залежності від рівня доступу користувачів та їх ролей. Кожен користувач повинен мати доступ лише до тих функцій та даних, які необхідні для виконання його робочих обов'язків. Система повинна забезпечувати моніторинг та журналювання дій користувачів, щоб виявляти можливі порушення безпеки та ідентифікувати винних осіб.

3.1.7. Вимоги до захисту від впливу зовнішніх факторів

Додаток повинен бути захищений від випадкових або зловмисних дій користувачів, які можуть призвести до витоку конфіденційної інформації або пошкодження системи. Додаток повинен бути захищений від вірусів, троянських програм, шкідливого ПЗ та інших зловмисних програм. Додаток повинен мати можливість забезпечення захисту від DDoS атак та інших типів кібератак. Додаток повинен мати можливість захисту від впливу зовнішніх факторів, таких як електростатичний розряд, електромагнітні поля, радіаційні впливи тощо. Додаток повинен мати можливість автоматично виявляти та ліквідовувати загрози безпеці.

3.1.8. Вимоги безпеки

Додаток повинен забезпечувати безпеку використання студійного обладнання, у тому числі вимоги до висоти розташування обладнання,

відстані до стін, максимальної потужності, захисту від перегріву тощо. Додаток повинен забезпечувати безпеку користувачів, у тому числі забезпечувати безпеку підключення до системи, безпеку вводу та збереження конфіденційної інформації, захист від шахрайства та крадіжки даних. Додаток повинен мати можливість резервного копіювання даних та відновлення роботи системи в разі аварійної ситуації. Додаток повинен дотримуватись стандартів безпеки, таких як PCI DSS, HIPAA, ISO 27001, GDPR тощо.

3.2. Перелік підсистем системи (при наявності підсистем).

Серверна підсистема для зберігання і обробки аудіо-відео даних. Клієнтська підсистема для керування записом та відтворенням аудіо-відео даних. Підсистема для забезпечення безпеки доступу до аудіо-відео даних. Підсистема для забезпечення захисту від впливу зовнішніх факторів. Підсистема для діагностування та моніторингу стану системи. Підсистема для забезпечення комунікації між компонентами системи. Підсистема для забезпечення автоматичної роботи системи в різних умовах. Підсистема для забезпечення технічного обслуговування та ремонту системи.

3.3. Вимоги до видів забезпечення

3.3.1. Вимоги до математичного забезпечення

Система повинна мати можливість обробки та аналізу вхідних даних з високою точністю та швидкістю. Система повинна мати можливість проведення математичних розрахунків та моделювання різних фізичних процесів, пов'язаних з роботою студійного обладнання. Система повинна підтримувати різні математичні бібліотеки та мови програмування, які дозволяють виконувати складні математичні операції.

3.3.2. Вимоги до інформаційного забезпечення

Система повинна забезпечувати безпечний обмін даними між клієнтом та сервером за допомогою захищеного каналу зв'язку. Система повинна мати

можливість зберігати та обробляти великі обсяги даних, пов'язаних з роботою студійного обладнання. Система повинна мати можливість забезпечувати авторизацію користувачів та контролювати рівень доступу до різних функцій та даних системи. Система повинна мати можливість здійснювати архівування та резервне копіювання даних, що забезпечить збереження інформації при можливих випадках втрати даних або несправності системи.

4. Вимоги до програмного забезпечення

Додаток має бути написаний на мові програмування з можливістю підтримки множини одночасних підключень. Додаток повинен мати графічний інтерфейс користувача, який забезпечує зручне інтуїтивне керування системою. Додаток повинен містити розширений набір функцій для редагування, збереження та відтворення аудіо- та відеофайлів у режимі реального часу. Додаток повинен забезпечувати можливість додавання ефектів та фільтрів до звукових та відео-записів. Додаток повинен мати функцію автоматичного створення звукових або відео-матеріалів з заданої послідовності фрагментів.

5. Вимоги до технічного забезпечення

Клієнтський додаток має бути сумісним з операційними системами Windows 10 та MacOS Catalina і вище. Серверна частина повинна бути розміщена на сервері з належними характеристиками, забезпечуючи швидкість передачі даних та доступність сервера для клієнтів. Клієнтський додаток має вимагати наявності мінімум 4 ГБ оперативної пам'яті та 1 ГБ вільного місця на жорсткому диску. Рекомендована роздільна здатність екрану для роботи з додатком - не менше 1280x768 пікселів. Мінімальна швидкість Інтернет-з'єднання для плавної роботи додатка - 5 Мбіт/с.

6. Вимоги до методичного забезпечення

Документація до додатку повинна бути доступною для користувачів в електронному вигляді та містити докладний опис функціоналу та принципів роботи.

1.4. Висновки до розділу 1

Сучасні студійні обладнання стають все більш компактними та зручними у використанні завдяки використанню цифрових технологій. Це забезпечує збільшення продуктивності та ефективності в роботі. За останні роки онлайн-стрімінг став дуже популярним, що призвело до збільшення попиту на студійне обладнання для відеозйомки та живого стрімінгу. Ринок студійного обладнання є досить конкурентним, що призводить до зниження цін на деякі продукти та зростання якості інших. Сучасні студійні обладнання мають все більше функцій, які забезпечують більш глибоку обробку та редагування аудіо- та відеоматеріалів. Це забезпечує більшу гнучкість та ефективність в роботі. За останні кілька років зросла важливість якості та розширення відео, що призвело до збільшення попиту на студійне обладнання, яке забезпечує зйомку та обробку відео в високій якості.

У цілому, розвиток студійного обладнання залежить від розвитку технологій та відповідного ринкового попиту. Якщо розвиток продовжиться на поточному рівні, можна очікувати появу нових технологій та виробів, які забезпечать більшу ефективність та продуктивність в галузі студійної роботи.

Основні етапи розробки та впровадження такого додатку можуть включати наступне: визначення вимог до додатку, вибір технологій, розробка серверної частини, розробка клієнтської частини, тестування та налагодження, впровадження, підтримка та доповнення.

Розробка та впровадження клієнт-серверного додатку для студійного обладнання може бути складним та часовим процесом. Проте, якщо ви маєте необхідний досвід та знання, а також використовуєте ефективні інструменти,

то ви зможете розробити потрібний додаток, який відповідатиме всім вимогам та задовольнить потреби користувачів.

РОЗДІЛ 2
ПРОЄКТУВАННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ З РЕЛІЗАЦІЇ СТУДІЙНОГО ОБЛАДНАННЯ

2.1. Чинники, що визначають хід і результати роботи клієнт-серверного додатку

Хід і результати роботи клієнт-серверного додатку з реалізації студійного обладнання залежать від декількох факторів. Основні з них наведені на рис. 2.1.

Рис. 2.1. Чинники, що визначають хід і результати роботи клієнт-серверного додатку

Джерело: побудовано автором

1. Якість програмного забезпечення: якість програмного забезпечення визначає, наскільки добре додаток може взаємодіяти з обладнанням студії і виконувати функції, які передбачені для такого типу додатку.

Якість програмного забезпечення є ключовим фактором, який визначає успішність клієнт-серверного додатку для студійного обладнання. Якість програмного забезпечення охоплює ряд показників, таких як функціональність, надійність, продуктивність та безпеку. Кожен з цих показників може впливати на ефективність додатку і його здатність до взаємодії з обладнанням студії.

Функціональність програмного забезпечення студійного обладнання визначає, наскільки добре додаток може виконувати задані функції та завдання. Це може включати функції запису, міксування та монтажу звукових доріжок, налаштування рівня гучності та інші функції, які необхідні для роботи зі звуковим обладнанням студії. Надійність програмного забезпечення визначається його здатністю до роботи без помилок та аварій.

Продуктивність програмного забезпечення визначається його здатністю до ефективної роботи з великим обсягом даних та завдань, що пов'язані зі звуковим обладнанням студії. Наприклад, програмне забезпечення повинно працювати швидко і без затримок під час роботи з багатьма доріжками та ефектами звукового монтажу.

Безпека програмного забезпечення також є важливим показником якості. Програмне забезпечення має бути захищеним від несанкціонованого доступу до звукових даних та інформації про користувачів. Такі заходи безпеки, як шифрування даних, автентифікація користувачів та забезпечення приватності даних, допоможуть забезпечити захист від можливих загроз

2. Кількість одночасних з'єднань: кількість одночасних з'єднань між клієнтом і сервером може впливати на продуктивність і швидкість виконання завдань. Недостатня кількість з'єднань може призвести до затримок у роботі додатку, тоді як забагато з'єднань може спричинити втрату продуктивності.

Кількість одночасних з'єднань є важливим показником для клієнт-серверних додатків, таких як додатки для студійного обладнання. Цей

показник визначає максимальну кількість з'єднань, яку сервер може обслуговувати одночасно. Якщо кількість з'єднань перевищує максимальне значення, сервер може почати повільно відповідати на запити клієнтів або взагалі зупинити роботу.

Кількість одночасних з'єднань може бути обмежено різними факторами, такими як обсяг ресурсів, доступних для сервера, або характеристики мережі, на якій працює додаток. Якщо сервер не може обслуговувати достатню кількість з'єднань, це може призвести до затримок у відповіді на запити клієнтів або відмови у підключенні.

Недостатня кількість одночасних з'єднань може призвести до того, що додаток не зможе виконувати всі завдання, які потрібно виконати, що може вплинути на його продуктивність та ефективність. З іншого боку, забагато з'єднань може призвести до втрати продуктивності, тому що серверу потрібно буде виконувати багато запитів одночасно, що може затримати відповідь на запити клієнтів.

Тому важливо правильно налаштувати кількість одночасних з'єднань для додатку студійного обладнання, забезпечивши оптимальний баланс між продуктивністю та ефективністю роботи.

3. Пропускна здатність мережі: пропускна здатність мережі може впливати на швидкість передачі даних між клієнтом і сервером. Низька пропускна здатність може призвести до затримок у роботі додатку, тоді як висока пропускна здатність може дозволити виконувати завдання швидше.

Пропускна здатність мережі визначає, яка кількість даних може бути передана через мережу протягом певного часового інтервалу. Цей фактор може впливати на продуктивність додатку, особливо коли він працює з великим обсягом даних або з великою кількістю користувачів.

Якщо пропускна здатність мережі низька, то передача даних між клієнтом і сервером буде повільною, що може призвести до затримок у

виконанні завдань і погіршення загальної продуктивності додатку. З іншого боку, висока пропускна здатність мережі дозволяє передавати дані швидше, що дозволяє додатку виконувати завдання швидше і збільшує загальну продуктивність.

Пропускна здатність мережі може бути обмежена різними факторами, такими як швидкість інтернет-з'єднання, тип мережевого обладнання та кількість користувачів, що використовують мережу. Щоб забезпечити високу продуктивність додатку, важливо мати належне мережеве обладнання та достатньо швидке інтернет-з'єднання.

4. Конфігурація обладнання: конфігурація обладнання студії може впливати на роботу клієнт-серверного додатку. Якщо обладнання не задовольняє вимогам програмного забезпечення, то можуть виникати проблеми з продуктивністю та затримки в роботі додатку.

Конфігурація обладнання студії охоплює в собі аспекти, такі як процесор, оперативна пам'ять, жорсткий диск та інші компоненти. Якщо обладнання не відповідає вимогам програмного забезпечення, то це може призвести до погіршення продуктивності та низької швидкості роботи додатку.

Наприклад, якщо обладнання має недостатню кількість процесорних ядер або обсяг оперативної пам'яті, то воно може не здатне ефективно обробляти запити від клієнтів і призводити до затримок. Недостатній обсяг жорсткого диска може призвести до відмови в записі або збереженні даних, що може призвести до втрати даних і порушення продуктивності додатку.

Таким чином, важливо забезпечити відповідну конфігурацію обладнання для забезпечення надійної та ефективної роботи клієнт-серверного додатку, що використовує студійне обладнання.

5. Навички користувачів: навички користувачів можуть впливати на ефективність використання клієнт-серверного додатку. Користувачі, які

мають високі навички використання програмного забезпечення, можуть бути більш ефективні в роботі з додатком і отримувати кращі результати. Навички користування додатком можуть включати розуміння функцій, правильне налаштування параметрів та здатність до швидкого виявлення та виправлення помилок.

Користувачі з недостатніми навичками можуть виконувати завдання повільно та невірно, що може призвести до помилок та простоїв у роботі додатку. Більш досвідчені користувачі зможуть швидше виявляти та вирішувати проблеми, що дозволить ефективніше використовувати додаток та отримувати кращі результати.

Однак, необхідно зазначити, що навички користувачів можуть бути поліпшені через навчання та підтримку, що дозволить зменшити вплив цього фактору на ефективність роботи додатку.

6. Розміщення сервера: розміщення сервера може впливати на продуктивність і швидкість роботи додатку. Розміщення сервера на віддаленому сервері може збільшувати час відповіді, тоді як розміщення сервера на місцевому сервері може забезпечувати більш швидку взаємодію між клієнтом і сервером.

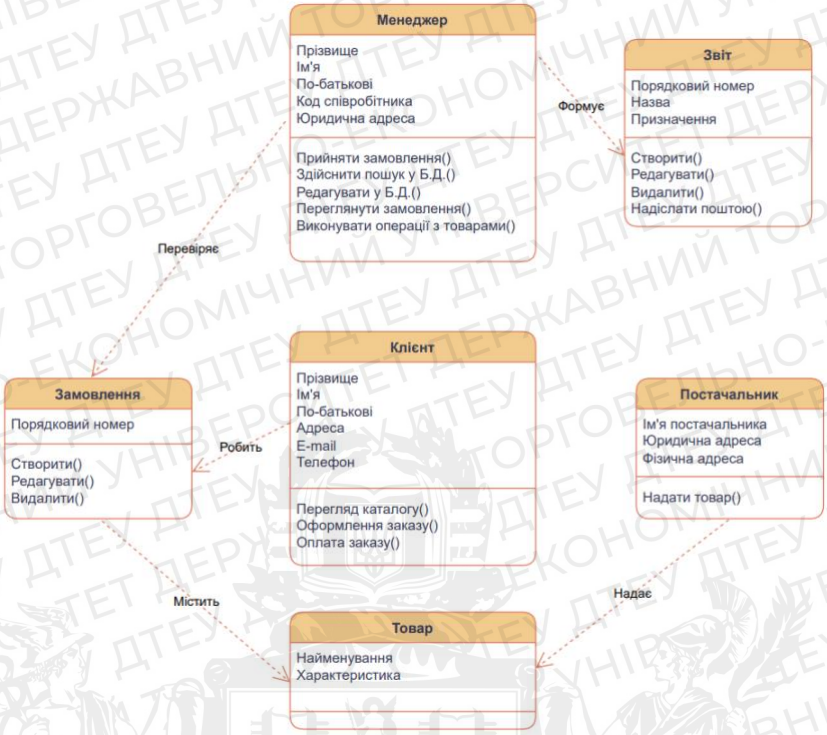

Рис. 2.2. Діаграма класів

Джерело: побудовано автором

Для розуміння впливу розміщення сервера на продуктивність та швидкість роботи додатку, слід розглянути наступні фактори:

– Відстань: чим більше відстань між клієнтом та сервером, тим більше часу потрібно на передачу даних між ними. Це може призвести до збільшення часу відповіді сервера на запити клієнтів та затримок у виконанні завдань.

– Пропускна здатність каналу зв'язку: пропускна здатність каналу зв'язку, який використовується для з'єднання клієнта та сервера, може впливати на швидкість передачі даних між ними. Якщо пропускна здатність низька, то можуть виникати затримки у виконанні завдань та обробці запитів.

– Навантаження на мережу: розміщення сервера може впливати на навантаження на мережу. Якщо сервер знаходиться на віддаленому місці, то можуть виникати проблеми з навантаженням на мережу під час передачі великої кількості даних між клієнтом та сервером.

– Безпека: розміщення сервера може впливати на безпеку додатку. Якщо сервер знаходиться в ненадійному місці або використовується небезпечний канал зв'язку, то можуть виникати проблеми з безпекою даних, що передаються між клієнтом та сервером.

Отже, вибір оптимального розміщення сервера може допомогти забезпечити ефективну та швидку роботу додатку та зменшити час відповіді сервера на запити клієнтів.

7. Рівень безпеки: рівень безпеки додатку може впливати на його роботу та результати. Низький рівень безпеки може призвести до витоку даних, тоді як високий рівень безпеки може забезпечити захист від несанкціонованого доступу до даних.

Рівень безпеки є дуже важливим аспектом клієнт-серверного додатку, особливо якщо додаток містить конфіденційну або особисту інформацію. Рівень безпеки визначає, наскільки добре захищені дані від несанкціонованого доступу, а також наскільки добре захищений додаток від зловмисників.

Основні проблеми, пов'язані з низьким рівнем безпеки, включають можливість витоку конфіденційної інформації, можливість несанкціонованого доступу до даних та можливість атак на додаток. Недостатній рівень безпеки може призвести до серйозних проблем для бізнесу та для користувачів.

Високий рівень безпеки може забезпечити захист від різних видів атак, таких як атаки на вразливості програмного забезпечення, відмова в обслуговуванні (DoS) та інші атаки, що мають на меті отримати доступ до конфіденційної інформації або завдати шкоди додатку. Для забезпечення високого рівня безпеки, можуть використовуватися різноманітні техніки, такі як шифрування даних, перевірка автентичності користувачів, контроль

доступу, виявлення та запобігання атакам, аудит безпеки та інші заходи безпеки.

Таким чином, рівень безпеки є важливим аспектом при розробці та експлуатації клієнт-серверних додатків, оскільки це може впливати на конфіденційність, цілісність та доступність даних користувачів, а також на репутацію та бізнес-показники компанії.

8. Обсяг оброблюваних даних: обсяг оброблюваних даних може впливати на продуктивність і швидкість роботи додатку.

Для обробки даних, необхідно забезпечити достатню кількість ресурсів, таких як процесор, оперативна пам'ять, дисковий простір тощо. Якщо обсяг даних, що оброблюється, є великим, то це може вимагати більш потужного обладнання для швидкої обробки та забезпечення продуктивності додатку. Також великий обсяг даних може вплинути на час відповіді додатку, який може збільшуватися при обробці більшої кількості даних.

У разі меншого обсягу даних, потрібне обладнання може бути менш потужним, що дозволяє знизити вартість та забезпечити оптимальну продуктивність. Однак, навіть з меншим обсягом даних, додаток може працювати повільно, якщо не буде оптимізовано правильно. Таким чином, обсяг оброблюваних даних є одним з факторів, які можуть впливати на продуктивність та швидкість роботи додатку.

9. Використовувані технології: використані технології можуть впливати на роботу додатку та результати. Вибір правильних технологій може забезпечити більш ефективну роботу додатку та отримання кращих результатів.

Вибір правильних технологій може бути важливим для успіху клієнт-серверного додатку. Технології можуть включати мови програмування, бази даних, фреймворки та інші інструменти, які використовуються для розробки додатку.

Наприклад, використання відкритих джерел може забезпечити більш ефективну роботу додатку, оскільки це дає можливість користуватися готовими бібліотеками та інструментами для розробки. Вибір правильної бази даних може забезпечити ефективне зберігання та обробку даних.

Крім того, вибір правильних технологій може впливати на швидкість розробки додатку, його продуктивність, надійність та безпеку. Наприклад, використання фреймворків може спростити розробку та забезпечити більшу стабільність додатку, тоді як використання мов програмування з високою продуктивністю може забезпечити швидку роботу додатку.

Таким чином, вибір правильних технологій може бути ключовим для успіху клієнт-серверного додатку, оскільки це може впливати на його продуктивність, швидкість роботи, надійність та безпеку.

10. Потужність обладнання: потужність обладнання може впливати на продуктивність і швидкість роботи додатку. Менш потужне обладнання може призводити до затримок та недостатньої продуктивності, тоді як більш потужне обладнання може забезпечувати більш швидку роботу додатку та отримання кращих результатів.

Потужність обладнання включає в себе різні параметри, такі як обсяг оперативної пам'яті, швидкість процесора, кількість та швидкість жорстких дисків і т. д. Якщо обладнання не відповідає вимогам програмного забезпечення, то можуть виникати проблеми з продуктивністю і швидкістю роботи додатку, що може призвести до зниження ефективності роботи користувача і негативно позначитися на результативності додатку. Більш потужне обладнання може забезпечити більш швидку обробку даних та збільшити продуктивність додатку, що дозволить користувачеві швидше виконувати свої завдання і отримувати кращі результати. Однак, не завжди потрібно вибирати найпотужніше обладнання, оскільки це може бути зайвим і дорогим. Вибір обладнання повинен відповідати вимогам додатку і

потребам користувачів, щоб забезпечити максимальну ефективність і оптимальні витрати.

2.2. Інструменти розробки клієнт-серверного додатку з реалізації студійного обладнання

Розробка клієнт-серверного додатку для студійного обладнання може включати в себе використання різних інструментів залежно від конкретних вимог проекту. Ось декілька загальних інструментів, які були корисні при розробці такого додатку:

1. Мови програмування: Для розробки клієнт-серверного додатку можна використовувати різні мови програмування, такі як Java, Python, C++, C# або JavaScript. Опис даних мов наведений в таблиці 2.1.

Таблиця 2.1.

Мови програмування

Мова програмування	Характеристика
Java	є популярною мовою програмування для розробки серверних додатків. Вона має багато бібліотек та фреймворків, які дозволяють розробляти ефективні та масштабовані серверні додатки. Java також має велику спільноту розробників, що забезпечує підтримку та поширення знань
Python	також є популярною мовою програмування для розробки серверних додатків. Вона має простий синтаксис та багато бібліотек, що дозволяє розробникам швидко створювати функціональні серверні додатки

Кінець табл. 2.1.

Мова програмування	Характеристика
C++ та C#	також можуть бути використані для розробки серверних додатків. Вони мають високу продуктивність та можуть бути корисні для розробки додатків, що потребують великої швидкодії
JavaScript	може бути використаний для розробки клієнтської частини додатку, якщо використовується архітектура клієнт-сервер. Він є популярним мовою програмування для розробки веб-додатків та має велику спільноту розробників.

Кожна з програмних мов має свої переваги та недоліки. Вибір мови залежить від вимог проекту, досвіду розробника та інших факторів.

У загальному, вибір мови програмування залежить від вимог проекту, досвіду розробника та інших факторів, таких як доступність бібліотек та фреймворків.

2. Бази даних: Для зберігання даних про обладнання та користувачів можна використовувати різні бази даних, такі як MySQL, PostgreSQL або MongoDB.

Для розробки клієнт-серверного додатку з реалізації студійного обладнання необхідно зберігати дані про обладнання, користувачів та їх взаємодію. Для цього можна використовувати різні бази даних, залежно від вимог проекту.

MySQL є однією з найбільш популярних реляційних баз даних, яка використовує мову SQL для зберігання та керування даними. Вона має велику спільноту розробників та підтримує багато функціональних можливостей, таких як транзакції, реплікація та шифрування даних.

PostgreSQL є іншою популярною реляційною базою даних, яка також використовує мову SQL. Вона має більш продвинуті можливості у порівнянні з MySQL, такі як підтримка відображень, схем та збережених процедур.

MongoDB є нереляційною базою даних, яка зберігає дані у вигляді документів. Вона має гнучку схему даних, що дозволяє легко додавати та змінювати поля документів. MongoDB також має вбудовану підтримку геопросторових запитів та може бути корисною для проектів, які потребують обробки геоданих.

Вибір бази даних залежить від вимог проекту, а також від вмінь розробника та наявних ресурсів. Якщо проект вимагає швидкості та простоти, MySQL може бути відмінним вибором. PostgreSQL може бути корисним, якщо проект потребує продвинутих можливостей баз даних. MongoDB може бути корисною для проектів, що використовують геодані або мають гнучку схему даних.

3. Фреймворки: Розробка за допомогою фреймворків може спростити процес розробки та зменшити кількість написаного коду. Наприклад, для розробки клієнт-серверного додатку можна використовувати фреймворки, такі як Spring для Java, Flask або Django для Python, або Express.js для JavaScript.

Фреймворки є набором інструментів та бібліотек, які допомагають розробникам створювати програмне забезпечення швидше та ефективніше. Вони забезпечують структуру проекту, вбудовані функціональність та механізми, що дозволяють розробникам зосередитися на бізнес-логіці програми, а не на технічних деталях.

Spring є одним з найбільш популярних фреймворків для розробки клієнт-серверних додатків на мові Java. Він забезпечує інверсію керування та внедрення залежностей, що спрощує процес розробки. Spring також має вбудовану підтримку для різних інших технологій, таких як JPA, Hibernate,

та Spring Security, що дозволяє розробникам ефективно використовувати ці технології у своїх проектах.

Flask та Django є двома популярними фреймворками для розробки веб-додатків на мові Python. Flask забезпечує мінімальну конфігурацію та простоту у використанні. Django, з іншого боку, надає більш продвинуті можливості, такі як адміністративний інтерфейс, підтримку ORM та вбудовану безпеку.

Express.js є фреймворком для розробки клієнт-серверних додатків на мові JavaScript. Він забезпечує простоту та швидкість у розробці, а також підтримує асинхронний код та маршрутизацію запитів.

Вибір фреймворку залежить від вимог проекту, а також від вмінь та досвіду розробника. Spring може бути корисним для великих проектів на Java, Flask або Django можуть бути корисними для проектів на Python.

4. REST API [10]: REST API може використовуватися для забезпечення комунікації між клієнтом та сервером, дозволяючи взаємодіяти з сервером через HTTP запити.

REST (Representational State Transfer) є стандартним архітектурним стилем для розробки веб-додатків, що підтримують легку та ефективну комунікацію між клієнтом та сервером. REST API є реалізацією цього стилю та забезпечує стандартизовану форму обміну даними між клієнтом та сервером через HTTP протокол.

REST API базується на ресурсах та методах HTTP протоколу. Кожен ресурс має унікальний ідентифікатор (URI), а методи HTTP (GET, POST, PUT, DELETE тощо) використовуються для взаємодії з ресурсами.

Наприклад, для отримання списку всіх обладнань клієнт може відправити GET запит на URI, що відповідає списку обладнань, наприклад, /api/equipment. Сервер оброблює цей запит та повертає клієнту список обладнань у форматі, який було вказано в запиті.

REST API є простим у використанні та забезпечує зручну взаємодію між клієнтом та сервером. Це дозволяє забезпечити підтримку декількох клієнтів з різними інтерфейсами, які можуть взаємодіяти з сервером за допомогою одного та того ж REST API.

5. Git: Git може бути корисним інструментом для контролю версій коду та спільної роботи над проектом.

Git є системою керування версіями, яка дозволяє зберігати та відстежувати зміни у коді програми. Користування Git допомагає зберігати різні версії коду та відновлювати попередні версії у разі потреби. Git забезпечує зручний інтерфейс для спільної роботи над проектом та дозволяє кільком розробникам працювати з одним і тим же кодом, зберігаючи історію змін та забезпечуючи зручний механізм злиття змін.

Git дозволяє створювати гілки (branches) коду, що дозволяє розробникам відокремлювати свою роботу від основного потоку розробки та експериментувати з новими функціями без ризику зіткнення з основним кодом. Після того, як розробник закінчує роботу з гілкою, він може об'єднати її з основним кодом, що дозволяє уникнути конфліктів та забезпечити більш ефективний процес розробки.

Git також дозволяє зберігати інформацію про те, хто вносив зміни та коли вони були внесені, що дозволяє відслідковувати внесені зміни та забезпечує більш прозорий та ефективний процес розробки.

6. WebSocket [10]: WebSocket дозволяє встановлювати двостороннє з'єднання між клієнтом та сервером, що може бути корисним для передачі даних в реальному часі.

WebSocket - це протокол, який дозволяє встановлювати постійне двостороннє з'єднання між клієнтом та сервером з мінімальною затримкою. Це означає, що замість того, щоб клієнт надсилати запити на сервер і чекати

відповіді, сервер може ініціювати передачу даних клієнту у режимі реального часу.

WebSocket забезпечує повністю функціональне двостороннє з'єднання між клієнтом та сервером, дозволяючи надсилати дані як клієнту, так і серверу в будь-який момент часу. Крім того, WebSocket підтримує відкрите з'єднання, що означає, що з'єднання може бути відкрите незалежно від того, чи виконується запит чи передача даних. Це забезпечує більш ефективну передачу даних і дозволяє клієнту та серверу більш ефективно взаємодіяти між собою.

WebSocket може бути використаний для різних цілей, таких як чати, онлайн-ігри, спільне відео, потокові медіа тощо. Для використання WebSocket зазвичай використовують спеціальні бібліотеки або фреймворки, такі як Socket.IO для JavaScript або Spring WebSocket для Java.

7. MQTT: MQTT - це протокол для передачі повідомлень між різними пристроями, що може бути корисним для передачі даних між студійним обладнанням та сервером.

MQTT (Message Queuing Telemetry Transport) - це легкий протокол передачі повідомлень для інтернету речей (IoT), що дозволяє ефективно передавати повідомлення між різними пристроями. Протокол MQTT забезпечує доставку повідомлень з гарантованою якістю обслуговування (QoS), тобто можливі різні рівні підтвердження доставки повідомлення, від найнижчого рівня (QoS 0) до найвищого рівня (QoS 2).

Протокол MQTT дозволяє відправляти повідомлення за допомогою таких параметрів, як "тема" (topic), "повідомлення" (message) та "якість обслуговування" (QoS). Клієнт може підписатися на певну тему (topic) та отримувати повідомлення, які були надіслані на цю тему. Це дозволяє клієнту отримувати оновлення в режимі реального часу та реагувати на них.

MQTT може бути корисним для передачі даних між студійним обладнанням та сервером, особливо якщо мається на увазі передача даних з великою кількістю показників в реальному часі. Протокол MQTT забезпечує ефективну передачу даних з низькою пропускною здатністю мережі, що робить його привабливим варіантом для використання у зв'язку зі студійним обладнанням.

8. Cloud Services: Cloud Services, такі як Amazon Web Services (AWS) або Microsoft Azure, можуть бути корисними для розгортання та підтримки клієнт-серверного додатку в хмарному середовищі. Це може забезпечити більшу масштабованість та доступність додатку.

Cloud Services - це інфраструктура та послуги, які надаються через Інтернет, а не на локальному сервері. Це можуть бути платформи для розгортання додатків, хмарні сервіси зберігання даних та інші корисні послуги для розробки та експлуатації програмного забезпечення.

Amazon Web Services (AWS) та Microsoft Azure - це дві найбільші платформи хмарних сервісів. AWS пропонує широкий спектр послуг, включаючи зберігання даних, розгортання та керування серверами, аналітичні та машинно-навчальні сервіси, а також послуги безпеки та ідентифікації. Azure також пропонує багато послуг, таких як зберігання даних, віртуальні машини та машинне навчання, а також має підтримку для різних мов програмування.

Використання хмарних сервісів для розгортання та підтримки клієнт-серверного додатку може мати декілька переваг. Спочатку, використання хмарних сервісів дозволяє зменшити витрати на розгортання та підтримку власної інфраструктури, замінивши їх передплатою на послуги в хмарі. Крім того, хмарні сервіси можуть забезпечити більшу масштабованість додатку, дозволяючи швидко змінювати кількість ресурсів, що використовуються, залежно від потреб користувачів. Нарешті, хмарні сервіси також

забезпечують високу доступність додатку, забезпечуючи зменшення часу простою та перебоїв в роботі.

Ці інструменти використані для розробки та підтримки клієнт-серверного додатку з реалізації студійного обладнання.

2.3. Висновки до розділу 2

Хід і результати роботи клієнт-серверного додатку з реалізації студійного обладнання залежать від декількох факторів: якість програмного забезпечення, кількість одночасних з'єднань, пропускна здатність мережі, конфігурація обладнання, навички користувачів, розміщення сервера, рівень безпеки, обсяг оброблюваних даних, використовувані технології, потужність обладнання.

Для розробки клієнт-серверного додатку можна використовувати різні мови програмування, такі як Java, Python, C++, C# або JavaScript. Для зберігання даних про обладнання та користувачів можна використовувати різні бази даних, такі як MySQL, PostgreSQL або MongoDB. Розробка за допомогою фреймворків може спростити процес розробки та зменшити кількість написаного коду. Наприклад, для розробки клієнт-серверного додатку можна використовувати фреймворки, такі як Spring для Java, Flask або Django для Python, або Express.js для JavaScript. REST API може використовуватися для забезпечення комунікації між клієнтом та сервером, дозволяючи взаємодіяти з сервером через HTTP запити. Git може бути корисним інструментом для контролю версій коду та спільної роботи над проектом. WebSocket дозволяє встановлювати двостороннє з'єднання між клієнтом та сервером, що може бути корисним для передачі даних в реальному часі. MQTT - це протокол для передачі повідомлень між різними пристроями, що може бути корисним для передачі даних між студійним обладнанням та сервером. Cloud Services, такі як Amazon Web Services (AWS)

або Microsoft Azure, можуть бути корисними для розгортання та підтримки клієнт-серверного додатку в хмарному середовищі. Це може забезпечити більшу масштабованість та доступність додатку.

РОЗДІЛ 3
РОЗРОБКА ТА ВПРОВАДЖЕННЯ КЛІЄНТ-СЕРВЕРНОГО ДОДАТКУ З РЕЛІЗАЦІЇ СТУДІЙНОГО ОБЛАДНАННЯ

3.1. Функціональні характеристики розробленого клієнт-серверного додатку «Muzline» з реалізації студійного обладнання

Клієнт-серверний додаток «Muzline» розроблений для керування студійним обладнанням і має наступні функціональні характеристики:

1. Забезпечення зв'язку між клієнтом та сервером: додаток дозволяє клієнту підключатись до серверу, щоб отримати доступ до студійного обладнання.

Ця функціональна характеристика розробленого клієнт-серверного додатку «Muzline» забезпечує зв'язок між клієнтом і сервером для передачі даних та керування студійним обладнанням. Клієнт зазвичай є користувачем додатку, а сервер - це комп'ютер або пристрій, на якому запущений додаток для керування обладнанням.

Для забезпечення зв'язку між клієнтом та сервером, додаток використовує мережеві протоколи, такі як TCP / IP або UDP, що дозволяє передавати дані між комп'ютерами по мережі. Клієнт підключається до сервера за допомогою інтерфейсу додатку, вводить необхідні дані для підключення, такі як IP-адресу та порт, та налаштовує з'єднання з сервером.

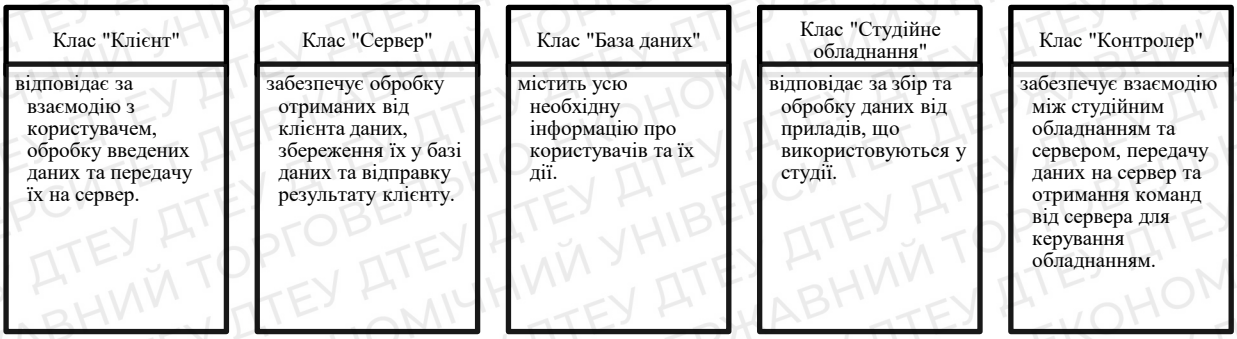

Рис. 3.1. Блок-схема зв'язків між класами клієнт-серверного додатку «Muzline» з реалізації студійного обладнання [складено автором на основі Додатку А]

Джерело: побудовано автором

Класи "Клієнт" та "Сервер" взаємодіють між собою через мережу Інтернет, використовуючи протоколи передачі даних, такі як HTTP або TCP. Клас "Клієнт" також може взаємодіяти з класом "Студійне обладнання" для отримання даних про стан приладів та передачі їх на сервер. Клас "Контролер" взаємодіє з класом "Студійне обладнання" для керування приладами з боку сервера. Клас "База даних" використовується для збереження інформації про користувачів та їх дії, а також для забезпечення безпеки даних.

Після успішного підключення до сервера, клієнт може взаємодіяти зі студійним обладнанням, виконувати різні функції та операції, такі як запис звуку, зміна налаштувань обладнання тощо. Додаток дозволяє передавати дані між клієнтом та сервером у режимі реального часу, що дозволяє користувачам ефективно керувати обладнанням та створювати професійну аудіо продукцію.

2. Керування обладнанням: користувач може керувати різними параметрами обладнання, такими як гучність, еквалайзер, панорамування та інші.

Ця функціональна характеристика дозволяє користувачеві керувати різними параметрами студійного обладнання за допомогою додатку «Muzline». Обладнання може включати аудіоінтерфейс, мікшерний пульт, монітори, мікрофони та інші елементи студійного обладнання.

Для керування параметрами обладнання додаток має інтерфейс, який дає користувачеві доступ до різних опцій керування обладнанням. Наприклад, користувач може змінювати гучність сигналу від окремих джерел, налаштовувати еквалайзер для оптимального звучання аудіо, панорамувати звук для розподілу звукових джерел у просторі тощо.

Додаток може мати різні режими керування, такі як ручне керування, автоматичний режим або змішування звуку. Ручне керування дозволяє користувачеві вручну налаштувати параметри обладнання, а автоматичний режим дозволяє програмно налаштувати параметри на основі аналізу вхідного аудіо. Змішування звуку дає можливість користувачеві створювати нові аудіо-треки з використанням різних джерел звуку, що можуть бути записані в реальному часі.

Крім того, додаток може забезпечувати режим збереження налаштувань обладнання, що дозволяє користувачеві зберігати налаштування для подальшого використання. Це дозволяє швидко переключатись між різними настройками обладнання та зберігати налаштування для різних проектів.

3. Збереження налаштувань: додаток може зберігати налаштування користувача для подальшого використання.

Ця функціональна характеристика дозволяє користувачам зберігати налаштування, які вони створюють для керування студійним обладнанням у додатку «Muzline». Це дозволяє користувачам зберігати настройки, які вони ввели у додаток, щоб не доводилось вводити їх знову при наступних використаннях додатку.

Збереження налаштувань може бути виконане у формі профілів користувача або шаблонів налаштувань. Профілі користувача дозволяють зберігати налаштування для конкретного користувача та використовувати їх при наступних використаннях додатку. Наприклад, якщо у користувача є ряд улюблених настройок для свого обладнання, він може зберегти їх у своєму профілі та використовувати їх знову та знову.

Шаблони налаштувань дозволяють зберігати настройки для конкретного проекту, де використовується студійне обладнання. Це дозволяє користувачам швидко встановлювати настройки для конкретного проекту, щоб сконцентруватись на творчості, а не на пошуку та встановленні необхідних настройок.

Крім того, додаток може забезпечувати можливість експорту та імпорту налаштувань, щоб користувачі могли зберігати свої настройки в інших форматах або обмінюватись ними з іншими користувачами. Наприклад, користувач може експортувати свої настройки у форматі файлу, а потім імпортувати їх на іншому комп'ютері або надіслати їх іншому користувачу, щоб він міг використовувати ті самі настройки.

4. Можливість підключення різних пристроїв: додаток може працювати з різними типами пристроїв, такими як мікшерні пульти, звукові карти та інші.

Функція можливості підключення різних пристроїв дозволяє користувачеві використовувати різноманітне студійне обладнання для змішування, запису та обробки аудіосигналів. Додаток може підключатися до різних типів пристроїв, таких як мікшерні пульти, звукові карти, аудіоінтерфейси та інші, які можуть бути використані для забезпечення потоку аудіоданих між клієнтом та сервером.

Ця функція забезпечує широкі можливості для користувачів, щоб працювати з різними типами обладнання та виконувати різноманітні задачі з

обробки звуку. Додаток може автоматично розпізнавати підключені пристрої та дозволяти користувачеві вибирати, який пристрій використовувати для роботи зі звуком.

Крім того, функція підключення різних пристроїв може дозволяти користувачеві працювати зі звуком в режимі реального часу та проводити звукові тестування, щоб перевірити різні параметри звуку, такі як гучність, еквалайзер та інші, на різних типах обладнання.

5. Зручний інтерфейс: додаток має простий і зрозумілий інтерфейс, що дозволяє користувачеві легко керувати обладнанням.

Функція зручного інтерфейсу є важливою для забезпечення зручності та ефективності використання додатку. Додаток «Muzline» як раз таки відповідає цьому критерію.

Інтерфейс додатку може бути організований за принципом меню, що дозволяє користувачеві легко навігувати по функціональних можливостях додатку та швидко знаходити необхідні опції та налаштування. Крім того, додаток може мати візуальні елементи, такі як кнопки, слайдери та інші елементи керування, що роблять роботу з обладнанням ще більш простою та зручною.

Для забезпечення ефективного використання додатку, інтерфейс може бути добре організований та логічно структурований. Усі ці функції забезпечують зручний та ефективний робочий процес, що дозволяє користувачеві легко керувати студійним обладнанням та виконувати різноманітні задачі з обробки звуку.

6. Можливість зберігання налаштувань за проектами: додаток може зберігати налаштування обладнання за проектами, що дозволяє користувачеві швидко перемикатись між різними проектами без необхідності повторного встановлення налаштувань.

Функціональність зберігання налаштувань за проектами є важливою складовою клієнт-серверного додатку "Muzline". Користувачі можуть створювати проекти та зберігати налаштування обладнання для кожного проекту окремо.

Така функціональність дозволяє зберігати налаштування для різних клієнтів або проектів, і користувачі можуть легко перемикатись між проектами, не втрачаючи свої налаштування. Крім того, ця функціональність дозволяє користувачеві швидко переключатись між різними сесіями та проектами, що дозволяє значно збільшити продуктивність роботи з обладнанням та спростити управління налаштуваннями.

Зберігання налаштувань за проектами дозволяє також зберігати налаштування в різних форматах та конфігураціях, що є особливо корисним для проектів з різними вимогами до звукового обладнання. Це дозволяє зменшити час, необхідний для зміни налаштувань, та дозволяє користувачеві ефективно працювати з різними налаштуваннями обладнання.

Таким чином, можливість зберігання налаштувань за проектами в додатку "Muzline" є дуже корисною та практичною функціональністю для користувачів, що працюють зі студійним звуковим обладнанням.

7. Підтримка різних форматів звукових файлів: додаток може працювати з різними форматами звукових файлів, такими як WAV, MP3, AIFF та інші.

8. Зручний пошук файлів: додаток має зручний інтерфейс для пошуку і відтворення звукових файлів.

Для зручності користувача, додаток «Muzline» має вбудований пошук файлів звуку. Користувач може шукати файли звуку з допомогою різних параметрів, таких як назва файлу, альбом, виконавець, жанр та інші.

Додаток підтримує різні формати файлів звуку, такі як MP3, WAV, FLAC та інші. Крім того, користувач може організувати свою музичну

бібліотеку, створювати плейлисти, додавати теги та робити інші операції зі звуковими файлами.

Зручний інтерфейс дозволяє користувачеві легко знайти потрібний файл звуку та відтворити його без зайвих зусиль.

9. Можливість додавання ефектів: додаток може додавати різні ефекти до звукових файлів, такі як реверберація, ехо та інші.

10. Відстеження історії: додаток може відстежувати історію змін налаштувань та інших дій користувача.

Можливість додавання ефектів - це важлива функціональна характеристика розробленого клієнт-серверного додатку "Muzline", що дозволяє користувачам редагувати та покращувати якість звукових файлів.

Додаток надає користувачеві можливість додавати різні ефекти до звукових файлів, такі як реверберація, ехо, хорус, фланжер та інші. Кожен ефект може мати свої налаштування, які користувач може змінювати за своїм бажанням, щоб досягти потрібного звукового ефекту.

Завдяки цій функціональності користувач може зробити звуковий файл більш насиченим та цікавим, а також виправити недоліки в оригінальному записі. Наприклад, користувач може додати реверберацію для створення більшої просторової глибини звуку або ехо для створення ефекту повторюваності звуку.

Ця функціональність дозволяє додатково налаштовувати звукові ефекти на свій смак, забезпечуючи більш гнучкість та контроль над створенням музики та звукового дизайну.

11. Можливість віддаленого керування: додаток може дозволяти віддалене керування студійним обладнанням через Інтернет.

Ця функція може бути корисною в різних ситуаціях. Наприклад, якщо користувач налаштував звук під час роботи над проектом, а потім відвів студію на деякий час і хоче продовжити роботу над тим же проектом, то він

може легко повернутись до своїх попередніх налаштувань, щоб продовжити роботу з того місця, де він зупинився.

Крім того, історія змін налаштувань може допомогти користувачеві виявити помилки чи неправильні дії, які можуть призвести до проблем зі звуковою продукцією. Користувач може переглянути свої попередні налаштування і виявити помилки, які він зробив, або відновити попередні налаштування, якщо поточні налаштування не задовольняють його.

У додатку «Muzline» історія змін налаштувань та дій користувача зберігається в спеціальному журналі, до якого користувач може звернутись для перегляду попередніх налаштувань.

12. Мультиплатформенність: додаток може працювати на різних операційних системах, таких як Windows, macOS та Linux.

Мультиплатформенність - це можливість додатку працювати на різних операційних системах, не залежно від апаратної платформи, на якій він запущений. Для розробки мультиплатформних додатків зазвичай використовують кросплатформні технології, які дозволяють створювати код, що працює на різних платформах.

У випадку з додатком «Muzline», він розроблений таким чином, щоб працювати на різних операційних системах, включаючи Windows, macOS та Linux. Це означає, що користувачі можуть використовувати додаток на будь-якій платформі, яка їм зручна.

Розробники додатку «Muzline» могли використовувати кросплатформні фреймворки, такі як Qt або Electron, для створення додатку, який може працювати на різних платформах. Крім того, можна використовувати мови програмування, які підтримують кросплатформність, такі як Java або Python.

Мультиплатформність дозволяє додатку «Muzline» бути більш доступним для користувачів на різних платформах та дозволяє розробникам

зменшити час та витрати на розробку та підтримку різних версій додатку для різних платформ.

13. Безпека: додаток має вбудовані механізми безпеки, такі як автентифікація користувача, захист від вторгнень та інші.

Додаток «Muzline» має вбудовані механізми безпеки, які гарантують безпечну роботу зі студійним обладнанням. Найважливішим з них є механізми автентифікації користувача, які забезпечують захист від несанкціонованого доступу до системи.

Під час першого запуску додатку користувач має створити обліковий запис з унікальним ім'ям та паролем. Для забезпечення безпеки, пароль зберігається у вигляді хешу, що зменшує ймовірність його зламування.

Для захисту від вторгнень та зломів додаток використовує різні заходи безпеки, такі як шифрування даних та захист від SQL-ін'єкцій. Крім того, система моніторингу може виявляти підозрілу активність та блокувати доступ до додатку.

Додаток також має механізми резервного копіювання та відновлення даних, що дозволяє відновити налаштування та історію в разі виникнення проблем з системою.

В цілому, завдяки вбудованим механізмам безпеки, додаток «Muzline» забезпечує надійний захист даних та безпеку взаємодії зі студійним обладнанням.

14. Підтримка мережевих протоколів: додаток може працювати з різними мережевими протоколами, такими як TCP / IP, UDP та інші.

Підтримка мережевих протоколів в додатку дозволяє йому працювати з іншими пристроями та програмами, які використовують різні протоколи зв'язку. TCP / IP і UDP є двома з найпоширеніших мережевих протоколів, які використовуються для передачі даних в Інтернеті.

TCP / IP - це протокол передачі даних, який забезпечує надійну і точну передачу даних між різними комп'ютерами в мережі. В додатку можна використовувати TCP / IP для забезпечення стабільного зв'язку між клієнтом та сервером, які працюють на різних комп'ютерах.

UDP - це протокол без з'єднання, який використовується для швидкої передачі даних у реальному часі. В додатку можна використовувати UDP для передачі аудіо та відео даних в режимі реального часу.

Додаток може також підтримувати інші мережеві протоколи, такі як HTTP, FTP та інші, що дозволяє забезпечувати більш гнучкий інтерфейс та взаємодію з іншими програмами та пристроями в мережі.

15. Підтримка різних мов: додаток може працювати на різних мовах програмування та мовах інтерфейсу.

Підтримка різних мов - це важлива можливість, яка забезпечує ширшу аудиторію користувачів додатку. Коли додаток підтримує різні мови програмування, це дозволяє розробникам створювати функціональність за допомогою різних мов програмування та розширювати можливості додатку.

Підтримка різних мов інтерфейсу дає можливість користувачам вибирати мову, яку вони розуміють краще. Це дозволяє забезпечувати зручну інтерактивність користувача з додатком і зробити його більш доступним для різноманітної аудиторії користувачів.

Крім того, підтримка різних мов програмування і мов інтерфейсу може допомогти підтримувати додаток на різних операційних системах та пристроях.

Ці характеристики роблять додаток «Muzline» потужним інструментом для керування студійним обладнанням, що дозволяє користувачеві ефективно працювати зі звуком та створювати професійну аудіо продукцію.

3.2. Особливості впровадження клієнт-серверного додатку «Muzline» з реалізації студійного обладнання

Клієнт-серверний додаток "Muzline" з реалізацією студійного обладнання має декілька особливостей впровадження. Деякі з них включають наступне:

1. Вимоги до обладнання: Для користування "Muzline" потрібне студійне обладнання, таке як мікшерні пульт, аудіоінтерфейс, монітори та інше. Це може становити виклик для користувачів, які не мають такого обладнання або не мають досвіду в роботі з ним.

2. Налаштування додатку: "Muzline" потребує налаштування перед використанням. Клієнтська частина додатку повинна бути налаштована на студійному комп'ютері, а серверна частина - на сервері. Для цього може знадобитися підтримка технічних спеціалістів або розробників програмного забезпечення.

3. Завантаження файлів: "Muzline" може вимагати завантаження файлів з клієнтської частини на сервер для обробки. Це може бути повільним процесом, якщо використовується великий обсяг аудіо-або відеофайлів. Крім того, це може вимагати достатньої пропускної здатності мережі.

4. Безпека: "Muzline" має важливу функцію збереження та обробки аудіо-і відеоданих, тому безпека є критично важливою. Клієнт-серверний додаток повинен бути захищений від несанкціонованого доступу та атак з зовнішньої сторони. Для цього можуть бути використані різні заходи безпеки, такі як аутентифікація, авторизація та шифрування даних.

5. Підтримка користувачів: З моменту впровадження "Muzline" має підтримуватися технічними спеціалістами, які можуть відповісти на запитання користувачів та надавати допомогу в разі проблем з додатком. Оскільки "Muzline" є спеціалізованим додатком, знання про студійне

обладнання та аудіотехнології можуть бути необхідними для розуміння його функціональності.

6. Оновлення та підтримка: "Muzline" має бути оновлюваним та підтримуватися відповідними розробниками, щоб забезпечити безпеку та стабільну роботу додатку. Оновлення можуть включати нові функції, підвищення продуктивності та різноманітні покращення. Регулярна підтримка може забезпечити, що "Muzline" буде працювати на нових версіях операційної системи та обладнання.

Узагалі, впровадження клієнт-серверного додатку "Muzline" з реалізацією студійного обладнання може бути складним завданням, оскільки вимагає досвіду в роботі з обладнанням, налаштування додатку та знання про безпеку мережі та даних. Однак, якщо все виконано правильно, "Muzline" може стати потужним інструментом для роботи зі звуком та музикою.

3.3. Висновок до розділу 3

Клієнт-серверний додаток «Muzline» розроблений для керування студійним обладнанням і має наступні функціональні характеристики: забезпечення зв'язку між клієнтом та сервером; керування обладнанням; збереження налаштувань; можливість підключення різних пристроїв; зручний інтерфейс; можливість зберігання налаштувань за проектами; підтримка різних форматів звукових файлів; зручний пошук файлів; можливість додавання ефектів; відстеження історії; можливість віддаленого керування; мультиплатформенність; безпека; підтримка мережевих протоколів; підтримка різних мов.

Отже, впровадження клієнт-серверного додатку "Muzline" з реалізацією студійного обладнання вимагає деяких особливостей та уваги до деталей. Необхідно правильно налаштувати мережу, забезпечити безпеку даних та

мережі, встановити та налаштувати сервер та клієнтські додатки, забезпечити надійність та швидкість роботи.

Також, необхідно мати достатні знання про студійне обладнання та аудіотехнології, щоб зрозуміти функціональність додатку та забезпечити належну роботу.

Регулярне оновлення та підтримка додатку можуть забезпечити безпеку та стабільну роботу додатку на нових версіях операційної системи та обладнання.

Узагалі, впровадження клієнт-серверного додатку "Muzline" може бути складним завданням, але з правильним налаштуванням та підтримкою, він може стати потужним інструментом для роботи зі звуком та музикою.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Завдяки використанню цифрових технологій, сучасні студійні обладнання стають все більш компактними та зручними у використанні, що забезпечує підвищення продуктивності та ефективності в роботі. Розвиток онлайн-стрімінгу призвів до збільшення попиту на студійне обладнання для відеозйомки та живого стрімінгу. Ринок студійного обладнання конкурентний, що призводить до зниження цін на деякі продукти та покращення якості інших. Сучасні студійні обладнання мають багато функцій, які забезпечують глибоку обробку та редагування аудіо- та відеоматеріалів, що забезпечує більшу гнучкість та ефективність в роботі. Зростання важливості якості та розширення відео призвело до збільшення попиту на студійне обладнання для зйомки та обробки відео в високій якості в останні кілька років.

Узагальнюючи, розвиток студійного обладнання залежить від прогресу технологій та відповідного попиту на ринку. Якщо розвиток продовжиться на тому ж рівні, можна очікувати появу нових технологій та продуктів, які забезпечать більшу продуктивність та ефективність в студійній роботі.

Основні кроки в розробці та впровадженні додатку можуть включати наступне: визначення вимог до додатку, вибір необхідних технологій, розробка серверної та клієнтської частини, тестування та налагодження, впровадження, підтримка та здійснення доповнень.

Розробка та впровадження клієнт-серверного додатку для студійного обладнання можуть бути складними та часоємними процесами. Однак, якщо у вас є необхідний досвід та знання, а також ви використовуєте ефективні

інструменти, то ви зможете успішно розробити необхідний додаток, який відповідатиме всім вимогам та задовольнить потреби користувачів.

Результати роботи клієнт-серверного додатку для студійного обладнання залежать від декількох факторів, таких як якість програмного забезпечення, кількість одночасних з'єднань, пропускна здатність мережі, конфігурація обладнання, рівень навичок користувачів, місце розміщення сервера, рівень безпеки, обсяг оброблюваних даних, використовувані технології та потужність обладнання.

Враховуючи потребу у високій якості відео та зростаючий попит на студійне обладнання для відеозйомки та живого стрімінгу, використання технологій, що забезпечують глибоку обробку та редагування аудіо- та відеоматеріалів, було надзвичайно важливим. Для цього в додатку було реалізовано функціональні можливості, що дозволяють глибоко обробляти та редагувати аудіо- та відеофайли, забезпечуючи більшу гнучкість та ефективність в роботі.

У процесі розробки та впровадження додатку було враховано різні фактори, такі як якість програмного забезпечення, пропускна здатність мережі, конфігурація обладнання та рівень навичок користувачів. Крім того, з метою забезпечення безпеки даних та масштабованості було використано хмарні сервіси, такі як Amazon Web Services (AWS) або Microsoft Azure, для розгортання та підтримки додатку.

Висновки підкреслюють, що успішна розробка клієнт-серверного додатку для студійного обладнання вимагає досвіду та знань у використанні відповідних технологій. Конкретизація вимог, використання ефективних інструментів та платформ, а також дотримання кращих практик розробки дозволять досягти бажаних результатів у розробці та впровадженні додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Anderson R. Security Engineering: A Guide to Building Dependable Distributed Systems 3rd Edition. Wiley; 3rd edition (December 22, 2020). 1232 p.
2. Audio engineering. URL: https://www.reddit.com/r/audioengineering/
3. Coronel C., Morris S. Database Systems: Design, Implementation, & Management 13th Edition. Cengage Learning; 13th edition (January 1, 2018). 816 p.
4. Gearspace. URL: https://gearspace.com/
5. Karner C., Falk J., Nguyen H.Q. Testing Computer Software, 2nd Edition. Wiley; 2nd edition (April 12, 1999). 480 p.
6. Krug S. Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability (3rd Edition). URL: https://www.amazon.com/Dont-Make-Think-Revisited-Usability/dp/0321965515
7. MongoDB Atlas. Fully managed MongoDB in the cloud. URL: https://www.mongodb.com/cloud/atlas/
8. Saternos C. Client-Server Web Apps with JavaScript and Java: Rich, Scalable, and RESTful. O'Reilly Media; 1st edition (April 29, 2014). 260 p.
9. Tanenbaum A.S., van Steen M. Distributed Systems: Principles and Paradigms 2nd Edition. CreateSpace Independent Publishing Platform; 2nd edition (February 26, 2016). 702 p.

10. WebSocket vs REST: Key differences and which to use. 2022. URL: https://ably.com/topic/websocket-vs-rest

11. Gary E. Economics: Today and Tomorrow's / E. Gary, Ph.D. Clayton. – Glencoe, McGraw-Hill, 2008. – 1050 p.

12. Пристрої введення та виведення даних [Електронний ресурс]. - Режим доступу: https://narodna-osvita.com.ua/5353-pristroyi-vvedennya-ta-vivedennya-danih.html

13. Апаратне забезпечення персонального комп'ютера [Електронний ресурс]. - Режим доступу: https://kafinfo.org.ua/files/Informatyka_10_11/Glava_2_5.pdf

ДОДАТКИ

ДОДАТОК А

Код класу CartController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;
using Muzline.Models;
using Newtonsoft.Json;
using Microsoft.AspNetCore.Http.Extensions;

namespace Muzline.Controllers
{
   public class CartController : Controller
   {
      private readonly DataContext _context;

      public CartController(DataContext context)
      {
         _context = context;
      }
      public IActionResult Index()
      {
         var cart = new Cart();
         if (HttpContext.Session.GetString("Cart") != null)
         {
            var jsonCart = HttpContext.Session.GetString("Cart");
            cart = JsonConvert.DeserializeObject<Cart>(jsonCart);
         }


         return View(cart);
      }

      [HttpPost]
      public IActionResult AddToCart(int productId, int quantity)
      {
         // отримуємо товар за його Id
         var product = _context.Products.FirstOrDefault(p => p.Id == productId);

         if (product != null)
         {
            var cart = new Cart();
            // отримуємо поточну корзину
            var cartString = HttpContext.Session.GetString("Cart");
            if (cartString != null)
            {
               cart = JsonConvert.DeserializeObject<Cart>(cartString);
               cart.AddItem(product.Id, product.Name, product.Price, quantity);
            }
            else
            {
               cart = new Cart();
               cart.AddItem(product.Id, product.Name, product.Price, quantity);
            }
            // зберігаємо корзину в сесії
            HttpContext.Session.SetString("Cart", JsonConvert.SerializeObject(cart));

         }

         return RedirectToAction("Index");
      }
   }
}
```

Кяд класу HomeController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;
using Muzline.Models;
using System.Diagnostics;

namespace Muzline.Controllers
{
    public class HomeController : Controller
    {
        private readonly DataContext _context;

        public HomeController(DataContext context)
        {
            _context = context;
        }

        public IActionResult Index()
        {
            var popularProducts =
_context.PopularProducts.Include(p=>p.Product).ThenInclude(x=>x.ProductImages).Select(x=>x.Product).ToList();
            //var result = popularProducts.Include(p => p.ProductImages).ToList();
            return View(popularProducts);
        }

        public IActionResult Privacy()
        {
            return View();
        }

        [ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]
        public IActionResult Error()
        {
            return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });
        }
    }
}
```

Код класу OrderController

```
using Microsoft.AspNetCore.Mvc;
using Muzline.Data;
using Muzline.Models;
using Muzline.ViewModels;
using Newtonsoft.Json;

namespace Muzline.Controllers
{
    public class OrderController : Controller
    {
        private readonly DataContext _context;

        public OrderController(DataContext context)
        {
            _context = context;
        }
        public IActionResult Index()
        {
            return View();
        }

        public IActionResult Checkout()
        {
            return View();
        }
        [HttpPost]
        public IActionResult Checkout(OrderViewModel order)
        {
            var cartString = HttpContext.Session.GetString("Cart");
            var cart = new Cart();
            if (cartString != null)
            {
                cart = JsonConvert.DeserializeObject<Cart>(cartString);

            }
            else
            {
                ModelState.AddModelError("", "Sorry, your cart is empty!");
            }

            if (ModelState.IsValid)
            {
                var newOrder = new Order()
                {
                    Name = order.Name,
                    Address = order.Address,
                    Email = order.Email,
                    Phone = order.Phone,
                    Comment = order.Comment,
                    OrderDate = DateTime.Now,
                    Total = cart.GetTotal(),

                };
                _context.Orders.Add(newOrder);
                _context.SaveChanges();

                foreach (var item in cart.Items)
                {
                    var orderItem = new OrderItem()
                    {
                        OrderId = newOrder.OrderId,
                        ProductId = item.ProductId,
```

```
                Quantity = item.Quantity

            };
            _context.OrderItems.Add(orderItem);
        }
        _context.SaveChanges();
        HttpContext.Session.Remove("Cart");
        return RedirectToAction("Index", "Home");
    }
    return View(order);


    }
  }
}
```

Код класу ProductController

```
using Microsoft.AspNetCore.Mvc;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;
using Muzline.Models;
using Muzline.ViewModels;

namespace Muzline.Controllers
{
    public class ProductController : Controller
    {
        private readonly DataContext _context;

        public ProductController(DataContext context)
        {
            _context = context;
        }

        public IActionResult Index(string brand, string searchString, int[] brandIds, int[] categoryIds, int page = 1, string sort
= "name", decimal minPrice = 0, decimal maxPrice = 100000)
        {
            int pageSize = 10;
                                var categories = _context.Categories.ToList();
                                var products = _context.Products.Include(p => p.Category).Include(p =>
p.ProductImages).AsQueryable();
                                //IQueryable<Product> products = _context.Products;
                                if (!string.IsNullOrEmpty(searchString))
                                {
                                        products = products.Where(p => p.Name.Contains(searchString) ||
p.Description.Contains(searchString));
                                }
                                if (categoryIds != null && categoryIds.Length > 0)
                                {
                                        products = products.Where(p => categoryIds.Contains(p.CategoryId));
                                }
            if (brandIds != null && brandIds.Length > 0)
            {
                products = products.Where(p => brandIds.Contains(p.BrandId));
            }
            switch (sort)
            {
                case "name":
                    products = products.OrderBy(p => p.Name);
                    break;
                case "price":
                    products = products.OrderBy(p => p.Price);
                    break;
                // інші варіанти сортування
                default:
                    products = products.OrderBy(p => p.Id);
                    break;
            }
            if(minPrice > 0 && maxPrice > 0)
            {
                products = products.Where(p => p.Price >= minPrice && p.Price <=maxPrice);
            }

            var viewModel = new ProductsViewModel
            {
                Products = products.Skip((page - 1) * pageSize).Take(pageSize),
                PagingInfo = new PagingInfo
                {
                    CurrentPage = page,
```

```
                ItemsPerPage = pageSize,
                TotalItems = products.Count()
            },
            CurrentSortOrder = sort,
            CategoryListViewModel = new CategoryListViewModel
            {
                Categories = categories,
                SelectedCategories = categoryIds
                                        },
            Brands = _context.Brands.ToList()

        };
        //var t = _context.ProductImages.ToList();
        return View(viewModel);
    }
    public async Task<IActionResult> Details(int id)
    {
        var product = await _context.Products
            .Include(p => p.Category)
            .Include(p => p.ProductImages)
            .Include(p => p.ProductSpecifications)
            .FirstOrDefaultAsync(p => p.Id == id);

        if (product == null)
        {
            return NotFound();
        }
                                ViewBag.Reviews = _context.ProductReviews.ToList();
        return View(product);
    }
    [HttpPost]
    public IActionResult AddReview(ProductReview review)
    {

        var product = _context.Products.SingleOrDefault(p => p.Id == review.ProductId);
        if (product == null)
        {
            return NotFound();
        }

        review.Date= DateTime.Now;
        review.IsApproved = true;

        _context.ProductReviews.Add(review);
        _context.SaveChanges();

        return RedirectToAction("Details", "Product", new { id = review.ProductId });
    }
  }
}
```

Код класу DataContext

```
using Microsoft.EntityFrameworkCore;
using Muzline.Models;

namespace Muzline.Data
{
  public class DataContext : DbContext
  {
    public DbSet<Product> Products { get; set; }
    public DbSet<ProductImage> ProductImages { get; set; }
    public DbSet<Category> Categories { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<OrderItem> OrderItems { get; set; }
    public DbSet<PopularProduct> PopularProducts { get; set; }
    public DbSet<Brand> Brands { get; set; }
    public DbSet<ProductSpecification> ProductSpecifications { get; set; }

    public DbSet<ProductReview> ProductReviews { get; set; }
    //public DbSet<Cart> Carts { get; set; }
    public DataContext(DbContextOptions<DataContext> options)
      : base(options)
    {
      Database.EnsureCreated();
    }

  }
}
```

Код класу Brand

```
using Muzline.Models;

public class Brand
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Description { get; set; }
    public ICollection<Product> Products { get; set; }
}
```

Код класу Cart

```
namespace Muzline.Models
{
    public class Cart
    {
        public List<CartItem> items = new List<CartItem>();

        public IReadOnlyCollection<CartItem> Items => items.AsReadOnly();

        public void AddItem(int productId, string productName, decimal price, int quantity)
        {
            var item = items.FirstOrDefault(i => i.ProductId == productId);

            if (item != null)
            {
                item.Quantity += quantity;
            }
            else
            {
                items.Add(new CartItem
                {
                    ProductId = productId,
                    ProductName = productName,
                    Price = price,
                    Quantity = quantity
                });
            }
        }

        public void RemoveItem(int productId)
        {
            items.RemoveAll(i => i.ProductId == productId);
        }

        public decimal GetTotal()
        {
            return items.Sum(i => i.Price * i.Quantity);
        }

        public void Clear()
        {
            items.Clear();
        }
    }
}
```

Код класу CartItem

```
namespace Muzline.Models
{
    public class CartItem
    {
        public int ProductId { get; set; }
        public string ProductName { get; set; }
        public decimal Price { get; set; }
        public int Quantity { get; set; }
    }
}
```

Код класу Category

```
namespace Muzline.Models
{
    public class Category
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public ICollection<Product> Products { get; set; }
    }
}
```

Код класу ErrorViewModel

```
namespace Muzline.Models
{
    public class ErrorViewModel
    {
        public string? RequestId { get; set; }

        public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);
    }
}
```

Код класу Order

```
using System.ComponentModel.DataAnnotations;

namespace Muzline.Models
{
    public class Order
    {
        [Key]
        public int OrderId { get; set; }

        [Required(ErrorMessage = "Не вказано ім'я клієнта")]
        [StringLength(50, ErrorMessage = "Довжина імені не повинна перевищувати 50 символів")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Не вказано адресу")]
        [StringLength(100, ErrorMessage = "Довжина адреси не повинна перевищувати 100 символів")]
        public string Address { get; set; }

        [Required(ErrorMessage = "Не вказано електронну пошту")]
        [EmailAddress(ErrorMessage = "Некоректна електронна пошта")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Не вказано номер телефону")]
        [Phone(ErrorMessage = "Некоректний номер телефону")]
        public string Phone { get; set; }

        [StringLength(500, ErrorMessage = "Довжина коментаря не повинна перевищувати 500 символів")]
        public string Comment { get; set; }

        public DateTime OrderDate { get; set; }

        [Required(ErrorMessage = "Не вказано суму замовлення")]
        public decimal Total { get; set; }

        public string? UserId { get; set; }

        public virtual ICollection<OrderItem> OrderItems { get; set; }
    }
}
```

Код класу OrderItem

```
using System.ComponentModel.DataAnnotations;

namespace Muzline.Models
{
    public class OrderItem
    {
        [Key]
        public int OrderItemId { get; set; }

        [Required(ErrorMessage = "Не вказано Id товару")]
        public int ProductId { get; set; }
        [Required(ErrorMessage = "Не вказано кількість товару")]
        [Range(1, int.MaxValue, ErrorMessage = "Кількість товару повинна бути більшою за 0")]
        public int Quantity { get; set; }

        [StringLength(200, ErrorMessage = "Довжина не повинна перевищувати 200 символів")]
        public string? Note { get; set; }

        public int OrderId { get; set; }

        public virtual Order Order { get; set; }
        public virtual Product Product { get; set; }
    }
}
```

Код класу PagingInfo

```
namespace Muzline.Models
{
    public class PagingInfo
    {
        public int TotalItems { get; set; }
        public int ItemsPerPage { get; set; }
        public int CurrentPage { get; set; }
        public int TotalPages => (int)Math.Ceiling((decimal)TotalItems / ItemsPerPage);
    }
}
```

Код класу PopularProduct

```
namespace Muzline.Models
{
    public class PopularProduct
    {
        public int Id { get; set; }
        public string? ImgUrl { get; set; }
        public int ProductId { get; set; }
        public Product Product { get; set; }
    }
}
```

Код класу Product

```
using System.Linq.Expressions;
namespace Muzline.Models
{
  public class Product
    {
      public int Id { get; set; }
      public string Name { get; set; }
      public string Description { get; set; }

      public decimal Price { get; set; }
      public int BrandId { get; set; }
      public Brand Brand { get; set; }
      public int CategoryId { get; set; }
      public Category Category { get; set; }
      public ICollection<ProductImage> ProductImages { get; set; }
      public ICollection<ProductSpecification> ProductSpecifications { get; set; }

                #region Filters
                public static Expression<Func<Product, bool>> FilterByCategory(int categoryId)
                {
                        return p => p.CategoryId == categoryId;
                }
                public static Expression<Func<Product, bool>> FilterByPrice(decimal minPrice, decimal maxPrice)
                {
                        return p => p.Price >= minPrice && p.Price <= maxPrice;
                }
                #endregion
        }
}
```

Код класу ProductImage

```
namespace Muzline.Models
{
    public class ProductImage
    {
        public int Id { get; set; }
        public string FileName { get; set; }
        public byte[] ImageData { get; set; }
        public int ProductId { get; set; }
        public Product Product { get; set; }
    }
}
```

Код класу ProductReview

```
namespace Muzline.Models
{
    public class ProductReview
    {
        public int Id { get; set; }
        public int ProductId { get; set; }
        public string UserName { get; set; }
        public string Comment { get; set; }
        public DateTime Date { get; set; }
        public int Rating { get; set; }
        public bool IsApproved { get; set; }

        public virtual Product Product { get; set; }
    }
}
```

Код класу ProductSpecification

```
namespace Muzline.Models
{
    public class ProductSpecification
    {
        public int Id { get; set; }
        public int ProductId { get; set; }
        public string Name { get; set; }
        public string Value { get; set; }
        public Product Product { get; set; }
    }
}
```

Код класу CategoryListViewModel

```
using Muzline.Models;

namespace Muzline.ViewModels
{
	public class CategoryListViewModel
	{
		public IEnumerable<Category> Categories { get; set; }
		public int[] SelectedCategories { get; set; }
	}
}
```

Код класу OrderViewModel

```
using System.ComponentModel.DataAnnotations;

namespace Muzline.ViewModels
{
    public class OrderViewModel
    {

        [Required(ErrorMessage = "Не вказано ім'я клієнта")]
        [StringLength(50, ErrorMessage = "Довжина імені не повинна перевищувати 50 символів")]
        public string Name { get; set; }

        [Required(ErrorMessage = "Не вказано адресу")]
        [StringLength(100, ErrorMessage = "Довжина адреси не повинна перевищувати 100 символів")]
        public string Address { get; set; }
        [Required(ErrorMessage = "Не вказано електронну пошту")]
        [EmailAddress(ErrorMessage = "Некоректна електронна пошта")]
        public string Email { get; set; }

        [Required(ErrorMessage = "Не вказано номер телефону")]
        [Phone(ErrorMessage = "Некоректний номер телефону")]
        public string Phone { get; set; }

        [StringLength(500, ErrorMessage = "Довжина коментаря не повинна перевищувати 500 символів")]
        public string Comment { get; set; }

    }
}
```

Код класу ProductsViewModel

```
namespace Muzline.ViewModels
{
    public class ProductsViewModel
    {
        public IEnumerable<Product> Products { get; set; }
        public PagingInfo PagingInfo { get; set; }
        public string CurrentSortOrder { get; set; }
        public string NameSortParm { get; set; }
        public string PriceSortParm { get; set; }
        public string CategorySortParm { get; set; }

        public string SearchString { get; set; }

        public List<Brand> Brands { get; set; }
        public CategoryListViewModel CategoryListViewModel { get; set; }

    }
}
```

Код HTML розмітки

```
@model Cart
@{
   ViewData["Title"] = "Корзина";
}
<!--================Cart Area ================-->
<section class="cart_area">
   <div class="container">
      <div class="cart_inner">
         <div class="table-responsive">
            <table class="table">
               <thead>
                  <tr>
                     <th scope="col">Product</th>
                     <th scope="col">Price</th>
                     <th scope="col">Quantity</th>
                     <th scope="col">Total</th>
                  </tr>
               </thead>
               <tbody>

                  @foreach(var cartItem in Model.Items){
                  <tr>
                     <td>
                        <div class="media">
                           <div class="d-flex">

                           </div>
                           <div class="media-body">
                              <p>@cartItem.ProductName</p>
                           </div>
                        </div>
                     </td>
                     <td>
                        <h5>@cartItem.Price₴</h5>
                     </td>
                     <td>
                        <div class="product_count">
                           <input type="text" name="qty" id="sst" maxlength="12" value="@cartItem.Quantity"
title="Quantity:"
                              class="input-text qty">
                           <button onclick="var result = document.getElementById('sst'); var sst = result.value; if( !isNaN(
sst )) result.value++;return false;"
                              class="increase items-count" type="button">
                              <i class="lnr lnr-chevron-up"></i>
                           </button>
                           <button onclick="var result = document.getElementById('sst'); var sst = result.value; if( !isNaN(
sst ) &amp;&amp; sst > 0 ) result.value--;return false;"
                              class="reduced items-count" type="button">
                              <i class="lnr lnr-chevron-down"></i>
                           </button>
                        </div>
                     </td>
                     <td>
                        <h5>@(cartItem.Quantity * cartItem.Price)₴</h5>
                     </td>
                  </tr>
                  }


               <tr class="bottom_button">
                  <td>
                     <a class="button" href="#">Update Cart</a>
```

```
            </td>
            <td>
            </td>
            <td>
            </td>
            <td>
              <div class="cupon_text d-flex align-items-center">
                <input type="text" placeholder="Coupon Code">
                <a class="primary-btn" href="#">Apply</a>
                <a class="button" href="#">Have a Coupon?</a>
              </div>
            </td>
          </tr>
          <tr>
            <td>
            </td>
            <td>
            </td>
            <td>
              <h5>Subtotal</h5>
            </td>
            <td>
              <h5>@Model.GetTotal()₴</h5>
            </td>
          </tr>
          <tr class="shipping_area">
            <td class="d-none d-md-block">
            </td>
            <td>
            </td>
            <td>
              <h5>Shipping</h5>
            </td>
            <td>
              <div class="shipping_box">
                <form action="Order/Checkout">
                  <div>
                    <label for="Name">Ім'я клієнта:</label>
                    <input type="text" id="Name" name="Name" required maxlength="50">
                  </div>
                  <div>
                    <label for="Address">Адреса:</label>
                    <input type="text" id="Address" name="Address" required maxlength="100">
                  </div>
                  <div>
                    <label for="Email">Електронна пошта:</label>
                    <input type="email" id="Email" name="Email" required>
                  </div>
                  <div>
                    <label for="Phone">Номер телефону:</label>
                    <input type="tel" id="Phone" name="Phone" required>
                  </div>
                  <div>
                    <label for="Comment">Коментар:</label>
                    <input id="Comment" name="Comment" maxlength="500"></input>
                  </div>
                  <button class="gray_btn" type="submit">Замовити</button>
                </form>

              </div>
            </td>
          </tr>
          <tr class="out_button_area">
            <td class="d-none-l">
```

```
                    </td>
                    <td class="">
                    </td>
                    <td>
                    </td>
                    <td>
                      <div class="checkout_btn_inner d-flex align-items-center">
                        <a class="gray_btn" href="#">Continue Shopping</a>
                        <a class="primary-btn ml-2" href="#">Proceed to checkout</a>
                      </div>
                    </td>
                  </tr>
                </tbody>
              </table>
            </div>
          </div>
        </div>
      </section>
      <!--================End Cart Area =================-->

      @using Muzline.Models;
      @model List<Product>;
      @{
          ViewData["Title"] = "Головна сторінка";
      }

      <main class="site-main">

        <!--================ Hero banner start =================-->
        <section class="hero-banner">
          <div class="container">
            <div class="row no-gutters align-items-center pt-60px">
              <div class="col-5 d-none d-sm-block">
                <div class="hero-banner__img">
                  <img class="img-fluid" src="~/img/home/hero-banner.png" alt="">
                </div>
              </div>
              <div class="col-sm-7 col-lg-6 offset-lg-1 pl-4 pl-md-5 pl-lg-0">
                <div class="hero-banner__content">
                  @*<h4>Shop is fun</h4>*@
                  <h1>Насолоджуйтесь музичним світом</h1>
                  <p>У нашому магазині музичної техніки ви знайдете все необхідне для створення неповторної
      атмосфери і насолоди музикою - від професійного обладнання до дрібних аксесуарів.</p>
                  <a class="button button-hero" href="/Product">Перегляньте зараз</a>
                </div>
              </div>
            </div>
          </div>
        </section>
        <!--================ Hero banner start =================-->
        <!--================ Hero Carousel start =================-->
        <section class="section-margin mt-0">
          <div class="owl-carousel owl-theme hero-carousel">
            <div class="hero-carousel__slide">
              <img src="~/img/home/home1.png" alt="" class="img-fluid">
              <a href="/Product/Deteils/2" class="hero-carousel__slideOverlay">
                <h3>Професійний мікрофон</h3>
                <p>Вокальні мікрофони</p>
              </a>
            </div>
            <div class="hero-carousel__slide">
              <img src="~/img/home/home2.jpg" alt="" class="img-fluid">
              <a href="#" class="hero-carousel__slideOverlay">
```

```
                <h3>Дротові навушники</h3>
                <p>навушники</p>
              </a>
            </div>
            <div class="hero-carousel__slide">
              <img src="~/img/home/home3.jpg" alt="" class="img-fluid">
              <a href="#" class="hero-carousel__slideOverlay">
                <h3>Студійний динамік</h3>
                <p>Колонки</p>
              </a>
            </div>
          </div>
        </section>
        <!--================= Hero Carousel end =================-->
        <!-- ================ trending product section start ================= -->
        <section class="section-margin calc-60px">
          <div class="container">
            <div class="section-intro pb-60px">
              <p></p>
              <h2>Популярні <span class="section-intro__style">Товари</span></h2>
            </div>
            <div class="row">
              @foreach(var product in Model)
              {
                <div class="col-md-6 col-lg-4 col-xl-3">
                  <div class="card text-center card-product">
                    <div class="card-product__img">
                      @if (product.ProductImages != null && product.ProductImages.Count > 0)
                        {
                        <img class="card-img"
src="@Convert.ToBase64String(product.ProductImages.First().ImageData)" />
                        }
                      @*<img class="card-img"
src="@Convert.ToBase64String(product.Product.ProductImages.First().ImageData)"  alt="">*@
                      <ul class="card-product__imgOverlay">
                        <li><button><i class="ti-search"></i></button></li>
                        <li><button><i class="ti-shopping-cart"></i></button></li>
                        <li><button><i class="ti-heart"></i></button></li>
                      </ul>
                    </div>
                    <div class="card-body">
                      <p>@product.Category</p>
                      <h4 class="card-product__title"><a href="/Product/Details/@product.Id">@product.Name</a></h4>
                      <p class="card-product__price">@product.Price</p>
                    </div>
                  </div>
                </div>
              }

            </div>
          </div>
        </section>
        <!-- ================ trending product section end ================= -->
        <!-- ================ offer section start ================= -->
        <section class="offer" id="parallax-1" data-anchor-target="#parallax-1" data-300-top="background-position: 20px
30px" data-top-bottom="background-position: 0 20px">
          <div class="container">
            <div class="row">
              <div class="col-xl-5">
                <div class="offer__content text-center">
                  <h3>Знижки до 50%</h3>
                  <h4>Весняний розпродаж</h4>
                  @*<p>Him she'd let them sixth saw light</p>*@
                  <a class="button button--active mt-3 mt-xl-4" href="/Product">Переглянути</a>
```

```
            </div>
          </div>
        </div>
      </div>
    </section>
    <!-- ================ offer section end ================ -->


    <!-- ================ Subscribe section start ================ -->
    <section class="subscribe-position">
      <div class="container">
        <div class="subscribe text-center">
          <h3 class="subscribe__title">Отримайте 10% знижку на нову колекцію!</h3>
          <p>Підпишіться на нашу розсилку, щоб отримати дисконтний код та бути у курсі нових надходжень</p>
          <div id="mc_embed_signup">
            <form target="_blank" action="https://spondonit.us12.list-
manage.com/subscribe/post?u=1462626880ade1ac87bd9c93a&amp;id=92a4423d01" method="get" class="subscribe-
form form-inline mt-5 pt-1">
              <div class="form-group ml-sm-auto">
                <input class="form-control mb-1" type="email" name="EMAIL" placeholder="Enter your email"
onfocus="this.placeholder = '" onblur="this.placeholder = 'Your Email Address '">
                <div class="info"></div>
              </div>
              <button class="button button-subscribe mr-auto mb-1" type="submit">Subscribe Now</button>
              <div style="position: absolute; left: -5000px;">
                <input name="b_36c4fd991d266f23781ded980_aefe40901a" tabindex="-1" value="" type="text">
              </div>

            </form>
          </div>

        </div>
      </div>
    </section>
    <!-- ================ Subscribe section end ================ -->


</main>

@using Muzline.Models;
@model Product
@{
    ViewData["Title"] = Model.Name;
}

<!--================Single Product Area =================-->
<div class="product_image_area">
        <div class="container">
                <div class="row s_product_inner">
                        <div class="col-lg-6">

                                <div class="owl-carousel owl-theme s_Product_carousel">
                                        @foreach(var item in Model.ProductImages)
                                        {
                                                <div class="single-prd-item">
                                                        <img class="img-fluid"
src="@Convert.ToBase64String(item.ImageData)" alt="">
                                                </div>

                                        }

                        </div>
                        <div class="col-lg-5 offset-lg-1">
                                <div class="s_product_text">
```

```
                                    <h3>@Model.Name</h3>
                                    <h2>@Model.Price ₴</h2>
                                    <ul class="list">

                                        <li><a class="active" href="#"><span>Категорія<span>:
@Model.Category.Name</a></li>

                                        <li><a href="#"><span>Наявність</span> : В
наявності</a></li>
                                    </ul>

                                    <div class="product_count">
                                        <form id="myForm" asp-controller="Cart" asp-
action="AddToCart" method="post">

                                        <label for="qty">Кількість:</label>

                                        <input style="visibility:hidden" value="@Model.Id"
name="productId"/>
                                        <input type="text" name="quantity" id="sst" size="2"
maxlength="12" value="1" title="Кількість:" class="input-text qty">


                                                <a class="button primary-btn"
onclick="document.getElementById('myForm').submit();">Додати до корзини</a>
                                        </form>
                                    </div>

                                </div>
                            </div>
                        </div>
                    </div>
                </div>
                <!--================End Single Product Area =================-->
                <!--================Product Description Area =================-->
                <section class="product_description_area">
                    <div class="container">
                        <ul class="nav nav-tabs" id="myTab" role="tablist">
                            <li class="nav-item">
                                <a class="nav-link" id="home-tab" data-toggle="tab" href="#home" role="tab" aria-
controls="home" aria-selected="true">Опис товару</a>
                            </li>
                            <li class="nav-item">
                                <a class="nav-link" id="profile-tab" data-toggle="tab" href="#profile" role="tab"
aria-controls="profile"
                                    aria-selected="false">Характеристики</a>
                            </li>

                            <li class="nav-item">
                                <a class="nav-link active" id="review-tab" data-toggle="tab" href="#review"
role="tab" aria-controls="review"
                                    aria-selected="false">Відгуки</a>
                            </li>
                        </ul>
                        <div class="tab-content" id="myTabContent">
                            <div class="tab-pane fade" id="home" role="tabpanel" aria-labelledby="home-tab">
                                <p>
                                    @Model.Description
                                </p>

                            </div>
                            <div class="tab-pane fade" id="profile" role="tabpanel" aria-labelledby="profile-tab">
                                <div class="table-responsive">
                                    <table class="table">
                                        <tbody>
                                            @foreach(var item in Model.ProductSpecifications)
```

```
                                    {
                                        <tr>
                                            <td>
                                                <h5>@item.Name</h5>
                                            </td>
                                            <td>
                                                <h5>@item.Value</h5>
                                            </td>
                                        </tr>
                                    }

                                </tbody>
                            </table>
                        </div>
                    </div>
                    <div class="tab-pane fade" id="contact" role="tabpanel" aria-labelledby="contact-tab">
                        <div class="row">
                            <div class="col-lg-6">
                                <div class="comment_list">
                                    <div class="review_item">
                                        <div class="media">
                                            <div class="d-flex">
                                                <img
src="~/img/product/review-1.png" alt="">
                                            </div>
                                            <div class="media-body">
                                                <h4>Blake Ruiz</h4>
                                                <h5>12th Feb, 2018 at 05:56
pm</h5>
                                                <a class="reply_btn"
href="#">Reply</a>
                                            </div>
                                        </div>
                                        <p>
                                            Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
                                            dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
                                            commodo
                                        </p>
                                    </div>
                                    <div class="review_item reply">
                                        <div class="media">
                                            <div class="d-flex">
                                                <img
src="~/img/product/review-2.png" alt="">
                                            </div>
                                            <div class="media-body">
                                                <h4>Blake Ruiz</h4>
                                                <h5>12th Feb, 2018 at 05:56
pm</h5>
                                                <a class="reply_btn"
href="#">Reply</a>
                                            </div>
                                        </div>
                                        <p>
                                            Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
                                            dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
                                            commodo
                                        </p>
                                    </div>
                                    <div class="review_item">
```

```
                                                <div class="media">
                                                    <div class="d-flex">
                                                        <img
src="~/img/product/review-3.png" alt="">
                                                    </div>
                                                    <div class="media-body">
                                                        <h4>Blake Ruiz</h4>
                                                        <h5>12th Feb, 2018 at 05:56
pm</h5>
                                                        <a class="reply_btn"
href="#">Reply</a>
                                                    </div>
                                                </div>
                                                <p>
                                                    Lorem ipsum dolor sit amet,
consectetur adipisicing elit, sed do eiusmod tempor incididunt ut labore et
                                                    dolore magna aliqua. Ut enim ad
minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea
                                                    commodo
                                                </p>
                                            </div>
                                        </div>
                                    </div>
                                    <div class="col-lg-6">
                                        <div class="review_box">
                                            <h4>Post a comment</h4>
                                            <form class="row contact_form"
action="contact_process.php" method="post" id="contactForm" novalidate="novalidate">
                                                <div class="col-md-12">
                                                    <div class="form-group">
                                                        <input type="text"
class="form-control" id="name" name="name" placeholder="Your Full name">
                                                    </div>
                                                </div>
                                                <div class="col-md-12">
                                                    <div class="form-group">
                                                        <input type="email"
class="form-control" id="email" name="email" placeholder="Email Address">
                                                    </div>
                                                </div>
                                                <div class="col-md-12">
                                                    <div class="form-group">
                                                        <input type="text"
class="form-control" id="number" name="number" placeholder="Phone Number">
                                                    </div>
                                                </div>
                                                <div class="col-md-12">
                                                    <div class="form-group">
                                                        <textarea class="form-
control" name="message" id="message" rows="1" placeholder="Message"></textarea>
                                                    </div>
                                                </div>
                                                <div class="col-md-12 text-right">
                                                    <button type="submit"
value="submit" class="btn primary-btn">Submit Now</button>
                                                </div>
                                            </form>
                                        </div>
                                    </div>
                                </div>
                            </div>
                            <div class="tab-pane fade show active" id="review" role="tabpanel" aria-labelledby="review-
tab">
                                <div class="row">
```

```
                    <div class="col-lg-6">
                        <div class="row total_rate">
                            <div class="col-6">
                                <div class="box_total">
                                    <h5>Середня оцінка</h5>
                                    @{
                                        var total = 0;
                                        if (ViewBag.Reviews != null
&& ViewBag.Reviews.Count > 0)
                                        {
                                            foreach (var item in
ViewBag.Reviews)
                                            {
                                                total +=
item.Rating;
                                            }
                                            var average = total /
ViewBag.Reviews.Count;

                                        <h4>@average</h4>
                                        }

                                        <h6>(@ViewBag.Reviews.Count
кількість відгуків)</h6>
                                    }
                                </div>
                            </div>
                            <div class="col-6">
                                <div class="rating_list">

                                </div>
                            </div>
                        </div>
                        <div class="review_list">
                            @foreach (var review in
(List<ProductReview>)ViewBag.Reviews)
                            {
                                <div class="review_item">
                                    <div class="media">

                                        <div class="media-body">

        <h4>@review.UserName</h4>

                                            @for(int i = 0; i <
review.Rating; i++)
                    {
                        <i class="fa fa-star"></i>
                    }

                                        </div>
                                    </div>
                                    <p>
                                        @review.Comment
                                    </p>
                                </div>
                            }


                        </div>
                    </div>
                    <div class="col-lg-6">
                        <div class="review_box">
                            <h4>Додай відгук</h4>
                            <p>Ваша оцінка товару:</p>
```

```
<form asp-controller="Product" asp-
action="AddReview" method="post" class="form-contact form-review mt-3">
    <ul class="list">
        <li><a href="#" class="star"><i
class="fa fa-star"></i></a></li>
        <li><a href="#" class="star"><i
class="fa fa-star"></i></a></li>
        <li><a href="#" class="star"><i
class="fa fa-star"></i></a></li>
        <li><a href="#" class="star"><i
class="fa fa-star"></i></a></li>
        <li><a href="#" class="star"><i
class="fa fa-star"></i></a></li>
    </ul>
    <style>
        .fa-star {
            content: "\f005";
            color: silver;
        }
        .fa-star.checked:before {
            content: "\f005";
            color: gold;
        }</style>
    <input type="hidden" name="rating"
id="rating" value="0">

    <script>
        const stars =
document.querySelectorAll('.star');

        stars.forEach((star, index) => {
            star.addEventListener('click',
(event) => {
                event.preventDefault();
                star.firstElementChild.classList.toggle('checked');

                // додаємо клас
"checked" до всіх попередніх зірочок
                for (let i = 0; i <
index; i++) {
                    stars[i].firstElementChild.classList.add('checked');
                }

                // зберігаємо
кількість увімкнених зірочок в input-полі
                const ratingInput =
document.querySelector('#rating');
                const checkedStars
= document.querySelectorAll('.checked').length;
                ratingInput.value =
checkedStars;
            });
        });
    </script>
    <p>Оцінка</p>
    <input style="visibility: hidden;"
type="number" value="@Model.Id" name="ProductId" />

    <div class="form-group">
        <input class="form-control"
name="username" type="text" placeholder="Введіть ваше ім'я" required>
    </div>
```

```
                                                    <div class="form-group">
                                                        <textarea class="form-control
different-control w-100" name="Comment" id="textarea" cols="30" rows="5" placeholder="Ваш відгук"></textarea>
                                                    </div>
                                                    <div class="form-group text-center text-md-
right mt-3">
                                                        <button type="submit" class="button
button--active button-review">Залишити відгук</button>
                                                    </div>
                                                </form>
                                            </div>
                                        </div>
                                    </div>
                                </div>
                            </div>
                        </div>
                    </section>

@using Muzline.ViewModels;
@model ProductsViewModel
@{
    ViewData["Title"] = "Каталог";
}


<!-- ================= category section start ================= -->
<section class="section-margin--small mb-5">
    <div class="container">
        <div class="row">
            <div class="col-xl-3 col-lg-4 col-md-5">
                <form asp-controller="Product" asp-action="Index">
                <div class="sidebar-categories">
                    <div class="head">Категорії</div>
                    <ul class="main-categories">
                        <li class="common-filter">

                            <ul>
                                @foreach (var category in Model.CategoryListViewModel.Categories)
                                {
                                    <li class="filter-list"><input class="pixel-radio" type="checkbox" value="@category.Id"
id="category" name="categoryIds"><label for="category">@category.Name<span> </span></label></li>
                                }

                            </ul>

                        </li>
                    </ul>
                </div>
                <div class="sidebar-filter">
                    <div class="top-filter-head">Фільтри</div>
                    <div class="common-filter">
                        <div class="head">Бренди</div>

                        <ul>
                            @foreach(var brand in Model.Brands)
                            {
                                <li class="filter-list"><input class="pixel-radio" type="checkbox" value="@brand.Id"
name="brandIds"><label for="apple">@brand.Name<span></span></label></li>
                            }


                        </ul>
```

```
                </div>

                <div class="common-filter">
                 <div  class="head">Ціна:</div>
                      <label style="margin-left: 15px;">Мін:</label>
                      <input style="
    border: 1px solid #eee;
    font-size: 14px;
    color: #999999;
    height: 38px;
    padding-left: 15px;
    margin-left:15px;
" type="number" min="0" id="price" name="minPrice" placeholder="Мінімальна вартість" value="0">
                      <label style="margin-left: 15px;">Макс:</label>
                      <input style="
    border: 1px solid #eee;
    font-size: 14px;
    color: #999999;
    height: 38px;
    padding-left: 15px;
margin-left:4px;
" type="number" min="0" id="price" name="maxPrice" placeholder="Максимальна вартість" value="10000">

                </div>

                </div>
                <button type="submit" class="button button-hero">Вибрати</button>
              </form>
            </div>

            <div class="col-xl-9 col-lg-8 col-md-7">
              <!-- Start Filter Bar -->
              <div class="filter-bar d-flex flex-wrap align-items-center">
                <div class="sorting">
                  Сортувати за:
                  <a asp-action="Index" asp-route-sort="name" value="1">За ім'ям</a>
                  <a asp-action="Index" asp-route-sort="price" value="1">За ціною</a>

                </div>

                <div>
                  <form asp-action="Index">
                  <div class="input-group filter-bar-search">
                       <input type="text" name="searchString" placeholder="Search">
                    <div class="input-group-append">
                         <button onclick="document.getElementById('myForm').submit();" type="submit"><i class="ti-
search"></i></button>
                    </div>
                  </div>
                  </form>
                </div>
              </div>
              <!-- End Filter Bar -->
              <!-- Start Best Seller -->
              <section class="lattest-product-area pb-40 category-list">
                <div class="row">

                  @if(Model.Products.Count() == 0)
                  {
                    <h4>Товарів не знайдено</h4>
                  }
                  @foreach(var item in Model.Products){
                    <div class="col-md-6 col-lg-4">
                    <div class="card text-center card-product">
```

```
                    <div class="card-product__img">
                        @if (item.ProductImages != null && item.ProductImages.Count > 0)
                        {
                            <img class="card-img"
src="@Convert.ToBase64String(item.ProductImages.First().ImageData)" />
                        }

                        <ul class="card-product__imgOverlay">
                            <li><button><i class="ti-search"></i></button></li>
                            <li><button><i class="ti-shopping-cart"></i></button></li>
                            <li><button><i class="ti-heart"></i></button></li>
                        </ul>
                    </div>
                    <div class="card-body">
                        <p>@item.Category.Name</p>
                        <h4 class="card-product__title"><a href="/Product/Details/@item.Id">@item.Name</a></h4>
                        <p class="card-product__price">@item.Price</p>
                    </div>
                </div>
            </div>
            }

        </div>
    </section>
    <!-- End Best Seller -->
</div>
</div>
</div>
</section>
<!-- ================ category section end ================= -->

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>@ViewData["Title"] - Muzline</title>

    <link rel="icon" href="~/img/Fevicon.png" type="image/png">
    <link rel="stylesheet" href="~/vendors/bootstrap/bootstrap.min.css">
    <link rel="stylesheet" href="~/vendors/fontawesome/css/all.min.css">
    <link rel="stylesheet" href="~/vendors/themify-icons/themify-icons.css">
    <link rel="stylesheet" href="~/vendors/nice-select/nice-select.css">
    <link rel="stylesheet" href="~/vendors/owl-carousel/owl.theme.default.min.css">
    <link rel="stylesheet" href="~/vendors/owl-carousel/owl.carousel.min.css">

    <link rel="stylesheet" href="~/css/style.css">
</head>
<body>
    <header class="header_area">
        <div class="main_menu">
            <nav class="navbar navbar-expand-lg navbar-light">
                <div class="container">
                    <a class="navbar-brand logo_h" href="/Home"><img src="~/img/logo.png" alt=""></a>
                    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent"
                        aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                        <span class="icon-bar"></span>
                    </button>
                    <div class="collapse navbar-collapse offset" id="navbarSupportedContent">
                        <ul class="nav navbar-nav menu_nav ml-auto mr-auto">
```

```
                    <li class="nav-item active"><a class="nav-link" href="/Home">Головна</a></li>
                    <li class="nav-item active"><a class="nav-link" href="/Product">Каталог товарів</a></li>

                </ul>

                <ul class="nav-shop">

                    <li class="nav-item"><a href="/Cart"><i class="ti-shopping-cart"></i><span class="nav-
shop__circle">3</span></a> </li>
                    <li class="nav-item"><a class="button button-header" href="/Product">Придбати зараз</a></li>
                </ul>
            </div>
        </div>
    </nav>
  </div>
</header>

    @RenderBody()

<!--================ Start footer Area  ================-->
<footer class="footer">
  <div class="footer-area">
    <div class="container">
      <div class="row section_gap">
        <div class="col-lg-3 col-md-6 col-sm-6">
          <div class="single-footer-widget tp_widgets">
            <h4 class="footer_title large_title">Muzline </h4>
            <p>
              Широкий асортимент - всі основні напрямки і бренди

Тільки професійна консультація і допомога при покупці

Робота в будь-який зручний для Вас час

Можливість замовлення товару, якого немає в каталозі

Максимально швидке обслуговування (доставка по всій Україні протягом 1-3 днів)
            </p>

          </div>
        </div>
        <div class="offset-lg-1 col-lg-2 col-md-6 col-sm-6">
          <div class="single-footer-widget tp_widgets">
            <h4 class="footer_title">Швидкі посилання</h4>
            <ul class="list">
              <li><a href="/Home">Головна </a></li>
              <li><a href="/Product">Продукція</a></li>

              <li><a href=/Contact">Контакти  </a></li>
            </ul>
          </div>
        </div>

        <div class="offset-lg-1 col-lg-3 col-md-6 col-sm-6">
          <div class="single-footer-widget tp_widgets">
            <h4 class="footer_title">Наші контакти</h4>
            <div class="ml-40">
              <p class="sm-head">
                <span class="fa fa-location-arrow"></span>
                Головний офіс
              </p>
              <p>123, Кіото, Київ</p>

              <p class="sm-head">
```

```
                        <span class="fa fa-phone"></span>
                        Номер телефону
                      </p>
                      <p>
                        +830 456 7890 <br>
                        +830 456 7890
                      </p>

                      <p class="sm-head">
                        <span class="fa fa-envelope"></span>
                        Email
                      </p>
                      <p>
                        free@infoexample.com <br>
                        www.infoexample.com
                      </p>
                    </div>
                  </div>
                </div>
              </div>
            </div>
          </div>

          <div class="footer-bottom">
            <div class="container">
              <div class="row d-flex">
                <p class="col-lg-12 footer-text text-center">
                  <!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0. -->

                  Copyright &copy;<script>document.write(new Date().getFullYear());</script> All rights reserved | This site is made with <i class="fa fa-heart" aria-hidden="true"></i> by Головченко Віолета <a style="visibility:hidden" href="https://colorlib.com" target="_blank">Colorlib</a>
                  <!-- Link back to Colorlib can't be removed. Template is licensed under CC BY 3.0. -->
                </p>
              </div>
            </div>
          </div>
        </footer>
        <!--================= End footer Area  =================-->


        <script src="~/vendors/jquery/jquery-3.2.1.min.js"></script>
        <script src="~/vendors/bootstrap/bootstrap.bundle.min.js"></script>
        <script src="~/vendors/skrollr.min.js"></script>
        <script src="~/vendors/owl-carousel/owl.carousel.min.js"></script>
        <script src="~/vendors/nice-select/jquery.nice-select.min.js"></script>
        <script src="~/vendors/jquery.ajaxchimp.min.js"></script>
        <script src="vendors/nouislider/nouislider.min.js"></script>
        <script src="~/vendors/mail-script.js"></script>
        <script src="~/js/main.js"></script>

    </body>
    </html>
```

Код класу BrandsController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;

namespace MuzlineAdminPanel.Controllers
{
    public class BrandsController : Controller
    {
        private readonly DataContext _context;

        public BrandsController(DataContext context)
        {
            _context = context;
        }

        // GET: Brands
        public async Task<IActionResult> Index()
        {
            return _context.Brands != null ?
                        View(await _context.Brands.ToListAsync()) :
                        Problem("Entity set 'DataContext.Brands'  is null.");
        }

        // GET: Brands/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Brands == null)
            {
                return NotFound();
            }

            var brand = await _context.Brands
                .FirstOrDefaultAsync(m => m.Id == id);
            if (brand == null)
            {
                return NotFound();
            }

            return View(brand);
        }

        // GET: Brands/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Brands/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,Name,Description")] Brand brand)
        {
            _context.Add(brand);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
```

```
}

// GET: Brands/Edit/5
public async Task<IActionResult> Edit(int? id)
{
    if (id == null || _context.Brands == null)
    {
        return NotFound();
    }

    var brand = await _context.Brands.FindAsync(id);
    if (brand == null)
    {
        return NotFound();
    }
    return View(brand);
}

// POST: Brands/Edit/5
// To protect from overposting attacks, enable the specific properties you want to bind to.
// For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
[HttpPost]
[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Description")] Brand brand)
{
    if (id != brand.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            _context.Update(brand);
            await _context.SaveChangesAsync();
        }
        catch (DbUpdateConcurrencyException)
        {
            if (!BrandExists(brand.Id))
            {
                return NotFound();
            }
            else
            {
                throw;
            }
        }
        return RedirectToAction(nameof(Index));
    }
    return View(brand);
}

// GET: Brands/Delete/5
public async Task<IActionResult> Delete(int? id)
{
    if (id == null || _context.Brands == null)
    {
        return NotFound();
    }

    var brand = await _context.Brands
        .FirstOrDefaultAsync(m => m.Id == id);
    if (brand == null)
```

```
            {
                return NotFound();
            }

            return View(brand);
        }

        // POST: Brands/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Brands == null)
            {
                return Problem("Entity set 'DataContext.Brands'  is null.");
            }
            var brand = await _context.Brands.FindAsync(id);
            if (brand != null)
            {
                _context.Brands.Remove(brand);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool BrandExists(int id)
        {
            return (_context.Brands?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```

Код класу CategoriesController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;
using Muzline.Models;

namespace MuzlineAdminPanel.Controllers
{
    public class CategoriesController : Controller
    {
        private readonly DataContext _context;

        public CategoriesController(DataContext context)
        {
            _context = context;
        }

        // GET: Categories
        public async Task<IActionResult> Index()
        {
            return _context.Categories != null ?
                        View(await _context.Categories.ToListAsync()) :
                        Problem("Entity set 'DataContext.Categories'  is null.");
        }

        // GET: Categories/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Categories == null)
            {
                return NotFound();
            }

            var category = await _context.Categories
                .FirstOrDefaultAsync(m => m.Id == id);
            if (category == null)
            {
                return NotFound();
            }

            return View(category);
        }

        // GET: Categories/Create
        public IActionResult Create()
        {
            return View();
        }

        // POST: Categories/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,Name")] Category category)
        {

            _context.Add(category);
```

```
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));


        }

        // GET: Categories/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.Categories == null)
            {
                return NotFound();
            }

            var category = await _context.Categories.FindAsync(id);
            if (category == null)
            {
                return NotFound();
            }
            return View(category);
        }

        // POST: Categories/Edit/5
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,Name")] Category category)
        {
            if (id != category.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(category);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!CategoryExists(category.Id))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            return View(category);
        }

        // GET: Categories/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.Categories == null)
            {
                return NotFound();
            }
```

```
            var category = await _context.Categories
                .FirstOrDefaultAsync(m => m.Id == id);
            if (category == null)
            {
                return NotFound();
            }

            return View(category);
        }

        // POST: Categories/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Categories == null)
            {
                return Problem("Entity set 'DataContext.Categories'  is null.");
            }
            var category = await _context.Categories.FindAsync(id);
            if (category != null)
            {
                _context.Categories.Remove(category);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool CategoryExists(int id)
        {
            return (_context.Categories?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```

Код класу NewProductController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;
using Muzline.Models;

namespace MuzlineAdminPanel.Controllers
{
    public class NewProductController : Controller
    {
        private readonly DataContext _context;

        public NewProductController(DataContext context)
        {
            _context = context;
        }

        // GET: NewProduct
        public async Task<IActionResult> Index()
        {
            var dataContext = _context.Products.Include(p => p.Brand).Include(p => p.Category);
            return View(await dataContext.ToListAsync());
        }

        // GET: NewProduct/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Products == null)
            {
                return NotFound();
            }

            var product = await _context.Products
                .Include(p => p.Brand)
                .Include(p => p.Category)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (product == null)
            {
                return NotFound();
            }

            return View(product);
        }

        // GET: NewProduct/Create
        public IActionResult Create()
        {
            ViewData["BrandId"] = new SelectList(_context.Brands, "Id", "Description");
            ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name");
            return View();
        }

        // POST: NewProduct/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,Name,Description,Price,BrandId,CategoryId")] Product product)
```

```
        {

            _context.Add(product);
            await _context.SaveChangesAsync();
            return Redirect("/ProductImages/Create");

            //ViewData["BrandId"] = new SelectList(_context.Brands, "Id", "Description", product.BrandId);
            //ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name", product.CategoryId);
            //return View(product);
        }

        // GET: NewProduct/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.Products == null)
            {
                return NotFound();
            }

            var product = await _context.Products.FindAsync(id);
            if (product == null)
            {
                return NotFound();
            }
            ViewData["BrandId"] = new SelectList(_context.Brands, "Id", "Description", product.BrandId);
            ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name", product.CategoryId);
            return View(product);
        }

        // POST: NewProduct/Edit/5
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Description,Price,BrandId,CategoryId")] Product
product)
        {
            if (id != product.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(product);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
                {
                    if (!ProductExists(product.Id))
                    {
                        return NotFound();
                    }
                    else
                    {
                        throw;
                    }
                }
                return RedirectToAction(nameof(Index));
            }
            ViewData["BrandId"] = new SelectList(_context.Brands, "Id", "Description", product.BrandId);
            ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name", product.CategoryId);
```

```
            return View(product);
        }

        // GET: NewProduct/Delete/5
        public async Task<IActionResult> Delete(int? id)
        {
            if (id == null || _context.Products == null)
            {
                return NotFound();
            }

            var product = await _context.Products
                .Include(p => p.Brand)
                .Include(p => p.Category)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (product == null)
            {
                return NotFound();
            }

            return View(product);
        }

        // POST: NewProduct/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Products == null)
            {
                return Problem("Entity set 'DataContext.Products'  is null.");
            }
            var product = await _context.Products.FindAsync(id);
            if (product != null)
            {
                _context.Products.Remove(product);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool ProductExists(int id)
        {
            return (_context.Products?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```

Код класу ProductImagesController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;
using Muzline.Models;

namespace MuzlineAdminPanel.Controllers
{
    public class ProductImagesController : Controller
    {
        private readonly DataContext _context;

        public ProductImagesController(DataContext context)
        {
            _context = context;
        }

        // GET: ProductImages
        public async Task<IActionResult> Index()
        {
            var dataContext = _context.ProductImages.Include(p => p.Product);
            return View(await dataContext.ToListAsync());
        }

        // GET: ProductImages/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.ProductImages == null)
            {
                return NotFound();
            }

            var productImage = await _context.ProductImages
                .Include(p => p.Product)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (productImage == null)
            {
                return NotFound();
            }

            return View(productImage);
        }

        // GET: ProductImages/Create
        public IActionResult Create()
        {
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Description");
            return View();
        }

        // POST: ProductImages/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,FileName,ImageData,ProductId")] ProductImage productImage,
IFormFile imageFile)
        {
```

```
            if (imageFile != null && imageFile.Length > 0)
            {
                using (var stream = new MemoryStream())
                {
                    await imageFile.CopyToAsync(stream);
                    productImage.ImageData = stream.ToArray();
                }
                productImage.FileName = imageFile.FileName;
            }

            _context.Add(productImage);
            await _context.SaveChangesAsync();
            return Redirect("/ProductSpecifications/Create");

            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Name", productImage.ProductId);
            //return View(productImage);
            //if (ModelState.IsValid)
            //{
            //    _context.Add(productImage);
            //    await _context.SaveChangesAsync();
            //    return RedirectToAction(nameof(Index));
            //}
            //ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Description", productImage.ProductId);
            //return View(productImage);
        }

        // GET: ProductImages/Edit/5
        public async Task<IActionResult> Edit(int? id)
        {
            if (id == null || _context.ProductImages == null)
            {
                return NotFound();
            }

            var productImage = await _context.ProductImages.FindAsync(id);
            if (productImage == null)
            {
                return NotFound();
            }
            ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Description", productImage.ProductId);
            return View(productImage);
        }

        // POST: ProductImages/Edit/5
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Edit(int id, [Bind("Id,FileName,ImageData,ProductId")] ProductImage
productImage)
        {
            if (id != productImage.Id)
            {
                return NotFound();
            }

            if (ModelState.IsValid)
            {
                try
                {
                    _context.Update(productImage);
                    await _context.SaveChangesAsync();
                }
                catch (DbUpdateConcurrencyException)
```

```
            {
                if (!ProductImageExists(productImage.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        ViewData["ProductId"] = new SelectList(_context.Products, "Id", "Description", productImage.ProductId);
        return View(productImage);
    }

    // GET: ProductImages/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
        if (id == null || _context.ProductImages == null)
        {
            return NotFound();
        }

        var productImage = await _context.ProductImages
            .Include(p => p.Product)
            .FirstOrDefaultAsync(m => m.Id == id);
        if (productImage == null)
        {
            return NotFound();
        }

        return View(productImage);
    }

    // POST: ProductImages/Delete/5
    [HttpPost, ActionName("Delete")]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> DeleteConfirmed(int id)
    {
        if (_context.ProductImages == null)
        {
            return Problem("Entity set 'DataContext.ProductImages'  is null.");
        }
        var productImage = await _context.ProductImages.FindAsync(id);
        if (productImage != null)
        {
            _context.ProductImages.Remove(productImage);
        }

        await _context.SaveChangesAsync();
        return RedirectToAction(nameof(Index));
    }

    private bool ProductImageExists(int id)
    {
        return (_context.ProductImages?.Any(e => e.Id == id)).GetValueOrDefault();
    }
  }
}
```

Код класу ProductsController

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using Microsoft.AspNetCore.Mvc;
using Microsoft.AspNetCore.Mvc.Rendering;
using Microsoft.EntityFrameworkCore;
using Muzline.Data;
using Muzline.Models;

namespace MuzlineAdminPanel.Controllers
{
    public class ProductsController : Controller
    {
        private readonly DataContext _context;

        public ProductsController(DataContext context)
        {
            _context = context;
        }

        // GET: Products
        public async Task<IActionResult> Index()
        {
            var dataContext = _context.Products.Include(p => p.Category);
            return View(await dataContext.ToListAsync());
        }

        // GET: Products/Details/5
        public async Task<IActionResult> Details(int? id)
        {
            if (id == null || _context.Products == null)
            {
                return NotFound();
            }

            var product = await _context.Products
                .Include(p => p.Category)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (product == null)
            {
                return NotFound();
            }

            return View(product);
        }

        // GET: Products/Create
        public IActionResult Create()
        {
            ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name");
            return View();
        }

        // POST: Products/Create
        // To protect from overposting attacks, enable the specific properties you want to bind to.
        // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
        [HttpPost]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> Create([Bind("Id,Name,Description,Price,CategoryId")] Product product)
        {
```

```
            _context.Add(product);
            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));

        //ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name", product.CategoryId);
        //return View(product);
    }

    // GET: Products/Edit/5
    public async Task<IActionResult> Edit(int? id)
    {
        if (id == null || _context.Products == null)
        {
            return NotFound();
        }

        var product = await _context.Products.FindAsync(id);
        if (product == null)
        {
            return NotFound();
        }
        ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name", product.CategoryId);
        return View(product);
    }

    // POST: Products/Edit/5
    // To protect from overposting attacks, enable the specific properties you want to bind to.
    // For more details, see http://go.microsoft.com/fwlink/?LinkId=317598.
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int id, [Bind("Id,Name,Description,Price,CategoryId")] Product product)
    {
        if (id != product.Id)
        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                _context.Update(product);
                await _context.SaveChangesAsync();
            }
            catch (DbUpdateConcurrencyException)
            {
                if (!ProductExists(product.Id))
                {
                    return NotFound();
                }
                else
                {
                    throw;
                }
            }
            return RedirectToAction(nameof(Index));
        }
        ViewData["CategoryId"] = new SelectList(_context.Categories, "Id", "Name", product.CategoryId);
        return View(product);
    }

    // GET: Products/Delete/5
    public async Task<IActionResult> Delete(int? id)
    {
```

```
            if (id == null || _context.Products == null)
            {
                return NotFound();
            }

            var product = await _context.Products
                .Include(p => p.Category)
                .FirstOrDefaultAsync(m => m.Id == id);
            if (product == null)
            {
                return NotFound();
            }

            return View(product);
        }

        // POST: Products/Delete/5
        [HttpPost, ActionName("Delete")]
        [ValidateAntiForgeryToken]
        public async Task<IActionResult> DeleteConfirmed(int id)
        {
            if (_context.Products == null)
            {
                return Problem("Entity set 'DataContext.Products'  is null.");
            }
            var product = await _context.Products.FindAsync(id);
            if (product != null)
            {
                _context.Products.Remove(product);
            }

            await _context.SaveChangesAsync();
            return RedirectToAction(nameof(Index));
        }

        private bool ProductExists(int id)
        {
            return (_context.Products?.Any(e => e.Id == id)).GetValueOrDefault();
        }
    }
}
```