МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра МОиПЭВМ

Направление подготовки 09.03.04 «Программная инженерия»

БАКАЛАВРСКАЯ РАБОТА

на тему

Аудио Сервис для пользователя интернет

Студент _____ Тарин Андрей Аркадьевич
(подпись, дата) (ФИО полностью)

Руководитель _____ Шашков Б.Д.
(подпись, дата) (фамилия, инициалы)

Нормоконтролёр _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2021

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____П.П. Макарычев
«__» _____2021 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ

БАКАЛАВРА

1. Студент Тарин Андрей Аркадьевич гр. 17ВП1 факультета ВТ
<Фамилия, имя, отчество полностью>
2. Тема работы Аудио Сервис для пользователя интернет

Тема утверждена приказом ПГУ №_____от "_____"_____2021 г.

2. Руководитель работы к.т.н., профессор кафедры «МО и ПЭВМ» Шашков Б.Д.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

1) Назначение: программное обеспечение предназначено для прослушивания, скачивания, загрузки, обсуждения, интерпретации музыки.

2) Среда выполнения: операционная система Windows 10.

Средства разработки: языки программирования C#, SQL.

Инструментальные средства: Microsoft Visual Studio, My SQL Workbench

Технологии разработки: ООП, UML, ASP.NET CORE.

3) Основные функции программного обеспечения аудио сервиса для пользователя интернет:

а) прослушать музыку в режиме онлайн и оффлайн;

б) обсудить музыку с другими пользователями;

в) интерпретировать музыку;

г) прослушать плейлист;

д) управлять прослушиванием музыки;

е) зарегистрироваться в сервисе;

ж) авторизироваться в сервисе;

з) редактировать данные о сотрудниках;

и) скачивать с сервера музыку и загружать музыку на сервер.

к) осуществлять поиск музыки по ключевым словам.

4. Перечень подлежащих разработке вопросов

Анализ предметной области

Анализ требований

Проектирование

Реализация

Тестирование

План разработки и оценка бюджета

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований	10%	14.04.2021 – 18.04.2021	
2	Проектирование	25%	19.05.2021 – 27.05.2021	
3	Реализация	35%	28.04.2021 – 26.05.2021	
4	Тестирование	15%	27.05.2021 – 02.06.2021	
5	План разработки бюджета	5%	03.06.2021 – 05.06.2021	
6	Оформление пояснительной записки	10%	06.06.2021 – 14.06.2021	

Дата выдачи задания "___" _____2021 г.

Руководитель бакалаврской работы _____
(подпись, дата)

Задание к исполнению принял студент _____
(подпись, дата)

Работу к защите допустить
Декан факультета _____
(подпись, дата)

Реферат

Пояснительная записка содержит 87 листов, 39 рисунков, 15 использованных источников.

ПРОГРАММНОЕ ОБЕСПЕЧЕНИЕ, АУДИО СЕРВИС, БАЗА ДАННЫХ, ООП, C#, КЛИЕНТ, СЕРВЕР, UML, MY SQL WORKBENCH, MICROSOFT VISUAL STUDIO, ASP.NET CORE, WPF.

Объект разработки – сервис прослушивания музыки, состоящий из клиентской и серверной частей.

Цель работы – разработать клиентскую и серверную части аудио сервиса.

Технология разработки – ООП, UML, ASP.NET Core.

Для создания серверной части использовался фреймворк ASP.NET Core. Для проектирования серверной БД использовалось приложение CA Erwin Data Modeler 7.3. Для управления БД использована СУБД MYSQL. Для создания клиентской части применялся язык C# в IDE Visual Studio 2019 на графической подсистеме WPF.

Результаты работы – в ходе выполнения выпускной квалификационной работы был создан аудио сервис для пользователя интернет. Все требования к разработке приложения соблюдены.

Содержание

Введение 6

1 Анализ предметной области и требований 7

1.1 Анализ предметной области 7

1.2 Анализ приложений для прослушивания аудио 8

1.3 Анализ требований 12

1.3.1 Функциональные требования 12

1.3.2 Нефункциональные требования 15

2 Проектирование программных средств 16

2.1 Архитектура сервиса 16

2.2 Логическая модель базы данных 16

2.3 Физическая модель базы данных 18

2.4 Диаграмма классов 19

2.5 Диаграмма состояний 20

2.6 Диаграмма последовательности 21

2.7 Диаграмма деятельности 22

2.8 Диаграмма кооперации 23

3 Реализация программных средств 24

3.1 Реализация пользовательского интерфейса 24

3.2 Диаграмма компонентов клиентского приложения 32

3.3 Диаграмма развёртывания 34

4 Тестирование программного продукта 36

4.1 Структурное тестирование 36

4.1.1 Метод покрытия операторов 37

4.1.2 Метод покрытия условий 37

4.1.3 Метод покрытия маршрутов 38
4.2 Функциональное тестирование 38
4.2.1 Классы эквивалентности 39
4.2.2 Анализ граничных условий 40
4.2.3 Функциональная диаграмма 41
4.2.4 Парное тестирование 43
4.2.5 Тестирование по вариантам использования 45
4.3 Модульное тестирование 46
4.4 Метрики кода 47
4.5 Планирование и оценка бюджета 49
Заключение 53
Список использованных источников 54
Приложение А. Глоссарий 56
Приложение Б. Листинг клиентского приложения на языке C# 58
Приложение В. Результаты функционального тестирования 77

Введение

Как верно отметил Бетховен: «Музыка выше, чем все откровения мудрости и философии». Развитие технологий упростило доступ человечества к искусству и к музыке в частности.

Если раньше человек был ограничен:

– расстоянием до театра, оперы и других мест, где бы он мог услышать музыку;

– финансовыми возможностями, чтобы попасть в такие места;

– наличием конкретных исполнителей, поскольку каждый раз необходимо было вживую проигрывать мелодию;

– качеством исполнителей;

– качеством инструментов и многими другими условиями.

То теперь любой пользователь интернета способен получить доступ ко всему многообразию искусства когда-либо созданного человеком.

Задачей же информационных технологий является обеспечение комфортного доступа пользователя. Предоставление удобного интерфейса для поиска желаемого произведения, его прослушивания, сохранения. Дополнительно, поскольку красота объединяет людей, то информационные технологии должны позволять людям обсуждать предметы их интереса, делиться мнениями для социализации, воспитания вкуса. И, в свою очередь, чтобы современные исполнители могли публиковать результаты своего созерцания для развития культуры человечества и конкретных личностей в отдельности.

Таким образом, проект аудио сервиса, позволяющий пользователям прослушивать музыку, является полезным и востребованным. Поэтому тема выпускной квалификационной работы является актуальной.

1 Анализ предметной области и требований

1.1 Анализ предметной области

На протяжении всей своей многовековой истории люди стремились проникнуть в тайны звуков окружающего мира. Удивительное влияние они оказали на все стороны человеческой жизни. Даже не обладая научным знанием, не понимая природы звука – древние люди на интуитивном уровне чувствовали красоту мелодии и создавали музыкальные инструменты поразительной тонкости [1].

И, понимая всю важность музыки для человека, необходимо разработать программу, использование которой бы было приятно пользователю. Это программа не для работы или праздности, поэтому подход к ее проектированию должен быть соответствующий, а именно:

– интерфейс не должен отвлекать своей яркостью или блеклостью, влиять на настроение пользователя;

– приложение должно быть устойчивым к нештатным ситуациям и непредсказуемым условиям;

– интерфейс должен быть интуитивно простым, и в то же время полно передавать происходящее;

– недопустимо наличие каких-либо ограничений для пользователя, либо наличие рекламы.

Последний пункт является особенно важным, учитывая, как неуместная реклама способна разрушить погружение слушателя в мир музыки.

Дополнительно, стоит отметить все возрастающую роль в жизни сервисов, облегчающих доступ к информации. Важным условием развития и безопасности является конкуренция сервисов, их разнообразие, так как при монополизации качество услуг падает. Например, ВКонтакте на пике своей рыночной гегемонии добавил рекламу при прослушивании аудиозаписей [2]. Следовательно, для улучшения сервисов необходимо создавать конкурентные альтернативы существующему предложению.

1.2 Анализ приложений для прослушивания аудио

Для анализа рассмотрены приложения Музыка Groove, Яндекс.Музыка, Spotify, Музыка ВКонтакте.

Музыка Groove – музыкальный проигрыватель Microsoft. Стандартный проигрыватель для операционной системы Windows 10. Приведён в сравнении, поскольку является предустановленным в самую популярную операционную систему в мире, его можно считать точкой отсчета для сравнения аудио сервисов (рисунок 1).

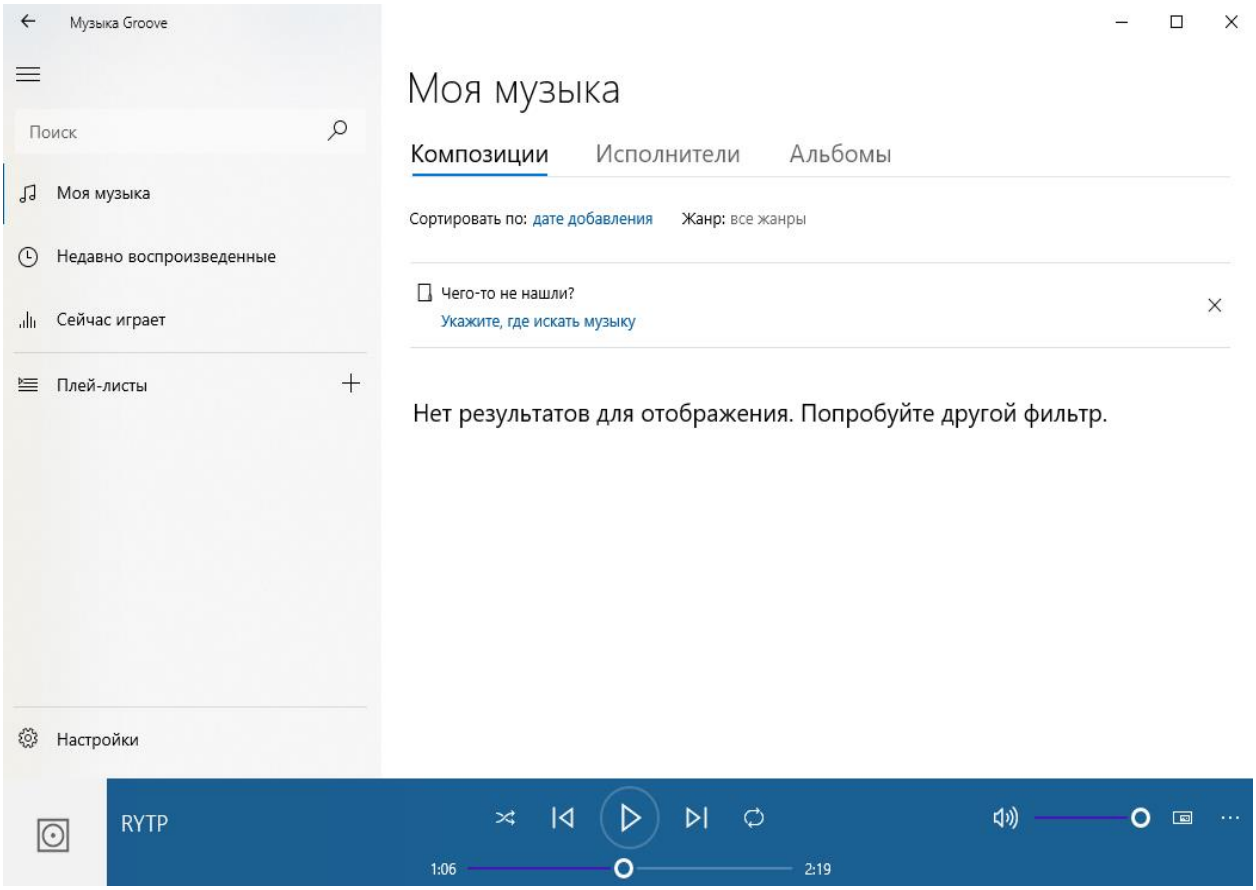

Рисунок 1 – Интерфейс Музыка Groove

Музыка ВКонтакте является популярным сервисом, поскольку ВКонтакте – крупнейшая социальная сеть в России [3] (рисунок 2).

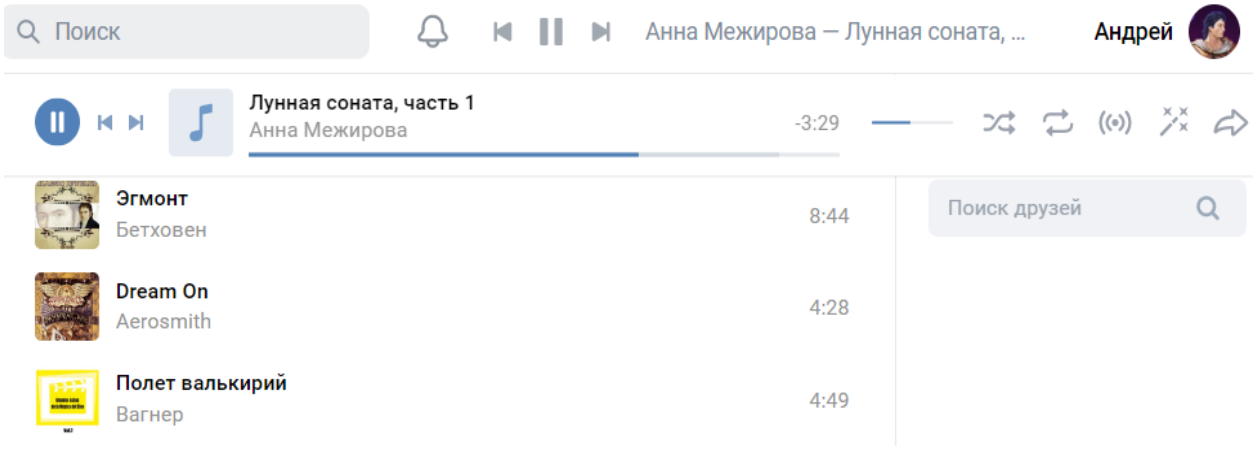

Рисунок 2 – Интерфейс Музыка ВКонтакте

Яндекс.Музыка приведена в таблице, поскольку Яндекс является крупнейшей IT – компанией в России [4] (рисунок 3).

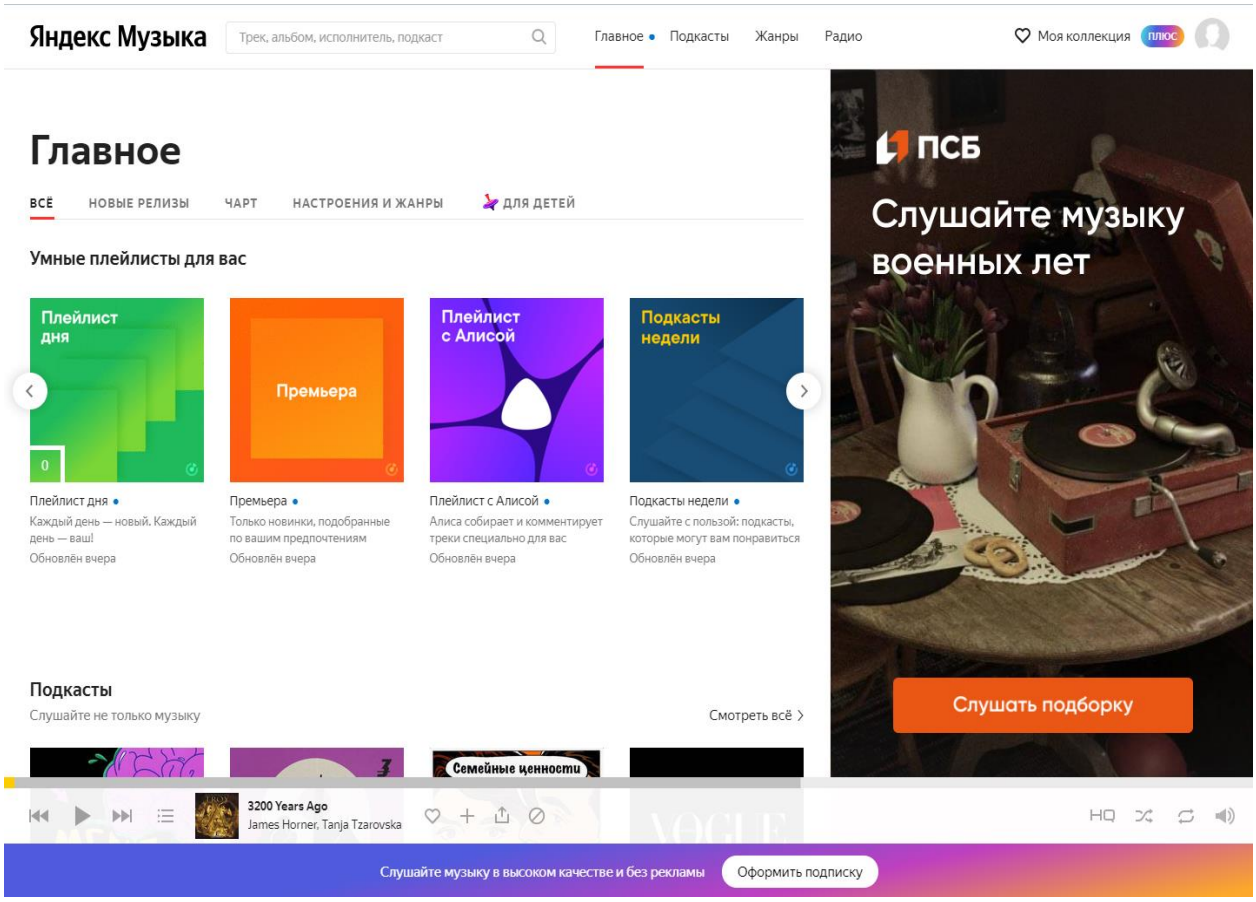

Рисунок 3 – Интерфейс Яндекс.Музыка

Spotify приведен, поскольку его запуск в России был самым успешным открытием за всю историю компании. Это значит, что пользователи российского рынка высоко ценят данный сервис [5] (рисунок 4).

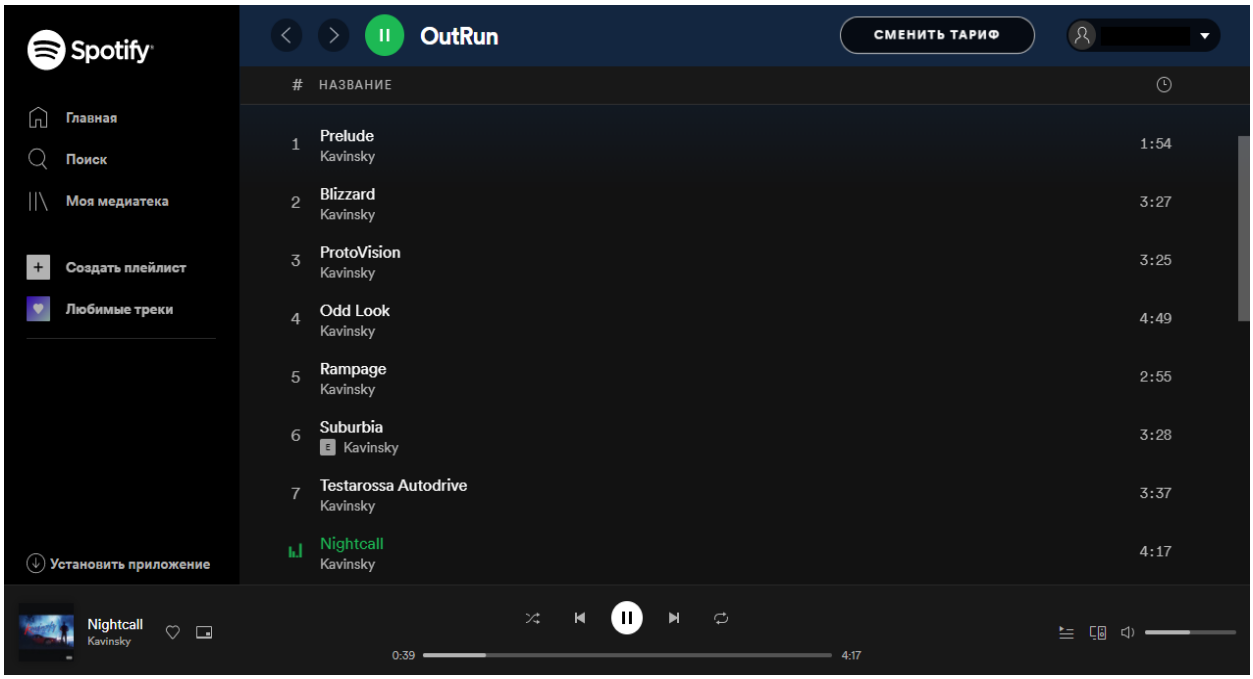

Рисунок 4 – Интерфейс Spotify

В таблице 1 приведено сравнение приложений аналогов.

Таблица 1 – Сравнение приложений

№	Черты	Groove	ВКонтакте	Яндекс	Spotify
1	**2**	**3**	**4**	**5**	**6**
1	Глобальный поиск	+	+	+	+
2	Реализованы плейлисты	+	+	+	+
3	Случайный порядок воспроизведения	+	+	+	+
4	Информация о композиции	+	+	+	+
5	Реклама	-	+	+	-
6	Возможность потокового скачивания музыки	-	+	+	+
7	Возможность использования без интернета	+	-	+	+
8	Ограничения бесплатной версии	-	+	+	+

Продолжение таблицы 1

1	2	3	4	5	6
9	Возможность указать текст песни	-	+	-	-
10	Возможность интерпретировать текст песни	-	-	-	-
11	Возможность ставить оценки	-	-	-+	-+
12	Наличие сервиса для обсуждения, комментирования песни	-	-	-	-

Сравнение аналогов позволяет выделить общие признаки, которые обязательно должны быть реализованы для конкурентоспособности проекта, а также конкурентные преимущества – признаки, которые являют выгодное превосходство проекта над аналогами.

Резюмируя, проект должен обладать базовой функциональностью:

- поиском музыки;
- работой с плейлистами;
- функцией случайного порядка воспроизведения;
- информацией о композиции;
- возможностью использования без интернета;
- возможностью потоковой загрузки музыки.

И конкурентными преимуществами:

- отсутствием рекламы;
- отсутствием ограничений бесплатной версии;
- возможностями социального взаимодействия пользователей;
- возможностью прикрепления текста и слов и их интерпретации.

Таким образом, создана база для создания функциональных требований к проекту.

1.3 Анализ требований

1.3.1 Функциональные требования

На основе анализа предметной области и сравнения приложений аналогов были построены диаграммы вариантов использования, которые выражают функциональные требования к проекту [6]. Поскольку общая диаграмма вариантов использования объемная, то было принято решение для упрощения разделить ее на две части: варианты использования в режиме оффлайн и онлайн.

На рисунке 5 показана диаграмма вариантов использования в режиме работы приложения оффлайн.

Рисунок 5 – Диаграмма вариантов использования приложения в режиме оффлайн

Выделен единственный актер «Пользователь», он описывает пользователя приложения для прослушивания музыки.

На рисунке 6 показана диаграмма вариантов использования в режиме работы приложения онлайн.

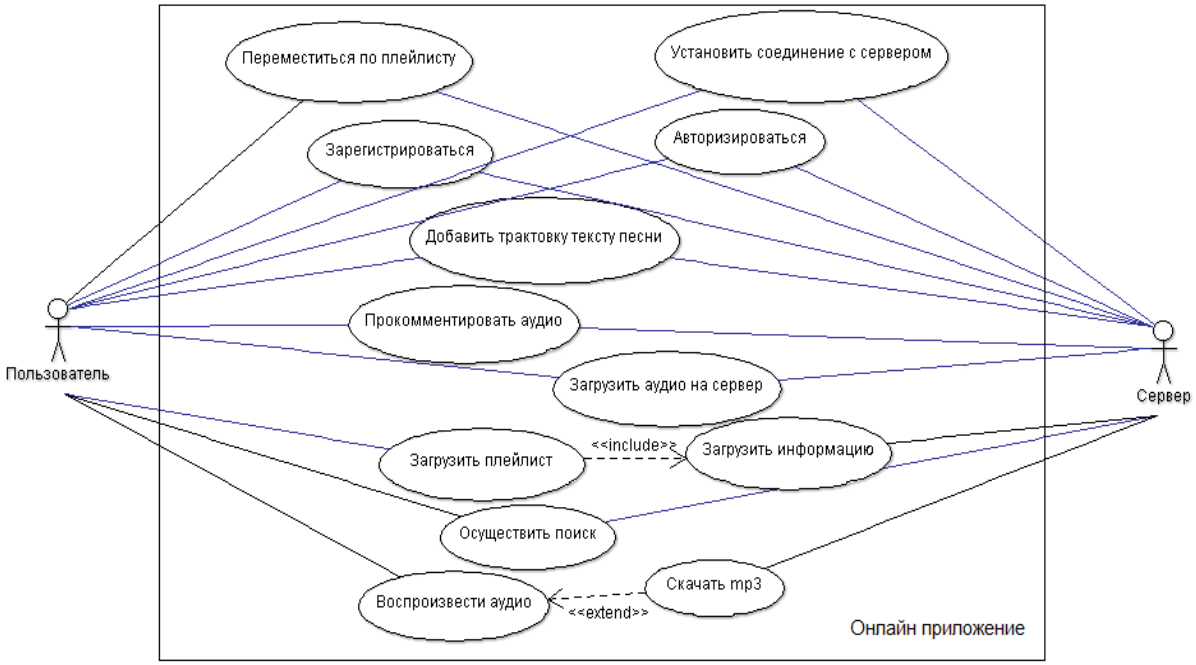

Рисунок 6 – Диаграмма вариантов использования приложения в режиме онлайн

Выделен дополнительный актер «Сервер», он описывает работу сервера, который выполняет запросы, которые поступают к нему с приложения в результате манипуляций пользователя сервиса. Запросы могут требовать передачи музыки, информации о ней, сбора информации от пользователей для социального взаимодействия между ними, обмена мнениями о музыкальных композициях. Для оптимизации работы клиент-серверной архитектуры и снижения нагрузки на сеть загрузка аудио-файла и информации о нем разделена.

Следует учитывать, что варианты использования, описанные на рисунке 5, выполняются всегда приложением, а варианты использования с рисунка 6 только при условии интернет-соединения.

Поскольку пользователь для прослушивания скачивает файлы с сервера, то после скачивания становится возможным их прослушивание без активного соединения.

Далее приведу сценарии для определенных вариантов использования (таблицы 2 – 4).

Таблица 2 – Сценарий «Переключить звук»

Наименование: Переключить звук.
ID: 1
Краткое описание: Переключение звука пользователем.
Главные актеры: Пользователь
Предусловия: Нет.
Основной поток: 1. Пользователь нажимает на кнопку переключения звука. 2. ЕСЛИ Звук активен 2.1 Звук выключается. 3. ИНАЧЕ 3.1 Звук включается.
Постусловия: Нет.

Таблица 3 – Сценарий «Загрузить плейлист»

Наименование: Загрузить плейлист.
ID: 2
Краткое описание: Загрузка плейлиста с сервера.
Главные актеры: Пользователь, Сервер
Предусловия: Установлено соединение с сервером.
Основной поток: 1. Пользователь совершает запрос на загрузку плейлиста. 2. Сервер определяет плейлист. 3. Сервер формирует json-файл с информацией о плейлисте и композициях. 4. Сервер отправляет json-файл приложению. 5. Приложение формирует представление плейлиста. 6. ЕСЛИ приложению необходимы картинки для аудио-файлов 6.1 Приложение отправляет запрос на загрузку картинки 6.2 Сервер возвращает картинку 6.3 Шаг 6
Постусловия: Нет.

Таблица 4 – Сценарий «Воспроизвести аудио»

Наименование: Воспроизвести аудио.
ID: 3
Краткое описание: Включение выбранной музыки.
Главные актеры: Пользователь, Сервер
Предусловия: Нет.
Основной поток: 1. Пользователь нажимает на кнопку воспроизведения. 2. ЕСЛИ на компьютере пользователя нет аудио файла 2.1 Приложение совершает запрос на загрузку файла. 2.2 Сервер отправляет файл приложению. 2.3 Приложение сохраняет файл в папке приложения. 2.4 Приложение запоминает путь до файла. 3 Приложение воспроизводит файл.
Постусловия: Нет.

Определены функциональные требования к разрабатываемому сервису. Далее дается определение нефункциональных требований.

1.3.2 Нефункциональные требования

Клиентская и серверная часть приложения должна быть реализована на языке C#, на платформе .Net Core. C# является мощным инструментом разработки и позволяет вести быструю разработку за счет большой библиотеки встроенных типов в платформу .Net Core [7]. Приложение должно корректно функционировать на ОС Windows 10.

Клиент должен быть написан на WPF . WPF – это современная подход к созданию графических приложений на ОС Windows . WPF использует язык разметки XAML, что упрощает создание пользовательского графического интерфейса. WPF имеет встроенные механизмы проектирования архитектуры приложения MVVM, позволяющие структурировать разработку [8].

Сервер должен использовать фреймворк ASP.NET Core, который удобен для создания серверных веб-приложений и который имеет встроенный архитектурный шаблон MVC [8]. Для хранения данных на сервере использовать СУБД MySQL Worckbench – является популярной, мощной и надежной базой данных [9].

При разработке использовать IDE Visual Studio 2019. VS является мощной интегрированной средой разработки, обладающей преимуществами:

- автоматизацией сборки;
- широким спектром инструментов для навигации в коде (поиск всех ссылок, всех упоминаний, глобальный поиск и т.д.);
- возможностью отладочной развертки веб-сервера;
- поддержкой возможностей автоматизированного рефакторинга;
- возможностями по генерации документации;
- визуализации графических элементов в реальном времени;
- отладчиком кода с возможностями замера производительности.

Поэтому Visual Studio 2019 является лучшим выбором.

2 Проектирование программных средств

2.1 Архитектура сервиса

Структура разрабатываемого аудио сервиса представлена на рисунке 7.

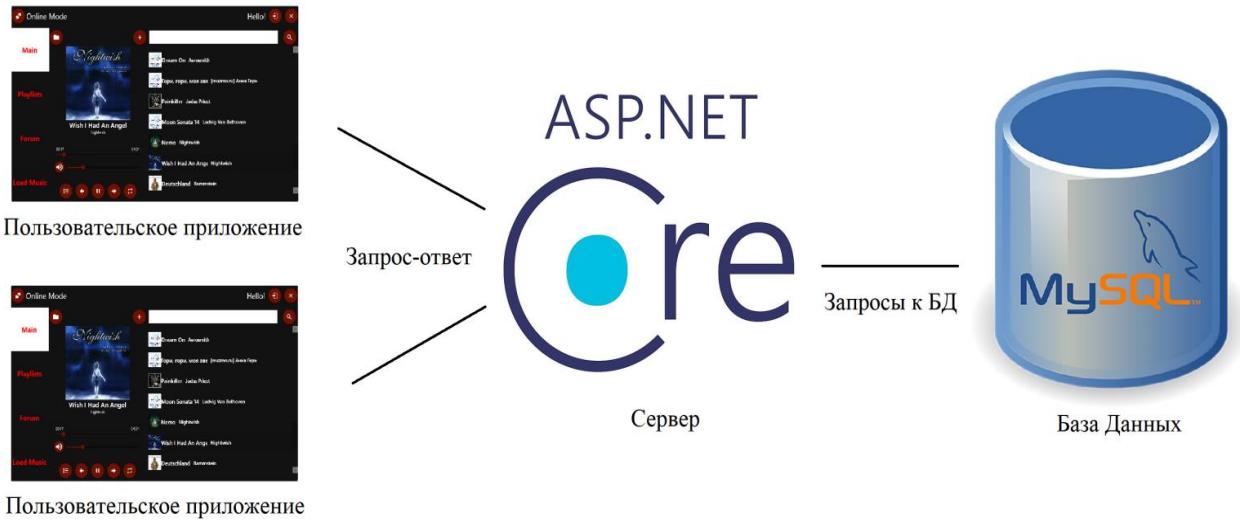

Рисунок 7 – Структура аудио сервиса

Сервис обладает клиент-серверной архитектурой. Клиентами выступают пользователи клиентских программ. Клиенты посылают запросы на сервер, реализованный на платформе ASP.NET Core. Сервер при необходимости получает необходимые данные от базы данных MySQL и возвращает клиенту ответ. В ответе могут быть данные в формате json или файлы. Несомненным преимуществом данной архитектуры является централизованное хранение данных, позволяющих любому пользователю получить к ним равный доступ, дающее возможность коммуникации пользователей через сервер.

2.2 Логическая модель базы данных

Для проектирования логической и физической модели базы данных выбрана программа Erwin Data Modeler. Данная программа предоставляет удобный функционал для разработки моделей на различных уровнях, имеет генератор кода, который можно использовать для создания реальной базы

данных. Поддерживаются различные базы данных, в том числе, MySQL, которая используется в данном проекте [10]. Логическая модель базы данных представлена на рисунке 8.

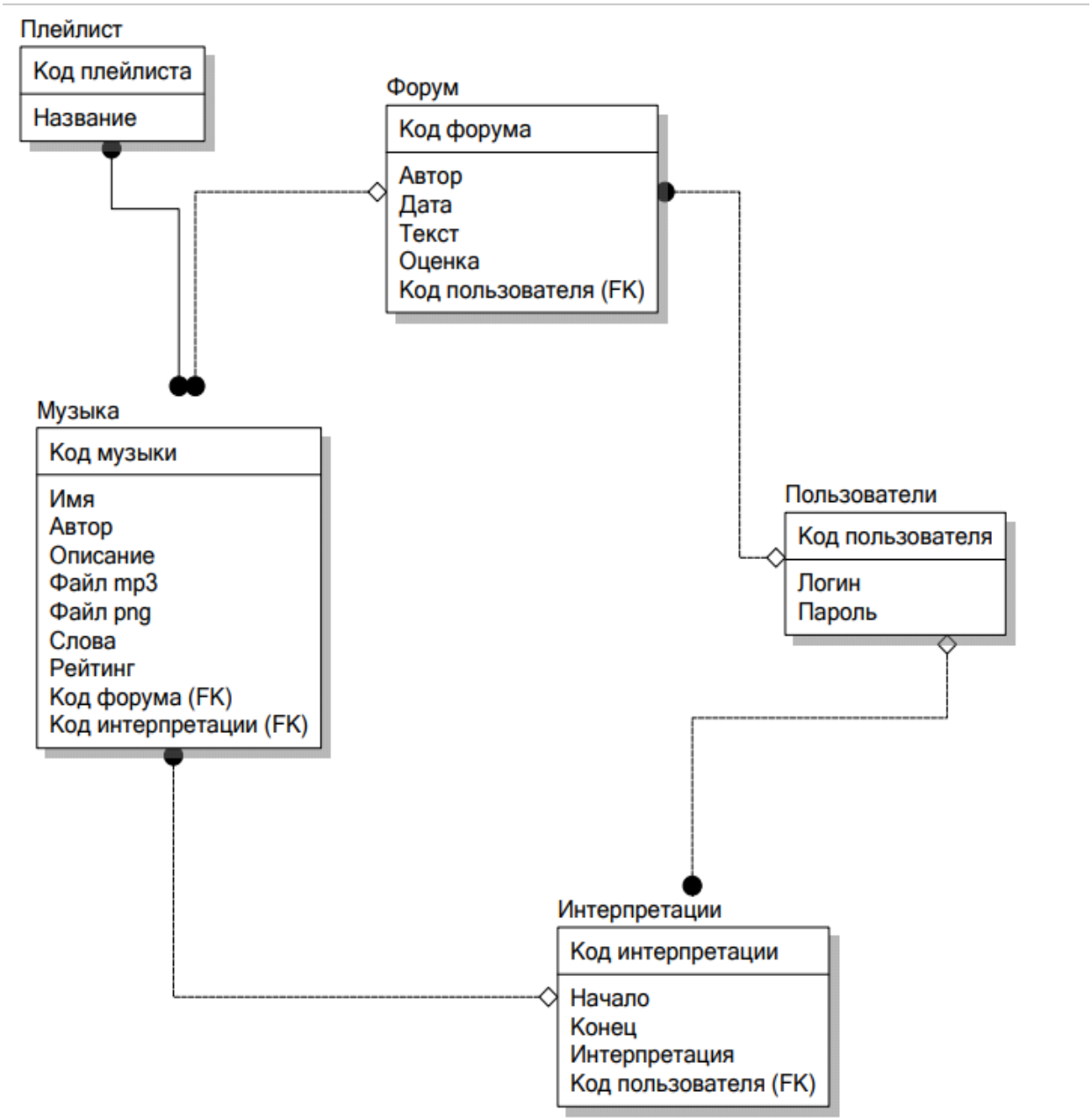

Рисунок 8 – Логическая модель базы данных

Логическая модель показывает выделенные сущности:

– «Музыка» – данные об аудио файле;

– «Плейлист» – список музыки;

– «Форум» – обсуждение музыки;

– «Интерпретации» – трактовка отдельных участков музыки;
– «Пользователи».

Далее осуществлен переход к физическому уровню.

2.3 Физическая модель базы данных

Физическая модель детализирует имена сущностей, типы данных, которые будут храниться непосредственно в базе данных.

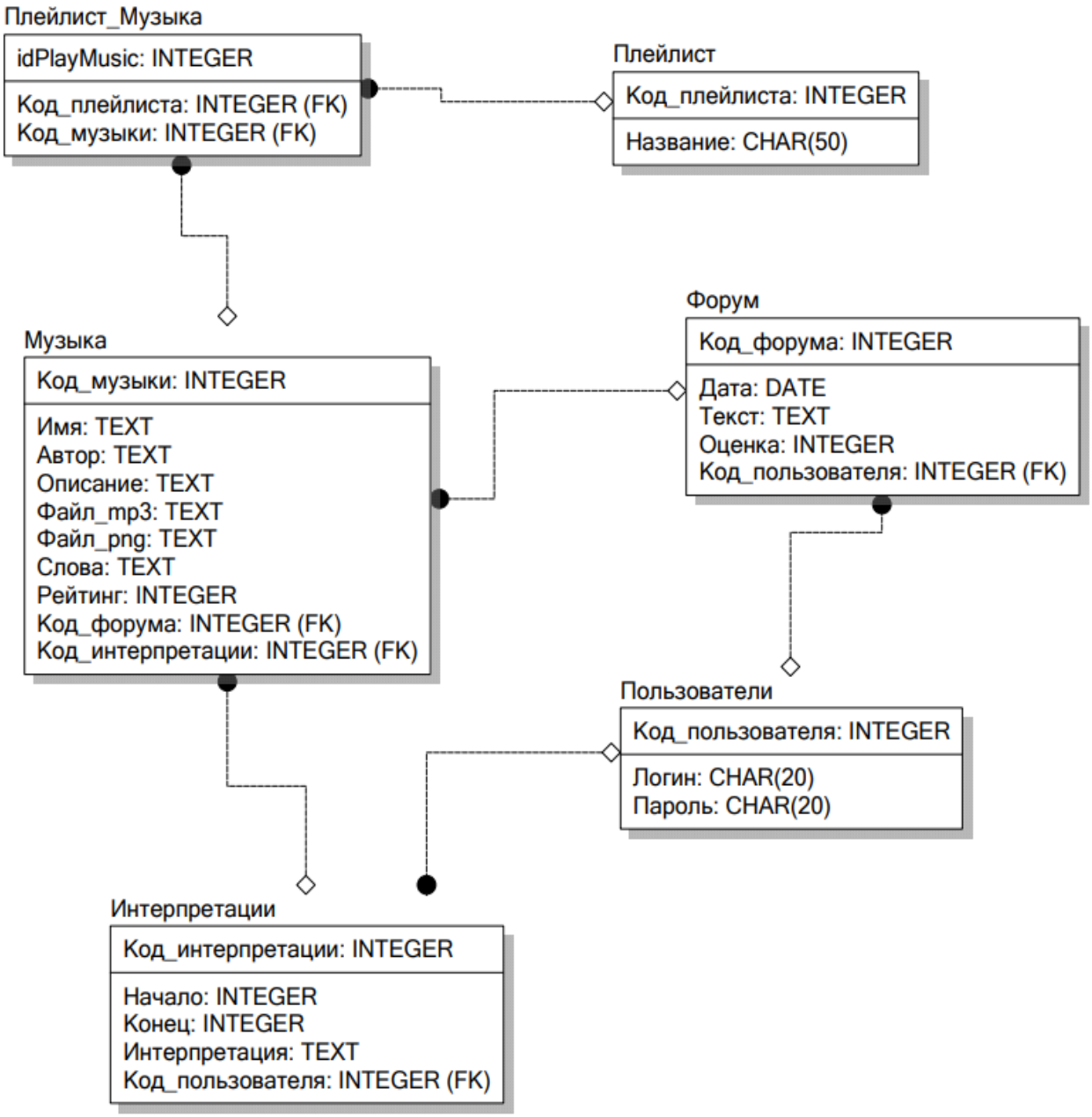

Рисунок 9 – Физическая модель базы данных

Также была детализирована связь многие ко многим между сущностью «Плейлист» и «Музыка». Для этого была создана вспомогательная сущность Плейлист_Музыка. Поскольку плейлист содержит всегда несколько композиций, но и музыка может входить в несколько плейлистов сразу.

2.4 Диаграмма классов

Для наглядности понимания внутренней структуры программы на рисунке 10 приведена диаграмма классов пользовательского приложения.

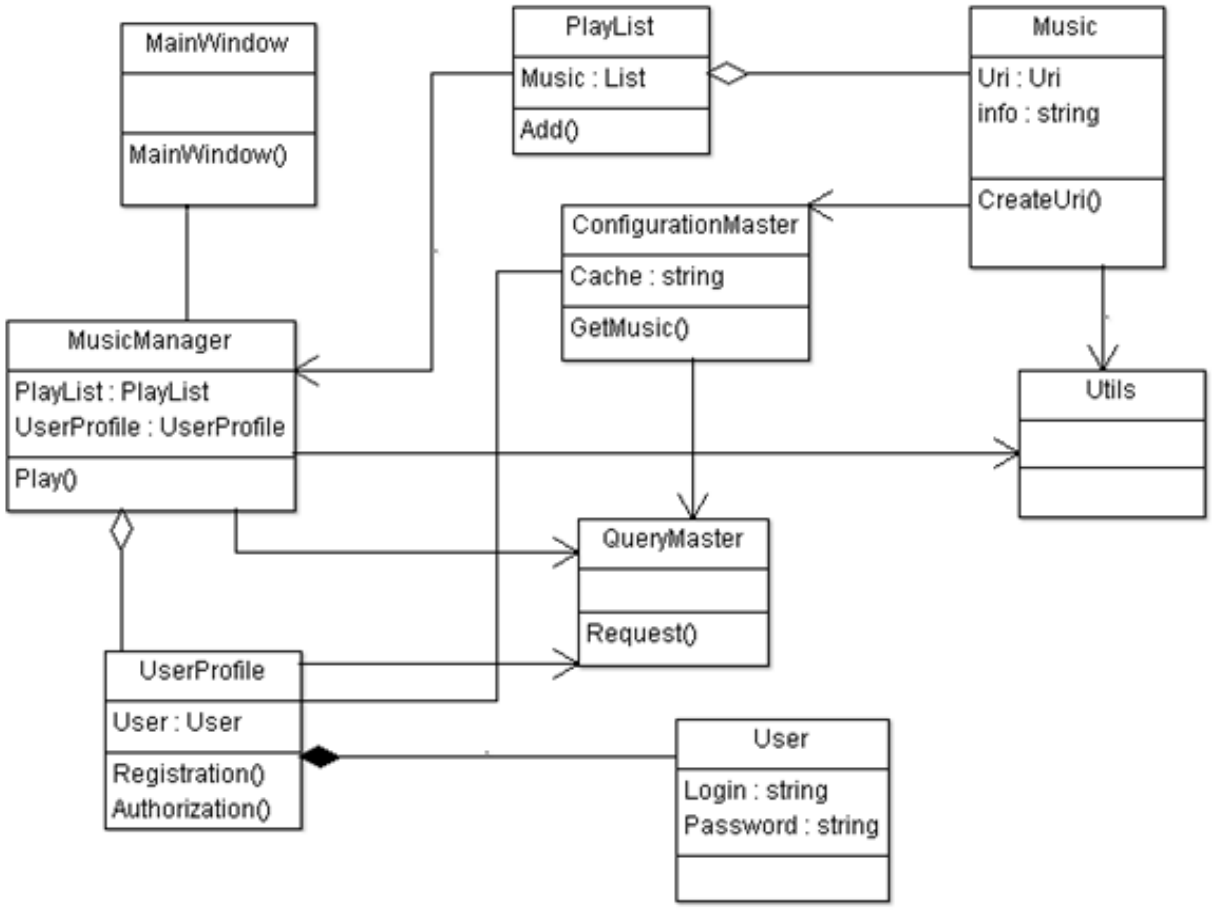

Рисунок 10 – Диаграмма классов пользовательского приложения

Описание классов приведено в таблице 4.

Таблица 4 – Описание классов

№	Класс	Описание
1	**2**	**3**
1	MainWindow	Точка входа в программу. Инициализирует пользовательское окно. Предоставляет пользователю интерфейс для взаимодействия с программой.
2	MusicManager	Содержит поля для воспроизведения музыки. Привязывается к элементам управления.
3	UserProfile	Содержит информацию о пользователе, музыке, которую он слушает.
4	User	Содержит информацию о логине и пароле пользователя.
5	PlayList	Содержит список музыки в плейлисте. Если плейлиста нет, то просто список музыки.
6	Music	Содержит информацию об аудио.
7	ConfigurationMaster	Содержит кеш аудио, выполняет поиск музыки по кешу или формирует запрос на сервер.
8	QueryMaster	Реализует различные запросы на сервер. Устанавливает соединение, возвращает информацию из Json-ответа, возвращает картинки и файлы.
9	Utils	Содержит вынесенные различные частые функции.

Таким образом, были описаны классы.

2.5 Диаграмма состояний

Показывает переход состояния системы из состояния в состояние при добавлении пользователем музыки на сервер (рисунок 11).

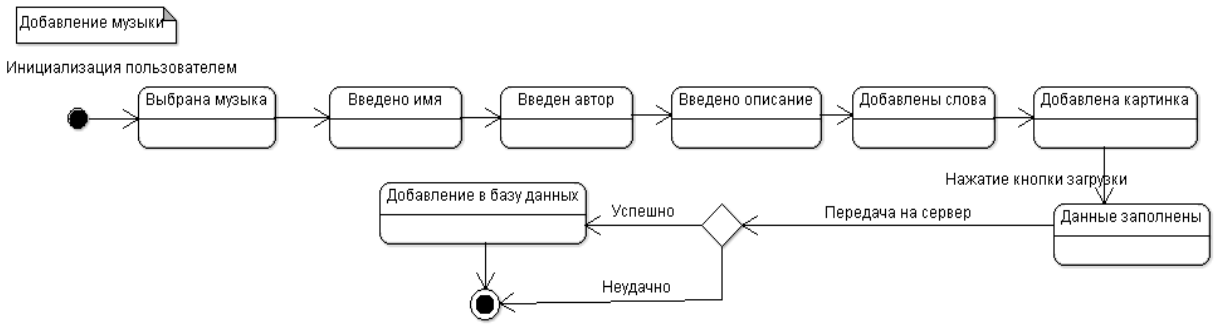

Рисунок 11 – Диаграмма последовательности клиент-серверного взаимодействия

Пользователь поочередно заполняет поля формы, далее нажимает на кнопку загрузки, осуществляется передача на сервер. При успешной передаче музыка добавляется в базу данных, иначе загрузка неудачно завершается.

2.6 Диаграмма последовательности

Далее на рисунке 12 представлена диаграмма последовательности, демонстрирующая клиент-серверное взаимодействие сервиса.

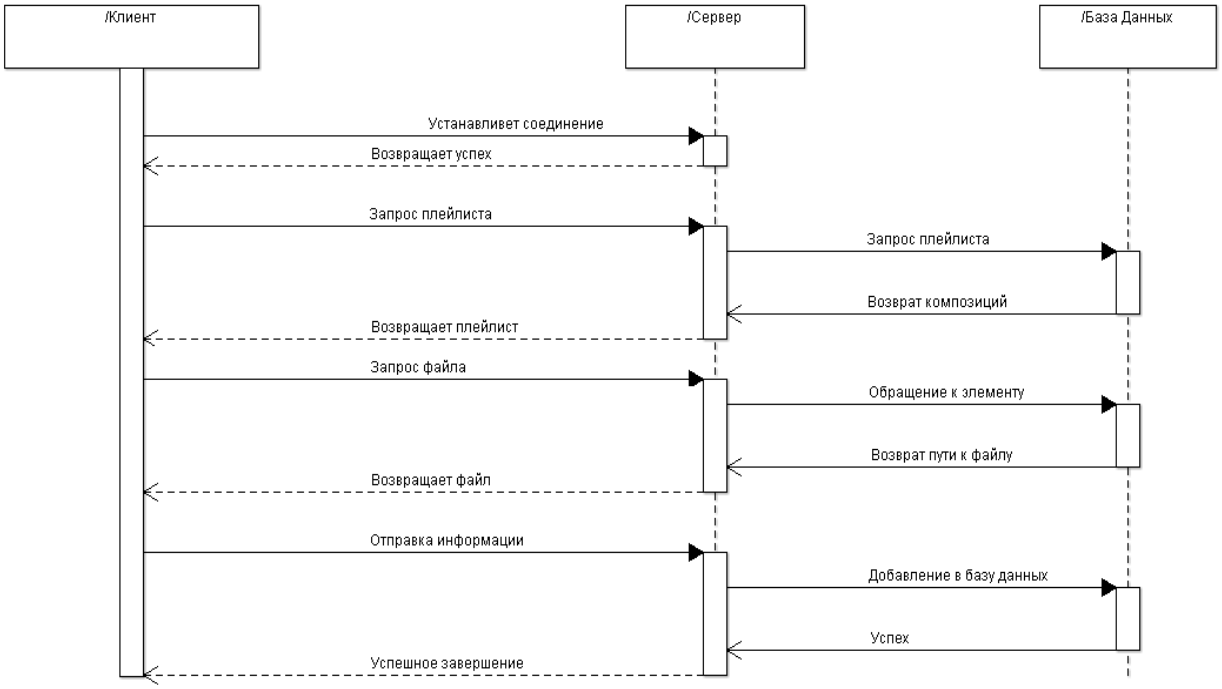

Рисунок 12 – Диаграмма последовательности клиент-серверного взаимодействия

Согласно вариантам использования, пользователю для работы с приложением необходимо взаимодействовать с сервером. Поэтому диаграмма показывает способ взаимодействия между клиентом и сервером, и между сервером и базой данных.

Дополнительно диаграмма наглядно представляет оптимизацию архитектуры приложения: загрузка плейлиста и аудио файлов, которые он содержит, разделена. Также для социальной составляющей сервер принимает информацию от каждого пользователя, сохраняет ее в базу данных, и через сервер пользователи могут взаимодействовать друг с другом.

Из структуры последовательностей можно сделать вывод, что при сбое какой-либо зависимости не произойдет нарушения работы всей системы в целом, потому что по обратной иерархии вызовов будет возвращена ошибка.

2.7 Диаграмма деятельности

Для понимания последовательности работы программы составлена диаграмма деятельности. Выделены три исполняемых «дорожки»: приложение, сервер, база данных (рисунок 13).

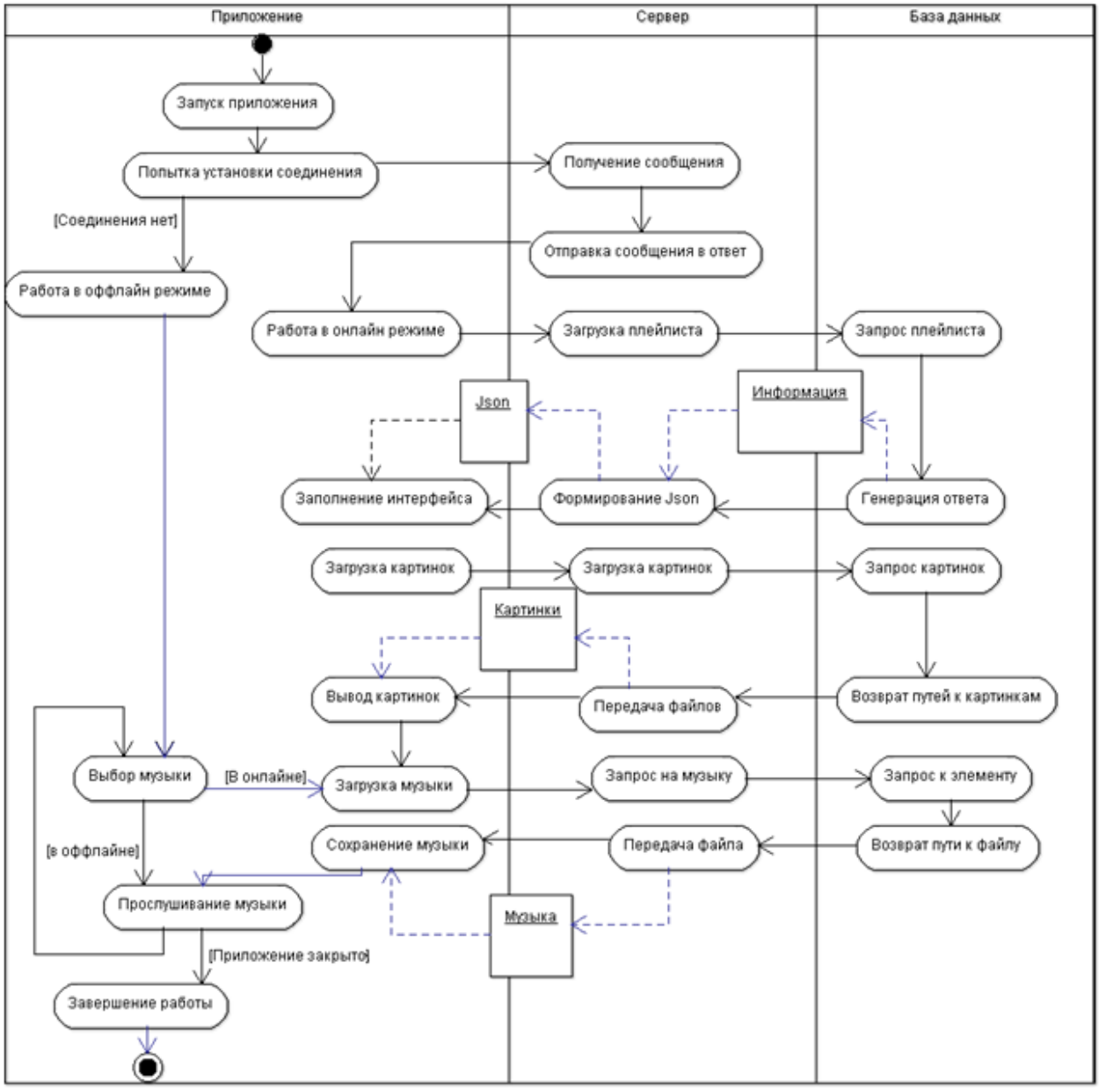

Рисунок 13 – Диаграмма деятельности сервиса

Диаграмма демонстрирует работу с сервисом. Присутствует ветвление, которое зависит режиме онлайн или оффлайн происходит работа с приложением. Поскольку онлайн пользователю доступна загрузка музыки с сервера. Присутствует цикл работы программы, условием выхода из которого является закрытие пользователем приложения. После выхода из цикла очищается память.

2.8 Диаграмма кооперации

Диаграмма кооперацией отображает взаимодействие между различными объектами. На рисунке 14 приведена диаграмма кооперации при добавлении комментария пользователем.

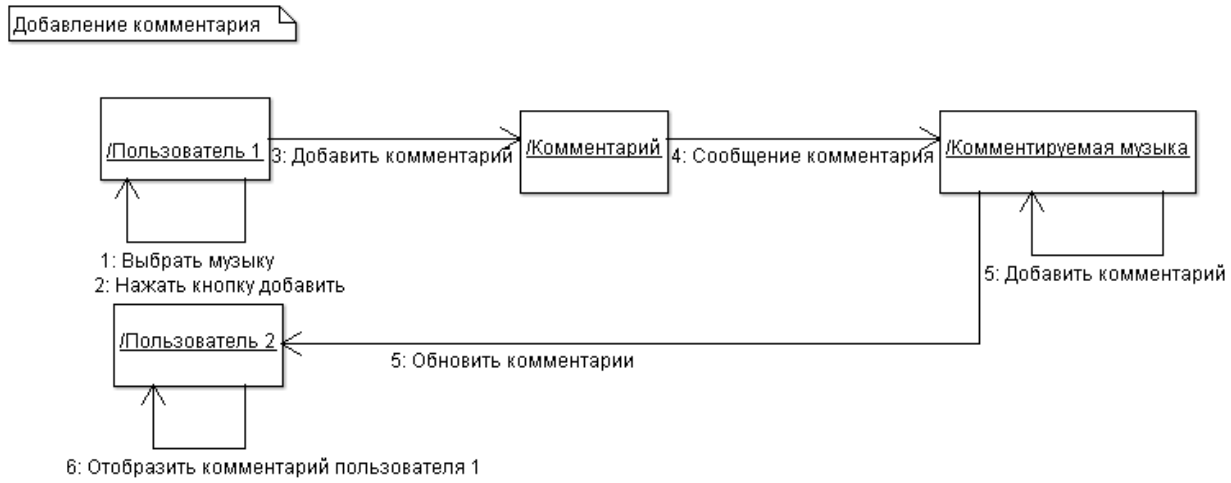

Рисунок 14 – Диаграмма кооперации

Диаграмма показывает взаимодействие между четырьмя объектами. Первый пользователь выбирает музыку, нажимает кнопку добавить. Далее первый пользователь добавляет комментарий, затем сообщает о нем на сервер, и комментируемой музыке добавляется комментарий. Далее второму пользователю приходит обновление комментариев, и комментарий первого пользователя отображается у второго.

Таким образом, диаграмма показывает механизм общения между пользователями через оставление комментариев к музыке.

3 Реализация программных средств

3.1 Реализация пользовательского интерфейса

В соответствии с анализом требований и определенными вариантами использования реализован пользовательский интерфейс на платформе WPF. При реализации интерфейса важным вопросом является его визуальный дизайн. Элементы управления должны быть привлекательными для пользователя. Для этого в качестве основных цветов выбраны контрастные цвета: светло-голубой и темно-малиновый [11]. Также было решено использовать интуитивно понятные иконки для изображения соответствующих вкладок и действий. Для дополнительной стилизации и отображения иконок использован пакет MaterialDesignThemes [12]. Интерфейс приложения представлен на рисунке 15.

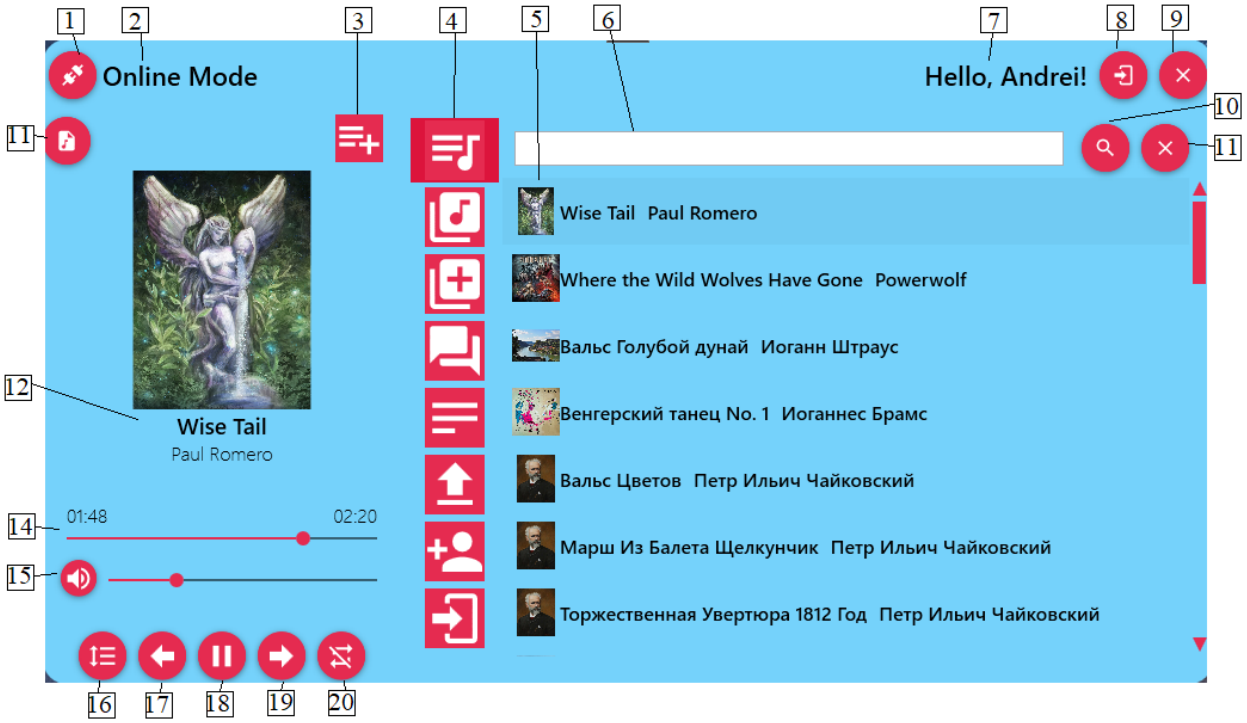

Рисунок 15 – Интерфейс приложения

Основными элементами управления являются:

1 – кнопка установки соединения с сервером и перехода из оффлайн режима в онлайн;

2 – текстовое поле, информирующее о режиме работы сервиса;

3 – кнопка добавления аудио в плейлист, работа представлена на рисунке 16;

4 – элемент управления TabControl, обеспечивающий работу с несколькими вкладками, по умолчанию активна вкладка с текущим прослушиваемым плейлистом;

5 – элемент управления ListBox, привязываемый к коллекции списка аудио активного плейлиста, при выборе элемента списка начинается его воспроизведение;

6 – поле для ввода ключевых слов поиска музыки;

7 – приветствующее текстовое поле, приветствие определяется логином пользователя;

8 – кнопка адресации на вкладку авторизации;

9 – кнопка завершения работы приложения;

10 – кнопка поиска по ключевым словам;

11 – кнопка сброса фильтра по ключевым словам;

12 – кнопка выбора аудио файла из файловой системы компьютера пользователя;

13 – панель вывода информации об исполняемой композиции;

14 – элемент управления, отвечающий за информирование о текущем времени воспроизведения и управление им;

15 – элементы управления звуком: кнопка переключает звук, а ползунок регулирует громкость;

16 – кнопка перемешивания воспроизведения композиций в текущем плейлисте, ее работа представлена на рисунке 17;

17 – кнопка воспроизведения предыдущего файла;

18 – кнопка воспроизведения, если неактивно воспроиведение, либо же паузы при активном воспроизведении;

19 – кнопка воспроизведения следующего файла;

20 – кнопка повтора текущей композиции.

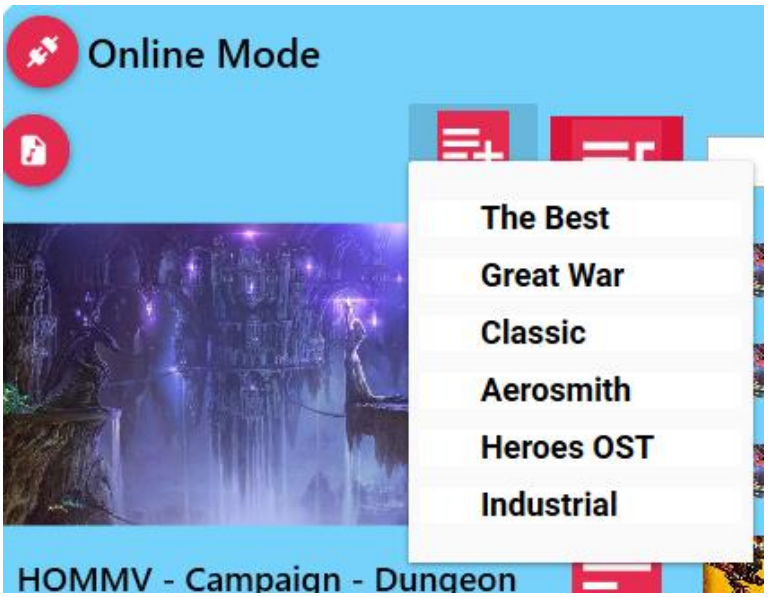

Рисунок 16 – Работа кнопки добавления в плейлист

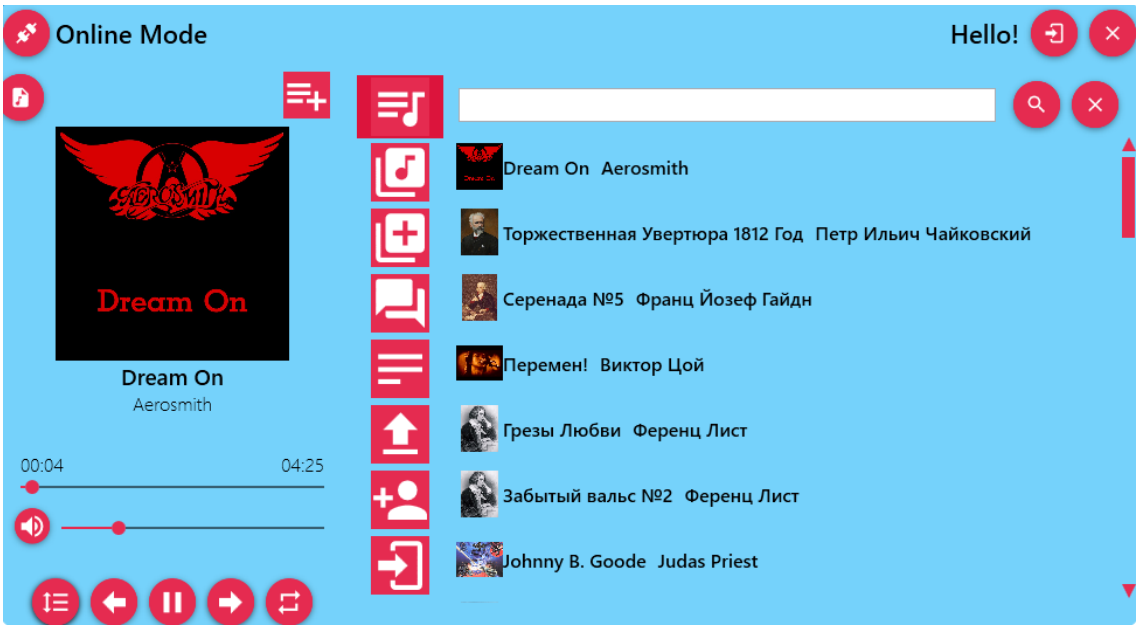

Рисунок 17 – Работа кнопки перемешивания

Далее рассмотрим следующие вкладки. Вкладка плейлисты представлена на 18 рисунке. Элементами управления являются:

1 – кнопка вкладки плейлистов;

2 – таблица плейлистов, каждый элемент таблицы содержит обложку плейлиста, название и исполнителя, при нажатии на элемент таблицы начинается воспроизведение плейлиста.

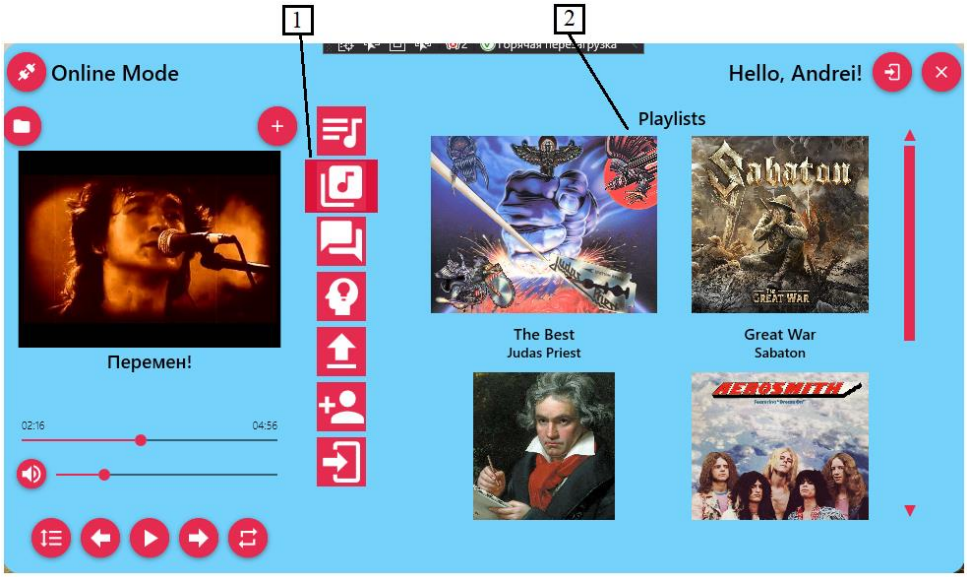

Рисунок 18 – Интерфейс вкладки плейлистов

Следующая вкладка позволяет создавать новые плейлисты, она представлена на 19 рисунке. Элементами управления являются:

1 – кнопка вкладки создания плейлиста;

2 – изображение обложки плейлиста;

3 – кнопка поиска изображения на компьютере пользователя;

4 – название плейлиста;

5 – автор плейлиста;

6 – кнопка добавить плейлист.

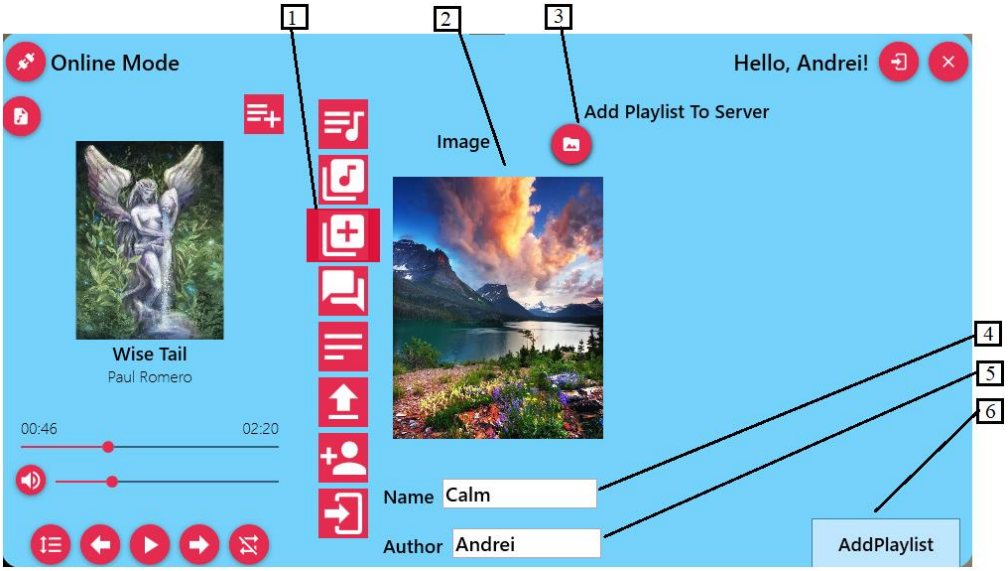

Рисунок 19 – Интерфейс вкладки создания плейлиста

Рассмотрим вкладку обсуждения. Она представлена на рисунке 20. Элементами управления являются:

1 – кнопка вкладки обсуждения;

2 – список комментариев к активной композиции;

3 – кнопка ответить на комментарий;

4 – текстовое поле для ввода пользователем своего комментария;

5 – кнопка добавить комментарий.

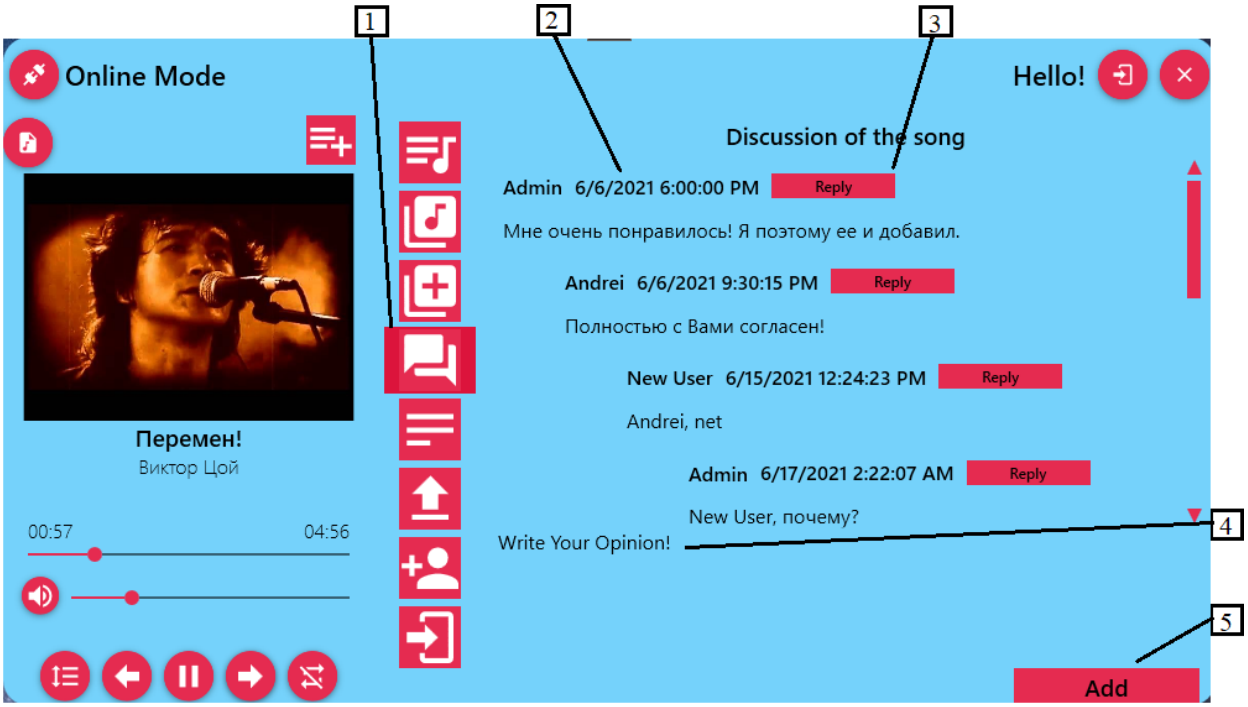

Рисунок 20 – Интерфейс вкладки обсуждения

Рассмотрим вкладку интерпретаций. Она реализована в двух вкладках, представлена на рисунках 21 и 22. Элементами управления являются:

1 – кнопка вкладки интерпретаций;

2 – вкладка лирики, содержит слова песни;

3 – вкладка интерпретаций, содержит список пользовательских интерпретаций к данной песне;

4 – слова песни;

5 – текстовое поле для ввода пользователем своей интерпретации;

6 – интерпретируемые слова;

7 – кнопка добавить интерпретацию;

8 – список пользовательских интепретаций.

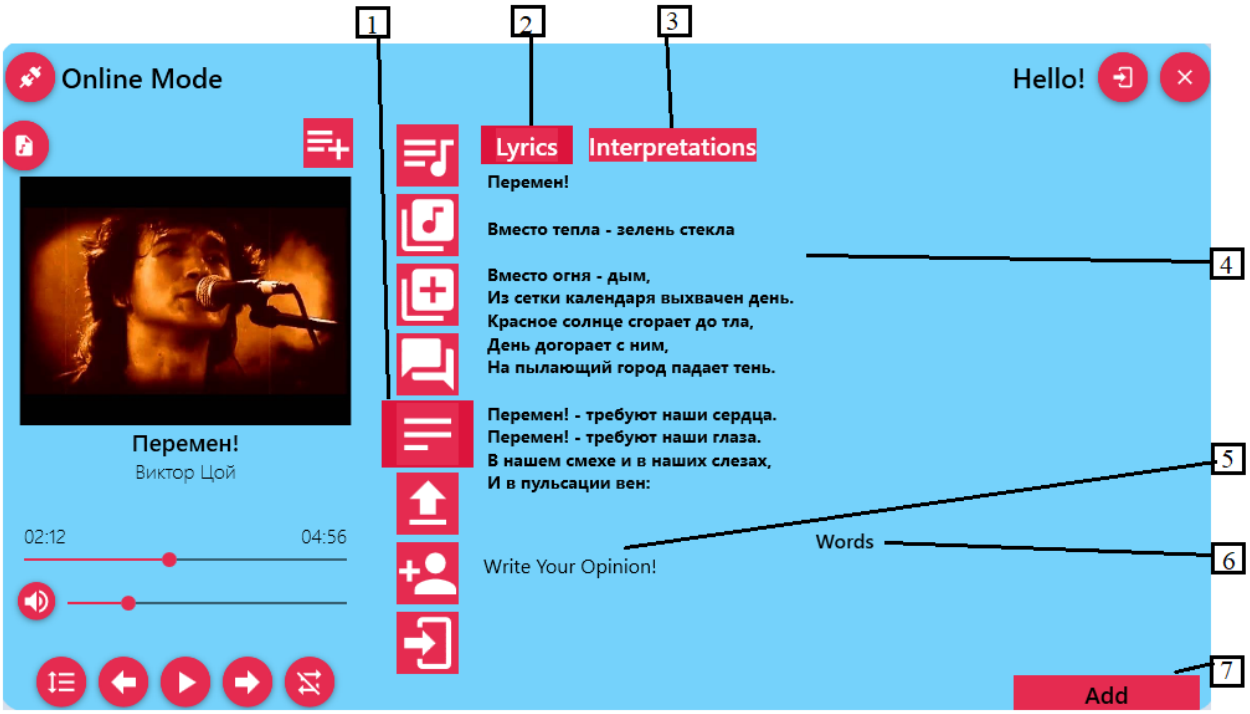

Рисунок 21 – Интерфейс вкладки интерпретации

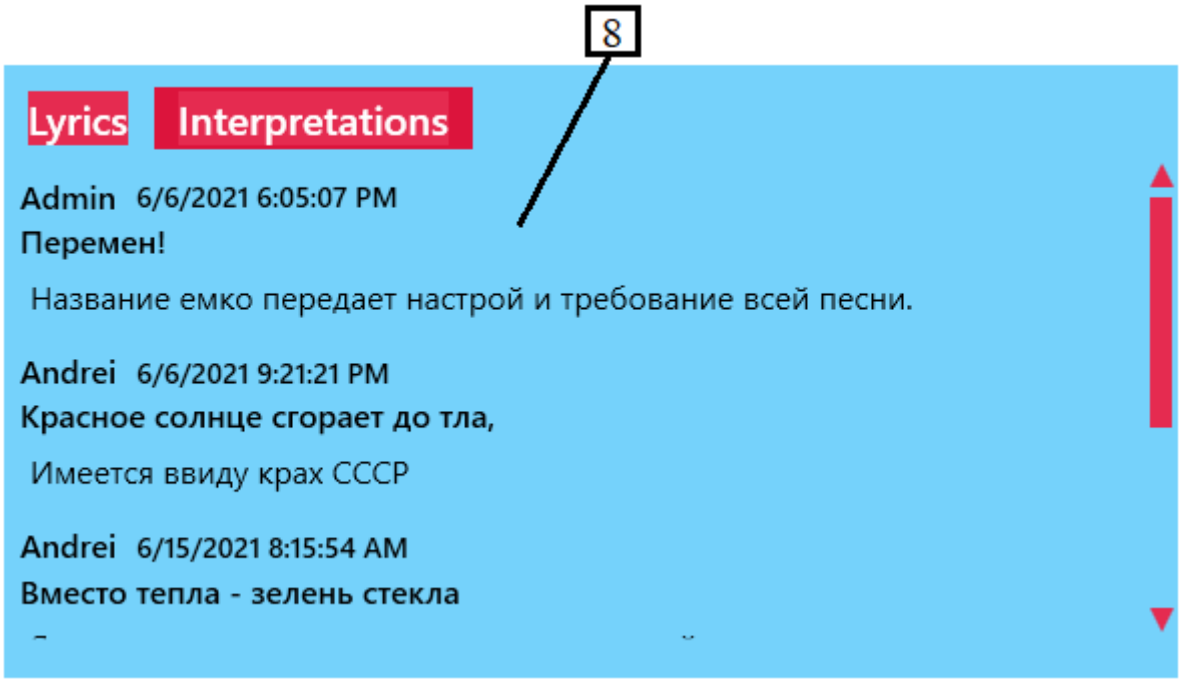

Рисунок 22 – Интерфейс вкладки плейлистов

Рассмотрим вкладку загрузки музыки на сервер. Она представлена на рисунке 23. Элементами управления являются:

1 – изображение, которое соответствует музыке;

2 – кнопка, вызывающая диалоговое окно выбора файла изображения на компьютере пользователя;

3 – поле для добавления слов к песне;

4 – поле для ввода название файла;

5 – поле для ввода автора;

6 – поле для ввода описания;

7 – кнопка загрузки на сервер;

8 – кнопка вкладки загрузки музыки на сервер.

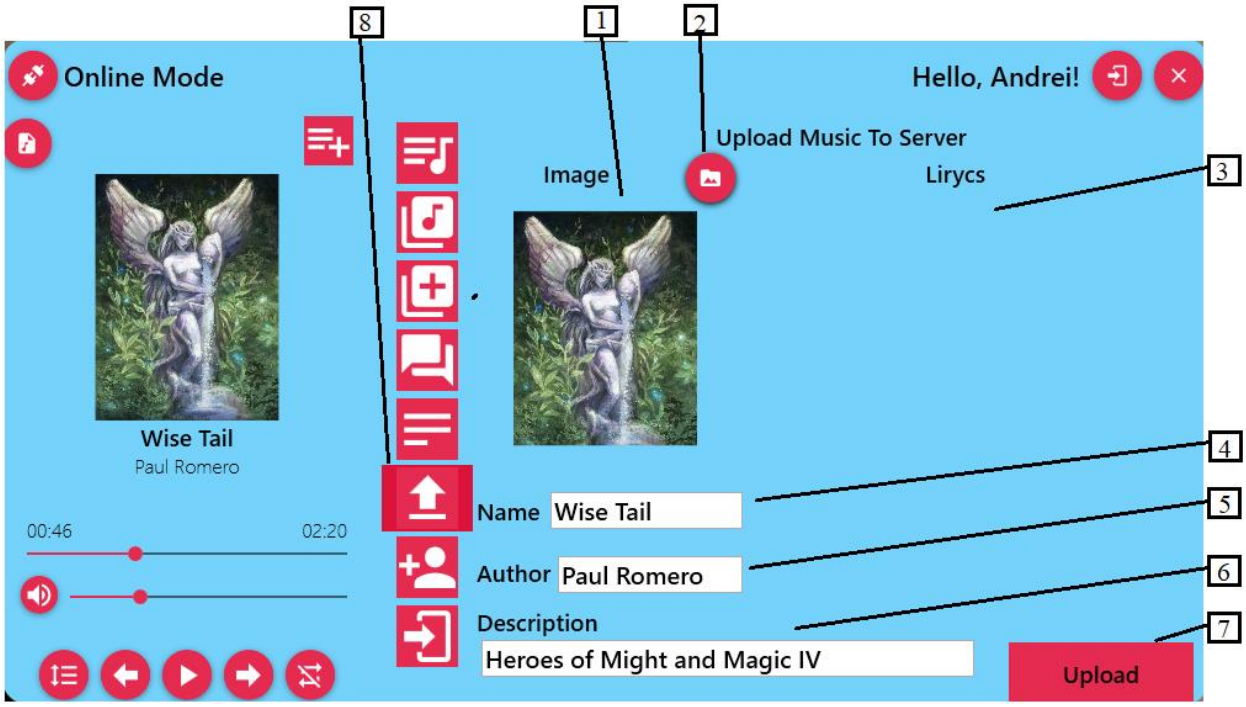

Рисунок 23 – Интерфейс вкладки загрузки на сервер

Рассмотрим вкладку регистрации. Она представлена на рисунке 24. Элементами управления являются:

1 – кнопка вкладки регистрации;

2 – поле ввода логина;

3 – поля ввода пароля;

4 – кнопка регистрации;

5 – при успешной регистрации пользователь автоматически авторизируется в приложении и приветственная информация обновляется.

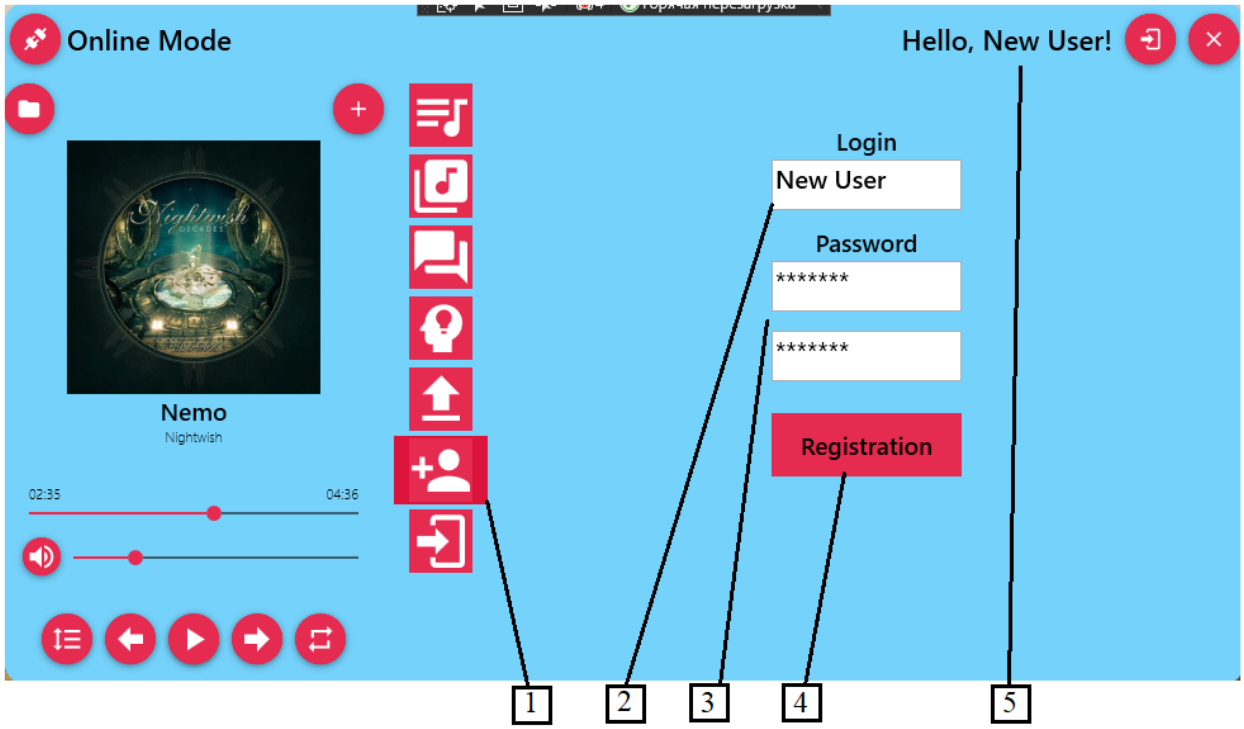

Рисунок 24 – Интерфейс вкладки регистрации

Рассмотрим вкладку авторизации. Она представлена на рисунке 25. Элементами управления являются:

1 – кнопка вкладки авторизации;

2 – кнопка навигации во вкладку регистрации;

3 – кнопка авторизации;

4 – поле ввода пароля;

5 – поле ввода логина;

6 – при успешной авторизации обновляется текст приветствия.

Вкладка содержит кнопку навигации во вкладку регистрации, хотя над кнопкой вкладки авторизации находится аналогичная кнопка, для того, что бы пользователь имел несколько возможностей сделать одно действие. Такой подход позволяет упростить пользователю работу с интерфейсом и делает его интуитивно понятным.

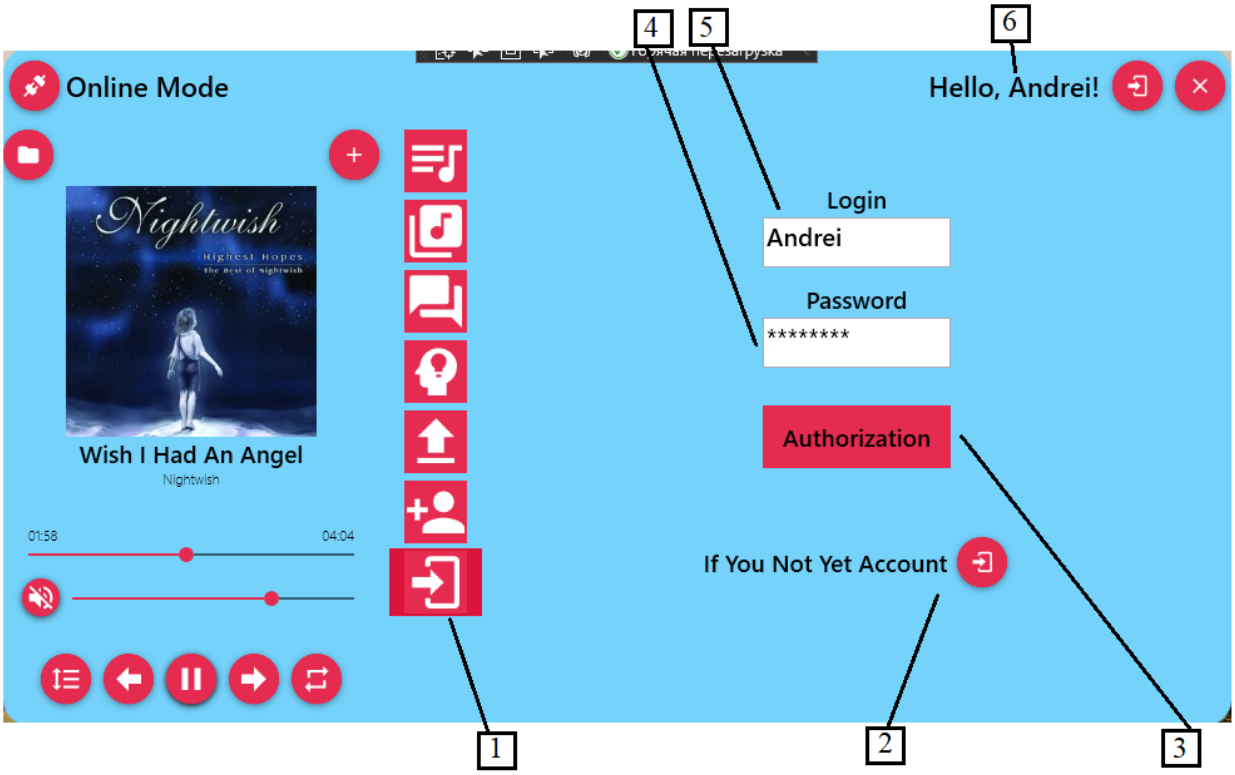

Рисунок 25 – Интерфейс вкладки авторизации

В дополнение к обзору интерфейса следует указать, что каждая кнопка действия в приложении сопровождается всплывающей надписью, которая описывает реализуемую функцию.

Таким образом, получившийся графический интерфейс удовлетворяет установленным требованиям, является визуально привлекательным и удобным для пользователя. Интерфейс реализован в xaml-разметке.

3.2 Диаграмма компонентов клиентского приложения

Диаграмма компонентов показывает взаимосвязь всех элементов проекта. Компонентами являются файлы с расширением .cs – файлы с кодом на языке C# – и .xaml – файлы с языком расширяемой разметки приложений, описывают интерфейс и реализуют привязку данных контролов к источникам данных, иначе это можно назвать поддержкой MVVM подхода. Диаграмма приведена на рисунке 26. Описание компонентов приведено в таблице 5.

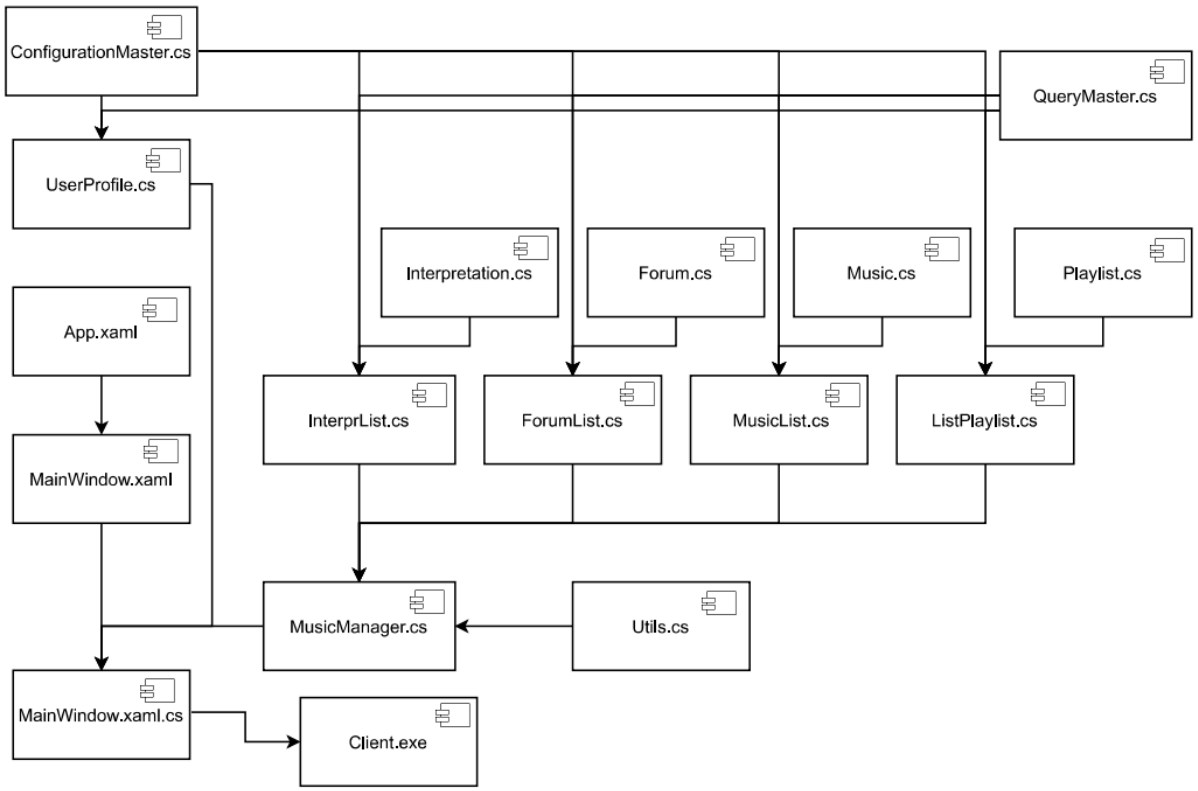

Рисунок 26 – Диаграмма компонентов клиентского приложения

Компонентами клиентского приложения, в основном, являются классы, необходимые для работы программы. Описание компонентов приведено в таблице 5.

Таблица 5 – Список компонентов

№	Компонент	Описание
1.	ConfigurationMaster.cs	Реализует класс ConfigurationMaster.
2.	QueryMaster.cs	Реализует класс QueryMaster. Управляет запросами к серверу.
3.	Interpretation.cs	Реализует класс Interpretation. Абстракция интерпретации.
4.	Forum.cs	Реализует класс Forum. Абстракция комментария.
5.	Music.cs	Реализует класс Music. Абстракция музыки.
6.	Playlist.cs	Реализует класс Playlist. Абстракция плейлиста.
7.	InterprList.cs	Реализует класс InterprList. Список интерпретаций.
8.	ForumList.cs	Реализует класс ForumList. Список комментариев.
9.	MusicList.cs	Реализует класс MusicList. Список музыки.
10.	ListPlaylist.cs	Реализует класс ListPlaylist. Список плейлистов.
11.	Utils.cs	Реализует различные общие функции.
12.	MusicManager.cs	Реализует класс MusicManager. Основной класс логики приложения.

Продолжение таблицы 5

13.	UserProfile.cs	Реализует класс UserProfile. Абстракция пользователя.
14.	App.xaml	Содержит пользовательские стили контролов.
15.	MainWindow.xaml	Содержит описание интерфейса и определение привязок данных.
16.	MainWindow.xaml.cs	Содержит точку входа в приложение. Инициализирует представление, логику и их связь.
17.	Client.exe	Основной исполняемый файл.

Таким образом, описана диаграмма компонентов клиентского приложения.

3.3 Диаграмма развёртывания

Диаграмма развёртывания демонстрирует непосредственное взаимодействие физических компонентов при работе всей системы. Диаграмма приведена на рисунке 27.

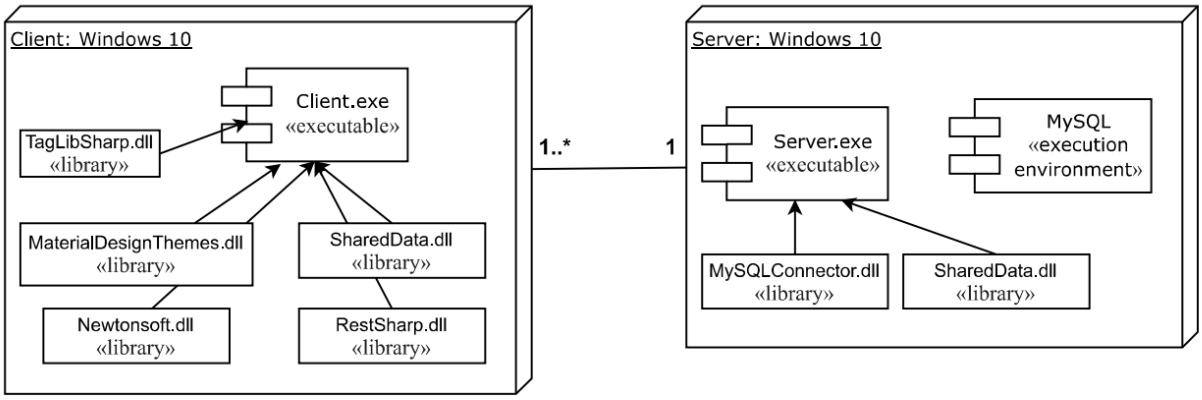

Рисунок 27 – Диаграмма развёртывания

На диаграмме можно наблюдать классическое клиент-серверное взаимодействие. Один сервер обслуживает множество клиентских приложений, которые настроены на его адрес в сети.

Сервер развертывается с помощью Internet Information Services (IIS). Службы IIS позволяют публиковать серверные части на ОС Windows 10. IIS являются для Windows стандартными средствами. Службы разработаны корпорацией Microsoft [8].

Сервер содержит контроллеры для обработки запросов. Запрос поступает на контроллер. Контроллер извлекает из запроса необходимую информацию, формирует запрос к MySQL [15]. По полученным данным от БД контроллер формирует ответ пользователю.

Клиентское и серверное приложение в своей работе ссылается на свободно распространяемые NuGet-пакеты. Пакеты - это распространяемые в стандартизированном виде библиотеки, поэтому физически приложение ссылается на dll-библиотеки. Описание пакетов и библиотек приведено ниже.

MP3-файлы содержат в себе помимо информации о звуковой дорожке теги, содержащие название, автора, описание и другую информацию. Для упрощения извлечения тегов из файлов использован пакет TagLibSharp [13].

Клиент отправляет запросы на сервер с помощью вспомогательного пакета RestSharp. Данный пакет упрощает формирование запроса и получение ответа [14].

Библиотека SharedData.dll – представляет собой описание передаваемых от клиента к серверу и в обратном порядке классов и структур. Необходимо описание общих классов для того, что бы передача данных была универсализирована.

Пакет Newtonsoft используется для сереализации в Json-формат структур SharedData для их дальнейшего добавления в тело запроса или ответа. После передачи необходим для десереализации в объект-экземпляр класса.

Пакет MySQLConnector упрощает установку соединений и выполнение команд к СУБД MySQL. Используется сервером для создания запросов к БД.

Подводя итог, в данной главе была описана реализация сервиса.

4 Тестирование программного продукта

Тестирование необходимо для контроля качества проекта. В процессе написания кода, так или иначе, неизбежны какие-либо неисправности и некорректная работа. Поэтому для уверенности в том, что пользователь получит качественное, стабильное и удобное приложение и что вероятность критических ситуаций минимальна, необходима система контроля разработки, которая включает в себя тестирование.

4.1 Структурное тестирование

Структурное тестирование необходимо для контроля программы на каждом шаге. Рассмотрим алгоритм загрузки нового аудио-файла на сервер. Для этого создадим блок-схему данного алгоритма, рисунок 28.

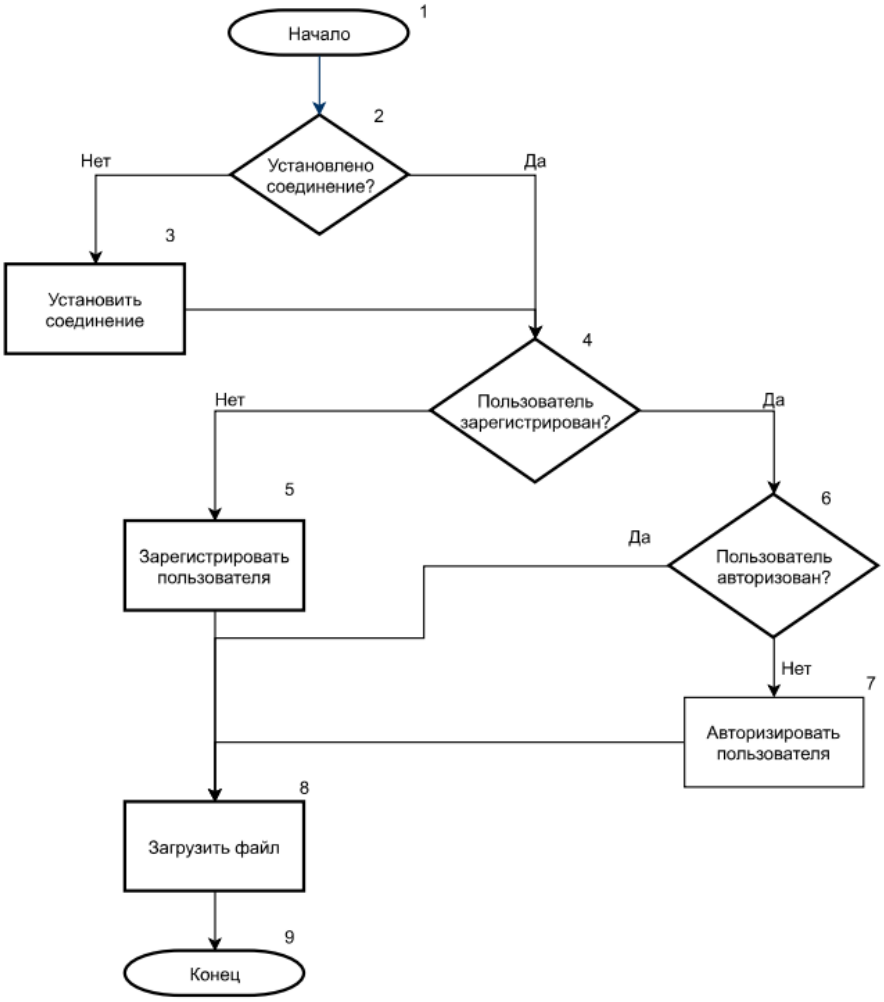

Рисунок 28 – Блок-схема «Загрузить аудио на сервер»

4.1.1 Метод покрытия операторов

Тестирование методом покрытия операндов означает, что каждый операнд будет протестирован как минимум один раз. Составлены тесты, которые обеспечат покрытие всех возможных случаев работы с данным вариантом использования. Тесты определены в таблице 6.

Таблица 6 – Результат тестирования

№	Операторы	Тест	Фактический результат	Результат тестирования
1	3, 5, 8	Не установлено соединение. Пользователь не зарегистрирован. Загружен аудио файл на сервер.	Успешная загрузка файла.	Успешно
2	5, 7, 8	Установлено соединение. Пользователь зарегистрирован. Пользователь не авторизирован. Загружен аудио файл на сервер.	Успешная загрузка файла.	Успешно

Выполнено успешное тестирование методом покрытия операндов.

4.1.2 Метод покрытия условий

Тестирование методом покрытия условий означает, что все комбинации результатов условий будут протестированы, плюс необходимо соблюдать условие покрытия операндов. Тесты определены в таблице 7.

Таблица 7 – Результат тестирования

№	Условия	Тест	Фактический результат	Результат
1	2 – нет	Соединение не установлено.	Переход к установке соединения.	Успешно
2	2 – да, 4 – нет	Соединение установлено. Пользователь не зарегистрирован.	Переход к регистрации.	Успешно
3	2 – да, 4 – да, 6 – нет	Соединение установлено. Пользователь зарегистрирован. Пользователь не авторизирован.	Переход к авторизации.	Успешно
4	2 – да, 4 – да, 6 – да	Соединение установлено. Пользователь зарегистрирован. Пользователь авторизирован.	Переход к загрузке файла.	Успешно

По итогам тестов можно говорить о корректной работе алгоритма.

4.1.3 Метод покрытия маршрутов

Тестирование методом покрытия условий означает, что для каждого возможного маршрута будет составлен тест. Тесты определены в таблице 8.

Таблица 8 – Результат тестирования

№	Маршрут	Тест	Фактический результат	Результат
1	1, 2, 4, 5, 8	Соединение установлено. Пользователь не зарегистрирован. Пользователь регистрируется. Пользователь загружает файл.	Файл загружен.	Успешно.
2	1, 2, 3, 4, 5, 8	Соединение не установлено. Установка соединения. Пользователь не зарегистрирован. Пользователь регистрируется. Пользователь загружает файл.	Файл загружен.	Успешно.
3	1, 2, 4, 6, 8	Соединение установлено. Пользователь зарегистрирован, авторизован. Пользователь загружает файл.	Файл загружен.	Успешно.
4	1, 2, 3, 4, 6, 8	Соединение не установлено. Установка соединения. Пользователь зарегистрирован, авторизован. Пользователь загружает файл.	Файл загружен.	Успешно.
5	1, 2, 4, 6 , 7, 8	Соединение установлено. Пользователь зарегистрирован, не авторизован. Пользователь авторизируется, загружает файл.	Файл загружен.	Успешно.
6	1, 2, 3, 4, 6, 7, 8	Соединение не установлено. Установка соединения. Пользователь зарегистрирован, не авторизован. Пользователь авторизируется, загружает файл.	Файл загружен.	Успешно.

По итогам тестов можно говорить о корректной работе алгоритма.

4.2 Функциональное тестирование

Для социального взаимодействия пользователей необходимо реализовать регистрацию по логину и паролю. Логин – уникальное имя пользователя в системе. Пароль – секретная комбинация символов для

доступа пользователя к своему аккаунту. При проектировании установлены следующие требования:

- длины логина и пароля не менее 5 и не более 20 символов;
- логин должен быть уникальным;
- логин и пароль могут содержать в себя любые символы;
- пароль должен содержать символы заглавного регистра;
- пароль должен содержать символы малого регистра;
- пароль должен содержать цифры.

В зависимости от выполнения условий, должно выдаваться сообщение:

- длина логина неверная;
- длина пароля неверная;
- данный логин занят;
- пароль не содержит заглавных символов;
- пароль не содержит малых символов;
- пароль не содержит цифр;
- регистрация успешно завершена.

4.2.1 Классы эквивалентности

Определение классов эквивалентности необходимо для системного подхода в тестировании – выделение параметров, которые в совокупности тестируются. Ниже представлена таблица 9, в которой собрана вся информация о классах эквивалентности. В таблице 10 приведены тесты.

Таблица 9 – Классы эквивалентности

Входные условия	**Правильные классы эквивалентности**	**Неправильные классы эквивалентности**
Длина логина	[3, 20]	< 3, > 20
Длина пароля	[3, 20]	< 3, > 20
Уникальность логина	Логин уникален	Логин занят
Состав пароля	Включает заглавный и строчный регистр и цифры	Не включает заглавный или строчный регистр или цифры

Таблица 10 – Тесты классов эквивалентности

№	Входные данные	Тест	Фактический результат	Результат
1	Логин: Admin	Длина логина	Успешная регистрация.	Успешно
2	Пароль: Qwerty1234	Длина пароля	Успешная регистрация.	Успешно
3	Логин: Andrei	Уникальность логина	Ошибка при регистрации. (неуникальный логин)	Успешно
4	Пароль: Qwerty1234	Состав пароля	Успешная регистрация.	Успешно

Описаны классы эквивалентности и проведено их успешное тестирование.

4.2.2 Анализ граничных условий

Описание граничных условий регистрации приведено в таблице 11, в таблице 12 приведены тесты.

Таблица 11 – Граничные условия

Тип границы	Длина логина	Длина пароля
Физическая	Не менее 0 (1)	Не менее 0 (2)
Логическая	Нет	Нет
Технологическая	Не менее 0. Не более 256 символов	Не менее 0. Не более 256 символов
Произвольная	От 3 до 20	От 3 до 20

Таблица 12 – Тесты граничных условий

№	Входные данные	Тест	Фактический результат	Результат
1	Логин: Alexey	Внутри диапазона	Успешная регистрация.	Успешно
2	Логин: Admin	На границе	Успешная регистрация.	Успешно
3	Логин: Aa	За границей	Ошибка при регистрации. (некорректный логин)	Успешно
4	Пароль: 12	За границей	Ошибка при регистрации. (некорректный пароль)	Успешно

Потребность в анализе граничных условий возникает при тестировании классов эквивалентности, в которых значения должны быть в определенном интервале. Выявленные граничные условия представлены на рисунке 29 и 30.

Рисунок 29 – Границы для длины логина и пароля

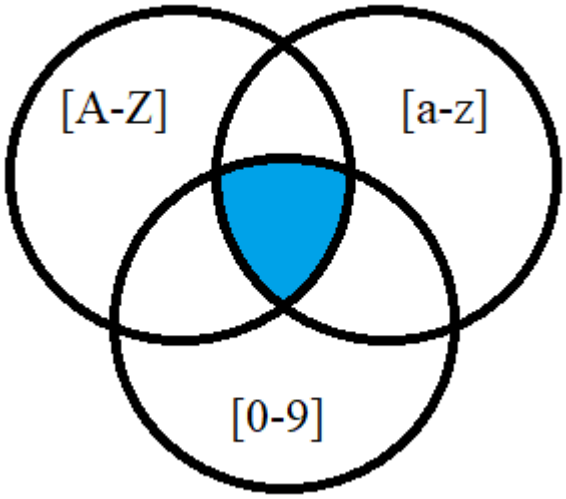

Рисунок 30 – Границы для состава пароля

Произведен анализ граничных условий. Далее расмотрена функциональная диаграмма.

4.2.3 Функциональная диаграмма

Функциональная диаграмма описывает возможные следствия при различных комбинациях входных причин.

При регистрации необходима проверка входных параметров, поскольку на логин и пароль наложены ограничения. Соответственно при нарушении ограничений программа должна выводить уведомления пользователю о соответствующем нарушении. На рисунке 31 представлена функциональная диаграмма регистрации.

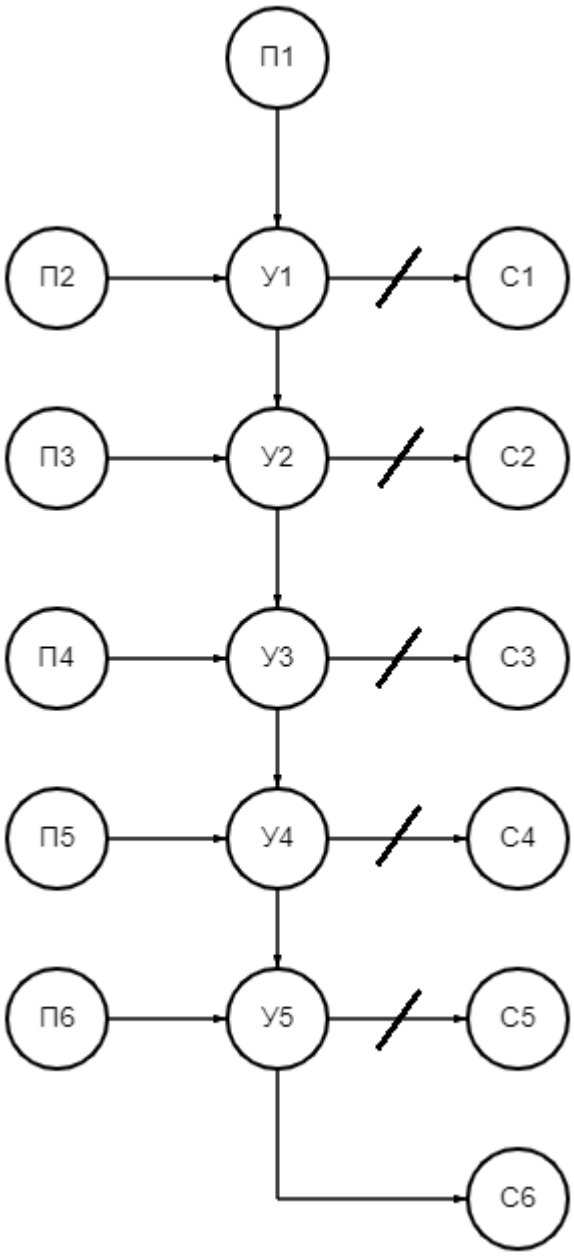

Рисунок 31 – Функциональная диаграмма регистрации

Описание причин и следствий представлено на таблицах 13 и 14.

Таблица 13 – Описание причин

Причина	**Описание**
1	Корректная длина логина
2	Корректная длина пароля
3	Логин уникален
4	Пароль содержит символы заглавного регистра
5	Пароль содержит символы строчного регистра
6	Пароль содержит цифры

Таблица 14 – Описание следствий

Следствие	Описание
1	Сообщение: «Некорректная длина логина»
2	Сообщение: «Некорректная длина пароля»
3	Сообщение: «Логин занят»
4	Сообщение: «Пароль не содержит заглавный регистр»
5	Сообщение: «Пароль не содержит строчный регистр»
6	Сообщение: «Пароль не содержит цифры»
7	Сообщение: «Регистрация успешно завершена»

По функциональной диаграмме составлена таблица решений. Она представлена на таблице 15. Таблица решений включает в себя входные причины и результаты.

Таблица 15 – Таблица решений

Тесты	1	2	3	4	5	6	7
П1	0	1	1	1	1	1	1
П2	-	0	1	1	1	1	1
П3	-	-	0	1	1	1	1
П4	-	-	-	0	1	1	1
П5	-	-	-	-	0	1	1
П6	-	-	-	-	-	0	1
С1	1	0	0	0	0	0	0
С2	0	1	0	0	0	0	0
С3	0	0	1	0	0	0	0
С4	0	0	0	1	0	0	0
С5	0	0	0	0	1	0	0
С6	0	0	0	0	0	1	0
С7	0	0	0	0	0	0	1

По таблице решений проведены тесты (таблица 16).

Таблица 16 – Таблица тестирования

№	Входные данные	Ожидаемый результат
1	Логин: Ada, Пароль: Qwerty1234	С1
2	Логин: Random, Пароль: Aa1	С2
3	Логин: Admin, Пароль: Qwerty1234	С3
4	Логин: Random, Пароль: qqqqq	С4
5	Логин: Random, Пароль: FFFFFF	С5
6	Логин: Random, Пароль: Asdfgh	С6
7	Логин: Random, Пароль: Qwerty1234	С7

Выполнено построение функциональной диаграммы и произведено ее тестирование.

4.2.4 Парное тестирование

Для парного тестирования в таблице 17 описаны входные классы эквивалентности. Каждое значение параметра эквивалентности должно сочетаться с каждым другим параметром.

Таблица 17 – Входные классы эквивалентности

Логин	**Пароль**
Admin	Qwerty1234
Ada	Aa1
Random	Asdfgh
-	FFFFFF
-	qqqqq
-	66666

Для создания кейсов для парного тестирования был использован открытый perl-скрипт Allpairs. Скрипт на основе классов эквивалентности сгенерировал таблицу случаев для парного тестирования (рисунок 32).

TEST CASES			
case	Login	Password	pairings
1	Admin	Qwerty12	1
2	Ada	Qwerty12	1
3	Random	Qwerty12	1
4	Admin	Aal	1
5	Ada	Aal	1
6	Random	Aal	1
7	Admin	Asdfgh	1
8	Ada	Asdfgh	1
9	Random	Asdfgh	1
10	Admin	FFFFFF	1
11	Ada	FFFFFF	1
12	Random	FFFFFF	1
13	Admin	qqqqq	1
14	Ada	qqqqq	1
15	Random	qqqqq	1
16	Admin	66666	1
17	Ada	66666	1
18	Random	66666	1

Рисунок 32 – Парное тестирование

В результате парного тестирования программа работала верно.

4.2.5 Тестирование по вариантам использования

Для вариантов использования были описаны тесты и ожидаемый результат. Тестовые наборы представлены в таблице 18.

Таблица 18 – Тесты вариантов использования

№	Вариант использования	Тест	Ожидаемый результат	Результат
1	2	3	4	5
1.	Установить соединение с сервером	Установить соединение автоматически при запуске приложения.	Иконка соединения показывает соединение с сервером.	Тест выполнен (рисунок В.1).
2.	Воспроизвести аудио	Выбрать из загруженных с сервера аудио-файлов.	Загрузка и воспроизведение выбранного файла.	Тест выполнен (рисунок В.2).
3.	Воспроизвести аудио	Выбрать аудио-файл, расположенный на компьютере пользователя.	Воспроизведение выбранного файла.	Тест выполнен (рисунок В.3).
4.	Переключить звук	Нажать на иконку звука.	Звук переключен. Иконка поменялась.	Тест выполнен (рисунок В.4).
5.	Изменить громкость	Переместить ползунок громкости на необходимое значение.	Громкость аудио изменена.	Тест выполнен (рисунок В.5).
6.	Поставить на паузу	Нажать на кнопку паузы.	Воспроизведение аудио остановлено.	Тест выполнен (рисунок В.6).
7.	Перемотать аудио	Переместить ползунок воспроизведения на необходимое значениею	Вопроизведение с указанного участка.	Тест выполнен (рисунок В.7).
8.	Поставить повтор воспроизведения	Нажать на кнопку повтора.	При завершении аудио воспроизведется тот же файл сначала.	Тест выполнен (рисунок В.8).
9.	Переместиться по плейлисту	Воспроизвести следующий файл.	Воспроизводится следующий файл.	Тест выполнен (рисунок В.9).
10.		Воспроизвести предыдущий файл.	Воспроизводится предыдущий файл.	Тест выполнен (рисунок В.10).
11.	Осуществить поиск	Ввести запрос и нажать поиск.	Вывод наиболее релевантных файлов.	Тест выполнен (рисунок В.11).

Продолжение таблицы 18

1	2	3	4	5
12.	Добавить в плейлист	Нажать на кнопку добавления.	Добавления выбранного файла в плейлист	Тест выполнен (рисунок В.12).
13.	Зарегистрироваться	Логин: Admin Пароль: Qwerty1234	Регистрация произведена и Admin авторизован в приложении.	Тест выполнен (рисунок В.13).
14.	Авторизироваться	Логин: Admin Пароль: Qwerty1234	Авторизация пользователя Admin произведена.	Тест выполнен (рисунок В.14).
15.		Логин: Admin Пароль: 12345	Ошибка авторизации	Тест выполнен (рисунок В.15).
16.	Загрузить аудио на сервер	Ввести информацию о файле и загрузить.	Файл загружен на сервер.	Тест выполнен (рисунок В.16).
17.	Прокомментировать аудио	Добавить комментарий к файлу.	Комментарий сохранен на сервере.	Тест выполнен (рисунок В.17, В.18).
18.	Добавить трактовку тексту песни	Выбрать участок слов и добавить ему интерпретацию.	Трактовка сохранена на сервере.	Тест выполнен (рисунок В.19, В.20).

Тестирование по вариантам использования было успешно пройдено. Результаты тестирования приведены в приложении В.

4.3 Модульное тестирование

Модельное тестирование позволяет провести тестирование отдельного блока кода, определенного функционала, определенного поведения, последовательности действий. Было решено использовать модульные тесты Visual Studio. Поскольку разработка проекта ведется на данной IDE, и функционал среды является оптимальным.

Модульному тестированию было подвергнуто клиент-серверное взаимодействие. Тесты симулировали работу пользователя, отправляли запросы серверу, выполняли контроль возвращаемого результата. Далее описаны тестовые случаи в таблице 19.

Таблица 19 – Модульные тесты

Тестовый случай	Описание	Модульные тесты
Start Working	Тестовый случай для симуляции работы пользователя с приложением для прослушивания музыки	Установка соединения Загрузка плейлиста Загрузка картинок Загрузка файла mp3
Registration	Тестовый случай для симуляции регистрации нового пользователя	Установка соединения Регистрация пользователя
Authorization	Тестовый случай для симуляции авторизации пользователя	Установка соединения Авторизация пользователя
Communicating	Тестовый случай для симуляции социального взаимодействия пользователей	Установка соединения Загрузка комментариев Загрузка интерпретаций Отправка комментария Отправка интерпретации

Все модульные тесты были успешно пройдены. Результаты тестирования приведены на рисунке 33.

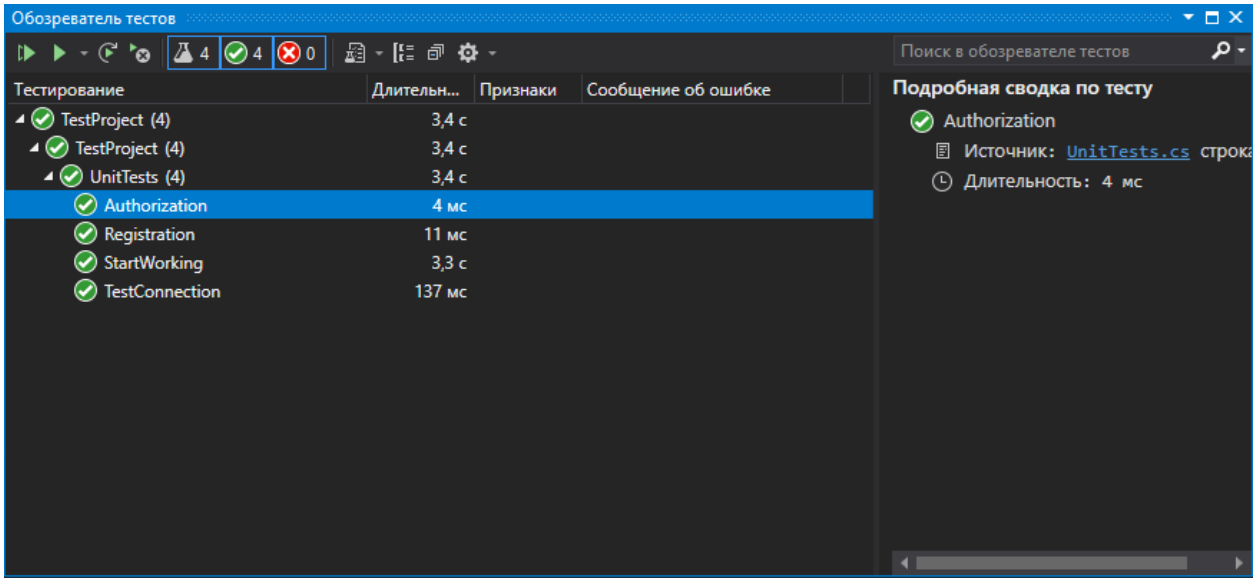

Рисунок 33 – Результаты модульных тестов

Таким образом, проведено модульное тестирование.

4.4 Метрики кода

Для определения метрик кода я воспользовался инструментарием IDE Visual Studio, в которой выполнялась разработка проекта.

Индекс удобства поддержки = 79. Значение индекс принимает в диапазоне от 0 до 100. Чем выше значение, тем легче поддерживать код. Диапазон от 0 до 9 говорит плохой сопровождаемости кода. Диапазон 10-19 является нейтральным, а 20-100 говорит о хорошей сопровождаемости кода.

Значение равно 79, что является хорошим показателем.

Сложность организации циклов = 163. Говорит о количестве путей потока управления. Соответственно, чем выше этот показатель, тем сложнее проводить тестирование программы.

Глубина наследования = 9 (1). Значение показывает максимальную длину иерархии классов. Чем выше число – тем выше сложность при необходимости внесений изменений в базовый класс. Значение 9 вычислено Visual Studio из-за иерархии элементов WPF. А значение непосредственно пользовательской программы равно 1, что является оптимальным при необходимости внесения изменений.

Взаимозависимость классов = 80. Говорит о связях между классами, использующих общие параметры, типы, методы, шаблоны, интерфейсы. Чем выше это число, тем выше сложность использования и поддержки проекта, поскольку при необходимости использования или изменения определенной части кода раскручивается спираль взаимозависимостей. Для примера, вы хотите изменить определенный класс, но он ссылается еще на три класса, а те в свою очередь еще на три класса и так далее, и тривиальная задача значительно усложняется.

Строки исходного кода = 1288. Говорит о количестве строк кода, включая пустые строки.

Строки исполняемого кода = 297. Говорит о строках, которые исполняются.

Таким образом, проведенное тестирование показывает корректную работу приложения согласно установленным требованиям.

4.5 Планирование и оценка бюджета

Планирование и оценка бюджета произведены при помощи программы управления проектами Microsoft Project 2007. Проект начат 20.04.21 года и по расчетам будет окончен 24.06.21 года. Были составлены задачи, которые необходимо решить в рамках разработки проекта. Задачи представлены на рисунке 34.

Начало проекта	0 дней	Вт 20.04.21	Вт 20.04.21		
Анализ предметной области и постановка задачи	**4 дней**	**Вт 20.04.21**	**Пт 23.04.21**	**1**	
Анализ предметной области	1 день	Вт 20.04.21	Вт 20.04.21	1	Директор
Сравнение приложений аналогов	1 день	Ср 21.04.21	Ср 21.04.21	3	Директор
Анализ функциональных требований	1 день	Чт 22.04.21	Чт 22.04.21	4	Директор;Программист
Постановка задачи	1 день	Пт 23.04.21	Пт 23.04.21	5	Директор;Программист
Проектирование программных средств	**11 дней**	**Пн 26.04.21**	**Пн 10.05.21**	**2**	
Проектирование архитектуры	5 дней	Пн 26.04.21	Пт 30.04.21		Программист
Проектирование базы данных	3 дней	Пн 03.05.21	Ср 05.05.21	8	Программист
Проектирование пользовательского интерфейса	3 дней	Чт 06.05.21	Пн 10.05.21	9	Программист
Реализация программных средств	**20 дней**	**Вт 11.05.21**	**Пн 07.06.21**	**7**	
Реализация пользовательского приложения	5 дней	Вт 11.05.21	Пн 17.05.21		Программист
Реализация сервера	8 дней	Вт 18.05.21	Чт 27.05.21	12	Программист
Реализация базы данных	4 дней	Пт 28.05.21	Ср 02.06.21	13	Программист
Итоговая сборка	3 дней	Чт 03.06.21	Пн 07.06.21	14	Программист
Тестирование	**9 дней**	**Вт 08.06.21**	**Пт 18.06.21**	**11**	
Отладка сервера и клиента	3 дней	Вт 08.06.21	Чт 10.06.21		Тестировщик;Программист
Рефакторинг	3 дней	Пт 11.06.21	Вт 15.06.21	17	Программист
Тестирование сервера и клиента	3 дней	Ср 16.06.21	Пт 18.06.21	18	Тестировщик
Завершение проекта	**4 дней**	**Пн 21.06.21**	**Чт 24.06.21**	**16**	
Составление документации	3 дней	Пн 21.06.21	Ср 23.06.21		Директор;Программист
Вывод в эксплуатацию	1 день	Чт 24.06.21	Чт 24.06.21	21	Директор
Конец проекта	0 дней	Чт 24.06.21	Чт 24.06.21	20	

Рисунок 34 – Список задач проекта Аудио Сервис

Помимо задач для успешного завершения проекта необходимы ресурсы. Ресурсы представлены на рисунке 35.

	Название ресурса	Тип	Единицы измерения материалов	Краткое название	Группа	Макс. единиц	Стандартная ставка	Ставка сверхурочных	Затраты на использ.	Начисление	Базовый календарь
1	Директор	Трудовой		Д		100%	800,00 ₽/ч	1 600,00 ₽/ч	0,00 ₽	Пропорциональное	Стандартный
2	Программист	Трудовой		П		100%	500,00 ₽/ч	1 000,00 ₽/ч	0,00 ₽	Пропорциональное	Стандартный
3	Тестировщик	Трудовой		Т		100%	300,00 ₽/ч	600,00 ₽/ч	0,00 ₽	Пропорциональное	Стандартный

Рисунок 35 – Список ресурсов проекта Аудио Сервис

По выделенным вехам и задачам была построена диаграмма Ганта.

Диаграмма Ганта приведена на рисунке 36. Диаграмма показывает связь задач, их последовательность при выполнении проекта, распределение ресурсов, даты начала и завершения работ.

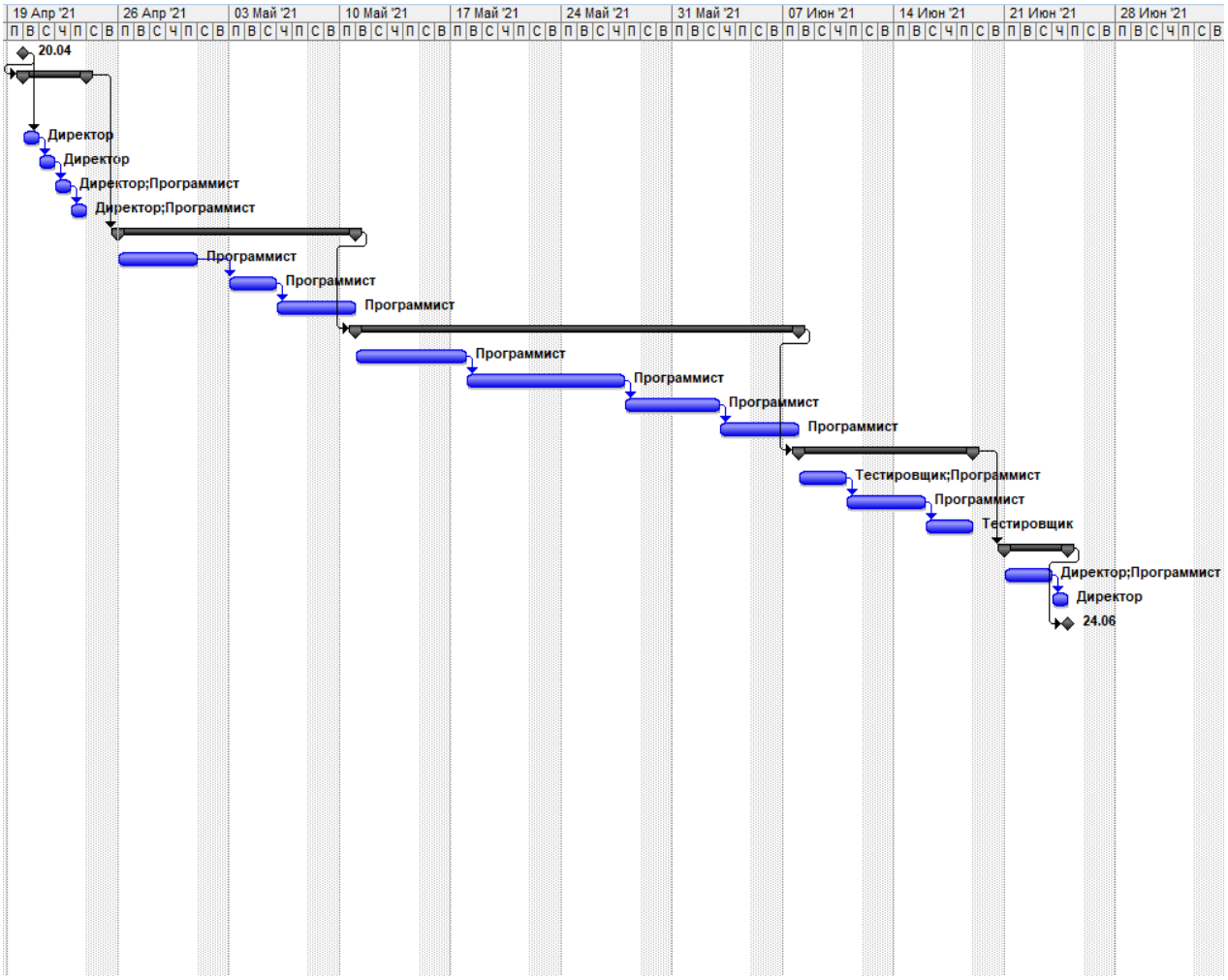

Рисунок 36 – Диаграмма Ганта проекта Аудио Сервис

Программа Microsoft Project 2007 позволяет генерировать отчеты по определенным задачам, их временными взаимосвязями и выделенным ресурсам. Так были сгенерированы диаграмма о базовых трудозатратах на всех вехах разработки и диаграмма о трудозатратах ресурсов – самих работников. Они показаны на рисунках 37, 38.

Рисунок 37 – Отчет о базовых трудозатратах

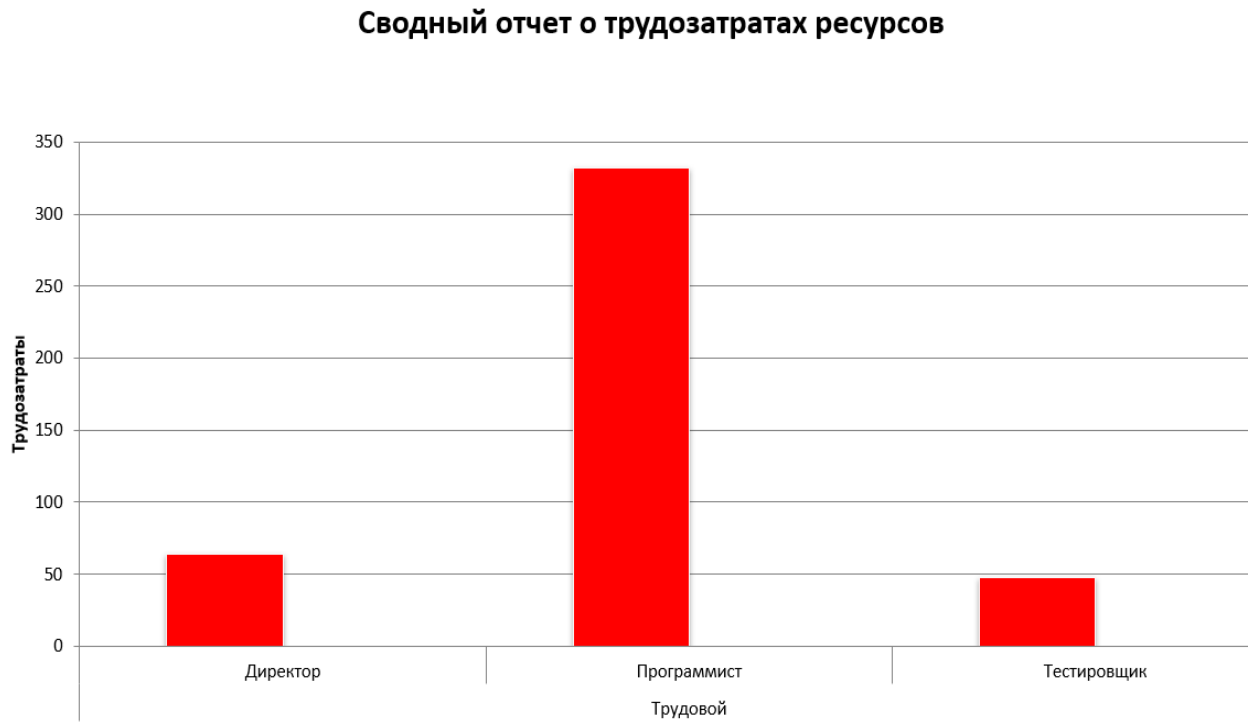

Рисунок 38 – Отчет о трудозатратах ресурсов

Дополнительно сгенерирован отчет о связи трудозатрат на вехах разработки с затратами финансовых ресурсов. Данная диаграмма показывает как складываются общие затраты из затрат на каждую веху разработки. Отчет представлен на рисунке 39.

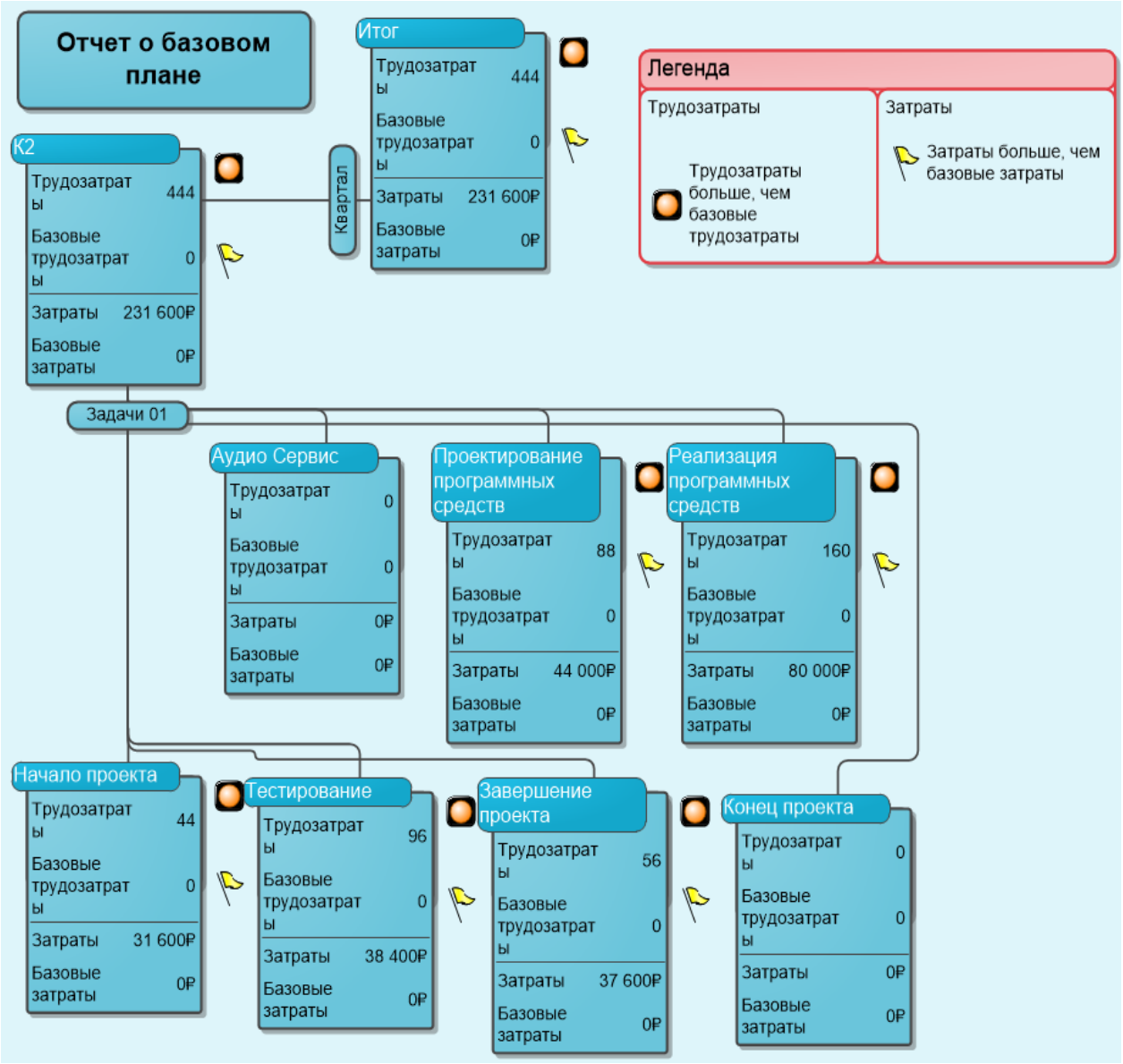

Рисунок 39 – Отчет о трудозатратах ресурсов

Из отчетов следует, что при установленных затратах на проект и стоимостью ресурсов, итоговая сумма оценки бюджета проекта составляет 231 тысячу 600 рублей. Время разработки составит 34 полных рабочих дня. И займет промежуток от 20 апреля 21 года до 24 июня 21 года.

Заключение

По итогам выпускной квалификационной работы был реализован аудио сервис для пользователя интернет, позволяющий пользователю прослушивать музыку, обсуждать ее с другими людьми по сети. Приложение состоит из клиентской и серверной части.

Клиентская часть обеспечивает пользователю необходимый интерфейс для настройки прослушивания, выбора музыки, отправки запросов на сервер. Клиент написан на языке C# на платформе WPF с использованием вспомогательных NuGet-пакетов.

Серверная часть обеспечивает взаимодействие с Базой Данных MySQL, ответы на запросы клиентов. Сервер написан на языке C# на фреймворке ASP.NET Core.

Проведено структурное и функциональное тестирование, модульное тестирование, анализ метрик кода. Требования тестирования успешно пройдены.

Выполнено планирование и произведена оценка бюджета. Так, стоимость проекта оценивается в 231 тысячу 600 рублей. А срок разработки составит 34 полных рабочих дня. Что является экономически целесообразным.

Итого, в ходе выполнения выпускной квалификационной работы было разработано программное обеспечение аудио сервиса для пользователя интернет, отвечающее всем требованиям технического задания.

Список использованных источников

1. Вартанян И.А. Звук-Слух-Мозг / И.А. Вартанян. - Академия наук СССР Ленинград, 1981. - 176 с.

2. Пользователи «ВКонтакте» массово жалуются на навязчивую рекламу в аудиозаписях // Digger [Электронный ресурс]. URL: https://www.digger.ru/news/polzovateli-vkontakte-massovo-zhaluyutsya-na-navyazchivuyu-reklamu-v-audiozapisyax (дата обращения: 26.04.2021).

3. Mediascope [Электронный ресурс]. URL: https://webindex.mediascope.net/top-resources (дата обращения: 26.04.2021).

4. «Яндекс» возглавил рейтинг самых дорогих компаний Рунета // Forbes [Электронный ресурс]. URL: https://www.forbes.ru/newsroom/tehnologii/421841-yandeks-vozglavil-reyting-samyh-dorogih-kompaniy-runeta (дата обращения: 26.04.2021).

5. Spotify назвал запуск сервиса в России самым успешным в своей истории // РБК [Электронный ресурс]. URL: https://www.rbc.ru/technology_and_media/29/10/2020/5f9accc19a79477a52253dc8 (дата обращения: 26.04.2021).

6. Арлоу Д. UML 2 и Унифицированный процесс, 2-е издание. Практический объектно-ориентированный анализ и проектирование. / Арлоу Д., Нейштадт А. – СПб: Символ Плюс, 2007. – 624 с.

7. Павловская Т.А. C# программирование на языке высокого уровня / Павловская Т. – СПб: Питер, 2007. – 432 с.

8. Microsoft [Электронный ресурс]. URL: https://docs.microsoft.com/ru-ru/ (дата обращения: 20.05.2021).

9. MySQL Workbench Documentation [Электронный ресурс]. URL: https://dev.mysql.com/doc/workbench/en/ (дата обращения: 22.05.2021).

10. Маклахов С.В. BPwin и Erwin. CASE-средства для разработки информационных систем / С.В. Маклахов. – Москва: ДИАЛОГ-МИФИ, 2000. – 347 с.

11. Jerry Cao Color Theory in Web UI Design / Jerry Cao - UXPin Inc., 2015. - 35 с.

12. Material Design in XAML [Электронный ресурс]. URL: http://materialdesigninxaml.net (дата обращения: 25.05.2021).

13. Удобная библиотека для работы с тегами медиафайлов // Habr [Электронный ресурс]. URL: https://habr.com/ru/post/177795/ (дата обращения: 25.05.2021).

14. RestSharp [Электронный ресурс]. URL: https://restsharp.dev (дата обращения: 21.05.2021).

15. Руководство по MySQL // Metanit [Электронный ресурс]. URL: https://metanit.com/sql/mysql/ (дата обращения: 23.05.2021).

Приложение А

ГЛОССАРИЙ

(справочное)

Объектно-ориентированное программирование (ООП) – это подход к программированию, предполагающий представление программы в виде взаимодействующих объектов – экземпляров классов. Проектирование классов должно опираться на основные подходы: полиморфизм, инкапсуляцию, наследование.

База данных (БД) – данные, организованные определенным образом и физически хранящиеся в памяти компьютера.

MySQL Workbench – Реляционная система управления базами данных. Обеспечивает создание базы данных, ее хранение, представление, управление. Реляционная СУБД при проектировании базы данных предполагает выделение сущностей и выстраивание связей между ними.

C# – это объектно-ориентированный, статически типизируемый язык, исполняемый на платформе .NET.

Unified Modeling Language (UML) – это унифицированный язык моделирования, графически описывающий различные стороны программного продукта при проектировании.

Windows Presentation Foundation (WPF) – это система для создания графических приложений. Входит в состав платформы .NET.

Клиент – приложение, отправляющее запрос на некий адрес.

Сервер – приложение, получающие запросы и формирующее ответы.

Приложение Б

ЛИСТИНГ КЛИЕНТСКОГО ПРИЛОЖЕНИЯ НА ЯЗЫКЕ C#

(рекомендуемое)

Листинг Б.1 – MainWindow.xaml.cs

```
using System;
using System.Collections.ObjectModel;
using System.IO;
using System.Text;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using Client.Logic;
using MaterialDesignThemes.Wpf;
using Microsoft.Win32;
using SharedData;

namespace Client
{
    /// <summary>
    /// Interaction logic for MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
        MusicManager m_MusicManager;
        UserProfile m_UserProfile;
        double m_LastPosByManager = 0;
        //Блокирует обновление позиции курсора при работе пользователя с
ползунком
        bool m_LockTimerSlider = true;
        bool VolumeSwitch = true;

        public MainWindow()
        {
            InitializeComponent();

            //-----------------------------------MusicManager----------------
--------------
            m_MusicManager = new MusicManager();

            Binding binding = new Binding();
            binding.Source = m_MusicManager;
            binding.Path = new PropertyPath("Image");
            MusicPng.SetBinding(Image.SourceProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager;
            binding.Path = new PropertyPath("Description");
            TextDescription.SetBinding(TextBlock.TextProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager;
            binding.Path = new PropertyPath("Name");
            TextSongName.SetBinding(TextBlock.TextProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager;
            binding.Path = new PropertyPath("MusicTime");
            TextMusicTime.SetBinding(TextBlock.TextProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager;
            binding.Path = new PropertyPath("CurrentTime");
            TextCurrentTime.SetBinding(TextBlock.TextProperty, binding);
```

```
            binding = new Binding();
            binding.Source = m_MusicManager;
            binding.Path = new PropertyPath("Maximum");
            TimerSlider.SetBinding(Slider.MaximumProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager;
            binding.Path = new PropertyPath("KindPlay");
            ButtonPlayDesign.SetBinding(PackIcon.KindProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager.Music;
            binding.Path = new PropertyPath("Data");
            MusicList.SetBinding(ListBox.ItemsSourceProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager.Playlist;
            binding.Path = new PropertyPath("Data");
            PlaylistsList.SetBinding(ListBox.ItemsSourceProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager.Forum;
            binding.Path = new PropertyPath("Data");
            CommentaryList.SetBinding(ListBox.ItemsSourceProperty, binding);

            binding = new Binding();
            binding.Source = m_MusicManager.Interpretation;
            binding.Path = new PropertyPath("Data");
            InterpretationList.SetBinding(ListBox.ItemsSourceProperty,
binding);



            m_MusicManager.NewPositionEvent += M_MusicManager_Notify;
            m_MusicManager.NewWordsEvent += M_MusicManager_NewWordsEvent;

            //-----------------------------------UserProfile-----------------
--------------
            m_UserProfile = new UserProfile();

            binding = new Binding();
            binding.Source = m_UserProfile;
            binding.Path = new PropertyPath("Greetings");
            TextHello.SetBinding(TextBlock.TextProperty, binding);

            binding = new Binding();
            binding.Source = m_UserProfile;
            binding.Path = new PropertyPath("AuthoErrorLogin");
            AuthoLogErr.SetBinding(TextBlock.TextProperty, binding);

            binding = new Binding();
            binding.Source = m_UserProfile;
            binding.Path = new PropertyPath("AuthoErrorPassword");
            AuthoPassErr.SetBinding(TextBlock.TextProperty, binding);

            binding = new Binding();
            binding.Source = m_UserProfile;
            binding.Path = new PropertyPath("RegisErrorLogin");
            RegisLogErr.SetBinding(TextBlock.TextProperty, binding);

            binding = new Binding();
            binding.Source = m_UserProfile;
```

```
            binding.Path = new PropertyPath("RegisErrorPassword");
            RegisPassErr.SetBinding(TextBlock.TextProperty, binding);

            ButtonConnect_Click(null, null);
        }

        private void M_MusicManager_NewWordsEvent(string words)
        {
            var document = new FlowDocument();
            var paragraph = new Paragraph();

            paragraph = new Paragraph();
            paragraph.Inlines.Add(new Bold(new Run(words)));

            document.Blocks.Add(paragraph);
            SongText.Document = document;
        }

        private void M_MusicManager_Notify(double pos)
        {
            if (m_LockTimerSlider)
            {
                m_LastPosByManager = pos;
                TimerSlider.Value = pos;
            }
        }

        private void Card_MouseDown(object sender, MouseButtonEventArgs e)
        {
            try
            {
                DragMove();
            }
            finally { }
        }

        private void ButtonClose_Click(object sender, RoutedEventArgs e)
        {
            Application.Current.Shutdown();
        }

        private void ButtonPlay_Click(object sender, RoutedEventArgs e)
        {
            m_MusicManager.Play();
        }

        private void Volume_Changed(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            VolumeSwitch = true;

            m_MusicManager?.NewVolume(e.NewValue / 100);
        }

        private void ButtonRandom_Click(object sender, RoutedEventArgs e)
        {
            m_MusicManager.PlayRandom();
        }

        private void ButtonPrev_Click(object sender, RoutedEventArgs e)
        {
            m_MusicManager.PlayPrevious();
        }
```

```
        private void ButtonNext_Click(object sender, RoutedEventArgs e)
        {
            m_MusicManager.PlayNext();
        }

        private void ButtonRepeat_Click(object sender, RoutedEventArgs e)
        {
            m_MusicManager.PlayRepeat();
        }

        private void MusicList_SelectionChanged(object sender,
SelectionChangedEventArgs e)
        {
            if (MusicList.SelectedItem != null)
            {

m_MusicManager.NewCompositionSelected((Logic.Music)MusicList.SelectedItem);
            }
        }

        private void ButtonFile_Click(object sender, RoutedEventArgs e)
        {
            OpenFileDialog dialog = new OpenFileDialog();
            dialog.Filter = "Audio files|*.mp3";
            dialog.FilterIndex = 2;

            if (dialog.ShowDialog() == true)
            {
                m_MusicManager.OpenMusicFile(dialog.FileName);
            }
        }

        private void TimeChanged(object sender,
RoutedPropertyChangedEventArgs<double> e)
        {
            if(m_LastPosByManager != e.NewValue)
            {
                m_LockTimerSlider = false;

                m_MusicManager.NewPosition(TimerSlider.Value);

                m_LockTimerSlider = true;
            }
        }

        private void ButtonFind_Click(object sender, RoutedEventArgs e)
        {
            m_MusicManager.FindMusic(TextSearch.Text);
        }

        private void ButtonVolumeSwitch_Click(object sender, RoutedEventArgs
e)
        {
            VolumeSwitch = !VolumeSwitch;

            if (!VolumeSwitch)
            {
                ButtonVolumeDesign.Kind = PackIconKind.AudioOff;
                m_MusicManager.Volume = 0;
            }
            else
            {
```

```
                ButtonVolumeDesign.Kind = PackIconKind.Audio;
                m_MusicManager.Volume = VolumeSlider.Value / 100;
            }
        }

        private void ButtonConnect_Click(object sender, RoutedEventArgs e)
        {
            if(QueryMaster.Connection)
            {
                return;
            }

            if(QueryMaster.SuccessConnect())
            {
                ConnectionStatus.Kind = PackIconKind.Connection;
                TextConnect.Text = "Online Mode";
                QueryMaster.Connection = true;
                m_MusicManager.LoadContentFromServer();
            }
            else
            {
                ConnectionStatus.Kind = PackIconKind.LanDisconnect;
                TextConnect.Text = "Offline Mode";
                QueryMaster.Connection = false;
            }
        }

        private void ButtonLogin_Click(object sender, RoutedEventArgs e)
        {
            TabMenu.SelectedValue = Authorization;
        }

        private void AuthoToRegistRedirect(object sender, RoutedEventArgs e)
        {
            TabMenu.SelectedValue = Registration;
        }

        private void AuthorizationClick(object sender, RoutedEventArgs e)
        {
            m_UserProfile.Authorization(AuthoLogin.Text,
AuthoPassword.Password);
        }

        private void RegistrationClick(object sender, RoutedEventArgs e)
        {
            m_UserProfile.Registration(RegisLogin.Text,
RegisPassword1.Password, RegisPassword2.Password);
        }

        private void UploadClick(object sender, RoutedEventArgs e)
        {

        }

        private void InterprClick(object sender, RoutedEventArgs e)
        {
            if (m_UserProfile.User == null)
            {
                MessageBox.Show("Please, Log Ining");
                return;
            }
```

```
            string text = new TextRange(UsersInterpr.Document.ContentStart,
UsersInterpr.Document.ContentEnd).Text;

            if(text == "")
            {
                return;
            }

            var start = m_MusicManager.Words.IndexOf(LyricsWordInterpr.Text);
            var end = start + LyricsWordInterpr.Text.Length;

            m_MusicManager.Interpretation.AddInterpretation(start, end, text,
                m_UserProfile.User.Login, m_MusicManager.ID);

            UsersInterpr.Document = new FlowDocument();
        }

        private void SelectionLyrics(object sender, RoutedEventArgs e)
        {
            LyricsWordInterpr.Text = SongText.Selection.Text;
        }

        private void CommentaryClick(object sender, RoutedEventArgs e)
        {
            if (m_UserProfile.User == null)
            {
                MessageBox.Show("Please, Log Ining");
                return;
            }

            string text = new
TextRange(UsersCommentary.Document.ContentStart,
UsersCommentary.Document.ContentEnd).Text;

            if (text == "")
            {
                return;
            }

            int prev = -1;

            if (AnswerAppeal != null && text.Substring(0,
AnswerAppeal.Length).Equals(AnswerAppeal))
            {
                prev = IdAppeal;
            }

            m_MusicManager.Forum.AddCommentary(text,
                m_UserProfile.User.Login, m_MusicManager.ID, prev,
SelectedIdCollection/*, Convert.ToInt32(Rate.Text)*/);

            UsersCommentary.Document = new FlowDocument();
        }

        string AnswerAppeal { get; set; }
        int IdAppeal { get; set; }
        int SelectedIdCollection { get; set; }

        private void ReplyCommentaryClick(object sender, RoutedEventArgs e)
        {
            var forum = ((Forum)((Button)sender).DataContext);
            SelectedIdCollection = m_MusicManager.Forum.Data.IndexOf(forum);
            IdAppeal = forum.Id;
```

```
            AnswerAppeal = forum.User + ",";

            var document = new FlowDocument();
            var paragraph = new Paragraph();
            paragraph.Inlines.Add(new Run(AnswerAppeal));

            document.Blocks.Add(paragraph);
            UsersCommentary.Document = document;
        }

        //Подгружает музыку в конце скролла
        private void MusicScrollChanged(object sender, ScrollChangedEventArgs
e)
        {
            if (MusicScroll.VerticalOffset == MusicScroll.ScrollableHeight)
            {
                m_MusicManager.Music.LoadMore();
            }
        }

        private void PlaylistDoubleClick(object sender, MouseButtonEventArgs
e)
        {
            var forum =
((Client.Logic.Playlist)((ListBox)sender).SelectedItem);

            m_MusicManager.Music.LoadPlaylist(forum.Id);

            if(m_MusicManager.Music.Data.Count > 0)
            {

m_MusicManager.NewCompositionSelected(m_MusicManager.Music.Data[0]);
            }
        }
    }

    public class ThicknessConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object
parameter, System.Globalization.CultureInfo culture)
        {
            return new Thickness((int)value, 0, -(int)value, 0);
        }

        public object ConvertBack(object value, Type targetType, object
parameter, System.Globalization.CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }

    public class WidthConverter : IValueConverter
    {
        public object Convert(object value, Type targetType, object
parameter, System.Globalization.CultureInfo culture)
        {
            return 520 - (int)value;
        }

        public object ConvertBack(object value, Type targetType, object
parameter, System.Globalization.CultureInfo culture)
        {
            throw new NotImplementedException();
```

```
        }
    }
}
```

Листинг Б.2 – MainWindow.xaml

```
<Window x:Class="Client.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
xmlns:materialDesign="http://materialdesigninxaml.net/winfx/xaml/themes"
        xmlns:local="clr-namespace:Client"
        mc:Ignorable="d"
        Height="560" Width="1000" WindowStartupLocation="CenterScreen"
WindowStyle="None"
        ResizeMode="CanResize" AllowsTransparency="True"
Background="{x:Null}">
    <Window.Resources>
        <local:ThicknessConverter x:Key="ThicknessConverter" />
        <local:WidthConverter x:Key="WidthConverter" />
    </Window.Resources>
    <Grid Background="{x:Null}">
        <materialDesign:Card Margin="10" UniformCornerRadius="20"
Background="{x:Null}" MouseDown="Card_MouseDown">
            <StackPanel>
                <StackPanel.Background>
                    <!--<LinearGradientBrush StartPoint="0.3,0"
EndPoint="0.7,0">
                        <GradientStop Color="#FFFFFF" Offset="1"/>
                        <GradientStop Color="#00FF00" Offset="0"/>
                    </LinearGradientBrush>-->
                    <LinearGradientBrush StartPoint="0,0" EndPoint="1,0">
                        <GradientStop Color="#FF75D2FB" Offset="1.0" />
                    </LinearGradientBrush>
                </StackPanel.Background>
                <!--Заголовок-->
                <Grid Height="60" VerticalAlignment="Top">
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="50"/>
                        <ColumnDefinition/>
                        <ColumnDefinition/>
                        <ColumnDefinition Width="50"/>
                        <ColumnDefinition Width="50"/>
                    </Grid.ColumnDefinitions>
                    <Button Grid.Column="0" Style="{StaticResource
ActionButton}" HorizontalAlignment="Center" ToolTip="Connection to Server"
x:Name="ButtonConnect" Click="ButtonConnect_Click" Cursor="Hand">
                        <materialDesign:PackIcon Kind="Connection"
x:Name="ConnectionStatus" Width="20" Height="20"/>
                    </Button>
                    <TextBlock Grid.Column="1" Text="Offline Mode"
x:Name="TextConnect" Style="{StaticResource TextBlock}"
HorizontalAlignment="Left"
                               VerticalAlignment="Center" FontSize="22"/>
                    <TextBlock Grid.Column="2" Margin="5" x:Name="TextHello"
Style="{StaticResource TextBlock}" HorizontalAlignment="Right"
                               VerticalAlignment="Center" FontSize="22"/>
                    <Button Grid.Column="3" Style="{StaticResource
ActionButton}" HorizontalAlignment="Center" ToolTip="Log In"
x:Name="ButtonLogin" Click="ButtonLogin_Click">
```

```
                        <materialDesign:PackIcon Kind="Login" Width="20"
Height="20"/>
                    </Button>
                    <Button Grid.Column="4" Style="{StaticResource
ActionButton}" HorizontalAlignment="Center" ToolTip="Close"
x:Name="ButtonClose" Click="ButtonClose_Click">
                        <materialDesign:PackIcon Kind="Close" Width="20"
Height="20"/>
                    </Button>
                </Grid>
                <Grid>
                    <Grid.ColumnDefinitions>
                        <ColumnDefinition Width="300"/>
                        <ColumnDefinition/>
                    </Grid.ColumnDefinitions>
                    <Grid Grid.Column="0" >
                        <Grid.RowDefinitions>
                            <RowDefinition Height="50"/>
                            <RowDefinition/>
                        </Grid.RowDefinitions>
                        <!-- Верхняя панель управления плеера -->
                        <Grid Grid.Row="0">
                            <Button Style="{StaticResource ActionButton}"
HorizontalAlignment="Left"  ToolTip="Open File" x:Name="ButtonFile"
Click="ButtonFile_Click">
                                <materialDesign:PackIcon Kind="Folder"
Width="20" Height="20"/>
                            </Button>
                            <Button Style="{StaticResource ActionButton}"
HorizontalAlignment="Right" ToolTip="Open File" x:Name="ButtonOptions">
                                <materialDesign:PackIcon Kind="Plus"
Width="20" Height="20"/>
                            </Button>
                        </Grid>
                        <!-- Музыкальный плеер -->
                        <StackPanel Grid.Row="1" Width="300"
HorizontalAlignment="Left">
                            <Image Height="200" Stretch="Uniform"
Source="based/icon.png" x:Name="MusicPng"/>
                            <TextBlock FontSize="19" Text="Song Name"
Style="{StaticResource TextBlock}" TextAlignment="Center"
x:Name="TextSongName" Height="26"/>
                            <TextBlock FontSize="16" Text="C# WPF Audio
Service" Style="{StaticResource TextBlock}" TextAlignment="Center"
FontWeight="ExtraLight" x:Name="TextDescription"/>
                            <Grid Margin="0 30 0 0">
                                <TextBlock FontSize="16" Text="0:00"
Margin="20 0" Style="{StaticResource TextBlock}" FontWeight="ExtraLight"
TextAlignment="Center" HorizontalAlignment="Left" x:Name="TextCurrentTime"/>
                                <TextBlock FontSize="16" Text="0:00"
Margin="20 0" Style="{StaticResource TextBlock}" FontWeight="ExtraLight"
TextAlignment="Center" HorizontalAlignment="Right" x:Name="TextMusicTime"/>
                            </Grid>
                            <Slider Value="0" Minimum="0" Maximum="100"
Margin="20 0" Style="{StaticResource Slider}" x:Name="TimerSlider"
IsMoveToPointEnabled="True" ValueChanged="TimeChanged"/>
                            <Grid Margin="15 10 10 0">
                                <Button Style="{StaticResource IconButton}"
HorizontalAlignment="Left" ToolTip="Volume" x:Name="ButtonVolumeSwitch"
Click="ButtonVolumeSwitch_Click">
                                    <StackPanel Margin="-5">
                                        <materialDesign:PackIcon
x:Name="ButtonVolumeDesign" Kind="Audio" Width="25" Height="25"/>
```

```
                                    </StackPanel>
                                </Button>
                                <Slider Value="10" Minimum="0" Maximum="50"
Margin="40 7 10 0" Style="{StaticResource Slider}" x:Name="VolumeSlider"
ValueChanged="Volume_Changed"/>
                            </Grid>
                            <!-- Нижняя панель управления плеера -->
                            <WrapPanel HorizontalAlignment="Center" Margin="0
30">
                                <Button Style="{StaticResource ActionButton}"
HorizontalAlignment="Left" Margin="5 0" ToolTip="Mixind"
x:Name="ButtonRandom" Click="ButtonRandom_Click">
                                    <StackPanel Margin="-5">
                                        <materialDesign:PackIcon
Kind="FormatLineSpacing" Width="25" Height="25"/>
                                    </StackPanel>
                                </Button>
                                <Button Style="{StaticResource ActionButton}"
HorizontalAlignment="Left" Margin="5 0" ToolTip="Play Previous"
x:Name="ButtonPrev" Click="ButtonPrev_Click">
                                    <StackPanel Margin="-5">
                                        <materialDesign:PackIcon
Kind="ArrowLeftBold" Width="30" Height="30"/>
                                    </StackPanel>
                                </Button>
                                <Button Style="{StaticResource ActionButton}"
HorizontalAlignment="Center" Margin="5 0" ToolTip="Play Music"
x:Name="ButtonPlay" Click="ButtonPlay_Click">
                                    <StackPanel Margin="-5">
                                        <materialDesign:PackIcon Kind="Play"
Width="30" Height="30" x:Name="ButtonPlayDesign"/>
                                    </StackPanel>
                                </Button>
                                <Button Style="{StaticResource ActionButton}"
HorizontalAlignment="Right" Margin="5 0" ToolTip="Play Next"
x:Name="ButtonNext" Click="ButtonNext_Click">
                                    <StackPanel Margin="-5">
                                        <materialDesign:PackIcon
Kind="ArrowRightBold" Width="30" Height="30"/>
                                    </StackPanel>
                                </Button>
                                <Button Style="{StaticResource ActionButton}"
HorizontalAlignment="Right" Margin="5 0" ToolTip="Repeat"
x:Name="ButtonRepeat" Click="ButtonRepeat_Click">
                                    <StackPanel Margin="-5">
                                        <materialDesign:PackIcon
Kind="Repeat" Width="25" Height="25"/>
                                    </StackPanel>
                                </Button>
                            </WrapPanel>
                        </StackPanel>
                    </Grid>

                    <TabControl Grid.Column="1" Style="{StaticResource
TabControl}" Margin="5,0,0,0"  TabStripPlacement="Left" Height="500"
x:Name="TabMenu">
                        <TabItem Style="{StaticResource TabItem}"
VerticalAlignment="Top">
                            <TabItem.Header>
                                <Grid>
                                    <materialDesign:PackIcon
Kind="PlaylistMusic" Width="50" Height="50" Background="#e52b50"
```

```
Foreground="White"/>
                            </Grid>
                        </TabItem.Header>

                        <Grid>
                            <Grid.RowDefinitions>
                                <RowDefinition Height="50"/>
                                <RowDefinition/>
                            </Grid.RowDefinitions>

                            <!-- Панель поиска -->
                            <Grid Grid.Column="1" Grid.Row="0">
                                <Grid.ColumnDefinitions>
                                    <ColumnDefinition Width="Auto"/>
                                    <ColumnDefinition/>
                                </Grid.ColumnDefinitions>

                                <TextBox Grid.Column="0" Margin="10 0"
Text="" TextAlignment="Left" FontSize="20" Width="520"
                                         Style="{StaticResource TextBox}"
HorizontalAlignment="Left" VerticalAlignment="Center" x:Name="TextSearch"/>

                                <Button Grid.Column="1"
Style="{StaticResource ActionButton}" Margin="5" HorizontalAlignment="Center"
ToolTip="Find" x:Name="ButtonFind" Click="ButtonFind_Click">
                                    <materialDesign:PackIcon
Kind="Magnify" Width="20" Height="20"/>
                                </Button>
                            </Grid>

                            <!-- Список музыки -->
                            <StackPanel  Grid.Column="1" Grid.Row="1">
                                <ScrollViewer CanContentScroll="True"
Height="400" x:Name="MusicScroll" ScrollChanged="MusicScrollChanged">
                                    <ListBox x:Name="MusicList"
Template="{StaticResource NoScroll}" VerticalAlignment="Top"
SelectionChanged="MusicList_SelectionChanged">
                                        <ListBox.ItemTemplate>
                                            <DataTemplate>
                                                <Grid>

<Grid.ColumnDefinitions>
<ColumnDefinition Width="Auto"/>
<ColumnDefinition Width="Auto"/>
<ColumnDefinition Width="10"/>
<ColumnDefinition Width="Auto"/>
</Grid.ColumnDefinitions>
                                                    <Image
Source="{Binding Image}" Grid.Column="0" Width="40" Height="40"/>
                                                    <TextBlock
Text="{Binding Name}" Grid.Column="1" FontSize="16" Style="{StaticResource
TextBlock}" TextAlignment="Center" VerticalAlignment="Center"/>
                                                    <TextBlock
Text="{Binding Author}" Grid.Column="3" FontSize="16" Style="{StaticResource
TextBlock}" TextAlignment="Center" VerticalAlignment="Center"/>
                                                </Grid>
                                            </DataTemplate>
                                        </ListBox.ItemTemplate>
                                    </ListBox>
                                </ScrollViewer>
                            </StackPanel>
                        </Grid>
```

```
                        </TabItem>

                        <TabItem Style="{StaticResource TabItem}">
                            <TabItem.Header>
                                <Grid VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                                    <materialDesign:PackIcon
Kind="LibraryMusic" Width="50" Height="50" Background="#e52b50"

Foreground="White"/>
                                </Grid>
                            </TabItem.Header>

                            <Grid>
                                <TextBlock FontSize="19" Text="Playlists"
Style="{StaticResource TextBlock}" VerticalAlignment="Top"
TextAlignment="Center" Height="26"/>

                                <StackPanel Margin="0 -50 0 0" Width="500"
Height="400">
                                    <ScrollViewer CanContentScroll="true"
Width="500" Height="400">
                                        <!--ItemsControl-->
                                        <ListBox x:Name="PlaylistsList"
Template="{StaticResource NoScroll}" MouseDoubleClick="PlaylistDoubleClick">
                                            <ListBox.ItemTemplate>
                                                <DataTemplate>
                                                    <Grid Width="230"
Height="210" Margin="-10 0 0 10">
                                                        <Grid.RowDefinitions>
                                                            <RowDefinition/>
                                                            <RowDefinition
Height="Auto"/>
                                                            <RowDefinition
Height="Auto"/>

</Grid.RowDefinitions>
                                                        <Image Margin="5"
Source="{Binding Image}" Grid.Row="0" Width="Auto" Height="Auto"
HorizontalAlignment="Center"/>
                                                        <TextBlock
Text="{Binding Name}" FontSize="16" Grid.Row="1" Style="{StaticResource
TextBlock}" VerticalAlignment="Top" TextAlignment="Center" Height="20"/>
                                                        <TextBlock
Text="{Binding Author}" FontSize="14" Grid.Row="2" Style="{StaticResource
TextBlock}" VerticalAlignment="Top" TextAlignment="Center" Height="20"/>
                                                    </Grid>
                                                </DataTemplate>
                                            </ListBox.ItemTemplate>
                                            <ItemsControl.ItemsPanel>
                                                <ItemsPanelTemplate>
                                                    <WrapPanel/>
                                                </ItemsPanelTemplate>
                                            </ItemsControl.ItemsPanel>
                                        </ListBox>
                                    </ScrollViewer>
                                </StackPanel>
                            </Grid>

                        </TabItem>
                        <TabItem Style="{StaticResource TabItem}">
                            <TabItem.Header>
```

```
                                <Grid VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                                    <materialDesign:PackIcon Kind="Text"
Width="50" Height="50" Background="#e52b50"

Foreground="White"/>
                                </Grid>
                            </TabItem.Header>

                            <Grid>
                                <Grid.RowDefinitions>
                                    <RowDefinition Height="Auto"/>
                                    <RowDefinition Height="Auto"/>
                                    <RowDefinition Height="Auto"/>
                                </Grid.RowDefinitions>

                                <TabControl Grid.Row="0" Height="325"
Width="Auto" Style="{StaticResource TabControl}">
                                    <TabItem Style="{StaticResource TabItem}"
HorizontalAlignment="Center">
                                        <TabItem.Header>
                                            <Grid VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                                                <TextBlock Text="Lyrics"
FontSize="20" Foreground="White" Style="{StaticResource TextBlock}"
Background="#e52b50" Height="27" VerticalAlignment="Top"/>
                                                <!--<materialDesign:PackIcon
Kind="Text" Width="50" Height="50" Background="#e52b50"

Foreground="White"/>-->
                                            </Grid>
                                        </TabItem.Header>
                                        <Grid>
                                            <RichTextBox Margin="0 0 0 20"
IsReadOnly="True" Grid.Column="0" Grid.Row="1" x:Name="SongText"
SelectionChanged="SelectionLyrics" Style="{StaticResource RichTextBox}"
FontSize="14" Cursor="IBeam">
                                            </RichTextBox>
                                        </Grid>
                                    </TabItem>

                                    <TabItem Style="{StaticResource
TabItem}">
                                        <TabItem.Header>
                                            <Grid VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                                                <TextBlock
Text="Interpretations" FontSize="20" Foreground="White"
Style="{StaticResource TextBlock}" Background="#e52b50"/>
                                                <!--<materialDesign:PackIcon
Kind="HeadBulb" Width="50" Height="50" Background="#e52b50"

Foreground="White"/>-->
                                            </Grid>
                                        </TabItem.Header>
                                        <Grid>
                                            <StackPanel>
                                                <ScrollViewer
CanContentScroll="True" Height="240">
                                                    <ListBox
x:Name="InterpretationList" Template="{StaticResource NoScroll}"
VerticalAlignment="Top">
```

```
<ListBox.ItemTemplate>
                                                                    <DataTemplate>
                                                                        <Grid>
                                                                            <Grid>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

                                                                                <Grid
Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<TextBlock Grid.Column="0" Text="{Binding User}" FontSize="16"
Style="{StaticResource TextBlock}"/>

<TextBlock Grid.Column="1" Margin="10 2" Text="{Binding Date}" FontSize="14"
Style="{StaticResource TextBlock}"/>

</Grid>


<TextBlock Grid.Row="1" Text="{Binding WordsLoop}" FontSize="16" Height="20"
Style="{StaticResource TextBlock}"/>
                                                                            </Grid>
                                                                            <Grid
Margin="0 50 0 0">

<RichTextBox FontSize="16" IsReadOnly="true" Style="{StaticResource
RichTextBox}">

<FlowDocument>

<Paragraph>

<Run Text="{Binding Words}">

</Run>

</Paragraph>

</FlowDocument>

</RichTextBox>
                                                                            </Grid>
                                                                        </Grid>
                                                                    </DataTemplate>

</ListBox.ItemTemplate>
```

```
                                        </ListBox>
                                    </ScrollViewer>
                                </StackPanel>
                            </Grid>
                        </TabItem>
                    </TabControl>

                    <TextBlock Grid.Row="1"
x:Name="LyricsWordInterpr" FontSize="16" Text="Words" Style="{StaticResource
TextBlock}" VerticalAlignment="Top" TextAlignment="Center"/>
                    <RichTextBox Grid.Row="1" Margin="0 20 0 0"
Height="100" x:Name="UsersInterpr" Style="{StaticResource RichTextBox}"
FontSize="16" Cursor="IBeam">
                        <FlowDocument>
                            <Paragraph>
                                <Run>
                                    Write Your Opinion!
                                </Run>
                            </Paragraph>
                        </FlowDocument>
                    </RichTextBox>


                    <Button Grid.Row="2" Margin="0 0 10 20"
Grid.Column="1" Style="{StaticResource ProfileButton}"
VerticalAlignment="Bottom" HorizontalAlignment="Right" ToolTip="Registration"
Click="InterprClick" Width="150" Height="30">
                        <TextBlock FontSize="19" Text="Add"
Style="{StaticResource TextBlock}" TextAlignment="Center" Height="26"/>
                    </Button>
                </Grid>
            </TabItem>

            <TabItem Style="{StaticResource TabItem}">
                <TabItem.Header>
                    <Grid VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                        <materialDesign:PackIcon Kind="Upload"
Width="50" Height="50" Background="#e52b50"

Foreground="White"/>
                    </Grid>
                </TabItem.Header>

                <Grid>
                    <TextBlock FontSize="19" Text="Upload Music
To Server" Style="{StaticResource TextBlock}" VerticalAlignment="Top"
TextAlignment="Center" Height="26"/>

                    <TextBlock Margin="-440,30,-10,438"
FontSize="19" Text="Image" Style="{StaticResource TextBlock}"
TextAlignment="Center" Height="26"/>
                    <Button Margin="170 -400 0 0"
Style="{StaticResource ActionButton}" HorizontalAlignment="Left"
ToolTip="Change Image" x:Name="UploadImageFile" Click="ButtonFile_Click">
                        <materialDesign:PackIcon Kind="Folder"
Width="20" Height="20"/>
                    </Button>

                    <Image Margin="11,64,394,230" Height="200"
Width="200" Stretch="Uniform" Source="based/icon.png"
x:Name="UploadMusicPng"/>
```

```
                                <TextBlock Margin="0 140 0 0" FontSize="19"
Text="Name" Width="55" Style="{StaticResource TextBlock}"
HorizontalAlignment="Left" Height="26"/>
                                <TextBox Margin="60 140 0 0" FontSize="20"
Width="156" Height="30"
                                         Style="{StaticResource TextBox}"
HorizontalAlignment="LEft" x:Name="UploadName"/>

                                <TextBlock Margin="0 240 0 0" FontSize="19"
Text="Author" Width="65" Style="{StaticResource TextBlock}"
HorizontalAlignment="Left" Height="26"/>
                                <TextBox Margin="66,319,0,75" FontSize="20"
Width="150" Height="30"
                                         Style="{StaticResource TextBox}"
HorizontalAlignment="Left" x:Name="UploadAuthor"/>

                                <TextBlock Margin="0 320 0 0" FontSize="19"
Text="Description" Style="{StaticResource TextBlock}"
HorizontalAlignment="Left" Height="26"/>
                                <TextBox Margin="4,417,0,37" FontSize="20"
Width="400" Height="30"
                                         Style="{StaticResource TextBox}"
HorizontalAlignment="Left" x:Name="UploadDescr"/>

                                <TextBlock Margin="0,30,179,438"
FontSize="19" Text="Lirycs" Style="{StaticResource TextBlock}"
HorizontalAlignment="Right" Height="26"/>

                                <RichTextBox Margin="0,67,10,0" Width="336"
Height="350" x:Name="UploadSongText" HorizontalAlignment="Right"
VerticalAlignment="Top" Style="{StaticResource RichTextBox}" FontSize="20"
Cursor="IBeam">
                                    <FlowDocument>
                                        <Paragraph>
                                            <Run>
                                                When the user clicks in the
RichTextBox, the selected
            text changes programmatically.
                                            </Run>
                                        </Paragraph>
                                    </FlowDocument>
                                </RichTextBox>

                                <Button Margin="0 0 10 20"
Style="{StaticResource ProfileButton}" VerticalAlignment="Bottom"
HorizontalAlignment="Right" ToolTip="Registration" Click="UploadClick"
Width="150" Height="50">
                                    <TextBlock FontSize="19" Text="Upload"
Style="{StaticResource TextBlock}" TextAlignment="Center" Height="26"/>
                                </Button>
                            </Grid>
                        </TabItem>

                        <TabItem Style="{StaticResource TabItem}"
x:Name="Registration">
                            <TabItem.Header>
                                <Grid VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                                    <materialDesign:PackIcon Kind="Register"
Width="50" Height="50" Background="#e52b50"

Foreground="White"/>
                                </Grid>
```

```
                            </TabItem.Header>

                            <Grid>
                                <TextBlock Margin="0 -400 0 0" FontSize="19"
Text="Login" Style="{StaticResource TextBlock}" VerticalAlignment="Center"
TextAlignment="Center" Height="26"/>
                                <TextBox Margin="0 -330 0 0" FontSize="20"
Width="150" Height="40"
                                         Style="{StaticResource TextBox}"
HorizontalAlignment="Center" x:Name="RegisLogin"/>
                                <TextBlock Margin="0 -270 0 0" FontSize="14"
Text="" Style="{StaticResource TextBlockError}" VerticalAlignment="Center"
TextAlignment="Center" Height="26" x:Name="RegisLogErr"/>


                                <TextBlock Margin="0 -240 0 0" FontSize="19"
Text="Password" Style="{StaticResource TextBlock}" VerticalAlignment="Center"
TextAlignment="Center" Height="26"/>
                                <PasswordBox Margin="0 -170 0 0"
FontSize="20" Width="150" Height="40" PasswordChar="*"
                                             Style="{StaticResource PasswordBox}"
HorizontalAlignment="Center" x:Name="RegisPassword1"/>
                                <PasswordBox Margin="0 -60 0 0" FontSize="20"
Width="150" Height="40" PasswordChar="*"
                                             Style="{StaticResource PasswordBox}"
HorizontalAlignment="Center" x:Name="RegisPassword2"/>
                                <TextBlock Margin="0 0 0 0" FontSize="14"
Text="" Style="{StaticResource TextBlockError}" VerticalAlignment="Center"
TextAlignment="Center" Height="26" x:Name="RegisPassErr"/>


                                <Button Margin="0 80 0 0"
Style="{StaticResource ProfileButton}" HorizontalAlignment="Center"
ToolTip="Registration" Click="RegistrationClick" Width="150" Height="50">
                                    <TextBlock FontSize="19"
Text="Registration" Style="{StaticResource TextBlock}"
VerticalAlignment="Top" TextAlignment="Center" Height="26"/>
                                </Button>
                            </Grid>
                        </TabItem>

                        <TabItem Style="{StaticResource TabItem}"
x:Name="Authorization">
                            <TabItem.Header>
                                <Grid VerticalAlignment="Stretch"
HorizontalAlignment="Stretch">
                                    <materialDesign:PackIcon Kind="Login"
Width="50" Height="50" Background="#e52b50"

Foreground="White"/>
                                </Grid>
                            </TabItem.Header>

                            <Grid>

                                <TextBlock Margin="0 -380 0 0" FontSize="19"
Text="Login" Style="{StaticResource TextBlock}" VerticalAlignment="Center"
TextAlignment="Center" Height="26"/>
                                <TextBox Margin="0 -310 0 0" FontSize="20"
Width="150" Height="40"
                                         Style="{StaticResource TextBox}"
HorizontalAlignment="Center" x:Name="AuthoLogin"/>
```

```
                                <TextBlock Margin="0 -250 0 0" FontSize="14"
Style="{StaticResource TextBlockError}" VerticalAlignment="Center"
TextAlignment="Center" Height="26" x:Name="AuthoLogErr"/>


                                <TextBlock Margin="0 -220 0 0" FontSize="19"
Text="Password" Style="{StaticResource TextBlock}" VerticalAlignment="Center"
TextAlignment="Center" Height="26"/>
                                <PasswordBox Margin="0 -150 0 0"
FontSize="20" Width="150" Height="40" PasswordChar="*"
                                             Style="{StaticResource PasswordBox}"
HorizontalAlignment="Center" x:Name="AuthoPassword" Cursor="IBeam"/>
                                <TextBlock Margin="0 -90 0 0" FontSize="14"
Style="{StaticResource TextBlockError}" VerticalAlignment="Center"
TextAlignment="Center" Height="26" x:Name="AuthoPassErr"/>


                                <Button Margin="0 0 0 0"
Style="{StaticResource ProfileButton}" HorizontalAlignment="Center"
ToolTip="Registration" Click="AuthorizationClick" Width="150" Height="50">
                                    <TextBlock FontSize="19"
Text="Authorization" Style="{StaticResource TextBlock}"
VerticalAlignment="Top" TextAlignment="Center" Height="26"/>
                                </Button>

                                <TextBlock Margin="-50 200 0 0" FontSize="19"
Text="If You Not Yet Account" Style="{StaticResource TextBlock}"
VerticalAlignment="Center" TextAlignment="Center" Height="26"/>
                                <Button Margin="200 200 0 0"
Style="{StaticResource ActionButton}" HorizontalAlignment="Center"
ToolTip="Registration" Click="AuthoToRegistRedirect">
                                    <materialDesign:PackIcon Kind="Login"
Width="20" Height="20"/>
                                </Button>
                            </Grid>
                        </TabItem>
                    </TabControl>
                </Grid>
            </StackPanel>
        </materialDesign:Card>
    </Grid>
</Window>
```

Приложение В

РЕЗУЛЬТАТЫ ФУНКЦИОНАЛЬНОГО ТЕСТИРОВАНИЯ

(рекомендуемое)

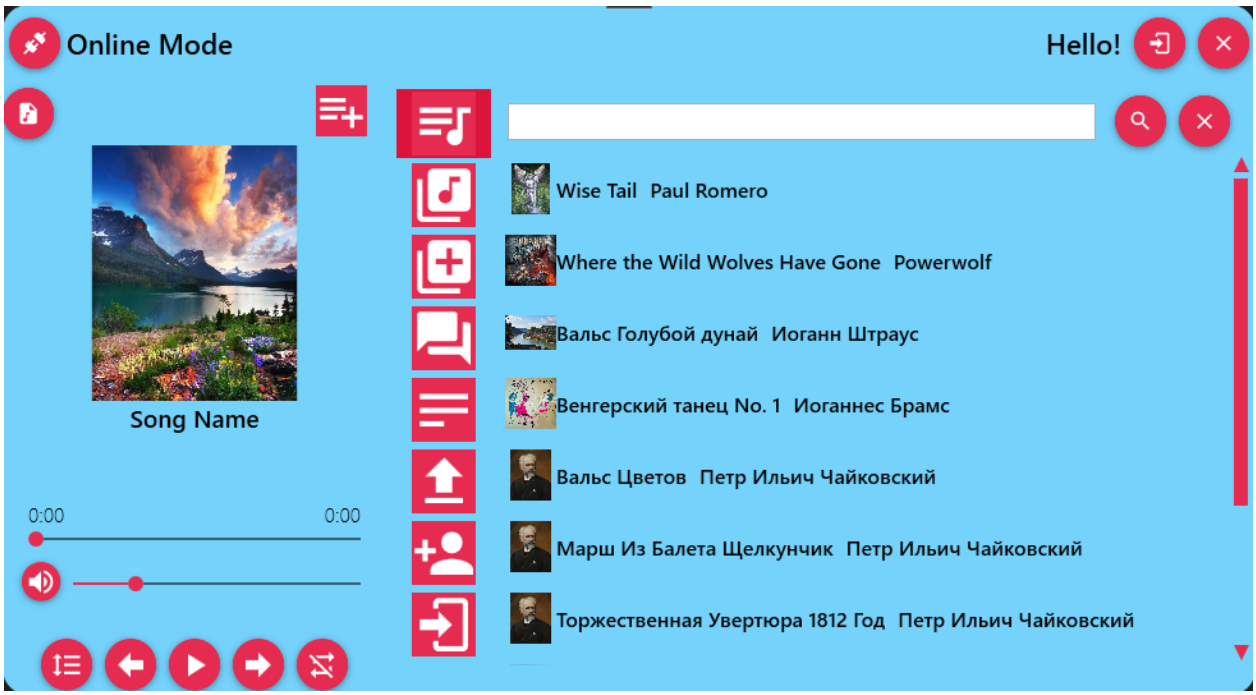

Рисунок В.1 – Установлено соединение с сервером

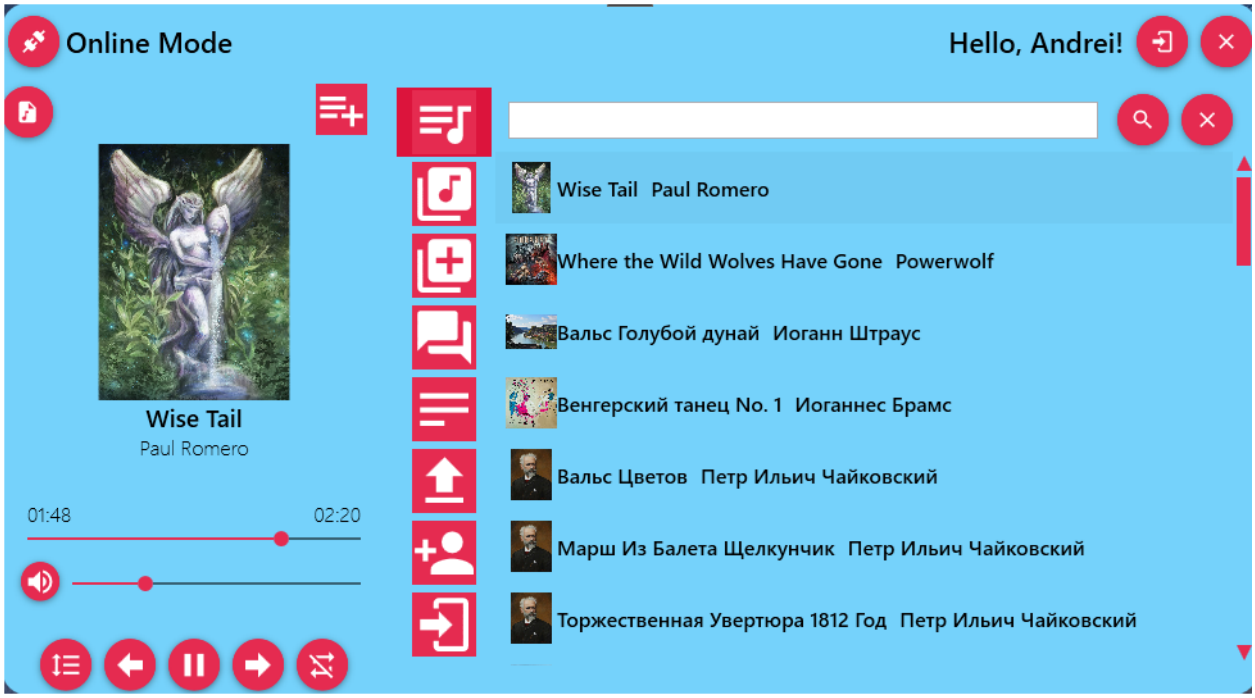

Рисунок В.2 – Воспроизведено аудио с сервера

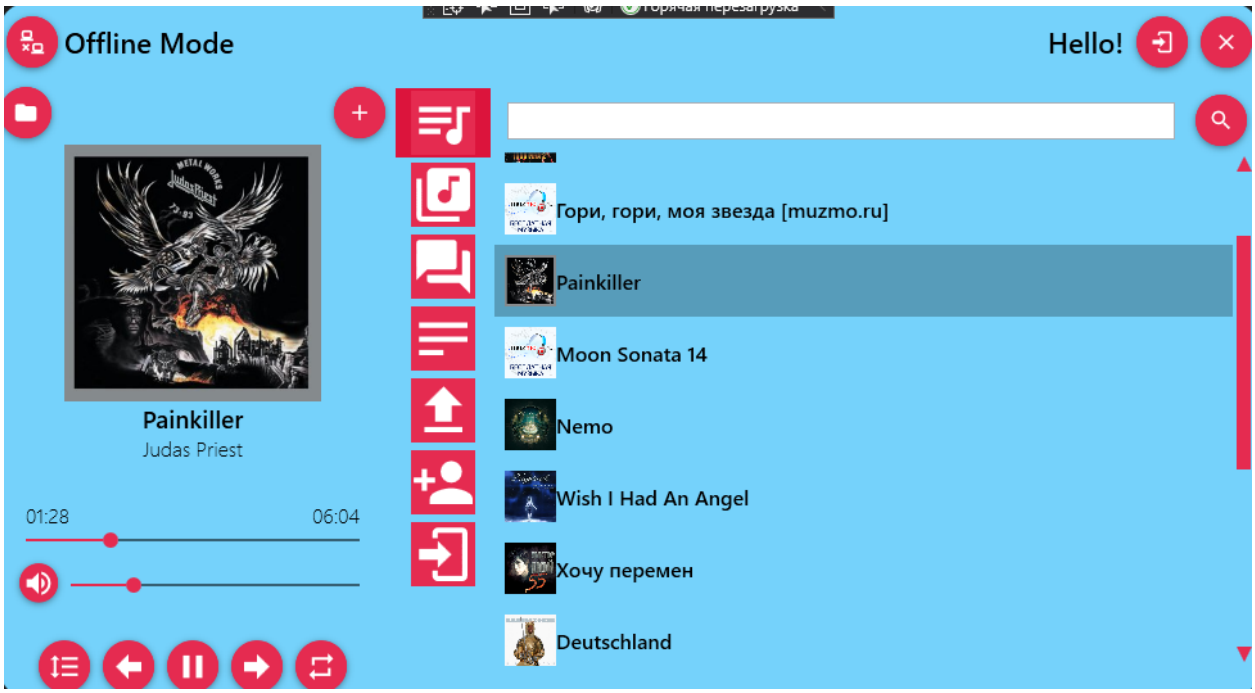

Рисунок В.3 – Воспроизведено аудио с компьютера

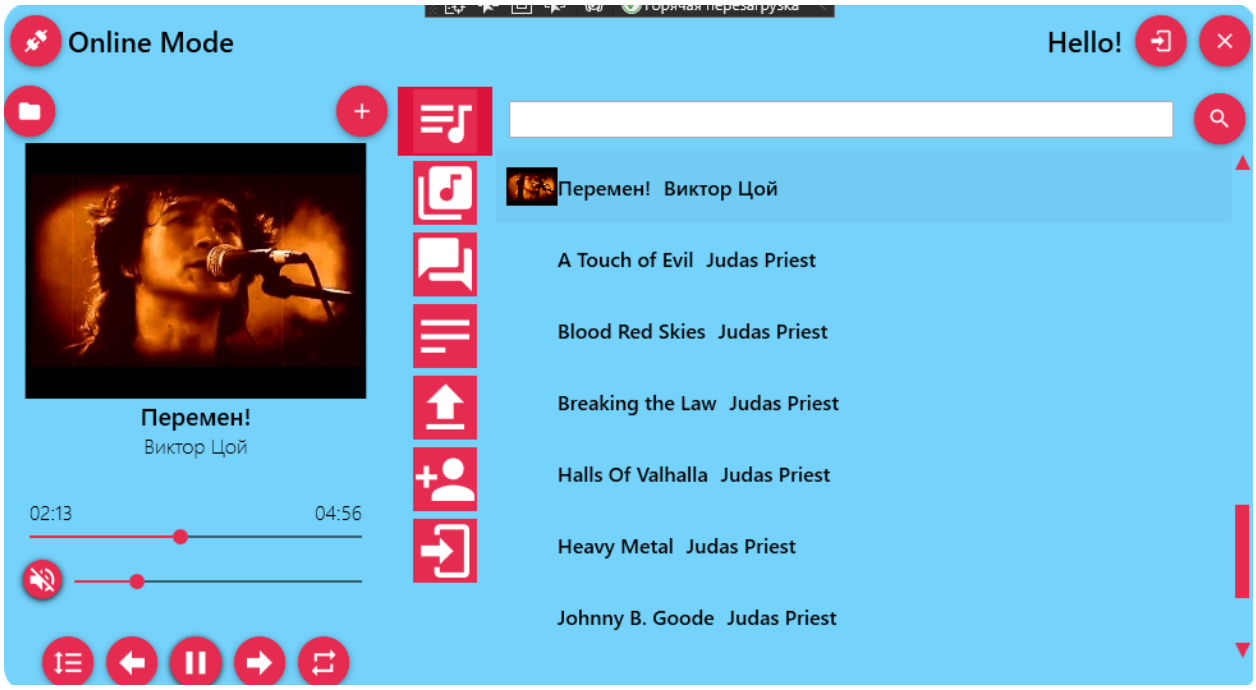

Рисунок В.4 – Переключен звук

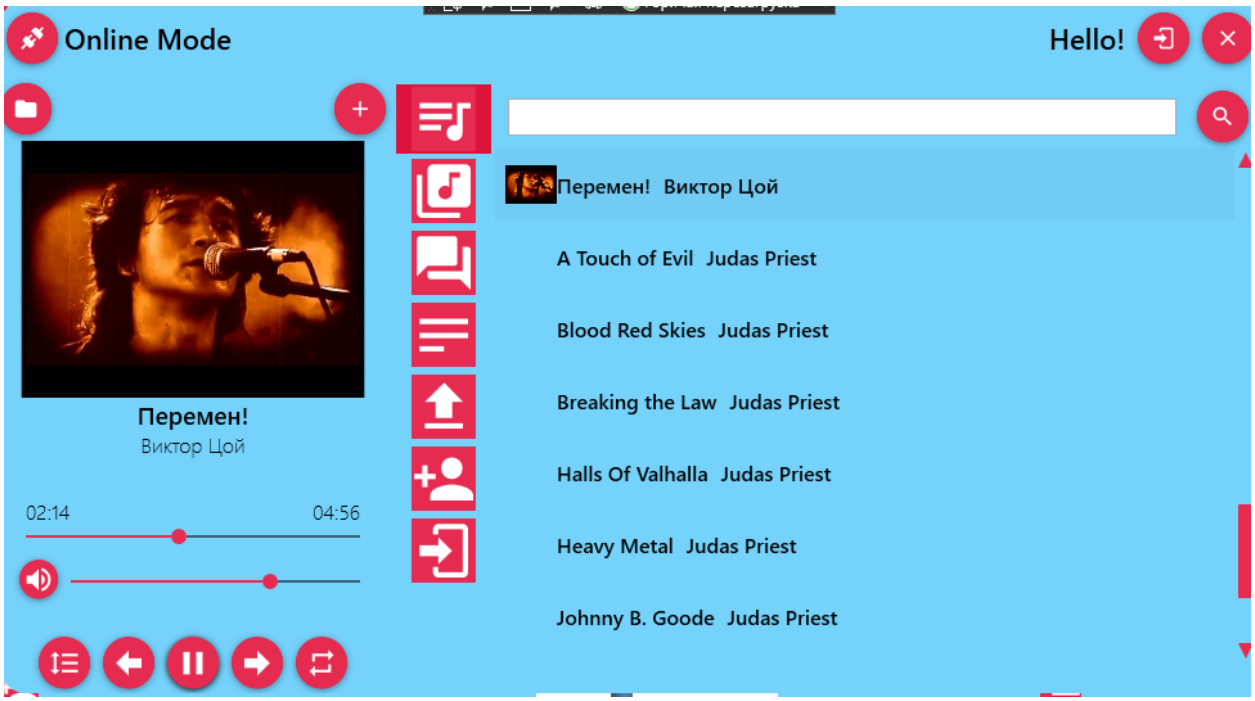

Рисунок В.5 – Изменена громкость

Рисунок В.6 – Поставлена пауза

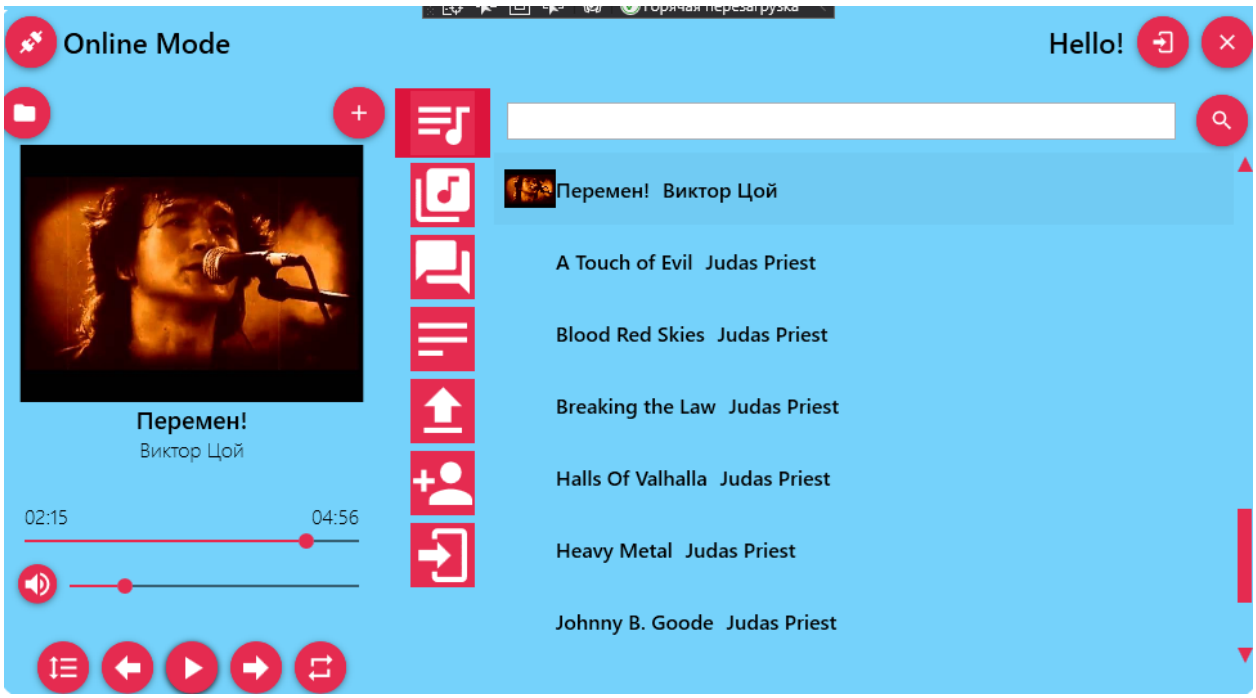

Рисунок В.7 – Перемотано аудио

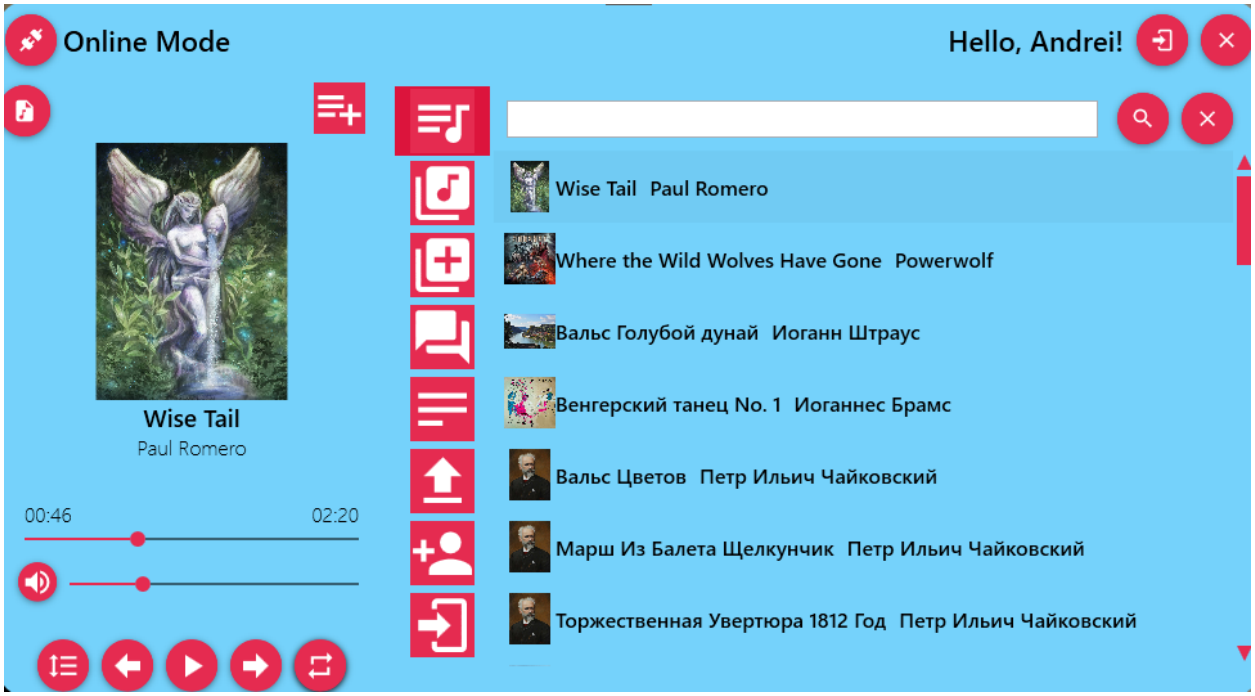

Рисунок В.8 – Поставлен повтор воспроизведения

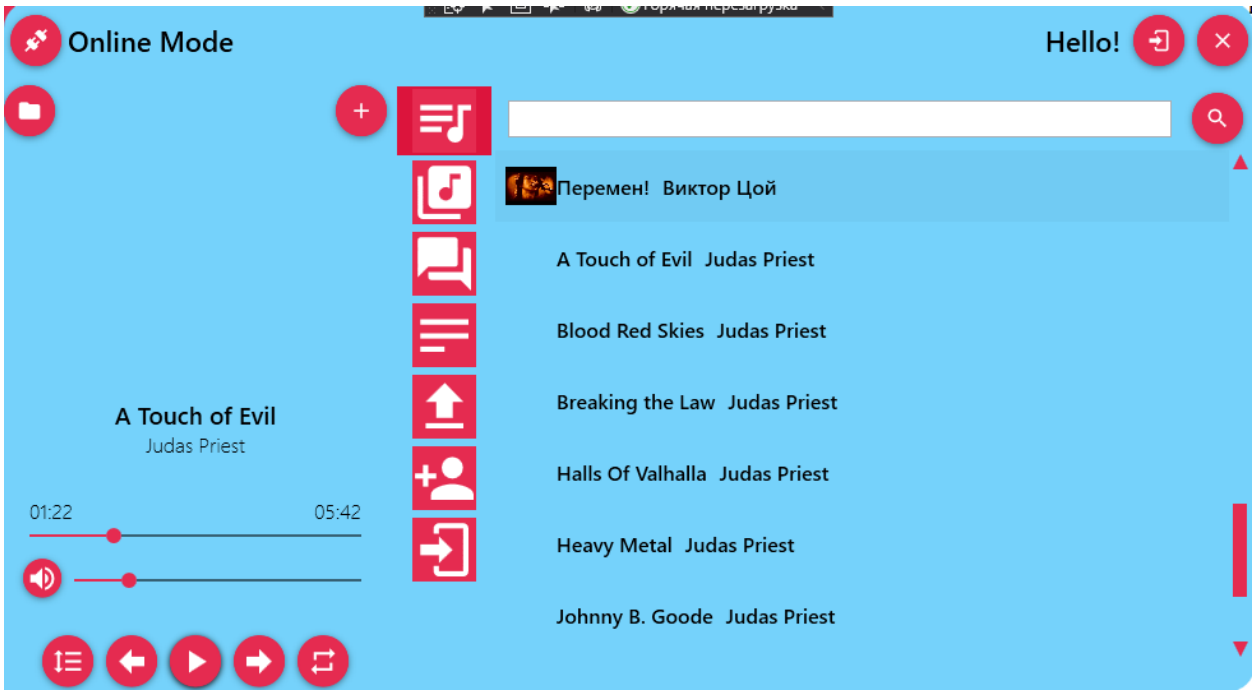

Рисунок В.9 – Воспроизведен следующий файл

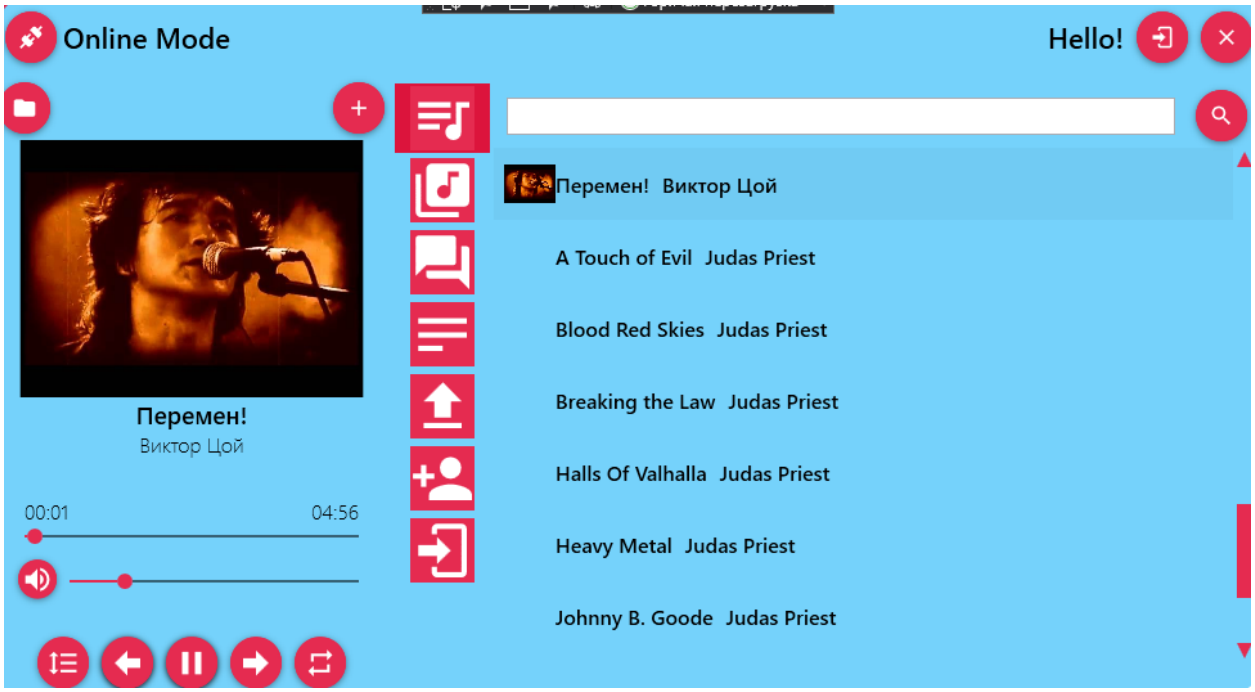

Рисунок В.10 – Воспроизведен предыдущий файл

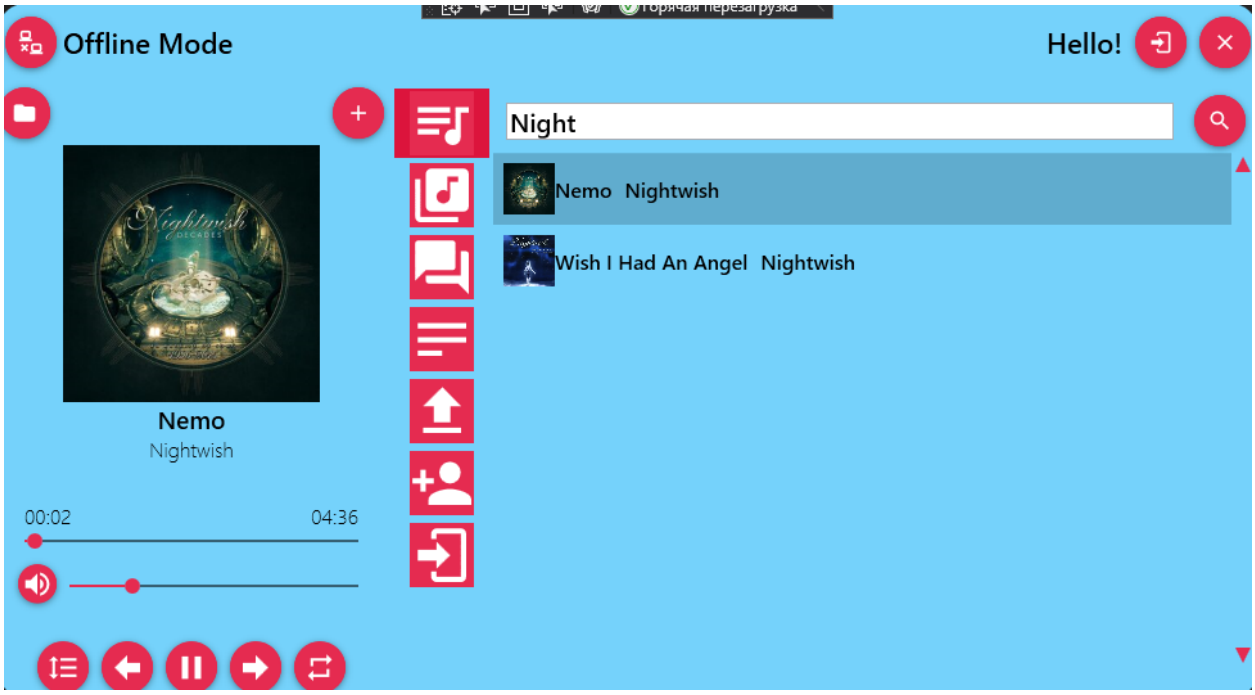

Рисунок В.11 – Осуществлен поиск

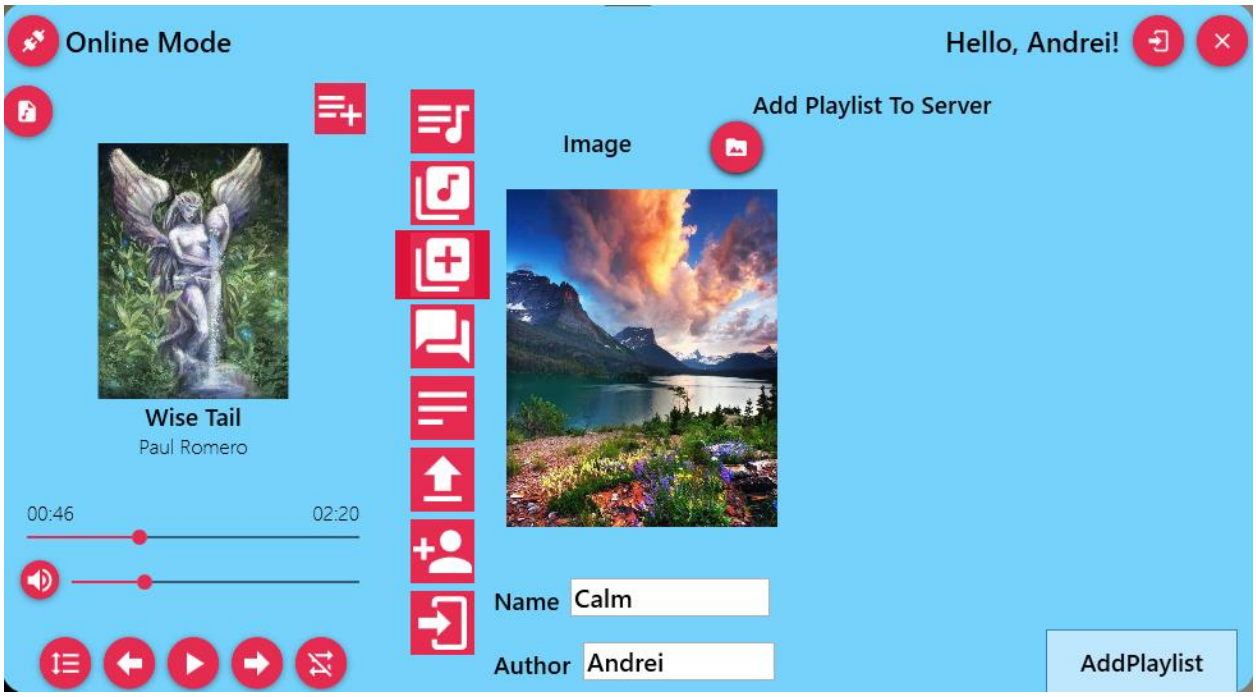

Рисунок В.12 – Добавлен плейлист

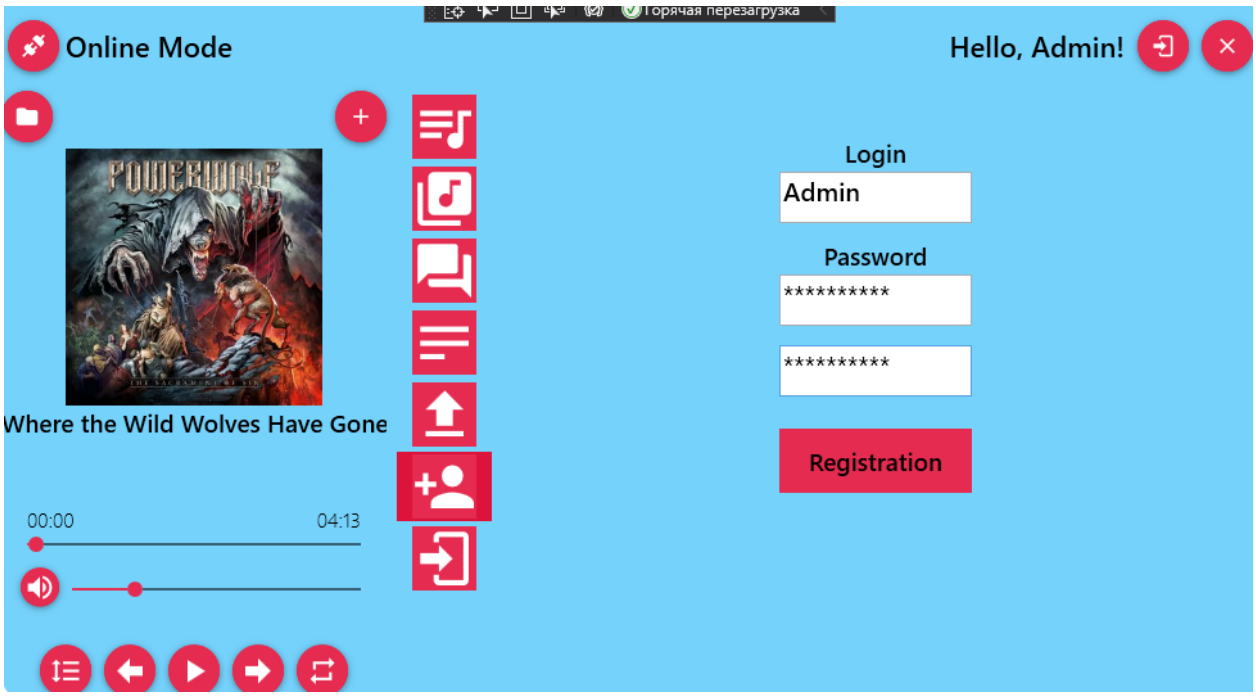

Рисунок В.13 – Успешная регистрация

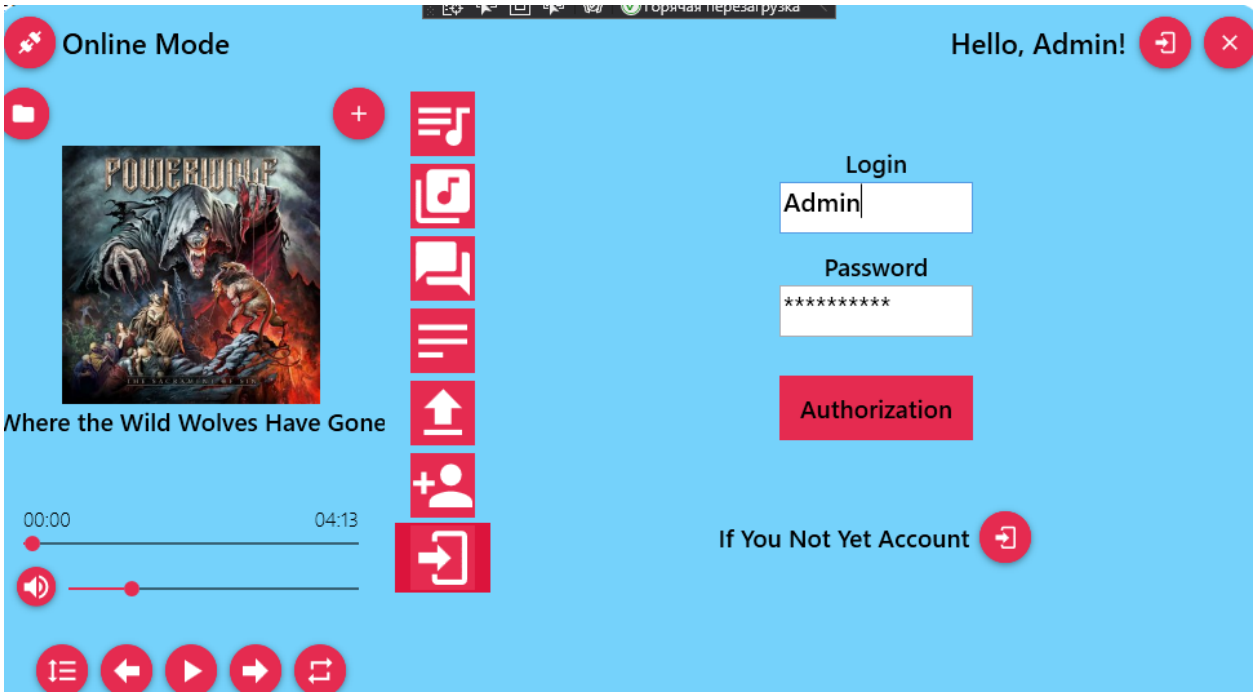

Рисунок В.14 – Успешная авторизация

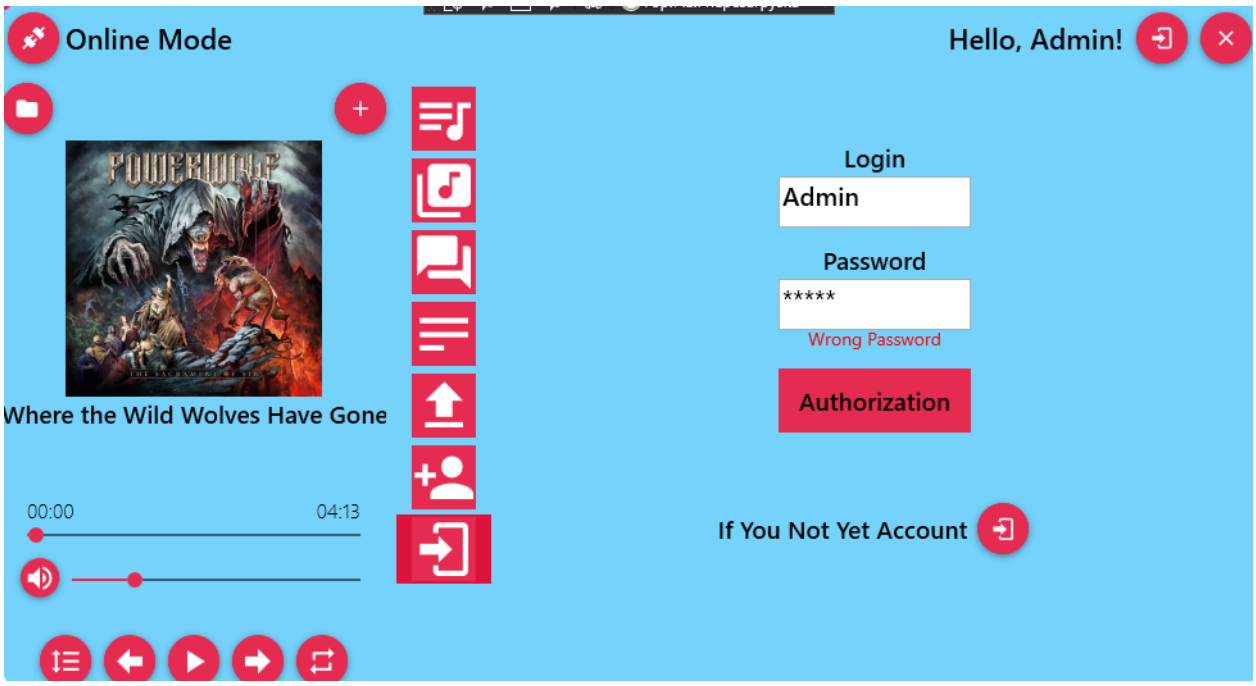

Рисунок В.15 – Неудачная авторизация

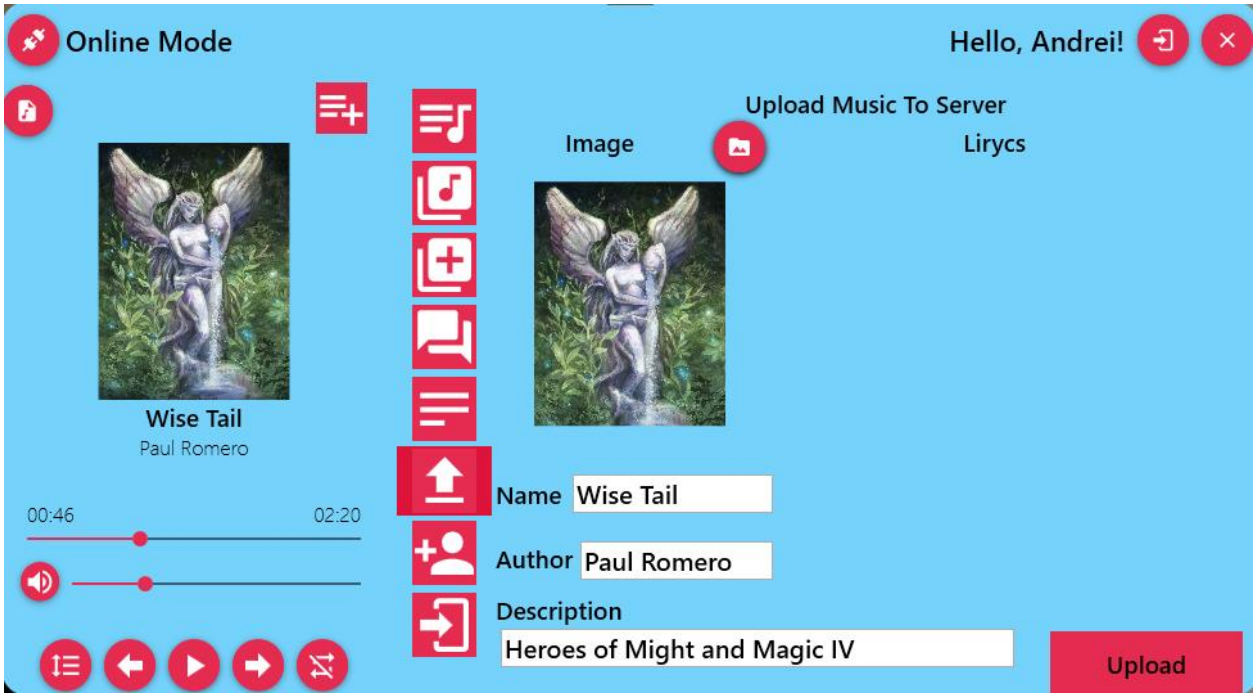

Рисунок В.16 – Файл загружен на сервер

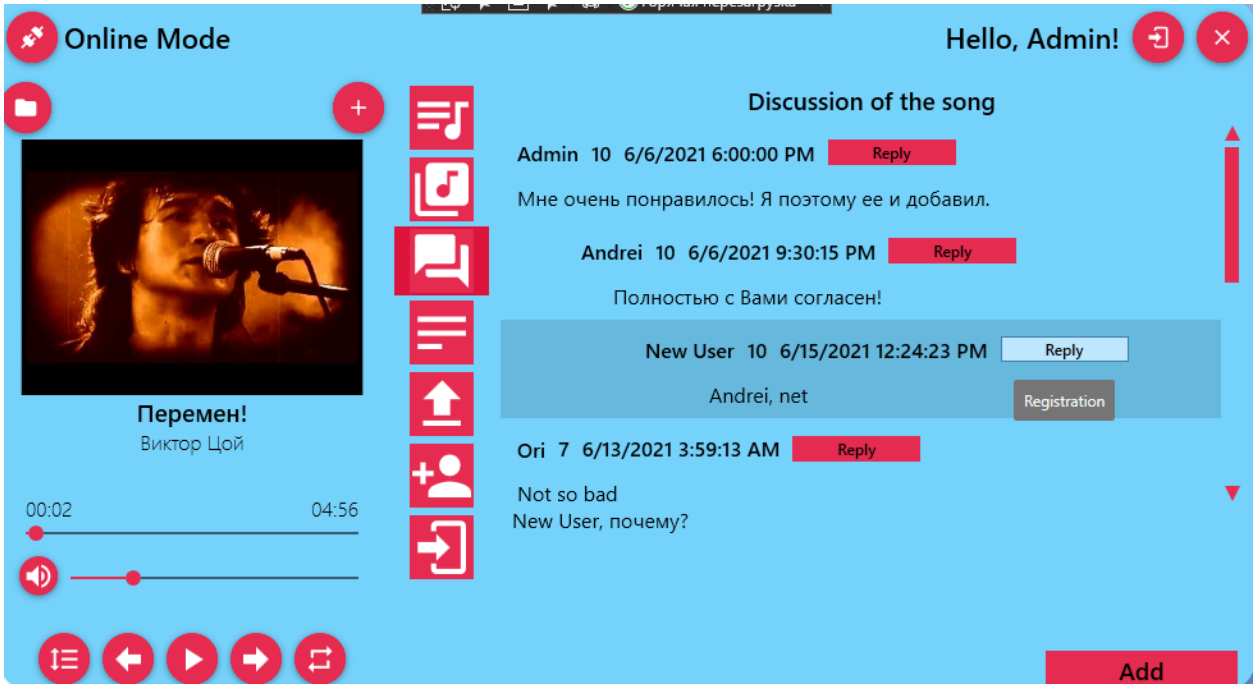

Рисунок В.17 – Добавлен комментарий шаг 1

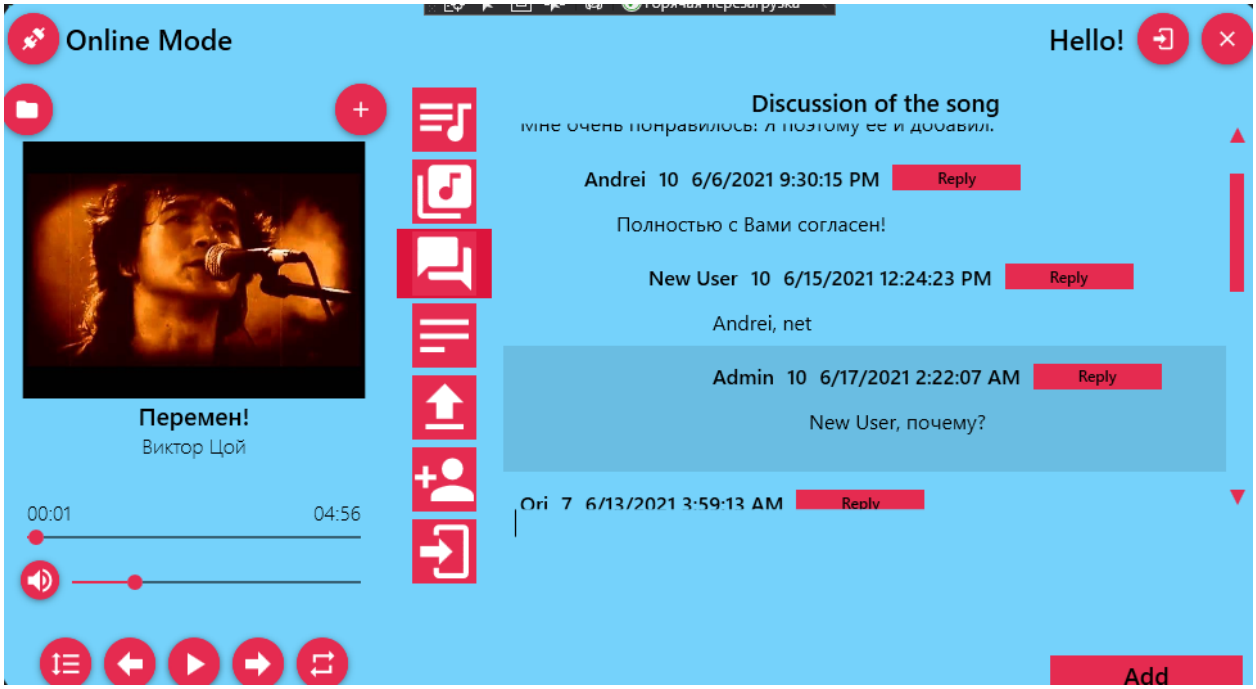

Рисунок В.18 – Добавлен комментарий шаг 2
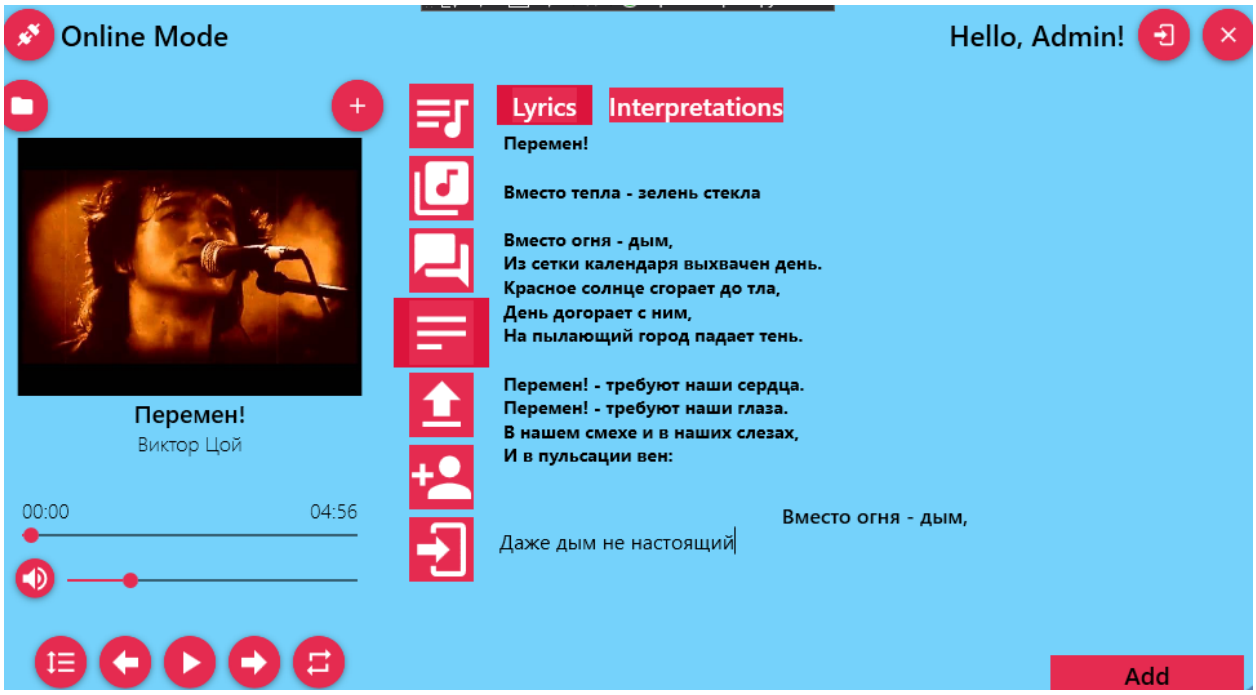
Рисунок В.19 – Добавлена трактовка шаг 1

Рисунок В.20 – Добавлена трактовка шаг 2