МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Информационных систем и технологий

Специальность 1-40 05 01 «Информационные системы и технологии»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема «Музыкальное приложение»

Исполнитель

студент (ка) 3 курса группы 3 Шутро А.С.

(Ф.И.О.)

Руководитель работы асс. Курилец А.В.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_ Курилец А.В.

подпись дата инициалы и фамилия

Минск 2024

**Содержание**

[Введение 3](#_Toc185966917)

[1. Аналитический обзор прототипов и литературных источников 4](#_Toc185966918)

[1.1 Анализ прототипов 4](#_Toc185966919)

[1.1.1 «Spotify» 4](#_Toc185966920)

[1.1.1 «Яндекс.Музыка» 6](#_Toc185966921)

[1.2 Постановка задачи 7](#_Toc185966922)

[2. Анализ требований к программному средству и разработка функциональных требований 8](#_Toc185966923)

[2.1 Спецификация функциональных требований к программному средству 8](#_Toc185966924)

[3. Проектирование программного средства 10](#_Toc185966925)

[3.1 Архитектура системы 10](#_Toc185966926)

[3.2 Взаимоотношения между классами 12](#_Toc185966927)

[3.3 Модель базы данных 12](#_Toc185966928)

[4. Реализация программного средства 14](#_Toc185966929)

[4.1 Реализация Media 14](#_Toc185966930)

[4.2 Реализация базы данных 15](#_Toc185966931)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 17](#_Toc185966932)

[5.1 Тестирование авторизации и регистрации 17](#_Toc185966933)

[6. Руководство по установке и использованию 20](#_Toc185966934)

[Заключение 23](#_Toc185966935)

[Список литературы 24](#_Toc185966936)

[Приложение А 25](#_Toc185966937)

[Приложение В 26](#_Toc185966938)

Введение

В условиях стремительного развития технологий пользователям становится всё сложнее найти удобные и функциональные музыкальные приложения. Несмотря на популярность онлайн-сервисов, многие предпочитают хранить свои музыкальные коллекции на локальных устройствах. Однако существующие приложения часто не соответствуют требованиям пользователей, которые хотят делится треками с друзьями не просто по мессенджерам, а по приложению. Это подчеркивает актуальность разработки программного обеспечения, способного эффективно воспроизводить музыку, добавляя ее в общий плейлист пользователей.

Цель данного курсового проекта – разработать музыкальное приложение, который станет удобным инструментом для воспроизведения аудиофайлов, поиска артистов, а также предоставит простой и функциональный интерфейс для пользователей.

Основные задачи, которые должен выполнять создаваемый проект, включают в себя:

* + Регистрация пользователей.
  + Поиск артистов.
  + Воспроизведение треков.
  + Загрузка треков с ПК.
  + Загрузка треков по URL с платформы YouTube.

Для реализации поставленных задач будут использоваться язык программирования C# и технология WPF (Windows Presentation Foundation), которая позволяет создавать современные пользовательские интерфейсы. Хранение данных пользовательских настройках будет осуществляться с использованием базы данных SQLite, которая является легковесной, но мощной СУБД, идеально подходящей для настольных приложений.

Разработка данного приложения позволит создать современное музыкальное приложение, отвечающий всем требованиям пользователей, которые предпочитают воспроизведение музыки в плейлисте.

1. Аналитический обзор прототипов и литературных источников

* 1. Анализ прототипов

В настоящее время существует множество программ для воспроизведения локальной музыки, и такие программные решения значительно упрощают управление музыкальной библиотекой и плейлистами. Использование современных технологий актуально при создании приложений на новой технологической основе. Для четкого определения задач курсового проекта по разработке музыкального приложения необходимо провести анализ существующих решений.

Анализ существующих решений позволяет выявить лучшие практики и определить ключевые функции, которые должны быть реализованы в создаваемом приложении. Это поможет создать удобный и функциональный проигрыватель, который обеспечит качественное воспроизведение аудиофайлов, удобную навигацию по музыкальной библиотеке, управление плейлистами и поддержку работы с метаданными.

При изучении программ-прототипов было выявлено несколько популярных решений, схожих по функциональности. Все они предназначены для воспроизведения локальной музыки и управления музыкальной коллекцией, предлагая пользователям базовые и продвинутые возможности для работы с аудиофайлами.

* + 1. «Spotify»

В центральной части главной страницы находится лента рекомендаций, включающая персонализированные плейлисты, такие как "Your Daily Mix", "Discover Weekly" и "Release Radar". Здесь же представлены популярные альбомы, подкасты и подборки, соответствующие интересам пользователя. Каждый элемент сопровождается изображением обложки, названием и именем исполнителя.

В верхней части главной страницы расположены кнопки для быстрого возврата к недавно прослушанным трекам и управления профилем пользователя.

На левой стороне экрана находится вертикальное меню для быстрого доступа к основным разделам приложения:

В нижней части интерфейса расположена панель управления воспроизведением, включающая:

* Кнопки воспроизведения, паузы и переключения треков.
* Индикатор текущего трека и прогресса воспроизведения.
* Регулировку громкости.
* Кнопку для добавления трека в избранное.
* Функцию подключения к другим устройствам для воспроизведения.
* Дополнительные функции

На главной странице предусмотрены разделы для сортировки треков, например, по жанрам или настроению. Пользователь также может найти тематические подборки, такие как музыка для тренировок или расслабления.

Интерфейс Spotify представлен на рисунке 1.1.

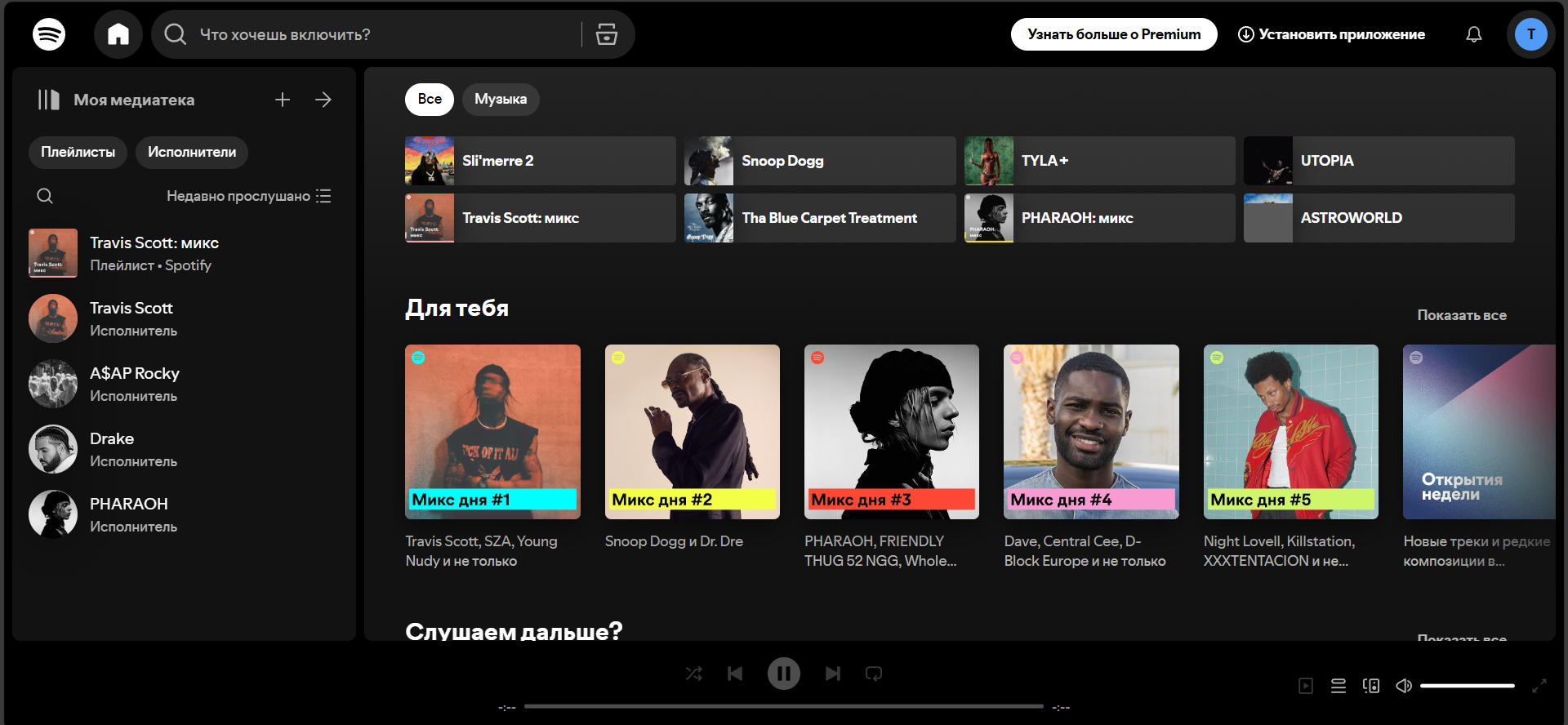


Рисунок 1.1 – Интерфейс Spotify

Достоинства Spotify:

* Простота использования благодаря интуитивному дизайну.
* Персонализированные рекомендации на основе истории прослушиваний.
* Поддержка работы на различных устройствах с синхронизацией.
* Возможность делиться музыкой и плейлистами.

Недостатки Spotify:

* Ограниченные возможности кастомизации интерфейса.
* Реклама в бесплатной версии, которая может отвлекать.
* Нет встроенных инструментов для работы с метаданными или обложками.
* Сложности с навигацией в больших библиотеках треков.
* Часть функций доступна только в премиум-версии.

1.1.2 «Яндекс.Музыка»

Интерфейс музыкального проигрывателя «Яндекс.Музыка» выполнен в светлой, но ненавязчивой цветовой гамме (с возможностью переключения в тёмный режим), где внимание пользователя акцентируется на обложках альбомов, плейлистах и удобной навигации. Основной упор сделан на персонализацию и лёгкость в использовании.

Главная страница представляет собой ленту персонализированных рекомендаций и популярных подборок:

1. Персонализированные плейлисты:

* "Моя волна" — бесконечный поток треков, подобранных на основе предпочтений.
* Плейлисты дня — ежедневные подборки, созданные для разных настроений и случаев.

1. Популярное сейчас:
   * Тренды и новинки, собранные редакторами или на основе пользовательских предпочтений в регионе.
2. Рекомендации:
   * Музыка, которую пользователь ещё не слушал, но она может понравиться.
3. Подкасты и аудиокниги:
   * Рекомендации по жанрам и популярные выпуски. Главное — лента рекомендаций и подборок.

Основное пространство интерфейса занято плиткой с обложками альбомов, треков и плейлистов. У каждой плитки указаны название, исполнитель и дополнительные параметры, такие как год выпуска или продолжительность.

Интерфейс Яндекс.Музыки представлен на рисунке 1.2

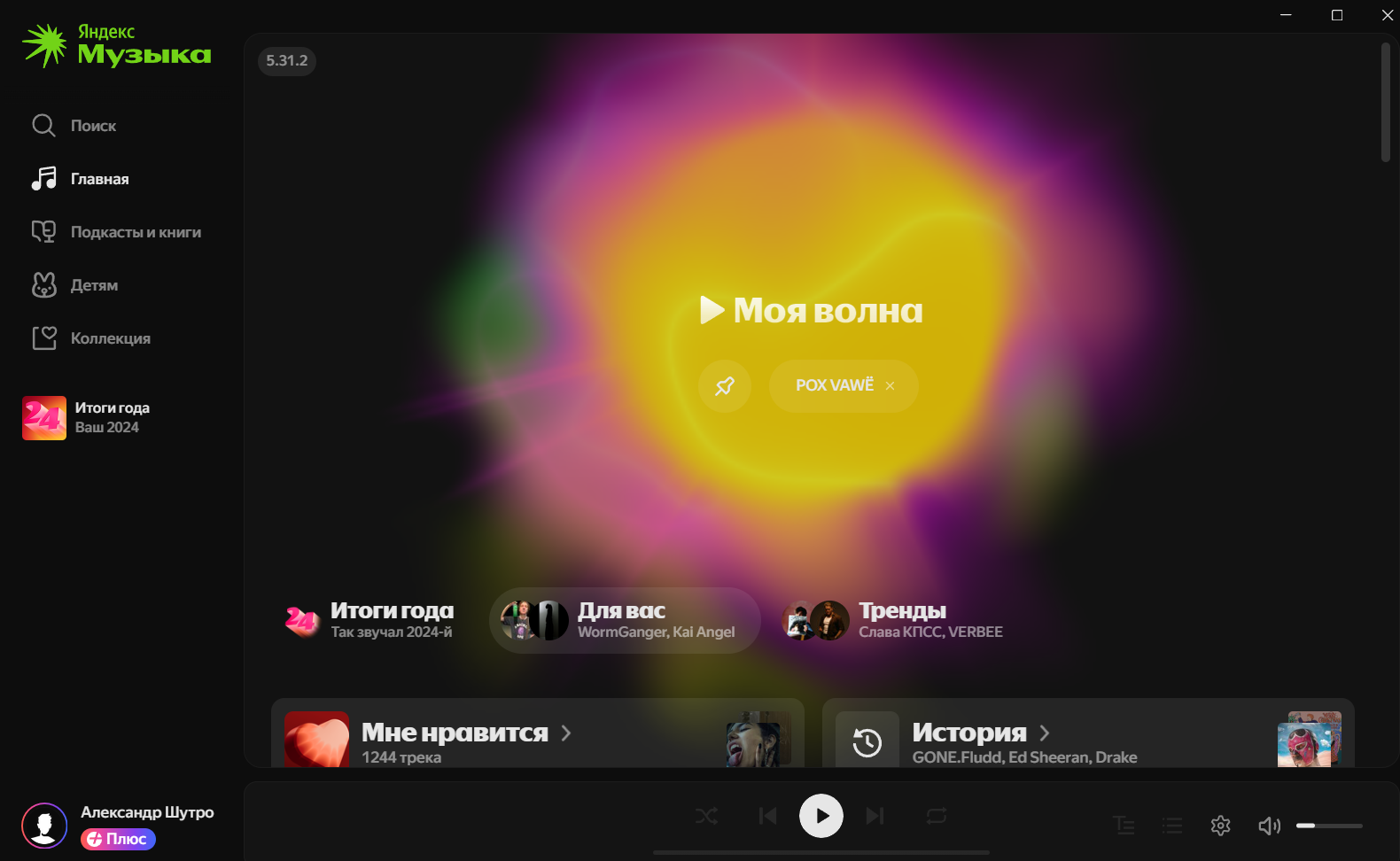


Рисунок 1.2 ­– Интерфейс «Яндекс.Музыки»

Достоинства «Яндекс.Музыки»:

* + Персонализация рекомендаций (например, "Моя волна").
  + Интеграция с умными устройствами и голосовым ассистентом Алисой.
  + Широкий выбор подкастов и аудиокниг.
  + Интуитивно понятный интерфейс.
  + Возможность кастомизации (тёмная тема, управление плейлистами).

Недостатки «Яндекс.Музыки»:

* + Ограниченная библиотека музыки в сравнении с международными сервисами.
  + Некоторый контент доступен только в России или странах СНГ.
  + Реклама в бесплатной версии.
* Отсутствие поддержки Lossless и других аудиоформатов высокого качества.

1.2 Постановка задачи

После анализа существующих музыкальных приложений и их функционала, а также изучения преимуществ и недостатков аналогичных решений, можно выделить основные задачи, которые должно решать разрабатываемое программное средство.

Основные задачи:

* разработать функционал для поиска артистов и просмотра их слушателей;
* создать каталог треков с возможностью просмотра и прослушивания, с возможностью добавления треков в плейлист;
* разработать функционал для управления плейлистом: добавление, удаление треков в плейлистах, а также поддержка воспроизведения аудиофайлов;

При разработке музыкального приложения необходимо учитывать требования пользователей, обеспечить удобство работы с интерфейсом, а также применить современные технологии и методы программирования для повышения производительности и безопасности использования.

2. Анализ требований к программному средству и разработка функциональных требований

Анализ требований — это процесс понимания, документирования, оценки и управления требованиями к системе. Цель анализа требований — создать четкое и понятное описание того, что должна делать система, каким образом и какие требования необходимы, чтобы система была полезной и эффективной для пользователей и бизнеса.

Этот процесс включает несколько этапов:

* сбор и документирование требований;
* анализ требований на противоречивость, неполноту и избыточность;
* разработка модели требований;
* оценка и проверка требований;
* управление требованиями в течение жизненного цикла проекта.

Анализ требований является ключевым этапом в разработке программного средства, поскольку он определяет, что должна сделать система, и гарантирует, что она будет соответствовать нуждам пользователей и бизнеса. Правильный анализ требований позволяет избежать ошибок в процессе разработки и значительно улучшить конечный продукт. Он обеспечивает точное понимание задач, которые система должна решать, и помогает разработчикам создавать эффективные, качественные решения.

При этом важно учитывать как функциональные, так и нефункциональные требования. Функциональные требования описывают, что должна делать система, а нефункциональные — как это должно быть реализовано. Например, требования к производительности, безопасности и удобству работы с системой. Оба типа требований должны быть согласованы и документированы для обеспечения успешной реализации проекта.

2.1 Спецификация функциональных требований к программному средству

Программное средство разрабатывается для использования несколькими пользователями, которые используют приложение для воспроизведения и управления музыкальными файлами. Все функции ориентированы на обеспечение удобного и простого интерфейса для нескольких пользователей.

Программное средство должно предоставлять следующие функциональные возможности для пользователя:

* просмотр каталога музыки: Пользователь должен иметь возможность просматривать доступные музыкальные треки и исполнителей в удобной и организованной форме;
* поиск и сортировка: Программное средство должно предоставлять возможность поиска артистов;
* функции воспроизведения музыки: Приложение должно позволять воспроизводить музыкальные треки, управлять воспроизведением (пауза, перемотка), а также регулировать громкость;
* просмотр информации о треках: Пользователь должен иметь доступ к детальной информации о треках, включая название, исполнителя и продолжительность.

Функциональные возможности приложения направлены на создание комфортного и простого интерфейса для конечного пользователя. Регистрация пользователей предназначена для добавления песен, которые могут понравится другим пользователем приложения.

Эта спецификация устанавливает минимальный набор требований к программному обеспечению и служит основой для дальнейшей разработки, тестирования и внедрения продукта.

3. Проектирование программного средства

3.1 Архитектура системы

В данном курсовом проекте реализована архитектура, обеспечивающая удобство разработки и гибкость настройки интерфейса приложения. Основное внимание уделено разделению логики приложения и визуальной части для обеспечения легкости в сопровождении и масштабировании.

Ключевые аспекты архитектуры:

* модель описывает основные данные, используемые в приложении, такие как музыкальные треки, загрузчик музыки с платформы YouTube и настройки;
* связь данных реализуется через четкое взаимодействие между элементами, что обеспечивает обновление интерфейса в реальном времени.

Программное средство «Музыкальный приложение» имеет логичную структурированную архитектуру, что делает его удобным для персонального использования. Визуальная схема архитектуры представлена на рисунке 3.1.





Рисунок 3.1 – Структура проекта

Описание структуры основных папок проекта представлено в таблице 3.1.

|  |  |
| --- | --- |
| Имя файла/папки | Содержание |
| Db | Контекст базы данных для пользователей. |
| Helpers | Поиск музыки по Spotify API. |
| Image | Графические ресурсы приложения. |
| Models | Модели для работы с пользователями, поиском, и данными. |
| styles | Стилизация кнопок приложения. |
| Utilities | Графические ресурсы приложения. |
| View | Представления для визуализацию данных и взаимодействие с пользователем |
| ViewModel | Модели представлений для обеспечения связи между пользовательским интерфейсом и бизнес-логикой, управляет состоянием интерфейса, обрабатывает команды. |

Таблица 3.1 – Описание структуры папок и файлов проекта

Таким образом, сформированная таблица помогает понять общую структуру проектируемого программного средства.

3.2 Взаимоотношения между классами

Для представления внутренней структуры программы в виде классов и связей между ними используется диаграмма классов. Диаграмма классов представлена в Приложении Б.

3.3 Модель базы данных

Разрабатываемое приложение использует единственную таблицу базы данных под названием **Users** для хранения информации, связанной с регистрацией и авторизацией пользователей. Эта таблица построена на базе реляционной системы управления базами данных SQLite и интегрирована в приложение с использованием Entity Framework Core. Такой подход обеспечивает стабильность, удобство работы с данными и гибкость при дальнейшей разработке приложения. Диаграмма базы данных и скрипт для создания представлены в приложении B.

Модель базы данных изображена на рисунке 3.2.



Рисунок 3.2 – Структура таблицы users

Таблица, содержащая информацию о зарегистрированных пользователях приложения, играет важную роль в обеспечении функциональности системы. Она содержит следующие поля:

* Name (уникальное имя пользователя):
* Является первичным ключом таблицы.
* Гарантирует уникальность каждого пользователя в системе.
* Используется для идентификации учетной записи.
* Password (пароль пользователя):
* Служит для аутентификации.
* Хранится в зашифрованной форме для обеспечения безопасности данных.

Поле Name является первичным ключом, указывающим путь к песне. Также в таблице есть поля для названия песни, исполнителя, продолжительности и изображения обложки.

4. Реализация программного средства

4.1 Реализация Media

Была выбрана библиотека Windows "Media"

Стандартный инструмент Windows: библиотека предоставляет мощные и хорошо протестированные возможности для работы с мультимедиа.

Простота интеграции: легко связывается с элементами WPF, такими как MediaPlayer.

Высокая производительность: оптимизирована для работы в среде Windows, что снижает затраты на обработку мультимедийных данных.

Поддержка современных форматов: гарантирует, что приложение будет работать с популярными аудиоформатами без необходимости использования сторонних библиотек.

В проекте для работы с мультимедийным контентом была реализована следующая функциональность:

Воспроизведение аудио:

Использован объект WPF MediaPlayer, который встроен в библиотеку Windows "Media".

Настроены базовые функции для воспроизведения, паузы, остановки и изменения громкости.

Работа с путями к файлам:

Для каждого аудиофайла из базы данных получен путь к его местоположению.

Путь передаётся в MediaPlayer через переменную MusicPath.

Обработка событий MediaElement:

Настроены события, такие как MediaOpened (открытие файла) и MediaEnded (завершение трека), для управления логикой воспроизведения, например, автоматическое переключение на следующий трек.

UI для плеера:

Создан пользовательский элемент управления (UserControl) песен. Каждый элемент содержит обложку, название песни и кнопку для воспроизведения.

Связь с базой данных:

При загрузке данных они передаются в элементы управления для отображения пользователю.

Реализация MediaPlayer представлена в листинге 4.1.

|  |
| --- |
| {  InitializeComponent();  var vm = (PlayerVM)DataContext;  mediaPLayer = vm.Player;  Timer = vm.Timer;  }  private void slider\_PreviewMouseLeftButtonDown(object sender, MouseButtonEventArgs e)  {  var PlayerVM = (PlayerVM)DataContext;  if (PlayerVM!.Player.HasAudio)  {  PlayerVM!.Player.Pause();  if (PlayerVM!.Timer != null)  PlayerVM!.Timer!.Stop();  }  } |

Листинг 4.1 – Реализация MediaPlayer

4.2 Реализация базы данных

Для создания базы данных и таблиц в проекте с используется Entity Framework Core и SQLite, а так же выполняются следующие шаги:

*Создание контекста базы данных:*

В классе MusicDbContext наследуемся от DbContext и определяем DbSet для каждой сущности, которая будет представлена таблицей в базе данных. В данном случае создаётся таблица для пользователей:

DbSet<User> users — таблица для хранения пользователей.

*Настройка подключения к базе данных:*

В методе OnConfiguring настраиваем подключение к базе данных SQLite с помощью строки подключения:

optionsBuilder.UseSqlite("Data Source= {App.path}\\DataFile.db");. база данных будет храниться в файле DataFile.db, путь к которому формируется с использованием статического свойства App.path

5. Тестирование, проверка работоспособности и анализ полученных результатов

5.1 Тестирование авторизации и регистрации

После запуска приложения пользователь оказывается на странице авторизации. Эта страница играет ключевую роль в обеспечении безопасности данных, управлении доступом и создании персонализированного пользовательского опыта. Она служит начальной точкой для работы с приложением и предоставляет два основных сценария: вход в уже существующую учетную запись или переход на страницу регистрации для создания новой учетной записи.

На странице авторизации отображается форма для ввода учетных данных, таких как:

* Имя пользователя (Name):

Поле для ввода уникального имени пользователя, которое было задано при регистрации.

* Пароль (Password):

Поле для ввода пароля. Для повышения безопасности вводимые символы скрываются (отображаются точками).

После ввода данных система проверяет их корректность: проверяется наличие имени пользователя в базе данных и выполняется сверка введенного пароля, хранящимся в базе данных.

Если данные введены корректно, пользователь автоматически перенаправляется на главную страницу приложения, где он может начать работу с функционалом, доступным после авторизации.

Если пользователь еще не имеет учетной записи, он может перейти на страницу регистрации через ссылку или кнопку на странице авторизации.

На странице регистрации пользователь заполняет форму с полями:

* Имя пользователя (Name):

Уникальное имя, которое будет использоваться для входа в систему. Система проверяет, не занято ли имя другим пользователем.

* Пароль (Password):

Пользователь создает надежный пароль. Система может предложить рекомендации, такие как минимальная длина пароля.

Система проверяет введенные данные на соответствие заданным критериям:

* Уникальность имени пользователя.
* Соответствие пароля требованиям безопасности.

После успешного заполнения формы и ее отправки пользовательская информация сохраняется в базе данных. Затем система автоматически перенаправляет пользователя на страницу авторизации.

Страницы авторизации и регистрации представлены на рисунках 5.1 и 5.2

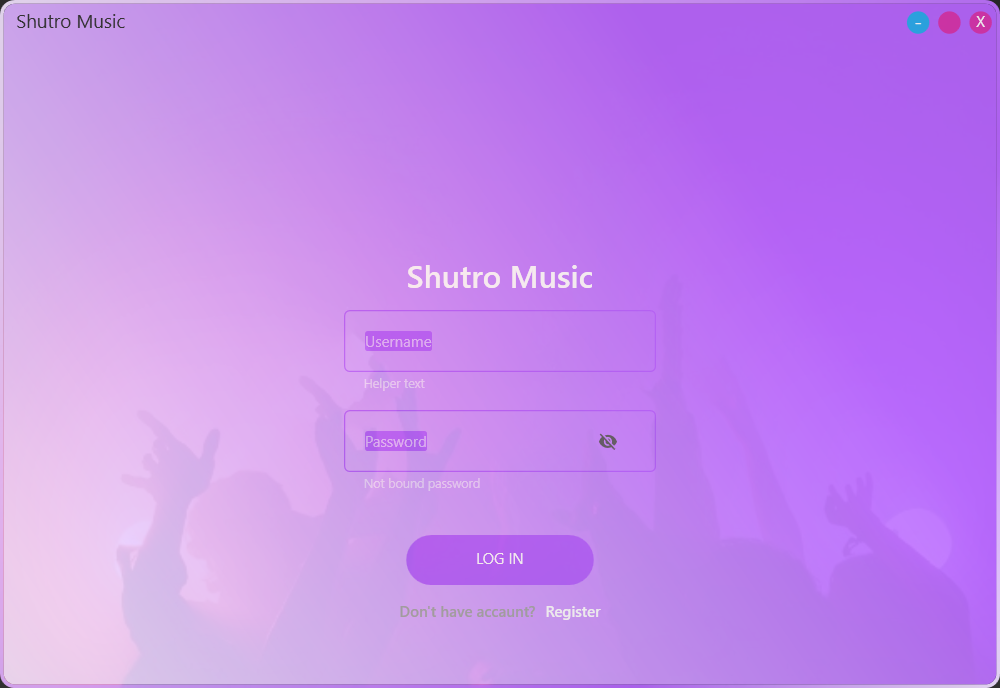
****

Рисунок 5.1 – Страница авторизации

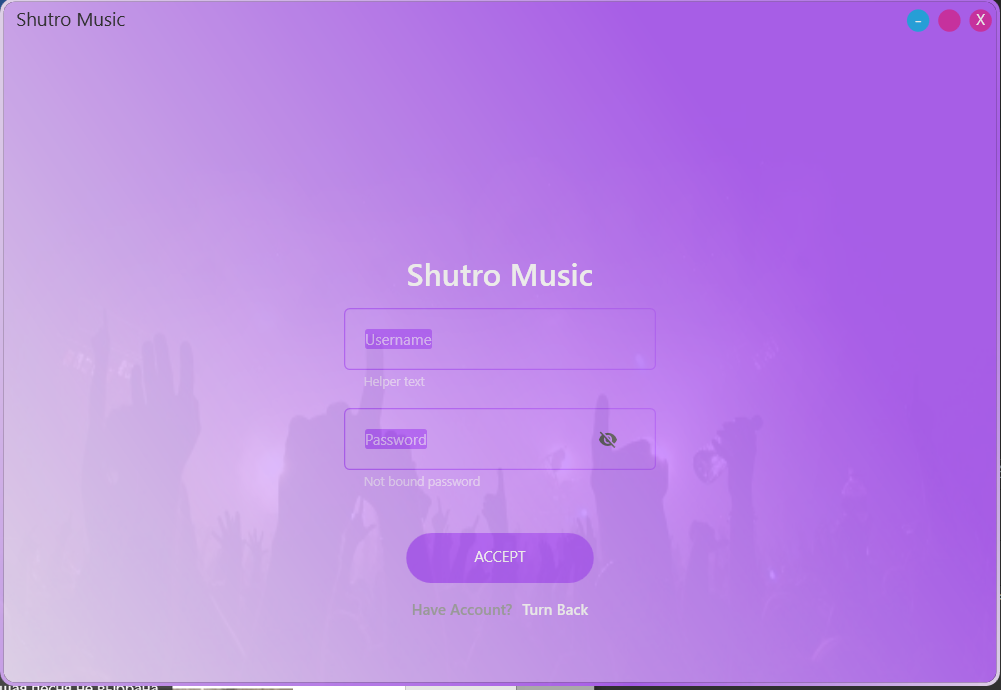


Рисунок 5.2 – Страница регистрации

Когда пользователь пытается создать новую учетную запись, введя имя, уже зарегистрированное в системе, приложение выполняет проверку уникальности. Цель этого шага — предотвратить дублирование записей и гарантировать, что каждый пользователь имеет уникальное имя для входа в систему.

Пользователь на странице регистрации вводит имя и пароль в соответствующие поля. После заполнения формы пользователь нажимает на кнопку "Зарегистрироваться", отправляя данные на обработку. Приложение проверяет базу данных, чтобы определить, существует ли уже запись с указанным именем. Если имя уникально, регистрация успешно завершается данные сохраняются в базе, и пользователь перенаправляется на страницу авторизации. Если имя уже занято, приложение уведомляет пользователя о том, что в системе уже есть пользователь с данным именем.

На рисунке 5.3 представлен сценарий, в котором пользователь пытается повторно зарегистрироваться в приложении.



Рисунок 5.3 – Повторная регистрация пользователя в систему

На странице авторизации пользователь заполняет форму с двумя обязательными полями: имя и пароль.

После заполнения формы пользователь нажимает кнопку "Войти", отправляя введенные данные на обработку.

Приложение выполняет запрос к базе данных для проверки наличия учетной записи с указанным именем. Если имя существует, система переходит к следующему этапу – проверке пароля. Если имя отсутствует, приложение уведомляет пользователя об ошибке.

Если введенное имя не найдено, пользователь видит сообщение о том, что учетная запись с таким именем отсутствует.

После получения уведомления об ошибке пользователь может:

* Перейти на страницу регистрации для создания новой учетной записи.
* Исправить ошибку в введенных данных и повторить попытку авторизации.

На рисунке 5.4 приведён сценарий подключения к системе незарегистрированного пользователя. В данном случае приложение сообщает об ошибке или уведомляет о том, что пользователя с данным имен нет в системе.

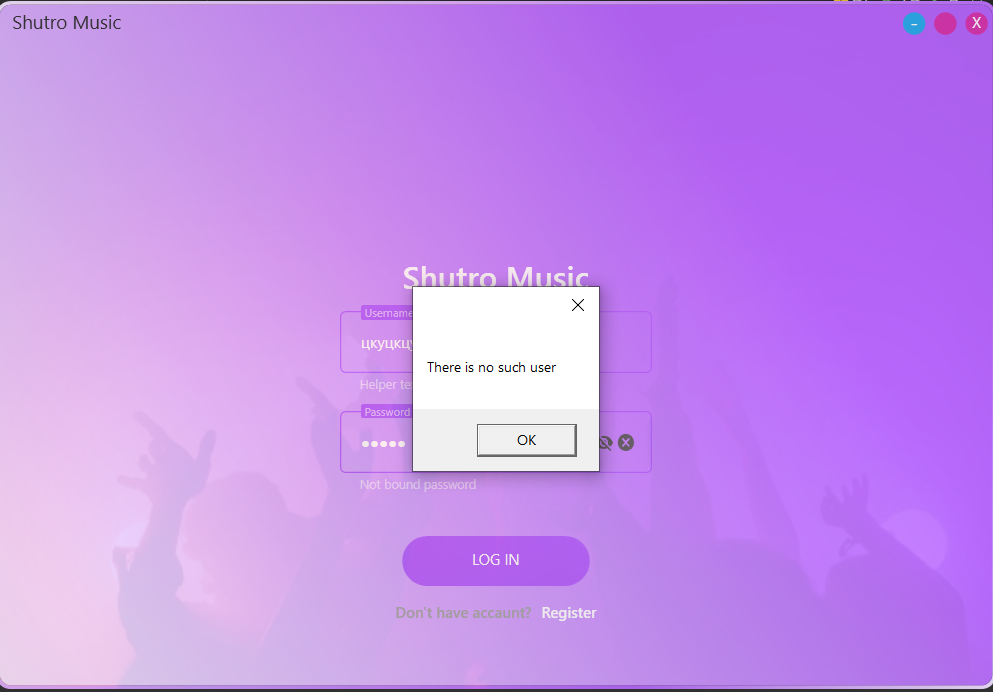


Рисунок 5.4 – Подключение к системе незарегистрированного пользователя

6. Руководство по установке и использованию

После успешной авторизации в программном средстве «Музыкальное приложение» пользователь получает доступ к главному окну приложения, которое представляет собой центральный узел для взаимодействия с основными функциями. Это окно является отправной точкой для работы с музыкальными треками, выполнения поиска и навигации по различным страницам.

Главное окно приложения разработано с учетом удобства пользователей, предлагая интуитивно понятный интерфейс и упрощенный доступ к основным возможностям. Основные задачи стартового окна включают:

* Поиск артистов.
* Переход на страницы, такие как «Songs» для управления песнями и «Mp3 Downloader» для загрузки треков по URL с платформы YouTube.

Упрощение начального взаимодействия позволяет пользователю сразу приступить к работе благодаря понятной организации интерфейса. Визуальные элементы, такие как кнопки и панели, позволяют быстро освоить функционал даже без предварительного обучения.

Центральная область отображает основные элементы интерфейса:

* **Результаты поиска:** После выполнения поиска пользователь видит список найденных артистов с указанием их подробной информации (исполнитель, продолжительность)
* **Рекомендуемые разделы:** В этой зоне могут отображаться наиболее популярные исполнители, основываясь на рекомендациях системы.

Визуальное представление стартового окна приведено на рисунке 6.1 и 6.2.

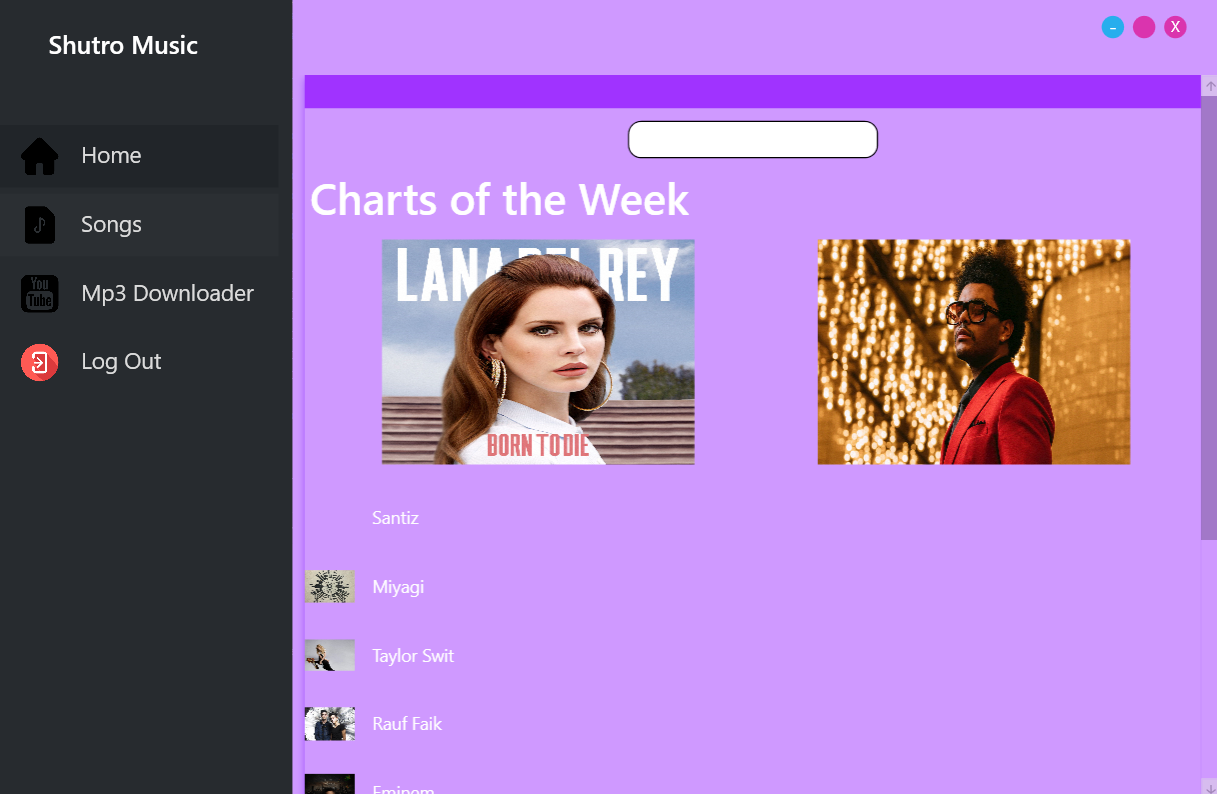
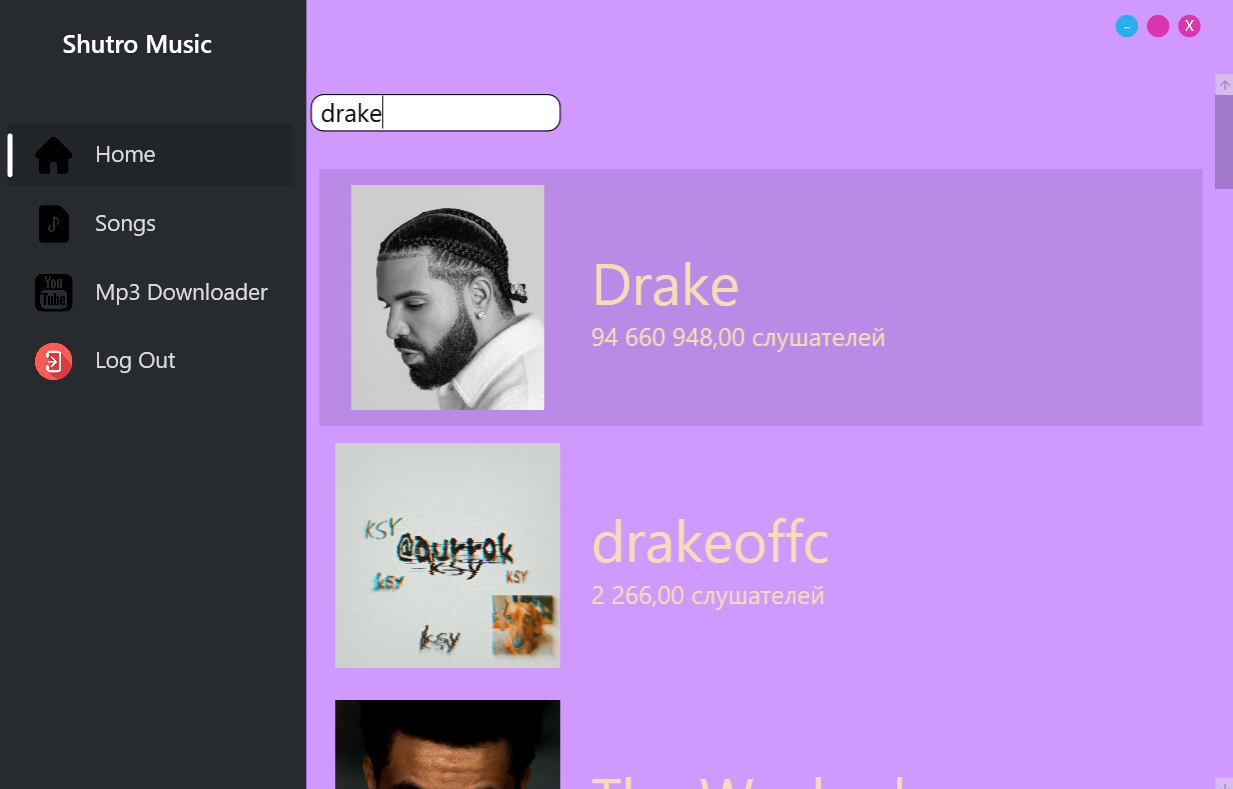


Рисунок 6.1 – Стартовое окно

Рисунок 6.2 – Результаты поиска

Страница «Songs» в музыкальном приложении разработана для управления аудиофайлами, их воспроизведения и загрузки новых треков с персонального компьютера. Этот раздел приложения предоставляет пользователю возможности, связанные с прослушиванием музыки, а также упрощает процесс добавления новых треков в библиотеку.

Основное назначение страницы «Songs» – это обеспечение комфортного взаимодействия с музыкальными файлами. Пользователи могут быстро добавлять треки с компьютера, управлять воспроизведением аудиофайлов и настроить удобные для себя параметры прослушивания.

Функционал страницы «Songs»:

1. Загрузка песен с компьютера:

На странице присутствует иконка папки, которая открывает окно выбора аудиофайлов.

Этапы загрузки песен:

* Пользователь нажимает на значок папки, расположенный в интерфейсе страницы.
* Открывается стандартное окно выбора файла, где пользователь может найти нужный аудиофайл в формате mp3 на своем устройстве.
* После выбора файла он автоматически добавляется в папку Folder, где хранятся все аудиофайлы приложения.
* Новый трек появляется в списке доступных песен на странице «Songs» и становится готовым для воспроизведения.
* Сразу после добавления файла приложение может автоматически начать воспроизведение трека, информируя пользователя об успешной загрузке.

2. Управление воспроизведением аудиофайлов:

Для удобного прослушивания музыки в приложении реализован аудиоплеер, предоставляющий следующие функции:

* Кнопка воспроизведения (Play): запускает воспроизведение выбранного трека.
* Кнопка паузы (Pause): приостанавливает воспроизведение без сброса текущего прогресса.
* Кнопка остановки (Stop): полностью завершает воспроизведение трека, сбрасывая его позицию на начало.
* Кнопка переключения треков:
* Кнопка «Next» позволяет перейти к следующему треку в списке.
* Кнопка «Previous» возвращает к предыдущему треку.
* Регулятор громкости: обеспечивает настройку уровня звука во время воспроизведения.

3. Сохранение файлов в библиотеку приложения:

После добавления аудиофайлов они автоматически сохраняются в папку Folder, где хранятся все музыкальные треки, используемые приложением. Это позволяет упорядочить музыкальную библиотеку и предоставляет удобный доступ к аудиофайлам.

Визуальное представление страницы «Songs» приведено на рисунке 6.3, 6.4 и 6.5

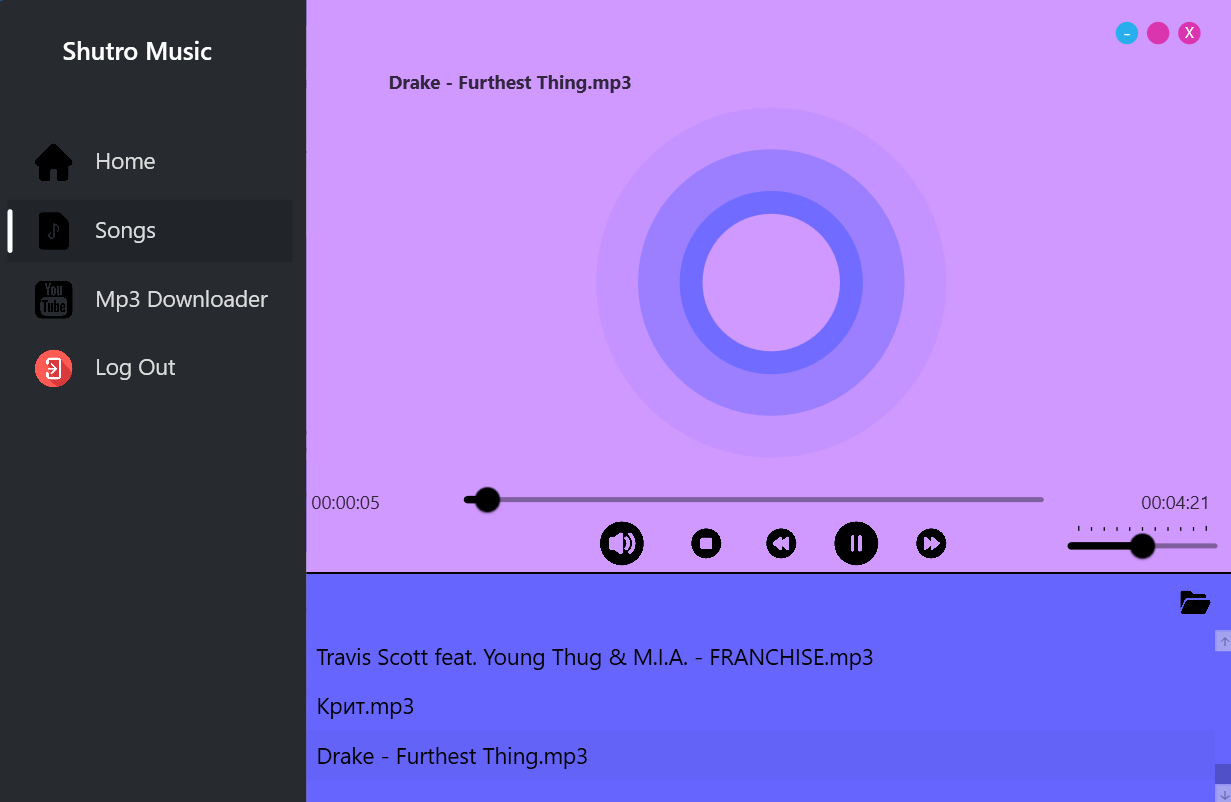


Рисунок 6.3 – Страница «Songs»

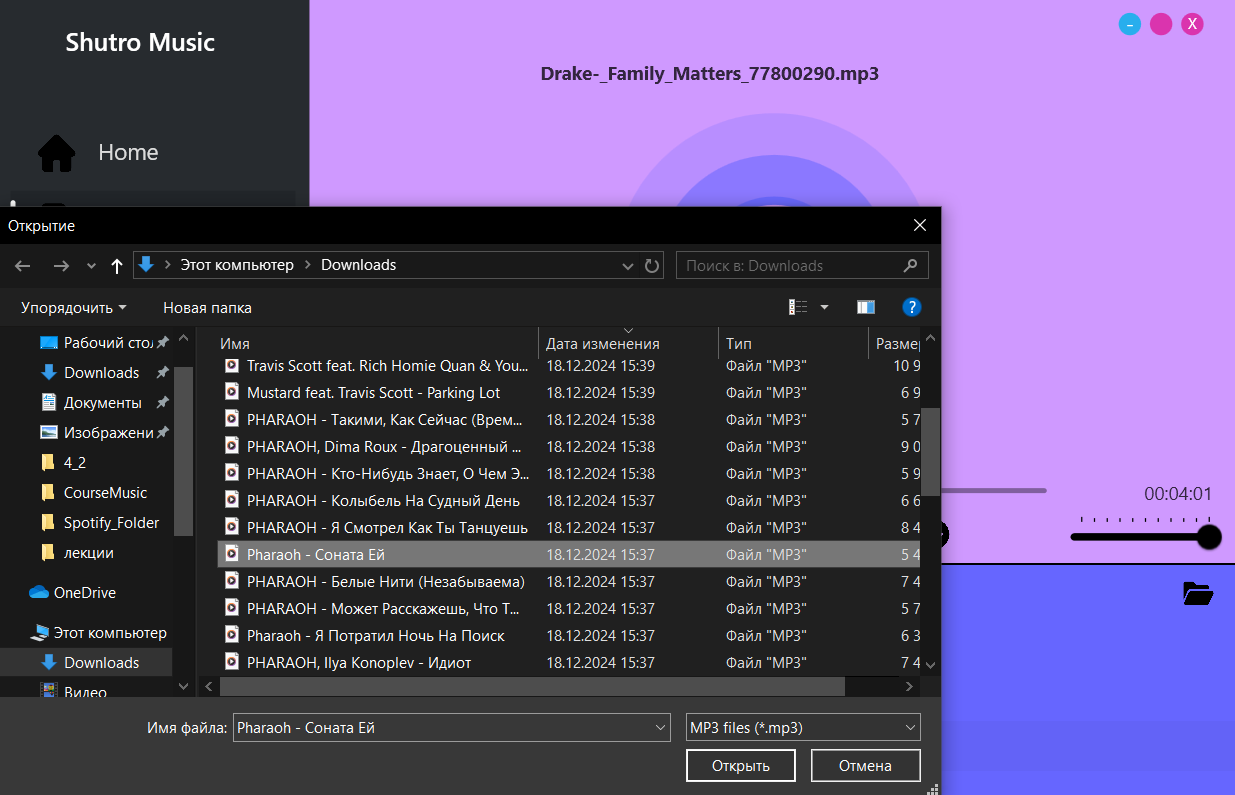
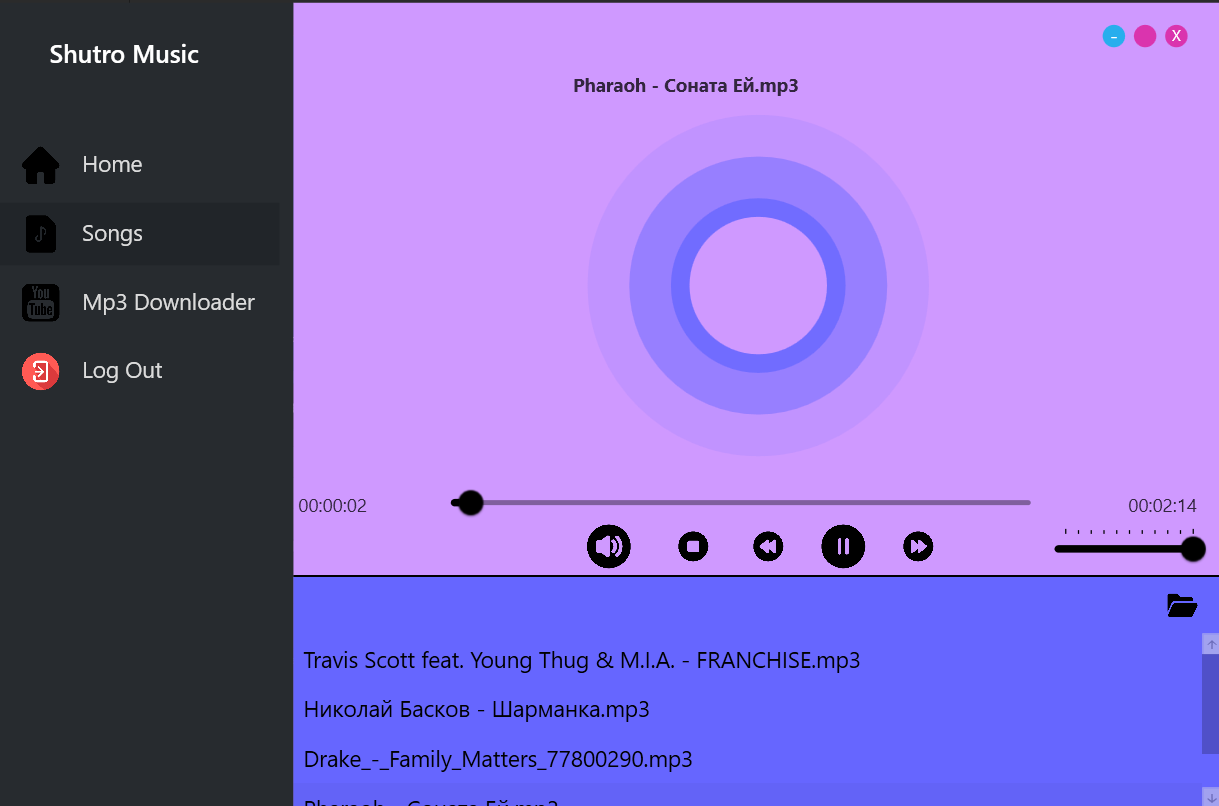


Рисунок 6.4 – Загрузка файла с персонального компьютера

Рисунок 6.5 – Воспроизведение загруженного трека

Страница «Mp3 Downloader» в музыкальном приложении разработана для выполнения двух основных функций: загрузки песен с платформы YouTube по URL и удаления уже существующих аудиофайлов из системы. Интерфейс страницы ориентирован на простоту и удобство использования, предоставляя все необходимые инструменты для работы с аудиофайлами.

Страница «Mp3 Downloader» предоставляет пользователям возможность расширять свою музыкальную коллекцию, загружая треки с платформы YouTube, а также управлять аудиофайлами, которые уже находятся в приложении. Это делает приложение более универсальным и удобным инструментом для создания персонализированной музыкальной библиотеки.

Основные задачи страницы:

* Загрузка треков с YouTube:
* Позволяет пользователю добавлять новые аудиофайлы в формате mp3, используя ссылки на видео с YouTube.
* Обеспечивает автоматическое преобразование видеофайлов в аудиофайлы.
* Удаление ненужных аудиофайлов:
* Реализует возможность удаления треков, которые больше не нужны пользователю.
* Упрощает управление библиотекой аудиофайлов.

Функционал страницы «Mp3 Downloader»

1. Загрузка аудиофайлов с YouTube:

На странице присутствует текстовое поле с меткой «Enter Youtube URL». Это поле предназначено для ввода URL видео с YouTube, которое пользователь хочет загрузить в формате mp3.

Этапы загрузки аудиофайлов:

* Пользователь копирует URL видео с YouTube.
* Вставляет ссылку в текстовое поле «Enter Youtube URL».
* Нажимает кнопку «Enter» или соответствующую кнопку загрузки на интерфейсе.
* Приложение начинает процесс конвертирования видеофайла в аудиофайл.
* Этот процесс включает обработку видео, извлечение звуковой дорожки и преобразование ее в формат mp3.
* После успешного завершения конвертации пользователь видит уведомление с подтверждением добавления трека. Сообщение может выглядеть следующим образом:

«Трек успешно добавлен в библиотеку приложения».

Новый трек появляется в списке доступных файлов и становится доступным для воспроизведения.

2. Удаление аудиофайлов:

Приложение позволяет пользователям эффективно управлять своей музыкальной библиотекой, удаляя ненужные треки.

Этапы удаления аудиофайлов:

* Пользователь открывает список треков в приложении.
* Выбирает аудиофайл, который нужно удалить, и нажимает на него правой кнопкой мыши.
* Появляется контекстное меню с опцией «delete».
* Нажав на «delete», пользователь подтверждает удаление трека.
* Аудиофайл немедленно удаляется из папки Folder, где хранятся все файлы приложения.
* После удаления файл становится недоступным для воспроизведения и исчезает из списка в приложении.
* Система может отобразить уведомление, подтверждающее успешное удаление файла.

Визуальное представление страницу «Songs» приведено на рисунке 6.3, 6.4 и 6.5



Рисунок 6.6 – Страница «Mp3 Downloader»

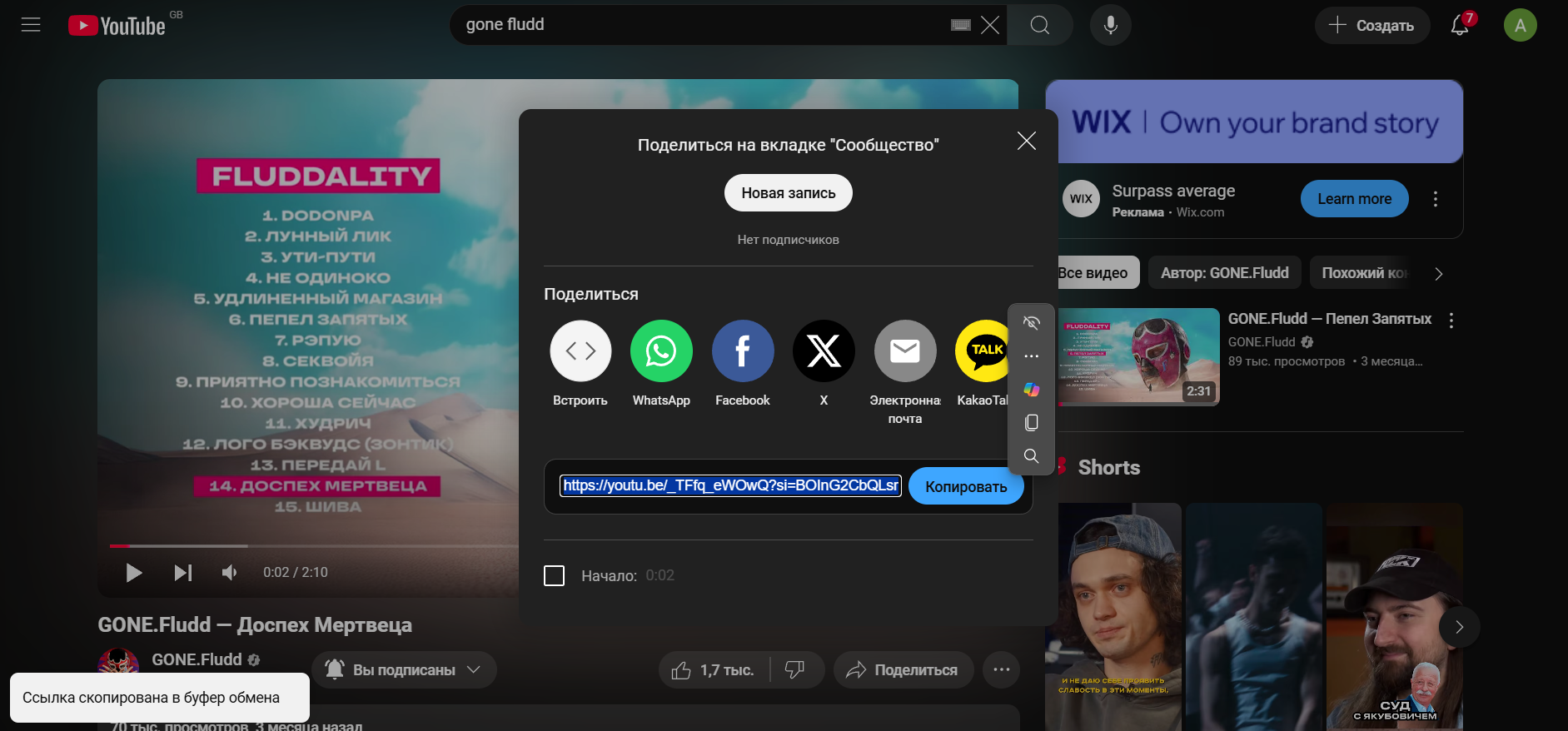
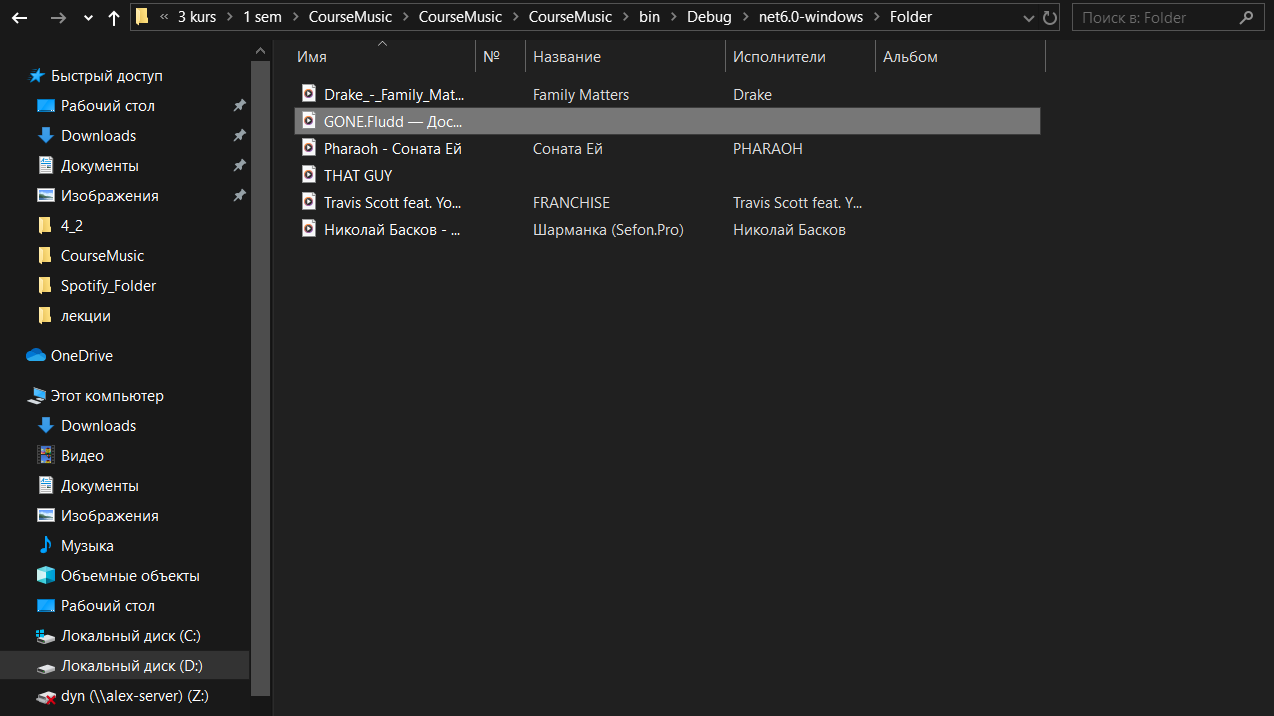


Рисунок 6.7 – Копирование URL с платформы Youtube

Рисунок 6.8 – Folder

Заключение

В ходе выполнения курсового проекта было разработано программное средство «Музыкальное приложение» на языке C# с использованием технологий WPF и SQLite.

При разработке программного средства были выполнены все пункты из указанного списка основного функционала приложения, а именно:

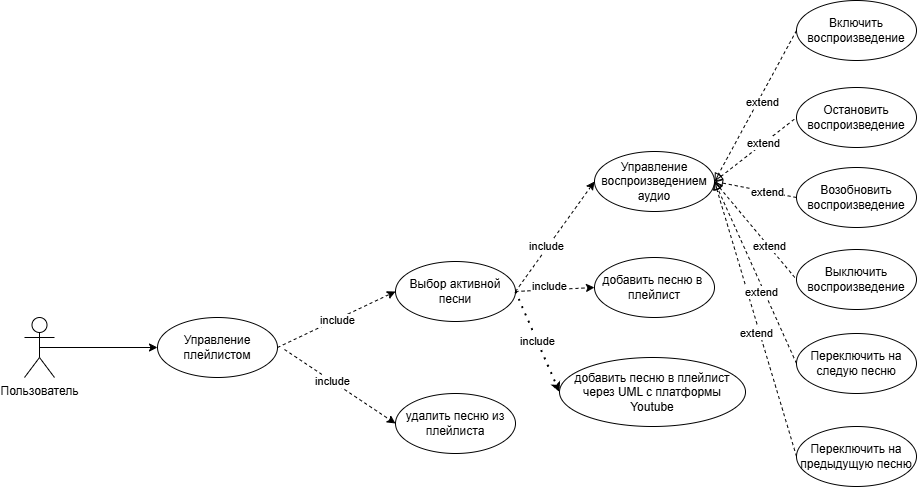
* добавление и удаление музыкальных треков в библиотеке;
* возможность поиска артистов;
* возможность загрузки песен по URL с платформы YouTube
* воспроизведение музыкальных треков с функциями паузы, перемотки и регулировки громкости;
* просмотр информации о треках (название, исполнитель, продолжительность);
* Тестирование программного средства показало, что оно работает корректно и стабильно выполняет все предусмотренные функции.

Также в процессе выполнения данного курсового проекта были закреплены навыки программирования на языке C#, создания приложений на WPF, работы с базой данных SQLite и проектирования пользовательского интерфейса.

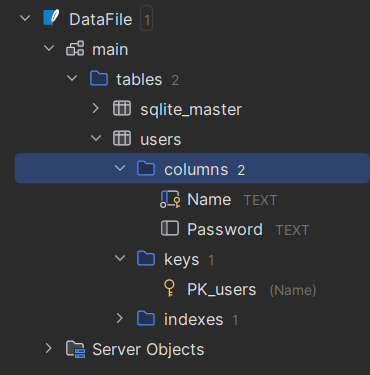
Список литературы

1. JetBrains Rider [Электронный ресурс] – https://www.jetbrains.com/ru-ru/rider – Дата доступа 28.11.2024
2. Полное руководство по языку программирования C# 7.0 и платформе .NET 6.0. Режим доступа: https://metanit.com/sharp/tutorial/ – Дата доступа: 30.11.2024
3. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
4. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 01.12.2024
5. Работа с Entity Framework Core [Электронный ресурс] – https://professorweb.ru/my/entity-framework/6/level1/ – Дата доступа 02.12.2024
6. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru – Дата доступа: 05.12.2024.

Приложение А



Приложение В

****

Листинг скрипта для подключения к базе данных

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  using CourseMusic.Models;  namespace CourseMusic.Db;  public class UserDataContext : DbContext  {  protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder)  {  optionsBuilder.UseSqlite($"Data Source = {App.path}\\DataFile.db");  }  public DbSet<User> users { get; set; }  } |