МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

|  |
| --- |
| ДОПУСТИТЬ К ЗАЩИТЕ  Заведующий кафедрой, д.ф.-м.н., профессор  \_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

**Разработка настольного приложения для загрузки плейлистов и аудиозаписей с видеохостинга YouTube**

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

ЮУрГУ – 02.03.02.2024.308-537.ВКР

|  |  |
| --- | --- |
|  | Научный руководитель,  доцент кафедры СП, к.ф.-м.н. \_\_\_\_\_\_\_\_\_\_ Г.И. Радченко  Автор работы, студент группы КЭ-402  \_\_\_\_\_\_\_\_\_\_ И.В. Башарова  Ученый секретарь  (нормоконтролер)  \_\_\_\_\_\_\_\_\_\_\_\_\_ И.Д. Володченко  «\_\_\_»\_\_\_\_\_\_\_\_\_\_\_ 2024 г. |

Челябинск, 2024 г.

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Южно-Уральский государственный университет**

**(национальный исследовательский университет)»**

**Высшая школа электроники и компьютерных наук**

**Кафедра системного программирования**

УТВЕРЖДАЮ

Зав. кафедрой СП

\_\_\_\_\_\_\_\_\_\_ Л.Б. Соколинский

29.01.2024 г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы бакалавра**

студентке группы КЭ-402

Башаровой Ильнаре Вадимовне,

обучающейся по направлению

02.03.02 «Фундаментальная информатика и информационные технологии»

1. **Тема работы** (утверждена приказом ректора от 22.04.2024 г. № 764-13/12)

Разработка настольного приложения для загрузки плейлистов и аудиозаписей с видеохостинга YouTube.

1. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
2. **Исходные данные к работе**
   1. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – Издательский дом «Питер», 2013. – 896 с.
   2. Windows Forms documentation. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms (дата обращения: 05.02.2024 г.).
   3. Command-line program to download videos from YouTube. [Электронный ресурс] URL: https://github.com/yt-dlp/yt-dlp (дата обращения: 05.02.2024 г.).
3. **Перечень подлежащих разработке вопросов**
   1. Изучить методы загрузки видео и аудио плейлистов с видеохостинга YouTube.
   2. Привести описание требований к разрабатываемому настольному приложению на основе диаграмм вариантов использования UML.
   3. Спроектировать структуру приложения и разработать необходимые модули.
   4. Протестировать возможности разработанного приложения.
   5. Продемонстрировать работоспособность приложения.
4. **Дата выдачи задания:** 29.01.2024 г.

**Научный руководитель,**

доцент кафедры СП, к.ф.-м.н. Г.И. Радченко

**Задание принял к исполнению** И.В. Башарова

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_Toc167545436)

[1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 7](#_Toc167545437)

[1.1. Описание предметной области 7](#_Toc167545438)

[1.2. Анализ существующих решений для реализации проекта 8](#_Toc167545439)

[2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ 10](#_Toc167545440)

[2.1. Диаграмма вариантов использования 10](#_Toc167545441)

[2.2. Основные актеры, взаимодействующие с системой 11](#_Toc167545442)

[3. АРХИТЕКТУРА СИСТЕМЫ 13](#_Toc167545443)

[3.1. Общее описание архитектуры системы 13](#_Toc167545444)

[3.2. Описание реализации архитектуры системы 14](#_Toc167545445)

[4. РЕАЛИЗАЦИЯ 16](#_Toc167545446)

[4.1. Обзор используемых технологий 16](#_Toc167545447)

[4.2. Реализация приложения 17](#_Toc167545448)

[4.3. Реализация интерфейса системы 22](#_Toc167545449)

[5. ТЕСТИРОВАНИЕ 23](#_Toc167545450)

[ЗАКЛЮЧЕНИЕ 25](#_Toc167545451)

[ЛИТЕРАТУРА 26](#_Toc167545452)

[ПРИЛОЖЕНИЯ 28](#_Toc167545453)

[Приложение А. Спецификация вариантов использования 28](#_Toc167545454)

[Приложение Б. Скриншоты разработанного приложения 33](#_Toc167545455)

# ВВЕДЕНИЕ

**Актуальность**

В современном информационном обществе наблюдается стремительный рост цифрового контента, включая аудио и видео. Видеохостинг YouTube, привлекающий миллионы пользователей ежедневно, не является исключением. Несмотря на широкие возможности онлайн-просмотра, существует потребность в скачивании аудиозаписей или плейлистов для офлайн-прослушивания, особенно в условиях ограниченного доступа к интернету или необходимости экономии траффика.

Для удобного использования и работы с ресурсами хостинга применяются приложения-загрузчики, однако некоторые из них, такие как youtube-dl и yt-dlp, не обладают графическим интерфейсом, что снижает их доступность для пользователей. Большинство существующих решений предлагают ограниченный функционал или требуют сложной настройки. В связи с этим разработка настольного приложения с графическим интерфейсом для загрузки плейлистов и аудиозаписей с YouTube становится актуальной научной задачей. Это позволит значительно повысить эффективность и удобство использования медиаконтента в повседневной жизни.

**Постановка задачи**

Целью выпускной квалификационной работы является разработка настольного приложения для загрузки плейлистов и аудиозаписей с видеохостинга YouTube. Для достижения поставленной цели необходимо решить следующие задачи:

* 1. изучить методы загрузки видео и аудио плейлистов с видеохостинга YouTube;
  2. привести описание требований к разрабатываемому настольному приложению на основе диаграмм вариантов использования UML;
  3. спроектировать структуру приложения и разработать необходимые модули;
  4. протестировать возможности разработанного приложения;
  5. продемонстрировать работоспособность приложения.

**Структура и содержание работы**

Работа состоит из введения, пяти глав, заключения и списка литературы. Объем работы составляет 34 страницы, объем списка литературы – 17 источников.

В первой главе описывается предметная область, анализ аналогичных проектов и существующие решения для реализации проекта.

Вторая глава посвящена описанию функциональных и нефункциональных требований к разрабатываемой программе, а также вариантам использования программы.

В третьей главе описывается архитектура системы и ее компоненты.

В четвертой главе представлена реализация приложения и интерфейс системы.

Пятая глава посвящена разработке функционального тестирования приложения.

В приложении А содержится спецификация основных вариантов использования.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

Основная задача заключается в разработке настольного приложения «Загрузчик плейлистов с видеохостинга YouTube». В рамках текущей работы предлагается следующее решение поставленной задачи: система состоит из графического приложения, основанного на формах, и библиотеки для получения ресурсов с видеохостинга YouTube.

**Анализ аналогичных проектов**

4K Youtube to MP3 [1] – настольное приложение с закрытым исходным кодом для скачивания аудиодорожек с видеохостинга YouTube в высоком качестве 320 кбит/с. Разработана компанией 4K Download [2] и распространяется по подписочной модели. Поддерживает операционные системы: Windows, Linux, MacOS.

YoutubeDownloader [3] – настольное приложение с открытым исходным кодом для скачивания видео и аудио ресурсов с видеохостинга YouTube. Разработан на языке программирования C# с использованием WPF [4]. Взаимодействия с видеохостингом происходит посредством библиотеки YoutubeExplode [5].

Youtube-dl-gui [6] – настольное приложение с открытым исходным кодом для скачивания видео с видеохостинга YouTube. Разработан на языке программирования Python с использованием wxWidgets [7]. Взаимодействия с видеохостингом происходит посредством библиотеки youtube-dl [8].

Сравнение перечисленных аналогов с разрабатываемым приложением по выделенным критериям приведено в таблице 1.

Таблица 1 – Сравнение аналогов и разрабатываемого приложения

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Критерий** | **4K Youtube to MP3** | **Youtube Downloader** | **youtube-dl-gui** | **Разрабатываемое приложение** |
| Поддерживаемые операционные системы | Windows, Linux, MacOS | Windows | Windows, Linux, MacOS | Windows |
| Язык написания | Неизвестно | C# | Python | C# |
| Библиотека для взаимодействия с видеохостингом YouTube | Неизвестно | YoutubeExplode | youtube-dl | YoutubeDLSharp [9] (yt-dlp [10]) |

## 1.2. Анализ существующих решений для реализации проекта

Существует множество подходов и решений для создания графических настольных приложений [11, 12]. Существующие решения можно разделить по нескольким критериям: язык разработки и целевая операционная система. Наиболее распространенными вариантами являются: WinForms [13] – языки разработки: C#, C++, Visual Basic, целевая операционная система – Windows; GTK [14] – языки разработки: C, C++, C#, Vala, целевые операционные системы: Windows, Linux, MacOS; Qt [15] – языки разработки: C++, Python, целевые операционные системы: Windows, Linux, MacOS, Android, IoT.

Для взаимодействия с интерфейсом видеохостинга YouTube сообществом были разработаны консольные программы и библиотеки такие, как youtube-dl, yt-dlp, YoutubeDLSharp. Для удобного взаимодействия с перечисленными программами из языка программирования C# была разработана библиотека YoutubeDLSharp, она поддерживает работу как с youtube-dl, так и с yt-dlp.

В результате анализа существующих решений для разработки приложений, были выбраны следующие технологии.

**Yt-dlp**

Yt-dlp представляет собой продвинутую альтернативу к утилите youtube-dl для загрузки видео с различных платформ видеохостинга. Эта технология разрабатывается сообществом и предлагает множество дополнительных функций и улучшений по сравнению с оригинальным youtube-dl. Yt-dlp активно поддерживается, имеет обширную базу пользователей и обновляется регулярно для обеспечения совместимости с изменениями на видеохостинговых платформах. Основное преимущество yt-dlp состоит в его гибкости, расширенных параметрах настройки и поддержке новых платформ, что делает его мощным инструментом для скачивания видеоконтента.

**C#**

C# (C-Sharp) – это объектно-ориентированный язык программирования, разработанный компанией Microsoft. Он широко используется для создания разнообразных приложений, включая веб-приложения, настольные программы, игры и мобильные приложения для платформы .NET. C# обладает сильной типизацией и современными возможностями языка, такими как асинхронное программирование и LINQ (Language Integrated Query). Он также предоставляет простой синтаксис, что способствует повышению производительности разработчиков и облегчает поддержку кода. C# является важным инструментом в разработке программного обеспечения под управлением платформы Microsoft и находит широкое применение в различных отраслях.

**Windows Forms**

Windows Forms (WinForms) представляет собой библиотеку пользовательского интерфейса для создания графических приложений под платформу Microsoft Windows. Она предоставляет разработчикам набор элементов управления и средства визуального проектирования, позволяя создавать настольные приложения с интерактивным интерфейсом. WinForms основан на языке программирования C# и поддерживает другие языки платформы .NET. При помощи WinForms разработчики могут создавать функциональные и интуитивно понятные пользовательские интерфейсы для широкого спектра приложений, включая утилиты, приложения для обработки данных.

**YoutubeDLSharp**

YoutubeDLSharp представляет собой обертку (wrapper) для популярной утилиты youtube-dl, написанную на языке программирования C#. Это позволяет разработчикам использовать функциональность youtube-dl в своих C# проектах, предоставляя удобный доступ к возможностям скачивания видео и аудио с различных платформ видеохостинга. YoutubeDLSharp обеспечивает интеграцию с приложениями на платформе .NET, позволяя разработчикам создавать клиенты для загрузки контента с YouTube и других поддерживаемых сервисов. Эта технология способствует упрощению процесса взаимодействия с youtube-dl в проектах на C#, делая его более доступным для программистов, работающих в этой экосистеме.

**Выводы по первой главе**

В результате обзора литературы и обзора схожих проектов была выявлена актуальность разработки программного решения. Также в результате обзора литературы был выявлен набор инструментальных средств для реализации поставленной задачи, наиболее полно удовлетворяющий требованиям к подобному рода системам. Приложение будет реализована на языке С# с использованием WinForms. Взаимодействие с видеохостингом YouTube будет обеспечено библиотекой YoutubeDLSharp посредством консольной программы yt-dlp.

# 2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ

**Функциональные требования к проектируемой системе**

Ниже представлен список основных функциональных требований для программы, предназначенной для скачивания аудио-файлов.

1. Программа должна позволять пользователю скачать аудио по заданному URL.
2. Программа должна позволять пользователю скачать все аудио из плейлиста по заданному URL.
3. Программа должна позволять пользователю выбирать путь для сохранения загружаемых файлов.
4. Программа должна позволять пользователю просмотр записей в плейлисте.
5. Программа должна позволять пользователю добавлять и удалять записи и плейлисты.
6. Программа должна позволять пользователю останавливать и возобновлять загрузку аудио.

**Нефункциональные требования к проектируемой системе**

Ниже представлен список основных нефункциональных требований для программы, предназначенной для скачивания аудио-файлов.

1. Программа должна быть написана на языке программирования C#.
2. Взаимодействие программы с видеохостингом YouTube должно быть обеспечено библиотекой YoutubeDLSharp.
3. Программа должна поддерживать работу с операционной системой Windows 10 и новее.

## 2.1. Диаграмма вариантов использования

Для составления диаграммы вариантов использования использован язык графического описания UML.

Диаграмма вариантов использования представлена на рисунке 1.

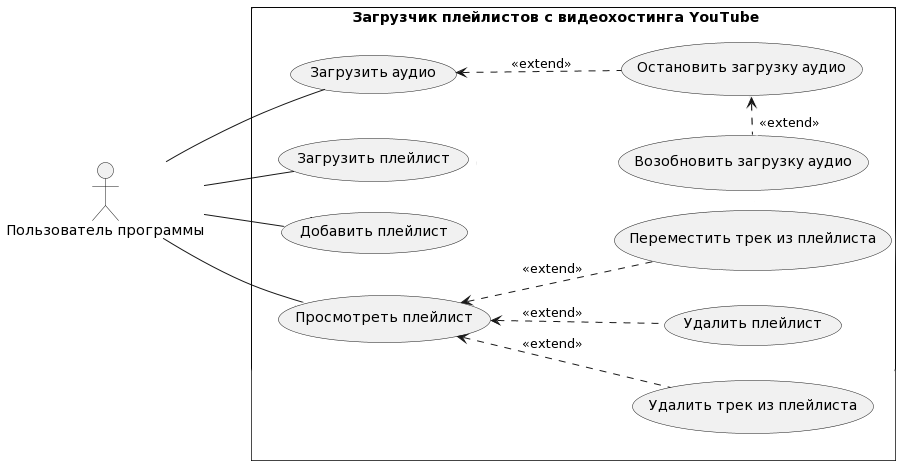


Рисунок 1 – Диаграмма вариантов использования

### 2.2. Основные актеры, взаимодействующие с системой

Основным актером, взаимодействующим с системой, выделен «Пользователь». Пользователь – человек, который использует настольное приложение. Ему доступен функционал загрузки аудио по заданному URL отдельных видео и плейлистов, а также добавление и удаление загруженных аудио.

**Краткое описание вариантов использования**

Краткое описание вариантов использования для актера «Пользователь» приведено ниже.

1. Загрузить аудио. Пользователь может загрузить файл на свой локальный компьютер по заданному URL.
2. Остановить загрузку аудио. Пользователь может остановить загрузку файла в просматриваемом плейлисте или во вкладке для всех текущих загрузок.
3. Возобновить загрузку аудио. Пользователь может возобновить остановленную загрузку файла.
4. Загрузить плейлист. Пользователь может загрузить файлы на свой локальный компьютер по заданному URL.
5. Просмотреть плейлист. Пользователь может просмотреть список треков в выбранном плейлисте.
6. Удалить плейлист. Пользователь может удалить выбранный плейлист из списка плейлистов.
7. Переместить трек из плейлиста. Пользователь может переместить трек из плейлиста в любой другой выбранный плейлист.
8. Добавить трек в плейлист. Пользователь может добавить трек в просматриваемый плейлист.
9. Удалить трек из плейлиста. Пользователь может удалить трек из просматриваемого плейлиста.
10. Добавить плейлист. Пользователь может добавить плейлист в список плейлистов.

**Выводы по второй главе**

В результате анализа требований к разрабатываемой системе был выделен ряд функциональных и нефункциональных требований. С использованием язык графического описания для объектного моделирования была разработана диаграмма вариантов использования. Были приведены варианты использования разрабатываемого приложения. Составлена спецификация вариантов использования, а также их краткое описание.

# 3. АРХИТЕКТУРА СИСТЕМЫ

## 3.1. Общее описание архитектуры системы

Архитектура разрабатываемой системы состоит из трех компонентов: «Окно приложения», «Загрузчик», «База данных». Классы из компонента «Окно приложения» отвечают за вывод информации и пользовательское взаимодействие с системой. Классы из компонента «Загрузчик» отвечают за загрузку ресурсов с видеохостинга. Классы из компонента «База данных» отвечают за хранение информации о ресурсах.

На рисунке 2 представлена диаграмма компонентов разрабатываемого приложения.

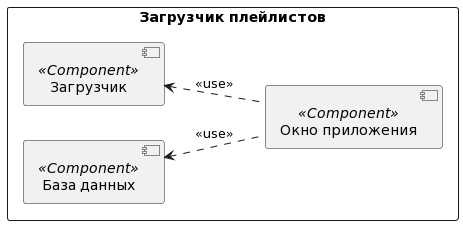
[](https://www.plantuml.com/plantuml/uml/hP1FIWCn5CRtESMZ1-W1f9I2nn0NnSOwC4racgv4iEAVvU3EMpZYH3CpJkOANsxaDwAuSLa2uNr_ydwGnTOj2xVRv2hNvqwSbI9RNpXPPOK-SvarIet3qgnpBHCy8U0L7U-ORk7HeXBKw58fQO-83ueAFG8DlAKRLEcEuX6USSlfNZ3G3kZOVeS7ksWcSgLEy8AQKJoLsKoExURI6csSpEUCdj6Fl9-L3mWZi6IVacH1n03_6rBrF14DCtRttsy_GUE_vN_SwLGEoZeYRRVLf5ohXJOh_lCN)

Рисунок 2 – Диаграмма компонентов

**Описание компонентов, составляющих систему**

**Компонент «Окно приложения»**

Данный компонент включает в себя следующие классы.

* 1. MainForm – главная форма приложения. В главной форме выводятся списки всех загружаемых и загруженных ресурсов, песен и плейлистов. Так же форма осуществляет взаимодействие пользователя с системой: позволяет загрузить ресурсы по заданному URL, изменять, удалять и воспроизводить загруженные ресурсы.
  2. CreatePlaylistForm – форма приложения, осуществляющая создание нового плейлиста с заданными пользователем параметрами.

**Компонент «Загрузчик»**

Данный компонент включает в себя класс YoutubeDownload. Данный класс осуществляет загрузку данных о заданном ресурсе по его URL. Загруженные данные включают в себя: название ресурса, тип ресурса, ссылку на ресурс, автора ресурса и другие поля. На основании загруженных данных происходит загрузка самого ресурса (аудио).

**Компонент «База данных»**

Данный компонент включает в себя следующие классы: Database – обеспечивает хранение данных, Song – представление ресурса «Песня», Playlist – представление ресурса «Плейлист». Компонент обеспечивает хранение и сохранение записей о ресурсах системы, состоянии системы.

## 3.2. Описание реализации архитектуры системы

Для реализации выбранной архитектуры и составляющих ее модулей, были разработаны следующие классы (рисунок 3).

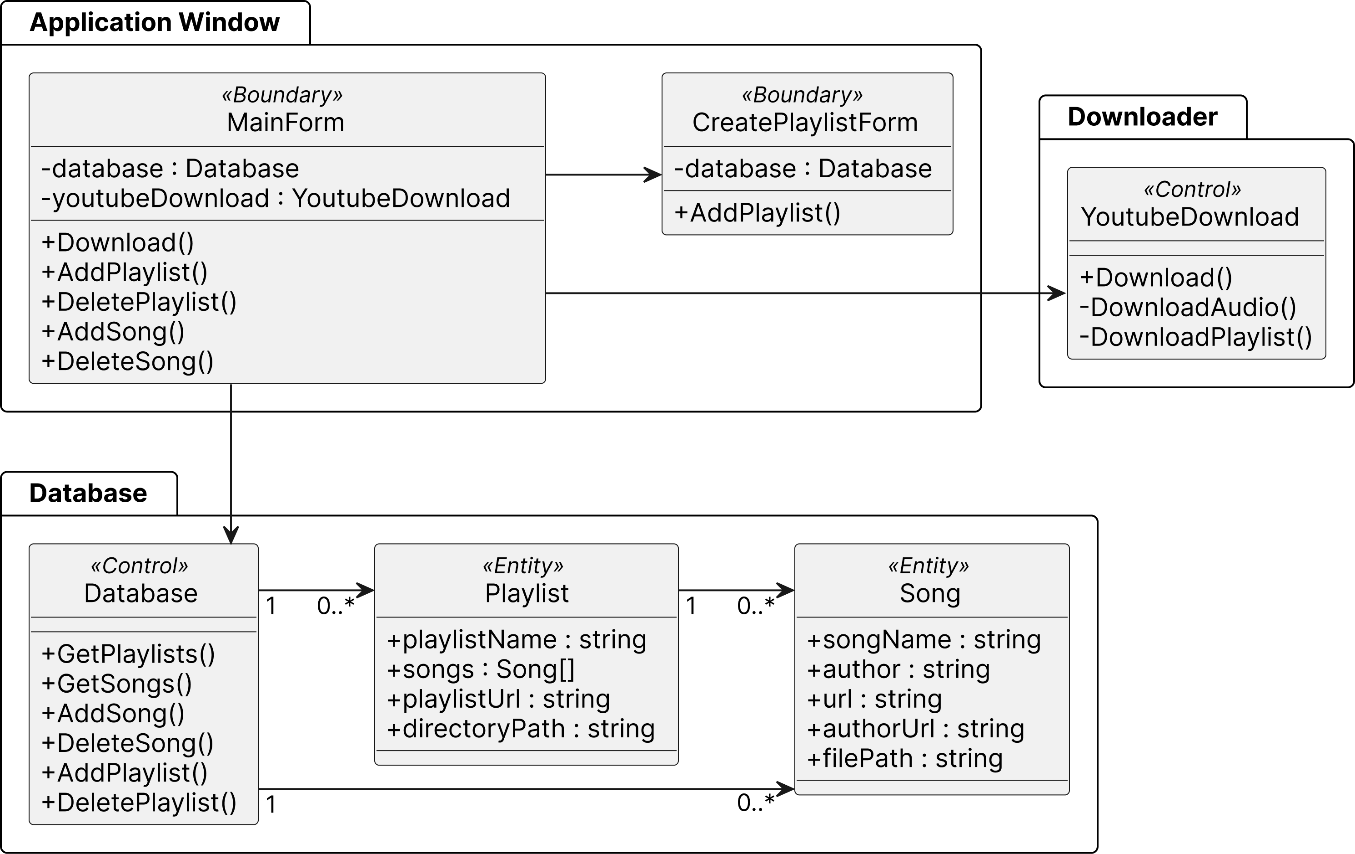


Рисунок 3 – Диаграмма классов

Компонент «Окно приложения» (Application Window) содержит следующие классы.

1. «Главная форма» (MainForm) – класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms». Класс формы содержит объект класса Database и объект класса YoutubeDownload, а также методы, использующие перечисленные объекты.
2. «Форма создания плейлиста» (CreatePlaylistForm) – класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms». Класс формы содержит объект класса Database и метод создания плейлиста.

Компонент «Загрузчик» (Downloader) содержит класс «Загрузчик из Youtube» (YoutubeDownload) – класс, использующий библиотеку YoutubeDLSharp для загрузки ресурсов и данных о них, предоставляет публичный метод загрузки ресурсов по заданному URL.

Компонент «База данных» (Database) содержит следующие классы.

* 1. «База данных» (Database) – класс, хранящий списки всех загружаемых и загруженных ресурсов системы. Осуществляет их сохранение и чтение из базы данных SQLite.
  2. «Песня» (Song) – представление ресурса «Песня». Содержит следующие поля: название песни, автор, ссылка на видео, ссылка на автора, путь до файла загрузки, прогресс загрузки.
  3. «Плейлист» (Playlist) – представление ресурса «Плейлист». Содержит следующие поля: название плейлиста, список песен, ссылка на плейлист, путь до папки, содержащей песни плейлиста.

**Выводы по третьей главе**

В результате проектирования была разработана архитектура приложения, удовлетворяющая приведенным требованиям. Архитектура состоит из нескольких связанных модулей, каждый из которых содержит приведенные классы.

# 4. РЕАЛИЗАЦИЯ

## 4.1. Реализация приложения

Разрабатываемая программа состоит из трех компонентов: окно приложения, загрузчик и база данных.

**Реализация компонента «Окно приложения»**

В ходе разработки компонента были реализованы следующие объекты.

1. MainForm – главная форма приложения. Класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms». Класс формы содержит объект класса Database и объект класса YoutubeDownload. Также в классе реализованы обработчики пользовательского ввода: нажатие на кнопку «Скачать», выбор элемента из списков песен и плейлистов.
2. CreatePlaylistForm – второстепенная форма приложения. Класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms». Класс формы содержит объект класса Database. Также в классе реализованы обработчики пользовательского ввода: нажатие на кнопку «Добавить».
3. BrowserForm – второстепенная форма приложения. Класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms». Класс формы содержит объект класса Database. Также в классе реализованы обработчики пользовательского ввода: нажатие на кнопку «Скачать», нажатие на кнопки «Вперед», «Назад».
4. SelectingPlaylistForm – второстепенная форма приложения. Класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms». Класс формы содержит объект класса Database. Также в классе реализованы обработчики пользовательского ввода: поиск плейлистов по заданному тексту, выбор плейлиста по нажатию на элемент.

Реализация метода обработки нажатия на кнопку «Скачать» класса MainForm приведена в листинге 1.

Листинг 1 – Обработка нажатия на кнопку «Скачать»

private void downloadAudioButton\_Click(object sender, EventArgs e) {

var link = linkInputTextBox.Text;

linkInputTextBox.Text = "";

Download(link);

}

private void Download(string link) {

youtubeDownload.Download(link);

}

**Реализация компонента «Загрузчик»**

Данный компонент включает в себя класс YoutubeDownload. Данный класс осуществляет загрузку данных о заданном ресурсе по его URL. В ходе реализации компонента был разработан класс YoutubeDownload. Класс, использует библиотеку YoutubeDLSharp для загрузки ресурсов и данных о них, предоставляет публичный метод загрузки ресурсов по заданному URL.

Для скачивания ресурса по заданному URL разработан публичный метод Download. Данный метод создаёт задачу на загрузку ресурса. Сначала осуществляется загрузка информации о ресурсе и его типе. В зависимости от типа ресурса вызывается определенный метод для его загрузки: DownloadAudio – для видео, DownloadPlaylist – для плейлистов. Реализация данного метода и структуры приведены в листинге 2.

Листинг 2 – Метод загрузки ресурсов

public async void Download(string link)

{

var result = await youtubeDownloader.RunVideoDataFetch(link);

if (!result.Success) {

return;

}

var data = result.Data;

if (data.Entries == null) {

await DownloadAudio(data);

} else {

await DownloadPlaylist(data);

}

}

Реализация метода загрузки аудио приведена в листинге 3.

Листинг 3 – Метод загрузки аудио

private async Task DownloadAudio(VideoData videoData, Playlist playlist = null)

{

var song = new Song {

songName = videoData.Title,

author = videoData.Channel,

url = videoData.Url ?? videoData.WebpageUrl,

authorUrl = videoData.ChannelUrl,

progress = 0

};

OnCreateSong?.Invoke(song, playlist);

await StartSongDownload(song, playlist);

}

public async Task StartSongDownload(Song song, Playlist playlist) {

var cts = new CancellationTokenSource();

songsCancelTokens[song.id] = cts;

try {

var result = await youtubeDownloader.RunAudioDownload(

song.url,

AudioConversionFormat.Mp3,

ct: cts.Token,

progress: new Progress<DownloadProgress>(p => {

OnUpdateSongProgress?.Invoke(song, (int)(p.Progress \* 100));

}),

overrideOptions: new OptionSet() {

Continue = true,

Output = Path.Combine(

playlist?.directoryPath ?? DownloadsPath,

youtubeDownloader.OutputFileTemplate

),

EmbedThumbnail = true,

ConvertThumbnails = "jpg",

WriteThumbnail = true,

}

);

if (!result.Success)

{

OnDeleteSong?.Invoke(song);

return;

}

OnUpdateSongProgress?.Invoke(song, 100);

OnUpdateSongPath?.Invoke(song, result.Data);

} catch (OperationCanceledException) {

OnStopSong?.Invoke(song);

} catch (Exception) {

OnDeleteSong?.Invoke(song);

}

}

При осуществлении задач загрузки ресурсов возникают события изменения состояния ресурсов. Внешние классы могут подписаться на возникающие события. Код описания событий приведен в листинге 4.

Листинг 4 – Описание событий

public delegate void CreateSongHandler(Song song, Playlist playlist);

public event CreateSongHandler OnCreateSong;

public delegate void SongHandler(Song song);

public event SongHandler OnUpdateSongProgress;

public event SongHandler OnUpdateSongPath;

public event SongHandler OnDeleteSong;

public delegate void PlaylistHandler(Playlist playlist);

public event PlaylistHandler OnCreatePlaylist;

public event PlaylistHandler OnDeletePlaylist;

Класс MainForm при возникновении событий вызывает обновление состояния ресурсов в базе данных и в отображаемых формах. Код обработки событий компонента «Загрузчик» приведен в листинге 5.

Листинг 5 – Обработка событий компонента «Загрузчик»

youtubeDownload = new YouTubeDownload(downloadsPath, applicationPath);

youtubeDownload.OnCreateSong += (Song song, Playlist playlist) => database.AddSong(song, playlist);

youtubeDownload.OnUpdateSongProgress += (Song song, int progress) => database.UpdateSongProgress(song, progress);

youtubeDownload.OnUpdateSongPath += (Song song, string path) => database.UpdateSongPath(song, path);

youtubeDownload.OnStopSong += (Song song) => database.UpdateSongStopped(song, true);

youtubeDownload.OnDeleteSong += (Song song) => database.DeleteSong(song);

youtubeDownload.OnCreatePlaylist += (Playlist playlist) => database.AddPlaylist(playlist);

youtubeDownload.OnDeletePlaylist += (Playlist playlist) => database.DeletePlaylist(playlist);

**Реализация компонента «База данных»**

Данный компонент обеспечивает хранение и сохранение записей о ресурсах системы, состоянии системы. В ходе разработки были реализованы следующие объекты.

**Класс «База данных»**

Класс, хранящий подключение к базе данных SQLite, методы для создания запросов к базе данных, методы для создания и получения хранимых ресурсов. Реализация метода для формирования и выполнения «INSERT» запроса к базе данных SQLite приведена в листинге 6.

Листинг 6 – Реализация метода добавления ресурса в базу данных

private int InsertInto(string tableName, string[] columns, object[] values) {

var command = connection.CreateCommand();

string[] parameters = new string[columns.Length];

for(int i = 0; i < parameters.Length; i++) {

parameters[i] = $"${i}";

}

command.CommandText = $"INSERT INTO {tableName} ({string.Join(",", columns)}) VALUES ({string.Join(",", parameters)});";

for(int i = 0; i < parameters.Length; i++) {

command.Parameters.AddWithValue($"${i}", values[i]);

}

command.ExecuteNonQuery();

command = connection.CreateCommand();

command.CommandText = $"SELECT last\_insert\_rowid();";

var reader = command.ExecuteReader();

reader.Read();

return reader.GetInt32(0);

}

**Класс «Песня»**

Класс представление ресурса «Песня». Содержит следующие поля: название песни, автор, ссылка на видео, ссылка на автора, путь до файла загрузки, прогресс загрузки. Реализация класса «Песня» представлена в листинге 7.

Листинг 7 – Реализация класса «Песня»

public class Song {

public int id;

public string songName;

public string author;

public string url;

public string authorUrl;

public string filePath;

public int progress;

public bool stopped;

}

**Класс «Плейлист»**

Класс представление ресурса «Плейлист». Содержит следующие поля: название плейлиста, список песен, ссылка на плейлист, путь до папки, содержащей песни плейлиста. Реализация класса «Плейлист» представлена в листинге 8.

Листинг 8 – Реализация класса «Плейлист»

public class Playlist {

public int id;

public string playlistName;

public List<Song> songs;

public string playlistUrl;

public string directoryPath;

}

## 4.2. Реализация интерфейса системы

Интерфейс в разрабатываемом приложении создается с помощью библиотеки WinForms и средств разработки, предоставляемых интегрированной средой разработки Visual Studio. Разрабатываемое приложение состоит из набора форм, перечисленных в их компоненте «Окно приложения»: MainForm, CreatePlaylistForm, BrowserForm, SelectingPlaylistForm. Дизайн и расположение элементов в формах хранятся в XML формате в файлах с расширением «.resx». Обработчики пользовательских событий, возникающих в формах, определяются в классах форм. Классы форм наследуются от класса Form из пакета «System.Windows.Forms». Скриншоты разработанных окон приложения приведены в приложении Б.

**Выводы по четвертой главе**

В результате реализации системы была разработана программная реализация модулей приложения. Была произведена разработка компонентов модулей «Окно приложения», «Загрузчик» и «База данных», а также реализация интерфейса системы.

# 5. ТЕСТИРОВАНИЕ

Тестирование проводилось на пробных ссылках на плейлисты и аудио записи на видеохостинге YouTube.

Функциональное тестирование – это тестирование программного обеспечения в целях проверки реализуемости функциональных требований, то есть способности программного обеспечения в определенных условиях решать задачи, нужные пользователям. Функциональные требования определяют, что именно делает программное обеспечение, какие задачи оно решает.

Набор тестов на функциональность представлен в таблице 2.

Таблица 2 – Функциональное тестирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **№** | **Название теста** | **Шаги** | **Ожидаемый результат** | **Тест пройден?** |
| 1 | Скачивание аудио | 1. Вставить ссылку на трек в поле ввода ссылки  2. Нажать на кнопку «Скачать» | Аудио было скачано и хранится в папке общих загрузок | Да |
| 2 | Остановка скачивания аудио | 1. Нажать правой кнопкой мыши на аудио  2. В контекстном меню выбрать пункт «Остановить» | Загрузка аудио была остановлена | Да |
| 3 | Возобновление скачивания аудио | 1. Нажать правой кнопкой мыши на аудио  2. В контекстном меню выбрать пункт «Возобновить» | Загрузка аудио была возобновлена | Да |
| 4 | Скачивание плейлиста | 1. Вставить ссылку на плейлист в поле ввода ссылки  2. Нажать на кнопку «Скачать» | Была создана папка с именем плейлиста, в которой хранятся загруженные треки плейлиста | Да |
| 5 | Скачивание аудио в плейлист | 1. Выбрать желаемый плейлист  2. Вставить ссылку на трек в поле ввода ссылки  3. Нажать на кнопку «Скачать» | Аудио было скачано и хранится в папке плейлиста | Да |

Окончание таблицы 2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 6 | Добавление плейлиста | 1. Нажать на кнопку «Добавить»  2. Ввести в поле ввода название плейлиста  3. Нажать на кнопку «Добавить» | Пустая папка с именем плейлиста была создана, пустой плейлист был добавлен в список плейлистов | Да |
| 7 | Удаление песни | 1. Нажать правой кнопкой мыши на трек  2. В контекстном меню выбрать пункт «Удалить» | Выбранная песня была удалена | Да |
| 8 | Удаление плейлиста | 1. Нажать правой кнопкой мыши на плейлист  2. В контекстном меню выбрать пункт «Удалить» | Выбранный плейлист и его содержимое были удалены | Да |
| 9 | Проигрывание трека | Нажать на трек дважды | Выбранный трек начал воспроизводиться | Да |
| 10 | Скачивание трека с помощью браузера | 1. В главном окне нажать на кнопку «Выбрать в браузере»  2. В открывшемся окне браузера найти нужное видео  3. Нажать на кнопку «Скачать» | Трек был скачен и хранится в папке общих загрузок | Да |
| 11 | Скачивание плейлиста с помощью браузера | 1. В главном окне нажать на кнопку «Выбрать в браузере»  2. В открывшемся окне браузера найти нужный плейлист  3. Нажать на кнопку «Скачать» | Была создана папка с именем плейлиста, в которой хранятся загруженные треки плейлиста | Да |

**Выводы по пятой главе**

В результате тестирования системы были разработаны процессы для функционального тестирования приложения. Разработанные тесты покрывают большинство функционала системы и облегчают исправление кода и разработку новых опций.

# ЗАКЛЮЧЕНИЕ

В рамках данной работы было разработано настольное приложение для загрузки плейлистов и аудиозаписей с видеохостинга YouTube. При этом были решены следующие задачи.

* 1. Произведен анализ предметной области, обзор существующих работ по данной тематике и обзор аналогичных проектов. Изучены методы загрузки видео и аудио плейлистов с видеохостинга YouTube.
  2. Приведено описание требований к разрабатываемому настольному приложению на основе диаграмм вариантов использования UML.
  3. Спроектирована структура приложения.
  4. Разработаны необходимые модули.
  5. Протестированы возможности разработанного приложения.
  6. Продемонстрирована работоспособность приложения.

Разработанное настольное приложения используется для загрузки плейлистов и аудиозаписей с видеохостинга YouTube. Приложение разработано на языке C# с использованием библиотеки WinForms, программы для загрузки ресурсов с видеохостинга YouTube – yt-dlp, а также библиотеки YouTubeDLSharp. Корректность и работоспособность приложения подтверждается проведенными тестами.

В дальнейшем планируется расширение функционала приложение, добавление большего числа тестов, а также написание руководства для пользователей и документации для разработчиков.

# ЛИТЕРАТУРА

1. Рихтер Д. CLR via C#. Программирование на платформе   
   Microsoft .NET Framework 4.5 на языке C#. – Издательский дом «Питер», 2013. – 896 с.
2. Windows Forms documentation. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms (дата обращения: 05.02.2024 г.).
3. Command-line program to download videos from YouTube. [Электронный ресурс] URL: https://github.com/yt-dlp/yt-dlp (дата обращения: 05.02.2024 г.).
4. Документация C#. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата обращения: 05.02.2024 г.).
5. Qt Framework. [Электронный ресурс] URL: <https://www.qt.io/product/framework> . (дата обращения: 05.02.2024 г.).
6. Документация GTK. [Электронный ресурс] URL: https://docs.gtk.org/ (дата обращения: 05.02.2024 г.).
7. Документация youtube-dl. [Электронный ресурс] URL: https://youtube-dl.readthedocs.io/en/latest/ (дата обращения: 05.02.2024 г.).
8. Документация yt-dlp. [Электронный ресурс] URL: https://yt-dlp.site/docs (дата обращения: 05.02.2024 г.).
9. Документация YoutubeDLSharp. [Электронный ресурс] URL: https://github.com/Bluegrams/YoutubeDLSharp?tab=readme-ov-file#youtubedlsharp (дата обращения: 05.02.2024 г.).
10. Головачев А.В. Разработка приложений для Windows на C# и .NET. – СПб.: Питер, 2018. – 512 с.
11. Windows – что это такое? | internet-lab.ru. [Электронный ресурс] URL: https://internet-lab.ru/windows\_os (дата обращения: 05.02.2024 г.).
12. Разработка на C# в Visual Studio – Visual Studio | Microsoft Docs. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/visualstudio/get-started/csharp/?view=vs-2019 (дата обращения: 05.02.2024 г.).
13. Лопатин Д.К. Разработка простых интерфейсов с использованием библиотеки WinForms. Журнал научных публикаций аспирантов и докторантов, 2014. – № 3 (93). – С. 265-267.
14. Общие сведения о платформе .NET Framework | Microsoft Docs. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/framework/get-started/overview (дата обращения: 05.02.2024 г.).
15. Версии .NET Framework и ОС Windows | Microsoft Docs. [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/dotnet/framework/migration-guide/versions-and-dependencies#net-framework-47 (дата обращения: 05.02.2024 г.).
16. wxWidgets Documentation [Электронный ресурс] URL: https://docs.wxwidgets.org/latest/ (дата обращения: 05.02.2024 г.).
17. Начало работы с WebView2 в приложениях WinForms [Электронный ресурс] URL: https://learn.microsoft.com/ru-ru/microsoft-edge/webview2/get-started/winforms (дата обращения: 05.02.2024 г.).

# ПРИЛОЖЕНИЯ

## Приложение А. Спецификация вариантов использования

Спецификация основных вариантов использования (ВИ) приведена в таблицах 1–10.

Таблица 1 – Спецификация ВИ «Загрузить аудио»

|  |
| --- |
| Прецедент: Загрузить аудио |
| ID: 1 |
| Краткое описание:  Пользователь может загрузить файл на свой локальный компьютер по заданному URL |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на главной странице |
| Основной поток:  1. Пользователь вводит URL в поле ввода.  2. Пользователь нажимает на кнопку «Скачать». |
| Постусловия:  Аудио было загружено в папку загрузок. |
| Альтернативные потоки:  Нет |

Таблица 2 – Спецификация ВИ «Остановить загрузку аудио»

|  |
| --- |
| Прецедент: Остановить загрузку аудио |
| ID: 1.1 |
| Краткое описание:  Пользователь может остановить загрузку аудио |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на странице просмотра плейлиста |
| Основной поток:  1. Пользователь выбирает аудио.  2. Пользователь нажимает на аудио правой кнопкой мыши.  3. Пользователь нажимает на кнопку «Остановить» в выпадающем меню. |
| Постусловия:  Загрузка аудио была остановлена. |
| Альтернативные потоки:  Нет |

Продолжение приложения А

Таблица 3 – Спецификация ВИ «Возобновить загрузку аудио»

|  |
| --- |
| Прецедент: Возобновить загрузку аудио |
| ID: 1.2 |
| Краткое описание:  Пользователь может возобновить остановленную загрузку аудио |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на странице просмотра плейлиста |
| Основной поток:  1. Пользователь выбирает аудио.  2. Пользователь нажимает на аудио правой кнопкой мыши.  3. Пользователь нажимает на кнопку «Возобновить» в выпадающем меню. |
| Постусловия:  Загрузка аудио была возобновлена. |
| Альтернативные потоки:  Нет |

Таблица 4 – Спецификация ВИ «Загрузить плейлист»

|  |
| --- |
| Прецедент: Загрузить плейлист |
| ID: 2 |
| Краткое описание:  Пользователь может загрузить файлы на свой локальный компьютер по заданному URL |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на главной странице |
| Основной поток:  1. Пользователь вводит URL в поле ввода.  2. Пользователь нажимает на кнопку «Скачать». |
| Постусловия:  Плейлист был загружен в папку загрузок. |
| Альтернативные потоки:  Нет |

Продолжение приложения А

Таблица 5 – Спецификация ВИ «Просмотреть плейлист»

|  |
| --- |
| Прецедент: Просмотреть плейлист |
| ID: 3 |
| Краткое описание:  Пользователь может просмотреть список треков в выбранном плейлисте |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на главной странице |
| Основной поток:  1. Пользователь выбирает плейлист из списка плейлистов.  2. Пользователь нажимает на выбранный плейлист. |
| Постусловия:  Пользователь переходит на страницу просмотра плейлиста. |
| Альтернативные потоки.  1. Удалить плейлист (ID: 3.1).  2. Добавить трек в плейлист (ID: 3.2).  3. Удалить трек из плейлиста (ID: 3.3). |

Таблица 6 – Спецификация ВИ «Переместить трек из плейлиста»

|  |
| --- |
| Прецедент: Переместить трек из плейлиста |
| ID: 3.1 |
| Краткое описание:  Пользователь может переместить трек из просматриваемого плейлиста |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на странице просмотра плейлиста |
| Основной поток:  1. Пользователь нажимает на выбранный трек правой кнопкой мыши.  2. Пользователь нажимает на кнопку «Добавить в».  3. Пользователь выбирает плейлист из списка в открывшемся окне. |
| Постусловия:  Трек перемещен. |
| Альтернативные потоки:  Нет |

Продолжение приложения А

Таблица 7 – Спецификация ВИ «Удалить плейлист»

|  |
| --- |
| Прецедент: Удалить плейлист |
| ID: 3.2 |
| Краткое описание:  Пользователь удалить выбранный плейлист из списка плейлистов |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на странице просмотра плейлиста |
| Основной поток:  1. Пользователь нажимает на кнопку «Удалить». |
| Постусловия:  Просматриваемый плейлист удален из списка плейлистов. |
| Альтернативные потоки:  Нет |

Таблица 8 – Спецификация ВИ «Добавить трек в плейлист»

|  |
| --- |
| Прецедент: Добавить трек в плейлист |
| ID: 3.3 |
| Краткое описание:  Пользователь может добавить трек в просматриваемый плейлист |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на странице просмотра плейлиста |
| Основной поток:  1. Пользователь вводит URL трека в поле ввода.  2. Пользователь нажимает на кнопку «Скачать». |
| Постусловия:  Трек, загруженный по введенному URL, добавлен в просматриваемый плейлист. |
| Альтернативные потоки:  Нет |

Окончание приложения А

Таблица 9 – Спецификация ВИ «Удалить трек из плейлиста»

|  |
| --- |
| Прецедент: Удалить трек из плейлиста |
| ID: 3.4 |
| Краткое описание:  Пользователь может удалить трек из просматриваемого плейлиста |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на странице просмотра плейлиста |
| Основной поток:  1. Пользователь выбирает трек в просматриваемом плейлисте.  2. Пользователь нажимает на кнопку «Удалить» в строке выбранного трека.  3. В сплывающем окне подтверждения пользователь выбирает один из вариантов:  1) если пользователь выбрал кнопку «Нет», то он возвращается на страницу просмотра плейлиста без изменений;  2) если пользователь выбрал кнопку «Да», то трек удаляется из плейлиста. |
| Постусловия:  При подтверждении пользователя, трек был удален. |
| Альтернативные потоки:  Нет |

Таблица 10 – Спецификация ВИ «Добавить плейлист»

|  |
| --- |
| Прецедент: Добавить плейлист |
| ID: 4 |
| Краткое описание:  Пользователь может добавить плейлист в список плейлистов |
| Главные актеры:  Пользователь |
| Второстепенные актеры:  Нет |
| Предусловия: пользователь находится на главной странице |
| Основной поток:  1. Пользователь нажимает на кнопку «Добавить плейлист».  2. В сплывающем окне пользователь вводит название плейлиста в поле ввода.  3. Пользователь нажимает на кнопку «Добавить». |
| Постусловия:  Плейлист был добавлен в список плейлистов. |
| Альтернативные потоки:  Нет |

## Приложение Б. Скриншоты разработанного приложения

Скриншоты разработанного приложения приведены на рисунках 1– 4.

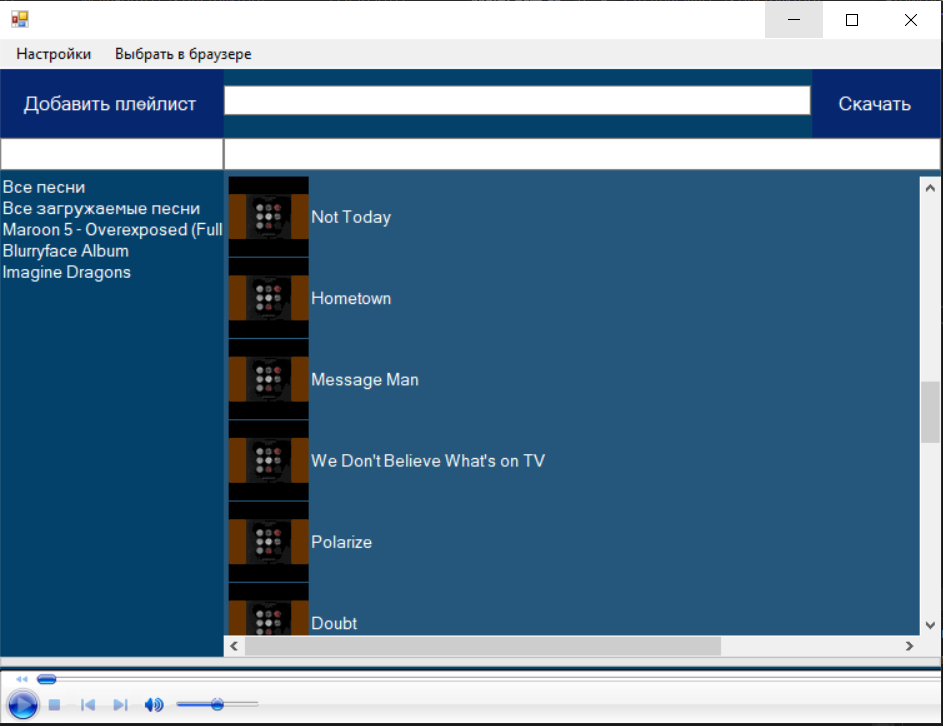


Рисунок 1 – Форма «Окно приложения»

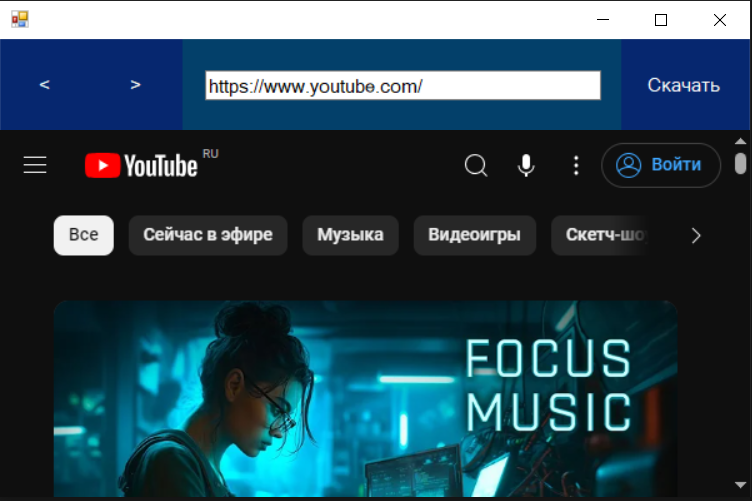


Рисунок 2 – Форма «Браузер»

Продолжение приложения Б

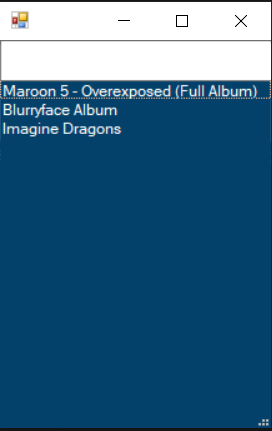


Рисунок 3 – Форма «Выбор плейлиста»



Рисунок 4 – Форма «Создание плейлиста»