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.
2. **Срок сдачи студентом законченной работы:** 03.06.2024 г.
3. **Исходные данные к работе**

3.1. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – Издательский дом «Питер», 2013. – 896 с.

3.2. Windows Forms documentation. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms (дата обращения: 05.02.2024 г.).

3.3. 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. **Перечень подлежащих разработке вопросов**

4.1. Изучить методы загрузки видео и аудио плейлистов с видеохостинга YouTube.

4.2. Привести описание требований к разрабатываемому настольному приложению на основе диаграмм вариантов использования UML.

4.3. Спроектировать структуру приложения и разработать необходимые модули.

4.4. Протестировать возможности разработанного приложения.

5. 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

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;

Структура и содержание работы

Работа состоит из введения, пяти глав, заключения и списка литературы. Объем работы составляет 36 страниц, объем списка литературы – 18 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 тестированию функционала приложения.

В приложении А содержится спецификация основных вариантов использования.

В приложении Б содержатся скриншоты разработанного приложения.

1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1. Описание предметной области

Основная задача заключается в разработке настольного приложения для загрузки аудио и 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 и распространяется по подписочной модели. Поддерживает операционные системы: Windows, Linux, MacOS.

YoutubeDownloader [2]– настольное приложение с открытым исходным кодом для скачивания видео и аудио ресурсов с видеохостинга YouTube. Разработан на языке программирования C# [3, 4, 5] с использованием WPF [6]. Взаимодействия с видеохостингом происходит посредством библиотеки YoutubeExplode [7].

Youtube-dl-gui [8] – настольное приложение с открытым исходным кодом для скачивания видео с видеохостинга YouTube. Разработан на языке программирования Python с использованием wxWidgets [9]. Программа предоставляет визуальный интерфейс для использования возможностей youtube-dl [10], позволяя пользователям находить, загружать и конвертировать видео без необходимости использования командной строки. Он поддерживает множество опций и настроек, которые доступны в оригинальной программе youtube-dl.

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 [11] (yt-dlp [12, 13])

1.2. Анализ существующих решений для реализации проекта

Существует множество подходов и решений для создания графических настольных приложений. Существующие решения можно разделить по нескольким критериям: язык разработки и целевая операционная система. Наиболее распространенными вариантами являются: WinForms [14, 15, 16] – языки разработки: C#, C++, Visual Basic, целевая операционная система – Windows; GTK [17] – языки разработки: C, C++, C#, Vala, целевые операционные системы: Windows, Linux, MacOS; Qt [18] – 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 приложения, были выбраны технологии, рассмотренные далее.

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 подобного рода системам. Приложение будет реализовано на языке C# с использованием WinForms. Взаимодействие с видеохостингом YouTube будет обеспечено библиотекой YoutubeDLSharp посредством консольной программы yt-dlp.

2. АНАЛИЗ ТРЕБОВАНИЙ К ПРОГРАММНОЙ СИСТЕМЕ

2.1. Требования к разрабатываемому приложению

Функциональные требования к проектируемой системе

Ниже представлен список основных функциональных требований для программы, предназначенной для скачивания аудиофайлов.

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 аудиофайлов.

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

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. Удалить трек из плейлиста. Пользователь может удалить трек из просматриваемого плейлиста.

9. 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 языка 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.

Рисунок 2 – Диаграмма компонентов

Далее рассмотрены компоненты, составляющие 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. Также форма осуществляет взаимодействие пользователя с системой: позволяет загрузить ресурсы, изменять, удалять и воспроизводить загруженные ресурсы, останавливать и возобновлять загрузку ресурсов.

2. CreatePlaylistForm – форма приложения, осуществляющая создание нового плейлиста с заданными пользователем параметрами.

3. BrowserForm – форма приложения, отображающая браузер и позволяющая искать и загружать из него ресурсы.

4. SelectingPlaylistForm – форма приложения, предоставляющая поиск плейлиста для перемещения песни.

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.

Схема базы данных представлена на рисунке 3.

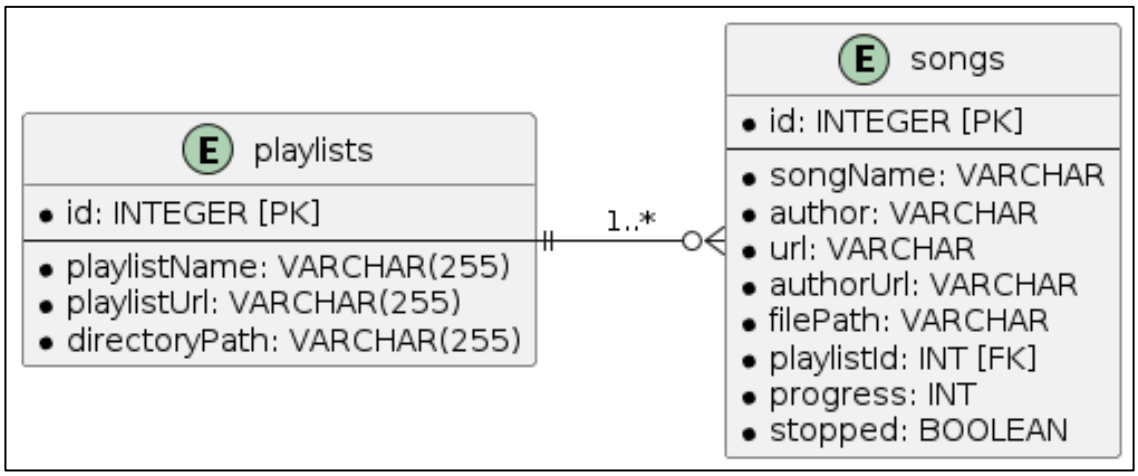

Рисунок 3 – Схема базы данных

3.2. Описание реализации архитектуры системы

Для реализации выбранной архитектуры и составляющих ее модулей, были разработаны следующие классы (рисунок 4).

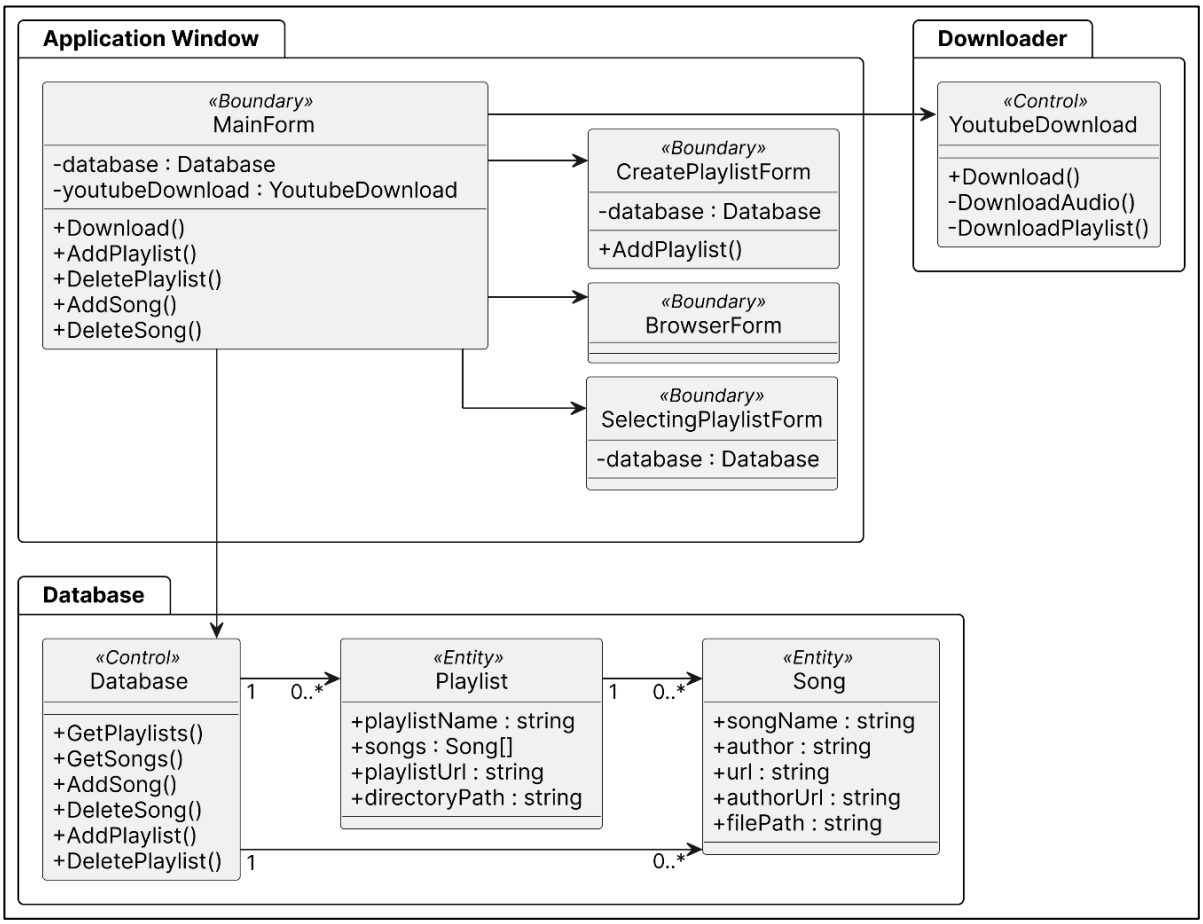

Рисунок 4 – Диаграмма классов

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.

3. «Форма браузера» (BrowserForm) – класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms».

4. «Форма поиска плейлиста» (SelectingPlaylistForm) – класс-форма, наследующийся от класса Form из пакета «System.Windows.Forms». Класс формы содержит объект класса Database.

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

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, поиск плейлистов по заданному тексту.

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 отображения списков песен и плейлистов класса `MainForm` приведена в листинге 2.

Листинг 2 – Отображение списков песен и плейлистов

```
var songItems = new List<ListViewItem>();
for (int i = 0; i < currentSongs.Count; i++) {
    var song = currentSongs[i];
    songItems.Add(MakeSongListViewItem(song, i));
}

var playlistItems = new List<ListViewItem> {
    new ListViewItem("Все песни"),
    new ListViewItem("Все загружаемые песни")
};
foreach (Playlist playlist in currentPlaylists) {
    playlistItems.Add(new ListViewItem(playlist.playlistName, 0));
}

trackListView.Items.Clear();
trackListView.SmallImageList = songImages;
trackListView.Items.AddRange(songItems.ToArray());

playlistListView.Items.Clear();
playlistListView.Items.AddRange(playlistItems.ToArray());
```

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 создает 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 3.

Листинг 3 – 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 приложении Б.

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 конструктора класса `Database` приведен в листинге 6.

Листинг 6 – Реализация метода добавления ресурса в базу данных

```
public Database(string path) {
    connection = new SQLiteConnection(path);
    connection.Open();

    CreateTable("playlists", new string[]{"id INTEGER PRIMARY KEY",
                                  "playlistName VARCHAR(255) NOT NULL",
                                  "playlistUrl VARCHAR(255) NOT NULL",
                                  "directoryPath VARCHAR(255) NOT NULL"});

    CreateTable("songs", new string[]{"id INTEGER PRIMARY KEY",
                                     "songName VARCHAR",
                                     "author VARCHAR",
                                     "url VARCHAR",
                                     "authorUrl VARCHAR",
                                     "filePath VARCHAR",
                                     "playlistId INT",
                                     "progress INT",
                                     "stopped BOOLEAN",
                                     "FOREIGN KEY (playlistId) REFERENCES
Playlist(id)"});

    MarkAllInProgressAsStopped();
}
```

Реализация метода для формирования и выполнения «INSERT» запроса к базе данных SQLite приведена в листинге 7.

Листинг 7 – 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 8.

Листинг 8 – 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 9.

Листинг 9 – Реализация класса «Плейлист»

```
public class Playlist {
  public int id;
  public string playlistName;
  public List<Song> songs;
  public string playlistUrl;
  public string directoryPath;
}
```

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».

Разработанная форма «Создание плейлиста» приведена на рисунке 5.

Рисунок 5 – Форма «Создание плейлиста»

Разработанная форма «Выбор плейлиста» приведена на рисунке 6.

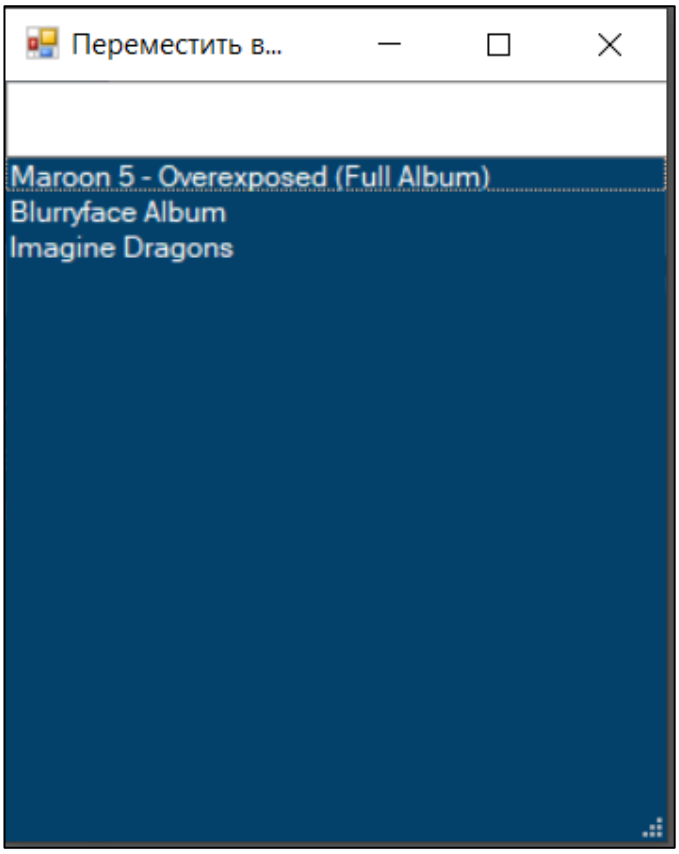

Рисунок 6 – Форма «Выбор плейлиста»

Разработанная форма «Браузер» приведена на рисунке 7.

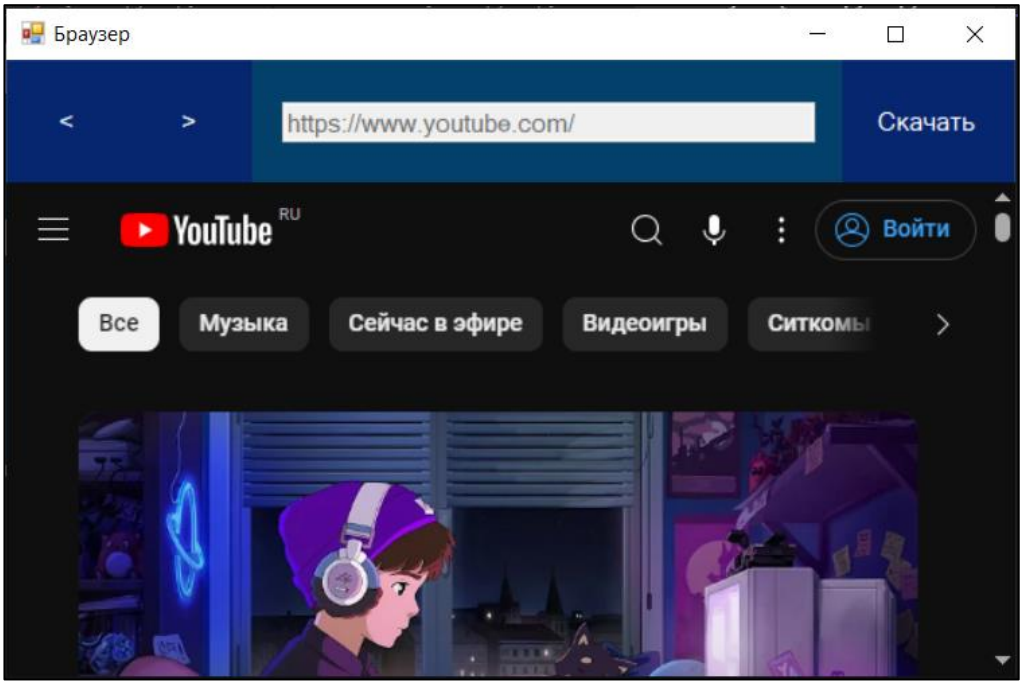

Рисунок 7 – Форма «Браузер»

Разработанная форма «Окно приложения» приведена на рисунке 8.

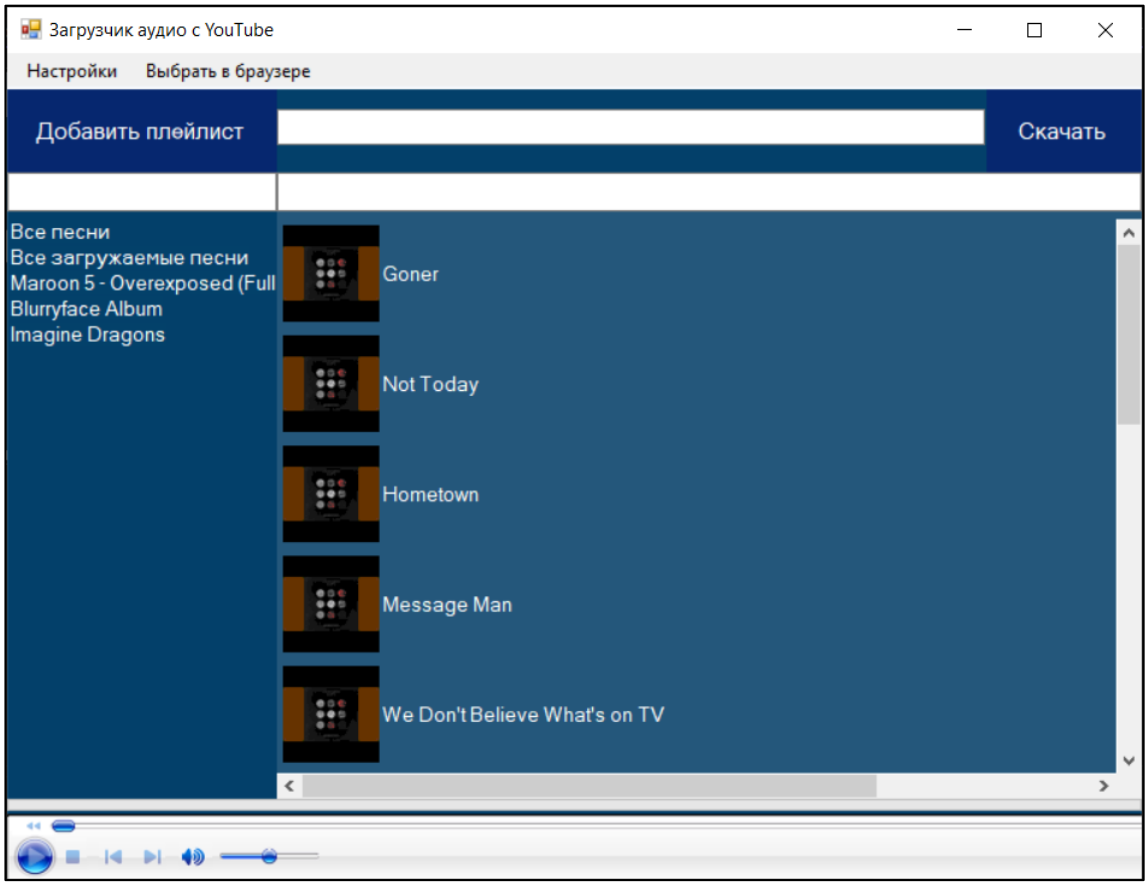

Рисунок 8 – Форма «Окно приложения»

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 аудиозаписи 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

№	Название теста	Шаги	Ожидаемый результат	Тест пройден?
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 песню 2. В выпадающем меню выбрать пункт «Удалить»	Выбранная песня была удалена	Да
8	Удаление плейлиста	1. Нажать правой кнопкой мыши на плейлист 2. В выпадающем меню выбрать пункт «Удалить»	Выбранный плейлист и его содержимое были удалены	Да
9	Проигрывание песни	Нажать на песню дважды	Выбранная песня начала воспроизводиться	Да
10	Скачивание аудио 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»	Аудио было скачано и хранится в папке общих загрузок	Да
11	Переместить трек из плейлиста	1. Нажать на трек правой кнопкой мыши. 2. Нажать на кнопку «Добавить в» в выпадающем меню. 3. Выбрать плейлист из списка в открывшемся окне.	Трек был перемещен в выбранный плейлист	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.

Разработанное настольное приложение 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 приложения, добавление большего числа тестов, а также написание руководства для пользователей и документации для разработчиков.

ЛИТЕРАТУРА

1. 4K YouTube to MP3. [Электронный ресурс] URL: https://www.4kdownload.com/-rpkdd/youtube-to-mp3 (дата обращения: 05.02.2024 г.).
2. YoutubeDownloader. [Электронный ресурс] URL: https://github.com/Tyrrrz/YoutubeDownloader (дата обращения: 05.02.2024 г.).
3. Рихтер Д. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C#. – Издательский дом «Питер», 2013. – 896 с.
4. Головачев А.В. Разработка приложений для Windows на C# и .NET. – СПб.: Питер, 2018. – 512 с.
5. 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.).
6. WPF is a .NET Core UI framework for building Windows desktop applications. [Электронный ресурс] URL: https://github.com/dotnet/wpf (дата обращения: 05.02.2024 г.).
7. Документация YoutubeExplode. [Электронный ресурс] URL: https://github.com/Tyrrrz/YoutubeExplode (дата обращения: 05.02.2024 г.).
8. Документация youtube-dl-gui. [Электронный ресурс] URL: https://github.com/MrS0m30n3/youtube-dl-gui (дата обращения: 05.02.2024 г.).
9. wxWidgets Documentation. [Электронный ресурс] URL: https://docs.wxwidgets.org/latest/ (дата обращения: 05.02.2024 г.).
10. Документация youtube-dl. [Электронный ресурс] URL: https://youtube-dl.readthedocs.io/en/latest/ (дата обращения: 05.02.2024 г.).

11. Документация YoutubeDLSharp. [Электронный ресурс] URL: https://github.com/Bluegrams/YoutubeDLSharp?tab=readme-ov-file#youtubedlsharp (дата обращения: 05.02.2024 г.).

12. Command-line program to download videos from YouTube. [Электронный ресурс] URL: https://github.com/yt-dlp/yt-dlp (дата обращения: 05.02.2024 г.).

13. Документация yt-dlp. [Электронный ресурс] URL: https://yt-dlp.site/docs (дата обращения: 05.02.2024 г.).

14. Windows Forms documentation. [Электронный ресурс] URL: https://learn.microsoft.com/en-us/dotnet/desktop/winforms (дата обращения: 05.02.2024 г.).

15. 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.).

16. Лопатин Д.К. Разработка простых интерфейсов с использованием библиотеки WinForms. Журнал научных публикаций аспирантов и докторантов, 2014. – № 3 (93). – С. 265-267.

17. Документация GTK. [Электронный ресурс] URL: https://docs.gtk.org/ (дата обращения: 05.02.2024 г.).

18. Qt Framework. [Электронный ресурс] URL: https://www.qt.io/product/framework . (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

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

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 дважды нажимает на выбранный плейлист левой кнопкой мыши.
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» в выпадающем меню. 3. Пользователь выбирает плейлист из списка в открывшемся окне.
Постусловия: Трек перемещен.
Альтернативные потоки: Нет

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 выбранный плейлист правой кнопкой мыши. 1. Пользователь нажимает на кнопку «Удалить» в выпадающем меню.
Постусловия: Плейлист удален из списка плейлистов.
Альтернативные потоки: Нет

Таблица 8 – 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 выпадающем меню.
Постусловия: При подтверждении пользователя, трек был удален.
Альтернативные потоки: Нет

Таблица 9 – 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. Реализация загрузки аудио

Реализация загрузки аудио приведена в листинге 1.

Листинг 1 – Загрузка 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);
  }
}
```