

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет
вычислительной техники

Кафедра
МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

БАКАЛАВРСКАЯ РАБОТА

на тему

**Программные средства воспроизведения и
редактирования системы аудиофайлов**

Студент

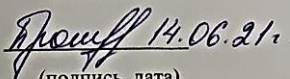


(подпись, дата)

Макаров Кирилл Андреевич

(ФИО полностью)

Руководитель

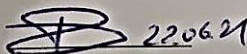


(подпись, дата)

Прошкина Е.Н.

(фамилия, инициалы)

Нормоконтролёр



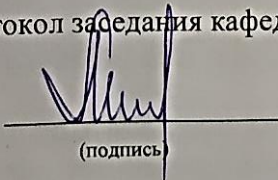
(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

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

Заведующий кафедрой



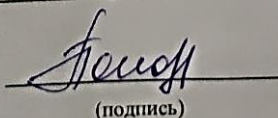
(подпись)

Макарычев П.П.

(фамилия, инициалы)

Работа защищена с отметкой 4 (протокол заседания ГЭК от 14.06.21 № 5)

Секретарь ГЭК



(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2021

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

Факультет
Вычислительной техники

Кафедра
МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

П.П. Макарычев

«28» декабря 2020 г

ЗАДАНИЕ

НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ БАКАЛАВРА

1. Студент Макаров Кирилл Андреевич гр. 17ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2. Тема работы
Программные средства воспроизведения и редактирования системы аудиофайлов

Тема утверждена приказом ПГУ № _____ от "28" декабря 2020 г

3. Руководитель работы к.т.н., доцент кафедры «МОиПЭВМ» Прошкина Е.Н.

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

1) Разработать программные средства воспроизведения и редактирования системы аудиофайлов

2) Средства выполнения: ОС Windows 10

Языковые средства разработки: C#, UML

Инструментальные средства разработки: Microsoft Visual Studio 2017

3) Основной функционал разрабатываемой системы:

а) обработка аудиопотока

б) воспроизведение аудиофайлов

в) работа с музыкальными форматами MP3, WAV, WMA, AIF, FLAC, OGG, AAC и др.

г) редактирование аудиофайлов: склейка, обрезка, конвертация форматов

5. Перечень подлежащих разработке вопросов

- 1) Анализ предметной области
- 2) Анализ требований и проектирование
- 3) Реализация
- 4) Тестирование
- 5) Оформление ВКР

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области		01.04.2021 – 09.04.2021	<i>Громов</i>
2	Анализ требований и проектирование		10.04.2021 – 19.04.2021	<i>Громов</i>
3	Реализация		20.04.2021 – 09.05.2021	<i>Громов</i>
4	Тестирование		10.05.2021 – 14.05.2021	<i>Громов</i>
5	Оформление ВКР		15.05.2021 – 20.05.2021	<i>Громов</i>

Дата выдачи задания "28" декабря 2020 г

Руководитель бакалаврской работы *Громов* 14.06.21.
(подпись, дата)

Задание к исполнению принял студент *ИИ*
(подпись, дата)

Работу к защите допустить
Декан факультета *А.С. Рогова*
(подпись, дата)
24.06.2021

Реферат

Пояснительная записка содержит 76 листов, 42 рисунка, 13 таблиц, 20 использованных источников, 2 приложения.

ПРОГРАММНЫЕ СРЕДСТВА ВОСПРОИЗВЕДЕНИЯ И РЕДАКТИРОВАНИЯ СИСТЕМЫ АУДИОФАЙЛОВ, MICROSOFT VISUAL STUDIO, C#, WINDOWS FORMS, UML 2.0.

Объектом разработки являются программные средства воспроизведения и редактирования системы аудиофайлов.

Цель работы – разработка программных средств воспроизведения и редактирования системы аудиофайлов.

В результате проведенной работы созданы программные средства воспроизведения и редактирования системы аудиофайлов, которые позволяют воспроизводить и редактировать аудиофайлы различных форматов.

Разработка проводилась на языке программирования C#, в качестве сред разработки использовалась среда программирования Visual Studio Professional 2017 и ОС Windows 10.

					ПГУ 09.03.02–8БР171.11 ПЗ				
Изм.	Лист	№ докум.	Подпись	Дата	Программные средства воспроизведения и редактирования системы аудиофайлов. Пояснительная записка.	Лит.	Лист	Листов	
Разраб.		Макаров К.А.		14.06.21			4	76	
Провер.		Прошкина Е.Н.		14.06.21					
Н. Контр.		Такташкин Д.В.		22.06.21					
Утверд.		Макарычев П.П.							
						Группа 17ВИ1			

Содержание

Введение	8
1 Анализ предметной области и постановка задачи	9
1.1 Анализ предметной области	9
1.2 Анализ существующих систем для работы с аудиофайлами.....	11
1.3 Постановка задачи.....	19
1.4 Анализ требований к системе	20
1.4.1 Анализ функциональных требований к системе	20
1.4.2 Анализ нефункциональных требований к системе	25
2 Проектирование системы.....	26
2.1 Выбор технологии проектирования	26
2.2 Выбор инструментальных средств разработки	26
2.3 Проектирование программных средств	30
2.3.1 Диаграмма вариантов использования	30
2.3.2 Диаграмма классов	33
2.3.3 Диаграмма состояний.....	34
2.3.4 Диаграмма деятельности.....	35
2.3.5 Диаграмма последовательности	36
2.3.6 Диаграмма коопераций	38
2.4 Проектирование механизма работы с аудиофайлами	40
2.4.1 Работа с аудиофайлами в модуле аудиоплеера	40
2.4.2 Работа с аудиофайлами в модуле аудиоредактора	41
2.5 Проектирование пользовательского интерфейса	43
2.6 Планирование разработки и оценка бюджета.....	44
3 Реализация системы	48

3.1 Реализация программных средств	48
3.1.1 Архитектура проигрывателя	48
3.1.2 Реализация модуля аудиоплеера	49
3.1.3 Реализация модуля аудиоредактора	53
3.2 Реализация пользовательского интерфейса	54
4 Контроль качества программного обеспечения	58
4.1 Выбор режима тестирования	58
4.2 Функциональное тестирование	59
4.3 Оценка качества программного обеспечения	60
Заключение	63
Список использованных источников	64
Приложение А. Результаты тестирования	66
Приложение Б. Код программы	71

Введение

На сегодняшний день музыка и всё, что с ней связано, занимают особое место в жизни человека. После появления портативных плееров, мобильных телефонов и стереосистем, композиции различных жанров, направлений и времён не перестают радовать слушателей ни на секунду. Ежедневно люди записывают, воспроизводят, слушают и редактируют миллионы аудиофайлов по всему миру.

Аудиопроигрыватель – приложение для воспроизведения и прослушивания музыкальных композиций.

Аудиоредактор – приложение для редактирования и создания аудиофайлов.

В настоящее время существует множество онлайн и офлайн музыкальных плееров и редакторов с различным функционалом и уровнем доступности. Однако даже среди такого разнообразия является редкостью продукт, который совмещал бы в себе функционал и проигрывателя, и редактора одновременно.

Целью выпускной квалификационной работы является разработка программного обеспечения, которое позволило бы пользователю быстро и удобно отредактировать и прослушать аудиофайл.

1 Анализ предметной области и постановка задачи

1.1 Анализ предметной области

При анализе предметной области выделяются основные сущности и моменты, определяются требования к функциональности системы, обозначаются границы проекта.

Анализ существующих аудиоплееров и аудиоредакторов дает представление о рынке подобных систем. Производя оценку уже созданных и проверенных временем приложений, можно грамотнее и точнее определить требования к своему продукту.

Аудиопроигрыватель – приложение для воспроизведения и прослушивания музыкальных композиций.

Аудиоредактор – приложение для редактирования и создания аудиофайлов.

В настоящее время существует множество онлайн и офлайн музыкальных плееров и редакторов с различным функционалом и уровнем доступности. Однако даже среди такого разнообразия является редкостью система, которая совмещала бы в себе функционал и проигрывателя, и редактора одновременно.

Аудиоредактор включает такие функции, как:

- конвертация форматов;
- склейка аудиофайлов;
- обрезка аудиофайлов.

Аудиоплеер включает все стандартные функции проигрывателя, например:

- прослушивание музыки;
- добавление аудиофайла в плейлист;
- изменение громкости.

Для достижения цели, были определены и решены следующие задачи:

- подобраны и изучены ресурсы по выбранной теме;

- проведен сравнительный анализ программных продуктов-аналогов;
- разработаны диаграммы, описывающие работу приложения;
- разработана архитектура приложения в рамках библиотек NAudio.dll и bass.dll;
- разработаны методы, обеспечивающие работу приложения.

В ходе выполнения работы, были изучены и применены на практике библиотеки NAudio.dll и bass.dll, язык программирования C#, технология Windows Forms. Были разработаны классы и методы для работы с аудиофайлами.

В результате работы было разработано приложение, содержащее функционал аудиопроигрывателя и аудиоредактора.

В процессе разработки были использованы следующие средства:

- язык программирования C#;
- библиотека для работы со звуком bass.dll;
- библиотека для работы со звуком NAudio.dll;
- инструмент для моделирования, анализа, документирования и оптимизации бизнес-процессов AllFusion Process Modeler;
- унифицированный язык моделирования (UML 2.0);
- технология для создания клиентских приложений Windows Forms.

Должен быть подключен весь стандартный набор библиотек, а также библиотеки для работы с аудиофайлами bass.dll и NAudio.dll.

Приложение должно работать с множеством музыкальных форматов: MP3, WAV, AIFF, FLAC, AAC, OGG, WMA, OGG, OPUS, APE, TTA, AC3.

В дальнейшем планируется продолжать работу над проектом и развивать функционал.

1.2 Анализ существующих систем для работы с аудиофайлами

Одним из самых ярких и интересных представителей рынка является бесплатный проигрыватель с закрытым исходным кодом AIMP, написанный на Delphi.

Пример интерфейса программы на рисунке 1.

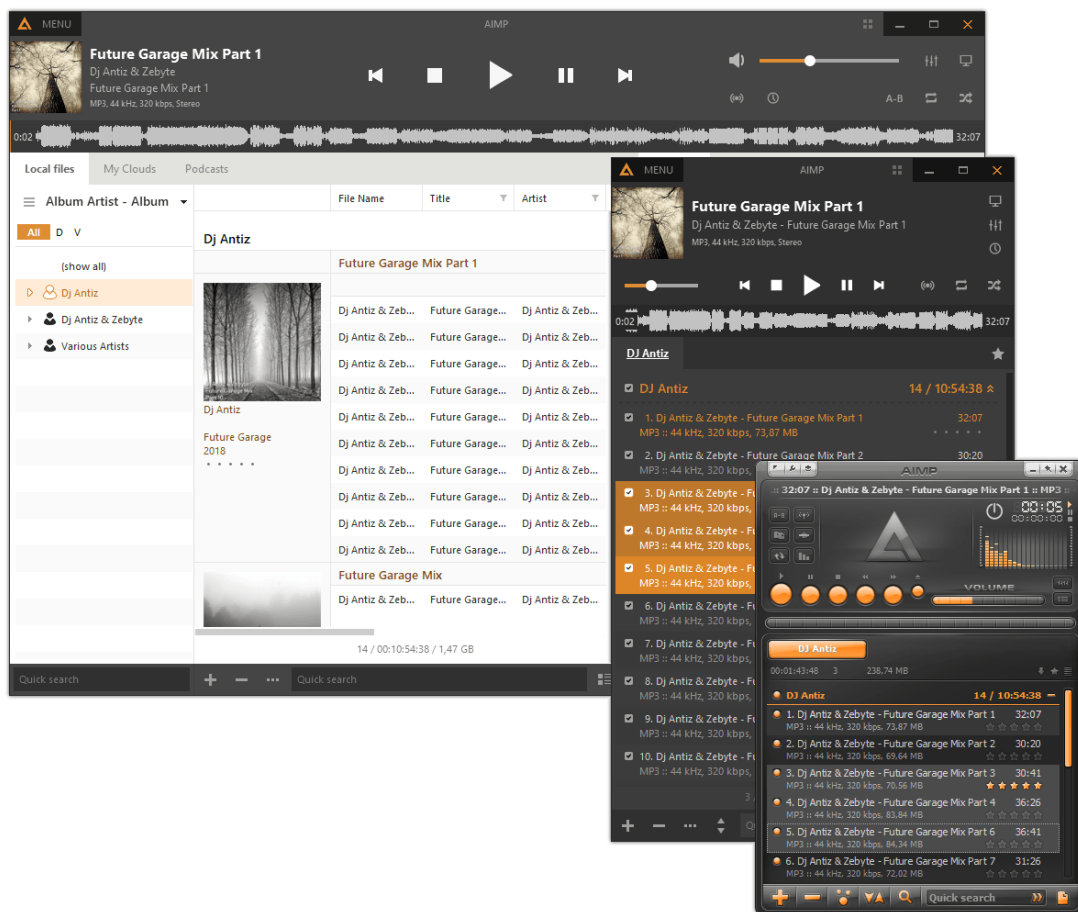


Рисунок 1 – Интерфейсы AIMP

Возможности и функции:

- одновременная работа с несколькими списками воспроизведения;
- чувствительный 18-полосный эквалайзер;
- встроенные звуковые эффекты: реверберация, фланжер, хорус, питч, темп, эхо, скорость, бас, усиление, подавление голоса;
- минимальное потребление системных ресурсов;
- создание закладок и очереди воспроизведения;

- встроенная фонотека имеет мощные инструменты для каталогизации музыкальной коллекции (поиск, сортировка, группировка, фильтрация);
- множество поддерживаемых музыкальных форматов, среди которых: MPEG-1, MPEG 2, OGG, WAV, FLAC, APE, Opus;
- возможность подключения внешней фонотеки для каталогизации музыкальной коллекции;
- поддержка плагинов;
- поддержка потокового содержимого (интернет-радио);
- совместимость со скробблером Last.fm;
- сворачивание проигрывателя в маленькую панельку (Tray Control) либо в область уведомлений;
- информация о файле – показ сведений о файле/формате при наведении мыши на значок проигрывателя в области уведомлений, а также (опционально) при старте следующей звуковой дорожки вверху/внизу экрана;
- поддержка работы с несколькими плейлистами;
- гибкие настройки программы;
- поиск файлов по всем открытым спискам воспроизведения;
- сохранение списка треков и воспроизведения;
- настраиваемые горячие клавиши;
- возможность выборочного добавления аудиофайлов и папок в плейлист [1].

Несмотря на наличие несущественных багов и порой запутанного, по мнению некоторых пользователей, интерфейса, AIMP является очень сильным программным продуктом, который постоянно дорабатывается и совершенствуется, получая положительные оценки.

Интерфейс AIMP часто сравнивают с Winamp, называя схожесть недостатком, но чаще этому не придаётся особого значения.

Winamp является ретро-гигантом в области аудиопроигрывателей, который очень сильно повлиял на развитие индустрии и, в своё время, задал большое количество определяющих сферу векторов.

Пример интерфейса программы на рисунке 2.

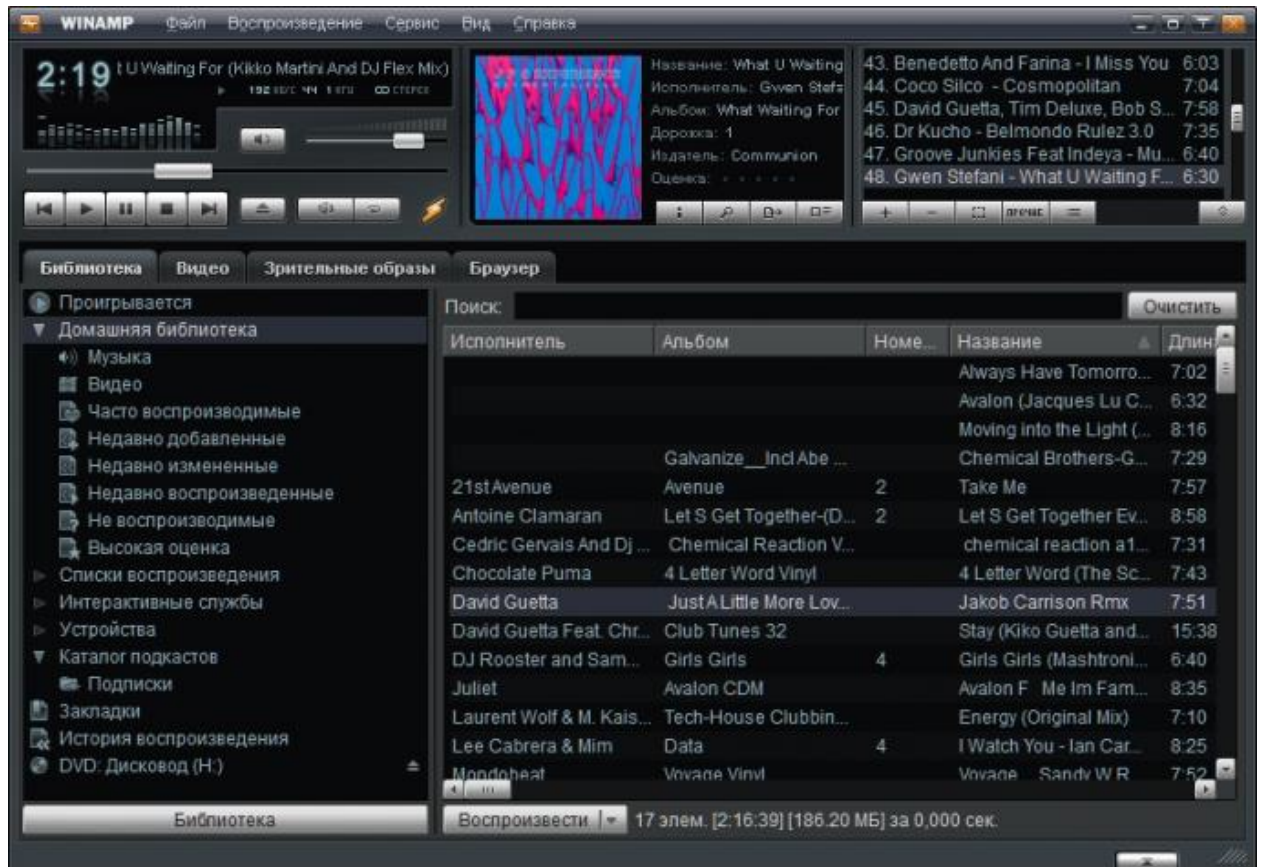


Рисунок 2 – Интерфейс Winamp

Плеер поддерживает большое количество музыкальных форматов и обладает богатым функционалом для работы с аудиофайлами:

- поддерживает большое количество форматов видео и аудио;
- воспроизводит MIDI и трекерную музыку;
- воспроизводит дорожки со звуковых компакт-дисков (Audio CD);
- воспроизводит потоки (например, SHOUTcast);
- поддерживает 24-битный вывод;
- поддерживает объёмное звучание (surround);
- поддерживает стандарт Replay Gain (выравнивание громкости);
- предотвращает клиппирование;

- поддерживает аппаратное ускорение;
- поддерживает плавный переход между дорожками (crossfade);
- поддерживает непрерывное воспроизведение (gapless);
- отображает спектрограмму и осциллограмму;
- отображает красивые зрительные образы, подобия которых в других мультимедийных проигрывателях и не встретишь;
- принимает аудиосигнал с линейного входа;
- имеет удобный и интуитивно понятный пользовательский интерфейс;
- предлагает множество настроек;
- позволяет украсить пользовательский интерфейс по вкусу пользователя;
- позволяет расширить возможности с помощью подключаемых модулей;
- имеет мультимедийную библиотеку, которая поможет с управлением и организацией файлов (поиск, сортировка, фильтрация);
- позволяет управлять воспроизведением с помощью сочетаний клавиш и мультимедийных кнопок;
- поддерживает работу с портативными плеерами и USB-накопителями [2].

Winamp является проверенным и надежным аудиопроигрывателем, но в настоящее время теряется среди множества других более молодых аналогов. ПО находится в свободном доступе, и любой желающий может без проблем попробовать его на деле, но в последнее время обновление функционала, добавление новых библиотек и установка новых актуальных плагинов происходят достаточно редко.

FL (Fruity Loops) Studio представляет собой полную программную среду или DAW (англ. digital audio workstation) для сочинения, аранжировки, записи, микширования и мастеринга современной музыки профессионального качества. FL разработан и написан на Delphi.

FL Studio позволяет осуществлять чистейшую мультитрековую аудиозапись и редактирование с возможностью растягивания времени (Time Stretching), изменения высоты звука (Pitch Shifting), искривления звука (Audio Warping) и всех стандартных манипуляций (копирование, вырезание) [3].

Пример интерфейса программы на рисунке 3.

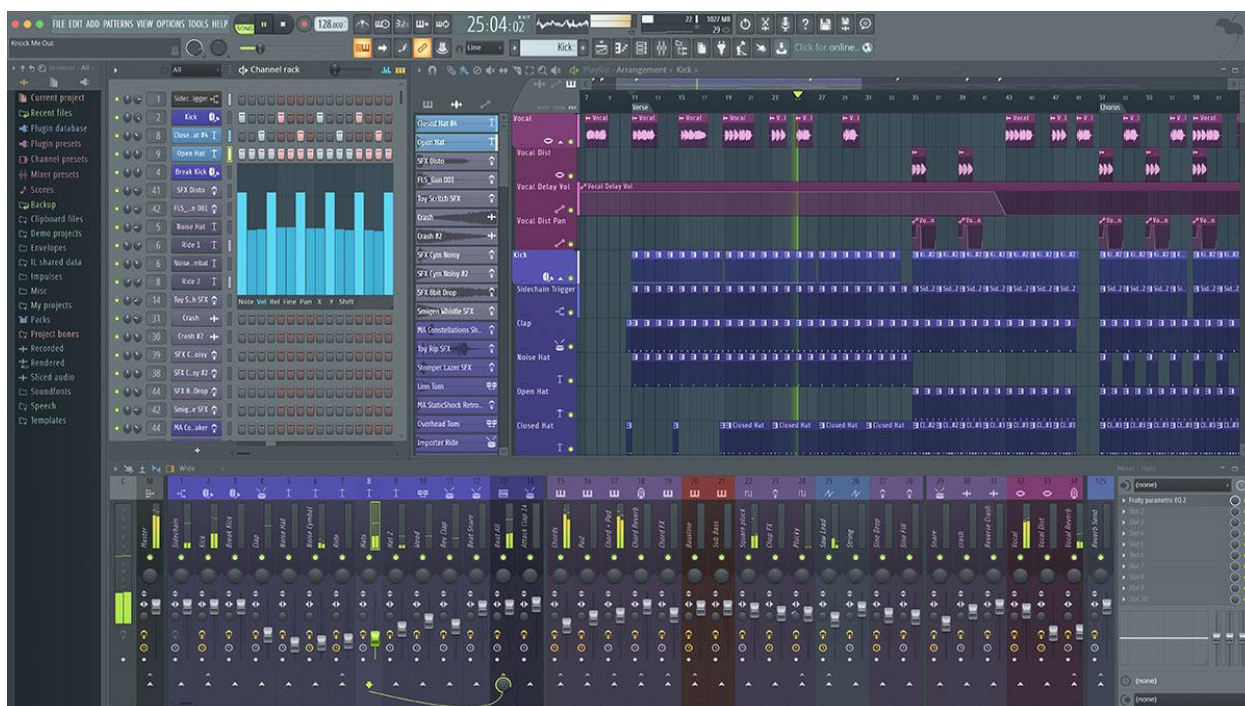


Рисунок 3 – Основное окно FL Studio

Функционал и возможности:

- обеспечивает полный доступ к мощному Piano Roll редактору, позволяющему пользоваться виртуальными инструментами для создания секвенций и аранжировки;
- дает полный доступ к мощному шаговому секвенсору (step sequencer), более чем к 18 виртуальным инструментам и более чем к дюжине встроенных эффектов и процессоров;

- позволяет автоматизировать большинство параметров интерфейса и плагинов с помощью записи, рисования, сплайн кривых и генераторов автоматизации;
- поддерживает все компьютерные стандарты плагинов инструментов, включая VST, DX, Buzz и Rewire;
- плейлист в FL Studio является чрезвычайно гибким и практически не ограниченным рабочим пространством, где нотные паттерны, аудио информация и данные автоматизации могут быть организованы и могут редактироваться;
- микшер обеспечивает точное и гибкое управление, а каждый трек микшера может содержать до 10 эффектов;
- имеет множество встроенных эффектов, включая эффект максимизации, лимиттер, компрессор, дилэй, дисторшн, эквалайзер (графический и параметрический) различные фильтры, фэйзер, флэнджер, хорус, вокодер и реверб [3].

Пример интерфейса редактора на рисунке 4.

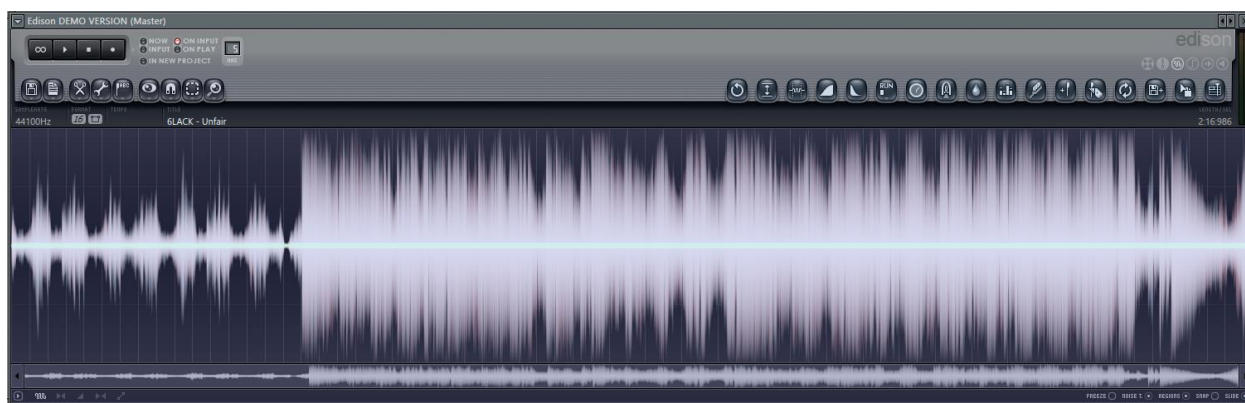


Рисунок 4 – Интерфейс функции «обрезать»

FL Studio является не просто редактором аудиофайлов, а полноценной профессиональной средой для написания и создания музыкальных композиций. FL имеет очень богатый функционал, поэтому, несмотря на достаточно органичный интерфейс, знакомство со средой может оказаться для новичков непростой задачей.

WavePad – полукоммерческая среда для работы и редактирования аудиофайлов, которая позволяет редактировать аудио, накладывать эффекты и конвертировать файлы в желаемое расширение. С его помощью можно ощутимо улучшить качество звучания через настройки эквалайзера и подавление шумов. Софт работает с большинством известных форматов (MP3, FLAC, AAC, M4A, M4R), а также способен проигрывать аудио из облачных хранилищ, таких как Dropbox и диск Google.

Пример интерфейса программы на рисунке 5.

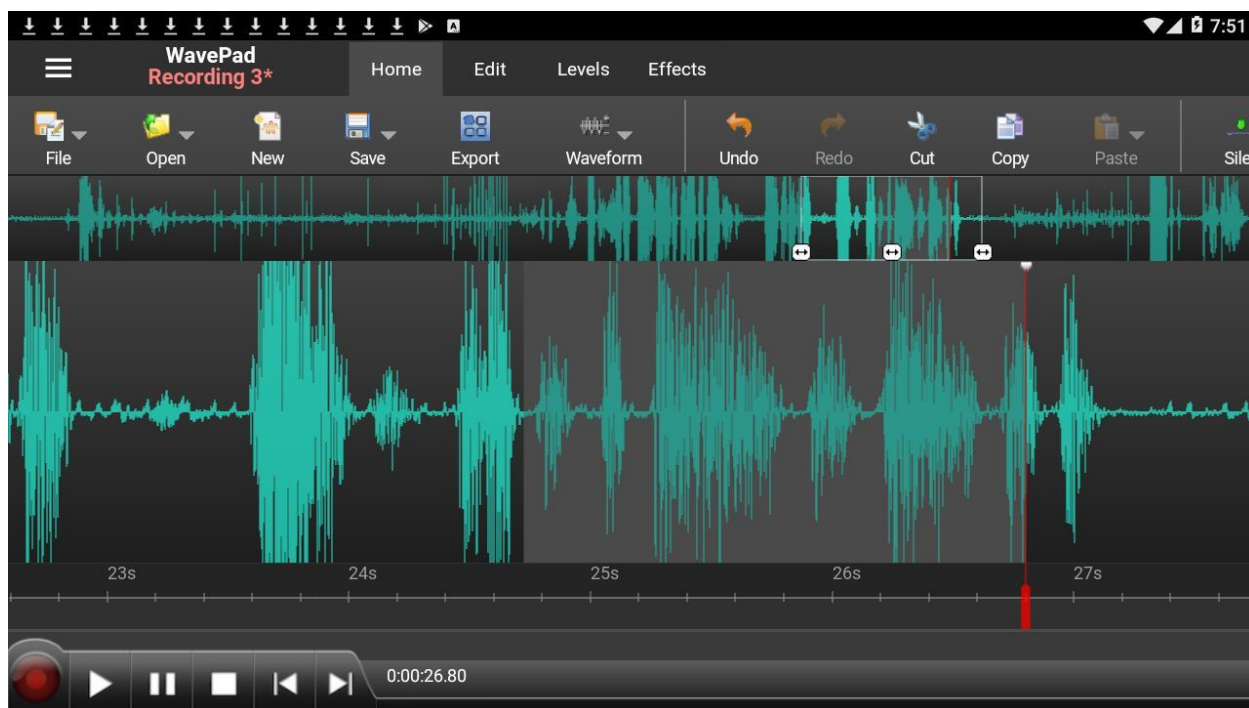


Рисунок 5 – Окно работы WavePad

Функционал и возможности:

- инструменты редактирования звука включают вырезание, копирование, вставку, удаление, затухание, автоматическую обрезку, сжатие, смещение высоты тона;
- звуковые эффекты включают усиление звука, нормализацию, эквалайзер, конвертацию, реверб, эхо, реверс звука;
- поддерживает форматы MP3, WAV, VOX, GSM, WMA, AU, AIF, FLAC, real audio, OGG, AAC, M4A, MID, AMR;
- прокрутка, поиск и закладки аудио для точного редактирования;

– простой и понятный интерфейс [4].

WavePad является средним аудиоредактором, в котором любой желающий может быстро и удобно совершить все основные необходимые манипуляции с аудиофайлом, не тратя много времени на знакомство с функционалом и не изучая подробно интерфейс.

Проанализировав ряд систем, можно сделать вывод, что важную роль играет целевая аудитория продукта. Задача состоит в том, чтобы оптимально совместить функции проигрывателя и редактора, при этом не усложняя интерфейс и учитывая весь набор необходимого (базового) и дополнительного функционалов для работы непрофессиональных пользователей. Результаты анализа представлены в таблице 1.

Таблица 1 – Результаты сравнительного анализа

Название программного продукта	Возможность воспроизведения музыкальных файлов	Возможность редактирования музыкальных файлов	Удобный и понятный интерфейс
AIMP	+	-	+
Winamp	+	-	-
FL Studio	+	+	-
WavePad	-	+	+
PLED (разрабатываемая система)	+	+	+

Проанализировав ряд аудиопроигрывателей и аудиоредакторов, можно утверждать, что в редакторах нельзя полноценно послушать музыку, а в проигрывателях, если и имеется, то самый скудный функционал по ее обработке.

Поэтому такое приложение как PLED (разрабатываемая система), которое совмещает в себе полноценный функционал и проигрывателя, и редактора, является редкостью.

1.3 Постановка задачи

В рамках выпускной квалификационной работы должны быть разработаны программные средства для воспроизведения и редактирования системы аудиофайлов.

В приложение должны входить:

- модуль аудиопроигрывателя;
- модуль аудиоредактора;
- вспомогательные модули для корректной работы двух предыдущих.

Аудиопроигрыватель, помимо своей основной функции – воспроизведения аудиофайлов различных форматов, должен обладать функциями, которые сделают работу с ним удобнее для пользователя.

Самой важной частью проигрывателя будет та, которая непосредственно отвечает за обработку и воспроизведение аудиофайла, организацию аудиопотоков и работу с форматами музыкальных файлов.

Не менее актуальными могут быть такие функции, как:

- музыкальные эффекты (эквалайзер, эффекты затухания, эхо);
- визуализация воспроизводимого потока;
- тегирование аудиофайлов;
- перемешивание композиций в плейлисте;
- функции радио и воспроизведения интернет-потоков.

Важную роль в подобных приложениях играет пользовательский интерфейс. Он должен быть максимально удобным для пользователя, а также соответствовать определенным нормам дизайна интерфейсов.

Аудиоредактор может обладать различным функционалом – всё зависит от целевой аудитории и масштаба самого редактора. Но существует набор функций, который должен включать в себя любой редактор от самого простого до самого продвинутого:

- изменение аудиофайла (обрезка, склеивание);

- работа с большинством известных форматов;
- конвертация форматов;
- звуковые эффекты (усиление звука, уменьшение шумов);
- простой и понятный интерфейс.

Основными задачами, которые решаются при выполнении выпускной квалификационной работы, являются:

- проектирование структуры обработки музыкальных файлов различных форматов;
- разработка модуля, отвечающего за работу музыкального плеера при помощи библиотеки BASS.dll;
- реализация классов для работы с тегами музыкальных файлов;
- разработка модуля, отвечающего за работу аудиоредактора при использовании библиотеки NAudio.dll;
- разработка удобного и понятного пользовательского интерфейса.

Исходя из этого, были определены требования, и спроектировано ПО для воспроизведения и редактирования системы аудиофайлов.

1.4 Анализ требований к системе

1.4.1 Анализ функциональных требований к системе

Функциональные требования описывают, что необходимо реализовать в продукте или системе, и какие действия должны выполнять пользователи при взаимодействии с ними.

Для определения функциональных требований используется методология SADT (англ. Structured Analysis and Design Technique, метод структурного анализа и проектирования), которая позволяет выделить, спроектировать, проанализировать и затем автоматизировать основные процессы, происходящие в системе.

Сначала строится контекстная диаграмма, которая является вершиной древовидной структуры диаграмм.

На рисунке 6 представлена контекстная диаграмма функционала аудиопроигрывателя в методологии IDEF0.

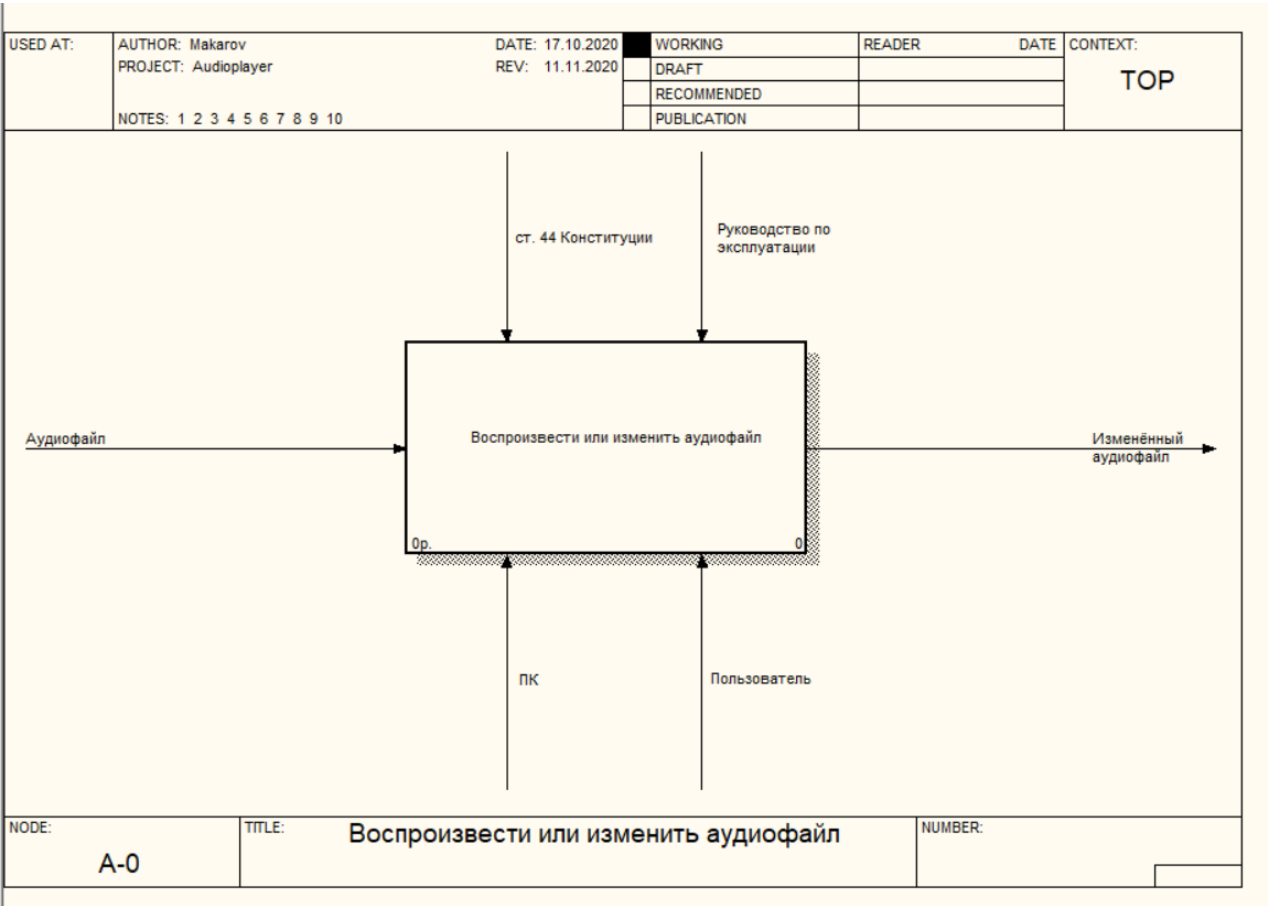


Рисунок 6 – Контекстная диаграмма IDEF0

Входными данными в данной контекстной диаграмме является аудиофайл. Выходными данными является изменённый аудиофайл. Ресурсами являются ПК и пользователь, а механизмами управления руководство по эксплуатации и ст. 44 Конституции РФ.

После описания основной функции выполняется функциональная декомпозиция, т.е. определяются функции, из которых состоит основная.

На основе созданной контекстной диаграммы в результате детализации обобщенного процесса «Воспроизвести или изменить аудиофайл» была разработана детализирующая диаграмма, представленная на рисунке 7.

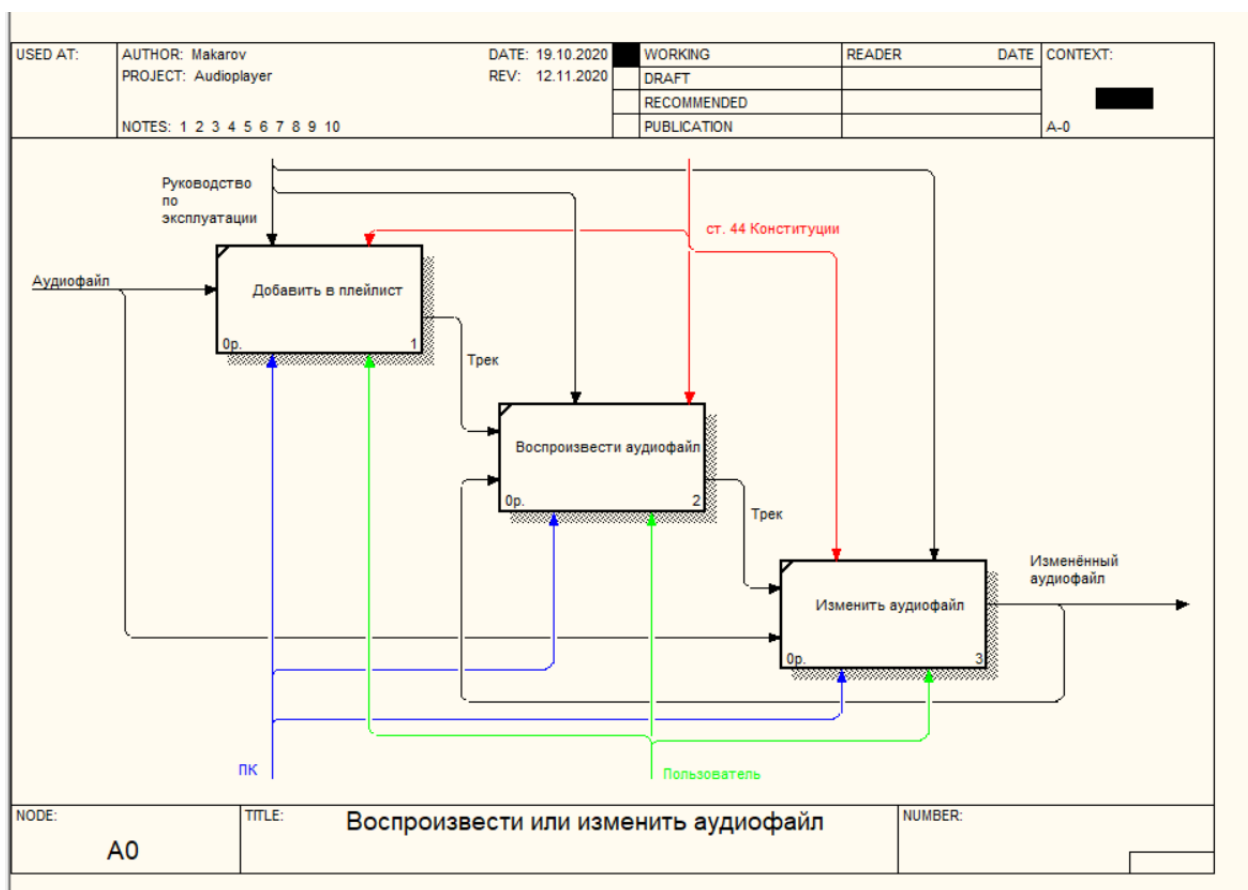


Рисунок 7 – Детализирующая диаграмма процесса «Воспроизвести или изменить аудиофайл»

При декомпозиции были выделены три процесса:

- процесс «Добавить в плейлист» предназначен для добавления аудиофайла в плейлист;
- процесс «Воспроизвести аудиофайл» предназначен для проигрывания композиции;
- процесс «Изменить аудиофайл» предназначен для редактирования аудиофайла.

Декомпозиция процесса «Воспроизвести аудиофайл» представлена на рисунке 8.

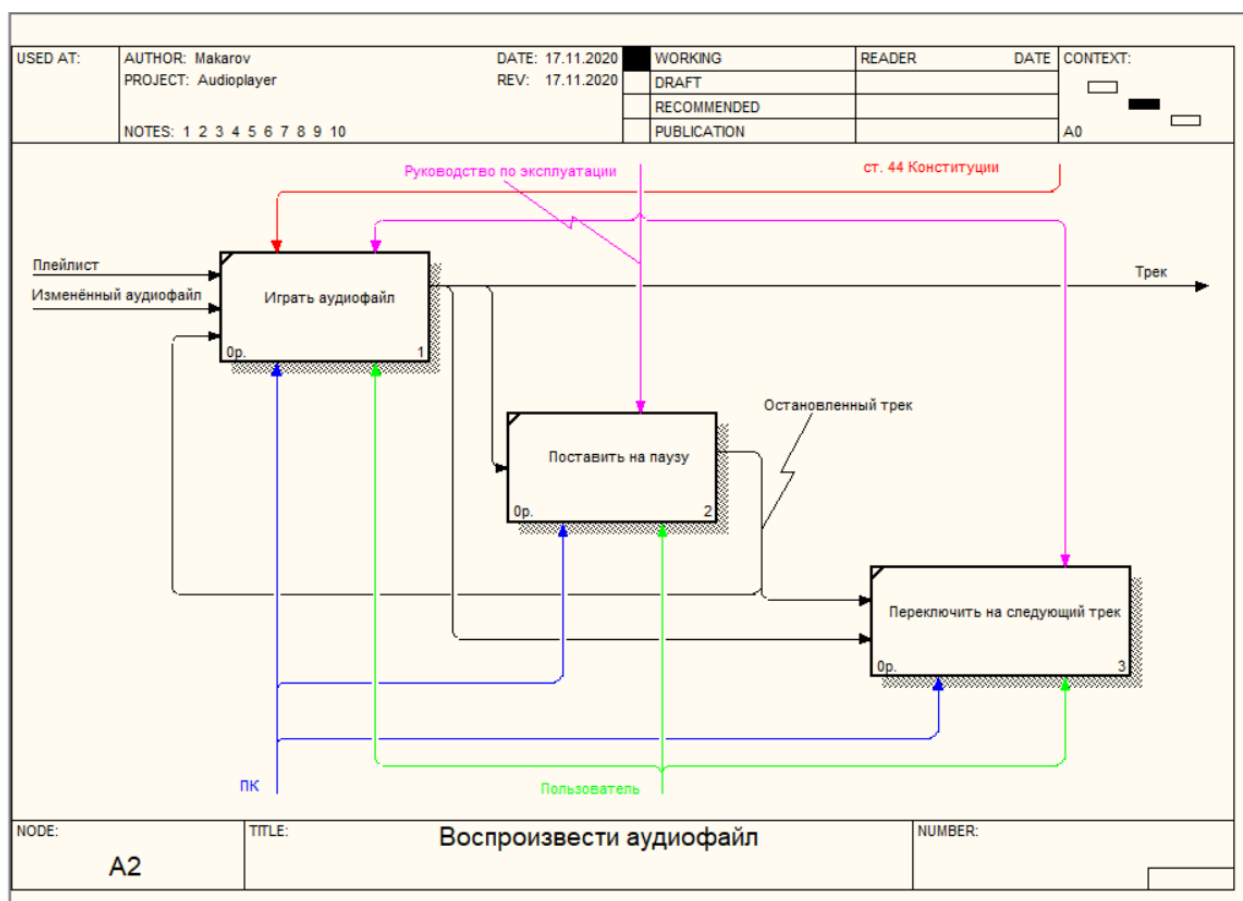


Рисунок 8 – Детализирующая диаграмма процесса «Воспроизвести аудиофайл»

При декомпозиции были выделены три процесса:

- процесс «Играть аудиофайл» предназначен для проигрывания аудиофайла;
- процесс «Поставить на паузу» предназначен для приостановки композиции;
- процесс «Переключить на следующий трек» предназначен для переключения на следующий/предыдущий аудиофайл.

Декомпозиция процесса «Изменить аудиофайл» в методологии IDEF3 представлена на рисунке 9.

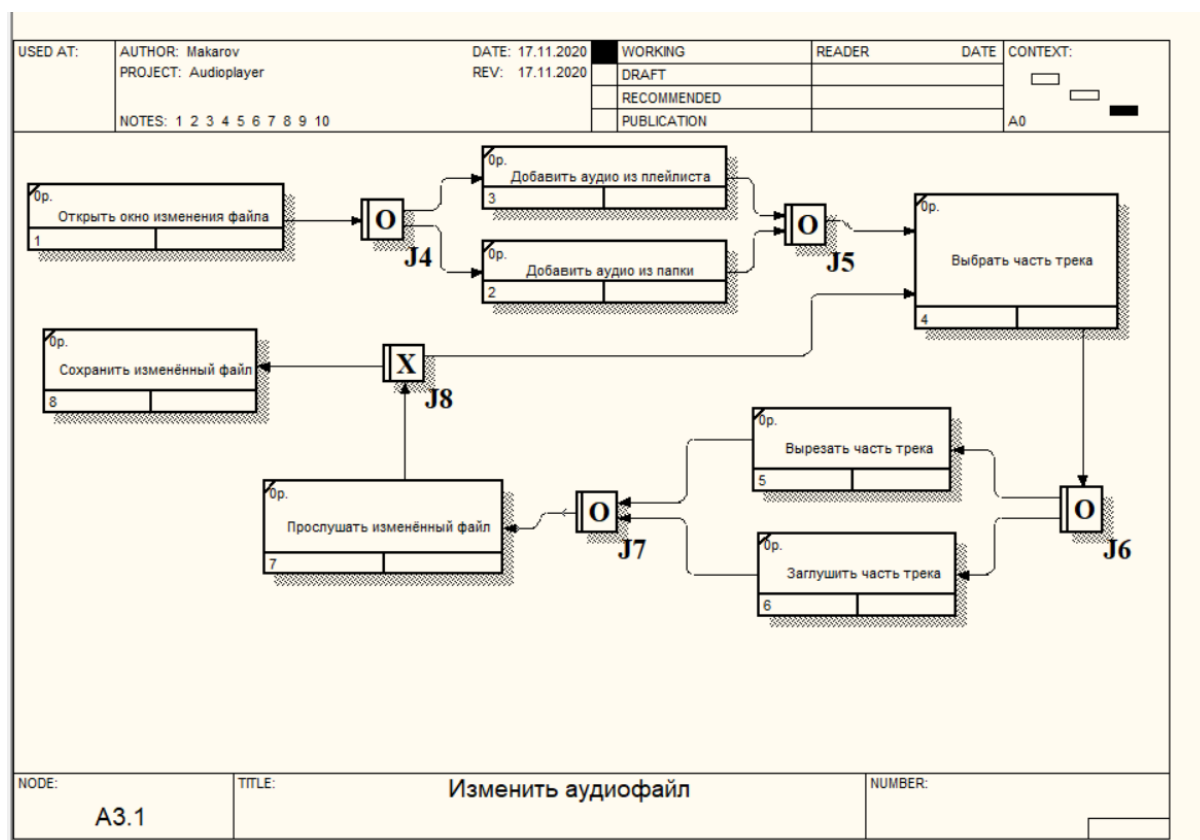


Рисунок 9 – Детализирующая диаграмма процесса «Изменить аудиофайл»

При декомпозиции были выделены следующие процессы:

- процесс «Открыть окно изменения аудиофайла» предназначен для открытия окна редактирования аудиофайла;
- процессы «Добавить аудио из плейлиста» и «Добавить аудио из папки» предназначены для добавления редактируемого аудиофайла;
- процесс «Выбрать часть трека» предназначен для установки указателя в определённый момент трека;
- процессы «Вырезать часть трека» и «Заглушить часть трека» предназначены для редактирования выбранной части трека;
- процесс «Прослушать измененный файл» предназначен для воспроизведения отредактированного аудиофайла;
- процесс «Сохранить измененный файл» предназначен для сохранения отредактированного аудиофайла.

В результате проектирования были определены функциональные требования к системе.

1.4.2 Анализ нефункциональных требований к системе

Нефункциональные требования описывают, как именно должен работать программный продукт или система, и какими свойствами или характеристиками она должна обладать.

К нефункциональным требованиям можно отнести:

- пользовательский интерфейс;
- производительность;
- использование программных средств;
- использование библиотек;
- форматы файлов, с которыми будет работать система.

Пользовательский интерфейс обеспечивает взаимодействие пользователя с системой и должен соответствовать определённым требованиям:

- интерфейс должен максимально покрывать функционал системы;
- основные элементы должны быть всегда видимы;
- возможность изменения интерфейса.

Приложение должно обеспечивать стабильную работу. FPS (количество кадров в секунду) не должно опускаться ниже значения в 30 кадров.

ПО должно быть реализовано с использованием среды разработки Visual Studio, язык программирования C# и технология Windows Forms.

Должен быть подключен весь стандартный набор библиотек, а также библиотеки для работы с аудиофайлами `bass.dll` и `NAudio.dll`.

Приложение должно работать с множеством музыкальных форматов: MP3, WAV, AIFF, FLAC, AAC, OGG, WMA, OGG, OPUS, APE, TTA, AC3.

2 Проектирование системы

2.1 Выбор технологии проектирования

При проектировании информационной системы были изучены и использованы такие средства, как инструмент UML-моделирования ArgoUML, инструмент для моделирования, анализа, документирования и оптимизации бизнес-процессов AllFusion Process Modeler и CASE средство ERwin Data Modeler.

Унифицированный язык моделирования (Unified Modeling Language) – UML – это система обозначений, которую можно применять для объектно-ориентированного анализа и проектирования. Его можно использовать для визуализации, спецификации, конструирования и документирования программных систем [5].

ArgoUML – разработанный на Java инструмент, который позволяет создавать UML-модели, совместимые с принятыми стандартами. С помощью ArgoUML можно создать 9 типов диаграмм: диаграммы прецедентов, диаграммы классов, диаграммы состояний, диаграммы последовательности, диаграммы деятельности, диаграммы взаимодействия и диаграммы развертывания.

AllFusion Process Modeler (далее BPwin) – CASE-средство для моделирования бизнес-процессов, позволяющая создавать диаграммы в нотации IDEF0, IDEF3, DFD. В процессе моделирования BPwin позволяет переключиться с нотации IDEF0 на любой ветви модели на нотацию IDEF3 или DFD и создать смешанную модель. BPwin поддерживает функционально-стоимостной анализ (ABC) [8].

2.2 Выбор инструментальных средств разработки

При выборе технологии для разработки и реализации задуманной системы, учитывались такие факторы, как:

- стабильность технологии;

- наличие необходимой документации;
- доступность технологии;
- относительно быстрое время разработки;
- производительность.

Учитывая эти пункты, была выбрана среда разработки Visual Studio, язык программирования C# и технология Windows Forms.

C# – современный объектно-ориентированный и типобезопасный язык программирования. C# позволяет разработчикам создавать множество типов безопасных и надежных приложений, работающих в экосистеме .NET.

C# предоставляет языковые конструкции для непосредственной поддержки объектно-ориентированной концепции работы. Благодаря этому C# подходит для создания и эксплуатации программных компонентов. С момента создания язык C# обогатился функциями поддержки новых рабочих нагрузок и современными рекомендациями по разработке ПО.

Программы C# выполняются в .NET, виртуальной системе выполнения, вызывающей общезыковую среду выполнения (CLR) и набор библиотек классов. Среда CLR – это реализация общезыковой инфраструктуры языка (CLI), являющейся международным стандартом. CLI является основой для создания сред выполнения и разработки, в которых языки и библиотеки прозрачно работают друг с другом.

Исходный код, написанный на языке C#, компилируется в промежуточный язык (IL), который соответствует спецификациям CLI. Код на языке IL и ресурсы, в том числе растровые изображения и строки, сохраняются в сборке, обычно с расширением .dll [9].

На рисунке 10 изображен процесс компиляции программы, написанной на языке C#.

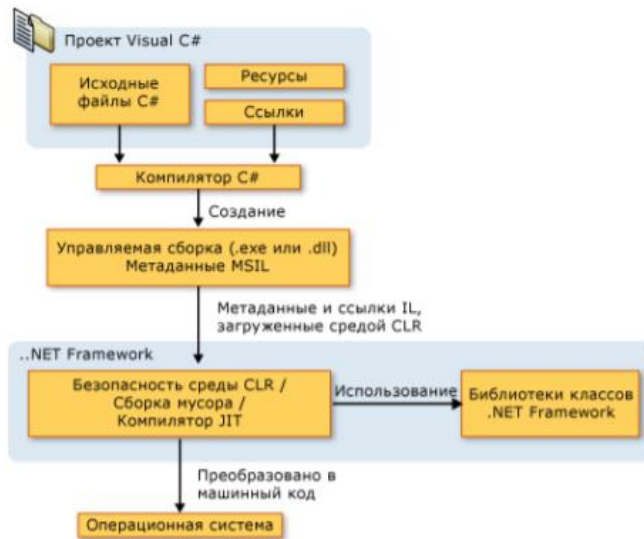


Рисунок 10 – Процесс компиляции программы на языке C#

Visual Studio – это набор инструментов разработки, основанных на использовании компонентов, и других технологий для создания мощных, производительных приложений.

На рисунке 11 представлен пример интерфейса Visual Studio.

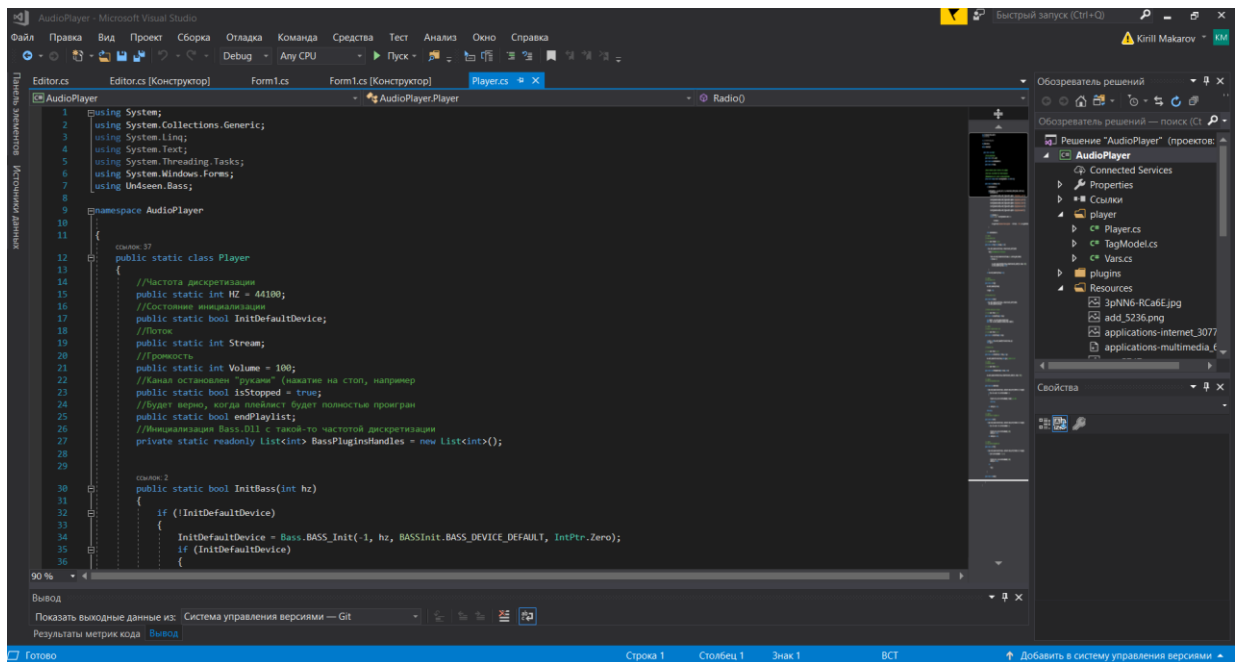


Рисунок 11 – Интегрированная среда разработки Visual Studio

Кроме того, среда Visual Studio оптимизирована для совместного проектирования, разработки и развертывания корпоративных решений.

Visual Studio позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ [10].

Windows Forms – интерфейс программирования приложений (API), отвечающий за графический интерфейс пользователя и являющийся частью Microsoft .NET Framework.

На рисунке 12 представлен пример интерфейса Windows Forms.

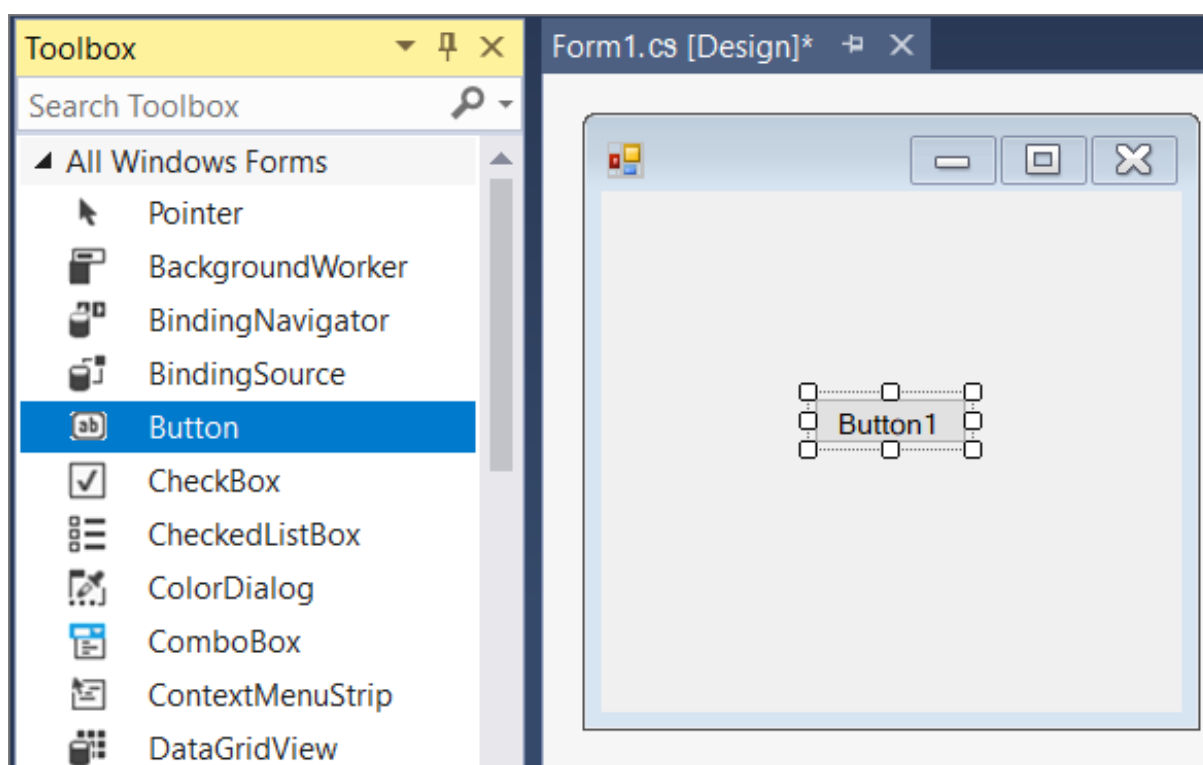


Рисунок 12 – Пример интерфейса Windows Forms

Данный интерфейс упрощает доступ к элементам интерфейса Microsoft Windows за счет создания обёртки для существующего Win32 API в управляемом коде.

2.3 Проектирование программных средств

Целью проектирования является определение внутренних свойств системы и детализации её внешних (видимых) свойств на основе требований к ПО (исходных условий задачи).

Проектирование ПО включает следующие основные виды деятельности:

- выбор метода и стратегии решения;
- выбор представления внутренних данных;
- разработка основного алгоритма;
- документирование ПО;
- тестирование и подбор тестов;
- выбор представления входных данных [11].

Проектирование системы осуществляется с использованием диаграмм UML с помощью CASE-средства ArgoUML.

2.3.1 Диаграмма вариантов использования

Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе ее проектирования и разработки. Данная диаграмма состоит из актеров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения.

Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества актеров, взаимодействующих с системой с помощью так называемых вариантов использования. При этом актером (действующим лицом, актантом, актором) называется любой объект, субъект или система, взаимодействующая с моделируемой системой извне. В свою очередь вариант использования – это спецификация сервисов (функций), которые система предоставляет актеру. Другими словами, каждый вариант использования определяет некоторый набор действий, совершаемых

системой при взаимодействии с актером. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий [5].

Диаграмма вариантов использования представлена на рисунке 13.

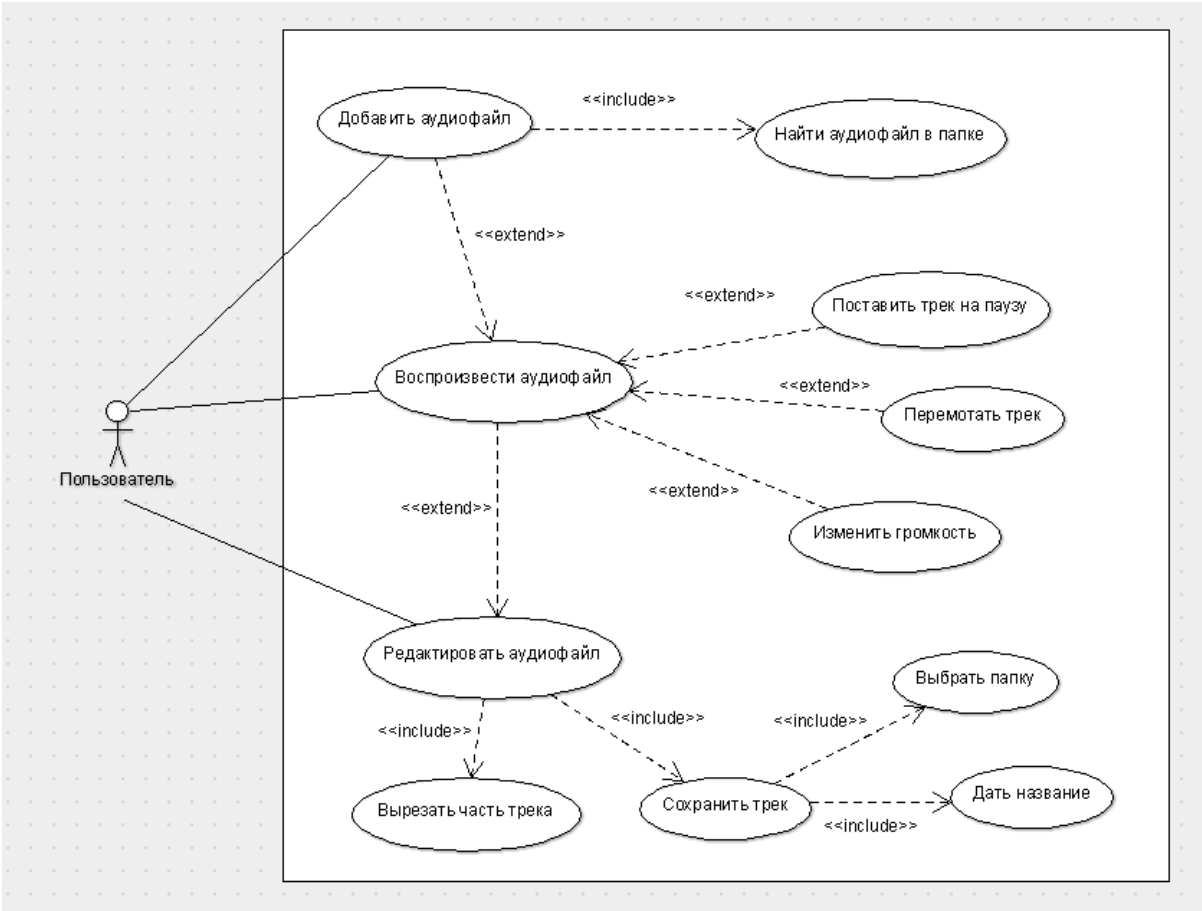


Рисунок 13 – Диаграмма вариантов использования

Действующим лицом является «Пользователь», то есть человек, который пользуется приложением.

Таблица 2 – Описание диаграммы вариантов использования

Вариант использования	Описание	Предусловие	Область действия	Основной сценарий
«Добавить аудиофайл»	Поиск и добавление аудиофайла в плейлист	Пользователь открыл аудиопроигрыватель	Аудиопроигрыватель, ФС ПК	– открыть аудиоплеер; – нажать кнопку «Добавить аудиофайл»; – найти папку с файлом; – найти файл в папке.

Продолжение таблицы 2

Вариант использования	Описание	Предусловие	Область действия	Основной сценарий
«Найти аудиофайл в папке»	Найти аудиофайл в папке на ПК	Пользователь открыл аудиопроигрыватель и находится в окне добавления аудиофайла.	Аудиопроигрыватель, ФС ПК	<ul style="list-style-type: none"> – Нажать кнопку «Добавить аудиофайл»; – Найти файл в папке в ФС.
«Редактировать аудиофайл»	Редактирование аудиофайла	Пользователь открыл окно редактирования, добавил трек	Аудиоредактор	<ul style="list-style-type: none"> – зайти «Редактирования аудиофайла»; – добавить файл; – выбрать часть трека, которую необходимо редактировать;
«Вырезать часть трека»	Вырезать необходимую часть трека	Пользователь находится в окне редактора.	Аудиоредактор	<ul style="list-style-type: none"> – выбрать первый тайминг; – выбрать второй тайминг; – вырезать часть трека.
«Сохранить аудиофайл»	Сохранить аудиофайл после редактирования	Файл отредактирован	Аудиоредактор	<ul style="list-style-type: none"> – нажать кнопку «Сохранить аудиофайл»; – выбрать директорию для сохранения.
«Воспроизвести аудиофайл»	Воспроизведение аудиофайла	Пользователь открыл аудиопроигрыватель, добавил аудиофайл в плейлист	Аудиопроигрыватель	<ul style="list-style-type: none"> – открыть аудиоплеер; – добавить аудиофайлы; – нажать кнопку воспроизведения.
«Поставить трек на паузу»	Остановить воспроизведение трека кнопкой «Пауза».	Пользователь прослушивает аудиофайл	Аудиопроигрыватель	<ul style="list-style-type: none"> – нажать на кнопку сохранения
«Изменить громкость»	Изменение громкости воспроизведения	Пользователь прослушивает аудиофайл	Аудиопроигрыватель	<ul style="list-style-type: none"> – перетащить слайдер; – включить режим «Без звука».
«Перемотать трек»	Перемотать трек в любой необходимый момент	Пользователь прослушивает аудиофайл.	Аудиопроигрыватель	<ul style="list-style-type: none"> – выбрать момент трека; – перемотать трек.

На основе прецедентов диаграммы вариантов использования строятся другие UML-диаграммы.

2.3.2 Диаграмма классов

Диаграмма классов отображает общую структуру иерархии классов. Диаграмма представлена на рисунке 14.

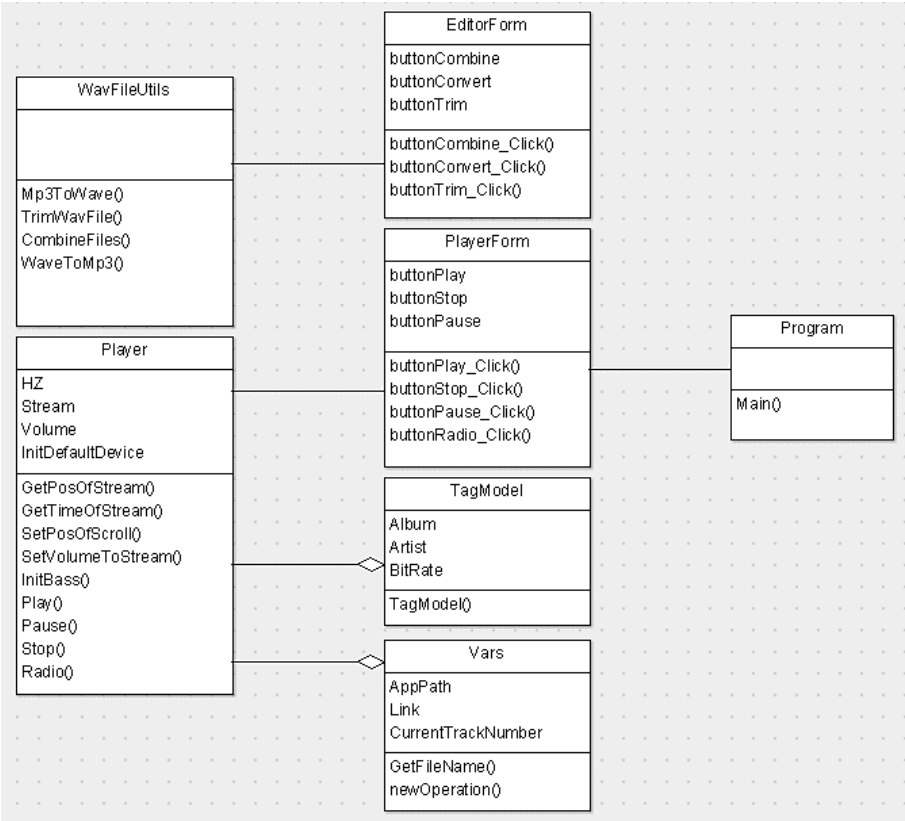


Рисунок 14 – Диаграмма классов

Описание отношений диаграммы классов представлено в таблице 3.

Таблица 3 – Описание диаграммы классов

Класс 1	Класс 2	Отношение	Описание
WavFileUtils	EditorForm	Ассоциация	Модуль редактора
Player	PlayerForm	Ассоциация	Модуль плеера
Player	TagModel	Агрегация	Использование тегов
Player	Vars	Агрегация	Использование переменных
PlayerForm	Program	Ассоциация	Program запускает PlayerForm

Поля и методы классов продемонстрированы и описаны в разделе разработки.

2.3.3 Диаграмма состояний

Диаграмма состояний описывает, как объект аудиофайла переходит из одного состояние в другое во время редактирования.

Диаграмма состояний представлена на рисунке 15.

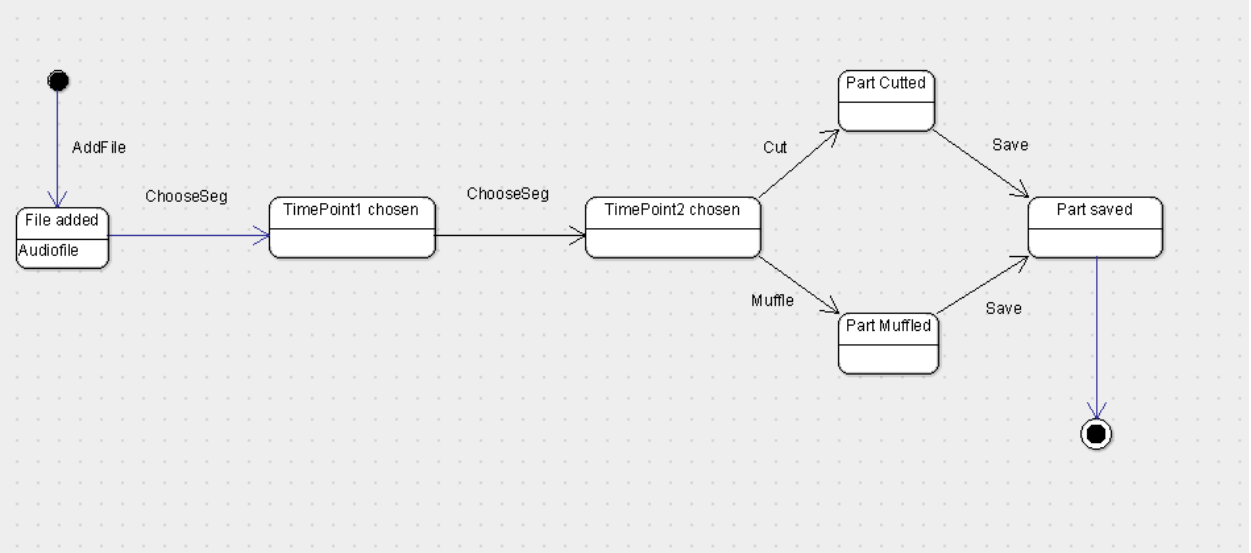


Рисунок 15 – Диаграмма состояний

Основными элементами диаграммы состояний являются «Состояние» и «Переход».

Описание диаграммы состояний представлено в таблице 4.

Таблица 4 – Описание диаграммы состояний

Состояние	Краткое описание	Вызывающие события (переходы)
Файл добавлен	Аудиофайл добавлен в приложение для дальнейшего редактирования	Добавление аудиофайла
Первый таймпойнт выбран	Выбран первый таймпойнт (точка времени) на треке	Поиск нужного для редактирования сегмента
Второй таймпойнт выбран	Выбран второй таймпойнт (точка времени) на треке	Поиск нужного для редактирования сегмента
Выбранный сегмент вырезан	Выбранный сегмент вырезан из трека	Вырезать часть трека, находящуюся между двумя таймпойнтами

Продолжение таблицы 4

Состояние	Краткое описание	Вызывающие события
Выбранный сегмент приглушён	Выбранный сегмент приглушён до определённого уровня громкости	Приглушить часть трека, находящуюся между двумя таймпойнтами
Изменённый аудиофайл сохранён	Сохранение отредактированного аудиофайла на ПК	Окончание редактирования

Диаграмма состояний отображает все возможные состояния, в которых может находиться объект, а также процесс смены этих состояний в результате внешнего влияния [6].

2.3.4 Диаграмма деятельности

Диаграмма деятельности наглядно отображает переходы потоков управления от одной деятельности к другой. Диаграмма описывает вариант использования «Воспроизвести аудиофайл».

Диаграмма деятельности представлена на рисунке 16.

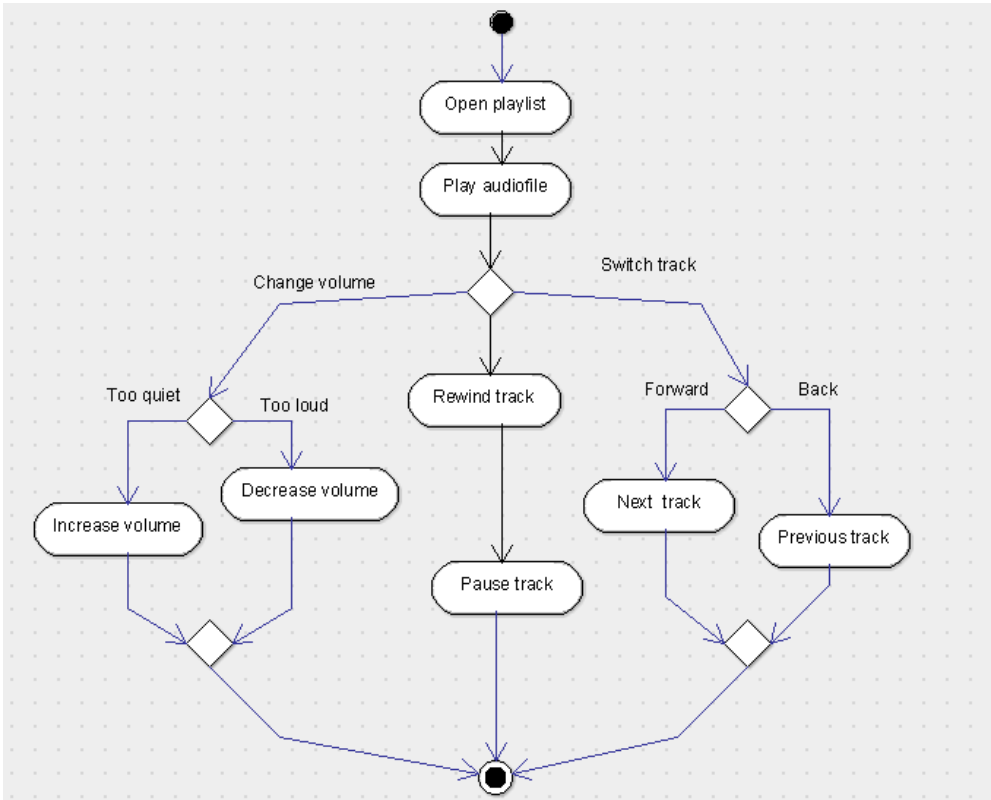


Рисунок 16 – Диаграмма деятельности

Описание диаграммы состояний представлено в таблице 5.

Таблица 5 – Описание диаграммы деятельности

Название узла	Тип узла	Описание	Входные потоки	Выходные потоки
Open playlist	Узел действия	Открыть плейлист	-	Play audiofile
Play audiofile	Узел действия	Воспроизвести аудиофайл	Open playlist	Increase/Decrease volume, Pause track, Rewind track, Next track, Previous track
Increase/Decrease volume	Узел действия	Увеличить уровень громкости	Play audiofile	-
Next/Previous track	Узел действия	Переключить на композицию	Play audiofile	-
Pause track	Узел действия	Приостановить воспроизведение	Play audiofile	-
Rewind track	Узел действия	Перемотать трек	Play audiofile	-

Узлы «Change volume/Switch track», «Too loud/Too quiet», «Forward/Back» являются узлами выбора решений.

2.3.5 Диаграмма последовательности

Диаграмма последовательности отображает жизненный цикл объекта и взаимодействие актеров для некоторого набора объектов на единой временной оси.

Диаграмма последовательности представлена на рисунке 17.

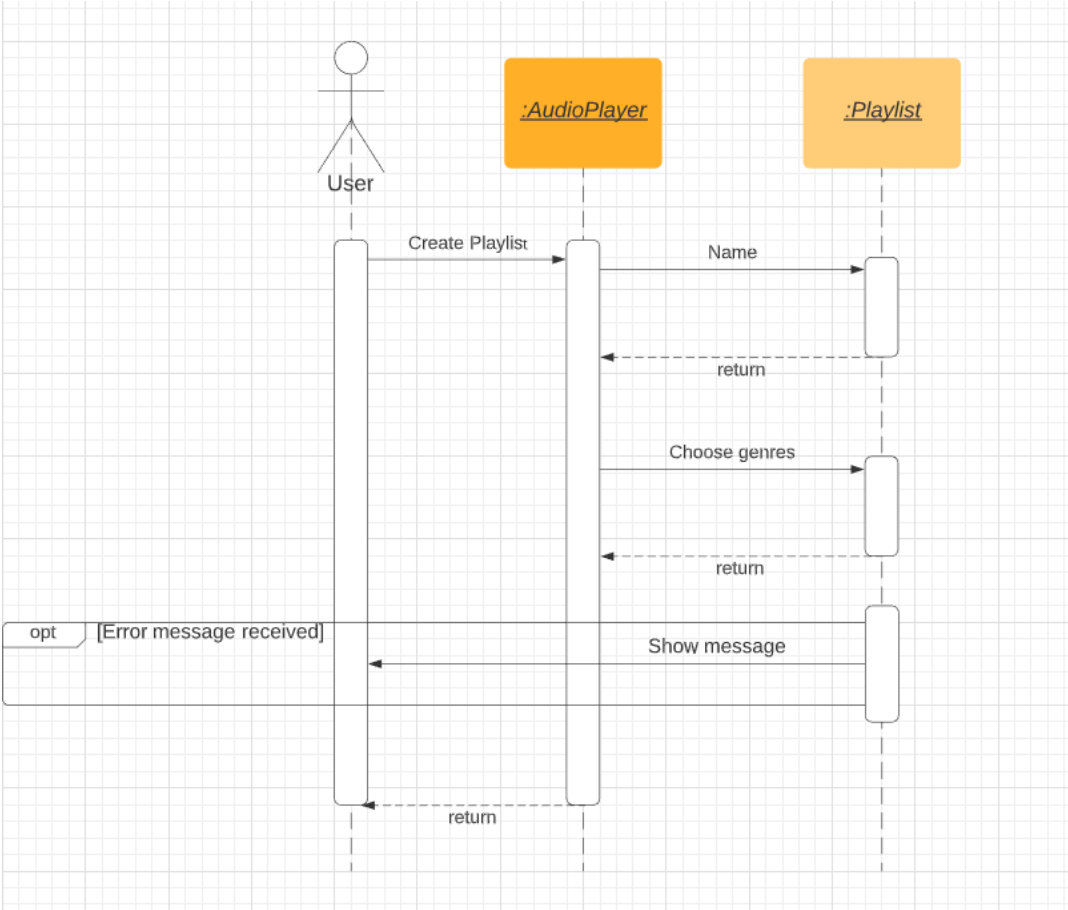


Рисунок 17 – Диаграмма последовательности

Описание линий жизни и сообщений диаграммы последовательности представлено в таблицах 6 и 7.

User – пользователь, человек, который пользуется системой.

Audio Player – система, воспроизводящая музыкальные файлы.

Таблица 6 – Линии жизни диаграммы последовательности

Линия жизни	Описание
User (Пользователь)	Человек, который пользуется аудиопроигрывателем
Audio Player (Плеер)	Основная область аудиопроигрывателя, в которой отображается трек, громкость и др.
Playlist (Плейлист)	Список добавленных композиции, список воспроизведения

Таблица 7 – Сообщения диаграммы последовательности

Сообщение	Описание	Инициатор	Получатель	Тип/Условие
Create Playlist	Создаёт новый плейлист	Playlist	Audio player	Вызов
Name	Даёт уникальное имя плейлисту	Playlist	User	Вызов
Choose genres	Выбирает основные жанры плейлиста	Playlist	User	Вызов
Show message	Передаёт сообщения об ошибке при названии/сохранении плейлиста	Playlist	User	Асинхронное/ opt [Error message received]

На диаграмме последовательности объекты в основном представляют экземпляры класса или сущности, обладающие поведением. В качестве объектов могут выступать пользователи, инициирующие взаимодействие, классы, обладающие поведением в системе или программные компоненты, а иногда и системы в целом.

2.3.6 Диаграмма коопераций

Диаграмма кооперации (collaboration) служит для обозначения множества взаимодействующих с определенной целью объектов в общем контексте моделируемой системы. Цель самой кооперации состоит в том, чтобы специфицировать особенности реализации отдельных наиболее значимых операций в системе.

Диаграмма коопераций представлена на рисунке 18.

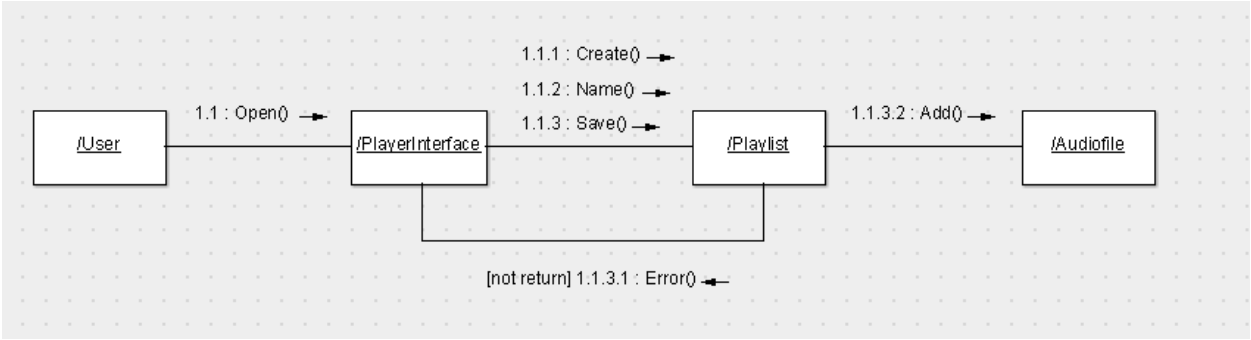


Рисунок 18 – Диаграмма кооперации

Описание объектов и связей диаграммы коопераций представлено в таблице 8.

Таблица 8 – Объекты и связи диаграммы коопераций

Название компонента	Описание	Входные потоки	Выходные потоки
: User	Интерфейс пользователя для работы с системой.	-	1.1. Open()
: PlayerInterface	Интерфейс плеера для работы с плейлистом, треком, функциями настройки воспроизведения.	1.1 : Open() 1.1.3.1 : Error()	1.1.1 : Create() 1.1.2 : Name() 1.1.3 : Save()
: AudioFile	Файл в формате .mp3/.wav, добавляемый в плейлист	1.1.3.2 Add()	-
: Playlist	Отображение добавленных композиций в виде списка (плейлиста)	1.1.1 : Create() 1.1.2 : Name() 1.1.3 : Save()	1.1.3.2 Add()

Кооперация определяет структуру поведения системы в терминах взаимодействия участников этой кооперации.

2.4 Проектирование механизма работы с аудиофайлами

2.4.1 Работа с аудиофайлами в модуле аудиоплеера

Для обработки аудиопотоков и аудиофайлов в плеере использовалась библиотека `bass.dll`.

BASS – это звуковая библиотека для использования в программном обеспечении на нескольких платформах. Библиотека предоставляет разработчикам мощный и эффективный инструмент для обработки воспроизведения аудиопотоков таких форматов, как: MP3, MP2, MP1, OGG, WAV, AIFF и многих других с помощью подключения плагинов, кодеков и надстроек ОС. Так же есть возможность прослушивания трекерной музыки (MOD) форматов: XM, IT, S3M, MOD, MTM, UMX, использование функции записи и работа с тегами музыкальных файлов.

Библиотека BASS.NET полностью написана на C# и работает с Microsoft .NET Framework v2.0, v3.0, v3.5 и v4.0 / 4.5 или выше (например, для платформ Windows x86 или x64), .NET Core v2.0 или выше и .NET Standard v2.0 или выше (например, для любой кроссплатформенной разработки).

Основной функционал аудиоплеера включает в себя:

- добавление аудиофайлов из ФС ПК в плейлист;
- загрузка плагинов музыкальных форматов;
- инициализация аудиопотока;
- воспроизведение музыкальных файлов;
- приостановка, переключение воспроизведения;
- изменение громкости воспроизведения;
- перемотка трека.

Помимо основного существует и дополнительный функционал:

- переключения композиций непосредственно в окне плейлиста;
- визуализация аудиопотока;
- перемешивание композиций;
- функция радио.

Примеры реализации функционала приведены в соответствующем разделе.

2.4.2 Работа с аудиофайлами в модуле аудиоредактора

Для организации работы редактора аудиофайлов использовалась библиотека NAudio.dll.

NAudio – это аудио API с открытым исходным кодом для .NET, написанный на C# Марком Хитом при участии многих других разработчиков. Библиотека предоставляет большой набор полезных служебных классов для воспроизведения, обработки и конвертации аудиофайлов.

Основной функционал аудиоредактора заключается в:

- конвертации форматов аудиофайлов;
- наложения различных эффектов на аудиофайл (эхо и др.);
- обрезки и изменения продолжительности аудиофайлов;
- склейки нескольких аудиофайлов.

Библиотека NAudio в основном работает с форматами WAV, AIFF, MP3 и WMA, которых вполне хватает для выполнения заявленных операций с музыкальными файлами. Другие форматы могут быть подключены и поддерживаются через Windows Media Foundation.

Существует два типа качества цифрового звука:

- lossless – без потерь: FLAC, ALAC, WAV;
- lossy – с потерями: MP3, AAC, OGG, WMA.

И lossless, и lossy представляют из себя мощные архиваторы, которые позволяют значительно уменьшить размер музыкального файла, однако в случае с lossless (например, WAV) это сжатие будет идти без потерь в качестве, а у lossy (например, MP3) – с потерями, но при этом сам трек по размерам будет значительно меньше.

Поэтому организовывая работу аудиоредактора, будет логично опираться на различия между этими двумя видами архивации и работать с форматами каждого вида.

Разница работы lossless и lossy представлена на рисунке 19.

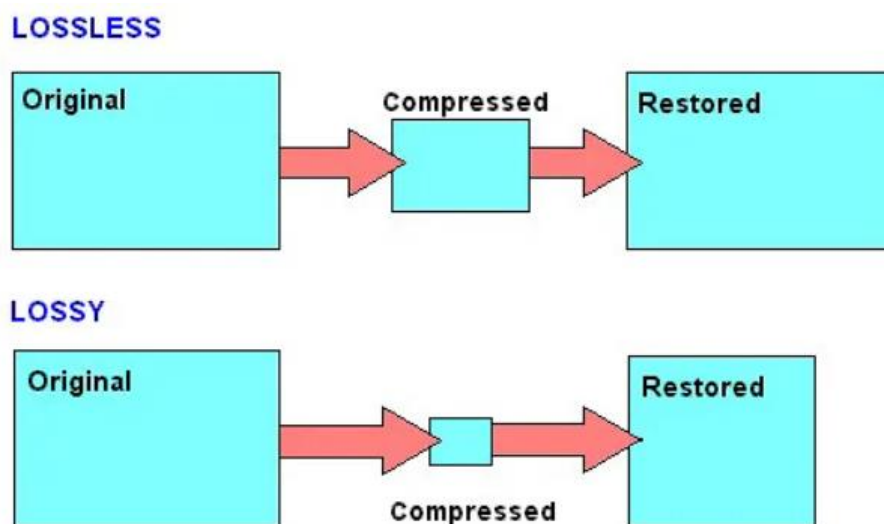


Рисунок 19 – Работа архиваторов

Учитывать особенности форматов необходимо, так как между ними имеются существенные различия.

Например, одним из недостатков MP3 является то, что файлы в этом формате не подходят для заикливания, так как в начале и в конце файла всегда будет небольшой тихий промежуток. Это связано с тем, что алгоритм сжатия MP3 оставляет паузу в начале и в конце файла от 10 мс до 50 мс.

Процедура обрезки, предоставляемая NAudio, осуществима только с аудиофайлами WAV формата PCM.

PCM (Linear PCM/LPCM) расшифровывается как импульсно-кодовая модуляция (pulse code modulation) и обеспечивает цифровое представление аналогового сигнала, который дискретизируется (оцифровывается) через равные промежутки времени (с заданной в герцах частотой) и представляется в двоичном виде (с заданной точностью – разрядностью в битах) [7].

Исходя из этих пунктов, было принято решение при работе с аудиоредактором сделать упор на самый распространённый lossy формат – MP3 и один из самых ходовых и удобных в обработке lossless форматов – формат WAV.

2.5 Проектирование пользовательского интерфейса

Пользовательский интерфейс является связующим звеном между пользователем и функционалом приложения.

Проектирование пользовательского интерфейса происходит на основании определенных ранее функциональных требований.

В интерфейсе аудиопроигрывателя должны присутствовать следующие элементы:

- кнопка добавления аудиофайлов, открывающая ФС ПК;
- плейлист, в который помещаются добавленные аудиофайлы;
- кнопки управления воспроизведением, переключением композиций, остановкой воспроизведения;
- слайдер для отображения текущей позиции трека и перемотки трека;
- слайдер регулирования громкости;
- элементы дополнительного функционала (визуализация аудиопотока, кнопка беззвучный режим, кнопка включения потокового радио);
- кнопки закрытия приложения и перехода в другие окна.

В интерфейсе аудиоредактора также должны присутствовать все необходимые для комфортной работы контроллеры:

- кнопки, управляющие работой функций обрезки, склеивания и конвертации;
- элементы визуализации аудиопотока для комфортной работы;
- необходимые для задания и отображения параметров текстовые окна;
- кнопки перехода между окнами приложения.

Интерфейс планируется разрабатывать в яркой цветовой гамме с возможностью изменения цветовой палитры. Иконки для кнопок планируется

взять из открытых источников. План интерфейса разрабатываемой системы представлен на рисунке 20.

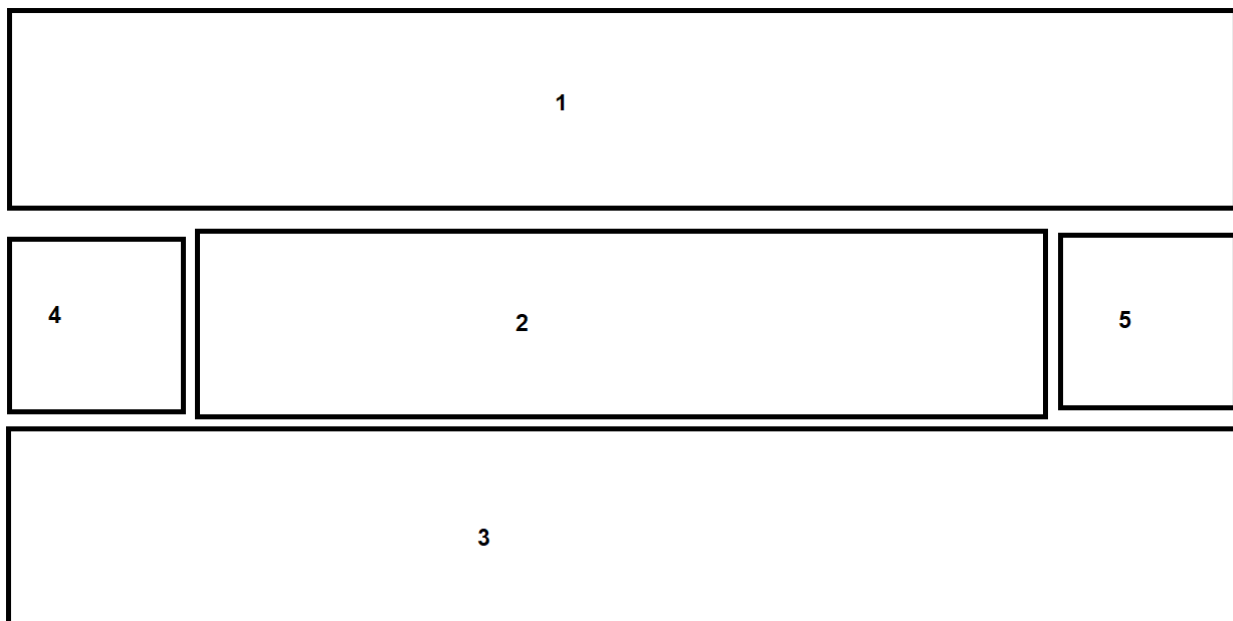


Рисунок 20 – План интерфейса

В область под номером 1 планируется поместить такие элементы, как: визуализатор аудиопотока, трек, название текущей композиции и её продолжительность.

В области под номером 2 будут находиться все основные контроллеры и кнопки.

В области под номером 3 планируется разместить плейлист.

В областях 4 и 5 будут находиться дополнительные контроллеры и кнопки.

2.6 Планирование разработки и оценка бюджета

Функционально-стоимостной анализ (ФСА, Activity Based Costing, ABC) – метод определения стоимости и других характеристик изделий, услуг и потребителей, в основе которого лежит использование функций и ресурсов, задействованных в производстве, маркетинге, продаже, доставке, технической поддержке, оказании услуг, обслуживании клиентов, а также в обеспечении качества [12].

Для определения стоимости разработки проекта можно взять такие моменты как: оплата труда программиста и стоимость электроэнергии. Также стоит учитывать, что сложность и объем разработки зависят от определенной среды и сроков выполнения работы.

Основные компоненты затрат показаны на рисунке 21.

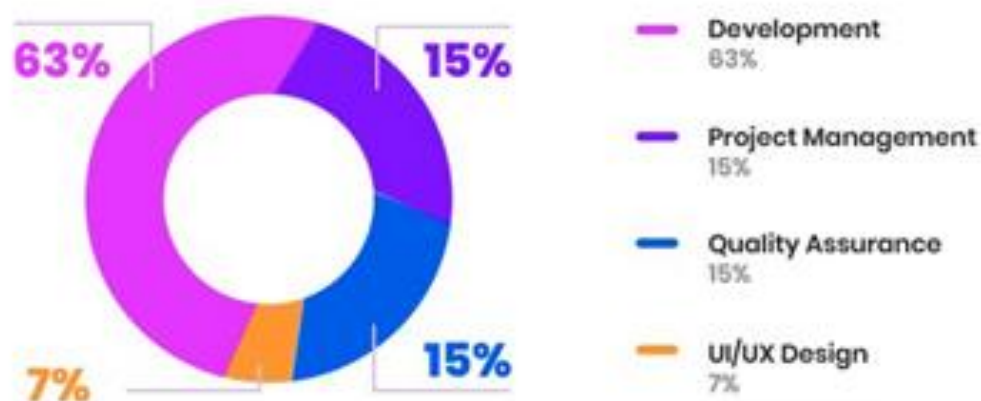


Рисунок 21 – Основные компоненты затрат

Затраты на разработку равны 63% от общего объема затрат. По 15% приходится на управление проектом и проверку качества продукта. И 7% затрат идет на проектирование и разработку интерфейсов.

Для определения стоимости выполнения работы была рассмотрена статистика заработной платы C#-программиста за 2020-2021 годы. Данные для статистики были взяты с сайта <http://russia.trud.com/>.

Медианная зарплата C#-разработчиков в целом (начиная от стажера и заканчивая лидом) – 95 000 руб. Если взять продолжительность рабочей недели в 40 часов, то в месяц выходит 160 часов работы. Это означает, что разработчик в среднем получает $95000/160=593$ руб/час.

Диаграмма заработных плат C#-разработчиков приведена на рисунке 22.

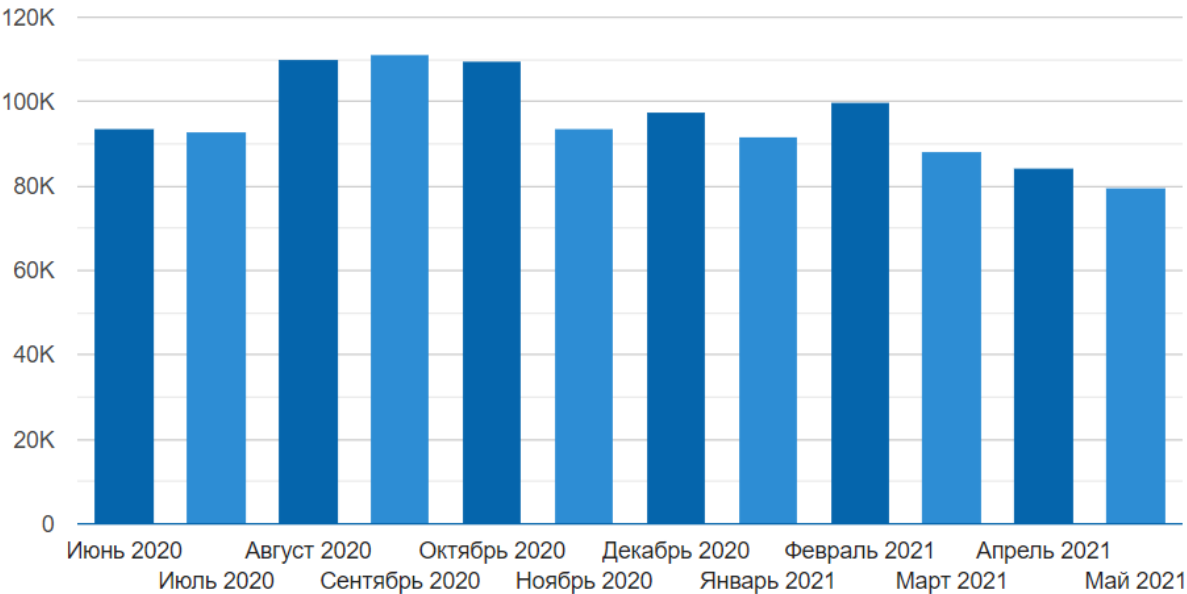


Рисунок 22 – статистика заработной платы C# программиста за 2020-2021 годы

Компьютер разработчика расходует приблизительно 0,6 кВт электроэнергии. Ставочный тариф составляет 3,63 рубля за кВт.

Выполнение выпускной работы состоит из пяти бизнес-процессов: «Анализ предметной области», «Анализ требований и проектирование», «Разработка», «Тестирование» и «Оформление ВКР».

Был проведен анализ затрат времени и количества дней выполнения на каждый этап для одного разработчика.

Диаграмма Ганта плана разработки представлена на рисунке 23.

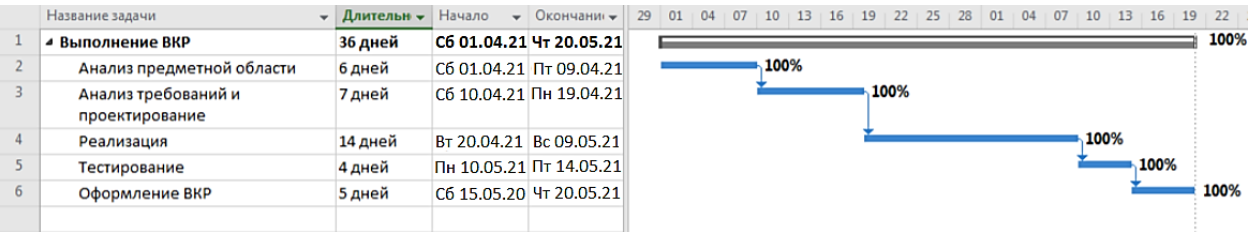


Рисунок 23 – Диаграмма Ганта

Учитывая все факторы, были сделаны вычисления, которые представлены в таблице 9.

Таблица 9 – Затраты на разработку

Этапы	Время на выполнение (часы)	Тариф (рублей в час)	Затраты на электроэнергию (рублей в час)	Всего
Анализ предметной области	10	593	2	5950
Анализ требований и проектирование	14	593	2	8330
Разработка	40	593	2	23800
Тестирование	6	593	2	3570
Оформление ВКР	8	593	2	4760
Всего	78			46410

Предположительная стоимость разработки продукта составляет 46410 рублей.

3 Реализация системы

3.1 Реализация программных средств

3.1.1 Архитектура проигрывателя

Для разработки приложения использовалась среда разработки Visual Studio 2017, язык программирования C#, средство разработки Windows Forms, библиотеки bass.dll и NAudio.dll.

Приложение состоит из двух основных модулей: плеера (Player.cs) и редактора (WavFileUtils.cs). А также из вспомогательных классов, с которыми они взаимодействуют.

То, как программные компоненты взаимодействуют в составе приложения, отображено на диаграмме компонентов на рисунке 24.

Описание диаграммы представлено в таблице 10.

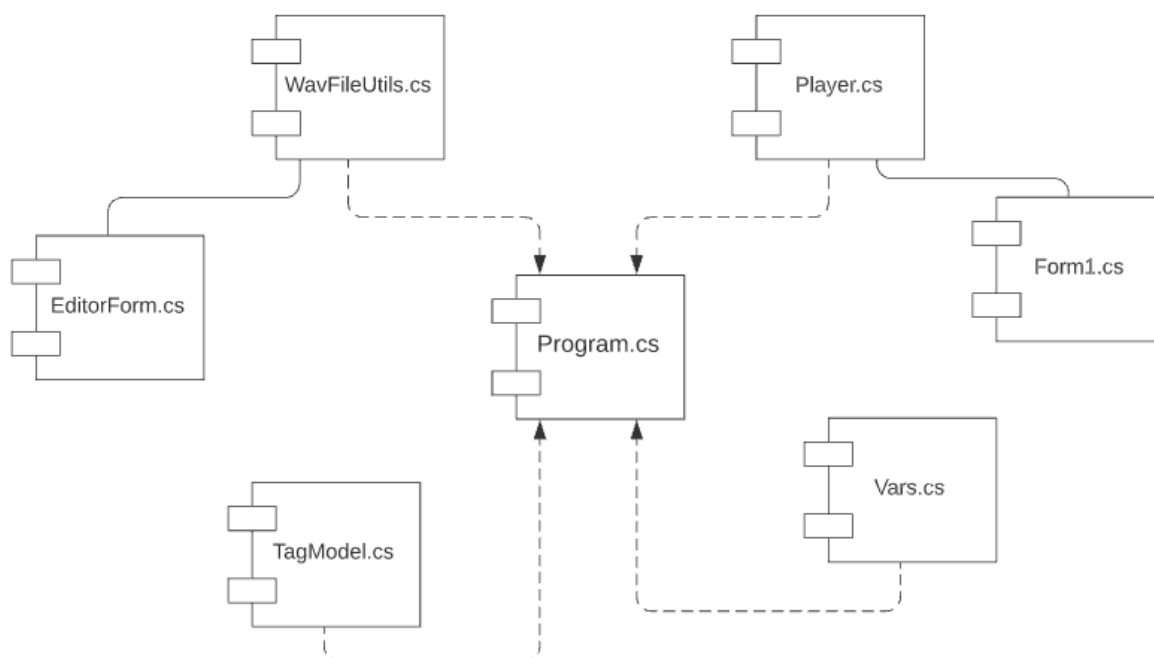


Рисунок 24 – Диаграмма компонентов

Таблица 10 – Описание диаграммы компонентов

Название компонента	Описание	Тип	Зависимости
Player.cs	Аудиоплеер	.cs	Form1.cs
PlayerForm1.cs	Форма плеера	.cs	-
WavFileUtils.cs	Аудиоредактор	.cs	EditorForm.cs
EditorForm.cs	Форма редактора	.cs	-
TagModel.cs	Работа с тегами	.cs	-
Vars.cs	Переменные	.cs	-
Program.cs	Точка входа в программу	.cs	Player.cs WavFileUtils.cs TagModel.cs Vars.cs

Классы Editor.cs и Form1.cs являются формами для отображения окон аудиоредактора и аудиоплеера соответственно. Во время работы приложения переход между окнами осуществляется нажатием специальной кнопки. Изначально открывается окно аудиоплеера, из которого можно перейти в окно редактора и обратно. При переходе, окно формы, из которой осуществляется переход, закрывается.

Класс Vars.cs предназначен для хранения переменных, с помощью которых осуществляется обработка пути до файла в плейлисте, текущая позиция файла в плейлисте и список имён всех файлов в плейлисте.

Класс TagModel.cs предназначен для работы с тегами музыкальных файлов.

Program.cs – точка входа в программу.

Иерархия классов, полей и методов представлена в виде диаграммы классов в разделе проектирование.

3.1.2 Реализация модуля аудиоплеера

Класс Player отвечает за воспроизведение музыкальных файлов, инициализацию аудиопотоков, изменение громкости проигрывания, перемотку трека и за работу потокового интернет радио.

При разработке класса использовалась использовалась библиотека `bass.dll`, которая отвечает за параметры звука в системе. Поля и методы класса представлены на рисунке 25.

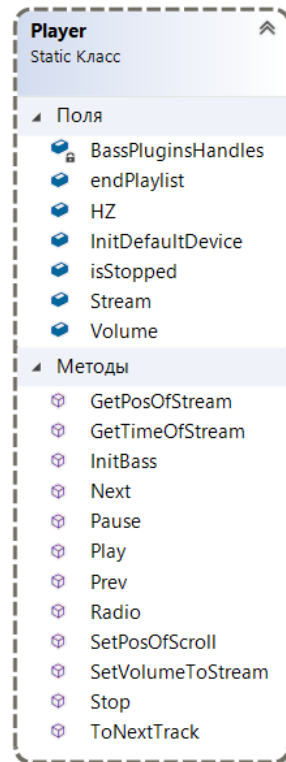


Рисунок 25 – Класс аудиоплеера

Рассмотрим поля данного класса:

- HZ – частота дискретизации;
- InitDefaultDevice – состояние инициализации библиотеки;
- Stream – хранит аудиопоток;
- Volume – хранит громкость;
- isStopped – состояние остановки канала;
- endPlaylis – состояние окончания плейлиста (верно, когда полностью проигран);
- BassPluginsHandles – инициализация `bass.dll` с определенной частотой дискретизации.

Рассмотрим методы данного класса:

- GetPosOfStream – получить текущую позицию трека;

- GetTimeOfStream – получить время трека в секундах;
- InitBass – проверка инициализации библиотеки, загрузка плагинов;
- Play – логика кнопки воспроизведения;
- Pause – логика кнопки пауза;
- Stop – логика кнопки остановки воспроизведения;
- Next – логика кнопки переключения на следующий трек;
- Prev – логика кнопки переключения на предыдущий трек;
- SetPosOfScroll – перемотка трека;
- SetVolumeToStream – изменение громкости;
- ToNextTrack – логика автопереключения треков.

Для детального рассмотрения решено взять некоторые методы класса, которые отвечают за инициализацию библиотеки и загрузку плагинов, и воспроизведение композиции.

Метод `InitBass(int hz)` проверяет, инициализирована ли библиотека, если нет, то инициализирует ее, затем загружает плагины музыкальных форматов и выдает сообщение, если не все из плагинов загрузились. Данный метод представлен в листинге 1.

Листинг 1 – Метод `InitBass(int hz)`

```
public static bool InitBass(int hz)
{
    if (!InitDefaultDevice)
    {
        InitDefaultDevice = Bass.BASS_Init(-1, hz,
BASSInit.BASS_DEVICE_DEFAULT, IntPtr.Zero);
        if (InitDefaultDevice)
        {
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_aac.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_ac3.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_ape.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_mpc.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_tta.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bassalac.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bassflac.dll"));
        }
    }
}
```

```

        BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bassopus.dll"));
        BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\basswma.dll"));
        BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\basswv.dll"));

        int ErrorCount = 0;
        for (int i = 0; i < BassPluginsHandles.Count; i++)
            if (BassPluginsHandles[i] == 0)
                ErrorCount++;
        if (ErrorCount != 0)
            MessageBox.Show("Плагинов не было загружено: " +
ErrorCount, " Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
        ErrorCount = 0;
    }

    }
    return InitDefaultDevice;
}

```

На рисунке 26 представлены плагины, которые лежат в специальной папке plugins в дереве проекта.

Метод загружает такие форматы, как: AAC, AC3, APE, MPC, TTA, ALAC, FLAC, OPUS, WMA.

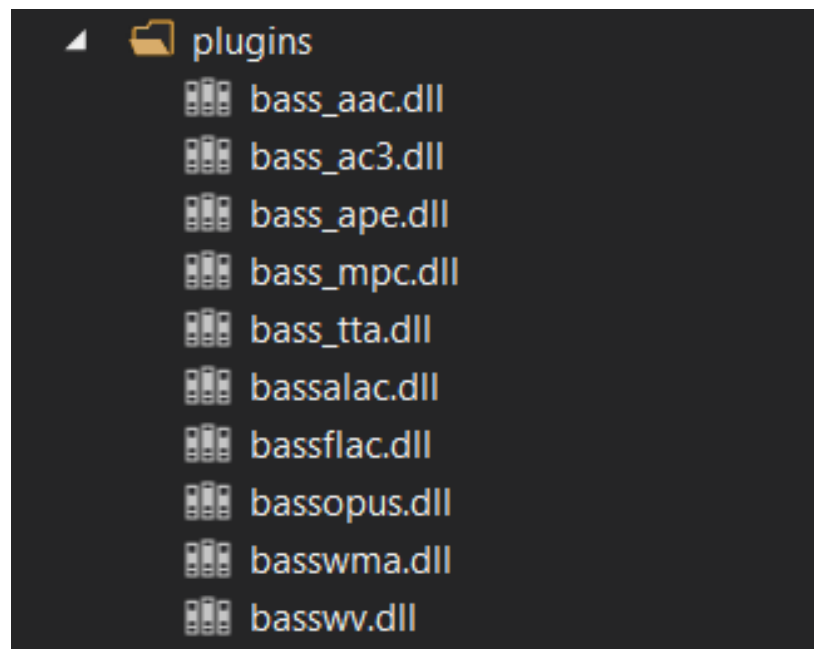


Рисунок 26 – Плагины музыкальных форматов

Метод Play служит для воспроизведения аудиопотока. Он принимает на вход файл для воспроизведения и уровень громкости воспроизведения. Метод останавливает игравший до этого поток, затем проверяет инициализирована

ли библиотека, присваивает потоку новый файл и устанавливает параметры громкости. Метод представлен в листинге 2.

Листинг 2 – Метод Play(string filename, int vol)

```
public static void Play (string filename, int vol)
{
    if (Bass.BASS_ChannelIsActive(Stream) !=
BASSActive.BASS_ACTIVE_PAUSED)
    {
        Stop();
        if (InitBass(HZ))
        {
            Stream =
Bass.BASS_StreamCreateFile(filename, 0, 0,
BASSFlag.BASS_DEFAULT);
            if (Stream != 0)
            {
                Volume = vol;
                Bass.BASS_ChannelSetAttribute(Stream,
BASSAttribute.BASS_ATTRIB_VOL, Volume / 100);
                Bass.BASS_ChannelPlay(Stream, false);
            }
        }
    }
    else Bass.BASS_ChannelPlay(Stream, false);
    isStopped = false;
}
```

Метод отвечает за работу кнопки воспроизведения.

3.1.3 Реализация модуля аудиоредактора

Модуль аудиоредактора разработан на основе библиотеки NAudio.dll и отвечает за редактирование аудиофайлов, конвертацию музыкальных форматов, обрезку и склейку аудиофайлов. Класс представлен на рисунке 27.

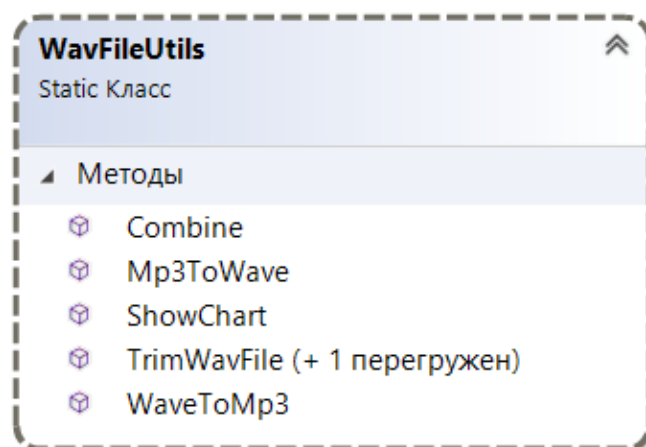


Рисунок 27 – Класс аудиоредактора

Рассмотрим методы данного класса:

- Combine – склейка нескольких композиций;
- Mp3ToWave – конвертация формата MP3 в WAV;
- WaveToMp3 – конвертация формата WAV в MP3;
- TrimWavFile – обрезка аудиофайла;
- ShowChart – отображение формы волны сигнала потока.

Для рассмотрения был взят метод Mp3ToWave, который используя методы библиотеки NAudio конвертирует формат MP3 в формат WAV. Метод представлен в листинге 3.

Листинг 3 – Метод Mp3ToWave()

```
public static void Mp3ToWave()
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "MP3 File (*.mp3)|*.mp3;";
    if (open.ShowDialog() != DialogResult.OK) return;

    SaveFileDialog save = new SaveFileDialog();
    save.Filter = "WAV File (*.wav)|*.wav;";
    if (save.ShowDialog() != DialogResult.OK) return;

    using (Mp3FileReader mp3 = new Mp3FileReader(open.FileName))
    {
        using (WaveStream pcm =
WaveFormatConversionStream.CreatePcmStream(mp3))
        {
            WaveFileWriter.CreateWaveFile(save.FileName, pcm);
        }
    }
}
```

Метод работает с файлами MP3 и WAV формата PCM.

Полный листинг системы представлен в приложении Б.

3.2 Реализация пользовательского интерфейса

Исходя из анализа требований, были реализованы интерфейсы окон аудиопроигрывателя и аудиоредактора. Иконки кнопок взяты из свободного источника v1.iconsearch.ru. Интерфейс аудиопроигрывателя представлен на рисунке 28. Описание компонентов интерфейса аудиопроигрывателя в таблице 11.

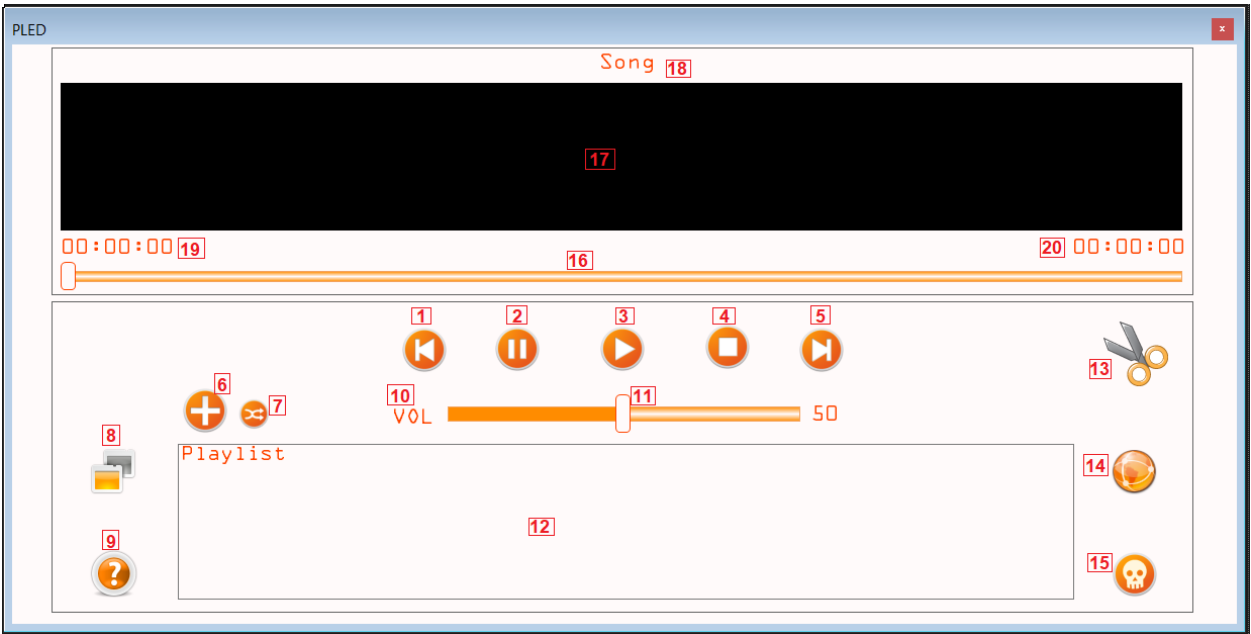


Рисунок 28 – Интерфейс аудиопроигрывателя

В квадратах с красной рамкой обозначены номера элементов интерфейса.

Таблица 11 – Описание компонентов интерфейса аудиопроигрывателя

Номер	Компонент	Назначение
1	Кнопка «Prev»	Переключение на предыдущий трек
2	Кнопка «Pause»	Приостановка проигрывания
3	Кнопка «Play»	Воспроизведение выбранной композиции
4	Кнопка «Stop»	Остановка воспроизведения текущей композиции
5	Кнопка «Next»	Переключение на следующий трек
6	Кнопка «FileDialog»	Добавление композиций в плейлист
7	Кнопка «Shuffle»	Перемешивание композиций в плейлисте
8	Кнопка «ChangeInterface»	Смена темы оформления
9	Кнопка «Info»	Получение справки и информации о продукте
10	Надпись «VOL»	Беззвучный режим
11	Слайдер «trackBarVol»	Изменение уровня громкости
12	Элемент «Playlist»	Отображение списка добавленных композиций
13	Кнопка «OpenEdit»	Переход в окно редактора
14	Кнопка «Radio»	Воспроизведение потокового интернет-радио
15	Кнопка «Exit»	Выход из приложения

Продолжение таблицы 11

Номер	Компонент	Назначение
16	Слайдер «trackBarTime»	Отображение текущего положения трека, перемотка трека
17	Элемент «wmp1»	Визуализация аудиопотока
18	Элемент «label3»	Отображение названия текущей композиции
19	Элемент «label1»	Отображение текущего времени композиции
20	Элемент «label2»	Отображение всей продолжительности композиции

Интерфейс аудиопроигрывателя представлен на рисунке 29.

Описание компонентов интерфейса аудиоредактора в таблице 12.

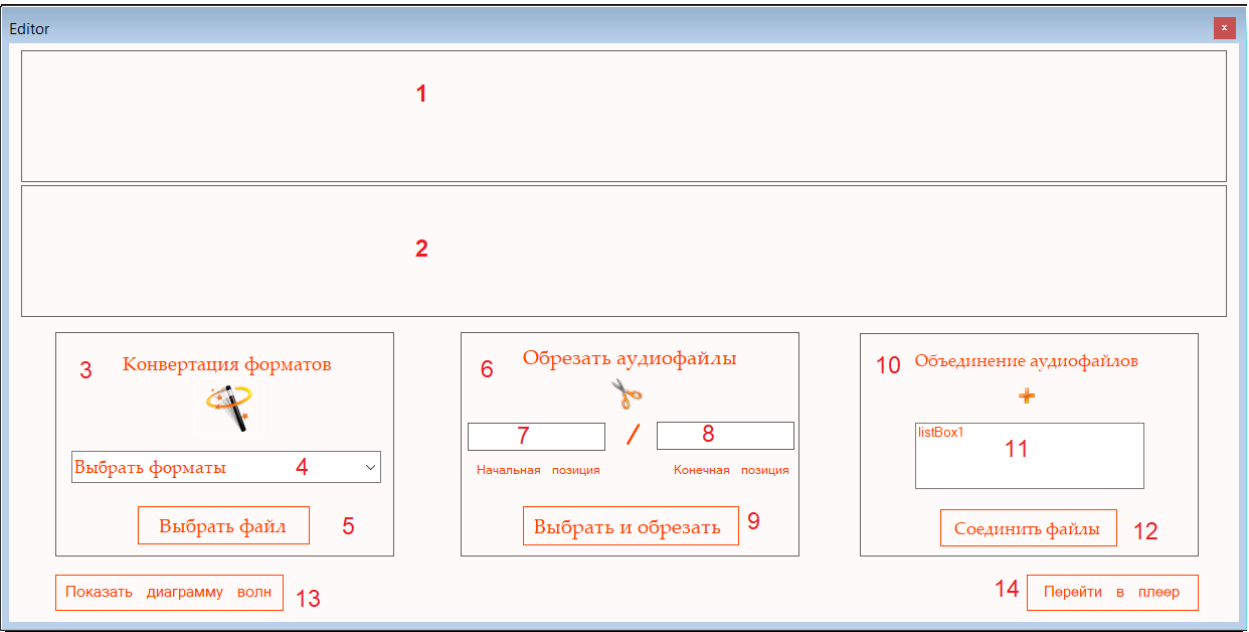


Рисунок 29 – Интерфейс аудиоредактора

Таблица 12 – Описание компонентов интерфейса аудиоредактора

Номер	Компонент	Назначение
1	Элемент «waveViewer1»	Отображение и визуализация аудиопотока
2	Элемент «waveViewer2»	Отображение и визуализация аудиопотока
3	Надпись «Конвертация форматов»	Группировка элементов, обозначение функционала
4	Элемент «Выбрать форматы»	Выбор конвертируемых форматов
5	Кнопка «Выбрать файл»	Выбор файлов для конвертации

Продолжение таблицы 12

Номер	Компонент	Назначение
6	Надпись «Обрезать аудиофайлы»	Группировка элементов, обозначение функционала
7	Элемент «textBoxTrim1»	Указание первого таймпойнта обрезки
8	Элемент «textBoxTrim2»	Указание второго таймпойнта обрезки
9	Кнопка «Выбрать и обрезать»	Выбор и обрезка файла по таймпоинтам
10	Надпись «Объединение аудиофайлов»	Группировка элементов, обозначение функционала
11	Элемент «listBox1»	Отображение названия склеиваемых композиций
12	Кнопка «Соединить файлы»	Склеивание выбранных композиций
13	Кнопка «Показать диаграмму»	Отображение и визуализация аудиопотока
14	Кнопка «Перейти в плеер»	Переход в аудиопроигрыватель

Графический интерфейс аудиопроигрывателя может изменяться в зависимости от выбранной темы оформления.

4 Контроль качества программного обеспечения

4.1 Выбор режима тестирования

Тестирование программного обеспечения – процесс исследования продукта с целью получения информации о его качестве и проверки соответствия заявленных к продукту требований и реально реализованной функциональности, осуществляемый путем наблюдения за его работой в искусственно созданных ситуациях и на ограниченном наборе тестов, выбранных определенным образом.

Выполнение тестирования включает в себя планирование тестирования, написание и выполнение тестов, анализ полученных результатов с последующим исправлением необходимых моментов.

Качество тестируемого программного обеспечения складывается из следующих составляющих:

- надежность;
- сопровождаемость;
- практичность;
- эффективность;
- мобильность;
- функциональность.

Тестирование, в зависимости от поставленных задач, можно условно разделить на следующие виды:

- функциональное;
- нефункциональное;
- связанное с изменениями.

Функциональное тестирование – один из ключевых видов тестирования, который предназначен для проверки соответствия конечного разработанного программного обеспечения изначальным требованиям к системе. Проведение функциональных тестов позволяет проверить работоспособность продукта и то, как он решает необходимые пользователю задачи.

При нефункциональном тестировании проводятся тесты для определения характеристик программного обеспечения. Нефункциональное тестирование не проверяет систему на соответствие требованиям заказчика, но проверяются такие свойства, как:

- работоспособность системы под различными нагрузками;
- адаптация приложения для различных платформ;
- требования к масштабируемости приложения.

Связанные с изменениями виды тестирования проводятся после исправления дефектов или багов для подтверждения того факта, что проблема была решена.

4.2 Функциональное тестирование

Составление плана для функционального тестирования производится по диаграмме вариантам использования и включает в себя описание тестов и их результаты.

План тестирования представлен в таблице 13. Результаты тестирования представлены в приложении А.

Таблица 13 – План тестирования

№ п/п	Назначение теста	Описание теста	Результаты теста
1	Тестирование кнопки «buttonPlay»	Нажатие кнопки, воспроизведение композиции	Рисунок А.1 (Успех)
2	Тестирование кнопки «buttonFileDialog»	Нажатие кнопки, добавление аудиофайла	Рисунок А.2 (Успех)
3	Тестирование кнопки «buttonStop»	Нажатие кнопки, остановка композиции	Рисунок А.3 (Успех)
4	Тестирование слайдера «trackBarTime»	Перетаскивание слайдера, перемотка композиции	Рисунок А.4 (Успех)
5	Тестирование слайдера «trackBarVol»	Перетаскивание слайдера, изменение громкости	Рисунок А.5 (Успех)

Продолжение таблицы 13

№ п/п	Назначение теста	Описание теста	Результаты теста
6	Тестирование кнопки «buttonOpenEdit»	Нажатие кнопки, переход в редактор	Рисунок А.6 (Успех)
7	Тестирование кнопки «buttonRadio»	Нажатие кнопки, прослушивание радио	Рисунок А.7 (Успех)
8	Тестирование элемента «wmp1»	Отображение формы волны аудиофайла	Рисунок А.8 (Успех)
9	Тестирование кнопки конвертации форматов «Choose files»	Нажатие кнопки, выбор композиции, конвертация формата	Рисунок А.9 Рисунок А.10 (Успех)
10	Тестирование метода «InitBass»	Загрузка плагинов, сообщение об ошибке	Рисунок А.11 Рисунок А.12 (Успех)

По результатам функционального тестирования можно утверждать, что продукт работает исправно.

4.3 Оценка качества программного обеспечения

Метрика программного обеспечения (англ. Software metric) – это некая мера позволяющая получить численное значение определенного свойства программного обеспечения или же его спецификаций. В Visual Studio есть встроенное средство для оценки кода проекта.

Таким образом, метрика будет отображать числовые значения определенных свойств программного обеспечения.

Индекс удобства обслуживания – вычисляет значение индекса от 0 до 100, представляющее относительную простоту обслуживания кода. Высокое значение означает лучшую сопровождаемость. Для быстрого обнаружения проблем в коде можно использовать цветовую маркировку. Зеленая Оценка находится в диапазоне от 20 до 100 и указывает на то, что код обладает хорошей сопровождаемостью. Желтая Оценка находится в диапазоне от 10 до

19 и указывает, что код является умеренно поддерживаемым. Красная Оценка – это оценка между 0 и 9 и указывает на низкую сопровождаемость.

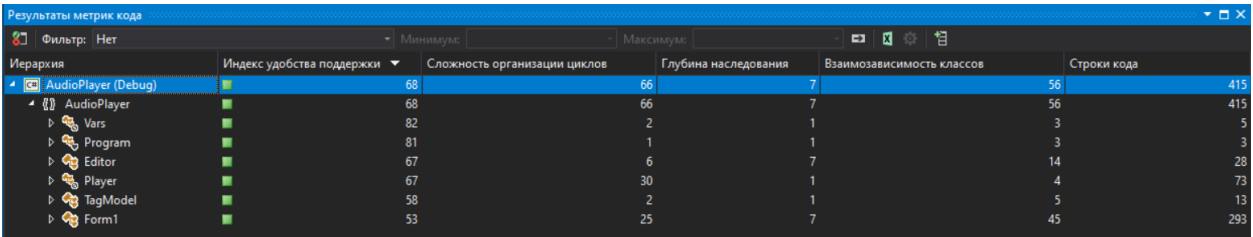
Сложностью организации циклов – измеряет структурную сложность кода. Он создается путем вычисления количества различных путей кода в потоке программы. Программа, имеющая сложный поток управления, требует больше тестов для достижения хорошего объема протестированного кода и менее сопровождаемой.

Глубина наследования – указывает количество различных классов, которые наследуют друг от друга, вплоть до базового класса. Глубина наследования аналогична взаимосвязанности классов в том, что изменение базового класса может повлиять на любые из его унаследованных классов. Чем выше это число, тем более глубокое наследование и тем выше вероятность внесения изменений в базовый класс, что приводит к критическому изменению. Для более глубокого наследования низкое значение хорошо, а высокое значение является недопустимым.

Взаимосвязь классов – измеряет связь с уникальными классами через параметры, локальные переменные, возвращаемые типы, вызовы методов, универсальные экземпляры или шаблоны шаблонов, базовые классы, реализации интерфейса, поля, определенные во внешних типах, и декорирование атрибутов. Хорошая разработка программного обеспечения определяет, что типы и методы должны иметь высокую связность и низкую связь. Высокая связь указывает на проект, который трудно использовать и поддерживать из-за множества взаимозависимостей от других типов [13].

Строки кода – показывает количество строк кода. Этот показатель отображает не точное количество строк. В расчет не берутся пустые строчки, комментарии, строчки со скобками, объявление типов и пространств имен.

На рисунке 1 представлены результаты метрик кода приложения.



Результаты метрик кода						
Фильтр: Нет		Минимум	Максимум			
Иерархия	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов	Строки кода	
AudioPlayer (Debug)	68	66	7	56	415	
AudioPlayer	68	66	7	56	415	
Vars	82	2	1	3	5	
Program	81	1	1	3	3	
Editor	67	6	7	14	28	
Player	67	30	1	4	73	
TagModel	58	2	1	5	13	
Form1	53	25	7	45	293	

Рисунок 30 – Результат метрик кода

Индекс удобства поддержки приложения равен 68, что является достаточно хорошим результатом. Это значит, что данное ПО будет удобно поддерживать и модифицировать в дальнейшем.

Заключение

В результате выполнения выпускной квалификационной работы были разработаны программные средства для воспроизведения и редактирования системы аудиофайлов.

На этапе анализа были рассмотрены аналогичные системы и определены функциональные и нефункциональные требования к системе.

На этапе проектирования были выбраны технологии проектирования и инструментальные средства для реализации системы, был разработан ряд UML-диаграмм и выполнено планирование проекта бюджета. Также были определены основные технологии работы с аудиофайлами в модулях аудиопроигрывателя и аудиоредактора.

На этапе реализации было создано приложение, написанное на языке программирования C# в среде разработки Visual Studio Professional 2017 с использованием технологии Windows Forms. Были подключены библиотеки для работы с аудиофайлами bass.dll и NAudio.dll.

Программные средства были протестированы с помощью функционального тестирования. По результатам измерения метрик кода, было установлено, что система может поддерживаться и совершенствоваться в дальнейшем.

Список использованных источников

1. Основные функции AIMP [Электронный ресурс]. – Режим доступа: <https://www.aimp.ru/?do=features> (дата обращения: 22.04.2021).
2. Возможности Winamp [Электронный ресурс]. – Режим доступа: <https://ru.winamp.com/features/> (дата обращения: 22.04.2021).
3. Image-Line FL Studio 20 Producer Edition звуковой редактор [Электронный ресурс]. – Режим доступа: <https://www.kombik.com/catalog/view/image-line-fl-studio-20-producer-edition/> (дата обращения: 23.04.2021).
4. WavePad Audio Editing Software [Электронный ресурс]. – Режим доступа: <https://www.nch.com.au/wavepad/index.html> (дата обращения: 23.04.2021).
5. UML-диаграммы [Электронный ресурс]. – Режим доступа: <https://prog-cpp.ru/uml-classes> (дата обращения: 17.05.2021).
6. Диаграмма состояний [Электронный ресурс]. – Режим доступа: http://it-gost.ru/articles/view_articles/97/ (дата обращения: 17.05.2021).
7. AVMARKET // PCM [Электронный ресурс]. – Режим доступа: <http://avmarket.com.ua/term/240-pcm/> (дата обращения: 18.05.2021).
8. Основы работы с CASE-средством AllFusion Process Modeler [Электронный ресурс]. – Режим доступа: <http://khpriip.mipk.kharkiv.edu/library/technpgm/labs> (дата обращения: 18.05.2021).
9. Microsoft Краткий обзор языка C# [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp> (дата обращения: 19.05.2021).
10. Microsoft Visual Studio [Электронный ресурс]. – Режим доступа: <https://www.microsoft.com/rus/business/smb/products-list/visualstudio2010> (дата обращения: 20.05.2021).
11. Турский В. Методология программирования. — М: Мир, 1981. — 264 с. — 30 000 экз.

12. Корпоративный менеджмент [Электронный ресурс]. – Режим доступа: https://www.cfin.ru/management/what_is_abc/ (дата обращения: 20.05.2021).
13. Microsoft Документация по Visual Studio [Электронный ресурс]. – Режим доступа: <https://docs.microsoft.com/ru-ru/visualstudio/code-quality/code-metrics-values?view=vs-2019> (дата обращения: 20.05.2021).
14. BASS.NET [Электронный ресурс]. – Режим доступа: <http://bass.radio42.com/> (дата обращения: 21.05.2021).
15. NAudio [Электронный ресурс]. – Режим доступа: <https://github.com/naudio/NAudio> (дата обращения: 21.05.2021).
16. Sound Code [Электронный ресурс]. – Режим доступа: <http://mark-dot-net.blogspot.com/2010/11/merging-mp3-files-with-naudio-in-c-and.html> (дата обращения: 21.05.2021).
17. OurCodeWorld [Электронный ресурс]. – Режим доступа: <https://ourcodeworld.com/articles/read/696/how-to-convert-a-mp3-file-to-wav-with-naudio-in-winforms-c-sharp> (дата обращения: 21.05.2021).
18. Хабр // Цифровой звук: DSD vs PSM [Электронный ресурс]. – Режим доступа: <https://habr.com/ru/post/275613/> (дата обращения: 22.05.2021).
19. Sound Code [Электронный ресурс]. – Режим доступа: <https://markheath.net/post/trimming-wav-file-using-naudio> (дата обращения: 23.05.2021).
20. SATURDAY MASTERING [Электронный ресурс]. – Режим доступа: <https://www.masteringonline.ru/articles/44100/> (дата обращения: 24.05.2021).

Приложение А
РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ
(рекомендуемое)

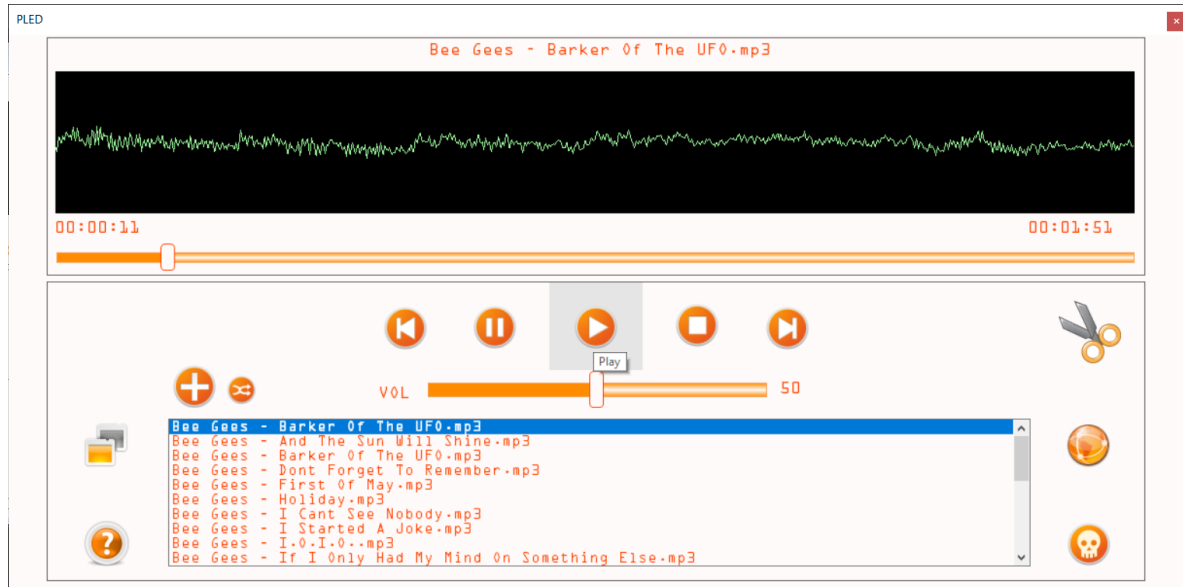


Рисунок А.1 – Тестирование кнопки «buttonPlay»

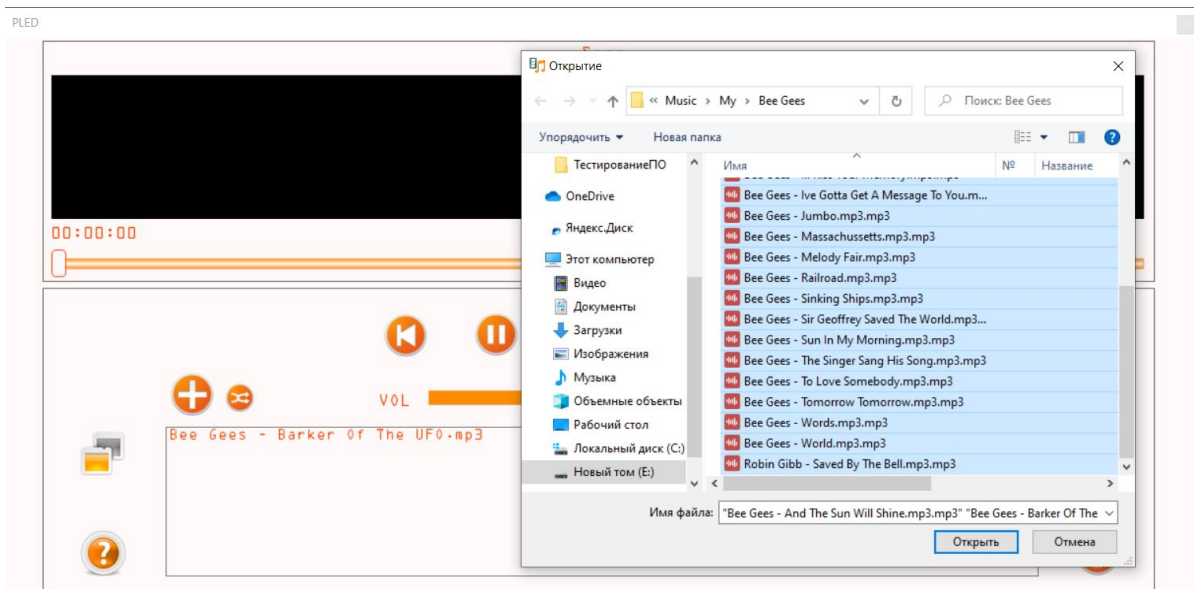


Рисунок А.2 – Тестирование кнопки «buttonFileDialog»



Рисунок А.3 – Тестирование кнопки «buttonStop»



Рисунок А.4 – Тестирование слайдера «trackBarTime»

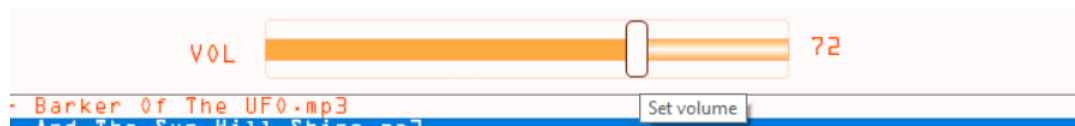


Рисунок А.5 – Тестирование слайдера «trackBarVol»

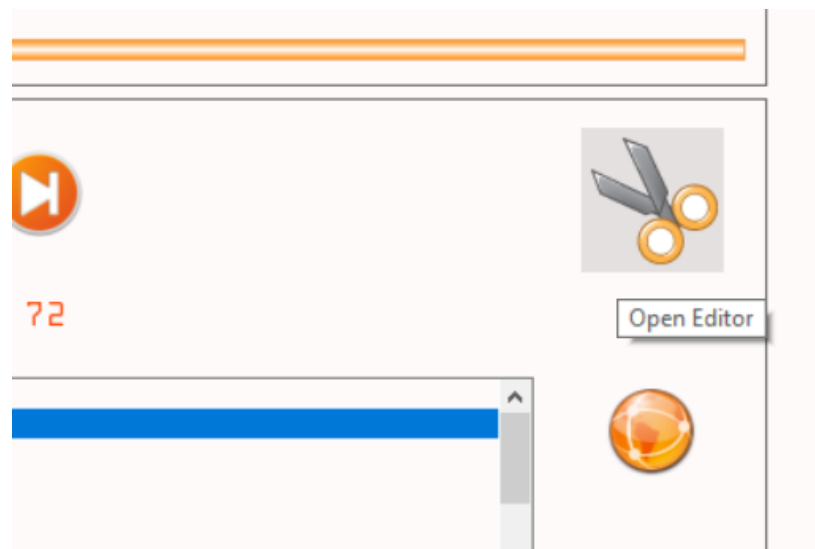


Рисунок А.6 – Тестирование кнопки «buttonOpenEdit»

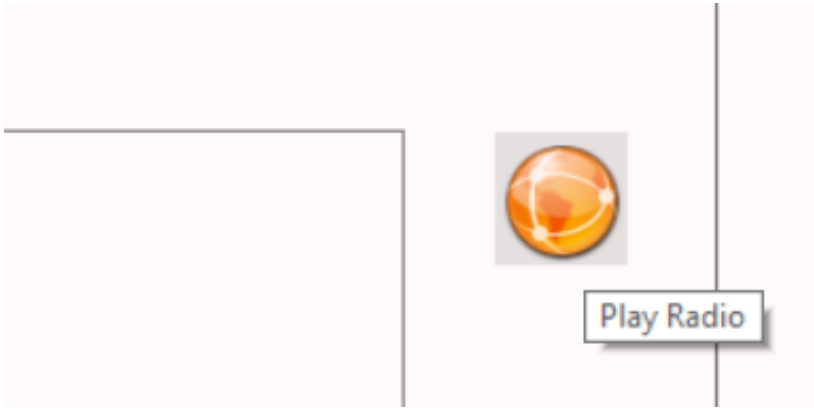


Рисунок А.7 – Тестирование кнопки «buttonRadio»

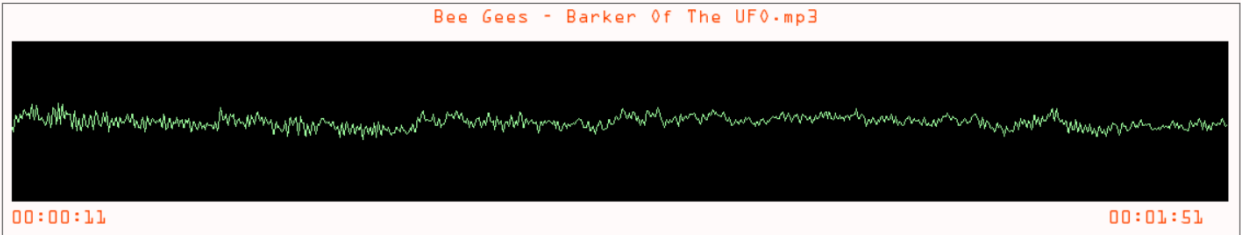


Рисунок А.8 – Тестирование элемента «wmp1»

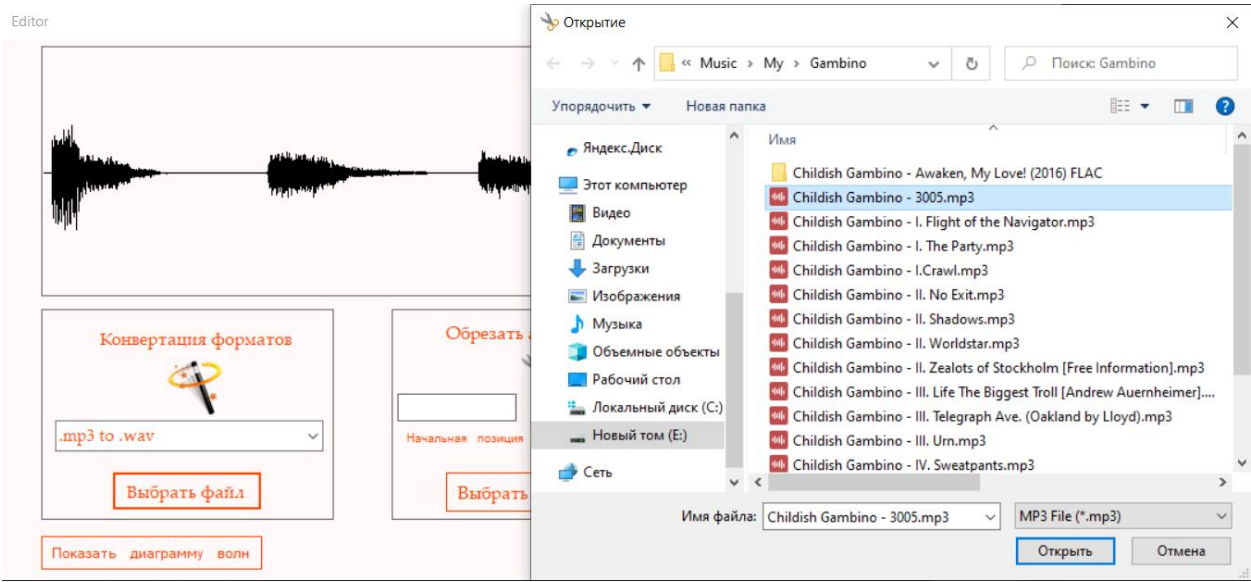


Рисунок А.9 – Тестирование кнопки конвертации форматов

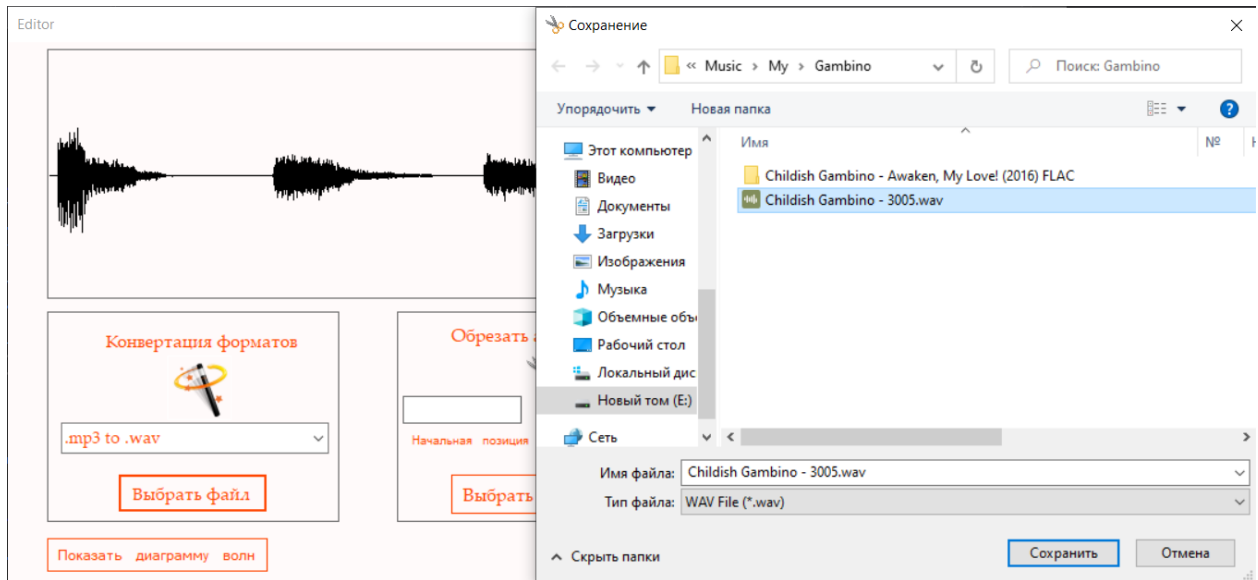


Рисунок А.10 – Тестирование кнопки конвертации форматов

```
InitDefaultDevice = Bass.BASS_Init(-1, hz, BASSInit.BASS_DEVICE_DEFAULT, IntPtr.Zero);
if (InitDefaultDevice)
{
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\ bass_aac.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\bass_ac3.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\ bass_ape.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\bass_mpc.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\b ass_tta.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\bassalac.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\bassflac.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\bassopus.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\basswma.dll"));
    BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath + @"plugins\bass wav.dll"));

    int ErrorCount = 0;
```

Рисунок А.11 – Тестирование загрузки плагинов (ошибка в коде 4-х)

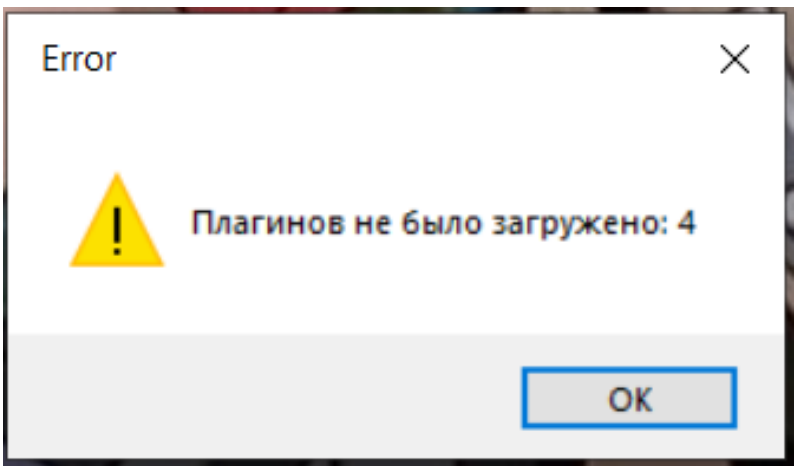


Рисунок А.12 – Тестирование сообщения об ошибке

Приложение Б
КОД ПРОГРАММЫ
(рекомендуемое)

```

public static bool InitBass(int hz)
{
    if (!InitDefaultDevice)
    {
        InitDefaultDevice = Bass.BASS_Init(-1, hz,
BASSInit.BASS_DEVICE_DEFAULT, IntPtr.Zero);
        if (InitDefaultDevice)
        {
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_aac.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_ac3.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_ape.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_mpc.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bass_tta.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bassalac.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bassflac.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\bassopus.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\basswma.dll"));
            BassPluginsHandles.Add(Bass.BASS_PluginLoad(Vars.AppPath
+ @"plugins\basswv.dll"));

            int ErrorCount = 0;
            for (int i = 0; i < BassPluginsHandles.Count; i++)
                if (BassPluginsHandles[i] == 0)
                    ErrorCount++;
            if (ErrorCount != 0)
                MessageBox.Show("Плаггинов не было загружено: " +
ErrorCount, " Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            ErrorCount = 0;
        }
    }
    return InitDefaultDevice;
}

/// <summary>
/// Логика кнопки "Плей"
/// </summary>
/// <param name="filename"></param>
/// <param name="vol"></param>
public static void Play(string filename, int vol)
{
    if (Bass.BASS_ChannelIsActive(Stream) !=
BASSActive.BASS_ACTIVE_PAUSED)
    {
        Stop();//Остановили канал, если он играл
        if (InitBass(HZ))
        {
            Stream = Bass.BASS_StreamCreateFile(filename, 0, 0,
BASSFlag.BASS_DEFAULT);
            if (Stream != 0)
            {
                Volume = vol;
                Bass.BASS_ChannelSetAttribute(Stream,
BASSAttribute.BASS_ATTRIB_VOL, Volume / 100);
                Bass.BASS_ChannelPlay(Stream, false);
            }
        }
    }
}

```

```

        }
    }
    else Bass.BASS_ChannelPlay(Stream, false);
    isStopped = false;
}
/// <summary>
/// Логика кнопки "Стоп"
/// </summary>
public static void Stop()
{
    Bass.BASS_ChannelStop(Stream);
    Bass.BASS_StreamFree(Stream);
    isStopped = true;
}
/// <summary>
/// Логика кнопки "Пауза"
/// </summary>
public static void Pause()
{
    if (Bass.BASS_ChannelIsActive(Stream) ==
BASSActive.BASS_ACTIVE_PLAYING)
        Bass.BASS_ChannelPause(Stream);
}
/// <summary>
/// Получить длину трека в секундах
/// </summary>
/// <param name="stream"></param>
/// <returns></returns>
public static int GetTimeOfStream(int stream)
{
    long TimeBytes = Bass.BASS_ChannelGetLength(stream);
    double Time = Bass.BASS_ChannelBytes2Seconds(stream, TimeBytes);
    return (int)Time;
}
/// <summary>
/// Получить текущую позицию в треке
/// </summary>
/// <param name="stream"></param>
/// <returns></returns>
public static int GetPosOfStream(int stream)
{
    long pos = Bass.BASS_ChannelGetPosition(stream);
    var posSec = (int)Bass.BASS_ChannelBytes2Seconds(stream, pos);
    return posSec;
}
/// <summary>
/// Перемотка трека
/// </summary>
/// <param name="stream"></param>
/// <param name="pos"></param>
public static void SetPosOfScroll(int stream, int pos)
{
    Bass.BASS_ChannelSetPosition(stream, (double)pos); //double ili
int?
}
/// <summary>
/// Изменение громкости
/// </summary>
/// <param name="stream"></param>
/// <param name="vol"></param>
public static void SetVolumeToStream(int stream, int vol)
{
    Volume = vol;
}

```

```

        Bass.BASS_ChannelSetAttribute(stream,
BASSAttribute.BASS_ATTRIB_VOL, Volume / 100F);
    }
    /// <summary>
    /// Логика автопереключения треков
    /// </summary>
    /// <returns></returns>
    public static bool ToNextTrack()
    {
        if ((Bass.BASS_ChannelIsActive(Stream) ==
BASSActive.BASS_ACTIVE_STOPPED) && (!isStopped))
        {
            if (Vars.Files.Count > Vars.CurrentTrackNumber + 1)
            {

                Play(Vars.Files[++Vars.CurrentTrackNumber], Volume); //++

ubral
                endPlaylist = false;
                return true;

            }
            else endPlaylist = true;
        }
        return false;
    }
    /// <summary>
    /// Логика кнопки "Следующий трек"
    /// </summary>
    public static void Next()
    {
        if ((Bass.BASS_ChannelIsActive(Stream) ==
BASSActive.BASS_ACTIVE_PLAYING) && (!isStopped))
        {
            if (Vars.Files.Count > Vars.CurrentTrackNumber + 1)
            {

                Play(Vars.Files[++Vars.CurrentTrackNumber], 100);
                endPlaylist = false;
            }
            else endPlaylist = true;
        }
    }
    /// <summary>
    /// Логика кнопки "Предыдущий трек"
    /// </summary>
    public static void Prev()
    {
        if ((Bass.BASS_ChannelIsActive(Stream) ==
BASSActive.BASS_ACTIVE_PLAYING) && (!isStopped))
        {
            if ((Vars.CurrentTrackNumber - 1) >= 0)
            {

                Play(Vars.Files[--Vars.CurrentTrackNumber], 100);
                endPlaylist = false;
            }
            else
            {
                Stop();
            }
        }
    }
    /// <summary>
    /// Логика кнопки "Radio"

```

```

    /// </summary>
    public static void Radio()
    {
        InitDefaultDevice = Bass.BASS_Init(-1, 44100,
        BASSInit.BASS_DEVICE_DEFAULT, IntPtr.Zero); //
        int stream =
        Bass.BASS_StreamCreateURL("http://air.radiorecord.ru:8102/tm_320", 0, 0,
        null, IntPtr.Zero); //
        Bass.BASS_ChannelPlay(stream, false);
    }
}

public static void Mp3ToWave()
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "MP3 File (*.mp3)|*.mp3;";
    if (open.ShowDialog() != DialogResult.OK) return;

    SaveFileDialog save = new SaveFileDialog();
    save.Filter = "WAV File (*.wav)|*.wav;";
    if (save.ShowDialog() != DialogResult.OK) return;

    using (Mp3FileReader mp3 = new Mp3FileReader(open.FileName))
    {
        using (WaveStream pcm =
        WaveFormatConversionStream.CreatePcmStream(mp3))
        {
            WaveFileWriter.CreateWaveFile(save.FileName, pcm);
        }
    }
}

public static void WaveToMp3()
{
    OpenFileDialog open = new OpenFileDialog();
    open.Filter = "WAV File (*.wav)|*.wav;";
    if (open.ShowDialog() != DialogResult.OK) return;

    SaveFileDialog save = new SaveFileDialog();
    save.Filter = "MP3 File (*.mp3)|*.mp3;";
    if (save.ShowDialog() != DialogResult.OK) return;

    using (WaveFileReader mp3 = new WaveFileReader(open.FileName))
    {
        using (WaveStream pcm =
        WaveFormatConversionStream.CreatePcmStream(mp3))
        {
            WaveFileWriter.CreateWaveFile(save.FileName, pcm);
        }
    }
}

public static void ShowChart()
{
}

public static void TrimWavFile(string inPath, string outPath,
    TimeSpan cutFromStart, TimeSpan cutFromEnd)
{
    using (WaveFileReader reader = new WaveFileReader(inPath))
    {

```

```

        using (WaveFileWriter writer = new WaveFileWriter(outPath,
reader.WaveFormat))
        {
            int bytesPerMillisecond =
reader.WaveFormat.AverageBytesPerSecond / 1000;

            int startPos = (int)cutFromStart.TotalMilliseconds *
bytesPerMillisecond;
            startPos = startPos - startPos %
reader.WaveFormat.BlockAlign;

            int endBytes = (int)cutFromEnd.TotalMilliseconds *
bytesPerMillisecond;
            endBytes = endBytes - endBytes %
reader.WaveFormat.BlockAlign;
            int endPos = (int)reader.Length - endBytes;

            TrimWavFile(reader, writer, startPos, endPos);
        }
    }

    public static void TrimWavFile(WaveFileReader reader, WaveFileWriter
writer, int startPos, int endPos)
    {
        reader.Position = startPos;
        byte[] buffer = new byte[1024];
        while (reader.Position < endPos)
        {
            int bytesRequired = (int)(endPos - reader.Position);
            if (bytesRequired > 0)
            {
                int bytesToRead = Math.Min(bytesRequired, buffer.Length);
                int bytesRead = reader.Read(buffer, 0, bytesToRead);
                if (bytesRead > 0)
                {
                    writer.Write(buffer, 0, bytesRead);
                }
            }
        }
    }

    public static void Combine(string[] inputFiles, Stream output)
    {
        foreach (string file in inputFiles)
        {
            Mp3FileReader reader = new Mp3FileReader(file);
            if ((output.Position == 0) && (reader.Id3v2Tag != null))
            {
                output.Write(reader.Id3v2Tag.RawData, 0,
reader.Id3v2Tag.RawData.Length);
            }
            Mp3Frame frame;
            while ((frame = reader.ReadNextFrame()) != null)
            {
                output.Write(frame.RawData, 0, frame.RawData.Length);
            }
        }
    }
}

public class TagModel

```

```

{
    public int BitRate;
    public int Freq;
    public string Channels;
    public string Artist;
    public string Album;
    public string Title;
    public string Year;

    Dictionary<int, string> ChannelsDict = new Dictionary<int, string>()
    {
        {0, "Null" },
        {1, "Mono" },
        {2, "Stereo" },
    };

    public TagModel(string file)
    {
        TAG_INFO tagInfo = new TAG_INFO();
        tagInfo = BassTags.BASS_TAG_GetFromFile(file);
        BitRate = tagInfo.bitrate;
        Freq = tagInfo.channelinfo.freq;
        Channels = ChannelsDict[tagInfo.channelinfo.chans];
        Artist = tagInfo.artist;
        Album = tagInfo.album;
        if (tagInfo.title == "")
            Title = Vars.GetFileName(file);
        else
            Title = tagInfo.title;
        Year = tagInfo.year;
    }
}

public static class Vars
{
    public static Form1 Link;
    //summary
    //Путь к исполняемому файлу
    //summary
    public static string AppPath = AppDomain.CurrentDomain.BaseDirectory;

    //Список полных имён файлов
    public static List<string> Files = new List<string>();
    //Текущая позиция трека в плейлисте
    public static int CurrentTrackNumber;
    //Получаем имя файла: парсим по / чтобы получить красивое имя, а не
    имя полного пути
    public static string GetFileName(string file)
    {
        string[] tmp = file.Split('\\');
        return tmp[tmp.Length - 1];
    }
}

```