**Содержание**

[Реферат 3](#_Toc484509804)

[Введение 4](#_Toc484509805)

[1.Постановка задачи 5](#_Toc484509806)

[1.1 Обзор и анализ прототипов 5](#_Toc484509807)

[1.2 Актуальность решаемой задачи 6](#_Toc484509808)

[1.3 Алгоритмы решения задачи 6](#_Toc484509809)

[2. Архитектура проекта 7](#_Toc484509810)

[2.1 Проектирование классов 7](#_Toc484509811)

[2.2 Проектирование базы данных 8](#_Toc484509812)

[3. Руководство пользователя 10](#_Toc484509813)

[4. Тестирование 16](#_Toc484509814)

[Заключение 18](#_Toc484509816)

[Список использованных источников 19](#_Toc484509817)

[Приложение А 20](#_Toc484509818)

[Приложение Б 21](#_Toc484509819)

# **Реферат**

Пояснительная записка курсового проекта содержит 20 страниц, 19 рисунков и 5 источников литературы.

Темой курсового проекта является разработка многофункционального плеера.

Пояснительная записка состоит из введения, 4 разделов и заключения.

Во введении представлена необходимость людей в использовании аудиоплееров.

В первом разделе представлена общая информация о проекте, актуальность и архитектура приложения.

Во втором разделе описана структура проекта и базы данных.

В третьем разделе представлено руководство для пользователя для данного приложения.

В четвёртом разделе представлен процесс тестирования.

# **Введение**

Аудиоплеер – это программа на ПК, позволяющая открывать и воспроизводить аудио файлы.

В своей повседневной жизни каждый человек сталкивается с необходимостью в прослушивании музыки. Люди любят слушать понравившиеся мелодии, занимаясь каким-либо делом. Так же многие книги выпускаются в звуковом формате. Для прослушивания музыки на компьютере необходимы специальные программы «Аудиоплееры».

Цель моего курсового проекта разработать многопользовательский аудиоплеер с визуализацией воспроизведения, с возможностью создавать и редактировать плейлисты для разных пользователей.

При использовании данного плеера, каждый пользователь компьютера сможет создать свою коллекцию музыкальных плейлистов различных форматов, о которой не будут знать другие пользователи. Также есть возможность оценивать композиции, что позволит сортировать музыку по различным параметрам.

# **1.Постановка задачи**

# **1.1 Обзор и анализ прототипов**

На сегодняшний день существует большое количество программ, предназначенных для прослушивания музыки и управления музыкальными файлами на компьютере. Одной из самых распространённых программ является AIMP.

В качестве примера рассмотрим плеер AIMP. Интерфейс данного приложения представлен на рисунке 1.

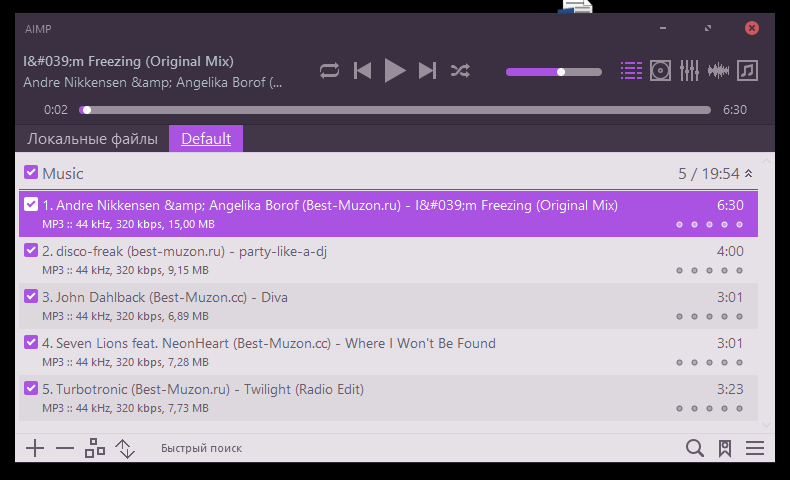


Рис. 1 – Интерфейс плеера AIMP

Проанализировав рисунок 1, а также прочие популярные плееры, видно их следующие сходства:

* плейлист для отображения композиций;
* кнопки для управления воспроизведением;
* слайдеры для перемотки воспроизведения и установки громкости;
* различная информация о композицияx;
* эквалайзер;
* визуализация воспроизведения.

## **1.2 Актуальность решаемой задачи**

В наше время почти каждый человек активно использует различные музыкальные плееры для прослушивания музыки, аудиокниг и т.д.

Таким образом данное приложения ещё многие годы будет актуальным, т.к. оно позволяет пользователям слушать аудиофайлы различных форматов, а также группировать имеющиеся аудиофайлы на своём компьютере.

## **1.3 Алгоритмы решения задачи**

При разработке плеера была использована библиотека Bass, одна из лучших библиотек в своём роде, с помощью которой написано много современных популярных плееров. Библиотека имеет множество возможностей для воспроизведения аудиофайлов, управлять воспроизведением, считывать теги с аудиофайла, работать с большим количеством форматов и многое другое.

Для организации работы с базой данных было решено использовать SQLite совместно с технологией ADO.NET.

Так же в плеере присутствует визуализация воспроизведения на основе фрактальной графики. Для её реализации был использован круговой фрактальный алгоритм: в окружность радиуса R вписывают семь окружностей радиуса R/3 таким образом, чтобы они все касались, но не пересекали друг друга. В каждую из этих семи окружностей вписываются по семь окружностей R/9 и т. д. Результат работы данного алгоритма представлен на рисунке 2. Листинг реализации алгоритма представлен в приложении А.

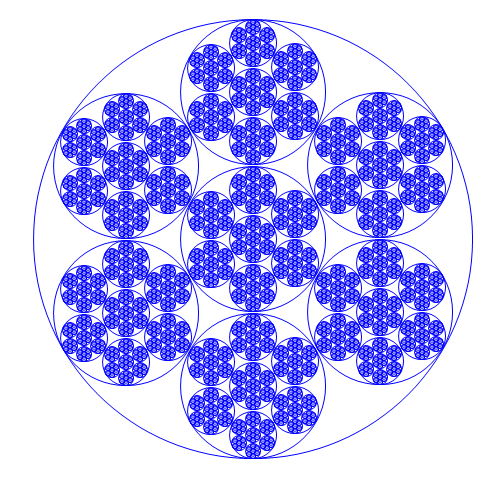


Рис. 2 – Круговой фрактал

# **2. Архитектура проекта**

## **2.1 Проектирование классов**

Диаграмма классов приложения представлена на рисунке 3.

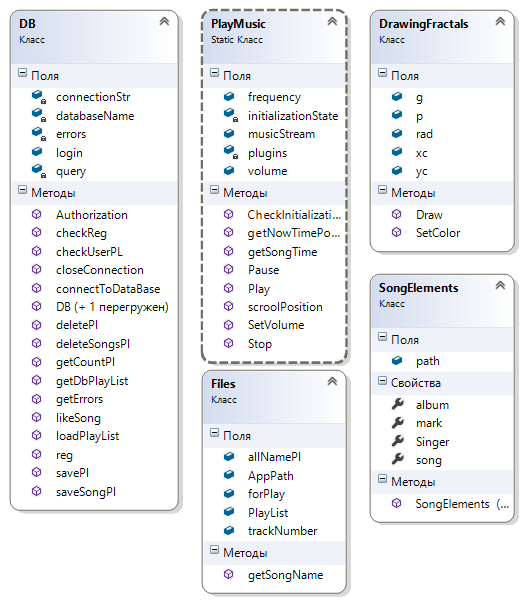


Рис. 3 – Диаграмма классов

Класс DB имеет методы, содержащие запросы для работы с базой данных.

Хранение информации о пользователях, их плейлисты, композиции.

Класс DrawingFractals имеет методы, для визуализации воспроизведения.

Статический класс PlayMusic содержит методы для управления воспроизведением, и поведением плеера. Листинг методов класса представлен в приложении Б.

Класс SongElements необходим для получения различной информации о композициях: исполнитель, альбом, название.

Класс Files содержит поля, которые предназначены для временного хранения композиций для воспроизведения, создания плейлистов.

Классы окон представлены на рисунке 6.

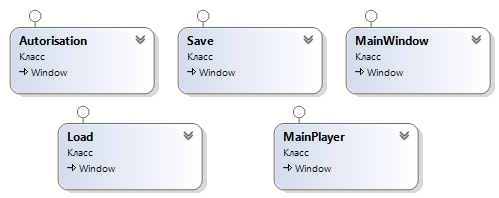


Рис. 6 – Классы окон

Класс MainWindow представляет собой начальное окно, необходимое для регистрации пользователя.

Класс Autorisation представляет собой окно авторизации.

Класс Save представляет собой окно для сохранения плейлиста для конкретного пользователя.

Класс Load представляет собой окно для получения, загрузки и удаления плейлистов.

Класс MainPlayer представляет собой главное окно плеера, на котором находится основное меню управления программой, кнопки для управления воспроизведением, анимация, плейлист.

## **2.2 Проектирование базы данных**

Диаграмма полученной базы данных представлена на рисунке 7.

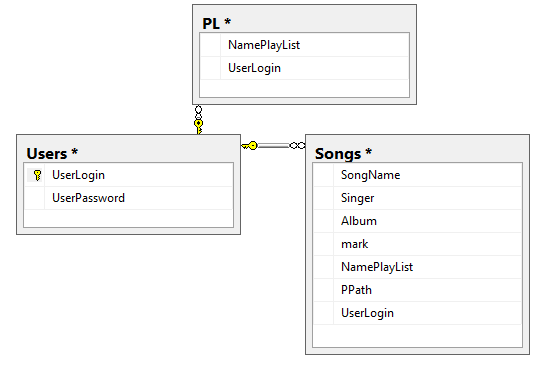


Рис. 7 – диаграмма базы данных

Всего в базе данных 3 таблицы. В таблице Users хранятся все пользователи, зарегистрированные в плеере, PL – созданные плейлисты для конкретных пользователей, а также таблица Songs для хранения аудиофайлов.

Таблица Users состоит из двух столбцов:

* UserLogin;
* UserPassword.

В столбце UserLogin хранится информация о логинах зарегистрированных пользователей, а в столбце UserPassword информация о их паролях.

Таблица PL состоит из двух столбцов:

* NamePlayList;
* UserLogin.

В столбце NamePlayList хранится информация о всех созданных плейлистах, а поле UserLogin указывает к какому пользователю плейлист относится.

Таблица Songs состоит из 7 столбцов:

* SongName;
* Singer;
* Album;
* Mark;
* NamePlayList;
* PPath;
* UserLogin.

В столбце SongName хранится информация о названиях аудиофайлов, Singer – информация о исполнителях, Album – информация о альбомах, Mark – информация о оценках, NamePlaylist указывает к какому плейлисту относится аудио файл, PPath хранит в себе файловые пути к аудиофайлам и UserLogin указывает какому пользователю принадлежит данный аудиофайл.

# 

# **3. Руководство пользователя**

После запуска приложения пользователь увидит окно регистрации. Оно представлено на рисунке 8.

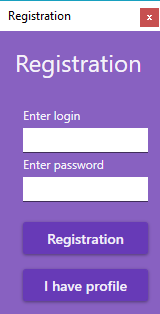


Рис. 8 – Окно регистрации

В случае, если пользователь уже имеет аккаунт, ему необходимо нажать кнопку I have profile, после чего он увидит окно авторизации, где пользователю будет необходимо ввести существующий логин и пароль. Окно авторизации представлено на рисунке 9.

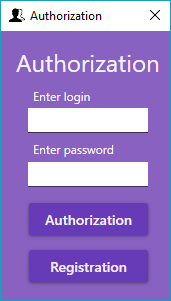


Рис. 9 – Окно авторизации

После авторизации пользователь увидит главное окно программы. Главное окно представлено на рисунке 10.

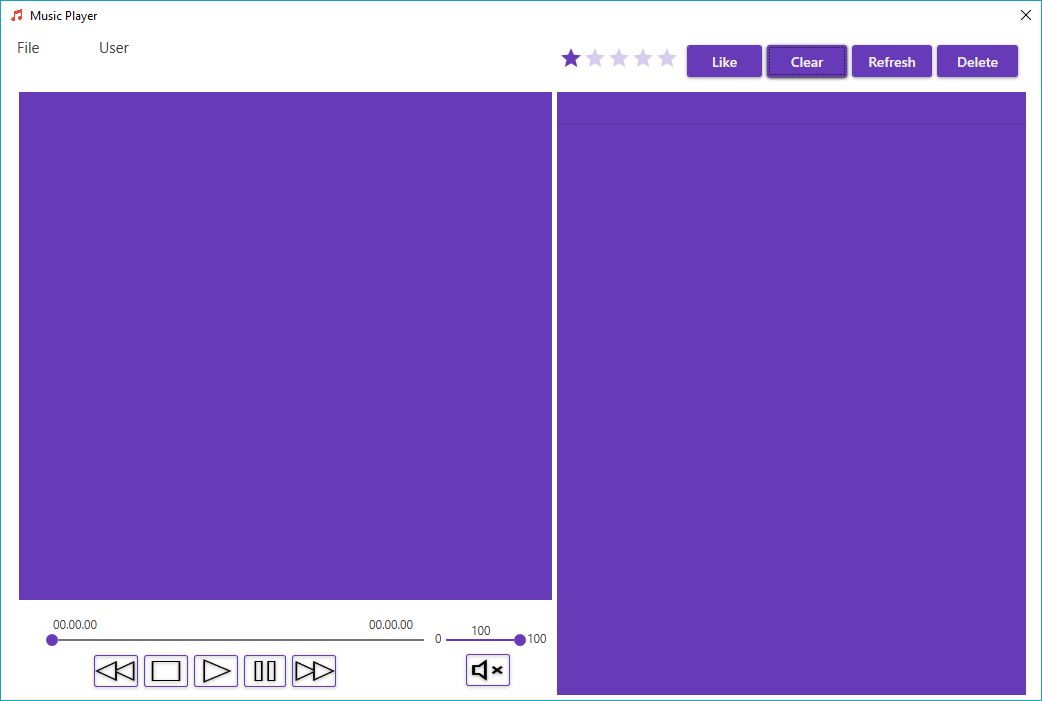


Рис. 10 – Главное окно программы

В правом верхнем углу располагается 4 кнопки: Like, Clear, Refresh, Delete и звёзды для оценивания композиций.

Для оценивания композиции необходимо выбрать её в плейлисте (ПКМ), выбрать количество звёзд и нажать кнопку Like. Результат операции представлен на рисунке 11.

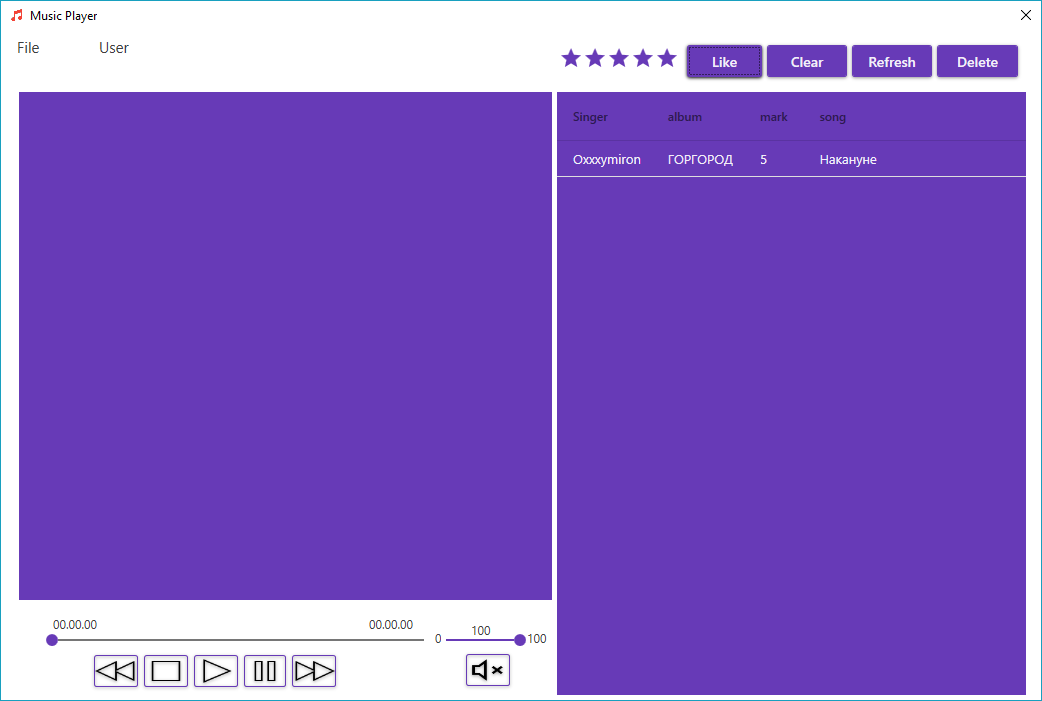


Рис. 11 – Оценка песни

Если нажать кнопку Delete, песня будет удалена из текущего плейлиста, а кнопка Clear удаляет весь текущий плейлист.

В левом нижнем углу находятся кнопки для управления воспроизведением: предыдущая песня, стоп, играть, пауза, следующая песня, режим без звука. Фон кнопки содержит картинку, которая полностью объясняет её назначение.

Здесь же и находятся 2 слайдера. Первый слайдер необходим для перемотки воспроизведения, а второй для установления громкости.

Плейлист находится в правой части окна и имеет следующие события: при клике левой кнопки мыши на строку плейлиста, произойдёт её воспроизведение, а при клике правой кнопкой мыши, обычное выделение

Окно плейлиста имеет 4 столбца: Singer, Album, Mark, Song для хранения информации о загруженных файлах. При нажатии левой кнопки мыши по каждому из них произойдёт сортировка плейлиста по соответствующему параметру.

Зона левее плейлиста отведена для анимации во время воспроизведения.

В левом верхнем меню располагается меню программы: File, User. Содержание этих пунктов представлено на рисунках 12 и 13.

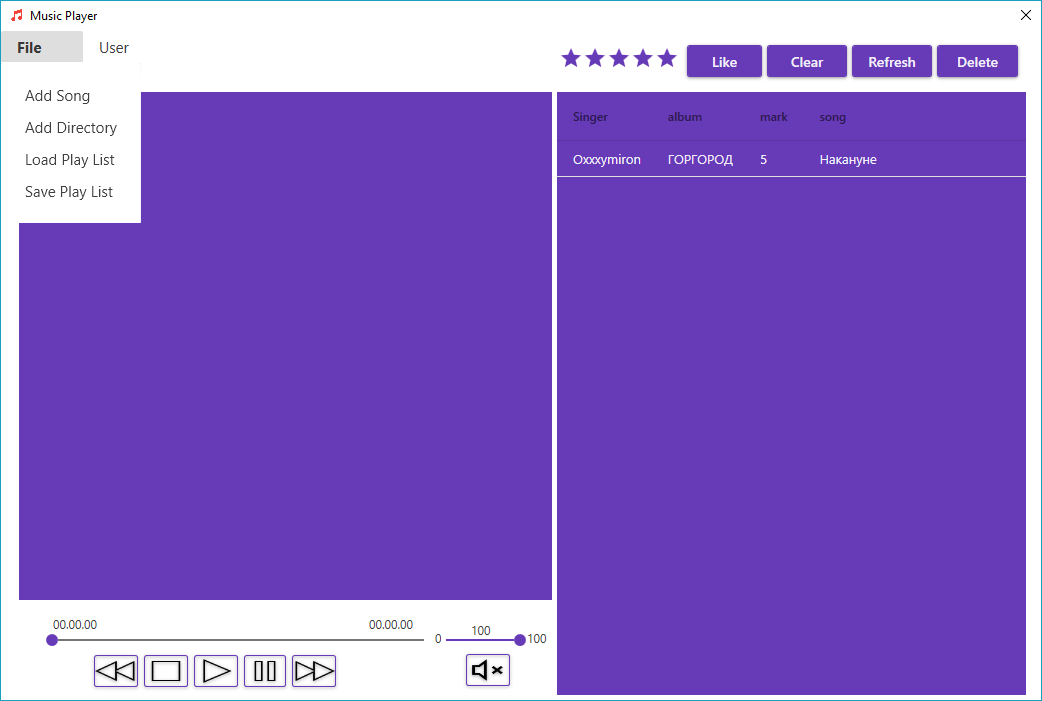


Рис.12 – Пункт меню File

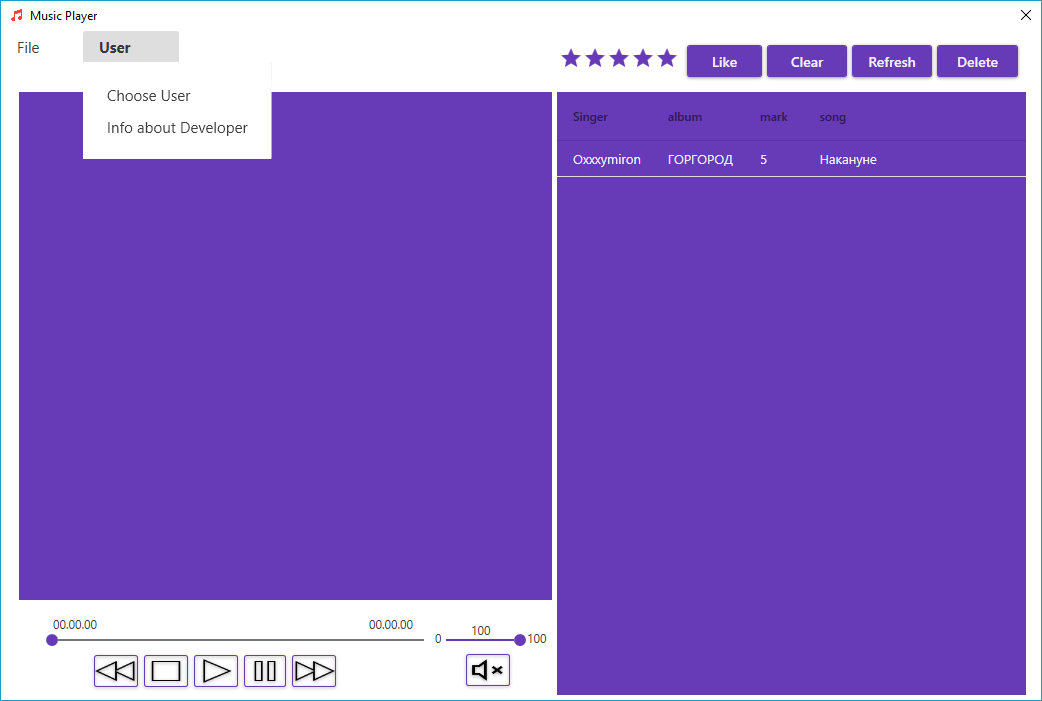


Рис. 13 – Пункт меню User

Рассмотрим содержимое File подробнее. При нажатии на пункт меню Add Song или Add directory откроется файловый диалог для выбора добавляемых файлов. Пример выполнения операции добавления продемонстрирован на рисунке 14 и 15.

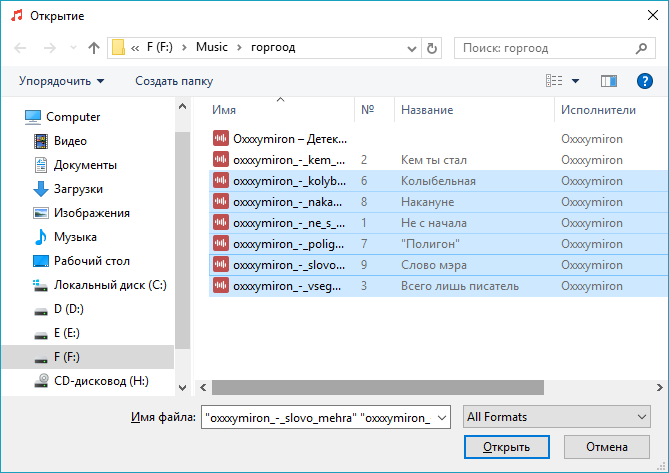


Рис.14 – Файловый диалог

При выделении нужных файлов и нажатии кнопки открыть произойдёт их добавление в плейлист основного окна.

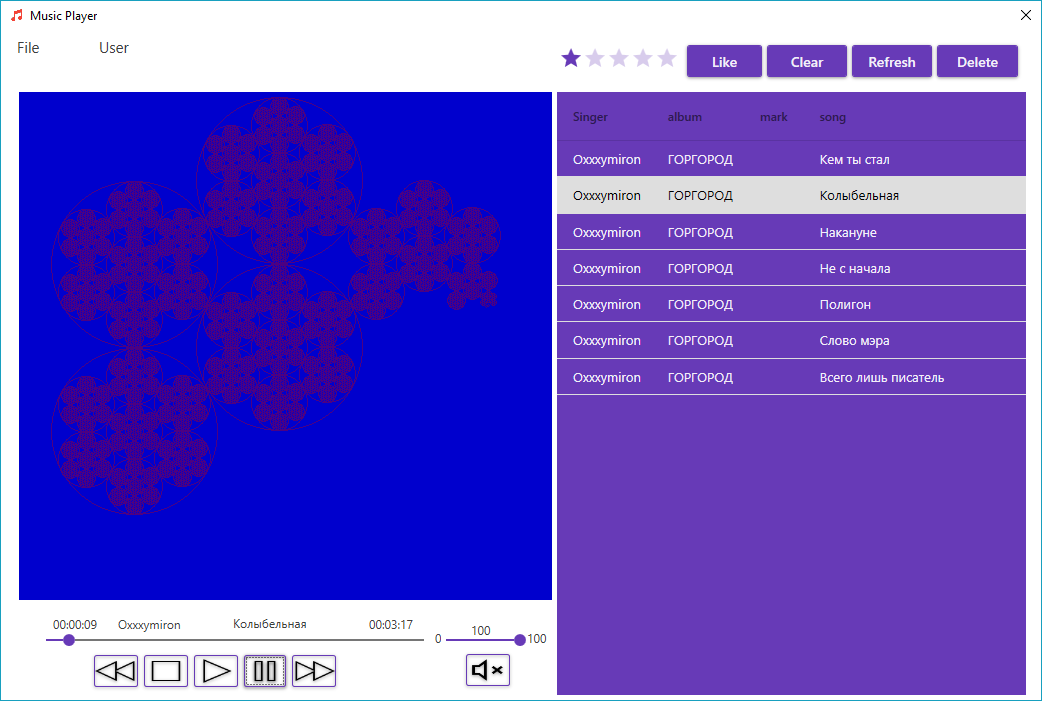


Рис.14 – Результат операции добавления

Единственная разница меду Add Song и Add Directory, что Add Directory позволяет добавить несколько файлов в отличии от Add Song.

Следующий пункт меню Load PlayList открывает окно с интерфейсом для загрузки и удаления сохранённых плейлистов. Окно Load PlayList представлено на рисунке 15.

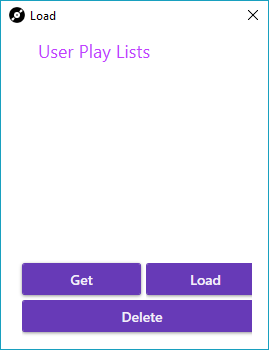


Рис.15 – Окно загрузки плейлистов

Окно имеет 3 кнопки: кнопка Get получает все сохранённые плейлисты текущего пользователя, Load загружает выбранный из списка плейлист, важно после выполнения операции загрузки нажать кнопку Refresh на основном окне плеера, т.е. обновить содержимое плейлиста. Кнопка Delete полностью удаляет сохранённый плейлист.

Следующий пункт меню Save PlayList. При выборе данного пункта откроется окно, которое продемонстрировано на рисунке 16.

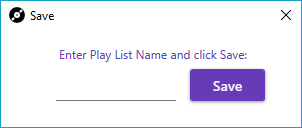


Рис.16 – Окно сохранения плейлистов

В текстовое поле ввода необходимо ввести названия плейлиста и нажать кнопку Save. После выполнения данной операции будет сохранён новый плейлист. Его содержимое – текущее содержимое плейлиста на главном окне.

В пункте меню User находится 2 подпункта Сhoose User. При выборе Сhoose User откроется окно авторизации для смены пользователя.

Также данное приложение поддерживает использование горячих клавиш для управления воспроизведением. При нажатии клавиши пробел произойдёт пауза текущего воспроизведения, при нажатии комбинации клавиш Shift + пробел произойдёт возобновление воспроизведения, с момента где была пауза. Клавиши Z и X отвечают за переключение песен: Z –предыдущая песня, X – следующая. При нажатии клавиши S произойдёт полная остановка воспроизведения.

# **4. Тестирование**

# При тестировании данного приложения были проведены следующие тесты на ситуации, которые могли бы привести к ошибке:

* Тесты авторизации и регистрации;
* Тест файлового диалога, т.е. исключить возможность добавлять файлы, которые не соответствуют аудио формату;
* Тест сохранения плейлиста;

В момент регистрации возможна ситуация, когда пользователь вводит уже существующий логин. Обработка данного исключения продемонстрирована на рисунке 17.

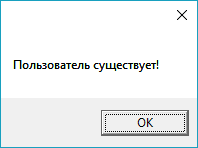


Рис.17– Тестирование регистрации

В момент авторизации возможна ситуация, когда пользователь вводит неверные данные пользователя, который ещё не зарегистрирован. Обработка данного исключения продемонстрирована на рисунке 18.

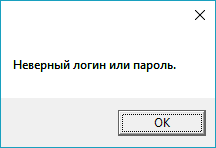


Рис.18– Тестирование авторизации

Для файлового диалога установлен фильтр, что полностью исключает возможность добавить файлы некорректного формата, т.к. пи использовании фильтра их просто не будет видно.

При сохранении плейлиста возможна ситуация, когда пользователь захочет сделать плейлист без имени, что крайне неудобно в дальнейшем использовании программы. Обработка данного исключения продемонстрирована на рисунке 19.

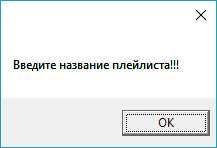


Рис.19– Тестирование сохранения

# **Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию приложения «Аудиоплеер». При разработке были выполнены все пункты из указанного списка предполагаемого основного функционала приложения. Таким образом, была достигнута цель и создан аудиоплеер.

В программном средстве были реализованы следующие функции:

* Воспроизведение 10 различных форматов аудиофайлов;
* Управление воспроизведением;
* Визуализация воспроизведения;
* Работа с несколькими пользователями;
* Система оценок и фильтры;
* Горячие клавиши для управления воспроизведением;

В соответствии с полученным результатом работы программы можно сделать вывод, что разработанная программа работает верно, а требования технического задания выполнены в полном объеме.

# **Список использованных источников**

1. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.
3. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/>. Дата доступа: 20.04.2017
4. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com>. Дата доступа: 21.05.2017
5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 13.04.2017

# **Приложение А Листинг алгоритма кругового фрактала**

public void Draw(int x, int y, int size, Graphics g, System.Drawing.Pen p)

{

int iter = 6, a = 3;

int r1, r2;

if (size > 1)

{

r1 = (int)Math.Round(size / (a \* 1.0));

r2 = (int)Math.Round(size \* (a - 1.0) / a);

Draw(x, y, r1, g, p);

for (int i = 1; i <= iter; i++)

Draw(x - (int)Math.Round(r2 \* Math.Sin(2 \* Math.PI / iter \* i)), y + (int)Math.Round(r2 \* Math.Cos(2 \* Math.PI / iter \* i)), r1, g, p);

}

try

{

g.DrawEllipse(p, x - size, y - size, 2 \* size, 2 \* size);

}

catch

{

}

}

# **Приложение Б Управление воспроизведением**

public static int frequency = 44100; // частота дискретизацции

private static bool initializationState; // состояние инициализации

public static int musicStream; // канал

public static int volume = 100; // громкость

private static readonly List<int> plugins = new List<int>();

// проверка инициализации

public static bool CheckInitializationState(int freq)

{

if (!initializationState)

{

initializationState = Bass.BASS\_Init(-1, frequency,BASSInit.BASS\_DEVICE\_DEFAULT,IntPtr.Zero);

if (initializationState)

{

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bass\_aac.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bass\_ac3.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bass\_ape.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bass\_mpc.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bass\_tta.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bassalac.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bassflac.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\bassopus.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\basswma.dll"));

plugins.Add(Bass.BASS\_PluginLoad(Files.AppPath + @"plugins\basswv.dll"));

}

int errors = 0;

for(int i = 0; i < plugins.Count; i ++)

{

if (plugins[i] == 0)

{

errors++;

}

}

if (errors !=0)

{

MessageBox.Show(errors + " плагина(ов) не было загружено");

errors = 0;

}

}

return initializationState;

}

// остановка воспроизведения

public static void Stop()

{

Bass.BASS\_ChannelStop(musicStream);

Bass.BASS\_StreamFree(musicStream);

}

public static void Pause()

{

if(Bass.BASS\_ChannelIsActive(musicStream) == BASSActive.BASS\_ACTIVE\_PLAYING)

{

Bass.BASS\_ChannelPause(musicStream);

}

}

// получить длительность аудиофайла

public static int getSongTime(int musicStream)

{

long timeBytes = Bass.BASS\_ChannelGetLength(musicStream);

double songTime = Bass.BASS\_ChannelBytes2Seconds(musicStream, timeBytes);

return (int)songTime;

}

//получить текущее время воспроизведения

public static int getNowTimePosition(int stream)

{

long position = Bass.BASS\_ChannelGetPosition(stream);

double nowSongTime = Bass.BASS\_ChannelBytes2Seconds(musicStream, position);

return (int) nowSongTime;

}

// перемотка воспроизведения

public static void scroolPosition(int stream, int position)

{

Bass.BASS\_ChannelSetPosition(stream, (double)position);

}

// установка громкости

public static void SetVolume(int stream, double vol)

{

volume = (int)vol;

Bass.BASS\_ChannelSetAttribute(musicStream,BASSAttribute.BASS\_ATTRIB\_VOL,volume/100F);

}

// воспроизвести

public static void Play(string fileName, int vol)

{

if (Bass.BASS\_ChannelIsActive(musicStream) != BASSActive.BASS\_ACTIVE\_PAUSED)

{

Stop();

if (CheckInitializationState(frequency))

{

musicStream = Bass.BASS\_StreamCreateFile(fileName, 0, 0, BASSFlag.BASS\_DEFAULT);

if (musicStream != 0)

{

volume = 100;

Bass.BASS\_ChannelSetAttribute(musicStream, BASSAttribute.BASS\_ATTRIB\_VOL, (float)volume / 100);

Bass.BASS\_ChannelPlay(musicStream, false);

}

}

}

else

{

Bass.BASS\_ChannelPlay(musicStream, false);

}

}