МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных Технологий

Кафедра Информационных систем и технологий

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

Специализация

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**К КУРСОВОМУ ПРОЕКТУ НА ТЕМУ:**

«Стриминговый сервис музыки» с применением полнотекстового поиска в БД

Выполнил студент Станкевич В.М.

(Ф.И.О.)

Руководитель проекта ассистент Колмаков М.В.

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

Заведующий кафедрой .т.н., доц. Смелов В.В.

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

Курсовой проект защищен с оценкой

Минск 2020

Реферат

Пояснительная записка курсового проекта содержит 31 страницу пояснительной записки, 24 иллюстрации, 5 источников литературы, 5 приложений.

.NET Framework, WPF, ORACLE DATABASE 12c, SQL DEVELOPER, ADO.NET

Цель курсового проекта – разработка программного средства «Cosmic». «Cosmic» это музыкальный стриминговый сервис, основной целью которого является облегчение поиска и прослушивания аудио. В настоящее время существует пару крупных аналогов, реализованных в качестве десктопных приложений. Большинство других аналогов таких сервисов представлено в виде Web-сайтов.

В первой главе проводится аналитический обзор схожих приложений по тематике курсового проекта.

Вторая глава посвящена процессу проектирования системы и описание технологий, использованных во время выполнения проекта.

В третьей главе описывается процесс разработки, принципы функционирования проекта.

В четвёртой главе описано тестирование, проверка работоспособности и анализ данных системы.

В заключении приведены результаты проделанной работы.

Содержание

[Реферат 2](#_Toc90576192)

[**Введение** 4](#_Toc90576193)

[**1 Аналитический обзор схожих приложений** 5](#_Toc90576194)

[**2 Проектирование и разработка базы данных** 7](#_Toc90576195)

[**2.1 Спецификация функциональных требований** 7](#_Toc90576196)

[**2.2 Разработка модели базы данных** 9](#_Toc90576197)

[**2.3 Пользователи** 10](#_Toc90576198)

[**2.4 Хранимые процедуры** 10](#_Toc90576199)

[**2.4.1 Выборка данных из таблиц** 10](#_Toc90576200)

[**2.4.2 Добавление пользователя и проверка данных при авторизации** 10](#_Toc90576201)

[**2.4.3 Удаление данных из таблиц** 12](#_Toc90576202)

[**2.4.4 Добавление данных в таблицы** 12](#_Toc90576203)

[**2.4.5 Изменение данных в таблицах** 12](#_Toc90576204)

[**2.4.6 Экспорт и импорт таблиц в формат json** 12](#_Toc90576205)

[**2.5** **Реализация технологий** 13](#_Toc90576206)

[**2.5.1 Реализация технологии мультимедийные типы данных в БД** 13](#_Toc90576207)

[**3 Проектирование и разработка программного средства** 16](#_Toc90576208)

[**4 Тестирование, проверка работоспособности и анализ полученных данных** 19](#_Toc90576209)

[**Заключение** 25](#_Toc90576210)

[**Список литературы** 26](#_Toc90576211)

[**Приложение А** 27](#_Toc90576212)

[**Приложение Б** 28](#_Toc90576213)

[**Приложение В** 29](#_Toc90576214)

[**Приложение Г** 30](#_Toc90576215)

[**Приложение Д** 31](#_Toc90576216)

# **Введение**

В данной записке приведено описание реляционной базы данных и desktop-приложения, разработанных в соответствии с заданием на курсовое проектирование по теме «Стриминговый сервис музыки» по дисциплине «Базы данных».

Приложение «Cosmic» предназначено для поиска и прослушивания аудиозаписей. Данная программа создана для решения всего спектра задач, связанного с прослушиванием музыки: поиск, прослушивание, скачивание музыки, создание плейлистов и управление ими. Исходя из этого, повышается легкость и скорость поиска нужной пользователю музыки. Программа интуитивно проста для пользователей, так как не требует квалифицируемых ИТ-ресурсов.

Задачей данной курсовой работы является разработка базы данных предназначенной для стримингового сервиса «Cosmic». Приложение будет обладать следующими функционалом:

* авторизация и регистрация пользователей;
* поддержка двух ролей: клиент и администратор;
* просмотр композиций в каталогах;
* создание плейлистов;
* шифрование данных клиента.

Система управления базами данных (СУБД) — совокупность программных и лингвистических средств общего или специального назначения, обеспечивающих управление созданием и использованием баз данных.

СУБД существует огромное множество: Oracle, MS SQL Server, Microsoft Access, MySql и так далее. В данной работе будет использовано решение MS SQL Server.

# **1 Аналитический обзор схожих приложений**

Немаловажным этапом в разработке программного продукта является аналитический обзор прототипов и литературных источников.

На сегодняшний день можно встретить достаточно большое количество Web решений, разработанных для прослушивания музыки, но довольно малое количество десктоп программ, имеющих интуитивно понятный интерфейс. Я рассмотрел несколько программ-аналогов. Также все программы разделяются на desktop-приложения и web-приложения.

Desktop-приложения для поиска и прослушивания музыки:

* «Spotify»

Web-приложения:

* «Spotify»
* «Яндекс Музыка»

«Spotify» - интернет-сервис потокового аудио (стриминговый), позволяющий легально прослушивать музыкальные композиции, аудиокниги и подкасты, не скачивая их на устройство. В одном окне: рекомендуемые станции, действия друзей, плейлисты пользователя, реклама новых альбомов.

Интерфейс «Spotify» представлен на рисунке 1.1.

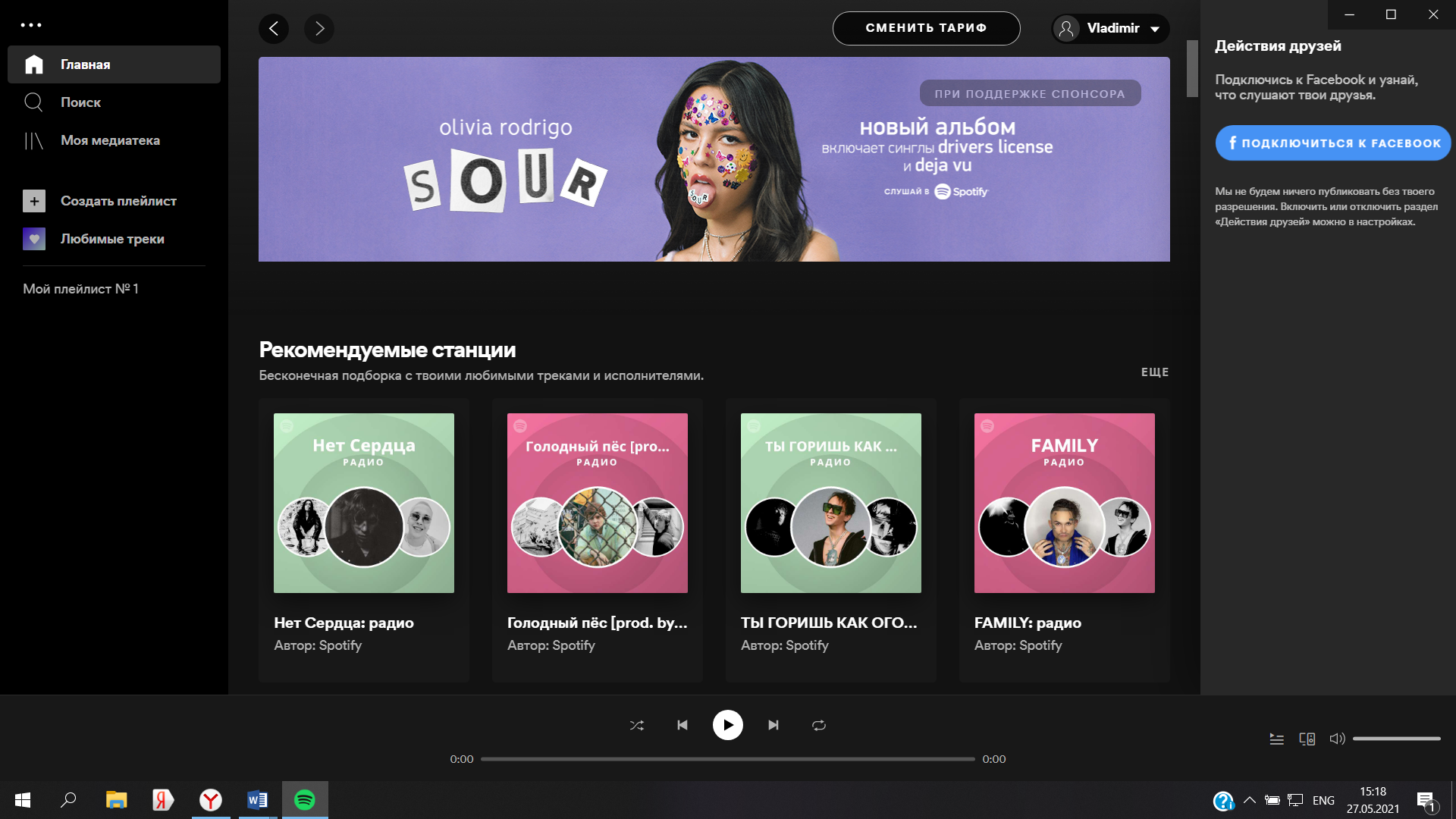


Рисунок 1.1 − Интерфейс «Spotify»

Проанализировав «Spotify», можно выделить основные минусы и плюсы данного программного средства.

Основные минусы:

* Отсутствие возможности слушать музыку без регистрации;
* Для скачивания музыки нужно покупать тариф.

Основные плюсы:

* Имеются «станции», которые подбирают музыку;
* Стильный дизайн;
* Приложение на телефон;
* Просмотр действий друзей.

Продукт «Яндекс Музыка» - стриминговый сервис компании «Яндекс», позволяющий прослушивать музыкальные композиции, альбомы, подборки музыкальных треков и получать персональные рекомендации.

Интерфейс «Яндекс Музыка» представлен на рисунке 1.2.

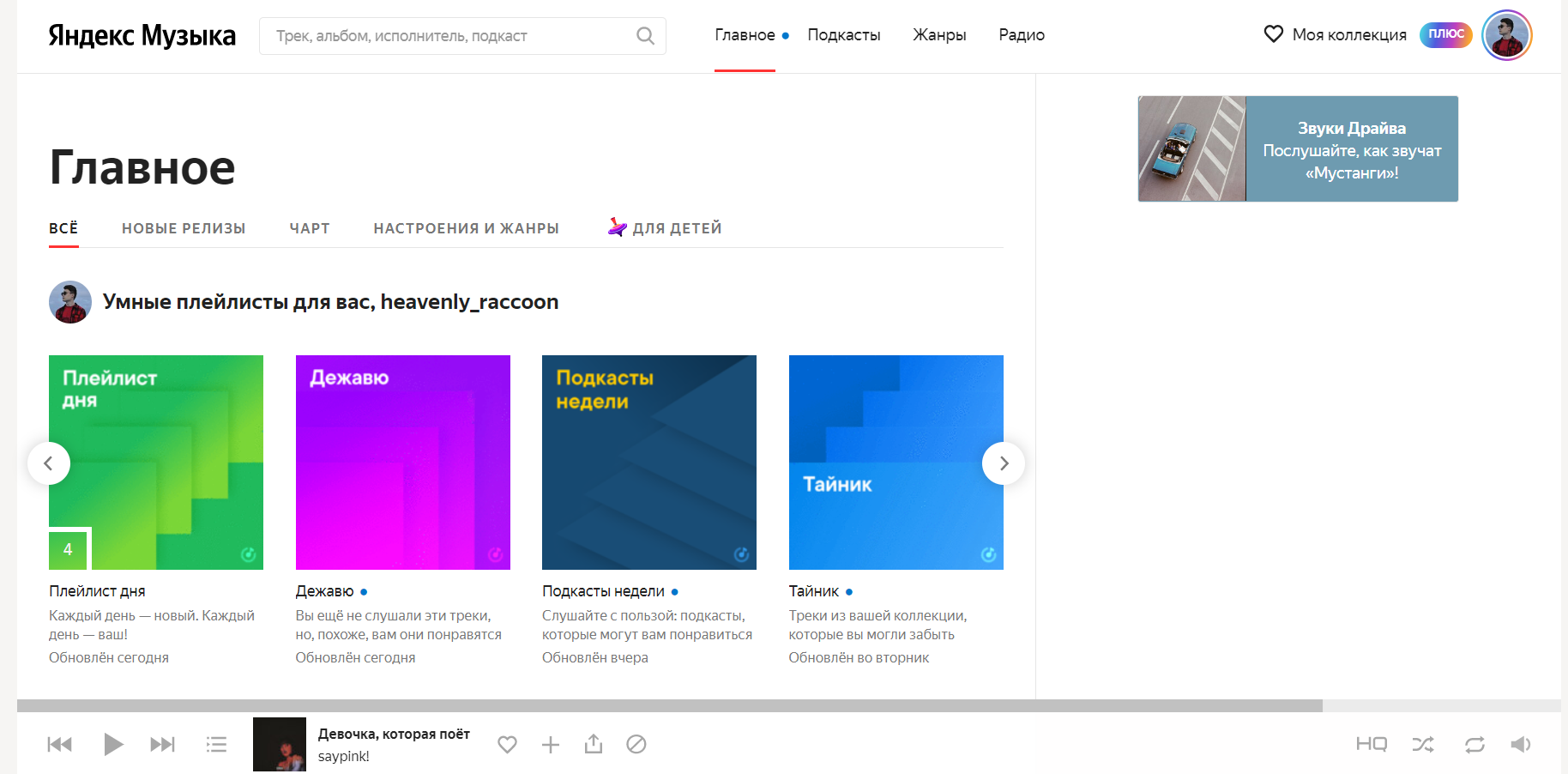


Рисунок 1.2 − Интерфейс «Яндекс Музыка»

Проанализировав «Яндекс Музыка», можно выделить её основные минусы и плюсы.

Основные минусы:

* Без подписки на «Яндекс Плюс» на экране много рекламы;
* Без подписки на «Яндекс Плюс» во время прослушивания будет проигрываться реклама;
* Нельзя скачать музыку на устройство.

Основные плюсы:

* Наличие мобильного приложения;
* Удобный интерфейс;
* Различные плейлисты, с подборками музыки по вкусу пользователя;
* Множество подкастов для прослушивания;

Для выполнения этого курсового проекта не обошлось без использования дополнительной литературы. Список представлен в разделе «Список литературы».

Самый большое количество информации было получено с официального сайта docs.microsoft.com и форума для программистов или разработчиков stackoverflow.com.

Дизайн приложения я брал с сайта mp3trip.info т.к. он максимально приближен к моей задумке, и дизайн сайта довольно стильный и удобный.

# **2 Проектирование и разработка базы данных**

При разработке курсового проекта понадобились следующие объекты:

1. Таблицы;

2. Хранимые процедуры;

3. Индексы;

## **2.1 Спецификация функциональных требований**

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований — шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

* бизнес-требования;
* пользовательские требования;
* функциональные требования.

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Курсовой проект не подразумевает наличие заказчика, который мог бы выдвинуть бизнес-требования, поэтому в качестве таких высокоуровневых требований можно рассматривать общие требования к разрабатываемому средству. К их числу относятся:

* простота и лёгкость интерфейса;
* использование принципов объектно-ориентированного программирования;
* использование архитектурных шаблонов проектирования;
* использование системы управления базами данных (СУБД);

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.

Следующими требованиями являются требования пользователей. Данные требования описывают цели и задачи, которые пользователям позволит решить система. Таким образом, в пользовательских требованиях указано, что клиенты смогут делать с помощью системы. Пользователь данного программного решения должен иметь возможность:

* регистрировать себя в системе;
* входить в приложение, после ввода данных, необходимых для аутентификации;
* прослушивать музыку;
* перематывать музыку;
* искать музыку;
* скачивать музыку;
* создавать плейлисты;
* добавлять и удалять музыку из плейлистов;
* редактировать личные данные (имя, о себе, пароль, аватар).

Администратор имеет возможность:

* выполнять действия пользователя;
* просматривать данные пользователей;
* блокировать пользователей;
* редактировать данные пользователей;
* редактировать данные плейлистов пользователей и музыки в них.

Наглядно это видно на UML-схеме, изображённой на рисунке 2.1 (Приложение А).

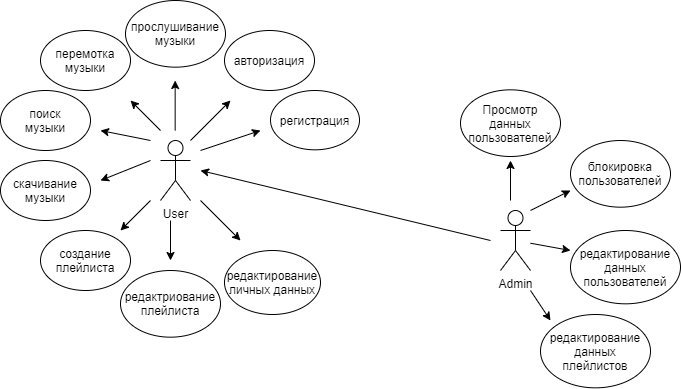


Рисунок 2.1 – UML-диаграмма возможностей

После проведения анализа были выявлены следующие функциональные требования:

* архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command;
* вся информация должна храниться в базе данных;
* приложение должно производить валидацию вводимых пользователем данных;
* приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;
* приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;
* приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;
* приложение должно предоставлять возможность поиска и прослушивании музыки.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

## **2.2 Разработка модели базы данных**

Для реализации базы данных для Cosmic было разработано 5 таблиц. Диаграмма базы со структурой связей представлена в приложении А, а в приложении Б представлен скрипт создания БД.

Логически можно вывести 5 основных таблиц: Users, Playlists, Songs, Roles и SongPlaylist.

Таблица Users представляет перечень пользователей, состоит из столбцов:

* Id – идентификатор пользователя, тип int, первичный ключ;
* Login – логин пользователя, тип nvarchar;
* Password – захешированный пароль пользователя, тип nvarchar;
* Name – имя пользователя, тип nvarchar.
* Avatar – аватар пользователя, тип varbinary.
* AboutUser – информация о пользователе, тип nvarchar.
* Role – роль пользователя, тип int.

Таблица Playlists представляет информацию о плейлистах, состоит из столбцов:

* Id – идентификатор плейлиста, тип int, первичный ключ;
* Name – название плейлиста, тип nvarchar;
* Avatar – аватар плейлиста, тип varbinary;
* UserId – Id владельца плейлиста, тип int, вторичный ключ.

Таблица Songs представляет информацию о музыкальной композиции, состоит из столбцов:

* Id – идентификатор композиции, тип int, первичный ключ;
* Name – название композиции, тип nvarchar;
* Author – исполнитель композиции, тип nvarchar;
* Image – картинка песни, тип nvarchar;
* Size – длительность композиции, тип nvarchar;
* Link ­– ссылка на песню в интернете, тип nvarchar.

Таблица Roles представляет информацию о ролях, состоит из столбцов:

* id – идентификатор роли, тип int, первичный ключ;
* Name – название роли, тип nvarchar.

Таблица SongPlaylists, вспомогательная таблица для связи многие ко многим, представляет информацию о содержании плейлистов, состоит из столбцов:

* SongId – идентификатор композиции, тип int, вторичный ключ;
* PlaylistId – идентификатор плейлиста, тип int, вторичный ключ.

## **2.3 Пользователи**

Пользователь базы данных – это физическое или юридическое лицо, которое имеет доступ к БД и пользуется услугами информационной системы для получения информации.

При проектировании базы данных было использовано 2 пользователя. Первый пользователь – User – обычный пользователь, имеет доступ для чтения таблиц, связанных с музыкой.

Второй пользователь – Admin – обладает более обширным перечнем прав. Имеет права для чтения, изменения и записи во всех таблицах.

## **2.4 Хранимые процедуры**

Хранимая процедура – объект базы данных, представляющий собой набор SQL-инструкций, который компилируется один раз и хранится на сервере.

При разработке курсового проекта было создано 15 процедур для следующих целей:

1. Выборка данных из таблиц по различным полям;

2. Добавление пользователя/ проверка при авторизации;

3. Удаление данных из таблиц;

4. Добавление данных в таблицы;

5. Изменение данных в таблицах;

6. Экспорт и импорт таблиц в формат json;

Весь перечень созданных процедур будет представлен в Приложении В.

### **2.4.1 Выборка данных из таблиц**

Для выборки данных из таблиц были написаны следующие процедуры: GetPlaylistsByUserId, GetSongsByNameOrAuthor, GetSongsByPlaylistId, GetUserById и другие. Поскольку таблица Songs хранит в себе огромное количество композиций было принято решение создать индексы, тем самым сократив время поиска по этой таблице.

### **2.4.2 Добавление пользователя и проверка данных при авторизации**

При регистрации нового пользователя с помощью процедуры CreateUser, ему необходимо заполнить необходимые поля: login, password. Для увеличения безопасности учетной записи пользователя его пароль шифруется с помощью алгоритма MD5. Также проверяется корректность ввода логина на повторение. Данная процедура представлена ниже на рисунке 2.4.1.

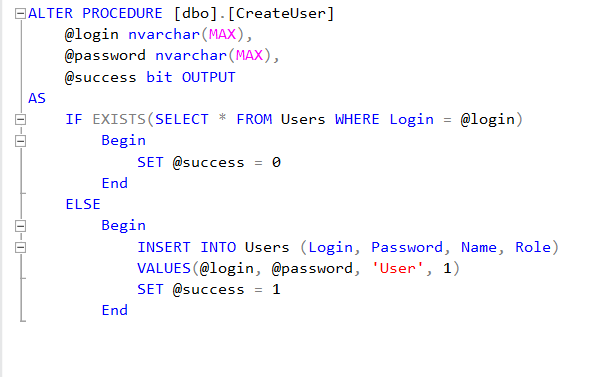


Рисунок 2.4.1 – Пример создания процедуры CreateUser

При авторизации с помощью процедуры SignIn, ему необходимо заполнить необходимые поля: login, password. Процедура проверяет наличие пользователя с такими данными и возвращает найденного пользователя либо ошибку. Данная процедура представлена ниже на рисунке 2.4.2.

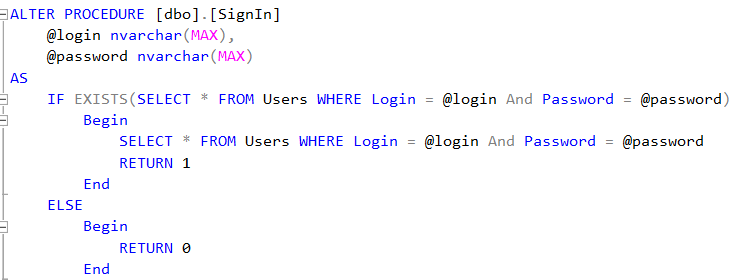


Рисунок 2.4.2 – Пример создания процедуры SignIn

### **2.4.3 Удаление данных из таблиц**

Для редактирования плейлистов была создана процедура для удаления композиций из них RemoveSongFromPlaylist. Она удаляет строки из таблицы SongPlaylist. Данная процедура представлена ниже на рисунке 2.4.3.

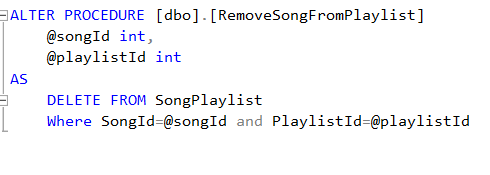


Рисунок 2.4.3 – Пример создания процедуры RemoveSongFromPlaylist

### **2.4.4 Добавление данных в таблицы**

Для управления базой данных были созданы процедуры добавления новых строк: CreateUser, AddSongToPlaylist, CreateSong. Они были разработаны для заполнения соответствующих таблиц.

Процедура AddSongToPlaylist позволяет пользователю добавить композицию в свой плейлист. Данная процедура представлена ниже на рисунке 2.4.4.

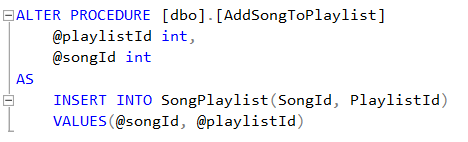
****

Рисунок 2.4.4 – Пример создания процедуры AddSongToPlaylist

### **2.4.5 Изменение данных в таблицах**

Для управления базой данных были созданы процедуры обновления: UpdateUserAboutUser, UpdateUserImage, UpdateUserName, UpdateUserPassword. Они были разработаны для обновления данных в соответствующих таблицах.

### **2.4.6 Экспорт и импорт таблиц в формат json**

JSON — это формат, который хранит структурированную информацию и в основном используется для передачи данных между сервером и клиентом.

Файл JSON представляет собой более простую и лёгкую альтернативу расширению с аналогичными функциями XML (Extensive Markup Language).

Часто возникает необходимость импортировать и экспортировать JSON-файлы в MS SQL, в данной курсовой работе используются функция OPENJSON для импорта. В данном курсовом проекте функция импорта используются для таблицы Songs, так как эта таблица является важнейшей в базе данных. Пример создания процедуры импорта можно посмотреть на рисунках 2.4.5.

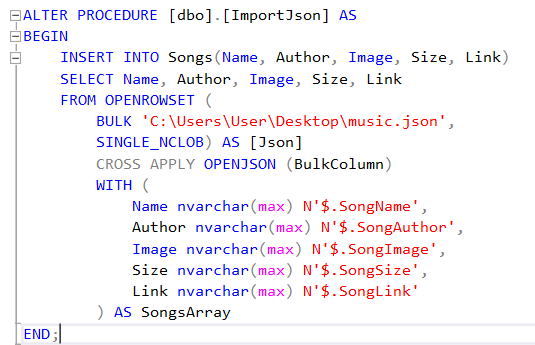


Рисунок 2.4.5 – Пример создания процедуры ImportJson

* 1. **Реализация технологий**

В процессе разработки проекта была разработана технология: полнотекстового поиска в Бд.

Полнотекстовый поиск – это поиск слов или фраз в текстовых данных. Обычно такой вид поиска используется для поиска текста в большом объёме данных, например, таблица с миллионом и более строк, так как он значительно быстрей обычного поиска, который можно осуществить, используя конструкцию LIKE. Полнотекстовый поиск подразумевает создание специального индекса (он отличается от обычных индексов) текстовых данных, который представляет собой некий словарь слов, которые встречаются в этих данных.

### **2.5.1** **Реализация технологии мультимедийные типы данных в БД**

Возможности полнотекстового поиска в Microsoft SQL Server:

* В полнотекстовом поиске SQL сервера можно осуществлять поиск не только по отдельным словам или фразам, но и по префиксным выражениям, например, задать текст начала слова или фразы;
* Также можно искать слова по словоформам, например, различные формы глаголов или существительные в единственном и во множественном числе, т.е. по производным выражениям;
* Можно построить запрос так, чтобы найти слова или фразы, находящиеся рядом с другими словами или фразами, т.е. выражения с учетом расположения;
* Есть возможность искать синонимические формы конкретного слова (тезаурус) т.е., например, если в тезаурусе определено, что «Автомобиль» и «Машина» – это синонимы, то при поиске слова «Автомобиль» в результирующий набор войдут и строки, содержащие слово «Машина»;
* Для того чтобы не учитывать в поиске некоторые слова можно использовать «список стоп-слов» т.е. по словам, включенным в этот список, поиск выполняться не будет.

Для реализации полнотекстового поиска нужно установить компонент «Полнотекстовый поиск» в Microsoft SQL Server. Установка представлена ниже на рисунке 2.5.1

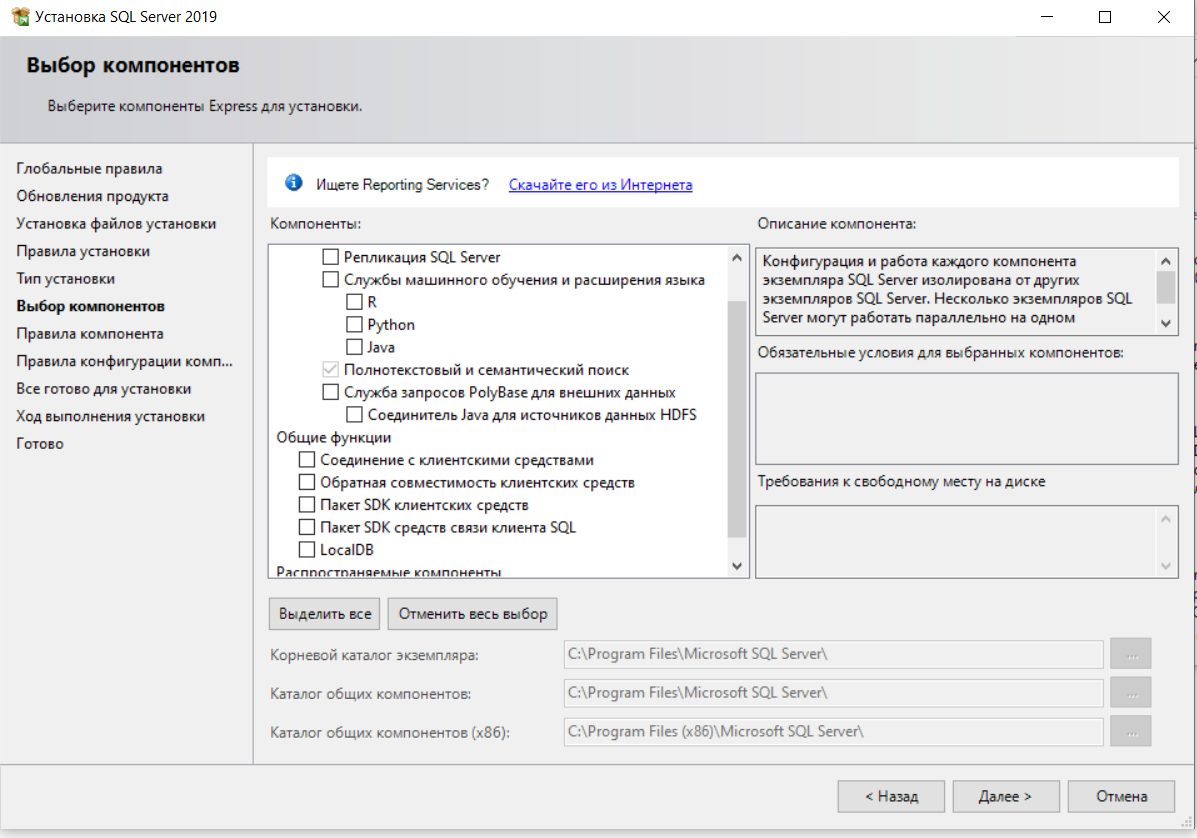


Рисунок 2.5.1 – Пример установки компонента «Полнотекстовый поиск»

Следующий шаг это создание полнотекстового каталога. Далее создание полнотекстового индекса и наконец тестирование с помощью функции contains. Все шаги представлены на рисунке 2.5.2

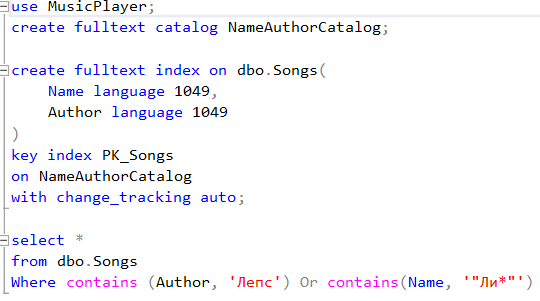


Рисунок 2.5.2 – Пример создания полнотекстового каталога и индекса.

# **3 Проектирование и разработка программного средства**

В процессе выполнения курсового проекта было разработано desktop-приложение, предназначенное для взаимодействия пользователя с базой данных. Приложение было разработано на языке программирования C# с использованием таких технологий как .NET Framework, WPF и Entity Framework. При создании приложения использовался паттерн проектирование MVVM. Структура разработанного программного средства представлена ниже.

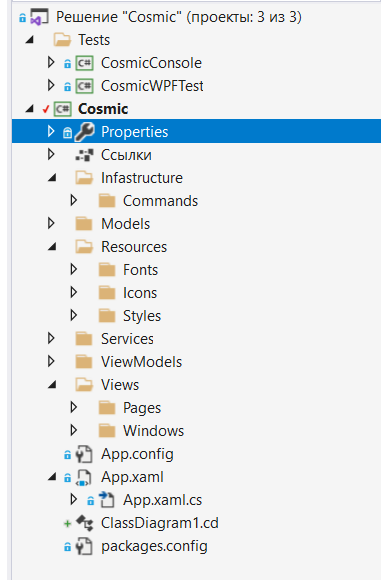


Рисунок 3.1 – Структура проекта

Описание обобщенной структуры проекта представлено в таблице 3.1.

Папка Tests хранит в себе проекты «CosmicConsole» и «CosmicWPFTest» в них тестировались функции и пользовательские стили. В папке Infastructure находится класс с пользовательской командой «LamdaCommand». Диаграмма классов Command представлена на рисунке 3.2.

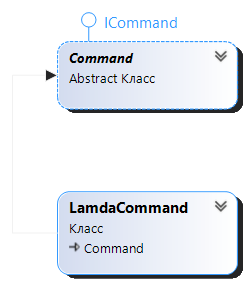


Рисунок 3.2 – Диаграмма классов Command

В папке Models содержатся сущностные классы, которые используются для создания БД и хранения массивов с музыкой.

Папка Resources разделена еще на три папки: Fonts – хранит шрифты, Icons – хранит картинки и иконки, Styles – хранит словарь ресурсов со стилями.

В папке Services хранятся различные вспомогательные классы: конвертер, парсер, проигрыватель, класс с функциями для Entity Framework.

Папка ViewModels хранит базовую реализацию ViewModel и классы наследуемые от него. Листинг базового класса представлен в приложении В. Диаграмма классов ViewModels представлена на рисунке 3.3.

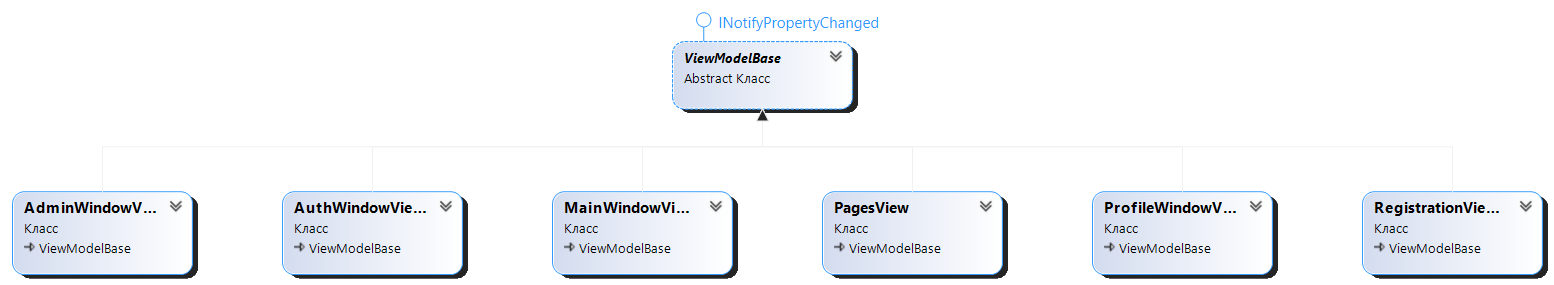


Рисунок 3.3 – Диаграмма классов ViewModel

Папка Views разделена на папки Pages и Windows которые хранят реализацию страниц и окон соответственно.

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде LamdaCommand. Класс реализует два метода:

* CanExecute: определяет, может ли команда выполняться
* Execute: собственно, выполняет логику команды

А также реализует событие CanExecuteChanged вызывается при изменении условий, указывающий, может ли команда выполняться. Для этого используется событие CommandManager.RequerySuggested

В приложении было разработаны 3 вспомогательных класса:

* Converter;
* EntityFunction;
* Player;

Диаграмма вспомогательных классов представлена на рисунке 3.4.

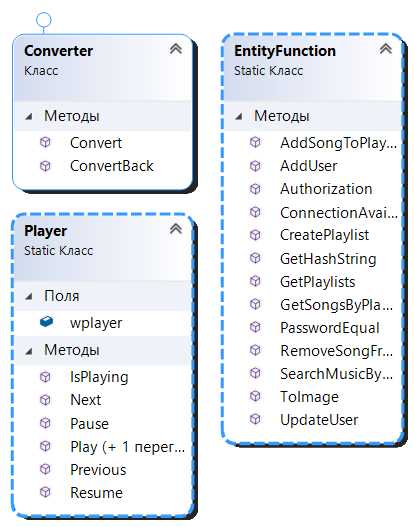


Рисунок 3.4 – Диаграмма вспомогательных классов

Класс Converter предназначен для передачи двух параметров через CommandParametr. Реализация класса представлена в приложении Г.

Класс EntityFunction содержит функции для работы с базой данных. Получение, редактирование, сохранение данных.

Класс Player реализует прослушивание, перемотку, переключение музыки. Прослушивание аудио потока из интернета без скачивания аудио реализовано с помощью библиотеки WMPLib. Листинг метода «Play» представлен в приложении Д.

Для взаимодействия через приложения с данными из БД была применена технология Entity Framework. Entity Framework предоставляет собой технологию работы с данными, которая основана на платформе .NET Framework. Эта технология представляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

# **4 Тестирование, проверка работоспособности и анализ полученных данных**

Прежде всего были проведены тесты авторизации: проверка на пустые и неверные данные. Валидация выполнена таким образом, что при нажатии на кнопки входа или регистрации при некорректных данных будет показано соответствующее сообщение и вход/регистрация не выполняться. При таком способе валидации шанс ввести невалидные данные минимален. На рисунке 4.1 видно, что валидация на странице регистрации не допускает ввода некорректных полей. Запрещается вводить пробелы в поля логина и пароля. Длина логина от 1 до 20 символов. Пароль должен содержать только латинские буквы, минимум 6 символов. Поле «Подтвердите пароль» должно совпадать с полем «Пароль».

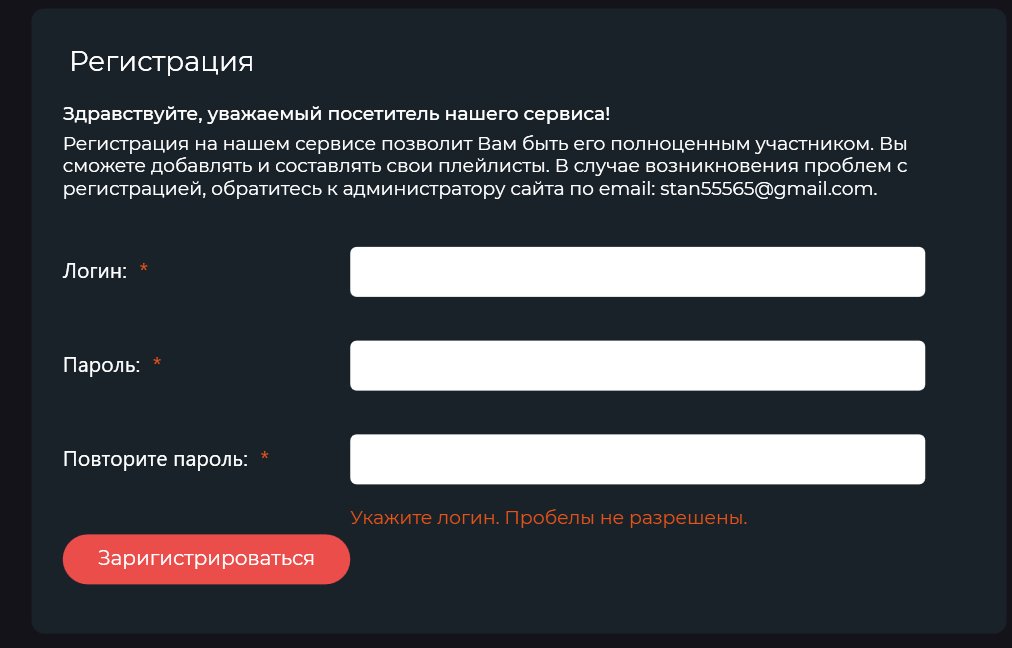


Рисунок 4.1 – Регистрация без ввода данных

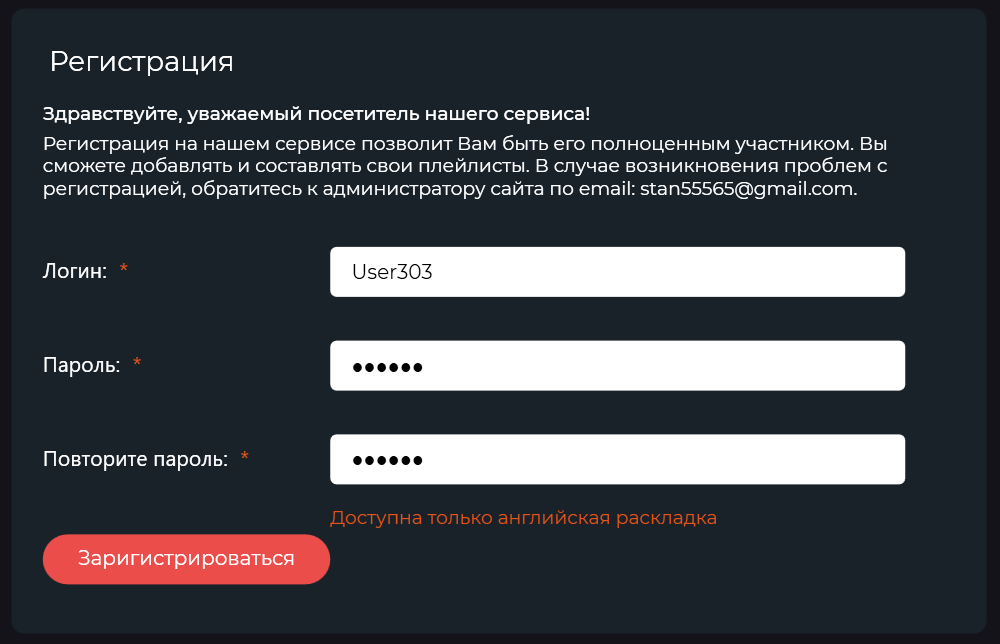


Рисунок 4.2 – Ввод пароля на кириллице

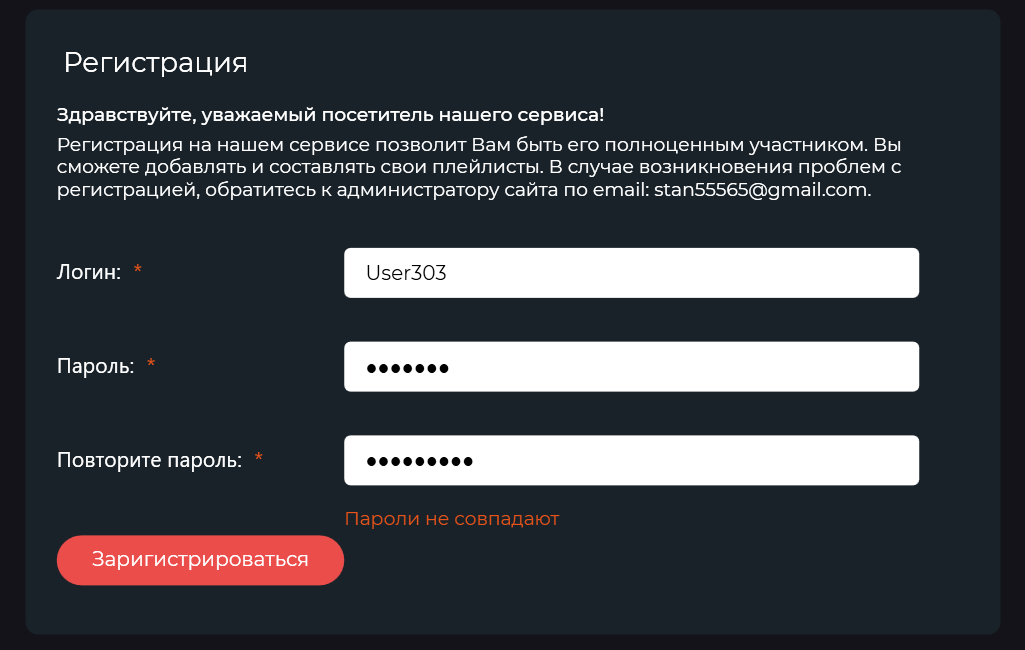


Рисунок 4.3 – Ввод несовпадающих паролей

На рисунке 4.4 представлена валидация формы входа

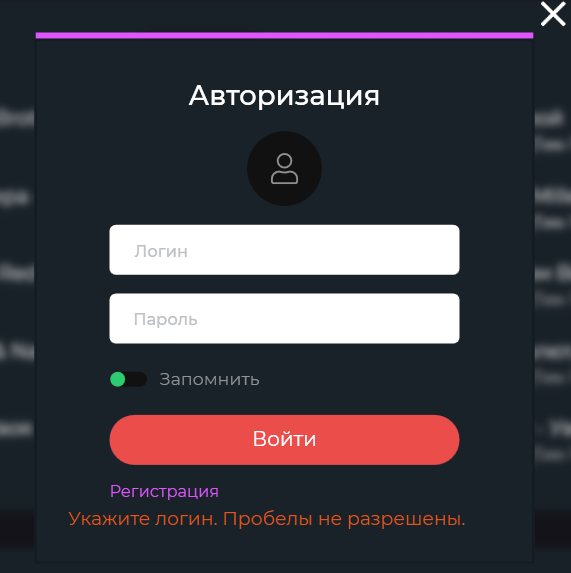


Рисунок 4.4 – Вход без ввода данных

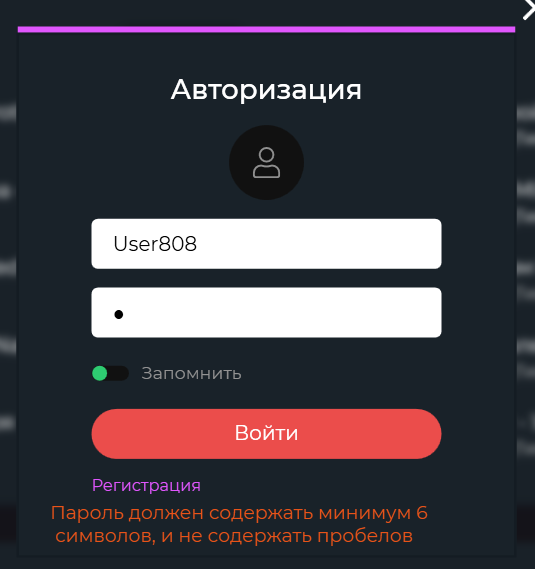


Рисунок 4.5 – Валидация пароля

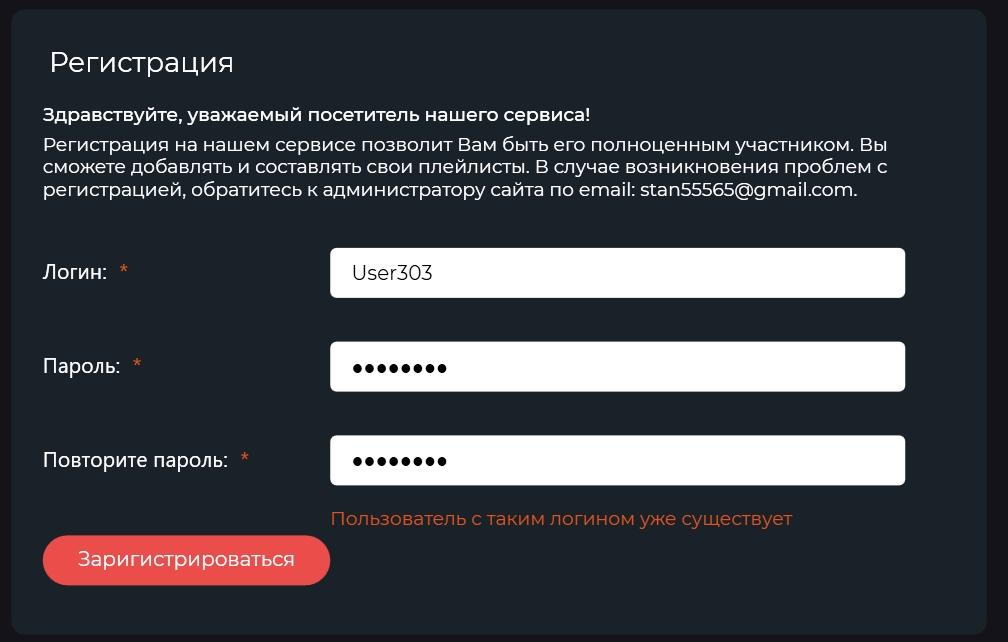


Рисунок 4.6 – Регистрация уже существующего пользователя

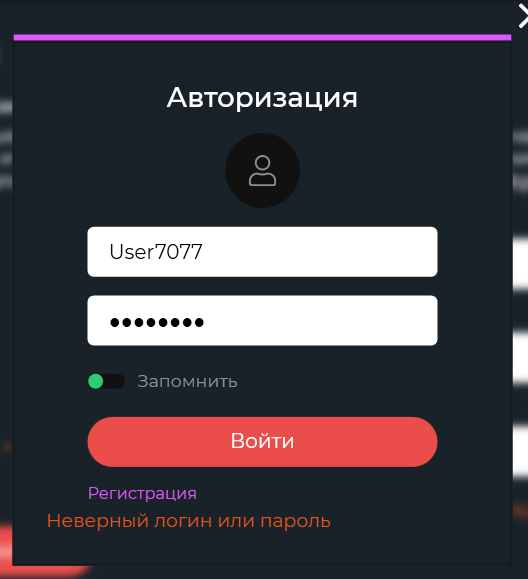


Рисунок 4.7 – Вход с неправильным логином или паролем

Также были проведены тесты на работу валидации внутри приложения. Валидация организованна таким же образом, как и на форме авторизации. Все ошибки, возникшие при валидации, доступны и отображены удобным образом, что позволяет пользователю легко понять какие данные не валидны и быстро исправить их.

Пример результата валидации данных и обработки различных экстремальных ситуаций внутри приложения приведены на рисунках 4.5-4.6

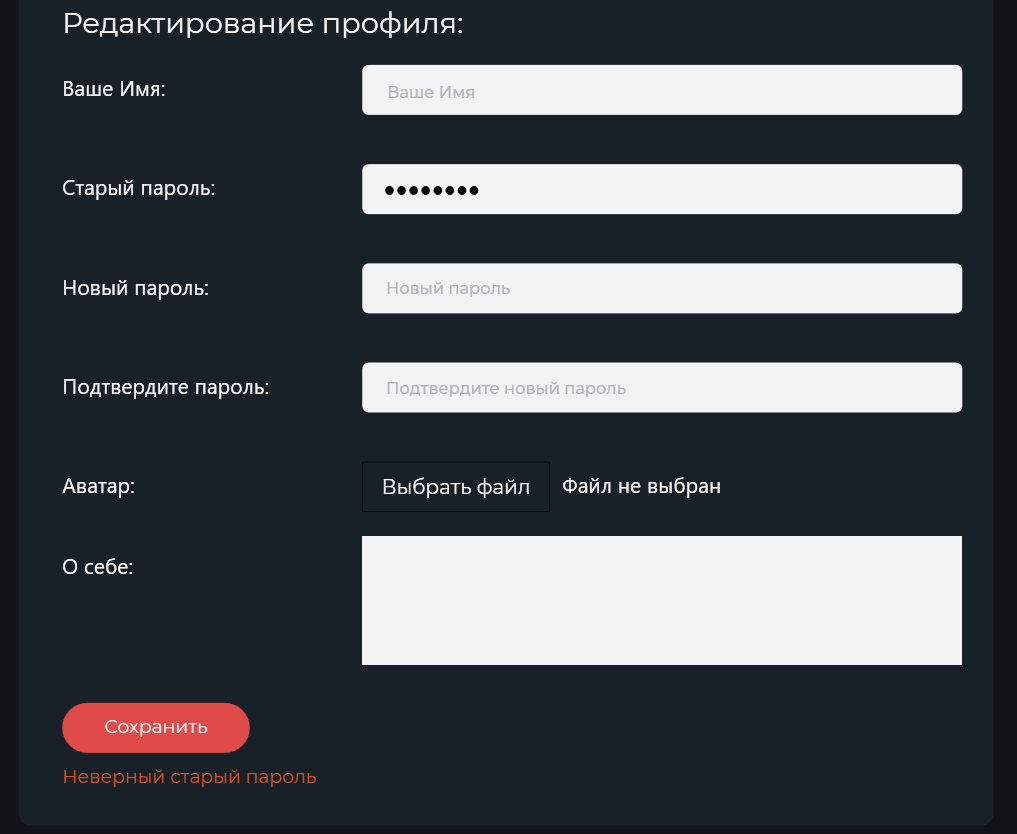


Рисунок 4.8 – Валидация смены пароля, при вводе неправильного старого пароля

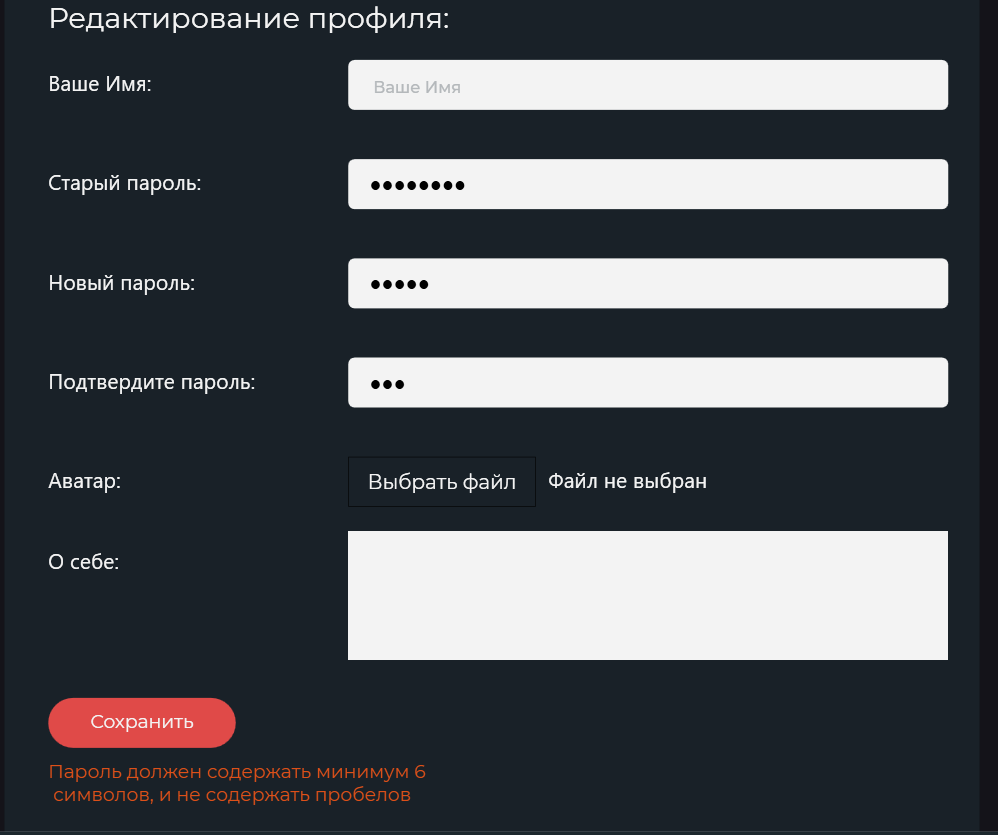


Рисунок 4.9 – Валидация ввода нового пароля

Так как приложение работает с интернет подключением были обработаны соответствующие ситуации. Например, запуск приложения без интернет соединения или обрыв интернет соединения при загрузке приложения. Обработка ситуации представлена на рисунке 4.10.

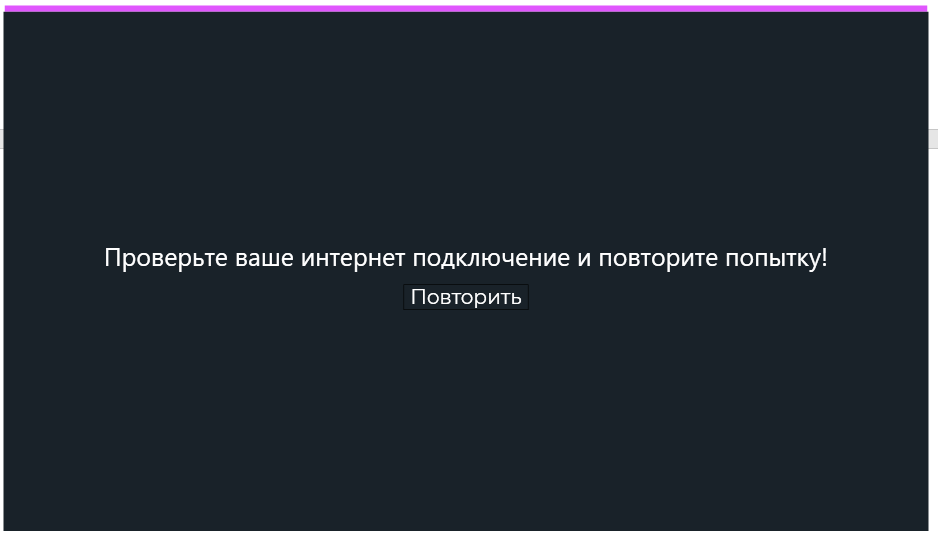


Рисунок 4.10 – Запуск приложения без интернет соединения

На данном этапе были выполнены тесты на проверку работоспособности приложения, а именно на проверку валидации и запуска приложения без подключения к интернету. Были проведены анализы результатов, которые показали, что валидация и проверка подключения в приложении работают в соответствии с тем, как они задумывались.

# **Заключение**

В процессе решения поставленной задачи была достигнута поставленная цель по созданию базы данных для стримингового сервиса «Cosmic», которая в совокупности с приложением формирует полноценное desktop-приложение для удобного поиска и прослушивания музыкальных композиций. Основной целью курсового проекта стало проектирование базы данных для дальнейшей интеграции с приложением, которое помогло облегчить взаимодействие с базой данных посредством программного интерфейса. При разработке выполнены следующие пункты:

* авторизация и регистрация пользователей;
* составление интерфейса для взаимодействия с БД;
* импорт/экспорт дынных из/в json;
* разделение ролей пользователей;
* шифрование данных клиента.

Приложение прошло тестирование при использовании в БД большого количество данных.

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

# **Список литературы**

1. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/> . Дата доступа: 02.12.2021

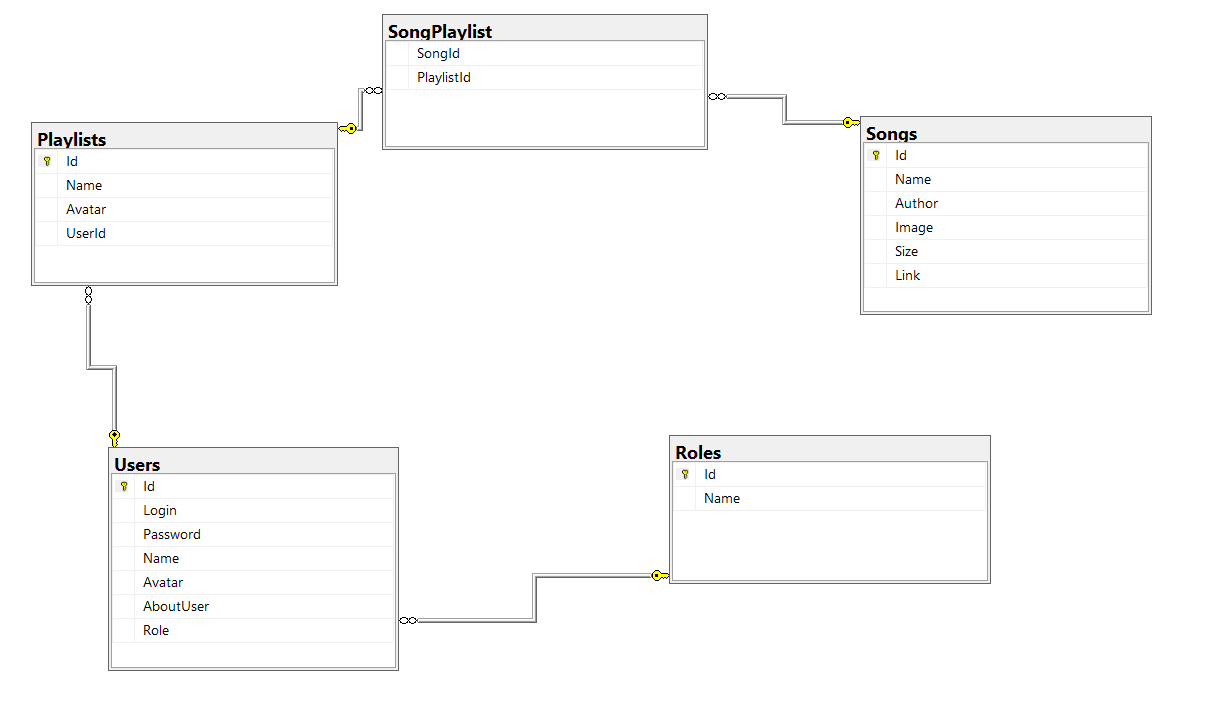
2. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> . Дата доступа: 02.012.2021

3. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 04.12.2021

4. Microsoft Docs Archived Content [Электронный ресурс]/ Режим доступа: <https://docs.microsoft.com/en-us/archive/> Дата доступа: 11.12.2021

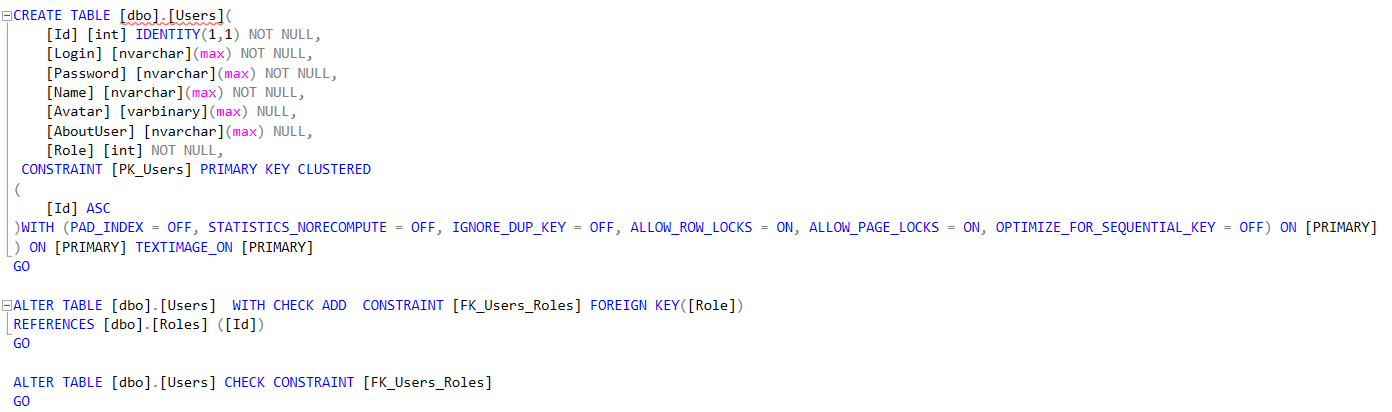
5. Форум для программистов или разработчиков [Электронный ресурс] – <https://stackoverflow.com/> – Дата доступа: 05.11.2021

# **Приложение А**

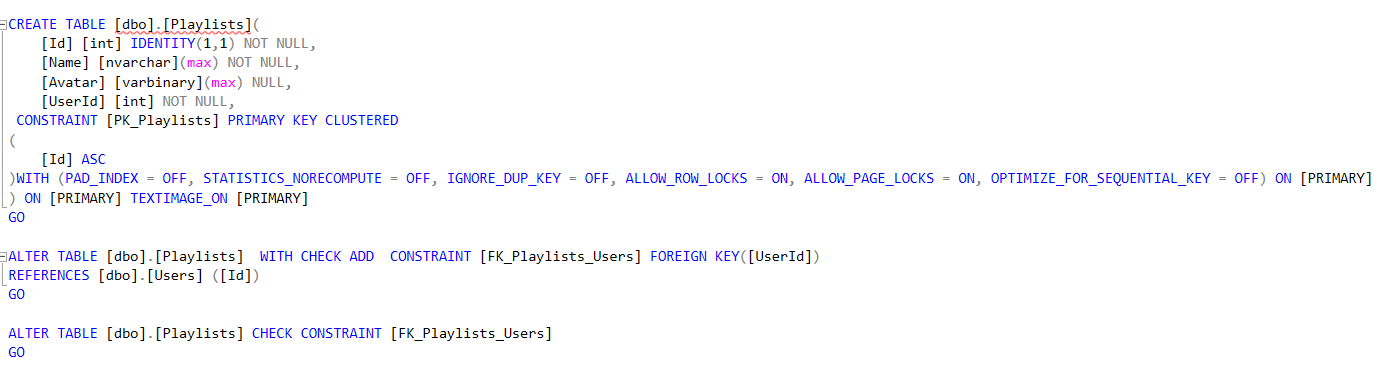
****

# **Приложение Б**

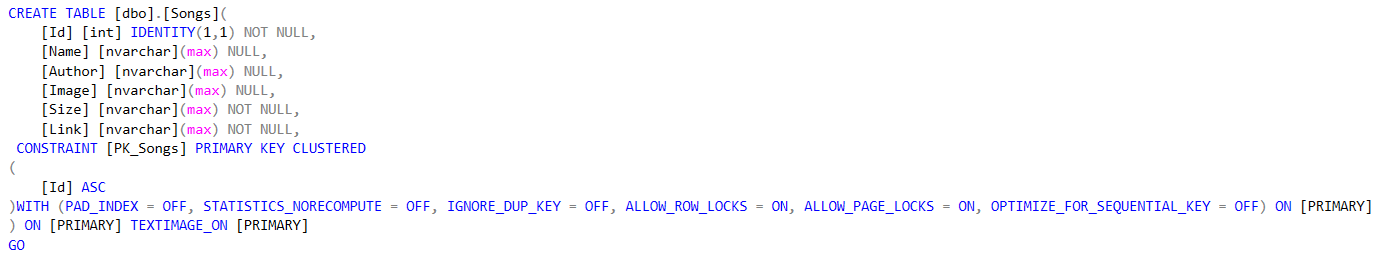
Листинг 1 – Создание таблицы «Users»



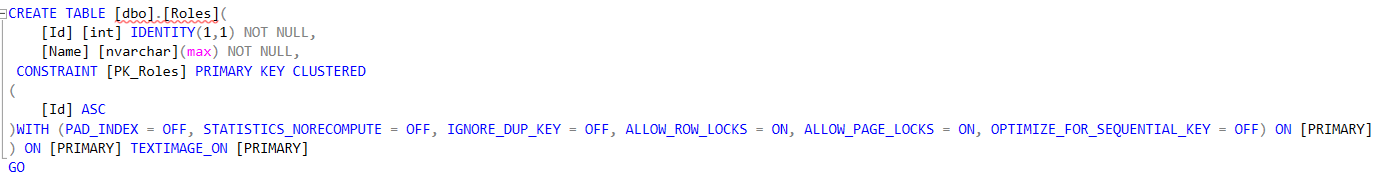
Листинг 2 – Создание таблицы «Playlists»



Листинг 3 – Создание таблицы «Songs»



Листинг 4 – Создание таблицы «Roles»

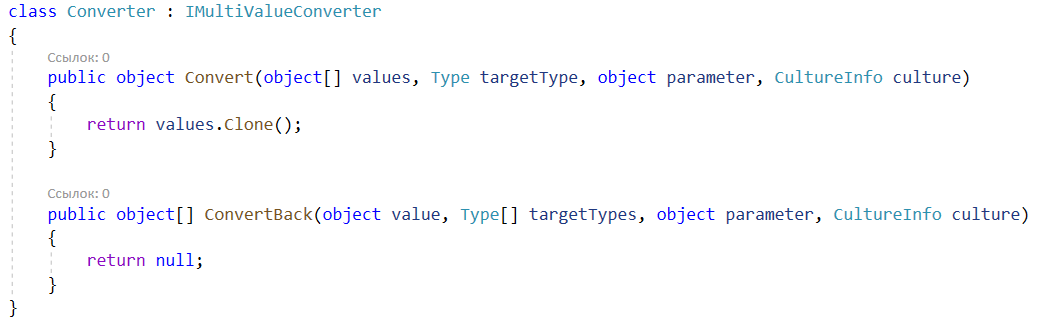


# **Приложение В**

|  |  |  |
| --- | --- | --- |
| Название процедуры | Описание | Принимаемые параметры |
| AddSongToPlaylist | Процедура добавления музыки в плейлист | PlaylistId in, SongId in |
| CreatePlaylist | Процедура добавления плейлиста | name in, user in |
| CreateUser | Процедура добавления пользователя | login in, password in, success out |
| GetPlaylistsByUserId | Процедура получения плейлистов | user in |
| GetSongsByNameOrAuthor | Поиск музыки по имени и автору | req in |
| GetSongsByPlaylistId | Получение музыки из плейлиста | PlaylistId in |
| GetUserById | Получение пользователя по Id | User in |
| ImportJson | Импорт данных из Json |  |
| RemoveSongFromPlaylist | Удаление песни из плейлиста | SongId in, PlaylistId in |
| SignIn | Авторизация пользователя | Login in, password in |
| UpdateUserAboutUser | Изменение секции о пользователе | userId in, aboutUser in |
| UpdateUserImage | Изменение аватара пользователя | userId in, image in |
| UpdateUserName | Изменение имени пользователя | userId in, name id |
| UpdateUserPassword | Изменение пароля пользователя | userId in, password in |

# **Приложение Г**

Листинг 5 – «Converter»



# **Приложение Д**

Листинг 6 – «Play»

