|  |
| --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ |
| **ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ ИМЕНИ К.Г. РАЗУМОВСКОГО (ПЕРВЫЙ КАЗАЧИЙ УНИВЕРСИТЕТ)»**  **(ФГБОУ ВО «МГУТУ ИМ. К.Г. РАЗУМОВСКОГО (ПКУ)»)** |
|  |
| **УНИВЕРСИТЕТСКИЙ КОЛЛЕДЖ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ** |

|  |
| --- |
| **КУРСОВОЙ ПРОЕКТ** |
|  |
| по междисциплинарному курсу: МДК.02.02. Технология разработки и зашиты баз данных. |
|  |
| на тему: Разработка информационной системы для любителя  музыкальных произведений. |
|  |
| студента группы ПКС-304  специальности 09.02.03 Программирование в компьютерных системах |
| Томского Николая Алексеевича |

|  |  |  |
| --- | --- | --- |
| Студент |  | Н.А. Томский |
| Руководитель курсового проекта |  | Е.А. Ларионова |
| Председатель ПЦК специальности 09.02.03 Программирование в компьютерных системах |  | А.И. Глускер |

|  |  |  |
| --- | --- | --- |
| Дата защиты «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_ 2020 г. | | |
| Оценка:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | |
| Заведующий отделением № 3 |  | Е.Л. Гусева |

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc54135359)

[1 ОБЩАЯ ЧАСТЬ 8](#_Toc54135360)

[1.1 Назначение и цели создания системы 8](#_Toc54135361)

[1.2 Обзор и анализ предметной области 8](#_Toc54135362)

[1.3 Жизненный цикл базы данных 10](#_Toc54135363)

[1.4 Выбор и характеристика СУБД 11](#_Toc54135364)

[1.5 Выбор и характеристика среды разработки приложения 11](#_Toc54135365)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 14](#_Toc54135366)

[2.1 Постановка задачи 14](#_Toc54135367)

[2.2 Архитектура информационной системы 15](#_Toc54135368)

[2.3 Логическая модель базы данных 16](#_Toc54135369)

[2.4 Нормализация таблиц 16](#_Toc54135370)

[2.5 Описание таблиц 17](#_Toc54135372)

[2.6 Разработка приложения 20](#_Toc54135373)

[2.6.1 Диаграмма вариантов использования 20](#_Toc54135374)

[2.6.2 Функции приложения 20](#_Toc54135375)

[2.6.3 Защита информационной системы 21](#_Toc54135376)

[2.7 Инструкция пользователю 22](#_Toc54135377)

[2.7.1 Общие сведения об информационной системе 22](#_Toc54135378)

[2.7.2 Требования к техническим средствам 22](#_Toc54135379)

[2.7.3 Требования к программным средствам 23](#_Toc54135380)

[2.7.4 Настройка информационной системы 23](#_Toc54135381)

[2.7.5 Формы приложения 23](#_Toc54135382)

[ЗАКЛЮЧЕНИЕ 28](#_Toc54135387)

[ПРИЛОЖЕНИЕ А. Скрипты на создание базы данных и хранимых процедур 29](#_Toc54135388)

[ПРИЛОЖЕНИЕ Б. Код приложения 54](#_Toc54135389)

# ВВЕДЕНИЕ

Курсовой проект посвящен разработке информационной системы для любителя музыкальных произведений.

Основная цель проекта – создание информационной системы с музыкой в различных жанрах, которая позволит оценивать произведения, возводя рейтинги альбомов и исполнителей, и объединит множество людей в одном месте.

В современном мире все держится на музыке, музыка становится одним из самых главных занятий в жизни многих граждан, практически каждый человек, к примеру, с утра в метро слушает в своих наушниках любимые треки. Без музыки жизнь для современных людей будет скучной и неинтересной, дело в том, что мы привыкли уже по жизни идти с музыкой, она задает нам правильный ритм, благодаря ей, даже, когда человеку грустно, настроение может улучшиться.

Музыка распространяется по всему миру очень быстрыми темпами, к примеру, в одной стране с утра ее только записали в студии, а вечером в другой стране, ей уже наслаждается много человек, которым данный трек очень понравился. Так же, сейчас не одна вечеринка в клубе, ни один концерт, фильм не обходятся без песен, без музыки, она добавляет различные ощущения, которые помогают понять, к примеру, фильм, с другой страны, музыка может передать ощущение того, что происходит, может передать главную тематику чего-либо.

Информационная система – это взаимосвязанная совокупность информационных, технических, программных, математических, организационных, правовых, эргономических, лингвистических, технологических и других средств, а также персонала, предназначенная для сбора, обработки, хранения и выдачи экономической информации и принятия управленческих решений.

Актуальность курсового проекта состоит в том, чтобы можно было объединить пользователей в автоматизированной системе, не требующей подключения Интернет, которое используется в большинстве существующих приложений, и дать возможность прослушивать музыкальные произведения в хорошем качестве записи.

В пример можно выделить такой сервис, как «Яндекс.Музыка». Позволяет прослушивать треки, добавлять их в избранное, слушать подборки песен по жанрам, настроению и т.д. Чтобы прослушивать песни необходимо подключение к Интернет, а, чтобы можно было слушать в высоком качестве и без рекламы, придется платить за подписку.

Таких сервисов множество – и мобильных, и десктопных, однако у всех них несколько ограничений – плата за подписку и необходимость в Интернет.

Анализ существующих систем показал, что необходима система, которая предполагает полный доступ ко всей аудиотеке без рекламы и без платежа за подписку, и, как было заверено ранее, без подключения к Интернет.

К задачам проекта, которые необходимо решить для достижения цели, относятся:

1. анализ предметной области,
2. проектирование базы данных средствами Microsoft SQL Server,
3. разработка приложения средствами Visual Studio,
4. защита базы данных и приложения.

Обоснование структуры курсового проекта:

1. введение: в данном разделе описываются цели и задачи проекта, а также его актуальность, так как изначально необходимо определить, что предстоит изучать и делать в проекте,
2. основная часть: этот раздел включает в себя анализ предметной области, для которой разрабатывается данное приложение, назначения и цели разработки, а также жизненный цикл базы данных,
3. специальная часть: в данном разделе описаны архитектурные решения, которые применяются в разработке, а также процесс разработки программного обеспечения и базы данных,
4. заключение: вывод о проделанной работе и полученных навыках за время разработки приложения,
5. список используемых источников: этот раздел содержит в себе перечень ресурсов, которые были использованы для получения необходимой информации во время разработки.

# 1 ОБЩАЯ ЧАСТЬ

## 1.1 Назначение и цели создания системы

Назначение курсового проекта - обеспечить поиск треков, альбомов и исполнителей, оценку произведений и их воспроизведение.

Основная цель курсового проекта – создание информационной системы для любителя музыкальных произведений, а также реализация учебного плана и формирование профессиональных компетенций по модулю ПМ.02:

* ­разработка объектов базы данных,
* реализация базы данных в конкретной системе управления базами данных,
* решение вопросов администрирования базы данных,
* реализация методов и технологий защиты информации в базах данных.

## 1.2 Обзор и анализ предметной области

База данных создается для информационного обслуживания любителей музыкальных произведений, редакторов и администраторов базы данных музыкальных произведений. База данных должна содержать данные о пользователях, их ролях, странах проживания, музыкальных произведениях, альбомах, исполнителях, странах и жанрах исполнителей, изображениях профилей пользователей, исполнителей, альбомов, музыкальных произведений.

В соответствии с предметной областью система строится с учётом следующих особенностей:

1) каждый пользователь:

* имеет одну из ролей,
* может не добавлять, добавить одно и более музыкальных произведений в Избранное,
* может иметь или не иметь изображение профиля,
* может проживать или не проживать в одной из стран;

2) каждая роль может быть у нескольких пользователей;

3) каждое музыкальное произведение:

* существует в единственном роде,
* может не быть, быть у одного или более пользователей,
* может быть записано в один и более альбомов,
* может быть записано одним и более исполнителями,
* наследует изображение альбома, в котором оно было записано;

4) каждый альбом:

* существует в единственном роде,
* может включать в себя одно или более музыкальных произведений,
* записывается одним или несколькими исполнителями,
* может иметь или не иметь изображение;

5) каждый исполнитель:

* существует в единственном роде,
* может записать одно и более музыкальных произведений,
* может записать один и более альбомов.
* может играть в одном или более жанрах,
* основался в одной из стран,
* может иметь или не иметь изображение;

6) каждый жанр может относится сразу к нескольким исполнителям;

7) каждая страна может быть у нескольких исполнителей и поьзователей.

Выделим базовые сущности этой предметной области:

1. Пользователи. Атрибуты: логин, пароль, фамилия, имя, пол, номер изображения, номер страны, номер роли;
2. Музыкальные произведения. Атрибуты: номер музыкального произведения, название музыкального произведения, бинарный код, номер изображения;
3. Альбомы. Атрибуты: номер альбома, название альбома, дата выхода, номер изображения;
4. Исполнители. Атрибуты: номер исполнителя, название исполнителя, дата основания, номер страны, номер изображения;
5. Роли. Атрибуты: номер роли, название роли;
6. Изображения. Атрибуты: номер изображения, название изображения, бинарный код;
7. Страны. Атрибуты: номер страны, название страны;
8. Жанры. Атрибуты: номер жанра, название жанра;
9. Связь между пользователями и музыкальными произведениями – Лайки. Атрибуты: логин, номер музыкального произведения;
10. Связь между исполнителями, альбомами и музыкальными произведениями – Записи. Атрибуты: номер записи, номер музыкального произведения, номер альбома, номер исполнителя;
11. Связь между исполнителями и жанрами – Стили. Атрибуты: номер исполнителя, номер жанра.

## 1.3 Жизненный цикл базы данных

Жизненный цикл – это процесс проектирования, реализации и управления базой данных. Он состоит из:

1. Анализа предметной области, который включает в себя планирование разработки (15.02.2020 – 24.03.2020), определение требований (15.02.2020 – 24.03.2020), сбор и анализ требований (15.02.2020 – 24.03.2020),
2. Проектирования базы данных, которое состоит из создания концептуальной модели (25.03.2020), логической модели (26.03.2020), физической модели (27.03.2020),
3. Реализации, в процессе которой пишутся скрипты на создание базы данных, связей сущностей в базе данных, а также разрабатывается интерфейс приложения и его функции (01.03.2020 – 11.05.2020).

## 1.4 Выбор и характеристика СУБД

Для разработки базы данных была выбрана среда Microsoft SQL Server Management Studio 2019.

SQL Server Management Studio (SSMS) — утилита для Microsoft SQL Server для конфигурирования, управления и администрирования компонентов базы данных. Данная утилита содержит редактор скриптов (который в основном и будет нами использоваться) и графическую программу, которая работает с объектами и настройками сервера. Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также управлять ими.

Весомыми плюсами SQL Server являются простота понимания принципов работы, синтаксиса, а также возможность создавать объекты баз данных либо с помощью скрипта, либо вручную, с помощью конструктора. Высокая производительность и надежность является главной отличительной чертой данной СУБД. MS SQL Server успешно применяется для создания многопользовательских приложений, где файлы базы данных являются разделяемыми ресурсами сети. Кроме того, в данной СУБД реализована надежная система защиты от несанкционированного доступа к файлам.

## 1.5 Выбор и характеристика среды разработки приложения

Для разработки приложения была выбрана среда программирования Microsoft Visual Studio 2019, потому что эта среда разработки обеспечивает возможность создания приложений с удобным интерфейсом с помощью языка C# и системы построения клиентских приложений WPF.Net Framework.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты 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, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework и Silverlight.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. Visual Studio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (как, например, Subversion и Visual SourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода на предметно-ориентированных языках программирования) или инструментов для прочих аспектов процесса разработки программного обеспечения (например, клиент Team Explorer для работы с Team Foundation Server).

Некоторые преимущества:

* удобная отладка и поиск ошибок,
* возможность создания специальной библиотеки для тестирования,
* подробная документация об ошибках, располагающаяся на сайте компании Microsoft,
* бесплатная (Версия «Community Edition»),
* функциональная, поддерживается большое количество плагинов,
* интеллектуальность системы, предупреждающая об ошибках и помогающая упростить и улучшить код.

ADO.NET (ActiveX Data Object для .NET) — технология, предоставляющая доступ и управление данными, хранящимся в базе данных или других источниках (Microsoft SQL Server, Microsoft Access, Microsoft Excel, Microsoft Outlook, Microsoft Exchange, Oracle, OLE DB, ODBC, XML, текстовые файлы)[1], основанных на платформе .NET Framework и входящая в состав .NET Framework 2.0, представляет собой набор библиотек[2]. В отличие от технологии ADO, которая была в основном предназначена для тесно связанных клиент-серверных систем, ADO.NET больше нацелена на автономную работу с помощью объектов DataSet. Объекты DataSet представляют локальные копии взаимосвязанных таблиц данных, каждая из которых содержит набор строк и столбцов. Объекты DataSet позволяют вызывающей сборке (наподобие веб-страницы или программы, выполняющейся на настольном компьютере) работать с содержимым DataSet, изменять его, не требуя подключения к источнику данных, и отправлять обратно блоки измененных данных для обработки с помощью соответствующего адаптера данных. Но, пожалуй, самое фундаментальное различие между классической ADO и ADO.NET состоит в том, что ADO.NET является управляемой кодовой библиотекой, и, значит, подчиняется тем же правилам, что и любая управляемая библиотека. Типы, составляющие ADO.NET, используют протокол управления памятью CLR, принадлежат к той же системе типов (классы, интерфейсы, перечисления, структуры и делегаты), и доступ к ним возможен с помощью любого языка .NET. Классы ADO.NET находятся в сборке System.Data.dll.

# 2 СПЕЦИАЛЬНАЯ ЧАСТЬ

## 2.1 Постановка задачи

Спроектировать средствами MS SQL SERVER базу данных и создать средствами WPF C# приложение, позволяющее:

1. обычному пользователю:

* регистрироваться,
* авторизовываться под своими логином и паролем,
* изменять личную информацию, пароль, изображение профиля,
* добавлять и удалять музыкальные произведения из Избранного,
* просматривать жанры музыкальных произведений,
* искать исполнителей по жанру,
* просматривать все музыкальные произведения, альбомы и исполнителей,
* проводить поиск по музыкальным произведениям, альбомам, исполнителям, жанрам;

1. редактору (кроме привилегий обычного пользователя):

* добавлять, изменять и удалять исполнителей, альбомы и музыкальные произведения,
* добавлять, изменять и удалять информацию и изображения исполнителей, альбомов и музыкальных произведений;

1. администратору (кроме привилегий обычного пользователя и редактора):

* добавлять, изменять и удалять роли пользователей,
* добавлять, изменять и удалять привилегии ролей пользователей,
* добавлять, изменять и удалять пользователей и их данные (личная информация, логин, пароль, изображение профиля, музыкальные произведения из разделов, сами разделы),
* добавлять, изменять и удалять страны

1. осуществлять поиск:

* исполнителей,
* альбомов,
* музыкальных произведений,
* жанров,
* музыкальных произведений в Избранном,
* ролей,
* стран,
* пользователей;

1. Обеспечить защиту базы данных и приложения.

## 2.2 Архитектура информационной системы

«Клиент-сервер» – вычислительная или сетевая архитектура, в которой задания или сетевая нагрузка распределены между поставщиками услуг, называемыми серверами, и заказчиками услуг, называемыми клиентами (Рисунок 1). Фактически клиент и сервер – это программное обеспечение. Обычно эти программы расположены на разных вычислительных машинах и взаимодействуют между собой через вычислительную сеть посредством сетевых протоколов, но они могут быть расположены также и на одной машине.

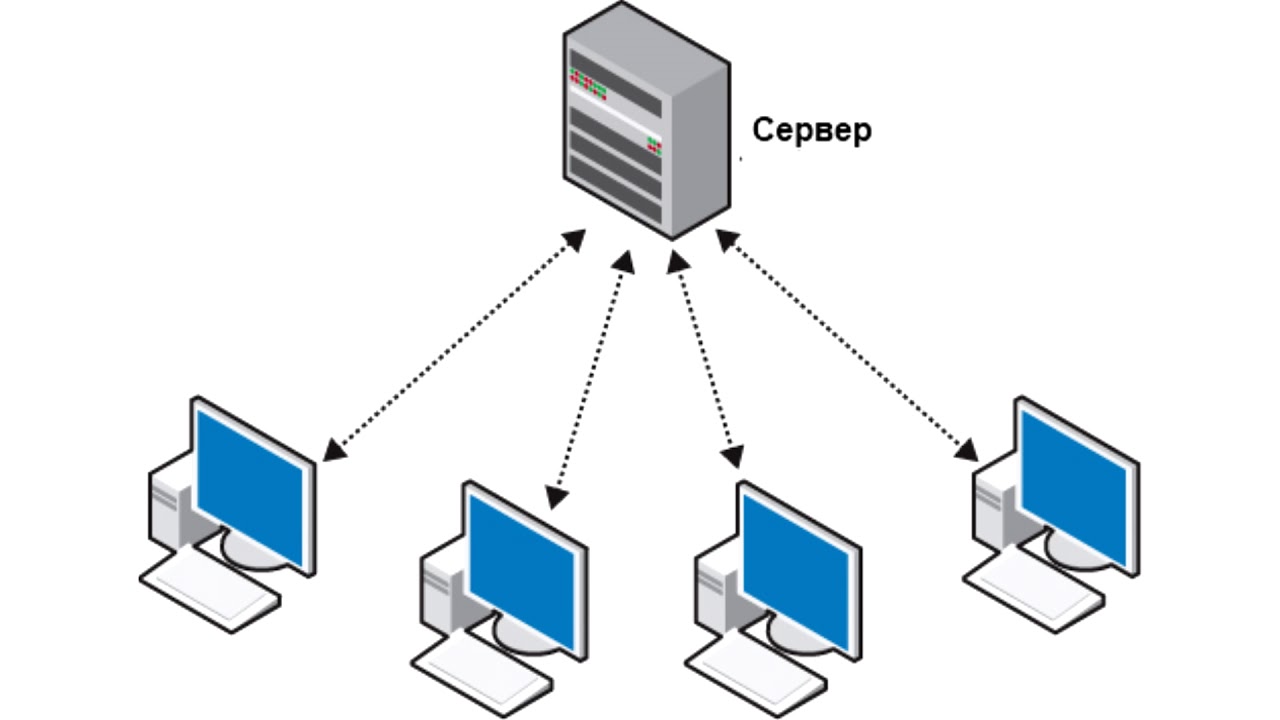


Рисунок 1­ – Модель архитектуры «клиент-сервер»

## 2.3 Логическая модель базы данных

Логическая модель базы данных курсового проекта (Рисунок 2).

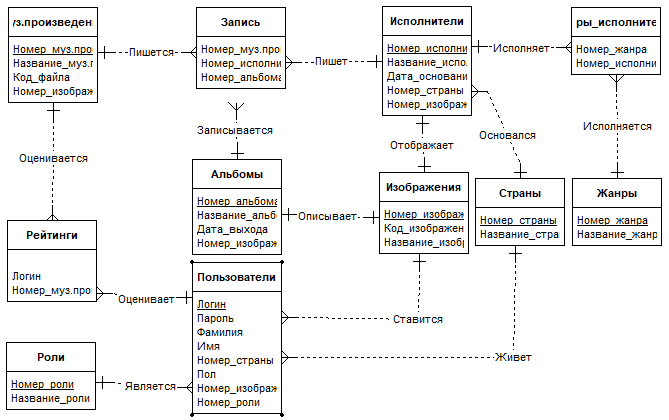


Рисунок 2 – Логическая модель базы данных

## 2.4 Нормализация таблиц

Первая нормальная форма – отношение находится в первой нормальной форме, если все атрибуты просты и неделимы, без повторений.

Вторая нормальная форма – отношение находится в первой нормальной форме и каждый не ключевой атрибут функционально зависит от первичного ключа.

Третья нормальная форма – это отношение, которое находится во второй нормальной форме, и между не ключевыми атрибутами нет функциональных зависимостей.

Таблица 1 – Приведение к 3НФ

|  |  |
| --- | --- |
| **Этап** | **Описание** |
| 1НФ | Выделим основные атрибуты Users:   * Login, * Password, * Surname, * Name, * Gender,   – Country\_id, |

## Продолжение таблицы 1

|  |  |
| --- | --- |
| **Этап** | **Описание** |
| 1НФ | * Image\_id, * Role\_id.   Выделим основные атрибуты Tracks:   * Track\_id, * Track\_name, * Track\_binary, * Image\_id.   Выделим основные атрибуты Records:   * Track\_id, * Album\_id, * Executor\_id.   Выделим основные атрибуты Images:   * Image\_id, * Image\_name,   Image\_binary. |
| 2НФ |  |
| 3НФ | |  | | --- | |  | |  |  | |

## 2.5 Описание таблиц

В этом разделе описаны названия таблиц, их атрибуты, свойства и типы атрибутов, первичные и внешние ключи.

Определения:

* FK – внешний ключ,
* PK – первичный ключ,
* NULL – разрешены нулевые значения.

Таблица 2 – Users

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Login | Nvarchar(20) | PK |
| Password | Nvarchar(20) |  |
| Surname | Nvarchar(20) |  |
| Name | Nvarchar(20) |  |
| Gender | Nvarchar(7) |  |
| Country\_id | Int | NULL, FK |
| Image\_id | Int | NULL, FK |
| Role\_id | Int | FK |

Таблица 3 – Roles

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Id | int | PK |
| Name | Nvarchar(50) |  |

Таблица 4 – Countries,

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Id | int | PK |
| Name | Nvarchar(100) |  |

Таблица 5 – Likes

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Track\_id | int | FK |
| Login | Nvarchar(20) | FK |

Таблица 6 – Images

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Id | int | PK |
| Name | Nvarchar(260) |  |
| Binary | Varbinary(MAX) | NULL |

Таблица 7 – Tracks

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Id | int | PK |
| Name | Nvarchar(50) |  |
| Binary | Varbinary(MAX) | NULL |
| Image\_id | int | FK, NULL |

Таблица 8 – Albums

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Id | int | PK |
| Name | Nvarchar(50) |  |
| Date | Date | NULL |
| Image\_id | int | FK, NULL |

Таблица 9 – Executors

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| id | int | PK |
| Name | Nvarchar(50) |  |
| Date | Date | NULL |
| Country\_id | int | FK, NULL |
| Image\_id | int | FK, NULL |

Таблица 10 – Genres

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Id | int | PK |
| Name | Nvarchar(50) |  |

Таблица 11 – Styles

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Executor\_id | int | FK |
| Genre\_id | int | FK |

Таблица 12 – Records

|  |  |  |
| --- | --- | --- |
| **Название** | **Тип** | **Дополнительно** |
| Track\_id | int | FK |
| Album\_id | int | FK |
| Executor\_id | int | FK |

## 2.6 Разработка приложения

## 2.6.1 Диаграмма вариантов использования

Диаграмма вариантов использования базы данных для любителей музыкальных произведений (Рисунок 3).

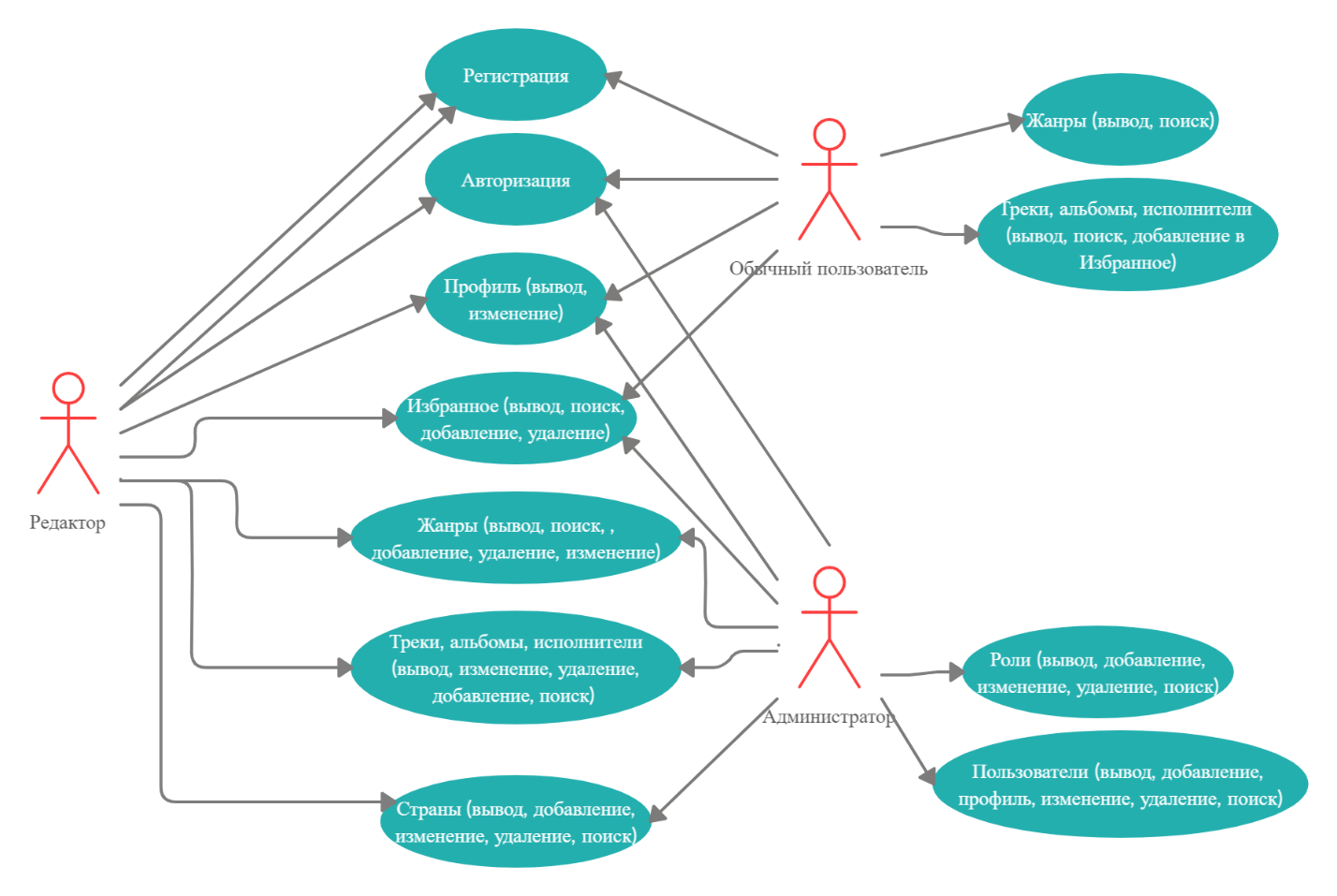


Рисунок 3 – Диаграмма вариантов использования

## 2.6.2 Функции приложения

Приложение имеет следующие функции:

* просмотр избранных треков,
* добавление избранного трека,
* редактирование трека,
* просмотр характеристик товара,
* просмотр списка пользователей,
* добавление нового пользователя,
* удаление пользователя,
* редактирование данных пользователя,
* просмотр списка жанров,
* добавление нового жанра,
* редактирование данных жанра,
* просмотр исполнителей в жанре,
* просмотр списка стран,
* просмотр списка избранных исполнителей,
* добавление новой страны,
* редактирование данных страны,
* просмотр списка ролей,
* добавление новой роли,
* просмотр списка всех треков,
* добавление нового трека,
* просмотр списка исполнителей,
* добавление нового исполнителя,
* изменение исполнителя,
* аутентификация пользователей и т.д.

## 2.6.3 Защита информационной системы

Так как в данном приложении есть поля для ввода данных пользователя, поэтому должна использоваться защита от SQL-инъекций и защиты базы данных в целом. Были использованы следующие средства:

– хранимые процедуры,

– разделение прав пользователей.

Так как у каждой роли пользователя есть различные функции, то необходим разделенный доступ определенной роли к списку процедур и функций. Например, редактор не имеет доступа к ролям, не может их добавлять и удалять. Эти ограничения сделаны программно, в коде приложения. Пользователи, не имеющие права управления определенными формами не имеют к ним доступа.

В самом коде приложения для защиты от SQL-инъекций реализованы запросы к БД через хранимые процедуры с передачей параметров.

## 2.7 Инструкция пользователю

## 2.7.1 Общие сведения об информационной системе

Информационная система для обслуживания любителей музыкальных произведений. Система предоставляет возможности:

* регистрироваться в системе,
* авторизовываться в системе по своему логину и паролю,
* заходить и изменять данные своего профиля,
* проводить поиск, изменять, добавлять и удалять страны, роли, пользователей, треки, альбомы и исполнителей,
* проводить поиск, изменять и удалять треки из Избранного.

## 2.7.2 Требования к техническим средствам

В состав технических средств должен входить компьютер пользователя, включающий:

* клавиатура и мышь,
* видеокарта,
* монитор,
* SSD/HDD – диск,
* процессор Intel Pentium 4 или другой, более производительный.

## 2.7.3 Требования к программным средствам

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы «Windows 10» или «Windows 8.1».

## 2.7.4 Настройка информационной системы

Для настройки информационной системы требуется:

* создать базу данных, запустив скрипт.

## 2.7.5 Формы приложения

## 

Рисунок 4 – Форма регистрации

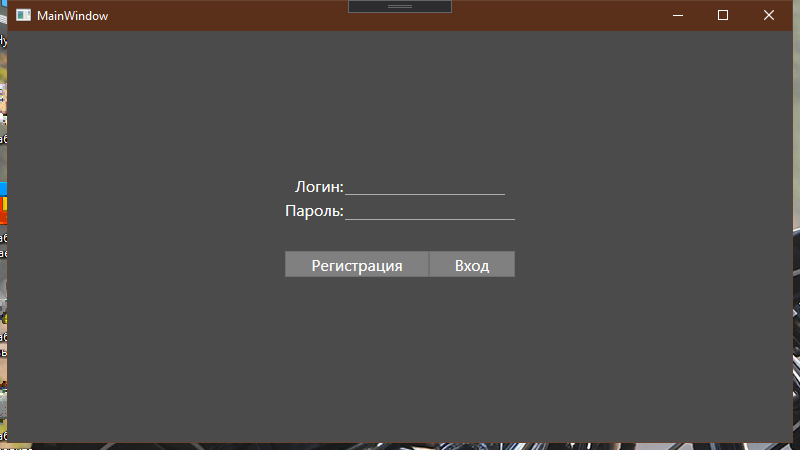


Рисунок 5 – Форма авторизации

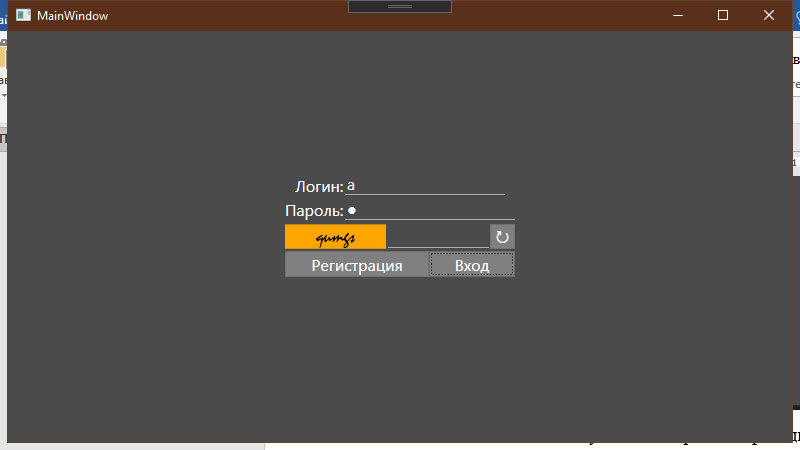


Рисунок 6 – Форма авторизации с капчей

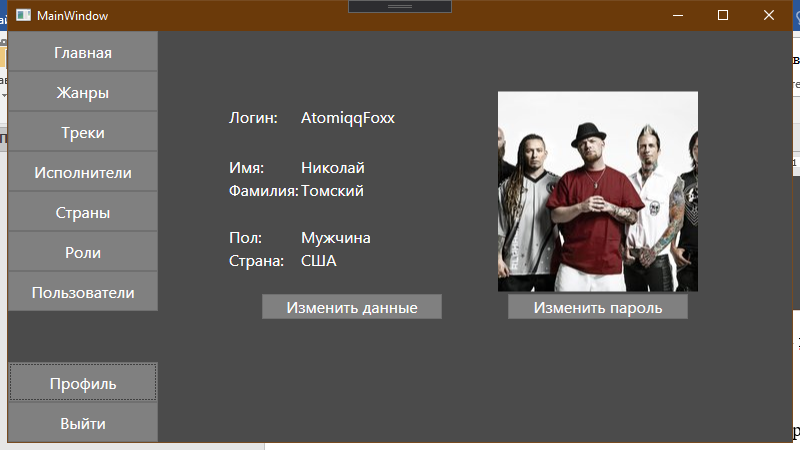


Рисунок 7 – Форма профиля

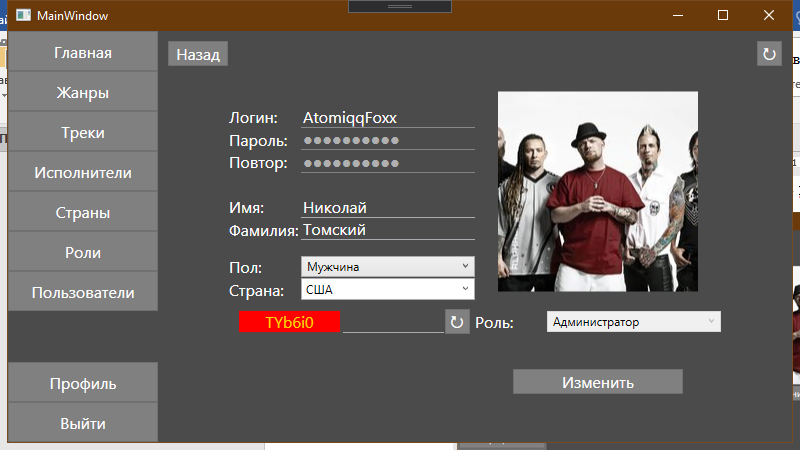


Рисунок 8 – Форма изменения данных пользователя

# 

Рисунок 9 – Форма изменения пароля

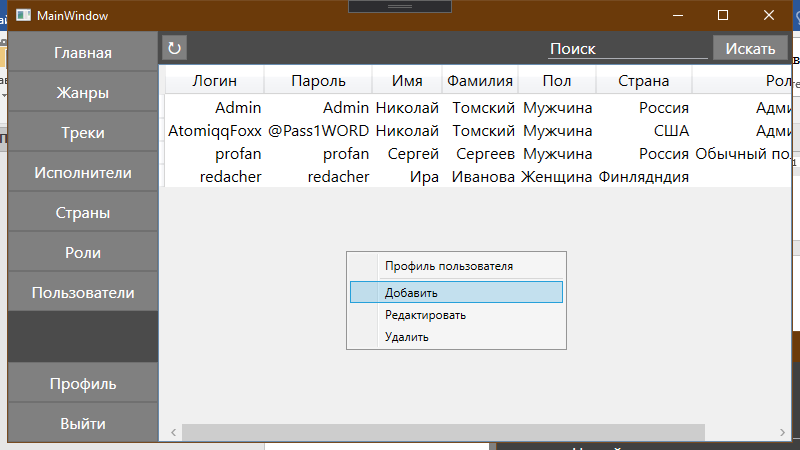


Рисунок 10 – Форма пользователей с дополнительным меню

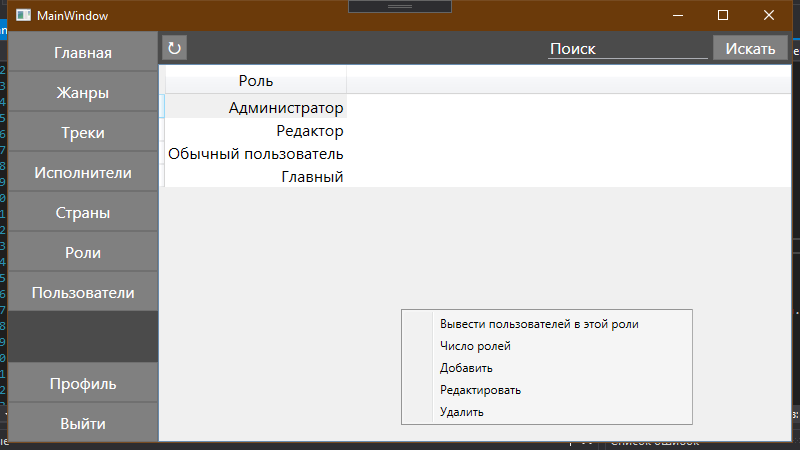


Рисунок 11 – Форма ролей с дополнительным меню

# 

Рисунок 12 – Форма пользователей с ролью администратора

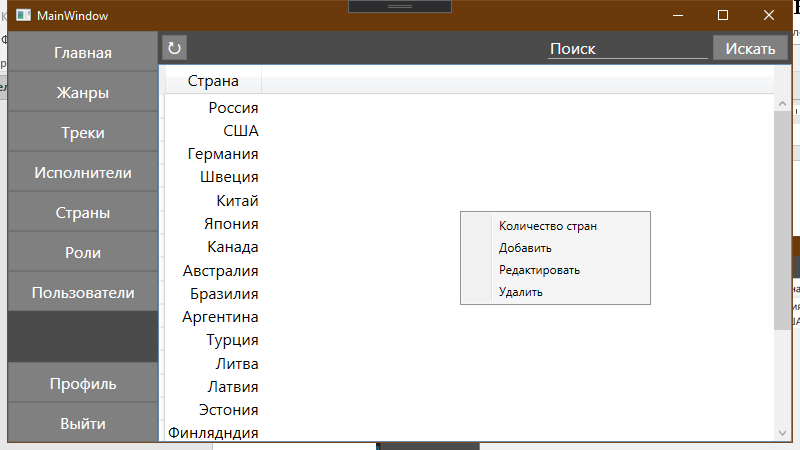


Рисунок 13 – Форма стран с дополнительным меню

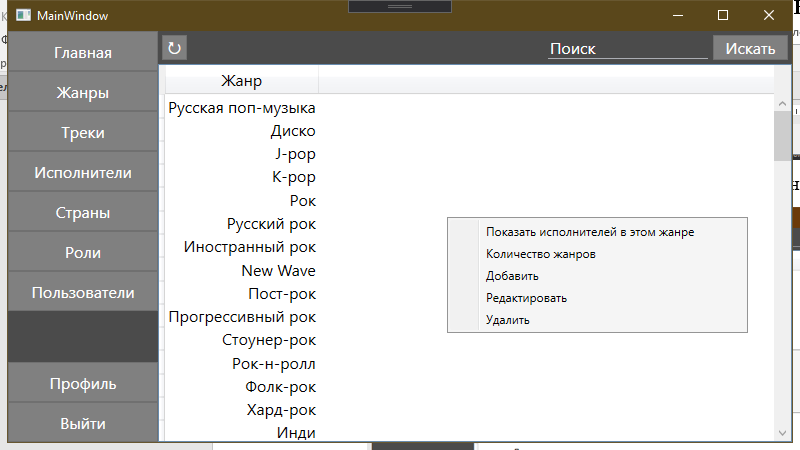


Рисунок 14 – Форма жанров с дополнительным меню

# 

Рисунок 15 – Форма треков, альбомов и исполнителей  
с дополнительным меню альбомов или исполнителей

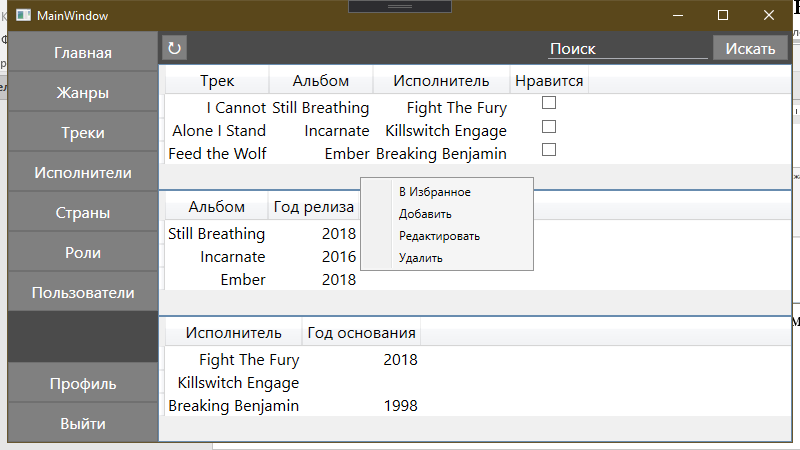


Рисунок 16 – Форма треков, альбомов и исполнителей  
с дополнительным меню треков

# ЗАКЛЮЧЕНИЕ

Курсовой проект посвящен разработке информационной системы для любителей музыкальных произведений.

В процессе выполнения курсового проекта:

1. Изучена предметная область;
2. Разработана база данных MS SQL Server Express;
3. Разработано приложение, позволяющее свободно пользоваться музыкальными произведениями;
4. Осуществлена защита приложения.

В ходе данной работы были выполнены поставленные задачи, повышен навык работы с платформой .NET Framework, приобретен новый опыт проектирования базы данных в SQL Server Express.

Дальнейшее развитие темы курсового проекта предполагает дальнейшую работу над безопасностью приложения, безопасностью данных в БД. Более тщательное скрытие паролей, используя алгоритмы хеширования.

# ПРИЛОЖЕНИЕ А. Скрипты на создание базы данных и хранимых процедур

USE [master]

GO

/\*\*\*\*\*\* Object: Database [KP] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

CREATE DATABASE [KP]

CONTAINMENT = NONE

ON PRIMARY

( NAME = N'KP', FILENAME = N'C:\Users\Public\KP.mdf' , SIZE = 8192KB , MAXSIZE = UNLIMITED, FILEGROWTH = 65536KB )

LOG ON

( NAME = N'KP\_log', FILENAME = N'C:\Users\Public\KP\_log.ldf' , SIZE = 8192KB , MAXSIZE = 2048GB , FILEGROWTH = 65536KB )

GO

ALTER DATABASE [KP] SET COMPATIBILITY\_LEVEL = 140

GO

IF (1 = FULLTEXTSERVICEPROPERTY('IsFullTextInstalled'))

begin

EXEC [KP].[dbo].[sp\_fulltext\_database] @action = 'enable'

end

GO

ALTER DATABASE [KP] SET ANSI\_NULL\_DEFAULT OFF

GO

ALTER DATABASE [KP] SET ANSI\_NULLS OFF

GO

ALTER DATABASE [KP] SET ANSI\_PADDING OFF

GO

ALTER DATABASE [KP] SET ANSI\_WARNINGS OFF

GO

ALTER DATABASE [KP] SET ARITHABORT OFF

GO

ALTER DATABASE [KP] SET AUTO\_CLOSE OFF

GO

ALTER DATABASE [KP] SET AUTO\_SHRINK OFF

GO

ALTER DATABASE [KP] SET AUTO\_UPDATE\_STATISTICS ON

GO

ALTER DATABASE [KP] SET CURSOR\_CLOSE\_ON\_COMMIT OFF

GO

ALTER DATABASE [KP] SET CURSOR\_DEFAULT GLOBAL

GO

ALTER DATABASE [KP] SET CONCAT\_NULL\_YIELDS\_NULL OFF

GO

ALTER DATABASE [KP] SET NUMERIC\_ROUNDABORT OFF

GO

ALTER DATABASE [KP] SET QUOTED\_IDENTIFIER OFF

GO

ALTER DATABASE [KP] SET RECURSIVE\_TRIGGERS OFF

GO

ALTER DATABASE [KP] SET DISABLE\_BROKER

GO

ALTER DATABASE [KP] SET AUTO\_UPDATE\_STATISTICS\_ASYNC OFF

GO

ALTER DATABASE [KP] SET DATE\_CORRELATION\_OPTIMIZATION OFF

GO

ALTER DATABASE [KP] SET TRUSTWORTHY OFF

GO

ALTER DATABASE [KP] SET ALLOW\_SNAPSHOT\_ISOLATION OFF

GO

ALTER DATABASE [KP] SET PARAMETERIZATION SIMPLE

GO

ALTER DATABASE [KP] SET READ\_COMMITTED\_SNAPSHOT OFF

GO

ALTER DATABASE [KP] SET HONOR\_BROKER\_PRIORITY OFF

GO

ALTER DATABASE [KP] SET RECOVERY SIMPLE

GO

ALTER DATABASE [KP] SET MULTI\_USER

GO

ALTER DATABASE [KP] SET PAGE\_VERIFY CHECKSUM

GO

ALTER DATABASE [KP] SET DB\_CHAINING OFF

GO

ALTER DATABASE [KP] SET FILESTREAM( NON\_TRANSACTED\_ACCESS = OFF )

GO

ALTER DATABASE [KP] SET TARGET\_RECOVERY\_TIME = 60 SECONDS

GO

ALTER DATABASE [KP] SET DELAYED\_DURABILITY = DISABLED

GO

ALTER DATABASE [KP] SET QUERY\_STORE = OFF

GO

USE [KP]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Albums] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Albums](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Date] [date] NULL,

[Image] [int] NULL,

CONSTRAINT [PK\_Albums] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Countries] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Countries](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](100) NOT NULL,

CONSTRAINT [PK\_Countries] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Executors] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Executors](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Country] [int] NULL,

[Date] [date] NULL,

[Image] [int] NULL,

CONSTRAINT [PK\_Executors] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Genres] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Genres](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Genres] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Images] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Images](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](260) NOT NULL,

[Binary] [varbinary](max) NOT NULL,

CONSTRAINT [PK\_Images] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Likes] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Likes](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Login] [nvarchar](20) NOT NULL,

[Track] [int] NOT NULL,

CONSTRAINT [PK\_Likes] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Records] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Records](

[Track] [int] NOT NULL,

[Album] [int] NOT NULL,

[Executor] [int] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Roles] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Roles](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

CONSTRAINT [PK\_Roles] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Styles] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Styles](

[Executor] [int] NOT NULL,

[Genre] [int] NOT NULL

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Tracks] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Tracks](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](50) NOT NULL,

[Binary] [varbinary](max) NULL,

[Image] [int] NULL,

CONSTRAINT [PK\_Tracks] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Users] Script Date: 21.10.2020 0:24:21 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Users](

[Login] [nvarchar](20) NOT NULL,

[Password] [nvarchar](20) NOT NULL,

[Name] [nvarchar](20) NOT NULL,

[Surname] [nvarchar](20) NOT NULL,

[Gender] [nvarchar](7) NOT NULL,

[Country] [int] NULL,

[Image] [int] NULL,

[Role] [int] NOT NULL,

CONSTRAINT [PK\_Users] PRIMARY KEY CLUSTERED

(

[Login] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[Albums] WITH CHECK ADD CONSTRAINT [FK\_Albums\_Images] FOREIGN KEY([Image])

REFERENCES [dbo].[Images] ([Id])

GO

ALTER TABLE [dbo].[Albums] CHECK CONSTRAINT [FK\_Albums\_Images]

GO

ALTER TABLE [dbo].[Executors] WITH CHECK ADD CONSTRAINT [FK\_Executors\_Countries] FOREIGN KEY([Country])

REFERENCES [dbo].[Countries] ([Id])

GO

ALTER TABLE [dbo].[Executors] CHECK CONSTRAINT [FK\_Executors\_Countries]

GO

ALTER TABLE [dbo].[Executors] WITH CHECK ADD CONSTRAINT [FK\_Executors\_Images] FOREIGN KEY([Image])

REFERENCES [dbo].[Images] ([Id])

GO

ALTER TABLE [dbo].[Executors] CHECK CONSTRAINT [FK\_Executors\_Images]

GO

ALTER TABLE [dbo].[Likes] WITH CHECK ADD CONSTRAINT [FK\_Likes\_Tracks] FOREIGN KEY([Track])

REFERENCES [dbo].[Tracks] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Likes] CHECK CONSTRAINT [FK\_Likes\_Tracks]

GO

ALTER TABLE [dbo].[Likes] WITH CHECK ADD CONSTRAINT [FK\_Likes\_Users] FOREIGN KEY([Login])

REFERENCES [dbo].[Users] ([Login])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Likes] CHECK CONSTRAINT [FK\_Likes\_Users]

GO

ALTER TABLE [dbo].[Records] WITH CHECK ADD CONSTRAINT [FK\_Records\_Albums] FOREIGN KEY([Album])

REFERENCES [dbo].[Albums] ([Id])

GO

ALTER TABLE [dbo].[Records] CHECK CONSTRAINT [FK\_Records\_Albums]

GO

ALTER TABLE [dbo].[Records] WITH CHECK ADD CONSTRAINT [FK\_Records\_Executors] FOREIGN KEY([Executor])

REFERENCES [dbo].[Executors] ([Id])

GO

ALTER TABLE [dbo].[Records] CHECK CONSTRAINT [FK\_Records\_Executors]

GO

ALTER TABLE [dbo].[Records] WITH CHECK ADD CONSTRAINT [FK\_Records\_Tracks] FOREIGN KEY([Track])

REFERENCES [dbo].[Tracks] ([Id])

GO

ALTER TABLE [dbo].[Records] CHECK CONSTRAINT [FK\_Records\_Tracks]

GO

ALTER TABLE [dbo].[Styles] WITH CHECK ADD CONSTRAINT [FK\_Styles\_Executors] FOREIGN KEY([Executor])

REFERENCES [dbo].[Executors] ([Id])

GO

ALTER TABLE [dbo].[Styles] CHECK CONSTRAINT [FK\_Styles\_Executors]

GO

ALTER TABLE [dbo].[Styles] WITH CHECK ADD CONSTRAINT [FK\_Styles\_Genres] FOREIGN KEY([Genre])

REFERENCES [dbo].[Genres] ([Id])

GO

ALTER TABLE [dbo].[Styles] CHECK CONSTRAINT [FK\_Styles\_Genres]

GO

ALTER TABLE [dbo].[Users] WITH CHECK ADD CONSTRAINT [FK\_Users\_Countries] FOREIGN KEY([Country])

REFERENCES [dbo].[Countries] ([Id])

GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [FK\_Users\_Countries]

GO

ALTER TABLE [dbo].[Users] WITH CHECK ADD CONSTRAINT [FK\_Users\_Images] FOREIGN KEY([Image])

REFERENCES [dbo].[Images] ([Id])

GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [FK\_Users\_Images]

GO

ALTER TABLE [dbo].[Users] WITH CHECK ADD CONSTRAINT [FK\_Users\_Roles] FOREIGN KEY([Role])

REFERENCES [dbo].[Roles] ([Id])

GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [FK\_Users\_Roles]

GO

ALTER TABLE [dbo].[Users] WITH CHECK ADD CONSTRAINT [MorF] CHECK (([Gender]='Мужчина' OR [Gender]='Женщина'))

GO

ALTER TABLE [dbo].[Users] CHECK CONSTRAINT [MorF]

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[AddNewUser] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[AddNewUser]

@login nvarchar(20),

@password nvarchar(20),

@name nvarchar(20),

@surname nvarchar(20),

@gender nvarchar(7),

@country nvarchar(100),

@image int

AS

BEGIN

SET NOCOUNT ON;

INSERT INTO Users VALUES (@login, @password, @name, @surname, @gender, (SELECT Id FROM Countries WHERE Name = @country), @image, (SELECT Id FROM Roles WHERE Name = 'Обычный пользователь'))

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[AddNewUsersUser] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[AddNewUsersUser]

-- Add the parameters for the stored procedure here

@login nvarchar(20),

@password nvarchar(20),

@name nvarchar(20),

@surname nvarchar(20),

@gender nvarchar(7),

@country nvarchar(100),

@image int,

@role nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

INSERT INTO Users VALUES (@login, @password, @name, @surname, @gender, (SELECT Id FROM Countries WHERE Name = @country), @image, (SELECT Id FROM Roles WHERE Name = @role))

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindAlbumPiece] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindAlbumPiece]

-- Add the parameters for the stored procedure here

@album nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

select Albums.Name as Album, YEar(Albums.Date) as Date

from Albums inner join Records on Albums.Id = Album

where Albums.Name LIKE('%' + @album + '%')

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindExecutors] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindExecutors]

-- Add the parameters for the stored procedure here

@login nvarchar(20)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

select Executors.Name as Executor, YEar(Executors.Date) as Date

from Executors inner join Records on Executors.Id = Executor

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindExecutorsInGenre] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindExecutorsInGenre]

-- Add the parameters for the stored procedure here

@name nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT Executors.Name as Executor, Year(Executors.Date) as Date

from Executors inner join Styles on Executors.Id = Executor

inner join Genres on Genres.Id = Genre

where Genres.Name = @name

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindExsInGenrePiece] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindExsInGenrePiece]

-- Add the parameters for the stored procedure here

@genre nvarchar(50),

@executor nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT Executors.Name as Executor, Year(Executors.Date) as Date

from Executors inner join Styles on Executors.Id = Executor

inner join Genres on Genres.Id = Genre

where (Genres.Name = @genre and Executors.Name LIKE ('%' + @executor + '%'))

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindExsPiece] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindExsPiece]

-- Add the parameters for the stored procedure here

@executor nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

select Executors.Name as Executor, YEar(Executors.Date) as Date

from Executors inner join Records on Executors.Id = Executor

where Executors.Name LIKE('%' + @executor + '%')

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindImage] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[FindImage]

@binary varbinary(MAX)

AS

BEGIN

SET NOCOUNT ON;

SELECT Id FROM Images WHERE Binary = @binary

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindLikedExecutors] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[FindLikedExecutors]

@login nvarchar(20)

AS

BEGIN

SET NOCOUNT ON;

select Executors.Name as Executor, YEAR(Executors.Date) as Date

from Executors inner join Records on Executor = Executors.Id

inner join Tracks on Track = Tracks.Id

inner join Likes on Likes.Track = Tracks.Id

inner join Users on Likes.Login = Users.Login

WHERE Users.Login = @login

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindLikedExsPiece] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindLikedExsPiece]

-- Add the parameters for the stored procedure here

@login nvarchar(50),

@executor nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

select Executors.Name as Executor, YEAR(Executors.Date) as Date

from Executors inner join Records on Executor = Executors.Id

inner join Tracks on Track = Tracks.Id

inner join Likes on Likes.Track = Tracks.Id

inner join Users on Likes.Login = Users.Login

WHERE (Users.Login = @login and Executors.Name LIKE('%' + @executor + '%'))

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindLikedTracks] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindLikedTracks]

-- Add the parameters for the stored procedure here

@login nvarchar(20)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT Tracks.Name as Track, Albums.Name as Album, Executors.Name as Executor, Binary

FROM Tracks inner join records on Tracks.Id = Track

inner join albums on Albums.Id = Album

inner join Executors on executors.Id = Executor

INNER JOIN Likes ON Tracks.Id = Likes.Track

inner join Users on Likes.Login = Users.Login

WHERE users.Login = @login

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindLikedTracksPiece] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindLikedTracksPiece]

-- Add the parameters for the stored procedure here

@login nvarchar(20),

@track nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT Tracks.Name as Track, Albums.Name as Album, Executors.Name as Executor, Binary

FROM Tracks inner join records on Tracks.Id = Track

inner join albums on Albums.Id = Album

inner join Executors on executors.Id = Executor

INNER JOIN Likes ON Tracks.Id = Likes.Track

inner join Users on Likes.Login = Users.Login

WHERE (users.Login = @login and tracks.Name LIKE('%' + @track + '%'))

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindRole] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROCEDURE [dbo].[FindRole]

@login nvarchar(20)

AS

BEGIN

SET NOCOUNT ON;

SELECT Roles.Name FROM Roles INNER JOIN Users ON Id = Role WHERE Login = @login

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindTracks] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindTracks]

-- Add the parameters for the stored procedure here

@login nvarchar(20)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

select Tracks.Name as Track, case

when tracks.Id not in (select Track

from Likes inner join users

on users.Login = likes.Login

where users.Login = @login)

then 'False'

else 'True'

end as [Like], Albums.Name as Album, Executors.Name as Executor, Binary

from Likes inner join Users on users.Login = likes.Login

full outer join tracks on tracks.Id = Track

inner join Records on Tracks.Id = Records.Track

inner join Albums on Albums.Id = Album

inner join Executors on Executors.Id = Executor

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindTracksPiece] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindTracksPiece]

-- Add the parameters for the stored procedure here

@login nvarchar(20),

@track nvarchar(50)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

select Tracks.Name as Track, case

when tracks.Id not in (select Track

from Likes inner join users

on users.Login = likes.Login

where users.Login = @login)

then 'False'

else 'True'

end as [Like], Albums.Name as Album, Executors.Name as Executor, Binary

from Likes inner join Users on users.Login = likes.Login

full outer join tracks on tracks.Id = Track

inner join Records on Tracks.Id = Records.Track

inner join Albums on Albums.Id = Album

inner join Executors on Executors.Id = Executor

where Tracks.Name LIKE('%' + @track + '%')

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindUsers] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindUsers]

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT Gender, Login, Password,

Users.Name as Name, Surname, Roles.Name as Role, Countries.Name as Country

FROM Users inner join Roles on Roles.Id = Role

left join Countries on Countries.Id = Country

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[FindUsersPiece] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[FindUsersPiece]

-- Add the parameters for the stored procedure here

@search nvarchar(20)

AS

BEGIN

SELECT Gender, Login, Password,

Users.Name as Name, Surname, Roles.Name as Role, Countries.Name as Country

FROM Users inner join Roles on Roles.Id = Role

left join Countries on Countries.Id = Country

WHERE Users.Name Like('%' + @search + '%') or Login Like('%' + @search + '%')

or Surname Like('%' + @search + '%') or Gender Like('%' + @search + '%')

or Countries.Name Like('%' + @search + '%') or Roles.Name Like('%' + @search + '%')

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[GetInfos] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[GetInfos]

-- Add the parameters for the stored procedure here

@login nvarchar(20)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

SELECT Login, Password, Users.Name, Surname, Gender, Countries.Name, Images.Binary, Roles.Name

FROM Users INNER JOIN Countries

ON Countries.Id = Country

LEFT JOIN Images

ON Images.Id = Image

LEFT JOIN Roles

ON Roles.Id = Role

WHERE Login = @login

END

GO

/\*\*\*\*\*\* Object: StoredProcedure [dbo].[UpdateUser] Script Date: 21.10.2020 0:24:22 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

-- =============================================

-- Author: <Author,,Name>

-- Create date: <Create Date,,>

-- Description: <Description,,>

-- =============================================

CREATE PROCEDURE [dbo].[UpdateUser]

-- Add the parameters for the stored procedure here

@login nvarchar(20),

@password nvarchar(20),

@name nvarchar(20),

@surname nvarchar(20),

@gender nvarchar(7),

@country nvarchar(100),

@image int,

@role nvarchar(30),

@search nvarchar(20)

AS

BEGIN

-- SET NOCOUNT ON added to prevent extra result sets from

-- interfering with SELECT statements.

SET NOCOUNT ON;

-- Insert statements for procedure here

UPDATE Users SET Login = @login, Password = @password, Users.Name = @name,

Users.Surname = @surname, Users.Gender = @gender, Users.Country = (select Id from Countries where Countries.Name = @country),

Users.Image = @image, Users.Role = (select id from Roles where Roles.Name = @role)

WHERE Login = @search

END

GO

USE [master]

GO

ALTER DATABASE [KP] SET READ\_WRITE

GO

# ПРИЛОЖЕНИЕ Б. Код приложения

<Page x:Class="KP.Pages.Auth"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="792"

Title="Auth">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="230"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Style="{StaticResource AuthBlock}" Text="Логин:" Margin="10,0,0,103" HorizontalAlignment="Left"/>

<TextBox Style="{StaticResource AuthBox}" x:Name="login" Margin="60,0,10,105"/>

<TextBlock Style="{StaticResource AuthBlock}" Text="Пароль:" Margin="0,0,0,55" HorizontalAlignment="Left"/>

<PasswordBox Style="{StaticResource AuthPass}" x:Name="password" Margin="60,0,0,55"/>

<TextBlock Style="{StaticResource AuthBlock}" x:Name="captcha" Margin="0,0,129,0" TextAlignment="Center" FontFamily="Mistral" FontSize="20" Visibility="Hidden"/>

<TextBox Style="{StaticResource AuthBox}" x:Name="verify" Margin="103,0,26,0" Visibility="Hidden"/>

<Button Style="{StaticResource AuthBtn}" x:Name="refresh" Content="⭮" Click="refresh\_Click" Margin="205,0,0,0" Visibility="Hidden"/>

<Button Style="{StaticResource AuthBtn}" x:Name="register" Content="Регистрация" Click="register\_Click" Margin="0,55,86,0"/>

<Button Style="{StaticResource AuthBtn}" x:Name="enter" Content="Вход" Click="enter\_Click" Margin="144,55,0,0"/>

</Grid>

</Page>

using System;

using System.Configuration;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Navigation;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Auth.xaml

/// </summary>

public partial class Auth : Page

{

public Auth()

{

InitializeComponent();

}

/// <summary>

/// Метод перехода на форму регистрации по нажатию кнопки "Зарегистрироваться"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void register\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Navigate(new Uri("Pages/Reg.xaml", UriKind.RelativeOrAbsolute));

}

/// <summary>

/// Метод обновления капчи по нажатию кнопки "Обновить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void refresh\_Click(object sender, RoutedEventArgs e)

{

Classes.Captcha.Refresh(captcha, verify);

}

private byte errorCount = 0; //Количество ошибок ввода

private bool errorMessage = false; //Показ/скрытие капчи

/// <summary>

/// Метод перехода на страницу пользователя

/// </summary>

/// <param name="reader"></param>

private void EnterPage(SqlDataReader reader)

{

while (reader.Read()) Classes.Login.Value = login.Text;

NavigationService.Navigate(new Uri("Pages/User.xaml", UriKind.RelativeOrAbsolute));

login.Text = "";

errorMessage = false;

errorCount = 0;

Classes.Captcha.Check(captcha, verify, refresh, false);

}

/// <summary>

/// Метод авторизации по нажатию на кнопку "Войти"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void enter\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT Login, Password FROM Users WHERE Login = @login AND Password = @password";

command.Parameters.AddWithValue("@login", login.Text);

command.Parameters.AddWithValue("@password", password.Password);

if (!string.IsNullOrEmpty(login.Text) && !string.IsNullOrEmpty(password.Password))

{

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows && errorCount < 3) EnterPage(reader);

else if (reader.HasRows && errorCount == 3)

{

if (verify.Text == captcha.Text) EnterPage(reader);

else

{

MessageBox.Show("Неверное значение Captcha!", "Ошибка ввода Captcha", MessageBoxButton.OK, MessageBoxImage.Error);

Classes.Captcha.Check(captcha, verify, refresh, false);

}

}

else if (!reader.HasRows && errorCount < 3)

{

MessageBox.Show("Пользователя не существует, или\nневерно указаны логин или пароль!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

errorCount++;

}

else if (!reader.HasRows && errorCount == 3)

{

if (errorMessage == false)

{

MessageBox.Show("Превышен лимит ввода неверных данных!\nВ следующий раз для подтверждения введите Captcha!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

Classes.Captcha.Check(captcha, verify, refresh, true);

errorMessage = true;

}

else if (verify.Text != captcha.Text)

MessageBox.Show("Неверное значение Captcha!", "Ошибка ввода Captcha", MessageBoxButton.OK, MessageBoxImage.Error);

else if (verify.Text == captcha.Text)

MessageBox.Show("Пользователя не существует, или\nневерно указаны логин или пароль!", "Ошибка ввода", MessageBoxButton.OK, MessageBoxImage.Error);

Classes.Captcha.Check(captcha, verify, refresh, false);

}

}

}

else MessageBox.Show("Введите логин и пароль!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Exclamation);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

}

}

<Page x:Class="KP.Pages.Countries"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Countries">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮" Click="updatePage\_Click"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn" Click="searchBtn\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="countriesGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Country}" Header="Страна"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="countOfCountries" Header="Количество стран" Click="countOfCountries\_Click"/>

<MenuItem x:Name="add" Header="Добавить" Click="add\_Click"/>

<MenuItem x:Name="edit" Header="Редактировать" Click="edit\_Click"/>

<MenuItem x:Name="delete" Header="Удалить" Click="delete\_Click"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Countries.xaml

/// </summary>

public partial class Countries : Page

{

public Countries()

{

InitializeComponent();

Classes.Get.TableOutput(countriesGrid, null, 9);

}

private void searchBtn\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Countries WHERE Name LIKE('%' + @name + '%')";

command.Parameters.AddWithValue("@name", searchBox.Text);

Classes.Get.FillGrid(command, countriesGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

searchBox.Text = "Поиск";

connection.Close();

}

}

}

private void delete\_Click(object sender, RoutedEventArgs e)

{

if (countriesGrid.SelectedItem != null)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить данную запись?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Countries WHERE Name = @name";

command.Parameters.AddWithValue("@name", ((DataRowView)countriesGrid.SelectedItem)[0]);

command.ExecuteNonQuery();

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

Classes.Get.TableOutput(countriesGrid, null, 9);

}

}

}

}

else

{

MessageBox.Show("Удалить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

private void updatePage\_Click(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(countriesGrid, null, 9);

searchBox.Text = "Поиск";

}

private void add\_Click(object sender, RoutedEventArgs e)

{

Windows.Country window = new Windows.Country(false, null, countriesGrid);

window.ShowDialog();

}

private void edit\_Click(object sender, RoutedEventArgs e)

{

if (countriesGrid.SelectedItem != null)

{

Windows.Country window = new Windows.Country(true, ((DataRowView)countriesGrid.SelectedItem)[0], countriesGrid);

window.ShowDialog();

}

else

{

MessageBox.Show("Изменить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

private void countOfCountries\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT COUNT(Id) FROM Countries";

MessageBox.Show("Количество стран: " + command.ExecuteScalar().ToString());

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

searchBox.Text = "Поиск";

}

}

}

}

}

<Page x:Class="KP.Pages.EditUser"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="EditUser">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="246"/>

<ColumnDefinition Width="246"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Button Style="{StaticResource RegBtn}" x:Name="back" Content="Назад" Click="back\_Click" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10,10,0,0" Width="60"/>

<Button Style="{StaticResource RegBtn}" x:Name="update" Content="⭮" Click="update\_Click" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,10,10,0" Grid.Column="3" Width="25"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Логин:" Margin="0,0,174,240"/>

<TextBox Style="{StaticResource RegBox}" x:Name="login" Margin="72,0,0,240"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Пароль:" Margin="0,0,174,195"/>

<PasswordBox Style="{StaticResource RegPass}" x:Name="password" Margin="72,0,0,195"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Повтор:" Margin="0,0,174,150"/>

<PasswordBox Style="{StaticResource RegPass}" x:Name="repeat" Margin="72,0,0,150"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Имя:" Margin="0,0,174,60"/>

<TextBox Style="{StaticResource RegBox}" x:Name="name" Margin="72,0,0,60"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Фамилия:" Margin="0,0,174,15"/>

<TextBox Style="{StaticResource RegBox}" x:Name="surname" Margin="72,0,0,15"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Пол:" Margin="0,60,174,0"/>

<ComboBox Style="{StaticResource RegCmbox}" x:Name="gender" Margin="72,60,0,0"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Страна:" Margin="0,105,174,0"/>

<ComboBox Style="{StaticResource RegCmbox}" x:Name="country" IsEditable="True" Margin="72,105,0,0"/>

<Image x:Name="image" PreviewMouseLeftButtonUp="image\_PreviewMouseLeftButtonUp" Height="200" Grid.Column="2" Width="200" Margin="0,0,0,90" Source="/KP;component/Resources/addImage.png"/>

<TextBlock Style="{StaticResource RegBlock}" x:Name="captcha" Margin="10,170,135,0" TextAlignment="Center" LineHeight="20"/>

<TextBox Style="{StaticResource RegBox}" x:Name="verify" Margin="114,170,31,0"/>

<Button Style="{StaticResource RegBtn}" x:Name="refresh" Content="⭮" Click="refresh\_Click" Grid.Column="1" Width="25" Margin="210,170,0,0"/>

<Button Style="{StaticResource RegBtn}" x:Name="updateUser" Content="Изменить" Grid.Column="2" Margin="0,290,0,0" Width="170"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Роль:" Margin="0,170,174,0" Grid.Column="2"/>

<ComboBox Style="{StaticResource RegCmbox}" x:Name="role" Margin="72,170,0,0" Grid.Column="2"/>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для EditUser.xaml

/// </summary>

public partial class EditUser : Page

{

BitmapImage bitmap = new BitmapImage();

public EditUser(object obj)

{

InitializeComponent();

gender.Items.Add("Мужчина");

gender.Items.Add("Женщина");

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT Name FROM Roles";

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

role.Items.Add(reader[0].ToString());

}

}

}

}

Classes.Get.Countries(connection, country);

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC GetInfos @login";

command.Parameters.AddWithValue("@login", obj);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

login.Text = reader[0].ToString();

password.Password = reader[1].ToString();

name.Text = reader[2].ToString();

surname.Text = reader[3].ToString();

gender.SelectedItem = reader[4].ToString();

country.SelectedItem = reader[5].ToString();

if (reader[6] != DBNull.Value)

{

bitmap.BeginInit();

bitmap.StreamSource = new MemoryStream((byte[])reader[6]);

bitmap.EndInit();

}

else

{

bitmap.BeginInit();

bitmap.UriSource = new Uri("/KP;component/Resources/addImage.png", UriKind.RelativeOrAbsolute);

bitmap.EndInit();

}

role.SelectedItem = reader[7].ToString();

}

image.Source = bitmap;

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

if (obj.ToString() == Classes.Login.Value)

{

role.IsEnabled = false;

repeat.Password = password.Password;

password.IsEnabled = false;

repeat.IsEnabled = false;

}

Classes.Captcha.Refresh(captcha, verify);

updateUser.Click += (s, e) =>

{

UpdateUser(obj);

if (obj.ToString() == Classes.Login.Value)

{

role.IsEnabled = false;

repeat.Password = password.Password;

password.IsEnabled = false;

repeat.IsEnabled = false;

}

};

}

private void UpdateUser(object obj)

{

if (login.Text != "" && password.Password != "" && repeat.Password != "" && gender.SelectedItem != null && (surname.Text != "" || name.Text != ""))

{

if (login.Text.Length > 2 && login.Text.Length <= 20)

{

if (password.Password.Length > 4 && password.Password.Length <= 20)

{

if (name.Text.Length > 0 && name.Text.Length <= 20)

{

if (surname.Text.Length > 0 && surname.Text.Length <= 20)

{

if (repeat.Password == password.Password)

{

if (verify.Text == captcha.Text)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC UpdateUser @login, @password, @name, @surname, @gender, @country, @image, @role, @search";

command.Parameters.AddWithValue("@search", obj);

command.Parameters.AddWithValue("@login", login.Text);

command.Parameters.AddWithValue("@password", password.Password);

command.Parameters.AddWithValue("@surname", surname.Text);

command.Parameters.AddWithValue("@name", name.Text);

command.Parameters.AddWithValue("@gender", gender.SelectedItem);

if (country.SelectedIndex != -1) command.Parameters.AddWithValue("@country", country.SelectedItem);

else command.Parameters.AddWithValue("@country", DBNull.Value);

if (image != null && file != null)

{

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "EXEC FindImage @binary";

command1.Parameters.AddWithValue("@binary", Classes.Get.BytesFromFile(file.FullName));

using (SqlDataReader reader = command1.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read()) command.Parameters.AddWithValue("@image", reader[0]);

}

else

{

reader.Close();

Classes.Add.NewImage(connection, file, command);

}

}

}

}

else if (image != null && file == null)

{

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "SELECT Image FROM Users WHERE Login = @login";

command1.Parameters.AddWithValue("@login", obj);

using (SqlDataReader reader = command1.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read()) command.Parameters.AddWithValue("@image", reader[0]);

}

else

{

reader.Close();

command.Parameters.AddWithValue("@image", DBNull.Value);

}

}

}

}

else if (image == null && file == null) command.Parameters.AddWithValue("@image", DBNull.Value);

command.Parameters.AddWithValue("@role", role.SelectedItem);

command.ExecuteNonQuery();

}

MessageBox.Show("Данные успешно изменены", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

NavigationService.GoBack();

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

else

{

MessageBox.Show("Неверное значения Captcha!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Введенные пароли не совпадают!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Фамилия должна быть в диапазоне от 1 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

surname.Text = "";

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Имя должно быть в диапазоне от 1 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

name.Text = "";

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Пароль должен быть в диапазоне от 5 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Логин должен быть в диапазоне от 3 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

login.Text = "";

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Заполните все необходимые данные!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

private void back\_Click(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

private void update\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Refresh();

}

private FileInfo file; //Для передачи пути изображения

/// <summary>

/// Метод вставки изображения из файла в форму регистрации после отжатия ЛКМ на форме изображения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void image\_PreviewMouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

image.Source = Classes.Set.ImageFromFile(out file);

}

/// <summary>

/// Метод обновления капчи по нажатию кнопки "Обновить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void refresh\_Click(object sender, RoutedEventArgs e)

{

Classes.Captcha.Refresh(captcha, verify);

}

}

}

<Page x:Class="KP.Pages.Executors"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Executors">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮" Click="updatePage\_Click"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn" Click="searchBtn\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="executorsGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Executor}" Header="Исполнитель"/>

<DataGridTextColumn Binding="{Binding Date}" Header="Год основания"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Executors.xaml

/// </summary>

public partial class Executors : Page

{

public Executors()

{

InitializeComponent();

Classes.Get.TableOutput(executorsGrid, null, 4);

}

private void updatePage\_Click(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(executorsGrid, null, 4);

searchBox.Text = "Поиск";

}

private void searchBtn\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindLikedExsPiece @login, @executor";

command.Parameters.AddWithValue("@executor", searchBox.Text);

command.Parameters.AddWithValue("@login", Classes.Login.Value);

Classes.Get.FillGrid(command, executorsGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

searchBox.Text = "Поиск";

connection.Close();

}

}

}

}

}

<Page x:Class="KP.Pages.ExecutorsInGenre"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="ExecutorsInGenre">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn"/>

<DataGrid AutoGenerateColumns="False" x:Name="exsInGenreGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Executor}" Header="Исполнитель"/>

<DataGridTextColumn Binding="{Binding Date}" Header="Год основания"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="add" Header="Добавить"/>

<MenuItem x:Name="edit" Header="Редактировать"/>

<MenuItem x:Name="delete" Header="Удалить"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для ExecutorsInGenre.xaml

/// </summary>

public partial class ExecutorsInGenre : Page

{

public ExecutorsInGenre(object obj)

{

InitializeComponent();

Classes.Get.TableOutput(exsInGenreGrid, obj, 7);

updatePage.Click += (s, e) =>

{

Classes.Get.TableOutput(exsInGenreGrid, obj, 7);

searchBox.Text = "Поиск";

};

searchBtn.Click += (s, e) =>

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindExsInGenrePiece @genre, @executor";

command.Parameters.AddWithValue("@executor", searchBox.Text);

command.Parameters.AddWithValue("@genre", obj);

Classes.Get.FillGrid(command, exsInGenreGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

searchBox.Text = "Поиск";

connection.Close();

}

}

};

}

}

}

<Page x:Class="KP.Pages.Genres"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Genres">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮" Click="updatePage\_Click"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn" Click="searchBtn\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="genresGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Name}" Header="Жанр"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="showExecutors" Header="Показать исполнителей в этом жанре" Click="showExecutors\_Click"/>

<MenuItem x:Name="countOfGenres" Header="Количество жанров" Click="countOfGenres\_Click"/>

<MenuItem x:Name="add" Header="Добавить" Click="add\_Click"/>

<MenuItem x:Name="edit" Header="Редактировать" Click="edit\_Click"/>

<MenuItem x:Name="delete" Header="Удалить" Click="delete\_Click"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Genres.xaml

/// </summary>

public partial class Genres : Page

{

public Genres()

{

InitializeComponent();

Classes.Get.TableOutput(genresGrid, null, 2);

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT Roles.Name FROM Roles INNER JOIN Users ON Role = Id WHERE Login = @login";

command.Parameters.AddWithValue("@login", Classes.Login.Value);

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

if (reader[0].ToString() == "Обычный пользователь")

{

add.Visibility = Visibility.Hidden;

edit.Visibility = Visibility.Hidden;

delete.Visibility = Visibility.Hidden;

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

connection.Close();

}

}

}

/// <summary>

/// Метод поиска сопадений в таблице жанров по слову или части слова в поисковой строке по нажатию кнопки "Искать"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void searchBtn\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Genres WHERE Name LIKE('%' + @name + '%')";

command.Parameters.AddWithValue("@name", searchBox.Text);

Classes.Get.FillGrid(command, genresGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

searchBox.Text = "Поиск";

connection.Close();

}

}

}

/// <summary>

/// Метод обновления страницы по нажатию кнопки "Обновить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void updatePage\_Click(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(genresGrid, null, 2);

searchBox.Text = "Поиск";

}

/// <summary>

/// Метод редактирования выделенной записи по нажатию кнопки меню ЛКМ "Редактировать"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void edit\_Click(object sender, RoutedEventArgs e)

{

if (genresGrid.SelectedItem != null)

{

Windows.Genre window = new Windows.Genre(true, ((DataRowView)genresGrid.SelectedItem)[1], genresGrid);

window.ShowDialog();

}

else

{

MessageBox.Show("Изменить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

/// <summary>

/// Метод удаления выделенной записи по нажатию кнопки меню ЛКМ "Удалить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void delete\_Click(object sender, RoutedEventArgs e)

{

if (genresGrid.SelectedItem != null)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить данную запись?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Genres WHERE Name = @name";

command.Parameters.AddWithValue("@name", ((DataRowView)genresGrid.SelectedItem)[1]);

command.ExecuteNonQuery();

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

Classes.Get.TableOutput(genresGrid, null, 2);

}

}

}

}

else

{

MessageBox.Show("Удалить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

/// <summary>

/// Метод добваления новой записи по нажатию кнопки меню ЛКМ "Добавить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void add\_Click(object sender, RoutedEventArgs e)

{

Windows.Genre window = new Windows.Genre(false, null, genresGrid);

window.ShowDialog();

}

private void showExecutors\_Click(object sender, RoutedEventArgs e)

{

if (genresGrid.SelectedItem != null)

{

NavigationService.Navigate(new ExecutorsInGenre(((DataRowView)genresGrid.SelectedItem)[1]));

}

else

{

MessageBox.Show("Перейти можно только по выделенной записи", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

private void countOfGenres\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT COUNT(Id) FROM Genres";

MessageBox.Show("Количество жанров: " + command.ExecuteScalar().ToString());

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

searchBox.Text = "Поиск";

}

}

}

}

}

<Page x:Class="KP.Pages.Main"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Main">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

<RowDefinition/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮" Click="updatePage\_Click"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn" Click="searchBtn\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="tracksGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Track}" Header="Трек"/>

<DataGridTextColumn Binding="{Binding Album}" Header="Альбом"/>

<DataGridTextColumn Binding="{Binding Executor}" Header="Исполнитель"/>

<DataGridCheckBoxColumn Binding="{Binding Like}" Header="Нравится"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="like" Header="В Избранное"/>

<MenuItem x:Name="add" Header="Добавить"/>

<MenuItem x:Name="edit" Header="Редактировать"/>

<MenuItem x:Name="delete" Header="Удалить"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

<DataGrid AutoGenerateColumns="False" x:Name="albumsGrid" Grid.Row="2">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Album}" Header="Альбом"/>

<DataGridTextColumn Binding="{Binding Date}" Header="Год релиза"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="edit2" Header="Редактировать"/>

<MenuItem x:Name="delete2" Header="Удалить"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

<DataGrid AutoGenerateColumns="False" x:Name="executorsGrid" Grid.Row="3">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Executor}" Header="Исполнитель"/>

<DataGridTextColumn Binding="{Binding Country}" Header="Страна"/>

<DataGridTextColumn Binding="{Binding Date}" Header="Год основания"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="edit1" Header="Редактировать"/>

<MenuItem x:Name="delete1" Header="Удалить"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Main.xaml

/// </summary>

public partial class Main : Page

{

public Main()

{

InitializeComponent();

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT Roles.Name FROM Roles INNER JOIN Users ON Role = Id WHERE Login = @login";

command.Parameters.AddWithValue("@login", Classes.Login.Value);

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

if (reader[0].ToString() == "Обычный пользователь")

{

add.Visibility = Visibility.Hidden;

edit.Visibility = Visibility.Hidden;

delete.Visibility = Visibility.Hidden;

edit1.Visibility = Visibility.Hidden;

delete1.Visibility = Visibility.Hidden;

edit2.Visibility = Visibility.Hidden;

delete2.Visibility = Visibility.Hidden;

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

connection.Close();

}

}

Classes.Get.TableOutput(tracksGrid, null, 0);

Classes.Get.TableOutput(executorsGrid, null, 1);

Classes.Get.TableOutput(albumsGrid, null, 10);

edit.Click += (s, e) =>

{

};

edit1.Click += (s, e) =>

{

Windows.Executor executor = new Windows.Executor(true, executorsGrid);

executor.ShowDialog();

};

like.Click += (s, e) =>

{

if (tracksGrid.SelectedItem != null)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "INSERT INTO Likes VALUES ((SELECT Login FROM Users WHERE Login = @login), (SELECT Id FROM Tracks WHERE Tracks.Name = @track))";

command.Parameters.AddWithValue("@login", Classes.Login.Value);

command.Parameters.AddWithValue("@track", ((DataRowView)tracksGrid.SelectedItem)[0]);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

searchBox.Text = "Поиск";

connection.Close();

}

}

}

else

{

MessageBox.Show("Добавить в избранное можно только выделенную запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

};

delete1.Click += (s, e) =>

{

if (executorsGrid.SelectedItem != null)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить данную запись?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Records WHERE Records.Executor = (SELECT Id FROM Executors WHERE Executors.Name = @executor)";

command.Parameters.AddWithValue("@executor", ((DataRowView)executorsGrid.SelectedItem)[0]);

Classes.Get.FillGrid(command, tracksGrid);

Classes.Get.FillGrid(command, executorsGrid);

Classes.Get.FillGrid(command, albumsGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

Classes.Get.TableOutput(tracksGrid, null, 0);

Classes.Get.TableOutput(executorsGrid, null, 1);

Classes.Get.TableOutput(albumsGrid, null, 10);

searchBox.Text = "Поиск";

connection.Close();

}

}

}

}

else

{

MessageBox.Show("Удалить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

};

delete2.Click += (s, e) =>

{

if (albumsGrid.SelectedItem != null)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить данную запись?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Records WHERE Records.Album = (SELECT Id FROM Albums WHERE Albums.Name = @album)";

command.Parameters.AddWithValue("@album", ((DataRowView)executorsGrid.SelectedItem)[0]);

Classes.Get.FillGrid(command, tracksGrid);

Classes.Get.FillGrid(command, executorsGrid);

Classes.Get.FillGrid(command, albumsGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

Classes.Get.TableOutput(tracksGrid, null, 0);

Classes.Get.TableOutput(executorsGrid, null, 1);

Classes.Get.TableOutput(albumsGrid, null, 10);

searchBox.Text = "Поиск";

connection.Close();

}

}

}

}

else

{

MessageBox.Show("Удалить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

};

}

private void updatePage\_Click(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(tracksGrid, null, 0);

Classes.Get.TableOutput(executorsGrid, null, 1);

Classes.Get.TableOutput(albumsGrid, null, 10);

searchBox.Text = "Поиск";

}

private void searchBtn\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindTracksPiece @login, @track";

command.Parameters.AddWithValue("@login", Classes.Login.Value);

command.Parameters.AddWithValue("@track", searchBox.Text);

Classes.Get.FillGrid(command, tracksGrid);

}

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindExsPiece @executor";

command.Parameters.AddWithValue("@executor", searchBox.Text);

Classes.Get.FillGrid(command, executorsGrid);

}

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindAlbumPiece @album";

command.Parameters.AddWithValue("@album", searchBox.Text);

Classes.Get.FillGrid(command, albumsGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

searchBox.Text = "Поиск";

connection.Close();

}

}

}

}

}

<Page x:Class="KP.Pages.Profile"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Profile">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="246"/>

<ColumnDefinition Width="246"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Style="{StaticResource RegBlock}" Text="Логин:" Margin="0,0,174,240"/>

<TextBlock Style="{StaticResource RegBlock}" x:Name="login" Margin="72,0,0,240"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Имя:" Margin="0,0,174,140"/>

<TextBlock Style="{StaticResource RegBlock}" x:Name="name" Margin="72,0,0,140"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Фамилия:" Margin="0,0,174,95"/>

<TextBlock Style="{StaticResource RegBlock}" x:Name="surname" Margin="72,0,0,95"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Пол:" Margin="0,0,174,0"/>

<TextBlock Style="{StaticResource RegBlock}" x:Name="gender" Margin="72,0,0,0"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Страна:" Margin="0,45,174,0"/>

<TextBlock Style="{StaticResource RegBlock}" x:Name="country" Margin="72,45,0,0"/>

<Image x:Name="image" Height="200" Width="200" Margin="0,0,0,90" Grid.Column="2"/>

<Button Style="{StaticResource RegBtn}" x:Name="changeInfos" Content="Изменить данные" Grid.Column="1" Width="180" Margin="0,140,0,0"/>

<Button Style="{StaticResource RegBtn}" x:Name="changePw" Content="Изменить пароль" Grid.Column="2" Width="180" Margin="0,140,0,0"/>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Profile.xaml

/// </summary>

public partial class Profile : Page

{

public Profile(object user)

{

InitializeComponent();

this.Loaded += (s, e) =>

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC GetInfos @login";

command.Parameters.AddWithValue("@login", user);

using (SqlDataReader reader = command.ExecuteReader())

{

BitmapImage bitmap = new BitmapImage();

while (reader.Read())

{

login.Text = reader[0].ToString();

name.Text = reader[2].ToString();

surname.Text = reader[3].ToString();

gender.Text = reader[4].ToString();

country.Text = reader[5].ToString();

if (reader[6] != DBNull.Value)

{

bitmap.BeginInit();

bitmap.StreamSource = new MemoryStream((byte[])reader[6]);

bitmap.EndInit();

}

else

{

bitmap.BeginInit();

bitmap.UriSource = new Uri("/KP;component/Resources/addImage.png", UriKind.RelativeOrAbsolute);

bitmap.EndInit();

}

}

image.Source = bitmap;

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

};

changeInfos.Click += (s, e) =>

{

NavigationService.Navigate(new EditUser(user));

};

changePw.Click += (s, e) =>

{

Windows.Password password = new Windows.Password(user);

password.ShowDialog();

};

}

}

}

<Page x:Class="KP.Pages.Reg"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="792"

Title="Reg">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="246"/>

<ColumnDefinition Width="246"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Button Style="{StaticResource RegBtn}" x:Name="back" Content="Назад" Click="back\_Click" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10,10,0,0" Width="60"/>

<Button Style="{StaticResource RegBtn}" x:Name="update" Content="⭮" Click="update\_Click" HorizontalAlignment="Right" VerticalAlignment="Top" Margin="0,10,10,0" Grid.Column="3" Width="25"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Логин:" Margin="0,0,174,240"/>

<TextBox Style="{StaticResource RegBox}" x:Name="login" Margin="72,0,0,240"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Пароль:" Margin="0,0,174,195"/>

<PasswordBox Style="{StaticResource RegPass}" x:Name="password" Margin="72,0,0,195"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Повтор:" Margin="0,0,174,150"/>

<PasswordBox Style="{StaticResource RegPass}" x:Name="repeat" Margin="72,0,0,150"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Имя:" Margin="0,0,174,60"/>

<TextBox Style="{StaticResource RegBox}" x:Name="name" Margin="72,0,0,60"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Фамилия:" Margin="0,0,174,15"/>

<TextBox Style="{StaticResource RegBox}" x:Name="surname" Margin="72,0,0,15"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Пол:" Margin="0,60,174,0"/>

<ComboBox Style="{StaticResource RegCmbox}" x:Name="gender" Margin="72,60,0,0"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Страна:" Margin="0,105,174,0"/>

<ComboBox Style="{StaticResource RegCmbox}" x:Name="country" IsEditable="True" Margin="72,105,0,0"/>

<Image x:Name="image" PreviewMouseLeftButtonUp="image\_PreviewMouseLeftButtonUp" Height="200" Grid.Column="2" Width="200" Margin="0,0,0,90" Source="/KP;component/Resources/addImage.png"/>

<TextBlock Style="{StaticResource RegBlock}" x:Name="captcha" Margin="10,170,135,0" TextAlignment="Center" LineHeight="20"/>

<TextBox Style="{StaticResource RegBox}" x:Name="verify" Margin="114,170,31,0"/>

<Button Style="{StaticResource RegBtn}" x:Name="refresh" Content="⭮" Click="refresh\_Click" Grid.Column="1" Width="25" Margin="210,170,0,0"/>

<Button Style="{StaticResource RegBtn}" x:Name="register" Content="Добавить" Click="register\_Click" Grid.Column="2" Margin="0,290,0,0" Width="170"/>

</Grid>

</Page>

using System;

using System.Configuration;

using System.Data.SqlClient;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

using System.Windows.Navigation;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Reg.xaml

/// </summary>

public partial class Reg : Page

{

public Reg()

{

InitializeComponent();

gender.Items.Add("Мужчина");

gender.Items.Add("Женщина");

Classes.Captcha.Refresh(captcha, verify);

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

Classes.Get.Countries(connection, country);

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally

{

connection.Close();

}

}

}

/// <summary>

/// Метод возврата на страницу авторизации по нажатию кнопки "Назад"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void back\_Click(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

/// <summary>

/// Метод обновления капчи по нажатию кнопки "Обновить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void refresh\_Click(object sender, RoutedEventArgs e)

{

Classes.Captcha.Refresh(captcha, verify);

}

private FileInfo file; //Для передачи пути изображения

/// <summary>

/// Метод регистрации пользователя понажатию кнопки "Зарегистрироваться"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void register\_Click(object sender, RoutedEventArgs e)

{

if (login.Text != "" && password.Password != "" && repeat.Password != "" && gender.SelectedItem != null && (surname.Text != "" || name.Text != ""))

{

if (login.Text.Length > 2 && login.Text.Length <= 20)

{

if (password.Password.Length > 4 && password.Password.Length <= 20)

{

if (name.Text.Length > 0 && name.Text.Length <= 20)

{

if (surname.Text.Length > 0 && surname.Text.Length <= 20)

{

if (repeat.Password == password.Password)

{

if (verify.Text == captcha.Text)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC AddNewUser @login, @password, @name, @surname, @gender, @country, @image";

command.Parameters.AddWithValue("@login", login.Text);

command.Parameters.AddWithValue("@password", password.Password);

command.Parameters.AddWithValue("@surname", surname.Text);

command.Parameters.AddWithValue("@name", name.Text);

command.Parameters.AddWithValue("@gender", gender.SelectedItem);

if (country.SelectedIndex != -1) command.Parameters.AddWithValue("@country", country.SelectedItem);

else command.Parameters.AddWithValue("@country", DBNull.Value);

if (file != null)

{

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "EXEC FindImage @binary";

command1.Parameters.AddWithValue("@binary", Classes.Get.BytesFromFile(file.FullName));

using (SqlDataReader reader = command1.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read()) command.Parameters.AddWithValue("@image", reader[0]);

}

else

{

reader.Close();

Classes.Add.NewImage(connection, file, command);

}

}

}

}

else command.Parameters.AddWithValue("@image", DBNull.Value);

command.ExecuteNonQuery();

}

MessageBox.Show("Пользователь успешно зарегистрирован!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

NavigationService.GoBack();

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

else

{

MessageBox.Show("Неверное значения Captcha!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Введенные пароли не совпадают!", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Фамилия должна быть в диапазоне от 1 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

surname.Text = "";

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Имя должно быть в диапазоне от 1 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

name.Text = "";

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Пароль должен быть в диапазоне от 5 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Логин должен быть в диапазоне от 3 до 20 символов!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

login.Text = "";

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

else

{

MessageBox.Show("Заполните все необходимые данные!", "Уведомление", MessageBoxButton.OK, MessageBoxImage.Information);

password.Password = "";

repeat.Password = "";

Classes.Captcha.Refresh(captcha, verify);

}

}

/// <summary>

/// Метод обновления страницы по нажатию кнопки "Обновить страницу"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void update\_Click(object sender, RoutedEventArgs e)

{

NavigationService.Refresh();

}

/// <summary>

/// Метод вставки изображения из файла в форму регистрации после отжатия ЛКМ на форме изображения

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void image\_PreviewMouseLeftButtonUp(object sender, MouseButtonEventArgs e)

{

image.Source = Classes.Set.ImageFromFile(out file);

}

}

}

<Page x:Class="KP.Pages.Roles"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Roles">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮" Click="updatePage\_Click"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn" Click="searchBtn\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="rolesGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Name}" Header="Роль"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="usersInRole" Header="Вывести пользователей в этой роли" Click="usersInRole\_Click"/>

<MenuItem x:Name="countOfRoles" Header="Число ролей" Click="countOfRoles\_Click"/>

<MenuItem x:Name="add" Header="Добавить" Click="add\_Click"/>

<MenuItem x:Name="edit" Header="Редактировать" Click="edit\_Click"/>

<MenuItem x:Name="delete" Header="Удалить" Click="delete\_Click"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Roles.xaml

/// </summary>

public partial class Roles : Page

{

public Roles()

{

InitializeComponent();

Classes.Get.TableOutput(rolesGrid, null, 5);

}

/// <summary>

/// Метод поиска сопадений в таблице ролей по слову или части слова в поисковой строке по нажатию кнопки "Искать"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void searchBtn\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Roles WHERE Name LIKE('%' + @name + '%')";

command.Parameters.AddWithValue("@name", searchBox.Text);

Classes.Get.FillGrid(command, rolesGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

searchBox.Text = "Поиск";

}

}

}

/// <summary>

/// Метод обновления страницы по нажатию кнопки "Обновить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void updatePage\_Click(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(rolesGrid, null, 5);

searchBox.Text = "Поиск";

}

/// <summary>

/// Метод редактирования выделенной записи по нажатию кнопки меню ЛКМ "Редактировать"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void edit\_Click(object sender, RoutedEventArgs e)

{

if (rolesGrid.SelectedItem != null)

{

Windows.Role window = new Windows.Role(true, ((DataRowView)rolesGrid.SelectedItem)[1], rolesGrid);

window.ShowDialog();

}

else

{

MessageBox.Show("Изменить можно только существующую роль", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

/// <summary>

/// Метод удаления выделенной записи по нажатию кнопки меню ЛКМ "Удалить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void delete\_Click(object sender, RoutedEventArgs e)

{

if (rolesGrid.SelectedItem != null)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить данную запись?", "Уведомление",MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Roles WHERE Name = @name";

command.Parameters.AddWithValue("@name", ((DataRowView)rolesGrid.SelectedItem)[1]);

command.ExecuteNonQuery();

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

Classes.Get.TableOutput(rolesGrid, null, 5);

}

}

}

}

else

{

MessageBox.Show("Удалить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

/// <summary>

/// Метод добваления новой записи по нажатию кнопки меню ЛКМ "Добавить"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void add\_Click(object sender, RoutedEventArgs e)

{

Windows.Role window = new Windows.Role(false, null, rolesGrid);

window.ShowDialog();

}

private void usersInRole\_Click(object sender, RoutedEventArgs e)

{

if (rolesGrid.SelectedItem != null)

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new UsersInRole(((DataRowView)rolesGrid.SelectedItem)[1]));

}

else

{

MessageBox.Show("Просматривать можно только по выделенной роли", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

private void countOfRoles\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT COUNT(Id) FROM Roles";

MessageBox.Show("Количество ролей: " + command.ExecuteScalar().ToString());

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

searchBox.Text = "Поиск";

}

}

}

}

}

<Page x:Class="KP.Pages.Tracks"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Tracks">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮" Click="updatePage\_Click"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn" Click="searchBtn\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="tracksGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Track}" Header="Трек"/>

<DataGridTextColumn Binding="{Binding Album}" Header="Альбом"/>

<DataGridTextColumn Binding="{Binding Executor}" Header="Исполнитель"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="deleteLiked" Header="Убрать из Избранного" Click="deleteLiked\_Click"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Tracks.xaml

/// </summary>

public partial class Tracks : Page

{

public Tracks()

{

InitializeComponent();

Classes.Get.TableOutput(tracksGrid, null, 3);

}

private void searchBtn\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindLikedTracksPiece @login, @track";

command.Parameters.AddWithValue("@login", Classes.Login.Value);

command.Parameters.AddWithValue("@track", searchBox.Text);

Classes.Get.FillGrid(command, tracksGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

searchBox.Text = "Поиск";

}

}

}

private void updatePage\_Click(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(tracksGrid, null, 3);

searchBox.Text = "Поиск";

}

private void deleteLiked\_Click(object sender, RoutedEventArgs e)

{

if (tracksGrid.SelectedItem != null)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить данную запись?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Likes WHERE Likes.Login = (SELECT Users.Login FROM Users" +

" WHERE Users.Login = @login) " +

"AND Likes.Track = (SELECT Id FROM Tracks WHERE Tracks.Name = @track)";

command.Parameters.AddWithValue("@track", ((DataRowView)tracksGrid.SelectedItem)[0]);

command.Parameters.AddWithValue("@login", Classes.Login.Value);

command.ExecuteNonQuery();

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

Classes.Get.TableOutput(tracksGrid, null, 3);

}

}

}

}

else

{

MessageBox.Show("Удалить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

}

}

<Page x:Class="KP.Pages.User"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="792"

Title="User">

<Grid>

<Button Style="{StaticResource UserBtn}" x:Name="main" Content="Главная" Click="main\_Click"/>

<Button Style="{StaticResource UserBtn}" x:Name="genres" Content="Жанры" Margin="0,40,0,0" Click="genres\_Click"/>

<Button Style="{StaticResource UserBtn}" x:Name="tracks" Content="Треки" Margin="0,80,0,0" Click="tracks\_Click"/>

<Button Style="{StaticResource UserBtn}" x:Name="executors" Content="Исполнители" Margin="0,120,0,0" Click="executors\_Click"/>

<Button Style="{StaticResource UserBtn}" x:Name="roles" Content="Роли" Margin="0,200,0,0" Click="roles\_Click" Visibility="Hidden"/>

<Button Style="{StaticResource UserBtn}" x:Name="users" Content="Пользователи" Margin="0,240,0,0" Click="users\_Click" Visibility="Hidden"/>

<Button Style="{StaticResource UserBtn}" x:Name="profile" Content="Профиль" VerticalAlignment="Bottom" Margin="0,0,0,40" Click="profile\_Click"/>

<Button Style="{StaticResource UserBtn}" x:Name="back" Content="Выйти" VerticalAlignment="Bottom" Click="back\_Click"/>

<Button Style="{StaticResource UserBtn}" x:Name="countries" Content="Страны" Click="countries\_Click" Margin="0,160,0,0" Visibility="Hidden"/>

<Frame x:Name="frame" NavigationUIVisibility="Hidden" Margin="150,0,0,0"/>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using System.Data.SqlClient;

using System.Configuration;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для User.xaml

/// </summary>

public partial class User : Page

{

public User()

{

InitializeComponent();

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindRole @login";

command.Parameters.AddWithValue("@login", Classes.Login.Value);

using (SqlDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

if (reader[0].ToString() == "Администратор")

{

roles.Visibility = Visibility.Visible;

users.Visibility = Visibility.Visible;

countries.Visibility = Visibility.Visible;

}

else if (reader[0].ToString() == "Редактор")

{

countries.Visibility = Visibility.Visible;

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

/// <summary>

/// Метод перехода на главную страницу по нажатию кнопки "Главная"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void main\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Uri("Pages/Main.xaml", UriKind.RelativeOrAbsolute));

}

/// <summary>

/// Метод перехода на страницу жанров по нажатию кнопки "Жанры"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void genres\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Uri("Pages/Genres.xaml", UriKind.RelativeOrAbsolute));

}

/// <summary>

/// Метод перехода на страницу треков по нажатию кнопки "Треки"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void tracks\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Uri("Pages/Tracks.xaml", UriKind.RelativeOrAbsolute));

}

/// <summary>

/// Метод переходв на страницу исполнителей по нажатию кнопки "Исполнители"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void executors\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Uri("Pages/Executors.xaml", UriKind.RelativeOrAbsolute));

}

/// <summary>

/// Метод перехода на страницу ролей по нажатию кнопки "Роли"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void roles\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Uri("Pages/Roles.xaml", UriKind.RelativeOrAbsolute));

}

/// <summary>

/// Метод перехода на страницу пользователей по нажаитю кнопки "Пользователи"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void users\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Uri("Pages/Users.xaml", UriKind.RelativeOrAbsolute));

}

/// <summary>

/// Метод перехода на страницу профиля пользователя по нажатию кнопки "Профиль"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void profile\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Profile(Classes.Login.Value));

}

/// <summary>

/// Метод возврата на страницу авторизации по нажатию кнопки "Выйти"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void back\_Click(object sender, RoutedEventArgs e)

{

this.NavigationService.Navigate(new Uri("Pages/Auth.xaml", UriKind.RelativeOrAbsolute));

Classes.Login.Value = "";

}

private void countries\_Click(object sender, RoutedEventArgs e)

{

frame.Navigate(new Uri("Pages/Countries.xaml", UriKind.RelativeOrAbsolute));

}

}

}

<Page x:Class="KP.Pages.Users"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="Users" Loaded="Page\_Loaded">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="updatePage" Content="⭮" Click="updatePage\_Click"/>

<TextBox Style="{StaticResource SearchBox}" x:Name="searchBox"/>

<Button Style="{StaticResource SearchBtn}" x:Name="searchBtn" Click="searchBtn\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="usersGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Login}" Header="Логин"/>

<DataGridTextColumn Binding="{Binding Password}" Header="Пароль"/>

<DataGridTextColumn Binding="{Binding Name}" Header="Имя"/>

<DataGridTextColumn Binding="{Binding Surname}" Header="Фамилия"/>

<DataGridTextColumn Binding="{Binding Gender}" Header="Пол"/>

<DataGridTextColumn Binding="{Binding Country}" Header="Страна"/>

<DataGridTextColumn Binding="{Binding Role}" Header="Роль"/>

</DataGrid.Columns>

<DataGrid.ContextMenu>

<ContextMenu>

<MenuItem x:Name="userInfos" Header="Профиль пользователя" Click="userInfos\_Click"/>

<Separator/>

<MenuItem x:Name="add" Header="Добавить" Click="add\_Click"/>

<MenuItem x:Name="edit" Header="Редактировать" Click="edit\_Click"/>

<MenuItem x:Name="delete" Header="Удалить" Click="delete\_Click"/>

</ContextMenu>

</DataGrid.ContextMenu>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для Users.xaml

/// </summary>

public partial class Users : Page

{

public Users()

{

InitializeComponent();

}

private void Page\_Loaded(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(usersGrid, null, 6);

}

/// <summary>

/// Метод поиска сопадений в таблице пользователей по слову или части слова в поисковой строке по нажатию кнопки "Искать"

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void searchBtn\_Click(object sender, RoutedEventArgs e)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "EXEC FindUsersPiece @search";

command.Parameters.AddWithValue("@search", searchBox.Text);

Classes.Get.FillGrid(command, usersGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

searchBox.Text = "Поиск";

}

}

}

private void updatePage\_Click(object sender, RoutedEventArgs e)

{

Classes.Get.TableOutput(usersGrid, null, 6);

searchBox.Text = "Поиск";

}

private void userInfos\_Click(object sender, RoutedEventArgs e)

{

if (usersGrid.SelectedItem != null)

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new Profile(((DataRowView)usersGrid.SelectedItem)[1]));

}

else

{

MessageBox.Show("Профиль можно вызвать только у выделенного пользователя", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

Classes.Get.TableOutput(usersGrid, null, 6);

searchBox.Text = "Поиск";

}

private void add\_Click(object sender, RoutedEventArgs e)

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new Uri("Pages/Reg.xaml", UriKind.RelativeOrAbsolute));

}

private void edit\_Click(object sender, RoutedEventArgs e)

{

if (usersGrid.SelectedItem != null)

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new EditUser(((DataRowView)usersGrid.SelectedItem)[1]));

}

else

{

MessageBox.Show("Изменить можно только существующего пользователя", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

private void delete\_Click(object sender, RoutedEventArgs e)

{

if (usersGrid.SelectedItem != null)

{

if (((DataRowView)usersGrid.SelectedItem)[1].ToString() != Classes.Login.Value)

{

MessageBoxResult result = MessageBox.Show("Вы уверены, что хотите удалить данную запись?", "Уведомление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "DELETE FROM Users WHERE Login = @name";

command.Parameters.AddWithValue("@name", ((DataRowView)usersGrid.SelectedItem)[1]);

command.ExecuteNonQuery();

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

Classes.Get.TableOutput(usersGrid, null, 6);

}

}

}

}

else

{

MessageBox.Show("Нельзя удалить текущего пользователя", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

else

{

MessageBox.Show("Удалить можно только существующую запись", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

}

}

}

<Page x:Class="KP.Pages.UsersInRole"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:KP.Pages"

mc:Ignorable="d"

d:DesignHeight="419" d:DesignWidth="642"

Title="UsersInRole">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="33"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Button Style="{StaticResource UpdatePageBtn}" x:Name="back" Content="Назад" Width="70" Click="back\_Click"/>

<DataGrid AutoGenerateColumns="False" x:Name="usersInRolesGrid">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Login}" Header="Логин"/>

<DataGridTextColumn Binding="{Binding Password}" Header="Пароль"/>

<DataGridTextColumn Binding="{Binding Name}" Header="Имя"/>

<DataGridTextColumn Binding="{Binding Surname}" Header="Фамилия"/>

<DataGridTextColumn Binding="{Binding Gender}" Header="Пол"/>

<DataGridTextColumn Binding="{Binding Country}" Header="Страна"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Page>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP.Pages

{

/// <summary>

/// Логика взаимодействия для UsersInRole.xaml

/// </summary>

public partial class UsersInRole : Page

{

public UsersInRole(object obj)

{

InitializeComponent();

Classes.Get.TableOutput(usersInRolesGrid, obj, 8);

}

private void back\_Click(object sender, RoutedEventArgs e)

{

NavigationService.GoBack();

}

}

}

<Window x:Class="KP.Windows.Country"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:KP.Windows"

mc:Ignorable="d"

Height="250" Width="400" ResizeMode="NoResize" Topmost="True" MinWidth="400" MinHeight="250" ShowInTaskbar="False" Background="#FF404040">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="234"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Style="{StaticResource RegBlock}" Text="Название:" Margin="0,0,160,40"/>

<TextBox Style="{StaticResource RegBox}" x:Name="name" Margin="76,0,0,40"/>

<Button Style="{StaticResource AuthBtn}" x:Name="save" Content="" Margin="0,40,0,0"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace KP.Windows

{

/// <summary>

/// Логика взаимодействия для Country.xaml

/// </summary>

public partial class Country : Window

{

public Country(bool old, object obj, DataGrid dataGrid)

{

InitializeComponent();

if (old == true)

{

this.Title = "Изменение страны";

save.Content = "Изменить";

save.Click += (s, e) =>

{

UpdateCountry(obj);

Classes.Get.TableOutput(dataGrid, null, 9);

};

}

else

{

this.Title = "Добавление страны";

save.Content = "Добавить";

save.Click += (s, e) =>

{

InsertCountry();

Classes.Get.TableOutput(dataGrid, null, 9);

};

}

}

private void UpdateCountry(object obj)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Countries WHERE Name = @name";

if (name.Text != null && (!name.Text.StartsWith(" ") || !name.Text.Contains(" "))) command.Parameters.AddWithValue("@name", name.Text.Trim());

else

{

MessageBox.Show("Неверный формат ввода", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

MessageBox.Show("Такая страна уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

reader.Close();

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "UPDATE Countries SET Name = @name WHERE Name = @old";

command1.Parameters.AddWithValue("@name", name.Text.Trim());

command1.Parameters.AddWithValue("@old", obj);

command1.ExecuteNonQuery();

}

this.Close();

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

private void InsertCountry()

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Countries WHERE Name = @name";

if (name.Text != null && (!name.Text.StartsWith(" ") || !name.Text.Contains(" "))) command.Parameters.AddWithValue("@name", name.Text.Trim());

else

{

MessageBox.Show("Неверный формат ввода", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

MessageBox.Show("Такая страна уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

reader.Close();

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "INSERT INTO Countries VALUES (@name)";

command1.Parameters.AddWithValue("@name", name.Text.Trim());

command1.ExecuteNonQuery();

}

this.Close();

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

}

}

<Window x:Class="KP.Windows.Genre"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:KP.Windows"

mc:Ignorable="d"

Height="250" Width="400" ResizeMode="NoResize" Topmost="True" MinWidth="400" MinHeight="250" ShowInTaskbar="False" Background="#FF404040">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="234"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Style="{StaticResource RegBlock}" Text="Название:" Margin="0,0,160,40"/>

<TextBox Style="{StaticResource RegBox}" x:Name="name" Margin="76,0,0,40"/>

<Button Style="{StaticResource AuthBtn}" x:Name="save" Content="" Margin="0,40,0,0"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace KP.Windows

{

/// <summary>

/// Логика взаимодействия для Genre.xaml

/// </summary>

public partial class Genre : Window

{

public Genre(bool old, object obj, DataGrid dataGrid)

{

InitializeComponent();

if (old == true)

{

this.Title = "Изменение жанра";

save.Content = "Изменить";

save.Click += (s, e) =>

{

UpdateGenre(obj);

Classes.Get.TableOutput(dataGrid, null, 2);

};

}

else

{

this.Title = "Добавление жанра";

save.Content = "Добавить";

save.Click += (s, e) =>

{

InsertGenre();

Classes.Get.TableOutput(dataGrid, null, 2);

};

}

}

/// <summary>

/// Метод обновления существующего жанра

/// </summary>

/// <param name="obj"></param>

private void UpdateGenre(object obj)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Genres WHERE Name = @name";

if (name.Text != null && (!name.Text.StartsWith(" ") || !name.Text.Contains(" "))) command.Parameters.AddWithValue("@name", name.Text.Trim());

else

{

MessageBox.Show("Неверный формат ввода", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

MessageBox.Show("Такой жанр уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

reader.Close();

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "UPDATE Genres SET Name = @name WHERE Name = @old";

command1.Parameters.AddWithValue("@name", name.Text.Trim());

command1.Parameters.AddWithValue("@old", obj);

command1.ExecuteNonQuery();

}

this.Close();

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

/// <summary>

/// Метод добавления нового жанра

/// </summary>

private void InsertGenre()

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Genres WHERE Name = @name";

if (name.Text != null && (!name.Text.StartsWith(" ") || !name.Text.Contains(" "))) command.Parameters.AddWithValue("@name", name.Text.Trim());

else

{

MessageBox.Show("Неверный формат ввода", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

MessageBox.Show("Такой жанр уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

reader.Close();

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "INSERT INTO Genres VALUES (@name)";

command1.Parameters.AddWithValue("@name", name.Text.Trim());

command1.ExecuteNonQuery();

}

this.Close();

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

}

}

<Window x:Class="KP.Windows.Password"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:KP.Windows"

mc:Ignorable="d"

Title="Изменение пароля" Height="250" Width="400" ResizeMode="NoResize" Topmost="True" MinWidth="400" MinHeight="250" ShowInTaskbar="False" Background="#FF404040">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="234"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Style="{StaticResource RegBlock}" Text="Старый:" Margin="0,0,175,80"/>

<TextBox Style="{StaticResource RegBox}" x:Name="old" Margin="60,0,0,80"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Новый:" Margin="0,0,175,30"/>

<PasswordBox Style="{StaticResource RegPass}" x:Name="\_new" Margin="60,0,0,30"/>

<TextBlock Style="{StaticResource RegBlock}" Text="Повтор:" Margin="0,20,175,0"/>

<PasswordBox Style="{StaticResource RegPass}" x:Name="repeat" Margin="60,20,0,0"/>

<Button Style="{StaticResource AuthBtn}" x:Name="save" Content="Изменить пароль" Margin="0,80,0,0"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace KP.Windows

{

/// <summary>

/// Логика взаимодействия для Password.xaml

/// </summary>

public partial class Password : Window

{

public Password(object user)

{

InitializeComponent();

save.Click += (s, e) =>

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT Password FROM Users WHERE Login = @login";

command.Parameters.AddWithValue("@login", user);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

reader.Read();

if (old.Text != reader[0].ToString())

{

MessageBox.Show("Старый пароль неверный", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

reader.Close();

if (\_new.Password.Length > 5 && \_new.Password.Length <= 20)

{

if (repeat.Password != \_new.Password)

{

MessageBox.Show("Пароли не совпадают", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

old.Text = "";

\_new.Password = "";

repeat.Password = "";

}

else

{

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "UPDATE Users SET Password = @pw WHERE Login = @login";

command1.Parameters.AddWithValue("@login", user);

command1.Parameters.AddWithValue("@pw", repeat.Password);

command1.ExecuteNonQuery();

}

this.Close();

}

}

else

{

MessageBox.Show("Пароль должен быть в диапазоне от 6 до 20 символов", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

old.Text = "";

\_new.Password = "";

}

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

};

}

}

}

<Window x:Class="KP.Windows.Role"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:KP.Windows"

mc:Ignorable="d"

Height="250" Width="400" ResizeMode="NoResize" Topmost="True" MinWidth="400" MinHeight="250" ShowInTaskbar="False" Background="#FF404040">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition Width="234"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<TextBlock Style="{StaticResource RegBlock}" Text="Название:" Margin="0,0,160,40"/>

<TextBox Style="{StaticResource RegBox}" x:Name="name" Margin="76,0,0,40"/>

<Button Style="{StaticResource AuthBtn}" x:Name="save" Content="" Margin="0,40,0,0"/>

</Grid>

</Window>

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace KP.Windows

{

/// <summary>

/// Логика взаимодействия для Role.xaml

/// </summary>

public partial class Role : Window

{

public Role(bool old, object obj,DataGrid dataGrid)

{

InitializeComponent();

if (old == true)

{

this.Title = "Изменение роли";

save.Content = "Изменить";

save.Click += (s, e) =>

{

UpdateRole(obj);

Classes.Get.TableOutput(dataGrid, null, 5);

};

}

else

{

this.Title = "Добавление роли";

save.Content = "Добавить";

save.Click += (s, e) =>

{

InsertRole();

Classes.Get.TableOutput(dataGrid, null, 5);

};

}

}

/// <summary>

/// Метод обновления существующей роли

/// </summary>

/// <param name="role"></param>

private void UpdateRole(object obj)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Roles WHERE Name = @name";

if (name.Text != null && (!name.Text.StartsWith(" ") || !name.Text.Contains(" "))) command.Parameters.AddWithValue("@name", name.Text.Trim());

else

{

MessageBox.Show("Неверный формат ввода", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

MessageBox.Show("Такая роль уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

reader.Close();

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "UPDATE Roles SET Name = @name WHERE Name = @old";

command1.Parameters.AddWithValue("@name", name.Text.Trim());

command1.Parameters.AddWithValue("@old", obj);

command1.ExecuteNonQuery();

}

this.Close();

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

/// <summary>

/// Метод добавления новой роли

/// </summary>

private void InsertRole()

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT \* FROM Roles WHERE Name = @name";

if (name.Text != null && (!name.Text.StartsWith(" ") || !name.Text.Contains(" "))) command.Parameters.AddWithValue("@name", name.Text.Trim());

else

{

MessageBox.Show("Неверный формат ввода", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

MessageBox.Show("Такая роль уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

else

{

reader.Close();

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "INSERT INTO Roles VALUES (@name)";

command1.Parameters.AddWithValue("@name", name.Text.Trim());

command1.ExecuteNonQuery();

}

this.Close();

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

}

}

<?xml version="1.0" encoding="utf-8" ?>

<configuration>

<startup>

<supportedRuntime version="v4.0" sku=".NETFramework,Version=v4.8" />

</startup>

<connectionStrings>

<add name="connStr" connectionString="Data Source=HP-NOTEBOOK\SQLEXPRESS;Initial Catalog=KP;Integrated Security=True" providerName="System.Data.SqlClient"/>

</connectionStrings>

</configuration>

<Application x:Class="KP.App"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:local="clr-namespace:KP"

StartupUri="MainWindow.xaml">

<Application.Resources>

<Style x:Key="AuthBlock" TargetType="TextBlock">

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="AuthBox" TargetType="TextBox">

<Setter Property="Background" Value="{x:Null}"/>

<Setter Property="BorderThickness" Value="0,0,0,1"/>

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="AuthPass" TargetType="PasswordBox">

<Setter Property="BorderThickness" Value="0,0,0,1"/>

<Setter Property="Background" Value="{x:Null}"/>

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="AuthBtn" TargetType="Button">

<Setter Property="Background" Value="Gray"/>

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="RegBlock" TargetType="TextBlock">

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="RegBox" TargetType="TextBox">

<Setter Property="Background" Value="{x:Null}"/>

<Setter Property="BorderThickness" Value="0,0,0,1"/>

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="RegPass" TargetType="PasswordBox">

<Setter Property="BorderThickness" Value="0,0,0,1"/>

<Setter Property="Background" Value="{x:Null}"/>

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="RegCmbox" TargetType="ComboBox">

<Setter Property="Grid.Column" Value="1"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="RegBtn" TargetType="Button">

<Setter Property="Background" Value="Gray"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="UserBtn" TargetType="Button">

<Setter Property="Background" Value="Gray"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Height" Value="40"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="VerticalAlignment" Value="Top"/>

<Setter Property="Width" Value="150"/>

</Style>

<Style x:Key="SearchBox" TargetType="TextBox">

<Setter Property="BorderThickness" Value="0,0,0,1"/>

<Setter Property="Background" Value="{x:Null}"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Margin" Value="0,0,84,0"/>

<Setter Property="HorizontalAlignment" Value="Right"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="Width" Value="160"/>

<Setter Property="Text" Value="Поиск"/>

</Style>

<Style x:Key="SearchBtn" TargetType="Button">

<Setter Property="Background" Value="Gray"/>

<Setter Property="Content" Value="Искать"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Margin" Value="0,0,4,0"/>

<Setter Property="HorizontalAlignment" Value="Right"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="Width" Value="75"/>

</Style>

<Style TargetType="DataGrid">

<Setter Property="CanUserResizeRows" Value="False"/>

<Setter Property="CanUserReorderColumns" Value="False"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="IsReadOnly" Value="True"/>

<Setter Property="Grid.Row" Value="1"/>

<Setter Property="GridLinesVisibility" Value="None"/>

<Setter Property="SelectionMode" Value="Single"/>

</Style>

<Style TargetType="CheckBox">

<Setter Property="Content" Value="Полная фраза"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Margin" Value="0,0,249,0"/>

<Setter Property="HorizontalAlignment" Value="Right"/>

<Setter Property="FlowDirection" Value="RightToLeft"/>

<Setter Property="VerticalAlignment" Value="Center"/>

</Style>

<Style x:Key="UpdatePageBtn" TargetType="Button">

<Setter Property="Background" Value="Gray"/>

<Setter Property="FontSize" Value="16"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="Margin" Value="4,0,0,0"/>

<Setter Property="HorizontalAlignment" Value="Left"/>

<Setter Property="VerticalAlignment" Value="Center"/>

<Setter Property="Width" Value="25"/>

</Style>

<Style TargetType="DataGridCell">

<Setter Property="TextBlock.TextAlignment" Value="Right"/>

</Style>

<Style TargetType="DataGridColumnHeader">

<Setter Property="HorizontalContentAlignment" Value="Center"/>

</Style>

</Application.Resources>

</Application>

<Window x:Class="KP.MainWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:KP"

mc:Ignorable="d"

Title="MainWindow" Height="450" Width="800" MinWidth="800" MinHeight="450">

<Frame x:Name="frame" NavigationUIVisibility="Hidden" Source="Pages\Auth.xaml" Background="#F0404040"/>

</Window>

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace KP

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

}

using Microsoft.Win32;

using System;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media.Imaging;

namespace KP.Classes

{

class Set

{

/// <summary>

/// Метод передачи файла изображения в форму изображения

/// </summary>

/// <param name="file"></param>

/// <returns></returns>

public static BitmapImage ImageFromFile(out FileInfo file)

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.Filter = "Изображения (\*.jpg, \*.png, \*.gif)|\*.jpg; \*.png; \*.gif";

file = null;

BitmapImage bitmap = new BitmapImage();

if (dialog.ShowDialog() == true)

{

file = new FileInfo(dialog.FileName);

}

if (file != null)

{

bitmap.BeginInit();

bitmap.UriSource = new Uri(file.FullName);

bitmap.EndInit();

}

else

{

bitmap.BeginInit();

bitmap.UriSource = new Uri("/KP;component/Resources/addImage.png", UriKind.RelativeOrAbsolute);

bitmap.EndInit();

}

return bitmap;

}

public static FileInfo TrackFile()

{

OpenFileDialog dialog = new OpenFileDialog();

dialog.Filter = "Музыка (\*.mp3, \*.wav, \*.flac)|\*.mp3; \*.wav; \*.flac";

FileInfo file = null;

BitmapImage bitmap = new BitmapImage();

if (dialog.ShowDialog() == true)

{

file = new FileInfo(dialog.FileName);

}

return file;

}

}

}

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

namespace KP.Classes

{

class Get

{

/// <summary>

/// Метод получения названий стран из бд в выпадающий список

/// </summary>

/// <param name="connection"></param>

/// <param name="country"></param>

public static void Countries(SqlConnection connection, ComboBox country)

{

using (SqlCommand command = connection.CreateCommand())

{

command.CommandText = "SELECT Name FROM Countries";

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read())

{

country.Items.Add(reader[0]);

}

}

}

}

}

/// <summary>

/// Метод получения изображения из бд

/// </summary>

/// <param name="filePath"></param>

/// <returns></returns>

public static byte[] BytesFromFile(string filePath)

{

FileStream stream = new FileStream(filePath, FileMode.Open, FileAccess.Read);

BinaryReader reader = new BinaryReader(stream);

byte[] bytes = reader.ReadBytes((int)stream.Length);

reader.Close();

stream.Close();

return bytes;

}

/// <summary>

/// Метод заполнения списка ролей

/// </summary>

/// <param name="command"></param>

public static void FillGrid(SqlCommand command, DataGrid rolesGrid)

{

command.ExecuteNonQuery();

using (SqlDataAdapter adapter = new SqlDataAdapter(command))

{

DataTable rolesTable = new DataTable();

adapter.Fill(rolesTable);

rolesGrid.ItemsSource = rolesTable.DefaultView;

}

}

/// <summary>

/// Метод показа таблицы ролей

/// </summary>

public static void TableOutput(DataGrid dataGrid, object old, byte i)

{

using (SqlConnection connection = new SqlConnection(ConfigurationManager.ConnectionStrings["connStr"].ConnectionString))

{

try

{

connection.Open();

using (SqlCommand command = connection.CreateCommand())

{

switch (i)

{

case 0:

command.CommandText = "EXEC FindTracks @login";

command.Parameters.AddWithValue("@login", Login.Value);

break;

case 1:

command.CommandText = "select Executors.Name as Executor, YEar(Executors.Date) as Date," +

"Countries.Name as Country" +

" from Executors inner join Records on Executors.Id = Executor" +

" left join Countries on Countries.Id = Executors.Country";

break;

case 2:

command.CommandText = "SELECT \* FROM Genres";

break;

case 3:

command.CommandText = "EXEC FindLikedTracks @login";

command.Parameters.AddWithValue("@login", Login.Value);

break;

case 4:

command.CommandText = "EXEC FindLikedExecutors @login";

command.Parameters.AddWithValue("@login", Login.Value);

break;

case 5:

command.CommandText = "SELECT \* FROM Roles";

break;

case 6:

command.CommandText = "EXEC FindUsers";

break;

case 7:

command.CommandText = "EXEC FindExecutorsInGenre @name";

command.Parameters.AddWithValue("@name", old);

break;

case 8:

command.CommandText = "SELECT Login, Password, Users.Name as Name, Surname, Gender, Countries.Name as Country FROM Users INNER JOIN" +

" Roles ON Roles.Id = Role Left JOIN Countries ON Countries.Id = Users.Country WHERE Roles.Name = @role";

command.Parameters.AddWithValue("@role", old);

break;

case 9:

command.CommandText = "Select Name as Country FROM countries";

break;

case 10:

command.CommandText = "SELECT Albums.Name AS Album, YEAR(Albums.Date) As Date FROM Albums " +

"inner join Records on Albums.Id = Records.Album";

break;

default:

break;

}

FillGrid(command, dataGrid);

}

}

catch (SqlException ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

finally

{

connection.Close();

}

}

}

}

}

using System;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Media;

namespace KP.Classes

{

class Captcha

{

/// <summary>

/// Метод обновления капчи и очистки поля ввода проверки

/// </summary>

/// <param name="textBlock"></param>

/// <param name="textBox"></param>

public static void Refresh(TextBlock textBlock, TextBox textBox)

{

textBlock.Text = "";

textBox.Text = "";

string symb = "1234567890qwertyuiopasdfghjklzxcvbnmQWERTYUIOPASDFGHJKLZXCVBNM";

char[] vs = symb.ToCharArray();

Brush[] backs = { Brushes.Red, Brushes.Orange, Brushes.Yellow, Brushes.Green, Brushes.LightBlue, Brushes.Blue, Brushes.Purple };

Brush[] fores = { Brushes.Black, Brushes.Gray, Brushes.Brown, Brushes.Gold, Brushes.White };

Random rnd = new Random();

for (int i = 0; i < rnd.Next(4, 7); i++)

{

textBlock.Text += vs[rnd.Next(vs.Length)];

textBlock.Background = backs[rnd.Next(backs.Length)];

textBlock.Foreground = fores[rnd.Next(fores.Length)];

}

}

/// <summary>

/// Метод появления капчи после трех неправильных попыток

/// </summary>

/// <param name="textBlock"></param>

/// <param name="textBox"></param>

/// <param name="button"></param>

/// <param name="check"></param>

public static void Check(TextBlock textBlock, TextBox textBox, Button button, bool check)

{

if (check == true)

{

textBlock.Visibility = Visibility.Hidden;

button.Visibility = Visibility.Hidden;

textBox.Visibility = Visibility.Hidden;

}

else if (check == false)

{

textBlock.Visibility = Visibility.Visible;

button.Visibility = Visibility.Visible;

textBox.Visibility = Visibility.Visible;

Refresh(textBlock, textBox);

}

}

}

}

using System.Data.SqlClient;

using System.IO;

namespace KP.Classes

{

class Add

{

/// <summary>

/// Метод доавления нового изображения в бд

/// </summary>

/// <param name="connection"></param>

/// <param name="file"></param>

/// <param name="command"></param>

public static void NewImage(SqlConnection connection, FileInfo file, SqlCommand command)

{

using (SqlCommand command1 = connection.CreateCommand())

{

command1.CommandText = "INSERT INTO Images VALUES (@name, @binary)";

command1.Parameters.AddWithValue("@name", file.Name);

command1.Parameters.AddWithValue("@binary", Get.BytesFromFile(file.FullName));

command1.ExecuteNonQuery();

}

using (SqlCommand command2 = connection.CreateCommand())

{

command2.CommandText = "SELECT MAX(Id) FROM Images";

using (SqlDataReader reader = command2.ExecuteReader())

{

if (reader.HasRows)

{

while (reader.Read()) command.Parameters.AddWithValue("@image", reader[0]);

}

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.Configuration;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

using System.Windows;

namespace KP

{

/// <summary>

/// Логика взаимодействия для App.xaml

/// </summary>

public partial class App : Application

{

}

}