**Министерство образования и науки Российской федерации  
Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный университет технологий и управления имени К.Г. Разумовского (Первый казачий университет)». Университетский колледж информационных технологий**

Специальность 09.02.03 Программирование в компьютерных системах

**КУРСОВОЙ ПРОЕКТ**

ПМ.02 Разработка и администрирование баз данных

МДК.02.02 Технология разработки и защиты баз данных

на тему: «Создание и ведение базы данных видеоигр»

**Пояснительная записка**

**УКИТ.09.02.03.2017.304.014ПЗ**

|  |  |  |
| --- | --- | --- |
| Группа | П-404 |  |
| Студент | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (личная подпись) | Казаков А. А, |
| Руководитель проекта | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (личная подпись) | Ларионова Е. А. |
| Нормоконтроль | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (личная подпись) | Гусева Е. Л. |

Москва 2019 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc28978352)

[1 ОСНОВНАЯ ЧАСТЬ 6](#_Toc28978353)

[1.1 Обзор и анализ предметной области 6](#_Toc28978354)

[1.2 Жизненный цикл базы данных 7](#_Toc28978355)

[1.3 Защита базы данных 9](#_Toc28978356)

[1.4 Выбор и характеристика СУБД 9](#_Toc28978357)

[1.5 Выбор и характеристика среды разработки приложения 9](#_Toc28978358)

[2 СПЕЦИАЛЬНАЯ ЧАСТЬ 10](#_Toc28978359)

[2.1 Постановка задачи 10](#_Toc28978360)

[2.2 Архитектура информационной системы 10](#_Toc28978361)

[2.3 Логическая модель базы данных 11](#_Toc28978362)

[2.4 Нормализация таблиц 11](#_Toc28978363)

[2.5 Описание таблиц 12](#_Toc28978364)

[2.6 Другие объекты базы данных 14](#_Toc28978365)

[2.7 Разработка приложения 14](#_Toc28978366)

[2.7.1 Структурная схема приложения 14](#_Toc28978367)

[2.7.2 Используемая технология доступа к данным 15](#_Toc28978368)

[2.7.3 Описание процесса отладки приложения 16](#_Toc28978369)

[2.7.4 Защита информационной системы 16](#_Toc28978370)

[2.8 Инструкция пользователю 16](#_Toc28978371)

[2.8.1 Общие сведения об информационной системе 16](#_Toc28978372)

[2.8.2 Требования к техническим средствам 16](#_Toc28978373)

[2.8.3 Требования к программным средствам 17](#_Toc28978374)

[2.8.4 Настройка информационной системы 17](#_Toc28978375)

[2.8.5 Формы ввода 17](#_Toc28978376)

[2.8.6 Отчеты 20](#_Toc28978377)

[ЗАКЛЮЧЕНИЕ 21](#_Toc28978378)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 22](#_Toc28978379)

[ПРИЛОЖЕНИЕ А. SQL СКРИПТЫ НА СОЗДАНИЕ И ЗАПОЛНЕНИЕ БАЗЫ ДАННЫХ, НА СОЗДАНИЕ УЧЕТНЫХ ЗАПИСЕЙ ПОЛЬЗОВАТЕЛЕЙ, НА НАЗНАЧЕНИЕ ПРИВИЛЕГИЙ 23](#_Toc28978380)

[ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ 28](#_Toc28978381)

[Б.1 Код LoginWindow.xaml 28](#_Toc28978382)

[Б.2 Код LoginWindow.xaml.cs 28](#_Toc28978383)

[Б.3 Код MainWindow.xaml 28](#_Toc28978384)

[Б.4 Код MainWindow.xaml.cs 28](#_Toc28978385)

[Б.5 Код GameWindow.xaml 28](#_Toc28978386)

[Б.6 Код GameWindow.xaml.cs 28](#_Toc28978387)

[Б.7 Код EditGameWindow.xaml 28](#_Toc28978388)

[Б.8 Код EditGameWindow.xaml.cs 28](#_Toc28978389)

ВВЕДЕНИЕ

В современном мире стремительно происходит развитие информационных технологий, за счет чего появляются виды развлечений, которые ранее были невозможны, одним из таких видов развлечений являются видеоигры. По данным сервиса Steamspy который предназначен для оценки объемов продаж в сервисе цифровой дистрибьюции Steam, количество игр в сервисе Steam в период на конец 2018 – начало 2019 превысило 30 тысяч, из них около 9300 вышло в 2018 году. В условиях быстро развивающегося рынка становиться все сложнее найти игру, которая будет удовлетворять требованиям игрока. По этой причине была написана база данных, которая предназначена сбора информации о видеоиграх, а также приложения, реализующего просмотр и поиск информации по базе данных.

Основная цель исследования – это приобретение и закрепление навыков в области объектно-ориентированного анализа и проектирования. Также исследование преследует цели:

* ­Получение навыков в работе со средой MS Visual Studio и WPF,
* Закрепление знаний SQL при работе с ним в различных СУБД.

Основные задачи проекта:

* Анализ предметной области,
* Проектирование базы данных средствами ERWin Data Modeler,
* Реализация защиты базы данных в среде MS SQL,
* Разработка основных функций приложения в среде MS Visual Studio,
* Тестирование приложения.

В рассматриваемой предметной области представлены следующие приложения:

* RAWG.IO – англоязычный сайт с самой крупной базой игр. На сайте у игр не указаны рецензии от критиков, а пользовательские рецензии разделены на 4 оценки Exceptional(Исключительная), Recommended(Рекомендую), Meh(Так себе) и Skip (Пропускаю),
* Metacritic – крупный англоязычный сайт агрегатор рецензий. Не специализируется только на играх, также на сайте представлены фильмы, сериалы и музыка,
* OpenCritic – англоязычный сайт агрегатор игровых рецензий. На сайте представлены только рецензии от критиков,
* MetaGames – русскоязычный сайт агрегатор игровых рецензий. На данный момент сайт не обновляется. Попытки входа на страницы большинства игр заканчиваются ошибкой 404,
* AG.RU – русскоязычная версия сайта RAWG.IO,
* Критиканство – русскоязычный сайт агрегатор рецензий. Не специализируется только на играх, на сайте также представлены фильмы.

Разработка базы данных видеоигр позволит:

* Хранить информацию о видеоиграх,
* Производить поиск игр по интересующим параметрам,
* Просматривать информацию о видеоиграх.

1 ОСНОВНАЯ ЧАСТЬ

* 1. Обзор и анализ предметной области

База данных создается для обеспечения пользователя возможностью поиска видеоигр. База данных должна содержать данные о играх, разработчиках, издателях, платформах, жанрах, рецензиях, рецензентах и предоставлять возможность поиска игр по параметрам.

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

* Каждая игра может разрабатываться одной или несколькими студиями разработки;
* Каждая игра может быть издаваться одним или несколькими издателями;
* Каждая игра может выпускаться на одну или несколько платформ;
* Каждая игра может соответствовать одному или нескольким жанрам;
* На каждую игру может быть написано одна или более рецензий;
* Рецензия может не содержать оценки;
* Игра может содержать только одну рецензию от каждого из рецензентов.

Выделим базовые сущности этой предметной области:

* Игры,
* Разработчики,
* Издатели,
* Платформы,
* Жанры,
* Рецензенты,
* Рецензии.

Атрибуты игры – название, дата выхода, описание официальный сайт.

Атрибуты разработчиков – название.

Атрибуты издателей – название.

Атрибуты платформ – название.

Атрибуты жанров – название.

Атрибуты рецензентов – название.

Атрибуты рецензий – оценка, описание, ссылка на рецензию.

* 1. Жизненный цикл базы данных

В жизненном цикле базы данных можно выделить пять этапов:

1) Анализ,

2) Проектирование,

3) Реализация,

4) Тестирование,

5) Эксплуатация.

На первом этапе был произведен анализ предметной области, в результате которого выяснилось, что необходимо создать следующие таблицы:

* Игры,
* Разработчики,
* Игры – Разработчики,
* Издатели,
* Игры – Издатели,
* Платформы,
* Игры – Платформы,
* Жанры,
* Игры – Жанры,
* Рецензенты,
* Рецензии.

На этапе проектирования была создана концептуальная модель базы данных:

* Games (id, title, release\_date, website, summary)
* Developers (id, title)
* Games\_Developers (game\_id, developer\_id)
* Publishers (id, title)
* Games\_Publishers (game\_id, publisher\_id)
* Platforms (id, title)
* Games\_Platforms (game\_id, platform\_id)
* Genres (id, title)
* Games\_Genres (game\_id, genre\_id)
* Reviewers (id, title)
* Reviews (game\_id, reviewer\_id, score, summary, website)

Также на этом этапе были спроектированы логическая модель базы данных, которая представлена в пункте 2.3, а также физическая модель, показанная в рисунке 1.

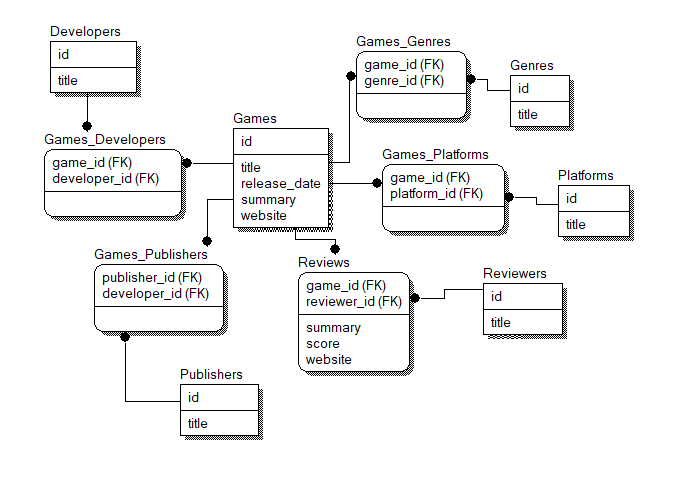


Рисунок 1 – Физическая модель базы данных

На этапе реализации был написан запрос для создания базы данных, создания и распределения прав пользователей, а также создания таблиц и других объектов БД. Код запроса указан в приложении А.

На этапе тестирования БД была заполнена тестовыми данными. Результаты тестирования показали, что схема БД является пригодной для ввода в эксплуатацию.

На этапе эксплуатации для взаимодействия с БД было разработано приложение. Код приложения представлен в приложении Б.

* 1. Защита базы данных

Для защиты базы данных используются параметризированные запросы. Использование параметризированных запросов позволяет избежать SQL-инъекций, которые могут стать причиной несанкционированного доступа к базе данных. Также для защиты базы данных используется вход по логину/паролю SQL, который позволяет ограничить права пользователя, использующего базу данных.

* 1. Выбор и характеристика СУБД

Для разработки информационной системы была выбрана СУБД Microsoft SQL Server разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact SQL. Transact SQL является реализацией стандарта ANSI/ISO по SQL с расширениями.

* 1. Выбор и характеристика среды разработки приложения

В качестве среды разработки приложения был выбран Microsoft Visual Studio 2019, в качестве технологии разработки использовалась технология WPF. У данной технологии можно выделить следующие преимущества:

* Возможность декларативного определения графического интерфейса с помощью языка разметки XAML,
* Объектная модель, которая позволяет не указывать явно размеры объектов, что позволяет создавать, интерфейсы, которые масштабируются в зависимости от размеров окна,
* Аппаратное ускорение графики.

1. СПЕЦИАЛЬНАЯ ЧАСТЬ
   1. Постановка задачи

Необходимо средствами MS SQL Server и MS Visual Studio разработать приложение, позволяющее производить поиск и просматривать информацию о видеоиграх.

Программа должна обеспечивать следующие функциональные возможности:

* Поиск игр по оценке, названию, разработчикам, издателям, платформам и жанрам,
* Просмотр информации о видеоигре,
* Добавление игр, разработчиков, издателей, платформ, жанров, рецензентов и рецензий,
* Удаление игр, разработчиков, издателей, платформ, жанров, рецензентов и рецензий,
* Изменение информации о играх, рецензентах и рецензиях.
  1. Архитектура информационной системы

Архитектура разрабатываемого приложения является двухуровневой, то есть клиент-сервер (Рисунок 2).

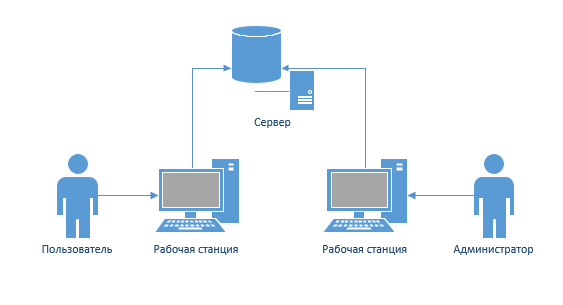


Рисунок 2 – Архитектура разрабатываемого приложения

Особенностью этой архитектуры является то, что все данные хранятся на сервере, все запросы также выполняются на стороне сервера. Это позволяет снизить требования к рабочим системам пользователей, но повышает нагрузку на сервер.

* 1. Логическая модель базы данных

На рисунке 3 представлена логическая модель базы данных видеоигр.

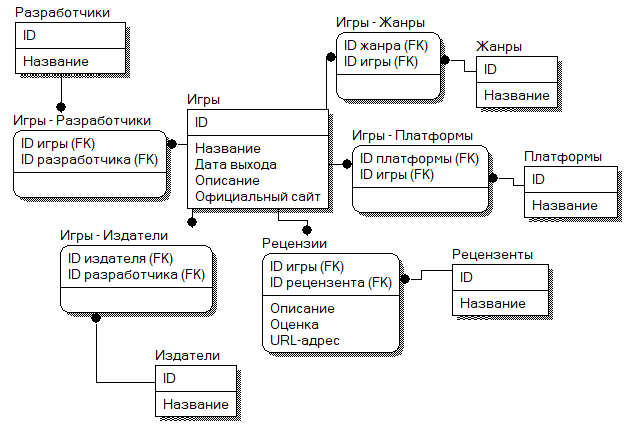


Рисунок 3. Логическая модель базы данных

* 1. Нормализация таблиц

Отношение находится в первой нормальной форме, если все атрибуты данного отношения атомарны.

Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый не ключевой атрибут зависит от первичного.

Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и между не ключевыми атрибутами нет функциональных зависимостей.

Каждый атрибут таблиц, которые указаны в разделе 2.5 настоящего документа, является атомарным, зависит от первичного ключа, только от первичного ключа и не от чего кроме первичного ключа. В связи с чем можно сделать вывод что таблицы соответствуют третьей нормальной форме.

* 1. Описание таблиц

В таблицах 1 – 11 указано описание атрибутов таблиц базы данных.

Таблица 1 – Описание таблицы Игры

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id | Идентификатор | INT | NOT NULL,  PRIMARY KEY,  IDENTITY |
| title | Название | NVARCHAR(255) | NOT NULL |
| release\_date | Дата выхода | DATE | NULL |
| website | Официальный сайт | NVARCHAR(255) | NOT NULL |
| summary | Описание | NVARCHAR(MAX) | NOT NULL |

Таблица 2 – Описание таблицы Разработчики

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id | Идентификатор | INT | NOT NULL  PRIMARY KEY,  IDENTITY |
| title | Название | NVARCHAR(255) | NOT NULL |

Таблица 3 – Описание таблицы Игры - Разработчики

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| game\_id | Идентификатор игры | INT | NOT NULL,  FOREIGN KEY |
| developer\_id | Идентификатор разработчика | INT | NOT NULL,  FOREIGN KEY |

Таблица 4 – Описание таблицы Издатели

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id | Идентификатор | INT | NOT NULL  PRIMARY KEY,  IDENTITY |
| title | Название | NVARCHAR(255) | NOT NULL |

Таблица 5 – Описание таблицы Игры – Издатели

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| game\_id | Идентификатор игры | INT | NOT NULL,  FOREIGN KEY |
| publisher\_id | Идентификатор издателя | INT | NOT NULL,  FOREIGN KEY |

Таблица 6 – Описание таблицы Платформы

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id | Идентификатор | INT | NOT NULL  PRIMARY KEY,  IDENTITY |
| title | Название | NVARCHAR(255) | NOT NULL |

Таблица 7 – Описание таблицы Игры – Платформы

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| game\_id | Идентификатор игры | INT | NOT NULL,  FOREIGN KEY |
| platform\_id | Идентификатор платформы | INT | NOT NULL,  FOREIGN KEY |

Таблица 8 – Описание таблицы Жанры

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id | Идентификатор | INT | NOT NULL  PRIMARY KEY,  IDENTITY |
| title | Название | NVARCHAR(255) | NOT NULL |

Таблица 9 – Описание таблицы Игры – Жанры

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| game\_id | Идентификатор игры | INT | NOT NULL,  FOREIGN KEY |
| genre\_id | Идентификатор жанра | INT | NOT NULL,  FOREIGN KEY |

Таблица 10 – Описание таблицы Рецензенты

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| id | Идентификатор | INT | NOT NULL  PRIMARY KEY,  IDENTITY |
| title | Название | NVARCHAR(255) | NOT NULL |

Таблица 11 – Описание таблицы Рецензии

|  |  |  |  |
| --- | --- | --- | --- |
| **Атрибут** | **Описание** | **Тип данных** | **Дополнительно** |
| game\_id | Идентификатор игры | INT | NOT NULL,  FOREIGN KEY |
| reviewer\_id | Идентификатор рецензента | INT | NOT NULL,  FOREIGN KEY |
| score | Оценка игры | TINYINT | NULL |
| website | Ссылка на рецензию | NVARCHAR(255) | NOT NULL |
| summary | Описание | NVARCHAR(MAX) | NOT NULL |

* 1. Другие объекты базы данных

При разработке ИС в базу данных была добавлена хранимая процедура StrProc\_AddGame которая добавляет новую запись в таблицу dbo.Games

2.7 Разработка приложения

2.7.1 Структурная схема приложения

Структурная схема приложения соответствует диаграммам использования для пользователя (Рисунок 4) и администратора (Рисунок 5).

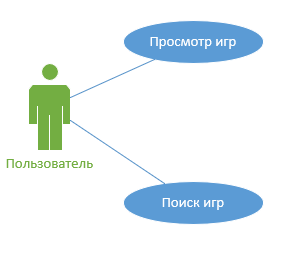


Рисунок 4 – Диаграмма использования для пользователя

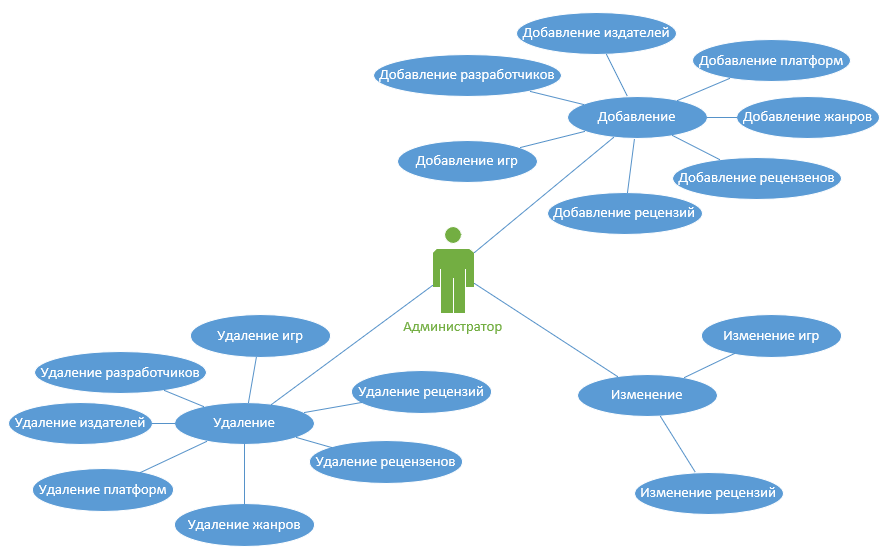


Рисунок 5 – Диаграмма использования для администратора

2.7.2 Используемая технология доступа к данным

Для доступа к данным используется технология ADO.NET, эта технология представляет нам набор классов, через которые мы можем взаимодействовать с базой данных.

Основу интерфейса взаимодействия с базами данных в ADO.NET составляют следующие объекты: Connection, Command, DataReader, DataSet и DataAdapter. Для разных источников данных в ADO.NET может быть свой провайдер данных, который определяет конкретную реализацию вышеуказанных классов. В случае провайдера для MS SQL Server классы будут иметь следующие названия: SqlConnection, SqlCommand, SqlDataReader, SqlDataAdapter.

2.7.3 Описание процесса отладки приложения

В процессе разработки ИС возникали следующие ошибки:

1. Ошибки при выполнении запросов,
2. Ошибки при входе по логину/паролю.

Для устранения ошибок, возникающих в процессе выполнения запросов, понадобилось проверять корректность написания запросов SQL.

Ошибки при входе были связаны с тем, что при установке по умолчанию был выбран способ аутентификации «Проверка подлинности Windows» из-за чего не работал вход по логину/паролю SQL. Для решения понадобилось зайти в свойства сервера, вкладка Безопасность и выбрать «Проверка подлинности Windows и SQL Server».

2.7.4 Защита информационной системы

Для защиты информационной системы были использованы параметризированные запросы, а также вход по логину/паролю SQL.

2.8 Инструкция пользователю

2.8.1 Общие сведения об информационной системе

Данная ИС предназначена для поиска и отображения информации о видеоиграх. ИС позволяет производить поиск игр по следующим параметрам: название, оценка, разработчики, издатели, платформы и жанры.

Для администратора ИС также предоставляет возможность:

* Добавления игр, разработчиков, издателей, платформ, игр, рецензентов и рецензий;
* Удаления игр, разработчиков, издателей, платформ, игр, рецензентов и рецензий;
* Изменения игр и рецензий.

2.8.2 Требования к техническим средствам

Системные требования к рабочему месту:

* Операционная система Windows (начиная с Windows 7),
* Процессор x86 или x64,
* ОЗУ объёмом не менее 4ГБ,
* Дисковой накопитель (не менее 100 МБ свободного места),
* Стабильное подключение к сети,
* Видеоадаптер,
* Монитор,
* Клавиатура и мышь.

2.8.3 Требования к программным средствам

Системные требования к серверу:

* Процессор x86 или x64,
* ОЗУ объёмом не менее 4ГБ,
* Дисковой накопитель (не менее 1 ТБ свободного места),
* Сеть 100/1000 Мбит/с.

2.8.4 Настройка информационной системы

Для запуска информационной системы необходимо:

* В экземпляре SQL Server запустить скрипт для создания базы данных видеоигр,
* Запустить проект в Visual Studio и заменить значение userConnection в файле Properties – Settings.Settings (Заменить Data Source=DESKTOP-HOME на Data Source=[Название вашего сервера]).

2.8.5 Формы ввода

В приложении представлены следующие формы ввода:

* В окне авторизации (Рисунок 6), поля ввода логина и пароля,
* В главном меню приложения (Рисунок 7), параметры поиска,
* В окне редактирования игр (Рисунок 8), поля для ввода информации о игре,
* Во вкладке рецензии (Рисунок 9), окна редактирования игр, формы для добавления, редактирования и удаления рецензий игры,
* В окне редактирования (Рисунок 10), формы для добавления и удаления объектов.

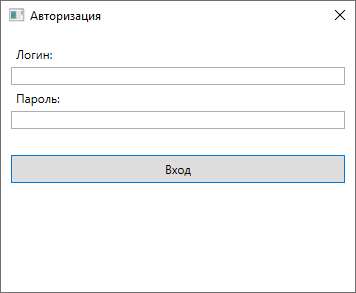


Рисунок 6 – Окно авторизации

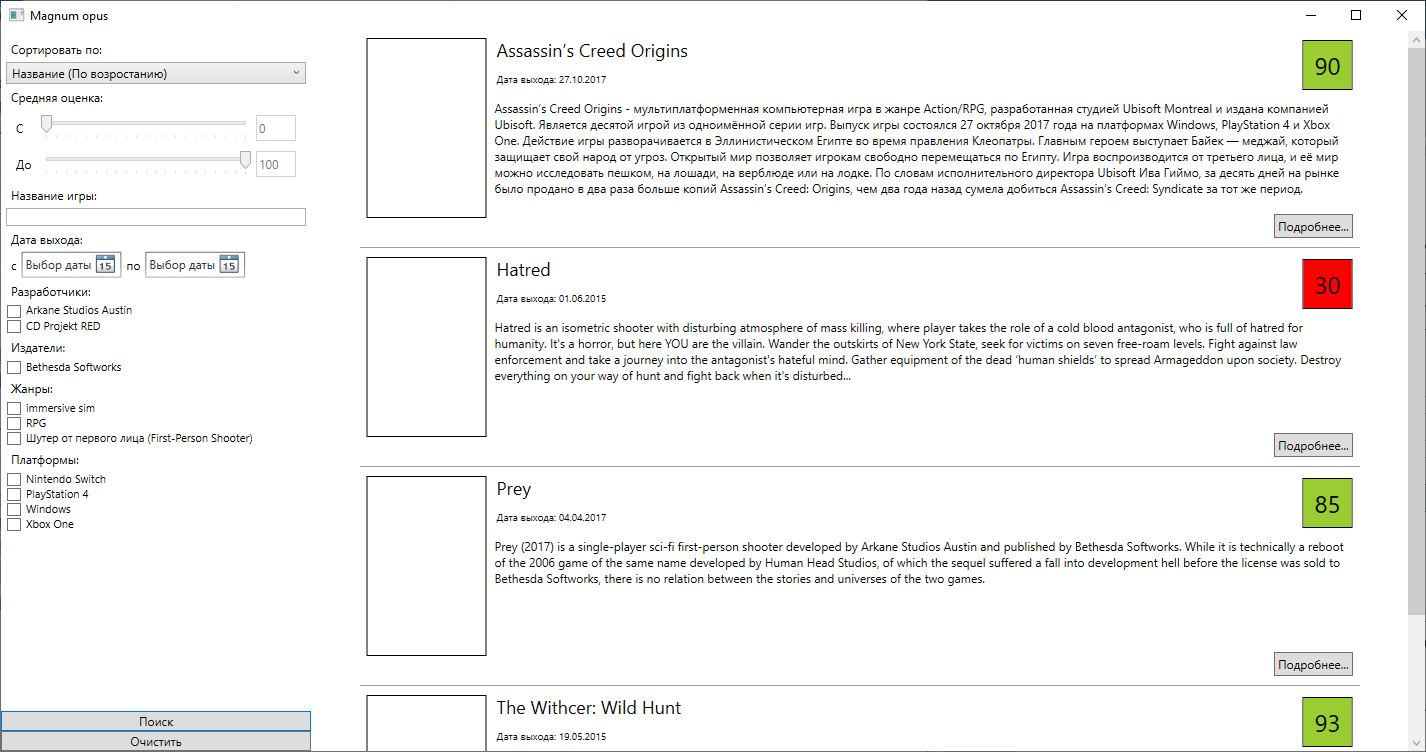


Рисунок 7 – Главное меню

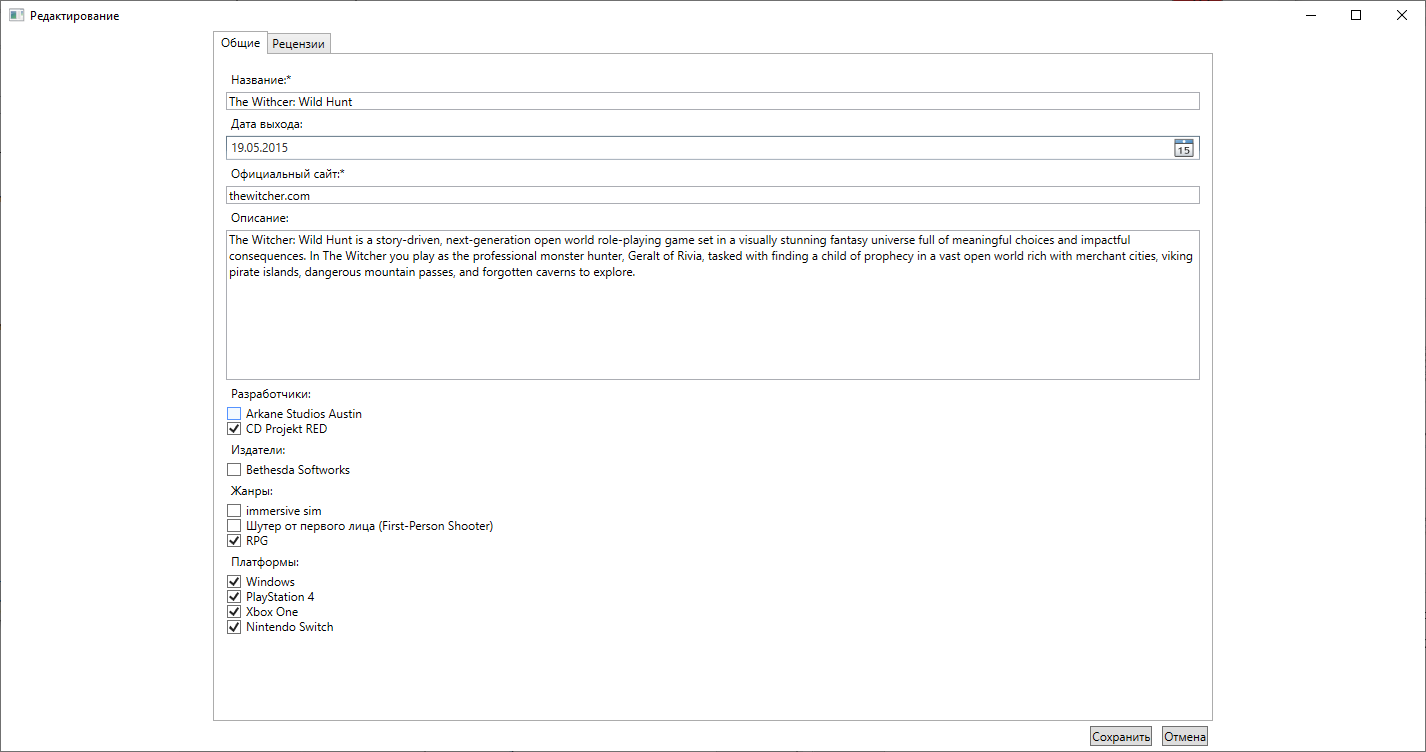


Рисунок 8 – Окно редактирования игр

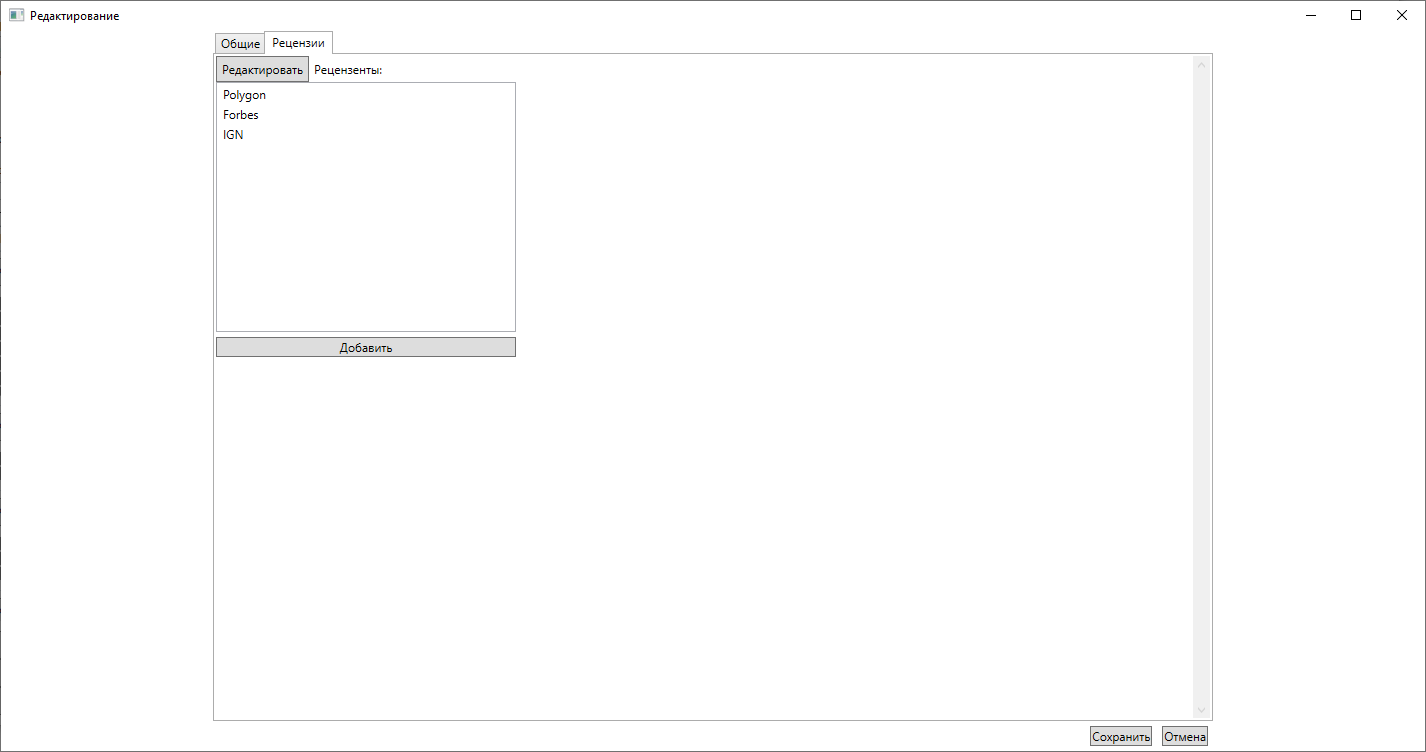


Рисунок 9 – Окно редактирования игр (Рецензии)

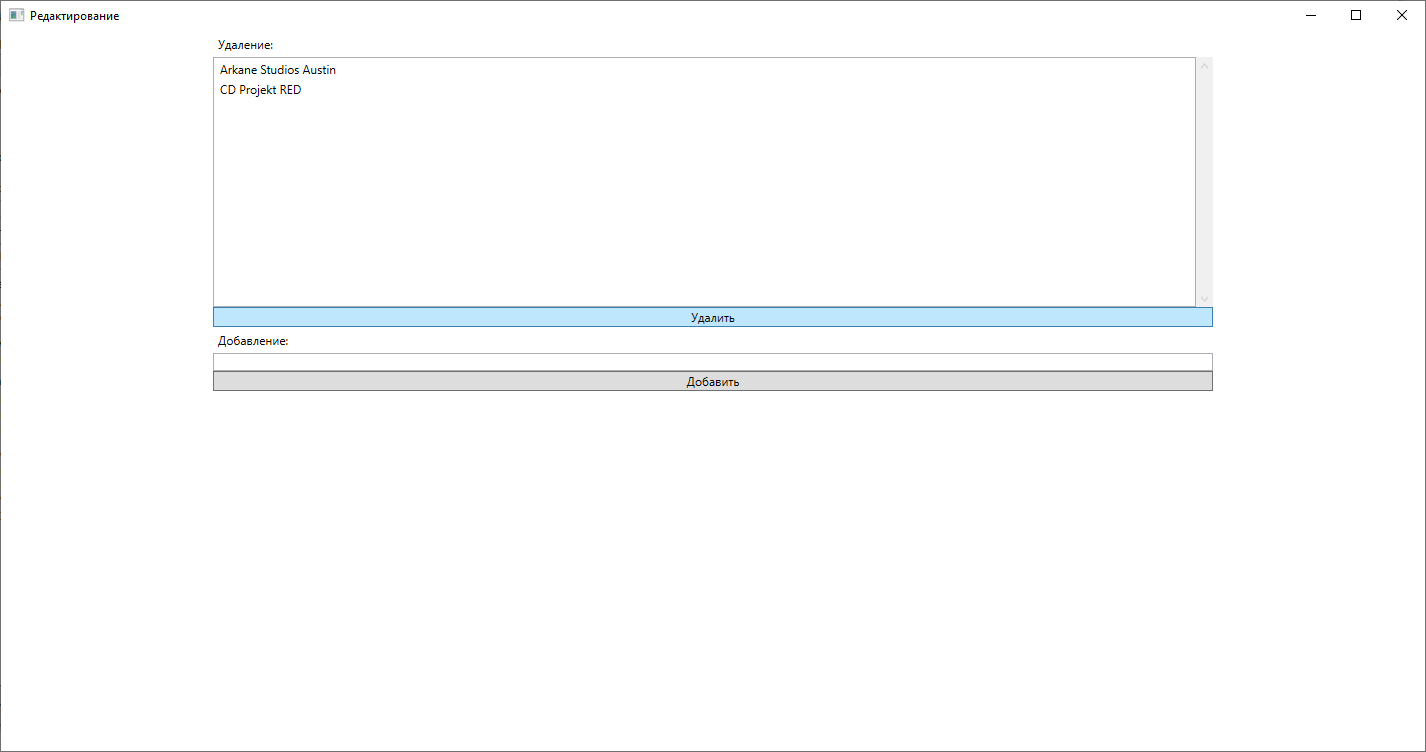


Рисунок 10 – Окно редактирования

2.8.6 Отчеты

В приложении представлены следующие отчеты:

* В главном меню (Рисунок 7), список игр по параметрам поиска,
* В окне игры (Рисунок 11), информация о игре.



Рисунок 11 – Окно игры

ЗАКЛЮЧЕНИЕ

Во время разработки курсового проекта были получены навыки:

* Проектирования БД для ИС,
* Создания физической модели в СУБД MS SQL Server,
* Работы с системой для построения клиентских приложений WPF,
* Работы с технологией ADO.NET.

В процессе выполнения курсового проекта:

* Изучена предметная область,
* Спроектирована база данных,
* Разработано приложение,
* Осуществлена защита от SQL инъекций.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Руководство по WPF - [metanit.com/sharp/wpf](https://metanit.com/sharp/wpf/)
2. Руководство по C# - [metanit.com/sharp/tutorial](https://metanit.com/sharp/tutorial/)
3. Руководство по MS SQL Server - [metanit.com/sql/sqlserver](https://metanit.com/sql/sqlserver/)
4. Руководство по ADO.NET - [metanit.com/sharp/adonet/](https://metanit.com/sharp/adonet/)
5. Руководство по C# - [professorweb.ru](http://professorweb.ru/)
6. НОУ Интуит. Основы SQL - [intuit.ru/studies/courses/5/5/info](https://www.intuit.ru/studies/courses/5/5/info)
7. Техническая документация Microsoft - [docs.microsoft.com/ru-ru](https://docs.microsoft.com/ru-ru/)
8. C# 6.0 Справочник. Полное описание языка. Джозеф Албахари и Бен Албахари
9. Microsoft SQL Server 2012. Основы T-SQL. Ицик Бен-Ган
10. WPF в .NET 4 с примерами на C# 2010 для профессионалов. Мэтью Мак-Дональд

ПРИЛОЖЕНИЕ А. SQL СКРИПТЫ НА СОЗДАНИЕ И ЗАПОЛНЕНИЕ БАЗЫ ДАННЫХ, НА СОЗДАНИЕ УЧЕТНЫХ ЗАПИСЕЙ ПОЛЬЗОВАТЕЛЕЙ, НА НАЗНАЧЕНИЕ ПРИВИЛЕГИЙ

Запрос на создание базы данных:

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

-- = 1. База данных

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

USE master;

-- Удаление существующей базы данных

IF DB\_ID('GameDatabase') IS NOT NULL

DROP DATABASE GameDatabase;

-- Создание новой базы данных

CREATE DATABASE GameDatabase;

GO

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

-- = 2. Добавление пользователей

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

USE GameDatabase;

-- Добавление пользователя с правами на чтение

IF EXISTS (SELECT \* FROM sys.sql\_logins WHERE name = 'reader')

DROP LOGIN reader

GO

-- Добавление логина

CREATE LOGIN reader WITH PASSWORD = 'reader'

GO

IF EXISTS (SELECT \* FROM sys.database\_principals WHERE name = 'reader')

DROP USER reader

GO

CREATE USER reader FOR LOGIN reader

EXEC sp\_addrolemember 'db\_datareader', 'reader'

-- Добавление администратора

IF EXISTS (SELECT \* FROM sys.sql\_logins WHERE name = 'admin')

DROP LOGIN admin

GO

-- Добавление логина

CREATE LOGIN admin WITH PASSWORD = 'admin'

GO

IF EXISTS (SELECT \* FROM sys.database\_principals WHERE name = 'admin')

DROP USER admin

GO

CREATE USER admin FOR LOGIN admin

EXEC sp\_addrolemember 'db\_datareader', 'admin'

EXEC sp\_addrolemember 'db\_datawriter', 'admin'

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

-- = 3. Таблицы

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

USE GameDatabase;

-- Таблица "Игры"

CREATE TABLE GameDatabase.dbo.Games

(

-- Столбцы

id INTEGER NOT NULL IDENTITY,

title NVARCHAR(255) NOT NULL,

summary NVARCHAR(MAX) NOT NULL,

website NVARCHAR(255) NOT NULL,

release\_date DATE NULL

-- Ограничения

CONSTRAINT PK\_Games

PRIMARY KEY CLUSTERED (id ASC)

);

-- Таблица "Разработчики"

CREATE TABLE GameDatabase.dbo.Developers

(

-- Столбцы

id INTEGER NOT NULL IDENTITY,

title NVARCHAR(255) NOT NULL

-- Ограничения

CONSTRAINT PK\_Developers

PRIMARY KEY CLUSTERED (id ASC)

);

-- Таблица "Игры - Разработчкики"

CREATE TABLE GameDatabase.dbo.Games\_Developers

(

-- Столбцы

game\_id INTEGER NOT NULL,

developer\_id INTEGER NOT NULL

-- Ограничения

CONSTRAINT PK\_GamesDevelopers

PRIMARY KEY CLUSTERED (game\_id ASC, developer\_id ASC),

CONSTRAINT FK\_GamesDevelopers\_Games

FOREIGN KEY (game\_id)

REFERENCES dbo.Games (id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_GamesDevelopers\_Developers

FOREIGN KEY (developer\_id)

REFERENCES dbo.Developers (id)

ON DELETE CASCADE

ON UPDATE CASCADE

);

-- Таблица "Издатели"

CREATE TABLE GameDatabase.dbo.Publishers

(

-- Столбцы

id INTEGER NOT NULL IDENTITY,

title NVARCHAR(255) NOT NULL

-- Ограничения

CONSTRAINT PK\_Publishers

PRIMARY KEY CLUSTERED (id ASC)

);

-- Таблица "Игры - Издатели"

CREATE TABLE GameDatabase.dbo.Games\_Publishers

(

-- Столбцы

game\_id INTEGER NOT NULL,

publisher\_id INTEGER NOT NULL

-- Ограничения

CONSTRAINT PK\_GamesPublishers

PRIMARY KEY CLUSTERED (game\_id ASC, publisher\_id ASC),

CONSTRAINT FK\_GamesPublishers\_Games

FOREIGN KEY (game\_id)

REFERENCES dbo.Games (id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_GamesPublishers\_Publishers

FOREIGN KEY (publisher\_id)

REFERENCES dbo.Publishers (id)

ON DELETE CASCADE

ON UPDATE CASCADE

);

-- Таблица "Жанры"

CREATE TABLE GameDatabase.dbo.Genres

(

-- Столбцы

id INTEGER NOT NULL IDENTITY,

title NVARCHAR(255) NOT NULL

-- Ограничения

CONSTRAINT PK\_Genres

PRIMARY KEY CLUSTERED (id ASC)

);

-- Таблица "Игры - Жанры"

CREATE TABLE GameDatabase.dbo.Games\_Genres

(

-- Столбцы

game\_id INTEGER NOT NULL,

genre\_id INTEGER NOT NULL

-- Ограничения

CONSTRAINT PK\_GamesGenres

PRIMARY KEY CLUSTERED (game\_id ASC, genre\_id ASC)

CONSTRAINT FK\_GamesGenres\_Games

FOREIGN KEY (game\_id)

REFERENCES dbo.Games (id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_GamesGenres\_Genres

FOREIGN KEY (genre\_id)

REFERENCES dbo.Genres (id)

ON DELETE CASCADE

ON UPDATE CASCADE

);

-- Таблица "Платформы"

CREATE TABLE GameDatabase.dbo.Platforms

(

-- Столбцы

id INTEGER NOT NULL IDENTITY,

title NVARCHAR(255) NOT NULL

-- Ограничения

CONSTRAINT PK\_Platforms

PRIMARY KEY CLUSTERED (id ASC)

);

-- Таблица "Игры - Платформы"

CREATE TABLE GameDatabase.dbo.Games\_Platforms

(

-- Столбцы

game\_id INTEGER NOT NULL,

platform\_id INTEGER NOT NULL

-- Ограничения

CONSTRAINT PK\_GamesPlatforms

PRIMARY KEY CLUSTERED (game\_id ASC, platform\_id ASC),

CONSTRAINT FK\_GamesPlatforms\_Games

FOREIGN KEY (game\_id)

REFERENCES dbo.Games (id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_GamesPlatforms\_Platforms

FOREIGN KEY (platform\_id)

REFERENCES dbo.Platforms (id)

ON DELETE CASCADE

ON UPDATE CASCADE

);

-- Таблица "Рецензенты"

CREATE TABLE GameDatabase.dbo.Reviewers

(

-- Столбцы

id INTEGER NOT NULL IDENTITY,

title NVARCHAR(255) NOT NULL

-- Ограничения

CONSTRAINT PK\_Reviewers

PRIMARY KEY CLUSTERED (id ASC)

);

-- Таблица "Рецензии"

CREATE TABLE GameDatabase.dbo.Reviews

(

-- Столбцы

game\_id INTEGER NOT NULL,

reviewer\_id INTEGER NOT NULL,

summary NVARCHAR(MAX) NOT NULL,

web\_address NVARCHAR(255) NOT NULL,

score TINYINT NULL

-- Ограничения

CONSTRAINT PK\_Reviews

PRIMARY KEY CLUSTERED (game\_id ASC, reviewer\_id ASC)

CONSTRAINT FK\_Reviews\_Games

FOREIGN KEY (game\_id)

REFERENCES dbo.Games (id)

ON DELETE CASCADE

ON UPDATE CASCADE,

CONSTRAINT FK\_Reviews\_Reviewers

FOREIGN KEY (reviewer\_id)

REFERENCES dbo.Reviewers (id)

ON DELETE CASCADE

ON UPDATE CASCADE

);

GO

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

-- = 4. Хранимые процедуры

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

-- Добавление игр

CREATE OR ALTER PROCEDURE StrProc\_AddGame

@title NVARCHAR(255),

@release DATE,

@website NVARCHAR(255),

@summary NVARCHAR(MAX)

AS

BEGIN

INSERT INTO dbo.Games (title, release\_date, website, summary)

VALUES (@title, @release, @website, @summary)

END;

GO

GRANT EXECUTE ON dbo.StrProc\_AddGame TO admin

GO

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

-- = \*. Возврат в master

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

USE master;

ПРИЛОЖЕНИЕ Б. КОД ПРОГРАММЫ

Код программы состоит из следующих модулей:

* LoginWindow.xaml
* LoginWindow.xaml.cs
* MainWindow.xaml
* MainWindow.xaml.cs
* GameWindow.xaml
* GameWindow.xaml.cs
* EditGameWindow.xaml
* EditGameWindow.xaml.cs
* EditWindow.xaml
* EditWindow.xaml.cs

Б.1 Код LoginWindow.xaml

В модуле LoginWindow.xaml содержится декларативное описание интерфейса окна авторизации на языке XAML:

<Window x:Class="GameDatabase\_App.LoginWindow"

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:GameDatabase\_App"

mc:Ignorable="d"

Title="Авторизация"

Width="370" Height="300" ResizeMode="NoResize">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Padding" Value="5"/>

</Style>

</Window.Resources>

<StackPanel Margin="10">

<Label Content="Логин:"/>

<TextBox x:Name="Login"/>

<Label Content="Пароль:"/>

<PasswordBox x:Name="Password"/>

<Label Name="InvalidLogOrPassHint"

Foreground="Red"

Content="Введен неверный логин или пароль"

Visibility="Hidden"/>

<Button Content="Вход"

Click="LoginButton\_Click"

IsDefault="True"/>

<Button Content="Вход с помощью Windows аутентификации"

Click="WinAuthButton\_Click"/>

<Button Content="Вход в режиме чтения"

Click="ReaderButton\_Click"/>

</StackPanel>

</Window>

Б.2 Код LoginWindow.xaml.cs

В модуле LoginWindow.xaml.cs содержится описание работы окна авторизации на языке C#:

using System.Windows;

using System.Data.SqlClient;

namespace GameDatabase\_App

{

/// <summary>

/// Логика взаимодействия для LoginWindow.xaml

/// </summary>

public partial class LoginWindow : Window

{

public LoginWindow()

{

InitializeComponent();

}

// Вход по логину и паролю SQL

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

{

try

{

SqlConnectionStringBuilder sqlConnectionString = new SqlConnectionStringBuilder(Properties.Settings.Default.userConnection)

{

UserID = Login.Text,

Password = Password.Password

};

using (SqlConnection connection = new SqlConnection(sqlConnectionString.ConnectionString))

{

connection.Open();

if (connection.State == System.Data.ConnectionState.Open)

{

Properties.Settings.Default.userConnection = sqlConnectionString.ConnectionString;

using (SqlCommand command = new SqlCommand("SELECT PERMISSIONS(OBJECT\_ID('Games', 'U'))", connection))

{

int value = (int)command.ExecuteScalar();

if (value > 20000)

{

OpenMainWindow(true);

}

else

{

OpenMainWindow();

}

}

}

}

}

catch (SqlException)

{

InvalidLogOrPassHint.Visibility = Visibility.Visible;

}

}

// Вход в режиме администратора с помощью Win аутентификации

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

{

SqlConnectionStringBuilder sqlConnectionString = new SqlConnectionStringBuilder(Properties.Settings.Default.userConnection)

{

IntegratedSecurity = true

};

Properties.Settings.Default.userConnection = sqlConnectionString.ConnectionString;

OpenMainWindow(true);

}

// Вход в режиме чтения

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

{

SqlConnectionStringBuilder sqlConnectionString = new SqlConnectionStringBuilder(Properties.Settings.Default.userConnection)

{

IntegratedSecurity = true

};

Properties.Settings.Default.userConnection = sqlConnectionString.ConnectionString;

OpenMainWindow(false);

}

// Открытие главного меню

private void OpenMainWindow(bool isAdmin = false)

{

MainWindow window = new MainWindow(isAdmin);

window.Show();

Close();

}

}

}

Б.3 Код MainWindow.xaml

В модуле MainWindow.xaml содержится декларативное описание интерфейса главного окна на языке XAML:

<Window x:Class="GameDatabase\_App.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:GameDatabase\_App"

mc:Ignorable="d"

MinHeight="600" MinWidth="800"

Title="Magnum opus">

<Window.Resources>

<Style x:Key="EditButtons" TargetType="Button">

<EventSetter Event="Click" Handler="EditButton\_Click"/>

<Setter Property="Padding" Value="5"/>

<Setter Property="Visibility" Value="Collapsed"/>

<Setter Property="Content" Value="Редактировать"/>

</Style>

</Window.Resources>

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<ScrollViewer Grid.Row="1">

<StackPanel Width="300" Margin="5">

<!-- Параметры сортировки -->

<Label Content="Сортировать по:"/>

<ComboBox x:Name="SortByComboBox">

<ComboBoxItem Tag="0" Content="Название (По возростанию)" IsSelected="True"/>

<ComboBoxItem Tag="1" Content="Название (По убыванию)"/>

<ComboBoxItem Tag="2" Content="Дата (По возростанию)"/>

<ComboBoxItem Tag="3" Content="Дата (По убыванию)"/>

<ComboBoxItem Tag="4" Content="Оценка (По возростанию)"/>

<ComboBoxItem Tag="5" Content="Оценка (По убыванию)"/>

</ComboBox>

<!-- Параметры поиска по средней оценки -->

<Label Content="Средняя оценка:"/>

<DockPanel VerticalAlignment="Center" Margin="5">

<TextBox IsEnabled="False"

Width="40"

Text="{Binding ElementName=GameScoreFromSlider, Path=Value, UpdateSourceTrigger=PropertyChanged}" DockPanel.Dock="Right" Margin="5 0" VerticalContentAlignment="Center" />

<Label Content="С "

Width="30"/>

<Slider x:Name="GameScoreFromSlider"

Minimum="0"

Maximum="100"

Value="0"

TickPlacement="BottomRight"

TickFrequency="5"

IsSnapToTickEnabled="True"

ValueChanged="GameScoreFromSlider\_ValueChanged"/>

</DockPanel>

<DockPanel VerticalAlignment="Center" Margin="5">

<TextBox IsEnabled="False"

Width="40"

Text="{Binding ElementName=GameScoreToSlider, Path=Value, UpdateSourceTrigger=PropertyChanged}" DockPanel.Dock="Right" Margin="5 0" VerticalContentAlignment="Center"/>

<Label Content="До" Width="30"/>

<Slider x:Name="GameScoreToSlider"

Minimum="0"

Maximum="100"

Value="100"

TickPlacement="BottomRight"

TickFrequency="5"

IsSnapToTickEnabled="True"

ValueChanged="GameScoreToSlider\_ValueChanged"/>

</DockPanel>

<!-- Параметры поиска по названию игры -->

<Label Content="Название игры:"/>

<TextBox x:Name="GameTitleSearchTextBlock" />

<!-- Параметры поиска по дате выхода -->

<Label Content="Дата выхода:"/>

<StackPanel Orientation="Horizontal">

<Label Content="с"/>

<DatePicker x:Name="GameReleaseFromDatePicker"/>

<Label Content="по"/>

<DatePicker x:Name="GameReleaseToDatePicker"/>

</StackPanel>

<!-- Параметры поиска по разработчикам -->

<GroupBox Padding="0 5">

<GroupBox.Header>

<StackPanel Orientation="Horizontal"

Margin="0 5">

<Button x:Name="EditDevelopersButton"

Style="{StaticResource EditButtons}"

Tag="0"/>

<Label Content="Разработчики:"/>

</StackPanel>

</GroupBox.Header>

<ScrollViewer VerticalScrollBarVisibility="Auto"

MaxHeight="100">

<StackPanel x:Name="SearchDevelopersList"/>

</ScrollViewer>

</GroupBox>

<!-- Параметры поиска по издателям -->

<GroupBox Padding="0 5">

<GroupBox.Header>

<StackPanel Orientation="Horizontal" Margin="0 5">

<Button x:Name="EditPublishersButton"

Style="{StaticResource EditButtons}"

Tag="1"/>

<Label Content="Издатели:"/>

</StackPanel>

</GroupBox.Header>

<ScrollViewer VerticalScrollBarVisibility="Auto"

MaxHeight="100">

<StackPanel x:Name="SearchPublishersList"/>

</ScrollViewer>

</GroupBox>

<!-- Параметры поиска по жанрам -->

<GroupBox Padding="0 5">

<GroupBox.Header>

<StackPanel Orientation="Horizontal" Margin="0 5">

<Button x:Name="EditGenresButton"

Style="{StaticResource EditButtons}"

Tag="2"/>

<Label Content="Жанры:"/>

</StackPanel>

</GroupBox.Header>

<ScrollViewer VerticalScrollBarVisibility="Auto"

MaxHeight="100">

<StackPanel x:Name="SearchGenresList"/>

</ScrollViewer>

</GroupBox>

<!-- Параметры поиска по платформам -->

<GroupBox Padding="0 5">

<GroupBox.Header>

<StackPanel Orientation="Horizontal" Margin="0 5">

<Button x:Name="EditPlatformsButton"

Style="{StaticResource EditButtons}"

Tag="3"/>

<Label Content="Платформы:"/>

</StackPanel>

</GroupBox.Header>

<ScrollViewer VerticalScrollBarVisibility="Auto"

MaxHeight="100">

<StackPanel x:Name="SearchPlatformsList"/>

</ScrollViewer>

</GroupBox>

</StackPanel>

</ScrollViewer>

<StackPanel Grid.Row="2" Margin="0 10">

<Button x:Name="SearchButton"

Click="SearchButton\_Click"

Content="Поиск"

IsDefault="True"

Padding="5"/>

<Button x:Name="ClearSearchButton"

Click="ClearSearchButton\_Click"

Content="Очистить"

Padding="5"/>

<Button x:Name="AddGameButton"

Click="AddGame\_Click"

Content="Добавить игру"

Padding="5"

Visibility="Collapsed"/>

</StackPanel>

<ScrollViewer Grid.Column="1"

Grid.Row="1"

Grid.RowSpan="2">

<Grid>

<TextBlock x:Name="NoResultsTextBlock"

FontSize="36"

VerticalAlignment="Center"

HorizontalAlignment="Center"

TextWrapping="Wrap"

TextAlignment="Center"

Text="По указанным параметрам поиска результатов не найдено"/>

<StackPanel x:Name="GamesList"

MaxWidth="1000"/>

</Grid>

</ScrollViewer>

</Grid>

</Window>

Б.4 Код MainWindow.xaml.cs

В модуле MainWindow.xaml.cs содержится описание работы главного окна на языке C#:

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;

namespace GameDatabase\_App

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow(bool isAdmin = false)

{

InitializeComponent();

if(isAdmin)

{

AddGameButton.Visibility = Visibility.Visible;

EditDevelopersButton.Visibility =

EditGenresButton.Visibility =

EditPublishersButton.Visibility =

EditPlatformsButton.Visibility = Visibility.Visible;

}

Tag = isAdmin;

ShowGames();

UpdateDevelopersList();

UpdatePublishersList();

UpdateGenresList();

UpdatePlatformsList();

}

// Поиск

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

{

ShowGames();

}

// Очистка параметров поиска

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

{

GameTitleSearchTextBlock.Clear();

GameReleaseFromDatePicker.SelectedDate = null;

GameReleaseToDatePicker.SelectedDate = null;

GameScoreFromSlider.Value = 0;

GameScoreToSlider.Value = 100;

foreach (CheckBox developer in SearchDevelopersList.Children)

{

developer.IsChecked = false;

}

foreach (CheckBox publisher in SearchPublishersList.Children)

{

publisher.IsChecked = false;

}

foreach (CheckBox genre in SearchGenresList.Children)

{

genre.IsChecked = false;

}

foreach (CheckBox platform in SearchPlatformsList.Children)

{

platform.IsChecked = false;

}

SortByComboBox.SelectedIndex = 0;

ShowGames();

}

// Значение слайдера не может быть выше значения второго слайдера

private void GameScoreFromSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

if (e.NewValue > GameScoreToSlider.Value)

GameScoreFromSlider.Value = e.OldValue;

}

// Значение слайдера не может быть ниже значения первого слайдера

private void GameScoreToSlider\_ValueChanged(object sender, RoutedPropertyChangedEventArgs<double> e)

{

if (e.NewValue < GameScoreFromSlider.Value)

GameScoreToSlider.Value = e.OldValue;

}

// Открытие окна игры

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

{

GameWindow gameWindow = new GameWindow((int)((Button)sender).Tag, (bool)Tag ? true : false);

gameWindow.Show();

}

// Открытие окна редактирования игры

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

{

EditGameWindow window = new EditGameWindow((int)((Button)sender).Tag);

window.ShowDialog();

ShowGames();

}

// Открытие окна добавления игры

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

{

EditGameWindow window = new EditGameWindow();

window.ShowDialog();

ShowGames();

}

// Удаление игры

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

{

MessageBoxResult result = MessageBox.Show($"Вы действительно хотите удалить {((Label)(((Grid)(((StackPanel)(((Button)sender).Parent)).Parent)).Children[1])).Content}?", "Удаление игры", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

try

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection.Open();

using (SqlCommand command = new SqlCommand($@"DELETE FROM dbo.Games WHERE dbo.Games.id = @id", connection))

{

command.Parameters.Add(new SqlParameter("@id", ((Button)sender).Tag));

command.ExecuteNonQuery();

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

ShowGames();

}

// Открытие окна редактирования

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

{

string id = ((Button)sender).Tag.ToString();

EditWindow window = new EditWindow(int.Parse(id));

window.ShowDialog();

switch (id)

{

case "0":

UpdateDevelopersList();

break;

case "1":

UpdatePublishersList();

break;

case "2":

UpdateGenresList();

break;

case "3":

UpdatePlatformsList();

break;

}

ShowGames();

}

// Получение списка игр по указанным параметрам поиска

private void ShowGames()

{

// Очистка списка

NoResultsTextBlock.Visibility = Visibility.Hidden;

GamesList.Children.Clear();

try

{

// Подключение

using (SqlConnection connection = new SqlConnection() { ConnectionString = Properties.Settings.Default.userConnection })

{

// Открытие подключения

connection.Open();

// Команда sql

using (SqlCommand command = GenerateSqlCommand(connection))

{

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

while (dataReader.Read())

{

// Добавление разделителя строк

// -----------------------------------------------

if (GamesList.Children.Count > 0)

GamesList.Children.Add(new Separator());

// -----------------------------------------------

GamesList.Children.Add(

GenerateGameTile(

dataReader.GetInt32(0),

dataReader.GetString(1),

dataReader.IsDBNull(3) ? null : (DateTime?)dataReader.GetDateTime(3),

dataReader.IsDBNull(4) ? null : (int?)dataReader.GetInt32(4),

dataReader.GetString(2)));

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

if (GamesList.Children.Count == 0)

NoResultsTextBlock.Visibility = Visibility.Visible;

}

private Grid GenerateGameTile(int id, string title, DateTime? release, int? score, string summary)

{

// Создание Grid в который будет компоноваться Tile

// -----------------------------------------------

Grid gameTileGrid = new Grid()

{

Height = 210,

Margin = new Thickness(2)

};

// Столбцы

gameTileGrid.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto });

gameTileGrid.ColumnDefinitions.Add(new ColumnDefinition());

gameTileGrid.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto });

// Строки

gameTileGrid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

gameTileGrid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

gameTileGrid.RowDefinitions.Add(new RowDefinition());

gameTileGrid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

// -----------------------------------------------

// Обложка игры

// -----------------------------------------------

Border gameCoverBorder = new Border()

{

BorderThickness = new Thickness(1),

BorderBrush = new SolidColorBrush(Colors.Black),

Margin = new Thickness(5),

Height = 180,

Width = 120,

VerticalAlignment = VerticalAlignment.Top

};

gameCoverBorder.Child = new Image()

{

Stretch = Stretch.Uniform

};

gameTileGrid.Children.Add(gameCoverBorder);

Grid.SetRowSpan(gameCoverBorder, 4);

// -----------------------------------------------

// Название игры

// -----------------------------------------------

Label gameTitleLabel = new Label()

{

Content = title,

FontSize = 18

};

gameTileGrid.Children.Add(gameTitleLabel);

Grid.SetColumn(gameTitleLabel, 1);

// -----------------------------------------------

// Дата выхода игры

// -----------------------------------------------

Label gameReleaseDateLabel = new Label()

{

Content = $"Дата выхода: " +

$"{(release == null ? "TBA" : DateTime.Parse(release.ToString()).ToShortDateString())}",

FontSize = 10

};

gameTileGrid.Children.Add(gameReleaseDateLabel);

Grid.SetColumn(gameReleaseDateLabel, 1);

Grid.SetRow(gameReleaseDateLabel, 1);

// -----------------------------------------------

// Оценка игры

// -----------------------------------------------

Border gameScoreBorder = new Border()

{

Background = score == null || score <= 50 ? new SolidColorBrush(Colors.Red) : (score <= 70 ? new SolidColorBrush(Colors.Gold) : new SolidColorBrush(Colors.YellowGreen)),

Margin = new Thickness(5),

BorderThickness = new Thickness(1),

BorderBrush = new SolidColorBrush(Colors.Black),

Height = 50,

Width = 50

};

gameScoreBorder.Child = new Label()

{

FontSize = 24,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Center,

Content = score == null ? "-" : score.ToString()

};

gameTileGrid.Children.Add(gameScoreBorder);

Grid.SetColumn(gameScoreBorder, 2);

Grid.SetRow(gameScoreBorder, 0);

Grid.SetRowSpan(gameScoreBorder, 2);

// -----------------------------------------------

// Описание игры

// -----------------------------------------------

TextBlock gameSummaryTextBlock = new TextBlock()

{

Text = summary,

TextWrapping = TextWrapping.Wrap,

Margin = new Thickness(3),

TextTrimming = TextTrimming.WordEllipsis,

};

gameTileGrid.Children.Add(gameSummaryTextBlock);

Grid.SetColumn(gameSummaryTextBlock, 1);

Grid.SetRow(gameSummaryTextBlock, 2);

Grid.SetColumnSpan(gameSummaryTextBlock, 2);

// -----------------------------------------------

// Панель кнопок в нижней части плитки

// -----------------------------------------------

StackPanel buttons = new StackPanel()

{

Orientation = Orientation.Horizontal,

HorizontalAlignment = HorizontalAlignment.Right,

Margin = new Thickness(5),

};

gameTileGrid.Children.Add(buttons);

Grid.SetColumn(buttons, 1);

Grid.SetColumnSpan(buttons, 3);

Grid.SetRow(buttons, 4);

// -----------------------------------------------

if ((bool)Tag)

{

// Кнопка удалить

// -----------------------------------------------

Button DeleteButton = new Button()

{

Tag = id,

Padding = new Thickness(3),

HorizontalAlignment = HorizontalAlignment.Right,

Content = "Удалить"

};

DeleteButton.Click += DeleteGame\_Click;

buttons.Children.Add(DeleteButton);

// -----------------------------------------------

// Кнопка редактировать

// -----------------------------------------------

Button EditButton = new Button()

{

Tag = id,

Padding = new Thickness(3),

HorizontalAlignment = HorizontalAlignment.Right,

Content = "Редактировать"

};

EditButton.Click += GameEditButton\_Click;

buttons.Children.Add(EditButton);

// -----------------------------------------------

}

// Кнопка подробнее

// -----------------------------------------------

Button gameMoreInfoButton = new Button()

{

Tag = id,

Padding = new Thickness(3),

Content = "Подробнее..."

};

gameMoreInfoButton.Click += GameMoreInfoButton\_Click;

buttons.Children.Add(gameMoreInfoButton);

// -----------------------------------------------

return gameTileGrid;

}

// Формирование поискового запроса для SQL Server

[System.Diagnostics.CodeAnalysis.SuppressMessage("Security", "CA2100:Review SQL queries for security vulnerabilities", Justification = "<Ожидание>")]

private SqlCommand GenerateSqlCommand(SqlConnection connection)

{

// Основной текст запроса, в случае отсутствие условий поиска выполняется только он

SqlCommand command = new SqlCommand()

{

CommandText =

" SELECT DISTINCT dbo.Games.id, dbo.Games.title, dbo.Games.summary, dbo.Games.release\_date, GameScore.avg\_score" +

" FROM dbo.Games" +

" LEFT OUTER JOIN (" +

" SELECT game\_id, AVG(score) AS avg\_score" +

" FROM dbo.Reviews" +

" GROUP BY dbo.Reviews.game\_id) AS GameScore" +

" ON dbo.Games.id = GameScore.game\_id",

Connection = connection

};

// Количество выделенных объектов

int count = 0;

// Поиск по разработчикам

foreach (CheckBox item in SearchDevelopersList.Children)

{

if ((bool)item.IsChecked)

{

command.Parameters.Add(new SqlParameter($"@developer{item.Tag}", item.Tag));

if (count == 0)

command.CommandText +=

" INNER JOIN dbo.Games\_Developers" +

" ON dbo.Games\_Developers.game\_id = dbo.Games.id" +

$" AND dbo.Games\_Developers.developer\_id IN (@developer{item.Tag}";

else

command.CommandText += $", @developer{item.Tag}";

count++;

}

}

if (count != 0)

{

command.CommandText += ')';

}

count = 0;

// Поиск по издателям

foreach (CheckBox item in SearchPublishersList.Children)

{

if ((bool)item.IsChecked)

{

command.Parameters.Add(new SqlParameter($"@publisher{item.Tag}", item.Tag));

if (count == 0)

command.CommandText +=

" INNER JOIN dbo.Ganes\_Publishers" +

" ON dbo.Games\_Publishers.game\_id = dbo.Games.id" +

$" AND dbo.Games\_Publishers IN (@publisher{item.Tag}";

else

command.CommandText += $", @publisher{item.Tag}";

count++;

}

}

if (count != 0)

{

command.CommandText += ')';

}

count = 0;

// Поиск по жанрам

foreach (CheckBox item in SearchGenresList.Children)

{

if ((bool)item.IsChecked)

{

command.Parameters.Add(new SqlParameter($"@genre{item.Tag}", item.Tag));

if (count == 0)

command.CommandText +=

" INNER JOIN dbo.Games\_Genres" +

" ON dbo.Games\_Genres.game\_id = dbo.Games.id" +

$" AND dbo.Games\_Genres.genre\_id IN (@genre{item.Tag}";

else

command.CommandText += $@", @genre{item.Tag}";

count++;

}

}

if (count != 0)

{

command.CommandText += ')';

}

count = 0;

// Поиск по платформам

foreach (CheckBox item in SearchPlatformsList.Children)

{

if ((bool)item.IsChecked)

{

command.Parameters.Add(new SqlParameter($"@platform{item.Tag}", item.Tag));

if (count == 0)

command.CommandText += " INNER JOIN dbo.Games\_Platforms " +

" ON dbo.Games\_Platforms.game\_id = dbo.Games.id" +

$" AND dbo.Games\_Platforms.platform\_id IN (@platform{item.Tag}";

else

command.CommandText += $@", @platform{item.Tag}";

count++;

}

}

if (count != 0)

{

command.CommandText += ')';

}

// Количество условий поиска

int terms = 0;

// Поиск по названию игры

if (GameTitleSearchTextBlock.Text.Length > 0)

{

command.Parameters.Add(new SqlParameter("@title", '%' + GameTitleSearchTextBlock.Text + '%'));

command.CommandText +=

" WHERE dbo.Games.Title LIKE @title";

terms++;

}

// Поиск по дате выхода (с)

if(GameReleaseFromDatePicker.SelectedDate != null)

{

if (terms == 0)

command.CommandText += " WHERE";

else

command.CommandText += " AND";

command.Parameters.Add(new SqlParameter("@releaseDateFrom", GameReleaseFromDatePicker.SelectedDate));

command.CommandText += " dbo.Games.release\_date >= @releaseDateFrom";

terms++;

}

// Поиск по дате выхода (до)

if (GameReleaseToDatePicker.SelectedDate != null)

{

if (terms == 0)

command.CommandText += " WHERE";

else

command.CommandText += " AND";

command.Parameters.Add(new SqlParameter("@releaseDateTo", GameReleaseToDatePicker.SelectedDate));

command.CommandText += " dbo.Games.release\_date <= @releaseDateTo";

terms++;

}

// Поиск по средней оценке (>)

if (GameScoreFromSlider.Value > 0)

{

if (terms == 0)

command.CommandText += " WHERE";

else

command.CommandText += " AND";

command.Parameters.Add(new SqlParameter("@scoreFrom", GameScoreFromSlider.Value));

command.CommandText += " GameScore.avg\_score >= @scoreFrom";

terms++;

}

// Поиск по средней оценке (<)

if (GameScoreToSlider.Value < 100)

{

if (terms == 0)

command.CommandText += " WHERE";

else

command.CommandText += " AND";

command.Parameters.Add(new SqlParameter("@scoreTo", GameScoreToSlider.Value));

command.CommandText += " GameScore.avg\_score <= @scoreTo";

}

switch (((ComboBoxItem)SortByComboBox.SelectedItem).Tag)

{

case "0":

command.CommandText += " ORDER BY dbo.Games.title ASC";

break;

case "1":

command.CommandText += " ORDER BY dbo.Games.title DESC";

break;

case "2":

command.CommandText += " ORDER BY dbo.Games.release\_date ASC";

break;

case "3":

command.CommandText += " ORDER BY dbo.Games.release\_date DESC";

break;

case "4":

command.CommandText += " ORDER BY GameScore.avg\_score ASC";

break;

case "5":

command.CommandText += " ORDER BY GameScore.avg\_score DESC";

break;

}

return command;

}

// Обновление списка разработиков

private void UpdateDevelopersList()

{

try

{

// Подключение

using (SqlConnection connection = new SqlConnection() { ConnectionString = Properties.Settings.Default.userConnection })

{

// Открытие подключения

connection.Open();

// Отображение списка разработчиков

using (SqlCommand command = new SqlCommand("SELECT id, title FROM dbo.Developers ORDER BY title ASC", connection))

{

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

SearchDevelopersList.Children.Clear();

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

Tag = dataReader.GetInt32(0),

Content = dataReader.GetString(1),

FontSize = 11

};

SearchDevelopersList.Children.Add(item);

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе получения данных произошла ошибка:\n{ex}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

// Обновление списка издателей

private void UpdatePublishersList()

{

try

{

// Подключение

using (SqlConnection connection = new SqlConnection() { ConnectionString = Properties.Settings.Default.userConnection })

{

// Открытие подключения

connection.Open();

// Отображение списка разработчиков

using (SqlCommand command = new SqlCommand("SELECT id, title FROM dbo.Publishers ORDER BY title ASC", connection))

{

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

SearchPublishersList.Children.Clear();

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

Tag = dataReader.GetInt32(0),

Content = dataReader.GetString(1),

FontSize = 11

};

SearchPublishersList.Children.Add(item);

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе получения данных произошла ошибка:\n{ex}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

// Обновление списка жанров

private void UpdateGenresList()

{

try

{

// Подключение

using (SqlConnection connection = new SqlConnection() { ConnectionString = Properties.Settings.Default.userConnection })

{

// Открытие подключения

connection.Open();

// Отображение списка разработчиков

using (SqlCommand command = new SqlCommand("SELECT id, title FROM dbo.Genres ORDER BY title ASC", connection))

{

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

SearchGenresList.Children.Clear();

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

Tag = dataReader.GetInt32(0),

Content = dataReader.GetString(1),

FontSize = 11

};

SearchGenresList.Children.Add(item);

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе получения данных произошла ошибка:\n{ex}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

// Обновление списка платформ

private void UpdatePlatformsList()

{

try

{

// Подключение

using (SqlConnection connection = new SqlConnection() { ConnectionString = Properties.Settings.Default.userConnection })

{

// Открытие подключения

connection.Open();

// Отображение списка разработчиков

using (SqlCommand command = new SqlCommand("SELECT id, title FROM dbo.Platforms ORDER BY title ASC", connection))

{

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

SearchPlatformsList.Children.Clear();

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

Tag = dataReader.GetInt32(0),

Content = dataReader.GetString(1),

FontSize = 11

};

SearchPlatformsList.Children.Add(item);

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе получения данных произошла ошибка:\n{ex}", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

Б.5 Код GameWindow.xaml

В модуле GameWindow.xaml содержится декларативное описание интерфейса окна игры на языке XAML:

<Window x:Class="GameDatabase\_App.GameWindow"

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:GameDatabase\_App"

mc:Ignorable="d"

MinHeight="600" MinWidth="800">

<Grid MaxWidth="1000">

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition/>

</Grid.RowDefinitions>

<Grid Grid.Row="0" >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition Height="Auto"/>

<RowDefinition />

</Grid.RowDefinitions>

<StackPanel Grid.RowSpan="3" >

<Border

Width="120" Height="180"

BorderThickness="1"

BorderBrush="Black"

Margin="10 5">

<Image />

</Border>

<Button x:Name="GameEditButton"

Visibility="Collapsed"

Content="Редактировать"

Click="GameEditButton\_Click"/>

</StackPanel>

<Label x:Name="GameTitle"

Grid.Column="1"

FontSize="36"

Content="Название"/>

<StackPanel Orientation="Horizontal" Grid.Column="1" Grid.Row="1">

<Label FontSize = "10">

<Hyperlink x:Name="GameOfficial" RequestNavigate="Hyperlink\_RequestNavigate"/>

</Label>

<Label x:Name="GameReleaseDate"

FontSize="10"

Content="Дата выхода: 20.10.2020"/>

</StackPanel>

<StackPanel Grid.Column="1" Grid.Row="2" >

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"/>

<RowDefinition />

</Grid.RowDefinitions>

<StackPanel Grid.Row="0" Grid.Column="0">

<Label FontWeight="Bold" Content="Разработчик(и):"/>

<TextBlock x:Name="GameDevelopers"

TextWrapping="Wrap"

Padding="10 0"/>

</StackPanel>

<StackPanel Grid.Row="0" Grid.Column="1">

<Label FontWeight="Bold" Content="Издател(ь/и):"/>

<TextBlock x:Name="GamePublishers"

TextWrapping="Wrap"

Padding="10 0"/>

</StackPanel>

<StackPanel Grid.Row="1" Grid.Column="0">

<Label FontWeight="Bold" Content="Жанр(ы):"/>

<TextBlock x:Name="GameGenres"

TextWrapping="Wrap"

Padding="10 0"/>

</StackPanel>

<StackPanel Grid.Row="1" Grid.Column="1">

<Label FontWeight="Bold" Content="Платформ(а/ы):"/>

<TextBlock x:Name="GamePlatforms"

TextWrapping="Wrap"

Padding="10 0"/>

</StackPanel>

</Grid>

<Label FontWeight="Bold" Content="Описание:"/>

<TextBlock x:Name="GameSummary"

Padding="10 0"

TextWrapping="Wrap"/>

</StackPanel>

<Border x:Name="GameAvgScoreBorder"

Grid.Column="3"

Grid.RowSpan="2"

Background="YellowGreen"

Height="80"

Width="80"

BorderThickness="1"

BorderBrush="Black"

Margin="20 20 20 0"

VerticalAlignment="Top">

<Label x:Name="GameAvgScore"

Content="99"

FontSize="36"

HorizontalContentAlignment="Center"

VerticalContentAlignment="Center"/>

</Border>

</Grid>

<TabControl Grid.Row="1" Margin="5">

<TabItem Header="Рецензии">

<ScrollViewer>

<StackPanel x:Name="ReviewsList" MaxWidth="1000"/>

</ScrollViewer>

</TabItem>

</TabControl>

</Grid>

</Window>

Б.6 Код GameWindow.xaml.cs

В модуле GameWindow.xaml.cs содержится описание работы окна игры на языке C#:

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.Shapes;

using System.Data.SqlClient;

using System.Diagnostics;

namespace GameDatabase\_App

{

/// <summary>

/// Логика взаимодействия для GameWindow.xaml

/// </summary>

public partial class GameWindow : Window

{

public GameWindow(int game\_id, bool adm = false)

{

InitializeComponent();

Tag = game\_id;

ShowGame();

if(adm)

{

GameEditButton.Visibility = Visibility.Visible;

}

}

private void ShowGame()

{

try

{

// Подключение

using (SqlConnection connection = new SqlConnection() { ConnectionString = Properties.Settings.Default.userConnection })

{

// Открытие подключения

connection.Open();

// Получение списка рецензий на игру

ReviewsList.Children.Clear();

using (SqlCommand command = new SqlCommand(

" SELECT dbo.Reviewers.title, dbo.Reviews.summary, dbo.Reviews.score, dbo.Reviews.web\_address" +

" FROM dbo.Reviews INNER JOIN dbo.Reviewers ON dbo.Reviews.reviewer\_id = dbo.Reviewers.id" +

" WHERE game\_id = @game\_id", connection))

{

command.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

while (dataReader.Read())

{

Grid reviewTileGrid = new Grid()

{

//ShowGridLines = true

};

ReviewsList.Children.Add(reviewTileGrid);

ReviewsList.Children.Add(new Separator());

reviewTileGrid.ColumnDefinitions.Add(new ColumnDefinition());

reviewTileGrid.ColumnDefinitions.Add(new ColumnDefinition() { Width = GridLength.Auto });

reviewTileGrid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

reviewTileGrid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

reviewTileGrid.RowDefinitions.Add(new RowDefinition() { Height = GridLength.Auto });

// Название рецензента

// ---------------------------------------

Label reviewerLabel = new Label()

{

Content = dataReader.GetString(0),

FontSize = 18

};

reviewTileGrid.Children.Add(reviewerLabel);

// ---------------------------------------

// Краткое содержание рецензии

// ---------------------------------------

TextBlock reviewSummaryTextBox = new TextBlock()

{

Text = dataReader.GetString(1),

TextWrapping = TextWrapping.Wrap,

Margin = new Thickness(3)

};

Grid.SetRow(reviewSummaryTextBox, 1);

reviewTileGrid.Children.Add(reviewSummaryTextBox);

// ---------------------------------------

// Оценка

// ---------------------------------------

Border reviewScoreBorder = new Border()

{

Background = dataReader.IsDBNull(2) || dataReader.GetByte(2) <= 50 ? new SolidColorBrush(Colors.Red) : (dataReader.GetByte(2) <= 70 ? new SolidColorBrush(Colors.Gold) : new SolidColorBrush(Colors.YellowGreen)),

Margin = new Thickness(5),

BorderThickness = new Thickness(1),

BorderBrush = new SolidColorBrush(Colors.Black),

Height = 50,

Width = 50,

VerticalAlignment = VerticalAlignment.Top

};

reviewScoreBorder.Child = new Label()

{

FontSize = 24,

HorizontalAlignment = HorizontalAlignment.Center,

VerticalAlignment = VerticalAlignment.Center,

Content = dataReader.IsDBNull(2) ? "-" : dataReader.GetByte(2).ToString()

};

Grid.SetColumn(reviewScoreBorder, 1);

Grid.SetRowSpan(reviewScoreBorder, 2);

reviewTileGrid.Children.Add(reviewScoreBorder);

// ---------------------------------------

// Ссылка на рецензию

// ---------------------------------------

Label moreInfoLabel = new Label

{

Content = new Hyperlink()

{

NavigateUri = new Uri(dataReader.GetString(3), UriKind.Relative)

}

};

((Hyperlink)(moreInfoLabel.Content)).Inlines.Add("Читать полностью");

((Hyperlink)(moreInfoLabel.Content)).RequestNavigate += Hyperlink\_RequestNavigate;

reviewTileGrid.Children.Add(moreInfoLabel);

Grid.SetColumnSpan(moreInfoLabel, 2);

Grid.SetRow(moreInfoLabel, 2);

// ---------------------------------------

}

}

}

// Получение информации о игре

using (SqlCommand command = new SqlCommand(

" SELECT dbo.Games.title, dbo.Games.website, dbo.Games.summary, dbo.Games.release\_date, GameScore.avg\_score" +

" FROM dbo.Games" +

" LEFT OUTER JOIN ( SELECT game\_id, AVG(score) AS avg\_score FROM dbo.Reviews GROUP BY dbo.Reviews.game\_id) AS GameScore" +

" ON dbo.Games.id = GameScore.game\_id" +

" WHERE dbo.Games.id = @game\_id", connection))

{

command.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

while (dataReader.Read())

{

// Заголовок окна

Title = dataReader.GetString(0);

// Название игры

GameTitle.Content = dataReader.GetString(0);

// Дата выхода игры

GameReleaseDate.Content = "Дата выхода: " +

$"{ (dataReader.IsDBNull(3) ? "TBA" : DateTime.Parse(dataReader.GetDateTime(3).ToString()).ToShortDateString())}";

// Средняя оценка игры

GameAvgScore.Content = dataReader.IsDBNull(4) ? "-" : dataReader.GetInt32(4).ToString();

// Цвет поля со средней оценкой

GameAvgScoreBorder.Background = dataReader.IsDBNull(4) || dataReader.GetInt32(4) <= 50 ? new SolidColorBrush(Colors.Red) : (dataReader.GetInt32(4) <= 70 ? new SolidColorBrush(Colors.Gold) : new SolidColorBrush(Colors.YellowGreen));

// Описание игры

GameSummary.Text = dataReader.GetString(2);

// Официальный сайт игры

GameOfficial.NavigateUri = new Uri(dataReader.GetString(1), UriKind.Relative);

GameOfficial.Inlines.Clear();

GameOfficial.Inlines.Add(dataReader.GetString(1));

}

}

}

// Разработчики игры

using (SqlCommand command = new SqlCommand(

" SELECT dbo.Developers.title" +

" FROM dbo.Games\_Developers INNER JOIN dbo.Developers" +

" ON dbo.Games\_Developers.developer\_id = dbo.Developers.id" +

" WHERE dbo.Games\_Developers.game\_id = @game\_id", connection))

{

command.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

int count = 0;

while (dataReader.Read())

{

if (count == 0)

GameDevelopers.Text = dataReader.GetString(0);

else

GameDevelopers.Text += $", {dataReader.GetString(0)}";

count++;

}

}

}

// Издатели игры

using (SqlCommand command = new SqlCommand(

" SELECT dbo.Publishers.title" +

" FROM dbo.Games\_Publishers INNER JOIN dbo.Publishers" +

" ON dbo.Games\_Publishers.publisher\_id = dbo.Publishers.id" +

" WHERE dbo.Games\_Publishers.game\_id = @game\_id", connection))

{

command.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

int count = 0;

while (dataReader.Read())

{

if (count == 0)

GamePublishers.Text = dataReader.GetString(0);

else

GamePublishers.Text += $", {dataReader.GetString(0)}";

count++;

}

}

}

// Жанры игры

using (SqlCommand command = new SqlCommand(

" SELECT dbo.Genres.title" +

" FROM dbo.Games\_Genres INNER JOIN dbo.Genres " +

" ON dbo.Games\_Genres.genre\_id = dbo.Genres.id" +

" WHERE dbo.Games\_Genres.game\_id = @game\_id", connection))

{

command.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

int count = 0;

while (dataReader.Read())

{

if (count == 0)

GameGenres.Text = dataReader.GetString(0);

else

GameGenres.Text += ", {dataReader.GetString(0)}";

count++;

}

}

}

// Платформы игры

using (SqlCommand command = new SqlCommand(

" SELECT dbo.Platforms.title" +

" FROM dbo.Games\_Platforms INNER JOIN dbo.Platforms" +

" ON dbo.Games\_Platforms.platform\_id = dbo.Platforms.id" +

" WHERE dbo.Games\_Platforms.game\_id = @game\_id", connection))

{

command.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

// Выполнение запроса

using (SqlDataReader dataReader = command.ExecuteReader())

{

int count = 0;

while (dataReader.Read())

{

if (count == 0)

GamePlatforms.Text = dataReader.GetString(0);

else

GamePlatforms.Text += $", {dataReader.GetString(0)}";

count++;

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

private void Hyperlink\_RequestNavigate(object sender, System.Windows.Navigation.RequestNavigateEventArgs e)

{

try

{

Process.Start(new ProcessStartInfo(e.Uri.IsAbsoluteUri ? e.Uri.AbsoluteUri : "http://" + e.Uri.OriginalString));

}

catch (Exception ex)

{

MessageBox.Show($"В процессе загрузки сайта произошла ошибка:{ex}", "Ошибка загрузки", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

// Открытие окна редактирования игры

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

{

EditGameWindow window = new EditGameWindow((int)Tag);

window.ShowDialog();

ShowGame();

}

}

}

Б.7 Код EditGameWindow.xaml

В модуле EditGameWindow.xaml содержится декларативное описание интерфейса окна редактирования игр на языке XAML:

<Window x:Class="GameDatabase\_App.EditGameWindow"

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:GameDatabase\_App"

mc:Ignorable="d"

Title="Редактирование" MinHeight="600" MinWidth="800">

<Window.Resources>

<Style TargetType="Button">

<Setter Property="Margin" Value="5"/>

</Style>

</Window.Resources>

<Grid MaxWidth="1000">

<Grid.RowDefinitions>

<RowDefinition/>

<RowDefinition Height="Auto"/>

</Grid.RowDefinitions>

<TabControl>

<TabItem Header="Общие">

<ScrollViewer VerticalScrollBarVisibility="Auto">

<StackPanel Margin="10">

<Label Content="Название:\*"/>

<TextBox x:Name="EditGameTitle"/>

<Label Content="Дата выхода:"/>

<DatePicker x:Name="EditGameRelease"/>

<Label Content="Официальный сайт:\*"/>

<TextBox x:Name="EditGameOfficial"/>

<Label Content="Описание:"/>

<TextBox x:Name="EditGameSummary"

TextWrapping="Wrap"

Height="150"/>

<Label Content="Разработчики:"/>

<StackPanel x:Name="GameEditDevelopers"/>

<Label Content="Издатели:"/>

<StackPanel x:Name="GameEditPublishers"/>

<Label Content="Жанры:"/>

<StackPanel x:Name="GameEditGenres"/>

<Label Content="Платформы:"/>

<StackPanel x:Name="GameEditPlatforms"/>

</StackPanel>

</ScrollViewer>

</TabItem>

<TabItem Header="Рецензии">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<StackPanel Width="300">

<StackPanel Orientation="Horizontal">

<Button Click="EditButton\_Click" Content="Редактировать" Margin="0" Padding="5 0"/>

<Label Content="Рецензенты:"/>

</StackPanel>

<ListBox x:Name="Reviewers" Height="250"/>

<Button Content="Добавить" Click="AddReview\_Click" Margin="0 5"/>

</StackPanel>

<ScrollViewer Grid.Column="1">

<StackPanel x:Name="ReviewsList" MaxWidth="1000"/>

</ScrollViewer>

</Grid>

</TabItem>

</TabControl>

<StackPanel Grid.Row="1"

Orientation="Horizontal"

HorizontalAlignment="Right">

<Button Content="Сохранить" Click="SaveButton\_Click"/>

<Button Content="Отмена" Click="CancelButton\_Click"/>

</StackPanel>

</Grid>

</Window>

Б.8 Код EditGameWindow.xaml.cs

В модуле EditGameWindow.xaml.cs содержится описание работы окна редкатирования игр на языке C#:

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.Shapes;

using System.Data.SqlClient;

namespace GameDatabase\_App

{

/// <summary>

/// Логика взаимодействия для EditGameWindow.xaml

/// </summary>

public partial class EditGameWindow : Window

{

public EditGameWindow(int gameId = 0)

{

InitializeComponent();

Tag = gameId;

ShowGame(gameId);

}

// Отмена изменений

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

{

this.Close();

}

// Сохранение изменений

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

{

if (EditGameTitle.Text.Length != 0 && EditGameSummary.Text.Length != 0 && EditGameOfficial.Text.Length != 0)

{

try

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection.Open();

// Добавление/Редактирование игры

using (SqlCommand command = new SqlCommand() { Connection = connection })

{

command.Parameters.Add(new SqlParameter("@title", EditGameTitle.Text));

command.Parameters.Add(new SqlParameter("@release", EditGameRelease.SelectedDate != null ? (object)EditGameRelease.SelectedDate : DBNull.Value));

command.Parameters.Add(new SqlParameter("@website", EditGameOfficial.Text));

command.Parameters.Add(new SqlParameter("@summary", EditGameSummary.Text));

if ((int)Tag > 0)

{

command.Parameters.Add(new SqlParameter("@id", (int)Tag));

command.CommandText = @"UPDATE dbo.Games SET title = @title, release\_date = @release, website = @website, summary = @summary WHERE id = @id";

command.ExecuteNonQuery();

}

else

{

command.CommandText = "StrProc\_AddGame";

command.CommandType = System.Data.CommandType.StoredProcedure;

command.ExecuteNonQuery();

command.CommandType = System.Data.CommandType.Text;

command.CommandText = @"SELECT TOP(1) id FROM dbo.Games ORDER BY id DESC";

using (SqlDataReader dataReader = command.ExecuteReader())

{

dataReader.Read();

Tag = dataReader.GetInt32(0);

}

}

}

// Добавление/Удаление разработчиков игры

using (SqlCommand command = new SqlCommand(@"SELECT dbo.Games\_Developers.game\_id, dbo.Games\_Developers.developer\_id FROM dbo.Games\_Developers WHERE game\_id = @id AND developer\_id = @dev\_id", connection))

{

command.Parameters.Add(new SqlParameter("@id", (int)Tag));

command.Parameters.Add(new SqlParameter("@dev\_id", 0));

foreach (CheckBox item in GameEditDevelopers.Children)

{

command.Parameters[1].Value = item.TabIndex;

using (SqlDataReader dataReader = command.ExecuteReader())

{

if ((bool)item.IsChecked)

{

if (!dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"INSERT INTO dbo.Games\_Developers (game\_id, developer\_id) VALUES (@game\_id, @dev\_id)", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

else

{

if (dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"DELETE FROM dbo.Games\_Developers WHERE game\_id = @game\_id AND developer\_id = @dev\_id", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

}

}

}

// Добавление/Удаление издателей игры

using (SqlCommand command = new SqlCommand(@"SELECT dbo.Games\_Publishers.game\_id, dbo.Games\_Publishers.publisher\_id FROM dbo.Games\_Publishers WHERE game\_id = @id AND publisher\_id = @dev\_id", connection))

{

command.Parameters.Add(new SqlParameter("@id", (int)Tag));

command.Parameters.Add(new SqlParameter("@dev\_id", 0));

foreach (CheckBox item in GameEditPublishers.Children)

{

command.Parameters[1].Value = item.TabIndex;

using (SqlDataReader dataReader = command.ExecuteReader())

{

if ((bool)item.IsChecked)

{

if (!dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"INSERT INTO dbo.Games\_Publishers (game\_id, publisher\_id) VALUES (@game\_id, @dev\_id)", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

else

{

if (dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"DELETE FROM dbo.Games\_Publishers WHERE game\_id = @game\_id AND publisher\_id = @dev\_id", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

}

}

}

// Добавление/Удаление жанров игры

using (SqlCommand command = new SqlCommand(@"SELECT dbo.Games\_Genres.game\_id, dbo.Games\_Genres.genre\_id FROM dbo.Games\_Genres WHERE game\_id = @id AND genre\_id = @dev\_id", connection))

{

command.Parameters.Add(new SqlParameter("@id", (int)Tag));

command.Parameters.Add(new SqlParameter("@dev\_id", 0));

foreach (CheckBox item in GameEditGenres.Children)

{

command.Parameters[1].Value = item.TabIndex;

using (SqlDataReader dataReader = command.ExecuteReader())

{

if ((bool)item.IsChecked)

{

if (!dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"INSERT INTO dbo.Games\_Genres (game\_id, genre\_id) VALUES (@game\_id, @dev\_id)", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

else

{

if (dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"DELETE FROM dbo.Games\_Genres WHERE game\_id = @game\_id AND genre\_id = @dev\_id", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

}

}

}

// Добавление/Удаление платформ игры

using (SqlCommand command = new SqlCommand(@"SELECT dbo.Games\_Platforms.game\_id, dbo.Games\_Platforms.platform\_id FROM dbo.Games\_Platforms WHERE game\_id = @id AND platform\_id = @dev\_id", connection))

{

command.Parameters.Add(new SqlParameter("@id", (int)Tag));

command.Parameters.Add(new SqlParameter("@dev\_id", 0));

foreach (CheckBox item in GameEditPlatforms.Children)

{

command.Parameters[1].Value = item.TabIndex;

using (SqlDataReader dataReader = command.ExecuteReader())

{

if ((bool)item.IsChecked)

{

if (!dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"INSERT INTO dbo.Games\_Platforms (game\_id, platform\_id) VALUES (@game\_id, @dev\_id)", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

else

{

if (dataReader.HasRows)

{

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand insertCommand = new SqlCommand(@"DELETE FROM dbo.Games\_Platforms WHERE game\_id = @game\_id AND platform\_id = @dev\_id", connection1))

{

insertCommand.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

insertCommand.Parameters.Add(new SqlParameter("@dev\_id", item.TabIndex));

insertCommand.ExecuteNonQuery();

}

}

}

}

}

}

}

// Добавление/Изменение/Удаление рецензий на игру

using (SqlCommand command = new SqlCommand(@"SELECT dbo.Reviews.game\_id, dbo.Reviews.reviewer\_id FROM dbo.Reviews WHERE game\_id = @id AND reviewer\_id = @reviewer\_id", connection))

{

command.Parameters.Add(new SqlParameter("@id", (int)Tag));

command.Parameters.Add(new SqlParameter("@reviewer\_id", 0));

using (SqlConnection connection1 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection1.Open();

using (SqlCommand command1 = new SqlCommand(@"SELECT id FROM dbo.Reviewers", connection1))

{

using (SqlDataReader dataReader1 = command1.ExecuteReader())

{

while (dataReader1.Read())

{

command.Parameters[1].Value = dataReader1.GetInt32(0);

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

bool having = false;

foreach (StackPanel item in ReviewsList.Children)

{

if ((int)(item.Tag) == dataReader1.GetInt32(0))

{

using (SqlConnection connection2 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection2.Open();

using (SqlCommand command2 = new SqlCommand(@"UPDATE dbo.Reviews SET score = @score, summary = @summary, web\_address = @url

WHERE dbo.Reviews.game\_id = @game\_id AND dbo.Reviews.reviewer\_id = @reviewer\_id", connection2))

{

command2.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

command2.Parameters.Add(new SqlParameter("@reviewer\_id", dataReader1.GetInt32(0)));

DockPanel dock = (DockPanel)item.Children[2];

Slider slider = (Slider)dock.Children[1];

if (slider.Value > 0)

{

command2.Parameters.Add(new SqlParameter("@score", slider.Value));

}

else

{

command2.Parameters.Add(new SqlParameter("@score", DBNull.Value));

}

command2.Parameters.Add(new SqlParameter("@summary", ((TextBox)item.Children[4]).Text));

command2.Parameters.Add(new SqlParameter("@url", ((TextBox)item.Children[6]).Text));

command2.ExecuteNonQuery();

}

}

having = true;

}

}

if (!having)

{

using (SqlConnection connection2 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection2.Open();

using (SqlCommand command2 = new SqlCommand(@"DELETE FROM dbo.Reviews WHERE dbo.Reviews.game\_id = @game\_id

AND dbo.Reviews.reviewer\_id = @reviewer\_id", connection2))

{

command2.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

command2.Parameters.Add(new SqlParameter("@reviewer\_id", dataReader1.GetInt32(0)));

command2.ExecuteNonQuery();

}

}

}

}

else

{

foreach (StackPanel item in ReviewsList.Children)

{

if ((int)(item.Tag) == dataReader1.GetInt32(0))

{

using (SqlConnection connection2 = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection2.Open();

using (SqlCommand command2 = new SqlCommand(@"INSERT INTO dbo.Reviews (game\_id, reviewer\_id, score, summary, web\_address)

VALUES (@game\_id, @reviewer\_id, @score, @summary, @url)", connection2))

{

command2.Parameters.Add(new SqlParameter("@game\_id", (int)Tag));

command2.Parameters.Add(new SqlParameter("@reviewer\_id", dataReader1.GetInt32(0)));

DockPanel dock = (DockPanel)item.Children[2];

Slider slider = (Slider)dock.Children[1];

if (slider.Value > 0)

{

command2.Parameters.Add(new SqlParameter("@score", slider.Value));

}

else

{

command2.Parameters.Add(new SqlParameter("@score", DBNull.Value));

}

command2.Parameters.Add(new SqlParameter("@summary", ((TextBox)item.Children[4]).Text));

command2.Parameters.Add(new SqlParameter("@url", ((TextBox)item.Children[6]).Text));

command2.ExecuteNonQuery();

}

}

}

}

}

}

}

}

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

this.Close();

}

else

{

MessageBox.Show("Поля заполнены некорректно", "Ошибка сохранения", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

// Добавление рецензента

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

{

if (Reviewers.SelectedItem != null)

{

int reviewer = (int)((ListBoxItem)(Reviewers.SelectedItem)).Tag;

// Панель компоновки рецензии

StackPanel reviewTile = new StackPanel() { Margin = new Thickness(10), Tag = reviewer };

ReviewsList.Children.Add(reviewTile);

DockPanel dock1 = new DockPanel()

{

VerticalAlignment = VerticalAlignment.Center

};

reviewTile.Children.Add(dock1);

// Название издания/рецензента

Label reviewerTitle = new Label()

{

FontSize = 18,

Content = ((ListBoxItem)(Reviewers.SelectedItem)).Content

};

dock1.Children.Add(reviewerTitle);

Button deleteReviewButton = new Button()

{

Tag = reviewer,

HorizontalAlignment = HorizontalAlignment.Right,

Width = 100,

Content = "Удалить"

};

deleteReviewButton.Click += DeleteReview\_Click;

DockPanel.SetDock(deleteReviewButton, Dock.Right);

dock1.Children.Add(deleteReviewButton);

// Оценка рецензии

// --------------------------------------------------------

Label scoreLabel = new Label()

{

Content = "Оценка:"

};

reviewTile.Children.Add(scoreLabel);

DockPanel dock = new DockPanel()

{

VerticalAlignment = VerticalAlignment.Center

};

reviewTile.Children.Add(dock);

TextBox scoreTextBox = new TextBox()

{

Width = 40,

VerticalContentAlignment = VerticalAlignment.Center,

IsEnabled = false

};

DockPanel.SetDock(scoreTextBox, Dock.Right);

dock.Children.Add(scoreTextBox);

Slider scoreSlider = new Slider()

{

Minimum = 0,

Maximum = 100,

TickFrequency = 1,

IsSnapToTickEnabled = true,

TickPlacement = System.Windows.Controls.Primitives.TickPlacement.BottomRight

};

dock.Children.Add(scoreSlider);

Binding binding = new Binding

{

Source = scoreSlider,

Path = new PropertyPath("Value"),

Mode = BindingMode.TwoWay

};

scoreTextBox.SetBinding(TextBox.TextProperty, binding);

// --------------------------------------------------------

// Краткое описание

// --------------------------------------------------------

Label summaryLabel = new Label() { Content = "Краткое содержание:" };

reviewTile.Children.Add(summaryLabel);

TextBox reviewSummary = new TextBox()

{

Height = 150,

TextWrapping = TextWrapping.Wrap

};

reviewTile.Children.Add(reviewSummary);

// --------------------------------------------------------

// Сайт

// --------------------------------------------------------

Label siteLabel = new Label() { Content = "Ссылка на полный текст статьи:" };

reviewTile.Children.Add(siteLabel);

TextBox reviewSite = new TextBox()

{

};

reviewTile.Children.Add(reviewSite);

foreach (ListBoxItem listBoxItem in Reviewers.Items)

{

if ((int)listBoxItem.Tag == reviewer)

{

Reviewers.Items.Remove(listBoxItem);

break;

}

}

}

}

// Удаление рецензии

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

{

foreach (StackPanel item in ReviewsList.Children)

{

if ((int)item.Tag == (int)((Button)sender).Tag)

{

ReviewsList.Children.Remove(item);

Label label = (Label)((DockPanel)item.Children[0]).Children[0];

ListBoxItem reviewer = new ListBoxItem()

{

Tag = item.Tag,

Content = label.Content

};

Reviewers.Items.Add(reviewer);

break;

}

}

}

// Открытие окна редактирования рецензентов

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

{

if (MessageBox.Show("Внесенные вами изменения не сохраняться. Вы действительно хотите перейти в окно редактирования рецензентов?",

"Предупреждение",

MessageBoxButton.YesNo,

MessageBoxImage.Warning) == MessageBoxResult.Yes)

{

EditWindow window = new EditWindow(4);

window.ShowDialog();

UpdateReviews((int)Tag);

}

}

// Загрузка списков, а также информации о игре при редактировании

private void ShowGame(int id = 0)

{

try

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection.Open();

using (SqlCommand command = new SqlCommand() { Connection = connection })

{

// Отображение списка разработчиков

command.CommandText = @"SELECT id, title FROM dbo.Developers";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

TabIndex = dataReader.GetInt32(0),

Content = dataReader.GetString(1)

};

GameEditDevelopers.Children.Add(item);

}

}

}

// Отображение списка издателей

command.CommandText = @"SELECT id, title FROM dbo.Publishers";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

TabIndex = dataReader.GetInt32(0),

Content = dataReader.GetString(1)

};

GameEditPublishers.Children.Add(item);

}

}

}

// Отображение списка жанров

command.CommandText = @"SELECT id, title FROM dbo.Genres";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

TabIndex = dataReader.GetInt32(0),

Content = dataReader.GetString(1)

};

GameEditGenres.Children.Add(item);

}

}

}

// Отображение списка платформ

command.CommandText = @"SELECT id, title FROM dbo.Platforms";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

CheckBox item = new CheckBox()

{

TabIndex = dataReader.GetInt32(0),

Content = dataReader.GetString(1)

};

GameEditPlatforms.Children.Add(item);

}

}

}

if (id > 0)

{

command.Parameters.Add(new SqlParameter("@id", id));

// Получение информации о игре

command.CommandText = @"SELECT title, website, release\_date, summary FROM dbo.Games WHERE id = @id";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if(dataReader.HasRows)

{

while (dataReader.Read())

{

EditGameTitle.Text = dataReader.GetString(0);

EditGameOfficial.Text = dataReader.GetString(1);

if (!dataReader.IsDBNull(2))

{

EditGameRelease.SelectedDate = dataReader.GetDateTime(2);

}

else

{

EditGameRelease.SelectedDate = null;

}

EditGameSummary.Text = dataReader.GetString(3);

}

}

}

// Отображение разработчиков игры

command.CommandText = @"SELECT developer\_id FROM dbo.Games\_Developers WHERE game\_id = @id";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

foreach (CheckBox item in GameEditDevelopers.Children)

{

if (item.TabIndex == dataReader.GetInt32(0))

{

item.IsChecked = true;

}

}

}

}

}

// Отображение издателей игры

command.CommandText = @"SELECT publisher\_id FROM dbo.Games\_Publishers WHERE game\_id = @id";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

foreach (CheckBox item in GameEditPublishers.Children)

{

if (item.TabIndex == dataReader.GetInt32(0))

{

item.IsChecked = true;

}

}

}

}

}

// Отображение жанров игры

command.CommandText = @"SELECT genre\_id FROM dbo.Games\_Genres WHERE game\_id = @id";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

foreach (CheckBox item in GameEditGenres.Children)

{

if (item.TabIndex == dataReader.GetInt32(0))

{

item.IsChecked = true;

}

}

}

}

}

// Отображение платформ игры

command.CommandText = @"SELECT platform\_id FROM dbo.Games\_Platforms WHERE game\_id = @id";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

foreach (CheckBox item in GameEditPlatforms.Children)

{

if (item.TabIndex == dataReader.GetInt32(0))

{

item.IsChecked = true;

}

}

}

}

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

UpdateReviews(id);

}

// Обновление вкладки рецензии

private void UpdateReviews(int id = 0)

{

try

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection.Open();

using (SqlCommand command = new SqlCommand() { Connection = connection })

{

// Отображение списка рецензентов

Reviewers.Items.Clear();

command.CommandText = @"SELECT id, title FROM dbo.Reviewers";

using (SqlDataReader dataReader = command.ExecuteReader())

{

while (dataReader.Read())

{

ListBoxItem reviewer = new ListBoxItem()

{

Tag = dataReader.GetInt32(0),

Content = dataReader.GetString(1)

};

Reviewers.Items.Add(reviewer);

}

}

if (id > 0)

{

ReviewsList.Children.Clear();

// Отображение рецензий игры

command.Parameters.Add(new SqlParameter("@id", id));

command.CommandText = @"SELECT dbo.Reviewers.id, dbo.Reviewers.title, dbo.Reviews.summary, dbo.Reviews.score, dbo.Reviews.web\_address

FROM dbo.Reviews INNER JOIN dbo.Reviewers ON dbo.Reviews.reviewer\_id = dbo.Reviewers.id WHERE game\_id = @id";

using (SqlDataReader dataReader = command.ExecuteReader())

{

if (dataReader.HasRows)

{

while (dataReader.Read())

{

foreach (ListBoxItem listBoxItem in Reviewers.Items)

{

if ((int)listBoxItem.Tag == dataReader.GetInt32(0))

{

Reviewers.Items.Remove(listBoxItem);

break;

}

}

// Панель компоновки рецензии

StackPanel reviewTile = new StackPanel() { Margin = new Thickness(10), Tag = dataReader.GetInt32(0) };

ReviewsList.Children.Add(reviewTile);

DockPanel dock1 = new DockPanel()

{

VerticalAlignment = VerticalAlignment.Center

};

reviewTile.Children.Add(dock1);

// Название издания/рецензента

Label reviewerTitle = new Label()

{

FontSize = 18,

Content = dataReader.GetString(1)

};

dock1.Children.Add(reviewerTitle);

Button deleteReviewButton = new Button()

{

Tag = dataReader.GetInt32(0),

HorizontalAlignment = HorizontalAlignment.Right,

Width = 100,

Content = "Удалить"

};

deleteReviewButton.Click += DeleteReview\_Click;

DockPanel.SetDock(deleteReviewButton, Dock.Right);

dock1.Children.Add(deleteReviewButton);

// Оценка рецензии

// --------------------------------------------------------

Label scoreLabel = new Label()

{

Content = "Оценка:"

};

reviewTile.Children.Add(scoreLabel);

DockPanel dock = new DockPanel()

{

VerticalAlignment = VerticalAlignment.Center

};

reviewTile.Children.Add(dock);

TextBox scoreTextBox = new TextBox()

{

Width = 40,

VerticalContentAlignment = VerticalAlignment.Center,

IsEnabled = false

};

DockPanel.SetDock(scoreTextBox, Dock.Right);

dock.Children.Add(scoreTextBox);

Slider scoreSlider = new Slider()

{

Value = dataReader.IsDBNull(3) ? 0 : dataReader.GetByte(3),

Minimum = 0,

Maximum = 100,

TickFrequency = 1,

IsSnapToTickEnabled = true,

TickPlacement = System.Windows.Controls.Primitives.TickPlacement.BottomRight

};

dock.Children.Add(scoreSlider);

Binding binding = new Binding

{

Source = scoreSlider,

Path = new PropertyPath("Value"),

Mode = BindingMode.TwoWay

};

scoreTextBox.SetBinding(TextBox.TextProperty, binding);

// --------------------------------------------------------

// Краткое описание

// --------------------------------------------------------

Label summaryLabel = new Label() { Content = "Краткое содержание:" };

reviewTile.Children.Add(summaryLabel);

TextBox reviewSummary = new TextBox()

{

Height = 150,

Text = dataReader.GetString(2),

TextWrapping = TextWrapping.Wrap

};

reviewTile.Children.Add(reviewSummary);

// --------------------------------------------------------

// Сайт

// --------------------------------------------------------

Label siteLabel = new Label() { Content = "Ссылка на полный текст статьи:" };

reviewTile.Children.Add(siteLabel);

TextBox reviewSite = new TextBox()

{

Text = dataReader.GetString(4)

};

reviewTile.Children.Add(reviewSite);

// --------------------------------------------------------

}

}

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

}

В модуле EditWindow.xaml содержится декларативное описание интерфейса окна редактирования объектов на языке XAML:

<Window x:Class="GameDatabase\_App.EditWindow"

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:GameDatabase\_App"

mc:Ignorable="d"

Title="Редактирование" MinHeight="600" MinWidth="800">

<StackPanel MaxWidth="1000">

<Label Content="Удаление:"/>

<ScrollViewer>

<ListBox x:Name="Items" Height="250" SelectionMode="Extended"/>

</ScrollViewer>

<Button Content="Удалить" Click="DeleteButton\_Click"/>

<Label Content="Добавление:"/>

<TextBox x:Name="AddItem"/>

<Button Content="Добавить" Click="AddButton\_Click"/>

</StackPanel>

</Window>

В модуле EditWindow.xaml.cs содержится описание работы окна редактирования объектов на языке C#:

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.Shapes;

using System.Data.SqlClient;

namespace GameDatabase\_App

{

/// <summary>

/// Логика взаимодействия для EditWindow.xaml

/// </summary>

public partial class EditWindow : Window

{

public EditWindow(int Type)

{

InitializeComponent();

Tag = Type;

Show(Type);

}

private void Show(int type)

{

Items.Items.Clear();

try

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection.Open();

using (SqlCommand command = new SqlCommand() { Connection = connection})

{

switch (type)

{

case 0:

command.CommandText = @"SELECT id, title FROM dbo.Developers";

break;

case 1:

command.CommandText = @"SELECT id, title FROM dbo.Publishers";

break;

case 2:

command.CommandText = @"SELECT id, title FROM dbo.Genres";

break;

case 3:

command.CommandText = @"SELECT id, title FROM dbo.Platforms";

break;

case 4:

command.CommandText = @"SELECT id, title FROM dbo.Reviewers";

break;

}

using (SqlDataReader dataReader = command.ExecuteReader())

{

while (dataReader.Read())

{

ListBoxItem item = new ListBoxItem()

{

Tag = dataReader.GetInt32(0),

Content = dataReader.GetString(1)

};

Items.Items.Add(item);

}

}

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

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

{

if(Items.SelectedItem != null)

{

foreach (ListBoxItem item in Items.SelectedItems)

{

try

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection.Open();

using (SqlCommand command = new SqlCommand() { Connection = connection })

{

command.Parameters.Add(new SqlParameter("@id", (int)(item.Tag)));

switch ((int)Tag)

{

case 0:

command.CommandText = @"DELETE FROM dbo.Developers WHERE id = @id";

break;

case 1:

command.CommandText = @"DELETE FROM dbo.Publishers WHERE id = @id";

break;

case 2:

command.CommandText = @"DELETE FROM dbo.Genres WHERE id = @id";

break;

case 3:

command.CommandText = @"DELETE FROM dbo.Platforms WHERE id = @id";

break;

case 4:

command.CommandText = @"DELETE FROM dbo.Reviewers WHERE id = @id";

break;

}

command.ExecuteNonQuery();

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

Show((int)Tag);

}

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

{

if (AddItem.Text != "")

{

try

{

using (SqlConnection connection = new SqlConnection(Properties.Settings.Default.userConnection))

{

connection.Open();

using (SqlCommand command = new SqlCommand() { Connection = connection })

{

command.Parameters.Add(new SqlParameter("@title", AddItem.Text));

switch ((int)Tag)

{

case 0:

command.CommandText = @"INSERT INTO dbo.Developers (title) VALUES (@title)";

break;

case 1:

command.CommandText = @"INSERT INTO dbo.Publishers (title) VALUES (@title)";

break;

case 2:

command.CommandText = @"INSERT INTO dbo.Genres (title) VALUES (@title)";

break;

case 3:

command.CommandText = @"INSERT INTO dbo.Platforms (title) VALUES (@title)";

break;

case 4:

command.CommandText = @"INSERT INTO dbo.Reviewers (title) VALUES (@title)";

break;

}

command.ExecuteNonQuery();

}

}

}

catch (SqlException ex)

{

MessageBox.Show($"В процессе обработки данных произошла ошибка:\n{ex}", "Ошибка обработки данных", MessageBoxButton.OK, MessageBoxImage.Error);

}

Show((int)Tag);

}

}

}

}