МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-98 01 03 «Программное обеспечение информационной безопасности мобильных систем»

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**КУРСОВОГО ПРОЕКТА:**

по дисциплине «Современные технологии программирования мобильных систем»

Тема Программное средство «Агрегатор отзывов о видеоиграх»

Исполнитель

Студент 2 курса группы 7 Заянковский Д. В.

(Ф.И.О.)

Руководитель работы ассистент Панченко О. Л.

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

Курсовой проект защищен с оценкой

Минск 2022

**Содержание**

[ВВЕДЕНИЕ 3](#_Toc104758041)

[1. Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования 4](#_Toc104758042)

[1.1 Обзор и анализ аналогов 4](#_Toc104758043)

[2. Анализ и проектирование архитектуры приложения 8](#_Toc104758044)

[2.1. Анализ требований к программному средству 8](#_Toc104758045)

[2.2 Разработка функциональных требований 9](#_Toc104758046)

[3. Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса 9](#_Toc104758047)

[3.1. Общая структура проекта 9](#_Toc104758048)

[3.2. Проектирование базы данных 12](#_Toc104758049)

[3.3. Проектирование и реализация классов приложения 13](#_Toc104758050)

[3.4. Диаграмма последовательности 13](#_Toc104758051)

[4. Реализация программного средства 14](#_Toc104758052)

[4.1. Основные классы программного средства 14](#_Toc104758053)

[4.2. Описание классов и методов программного средства 17](#_Toc104758054)

[4.2.1. Управление учетной записью пользователя 17](#_Toc104758055)

[4.2.2. Просмотр игр 17](#_Toc104758056)

[4.2.3. Просмотр пользователей 17](#_Toc104758057)

[4.2.4. Просмотр отзывов 18](#_Toc104758058)

[4.2.5. Добавление отзыва 18](#_Toc104758059)

[4.2.6. Добавление/удаление игр 18](#_Toc104758060)

[4.3. Реализация паттерна Singleton 18](#_Toc104758061)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 19](#_Toc104758062)

[5.1. Тестирование авторизации и регистрации 19](#_Toc104758063)

[5.2. Тестирование редактирования учетной записи 22](#_Toc104758064)

[5.3. Тестирование оставления отзыва 24](#_Toc104758065)

[5.4. Тестирование добавления игры 24](#_Toc104758066)

[6. Руководство по использованию 26](#_Toc104758067)

[Заключение 31](#_Toc104758068)

[Список литературы 32](#_Toc104758069)

[ПРИЛОЖЕНИЕ А 33](#_Toc104758070)

[ПРИЛОЖЕНИЕ Б 34](#_Toc104758071)

[ПРИЛОЖЕНИЕ В 35](#_Toc104758072)

[ПРИЛОЖЕНИЕ Г 36](#_Toc104758073)

[ПРИЛОЖЕНИЕ Д 37](#_Toc104758074)

[ПРИЛОЖЕНИЕ Е 38](#_Toc104758075)

# ВВЕДЕНИЕ

Современные технологии позволяют решать различные задачи множеством способов. Приложения проникают в повседневную жизнь и делают её проще и лучше, решая задачи намного быстрее, позволяя пользователю экономить своё время.

У людей в зависимости от их привычек и предпочтений присутствует множество способов проведения досуга. Некоторые предпочитают занятие спортом (различные спортивные игры), некоторые проводят время за просмотром любимых телепередач, фильмов или сериалов. Некоторые предпочитают провести свое свободное время за видеоиграми. На сегодняшний день существует множество различных игр разных жанров, и становиться всё тяжелее найти ту игру, которая принесет удовольствие.

Решить задачу по поиску игры позволяют рецензии или отзывы других пользователей. Отзывы реальных игроков намного лучше раскрывают все плюсы и минусы той или иной видеоигры. Для того, чтобы облегчить поиск таких отзывов, следует систематизировать все игры и собрать все отзывы в одном месте для удобства пользователя.

Программное обеспечение «Агрегатор отзывов» позволить решить поставленные задачи, а также сэкономить средства на покупку видеоигр, которые пользователю могут и не понравиться вовсе. С помощью приложения появиться возможность искать видеоигры с лучшим пользовательским рейтингом, а также сортировать игры по этому рейтингу, находя для себя лучшие продукты. Такой программный продукт может ответить на вопрос «Какой видеоигрой занять свободное время?».

Также если пользователь играл в некоторые видеоигры, то с помощью такого приложения у него есть возможность поделиться своими впечатлениями, тем самым облегчить поиск другим.

# 1. Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования

## 1.1 Обзор и анализ аналогов

На данном этапе рассмотрим несколько примеров интерфейсов на основе существующих аналогов.

В текущих реалиях сервис по сбору отзывов существует как часть какой-либо системы. Будут рассмотрены два варианта интернет-ресурса, позволяющие собирать отзывы о видеоиграх.

Первый пример – Metacritic (<https://www.metacritic.com/game>) (рис. 1.1).

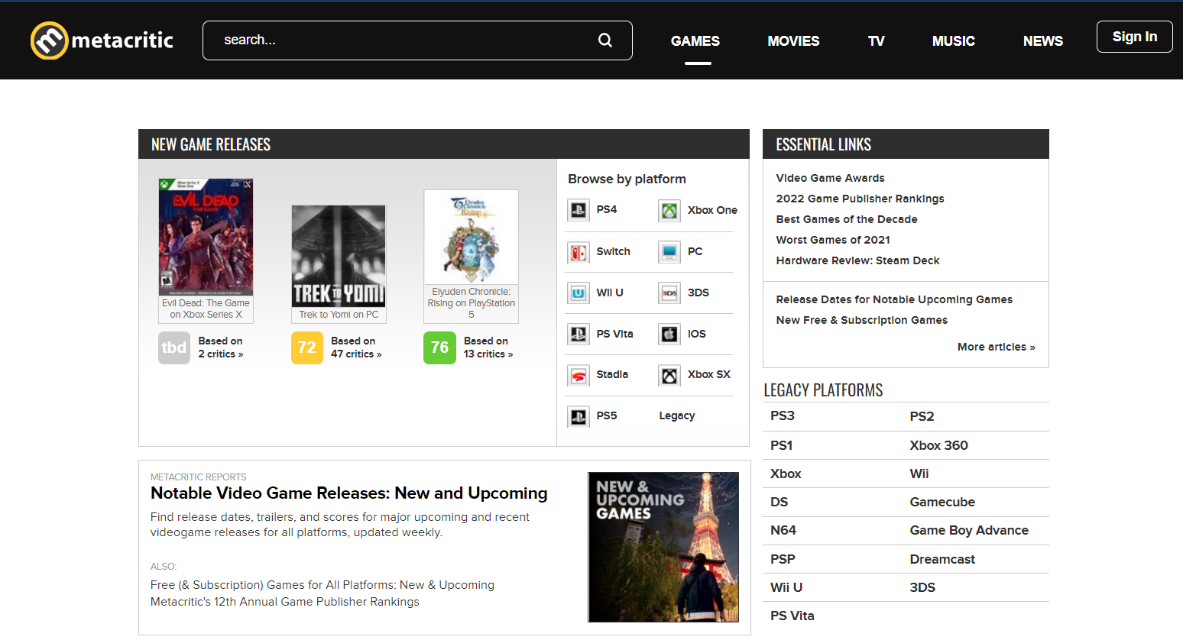


Рисунок 1.1 –Веб-сайт Metacritic.com

На главной странице представлены недавно вышедшие игры, поиск по платформе, на которой игра была выпущена, а также актуальная статья.

Список всех игр доступен с фильтрацией по жанрам (рис 1.2).

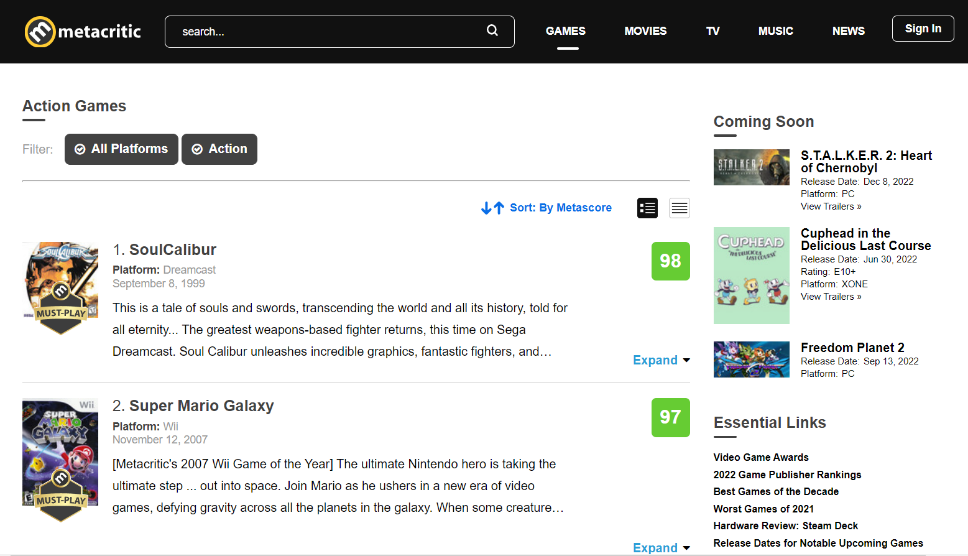


Рисунок 1.2 –Список игр Metacritic

Каждая страница с игрой включает в себя информацию о разработчике, издателе. Присутствует краткое описание игры. Ниже расположены отзывы критиков и пользователей. Каждый отзыв пользователя включает в себя оценку, которую пользователь выбрал для игры, сам текст отзыва, дату публикации (рис. 1.3).



Рисунок 1.3 –Страница игры Metacritic

Интерфейс Metacritic.com позволяет нам получать информацию о видеоигре, а также написать свой отзыв.

По итогу можно сказать, что данный ресурс является хорошим программным средством для сбора отзывов о видеоиграх с хорошим дизайном.

Второй пример – раздел игр сайта Kanobu.ru (рис. 1.4)

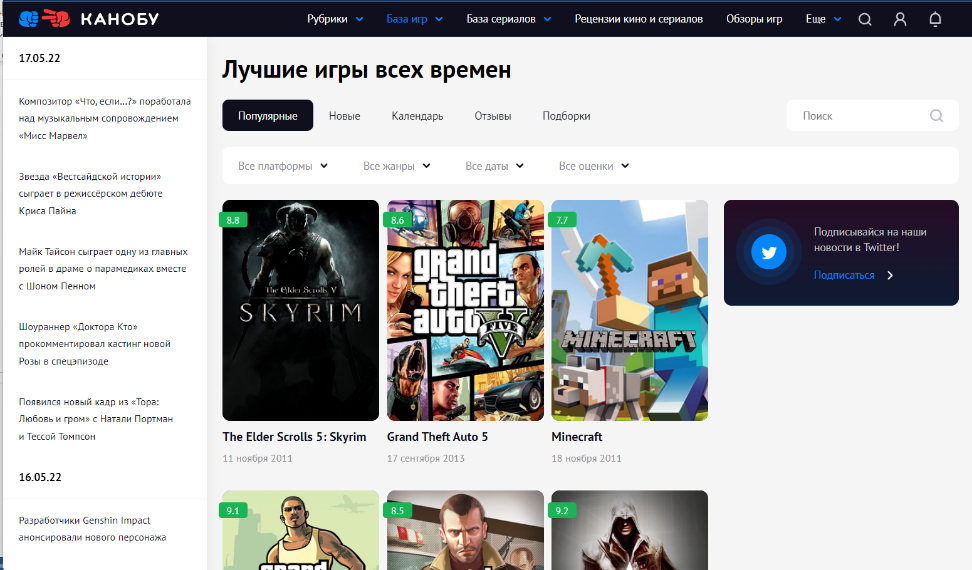


Рисунок 1.4 –Сайт Kanobu.ru

Интерфейс данного ресурса предоставляет нам список всех игр, каждый элемент которого включает в себя изображение, название, а также дату выхода конкретной игры. Для удобства пользователя доступен поиск и фильтрация.

Каждая страница с игрой содержит в себе подробную информацию о видеоигре, содержит минимальные и рекомендуемые системные требования, изображения игры (рис. 1.5).

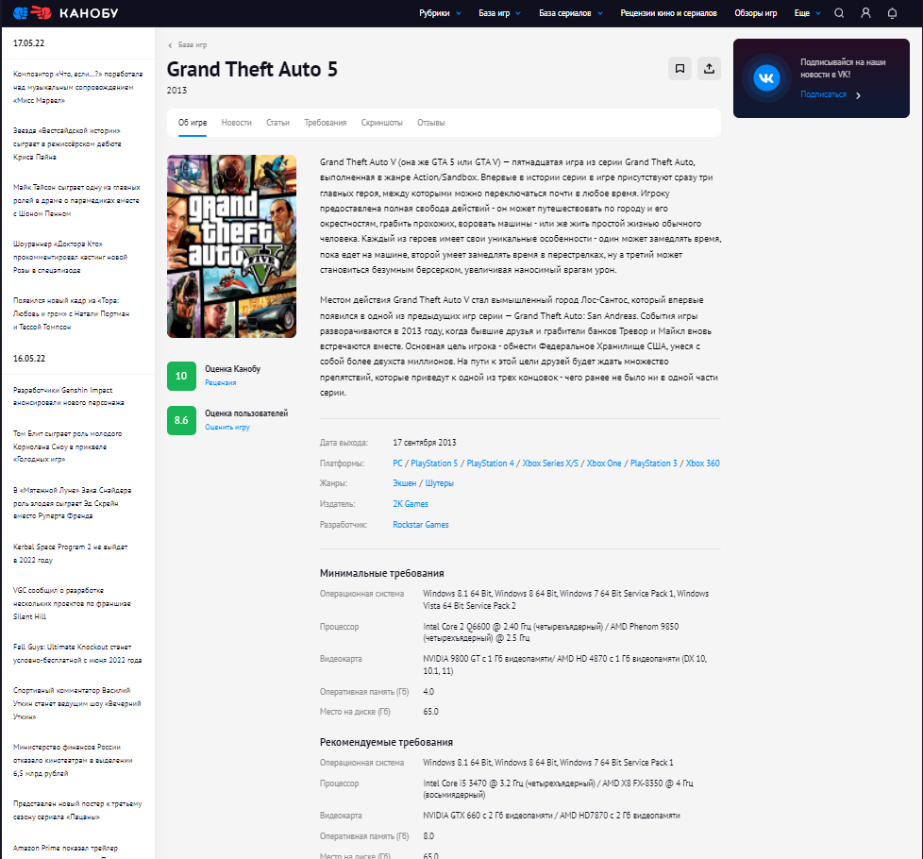


Рисунок 1.5 –Страница игры Kanobu.ru

После блока с информацией расположены отзывы пользователей. Для того, чтобы свой отзыв, ресурс предлагает создать или войти в учетную запись (рис. 1.6).

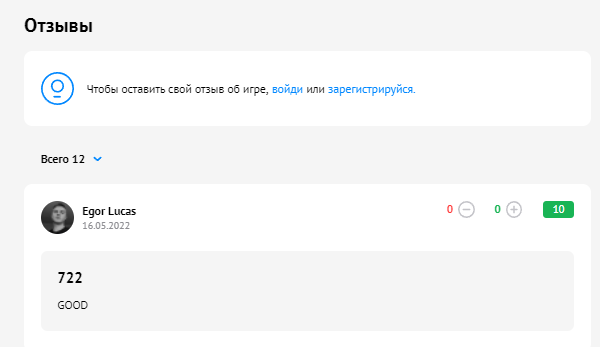


Рисунок 1.6 –Отзывы Kanobu.ru

В целом, данное программное средство обладает всем необходимым функционалом. Присутствует возможность оставить отзыв после регистрации или авторизации, доступна информация про конкретную игру. Также у пользователя есть возможность воспользоваться поиском. Вероятно, избыточна информация об игре, если пользователя интересуют лишь отзывы реальных людей.

# 2. Анализ и проектирование архитектуры приложения

## 2.1. Анализ требований к программному средству

На предыдущем этапе курсового проекта были рассмотрены различные аналоги программных средств для сбора отзывов о видеоиграх. Функционал разрабатываемого приложения будет включать в себя основные возможности рассмотренных аналогов с возможной доработкой или изменением отдельных элементов. Для получения качественного программного продукта необходимо учитывать проблемы и излишки рассмотренных аналогов. Также приложению стоит добавить некоторые дополнительные возможности.

Благодаря анализу похожих по функционалу приложений, было принято решение проектировки функционала для трех категорий пользователей: администраторы и авторизированные пользователи, гости. Представление общего функционала для двух групп можно увидеть на Use Case диаграмме (рис. 2.1).

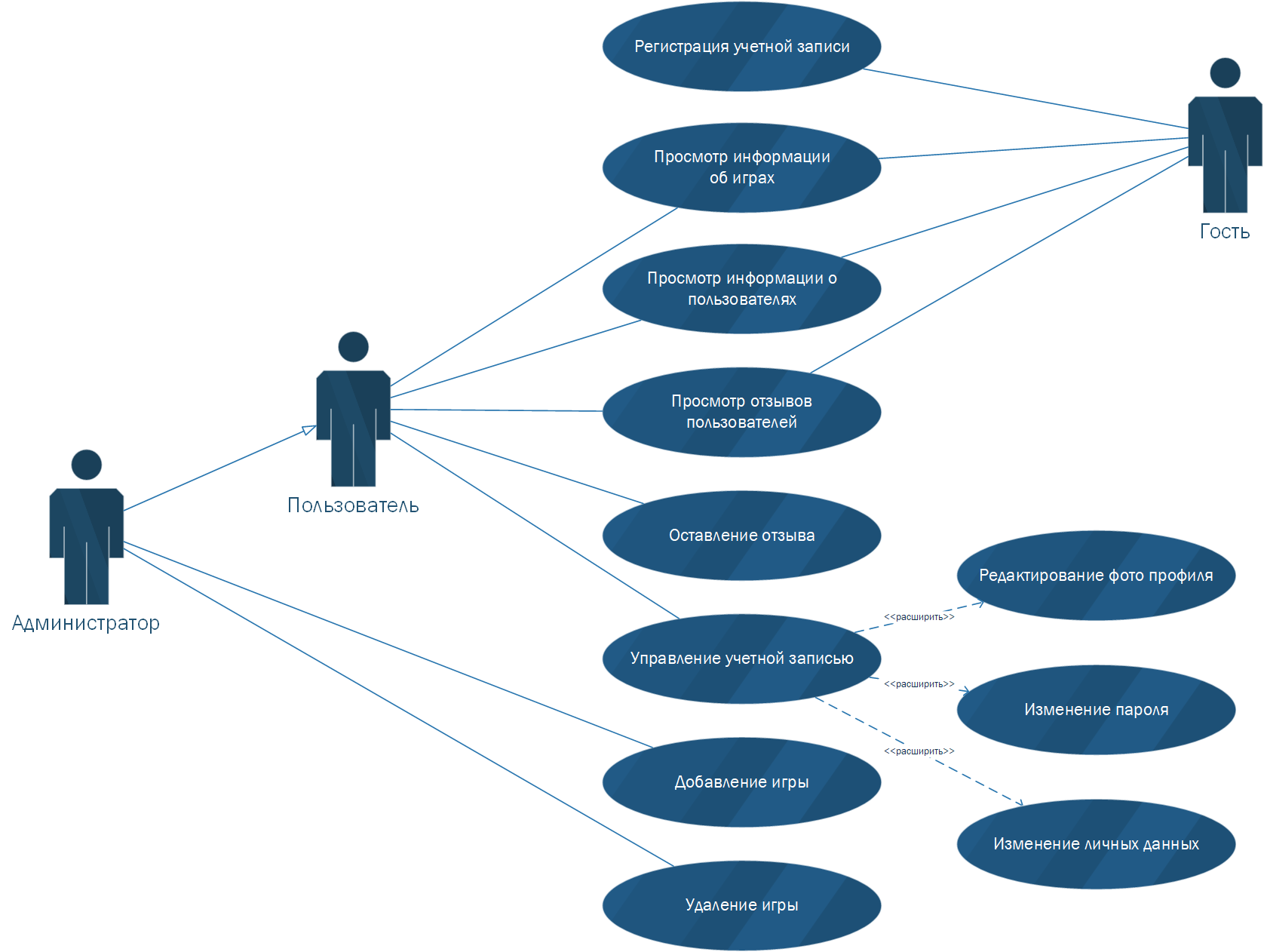


Рисунок 2.1 –Use Case диаграмма

## 2.2 Разработка функциональных требований

Составим список функций, которые сможет выполнить пользователь в зависимости от его роли.

**Пользователь.** Это основной пользователь приложения. Пользователю предоставляется возможность просматривать информацию о пользователях, играх. Также присутствует возможность просмотра отзывов других пользователей, оставление собственного отзыва и управление учетной записью.

**Администратор.** Пользователь с максимально возможным функционалом. Администратору доступны все возможности Пользователя, а также добавление и удаление игр.

**Гость.** Пользователь с минимально возможным функционалом. Гостю предоставляется возможность просмотра информации о пользователях и играх. Присутствует возможность просмотра отзывов и регистрации учетной записи.

# 3. Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса

## 3.1. Общая структура проекта

В данной главе будут описаны все шаги проектирования и разработки приложения, в результате которых получилась определенная структура программного продукта (рис. 3.1).

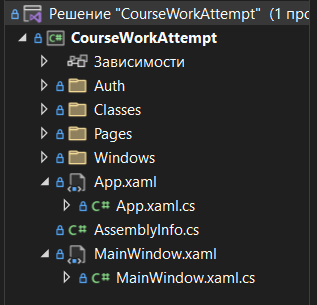


Рисунок 3.1 –Структура проекта

В проекте присутствуют различные виды классов и ресурсов. Рассмотрим их подробнее:

* файл MainWindow.xaml является основным классом, реализующим главное окно WPF приложения, содержит xaml разметку. Аналоги данных файлов с расширением xaml.cs содержат всю необходимую логику взаимодействия, описанную на языке программирования C#. MainWindow – окно, отвечающее за навигацию по всему приложению, путем переключения между страницами. Стартовая страница по умолчанию – MainPage.xaml(страница авторизации);
* файл App.xaml отвечает за подключение словарей ресурсов ко всему приложению. В одном из словарей ресурсов, а именно Styles.xaml, определены стили для основных элементов;
* папка Auth содержит в себе два файла, реализующие соответствующие классы Authorization.cs и Crypto.cs. Класс Authorization отвечает за регистрацию и авторизацию пользователя. Класс Crypto используется для хэширования паролей с помощью алгоритма SHA-256 (рис 3.2);

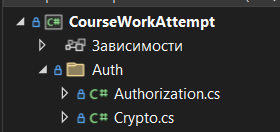


Рисунок 3.2 –Папка Auth

* папка Classes содержит в себе все классы приложения, над объектами которых производятся различные операции. Класс Developer определяет объект разработчика игры. Класс Publisher определяет объект издателя игры. Класс User определяет объекты пользователей приложения. Класс Game определяет объекты игр, на которые оставляются отзывы. Класс Review определяет отзыв на видеоигру (рис 3.3);

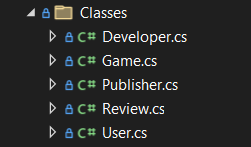


Рисунок 3.3 –Папка Classes

* папка Pages хранит в себе все классы WPF приложения, реализующие страницы программы с помощью xaml разметки. CurrentGame.xaml – это страница, на которую пользователь переходить при выборе игры из списка. CurrentUser.xaml ­­­– это страница, на которой отображается информация о выбранном пользователе. Game.xaml – это страница со всеми играми. MainPage.xaml – стартовая страница приложения. Profile.xaml – страница, на которой отображается информация о текущем(авторизированном) пользователе. Users.xaml – страница, на которой отображаются все пользователи приложения. (рис 3.4);

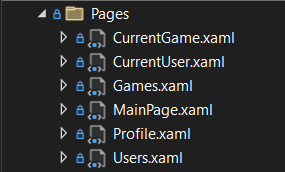


Рисунок 3.4 –Папка Pages

* папка Windows содержит в себе все классы WPF приложения, реализующие окна программы с помощью xaml разметки. AddGameWindow.xaml – страница добавления новой игры(функция администратора). AddReviewWindow – страница добавления отзыва. EditProfileWindow.xaml – окно редактирования учетной записи. RegisterWindow – окно регистрации нового пользователя. AgregatoIcon.png – иконка всех окон приложения. Styles.xaml – словарь ресурсов, в котором определены стили для основных элементов программы (рис 3.5).

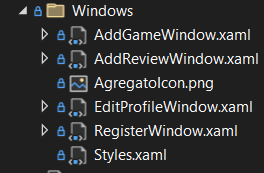


Рисунок 3.5 –Папка Windows

Разработка приложения выполнялась в следующей последовательности:

* проектирование и реализация базы данных;
* подключение базы данных к проекту;
* генерация модели данных;
* проектирование и реализация xaml разметки окон приложения;
* создание дополнительных классов для обеспечения работы функционала;
* улучшение внешнего вида программы.

Все этапы будут подробно рассмотрены в главе 4.

## 3.2. Проектирование базы данных

Было принято решение в данном проекте использовать базу данных MSSQL Server. Разработка приложения начинается с проектирования и реализации базы данных. В процессе разработки структура базы данных менялась для того, чтобы была возможность реализовать поставленный функционал программного продукта. Итоговым решением стала база данных, логическая схема которой представлена на рис. 3.6.

База данных проекта состоит из 7 таблиц, связанных между собой.

USERS. Данная таблица содержит всю информацию о пользователях, включая имя пользователя, пароль и права доступа.

REVIEWS. Данная таблица содержит информацию о отзывах, оставленных всеми пользователями приложения.

GAMES. Таблица содержит информацию о играх, на которые пользователи могут оставлять отзывы.

DEVELOPERS. Таблица содержит информацию о разработчиках игр.

PUBLISHERS. Таблица содержит информацию об издателях игр.

GENRES. Таблица хранит всевозможные жанры игр, которые могут содержаться в таблице GAMES.

PLATFORMS. Таблица хранит всевозможные платформы, на которых может быть запущена игра.

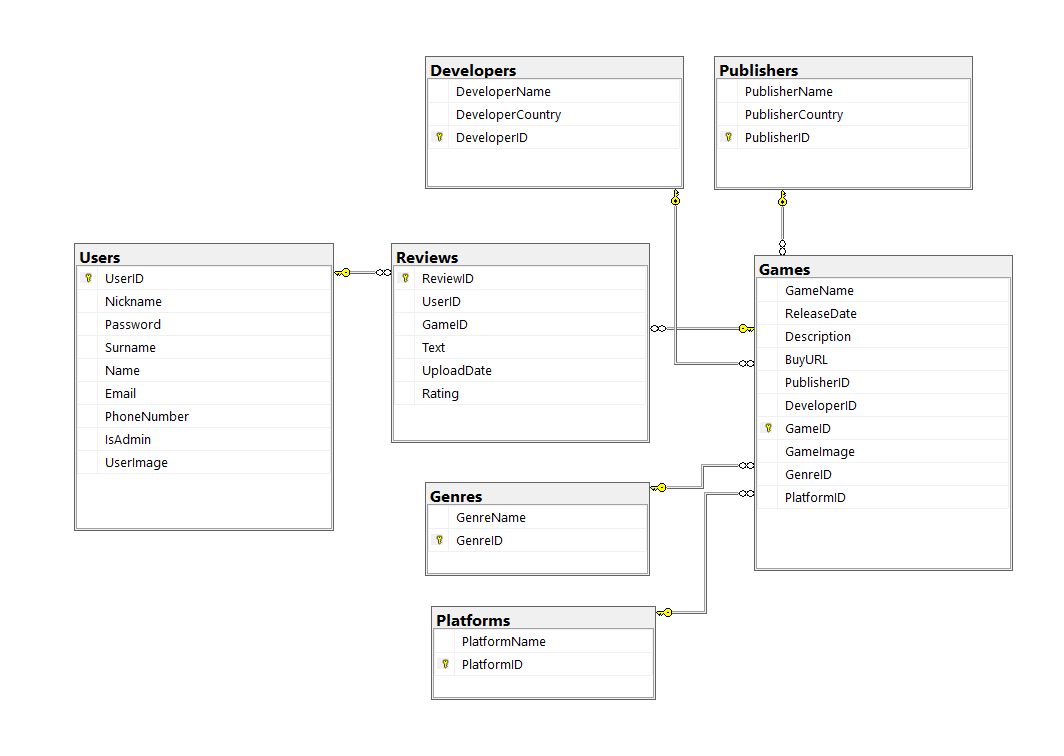


Рисунок 3.6 –Логическая схема базы данных

## 3.3. Проектирование и реализация классов приложения

Все классы приложения были выделены в свои отдельные группы.

Первая группа – классы, которые формируют окна приложения. В них задается вся логика взаимодействия соответствующих им xaml-разметок и пользователя. В них описываются обработчики событий для элементов управления, задаются контексты и источники данных. Диаграмма классов окон представлена в приложении А. Диаграмма классов страниц представлена в приложении Б.

Вторая категория – классы, с объектами которых мы взаимодействуем в процессе выполнения приложения. Диаграмма классов данной группы представлена в приложении В.

Третья категория – вспомогательные классы, которые необходимы для реализации дополнительного функционала. В данном проекте вспомогательные классы используются для авторизации пользователей и хэширования паролей.

Общая диаграмма классов представлена в приложении Г.

## 3.4. Диаграмма последовательности

Для авторизации пользователь вводит логин и пароль. Затем нажимается кнопка «Войти». Событие данной кнопки вызывает метод, отвечающий за авторизацию. Этот метод в свою очередь проверяет наличие данного пользователя в базе данных. Если, такой пользователь существует, то его данные сохраняются внутри статического поля класса Authorization. Затем пользователю предоставляется доступ к его профилю, появляется возможность оставить отзыв и удалить игру (если пользователь является администратором)

Диаграмма последовательности входа в приложение представлена на рис. 3.7.

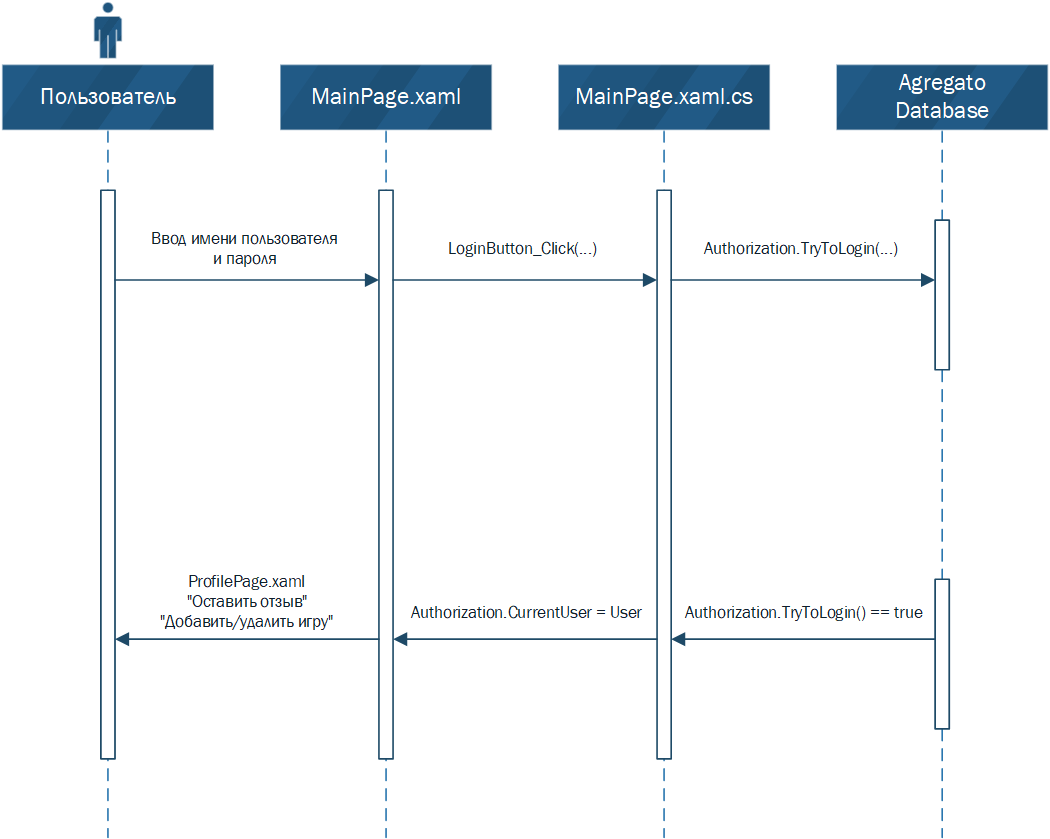


Рисунок 3.7 –Диаграмма последовательности входа в приложение

# 4. Реализация программного средства

## 4.1. Основные классы программного средства

Данное программное средство реализовано с помощью технологии Windows Presentation Foundation (WPF). Это система построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. Выбранная технология позволяет реализовать все поставленные задачи.

Было принято решение реализовать страничную систему приложения, так как именно она позволяет обеспечить наиболее удобную навигацию.

Стили отдельных элементов управления располагаются в словаре ресурсов Styles.xaml.

В главном окне приложения MainWindow.xaml используется элемент Frame, с помощью которого реализуется страничная система приложения. Frame позволяет переключаться с одной страницы на другую с помощью NavigationService (xaml-код окна MainWindow.xaml представлен на листинге 4.1).

<Window FontFamily="Verdana" x:Class="CourseWorkAttempt.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:CourseWorkAttempt"

mc:Ignorable="d"

WindowStartupLocation="CenterScreen"

Title="Agregato" Height="600" Width="1024" MinHeight="600" MinWidth="1024" Icon="/Windows/AgregatoIcon.png">

<Grid>

<Grid.RowDefinitions>

<RowDefinition Height="60px" MaxHeight="60px"></RowDefinition>

<RowDefinition Height="\*"></RowDefinition>

</Grid.RowDefinitions>

<Rectangle HorizontalAlignment="Center" VerticalAlignment="Bottom" Width="2000" Height="100" Grid.Row="0">

<Rectangle.Stroke>

<LinearGradientBrush EndPoint="1,0" StartPoint="0,0">

<GradientStop Color="#FFFC00FF" Offset="0.7"/>

<GradientStop Color="#FF1467FF" Offset="0.5"/>

<GradientStop Color="#FFFC00FF" Offset="0.3"/>

</LinearGradientBrush>

</Rectangle.Stroke>

</Rectangle>

<Grid Grid.Row="0">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="5\*"></ColumnDefinition>

<ColumnDefinition Width="3\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Orientation="Horizontal">

<Button FontSize="18" Click="BackButton\_Click" x:Name="BackButton" Style="{DynamicResource MenuButton}" Padding="16, -4,16,0" Margin="32,16,4, 16" Content="←">

<Button.BorderBrush>

<LinearGradientBrush EndPoint="1,0" StartPoint="0,0">

<GradientStop Color="#FF1467FF" Offset="1"/>

<GradientStop Color="#FFFC00FF" Offset="0.208"/>

</LinearGradientBrush>

</Button.BorderBrush>

<Button.Background>

<LinearGradientBrush EndPoint="1,0" StartPoint="0,0">

<GradientStop Color="#FF1467FF" Offset="1"/>

<GradientStop Color="#FFFC00FF" Offset="0.208"/>

</LinearGradientBrush>

</Button.Background>

</Button>

<Button FontSize="18" Click="ForeButton\_Click" x:Name="ForeButton" Style="{DynamicResource MenuButton}" Padding="16,-4,16,0" Margin="4,16,8, 16" Content="→"></Button>

<Button Click="MainPageButton\_Click" x:Name="MainPageButton" Style="{DynamicResource MenuButton}" Padding="16,0,16,0" Margin="8,16,8, 16" Content="Главная"></Button>

<Button IsEnabled="false" Click="ProfilePageButton\_Click" x:Name="ProfilePageButton" Style="{DynamicResource MenuButton}" Padding="16,0,16,0" Margin="8,16,8, 16" Content="Профиль"></Button>

<Button Click="GamesPageButton\_Click" x:Name="GamesPageButton" Style="{DynamicResource MenuButton}" Padding="16,0,16,0" Margin="8,16,8, 16" Content="Игры"></Button>

<Button Click="UsersPageButton\_Click" x:Name="UsersPageButton" Style="{DynamicResource MenuButton}" Padding="16,0,16,0" Margin="8,16,8, 16" Content="Пользователи">

<Button.BorderBrush>

<LinearGradientBrush EndPoint="0,0" StartPoint="1,0">

<GradientStop Color="#FF1467FF" Offset="1"/>

<GradientStop Color="#FFFC00FF" Offset="0.208"/>

</LinearGradientBrush>

</Button.BorderBrush>

<Button.Background>

<LinearGradientBrush EndPoint="0,0" StartPoint="1,0">

<GradientStop Color="#FF1467FF" Offset="1"/>

<GradientStop Color="#FFFC00FF" Offset="0.208"/>

</LinearGradientBrush>

</Button.Background>

</Button>

</StackPanel>

<TextBlock x:Name="StatusBlock" Grid.Column="1" Foreground="Gray" FontWeight="Light" FontSize="11" Margin="0,-8,32,0" Text="" TextAlignment="Right" VerticalAlignment="Center"></TextBlock>

</Grid>

<Frame NavigationUIVisibility="Hidden" x:Name="MainFrame" Grid.Row="1"></Frame>

</Grid>

</Window>

Листинг 4.1 –MainWindow.xaml

При запуске приложения открывается именно окно MainWindow.xaml. В качестве стартовой страницы открывается страница MainPage.xaml, на которой пользователь может авторизироваться либо зарегистрировать новую учетную запись.

Дополнительно внутри окна MainWindow.xaml реализована функция отслеживания подключения к интернету, так как изображения загружаются из всемирной паутины, и пользователь должен быть уведомлен о том, могут ли быть загружены изображения в данный момент.

public static async void CheckConnection(object sender, EventArgs e)

{

var client = new HttpClient();

try

{

var result = await client.GetAsync("http://google.com");

if (result.StatusCode.ToString() != "OK")

{

MainWindow.link.StatusBlock.Text = "Оффлайн режим \n Изображения недоступны";

}

else

{

MainWindow.link.StatusBlock.Text = "";

}

}

catch

{

MainWindow.link.StatusBlock.Text = "Оффлайн режим \n Изображения недоступны";

}

}

Листинг 4.2 –Метод CheckConnection

Весь функционал приложения реализуется в окнах и на страницах. Функции регистрации, редактирования профиля, оставления отзыва и добавления игры реализованы с помощью окон. Функции авторизации, поиска игр и отзывов, отображение информации о пользователях и играх реализованы с помощью страниц.

Процесс авторизации реализован с помощью сверки данных, введенных пользователем и данных хранящихся в таблице с пользователями базы данных. Для подключения к базе данных используется технология ADO.NET. Так как в проекте используется база данных MSSQL Server, то в качестве провайдера данных был выбран Microsoft.Data.SqlClient. Технология подключается к проекту с помощью службы управления пакетами NuGet (рис. 4.1)

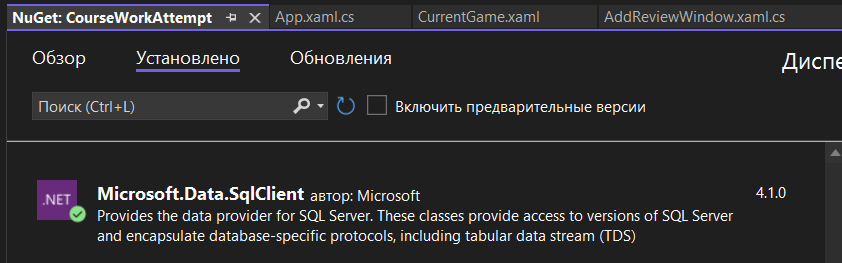


Рисунок 4.1 –Подключение Microsoft.Data.SQLClient

## 4.2. Описание классов и методов программного средства

## 4.2.1. Управление учетной записью пользователя

Функционал для возможности авторизации, регистрации, удаления учетной записи пользователя, а также проверка подключения к интернету реализован в классе Authorization.

Основным методом класса является метод TryToLogin(), который выполняется по нажатию кнопки «Войти». При сверке введенных пользователем данных пароль хешируется с помощью метода GetHash() класса Crypto для проверки на соответствие учетной записи в базе данных. Проведя успешную авторизацию, пользователю становится доступна страница профиля, а также появляются возможность оставить отзыв, добавить/удалить игру (для администратора).

Для регистрации используется метод RegisterAccount(). После введения гостем корректных данных в окне регистрации запись о нем появляется в базе данных, и он становиться пользователем приложения и может успешно войти в систему.

Редактирование учетной записи осуществляется в окне EditProfileWindow.xaml посредством изменения пользовательских данных и ввода текущего пароля пользователя. За обновление данных отвечает метод UpdateUser() класса User.

Для удаления учетной записи применяется метод DeleteAccount(). Удаление аккауна возможно после успешной авторизации.

## 4.2.2. Просмотр игр

Все игры, доступные в приложении отображаются на странице Games.xaml. Игры как объект программы представлены классами Game, Publisher, Developer. Метод GetList() класса Game позволяет получить список всех игр из базы данных. Метод AddGame() позволяет администратору пополнить список новой игрой. Метод Delete() отвечает за удаление игры.

За сортировку и поиск по всем играм отвечает метод SearchFunc() страницы Games, который позволяет выбрать игры определенного жанра, либо осуществить поиск по названию и платформе. Блок-схема алгоритма поиска представлена в приложении Д.

Для просмотра информации о конкретной игре создана страница CurrentGame.xaml.

## 4.2.3. Просмотр пользователей

Все пользователи, зарегистрированные в приложении, отображаются на странице Users.xaml. Каждый пользователь представлен в виде объекта класса User. Получить список всех пользователей можно с помощью метода GetList() класса User.

Поиск пользователей доступен по имени, фамилии, нику, адресу электронной почты и реализуется методом SearchFunc() страницы Users.

Для просмотра информации о пользователе создана страница CurrentUser.xaml.

## 4.2.4. Просмотр отзывов

Отзывы, которые оставили пользователи можно найти на странице с игрой либо на странице конкретного пользователя. Отзыв реализован с помощью класса Review. Данный класс реализует метод GetCurrentGameReviews(), с помощью которого можно получить все отзывы на определенную игру. Метод GetTopGames() позволяет получить пять самых высоко оцениваемых игр пользователя. Метод GetAverageRate() на основе оставленных отзывах об игре вычисляет ее средний рейтинг.

## 4.2.5. Добавление отзыва

Пользователь может оставить отзыв на странице с игрой, которая его интересует. За добавление отзыва отвечает метод AddReview() класса Review. Предусмотрена возможность обновление информации в отзыве с помощью метода Update().

## 4.2.6. Добавление/удаление игр

Добавление/удаление игр администратором приложения реализуется с помощью методов AddGame() и Delete() класса Game. Для добавления игры создано окно AddGameWindow.xaml.

## 4.3. Реализация паттерна Singleton

Для гарантирования того, что главные страницы приложения будут созданы лишь один раз, с последующим редактированием компонентов внутри них в проекте применяется паттерн Singleton. Его реализация представлена на листинге 4.3.

private static Games gamesPage;

public static Games GetPage()

{

if (gamesPage == null)

gamesPage = new Games();

return gamesPage;

}

Листинг 4.3 –Паттерн Singleton в классе Games

# 5. Тестирование, проверка работоспособности и анализ полученных результатов

## 5.1. Тестирование авторизации и регистрации

Во время авторизации и регистрации пользователь может вводить некорректные или несуществующие данные. Эти ситуации называются исключениями и обрабатывается программным продуктом с помощью показа пользователю сообщения об ошибке. Примеры обработки различных видов исключительных ситуаций представлены на рисунках 5.1 – 5.5.

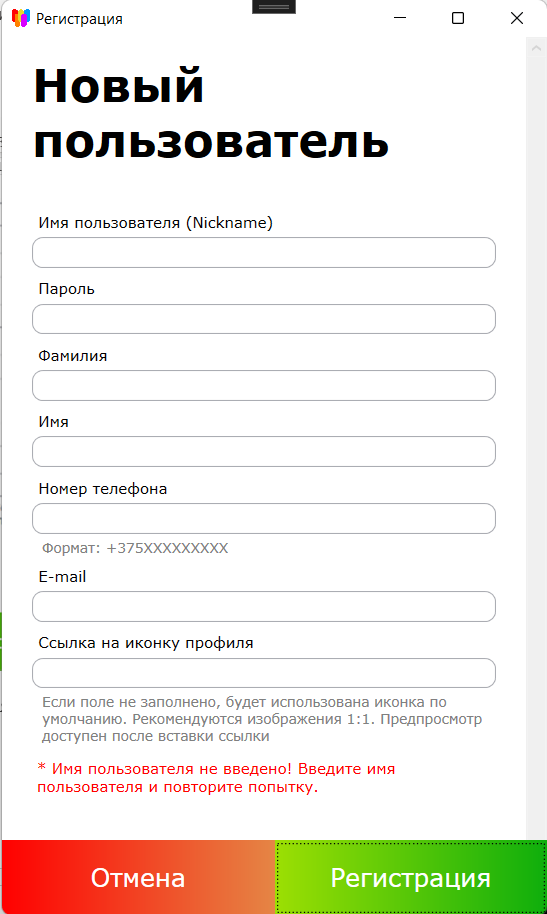


Рисунок 5.1 –Незаполненное поле при регистрации

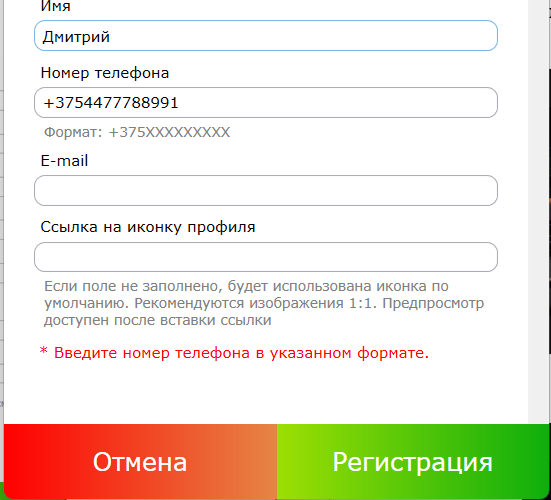


Рисунок 5.2 –Неверный формат ввода данных

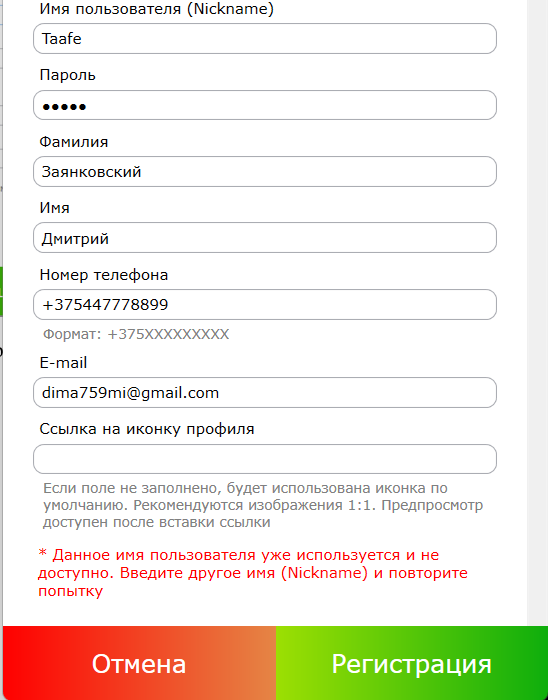


Рисунок 5.3 –Попытка использования занятого имени пользователя

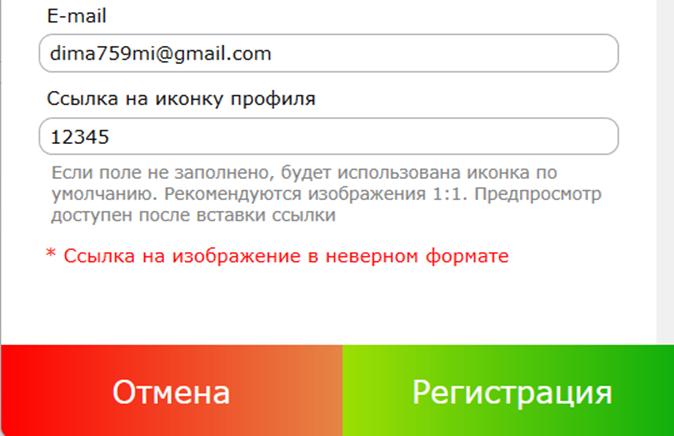


Рисунок 5.4 –Неверный формат ссылки на изображение

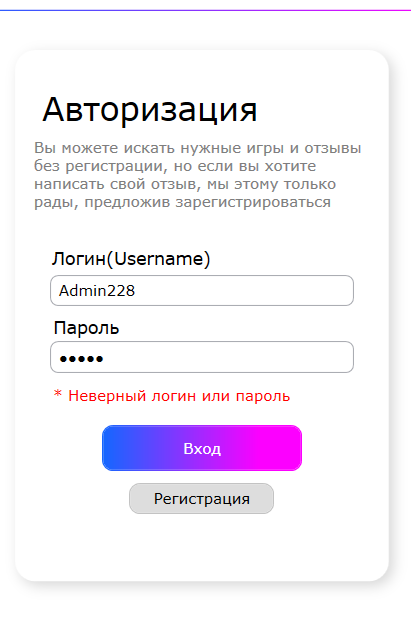


Рисунок 5.5 –Некорректные данные при авторизации

## 5.2. Тестирование редактирования учетной записи

При редактировании учетной записи срабатывают ограничения как при регистрации. Для того, чтобы редактировать учетную запись, пользователю необходимо вводить текущий пароль. При неверном пароле возникает ситуация, представленная на рисунке 5.6.

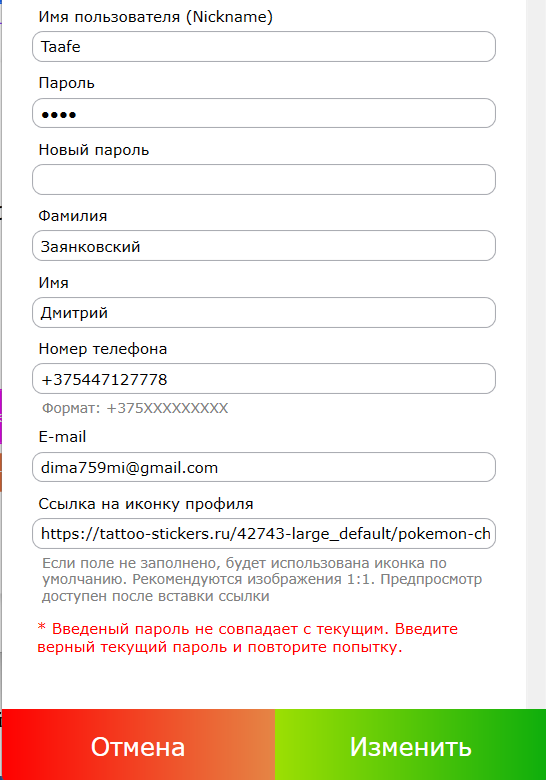


Рисунок 5.6 –Неверный текущий пароль

При изменении ссылки на изображение сразу доступен предпросмотр. Если поле для ссылки оставить пустым, будет использовано изображение по умолчанию. Изменение изображения профиля представлено на рисунке 5.7.

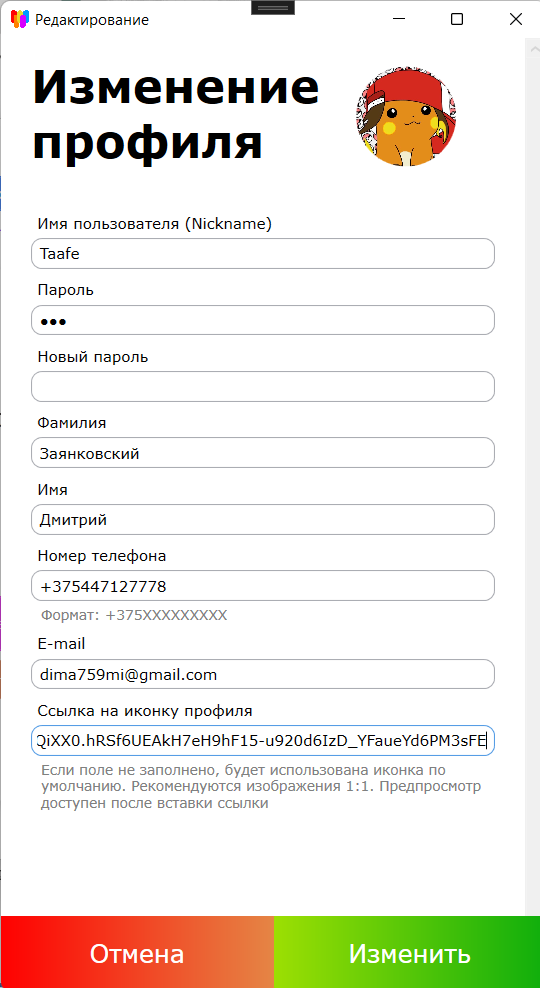


Рисунок 5.7 –Предпросмотр иконки профиля

## 5.3. Тестирование оставления отзыва

Пользователь, выбрав интересующую игру может оставить на неё отзыв. Текстовое поле для оставления отзыва должно быть заполнено, иначе возникнет исключительная ситуация, представленная на рисунке 5.8.

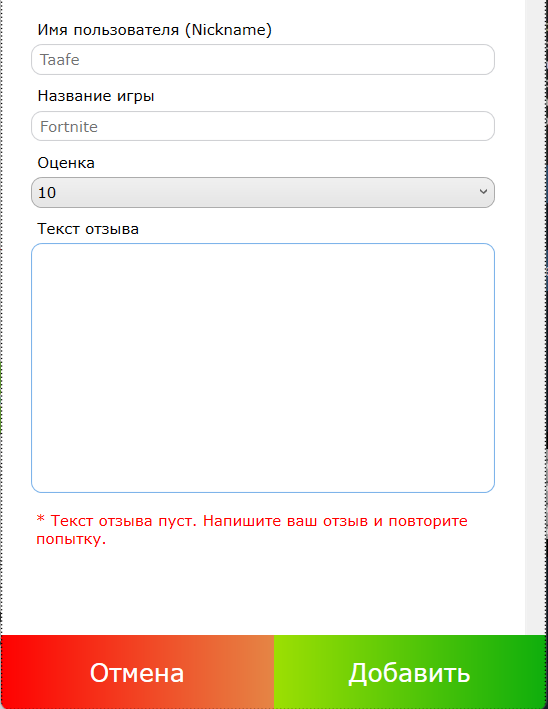


Рисунок 5.8 –Незаполненный текст отзыва

## 5.4. Тестирование добавления игры

Добавление новой игры в базу данных – это функция администратора. Для этого доступно окно, все поля которого необходимо заполнить. При этом все исключительные ситуации обрабатываются. Для изображений игры доступен предпросмотр сразу же после изменения ссылки. Сообщение об ошибке доступно внизу окна. Пример возникновения ошибки при заполнении полей представлен на рисунках 5.9 – 5.10.

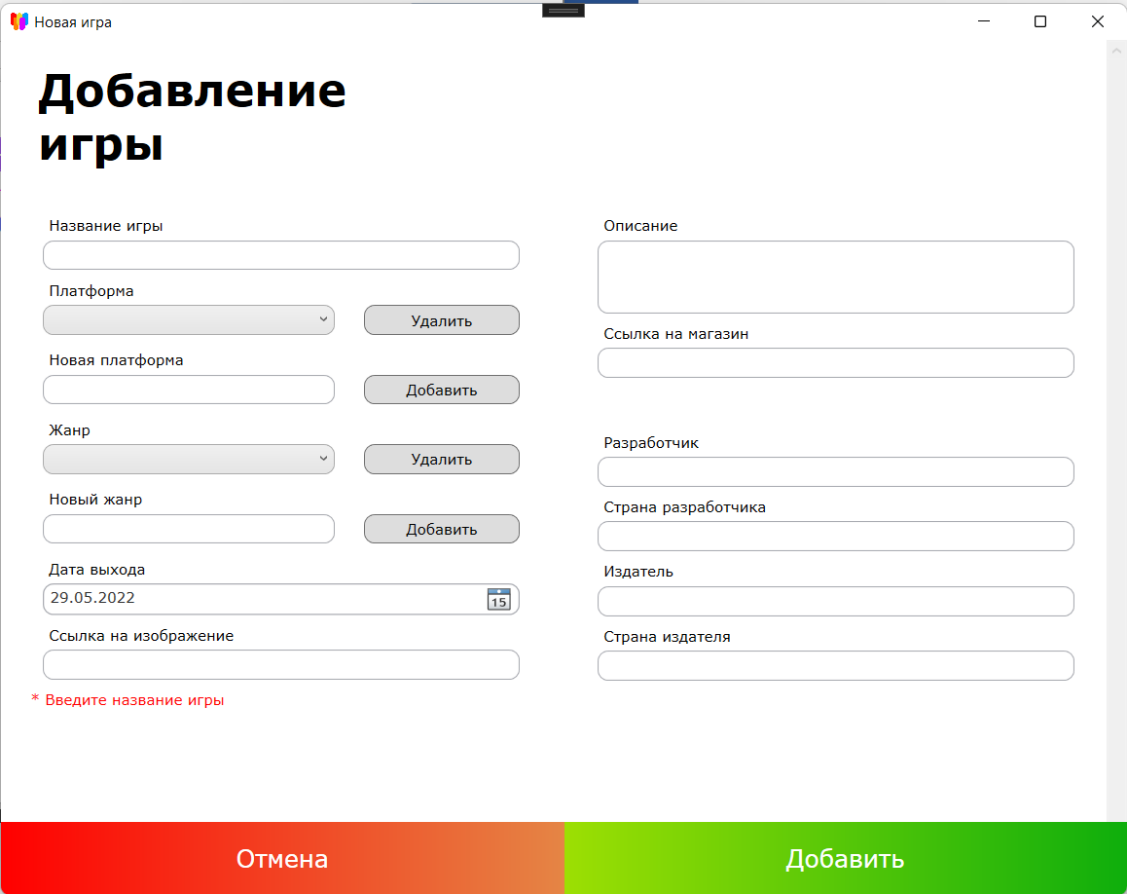


Рисунок 5.9 –Незаполненное поле

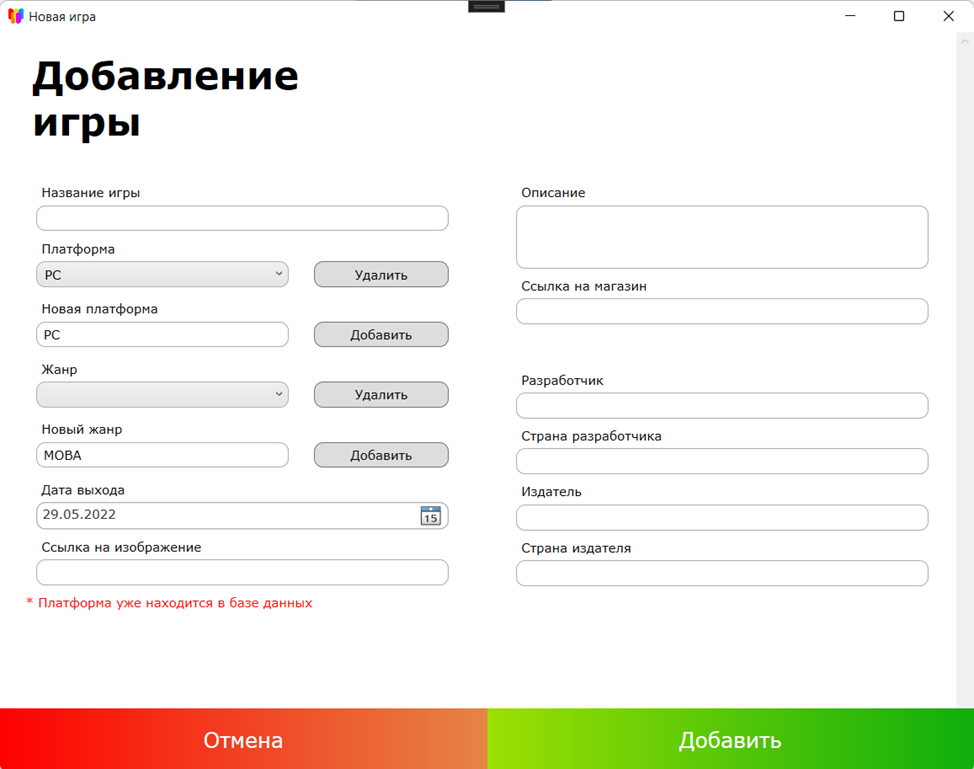


Рисунок 5.10 –Добавление уже существующих данных

# 6. Руководство по использованию

При запуске приложения пользователь попадает на главную страницу, на которой ему предлагается авторизоваться либо зарегистрироваться.

По умолчанию новый пользователь получает одноименную роль, а роль «Администратор» заранее установлена для определенной группы пользователей.

Окно регистрации представлено на рисунке 6.1.

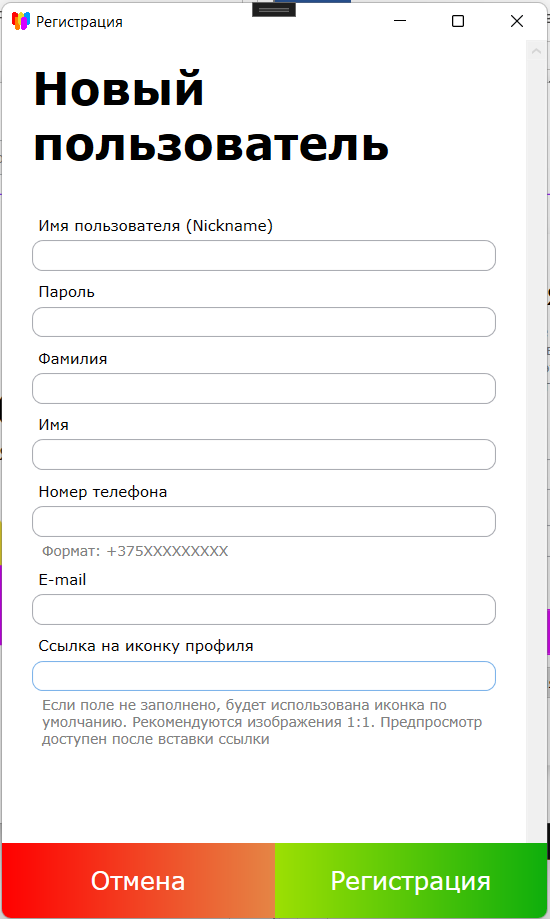


Рисунок 6.1 –Окно регистрации

Страница авторизации представлена на рисунке 6.2.

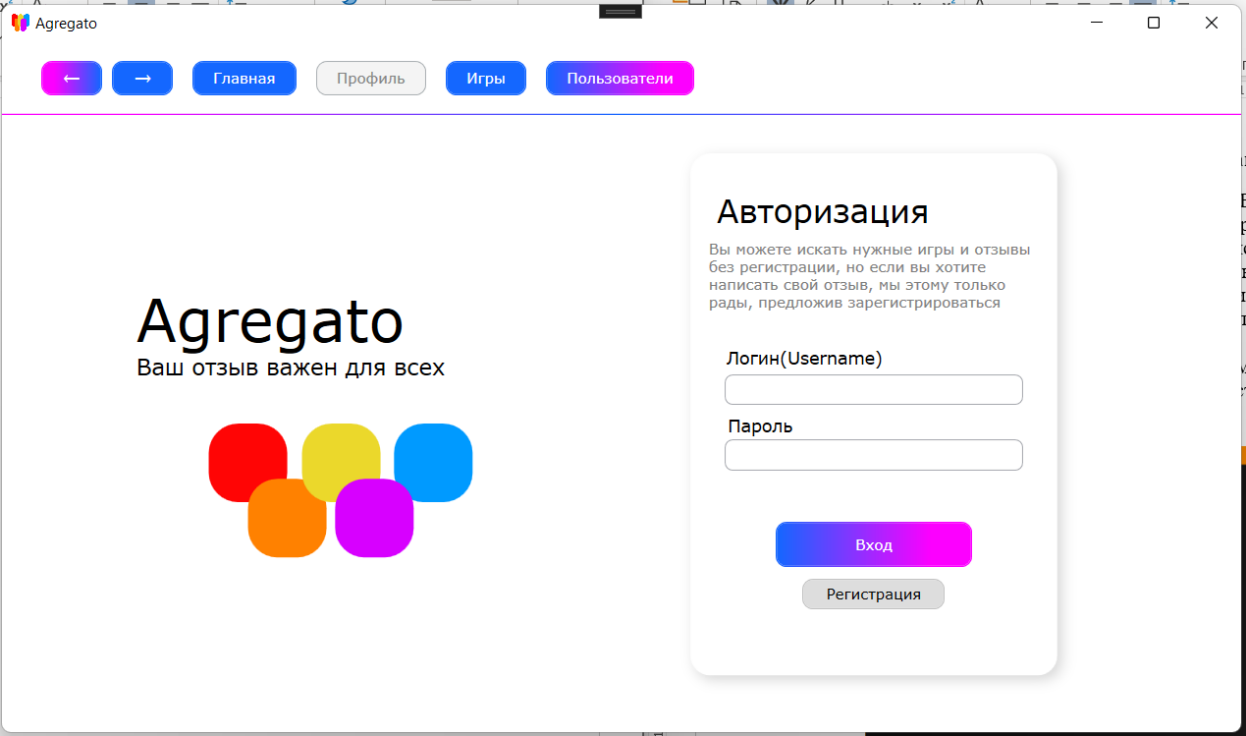


Рисунок 6.2 –Страница авторизации

После авторизации пользователю становится доступна страница «Профиль». Кроме того, пользователь может оставить отзыв на любую доступную игру. На странице «Профиль» отображается вся личная информация. Присутствует возможность редактирования, удаления учетной записи. Страница «Профиль» представлена на рисунке 6.3.

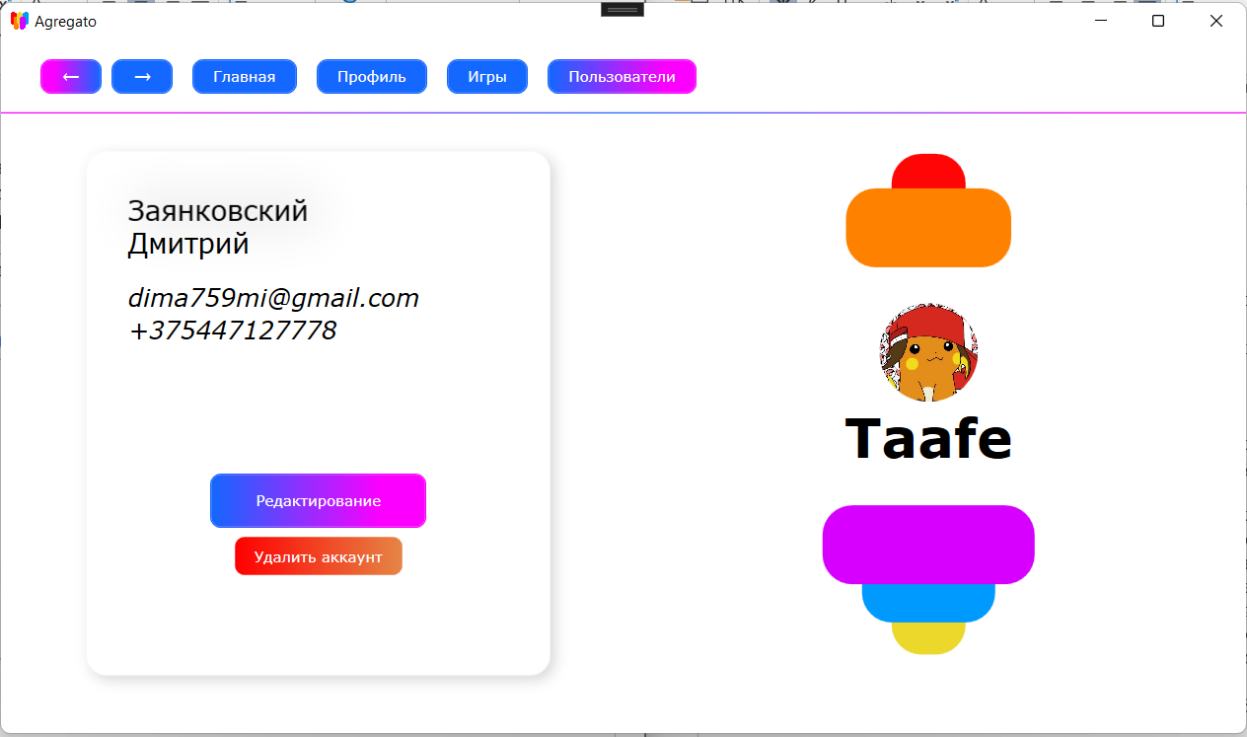


Рисунок 6.3 –Страница «Профиль»

Список всех доступных игр доступен пользователю на странице «Игры». На этой странице есть поиск и фильтрация по жанру. Страница «Игры» представлена на рисунке 6.4.

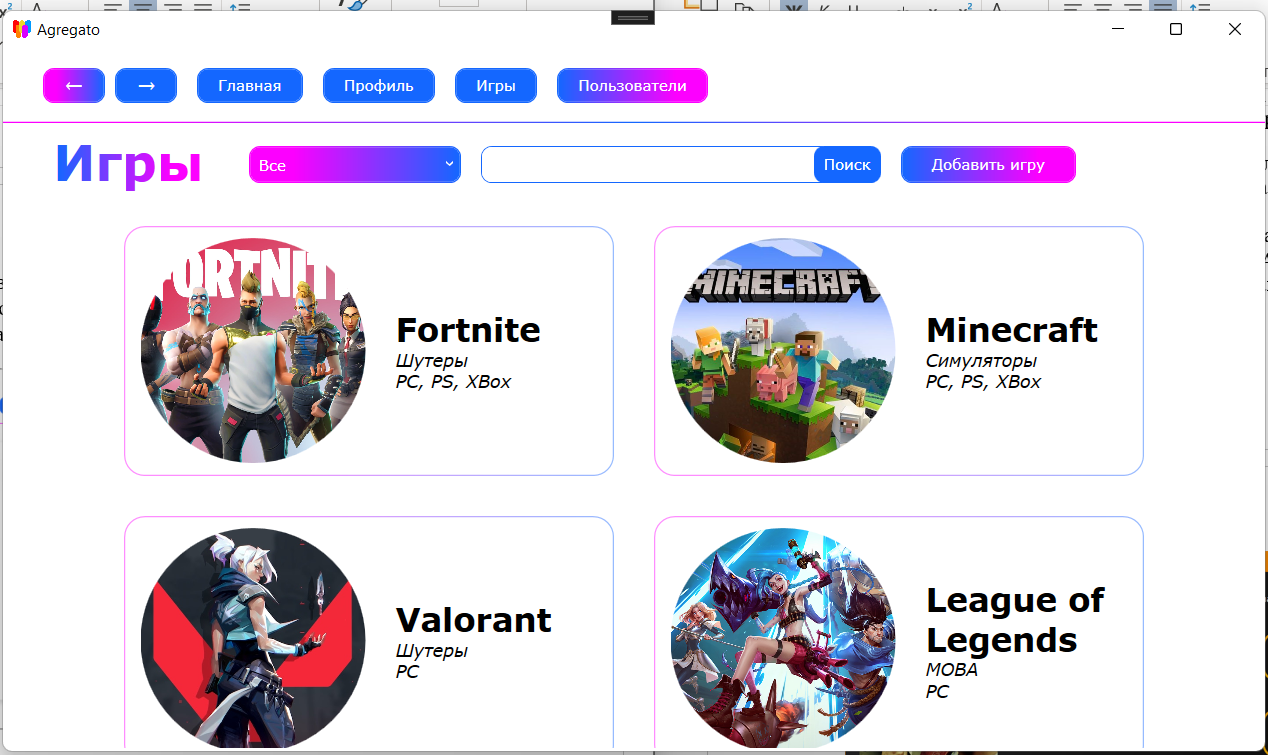


Рисунок 6.4 –Страница «Игры»

Список всех зарегистрированных пользователей доступен на странице «Пользователи». Данная страница представлена на рисунке 6.5.

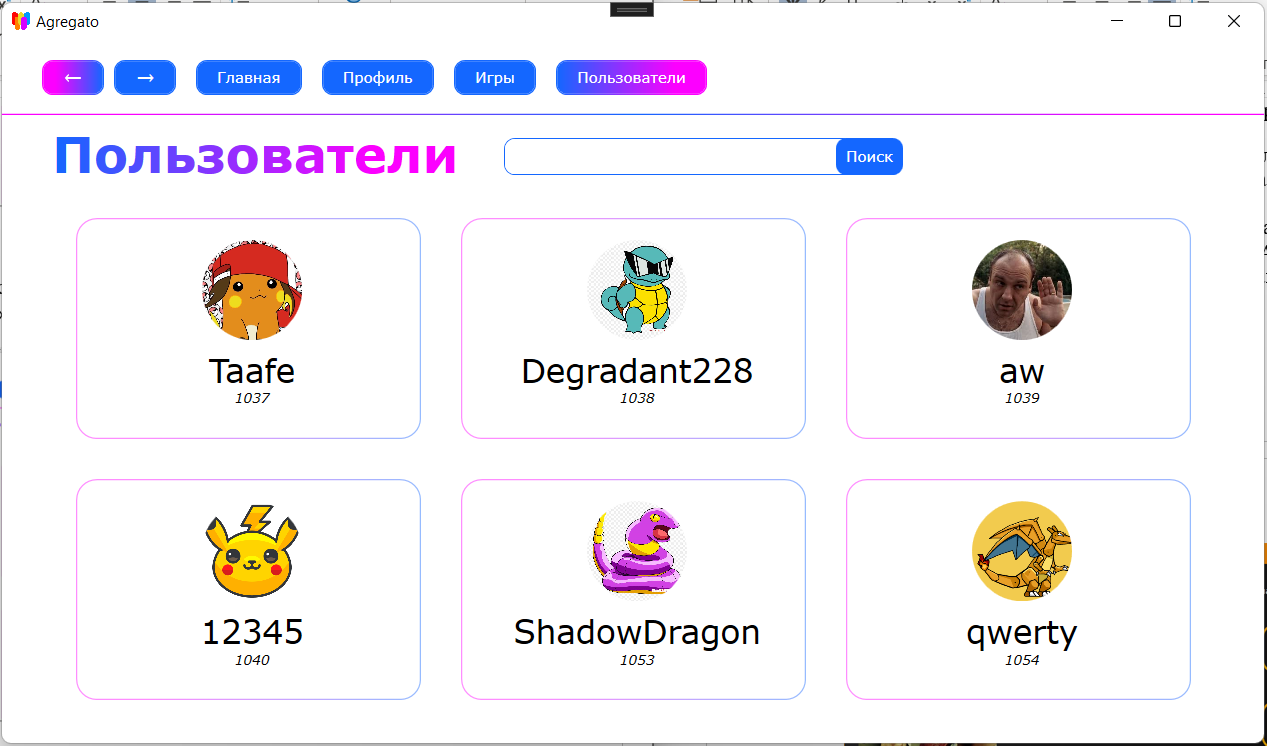


Рисунок 6.5 –Страница «Пользователи»

Перейдя на страницу пользователя, можно просмотреть информацию о нем, а также его топ 5 игр. Пример такой страницы представлен на рисунке 6.6.

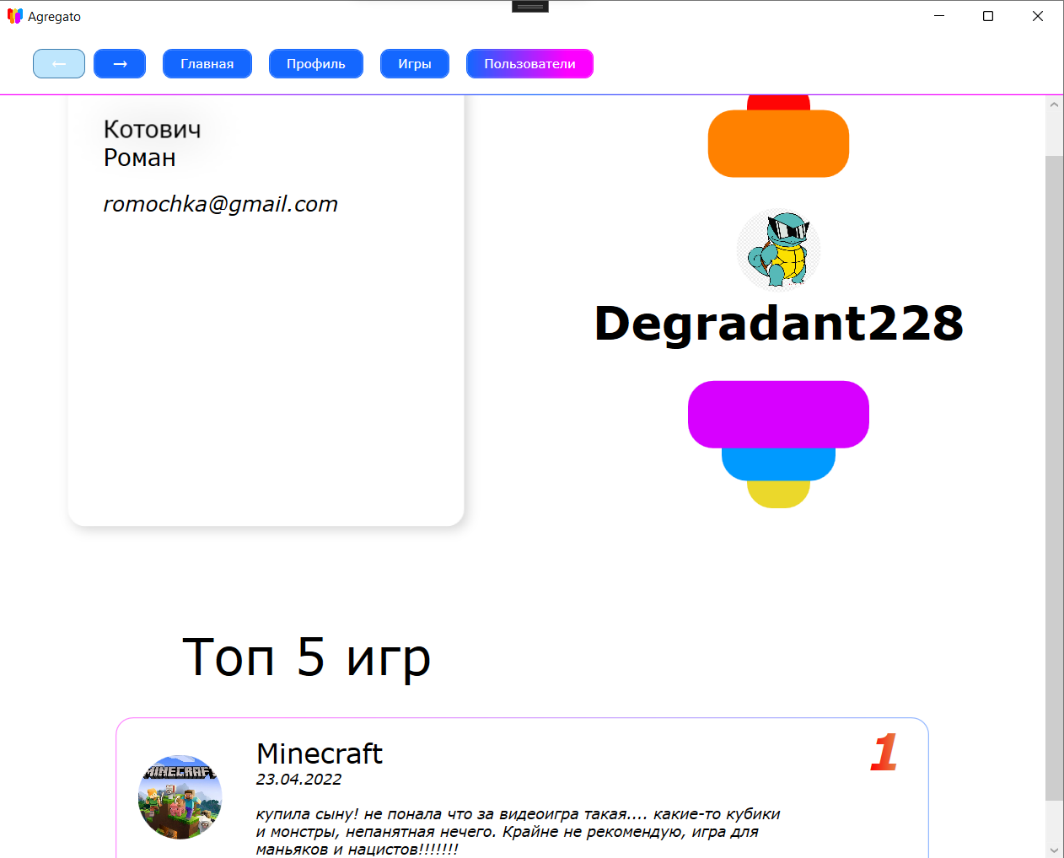


Рисунок 6.6 –Страница пользователя

Перейдя на страницу с игрой, пользователь может ознакомиться с информацией об игре, просмотреть её общий рейтинг и оставить собственный отзыв. Страница игры представлена на рисунке 6.7.

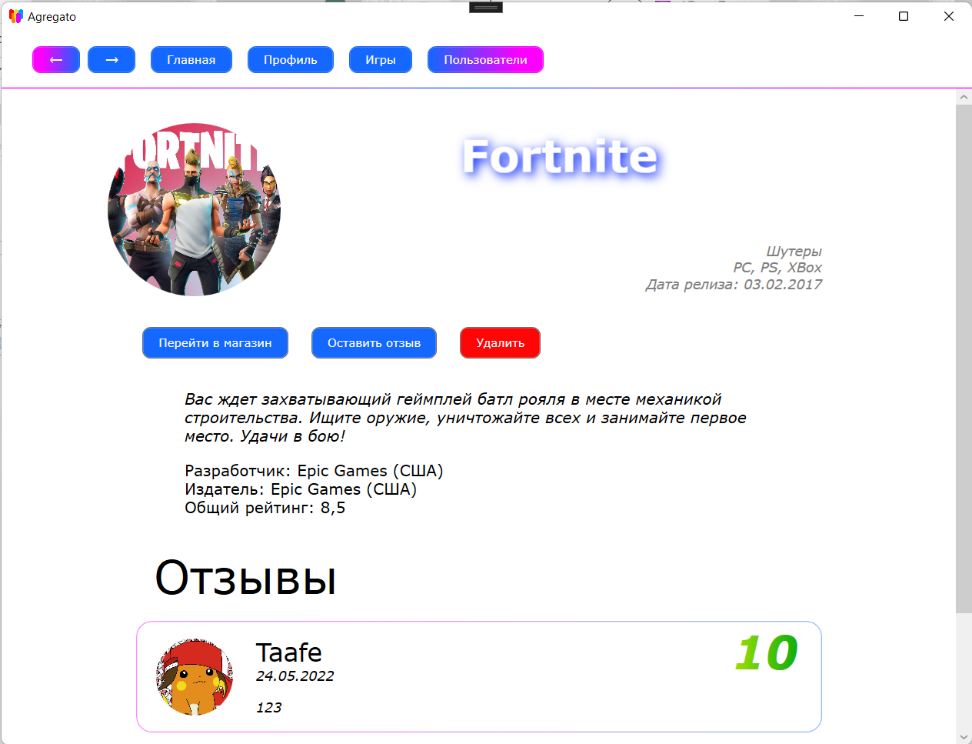


Рисунок 6.7 –Страница игры

Окно для оставления отзыва представлено на рисунке 6.8.

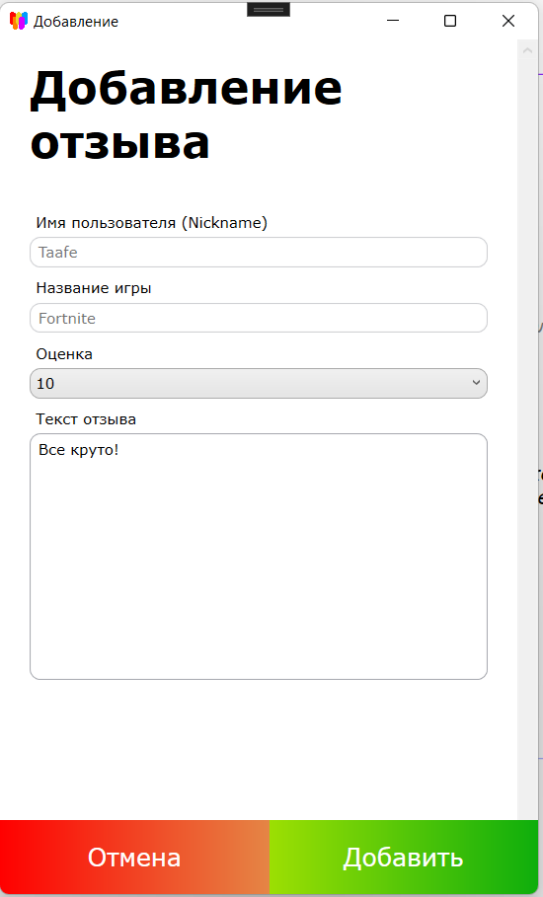


Рисунок 6.8 –Окно «Добавить отзыв»

Для администратора доступно окно добавления игры. Это окно представлено на рисунке 6.9.

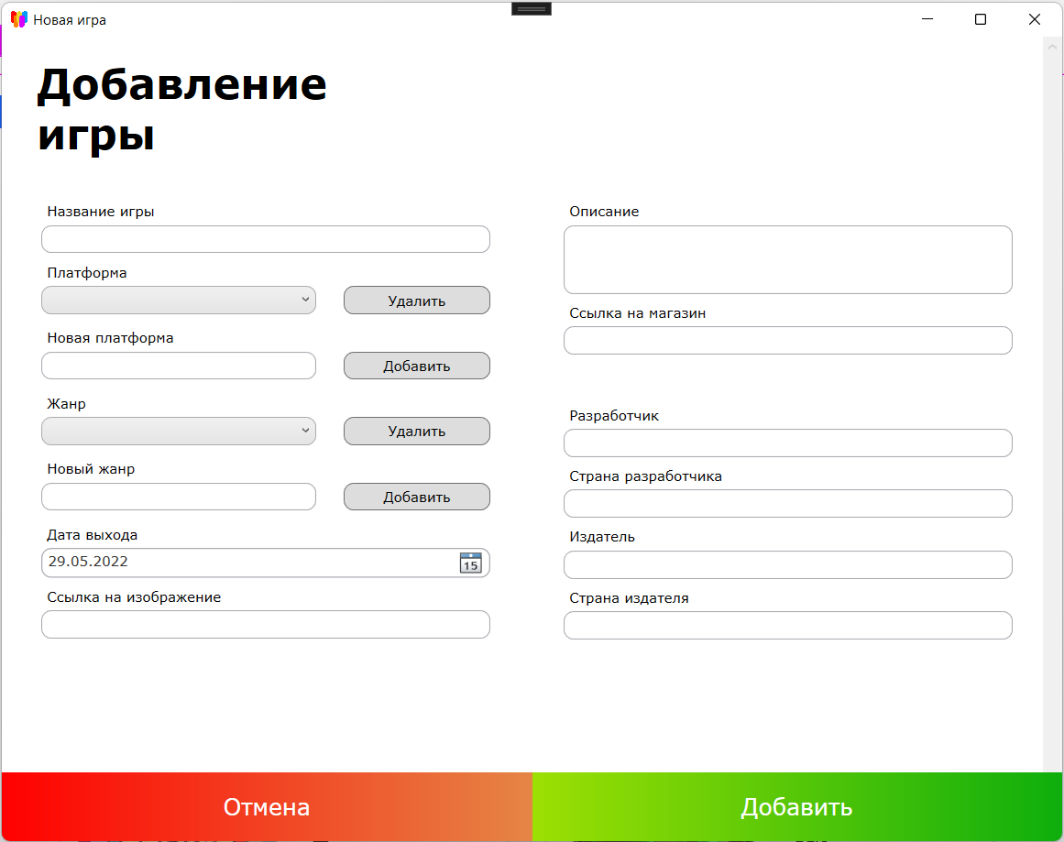


Рисунок 6.9 –Окно «Добавить игру»

# Заключение

В результате выполнения курсовой работы было разработано приложение агрегатор отзывов «Agregato» на языке C# с использованием технологий ADO.NET, WPF.

При разработке программного продукта были реализованы все предполагаемые функции приложения, а именно:

* создана база данных;
* реализована возможность авторизации/регистрации;
* реализована возможность просмотра информации об играх и пользователях;
* реализована возможность оставления отзыва;
* реализована возможность добавления/удаления игры для администратора;
* реализована возможность редактирования учетной записи:
* реализована функция подсчета средней оценки игры на основе оставленных отзывов.

Тестирование приложения показало, что оно работает корректно и способно выполнить свои функции.

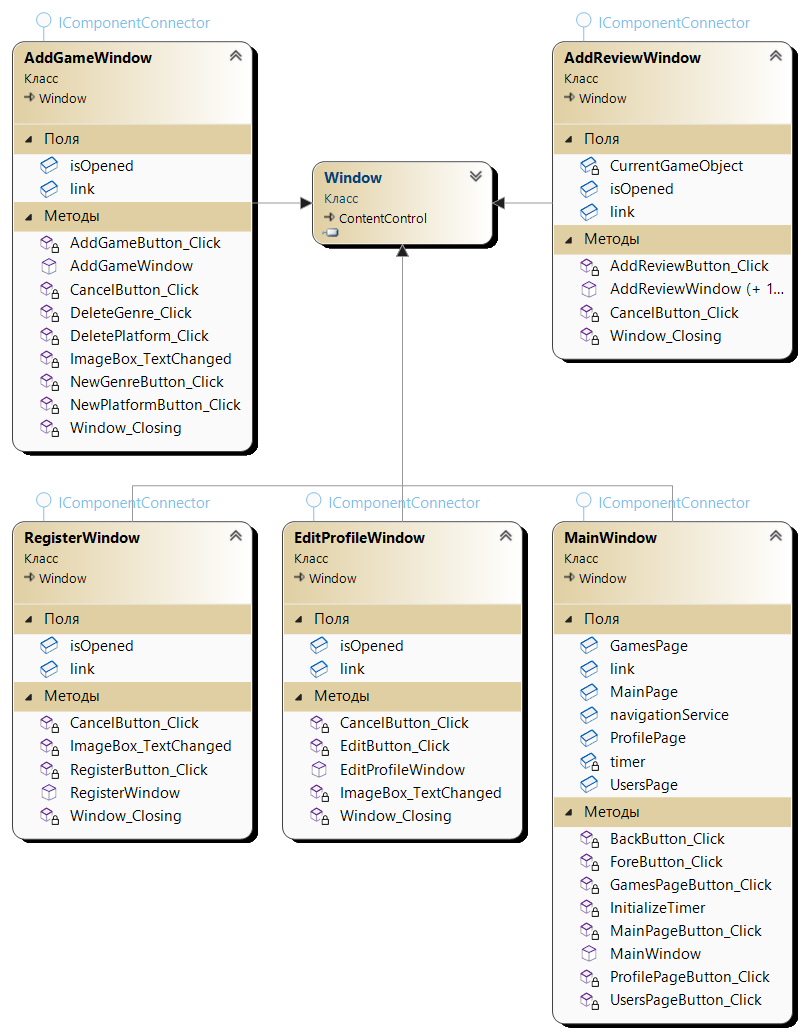
В соответствии с полученным результатом работы программы можно сделать вывод, что разработанный продукт работает верно, а требования технического задания выполнены в полном объеме.

# Список литературы

1. Полное руководство по языку программирования С# 10 и платформе .NET 6 [Электронный ресурс] – https://metanit.com/sharp/tutorial/ – Дата доступа 13.03.2022
2. Пацей, Н.В. Курс лекций по языку программирования C# / Н. В. Пацей. – Минск: БГТУ, 2018. – 175 с.
3. Руководство по WPF // [Электронный ресурс]. – Режим доступа: https://metanit.com/sharp/wpf/ – Дата доступа: 09.04.2022
4. Руководство по ADO.NET и работе с базами данных в .NET 6 [Электронный ресурс] – https://metanit.com/sharp/adonetcore/ – Дата доступа 03.05.2022
5. Блинова, Е.А. Курс лекций по Базам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

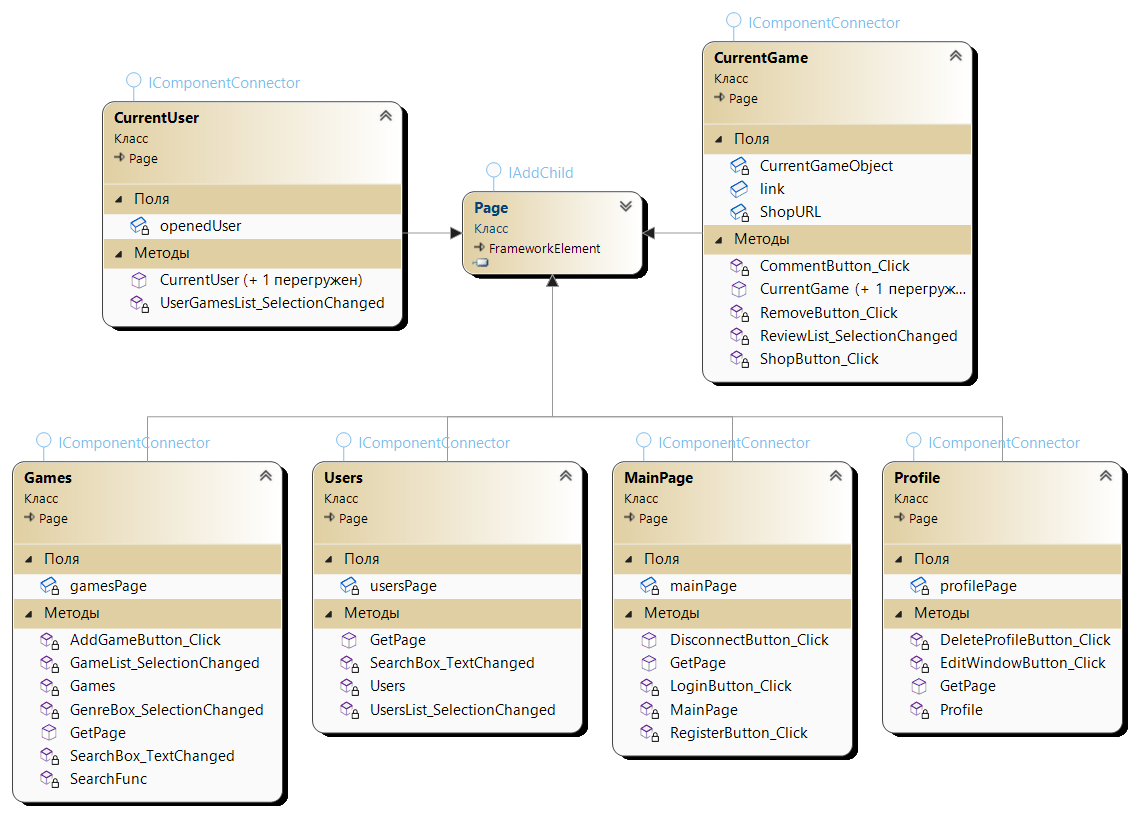
# ПРИЛОЖЕНИЕ А

Диаграмма классов окон приложения



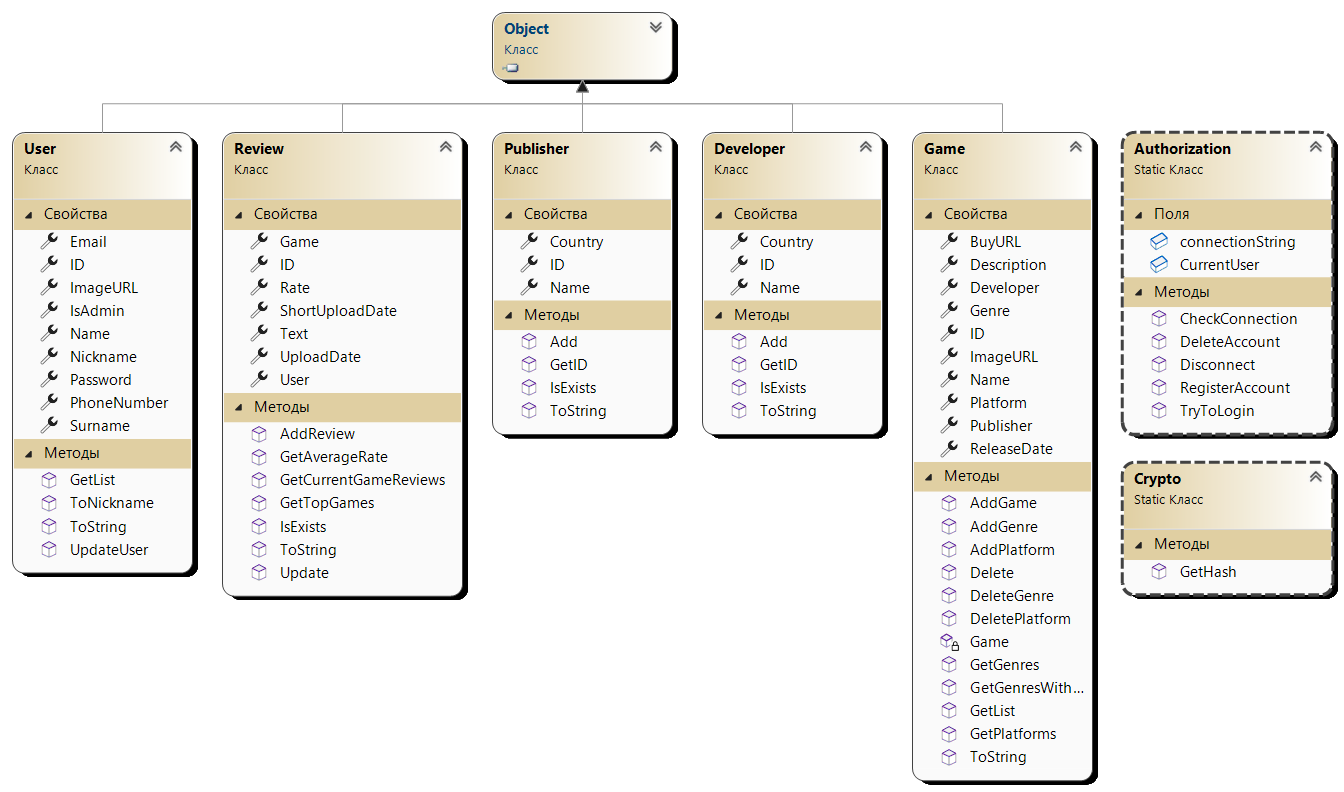
# ПРИЛОЖЕНИЕ Б

Диаграмма классов страниц приложения



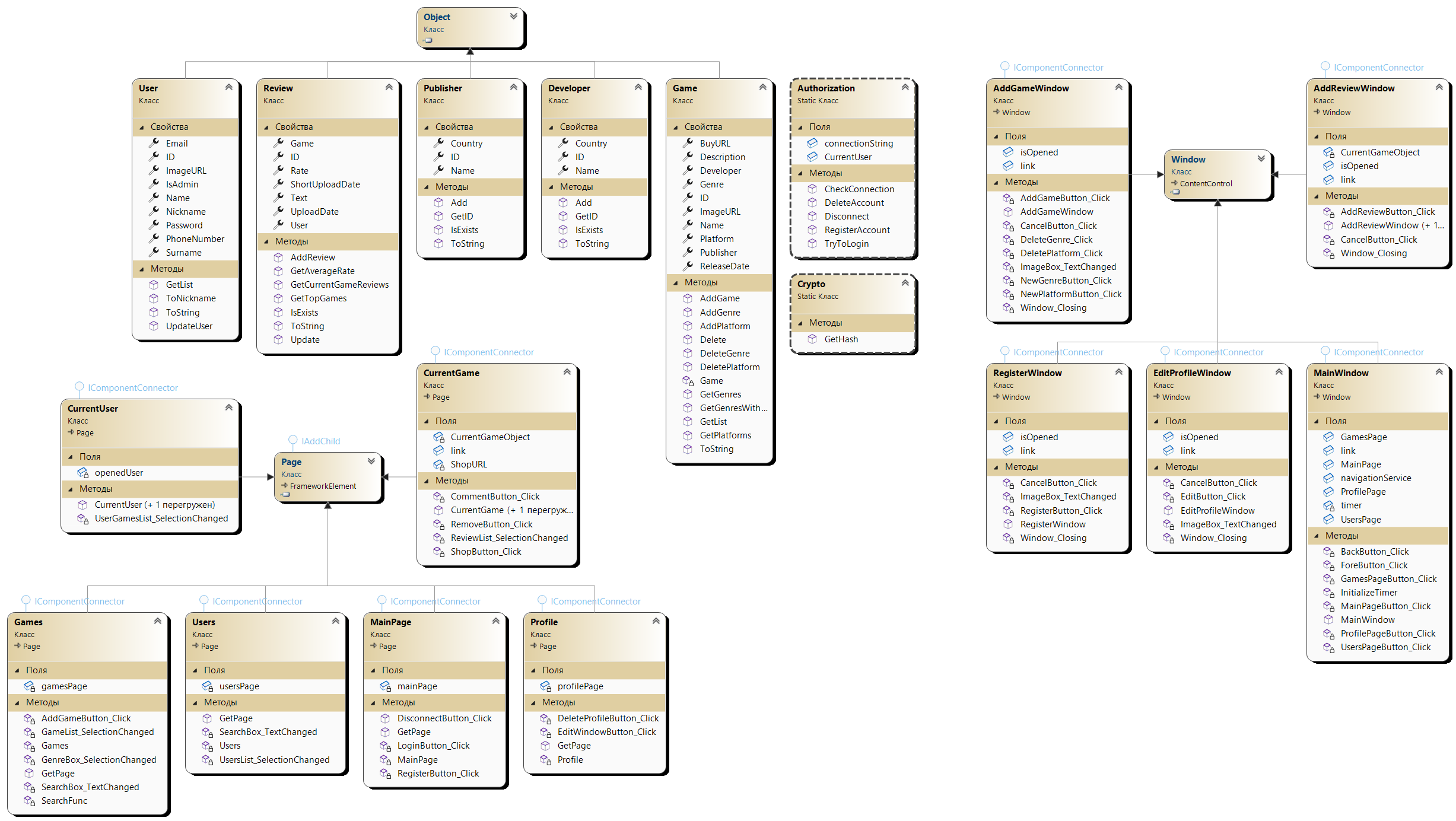
# ПРИЛОЖЕНИЕ В

Диаграмма функциональных классов



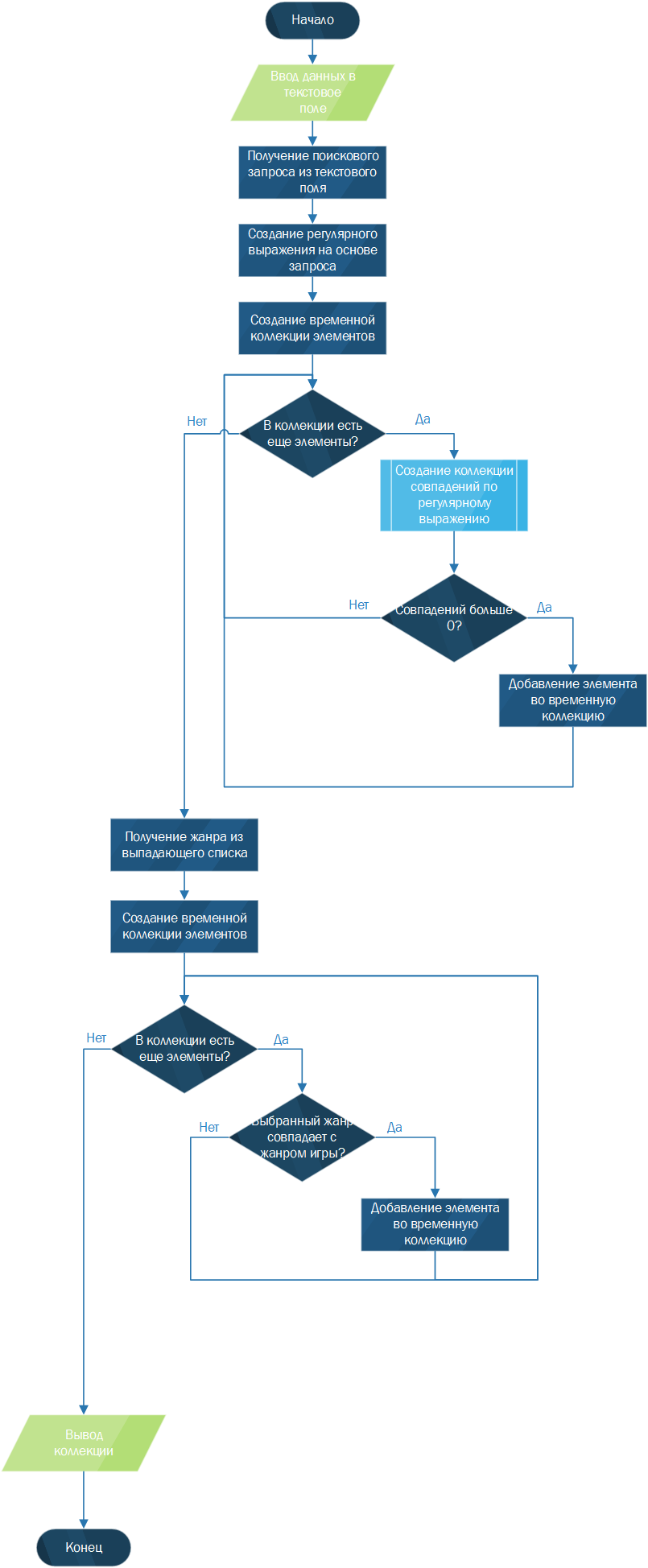
# ПРИЛОЖЕНИЕ Г

Диаграмма классов



# ПРИЛОЖЕНИЕ Д

Блок-схема алгоритма поиска



# ПРИЛОЖЕНИЕ Е

Листинг 1: Developer.cs

using CourseWorkAttempt.Auth;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace CourseWorkAttempt.Classes

{

public class Developer

{

public int ID { get; set; }

public string? Name { get; set; }

public string? Country { get; set; }

public override string ToString()

{

return Name + " " + Country;

}

public static bool IsExists(string devName)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = $"select count(DeveloperName) from Developers where DeveloperName = @devName";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

var state = Convert.ToInt32(command.ExecuteScalar());

if (state > 0)

{

isSuccessful = true;

}

}

catch (Exception ex)

{

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

//AddReviewWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

return isSuccessful;

}

}

public static bool Add(string name, string country)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

//MessageBox.Show(DateTime.Now.ToShortDateString());

string sqlExpression = $"insert into Developers values(@name, @country)";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

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

try

{

var state = command.ExecuteNonQuery();

//MessageBox.Show("Игра добавлена!", "Новая игра", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

}

catch (Exception ex)

{

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

//AddGameWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

return isSuccessful;

}

}

public static int GetID(string devName)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

int id = 0;

connection.Open();

string sqlExpression = $"select DeveloperID from Developers where DeveloperName = @devName";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

if (reader.Read()) // построчно считываем данные

{

id = (int)reader["DeveloperID"];

}

}

}

return id;

}

}

}

}

Листинг 2: Publisher.cs

using CourseWorkAttempt.Auth;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace CourseWorkAttempt.Classes

{

public class Publisher

{

public int ID { get; set; }

public string? Name { get; set;}

public string? Country { get; set; }

public override string ToString()

{

return Name + " " + Country;

}

public static bool IsExists(string pubName)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = $"select count(PublisherName) from Publishers where PublisherName = @pubName";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

var state = Convert.ToInt32(command.ExecuteScalar());

if (state > 0)

{

isSuccessful = true;

}

}

catch (Exception ex)

{

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

//AddReviewWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

return isSuccessful;

}

}

public static bool Add(string name, string country)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

//MessageBox.Show(DateTime.Now.ToShortDateString());

string sqlExpression = $"insert into Publishers values(@name, @country)";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

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

try

{

var state = command.ExecuteNonQuery();

//MessageBox.Show("Игра добавлена!", "Новая игра", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

}

catch (Exception ex)

{

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

//AddGameWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

return isSuccessful;

}

}

public static int GetID(string pubName)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

int id = 0;

connection.Open();

string sqlExpression = $"select PublisherID from Publishers where PublisherName = @pubName";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

if (reader.Read()) // построчно считываем данные

{

id = (int)reader["PublisherID"];

}

}

}

return id;

}

}

}

}

Листинг 3: User.cs

using CourseWorkAttempt.Auth;

using CourseWorkAttempt.Windows;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace CourseWorkAttempt.Classes

{

public class User

{

public int ID { get; set; }

public string? Nickname { get; set; }

public string? Name { get; set; }

public string? Surname { get; set; }

public string? Password { get; set; }

public string? PhoneNumber { get; set; }

public string? Email { get; set; }

public bool IsAdmin { get; set; }

public string? ImageURL { get; set; }

public override string ToString()

{

return $"{ID} {Nickname} {Name} {Surname} {Email}";

}

public string ToNickname()

{

return Nickname;

}

public static bool UpdateUser()

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

User updatedUser = Authorization.CurrentUser;

bool isSuccessful = false;

connection.Open();

string sqlExpression = @$"update Users set

Nickname = @nickname,

Password = @password,

Surname = @surname,

Name = @name,

Email = @email,

PhoneNumber = @phone,

UserImage = @image

where UserID = @userID";

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.Add(new SqlParameter("@nickname", updatedUser.Nickname));

command.Parameters.Add(new SqlParameter("@password", updatedUser.Password));

command.Parameters.Add(new SqlParameter("@surname", updatedUser.Surname));

command.Parameters.Add(new SqlParameter("@name", updatedUser.Name));

command.Parameters.Add(new SqlParameter("@email", updatedUser.Email));

command.Parameters.Add(new SqlParameter("@phone", updatedUser.PhoneNumber));

command.Parameters.Add(new SqlParameter("@image", updatedUser.ImageURL));

command.Parameters.Add(new SqlParameter("@userID", updatedUser.ID));

try

{

var state = command.ExecuteNonQuery();

isSuccessful = true;

}

catch (Exception ex)

{

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

if (ex.Message.Contains("\"UQ\_Nickname\""))

{

EditProfileWindow.link.ErrorMessageBlock.Text = "\* " + "Данное имя пользователя уже используется и не доступно. Введите другое имя (Nickname) и повторите попытку";

}

else

{

EditProfileWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

}

return isSuccessful;

}

}

public static List<User> GetList()

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

connection.Open();

string sqlExpression = $"select \* from Users";

SqlCommand command = new SqlCommand(sqlExpression, connection);

List<User> usersCollection = new();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

var user = new User();

user.ID = (int)reader["UserID"];

user.Nickname = reader["Nickname"] as string;

user.Name = reader["Name"] as string;

user.Surname = reader["Surname"] as string;

user.PhoneNumber = reader["PhoneNumber"] as string;

user.Email = reader["Email"] as string;

user.IsAdmin = (bool)reader["IsAdmin"];

user.ImageURL = reader["UserImage"] as string;

usersCollection.Add(user);

}

}

}

return usersCollection;

}

}

}

}

Листинг 4: Review.cs

using CourseWorkAttempt.Auth;

using CourseWorkAttempt.Windows;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace CourseWorkAttempt.Classes

{

public class Review

{

public int ID { get; set; }

public User? User { get; set; }

public Game? Game { get; set; }

public string? Text { get; set; }

public DateTime UploadDate { get; set; }

public string? ShortUploadDate { get; set; }

public int Rate { get; set; }

public override string ToString()

{

return Text;

}

public static List<Review> GetTopGames(int userID)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

connection.Open();

string sqlExpression = @$"select top(5) Reviews.GameID, Users.Nickname, Users.UserID, Reviews.UploadDate, Reviews.Rating, Reviews.Text,

Games.GameName, Games.GameImage, Games.GameID

from reviews inner join Games on Reviews.GameID = Games.GameID inner join

users on Users.UserID = Reviews.UserID

where Users.UserID = @userID

order by Rating desc";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

List<Review> ReviewsCollection = new();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

var review = new Review();

var user = new User();

var game = new Game();

game.Name = reader["GameName"] as string;

game.ImageURL = reader["GameImage"] as string;

game.ID = Convert.ToInt32(reader["GameID"]);

user.Nickname = reader["Nickname"] as string;

review.Game = game;

review.User = user;

review.Rate = Convert.ToInt32(reader["Rating"]);

review.Text = reader["Text"] as string;

DateTime uploadTime = Convert.ToDateTime(reader["UploadDate"].ToString());

review.UploadDate = uploadTime;

review.ShortUploadDate = uploadTime.ToShortDateString();

ReviewsCollection.Add(review);

}

}

}

return ReviewsCollection;

}

}

public static List<Review> GetCurrentGameReviews(int gameID)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

connection.Open();

string sqlExpression = $"select Users.UserImage, Reviews.GameID, Users.Nickname, Reviews.UploadDate, Reviews.Rating, Reviews.Text from users inner join reviews on Users.UserID = Reviews.UserID where Reviews.GameID = @gameID order by Reviews.UploadDate desc";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

List<Review> ReviewsCollection = new();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

var review = new Review();

var user = new User();

user.Nickname = reader["Nickname"] as string;

user.ImageURL = reader["UserImage"] as string;

review.User = user;

review.Rate = Convert.ToInt32(reader["Rating"]);

review.Text = reader["Text"] as string;

DateTime uploadTime = Convert.ToDateTime(reader["UploadDate"].ToString());

review.UploadDate = uploadTime;

review.ShortUploadDate = uploadTime.ToShortDateString();

ReviewsCollection.Add(review);

}

}

}

return ReviewsCollection;

}

}

public static bool AddReview(int gameID, string reviewText, int rate)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

//MessageBox.Show(DateTime.Now.ToShortDateString());

string sqlExpression = $"insert into Reviews values(@userID, @gameID, @text, @dateNow, @rate)";

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.Add(new SqlParameter("@userID", Authorization.CurrentUser.ID));

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

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

command.Parameters.Add(new SqlParameter("@dateNow", DateTime.Now.ToShortDateString()));

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

try

{

var state = command.ExecuteNonQuery();

MessageBox.Show("Отзыв добавлен!", "Новый отзыв", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

}

catch (Exception ex)

{

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

AddReviewWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

return isSuccessful;

}

}

public static bool Update(int gameID, string text, int rating)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

User updatedUser = Authorization.CurrentUser;

bool isSuccessful = false;

connection.Open();

string sqlExpression = @$"update Reviews set UploadDate = @dateNow,

Text = @text,

Rating = @rating

where GameID = @gameID and UserID = @userID";

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.Add(new SqlParameter("@dateNow", DateTime.Now.ToShortDateString()));

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

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

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

command.Parameters.Add(new SqlParameter("@userID", updatedUser.ID));

try

{

var state = command.ExecuteNonQuery();

isSuccessful = true;

MessageBox.Show("Последний оставленный отзыв на игру был обновлен", "Обновление отзыва", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch (Exception ex)

{

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

AddReviewWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

return isSuccessful;

}

}

public static bool IsExists(int gameID)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

//MessageBox.Show(DateTime.Now.ToShortDateString());

string sqlExpression = $"select COUNT(ReviewID) from reviews where UserID = @userID and GameID = @gameID";

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.Add(new SqlParameter("@userID", Authorization.CurrentUser.ID));

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

try

{

var state = Convert.ToInt32(command.ExecuteScalar());

//MessageBox.Show("Отзыв добавлен!", "Новый отзыв", MessageBoxButton.OK, MessageBoxImage.Information);

//MessageBox.Show(Convert.ToString(state));

if(state > 0)

{

isSuccessful = true;

}

}

catch (Exception ex)

{

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

AddReviewWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

return isSuccessful;

}

}

public static double GetAverageRate(int gameid)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

double rate = 0;

connection.Open();

string sqlExpression = $"select AVG(Cast(rating as Float)) as rating from Reviews where GameID = @gameID";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

if(command.ExecuteScalar() is DBNull)

{

rate = -1.0;

}

else

{

var state = Convert.ToDouble(command.ExecuteScalar());

rate = state;

}

}

catch (Exception ex)

{

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

//AddReviewWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

if(rate > 0)

{

return Math.Round(rate, 1);

}

else

{

return -1.0;

}

}

}

}

}

Листинг 5: Game.cs

using CourseWorkAttempt.Auth;

using CourseWorkAttempt.Pages;

using CourseWorkAttempt.Windows;

using Microsoft.Data.SqlClient;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

namespace CourseWorkAttempt.Classes

{

public class Game

{

public int ID { get; set; }

public string? Name { get; set; }

public DateTime? ReleaseDate { get; set; }

public string? Description { get; set; }

public string? BuyURL { get; set; }

public Publisher? Publisher { get; set; }

public Developer? Developer { get; set; }

public string? Genre { get; set; }

public string? Platform { get; set; }

public string? ImageURL { get; set; }

public override string ToString()

{

return Name + " " + Genre + " " + Platform;

}

//public static List<Game> Collection = new List<Game>();

static Game()

{

}

public static List<ComboBoxItem> GetGenres()

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

connection.Open();

string sqlExpression = $"select \* from genres";

SqlCommand command = new SqlCommand(sqlExpression, connection);

List<ComboBoxItem> genreCollection = new();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

string genre = reader["GenreName"] as string;

ComboBoxItem item = new();

item.Content = genre;

genreCollection.Add(item);

}

}

}

return genreCollection;

}

}

public static List<ComboBoxItem> GetGenresWithAll()

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

connection.Open();

string sqlExpression = $"select \* from genres";

SqlCommand command = new SqlCommand(sqlExpression, connection);

List<ComboBoxItem> genreCollection = new();

ComboBoxItem fitem = new();

fitem.Content = "Все";

genreCollection.Add(fitem);

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

string genre = reader["GenreName"] as string;

ComboBoxItem item = new();

item.Content = genre;

genreCollection.Add(item);

}

}

}

return genreCollection;

}

}

public static bool AddGenre(string genre)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = @$"insert into Genres values(@genre)";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

var state = command.ExecuteNonQuery();

MessageBox.Show("Жанр добавлен!", "Новая жанр", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

if (AddGameWindow.link != null)

{

AddGameWindow.link.GenreBox.ItemsSource = GetGenres();

}

Games.GetPage().GameList.ItemsSource = null;

Games.GetPage().GameList.ItemsSource = GetList();

Games.GetPage().GenreBox.ItemsSource = GetGenresWithAll();

Games.GetPage().GenreBox.SelectedIndex = 0;

}

catch (Exception ex)

{

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

if (ex.Message.Contains("UNIQUE"))

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* Жанр уже находится в базе данных";

}

else

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

}

return isSuccessful;

}

}

public static bool DeleteGenre(string genre)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = @$"delete from Genres where GenreName = @genre";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

var state = command.ExecuteNonQuery();

MessageBox.Show("Жанр удален!", "Удаление жанра", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

if (AddGameWindow.link != null)

{

AddGameWindow.link.GenreBox.ItemsSource = GetGenres();

}

Games.GetPage().GameList.ItemsSource = null;

Games.GetPage().GameList.ItemsSource = GetList();

Games.GetPage().GenreBox.ItemsSource = GetGenresWithAll();

Games.GetPage().GenreBox.SelectedIndex = 0;

}

catch (Exception ex)

{

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

if (ex.Message.Contains("UNIQUE"))

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* Жанр уже находится в базе данных";

}

else

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

}

return isSuccessful;

}

}

public static bool AddPlatform(string platform)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = @$"insert into Platforms values(@platform)";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

var state = command.ExecuteNonQuery();

MessageBox.Show("Платформа добавлена!", "Новая платформа", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

if (AddGameWindow.link != null)

{

AddGameWindow.link.PlatformBox.ItemsSource = GetPlatforms();

}

}

catch (Exception ex)

{

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

if (ex.Message.Contains("UNIQUE"))

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* Платформа уже находится в базе данных";

}

else

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

}

return isSuccessful;

}

}

public static bool DeletePlatform(string platform)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = @$"delete from Platforms where PlatformName = @platform";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

var state = command.ExecuteNonQuery();

MessageBox.Show("Платформа удалена!", "Удаление платформы", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

if (AddGameWindow.link != null)

{

AddGameWindow.link.PlatformBox.ItemsSource = GetPlatforms();

}

Games.GetPage().GameList.ItemsSource = null;

Games.GetPage().GameList.ItemsSource = GetList();

}

catch (Exception ex)

{

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

if (ex.Message.Contains("UNIQUE"))

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* Платформа уже находится в базе данных";

}

else

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

}

return isSuccessful;

}

}

public static List<ComboBoxItem> GetPlatforms()

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

connection.Open();

string sqlExpression = $"select \* from platforms";

SqlCommand command = new SqlCommand(sqlExpression, connection);

List<ComboBoxItem> platformCollection = new();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

string platform = reader["PlatformName"] as string;

ComboBoxItem item = new();

item.Content = platform;

platformCollection.Add(item);

}

}

}

return platformCollection;

}

}

public static List<Game> GetList()

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

connection.Open();

string sqlExpression = @$"select \* from Games

inner join Developers on Games.DeveloperID = Developers.DeveloperID

inner join Publishers on Publishers.PublisherID = Games.PublisherID

inner join Genres on Genres.GenreID = Games.GenreID

inner join Platforms on Platforms.PlatformID = Games.PlatformID";

SqlCommand command = new SqlCommand(sqlExpression, connection);

List<Game> GamesCollection = new();

using (SqlDataReader reader = command.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

var game = new Game();

var publisher = new Publisher();

var developer = new Developer();

publisher.ID = (int)reader["PublisherID"];

publisher.Name = reader["PublisherName"] as string;

publisher.Country = reader["PublisherCountry"] as string;

developer.ID = (int)reader["DeveloperID"];

developer.Name = reader["DeveloperName"] as string;

developer.Country = reader["DeveloperCountry"] as string;

game.ID = (int)reader["GameID"];

game.Developer = developer;

game.Publisher = publisher;

game.Name = reader["GameName"] as string;

game.Description = reader["Description"] as string;

game.Genre = reader["GenreName"] as string;

game.ImageURL = reader["GameImage"] as string;

game.Platform = reader["PlatformName"] as string;

game.BuyURL = reader["BuyURL"] as string;

DateTime releaseDate = Convert.ToDateTime(reader["ReleaseDate"].ToString());

game.ReleaseDate = releaseDate;

GamesCollection.Add(game);

}

}

}

return GamesCollection;

}

}

public static bool AddGame(Game game)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = @$"insert into Games values(@name, @releaseDate,

@description, @buyURL, @publisherID, @developerID, @image, @genreID, @platformID)";

SqlCommand command = new SqlCommand(sqlExpression, connection);

command.Parameters.Add(new SqlParameter("@name", game.Name));

command.Parameters.Add(new SqlParameter("@releaseDate", game.ReleaseDate));

command.Parameters.Add(new SqlParameter("@description", game.Description));

command.Parameters.Add(new SqlParameter("@buyURL", game.BuyURL));

command.Parameters.Add(new SqlParameter("@publisherID", game.Publisher.ID));

command.Parameters.Add(new SqlParameter("@developerID", game.Developer.ID));

command.Parameters.Add(new SqlParameter("@image", game.ImageURL));

int platformID = -1, genreID = -1;

string platformSql = "select PlatformID, GenreID from Platforms, Genres where PlatformName = @platform and GenreName = @genre";

var platformCommand = new SqlCommand(platformSql, connection);

platformCommand.Parameters.Add(new SqlParameter("@platform", game.Platform));

platformCommand.Parameters.Add(new SqlParameter("@genre", game.Genre));

using (SqlDataReader reader = platformCommand.ExecuteReader())

{

if (reader.HasRows) // если есть данные

{

while (reader.Read()) // построчно считываем данные

{

platformID = (int)reader["PlatformID"];

genreID = (int)reader["GenreID"];

}

}

}

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

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

try

{

var state = command.ExecuteNonQuery();

MessageBox.Show("Игра добавлена!", "Новая игра", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

Games.GetPage().GameList.ItemsSource = null;

Games.GetPage().GameList.ItemsSource = GetList();

Games.GetPage().GenreBox.ItemsSource = GetGenresWithAll();

Games.GetPage().GenreBox.SelectedIndex = 0;

}

catch (Exception ex)

{

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

if (ex.Message.Contains("UNIQUE"))

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* Игра с таким названием уже находится в базе данных";

}

else

{

AddGameWindow.link.ErrorMessageBlock.Text = "\* " + ex.Message;

}

}

return isSuccessful;

}

}

public static bool Delete(int id)

{

using (SqlConnection connection = new SqlConnection(Authorization.connectionString))

{

bool isSuccessful = false;

connection.Open();

string sqlExpression = $"delete Games where GameID = @id";

SqlCommand command = new SqlCommand(sqlExpression, connection);

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

try

{

var state = command.ExecuteNonQuery();

MessageBox.Show("Игра удалена!", "Удаление", MessageBoxButton.OK, MessageBoxImage.Information);

isSuccessful = true;

Games.GetPage().GameList.ItemsSource = null;

Games.GetPage().GameList.ItemsSource = GetList();

}

catch (Exception ex)

{

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

}

return isSuccessful;

}

}

}

}