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 «Объектно-ориентированное проектирование и программирование»

Тема «Игровой помощник»

Исполнитель

студент 3 курса группы 7 Тышкевич Роман Антонович

(Ф.И.О.)

Руководитель работы преподаватель-стажер Якунович А.В.

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

Курсовой проект защищен с оценкой

Председатель

(подпись)

Минск 2023

**Содержание**

[ВВЕДЕНИЕ 4](#_Toc154138557)

[1. Аналитический обзор литературы и постановка задачи 5](#_Toc154138558)

[1.1 Обзор прототипов 5](#_Toc154138559)

[1.1.1 Приложение GamerHub 5](#_Toc154138560)

[1.1.2 GameTracker Pro 6](#_Toc154138561)

[1.1.3 GamingPal 6](#_Toc154138562)

[1.1.4 QuestMaster 7](#_Toc154138563)

[1.2 Постановка задачи 8](#_Toc154138564)

[2. Анализ требований к программному средству и разработка функциональных требований 9](#_Toc154138565)

[2.1 Описание средств разработки 9](#_Toc154138566)

[2.2 Описание разрабатываемой функциональности программного средства 10](#_Toc154138567)

[3. Проектирование программного средства 11](#_Toc154138568)

[3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства 11](#_Toc154138569)

[3.2 Разработка общей схемы приложения 12](#_Toc154138570)

[3.3 Проектирование архитектуры приложения 13](#_Toc154138571)

[3.4 Проектирование базы данных 15](#_Toc154138572)

[4. Создание (реализация) программного средства 17](#_Toc154138573)

[4.1 Диаграмма классов 17](#_Toc154138574)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 17](#_Toc154138575)

[6. Руководство по использованию 20](#_Toc154138576)

[ЗАКЛЮЧЕНИЕ 24](#_Toc154138577)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 25](#_Toc154138578)

[Приложение А 26](#_Toc154138579)

[Приложение Б 28](#_Toc154138580)

[Графический материал 31](#_Toc154138581)

[Диаграмма вариантов использования 31](#_Toc154138582)

[Логическая схема базы данных 32](#_Toc154138583)

[Диаграммы классов 33](#_Toc154138584)

[Диаграмма последовательностей 34](#_Toc154138585)

[34](#_Toc154138586)

ВВЕДЕНИЕ

Game Helper представляет инновационное решение в сфере организации и повышения продуктивности, сфокусированное на потребностях геймеров. В современном игровом мире актуальным становится направление самоорганизации, и Game Helper является ответом на этот вызов.

В последние годы наблюдается рост интереса к приложениям, направленным на увеличение эффективности и управление игровой жизнью. Такие приложения становятся популярными в магазинах приложений, таких как PlayMarket и AppStore. Очевиден тренд среди современных геймеров, стремящихся контролировать и анализировать свой игровой процесс.

Подобно тому, как организованность связана с производительностью в повседневной жизни, Game Helper призван улучшить игровой опыт, предоставляя инструменты для более эффективного управления игровыми сессиями. Геймеры часто сталкиваются с проблемой отслеживания игрового прогресса и выполнения задач, и Game Helper призван решить эти трудности.

Game Helper предоставляет две ключевые роли: Пользователь и Администратор. Геймер может мониторить свои достижения, отмечать выполненные задания, а также взаимодействовать с другими пользователями. Администратор, в свою очередь, обеспечивает административное управление, добавляя новые игры, контролируя доступ к функциям и следя за общим игровым пространством.

Таким образом, Game Helper создан для того, чтобы сделать игровой опыт более структурированным и удовлетворительным, предоставляя инструменты для эффективного управления всеми аспектами игровой деятельности.

1. Аналитический обзор литературы и постановка задачи

Для того чтобы окончательно определиться с постановкой задачи моего курсового проекта, необходимо проанализировать прототипы программных средств выбранной темы.

* 1. Обзор прототипов

1.1.1 Приложение GamerHub

GamerHub - это приложение, сфокусированное на улучшении организации и взаимодействии геймеров. Пользователи могут отслеживать свои достижения в различных играх, обмениваться опытом и рекомендациями. Основные функции включают в себя создание персональных списков задач, управление игровыми сессиями и ведение статистики по прохождению уровней. Важным элементом является возможность общения внутри приложения, что создает игровое сообщество.

Интерфейс приложения показан на рисунке 1.1.

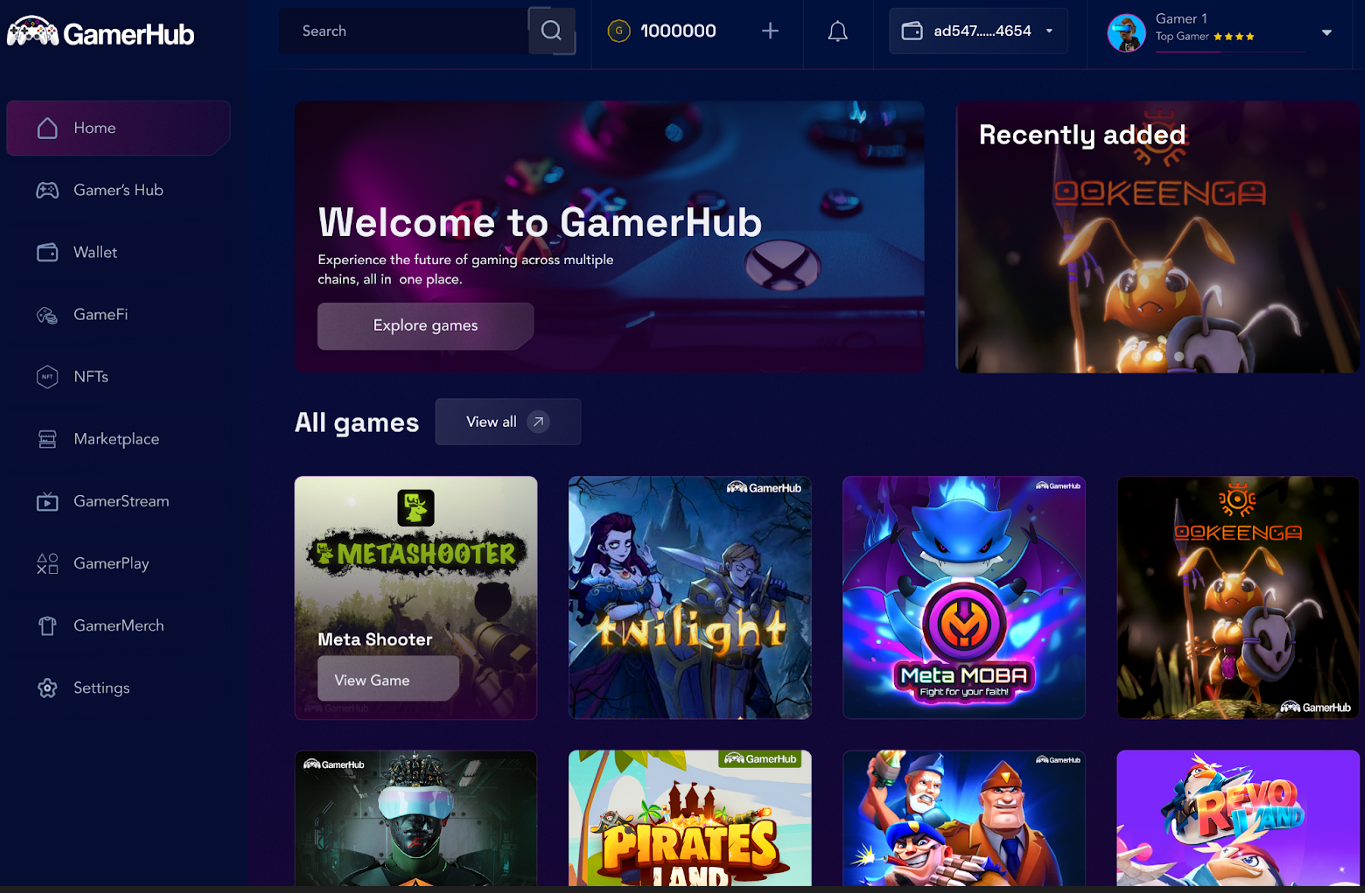


Рисунок 1.1 – Интерфейс приложения GamerHub

1.1.2 GameTracker Pro

GameTracker Pro предоставляет геймерам мощный инструмент для отслеживания игрового прогресса и достижений. Приложение позволяет пользователям управлять списками задач, создавать заметки и устанавливать цели для каждой игры. Одной из ключевых функций является возможность автоматического отслеживания времени игры и статистики по прохождению уровней. GameTracker Pro также предоставляет возможность сравнения своих результатов с другими геймерами.

Интерфейс представлен на рисунке 1.2.

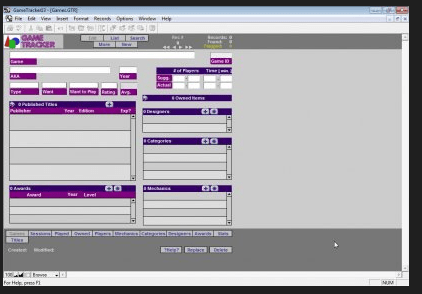


Рисунок 1.2 – Интерфейс приложения GameTracker Pro

1.1.3 GamingPal

GamingPal - это социальная платформа для геймеров, предлагающая функции организации и взаимодействия. Пользователи могут создавать группы для совместной игры, планировать события и обмениваться опытом. Приложение также включает в себя инструменты для ведения личного игрового дневника, в котором можно отслеживать пройденные уровни, полученные награды и составленные стратегии. Геймеры могут оценивать другие профили и создавать соревнования. Интерфейс представлен на рисунке 1.3.



Рисунок 1.3 – Интерфейс GamingPal

1.1.4 QuestMaster

QuestMaster - это приложение, разработанное специально для любителей игр с элементами квестов. Пользователи могут создавать собственные квесты, делиться ими с другими пользователями и присоединяться к квестам, созданным сообществом. Основные функции включают в себя управление текущими заданиями, отслеживание прогресса и возможность получения наград за выполнение квестов. QuestMaster также предоставляет возможность общения внутри приложения и обсуждения игровых событий. Интерфейс представлен на рисунке 1.4.



Рисунок 1.4 – Интерфейс QuestMaster

1.2 Постановка задачи

Проанализировав концепцию, выделены основные задачи и функциональные требования для моего проекта "Игровой Помощник".

Целью данного проекта является создание инновационного инструмента, предназначенного для оптимизации игрового процесса и обеспечения удобства геймеров. Игровой помощник будет включать в себя информацию о персонажах, мирах, заданиях, а также предоставлять возможность отслеживать игровое время и важные события.

Пользователи системы смогут просматривать данные по различным критериям, таким как типы заданий, локации и даты. Также будет предоставлена функция поиска конкретных элементов игры.

Администратор системы получит возможность управлять учетными записями игроков и модерировать игровое пространство.

Игрок сможет вносить изменения в свой профиль, отмечать выполненные задания, а также управлять информацией о своих достижениях. Предусмотрена автоматическая система уведомлений об обновлениях и событиях в игре.

Возможность регистрации в системе будет предоставлена игрокам, после чего они смогут персонализировать свой опыт, отслеживать свой игровой прогресс и делиться им с друзьями. Дополнительно будет реализована функция создания заметок и комментариев по игровым событиям. Функциональные требования описаны в главе 2.

2. Анализ требований к программному средству и разработка функциональных требований

Для начала этапа разработки необходимо сформулировать функциональные требования к программному средству.

2.1 Описание средств разработки

Языком разработки курсового проекта является C#. При выполнении курсового проекта будут использованы принципы и приемы ООП. Также будут использоваться технологии Windows Presentation Foundation (WPF). Программный продукт реализован на объектно-ориентированном языке C# с использованием технологий WPF.

Windows Presentation Foundation (WPF) — система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от WindowsForms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

База данных была разработана на SQL Server в приложении DataGrip.

SQL Server является одной из наиболее популярных систем управления базами данных (СУБД) в мире.

SQL Server характеризуется такими особенностями как:

* Производительность. SQL Server работает очень быстро.
* Надежность и безопасность. SQL Server предоставляет шифрование данных.
* Простота. С данной СУБД относительно легко работать и вести администрирование.

Для организации баз данных MS SQL Server использует реляционную модель, которая предполагает хранение данных в виде таблиц, каждая из которых состоит из строк и столбцов. Каждая строка хранит отдельный объект, а в столбцах размещаются атрибуты этого объекта.

Для взаимодействия с базой данных применяется язык SQL (Structured Query Language).

Клиент (например, внешняя программа) отправляет запрос на языке SQL должным образом интерпретирует и выполняет запрос, а затем посылает клиенту результат выполнения.

Основной используемый язык запросов – Transact-SQL – реализован на структурированном языке запросов (SQL) с расширениями.

Для осуществления связи между базой данных и приложением на C# необходим посредник, которым будет являться технология Entity Framework. Она основана на платформе .NET Framework и предназначена для работы с данными.

Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Эта технология предоставляет нам набор классов, через которые мы можем отправлять запросы к базам данных, устанавливать подключения, получать ответ от базы данных и производить ряд других операций.

Entity Framework предполагает три возможных способа взаимодействия с базой данных:

* database first: Entity Framework создает набор классов, которые отражают модель конкретной базы данных;
* model first: сначала разработчик создает модель базы данных, по которой затем Entity Framework создает реальную базу данных на сервере;
* code first: разработчик создает класс модели данных, которые будут храниться в базе данных, а затем Entity Framework по этой модели генерирует базу данных и ее таблицы.

В курсовом проектировании для разработки был выбран первый подход.

2.2 Описание разрабатываемой функциональности программного средства

В программном средстве присутствуют 2 роли: пользователь, администратор.

У роли пользователя следующие функциональные возможности:

* авторизация в приложении;
* просмотр игр, имеющихся в библиотеке пользователя;
* просмотр всех игр, имеющихся в базе данных;
* оставлять пометки и записки;
* добавлять свои собственные данные.

У роли администратора следующие функциональные возможности:

* изменять общую библиотеку игр;
* просматривать список пользователей;
* добавлять игры в базу данных;

По итогу работы над продуктом, его необходимо протестировать и отладить, проанализировать его возможности и, по необходимости, внести дополнения или улучшения проекта, с возможностью дальнейшего расширения проекта.

В графической части записки представлена UML-диаграмма возможностей, которая отражает функциональность программного средства с точки зрения получения значимого результата для пользователя.

3. Проектирование программного средства

Проектирование программного средства – процесс создания проекта программного обеспечения. Целью проектирования является определение внутренних свойств системы и детализации её внешних свойств на основе исходных условий задачи. Исходные условия задачи были сформулированы во втором разделе данной пояснительной записки. Этап проектирования подразумевает их анализ.

3.1 Обобщенная структура проекта и детализация функций проектируемого программного средства

В данном приложении используется следующая структура:

* Модель описывает используемые в приложении данные.
* Представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением.
* Модель представления связывает модель и представление через механизм привязки данных. ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление.

Структура проекта представлена на рисунке 2.2.1.

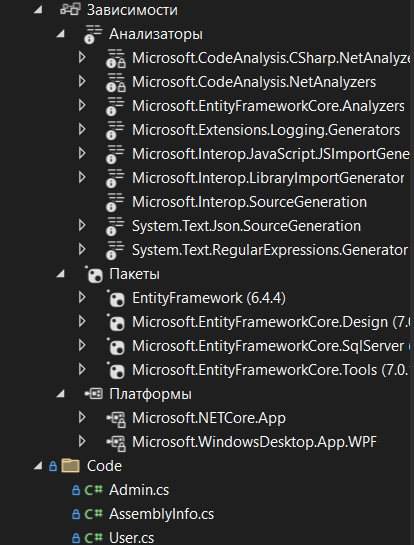
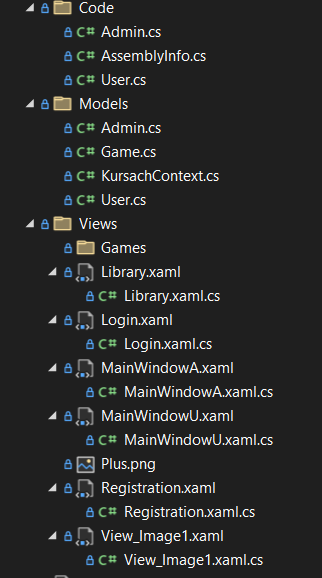
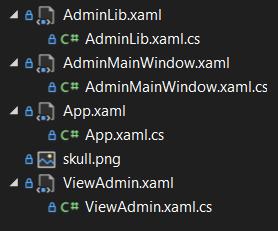
  

Рисунок 2.2.1 – Структура проекта

Описание структуры проекта представлено в таблице 2.1.

Таблица 2.1 – Структура проекта «Game Helper»

|  |  |
| --- | --- |
| Имя файла | Содержание |
| Models | Подпространство имён, содержащее описание сущностей для работы с БД. |
| View | Представления, которые определяют визуальный интерфейс, через который пользователь взаимодействует с приложением |
| ViewModel | Содержит логику по получению данных из модели, которые затем передаются в представление. |
| App.config | Файл с параметрами проекта. |

Таким образом, сформированная таблица помогает понять общую структуру проекта проектируемого программного средства.

3.2 Разработка общей схемы приложения

Общая схема окон приложения пользователя показана на рисунке 3.2.1.

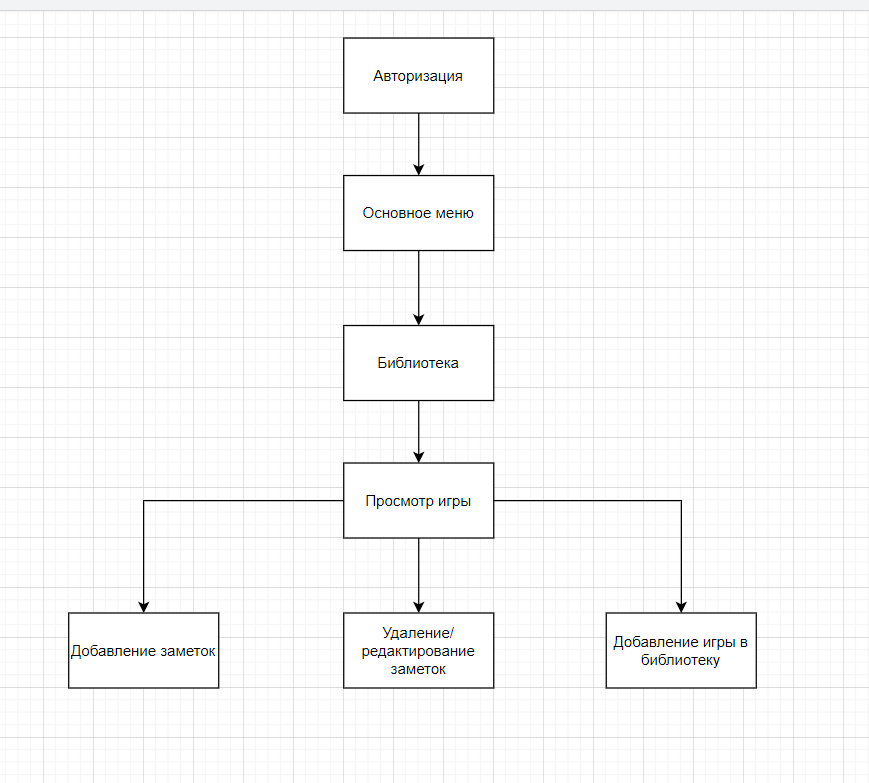


Рисунок 3.2.1 – Схема окон пользователя

Общая схема окон приложения для администратора представлена на рисунке 3.2.2.

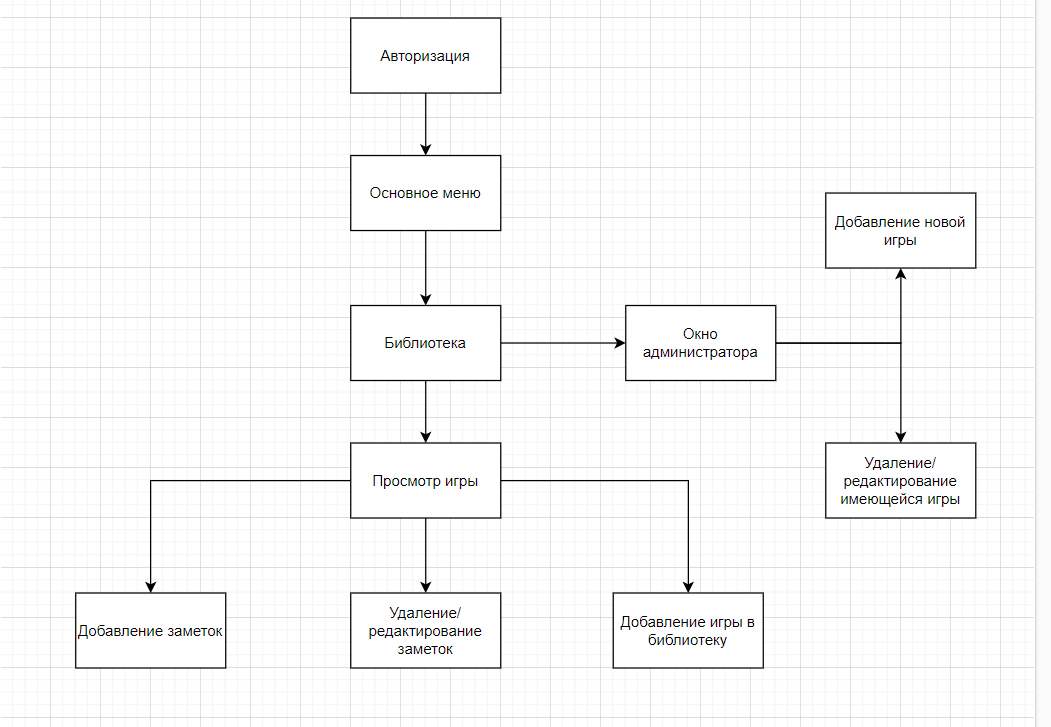


Рисунок 3.2.2 – Схема окон администратора

При запуске приложения открывается окно авторизации.

После успешной авторизации запускается главное окно приложения, где происходит работа с отдельными страницами.

3.3 Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность вaжнейших решений об организации программной системы. Архитектура включает:

* выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;
* соединение выбрaнных элементов структуры и поведения во всё более крупные системы;
* архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей.

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Она содержит Модель, преобразованную к Представлению, а также команды, которыми может пользоваться Представление, чтобы влиять на Модель.

Нередко все сущности (модель) наследует интерфейс INotifyPropertyChanged или INotifyCollectionChanged, которые позволяют отлавливать изменения и правильно на их реагировать и сохранять.

Для того, чтобы реализовать интерфейс INotifyPropertyChanged, нужно объявить свойство PropertyChanged и метод OnPropertyChanged()[5].

Для того, чтобы начать отслеживание свойства, необходимо вызывать метод OnPropertyChanged(“Имя свойства”).

Model или Модель представляет собой логику работы с данными и описание фундаментальных данных, необходимых для работы приложения.

На диаграмме размещения, представленной на рисунке 3.3.2 показано графическое представление инфраструктуры приложения.

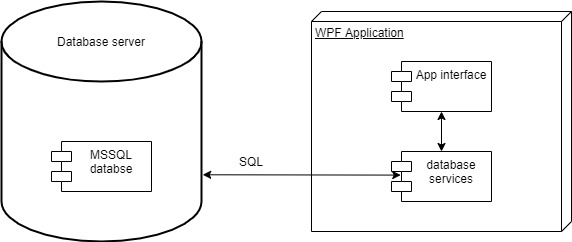


Рисунок 3.3.2 – Диаграмма размещения

Для разработки приложения в качестве сервиса для хранения данных была выбрана СУБД Microsoft SQL Server.

3.4 Проектирование базы данных

Проектирование баз данных – процесс создания схемы базы данных и определения необходимы ограничений целостности.

При проектировании базы данных курсового проекта использовался Entity Framework. Он представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища.

Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работаем с объектами.

Отличительной чертой Entity Framework является использование запросов LINQ для выборки данных из БД. С помощью LINQ мы можем не только извлекать определенные строки, хранящие объекты, из базы данных, но и получать объекты, связанные различными ассоциативными связями.

Логическая модель базы данных представлена в графической части. Для реализации, поставленной в курсовом проектировании задачи была создана база данных Game Helper. Для её создания использовалась система управления реляционными базами данных Microsoft SQL Server 2018[3].

Созданная база данных содержит в себе 3 таблицы: Users, Admins, Games. На рисунке 3.4.1 представлена структура таблицы Users.

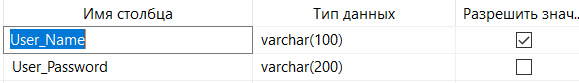


Рисунок 3.4.1 – Структура таблицы Users

В данной таблице хранится логин, пароль.

Всю информацию о администраторе можно посмотреть в таблице Admins, структура которой представлена на рисунке 3.4.3.

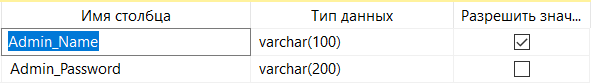


Рисунок 3.4.3 – Структура таблицы Admins

Для игр есть таблица Games, показанная на рисунке 3.4.5.

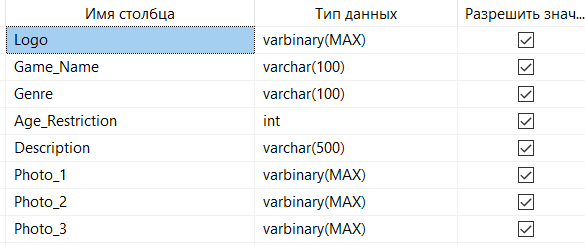


Рисунок 3.4.5 – Структура таблицы Games

4. Создание (реализация) программного средства

Следующим этапом разработки приложения является непосредственная реализация программного решения в соответствии с уже сформированными требованиями и шаблонами.

4.1 Диаграмма классов

Структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На данной диаграмме классов изображены классы программы, методы, поля и свойства классов. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов). Диаграмма классов проекта   
 представлена в графической части.

5. Тестирование, проверка работоспособности и анализ полученных результатов

Для обеспечения корректности работы программы обрабатываются различные ошибки, возникающие в процессе работы. Данное программное средство использует подключение к базе данных, следовательно, неправильно введенные данные или же их отсутствие может повлечь за собой неработоспособность приложения.

При тестировании данного приложения, были применены сценарии, которые могли бы привести к ошибке. В этой главе мы рассмотрим некоторые такие сценарии и посмотрим на их обработку.

В момент регистрации, возможна такая ситуация, в которой пользователь ничего не ввёл или ввёл неверные данные. Обработка данного сценария приведена на рисунке 5.1.

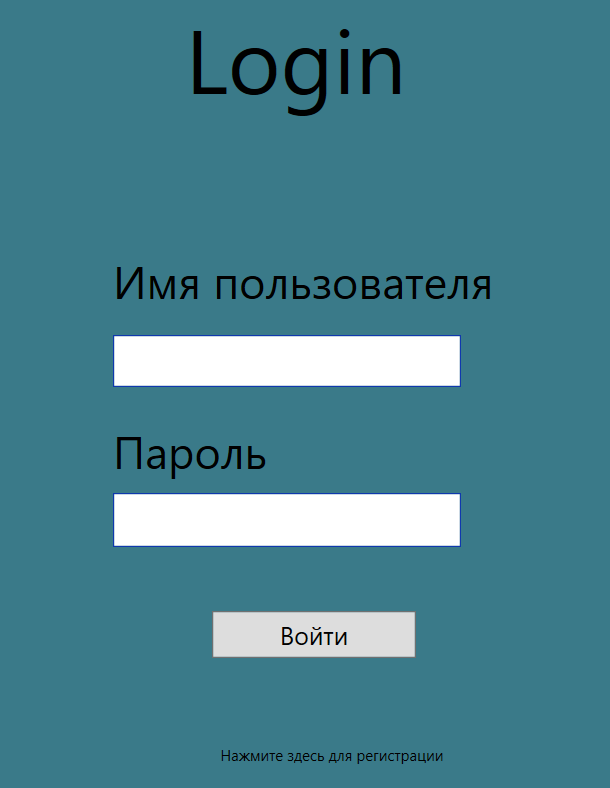


Рисунок 5.1 – Любое поле оставлено пустым

Кнопка Войти не активна до момента ввода обоих полей.

Сценарий, при котором введён логин, и пароль но данной комбинации нет в базе данных, приведен на рисунке 5.2.

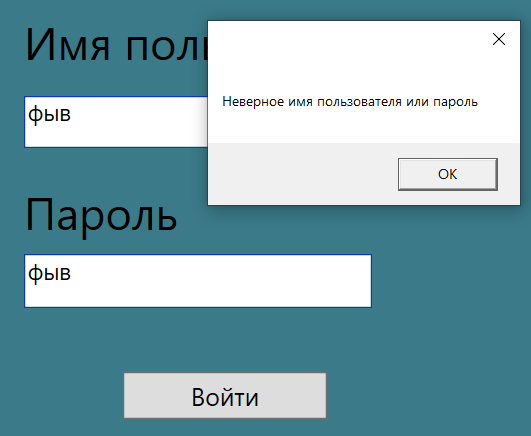
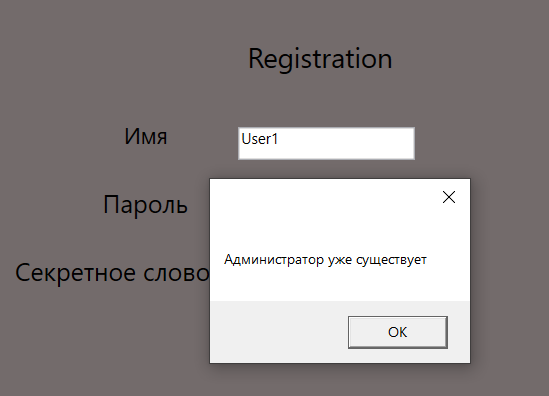


Рисунок 5.2 – Несуществующий пользователь

Сценарий, при котором администратор пытается добавить пользователя с уже существующим логином представлен на рисунке 5.3.



Все поля, предназначенные для ввода данных, проходят валидацию в соответствии с тем типом данных, для которых данное поле предназначено.

6. Руководство по использованию

При запуске приложения у вас появится окно авторизации, где вы выбрать либо зарегистрировать нового пользователя, либо совершить вход в существующий аккаунт. Для смены регистрации и входа существует отдельная кнопка.

Окно авторизации представлено на рисунке 6.1.

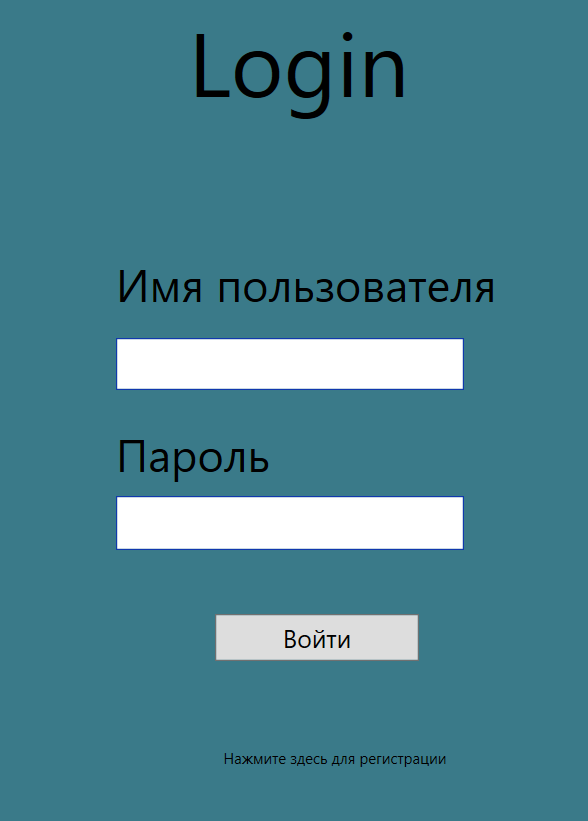


Рисунок 6.1 – Окно авторизации

После успешного входа под вашей учётной записью у вас появится главное меню, которое изображено на рисунке 6.2.



Рисунок 6.2 – Главное меню приложения

В нем изображены кнопки, ведущие на все страницы приложения.. Данное вспомогательное меню представлено на каждой странице приложения.

Рассмотрим подробнее каждую из страниц.

На рисунке 6.3 представлено окно библиотека, где можно просмотреть все имеющиеся игры.

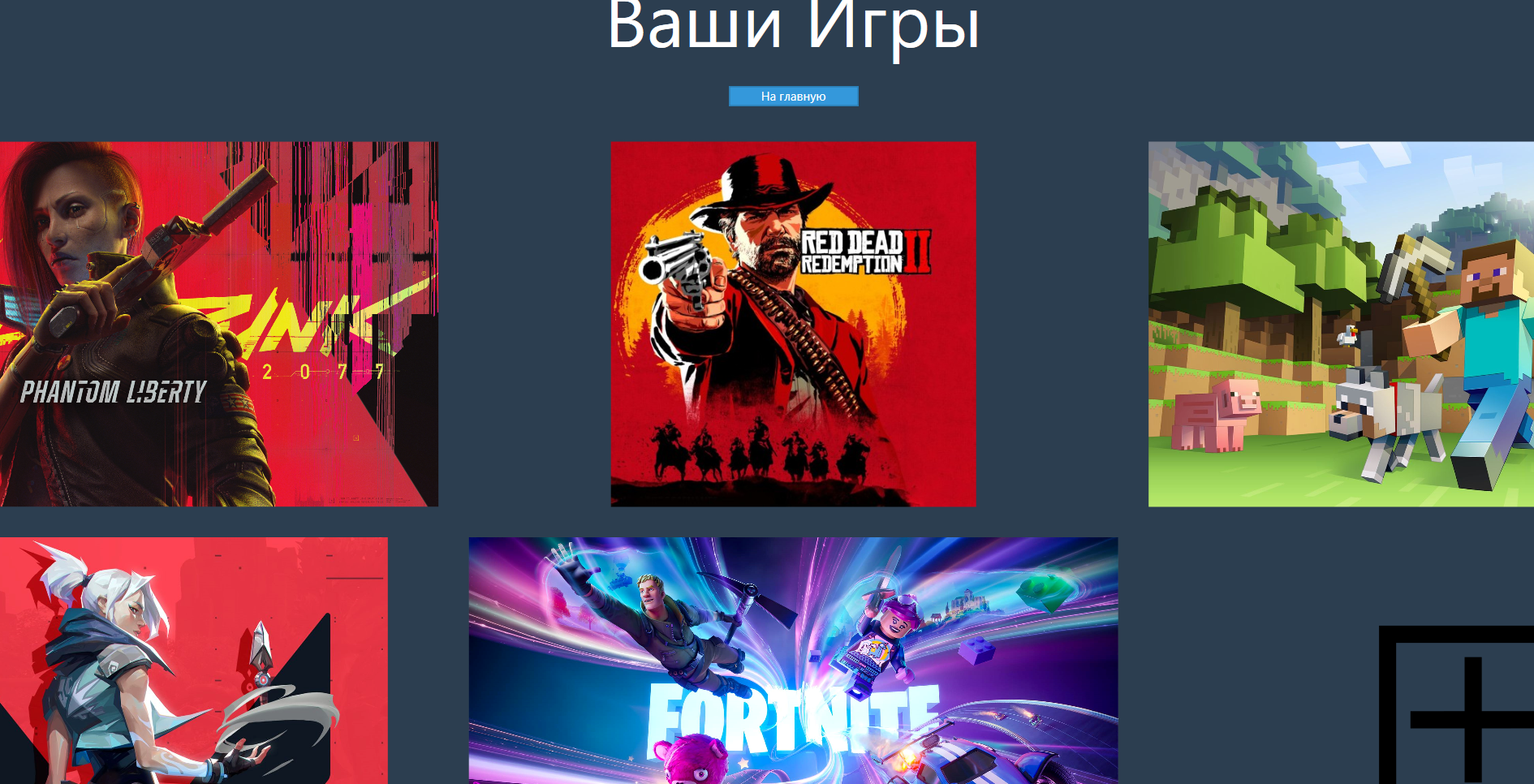


Рисунок 6.3 – Окно библиотеки

На рисунке 6.4 представлено окно заметок.

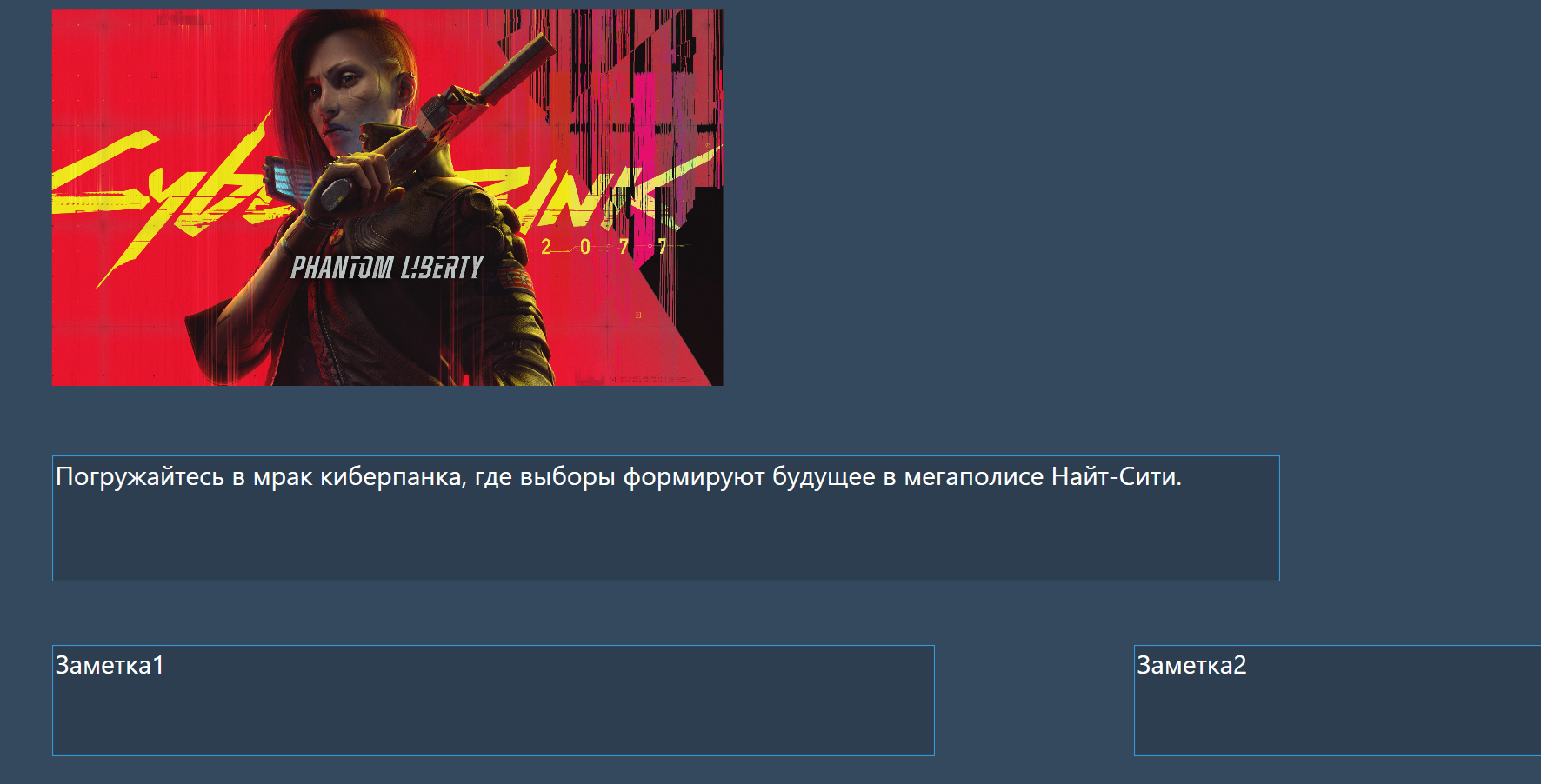
****

Рисунок 6.4 – Окно заметок

На рисунке 6.5 представлено окно добавление новой игры.

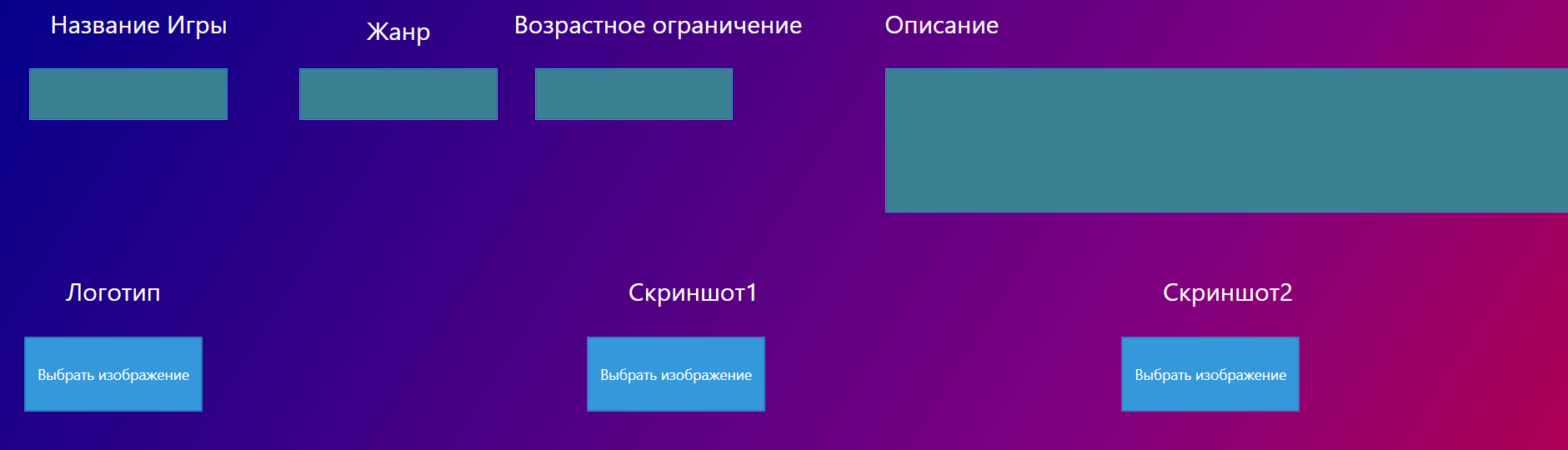


Рисунок 6.5–Окно добавление игры

ЗАКЛЮЧЕНИЕ

В результате выполнения данной курсовой работы было разработано программное средство «Game Helper», которое помогает игрокам в играх.

Разработанное программное средство предоставляет пользователю следующие функциональные возможности:

* регистрация пользователей;
* авторизация пользователей;
* создание и просмотр игр;
* создание и просмотр оценок;
* хранение и отображение прогресса по играм;
* хранение личной информации в базе данных.

Разработанное программное средство реагирует на ошибочный ввод данных выводя при этом соответствующее сообщение об ошибке.

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

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Пацей, Н.В. Курс лекций по языку программирования С# / Н.В. Пацей. – Минск: БГТУ, 2021. – 175 с.
2. Пацей, Н.В. Технология разработки программного обеспечения / Н.В. Пацей. – Минск: БГТУ, 2016. – 129 с.
3. Документация SQLServer [Электронный ресурс] / Foundation, Inc. https://docs.sqlserver.com/cd/B28359\_01/server.111/b31222/toc.htm
4. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: https://metanit.com
5. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: https://professorweb.ru
6. StackOverflow [Электронный ресурс] / Режим доступа: <https://stackoverflow.com>
7. Хабр [Электронный ресурс] / Режим доступа: <https://habr.com>.

Приложение А

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity;

using System.Data.SqlClient;

using System.Windows;

namespace Test

{

public class User

{

//Server=NEWTRPC;User ID=Newtr;Password=;Database=Kursach

string connectionString = "Server=NEWTRPC;User ID=Newtr;Password=;Database=Kursach;Trusted\_Connection=True;TrustServerCertificate=true;";

public void InsertUser(string userName, string userPassword)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

// Проверьте, существует ли уже пользователь с таким именем

using (SqlCommand checkUserCmd = new SqlCommand("SELECT COUNT(\*) FROM Users WHERE User\_Name = @UserName", conn))

{

checkUserCmd.Parameters.Add(new SqlParameter("@UserName", userName));

int userExists = (int)checkUserCmd.ExecuteScalar();

if (userExists > 0)

{

MessageBox.Show("Пользователь с таким именем уже существует.");

return; // Возвращаемся из метода, не выполняя код ниже

}

}

// Если пользователя с таким именем нет, вставьте нового пользователя

using (SqlCommand cmd = new SqlCommand("INSERT INTO Users (User\_Name, User\_Password) VALUES (@UserName, @UserPassword)", conn))

{

cmd.Parameters.Add(new SqlParameter("@UserName", userName));

cmd.Parameters.Add(new SqlParameter("@UserPassword", userPassword));

cmd.ExecuteNonQuery();

}

}

}

public bool CheckUser(string userName, string userPassword)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

// Проверьте, существует ли пользователь с таким именем и паролем

using (SqlCommand checkUserCmd = new SqlCommand("SELECT COUNT(\*) FROM Users WHERE User\_Name = @UserName AND User\_Password = @UserPassword", conn))

{

checkUserCmd.Parameters.Add(new SqlParameter("@UserName", userName));

checkUserCmd.Parameters.Add(new SqlParameter("@UserPassword", userPassword));

int userExists = (int)checkUserCmd.ExecuteScalar();

if (userExists > 0)

{

// Пользователь с таким именем и паролем существует

return true;

}

else

{

// Пользователь с таким именем и паролем не существует

return false;

}

}

}

}

}

}

Листинг – Класс User

Приложение Б

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.Entity;

using System.Data.SqlClient;

using System.Windows;

namespace Test

{

public class Admin

{

//Server=NEWTRPC;User ID=Newtr;Password=;Database=Kursach

string connectionString = "Server=NEWTRPC;User ID=Newtr;Password=;Database=Kursach;Trusted\_Connection=True;TrustServerCertificate=true;";

public void InsertAdmin(string adminName, string adminPassword)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

// Проверьте, существует ли уже пользователь с таким именем

using (SqlCommand checkUserCmd = new SqlCommand("SELECT COUNT(\*) FROM Admins WHERE Admin\_Name = @AdminName", conn))

{

checkUserCmd.Parameters.Add(new SqlParameter("@AdminName", adminName));

int userExists = (int)checkUserCmd.ExecuteScalar();

if (userExists > 0)

{

MessageBox.Show("Администратор с таким именем уже существует.");

return; // Возвращаемся из метода, не выполняя код ниже

}

}

// Если админ с таким именем нет, вставьте нового админа

using (SqlCommand cmd = new SqlCommand("INSERT INTO Admins (Admin\_Name, Admin\_Password) VALUES (@AdminName, @AdminPassword)", conn))

{

cmd.Parameters.Add(new SqlParameter("@AdminName", adminName));

cmd.Parameters.Add(new SqlParameter("@AdminPassword", adminPassword));

cmd.ExecuteNonQuery();

}

}

}

public bool CheckAdmin(string adminName, string adminPassword)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

// Проверьте, существует ли пользователь с таким именем и паролем

using (SqlCommand checkUserCmd = new SqlCommand("SELECT COUNT(\*) FROM Admins WHERE Admin\_Name = @AdminName AND Admin\_Password = @AdminPassword", conn))

{

checkUserCmd.Parameters.Add(new SqlParameter("@AdminName", adminName));

checkUserCmd.Parameters.Add(new SqlParameter("@AdminPassword", adminPassword));

int userExists = (int)checkUserCmd.ExecuteScalar();

if (userExists > 0)

{

// Пользователь с таким именем и паролем существует

return true;

}

else

{

// Пользователь с таким именем и паролем не существует

return false;

}

}

}

}

public void InsertGame(byte[] logo, string gameName, string genre, int ageRestriction, string description, byte[] photo1, byte[] photo2, byte[] photo3)

{

using (SqlConnection conn = new SqlConnection(connectionString))

{

conn.Open();

using (SqlCommand insertGameCmd = new SqlCommand("INSERT INTO Games (Logo, Game\_Name, Genre, Age\_Restriction, Description, Photo\_1, Photo\_2, Photo\_3) VALUES (@Logo, @GameName, @Genre, @AgeRestriction, @Description, @Photo1, @Photo2, @Photo3)", conn))

{

insertGameCmd.Parameters.Add(new SqlParameter("@Logo", logo));

insertGameCmd.Parameters.Add(new SqlParameter("@GameName", gameName));

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

insertGameCmd.Parameters.Add(new SqlParameter("@AgeRestriction", ageRestriction));

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

insertGameCmd.Parameters.Add(new SqlParameter("@Photo1", photo1));

insertGameCmd.Parameters.Add(new SqlParameter("@Photo2", photo2));

insertGameCmd.Parameters.Add(new SqlParameter("@Photo3", photo3));

insertGameCmd.ExecuteNonQuery();

}

}

MessageBox.Show("Да");

}

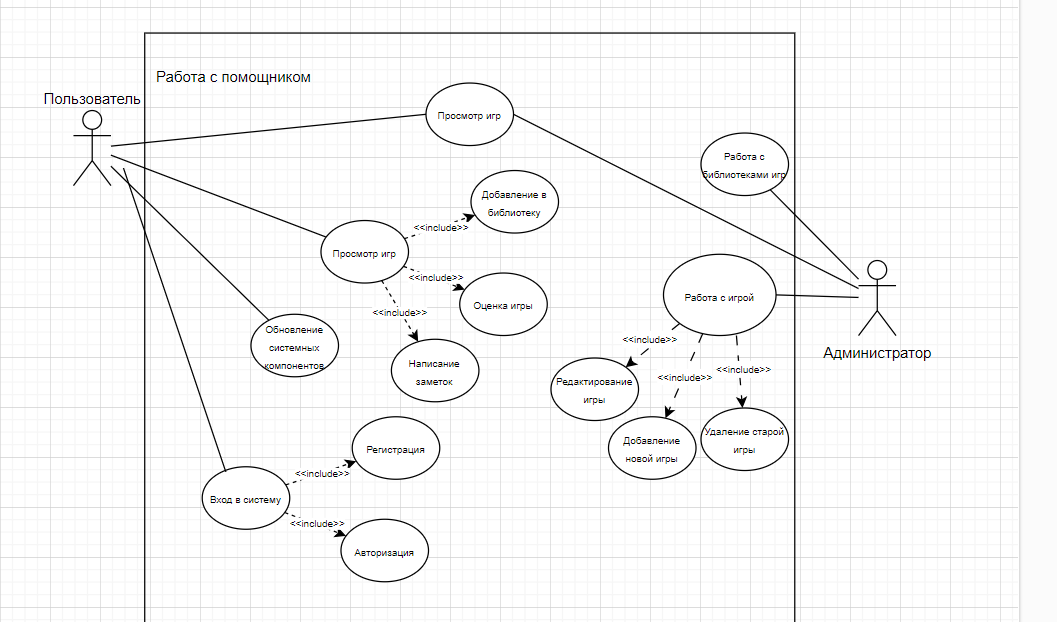
}

}

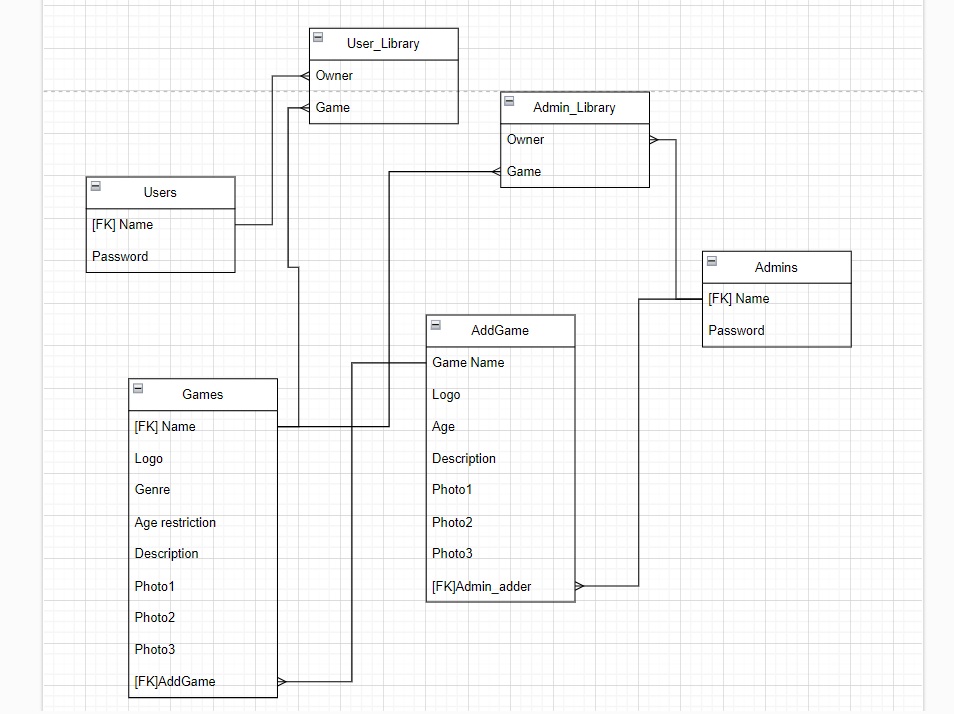
Листинг – Класс Admin

Графический материал

Диаграмма вариантов использования



Логическая схема базы данных



Диаграммы классов

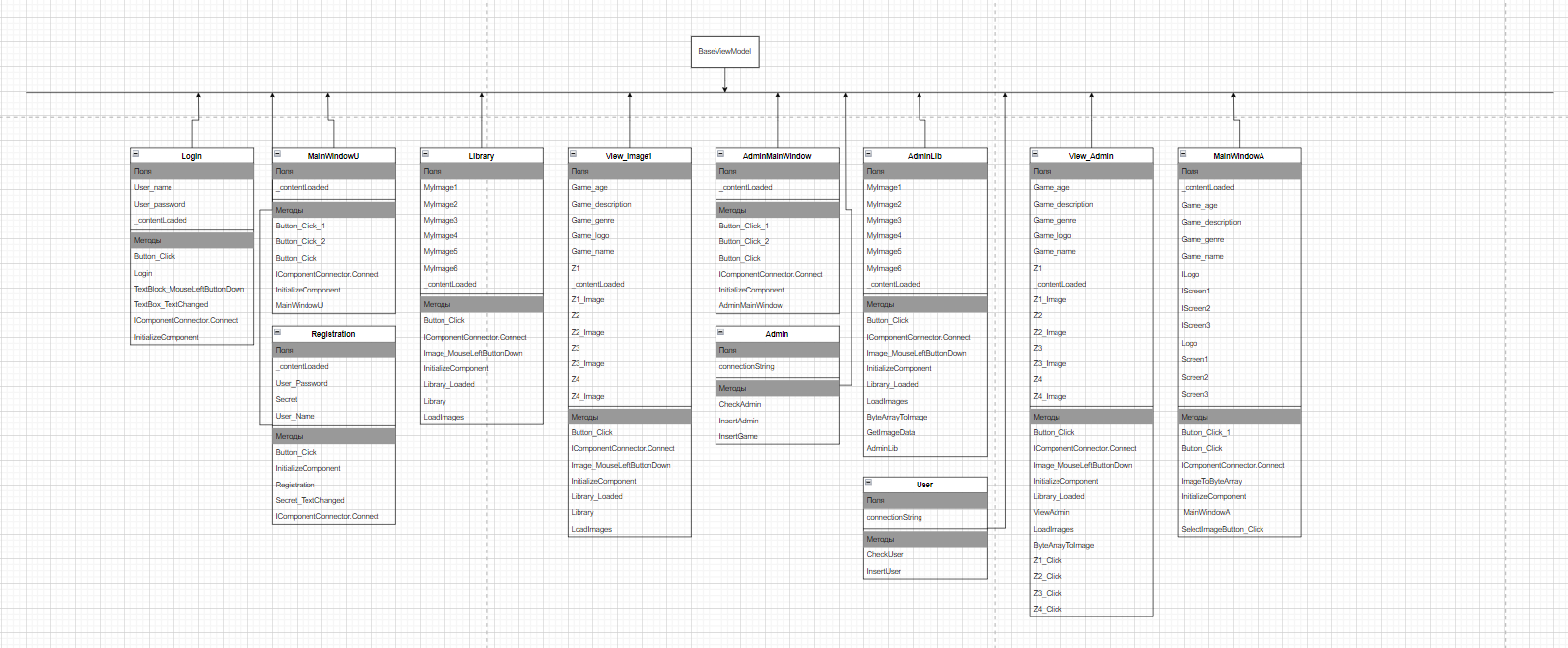


Диаграмма последовательностей

