Федеральное государственное образовательное бюджетное учреждение

высшего образования

**«Финансовый университет при Правительстве**

**Российской Федерации»**

**(Финансовый университет)**

Колледж информатики и программирования

(наименование структурного подразделения)

Дипломный проект

Тема « Разработка развлекательного мультиплатформенного игрового сервиса »

(наименование)

Студент Мариничев Андрей Романович (фамилия, имя, отчество полностью)

Учебная группа 4ИСИП-619

Специальность 09.02.07 Информационные системы и программирование

(код и наименование специальности)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Руководитель  дипломного проекта |  |  |  | И.В. Сибирев |
|  |  | (подпись) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Консультант  дипломного проекта |  |  |  |  |
| (при наличии) |  | (подпись) |  | (инициалы, фамилия) |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Председатель предметной  (цикловой) комиссии |  |  |  | Н.Г. Титов |
|  |  | (подпись) |  | (инициалы, фамилия) |

Москва – 2023 г.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc31460)

[ГЛАВА 1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ 4](#_Toc9783)

[1.1 Описание предметной области 4](#_Toc8457)

[1.2 Сравнительный анализ программ аналогов 4](#_Toc32541)

[1.3 Постановка задачи 8](#_Toc27250)

[1.4 Характеристика инструментальных средств разработки 9](#_Toc1076)

[1.5 Обоснование выбора инструментальных средств разработки 12](#_Toc7776)

[ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ 14](#_Toc26994)

[2.1 Анализ требований и определение спецификаций 14](#_Toc2984)

[2.2 Проектирование программного обеспечения 17](#_Toc27183)

[2.3 Разработка программного обеспечения 19](#_Toc20689)

[2.4 Отладка и тестирование программы 38](#_Toc24890)

[2.5 Руководство по использованию программы 39](#_Toc16965)

[ЗАКЛЮЧЕНИЕ 42](#_Toc2753)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 43](#_Toc18211)

[ПРИЛОЖЕНИЕ 1 46](#_Toc1647)

[ПРИЛОЖЕНИЕ 2 54](#_Toc2406)

# ВВЕДЕНИЕ

В наше время люди часто устают на работе. После рабочего дня они хотят эмоционально и физически расслабиться. Для того, чтобы помочь людям отдохнуть после выполнения всех дел, было решено сделать развлекательный мультиплатформенный игровой сервис. В приложении предполагаются простые игры, которые будут нести развлекательную функцию. Результат будет единым на всех платформах. В соревновательных целях будет добавлена таблица с наивысшими результатами среди всех зарегистрированных пользователей. Свой результат пользователь будет показывать только при собственном желании, нажимая на кнопку.

Одна из игр будет представлять собой уже ставшую классикой змейку. По мере поедания яблок, появляющихся на игровом поле, размер змейки, которой управляет пользователь, будет увеличиваться. При столкновении головы змейки со своим телом игра будет заканчиваться.

Ещё одна игра – «Крестики-нолики». Правила простые, игрок выбирает сторону, крестики или нолики, после чего по очереди с компьютером ставят либо крестик, либо нолик. Победа присуждается стороне, которая смогла сделать ряд из своих символов.

Цель проекта – разработка десктопной версии развлекательного мультиплатформенного игрового сервиса, имеющего соревновательные элементы.

Для того, чтобы сохранять прогресс на всех платформах и выводить список лучших игроков планируется использовать удалённую базу данных. Для того, чтобы обеспечить доступ к базе данных, будет использоваться API, который позволяет работать с данными, содержащимися в базе данных.

Для того, чтобы достигнуть цели ВКР, были поставлены следующие задачи:

* проанализировать существующие развлекательные мультиплатформенные игровые сервисы;
* определить инструментальную базу для разработки приложения;
* изучить требования и спецификацию ПО;
* разработать архитектуру развлекательного мультиплатформенного игрового сервиса;
* реализовать разработанную архитектуру;
* провести отладку и тестирования программы;
* разработать руководство по использованию программы.

Объект исследования – развлекательный мультиплатформенный игровой сервис.

Предмет исследования – инструменты и технологии разработки развлекательного мультиплатформенного игрового сервиса.

Источниками для исследования будут являться интернет-ресурсы, научная и учебная литература, нормативные и законодательные акты.

Одной из особенностей предлагаемого переводчика является его мультиплатформенность. Это означает, что может быть разработано под различные операционные системы, таких как Windows и Android, что значительно расширяет круг потенциальных пользователей.

Актуальность выбранной темы для ВКР обусловлена высокой напряжённостью людей после рабочего дня. Многие очень устают после работы, и хотят в небольшой промежуток времени, который остаётся после завершения всех дел, эмоционально и физически расслабиться. Разработанное приложение поможет людям немного отвлечься и отдохнуть.

Для реализации проекта будет использоваться среда разработки Microsoft Visual Studio. Для написания кода приложения и API будет использоваться язык программирования C#. Для хранения и управления данными будет использоваться система управления базами данных MySQL. Для проектирования и создания макетов будет использоваться draw.io. Для написания отчёта будет использоваться текстовый редактор Microsoft Word.

# ГЛАВА 1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

## 1.1 Описание предметной области

Предметной областью для данного проекта является развлекательный мультиплатформенный игровой сервис. Предполагается, что его функционал приложения поможет пользователям отдохнуть и расслабиться после тяжёлого рабочего дня,. Будет иметься также и соревновательный элемент в виде таблицы с лучшими результатами для каждой мини-игры.

Игровой сервис позволит пользователям получить психологическую разгрузку, что поможет улучшить общее моральное состояние человека, а это в дальнейшем может также повлиять и на улучшение результатов на работе и в увеличении эффективности затрат времени в быту.

Пользователями программного продукта будут являться люди разных возрастов, интерфейс приложения должен быть доступным и понятным на интуитивном уровне, а также лёгким в использовании.

Приложение должно иметь следующий функционал:

* регистрация и авторизация;
* возможность выхода из аккаунта;
* возможность выбора мини-игры;
* игровой процесс;
* возможность публикации результата;
* возможность просмотра списка лучших результатов.

## 1.2 Сравнительный анализ программ аналогов

Для проведения сравнительного анализа были выбраны несколько программ-аналогов, которые имеют похожие функции и предназначены для использования на различных платформах. В данном разделе будет проведён анализ каждой из этих программ по нескольким критериям:

* публикация результата: будет рассмотрена возможность сохранения и публикации результатов;
* удобство использования: будет оценено удобство использования каждой программы, включая интерфейс пользователя и доступность функций;
* назойливость рекламы: будет рассмотрена частота появления рекламных баннеров и приносимый дискомфорт от неё во время пользования приложением;
* дополнительные функции: будет рассмотрено наличие дополнительных функций в каждой программе, таких как возможность использования голосового ввода или функций машинного обучения;
* стоимость: будет оценена стоимость каждой программы, включая цену за использование и возможные дополнительные расходы.

Программный продукт «Google Play Игры» –официальное приложение от компании Google. Является библиотекой скачанных из приложения «Google Play», но помимо этого предоставляет доступ к играм без скачивания. По сути, является площадкой для публикации игр различными разработчиками.

Публикация результата: программа не предоставляет возможности публикации результата, имеются только достижения в играх, которые добавляются разработчиками

Мультиплатформенность: только мобильные устройства

Назойливость рекламы: пользователи приложения жалуются на рекламу, которая, иногда, загораживает игру и не убирается

Удобство использования: Интерфейс интуитивно понятный и удобный

Дополнительные функции: поддержка открытия скаченных игр

Стоимость: Бесплатно

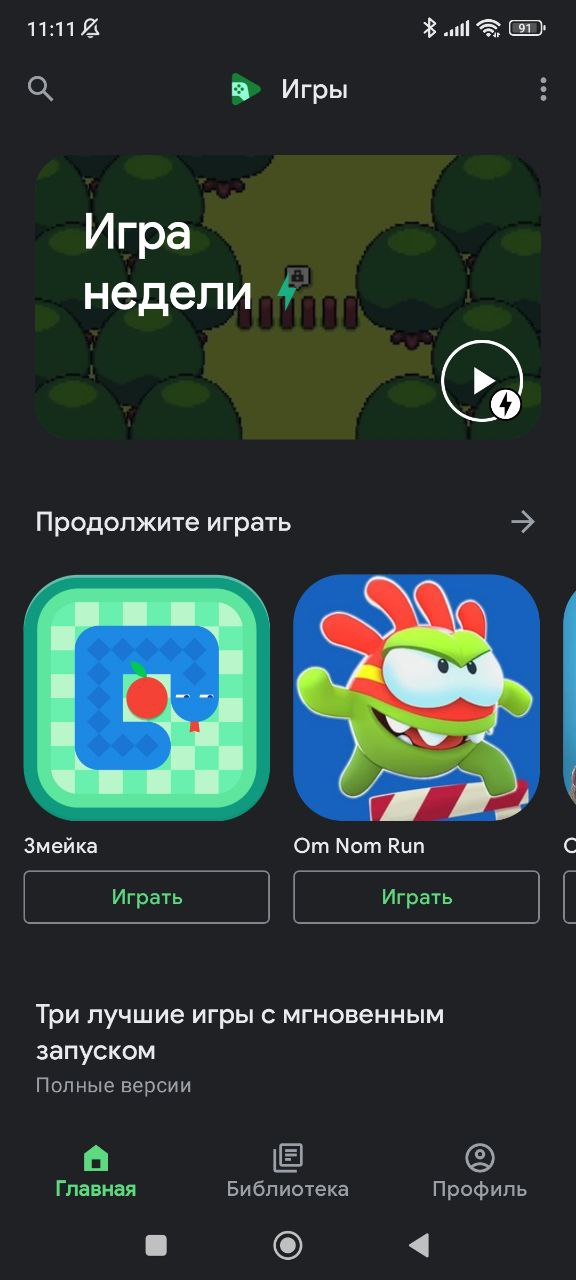


Рисунок 1. Скриншот программы-аналога «Google Play Игры»

Программный продукт «Яндекс игры» **–** это сервис с каталогом игр, которые запускаются онлайн в браузерной версии Яндекс игры. По сути, является площадкой для публикации игр различными разработчиками.

Публикация результата: в некоторых играх есть сохранение результата, но это зависит от разработчиков игр. Возможности публикации и общей таблицы с лучшими результатами нет.

Мультиплатформенность: имеется браузерная версия для персональных компьютеров и мобильная версия в виде приложения.

Назойливость рекламы: реклама появляется в списке игр, при запуске игры, во время игрового процесса сбоку. Также регулярно появляются рекламные баннеры во время игрового процесса.

Удобство использования: интерфейс понятный, удобный. Отталкивает своей рекламой. Из-за обилия цветных иконок, пользователь может перепутать рекламный баннер и игру.

Дополнительные функции: возможность отключения рекламы за деньги, внутриигровые платные услуги.

Стоимость: Бесплатно

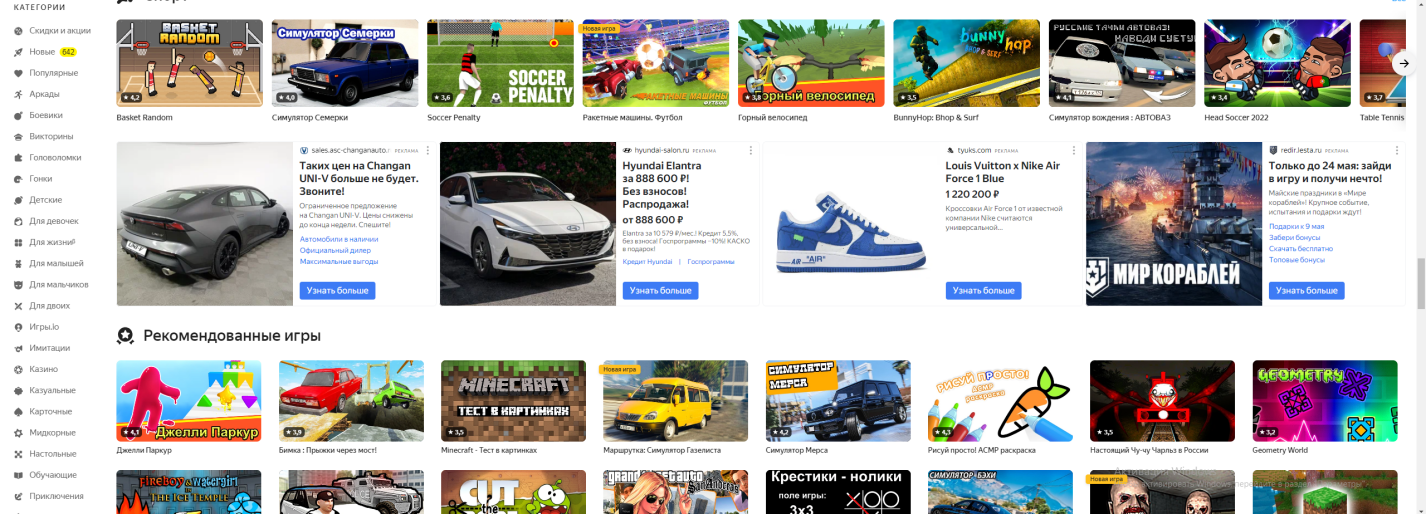


Рисунок 2. Скриншот программы-аналога «Яндекс игры»

После проведения сравнительного анализа аналогов, перечисленных выше, были выявлены достоинства и недостатки каждого из них, результаты сравнительного анализа представлены в таблице 1.

Таблица 1. Результаты сравнительного анализа

|  |  |  |  |
| --- | --- | --- | --- |
| Название  Функции | Google Play Игры | Яндекс игры | Разрабатываемое приложение |
| Публикация результата | - | - | + |
| Мультиплатформенность | - | + | + |
| Назойливость рекламы | + | - | + |
| Удобство использования | + | + | + |
| Дополнительные функции | + | - | - |
| Стоимость | + | + | + |

На основании результатов проведённого анализа функционала можно сделать вывод о том, что разрабатываемый проект, на фоне аналогов, имеет как преимущества, так и недостатки.

Преимуществами разрабатываемого программного решения «Плейзер» является отсутствие рекламы, общая таблица с лучшими результатами для каждой игры, где это возможно, удобный и понятный интерфейс.

Недостатком является отсутствие значительных дополнительных функций.

## 1.3 Постановка задачи

При запуске приложения, первым делом пользователь будет видеть страницу авторизации. На ней будет отображаться поле для ввода логина, поле для ввода пароля и две кнопки: одна – для входа в приложение, а вторая – для перехода на страницу авторизации. Если данные пользователя будут не верны, он увидит уведомление об этом. При верном вводе данных, пользователь будет переходить на главную страницу.

На странице регистрации будут располагаться три поля: одно для логина, второе – для пароля, а третье – для подтверждения пароля, т.е. будет необходимо повторить пароль. Ниже будет находиться кнопка для подтверждения регистрации. При успешной регистрации, пользователь увидит уведомление об этом, после чего автоматически вернётся на страницу авторизации.

На главной странице приложения будет располагаться кнопка выхода в левом верхнем углу. При нажатии на неё пользователю будет необходимо подтвердить желание выйти в диалоговом окне. При подтверждении пользователь выйдет на страницу авторизации. При отказе пользователь останется в главном меню. В правом верхнем углу будет находиться логин пользователя. Ниже будут представлены все доступные игры. При нажатии на «игру», пользователь будет переходить на страницу игры. Там в правой части экрана будет располагаться таблица с лучшими результатами среди всех игроков в выбранной игре, а в левой части будет находиться название игры и кнопка «Играть». При нажатии на эту кнопку будет запускаться игра, которую выбирал пользователь. После игры, если она подразумевает набор каких-либо очков, пользователь сможет отправить результат в базу данных по нажатию на указанную кнопку. После этого результат отобразится в списке лучших результатов. Также пользователь сможет перезапустить игру и выйти из неё в любой момент.

Эксплуатационные требования:

* операционная система Windows 7 и выше;
* контроллер: клавиатура, мышь;

## 1.4 Характеристика инструментальных средств разработки

Для разработки ПО будет использоваться язык программирования C#.

C# – объектно-ориентированный язык программирования общего назначения. Разработан в 1998—2001 годах группой инженеров компании Microsoft под руководством Андерса Хейлсберга и Скотта Вильтаумота как язык разработки приложений для платформы Microsoft .NET Framework и .NET Core. Впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, переменные, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Delphi, Модула, Smalltalk и, в особенности, Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественная реализация интерфейсов).

Выбранной средой разработки стала Microsoft Visual Studio.

Microsoft Visual Studio — линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментов. Данные продукты позволяют разрабатывать как консольные приложения, так и игры и приложения с графическим интерфейсом, в том числе с поддержкой технологии Windows Forms, UWP а также веб-сайты, веб-приложения, веб-службы как в родном, так и в управляемом кодах для всех платформ, поддерживаемых Windows, Windows Mobile, Windows CE, .NET Framework, .NET Core, .NET, MAUI, Xbox, Windows Phone .NET Compact Framework и Silverlight. После покупки компании Xamarin корпорацией Microsoft появилась возможность разработки IOS и Android программ.

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

Для работы с базой данных был выбран язык запросов SQL.

SQL — это язык структурированных запросов, который является компьютерным языком для хранения, обработки и извлечения данных, хранящихся в реляционной базе данных.

SQL является стандартным языком для реляционной системы баз данных. Все системы управления реляционными базами данных (RDMS), такие как MySQL, MS Access, Oracle, Sybase, Informix, Postgres и SQL Server, используют SQL в качестве стандартного языка баз данных.

Для управления базой данных была выбрана система управления базами данных MySQL.

MySQL — свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Продукт распространяется как под GNU General Public License, так и под собственной коммерческой лицензией. Помимо этого, разработчики создают функциональность по заказу лицензионных пользователей. Именно благодаря такому заказу почти в самых ранних версиях появился механизм репликации.

MySQL является решением для малых и средних приложений. Входит в состав серверов WAMP, AppServ, LAMP и в портативные сборки серверов Денвер, XAMPP, VertrigoServ. Обычно MySQL используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы.

Для написания отчёта будет использоваться текстовый редактор Microsoft Word.

Microsoft Word — текстовый процессор, предназначенный для создания, просмотра, редактирования и форматирования текстов статей, деловых бумаг, а также иных документов, с локальным применением простейших форм таблично-матричных алгоритмов. Выпускается корпорацией Microsoft в составе пакета Microsoft Office. Первая версия была написана Ричардом Броди (Richard Brodie) для IBM PC, использующих DOS, в 1983 году. Позднее выпускались версии для Apple Macintosh (1984), SCO UNIX и Microsoft Windows (1989). Текущей версией является Microsoft Office Word 2021 для Windows и macOS, а также веб-версия Word Online (Office Online), не требующая установки программы на компьютер.

Для проектирования и создания макетов будет использоваться приложение draw.io. draw.io — это бесплатное кроссплатформенное программное обеспечение для рисования графиков с открытым исходным кодом, разработанное на HTML5 и JavaScript. Его можно использовать для создания различных диаграмм, таких как блок-схемы, диаграммы UML, сетевые диаграммы и многих других.

draw.io доступна как онлайн в виде кросс-браузерного веб-приложения, так и в виде автономного настольного приложения для Linux, macOS и Windows. Автономная версия приложения разработана с использованием фреймворка Electron. Веб-приложение не требует онлайн-входа или регистрации и может сохраняться на локальном жестком диске. Поддерживаемые форматы хранения и экспорта для загрузки включают PNG, JPEG, SVG и PDF.

## 1.5 Обоснование выбора инструментальных средств разработки

Для разработки развлекательного мультиплатформенного игрового сервиса были выбраны инструменты и технологии, которые позволяют решать поставленные задачи и соответствуют требованиям проекта.

Основные требования к проекту включают разработку приложения, работающего на нескольких платформах.

Для разработки приложения на нескольких платформах были выбраны такие фреймворки как: .Net Framework и Android SDK. .Net Framework был выбран из-за поддержки языка C#, на котором будет реализована desktop версия приложения. Android SDK – из-за поддержки языка Kotlin для мобильной версии приложения.

Фреймворки .NET Framework и Android SDK были выбраны из-за доступности для разработчиков с разным уровнем опыта и знаний, что является важным фактором, который определил выбор данных инструментов. .NET Framework является одним из наиболее популярных инструментов для разработки приложений под Windows, обладает большим сообществом пользователей и имеет обширную документацию. Android SDK, в свою очередь, является стандартным комплектом разработки для Android-приложений и имеет множество учебных материалов и документации, что облегчает разработку для этой платформы.

В связи с тем, что фреймворки .NET Framework и Android SDK являются инструментами разработки в средах Vusial Studio и Android Studio, то для реализации проекта были выбраны именно эти среды разработки.

Приложение будет создано с помощью языка программирования C#, так как этот язык обладает богатым набором библиотек классов, которые позволяют создавать приложения быстро и эффективно. Также C# поддерживает асинхронное программирование и Language Integrated Query. Асинхронное программирование даёт возможность создавать приложения, выполняющие несколько задач одновременно. LINQ обеспечивает работу с данными различных источников (например, баз данных, XML-файлов и т.д.) с помощью унифицированного синтаксиса. C# является кроссплатформенным языком программирования, что позволяет разработчикам создавать приложения, которые могут запускаться на различных операционных системах, таких как Windows, Linux и MacOS.

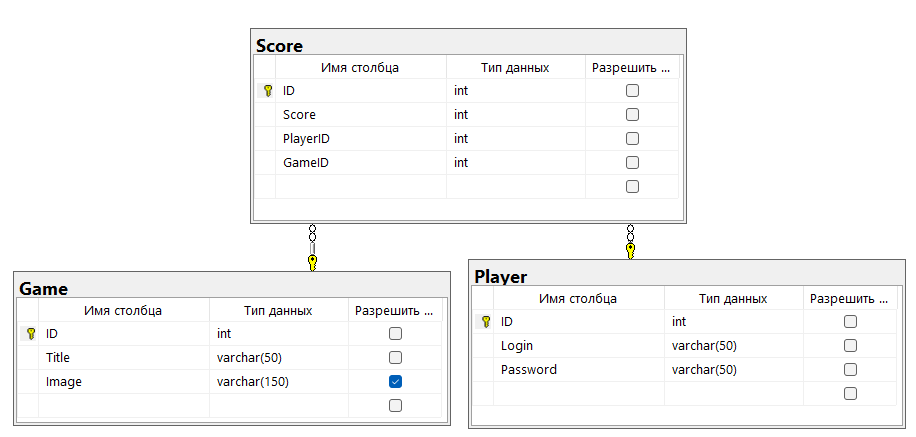
# ГЛАВА 2. ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

## 2.1 Анализ требований и определение спецификаций

Спецификация данного проекта определяется следующими диаграммами:

* диаграмма «Сущность-связь» (ER-диаграмма);
* диаграмма классов;
* диаграмма вариантов использования;
* диаграмма потоков данных верхнего уровня;
* детализированная диаграмма потоков данных.

На рисунке 3 представлена диаграмма «сущность-связь» программного проекта «Развлекательный мультиплатформенный игровой сервис», построенная с использованием Microsoft SQL Server Management Studio.

  
Рисунок 3. Диаграмма «Сущность-связь»

На рисунке 4 представлена диаграмма классов проекта «Развлекательный мультиплатформенный игровой сервис».

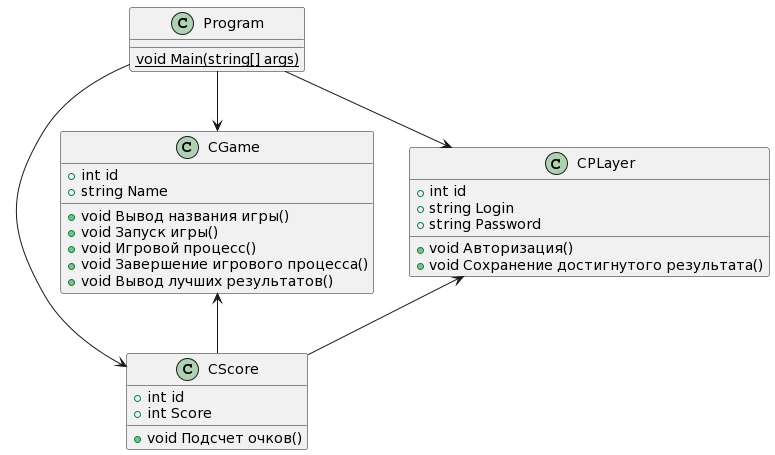
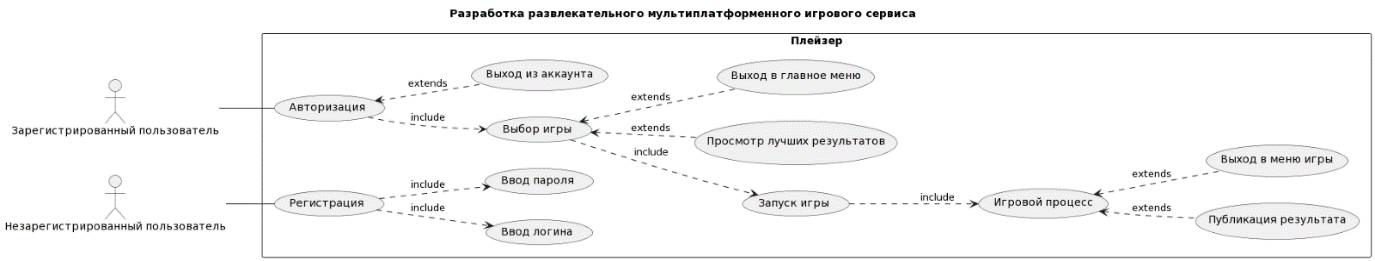
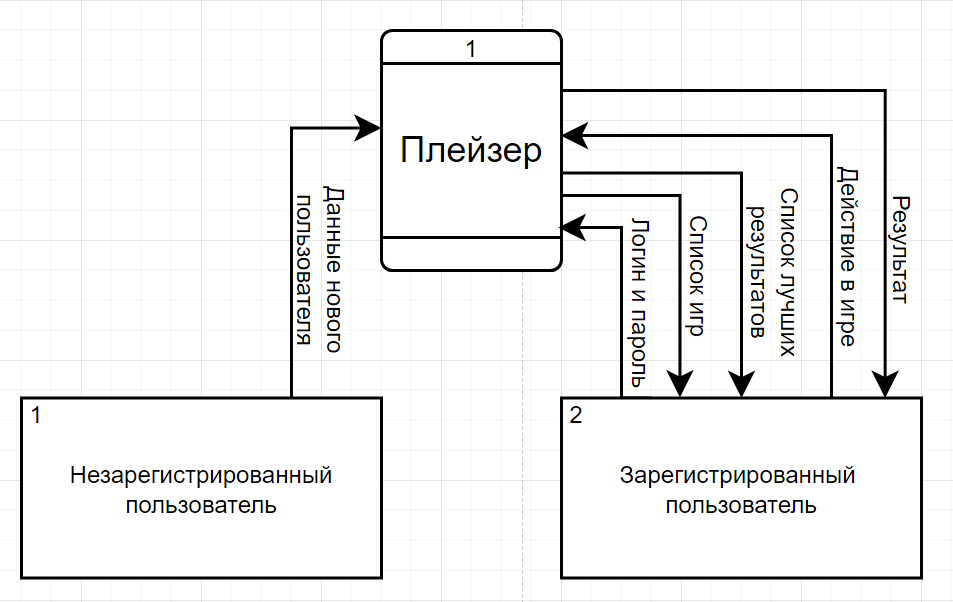
  
Рисунок 4. Диаграмма классов

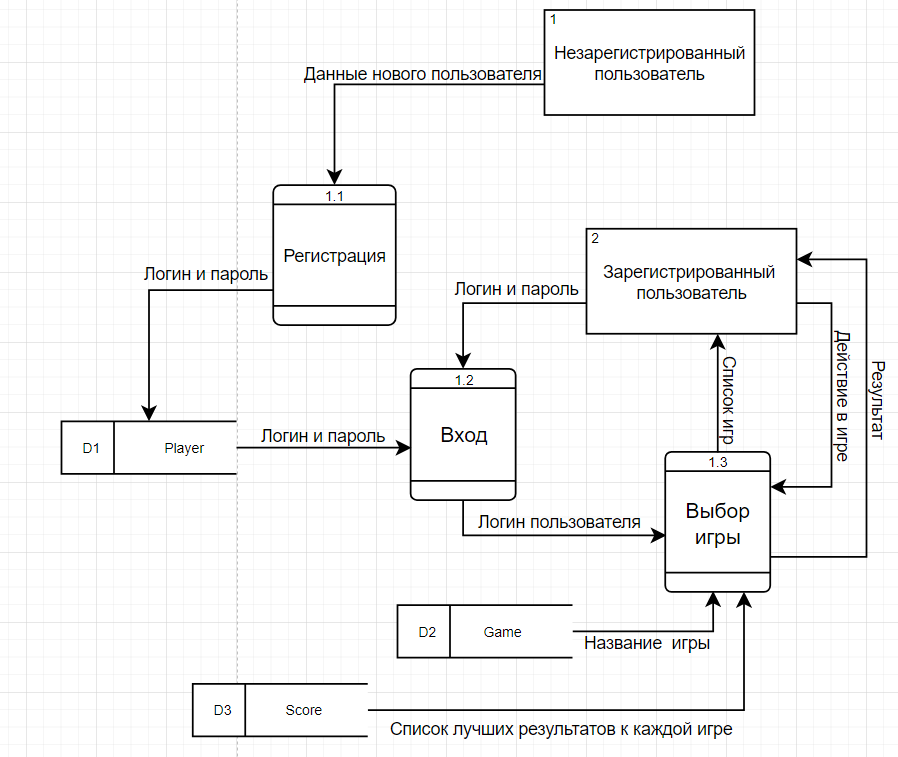
Диаграмма вариантов использования (сценариев поведения, прецедентов) является исходным концептуальным представлением системы в процессе её проектирования и разработки. Данная диаграмма состоит из актёров, вариантов использования и отношений между ними. При построении диаграммы могут использоваться также общие элементы нотации: примечания и механизмы расширения (Рисунок 5).

  
Рисунок 5. Диаграмма вариантов использования

На рисунке 6 изображена диаграмма потоков данных верхнего уровня для программного продукта «Развлекательный мультиплатформенный игровой сервис».

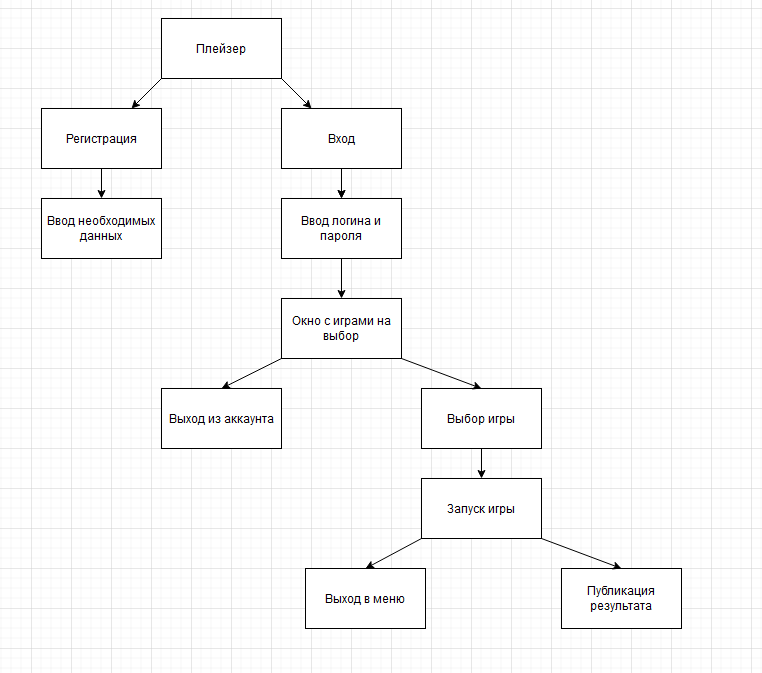
  
Рисунок 6. Диаграмма потоков данных верхнего уровня

На рисунке 7 изображена детализированная диаграмма потоков данных для программного продукта «Развлекательный мультиплатформенный игровой сервис».

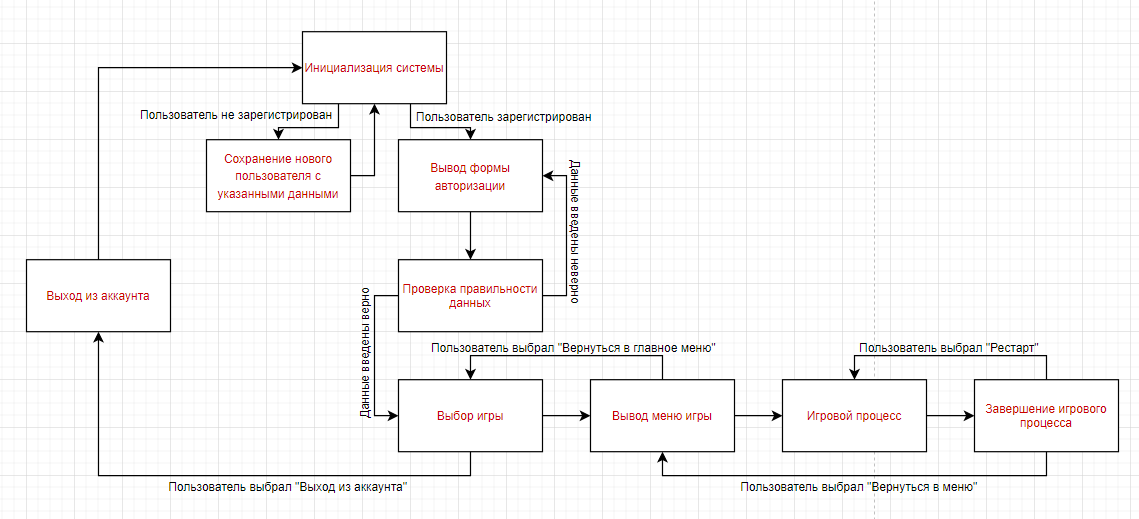
  
Рисунок 7. Детализированная диаграмма потоков данных

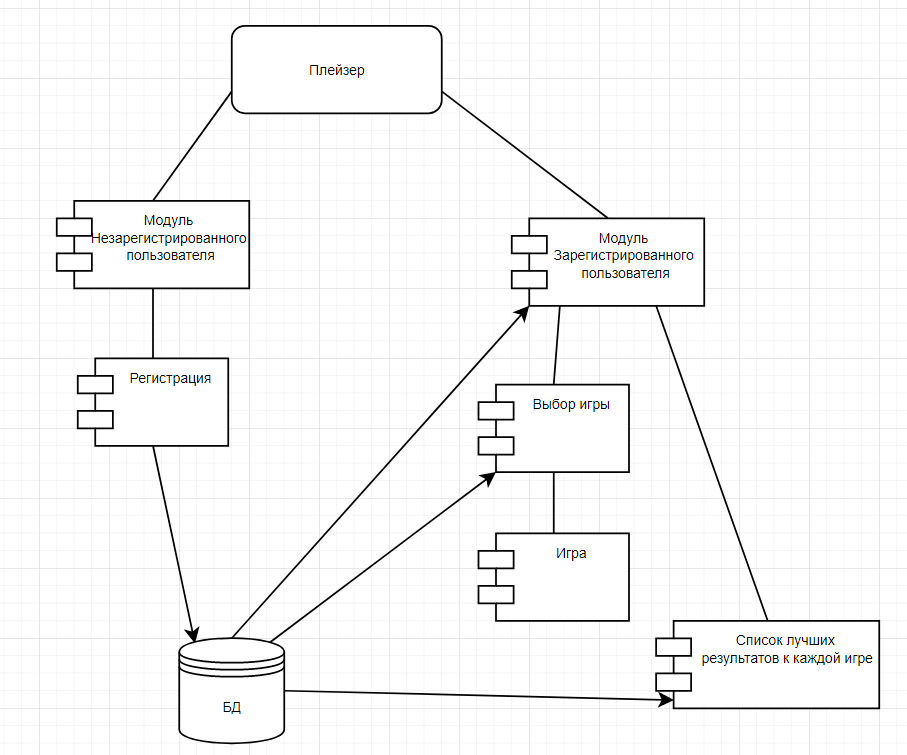
## 2.2 Проектирование программного обеспечения

На рисунке 8 представлена структурная схема, которая содержит в себе основные функции разрабатываемого программного продукта.

Рисунок 8. Структурная схема

На рисунке 9 представлена диаграмма состояний для разработки программного продукта «Развлекательный мультиплатформенный игровой сервис».

Рисунок 9. Диаграмма состояний

На рисунке 10 представлена диаграмма компонентов программного продукта «Развлекательный мультиплатформенный игровой сервис».  
Рисунок 10. Диаграмма компонентов

Игровой сервис позволит пользователям получить психологическую разгрузку, что поможет улучшить общее моральное состояние человека, а это в дальнейшем может также повлиять и на улучшение результатов на работе и в увеличении эффективности затрат времени в быту. Приложение будет разрабатываться в развлекательных целях, будет доступно без возрастного ограничения, так как не имеет в себе никаких оскорбительных или недопустимых для детей фраз или изображений.

## 2.3 Разработка программного обеспечения

Программный продукт «Плейзер» – это развлекательный мультиплатформенный игровой сервис, разработанный на базе платформ Android и Windows. Основная функция приложения «Плейзер» - разнообразить досуг людей, объединив различные классические игры в одном месте. Дополнительно имеется соревновательный элемент, позволяющий конкурировать с другими игроками по набору очков в играх. Также есть игры для двоих на одном устройстве.

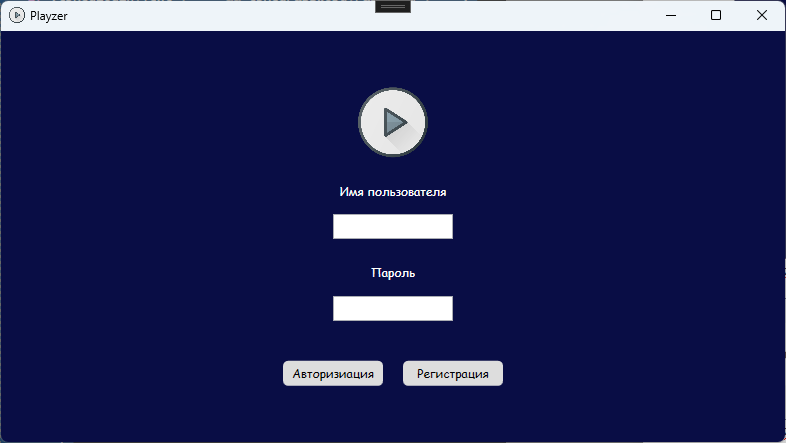
При запуске приложения «Плейзер» выводится окно авторизации. На ней представлено два поля для ввода логина и пароля, а также две кнопки (Рисунок 11).

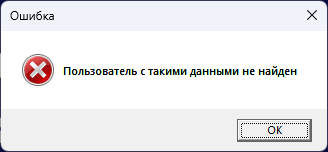
Верхнее текстовое поле предназначено для ввода логина

Нижнее текстовое поля предназначено для ввода пароля

Кнопка «Авторизация» при нажатии переводит пользователя в главное меню, если пользователь правильно ввёл свои данные. Если данные были введены неверно, пользователь увидит уведомление об этом (Рисунок 12).

Кнопка «Регистрация» при нажатии переводит пользователя на страницу регистрации.

Рисунок 11. Страница авторизации

  
Рисунок 12. Уведомление при неверном вводе данных

Ниже представлены вёрстка и код окна:

MainWindow.xaml

<Window x:Class="Playzer.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:Playzer"

mc:Ignorable="d"

Title="Playzer" Height="450" Width="800" MinHeight="450" MinWidth="540">

<Grid>

<Frame Name="MainFrame" NavigationUIVisibility="Hidden" Source="Pages/AuthPage.xaml"/>

</Grid>

</Window>

MainWindow.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Playzer

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

}

}

}

Ниже представлены вёрстка и код страницы авторизации:

AuthPage.xaml

<Page x:Class="Playzer.Pages.AuthPage"

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

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

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

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

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

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="AuthPage">

<Grid Background="#090d45">

<StackPanel Orientation="Vertical" HorizontalAlignment="Center" VerticalAlignment="Center">

<Image Source="/Images/PlayzerIcon\_Demo.png" Width="70"/>

<Label Content="Имя пользователя" Foreground="White" HorizontalAlignment="Center" Margin="0,20,0,0"/>

<TextBox Name="tbLogin" ToolTip="Введите имя пользователя" TextWrapping="Wrap" Width="120" Height="25" HorizontalAlignment="Center" Margin="0,10,0,0"/>

<Label Content="Пароль" Foreground="White" HorizontalAlignment="Center" Margin="0,20,0,0"/>

<PasswordBox Name="tbPassword" ToolTip="Введите пароль" Width="120" Height="25" HorizontalAlignment="Center" Margin="0,10,0,0"/>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="0,40,0,0">

<Button Name="btnAuth" Content="Авторизиация" Height="25" Width="100" Click="btnAuth\_Click"/>

<Button Name="btnReg" Content="Регистрация" Height="25" Width="100" Margin="20,0,0,0" Click="btnReg\_Click"/>

</StackPanel>

</StackPanel>

</Grid>

</Page>

AuthPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Playzer.Pages

{

/// <summary>

/// Логика взаимодействия для AuthPage.xaml

/// </summary>

public partial class AuthPage : Page

{

public AuthPage()

{

InitializeComponent();

}

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

{

var usersData = Entities.GetContext().Player.AsNoTracking().FirstOrDefault(u => u.Login == tbLogin.Text && u.Password == tbPassword.Password);

if (usersData == null)

MessageBox.Show("Пользователь с такими данными не найден", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

else

{

var userToken = usersData as Player;

NavigationService.Navigate(new MainMenuPage(userToken));

}

}

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

{

NavigationService.Navigate(new RegPage());

}

}

}

На странице регистрации имеется три текстовых поля, расположенных в столбик, а также две кнопки. Одна в левом верхнем углу - для возврата на страницу авторизации, а вторая для подтверждения регистрации, находится в самом низу (Рисунок 13).

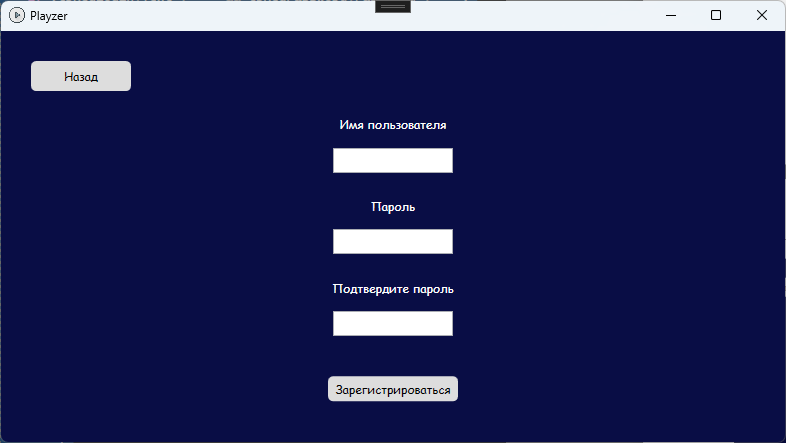
– Верхнее текстовое поле предназначено для ввода логина;

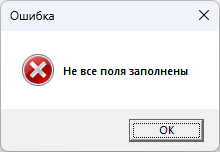
– Текстовое поле в середине предназначено для ввода пароля;

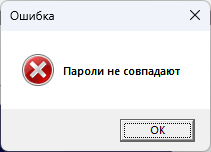
– Нижнее текстовое поле предназначено для повторного ввода пароля в целях подтверждения;

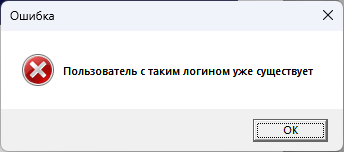
– Кнопка «Назад» при нажатии возвращает пользователя на страницу авторизации;

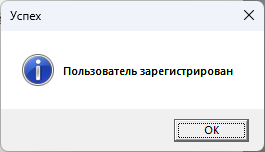
– Кнопка «Зарегистрироваться» при нажатии проверяет поля на ввод данных. Если одно из полей пустое - пользователь увидит сообщение об этом (Рисунок 14). Если пароли не совпадают - программа оповестит пользователя об этом (Рисунок 15). Если логин будет занят, пользователь также не сможет зарегистрироваться, и увидит сообщение об этом (Рисунок 16). При соблюдении всех условий, пользователь сможет зарегистрировать аккаунт, после чего будет об этом оповещён (Рисунок 17).

Рисунок 13. Страница регистрации

  
Рисунок 14. Ошибка при пустых полях

  
Рисунок 15. Ошибка несовпадения паролей

  
Рисунок 16. Ошибка при попытке создать аккаунт с существующим логином

  
Рисунок 17. Успешная регистрация аккаунта

Ниже представлены вёрстка и код страницы регистрации:

RegPage.xaml

<Page x:Class="Playzer.Pages.RegPage"

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

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

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

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

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

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="RegPage">

<Grid Background="#090d45">

<Button Name="btnBack" Height="30" Width="100" Content="Назад" VerticalAlignment="Top" HorizontalAlignment="Left" Margin="30,30,0,0" Click="btnBack\_Click"/>

<StackPanel Orientation="Vertical" HorizontalAlignment="Center">

<Label Content="Имя пользователя" Foreground="White" HorizontalAlignment="Center" Margin="0,80,0,0"/>

<TextBox Name="tbRegLogin" ToolTip="Введите имя пользователя" TextWrapping="Wrap" Width="120" Height="25" HorizontalAlignment="Center" Margin="0,10,0,0"/>

<Label Content="Пароль" Foreground="White" HorizontalAlignment="Center" Margin="0,20,0,0"/>

<PasswordBox Name="tbRegPassword" ToolTip="Введите пароль" Width="120" Height="25" HorizontalAlignment="Center" Margin="0,10,0,0"/>

<Label Content="Подтвердите пароль" Foreground="White" HorizontalAlignment="Center" Margin="0,20,0,0"/>

<PasswordBox Name="tbRegPasswordConfirm" ToolTip="Введите пароль" Width="120" Height="25" HorizontalAlignment="Center" Margin="0,10,0,0"/>

<StackPanel Orientation="Horizontal" HorizontalAlignment="Center" Margin="0,40,0,0">

<Button Name="btnRegistration" Content="Зарегистрироваться" Height="25" Width="130" Click="btnRegistration\_Click"/>

</StackPanel>

</StackPanel>

</Grid>

</Page>

RegPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Playzer.Pages

{

/// <summary>

/// Логика взаимодействия для RegPage.xaml

/// </summary>

public partial class RegPage : Page

{

public RegPage()

{

InitializeComponent();

}

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

{

NavigationService.GoBack();

}

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

{

if (tbRegLogin.Text != "" && tbRegPassword.Password != "")

{

if (tbRegPassword.Password == tbRegPasswordConfirm.Password)

{

var checkLogin = Entities.GetContext().Player.AsNoTracking().FirstOrDefault(x => x.Login == tbRegLogin.Text);

if (checkLogin != null)

{

MessageBox.Show("Пользователь с таким логином уже существует", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

else if (checkLogin == null)

{

Player newPlayer = new Player();

newPlayer.Login = tbRegLogin.Text;

newPlayer.Password = tbRegPassword.Password;

try

{

Entities.GetContext().Player.Add(newPlayer);

Entities.GetContext().SaveChanges();

MessageBox.Show("Пользователь зарегистрирован", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

NavigationService.GoBack();

}

catch

{

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

}

}

}

else

MessageBox.Show("Пароли не совпадают", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

else

MessageBox.Show("Не все поля заполнены", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

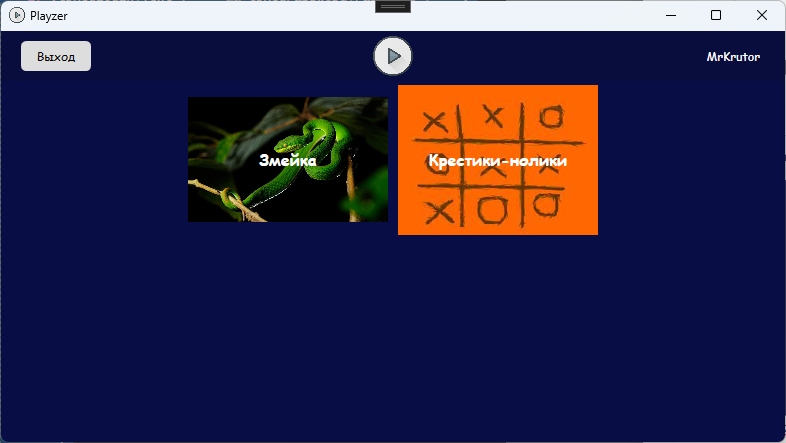
}

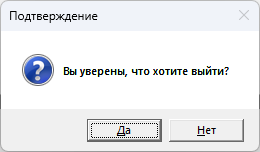
}

При успешной авторизации пользователь перейдёт в главное меню, где он сможет увидеть в верхнем левом углу кнопку для выхода из аккаунта, в правом верхнем углу свой логин, а немного ниже - список доступных игр (Рисунок 18).

Кнопка «Выход» при нажатии вызовет диалоговое окно с просьбой подтвердить намерение выйти из аккаунта (Рисунок 19). Если пользователь согласится, то он перейдёт обратно на страницу авторизации.

Нажатие на картинку с названием игры откроет игровое меню выбранной игры.

Рисунок 18. Главное меню

  
Рисунок 19. Подтверждение выхода

Ниже представлены вёрстка и код страницы главного меню.

MainMenuPage.xaml

<Page x:Class="Playzer.Pages.MainMenuPage"

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

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

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

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

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

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="MainMenuPage">

<Grid Background="#090d45">

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition Height="\*"/>

</Grid.RowDefinitions>

<Rectangle Fill="#090d3d" Grid.Row="0"/>

<Button Name="btnExit" Height="30" Width="70" Content="Выход" HorizontalAlignment="Left" VerticalAlignment="Center" Margin="20,0,0,0" Click="btnExit\_Click"/>

<Label Name="lbUserName" HorizontalAlignment="Right" VerticalAlignment="Center" Content="Имя пользователя" Margin="0,0,20,0" Foreground="White"/>

<Image Source="\Images\PlayzerIcon\_Demo.png" Height="40" Width="40"/>

<ListView Name="gamesList" Grid.Row="1" Background="#090d45" BorderBrush="#090d45" ScrollViewer.VerticalScrollBarVisibility="Hidden" HorizontalContentAlignment="Center" HorizontalAlignment="Center" ScrollViewer.HorizontalScrollBarVisibility="Disabled">

<ListView.ItemsPanel>

<ItemsPanelTemplate>

<WrapPanel Orientation="Horizontal"/>

</ItemsPanelTemplate>

</ListView.ItemsPanel>

<ListView.ItemTemplate>

<DataTemplate>

<Grid MouseLeftButtonUp="listViewElement\_Click">

<Image Width="200">

<Image.Source>

<Binding Path="Image"/>

</Image.Source>

</Image>

<Label Name="lbSelectedGameTitle" FontSize="16" FontWeight="Bold" Content="{Binding Title}" Foreground="White" HorizontalAlignment="Center" VerticalAlignment="Center"/>

</Grid>

</DataTemplate>

</ListView.ItemTemplate>

</ListView>

</Grid>

</Page>

MainMenuPage.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Playzer.Pages

{

/// <summary>

/// Логика взаимодействия для MainMenuPage.xaml

/// </summary>

public partial class MainMenuPage : Page

{

Player playerInfo = new Player();

public MainMenuPage(Player userToken)

{

InitializeComponent();

lbUserName.Content = userToken.Login;

playerInfo = userToken;

var gamesData = Entities.GetContext().Game.ToList();

gamesList.ItemsSource = gamesData;

}

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

{

var result = MessageBox.Show("Вы уверены, что хотите выйти?", "Подтверждение", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

NavigationService.Navigate(new AuthPage());

}

private void listViewElement\_Click(object sender, MouseButtonEventArgs e)

{

var selectedGame = (gamesList.SelectedIndex + 1);

NavigationService.Navigate(new GameMenuPage(selectedGame, playerInfo));

}

}

}

На странице игры в левой части располагаются кнопки для возврата в главное меню и начала игры, а также название выбранной игры. В правой части экрана находится список лучших результатов в выбранной игре (Рисунок 20).

Кнопка «Назад» при нажатии возвращает пользователя в главное меню.

Кнопка «Играть» при нажатии запускает выбранную игру.

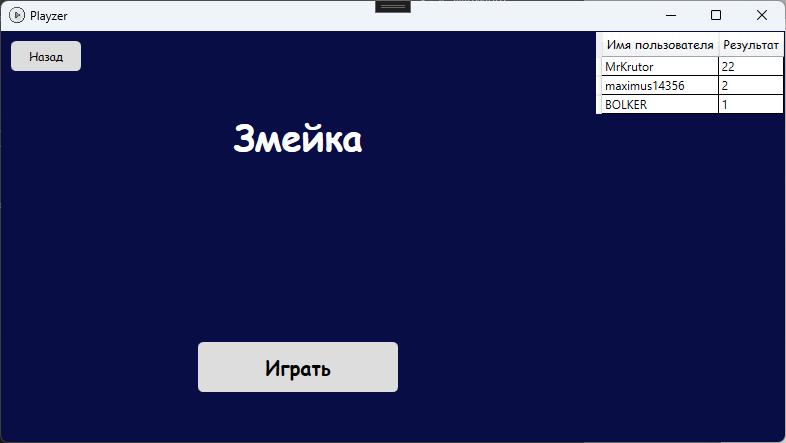


Рисунок 20. Меню игры

Ниже представлены вёрстка и код страницы игрового меню:

GameMenuPage.xaml

<Page x:Class="Playzer.Pages.GameMenuPage"

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

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

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

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

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

mc:Ignorable="d"

d:DesignHeight="450" d:DesignWidth="800"

Title="GameMenuPage">

<Grid Background="#090d45">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="auto"/>

</Grid.ColumnDefinitions>

<Button Name="btnBack" Click="btnBack\_Click" Height="30" Width="70" Content="Назад" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10,10,0,0"/>

<Button Name="btnPlay" Click="btnPlay\_Click" Height="50" Width="200" Content="Играть" FontSize="20" FontWeight="Bold" VerticalAlignment="Bottom" HorizontalAlignment="Center" Margin="0,0,0,50"/>

<Label Name="lbGameTitle" Content="Название игры" FontSize="36" HorizontalAlignment="Center" VerticalAlignment="Center" Margin="0,0,0,200" FontWeight="Bold" Foreground="White"/>

<DataGrid Name="bestScoreList" BorderBrush="#090d45" Grid.Column="1" CanUserAddRows="False" IsReadOnly="True" Background="#090d45" HorizontalAlignment="Right" AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn Header="Имя пользователя" Binding="{Binding Player.Login}"/>

<DataGridTextColumn Header="Результат" Binding="{Binding Score1}"/>

</DataGrid.Columns>

</DataGrid>

</Grid>

</Page>

GameMenuPage.xaml.cs

using Playzer.GameWindows;

using Playzer.TicTacToeFolder;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

namespace Playzer.Pages

{

/// <summary>

/// Логика взаимодействия для GameMenuPage.xaml

/// </summary>

public partial class GameMenuPage : Page

{

int gameID;

Player playerInfo = new Player();

public GameMenuPage(int selectedGameID, Player playerData)

{

InitializeComponent();

var selectedGameData = Entities.GetContext().Game.AsNoTracking().FirstOrDefault(x => x.ID == selectedGameID);

var bestScore = Entities.GetContext().Score.Where(x => x.GameID == selectedGameID).OrderByDescending(x => x.Score1).ToList();

bestScoreList.ItemsSource = bestScore;

lbGameTitle.Content = selectedGameData.Title;

gameID = selectedGameID;

playerInfo = playerData;

}

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

{

NavigationService.GoBack();

}

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

{

switch (gameID)

{

default:

MessageBox.Show("Игра не готова", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Information);

break;

case 1:

var gameInfo = Entities.GetContext().Game.AsNoTracking().FirstOrDefault(x => x.ID == gameID);

Window goToSnake = new SnakeGame(gameInfo, playerInfo);

goToSnake.Show();

break;

case 2:

Window goToTicTacToe = new TicTacToe();

goToTicTacToe.Show();

break;

}

}

}

}

Пользователь, на данный момент, имеет возможность поиграть в 2 игры: «Змейка» и «Крестики-нолики». При запуске змейки пользователя встречает окно с кратким объяснением правил игры и управлением (Рисунок 21). Во время игры, пользователь управляет змейкой, должен собирать яблоки. Нельзя врезаться в стены и в свой хвост - это приведёт к поражению. Сверху, во время игры, отображается количество набранных баллов и скорость тика игры. При подборе яблока, увеличивается скорость игры и итоговый счёт (Рисунок 22). После окончания игры, пользователь увидит окно, сообщающее о достигнутом результате с возможностью сохранить результат (Рисунок 23). При подтверждении в диалоговом окне (Рисунок 24), результат сохранится, а пользователь увидит сообщение об этом (Рисунок 25). Если сохранить результат не получится, программа оповестит об этом пользователя.

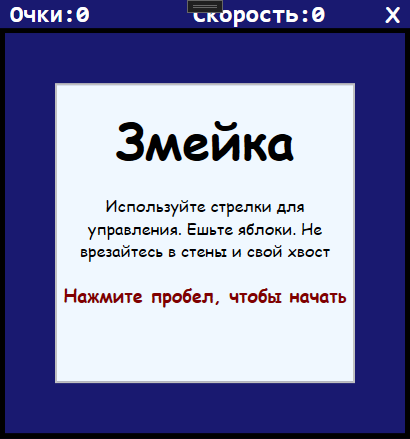
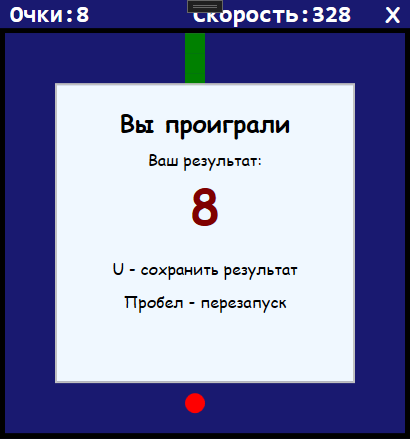
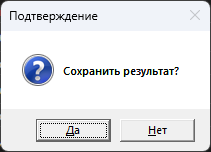
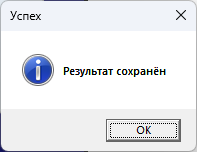
  
Рисунок 21. Приветственное окно игры «Змейка»

  
Рисунок 22. Игровой процесс игры «Змейка»

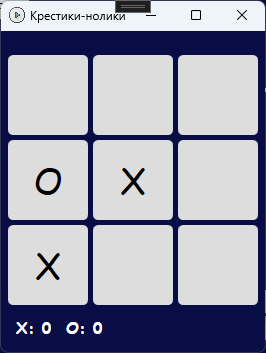
  
Рисунок 23. Окно конца игры «Змейка»

  
Рисунок 24. Подтверждение сохранения результата игры «Змейка»

  
Рисунок 25. Уведомление об успехе сохранения результата

Вёрстка и код игры «Змейка» представлены в приложении 1.

Игра «Крестики-нолики» является игрой, предназначенной для 2-х игроков. В ней не предусмотрен итоговый результат за один раунд, так что результат не записывается. При запуске игры можно увидеть 9 ячеек для крестиков и ноликов. Внизу располагается количество побед обеих сторон. (Рисунок 26). При победе появляется надпись, которая показывает либо победившую сторону (Рисунок 27) или оповещает о ничьей (Рисунок 28).

  
Рисунок 26. Окно игры «Крестики-нолики»

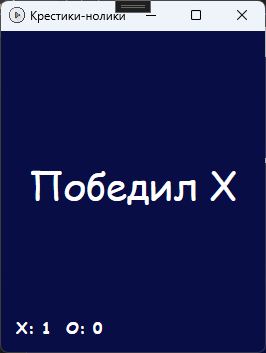
  
Рисунок 27. Игрок победил в раунде игры «Крестики-нолики»

  
Рисунок 28. Ничья в раунде игры «Крестики-нолики»

Вёрстка и код игры «Крестики-нолики» представлены в приложении 2.

Ниже представлен код словаря стилей:

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

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

<Style TargetType="Button">

<Setter Property="Template">

<Setter.Value>

<ControlTemplate TargetType="Button">

<Border CornerRadius="5" Background="{TemplateBinding Background}">

<ContentPresenter HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}"

VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>

</Border>

</ControlTemplate>

</Setter.Value>

</Setter>

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="Label">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="TextBlock">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="TextBox">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

<Style TargetType="PasswordBox">

<Setter Property="FontFamily" Value="Comic Sans MS"/>

</Style>

</ResourceDictionary>

## 2.4 Отладка и тестирование программы

Для проверки реализованного функционала необходимо провести отладку и тестирование программного обеспечения.

В таблице 2 представлены результаты отладки и тестирования программного обеспечения

Таблица 2. Тестирование и отладка программного обеспечения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № Теста | Входные данные | Вводимое значение | Ожидаемый результат | Фактический результат | Ошибка выявлена |
| 1 | Незарегистрированный логин | qwe123 | “Пользователь с такими данными не найден” |  | Нет |
| 2 | Зарегистрированный логин, неправильный пароль | Логин - MrKrutor  Пароль - qwe123 | “Пользователь с такими данными не найден” |  | Нет |
| 3 | Пустые поля логина и пароля |  | “Пользователь с такими данными не найден” |  | Нет |
| 4 | Пустые поля логина, пароля и подтверждения пароля |  | “Не все поля заполнены” |  | Нет |
| 5 | Зарегистрированный логин и корректно введённый пароль | Логин - MrKrutor  Пароль - qwe  Подтвердите пароль - qwe | “Пользователь с таким логином уже существует” |  | Нет |
| 6 | Уникальный логин, пароль повторён не верно | Логин - qwe123  Пароль - 123  Подтвердите пароль - qwe | “Пароли не совпадают” |  | Нет |
| 7 | Уникальный логин, правильно введённый пароль | Логин - qwe123  Пароль - 123  Подтвердите пароль - 123 | “Пользователь зарегистрирован” |  | Нет |
| 8 | Нажатие кнопки “Регистрация” |  | Переход на страницу регистрации | Переход на страницу регистрации | Нет |
| 9 | Нажатие кнопки “Назад” на странице регистрации |  | Переход на страницу авторизации | Переход на страницу авторизации | Нет |
| 10 | Нажатие кнопки “Выход” в главном меню | “Да” в диалоговом окне | Переход на страницу авторизации | Переход на страницу авторизации | Нет |
| 11 | Нажатие кнопки “Выход” в главном меню | “Нет” в диалоговом окне | Диалоговое окно закроется, открыто главное меню | Диалоговое окно закрылось, открыто главное меню | Нет |
| 12 | Нажатие кнопки “U” во время игрового процесса игры «Змейка» | Клавиша “U” | Подтверждение сохранения результата не откроется | Подтверждение сохранения результата не открылось | Нет |
| 13 | Нажатие пробела во время игрового процесса игры «Змейка» | Клавиша “Пробел” | Игра перезапустится | Игра перезапустилась | Нет |

Проведя работу с отладкой и тестированием, можно утверждать, что программа соответствует как функциональным, так и нефункциональным требованиям.

## 2.5 Руководство по использованию программы

Назначением программы является развлечение пользователей.

Функции программы:

* регистрация;
* авторизация;
* запуск игр и игровой процесс;
* выход из аккаунта;
* сохранение результатов;
* просмотр списка лучших результатов.

Ниже представлены минимальные требования для пользования приложением:

* операционная система: Windows;
* разрядность: х86(32-bit) x64(64-bit);
* процессор: от 500 MHz;
* видеоадаптер: 3D;
* видеопамять: 64 Mb;
* место на носителе: 70Mb;
* оперативная память [RAM]: 256 Mb;
* контроллер: клавиатура, мышь.

Программа «Плейзер» - это развлекательный мультиплатформенный игровой сервис. В нём имеются такие функции, как: регистрация, авторизация, запуск игр и игровой процесс, выход из аккаунта, сохранение результатов, просмотр лучших результатов. Функция «регистрация» - пользователь может зарегистрировать себе аккаунт для использования приложения. Функция «авторизация» - пользователь может войти в приложение, используя данные своего аккаунта (логин и пароль). Функция «запуск игр и игровой процесс» - пользователь может запускать различные игры в программе и играть в них. Функция «выход из аккаунта» - пользователь может выйти из аккаунта в главном меню, вернувшись на страницу авторизации. Функция «сохранение результатов» - пользователь может сохранить свой итоговый счёт в базе данных, после чего на странице игры будет отображаться его результат, если он попадёт в список лучших. Функция «просмотр результатов» - пользователь может просматривать список лучших результатов в играх, где подразумевается набор очков.

Чтобы начать пользоваться приложением, пользователю необходимо перейти на страницу регистрации и зарегистрироваться. После этого, он попадёт на страницу авторизации, где нужно заполнить поля данными своего аккаунта. При правильном вводе данных, пользователь увидит главное меню со списком игр. Он может нажать на любую игру и перейти на её страницу. Там он сможет запустить её и начать игровой процесс. После окончания, он может сохранить результат или выйти из игры, вернуться в главное меню и выбрать другую игру.

# ЗАКЛЮЧЕНИЕ

В ходе работы над проектом разработан развлекательный мультиплатформенный игровой сервис, который может использоваться на мобильных устройствах и персональных компьютерах Для разработки desktop версии использовался .NET Framework, а для мобильной версии - Android Studio.

При разработке использовались современные методы и технологии: .NET Framework для разработки desktop версии приложения и Android Studio для мобильной версии. Разработанный игровой сервис имеет приятный дизайн и понятный интерфейс, благодаря чему он доступен для широкого круга пользователей.

В результате дипломного проекта был создан развлекательный мульиплатформенный игровой сервис, который позволит улучшить качество досуга для пользователей. Это может положительно сказаться на общем эмоциональном состоянии пользователей.

Игровой сервис имеет потенциал для развития. К примеру, введение большего числа игр, добавление личного кабинета, привязка к почте для восстановления пароля и др.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

*Законодательные и нормативные акты:*

1. ГОСТ Р 7.0.12-2011 Библиографическая запись. Сокращение слов и словосочетаний на русском языке. Общие требования и правила. – М.: Стандартинформ, 2012. – 61 с.
2. ГОСТ 7.32-2017 Отчет о научно-исследовательской работе. Структура и правила оформления. – М.: Стандартинформ, 2017. – 47 с.
3. ГОСТ 7.82-2001 Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления. – М.: ИПК Издательство стандартов, 2001. – 39 с.
4. ГОСТ Р 7.0.100-2018 Библиографическая запись. Библиографическое описание. Общие требования и правила составления. – М.: Стандартинформ, 2018. – 122 с.
5. ГОСТ Р 7.0.5-2008 Библиографическая ссылка. Общие требования и правила составления. – М.: Стандартинформ, 2008. – 32 с.

*Интернет-документы:*

1. Интернет-сервис для построения схем и диаграмм Draw.io. – [Электронный ресурс]. – URL: <https://www.draw.io/>
2. Интернет-ресурс с помощью по создании игры «Змейка» – [Электронный ресурс]. – URL: <https://goo.su/kUm2x>
3. Автор видеороликов с обучением по созданию игр на C# – [Электронный ресурс]. – URL: <https://youtube.com/@developerscityarmenia228>
4. Интернет-ресурс с документацией по C# – [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/>
5. Интернет-ресурс с описанием возможностей элемента ListView в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/5.9.php>
6. Интернет-ресурс с описанием возможностей элемента Button в WPF – [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.controls.button?view=windowsdesktop-8.0>
7. Интернет-ресурс с информацией по созданию словаря стилей WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/9.3.php>
8. Интернет-ресурс с описанием возможностей элемента Grid в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/4.2.php>
9. Интернет-ресурс с информацией об элементах Frame и Window в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/uwp/8.2.php>
10. Интернет-ресурс с информацией об элементе Label в WPF – [Электронный ресурс]. – URL: <https://clck.ru/34ZkpG>
11. Интернет-ресурс с информацией привязке данных WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/11.php>
12. Интернет-ресурс с информацией о свойстве элементов Visibility WPF – [Электронный ресурс]. – URL: <https://www.c-sharpcorner.com/article/visibilitycollapsed-vs-visibilityhidden-in-wpf/>
13. Интернет-ресурс с информацией об элементе DockPanel в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/4.5.php>
14. Интернет-ресурс с информацией об элементе StackPanel в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/4.4.php>
15. Интернет-ресурс с информацией об элементе WrapPanel в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/4.6.php>
16. Интернет-ресурс с информацией об элементе Border в WPF – [Электронный ресурс]. – URL: <https://clck.ru/34Zm4r>
17. Интернет-ресурс с информацией об элементе Canvas в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/4.7.php>
18. Интернет-ресурс с информацией о текстовых элементах в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/wpf/5.5.php>
19. Интернет-ресурс с информацией об элементе TextBox в WPF – [Электронный ресурс]. – URL: <https://clck.ru/34Zm9p>
20. Интернет-ресурс с информацией об элементе Brush в WPF – [Электронный ресурс]. – URL: <https://clck.ru/34ZmDV>
21. Интернет-ресурс с информацией о классе UIElement в WPF – [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.windows.uielement?view=windowsdesktop-8.0>
22. Интернет-ресурс с информацией о циклах в WPF – [Электронный ресурс]. – URL: <https://metanit.com/sharp/tutorial/2.6.php>
23. Интернет-ресурс с информацией о конструкции switch в C# – [Электронный ресурс]. – URL: <https://metanit.com/sharp/tutorial/3.45.php>
24. Интернет-ресурс с информацией о классе Timer в C# – [Электронный ресурс]. – URL: <https://learn.microsoft.com/ru-ru/dotnet/api/system.timers.timer?view=net-7.0>
25. Интернет-ресурс с информацией об обработке исключений try, catch в C# и .NET – [Электронный ресурс]. – URL: <https://metanit.com/sharp/tutorial/2.14.php>

# ПРИЛОЖЕНИЕ 1

SnakeGame.xaml

<Window x:Class="Playzer.GameWindows.SnakeGame"

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:Playzer.GameWindows"

mc:Ignorable="d"

Title="Змейка - Результат: 0" SizeToContent="WidthAndHeight" ContentRendered="Window\_ContentRendered"

KeyUp="Window\_KeyUp" ResizeMode="NoResize" WindowStyle="None" Background="Black" MouseDown="Window\_MouseDown">

<DockPanel Background="MidnightBlue">

<Grid DockPanel.Dock="Top" Name="pnlTitleBar">

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="\*"/>

<ColumnDefinition Width="Auto"/>

</Grid.ColumnDefinitions>

<Grid.Resources>

<Style TargetType="TextBlock">

<Setter Property="FontFamily" Value="Consolas"/>

<Setter Property="Foreground" Value="White"/>

<Setter Property="FontSize" Value="24"/>

<Setter Property="FontWeight" Value="Bold"/>

</Style>

</Grid.Resources>

<WrapPanel Margin="10,0,0,0">

<TextBlock Text="Очки:"/>

<TextBlock Name="tbStatusScore" Text="0"/>

</WrapPanel>

<WrapPanel Grid.Column="1">

<TextBlock Text="Скорость:"/>

<TextBlock Name="tbStatusSpeed" Text="0"/>

</WrapPanel>

<Button Grid.Column="2" DockPanel.Dock="Right" Background="Transparent" Foreground="White" FontWeight="Bold" FontSize="20" BorderThickness="0" Name="btnClose" Click="btnClose\_Click" Margin="0,0,10,0">X</Button>

</Grid>

<Border BorderBrush="Black" BorderThickness="5">

<Canvas Name="gameField" ClipToBounds="True" Width="400" Height="400">

<Border BorderBrush="Silver" BorderThickness="2" Width="300" Height="300" Canvas.Left="50" Canvas.Top="50" Name="bdrWelcomeMessage" Panel.ZIndex="1">

<StackPanel Orientation="Vertical" Background="AliceBlue">

<TextBlock FontWeight="Bold" FontSize="50" HorizontalAlignment="Center" Margin="0,20,0,0" Text="Змейка"/>

<TextBlock TextWrapping="Wrap" Margin="20" FontSize="16" Text="Используйте стрелки для управления. Ешьте яблоки. Не врезайтесь в стены и свой хвост" HorizontalAlignment="Center" TextAlignment="Center"/>

<TextBlock FontWeight="Bold" HorizontalAlignment="Center" FontSize="18" Foreground="Maroon" Text="Нажмите пробел, чтобы начать"/>

</StackPanel>

</Border>

<Border BorderBrush="Silver" BorderThickness="2" Width="300" Height="300" Canvas.Left="50" Canvas.Top="50" Name="bdrEndOfGame" Panel.ZIndex="1" Visibility="Collapsed">

<StackPanel Orientation="Vertical" Background="AliceBlue">

<TextBlock HorizontalAlignment="Center" FontSize="24" Margin="0,20,0,0" FontWeight="Bold" Text="Вы проиграли"/>

<TextBlock TextAlignment="Center" FontSize="16" Text="Ваш результат:" Margin="0,10,0,0"/>

<TextBlock Name="tbFinalScore" TextAlignment="Center" FontSize="48" FontWeight="Bold" Foreground="Maroon" Text="0"/>

<TextBlock TextAlignment="Center" FontSize="16" Text="U - сохранить результат" Margin="0,20,0,0"/>

<TextBlock TextAlignment="Center" FontSize="16" Text="Пробел - перезапуск" Margin="0,10,0,0"/>

</StackPanel>

</Border>

</Canvas>

</Border>

</DockPanel>

</Window>

SnakeGame.xaml.cs

using Playzer.SnakeFolder;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Playzer.GameWindows

{

/// <summary>

/// Логика взаимодействия для SnakeGame.xaml

/// </summary>

public partial class SnakeGame : Window

{

private System.Windows.Threading.DispatcherTimer gameTickTimer = new System.Windows.Threading.DispatcherTimer();

private Random rnd = new Random();

private UIElement snakeFood = null;

private SolidColorBrush foodBrush = Brushes.Red;

private int snakeLength;

private int currentScore = 0;

const int snakeSquareSize = 20;

const int snakeStartLength = 3;

const int snakeStartSpeed = 400;

const int snakeSpeedThreshold = 100;

private SolidColorBrush snakeBodyBrush = Brushes.Green;

private SolidColorBrush snakeHeadBrush = Brushes.YellowGreen;

private List<SnakePart> snakeParts = new List<SnakePart>();

public enum SnakeDirection { Left, Right, Up, Down };

private SnakeDirection snakeDirection = SnakeDirection.Right;

private int snakeLenght;

Game gameData = new Game();

Player playerInfo = new Player();

public SnakeGame(Game gameInfo, Player playerData)

{

InitializeComponent();

gameData = gameInfo;

playerInfo = playerData;

gameTickTimer.Tick += GameTickTimer\_Tick;

}

private void Window\_ContentRendered(object sender, EventArgs e)

{

DrawGameArea();

}

private void DrawGameArea()

{

bool doneDrawingBackground = false;

int nextX = 0, nextY = 0;

int rowCounter = 0;

bool nextIsOdd = false;

while (doneDrawingBackground == false)

{

Rectangle rect = new Rectangle

{

Width = snakeSquareSize,

Height = snakeSquareSize,

Fill = nextIsOdd ? Brushes.MidnightBlue : Brushes.MidnightBlue

};

gameField.Children.Add(rect);

Canvas.SetTop(rect, nextY);

Canvas.SetLeft(rect, nextX);

nextIsOdd = !nextIsOdd;

nextX += snakeSquareSize;

if(nextX >= gameField.ActualWidth)

{

nextX = 0;

nextY += snakeSquareSize;

rowCounter++;

nextIsOdd = (rowCounter % 2 != 0);

}

if(nextY >= gameField.ActualHeight)

{

doneDrawingBackground = true;

}

}

}

private void DrawSnake()

{

foreach(SnakePart snakePart in snakeParts)

{

if(snakePart.UiElement == null)

{

snakePart.UiElement = new Rectangle()

{

Width = snakeSquareSize,

Height = snakeSquareSize,

Fill = (snakePart.IsHead ? snakeHeadBrush : snakeBodyBrush)

};

gameField.Children.Add(snakePart.UiElement);

Canvas.SetTop(snakePart.UiElement, snakePart.Position.Y);

Canvas.SetLeft(snakePart.UiElement , snakePart.Position.X);

}

}

}

private void MoveSnake()

{

while(snakeParts.Count >= snakeLenght)

{

gameField.Children.Remove(snakeParts[0].UiElement);

snakeParts.RemoveAt(0);

}

foreach(SnakePart snakePart in snakeParts)

{

(snakePart.UiElement as Rectangle).Fill = snakeBodyBrush;

snakePart.IsHead = false;

}

SnakePart snakeHead = snakeParts[snakeParts.Count - 1];

double nextX = snakeHead.Position.X;

double nextY = snakeHead.Position.Y;

switch(snakeDirection)

{

case SnakeDirection.Left:

nextX -= snakeSquareSize;

break;

case SnakeDirection.Right:

nextX += snakeSquareSize;

break;

case SnakeDirection.Up:

nextY -= snakeSquareSize;

break;

case SnakeDirection.Down:

nextY += snakeSquareSize;

break;

}

snakeParts.Add(new SnakePart()

{

Position = new Point(nextX, nextY),

IsHead = true

});

DrawSnake();

DoCollisionCheck();

}

private void GameTickTimer\_Tick(object sender, EventArgs e)

{

MoveSnake();

}

private void StartNewGame()

{

bdrWelcomeMessage.Visibility = Visibility.Collapsed;

bdrEndOfGame.Visibility = Visibility.Collapsed;

foreach(SnakePart snakeBodyPart in snakeParts)

{

if (snakeBodyPart.UiElement != null)

gameField.Children.Remove(snakeBodyPart.UiElement);

}

snakeParts.Clear();

if (snakeFood != null)

gameField.Children.Remove(snakeFood);

currentScore = 0;

snakeLenght = snakeStartLength;

snakeDirection = SnakeDirection.Right;

snakeParts.Add(new SnakePart() { Position = new Point(snakeSquareSize \* 5, snakeSquareSize \* 5) });

gameTickTimer.Interval = TimeSpan.FromMilliseconds(snakeStartSpeed);

DrawSnake();

DrawSnakeFood();

UpdateGameStatus();

gameTickTimer.IsEnabled = true;

}

private Point GetNextFoodPosition()

{

int maxX = (int)(gameField.ActualWidth / snakeSquareSize);

int maxY = (int)(gameField.ActualHeight / snakeSquareSize);

int foodX = rnd.Next(0, maxX) \* snakeSquareSize;

int foodY = rnd.Next(0, maxY) \* snakeSquareSize;

foreach(SnakePart snakePart in snakeParts)

{

if((snakePart.Position.X == foodX) && (snakePart.Position.Y == foodY))

return GetNextFoodPosition();

}

return new Point(foodX, foodY);

}

private void DrawSnakeFood()

{

Point foodPosition = GetNextFoodPosition();

snakeFood = new Ellipse()

{

Width = snakeSquareSize,

Height = snakeSquareSize,

Fill = foodBrush

};

gameField.Children.Add(snakeFood);

Canvas.SetTop(snakeFood, foodPosition.Y);

Canvas.SetLeft(snakeFood, foodPosition.X);

}

private void Window\_KeyUp(object sender, KeyEventArgs e)

{

SnakeDirection originalSnakeDirection = snakeDirection;

switch(e.Key)

{

case Key.Up:

if(snakeDirection != SnakeDirection.Down)

snakeDirection = SnakeDirection.Up;

break;

case Key.Down:

if (snakeDirection != SnakeDirection.Up)

snakeDirection = SnakeDirection.Down;

break;

case Key.Left:

if (snakeDirection != SnakeDirection.Right)

snakeDirection = SnakeDirection.Left;

break;

case Key.Right:

if (snakeDirection != SnakeDirection.Left)

snakeDirection = SnakeDirection.Right;

break;

case Key.Space:

StartNewGame();

break;

case Key.U:

if (gameTickTimer.IsEnabled == false)

{

var result = MessageBox.Show("Сохранить результат?", "Подтверждение", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

try

{

Score resultSave = new Score();

resultSave.Score1 = currentScore;

resultSave.PlayerID = playerInfo.ID;

resultSave.GameID = gameData.ID;

Entities.GetContext().Score.Add(resultSave);

Entities.GetContext().SaveChanges();

MessageBox.Show("Результат сохранён", "Успех", MessageBoxButton.OK, MessageBoxImage.Information);

}

catch

{

MessageBox.Show("Не удалось сохранить результат", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

}

}

}

break;

}

if (snakeDirection != originalSnakeDirection)

MoveSnake();

}

private void DoCollisionCheck()

{

SnakePart snakeHead = snakeParts[snakeParts.Count - 1];

if((snakeHead.Position.X == Canvas.GetLeft(snakeFood)) && (snakeHead.Position.Y == Canvas.GetTop(snakeFood)))

{

EatSnakeFood();

return;

}

if((snakeHead.Position.Y < 0) || (snakeHead.Position.Y >= gameField.ActualHeight) ||

(snakeHead.Position.X < 0) || (snakeHead.Position.X >= gameField.ActualWidth))

{

EndGame();

}

foreach(SnakePart snakeBodyPart in snakeParts.Take(snakeParts.Count - 1))

{

if ((snakeHead.Position.X == snakeBodyPart.Position.X) && (snakeHead.Position.Y == snakeBodyPart.Position.Y))

EndGame();

}

}

private void EatSnakeFood()

{

snakeLenght++;

currentScore++;

int timerInterval = Math.Max(snakeSpeedThreshold, (int)gameTickTimer.Interval.TotalMilliseconds - (currentScore \* 2));

gameTickTimer.Interval = TimeSpan.FromMilliseconds(timerInterval);

gameField.Children.Remove(snakeFood);

DrawSnakeFood();

UpdateGameStatus();

}

private void UpdateGameStatus()

{

this.tbStatusScore.Text = currentScore.ToString();

this.tbStatusSpeed.Text = gameTickTimer.Interval.TotalMilliseconds.ToString();

}

private void EndGame()

{

gameTickTimer.IsEnabled = false;

tbFinalScore.Text = currentScore.ToString();

bdrEndOfGame.Visibility = Visibility.Visible;

}

private void Window\_MouseDown(object sender, MouseButtonEventArgs e)

{

this.DragMove();

}

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

{

this.Close();

}

}

}

SnakePart.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

namespace Playzer.SnakeFolder

{

public class SnakePart

{

public UIElement UiElement { get; set; }

public Point Position { get; set; }

public bool IsHead { get; set; }

}

}

# ПРИЛОЖЕНИЕ 2

TicTacToe.xaml

<Window x:Class="Playzer.TicTacToeFolder.TicTacToe"

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:Playzer.TicTacToeFolder"

mc:Ignorable="d"

Title="Крестики-нолики" Height="350" Width="280" MinHeight="360" MinWidth="280">

<Grid Background="#090d45">

<Grid HorizontalAlignment="Center">

<Button Name="btnA1" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="0,24,0,0"/>

<Button Name="btnA2" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="85,24,0,0"/>

<Button Name="btnA3" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="170,24,0,0"/>

<Button Name="btnB1" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="0,109,0,0"/>

<Button Name="btnB2" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="85,109,0,0"/>

<Button Name="btnB3" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="170,109,0,0"/>

<Button Name="btnC1" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="0,194,0,0"/>

<Button Name="btnC2" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="85,194,0,0"/>

<Button Name="btnC3" HorizontalAlignment="Left" VerticalAlignment="Top" Width="80" Height="80" IsTabStop="False" Focusable="False" FontSize="36" FontFamily="Adobe Gothic Std B" Click="btn\_Click" MouseEnter="btn\_Enter" MouseLeave="btn\_Leave" Margin="170,194,0,0"/>

</Grid>

<Label Name="lbXWins" Content="X: 0" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="10,280,0,0" FontSize="16" FontWeight="Bold" FontFamily="Arial Black" Foreground="White"/>

<Label Name="lbOWins" Content="O: 0" HorizontalAlignment="Left" VerticalAlignment="Top" Margin="60,280,0,0" FontSize="16" FontWeight="Bold" FontFamily="Arial Black" Foreground="White"/>

<Label Name="lbWinner" Content="Победил X" HorizontalAlignment="Center" VerticalAlignment="Top" Margin="0,19,0,0" FontSize="38" FontFamily="Adobe Fan Heiti Std B" Foreground="White" Background="#090d45" HorizontalContentAlignment="Center" VerticalContentAlignment="Center" Width="270" Height="270" Visibility="Hidden"/>

</Grid>

</Window>

TicTacToe.xaml.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace Playzer.TicTacToeFolder

{

/// <summary>

/// Логика взаимодействия для TicTacToe.xaml

/// </summary>

public partial class TicTacToe : Window

{

private string value = "X";

private int xWins = 0;

private int oWins = 0;

private static readonly Brush DEFAULTBRUSH = new SolidColorBrush(Color.FromArgb(255, 142, 142, 166));

public TicTacToe()

{

InitializeComponent();

}

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

{

Button btn = (Button)sender;

btn.Foreground = Brushes.Black;

btn.IsEnabled = false;

CheckGame();

if (!btnA1.IsEnabled && !btnA2.IsEnabled && !btnA3.IsEnabled &&

!btnB1.IsEnabled && !btnB2.IsEnabled && !btnB3.IsEnabled &&

!btnC1.IsEnabled && !btnC2.IsEnabled && !btnC3.IsEnabled)

GameOver("");

value = value == "X" ? "O" : "X";

}

private void CheckGame()

{

if (btnA1.Content == "X" && btnA2.Content == "X" && btnA3.Content == "X"

|| btnB1.Content == "X" && btnB2.Content == "X" && btnB3.Content == "X"

|| btnC1.Content == "X" && btnC2.Content == "X" && btnC3.Content == "X"

|| btnA1.Content == "X" && btnB1.Content == "X" && btnC1.Content == "X"

|| btnA2.Content == "X" && btnB2.Content == "X" && btnC2.Content == "X"

|| btnA3.Content == "X" && btnB3.Content == "X" && btnC3.Content == "X"

|| btnA1.Content == "X" && btnB2.Content == "X" && btnC3.Content == "X"

|| btnA3.Content == "X" && btnB2.Content == "X" && btnC1.Content == "X")

{

GameOver("X");

}

else if (btnA1.Content == "O" && btnA2.Content == "O" && btnA3.Content == "O"

|| btnB1.Content == "O" && btnB2.Content == "O" && btnB3.Content == "O"

|| btnC1.Content == "O" && btnC2.Content == "O" && btnC3.Content == "O"

|| btnA1.Content == "O" && btnB1.Content == "O" && btnC1.Content == "O"

|| btnA2.Content == "O" && btnB2.Content == "O" && btnC2.Content == "O"

|| btnA3.Content == "O" && btnB3.Content == "O" && btnC3.Content == "O"

|| btnA1.Content == "O" && btnB2.Content == "O" && btnC3.Content == "O"

|| btnA3.Content == "O" && btnB2.Content == "O" && btnC1.Content == "O")

{

GameOver("O");

}

}

private void GameOver(string whoWin)

{

if (lbWinner.Visibility == Visibility.Visible)

return;

if (whoWin == "X")

{

lbWinner.Content = "Победил X";

lbXWins.Content = $"X: {++xWins}";

}

else if (whoWin == "O")

{

lbWinner.Content = "Победил O";

lbOWins.Content = $"O: {++oWins}";

}

else

lbWinner.Content = "Ничья";

lbWinner.Visibility = Visibility.Visible;

AutoRestart();

}

private async void AutoRestart()

{

await Task.Delay(1000);

lbWinner.Visibility = Visibility.Hidden;

ResetButtons();

}

private void ResetButtons()

{

ResetButton(btnA1);

ResetButton(btnA2);

ResetButton(btnA3);

ResetButton(btnB1);

ResetButton(btnB2);

ResetButton(btnB3);

ResetButton(btnC1);

ResetButton(btnC2);

ResetButton(btnC3);

}

private void ResetButton(Button btn)

{

btn.Content = "";

btn.IsEnabled = true;

btn.Foreground = DEFAULTBRUSH;

}

private bool IsWin(Button btn1, Button btn2, Button btn3) =>

!btn1.IsEnabled && btn1.Content == btn2.Content && btn1.Content == btn3.Content;

private void btn\_Enter(object sender, MouseEventArgs e)

{

Button btn = (Button)sender;

btn.Content = value;

}

private void btn\_Leave(object sender, MouseEventArgs e)

{

Button btn = (Button)sender;

if (btn.IsEnabled)

btn.Content = "";

}

}

}