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 Гордов Максим Олегович (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 3](#_Toc136432510)

[ГЛАВА 1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ 5](#_Toc136432511)

[1.1. Описание предметной области 5](#_Toc136432512)

[1.3. Постановка задачи 10](#_Toc136432513)

[1.4. Характеристика инструментальных средств 11](#_Toc136432514)

[1.5. Обоснование выбора инструментальных средств разработки 14](#_Toc136432515)

[ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ 15](#_Toc136432516)

[2.1. Анализ требований и определение спецификаций ПО 15](#_Toc136432517)

[2.2. Проектирование программного обеспечения 19](#_Toc136432518)

[2.3. Разработка программного обеспечения 22](#_Toc136432519)

[2.4. Отладка и тестирование программы 32](#_Toc136432520)

[2.5. Руководство по использованию программы 33](#_Toc136432521)

[ЗАКЛЮЧЕНИЕ 35](#_Toc136432522)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 36](#_Toc136432523)

[ПРИЛОЖЕНИЕ 37](#_Toc136432524)

# 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 дел было решено сделать развлекательный мультиплатформенный игровой сервис. В приложении предполагаются простые игры, которые будут нести развлекательную функцию. В соревновательных целях будет добавлена таблица с наивысшими результатами среди всех зарегистрированных пользователей.

Одна из игр будет представлять из себя уже ставшую классикой змейку. По мере поедания яблок, появляющихся на игровом поле, размер змейки, которой управляет пользователь будет увеличиваться. При столкновении головы змейки со стенкой 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 ВКР – разработка мобильного приложения 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:

– определение требований,

– проектирование и создание базы данных,

– интеграция API для работы с базой данных,

– проектирование и реализация приложения,

– написание руководства пользователя.

Объект исследования – развлекательный мультиплатформенный игровой сервис.

Предмет исследования – технологии разработки мобильных приложений.

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

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

Для реализации проекта будет использоваться среда разработки Android Studio. Для написания кода приложения и API будет использоваться язык программирования Kotlin. Для хранения и управления данными будет использоваться система управления базами данных Firebase.

# ГЛАВА 1. ПРЕДПРОЕКТНОЕ ИССЛЕДОВАНИЕ

## 1.1. Описание предметной области

Предметной областью для данного проекта являются развлекательные игровые сервисы. Предполагается, что функционал разрабатываемого приложения поможет пользователям отдохнуть после тяжёлого рабочего дня, расслабиться и не нагружать себя. Вводится также 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 и разных социальных групп, так что 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,

– возможность сохранения результата.

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 проведен анализ каждой из этих программ по нескольким критериям, таким как:

* Сохранение прогресса: будет оценено возможность сохранения рекордов в играх, скаченных по возможности вместе с приложением.
* Удобство использования: будет оценено удобство использования каждой программы, включая интерфейс пользователя и доступность функций.
* Стоимость: будет оценена стоимость каждой программы, включая цену за использование и возможные дополнительные расходы.
* Неудобство реклам: будет рассмотрен частость рекламы и ее отрицательного воздействия на возможность пользования.
* Дополнительные функции: будет рассмотрено наличие дополнительных функций в каждой программе, таких как возможность открыть скаченные игры.

Программный продукт «Google Play Игры» –официальное приложение от компании Google, в нем есть множество игр, среди них несколько классических. Все же остальные игры работают по специальной системе быстрого запуска без скачивания на телефон.

Сохранение прогресса: у программы нет сохранения очков в мини-играх, есть только очки, которые добавляются из скаченных заранее игр, и то, если они предусмотрены разработчиками.

Совместимость: Требованная OC Android11 и более поздние версии

Стоимость: Бесплатно.

Неудобство реклам: Пользователи жалуются на рекламу, которая перегораживает контент игр и не убирается.

Удобство использования: Интерфейс интуитивно понятен.

Дополнительные функции: Поддержка открытия скаченных игр.

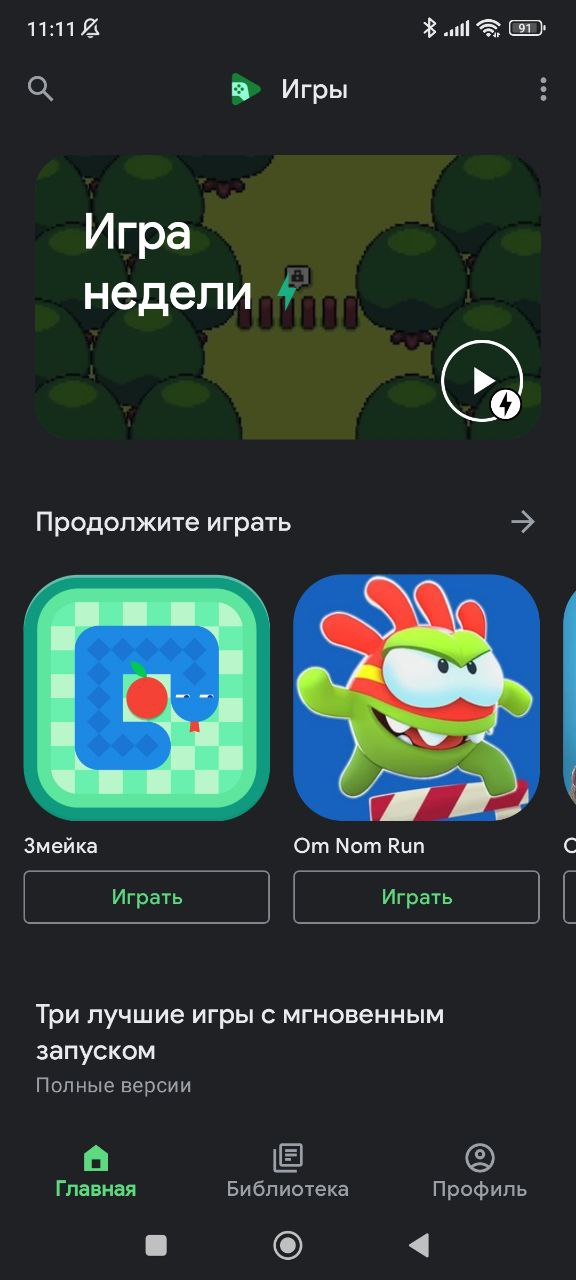


Рисунок 1. Скриншот программы-аналога «Google Play Игры»

Программный продукт «Яндекс игры»

Яндекс игры - это программа с каталогом игр которые они запускают онлайн с браузерной версии Яндекс игры.

Сохранение прогресса: В некоторых играх есть сохранения очков, но это зависит от разработчиков игр.

Совместимость: Требованная OC Android11 и более поздние версии

Стоимость: Бесплатно.

Неудобство реклам: Реклама появляется перед началом каждой игры. люди жалуются на рекламу, которая перегораживает контент игр и не убирается.

Удобство использования: Интерфейс пугает своей загруженностью из-за этого не сразу понятно, что игра, а что рекламный баннер.

Дополнительные функции: Возможность отключения рекламы за деньги, а также покупки внутри игровых благ для игр на этой площадке.

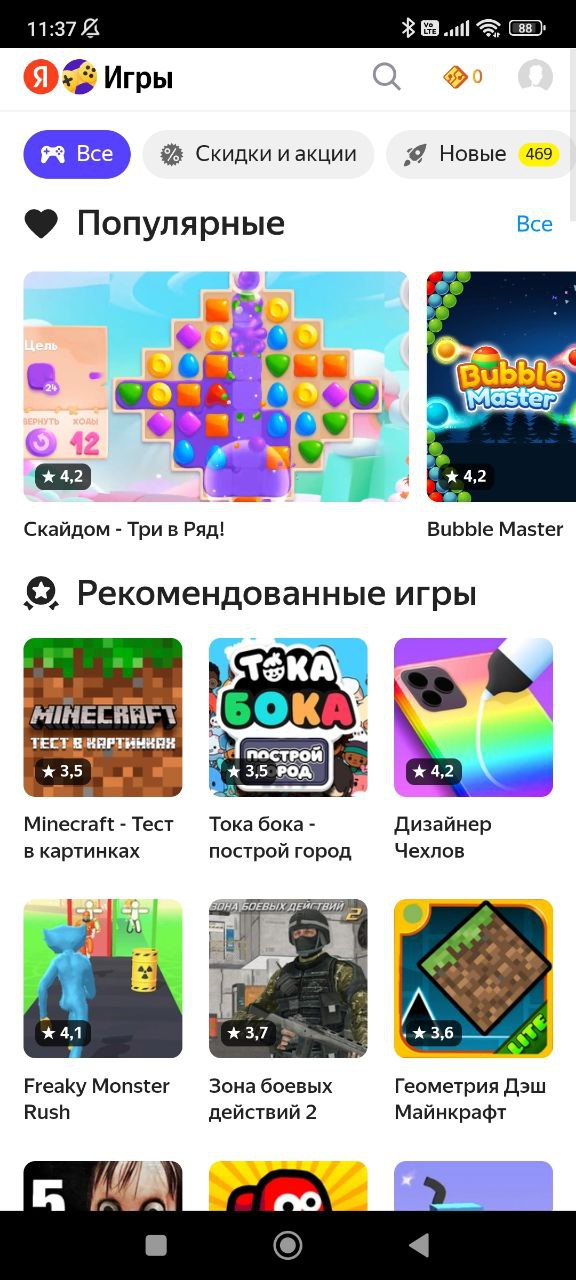


Рисунок 2. Скриншот программы-аналога «Яндекс игры»

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

Таблица 1. Результаты сравнительного анализа

|  |  |  |
| --- | --- | --- |
| Название Приложения:  Функции: | Google Play Игры | Яндекс игры |
| Сохранение прогресса | - | +/- |
| Совместимость | + | + |
| Стоимость(бесплатно) | + | + |
| Неудобство реклам | +/- | + |
| Удобство использования | + | - |
| Дополнительные функции | + | + |

Тесты проводились на 1 устройстве, подключенном к сети интернет. Были проведены тесты в нескольких похожих играх.

На основании результатов проведённого анализа функционала можно сделать вывод о том, что разрабатываемый проект по качеству, скорости и функционалу не уступает программам-аналогам.

Преимуществами разрабатываемого программного решения «Плейзер» будет контент без рекламы, сохранение прогресса, понятный интерфейс и игры для всех возрастов.

## 1.3. Постановка задачи

При первом запуске приложения пользователя будет отображаться главное окно, в котором он сможет войти, введя данные учетной записи или перейти на страницу регистрации, где он сможет зарегистрировать новую учетную запись.

После входа перед пользователями открывается новое окно со списком игр, которые он может запустить.

Далее после выбора игры откроется окно игры, в котором отображается кнопка играть и топ игроков.

После запускается выбранной игры пользователем будет протекать геймплей для 2 людей или 1 человека.

Эксплуатационные требования для разрабатываемого приложения:

операционная система: Android.

Минимальная система: требованная OC Android11 и более поздние версии

Требуемые разрешения: доступ в интернет.

## 1.4. Характеристика инструментальных средств

В процессе разработки программного продукта были использованы следующие языки программирования:

Kotlin – это объектно-ориентированный язык программирования, созданный в компании JetBrains. Его разработали в 2011 году на замену Java, который в компании считали чересчур многословным. Новый язык получился на 40% компактнее предшественника, что помогло ускорить работу над основным продуктом JetBrains — средой разработки IntelliJ IDEA. При этом Kotlin полностью совместим с Java, потому что запускается на его виртуальной машине JVM.

Благодаря совместимости с JVM, Kotlin можно запустить на любых серверах, где работает Java. Поэтому его часто используют для создания бэкенда — той части систем, которая выполняется на сервере и не видна обычному пользователю.

Firebase — это платформа для разработки мобильных и веб-приложений, которая была приобретена компанией Google в 2014 году. Firebase предоставляет множество инструментов и сервисов для разработки и развертывания приложений, таких как аутентификация пользователей, база данных в реальном времени, облачное хранилище, обработка сообщений, аналитика, тестирование и др.

Firebase предоставляет базу данных типа NoSQL, называемую Realtime Database. Realtime Database - это гибкая и масштабируемая база данных, которая может хранить и синхронизировать данные в режиме реального времени между различными клиентами приложения. Realtime Database использует модель JSON для хранения данных и позволяет быстро читать и записывать данные, что делает ее отличным выбором для приложений, требующих быстрой и эффективной обработки данных в режиме реального времени. Кроме того, Firebase также предоставляет другую базу данных, Firestore, которая также является NoSQL-базой данных и предлагает более мощный и гибкий запрос данных и другие функции.

NoSQL - это аббревиатура от "Not Only SQL" (не только SQL), что означает, что NoSQL - это широкий класс баз данных, которые отличаются от реляционных баз данных (SQL) в нескольких аспектах. NoSQL-базы данных предоставляют гибкую модель данных, не требуют жесткой схемы данных и могут обрабатывать большие объемы структурированных, полуструктурированных и неструктурированных данных. Также в процессе разработки были использованы следующие программные средства:

Android Studio — это интегрированная среда разработки (IDE) производства Google, с помощью которой разработчикам становятся доступны инструменты для создания приложений на платформе Android OS. Android Studio можно установить на Windows, Mac и Linux.

Среда Android Studio предназначена как для небольших команд разработчиков мобильных приложений (даже в количестве одного человека), или же крупных международных организаций с Git или другими подобными системами управления версиями. Опытные разработчики смогут выбрать инструменты, которые больше подходят для масштабных проектов. Решения для Android разрабатываются в Android Studio с использованием Java, Kotlin или C++. В основе рабочего процесса Android Studio заложен концепт непрерывной интеграции, позволяющий сразу же обнаруживать имеющиеся проблемы. Продолжительная проверка кода обеспечивает возможность эффективной обратной связи с разработчиками. Такая опция позволяет быстрее опубликовать версию мобильного приложения в Google Play.

Android Studio совместима с платформой Google App Engine для быстрой интеграции в облаке новых API и функций. В среде разработки вы найдете различные API, такие как Google Play, Android Pay и Health. Присутствует поддержка всех платформ Android, начиная с Android 1.6. Есть варианты Android, которые существенно отличаются от версии Google Android. Самая популярная из них — это Amazon Fire OS. В Android Studio можно создавать APK для этой ОС.

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 с сохранением очков 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, высокую точность распознавания текста и качество перевода.

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 «KIP Translator». 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 Kotlin, так как этот язык обладает богатым набором библиотек классов, которые позволяют создавать приложения быстро и эффективно.

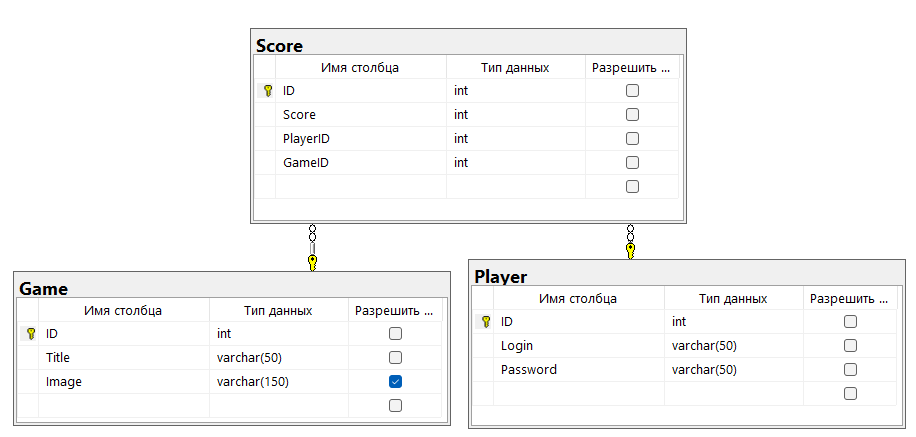
# ГЛАВА 2. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 2.1. Анализ требований и определение спецификаций ПО

Спецификации данного проекта определяются следующими диаграммами из списка:

* 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».

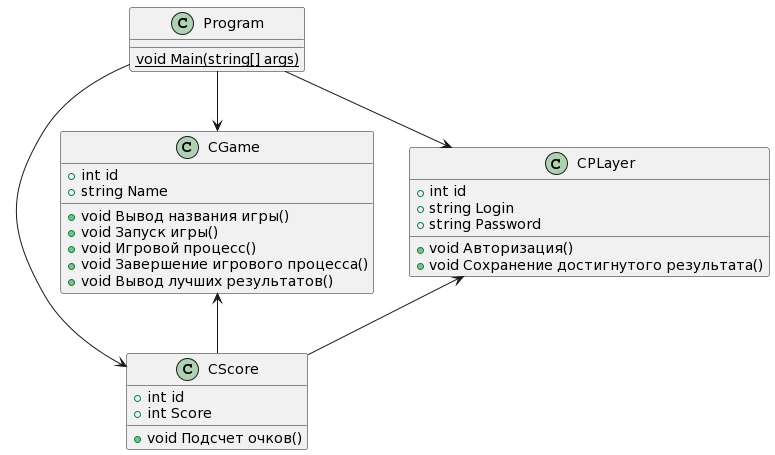
  
Рисунок 4. Диаграмма классов

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

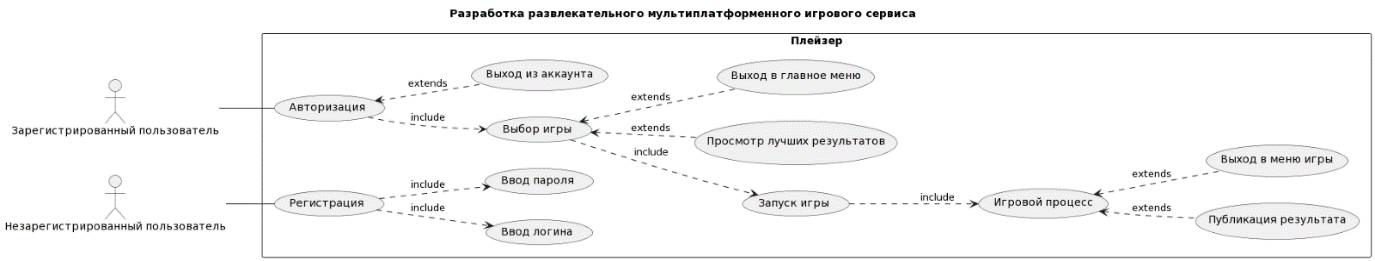
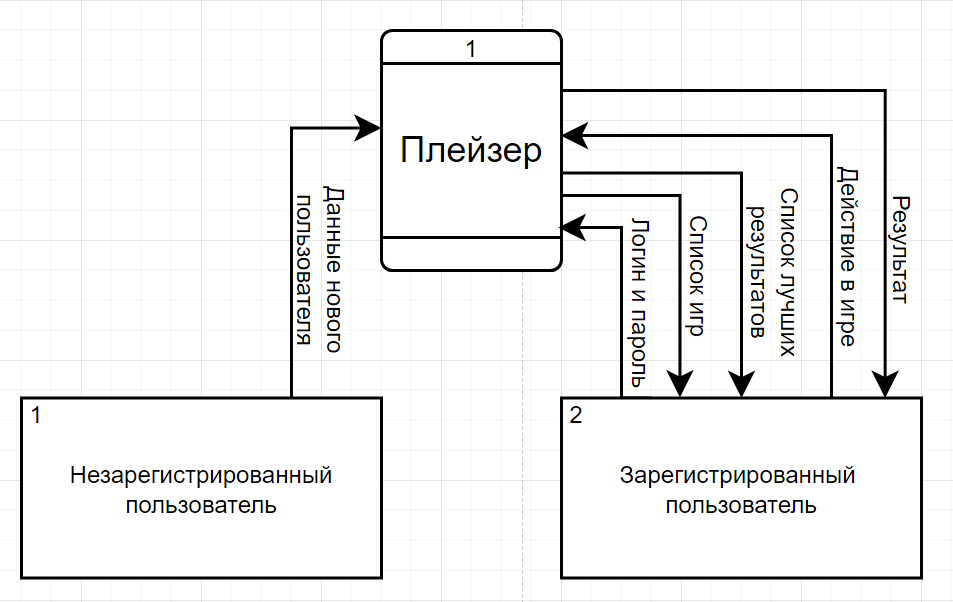
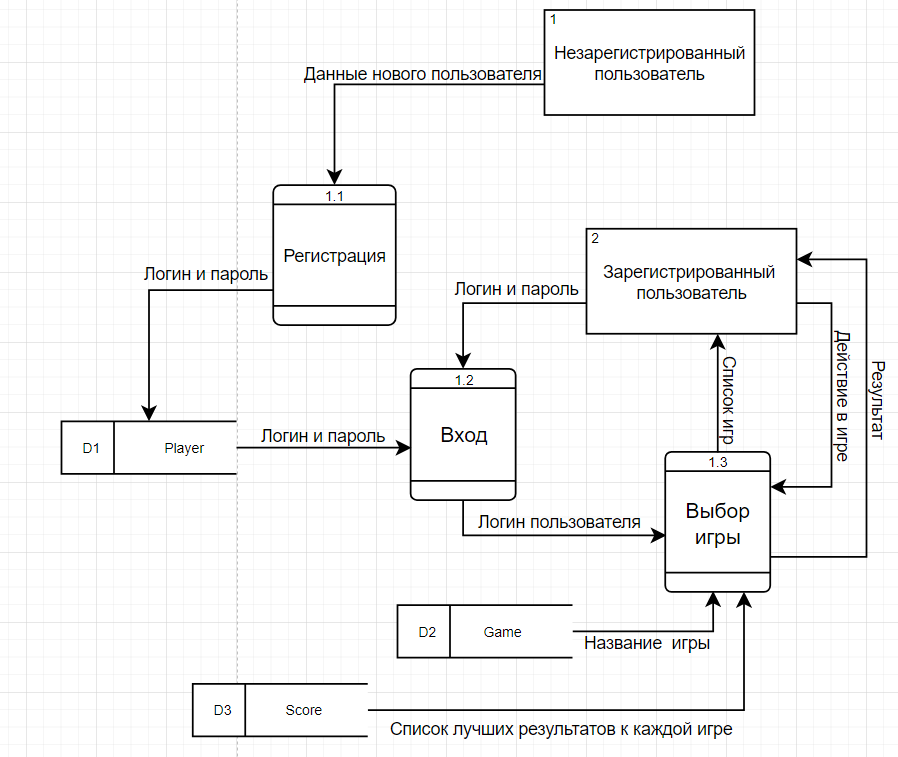


Рисунок 5. Диаграмма вариантов использования

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

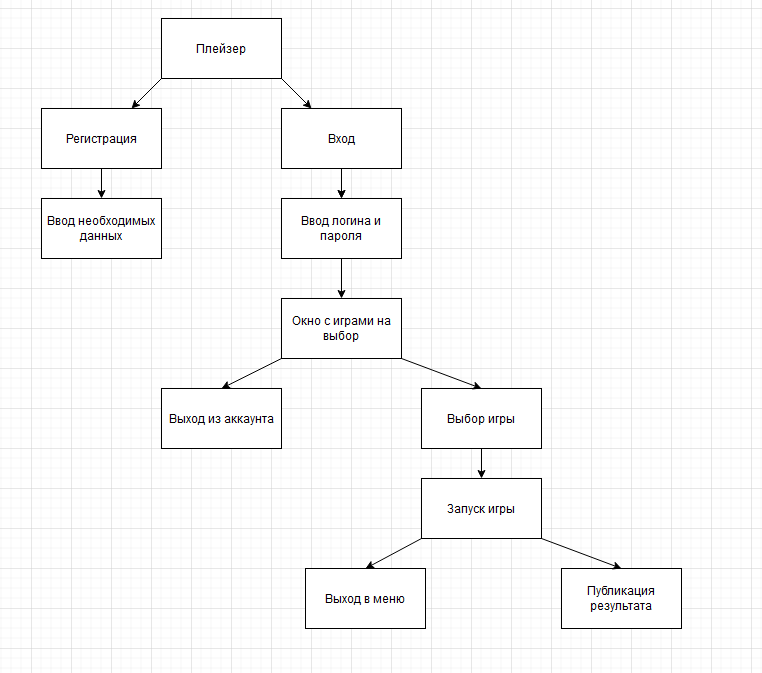
Рисунок 6. Диаграмма потоков данных верхнего уровня

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

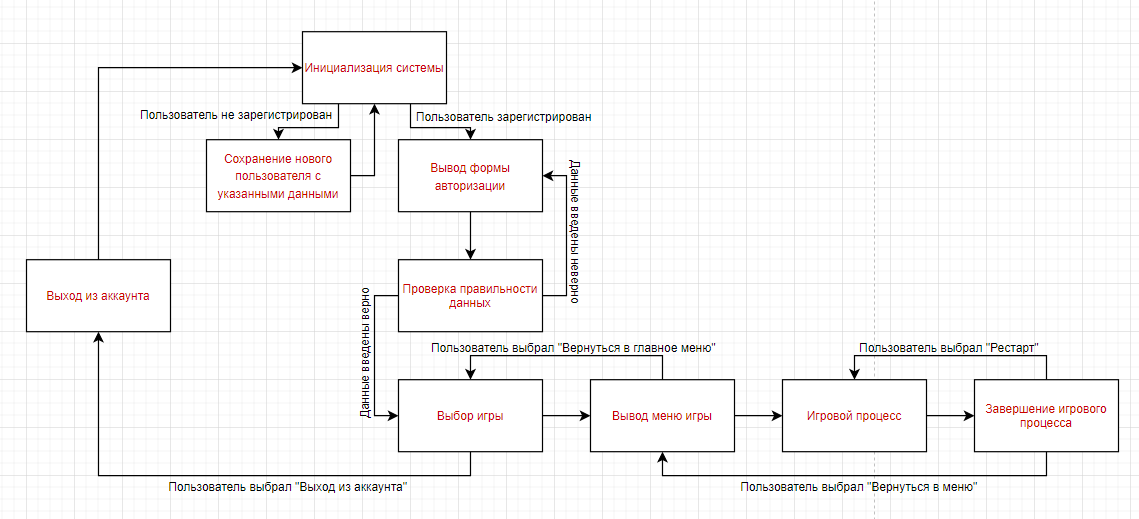
  
Рисунок 7. Детализированная диаграмма потоков данных

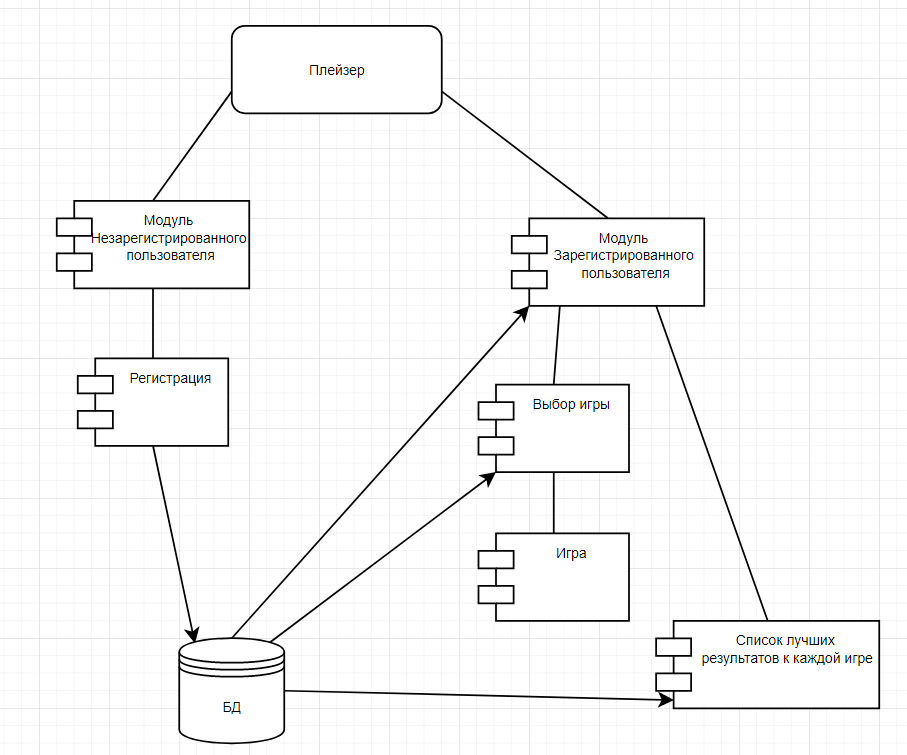
## 2.2. Проектирование программного обеспечения

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.

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

Главная страница (Авторизация):

При запуске приложения «Плейзер» главное окно, на котором представлены 2 кнопки, 2 EditText и 3 text view. Первая из кнопок предназначена для входа, вторая для перехода на страницу регистрации (рисунок 11).

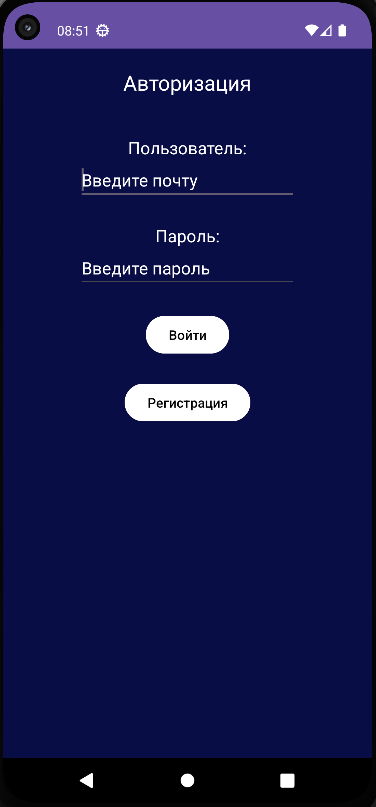


Рисунок 11. Главное окно приложения

Верстка и программный код страницы 11

XML:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#090d45"  
 android:orientation="vertical"  
 tools:context=".MainActivity">  
 <RelativeLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.0">  
 <TextView  
 android:id="@+id/Login"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginTop="21dp"  
 android:layout\_centerHorizontal="true"  
 android:inputType="textCapWords"  
 android:minHeight="48dp"  
 android:textColor="@android:color/white"  
 android:text="Авторизация"  
 android:textAppearance="?android:attr/textAppearanceLarge" />  
 <Button  
 android:id="@+id/button\_login"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:backgroundTint="@android:color/white"  
 android:textColor="@android:color/black"  
 android:layout\_below="@id/edit\_password"  
 android:layout\_marginTop="24dp"  
 android:text="Войти" />  
 <Button  
 android:id="@+id/button\_reg"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:backgroundTint="@android:color/white"  
 android:textColor="@android:color/black"  
 android:layout\_below="@id/button\_login"  
 android:layout\_marginTop="24dp"  
 android:text="Регистрация" />  
 <EditText  
 android:id="@+id/edit\_password"  
 android:layout\_width="233dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/password\_text"  
 android:layout\_centerHorizontal="true"  
 android:textColor="@android:color/white"  
 android:textColorHint="@android:color/white"  
 android:hint="Введите пароль"  
 android:inputType="textPassword" />  
 <TextView  
 android:id="@+id/password\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/edit\_user"  
 android:layout\_centerHorizontal="true"  
 android:text="Пароль:"  
 android:textColor="@android:color/white"  
 android:layout\_marginTop="24dp"  
 android:textAppearance="?android:attr/textAppearanceMedium" />  
 <EditText  
 android:id="@+id/edit\_user"  
 android:layout\_width="233dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/user\_text"  
 android:textColor="@android:color/white"  
 android:textColorHint="@android:color/white"  
 android:layout\_centerHorizontal="true"  
 android:hint="Введите почту">  
 <requestFocus />  
 </EditText>  
 <TextView  
 android:id="@+id/number\_of\_attempts"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignRight="@+id/user\_text"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentBottom="true"  
 android:visibility="invisible" />  
 <TextView  
 android:id="@+id/login\_locked"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentBottom="true"  
 android:textAppearance="?android:attr/textAppearanceMedium"  
 android:visibility="invisible" />  
 <TextView  
 android:id="@+id/user\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="24dp"  
 android:layout\_below="@+id/Login"  
 android:layout\_centerHorizontal="true"  
 android:textColor="@android:color/white"  
 android:text="Пользователь:"  
 android:textAppearance="?android:attr/textAppearanceMedium" />  
 </RelativeLayout>  
</androidx.constraintlayout.widget.ConstraintLayout>

kt:

package ru.maximus.playzer  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.lifecycle.Observer  
import androidx.lifecycle.lifecycleScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
class MainActivity : AppCompatActivity() {  
 private lateinit var firebaseAuth: FirebaseAuthentication  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_main)  
 firebaseAuth = FirebaseAuthentication.get()  
 if (firebaseAuth.getSignedInUser() != null) {  
 updateUI()  
 }  
 firebaseAuth.signInResult.observe(this,  
 Observer {  
 it?.let {  
 if (it) {  
 updateUI()  
 } else {  
 showToastFail()  
 }  
 }  
 })  
 val reg\_but = findViewById<Button>(R.id.button\_reg)  
 reg\_but.setOnClickListener {  
 val intent = Intent(this, Auth::class.java)  
 startActivityForResult(intent,0)  
 }  
 val username = findViewById<EditText>(R.id.edit\_user)  
 val password = findViewById<EditText>(R.id.edit\_password)  
 val loginButton = findViewById<Button>(R.id.button\_login)  
 loginButton.setOnClickListener {  
 lifecycleScope.launch(Dispatchers.IO){  
 firebaseAuth.signIn(username.text.toString(), password.text.toString())  
 }  
 }  
 }  
 private fun showToastFail() {  
 Toast.makeText(this, "Не верный логин или пароль", Toast.LENGTH\_LONG).show()  
 }  
 fun updateUI() {  
 val intent = Intent(this, Menu::class.java)  
 startActivity(intent)  
 finish()  
 }  
}

На странице регистрации (рис 12) есть:

Верхнее текстовое поле предназначено для ввода имени аккаунта.

Нижнее текстовое поля предназначено для ввода почты.

Нижнее текстовое поля предназначено для ввода пароля.

Нижнее текстовое поля предназначено для повторного ввода пароля.

Кнопка «Регистрация» при нажатии переводит пользователя на страницу Авторизации и добавляет его аккаунт в бд.

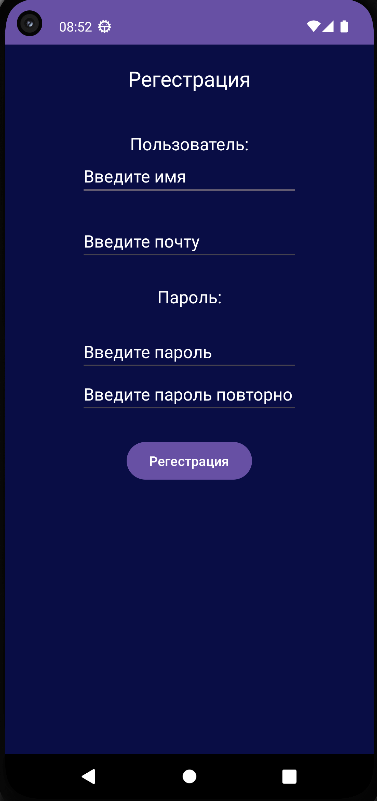


Рисунок 12. Страница регистрации

Верстка и программный код страницы 12

Xml:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#090d45"  
 android:orientation="vertical"  
 tools:context=".Auth">  
 <RelativeLayout  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="1.0"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:layout\_constraintVertical\_bias="0.0">  
 <TextView  
 android:id="@+id/Login"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_marginTop="21dp"  
 android:layout\_centerHorizontal="true"  
 android:inputType="textCapWords"  
 android:minHeight="48dp"  
 android:text="Регестрация"  
 android:textColor="@android:color/white"  
 android:textAppearance="?android:attr/textAppearanceLarge" />  
 <Button  
 android:id="@+id/button\_auth"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_below="@id/repasword"  
 android:layout\_marginTop="24dp"  
 android:text="Регестрация" />  
 <EditText  
 android:id="@+id/edit\_password"  
 android:layout\_width="233dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/password\_text"  
 android:layout\_centerHorizontal="true"  
 android:hint="Введите пароль"  
 android:textColor="@android:color/white"  
 android:textColorHint="@android:color/white"  
 android:layout\_marginTop="24dp"  
 android:inputType="textPassword" />  
  
 <EditText  
 android:id="@+id/repasword"  
 android:layout\_width="233dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/edit\_password"  
 android:layout\_centerHorizontal="true"  
 android:textColor="@android:color/white"  
 android:textColorHint="@android:color/white"  
 android:hint="Введите пароль повторно"  
 android:inputType="textPassword" />  
 <TextView  
 android:id="@+id/password\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/edit\_mail"  
 android:layout\_centerHorizontal="true"  
 android:text="Пароль:"  
 android:textColor="@android:color/white"  
 android:layout\_marginTop="24dp"  
 android:textAppearance="?android:attr/textAppearanceMedium" />  
 <EditText  
 android:id="@+id/edit\_user"  
 android:layout\_width="233dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/user\_text"  
 android:textColor="@android:color/white"  
 android:textColorHint="@android:color/white"  
 android:layout\_centerHorizontal="true"  
 android:hint="Введите имя">  
 <requestFocus />  
 </EditText>  
 <EditText  
 android:id="@+id/edit\_mail"  
 android:layout\_width="233dp"  
 android:layout\_marginTop="24dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_below="@+id/edit\_user"  
 android:textColor="@android:color/white"  
 android:textColorHint="@android:color/white"  
 android:layout\_centerHorizontal="true"  
 android:hint="Введите почту"/>  
 <TextView  
 android:id="@+id/number\_of\_attempts"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignRight="@+id/user\_text"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentBottom="true"  
 android:visibility="invisible" />  
 <TextView  
 android:id="@+id/login\_locked"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentBottom="true"  
 android:textAppearance="?android:attr/textAppearanceMedium"  
 android:visibility="invisible" />  
 <TextView  
 android:id="@+id/user\_text"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="24dp"  
 android:layout\_below="@+id/Login"  
 android:layout\_centerHorizontal="true"  
 android:textColor="@android:color/white"  
 android:text="Пользователь:"  
 android:textAppearance="?android:attr/textAppearanceMedium" />  
 </RelativeLayout>  
</androidx.constraintlayout.widget.ConstraintLayout>

Kt:

package ru.maximus.playzer  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import android.widget.EditText  
import android.widget.Toast  
import androidx.lifecycle.Observer  
import androidx.lifecycle.lifecycleScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
class Auth : AppCompatActivity() {  
 private lateinit var firebaseAuth: FirebaseAuthentication  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_auth)  
 FirebaseAuthentication.initialize()  
 firebaseAuth = FirebaseAuthentication.get()  
 firebaseAuth.signInResult.observe(this,  
 Observer {  
 it?.let {  
 if (it) {  
 updateUI()  
 } else {  
 showToastFail()  
 }  
 }  
 })  
 val username = findViewById<EditText>(R.id.edit\_user)  
 val mail = findViewById<EditText>(R.id.edit\_mail)  
 val password = findViewById<EditText>(R.id.edit\_password)  
 val repassword = findViewById<EditText>(R.id.repasword)  
 val loginButton = findViewById<Button>(R.id.button\_auth)  
 loginButton.setOnClickListener{  
 if (password.text.count() >= 6){  
 if(password.text.toString() == repassword.text.toString()) {  
 lifecycleScope.launch(Dispatchers.IO) {  
 firebaseAuth.signUp(mail.text.toString(), password.text.toString(), username.text.toString())  
 updateUI()  
 }  
 }  
 else{Toast.makeText(this, "Пароли не совподают", Toast.LENGTH\_LONG).show()}  
 }  
 else{Toast.makeText(this, "Пароль меньше чем 6 символов", Toast.LENGTH\_LONG).show()}  
  
 }  
 }  
  
 private fun showToastFail() {  
 Toast.makeText(this, "Проверьте свою почту", Toast.LENGTH\_LONG).show()  
 }  
  
 fun updateUI() {  
 finish()  
 }  
}

На странице регистрации (рис 13) есть 3 кнопки:

Кнопка «Крестики нолики» при нажатии переводит пользователя на страницу меню игры крестики нолики.

Кнопка «Змейка» при нажатии переводит пользователя на страницу меню игры змейка.

Кнопка «Выйти» при нажатии переводит пользователя на страницу Авторизации и выходит из аккаунта.

Нижнее текстовое поля предназначено для вывода имени аккаунта.

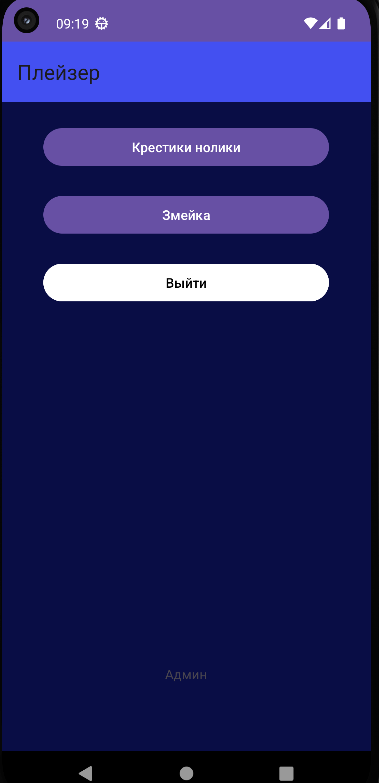


Рисунок 13. Меню

Верстка и программный код страницы 13

xaml:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#090d45"  
 android:orientation="vertical"  
 tools:context=".Menu">  
 <com.google.android.material.appbar.AppBarLayout  
 android:id="@+id/appBarLayout"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:fitsSystemWindows="true"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <com.google.android.material.appbar.MaterialToolbar  
 android:background="#4350F1"  
 android:id="@+id/toolbar"  
 android:layout\_width="match\_parent"  
 android:layout\_height="?attr/actionBarSize"  
 android:titleTextColor="#FFFFFF"  
 app:title="@string/app\_name" />  
  
 </com.google.android.material.appbar.AppBarLayout>  
  
 <TextView  
 android:id="@+id/nameView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginBottom="72dp"  
  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.498"  
 app:layout\_constraintStart\_toStartOf="parent" />  
  
 <Button  
 android:id="@+id/button"  
 android:layout\_width="304dp"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="24dp"  
 android:text="Крестики нолики"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.495"  
 app:layout\_constraintStart\_toStartOf="parent"  
  
 app:layout\_constraintTop\_toBottomOf="@+id/appBarLayout"/>  
  
 <Button  
 android:id="@+id/Snake"  
 android:layout\_width="304dp"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="24dp"  
 android:text="Змейка"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.495"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/button" />  
  
 <Button  
 android:id="@+id/signout"  
 android:layout\_width="304dp"  
 android:layout\_height="48dp"  
 android:layout\_marginTop="24dp"  
 android:text="Выйти"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.495"  
 android:backgroundTint="@android:color/white"  
 android:textColor="@android:color/black"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/ Snake " />  
</androidx.constraintlayout.widget.ConstraintLayout>

kt:

package ru.maximus.playzer  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import android.widget.TextView  
import android.widget.Toolbar  
import androidx.lifecycle.Observer  
import androidx.lifecycle.lifecycleScope  
import com.google.firebase.auth.FirebaseAuth  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import ru.maximus.playzer.SnakeGame.FirstPSnake  
import ru.maximus.playzer.TicGame.FirstPageTicGame  
import ru.maximus.playzer.trickmath.FirstPtrickmath  
class Menu : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_menu)  
 val TicTac = findViewById<Button>(R.id.button)  
 TicTac.setOnClickListener {  
 val intent = Intent (this, FirstPageTicGame::class.java)  
 startActivityForResult(intent,0)  
 }  
 val Snake = findViewById<Button>(R.id.Snake)  
 Snake.setOnClickListener{  
 val intent = Intent (this, FirstPSnake::class.java)  
 startActivityForResult(intent,0)  
 }  
 var nameView = findViewById<TextView>(R.id.nameView)  
 val firebase = FirebaseAuthentication.get()  
 firebase.userResult.observe(this,  
 Observer {  
 it?.let {  
 nameView.text = it.name  
 }  
 })  
 if(FirebaseAuth.getInstance().currentUser != null){lifecycleScope.launch(Dispatchers.IO) { firebase.usernam() }}  
 val signout = findViewById<Button>(R.id.signout)  
 signout.setOnClickListener {  
 firebase.signOut()  
 val intent = Intent (this, MainActivity::class.java)  
 startActivity(intent)  
 finish()  
 }  
 }  
}

На рисунке 14 представлено меню игры «Крестики-нолики». Здесь есть кнопка «Играть», она переносит к началу игры.



Рисунок 14. Меню крестики нолики

Верстка и программный код страницы 14

xml:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#090d45"  
 android:orientation="vertical"  
 tools:context=".TicGame.FirstPageTicGame">  
 <TextView  
 android:id="@+id/textView"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginTop="30dp"  
 android:text="Игра для двоих!"  
 android:textColor="@android:color/white"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/imageView2" />  
 <ImageView  
 android:id="@+id/imageView2"  
 android:layout\_width="426dp"  
 android:layout\_height="209dp"  
 android:layout\_marginTop="4dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.533"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent"  
 app:srcCompat="@drawable/tictac" />  
 <Button  
 android:id="@+id/button2"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="20dp"  
 android:backgroundTint="@android:color/white"  
 android:textColor="@android:color/black"  
 android:text="Играть"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/textView" />  
</androidx.constraintlayout.widget.ConstraintLayout>

kt:

package ru.maximus.playzer.TicGame  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import ru.maximus.playzer.R  
  
class FirstPageTicGame : AppCompatActivity() {  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_first\_page\_tic\_game)  
 var button = findViewById<Button>(R.id.button2)  
 button.setOnClickListener {  
 val intent = Intent (this, TicTacTaoGame::class.java)  
 startActivityForResult(intent,0)  
 }  
 }  
}

После нажатия играть открывается страница игры (Рисунок 15), где есть игровое поле и кнопка рестарт. Эта игра сделана для 2 людей.

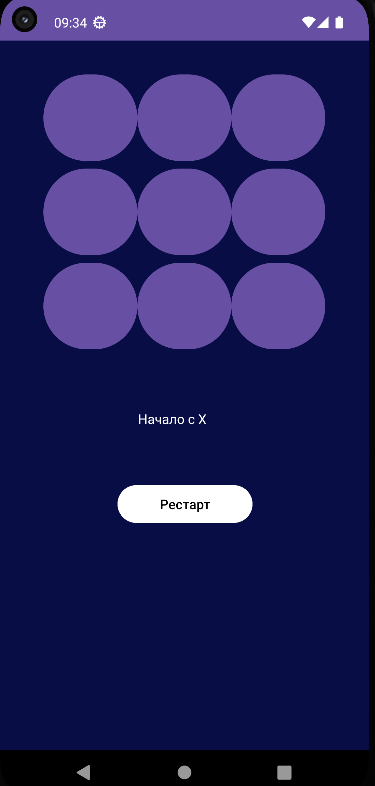


Рисунок 15. Игра Крестики нолики

Верстка и программный код страницы 15

xml:

<?xml version="1.0" encoding="utf-8"?>  
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#090d45"  
 android:orientation="vertical"  
 tools:context=".TicGame.TicTacTaoGame">  
 <GridLayout  
 android:id="@+id/grid\_layout"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentTop="true"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="32dp"  
 android:columnCount="3"  
 android:rowCount="3">  
 <Button  
 android:id="@+id/button\_00"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 0 0"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_01"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 0 1"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_02"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 0 2"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_10"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 1 0"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_11"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 1 1"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_12"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 1 2"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_20"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 2 0"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_21"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 2 1"  
 android:textSize="40sp" />  
 <Button  
 android:id="@+id/button\_22"  
 android:layout\_width="100dp"  
 android:layout\_height="100dp"  
 android:contentDescription="Button 2 2"  
 android:textSize="40sp" />  
 </GridLayout>  
 <Button  
 android:id="@+id/new\_game\_button"  
 android:layout\_width="162dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_alignParentBottom="true"  
 android:layout\_marginStart="125dp"  
 android:layout\_marginEnd="124dp"  
 android:backgroundTint="@android:color/white"  
 android:textColor="@android:color/black"  
 android:layout\_marginBottom="237dp"  
 android:text="Рестарт" />  
 <TextView  
 android:id="@+id/playernow"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_alignParentStart="true"  
 android:layout\_alignParentTop="true"  
 android:layout\_alignParentEnd="true"  
 android:layout\_marginStart="147dp"  
 android:layout\_marginTop="392dp"  
 android:layout\_marginEnd="147dp"  
 android:textColor="@android:color/white"  
 android:text="Начало с X"  
 />  
</RelativeLayout>

kt:

package ru.maximus.playzer.TicGame  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import ru.maximus.playzer.R  
class TicTacTaoGame : AppCompatActivity() {  
 private lateinit var game: TicTacLogic  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_tic\_tac\_tao\_game)  
 game = TicTacLogic(this)  
 // Set click listeners for the game board buttons  
 val buttons = Array(3) { arrayOfNulls<Button>(3) }  
 for (i in 0..2) {  
 for (j in 0..2) {  
 val buttonId = resources.getIdentifier("button\_$i$j", "id", packageName)  
 buttons[i][j] = findViewById(buttonId)  
 buttons[i][j]?.setOnClickListener {  
 if (!game.isGameEnded() && game.makeMove(i, j)) {  
 updateBoard(buttons)  
 //game.checkForWin()  
 }  
 }  
 }  
 }  
 // Set click listener for the "New Game" button  
 val newGameButton = findViewById<Button>(R.id.new\_game\_button)  
 newGameButton.setOnClickListener {  
 game.resetGame()  
 updateBoard(buttons)  
 }  
 }  
 private fun updateBoard(buttons: Array<Array<Button?>>) {  
 for (i in 0..2) {  
 for (j in 0..2) {  
 buttons[i][j]?.text = game.board[i][j]  
 }  
 }  
 }  
}

Все классы и дальнейший код можно найти в приложении.

На (Рисунке 16) представлено меню игры «Крестики-нолики». Здесь есть кнопка «Играть», она переносит к началу игры и recycleview с выводом топ игроков.

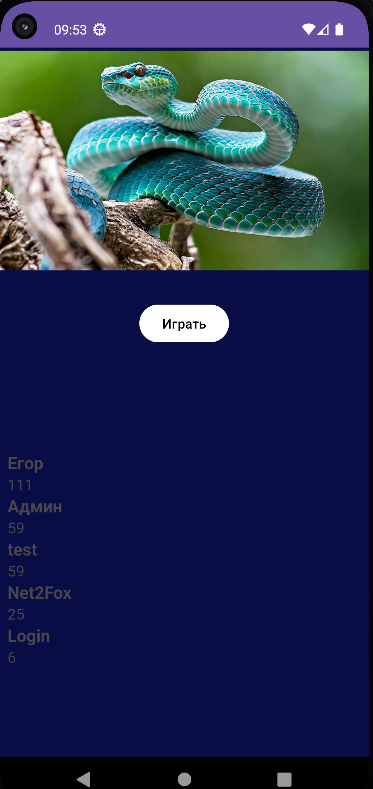


Рисунок 16. Меню игры змейка

Страница игры «Змейка» делится на 2 части (Рисунок 17). Игра со змейкой подразумевает 2 возможности управления по стрелочкам и водя по игровому полю. Указывается количество очков, набранных в данный момент и с прошлым рекордом.

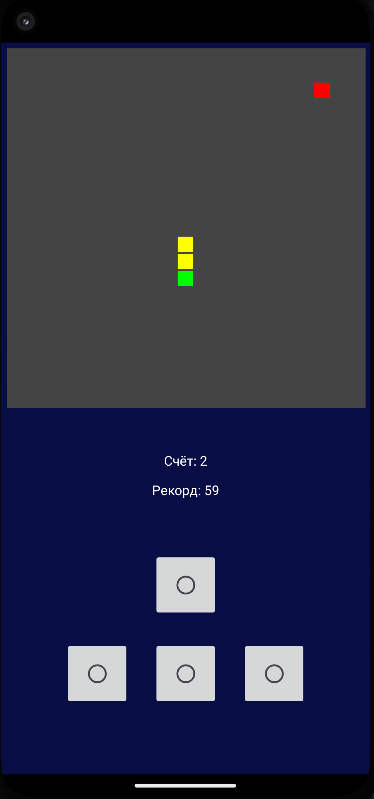
****

Рисунок 17. Игра змейка

## 2.4. Отладка и тестирование программы

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

В таблице представлены результаты отладки и тестирования программного обеспечения.

Таблица 1. Тестирование и отладка программного обеспечения

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  теста | Название теста | Вводимое значение | Ожидаемая реакция программы | Фактический результат | Ошибка выявлена |
| 1 | Вход | [admin@admin.com](mailto:admin@admin.com)  123456 | Вход в меню |  | нет |
| 2 | Регистрация | Login  [Test@test.ru](mailto:Test@test.ru)  1234567 | Создание нового пользователя |  | нет |
| 3 | Выход |  | Выход на странницу авторизоцию |  | нет |
| 4 | Змейка с лучшем счетом |  | Вывод рядом с игровым полем лучшего значения для главного аккаунта |  | 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 игра.

Функции программы:

– вход,

– регистрация,

– возможность запустить игры,

– выход из аккаунта,

– сохранение результатов.

Ниже представлены минимальные требования к телефону для установки приложения:

* операционная система: Android.
* версия: Android 7 и новее.
* разрешение: Доступ к интернету, доступ к изменению внутренней памяти.

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 дипломным проектом решалась задача разработки развлекательного мультиплатформенного игрового сервиса, который может использоваться на мобильных устройствах и персональных компьютерах.

Для разработки desktop-версии использовался .NET Framework, а для мобильной версии - Android Studio. Для того, чтобы достигнуть этой цели, 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. Разработан развлекательный мультиплатформенный игровой сервис, который позволит улучшить качество досуга для пользователей. Это может положительно сказаться на общем эмоциональном состоянии пользователей.

Игровой сервис имеет потенциал для развития и улучшения. К примеру, можно увеличить общее число игр, добавить личный кабинет, сделать привязку 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/> (дата обращения: 25.02.2022) – Текст: электронный.
2. Пример для игры змейка– [Электронный ресурс] – URL: <https://github.com/d-lehel/snake-android-game> (дата обращения: 11.01.2022) – Текст: электронный.
3. Flaticon сайт с иконками – [Электронный ресурс] – URL: <https://www.flaticon.com/ru/free-icon/tic-tac-toe_819972> (дата обращения: 15.03.2023) – Текст: электронный.
4. Документация языка kotlin – [Электронный ресурс] – URL: <https://kotlinlang.org/docs/home.html> (дата обращения: 02.02.2023) – Текст: электронный.
5. Документация для Androis studio – [Электронный ресурс] – URL: <https://developer.android.com/docs> (дата обращения: 21.01.2023) – Текст: электронный.
6. Сайт с примерами многих игр– [Электронный ресурс] – URL: <https://developer.alexanderklimov.ru/android/games/> (дата обращения: 02.01.2022) – Текст: электронный.
7. Firebase – [Электронный ресурс]– URL: <https://console.firebase.google.com/?hl=ru> (дата обращения: 02.04.2023) – Текст: электронный.
8. Документация для Firebase – [Электронный ресурс] – URL: <https://firebase.google.com/docs?authuser=0&hl=en> (дата обращения: 02.10.2022) – Текст: электронный.
9. Документация для MySQl  – [Электронный ресурс] – URL: <https://dev.mysql.com/doc/> (дата обращения: 08.11.2022) – Текст: электронный.
10. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
11. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
12. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
13. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
14. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
15. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
16. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
17. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.
18. Руководство по языку Kotlin – [Электронный ресурс]. – URL: [http://metanit.com/sharp/tutorial/](http://metanit.com/sharp/tutorial/%20) (дата обращения: 02.11.2022) – Текст: электронный.

# ПРИЛОЖЕНИЕ

Class TicTacLogic

package ru.maximus.playzer.TicGame  
import android.content.Context  
import android.widget.Toast  
class TicTacLogic(private val context: Context) {  
 val board = Array(3) { Array(3) { "" } }  
 var currentPlayer = "X"  
 var gameEnded = false  
 fun makeMove(row: Int, col: Int): Boolean {  
 if (board[row][col] == "" && !gameEnded) {  
 board[row][col] = currentPlayer  
 checkForWin()  
 currentPlayer = if (currentPlayer == "X") "O" else "X"  
 return true  
 }  
 return false  
 }  
 fun isGameEnded(): Boolean {  
 return gameEnded  
 }  
 fun checkForWin() {  
 // Check rows  
 for (i in 0..2) {  
 if (board[i][0] == board[i][1] && board[i][0] == board[i][2] && board[i][0] != "") {  
 gameEnded = true  
 showToastWin()  
 // Display win message and prompt for new game  
 return  
 }  
 }  
 // Check columns  
 for (j in 0..2) {  
 if (board[0][j] == board[1][j] && board[0][j] == board[2][j] && board[0][j] != "") {  
 gameEnded = true  
 showToastWin()  
 // Display win message and prompt for new game  
 return  
 }  
 }  
 // Check diagonals  
 if (board[0][0] == board[1][1] && board[0][0] == board[2][2] && board[0][0] != "") {  
 gameEnded = true  
 showToastWin()  
 // Display win message and prompt for new game  
 return  
 }  
 if (board[0][2] == board[1][1] && board[0][2] == board[2][0] && board[0][2] != "") {  
 gameEnded = true  
 showToastWin()  
// Display win message and prompt for new game  
 return  
 }  
 // Check for tie  
 if (isBoardFull() && !gameEnded) {  
 gameEnded = true  
 showToastWin()  
 // Display tie message and prompt for new game  
 }  
  
 }  
 fun showToastWin(){  
 val message = "Игрок $currentPlayer выйграл!"  
 Toast.makeText(context, message, Toast.LENGTH\_LONG).show()  
 }  
 fun isBoardFull(): Boolean {  
 for (i in 0..2) {  
 for (j in 0..2) {  
 if (board[i][j] == "") {  
 return false  
 }  
 }  
 }  
 return true  
 }  
 fun resetGame() {  
 board.forEach { it.fill("") }  
 currentPlayer = "X"  
 gameEnded = false  
 }  
}

class FirebaseAuthentication

package ru.maximus.playzer  
  
import androidx.lifecycle.MutableLiveData  
import com.google.firebase.auth.FirebaseAuth  
import com.google.firebase.auth.FirebaseUser  
import com.google.firebase.firestore.Query  
import com.google.firebase.firestore.QuerySnapshot  
import com.google.firebase.firestore.SetOptions  
import com.google.firebase.firestore.ktx.firestore  
import com.google.firebase.ktx.Firebase  
import kotlinx.coroutines.tasks.await  
  
class FirebaseAuthentication {  
 val signInResult = MutableLiveData<Boolean?>()  
 val scoreResult = MutableLiveData<Long?>()  
 val usersResult = MutableLiveData<List<User>>()  
 val userResult = MutableLiveData<User>()  
 var userList: MutableList<User> = mutableListOf()  
 private val firebaseAuth = FirebaseAuth.getInstance()  
 private val db = Firebase.firestore  
  
 // Реализация функции для регистрации нового пользователя по электронной почте и паролю  
 suspend fun signUp(email: String, password: String, name: String): Boolean {  
 return try {  
 val authResult = firebaseAuth.createUserWithEmailAndPassword(email, password).await()  
 if (authResult.user != null) {  
 db.collection("players")  
 .document(authResult.user!!.uid)  
 .set(hashMapOf(  
 "name" to name  
 ))  
 .await()  
 firebaseAuth.signOut()  
 }  
 authResult.user != null  
 } catch (e: Exception) {  
 return false  
 }  
 }  
  
  
 // Реализация функции для входа существующего пользователя по электронной почте и паролю  
 suspend fun signIn(email: String, password: String): Boolean {  
 return try {  
 val authResult = firebaseAuth.signInWithEmailAndPassword(email, password).await()  
 signInResult.postValue(authResult.user != null)  
 authResult.user != null  
 } catch (e: Exception) {  
 return false  
 }  
 }  
  
 fun signOut() {  
 firebaseAuth.signOut()  
 }  
  
 fun getSignedInUser(): FirebaseUser? {  
 return firebaseAuth.currentUser  
 }  
  
 suspend fun readSnakeBestScore() {  
 val user = firebaseAuth.currentUser!!  
 val result = db.collection("players")  
 .document(user.uid)  
 .get()  
 .addOnSuccessListener {  
 val data = it.data  
 if (data != null) {  
 val score = data["snakeScore"] as Long?  
 scoreResult.postValue(score)  
 }  
  
 }  
 .await()  
 }  
  
 suspend fun username(){  
 }  
  
 suspend fun getUsers() {  
 val users = db.collection("players").orderBy("snakeScore", Query.Direction.DESCENDING).get().await()  
 userList = mutableListOf()  
 for (user in users) {  
 userList.add(User(  
 user.getString("name")!!,  
 user.getLong("snakeScore")  
 ))  
 }  
 usersResult.postValue(userList)  
 }  
  
 suspend fun usernam(){  
 var postDocument = db.collection("players").document(firebaseAuth.currentUser!!.uid).get().await()  
 var user = User(  
 postDocument.getString("name")!!,  
 postDocument.get("snakeScore", Long::class.java)  
 )  
 userResult.postValue(user)  
 }  
  
 suspend fun writeSnakeBestScore(bestScore: Long) {  
 val user = firebaseAuth.currentUser!!  
 val userMap: Map<String, Long> = hashMapOf(  
 "snakeScore" to bestScore  
 )  
 db.collection("players")  
 .document(user.uid)  
 .set(userMap, SetOptions.merge())  
 .await()  
 }  
  
 companion object {  
 private var INSTANCE: FirebaseAuthentication? = null  
  
 fun initialize() {  
 if (INSTANCE == null) {  
 INSTANCE = FirebaseAuthentication()  
 }  
 }  
  
 fun get(): FirebaseAuthentication {  
 return INSTANCE ?:  
 throw IllegalStateException("FirebaseAuthentication must be initialized")  
 }  
 }  
}

class Playzer

package ru.maximus.playzer  
  
import android.app.Application  
import com.google.firebase.FirebaseApp  
  
class Playzer : Application() {  
  
 override fun onCreate() {  
 super.onCreate()  
 FirebaseApp.initializeApp(this)  
 FirebaseAuthentication.initialize()  
 }  
}

class User

package ru.maximus.playzer  
class User (val name:String, val snakeScore:Long?)

Игра Змейка

Xml:

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#090d45"  
 android:orientation="vertical"  
 tools:context=".SnakeGame.SnakeGa">  
 <ru.maximus.playzer.SnakeGame.CanvasView  
 android:id="@+id/canvas"  
 android:layout\_width="match\_parent"  
 android:layout\_height="400dp"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginStart="6dp"  
 android:layout\_marginTop="6dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
 <ImageButton  
 android:id="@+id/button\_down"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginBottom="72dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle"  
 tools:ignore="MissingConstraints" />  
 <ImageButton  
 android:id="@+id/button\_right"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginStart="24dp"  
 android:layout\_marginBottom="72dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/button\_down"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle" />  
 <ImageButton  
 android:id="@+id/button\_left"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginEnd="24dp"  
 android:layout\_marginBottom="72dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/button\_down"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle" />  
 <ImageButton  
 android:id="@+id/button\_up"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginBottom="24dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/button\_down"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle" />  
 <TextView  
 android:id="@+id/scorebar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="28dp"  
 android:textColor="@android:color/white"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/canvas"  
 tools:ignore="MissingConstraints" />  
 <TextView  
 android:id="@+id/storyscorebar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="12dp"  
 android:textColor="@android:color/white"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.501"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/scorebar"  
 tools:ignore="MissingConstraints" />  
</androidx.constraintlayout.widget.ConstraintLayout>

kt:

package ru.maximus.playzer.SnakeGame  
import android.content.Intent  
import androidx.appcompat.app.AppCompatActivity  
import android.os.Bundle  
import android.widget.Button  
import androidx.lifecycle.Observer  
import androidx.lifecycle.lifecycleScope  
import androidx.recyclerview.widget.LinearLayoutManager  
import androidx.recyclerview.widget.RecyclerView  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.launch  
import ru.maximus.playzer.Adaptr  
import ru.maximus.playzer.FirebaseAuthentication  
import ru.maximus.playzer.R  
import ru.maximus.playzer.User  
class FirstPSnake : AppCompatActivity() {  
 private lateinit var recyclerView: RecyclerView  
 private lateinit var adapter: Adaptr  
 private lateinit var firebase: FirebaseAuthentication  
 fun getTopPlayers(): List<User> {  
 firebase = FirebaseAuthentication.get()  
 return (firebase.userList.map { User(it.name, it.snakeScore) })  
 }  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_first\_psnake)  
 firebase = FirebaseAuthentication.get()  
 recyclerView = findViewById(R.id.recycler\_view)  
 recyclerView.layoutManager = LinearLayoutManager(this)  
 firebase.usersResult.observe(  
 this,  
 Observer { result ->  
 adapter = Adaptr(result)  
 recyclerView.adapter = adapter  
 updateUI()  
 }  
 )  
 lifecycleScope.launch(Dispatchers.IO) {  
 firebase.getUsers()  
 }  
 var butten = findViewById<Button>(R.id.Play)  
 butten.setOnClickListener{  
 val intent = Intent (this, SnakeGa::class.java)  
 startActivityForResult(intent,0)  
 }  
 }  
 private fun updateUI() {  
 recyclerView.adapter!!.notifyDataSetChanged()  
 }  
}

xml snake\_game

<?xml version="1.0" encoding="utf-8"?>  
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:background="#090d45"  
 android:orientation="vertical"  
 tools:context=".SnakeGame.SnakeGa">  
  
 <ru.maximus.playzer.SnakeGame.CanvasView  
 android:id="@+id/canvas"  
 android:layout\_width="match\_parent"  
 android:layout\_height="400dp"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginStart="6dp"  
 android:layout\_marginTop="6dp"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent" />  
  
 <ImageButton  
 android:id="@+id/button\_down"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginBottom="72dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle"  
 tools:ignore="MissingConstraints" />  
  
 <ImageButton  
 android:id="@+id/button\_right"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginStart="24dp"  
 android:layout\_marginBottom="72dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintStart\_toEndOf="@+id/button\_down"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle" />  
  
 <ImageButton  
 android:id="@+id/button\_left"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginEnd="24dp"  
 android:layout\_marginBottom="72dp"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toStartOf="@+id/button\_down"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle" />  
  
 <ImageButton  
 android:id="@+id/button\_up"  
 android:layout\_width="70dp"  
 android:layout\_height="70dp"  
 android:layout\_marginBottom="24dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/button\_down"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:srcCompat="?android:attr/listChoiceIndicatorSingle" />  
  
 <TextView  
 android:id="@+id/scorebar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="28dp"  
 android:textColor="@android:color/white"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/canvas"  
 tools:ignore="MissingConstraints" />  
  
 <TextView  
 android:id="@+id/storyscorebar"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_centerHorizontal="true"  
 android:layout\_marginTop="12dp"  
 android:textColor="@android:color/white"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.501"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toBottomOf="@+id/scorebar"  
 tools:ignore="MissingConstraints" />  
  
</androidx.constraintlayout.widget.ConstraintLayout>

kt:

package ru.maximus.playzer.SnakeGame  
  
import android.annotation.SuppressLint  
import android.content.pm.ActivityInfo  
import android.os.Bundle  
import android.view.GestureDetector  
import android.view.MotionEvent  
import android.view.View  
import android.widget.ImageButton  
import android.widget.TextView  
import androidx.appcompat.app.AppCompatActivity  
import androidx.lifecycle.Observer  
import androidx.lifecycle.lifecycleScope  
import kotlinx.coroutines.CoroutineScope  
import kotlinx.coroutines.Dispatchers  
import kotlinx.coroutines.delay  
import kotlinx.coroutines.launch  
import ru.maximus.playzer.FirebaseAuthentication  
import ru.maximus.playzer.R  
import kotlin.math.abs  
  
  
class SnakeGa : AppCompatActivity() {  
  
 private lateinit var scorebar: TextView  
 private lateinit var stroyscorebar: TextView  
 var score: Long = 0  
 var best\_score: Long = 0  
 var time: Long = 150  
 private lateinit var firebase: FirebaseAuthentication  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.activity\_snake\_ga)  
  
 scorebar = findViewById<TextView>(R.id.scorebar)  
 stroyscorebar = findViewById<TextView>(R.id.storyscorebar)  
 firebase = FirebaseAuthentication.get()  
 firebase.scoreResult.observe(this,  
 Observer {  
 it?.let {  
 best\_score = it  
 stroyscorebar.text = getString(R.string.best\_score\_text, best\_score)  
 }  
 })  
 lifecycleScope.launch(Dispatchers.IO) {  
 firebase.readSnakeBestScore()  
 }  
 window.decorView.systemUiVisibility = View.SYSTEM\_UI\_FLAG\_FULLSCREEN  
 requestedOrientation = ActivityInfo.SCREEN\_ORIENTATION\_PORTRAIT  
 supportActionBar?.hide()  
 var snake: Snake = Snake()  
 //возможность управлять жестами  
 open class OnSwipeTouchListener : View.OnTouchListener {  
  
 private val gestureDetector = GestureDetector(GestureListener())  
  
 fun onTouch(event: MotionEvent): Boolean {  
 return gestureDetector.onTouchEvent(event)  
 }  
  
 private inner class GestureListener : GestureDetector.SimpleOnGestureListener() {  
  
 private val SWIPE\_THRESHOLD = 100  
 private val SWIPE\_VELOCITY\_THRESHOLD = 100  
  
 override fun onDown(e: MotionEvent): Boolean {  
 return true  
 }  
  
 override fun onSingleTapConfirmed(e: MotionEvent): Boolean {  
 onTouch(e)  
 return true  
 }  
  
 override fun onFling(  
 e1: MotionEvent,  
 e2: MotionEvent,  
 velocityX: Float,  
 velocityY: Float  
 ): Boolean {  
 val result = false  
 try {  
 val diffY = e2.y - e1.y  
 val diffX = e2.x - e1.x  
 if (abs(diffX) > abs(diffY)) {  
 if (abs(diffX) > SWIPE\_THRESHOLD && abs(velocityX) > SWIPE\_VELOCITY\_THRESHOLD) {  
 if (diffX > 0) {  
 onSwipeRight()  
 } else {  
 onSwipeLeft()  
 }  
 }  
 } else {  
 // this is either a bottom or top swipe.  
 if (abs(diffY) > SWIPE\_THRESHOLD && abs(velocityY) > SWIPE\_VELOCITY\_THRESHOLD) {  
 if (diffY > 0) {  
 onSwipeTop()  
 } else {  
 onSwipeBottom()  
 }  
 }  
 }  
 } catch (exception: Exception) {  
 exception.printStackTrace()  
 }  
 return result  
 }  
 }  
  
 @SuppressLint("ClickableViewAccessibility")  
 override fun onTouch(v: View, event: MotionEvent): Boolean {  
 return gestureDetector.onTouchEvent(event)  
 }  
  
 open fun onSwipeRight() {}  
 open fun onSwipeLeft() {}  
 open fun onSwipeTop() {}  
 open fun onSwipeBottom() {}  
 }  
  
  
 var canvas = findViewById<CanvasView>(R.id.canvas)  
  
 canvas.setOnTouchListener(object : OnSwipeTouchListener() {  
 override fun onSwipeLeft() {  
 snake.alive = true  
 if (snake.direction != "right")  
 snake.direction = "left"  
 }  
  
 override fun onSwipeRight() {  
 snake.alive = true  
 if (snake.direction != "left")  
 snake.direction = "right"  
 }  
  
 override fun onSwipeTop() {  
 snake.alive = true  
 if (snake.direction != "up")  
 snake.direction = "down"  
 }  
  
 override fun onSwipeBottom() {  
 snake.alive = true  
 if (snake.direction != "down")  
 snake.direction = "up"  
 }  
 })  
////////////////////////////////////////////////////////////////////////////////////////  
  
 // движение змеи  
 CoroutineScope(Dispatchers.IO).launch {  
 while (true) {  
 while (snake.alive) {  
 when (snake.direction) {  
 "up" -> {  
  
 snake.headY -= 50  
 if (!snake.possibleMove()) {  
 snake.alive = false  
 snake.reset()  
 lifecycleScope.launch(Dispatchers.Main) {  
 showScoreBar()  
 }  
 }  
 }  
  
 "down" -> {  
  
 snake.headY += 50  
 if (!snake.possibleMove()) {  
 snake.alive = false  
 snake.reset()  
 lifecycleScope.launch(Dispatchers.Main) {  
 showScoreBar()  
 }  
  
 }  
 }  
  
 "left" -> {  
  
 snake.headX -= 50  
 if (!snake.possibleMove()) {  
 snake.alive = false  
 snake.reset()  
  
 lifecycleScope.launch(Dispatchers.Main) {  
 showScoreBar()  
 }  
 }  
  
 }  
  
 "right" -> {  
  
 snake.headX += 50  
 if (!snake.possibleMove()) {  
 snake.alive = false  
 snake.reset()  
 lifecycleScope.launch(Dispatchers.Main) {  
 showScoreBar()  
 }  
  
 }  
 }  
 }  
 // преобразование головы в тело  
 Snake.bodyParts.add(arrayOf(snake.headX, snake.headY))  
  
  
 if (snake.headX == Food.posX && snake.headY == Food.posY){  
 Food.generate()  
  
 lifecycleScope.launch(Dispatchers.Main) {  
 scorebar.text = getString(R.string.score\_text,++score)  
  
 if (score.toInt() % 3 == 0)  
 {  
 time -= 2 // усорение скорости змеи  
 }  
  
  
 }  
 }  
 else  
 Snake.bodyParts.removeAt(0)  
  
  
 lifecycleScope.launch(Dispatchers.Main){  
 canvas.invalidate()  
 }  
  
 delay(time)  
 }  
 }  
 }  
 val button\_up = findViewById<ImageButton>(R.id.button\_up)  
 button\_up.setOnClickListener {  
 snake.alive = true  
 if (snake.direction != "down")  
 snake.direction = "up"  
 }  
 val button\_down = findViewById<ImageButton>(R.id.button\_down)  
 button\_down.setOnClickListener {  
 snake.alive = true  
 if (snake.direction != "up")  
 snake.direction = "down"  
 }  
 val button\_left = findViewById<ImageButton>(R.id.button\_left)  
 button\_left.setOnClickListener {  
 snake.alive = true  
 if (snake.direction != "right")  
 snake.direction = "left"  
 }  
 val button\_right = findViewById<ImageButton>(R.id.button\_right)  
 button\_right.setOnClickListener {  
 snake.alive = true  
 if (snake.direction != "left")  
 snake.direction = "right"  
 }  
 }  
  
 private fun showScoreBar() {  
 if (best\_score < score) {  
 stroyscorebar.text = getString(R.string.best\_score\_text, score)  
 best\_score = score  
 lifecycleScope.launch(Dispatchers.IO) {  
 firebase.writeSnakeBestScore(best\_score)  
 }  
  
 }  
  
 scorebar.text = getString(R.string.score\_text,0)  
 score = 0  
 time = 150  
 }

}

class Snake

package ru.maximus.playzer.SnakeGame  
  
class Snake {  
 // по умолчанию: одна часть тела  
 var headX = 0f  
 var headY = 0f  
  
  
 var direction = "right";  
 var alive = false;  
  
  
 fun possibleMove(): Boolean {  
 if (headX < 0f || headX > 1000f || headY < 0f || headY > 1000f)  
 return false  
 return true  
 }  
  
 fun reset() {  
 headX = 0f;  
 headY = 0f;  
 bodyParts = mutableListOf(arrayOf(0f, 0f))  
 direction = "right";  
 }  
  
 companion object {  
 var bodyParts =  
 mutableListOf(arrayOf(0f, 0f))  
 }  
}

class Food

package ru.maximus.playzer.SnakeGame  
  
class Food {  
 companion object {  
 var posX = 400f  
 var posY = 400f  
  
 fun generate() {  
 posX = (1..20).random().toFloat() \* 50  
 posY = (1..20).random().toFloat() \* 50  
 }  
 }  
}

class CanvasView

package ru.maximus.playzer.SnakeGame  
  
import android.content.Context  
import android.graphics.Canvas  
import android.graphics.Color  
import android.graphics.Paint  
import android.util.AttributeSet  
import android.view.View  
  
class CanvasView @JvmOverloads constructor(  
 context: Context,  
 attrs: AttributeSet? = null,  
 defStyleAttr: Int = 0  
) : View(context, attrs, defStyleAttr) {  
  
 override fun onDraw(canvas: Canvas?) {  
 super.onDraw(canvas)  
 val snakeHead = Paint()  
 snakeHead.setColor(Color.GREEN)  
 val snakeBody = Paint()  
 snakeBody.setColor(Color.YELLOW)  
  
 val food = Paint()  
 food.setColor(Color.RED)  
 val level = Paint()  
 level.setColor(Color.DKGRAY)  
  
 // Задняя часть картинки  
 canvas?.drawRect(0f,0f,1050f,1050f,level)  
  
 // Змея из массива  
 // left x, top y, right x+50, bottom y +50  
  
 for (i in Snake.bodyParts){  
 canvas?.drawRect(i[0], i[1], i[0]+45, i[1]+45, snakeBody)  
 }  
 canvas?.drawRect(Snake.bodyParts[Snake.bodyParts.size-1][0], Snake.bodyParts[Snake.bodyParts.size-1][1], Snake.bodyParts[Snake.bodyParts.size-1][0]+45, Snake.bodyParts[Snake.bodyParts.size-1][1]+45, snakeHead)  
 // еда из массива  
 // left x, top y, right x+50, bottom y +50  
 canvas?.drawRect(Food.posX, Food.posY, Food.posX +45, Food.posY +45,food)  
  
  
 }  
}