Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

К ЗАЩИТЕ ДОПУСТИТЬ

Заведующий кафедрой АОИ

канд. экон. наук, доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Сидоров

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2022 г.

Магистерская диссертация по направлению 09.04.04

«Программная инженерия»

**КРОССПЛАТФОРМЕННОЕ ПРИЛОЖЕНИЕ ДЛЯ ПРОВЕДЕНИЯ СОРЕВНОВАНИЙ ПО НАСТОЛЬНОМУ ТЕННИСУ**

|  |
| --- |
| Студент гр. 420-М1  \_\_\_\_\_\_\_\_\_\_\_\_\_ Е. Баринов  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_2022 г. |
| Руководитель  доцент кафедры АОИ, канд. техн. наук  \_\_\_\_\_\_\_\_\_\_\_\_\_ Л.П.Турунтаев  «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_2022 г. |

Томск 2022

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ

УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)

Кафедра автоматизации обработки информации (АОИ)

УТВЕРЖДАЮ

Заведующий кафедрой АОИ

канд. экон. наук, доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Сидоров

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_2022 г.

ЗАДАНИЕ

на выполнение магистерской диссертации

студенту Баринову Егору группы 420-М1, факультета систем управления

1. Тема работы: «Кроссплатформенное приложение для проведения соревнований по настольному теннису»

утверждена приказом по вузу от \_\_\_\_\_\_\_\_\_\_\_ №\_\_\_\_\_\_\_\_\_\_\_

1. Срок сдачи работы на кафедру: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_
2. Содержание работы (перечень подлежащих разработке вопросов):
3. исследование предметной области;
4. обзор и анализ аналогов;
5. обзор и анализ типов разработки;
6. обзор и анализ кроссплатформенных Фреймворков;
7. проектирование кроссплатформенного приложения;
8. разработка кроссплатформенного приложения.
9. Дата выдачи задания: \_\_\_\_\_\_\_\_\_\_

Руководитель:

доцент кафедры АОИ, канд. техн. наук \_\_\_\_\_\_\_ Л.П.Турунтаев

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2022 г.

Задание принял к исполнению:

студент гр. 420-М1 \_\_\_\_\_\_\_ Е. Баринов

«\_\_\_\_» \_\_\_\_\_\_\_\_ 2022 г.

**РЕФЕРАТ**

Выпускная квалификационная работа, 70 страниц, 59 рисунков, 4 таблицы, 10 источников.

КРОССПЛАТФОРМЕННОЕ ПРИЛОЖЕНИЕ, DART, FLUTTER, СОРЕВНОВАНИЯ ПО НАСТОЛЬНОМУ ТЕННИСУ, ТРЕНИРОВКИ ПО НАСТОЛЬНОМУ ТЕННИСУ

Объектом разработки является программное обеспечение, позволяющее автоматизировать процесс проведения соревнований и тренировок по настольному теннису.

Цель работы – упрощение и автоматизация процесса подготовки и проведения соревнований по настольному теннису с помощью разрабатываемого кроссплатформенного программного приложения.

Выпускная квалификационная работа содержит:

* описание предметной области;
* обзор и анализ аналогов;
* обзор и анализ типов разработки;
* обзор и анализ кроссплатформенных Фреймворков;
* проектирование кроссплатформенного приложения;
* разработку кроссплатформенного приложения.

В результате исследования была изучена предметная область, проанализированы существующие решения и типы разработки, выбран кроссплатформенный Фреймворк, спроектировано кроссплатформенное приложение и разработано полностью готовое приложение.

Выпускная квалификационная работа выполнена в текстовом редакторе Microsoft Word 2016.

Средства реализации кроссплатформенного приложения – язык программирования Dart, Фреймворк Flutter, NoSQL база данных реального времени.

**АBSTRACT**

The final qualifying work contains 70 pages, 59 figures, 4 tables, 10 sources.

CROSS-PLATFORM APPLICATION, DART, FLUTTER, TABLE TENNIS COMPETITIONS, TABLE TENNIS TRAINING.

The object of development is software that allows you to automate the process of competitions and training in table tennis.

The purpose of the work is to simplify and automate the process of preparing and holding table tennis competitions using a cross–platform software application being developed.

The final qualifying work contains:

* description of the subject area;
* review and analysis of analogues;
* overview and analysis of development types;
* overview and analysis of cross-platform frameworks;
* designing a cross-platform application;
* development of a cross-platform application

As a result of the research, the subject area was studied, existing solutions and types of development were analyzed, a cross-platform framework was selected, a cross-platform application was designed and a fully ready application was developed.

The final qualifying work was done in a Microsoft Word 2016 text editor.

Cross–platform application implementation tools - Dart programming language, Flutter framework, NoSQL real-time database.

Оглавление

[Введение 7](#_Toc107353767)

[1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 9](#_Toc107353768)

[1.1 Описание предметной области 9](#_Toc107353769)

[1.2 Обзор и анализ существующих решений 10](#_Toc107353770)

[2 ТЕХНОЛОГИИ РАЗРАБОТКИ 15](#_Toc107353771)

[2.1 Обзор типов разработки 15](#_Toc107353772)

[2.2 Обзор кроссплатформенных фреймворков 17](#_Toc107353773)

[2.2.1 Flutter 17](#_Toc107353774)

[2.2.2 React Native 19](#_Toc107353775)

[2.2.3 Xamarin 22](#_Toc107353776)

[2.3 База данных 25](#_Toc107353777)

[3 ПРОЕКТИРОВАНИЕ 27](#_Toc107353778)

[3.1 Пользовательские требования 27](#_Toc107353779)

[3.2 Функциональные требования 28](#_Toc107353780)

[3.3 Диаграммы прецедентов 31](#_Toc107353781)

[3.4 Диаграммы последовательностей 33](#_Toc107353782)

[3.4 Диаграммы деятельности 34](#_Toc107353783)

[3.5 Проектирование базы данных 36](#_Toc107353784)

[4 РЕАЛИЗАЦИЯ 40](#_Toc107353785)

[4.1 Выбор типа разработки 40](#_Toc107353786)

[4.2 Выбор кроссплатформенных фреймворков 40](#_Toc107353787)

[4.3 Выбор архитектур управления состоянием 41](#_Toc107353788)

[4.4 Устройство приложения 47](#_Toc107353789)

[4.5 Адаптивность приложения 50](#_Toc107353790)

[4.6 Интерфейс программного приложения 52](#_Toc107353791)

[Заключение 69](#_Toc107353792)

[Список литературы 70](#_Toc107353793)

# Введение

Спорт в настоящее время становится все более популярным. Он широко пропагандируется и рекламируется как один из компонентов здорового образа жизни. Специально организованные спортивные соревнования направлены на максимальную реализацию духовных и физических возможностей человека, группы людей, команды, демонстрацию и сопоставление уровня подготовленности, достижение высших результатов или победы в регламентированных специальными правилами. Соревнования в современном спорте являются не только способом выявления победителя, но и важнейшим средством подготовки спортсмена.

Большинство спортивных соревнований проводятся с использованием бумажных носителей и без использования информационных систем, приложений. Тем самым усложняя процесс проведения соревнований. А если рассматривать аналоги из профессиональных соревнований, то распространение производится в закрытом кругу лиц. И при проведении любительских соревнований становится проблемой отсутствие функциональных программ для проведения соревнований. Использование разрабатываемого программного приложения является актуальным, ввиду не доступности профессиональных программ, а также решает проблему автоматизации проведения соревнований. Тем самым упрощая процесс проведения соревнований.

Целью данной работы является разработка программного приложения, которое поможет автоматизировать процесс подготовки и проведения соревнований по настольному теннису.

Для достижения поставленной цели необходимо:

* провести исследование предметной области;
* провести обзор и анализ существующих аналогов;
* провести обзор и анализ типов разработки;
* провести обзор и анализ кроссплатформенных Фреймворков;
* спроектировать и разработать кроссплатформенное программное приложение.

Объектом исследования является процесс автоматизации проведения соревнований по настольному теннису.

Предмет исследования является программное приложение, предназначенные для сохранения и обработки данных получаемых в ходе проведения соревнований.

Практическая значимость данной работы заключается в разработке кроссплатформенного программного приложения, охватывающего весь процесс автоматизации проведения соревнований по настольному теннису.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## **1.1 Описание предметной области**

Настольный теннис – вид спорта, спортивная игра с мячом, в которой используют специальные ракетки и игровой стол, разграниченный сеткой пополам. Задачей игроков является удерживать мяч в игре при помощи ракеток, каждый игрок после одного отскока мяча на своей половине стола должен отправить мяч на половину стола соперника. Очко начисляется игроку или паре игроков, когда соперник не может вернуть мяч в соответствии с правилами. Каждая партия продолжается до 11 очков, матч состоит из нечётного количества партий, и играется на большинство побед в партиях [1].

Соревнования по настольному теннису подразделяют [2]:

* личные, в которых результаты личных встреч засчитывают каждому участнику отдельно;
* командные, в которых результаты личных встреч участников команды засчитывают команде в целом;
* лично-командные, личные соревнования, в которых результаты засчитывают одновременно каждому участнику отдельно и команде в целом;
* командно-личные, командные соревнования, в которых результаты засчитывают одновременно команде в целом и каждому участнику отдельно.

Каждый вид соревнований может состоять из личных встреч:

* одиночные: мужские, женские;
* парные: мужские, женские, смешанные;
* командные: одиночные или парные личные встречи в командном матче.

Вид соревнования и остальные правила определяются положением о соревновании. У разных организаций оно различается. Но можно выделить общие этапы проведения соревнований и подготовки:

* Оповещение о проведении и положении соревнования;
* Приём заявок;
* Проведение жеребьёвки;
* Cоставление расписания игр;
* Визуализация игр;
* Итоги прошедших матчей.

Программное приложения для проведения соревнований по настольному теннису представляет собой кроссплатформенное приложение, функционирующие как для персональных компьютеров, мобильных устройств, так и для web.

## **1.2 Обзор и анализ существующих решений**

ИСТОК-ТУРНИР – компьютерная программа для обслуживания турниров по индивидуальным видам спорта, включая игровые и единоборства [3]. Такие как бокс, борьба, бадминтон, дзюдо, теннис, сумо, сквош, ушу, кикбоксинг, настольный теннис.

Программа предназначена для организаторов, главных судей и секретарей, проводящих турниры международного, всероссийского, районного или клубного уровня среди профессионалов или любителей, взрослых, детей, юношей или ветеранов.

Поддерживаются как личные, так и командные соревнования. Допустимо использовать программу и для командных видов спорта (футбол, волейбол и т.д.): в спортивные таблицы вместо фамилий участников вводить названия команд. Но список членов команд не поддерживается в текущей версии.

Основные возможности:

* настройка вида спорта;
* терминология;
* особенности ведения счёта и допустимые сокращения;
* перечень дисциплин (разделов, стилей);
* перечни возрастных и весовых категорий;
* перечень упражнений (ката, ситуаций, тулей);
* база данных спортсменов, позволяющая избежать повторного ввода анкетных данных при проведении очередного турнира.

Второстепенные функции:

* Быстрое переключение между видами спорта. Настроив параметры, характерные для одного вида, можно переключиться на другой, а затем быстро вернуться к первому: ваши настройки будут восстановлены;
* Автоматическое вычисление возраста либо по году рождения, либо по дате рождения и дате начала турнира. Возможность выбора в распечатке списка спортсменов одного из столбцов: дата рождения, год рождения или возраст;
* Мастер объединения турниров, позволяющий копировать заявки и группы вместе со всеми турнирными таблицами из одного турнирного файла в другой;
* Подсчёт разницы выигранных и проигранных партий (сетов) и очков (геймов) при круговой схеме;
* Показ дополнительной информации в подчинённых окнах, "плавающих" поверх основных;
* карточка Спортсмен;
* карточка Заявка;
* карточка Работник (судья, тренер);
* карточка Поединок (матч, бой) с возможностью выбора цвета одежды каждого из участников.

Стандартные версии программы – Исток-Турнир поставляется только для персональных компьютеров, в одной из редакций. Для шеф-версии и федеральной версии доступны дополнительные модули. Шеф-версия в сетевом варианте превращает компьютер в сервер, к которому по локальной сети могут подключаться. Интерфейс программы представлен на рисунке 1.1

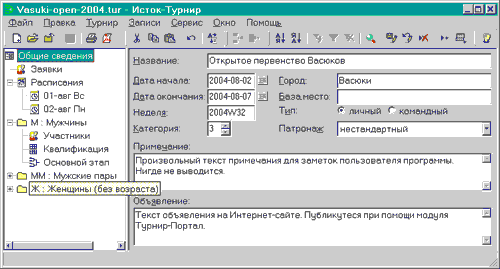


Рисунок 1.1 – интерфейс «Исток-Турнир»

ТУРНИР 2.0 – онлайн программа проведения соревнований с начальной стадии (регистрация участников) до завершающей (награждение победителей) и формирование обязательных отчётов для представления в Комитет по спорту своего региона [4].

Программа обеспечивает компактное хранение, быструю сортировку и фильтрацию данных, ручную модерацию на различных этапах, синхронизацию с табло для участников. Программа позволяет проводить турниры международного, всероссийского, районного или клубного уровня.

Основные возможности:

* настройка справочников;
* наименование спортивных организаций;
* списки тренеров;
* список судей;
* перечень подвидов (дисциплин, разделов, стилей);
* перечни возрастных и гендерных категорий;
* перечни возрастных и гендерных категорий;
* база данных спортсменов, клубов и тренеров – позволяет избежать повторного ввода анкетных данных при проведении очередного турнира;
* Первичная регистрация участников;
* Взвешивание участников и любые манипуляции с данными каждого участника или групп: исправление ошибок, смена веса, исключение спортсмена из списка участников без удаления из базы;
* ручная или автоматическая жеребьевка участников;
* составление турнирных таблиц соревнований;
* оперативная распечатка и сохранение в виде отдельных файлов следующих видов документов, бланков и отчётов;

Программа ТУРНИР 2.0 не накладывает никаких ограничений на своих пользователей по количеству турниров или зарегистрированных спортсменов, но распространяется только по подписке. Имеет только веб версию. Интерфейс программы ТУРНИР 2.0 представлен на рисунке 1.2

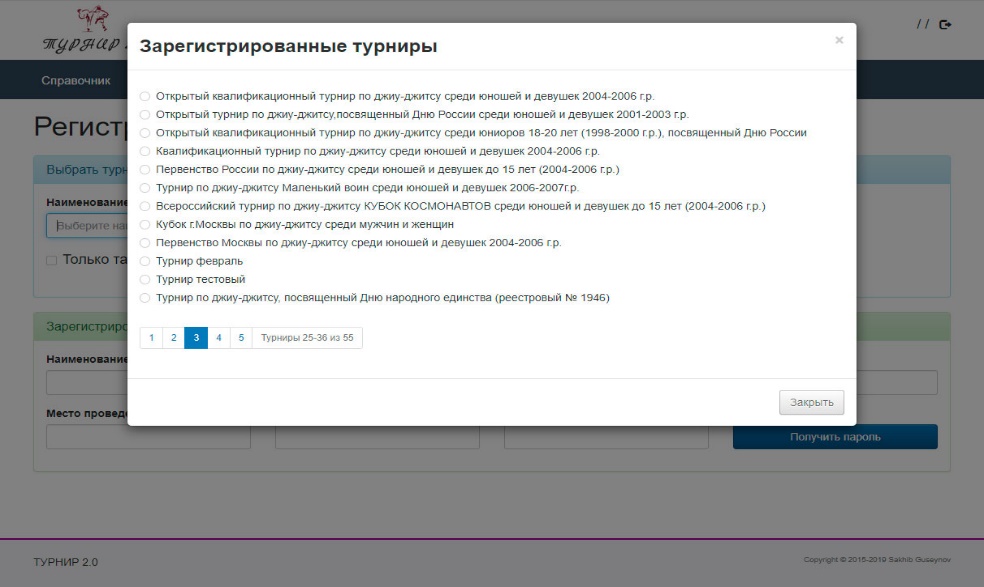


Рисунок 1.2 – интерфейс «ТУРНИР 2.0»

После обзора существующих решений можно составить таблицу, для сравнения основных функций (таблица 1.1).

Таблица 1.1 – Сравнение существующих решений

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии сравнения | ИСТОК-ТУРНИР | ТУРНИР 2.0 | Разрабатываемое приложение |
| Платформы | Desktop | Web | Desktop, Web, Android, IOS |
| Оповещение о проведении соревнований | - | - | + |
| Приём заявок | + | + | + |
| Проведение жеребьёвки | - | + | + |
| Составление расписаний | + | + | + |
| Проведения соревнований | + | + | + |
| Подведение итогов | + | + | + |
| Стоимость | Ограниченная бесплатная версия | Платная версия | Бесплатная версия |

По данным таблицы получаем вывод, что аналоги не решают в полном объёме поставленные задачи и разрабатываемое приложение будет актуальным для проведения соревнований по настольному теннису.

# 2 ТЕХНОЛОГИИ РАЗРАБОТКИ

## **2.1 Обзор типов разработки**

Разработка приложений, становится все более и более востребованной, таким образом разработчики ищут новые возможности для быстрой и качественной реализации проектов.

На данный момент разработку подразделяют на два вида:

1) Нативная разработка – при нативной разработке программисты используют оригинальные языки программирования и инструменты мобильной операционной системы. Разработка iOS приложений ведется в интегрированной в OS X и iOS среде Xcode, на языках Objective-C, Swift, C и С++. Для разработки Android приложений используется среда Android Studio и языки Java или Kotlin. Среда разработки – это специальный комплекс инструментов, созданный для максимально удобного написания кода, проектирования интерфейса, отладки, мониторинга и сборки приложения.

2) Кроссплатформенная разработка – в кроссплатформенной разработке используются специальные инструменты, которые позволяют создавать приложения сразу для нескольких операционных систем. Каркас и логика подобных приложений сначала программируются в кроссплатформенном инструменте, далее код интерпретируется в XCode или Android Studio проект.

Преимущества кроссплатформенной разработки:

* Экономия бюджета – использование одной технологии и набора графики снижает количество рабочих часов и бюджет проекта;
* Время разработки – отсутствие уникальных элементов интерфейса и одна технологическая платформа сокращает сроки разработки;
* Поддержка и обновление продукта – добавление функционала или исправление ошибок сразу для всех платформ;
* Мобильная версия сайта – большинство кроссплатформенных решений позволяют генерировать мобильную версию сайта из приложения;
* Единая логика приложения – логика приложения будет одинаково работать для всех платформ. Написанная и отлаженная логика содержит потенциально меньшее количество ошибок и расхождений в своей работе.

Преимущества нативной разработки:

* Скорость работы приложения – скомпилированный код проекта оптимален для родной платформы. Приложение получает полную аппаратную поддержку устройства и использует многопоточность для сложных задач. В процессе разработки приложения программисты могут измерять скорость работы всех участков кода и при необходимости их оптимизировать;
* Гибкость в реализации – нативная разработка использует все возможности мобильной операционной системы;
* Новый программный и аппаратный функционал — доступен для реализации сразу после выпуска обновлений;
* Простота тестирования – в распоряжении разработчиков и тестировщиков есть целых комплекс технологий: все параметры системы в процессе работы приложения контролируются автоматически, например, если приложение стало использовать больше памяти или ресурсов процессора. В нативной разработке доступны широкие возможности автоматического тестирования для любого метода в приложении, например, если часть приложения перестанет корректно работать после изменений кода, новая версия не соберется, а программист сразу увидит причину. Для нативных проектов встроен функционал удаленного мониторинга ошибок, который позволяет увидеть ошибку и ее причину на устройстве пользователя.

## **2.2 Обзор кроссплатформенных фреймворков**

### **2.2.1 Flutter**

Flutter – это кроссплатформенный фреймворк для создания приложений с единой кодовой базой, используемых на мобильных устройствах, настольных и веб-платформах [5]. Flutter не переводит код в нативный, который выполнится платформой. Он работает с канвасом (холстом), который размещает на экране телефона, где самостоятельно производит отрисовку (отображение, рендеринг) элементов. Flutter был разработан компанией Google, а с помощью стремительно растущего сообщества он доводится до совершенства с высокой скоростью.

Использует язык программирования Dart, также разработанный компанией Google. Dart – объектно-ориентированный язык программирования, который позиционируется в качестве альтернативы или в скором времени заменой JavaScript. Он был разработан как оптимизированный язык программирования для быстрого создания пользовательского интерфейса приложений под любые платформы [6].

Flutter относительно молодая технология, так как официальный релиз на рынке произошёл недавно в 2018 году. Однако, он мгновенно набрал популярность среди разработчиков и крупных компаний. На данный момент самые известные компании, у которых уже можно увидеть приложения, – это Google, Alibaba, Hamilton Musical, Greentea.

Стоит отметить, что у фреймворка Flutter большое количество сильных сторон за счёт его полной ориентированности под кроссплатформенную разработку и использования такого же языка программирования. Его кроссплатформенность позволяет использовать единую кодовую базу для всех платформ, что значительно ускоряет разработку. Благодаря собственному графическому движку, у фреймворка нет необходимости в создании отдельных интерфейсов под разные операционные системы. Фреймворк использует быстрый движок для рендеринга Skia, который является библиотекой 2D-графики с открытым исходным кодом и предоставляет общие API-интерфейсы. В свою очередь, они работают на различных аппаратных и программных платформах [7]. Вследствие этого, получается создавать максимально качественные и красивые интерфейсы, которые практически ничем не уступают нативным. Интерфейс легко разбивается на отдельные модули. Стоит выделить функцию Hot Reloading, которая позволяет добавлять новый код без повторной сборки приложения, и внушительное количество готовых виджетов. Одним из главных плюсов Flutter является высокая скорость. Она достигается за счёт компиляции приложения в машинный код с использованием графики и движков рендеринга, встроенных в C/C++. Отличительной особенностью Flutter является самостоятельная отрисовка каждого пикселя, контролирование жестов и анимаций. Так как Flutter использует собственные компоненты, которые называют виджетами, потребовалось разделение виджетов на два набора, где Material - для Androd и Cupertino - для IOS. На рисунке 2.1 представлена архитектура фреймворка Flutter.

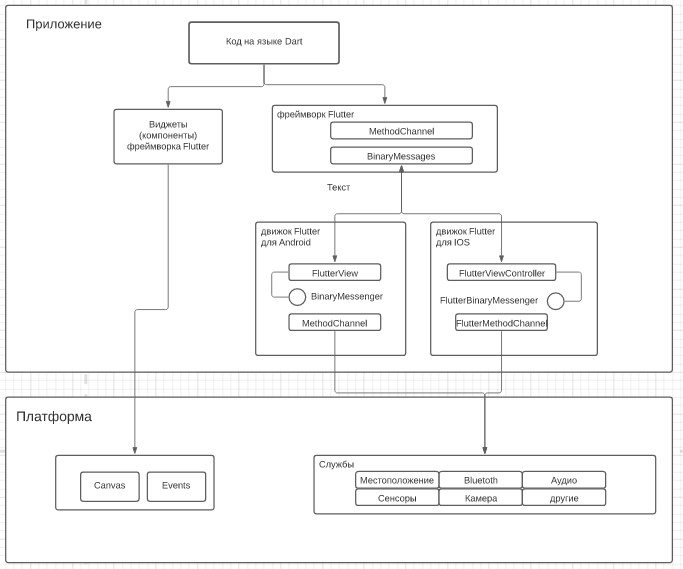


Рисунок 2.1 – Архитектура Flutter

Вследствие этого, приложения становятся быстрыми и надежными. В сфере кроссплатформенных технологий данная особенность дает существенное преимущество перед конкурентами.

Flutter имеет свои недостатки, которые выражаются в размере конечного установочного пакета из-за добавления в него виртуальной машины Dart. Так как интерфейс создается с помощью кода, грань между логикой и дизайном практически исчезает. Как и любая молодая технология, Flutter имеет определенные сложности при разработке, когда требуются специфические функции. Это выражается в том, что во фреймворке имеются только встроенные клиенты HTTP(S) и WebSockets, парсинг JSON необходимо производить вручную, а также имеются сложности в использовании WebView.

Flutter поддерживается в таких интегрированных средах разработки как Android Studio, IntelliJ и Visual Studio Code. На данный момент Flutter значительно обходит аналогичные кроссплатформенные фреймворки и показывает очень высокую производительность, в этом ему помогает современный язык программирования Dart и свой специальный движок для рендеринга, временами обходя даже нативные языки, чем и заслуживает лидерские позиции в кроссплатформенной разработке. В результате этого, сообщество Flutter стремительно растёт и благодаря тому, что компания Google вкладывает значительные ресурсы для развития фреймворка, это открывает новые перспективы развития для него

### **2.2.2 React Native**

React Native – это кроссплатформенная технология от Facebook, с полностью открытым исходным кодом, основная цель которой заключается в достижении такой же производительности как и у нативных приложений с помощью перевода кода приложения на машинный язык и обеспечение нативного внешнего вида мобильным приложениям [8]. Для основы React Native был выбран язык программирования JavaScript, который в настоящее время занимает лидирующие позиции по популярности среди языков программирования, и библиотека React, которая применяется для разработки пользовательских интерфейсов. При работе с React Native происходит сборка собственных элементов управления, чтобы построить иерархию с учетом интеграции и разработать пользовательский интерфейс используя фреймворк. Так как React Native вышел на рынок почти 6 лет назад, его поддерживают практически все популярные интегрированные среды для разработки. Одним из достоинств в React Native выделяют, возможность написания модулей на языках программирования Java, Kotlin, Objective-C и Swift. В приложениях React Native при использовании собственных модулей и библиотек, имеется возможность проводить сложные вычислительные операции.

Главный принцип React Native звучит следующим образом: «изучи один раз, используй везде». Данный принцип часто используется в кроссплатформенности и подразумевает, что при изучении определенного набора технологий, его можно будет использовать для разработки под различные платформы. Уже долгое время большинство разработчиков мобильных приложений больше ценят те фреймворки, в которых доступна функция «Горячая перезагрузка». Эта функция позволяет вносить изменение или добавлять новый блок кода во время запуска приложения, что ускоряет разработку пользовательского интерфейса. React Native имеет большой набор уже готовых компонентов, а при помощи сообщества с каждым разом увеличивает количество сторонних библиотек и их качество. Приложения, созданные с помощью React Native, по поведению и внешнему виду близки к нативным.

Стоит учитывать, что у React Native производительность приложений приближается в большинстве случаев к нативным приложением. Так как, происходит отображение компонентов через свои API-интерфейсы, с помощью виртуальной машины JavaScriptCore для Android и iOS платформ. Приложения, написанные на React Native в большинстве случаев, могут использовать от восьмидесяти до девяноста процентов кода при работе с двумя платформами.

React Native использует динамическую компиляцию для Android платформы, а для iOS платформы компиляция перед исполнением приложения не поддерживается. Поэтому на iOS платформе происходит интерпретирования JavaScript кода. Для верстки элементов специального редактора нет, поэтому в Android приложениях используется язык разметки XML, а в IOS приложениях разметку JSX.

Стоит отметить, что в каждом React Native-приложении параллельно выполняются потоки JS, Native и Shadow. JS поток осуществляет чтение и компиляцию JavaScript-кода. В этом потоке выполняется основная часть логики приложения. Бандлер Metro комбинирует весь JS-код в единый файл. Native поток производит выполнение нативного кода. Shadow поток выполняет пересчёт макета приложения. В этом потоке используется движок Yoga. На рисунке 2.2 представлена архитектура фреймворка React Native.

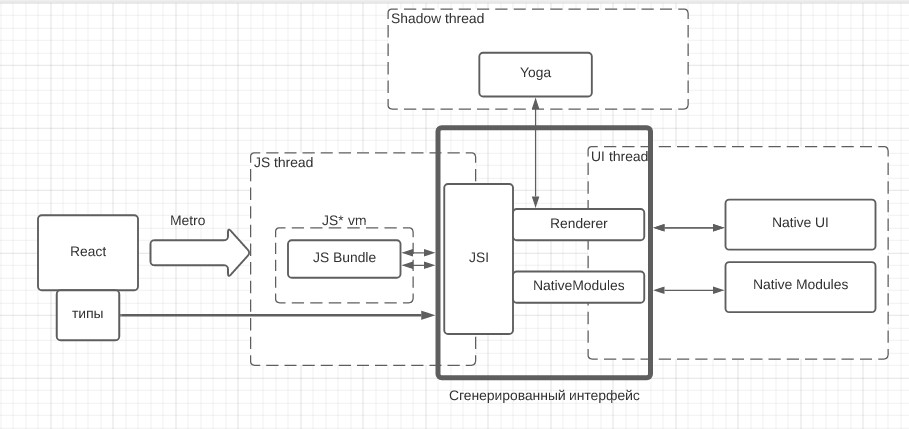


Рисунок 2.2 – Архитектура фреймворка React Native

В React Native модули фреймворка взаимодействуют с контроллерами пользовательского интерфейса Android и iOS, что позволяет максимально приблизить UX (пользовательский опыт взаимодействия) к уровню нативных приложений. Он также использует библиотеку ReactJS с обширными компонентами пользовательского интерфейса, которые могут упростить разработку пользовательского интерфейса. В результате React Native имеет множество готовых компонентов пользовательского интерфейса, таких как кнопки, модули и ползунки, которые ускоряют работу разработчика. React Native довольно похож на Ionic с точки зрения процесса написания кода и сборки. Однако, фреймворк не подходит для проектов с тяжелой и сложной графикой и анимацией.

Сообщество разработчиков React Native быстро растет и уже насчитывает большое количество разработчиков. Это значительно упрощает разработку проекта на React Native, так как разработчик может легко получить ответы на обычные вопросы и большое сообщество уже создало ряд сторонних библиотек.

React Native очень популярен, поскольку его уже применяют высокотехнологичные компании. Среди них можно выделить Instagram, Facebook, Walmart, Tesla, Pinterest, UberEats, Airbnb и другие.

### **2.2.3 Xamarin**

Xamarin – это кроссплатформенный фреймворк для разработки кроссплатформенных мобильных приложений с использованием языка программирования C# [9]. Xamarin использует язык программирования C# и платформу .NET. Приложения, разработанные на Xamarin, могут достигать такой же скорости, как и нативные приложения. Так как, разработчики связывают нативные библиотеки с открытым исходным кодом, который доступен для фреймворка с платформами Android и IOS. А также большая часть существующих проектов с открытым и закрытым исходным кодом, могут быть использованы в проектах Xamarin.iOS или Xamarin.Android повторно. У фреймворка наблюдается хорошая совместимость с устройствами и оптимальные условия тестирования. Интегрированные среды разработки (IDE), используемые для фреймворка это IDE Xamarin Studio и Visual Studio.

У Xamarin есть два основных инструмента: Xamarin.Android/iOS и Xamarin.Forms. В кроссплатформенной разработке Xamarin позволяет использовать единый API Xamarin.Essentials. Xamarin.Android и Xamarin.iOS предоставляют приложениям почти такие же возможности и интерфейс, которые можно встретить в нативной разработке. Фреймворк также предоставляет возможность работы со своим инструметом Xamarin.Forms, который значительно ускоряет процесс разработки и способствует совместимости всех элементов со всеми допустимыми платформами.

Производительность Xamarin также считается близкой к нативной, но зависит от того, используется ли Xamarin.Android, Xamarin.iOS или Xamarin.Forms. Приложения Xamarin.Android/iOS очень схожи с нативными, так как основной акцент делается на бизнес-логики. В приложениях используется элементы управления пользовательского интерфейса соответствующих платформ и улучшается ускорение устройства, которое в значительно степени зависит от платформы. И так как фреймворк компилирует C# в машинный код, получается приблизить производительность таких приложений к производительности нативных приложений. В свою очередь, Xamarin.Forms ориентирован на разработку, не опираясь на определенную платформу. Однако, это существенно снижает производительность приложения.

На платформе Android допустимо использовать JIT и AOT компиляцию, а на IOS платформе только AOT компиляцию. Xamarin компилирует язык C# низкоуровневые языки платформ по-своему, с помощью среды Mono. На рисунке 2.3 представлена архитектура фреймворка Xamarin.

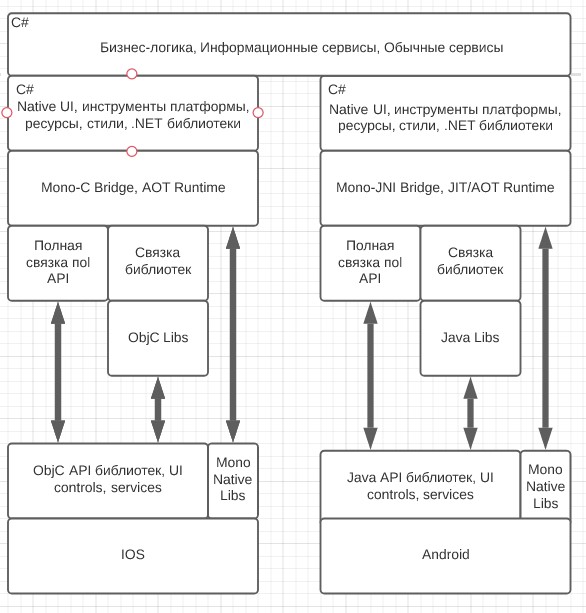


Рисунок 2.3 – Архитектура Xamarin

Стоит учитывать, работая с Xamarin, что можно столкнуться со сложностями, связанными с задержками обновлений платформы. Бывают случаи, когда требуются функции или интеграция в приложении, которые не предоставляются платформой. Однако, у платформы довольно ограниченная экосистема, это также ограничивает доступ к open-source библиотекам. В связи с этим разработчикам остается использовать компоненты, которые предоставляет платформа, и некоторые .NET ресурсы. Данный фреймворк не подходит для разработки приложений, которые требуют высокопроизводительную графику. Одним из главных недостатков принято считать внушительный размер приложений, которые в большинстве случаев имеют размер в 2 раза больше нативных.

На сегодняшний день Microsoft вкладывает много усилий в обучение и развитие сообщества Xamarin. Разработчики, работающие в экосистеме Microsoft, могут легко начать работу с технологией благодаря активной поддержке Microsoft. Для начинающих разработчиков имеется Xamarin University. Это портал электронного обучения, который включает классы от начального до продвинутого.

Xamarin стал продуктом с открытым исходным кодом по лицензии MIT. Однако, необходимо установить Visual Studio IDE, которая является проприетарной и распространяется по подписке. Он доступен бесплатно для некорпоративных проектов с количеством пользователей до 5. Но некоторые функции заблокированы за лицензиями Professional и Enterprise. Цена подписки Visual Studio начинается с 539 долларов США на пользователя в год и до 6000 долларов США в первый год для одного пользователя версии Enterprise. За это разработчики получают различные облачные сервисы, костюмы для корпоративной мобильности и многое другое.

На Xamarin разработаны такие приложения как Olo, The World Bank, Storyo, Slack, Pinterest, Siemens и другие.

## **2.3 База данных**

NoSQL база данных Firebase предоставляет пользователям удобные возможности для хранения и получения данных, а также имеет различные сервисы необходимые для любого мобильного приложения.

Стоит учитывать, что база данных была разработана компанией Google для разработки мобильных приложений, которая нацелена на помощь в создании, улучшении и расширении мобильных приложений. Текстовые данные хранятся в JSON формате. Доступны комфортные методы для чтения, обновления и извлечения данных. JSON (англ. JavaScript Object Notation) – это специальный формат обмена данными, который может взаимодействовать с любыми языками программирования, которые поддерживают данный формат. Особенность JSON состоит в том, что он является обычным текстом, поэтому при его использовании передача и обмен данных между разными языками программирования, БД и серверами происходит довольно просто.

Облачные базы данных удобны для использования их в регистрации и авторизации пользователей с хранением сессий авторизованных пользователей, с хранением медиа файлов. Ключевой особенностью является постоянная синхронизация данных в течение миллисекунд, когда данные изменяются.

В связи с тем, что изначально Firebase был нацелен на поддержку мобильных приложений, то основными поддерживаемыми ОС являются Android и IOS. Также имеется расширенная поддержка Flutter.

# 3 ПРОЕКТИРОВАНИЕ

## **3.1 Пользовательские требования**

Пользовательские требования - это решение с помощью приложения, списка целей и задач пользователей. Требования бывают в виде реализаций применения, в виде фраз, сценариев. Разработанные пользовательские требования описаны в таблице 3.1.

Таблица 3.1 – Пользовательские требования

|  |  |
| --- | --- |
| Название требования | Описание требования |
| Возможность создания профиля | Любому человеку должна предоставляться возможность создания собственного профиля |
| Возможность входа в систему | Любому прошедшему регистрацию должна быть предоставлена возможность входа в систему |
| Возможность получений информации о соревновании | После входа, пользователю должна быть предоставлена возможность получения информации о соревнованиях |
| Возможность получения приглашений для участия в соревнованиях | После входа, пользователю должна быть предоставлена возможность получения приглашений для участия в соревнованиях |
| Возможность добавления администраторов в соревнования | После создания тренировки или соревнования, пользователю должна быть предоставлена возможность добавления администраторов |
| Возможность записи в соревнованиях | После входа, пользователю должна быть возможность регистрации в соревнованиях |
| Возможность создания и проведения тренировок | После входа, пользователю должна быть предоставлена возможность создания и проведения тренировок |

Продолжение таблицы 3.1

|  |  |
| --- | --- |
| Название требования | Описание требования |
| Возможность проведения и получения результатов жеребьевки | После входа, пользователю должна быть предоставлена возможность проведения жеребьёвки при организации соревнований, либо получения результата в случае участия в соревнованиях |
| Возможность составления и получения результатов расписаний | После входа, пользователю должна быть предоставлена возможность получении результатов расписаний |
| Возможность создания и проведения соревнований | После входа, пользователю должна быть предоставлена возможность создавать и проводить соревнования |
| Возможность подведения итогов | После входа, должна быть предоставлена возможность получения результатов итогов |
| Возможность просмотра собственных данных и статистики | После входа, должна быть предоставлена возможность просмотра своего профиля и просмотра итогов и статистики |
| Возможность редактирования личной информации | После входа, должна быть предоставлена возможность редактирования личной, контактной информации |

## **3.2 Функциональные требования**

Функциональные требования – это требования, показывающие разработчикам цели и задачи пользователей, которые необходимо выполнить. Выработанные функциональные требования занесены в таблицу 3.2.

Таблица 3.2 – Функциональные требования

|  |  |
| --- | --- |
| Название требования | Описание требования |
| Возможность регистрации аккаунта | Не авторизованному пользователю приложения должна предоставляться возможность регистрации |
| Возможность входа в систему | Зарегистрированному пользователю должна быть предоставлена возможность авторизации |
| Возможность получений уведомлений о соревновании | Зарегистрированному пользователю должна быть предоставлена возможность получения уведомлений с информацией о соревнованиях |
| Возможность получения приглашений соревнований | Зарегистрированному пользователю должна быть предоставлена возможность получения приглашений для участия в соревнованиях |
| Возможность добавления администраторов в соревнования | Зарегистрированному пользователю должна быть предоставлена возможность добавления администраторов в соревнованиях |
| Возможность создания и проведения тренировок | Зарегистрированному пользователю должна быть предоставлена возможность создания и проведения тренировок |
| Возможность отправлять заявки | Зарегистрированному пользователю должна быть предоставлена возможность отправлять заявки на участии в соревнованиях |

Продолжение таблицы 3.2

|  |  |
| --- | --- |
| Название требования | Описание требования |
| Возможность проведения и получения результатов жеребьевки | Зарегистрированному пользователю должна быть предоставлена возможность проведения жеребьёвки при организации соревнований, либо получения результата в случае участия в соревнованиях |
| Возможность составления и получения результатов расписаний | Зарегистрированному пользователю должна быть предоставлена возможность получении результатов расписаний |
| Возможность проведения и создания соревнований | Зарегистрированному пользователю должна быть предоставлена возможность проводить соревнования, заносить информацию при организации соревнований, либо визуализация соревнований |
| Возможность подведения итогов | Зарегистрированному пользователю должна быть предоставлена возможность получения результатов итогов |
| Возможность просмотра собственного профиля и статистики | Зарегистрированному пользователю должна быть предоставлена возможность просмотра своего профиля и просмотра итогов и статистики |
| Возможность редактирования личной информации | Зарегистрированному пользователю должна быть предоставлена возможность редактирования личной, контактной информации |

Таким образом, установленные функциональные требования, основанные на пользовательских требованиях в полной мере, удовлетворяют поставленным задачам.

## **3.3 Диаграммы прецедентов**

UML диаграммы прецедентов составляют модель прецедентов (вариантов использования, use-cases). Диаграммы для данной системы представлена на рисунках 3.1 и 3.2.



Рисунок 3.1 – Диаграмма прецедентов неавторизированного пользователя

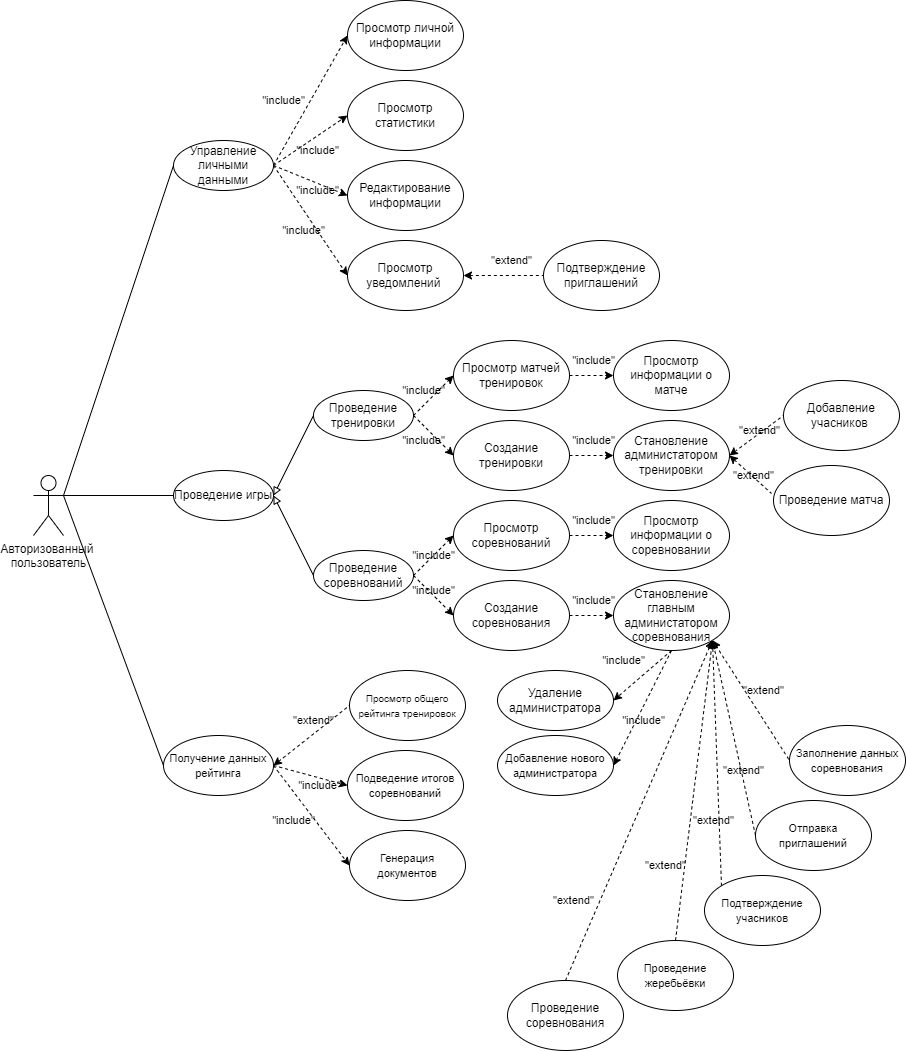


Рисунок 3.2 – Диаграмма прецедентов авторизированного пользователя

Данное приложение позволяет автоматизировать процесс проведения турниров международного, всероссийского, районного или клубного уровня. Где авторизованный пользователь может выступать как в роли организатора, так и в роли участника или зрителя.

## **3.4 Диаграммы последовательностей**

Для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности.

В разрабатываемой системе предусмотрено данные объекты взаимодействия:

* Пользователь. Может быть зарегистрированным и не зарегистрированным. Регистрация осуществляется посредством ввода электронной почты и пароля. Для использования функционала необходима регистрация
* Программное приложение. Обрабатывает действия пользователя и отправляет запросы на сервер. Отображает результаты обработки запросов, а также выводит определенное состояние пользователю.
* Сервер. Обрабатывает информацию и предоставляет доступ к ней.

Диаграмма последовательностей авторизации и регистрации неавторизированного пользователя изображена на рисунке 3.3.

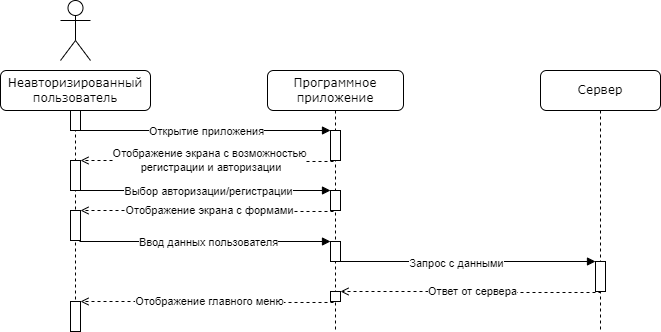


Рисунок 3.3 – Диаграмма прецедентов авторизации и регистрации неавторизированного пользователя

После этого пользователю доступен весь функционал приложения. Например, отправление заявки, как и жеребьёвка и расписание доступны в разделе “Игра”. Соответствующая диаграмма представлена на рисунке 3.4.

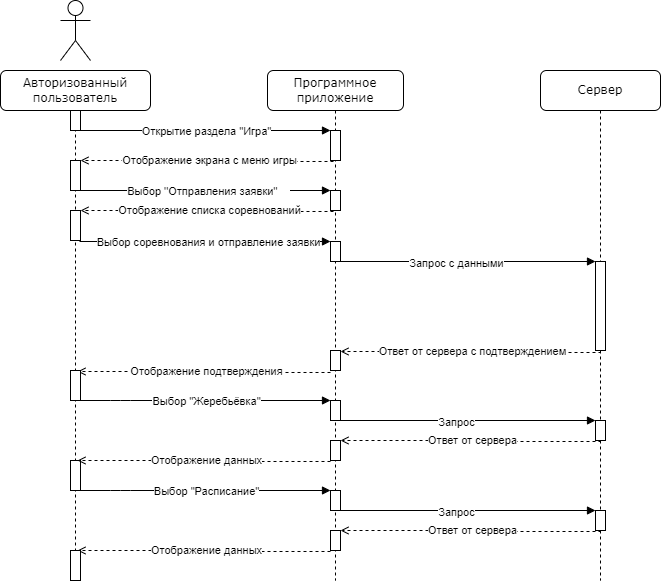


Рисунок 3.4 – Диаграмма прецедентов отправления заявки, получение результатов жеребьевки и получение расписания

## **3.4 Диаграммы деятельности**

Диаграммы деятельности – вид UML диаграмм, применяемых для моделирования динамических аспектов поведения системы. Как правило, они применяются, чтобы промоделировать последовательные (иногда и параллельные) шаги вычислительного процесса.

Общая диаграмма деятельности приложения представлена на рисунке 3.5.

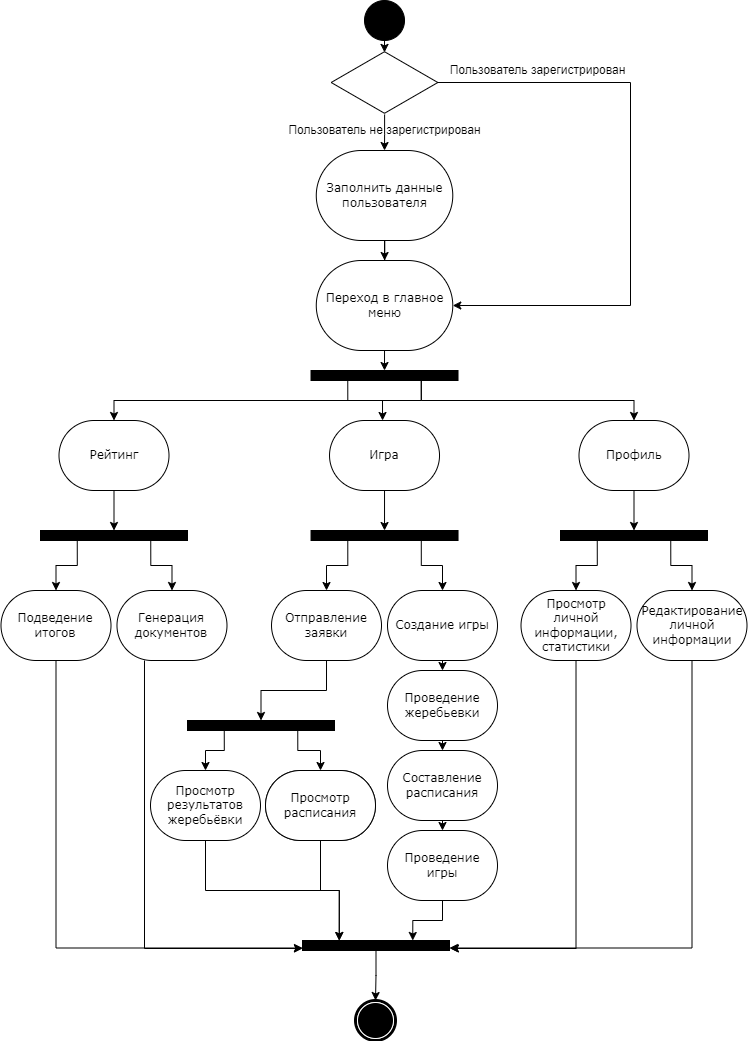


Рисунок 3.5 – Общая диаграмма деятельности

## **3.5 Проектирование базы данных**

Для работы приложения с базой данных, была выбрана Firebase realtime database, которая является облачной NoSQL базой данных. На рисунке 3.6 представлена спроектированная база данных, которая отображает приложение в целом.



Рисунок 3.6 – Спроектированная база данных

Данные в Firebase realtime database основываются на документно-ориентированном хранилище, хранятся в формате JSON и синхронизируются в режиме реального времени. База данных состоит из ключей и данных, поэтому её можно представить, как дерево JSON, размещенное в облаке. При добавлении данных в дерево JSON, оно становится узлом в уже существующей структуре JSON со связанным ключом. База данных имеет три главных коллекции:

* Users;
* Games;
* Lottery.

Коллекция users содержит множество ключей keyUser. Сам keyUser обозначает уникального пользователя и содержит следующие ключи:

* firstName — ключ, который содержит имя пользователя;
* secondName — ключ, который содержит фамилию пользователя;
* email — ключ, который содержит электронную почту;
* photo — ключ, который содержит фото пользователя или стандартное изображение;
* phone — ключ, который содержит номер телефона пользователя;
* about — ключ, который содержит информацию о пользователе;
* overallRating — ключ, который содержит данные о рейтинге пользователя;
* notification — ключ, который содержит коллекцию ключей KeyNotifications. Где элемент KeyNotifications является уникальным уведомлением либо приглашением. В свою очередь keyNotifications содержит следующие ключи:
  + title — заголовок уведомления либо приглашения;
  + body — тело уведомления либо приглашения;
  + keyCompetitions — содержит ключ определенного соревнования.

Коллекция Games содержит два ключа:

* training — ключ, который содержит коллекцию ключей keyMatch. Где keyMatch обозначает уникальный матч тренировки, который содержит ключи:
  + firstTeam — ключ, который обозначает команду и содержит коллекцию ключей keyUser. keyUser обозначает определенного пользователя;
  + secondTeam — ключ, который обозначает команду и содержит коллекцию ключей keyUser. keyUser обозначает определенного пользователя;
  + GeneralAdmin — ключ, который содержит keyUser пользователя;
  + adminRights — ключ, который содержит коллекцию ключей keyUser. keyUser обозначает определенного пользователя;
  + resultsMatch — ключ, который содержит результаты матча;
  + score — ключ, который содержит счёт матча.
* competitions — ключ, который содержит коллекцию ключей keyCompetitions. Где keyCompetitions обозначает уникальное соревнование, которое содержит ключи:
  + filedRequests — ключ, который обозначает множество пользователей отправивших заявки на участие. Содержит список ключей keyUser. keyUser обозначает определенного пользователя;
  + acceptedRequests — ключ, который обозначает множество подтвержденных пользователей. Содержит список ключей keyUser. keyUser обозначает определенного пользователя;
  + title — ключ, содержащий заголовок соревнования;
  + body — ключ, содержащий тело соревнования;
  + regulations — ключ, содержащий регламенты соревнования;
  + logoCompetitions — ключ, содержащий логотип соревнования;
  + generalAdmin — ключ, который содержит keyUser пользователя
  + adminRights — ключ, который содержит коллекцию ключей keyUser. keyUser обозначает определенного пользователя;
  + theLot — ключ, который содержит ключ KeyLottery. KeyLottery обозначает определенную жеребьёвку;
  + resultsCompetitions — ключ, который содержит результаты соревнования;
  + startDate — ключ, который содержит дату и время начала соревнования;
  + finishDate — ключ, который содержит дату и время окончания соревнования;
  + place — ключ, который содержит информацию о месте проведения соревнования.

Коллекция lottery содержит список жеребьёвок с ключом keyLottery. keyLottery содержит список этапов соревнования с ключом keyStage. keyStage содержит список матчей с ключом keyMatch. keyMatch обозначает матч и содержит список ключей:

* firstTeam — ключ, который обозначает команду и содержит коллекцию ключей keyUser. keyUser обозначает определенного пользователя;
* secondTeam — ключ, который обозначает команду и содержит коллекцию ключей keyUser. keyUser обозначает определенного пользователя;
* resultsMatch — ключ, который содержит результаты матча;
* score — ключ, который содержит счёт матча.
* date — ключ, который содержит дату и время матча;
* place — ключ, который содержит информацию о месте проведения соревнования.

# 4 РЕАЛИЗАЦИЯ

## **4.1 Выбор типа разработки**

При проведении анализа рынка возможностей разработки приложений, был выбран кроссплатформенный подход, который подразумевает использование одного набора технологий и инструментов для разработки на нескольких платформах, что помогает намного быстрее адаптировать разрабатываемое приложение и экономить значительное количество ресурсов [10]. Кроссплатформенный подход с каждым годом привлекает всё больше внимание разработчиков, благодаря высокой скорости в разработке и развертывание приложений, низкой стоимости, кроссплатформенности и использованию единого технологического стека.

Современное приложение, для лучшей конкуренции требует размещение сразу на нескольких платформах. В связи с этим, кроссплатформенный подход является более лучшим вариантом, в отличие от нативного подхода, он расходует намного меньше ресурсов.

## **4.2 Выбор кроссплатформенных фреймворков**

Каждый из рассмотренных фреймворков обладает как достоинствами, так и недостатками. Для того чтобы выбрать наиболее подходящий фреймворк для разработки приложения, их необходимо сравнить. В следствие обзора кроссплатформенных фреймворков можно составить таблицу, для сравнения по основным критериям (таблица 4.1).

Таблица 4.1 – Сравнение кроссплатформенных фреймворков

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии сравнения | Flutter | React Native | Xamarin |
| Компиляция IOS | AOT, JIT | Интерпретатор | AOT |
| Компиляция Android | + DartVM | JIT | JIT/AOT |
| Платформы | IOS, Android, Google Fuchsia, Web, Desktop | IOS, Android | IOS, Android, Windows, Mac OS |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| Критерии сравнения | Flutter | React Native | Xamarin |
| Проектирование пользовательского интерфейса | Собственные виджеты | Нативные контроллеры пользовательского интерфейса | Нативные контроллеры пользовательског о интерфейса |
| Качество пользовательского интерфейса | Очень высокое | Высокое | Среднее |
| Стоимость | Бесплатный | Бесплатный | Часть бесплатная, но имеет платные пакеты |
| Производительность | Очень высокая | Средняя, близкая к высокой | Высокая, близка к нативной |

Таким образом, наиболее подходящим фреймворком для разработки кроссплатформенного мобильного приложения был выбран Flutter.

## **4.3 Выбор архитектур управления состоянием**

При рассмотрении фреймворка Flutter существует множество архитектур управления состоянием. Основными являются:

* Native state;
* Provider (Scoped Model);
* BLoC (Cubit).

Native state — все, что используется во Flutter, состоит из виджетов. Они могут быть видимыми, невидимыми, содержать дочерние виджеты и взаимодействовать между собой. Каждый из них может быть, как виджетом без состояния (Stateless Widget), так и виджетом, у которого есть состояние (Stateful Widget). Основное отличие —  возможность повторно отрисовывать виджеты во время выполнения приложения. Stateless Widget будет отрисовываться только один раз и является неизменяемым. Stateful Widget может отрисовываться множество раз в зависимости от изменения внутреннего состояния виджета (рисунок 4.1).



Рисунок 4.1 – Визуальное представление Native state

Для создания Stateful Widget потребуется создать 2 класса:

* Первый класс должен наследоваться от Stateful Widget, который в свою очередь наследуется от Widget и является неизменяемым. Экземпляр этого класса не пересоздается при каждой отрисовке и используется для хранения переданных параметров и инициализации состояния.
* Второй —  класс состояния, который имеет доступ к Stateful Widget через внутреннее свойство и занимается непосредственно отрисовкой состояния, реагируя на его изменение.

Глобальное состояние приложения, так же, как и локальное, может быть реализовано с помощью Stateful Widget. Этот виджет создается в самом верхнем узле приложения и передается вниз по виджетам с помощью InheritedWidget. Каждый дочерний виджет приложения может получить доступ к виджету глобального состояния для изменения и использования его полей.

Преимуществами такого подхода является простота и скорость внедрения в приложение с небольшим количеством экранов, которая выражается в отсутствии каких-либо дополнительных библиотек.

Из недостатков можно выделить:

* представление и бизнес-логика никак не разделены;
* сложность в модульном тестировании;
* сложность в масштабировании и поддержке;
* большое количество кода, который не может быть переиспользован;

Этот подход желательно применять только к очень маленьким приложениям с фиксированным количеством экранов.

Provider (Scoped Model) — тип архитектуры позволяет вынести бизнес-логику из представления и дает возможность переиспользовать эту логику в разных модулях системы. Результат достигается с помощью создания модели и реагирования подписанных виджетов на ее изменение (рисунок 4.2).

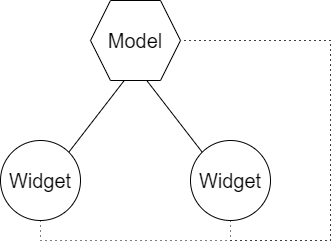


Рисунок 4.2 – Визуальное представление Provider (Scoped Model)

Для локального состояния виджета первым делом необходимо создать модель со всеми полями, которые будут использоваться в виджете. После изменения каждого поля необходимо сообщить подписчикам, что модель изменилась, и выполнить отрисовку подписанных виджетов.

Чтобы подписать виджет на модель, используется класс ChangeNotifierProvider, который является частью библиотеки provider. Подписка происходит непосредственно к тому виджету, который будет зависеть от данных из созданной модели.

Глобальное состояние ничем не отличается от локального, кроме того, что модель подписывается к самому верхнему виджету приложения.

Из преимуществ можно выделить разделение бизнес-логики и представления с помощью создания моделей и event-based-архитектуры. Тестировать такие модели легче, чем в Native state, за счет отсутствия дополнительных усилий на создание виджетов, в которых это состояние используется. Немаловажно, что этот тип архитектуры поддерживается Google, так что можно не беспокоиться о популярности и поддержке этого решения.

Одна из основных проблем такой архитектуры — сложность в понимании того, какое свойство было изменено и с какой модели произошло уведомление виджетов об изменении. Вариантом решения этой проблемы является соглашение на уровне команды о введении так называемых экшенов. Это единственное место, где модель может вызвать метод уведомления либо другие экшены. Также вам нужно быть готовым, что многие вещи вроде persist-хранилища моделей недоступны из коробки и придется написать большое количество сопровождающего кода для их внедрения.

Provider отлично подходит для средних проектов, в которых нет большого зацепления между модулями.

BLoC (Cubit) — шаблон, созданный Google для управления сложным состоянием приложения, основываясь на реактивной парадигме (рисунок 4.3).

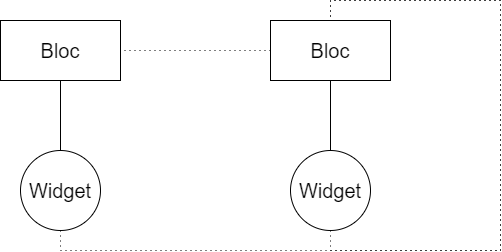


Рисунок 4.3 – Визуальное представление BLoC

Основная идея заключается в том, что наше приложение разбито на модули, реализующие бизнес-логику. Каждый модуль имеет одну или несколько Sink, которые являются некоторым входным потоком для агрегирования событий извне. В качестве выходных данных выступает Stream, который определяет асинхронный формат данных для наших виджетов. Чтобы воспользоваться модулем на уровне виджета, применяют StreamBuilder, который управляет потоком данных и автоматически решает проблемы подписки и перерисовки дочернего дерева виджетов.

Несмотря на это, использовать BLoC в чистом виде — достаточно сложная работа, поскольку надо применять библиотеку RxDart для манипуляции с потоками, вручную отписываться от потоков, иначе можно получить серьезную утечку памяти на больших приложениях. С целью решения этих проблем была изобретена библиотека BLoC, которая по максимуму упростила использование этого шаблона и предоставила удобное API для управления состоянием с возможностью легкого тестирования модулей.

Сам подход интересный и имеет большое количество преимуществ:

* богатое API при работе с потоками, что позволяет их легко группировать, совмещать и трансформировать;
* группировка логики в одном месте;
* легкость в тестировании состояния с сайд-эффектами за счет встроенного в Dart API тестирования потоков;
* минимальное количество отрисовок благодаря использованию StreamBuilder.

Наиболее подходящим для данной работы из всех перечисленных архитектур управления состоянием является BLoC (Cubit). На рисунках 4.4 – 4.6 приведены примеры класса Cubit, его возможные состояния и отклики экрана на состояния.



Рисунок 4.4 – Реализация класса Bloc (Cubit)

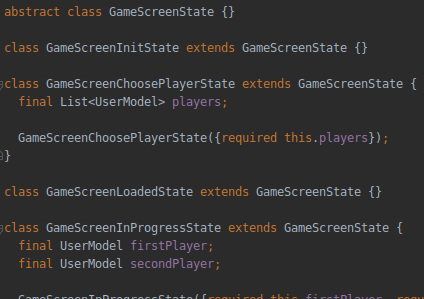


Рисунок 4.5 – Состояния, связывающие Bloc (Cubit) c экраном

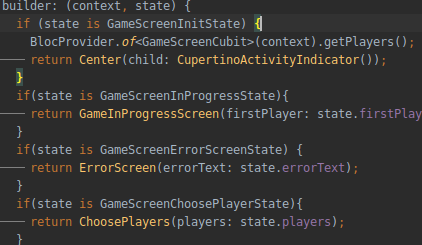


Рисунок 4.6 – Отклики экрана на различные состояния Bloc (Cubit)

## **4.4 Устройство приложения**

Все пользователи мобильного приложения могут зарегистрироваться или авторизоваться под своей почтой. При регистрации пользователя осуществляется соответствующий вызов одного из методов класса, что приводит к добавлению данных в Firebase. После авторизации, идёт сбор данных из БД и их вывод в виджеты. При их отсутствии дается возможность внесения данных, в обратном случае их редактирование. На рисунке 4.7 – 4.9 приведено взаимодействие с Firebase Realtme Database на примере авторизации.

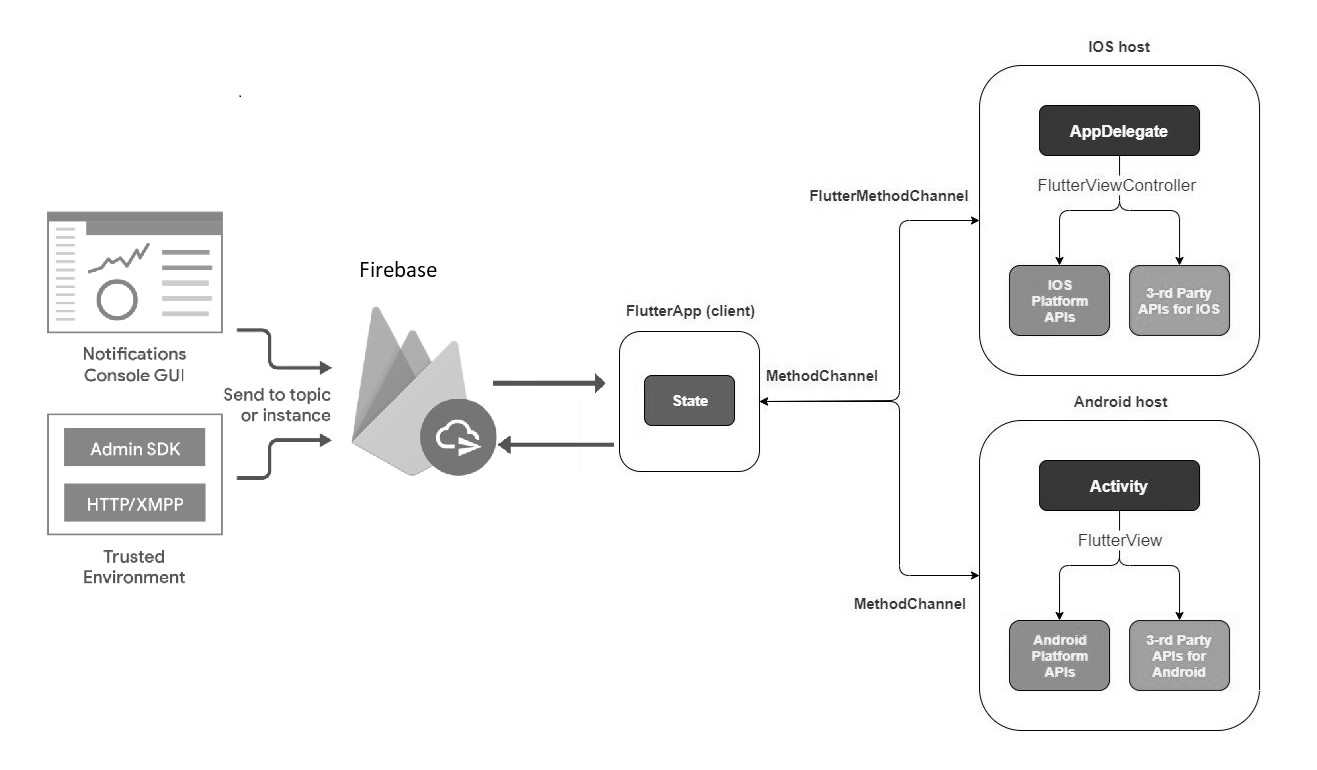


Рисунок 4.7 – Взаимодействие Firebase c приложением

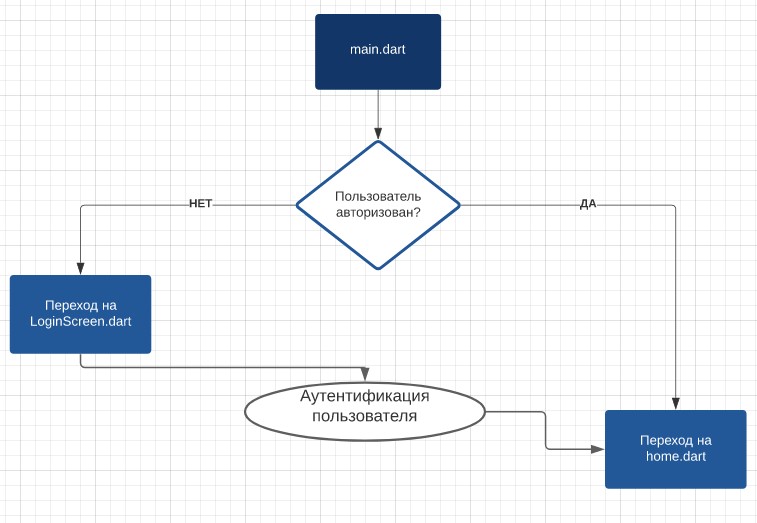


Рисунок 4.8 – Взаимодействие c приложением на примере авторизации

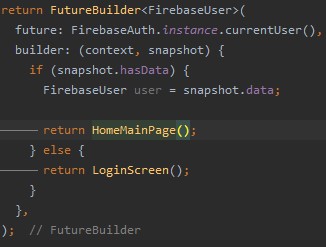


Рисунок 4.9 – Пример взаимодействия c приложением на примере авторизации

При использовании приложения, пользователь может задействовать 4 этапа архитектурного взаимодействия. На первом этапе, Flutter Framework инициализирует слой абстракции, при помощи которой происходит взаимодействие с функцией рендеринга Flutter Engine. После начинают отображаться и выстраиваться виджеты, компоненты интерфейса приложения из локальной памяти устройства с помощью GPU. На втором этапе, данные в формате документа записываются в базу данных Firebase через статичные виджеты. На следующем этапе, по запросу динамических виджетов поступают данные из БД. В основе мобильного приложения лежит бессерверная архитектура с использованием сервисов Google Cloud. В приложении используется подход взаимодействия клиента с сервером. Клиентской частью выступает само мобильное приложение, а серверной частью выступает Firebase (рисунок 4.10).

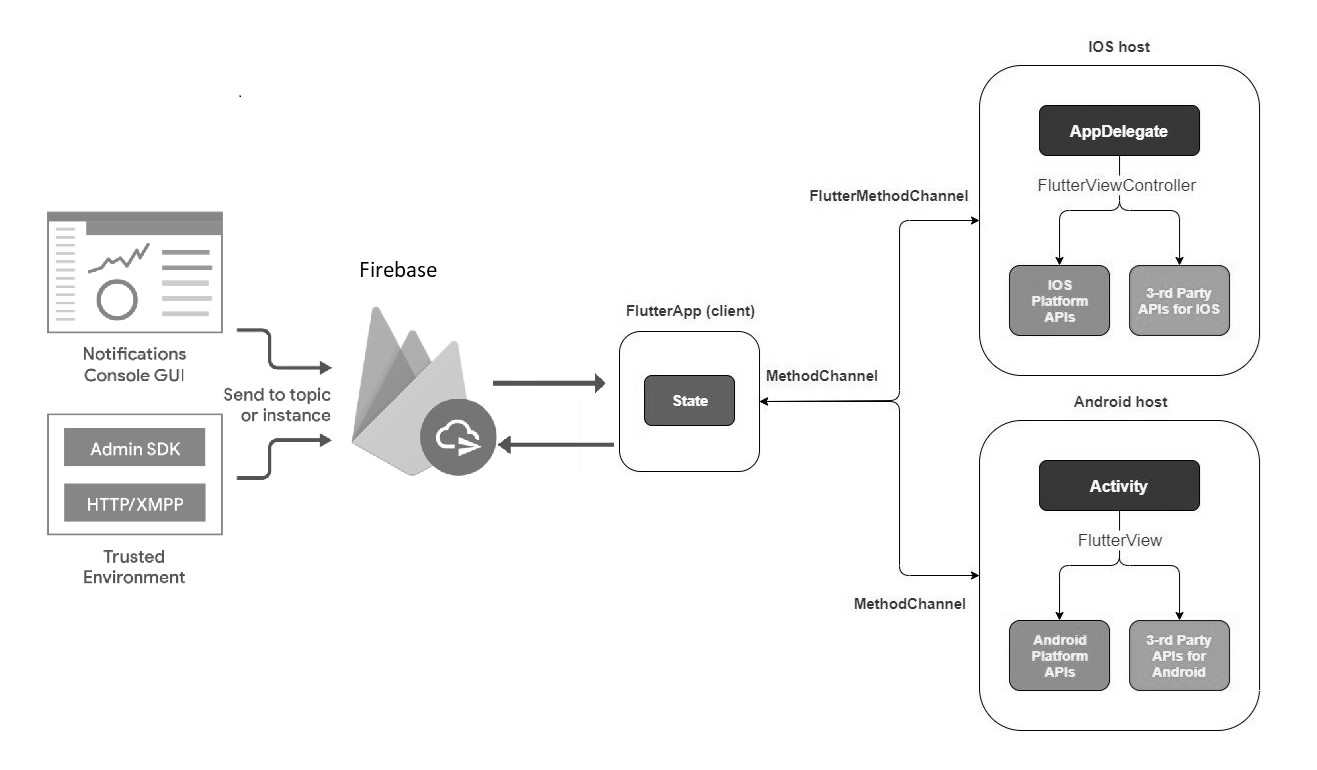


Рисунок 4.10 – Клиент-серверная архитектура приложения

Таким образом архитектура кроссплатформенного программного приложения для проведения соревнований по настольному теннису представляет собой двухзвенную клиент серверную архитектуру, состоящую из клиента и веб-сервера, коим выступает Firebase.

На основе приложения, можно получить диаграмму компонентов. Такая диаграмма определяет состав программных компонент, а также устанавливает зависимости между ними. На рисунке 4.11 представлена диаграмма компонентов, которая состоит из клиента и веб сервера. Веб сервером выступает облачная база данных Firebase, которая позволяет мобильному приложению использовать методы, предоставляемые веб сервером, позволяет взаимодействовать с данными.



Рисунок 4.11 – Двухзвенная клиент-серверная архитектура приложения

## **4.5 Адаптивность приложения**

Разработка приложения чаще всего ведется под размеры одного макета дизайна. Однако устройства имеют кардинально разные размеры экранов. Появляется необходимость в адаптивности приложения. Примеры реализации адаптивности представлены на рисунках 4.12 – 4.14



Рисунок 4.12 – Передача размеров макета

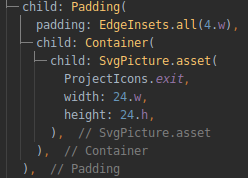




Рисунок 4.13 – Указание статических размеров и получения адаптивных размеров

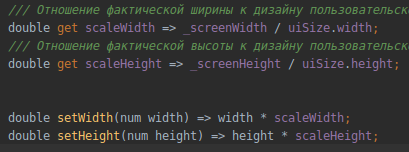


Рисунок 4.14 – Расчёт адаптивных размеров

Для демонстрации работы адаптивности было запущено кроссплатформенно приложение в браузере Google Chrome (рисунок 4.15) и в операционной системе Android (рисунок 4.16).

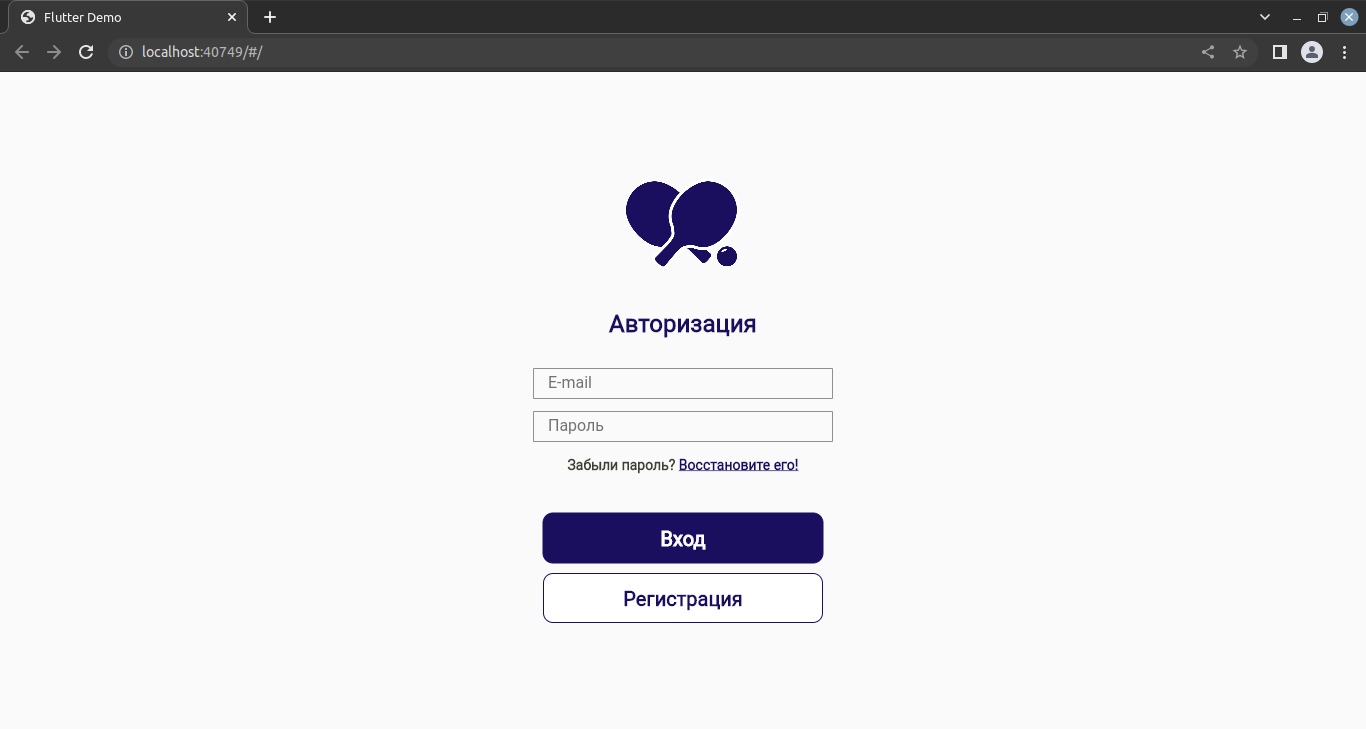


Рисунок 4.15 – Демонстрация работы в браузере Google Chrome

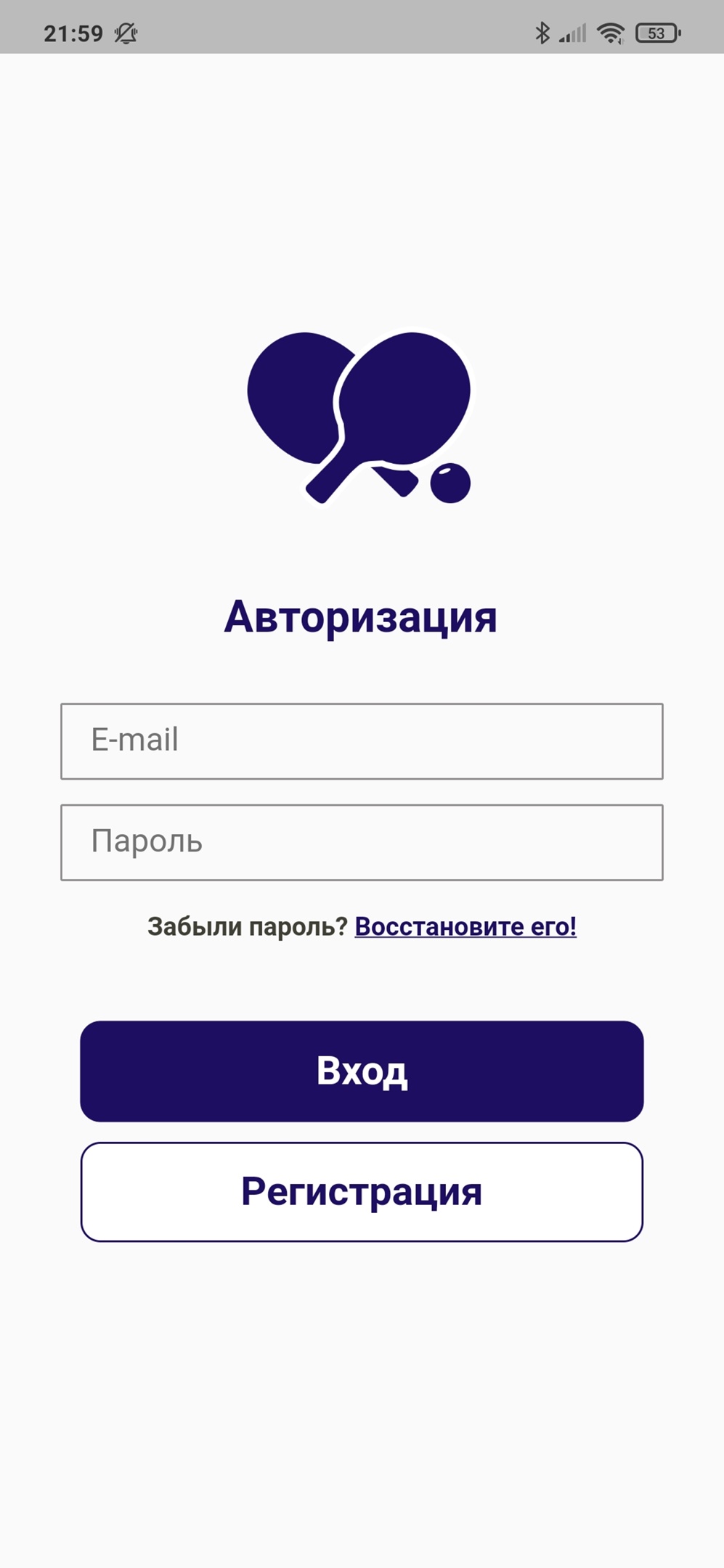


Рисунок 4.16 – Демонстрация работы под операционной системой Android

## **4.6 Интерфейс программного приложения**

При запуске приложения открывается экран авторизации (рисунок 4.17), где доступен экран регистрации (рисунок 4.18) и экран восстановления пароля (рисунок 4.19). В случае неверных данных будет открыта ошибка (рисунок 4.20)

|  |  |
| --- | --- |
| C:\Users\l1der\AppData\Local\Microsoft\Windows\INetCache\Content.Word\YsNWp4Bupws.jpg  Рисунок 4.17 – Экран авторизации | Рисунок 4.18 – Экран регистрации |
| C:\Users\l1der\AppData\Local\Microsoft\Windows\INetCache\Content.Word\LvCi9kOOeJg.jpg  Рисунок 4.19 – Экран восстановления пароля | Рисунок 4.20 – Открытие диалога с ошибкой |

После регистрации пользователь попадает на экран подтверждения почты (рисунок 4.21) и на указанную почту приходит письмо. Также на экране подтверждения доступен экран смены почты (рисунки 4.22 - 4.23)

|  |  |
| --- | --- |
| C:\Users\l1der\AppData\Local\Microsoft\Windows\INetCache\Content.Word\SEFpnSbIr6w.jpg  Рисунок 4.21 – Экран подтверждения почты | Рисунок 4.22 – Экран смены почты |

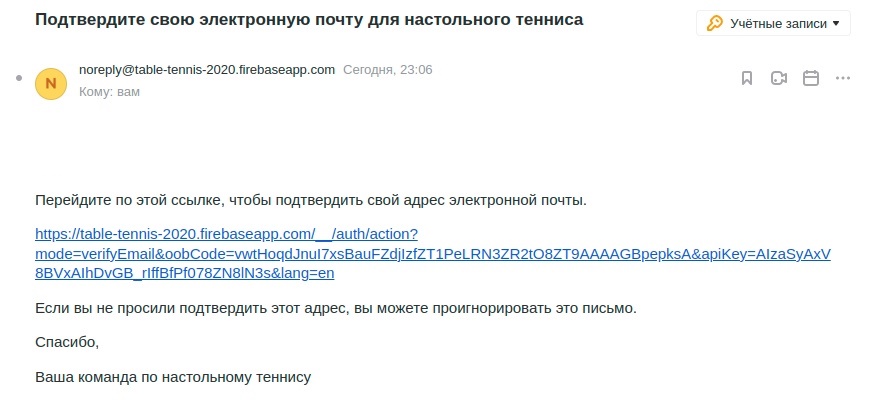


Рисунок 4.23 – Письмо подтверждения почты

После авторизации или подтверждения почты становятся доступны главные разделы:

* рейтинг;
* игра;
* профиль.

Первым открывается раздел профиль. В профиле отображается вся основная информация пользователя, а также наличие (рисунок 4.24) или отсутствие (рисунок 4.25) уведомлений и приглашений.

|  |  |
| --- | --- |
| Рисунок 4.24 – Экран профиль с отсутствием уведомлений и приглашений | Рисунок 4.25 – Экран профиль с наличием уведомлений и приглашений |

В профиле существует возможность просматривать статистику тренировок и статистику матчей. При нажатии на матч будет осуществлен переход в профиль другого пользователя (рисунок 4.26).

Также существует возможность редактировать личную информацию пользователя (рисунок 4.27).

|  |  |
| --- | --- |
| Рисунок 4.26 – Просмотр профиля другого пользователя | C:\Users\l1der\AppData\Local\Microsoft\Windows\INetCache\Content.Word\TTaPx_pBEFA.JPG  Рисунок 4.27 – Экран редактирования личной информации |

В разделе профиль существует возможность загружать и удалять фотографии (рисунки 4.28 – 4.29).

В случае отсутствия фотографий, пользователю будет предложено выбрать готовые эмодзи – заглушки (рисунки 4.30 – 4.31).

|  |  |
| --- | --- |
| Рисунок 4.28 – Редактирование фотографии | Рисунок 4.29 – Удаление фотографии |
| Рисунок 4.30 – Выбор заглушки вместо фотографии | Рисунок 4.31 – Установка заглушки |

При переходе в раздел игра, пользователю становятся доступны подразделы:

* тренировка;
* соревнования.

При открытии подраздела тренировка, у пользователя появляется возможность просматривать списки активных тренировок (рисунки 4.32 – 4.33).

|  |  |
| --- | --- |
| Рисунок 4.32 – Подраздел тренировка с активными матчами | Рисунок 4.33 – Подраздел тренировка с отсутствующими матчами |

При переходе по кнопке создать тренировку, создаётся матч. У пользователя появляется возможность выбрать участников (рисунок 4.34 – 4.35) и провести тренировку (рисунок 4.36).

|  |  |
| --- | --- |
| Рисунок 4.34 – Экран старта тренировки | Рисунок 4.35 – Выбор игроков тренировки |



Рисунок 4.36 – Проведение тренировки

При открытии подраздела соревнования, у пользователя появляется возможность просматривать списки соревнований (рисунки 4.37 – 4.38).

|  |  |
| --- | --- |
| Рисунок 4.37 – Подраздел соревнования с активными соревнованиями | Рисунок 4.38 – Подраздел соревнования с отсутствующими соревнованиями |

При переходе по кнопке создать соревнование, открывается экран создания соревнования, где пользователь может указать всю информацию (рисунок 4.39).

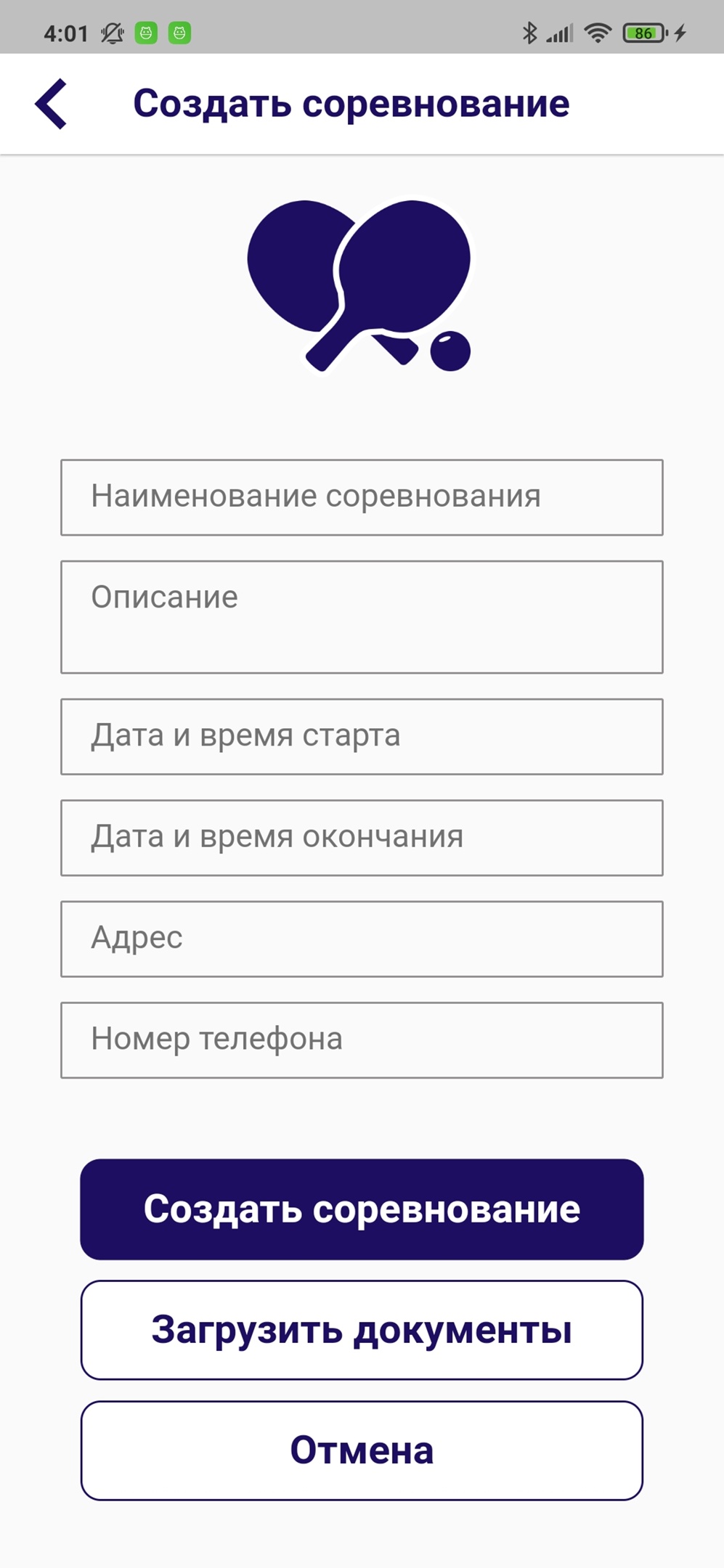


Рисунок 4.39 – Экран создания соревнования

После создания соревнования можно просматривать детали соревнования и настраивать (рисунки 4.40 – 4.41).

|  |  |
| --- | --- |
| Рисунок 4.40 – Экран детали соревнования с возможностью управления | Рисунок 4.41 – Экран детали соревнования |

После создания соревнования, появляется возможность:

* отправлять приглашения участникам (рисунок 4.42);
* подтверждать участников (рисунок 4.43);
* просматривать список участников;
* завершать приём заявок;
* после завершения приёма заявок, возможность проводить жеребьёвку;
* добавлять и удалять дополнительного администратора;
* удалять соревнование;
* проводить матчи соревнования.

|  |  |
| --- | --- |
| Рисунок 4.42 – Экран подтверждения участников | Рисунок 4.43 – Экран отправки приглашений |

В разделе игра существует возможность просмотра активных матчей тренировок и соревнований (рисунки 4.44 – 4.45)

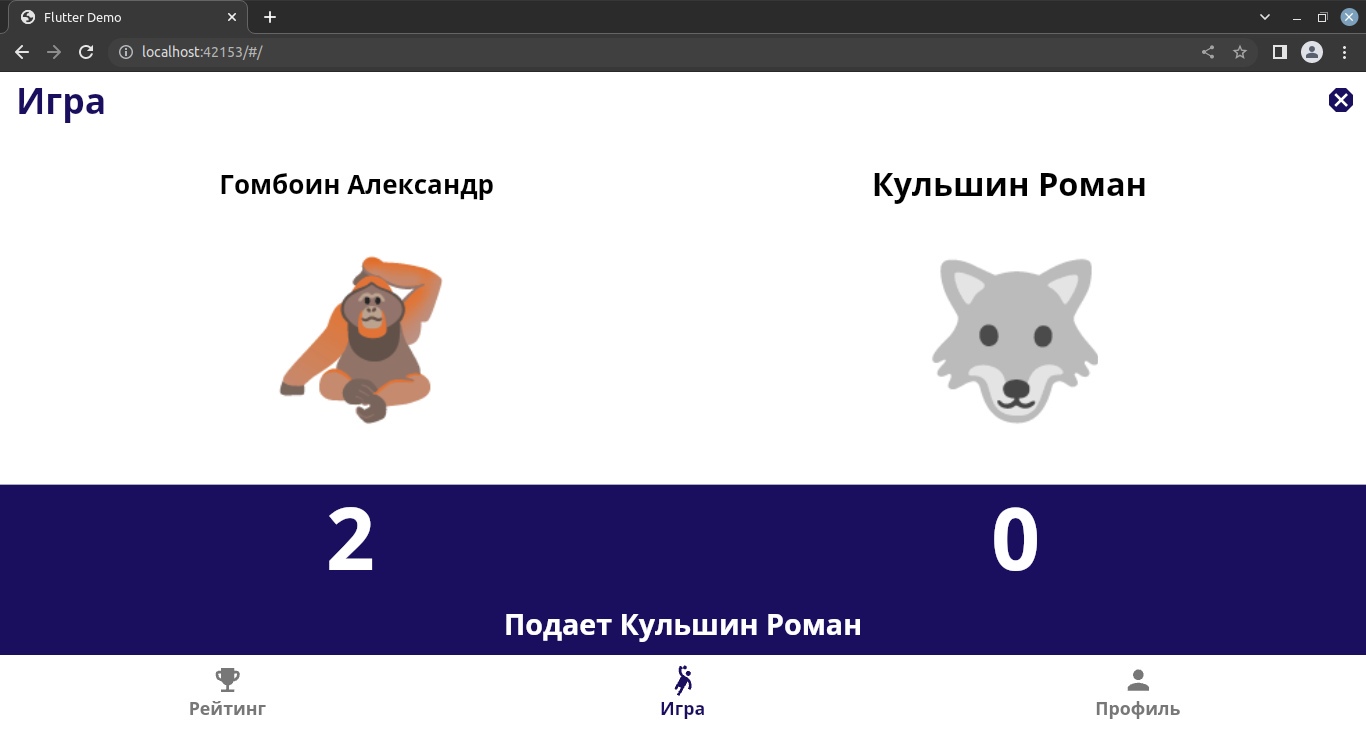


Рисунок 4.44 – Пример экрана просмотра активных матчей соревнований и тренировок в браузере Google Chrome



Рисунок 4.45 – Пример экрана просмотра активных матчей соревнований и тренировок в операционной системе Android

В разделе рейтинг доступна рейтинговая таблица тренировок (рисунок 4.46) и подведение итогов соревнований, с последующим завершением соревнования и формированием итогового документа (рисунок 4.47).

|  |  |
| --- | --- |
| Рисунок 4.46 – Экран рейтинг | Рисунок 4.47 – Экран подведения итогов |

# Заключение

Таким образом, в рамках магистерской диссертации удалось решить поставленные задачи:

* + был произведен анализ предметной области;
  + произведено изучение существующих аналогов;
  + произведен анализ типов разработки;
  + произведен анализ кроссплатформенных фрейморков;
  + спроектировано кроссплатформенное программное приложение;
  + разработано кроссплатформенное приложение.

Спроектированное и разработанное приложение позволяет автоматизировать процесс проведения турниров международного, всероссийского, районного или клубного уровня. Где авторизованный пользователь может выступать как в роли организатора, так и в роли участника или зрителя.

Дальнейшими направлениями работы должны стать мероприятия по улучшению и дополнению функциональности программной реализации спроектированного программного приложения.

# Список литературы

1. Настольный теннис в Томске [Электронный ресурс]. - режим доступа: URL: http://fnt.tomsk.ru/ (дата обращения – 14.04.2022)

2. Федерация настольного тенниса России [Электронный ресурс]. - режим доступа: URL: http://ttfr.ru/ (дата обращения – 14.04.2022)

3. ИСТОК-ТУРНИР [Электронный ресурс]. - режим доступа: URL: http://ystok.ru/tournament/ (дата обращения – 14.04.2022)

4. ТУРНИР 2.0 [Электронный ресурс]. - режим доступа: URL: https://werus.ru/obzor/ (дата обращения – 14.04.2022)

5. Flutter [Электронный ресурс]. - режим доступа: URL: https://flutter.dev/ (дата обращения – 15.04.2022)

6. Eric Windmill. Flutter in Action/ Eric Windmill. – Shelter Island: Manning Publications, 2020. – 341 p. (дата обращения – 15.04.2022)

7. Skia in Flutter & Fuchsia [Электронный ресурс]. - режим доступа: URL: https://skia.org/docs/dev/flutter/ (дата обращения – 15.04.2022)

8. React Native [Электронный ресурс]. - режим доступа: URL: https://reactnative.dev/ (дата обращения – 15.04.2022)

9. Xamarin [Электронный ресурс]. - режим доступа: URL: https://docs.microsoft.com/en-us/xamarin/ (дата обращения – 15.04.2022)

10. Кроссплатформенная разработка приложений [Электронный ресурс]. - режим доступа: URL: https://habr.com/ru/post/491926/ (дата обращения – 15.04.2022)