МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных наук

Кафедра программирования и информационных технологий

Приложение для организации встреч участников настольных игр «Onboard»

Курсовой проект

09.03.03 Информационные системы и технологии

Программная инженерия в информационных системах

Зав. кафедрой _____ С.Д. Махортов, д. т. н., профессор __.__.2023

Обучающийся _____ В.А. Ефремов, 3 курс, д/о

Обучающийся _____ Ю.А. Богданова, 3 курс, д/о

Обучающийся _____ Е.А. Бродская, 3 курс, д/о

Руководитель _____ В.С. Тарасов, ст. преподаватель

Руководитель _____ И.В. Клейменов, ассистент

Воронеж 2023

Содержание

Введение

Настольные игры уже давно стали общепризнанным хобби, пользующимся популярностью ещё с древних времён. Они являются прекрасным способом времяпрепровождения, предоставляющим самые различные варианты развлечения: от лёгких и расслабляющих игр до невероятно интеллектуальных и сложных, для всех возрастов и категорий населения. Но поиск соратников по увлечению и процесс организации игровых настольных сессий встречает множество трудностей на своём пути: ресурсы для поиска и размещения объявлений крайне разрознены, коммуникация между игроками не централизована, и весь этот функционал ложится на множество мелких сообществ в соцсетях и мессенджерах, что делает этот процесс крайне затруднительным. Ввиду этого игроки в настольные игры нуждаются в централизованном сервисе, предоставляющем максимально комфортный уровень сервиса непосредственно для них. Кроме того, аудитория настольных игр постоянно растёт по всему миру и зачастую отличается постоянностью. поэтому разработка мобильного приложения для решения данной задачи является актуальной.

1 Терминология (глоссарий) предметной области

— Мобильное приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для конкретной платформы (iOS, Android, Windows Phone и т. д.).

— Android — это операционная система с открытым исходным кодом, созданная для мобильных устройств на основе модифицированного ядра Linux.

— Android-приложение — программное обеспечение, предназначенное для работы на смартфонах, планшетах и других мобильных устройствах, разработанное для платформы Android.

— Фреймворк — программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

— Frontend — клиентская сторона пользовательского интерфейса к программно-аппаратной части сервиса.

— Backend — программно-аппаратная часть сервиса, отвечающая за функционирование его внутренней части.

— REST — архитектурный стиль взаимодействия компонентов распределённого приложения в сети.

— API — описание взаимодействия одной компьютерной программы с другой.

— Сериализация — это процесс преобразования объекта в поток байтов для сохранения или передачи в память, базу данных или файл.

— JSON (JavaScript Object Notation) — это открытый стандарт формата файла и обмена данными формат, который использует удобочитаемый текст, чтобы сохранить и передать данные объекты, состоящие из пар атрибут–значение и массивов (или других сериализуемых значений).

2 Постановка задачи

2.1 Цель создания системы

Основной целью данной работы является разработка мобильного приложения для поиска игроков в настольные игры с использованием Firebase REST API и его модулей Firebase Realtime Database и Firebase Auth. Главная задача заключается в создании функциональной и безопасной платформы, которая позволит пользователям находить друг друга и организовывать игровые сессии.

2.2 Требования к разрабатываемой системе

— Обеспечение авторизации и аутентификации пользователей;

— использование протокола передачи данных HTTP;

— приложение должно быть построено на трехуровневой архитектуре: клиент, сервер, база данных.

2.3 Задачи, решаемые в процессе разработки

2.3.1 Изучение Firebase REST API и его модулей

Первоначальная задача включает изучение основных концепций, возможностей и функциональности Firebase REST API, Firebase Realtime Database и Firebase Auth. Будут рассмотрены основные принципы работы с базой данных в реальном времени, аутентификация пользователей и методы взаимодействия с Firebase через REST API.

2.3.2 Анализ предметной области

Для установления необходимого функционала приложения и актуальности разработки требуется произвести анализ предметной области, включающий в себя обзор функционала аналогов приложения с подробным разбором их преимуществ и недостатков, а также анализ целевой аудитории данного приложения.

2.3.3 Проектирование архитектуры приложения

На основе анализа требований будет проведено проектирование архитектуры мобильного приложения. В этом этапе будут определены

основные компоненты и модули приложения, их взаимосвязь и функциональность. Также будет рассмотрено интеграция Firebase REST API и его модулей в архитектуру приложения, чтобы обеспечить эффективную обработку данных и безопасность пользователей.

2.3.4 Реализация мобильного приложения

После завершения проектирования будет приступлено к реализации мобильного приложения. В этом этапе будут использованы полученные знания о Firebase REST API и его модулях для разработки функциональности поиска игровых сессий, создания пользовательских профилей, обмена сообщениями и других функций, соответствующих требованиям к системе.

2.4 Моделирование системы

2.4.1 Диаграмма прецедентов

Диаграмма прецедентов – это диаграмма, которую используют для анализа различных систем. Она позволяет визуализировать различные типы ролей в системе и то, как эти роли взаимодействуют с системой. Диаграмма прецедентов представлена на рисунке 1.

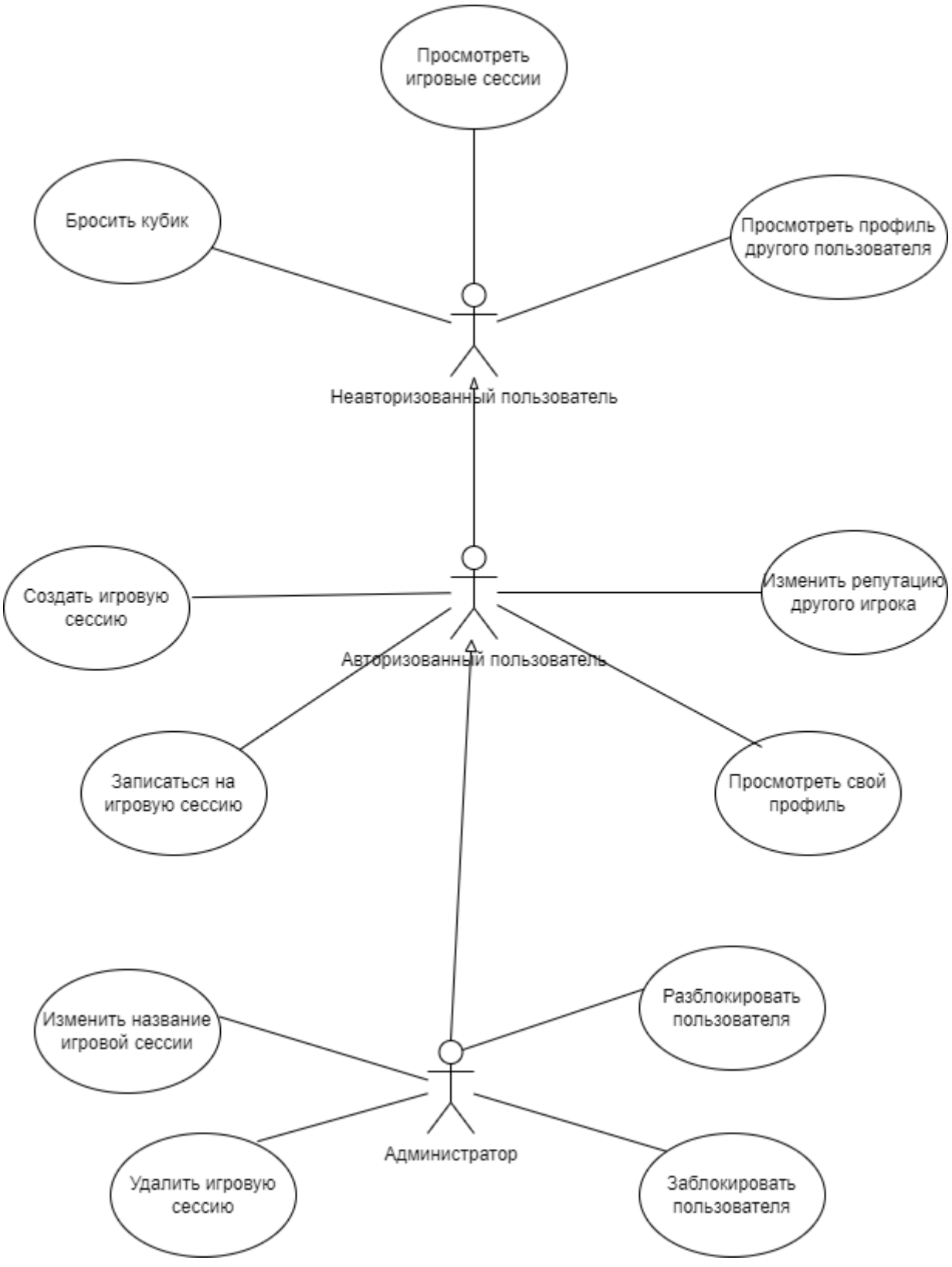

Рисунок 1 - Диаграмма прецедентов

2.4.2 Диаграмма последовательности

Диаграммы последовательностей используются для более детального описания логики сценариев использования. Диаграммы последовательностей

содержат объекты, которые взаимодействуют в рамках сценария, сообщения, которыми они обмениваются, и возвращаемые результаты, связанные с сообщениями. Диаграмма последовательности представлена на рисунке 2.

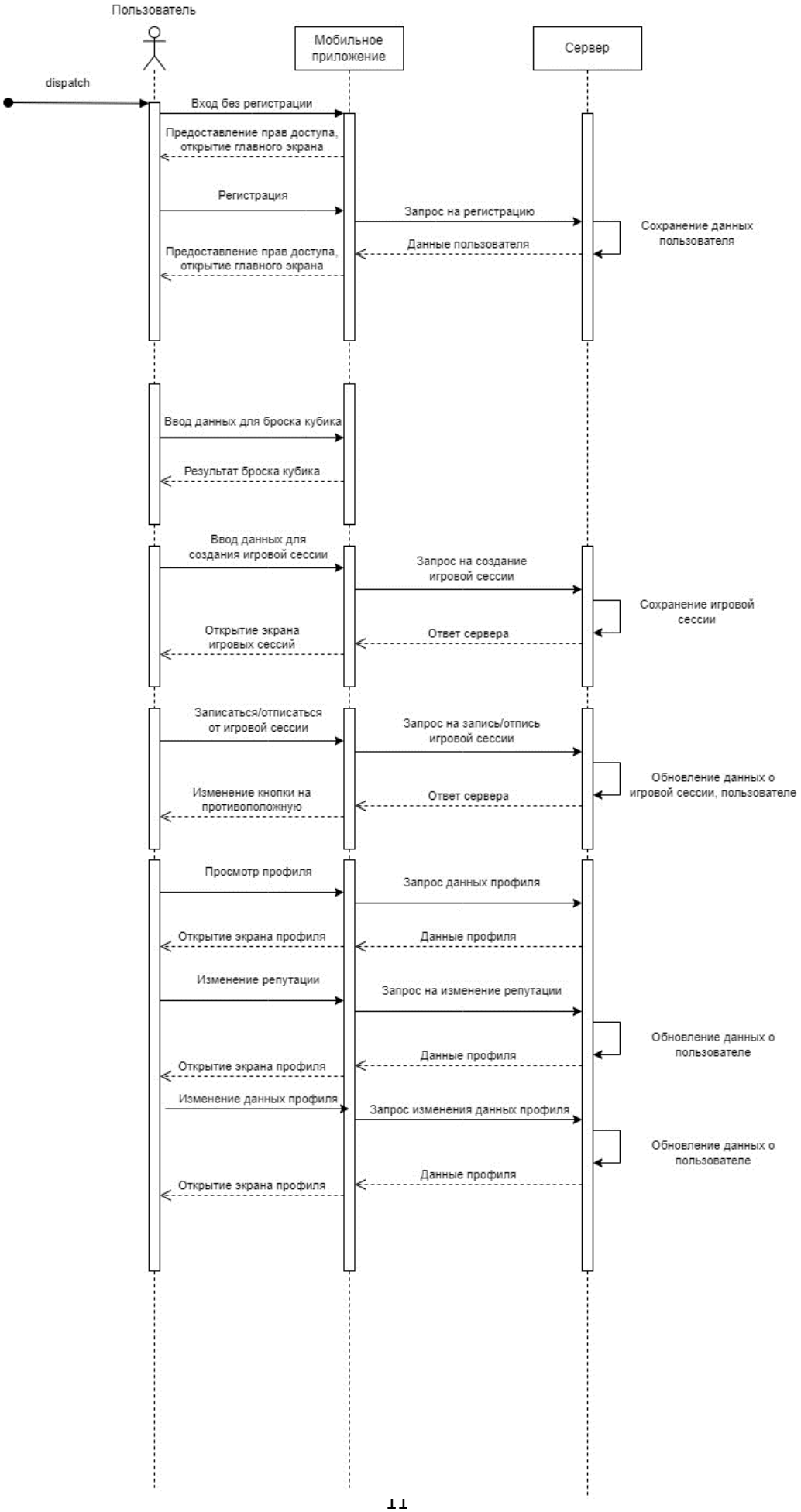
Пользователь
Мобильное приложение
Сервер
dispatch
Вход без регистрации
Предоставление прав доступа, открытие главного экрана
Регистрация
Запрос на регистрацию
Сохранение данных пользователя
Данные пользователя
Предоставление прав доступа, открытие главного экрана
Ввод данных для броска кубика
Результат броска кубика
Ввод данных для создания игровой сессии
Запрос на создание игровой сессии
Сохранение игровой сессии
Открытие экрана игровых сессий
Ответ сервера
Записаться/отписаться от игровой сессии
Запрос на запись/отпись игровой сессии
Обновление данных о игровой сессии, пользователе
Изменение кнопки на противоположную
Ответ сервера
Просмотр профиля
Запрос данных профиля
Открытие экрана профиля
Данные профиля
Изменение репутации
Запрос на изменение репутации
Обновление данных о пользователе
Данные профиля
Открытие экрана профиля
Изменение данных профиля
Запрос изменения данных профиля
Обновление данных о пользователе
Данные профиля
Открытие экрана профиля

Рисунок 2 - Диаграмма последовательности

2.4.3 Диаграмма активностей

Диаграмма активностей позволяет более детально визуализировать конкретный случай использования. Это поведенческая диаграмма, которая иллюстрирует поток деятельности через систему. Диаграмма активностей представлена на рисунке 3.

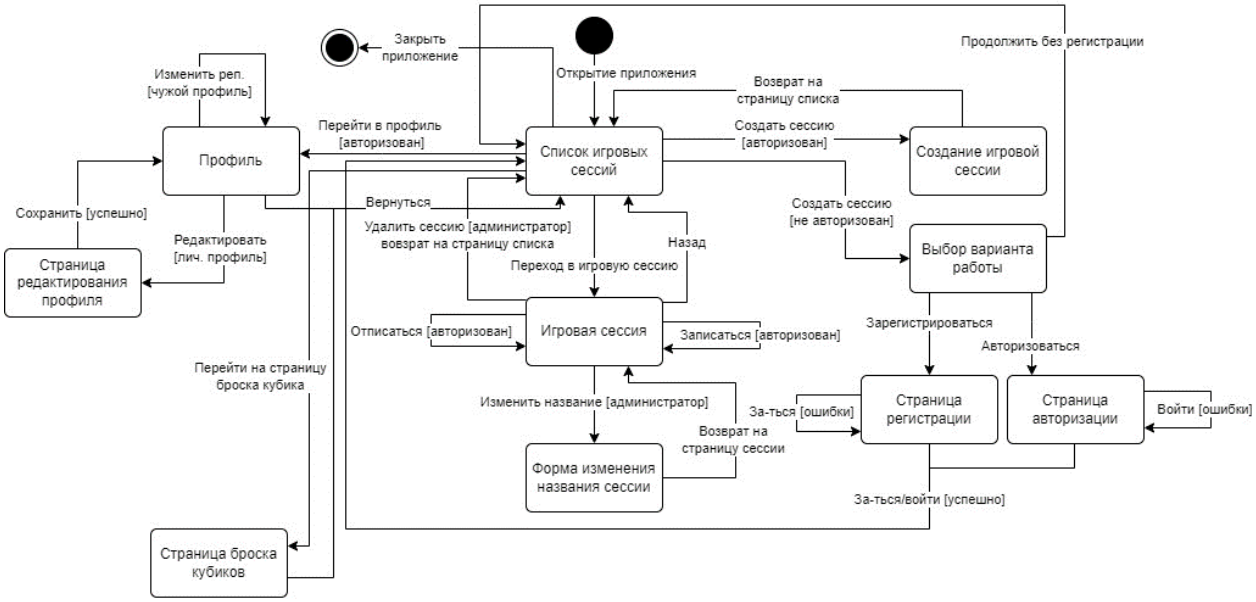

Рисунок 3 - Диаграмма активностей

3 Изучение Firebase REST API и его модулей

3.1 Firebase REST API

Firebase REST API предоставляет разработчикам широкий набор возможностей для работы с Firebase-платформой и ее сервисами. В контексте разработки мобильного приложения для поиска игроков в настольные игры, следующие возможности Firebase REST API могут быть применены:

3.1.1 Аутентификация пользователей с помощью Firebase Auth

Firebase Auth предоставляет надежную систему аутентификации и авторизации пользователей. С помощью Firebase REST API можно реализовать функциональность регистрации и входа пользователей в приложение, а также управление их учетными записями.

3.1.2 Работа с Firebase Realtime Database

Firebase Realtime Database предоставляет возможность реально-временного хранения и синхронизации данных. С помощью Firebase REST API можно осуществлять чтение и запись данных в базу данных, а также организовывать сложные запросы для извлечения необходимой информации.

3.2 Модули Firebase: Firebase Realtime Database и Firebase Auth

3.2.3 Модуль Firebase Realtime Database

Firebase Realtime Database[1] предлагает гибкое решение для хранения и синхронизации данных. В рамках разработки мобильного приложения для поиска игроков в настольные игры, Firebase Realtime Database может быть использован для следующих задач:

— Хранение профилей пользователей: Firebase Realtime Database позволяет сохранять информацию о пользователях, такую как их имя, возраст, предпочтения в играх и другие данные, необходимые для создания и управления профилями пользователей.

— Хранение данных об игровых сессиях: Firebase Realtime Database может быть использован для хранения информации о существующих игровых сессиях, включая их название, описание, местоположение, время начала и другие атрибуты, которые помогут пользователям находить и присоединяться к интересующим их игровым сессиям.

3.2.4 Модуль Firebase Auth

Firebase Auth[2] предоставляет мощную систему аутентификации пользователей. Для мобильного приложения по поиску игроков в настольные игры Firebase Auth может быть использован для решения следующей задачи:

Регистрация и вход пользователей: Firebase Auth позволяет пользователям создавать новые учетные записи и входить в систему с помощью различных методов аутентификации, в частности с помощью электронной почты и пароля.

3.3 Применение Firebase REST API и его модулей в мобильном приложении

Совместное использование Firebase REST API и его модулей в разрабатываемом мобильном приложении позволит достичь следующих результатов:

— Реализация аутентификации пользователей: Firebase Auth позволит пользователям регистрироваться и входить в систему, обеспечивая безопасность и управление учетными записями.

— Хранение и синхронизация данных: Firebase Realtime Database будет использоваться для хранения информации о пользователях, игровых сессиях и других сущностях приложения, обеспечивая реально-временный доступ к данным и синхронизацию между устройствами.

Изучение и применение Firebase REST API и его модулей в разрабатываемом мобильном приложении позволит создать надежную и функциональную систему для поиска игроков в настольные игры, обеспечивая удобство использования и безопасность пользователей.

4 Анализ предметной области

4.1 Анализ целевой аудитории

По данным из доклада об анализе рынка настольных игр в России Пензенского Государственного Университета [9] основными потенциальными покупателями настольных игр являются люди в возрасте 26-35 лет, их доля составляет 40%, следующей возрастной категорией потенциальных покупателей являются люди в возрасте 36-45 лет – 23%, немного меньше доля людей в возрасте 19-25 лет – 22%, далее идёт возрастная группа 15-18 лет, составляющая 10%, за ними идут люди в возрасте 46-55 лет, составляющая 4% и самой меньшей возрастной группой являются люди в возрасте от 56 лет, составляющие 1%. Для исследования потребительских предпочтений при выборе настольных игр был поведен опрос методом анкетирования. Анкетирование проводилось в точках продаж настольных игр г. Пенза в сентябре 2017 года. В опросе приняло участие 120 человек.

Исходя из этих данных можно сделать вывод, что основной частью аудитории настольных игр являются люди в возрасте 26-45 лет, что говорит нам о том, что они являются молодёжью, заинтересованной в использовании современных технологий, в частности мобильных приложений, и могут стать аудиторией нашего приложения.

4.2 Обзор аналогов

Существует много способов поиска единомышленников для организации какого-либо мероприятия. Это может быть и обычная группа в социальной сети «VK», для онлайн-игр хорошо подойдет сервер на платформе «Discord» с чат-ботами, упрощающими сбор игроков. Также есть сервисы, предоставляющие информацию о мероприятиях в городе на основе выбранных пользователем интересов и возможность присоединиться к ним, например, Meetup[12] или “Кто куда”[13]. Но намного комфортнее пользоваться узконаправленным приложением, учитывающим тонкости

организации встречи для конкретного вида занятий, такими, как сервис Plink[11] для поиска тиммейтов для компьютерных игр, который основан на принципе приложений для знакомств, где пользователи отмечают друг друга на основе общих интересов, и в случае взаимного интереса они могут связаться и начать игру. Для крупных событий существуют приложения, позволяющие спланировать рассадку гостей, список персонала, рассылку билетов и т. д., такие, как русскоязычная платформа Timepad[10], но порой они достаточно сложны в использовании из-за большого количества опций и сложного интерфейса. К тому же, для организации небольших ивентов с одной тематикой такая обширная функциональность будет излишней, и в то же время пользователь не будет иметь возможность детализировать мероприятие в нужном объеме.

Таблица 1 - Аналоги приложения

Название	Онлайн-запись	Создание ивента	История ивентов	Фильтрация по времени	Репутация пользователя	Статистика пользователя	Мобильное приложение
Timepad	да	да	да	да	нет	нет	нет
Plink	да	нет	да	нет	нет	да	да
Meetup	да	да	да	Да	нет	да	да
“Кто куда”	да	нет	да	Да	нет	да	да

Таким образом, на основе анализа подобных приложений можно сделать вывод, что приложение для организации тематических мероприятий должно быть лаконичным, обладать простым интерфейсом и быть понятным для любого пользователя, и в то же время иметь функционал, учитывающий все нюансы выбранной тематики. При обзоре существующих решений не было найдено прямых аналогов разрабатываемой системы, так как ни одно из них не направлено на организацию сессий настольных игр, и, вследствие, не позволяет обеспечить должный уровень комфорта при поиске и организации

сессий. Приложение "Onboard", в свою очередь, дает возможность сузить круг потенциальных участников, конкретизировать запрос на поиск сессии и предоставляет инструментарий для релевантного подбора игроков.

5 Реализация

5.1 Технологический стек backend-разработки

— Firebase Realtime Database – бесплатная удалённая СУБД, разрабатываемая Google. Имеется интерфейс REST API. Основными плюсами данной удалённой СУБД является отсутствие необходимости поиска собственного локального хостинга.

— Firebase Auth – сервис авторизации пользователей от компании Google, предоставляющий различные способы входа в аккаунт, в частности через почту и пароль, и в дальнейшем использовать для аутентификации полученный персональный токен (далее idToken), а также предоставляющий персональные пользовательские аутентификаторы (далее uid).

— Django – фреймворк для веб-приложений на языке Python, использующий шаблон проектирования MVC.

— Docker – программное обеспечение для автоматизации развёртывания и управления приложениями в средах с поддержкой контейнеризации, контейнеризатор приложений.

5.2 Технологический стек frontend-разработки

— Kotlin – статически типизированный, объектно-ориентированный язык программирования, работающий поверх Java Virtual Machine и полностью совместимый с языком Java, что позволит интегрировать библиотеки, написанные на Java, в код на Kotlin.

— Android SDK – универсальное средство разработки мобильных приложений для операционной системы Android.

— Spring Android – это расширение Spring Framework, специально адаптированное для разработки клиентской части мобильных приложений под платформу Android. Он предоставляет интеграцию с Android SDK и инструментарий для взаимодействия с сетевыми сервисами и API.

— SQLite[8] – встраиваемая СУБД, подходящая для хранения локальных данных приложения.

5.3 Реализация базы данных

Для хранения данных была выбрана СУБД Firebase Realtime Database, хранящая данные в формате JSON. На рисунке 4 приведена ER-диаграмма используемой базы данных:

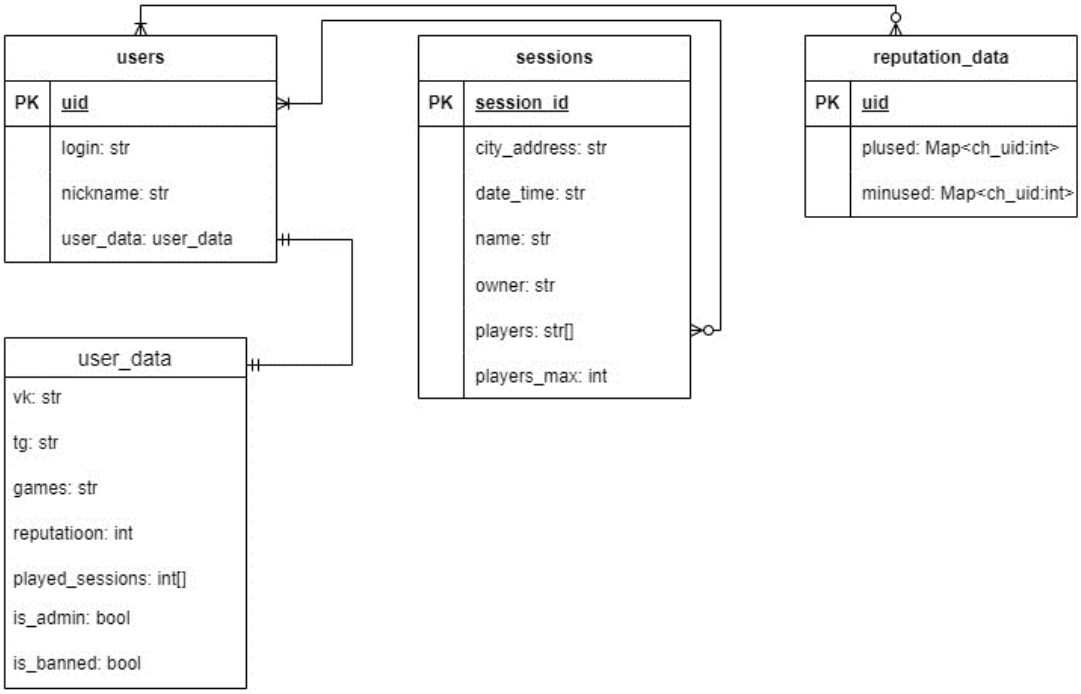

Рисунок 4 - ER-диаграмма базы данных

5.4 Реализация клиентской части

Для реализации клиентской части приложения было принято решение воспользоваться Android SDK, Spring Adroid и языком программирования Kotlin. Этот выбор сочетает в себе множество преимуществ и предоставляет разработчикам широкий набор инструментов для создания современных мобильных приложений на платформе Android.

Android SDK[6], или набор разработчика для Android, является ключевым компонентом при разработке приложений под Android. Он обеспечивает разработчикам доступ к различным API и сервисам, а также предоставляет богатый набор инструментов для разработки пользовательского интерфейса, управления жизненным циклом приложения и обработки событий.

Kotlin[5] – это современный язык программирования, полностью совместимый с Java и специально разработанный для платформы Android. Он обладает богатым набором функций и усовершенствований по сравнению с Java, что делает процесс разработки более эффективным и удобным.

Основным преимуществом Kotlin является его поддержка функционального программирования и расширений языка. Он позволяет разработчикам писать более компактный и читаемый код, сокращая количество шаблонного кода, необходимого в Java. Благодаря таким возможностям как null-безопасность и инференция типов, Kotlin помогает предотвратить множество ошибок на этапе компиляции и повысить общую надежность приложения.

Для дополнительной поддержки разработки клиентской части приложения на платформе Android было принято решение использовать фреймворк Spring Android[7]. Spring Android представляет собой расширение Spring Framework, специально адаптированное для разработки приложений под Android.

Одним из ключевых преимуществ Spring Android является его интеграция с Android SDK и удобный способ взаимодействия с различными сервисами и API. Фреймворк предоставляет удобные классы и абстракции для работы с сетевыми запросами, обработки ответов, управления сеансами связи и другими аспектами коммуникации между клиентом и сервером. Это значительно упрощает процесс разработки и позволяет разработчикам сосредоточиться на бизнес-логике приложения, минимизируя сложность взаимодействия с внешними сервисами.

В итоге, добавление Spring Android к выбору Android SDK и языка программирования Kotlin позволяет разработчикам использовать все преимущества Spring Framework и обеспечивает современные и эффективные инструменты для разработки клиентской части мобильного приложения под Android.

5.4.1 Форма для просмотра игровых сессий

Рисунок 5 демонстрирует функционал просмотра игровых сессий для пользователя (авторизованного или неавторизованного), который имеет возможность просматривать список актуальных игровых сессий, т.е. не завершённых и не начатых сессий, а также сессий, где не набрано нужное количество игроков. Список содержит краткую информацию: название, основная игра, дата, город, количество игроков. Каждая сессия имеет свой идентификатор (ID).

Сессии сортируются по дате от самых новых до самых старых.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

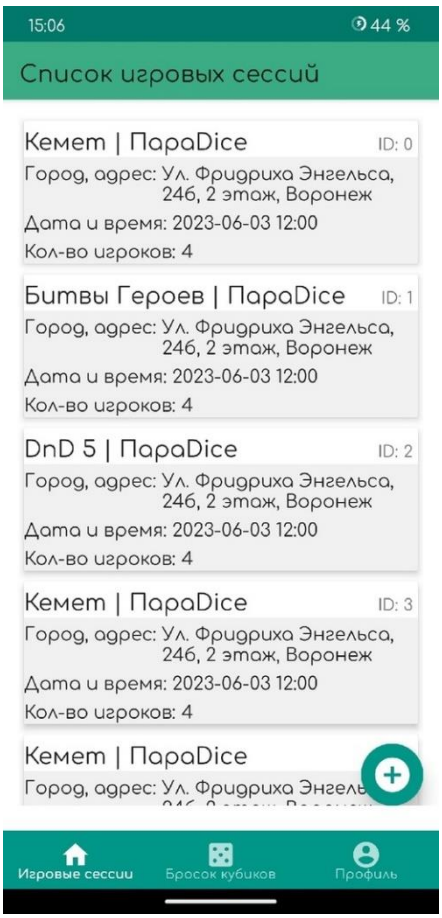

Рисунок 5 - Форма экрана списка игровых сессий

5.4.2 Форма экрана сессии

На экране, изображённом на рисунке 6, расположена информация о сессии:

— Название

— Адрес проведения

— Дата и время

— Количество игроков

— Список игр

— Список игроков

Список игроков содержит краткую информацию о каждом игроке: никнейм и его репутацию.

Ниже информации о сессии расположены три кнопки:

— Изменить название

— Удалить

— Записаться/Отписаться

Кнопка изменения названия доступна администраторам и позволяет изменить название текущей сессии. Кнопка удаления доступна создателю сессии и администратору и удаляет текущую сессию. Кнопка для записи и отписки доступна всем авторизованным пользователям и позволяет им записаться на сессию или отписаться от неё, добавляя их в список игроков данной сессии или удаляя из него.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

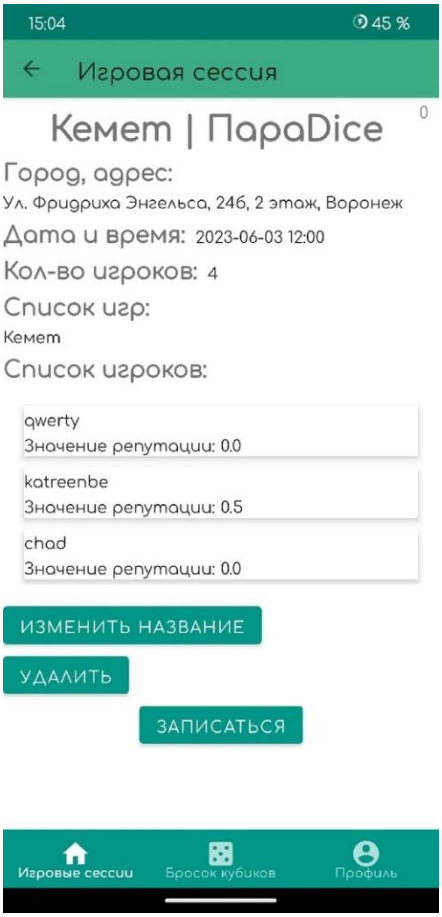

Рисунок 6 - Форма экрана сессии

5.4.3 Форма экрана создания сессии

На экране, изображённом на рисунке 7, расположены поля для ввода информации о сессии: название сессии, количество необходимых игроков, список игр, дата и время, адрес.

Ниже расположена кнопка кнопка создать, которая сохраняет сессию на сервере.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

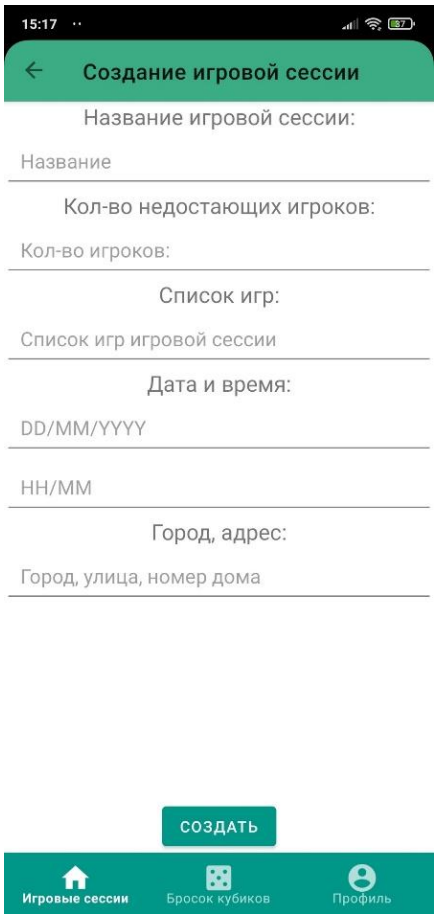

Рисунок 7 - Форма экрана создания сессии

5.4.4 Форма экрана выбора варианта работы

На экране, изображённом на рисунке 8, расположены три кнопки: войти, зарегистрироваться, продолжить без регистрации. Кнопки войти и

зарегистрироваться перенаправляют пользователя на соответствующие экраны. Кнопка продолжить без регистрации возвращает пользователя на главный экран.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

Рисунок 8 - Форма экрана выбора варианта работы

5.4.5 Форма экрана авторизации

На экране, изображённом на рисунке 9, отображены поля с вводом логина и пароля к аккаунту и кнопка войти.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

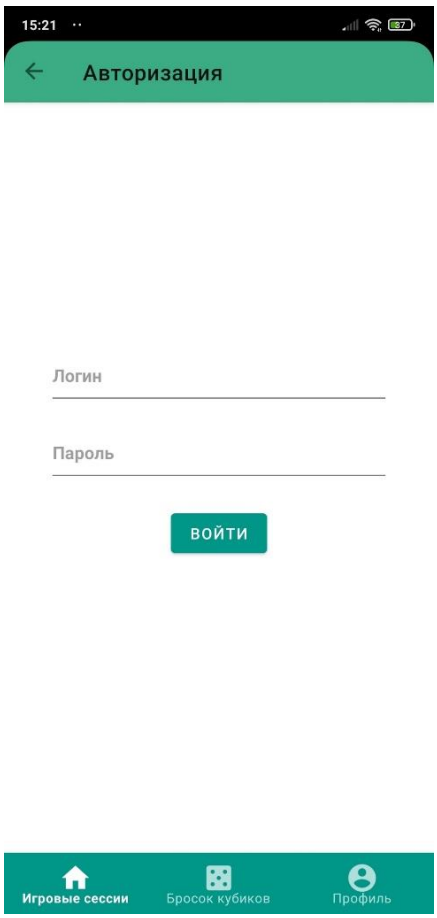

Рисунок 9 - Форма экрана авторизации

5.4.6 Форма экрана регистрации

На экране, изображённом на рисунке 10, отображены поля для ввода никнейма, логина, пароля. Ниже расположена кнопка зарегистрироваться, после регистрации перенаправляющая на экран профиля.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

Рисунок 10 - Форма экрана регистрации

5.4.7 Форма экрана профиля

На экране, изображённом на рисунке 11, указана информация о пользователе: никнейм, значение репутации, возраст, любимые игры, ссылки на профили в социальной сети «VK» и мессенджере «Telegram».

На экране расположены кнопки редактирования профиля, изменения репутации, открытия списка игровых сессий, блокирования и разблокирования пользователя.

Кнопка изменения профиля перенаправляет на экран редактирования.

Кнопки изменения репутации (увеличения/уменьшения) меняют величину репутации у адресуемого пользователя.

Кнопка открытия списка игровых сессий перенаправляет на станицу списка игровых сессий конкретного пользователя, на которые он подписан.

Кнопки заблокировать/разблокировать доступны только администратору и соответственно блокируют/разблокируют пользователя с соответственным изменением статуса пользователя и отображением статуса под никнеймом.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

Также приведён экран обычного пользователя на рисунке 12.

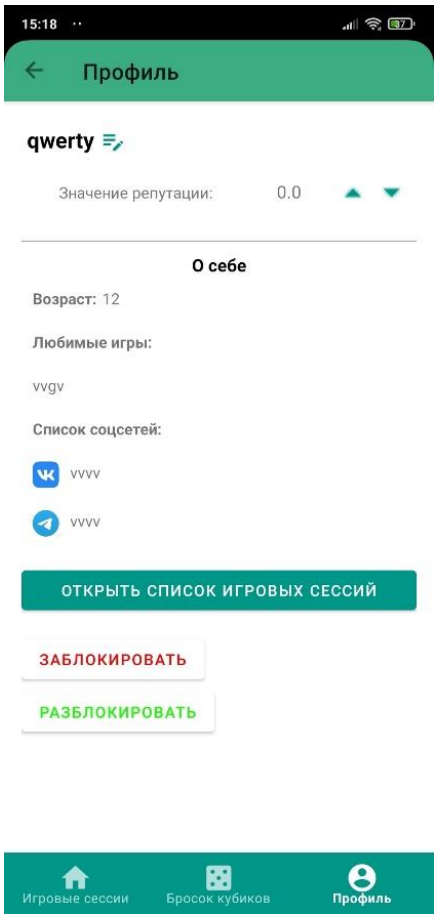

Рисунок 11 - Форма экрана профиля администратора

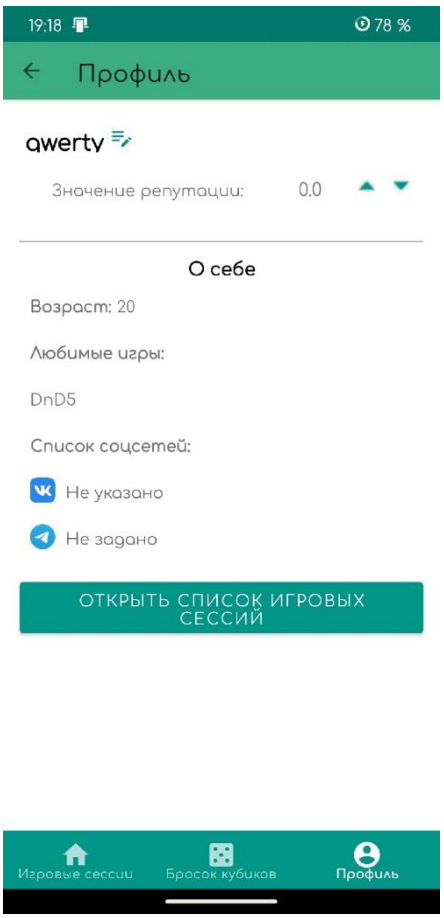

Рисунок 12 - Форма экрана профиля обычного пользователя

5.4.8 Форма экрана редактирования профиля

На экране, изображённом на рисунке 13, расположены формы для ввода данных о пользователе: поля для ввода возраста, любимых игр, ссылок на соцсеть «VK» и мессенджер «Telegram».

Ниже расположена кнопка изменения, которая сохраняет изменения и перенаправляет обратно на экран профиля.

Также ниже находится кнопка выйти, которая позволяет пользователю выйти из своего аккаунта, перенаправляя его на главный экран и изменяя его статус на неавторизованного.

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

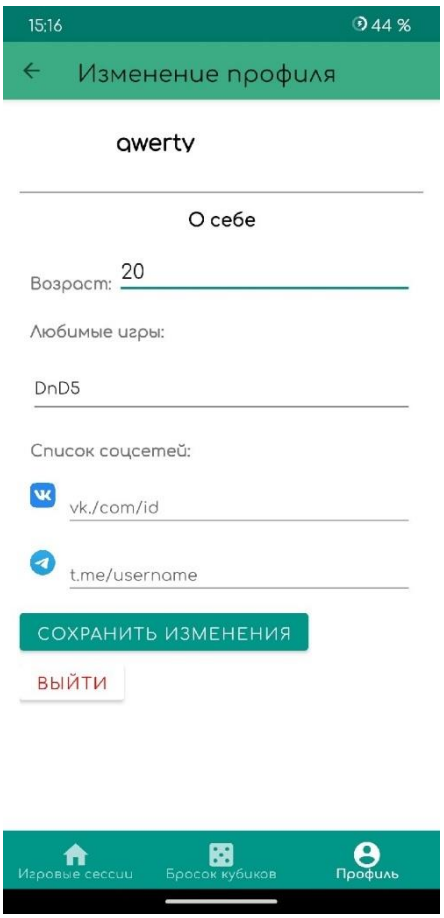

Рисунок 13 - Форма экрана редактирования профиля

5.4.9 Форма экрана броска кубика

На экране, изображённом на рисунке 14, расположены поля для ввода данных о необходимом количестве кубиков для броска, количестве граней кубиков и о модификаторе (числе, которое добавляется к значению).

Снизу расположена панель с кнопками, при нажатии на соответствующую кнопку пользователь переходит на страницы: список игровых сессий, бросок кубиков, профиль.

Рисунок 14 - Форма экрана броска кубика

5.5 Серверная часть

В качестве серверной части приложения используются фреймворки Django и Django REST Framework (далее – DRF). Использование Django в сочетании с DRF для создания API для мобильного приложения на Android предлагает ряд преимуществ. DRF предоставляет удобные инструменты для создания RESTful API, обеспечивая автоматическую сериализацию и десериализацию данных, обработку запросов и другие функции, необходимые для разработки API. Django и DRF также обеспечивают гибкость и масштабируемость, позволяя легко добавлять новые конечные точки API и настраивать функциональность. Безопасность обеспечивается встроенными механизмами Django и механизмами аутентификации и авторизации DRF.

Архитектура данного решения основана на шаблоне проектирования MSV (Model-Serializer-View) в Django Rest Framework. Модели описывают

структуру данных и взаимодействие с базой данных. Сериализаторы используются для преобразования моделей Django в JSON или другие форматы данных, которые могут быть переданы по сети. Они также обрабатывают входящие данные и преобразуют их в формат, понятный Django.

Также применяется программная платформа Docker для быстрой разработки, тестирования и развертывания приложений.

Вся логика разбита на отдельные пакеты, выполняющие различные функции и представляющие собой отдельные приложения MVC:

- users – приложение, отвечающее за функциональность пользователей;
- sessions – приложение, отвечающее за функциональность игровых сессий.

5.5.1 Приложение users

В данном приложении имеется следующая структура:

- models.py: в этом файле определяются классы модели профиля пользователя приложения и модели, отвечающей за сопоставление пользователей группам;
- serializers.py: данный файл предоставляет классы, на основе которых происходит сериализация/десериализация моделей в JSON формат;
- views.py: в этом файле имеются методы, привязанные к различным функциям API;
- apps.py: данный файл хранит в себе конфигурацию для моделей текущего модуля.

В данном модуле имеются модели User и UserData, хранящие данные о пользователе. User имеет следующие поля:

— uid;

— login: логин;

— nickname: никнейм;

— user_data: информация о пользователе, основанная на модели UserData.

UserData имеет следующие поля:

— age: возраст;

— games: любимые игры;

— vk: ссылка на страницу социальной сети «ВКонтакте»;

— tg: ссылка на профиль в мессенджере «Telegram»;

— played_sessions: массив идентификаторов игровых сессий, в которых пользователь принял или принимает участие;

— is_admin: статус администратора;

— is_banned: статус блокировки.

Функции-представления, помимо Create, Read и Update для модели User, в модуле добавляют следующие возможности:

— выдача или снятие блокировки пользователя (функции ban и unban, POST);

— увеличение или снижение репутации одним пользователем другого (функции plus_reputation и minus_reputation, POST);

— авторизация в приложении с регенерацией idToken и выдачей nickname (функция authorize, POST).

Данное приложение было реализовано с помощью модулей Firebase Auth, использованного для регистрации и авторизации пользователей, и Firebase Realtime Database, использованного для сохранения пользовательских данных.

5.5.2 Приложение sessions

В данном приложении имеется следующая структура:

— models.py: в этом файле определяются классы модели профиля пользователя приложения и модели, отвечающей за сопоставление пользователей группам;

— serializers.py: данный файл предоставляет классы, на основе которых происходит сериализация/десериализация моделей в JSON формат;

— views.py: в этом файле имеются методы, привязанные к различным функциям API;

— apps.py: данный файл хранит в себе конфигурацию для моделей текущего модуля.

В данном модуле имеется модель Session, хранящая данные о игровой сессии. Она имеет следующие поля:

— session_id: идентификатор;

— city_address: адрес проведения;

— name: название;

— owner: никнейм пользователя-владельца;

— players: массив никнеймов пользователей, принимающих участие в игровой сессии;

— players_max: максимальное количество игроков в игровой сессии.

Данное приложение было реализовано с помощью модулей Firebase Auth, использованного для аутентификации и разграничения доступа между пользователями, и Firebase Realtime Database, использованного для сохранения данных о игровых сессиях.

5.6 Развёртывание сервера

Также была применена программная платформа Docker для быстрой разработки, тестирования и развертывания приложений[15]. Серверная часть приложения была развернута на внешнем сервере с использованием Docker и DockerHub. Для этого были выполнены следующие шаги[4]:

— Создание Docker-контейнера: был создан Docker-контейнер, в котором размещена серверная часть приложения. Для этого был создан Dockerfile, в котором были описаны инструкции по настройке окружения и установке зависимостей. Dockerfile содержал указание базового образа, установку Python, установку необходимых пакетов и копирование приложения в контейнер.

— Сборка Docker-образа: после создания Dockerfile была выполнена команда для сборки Docker-образа, который содержит серверную часть приложения. Команда для сборки выглядела примерно так: "docker build -t <image_name>.", где <image_name> - имя выбранного образа.

— Загрузка Docker-образа на DockerHub: после сборки Docker-образа он был загружен на DockerHub, который является репозиторием Docker-образов. Для этого был создан аккаунт на DockerHub, и были выполнены команды для входа в систему (docker login) и загрузки образа (docker push <image_name>).

— Развёртывание на внешнем сервере: на внешнем сервере, где планировалось развернуть серверную часть приложения, был установлен Docker, и была выполнена команда для загрузки Docker-образа с DockerHub (docker-compose pull). Затем был запущен контейнер, используя загруженный образ (docker-compose up -d).

Таким образом, с использованием Docker и DockerHub была упакована и развернута серверная часть приложения на внешнем сервере с минимальными усилиями, обеспечивая масштабируемость и удобство обновлений.

Заключение

В результате выполнения данной курсовой работы было разработано мобильное приложение, предназначенное для поиска игроков в настольные игры. Приложение использует Firebase Realtime Database и Firebase Auth для хранения и аутентификации данных пользователей.

В процессе разработки были успешно достигнуты все поставленные цели и решены задачи. Были изучены возможности Firebase REST API и его модулей, а также особенности их применения в контексте мобильного приложения для поиска игроков. Firebase Realtime Database позволил реализовать эффективное хранение и синхронизацию данных между различными устройствами, а Firebase Auth обеспечил безопасную аутентификацию пользователей.

Выполнение данной курсовой работы позволило приобрести ценные знания и навыки в области разработки мобильных приложений и работы с Firebase REST API. Изучение Firebase Realtime Database и Firebase Auth позволило понять основы удаленного хранения данных и обеспечения безопасности в мобильных приложениях.

Список использованных источников

1. Firebase Realtime Database [электронный ресурс] – Режим доступа: https://firebase.google.com/docs/database – Заглавие с экрана. – (дата обращения: 07.06.2023).
2. Firebase Auth [электронный ресурс] – Режим доступа: https://firebase.google.com/docs/auth – Заглавие с экрана. – (дата обращения: 07.06.2023).
3. Django [электронный ресурс] – Режим доступа: https://www.djangoproject.com/ – Заглавие с экрана. – (дата обращения: 07.06.2023).
4. Docker [электронный ресурс] – Режим доступа: https://www.docker.com/ – Заглавие с экрана. – (дата обращения: 07.06.2023).
5. Kotlin [электронный ресурс] – Режим доступа: https://kotlinlang.org/ – Заглавие с экрана. – (дата обращения: 07.06.2023).
6. Android SDK [электронный ресурс] – Режим доступа: https://developer.android.com/studio – Заглавие с экрана. – (дата обращения: 07.06.2023).
7. Spring Android [электронный ресурс] – Режим доступа: https://spring.io/guides/gs/consuming-rest-android/ – Заглавие с экрана. – (дата обращения: 07.06.2023).
8. SQLite [электронный ресурс] – Режим доступа: https://www.sqlite.org/ – Заглавие с экрана. – (дата обращения: 07.06.2023).
9. Исследование рынка настольных игр в России [электронный ресурс] – Режим доступа: https://elib.pnzgu.ru/files/eb/doc/FkGzMNuE7b6.pdf – Заглавие с экрана. – (дата обращения: 07.06.2023).
10. Timepad [электронный ресурс] – Режим доступа: https://timepad.ru – Заглавие с экрана. – (дата обращения: 07.06.2023).

11. Plink [электронный ресурс] – Режим доступа: https://plink.gg/ – Заглавие с экрана. – (дата обращения: 07.06.2023).
12. Meetup [электронный ресурс] – Режим доступа: https://www.meetup.com – Заглавие с экрана. – (дата обращения: 07.06.2023).
13. «Кто куда» [электронный ресурс] – Режим доступа: https://ktokyda.ru/ – Заглавие с экрана. – (дата обращения: 07.06.2023).
14. Кант М., Сумароков М. Разработка Android-приложений на языке Kotlin. - СПб.: БХВ-Петербург, 2022.
15. Маклафлин Б., Бэннетт Дж. Docker в действии. - М.: ДМК Пресс, 2020.