Титульный лист

Задание на курсовую работу

ОГЛАВЛЕНИЕ

Введение

Компьютерные игры давно стали неотъемлемой частью современной культуры, привлекая миллионы людей по всему миру, включая Россию. Это уникальное явление, соединяющее людей разных возрастов, профессий и интересов. Тема для курсовой работы “Игровой календарь” была выбрана не случайно, несмотря на широкую аудиторию и высокий спрос, я не смог найти решения для отслеживания дат выхода новых игр и организации личных игровых списков.

Отсутствие такого инструмента создает определенные неудобства для геймеров, которые стремятся быть в курсе последних новинок и управлять своими игровыми достижениями. В свете этого, целью данной курсовой работы является разработка мобильного приложения, которое не только поможет игрокам следить за актуальными и предстоящими релизами, но и предоставит возможность вести учет своих игровых достижений.

Создание “Игрового календаря” предоставит людям инструмент, который не только удовлетворит их запросы, но и повысит их вовлеченность и удовольствие, получаемое от игр.

Цель курсовой работы

Разработать мобильное приложение календарь – данное приложение должно будет помочь отслеживать свои списки, а также быть в курсе последних и ближайших релизов. Приложение должно быть реализовано на языке высокого уровня Java.

Задачи курсовой работы

1. проанализировать предметную область мобильного приложения;
2. сделать обзор существующих аналогов разрабатываемого приложения;
3. сформировать техническое задание на разработку программы в соответствии с ГОСТ 19.201-78;
4. описать архитектуру программной системы, привести структурную и функциональную диаграммы, схему базы данных;

5. спроектировать интерфейс мобильного приложения;

6. реализовать код программы на языке высокого уровня Java, протестировать его и отладить;

7. реализовать контрольный пример работы программы, начиная с открытия, показать все этапы работы вашего приложения.

1. Анализ предметной области

Календарь специализируется не только в отслеживании анонсированных игр, вызывающих интерес в игровом сообществе, но и предоставляет пользователям возможность организовывать свои личные списки ожидаемых релизов. Это позволяет геймерам оставаться в курсе последних новостей и быть уверенными, что они не пропустят выход игр, которые они с нетерпением ждут.

Для эффективного функционирования календаря требуется база данных, которая будет способная хранить информацию. Данная база данных должна включать в себя:

8. Детальные сведения об игре: название, жанр, разработчик, издатель и описание;
9. Текущее состояние игры: в разработке или уже можно играть;
10. Дата выхода игры: точная дата или предполагаемый период релиза.
11. Информация о пользователе: всевозможные данные пользователя (его статус, никнейм и фотография профиля);
12. Состояние игры в списках у пользователя: отметки ожидания, прохождения, избранное и т.д.

База данных строиться с учетом следующих особенностей:

1. Состояние игры должно меняться в реальном времени;
2. Каждый пользователь может сохранять в свои списки по несколько игр;
3. Каждая игра включает в себя связанные с этой релизы.

Администраторы базы данных играют ключевую роль в поддержании актуальности и точности хранимой информации. Их задачи включают в себя:

1. Добавление новых игр в календарь, обеспечивая, что данные о новых анонсах быстро становятся доступны пользователям;

2. Обновление информации об играх: изменение дат, добавление новостей о разработке, редактирование описаний и т.д.;

3. Мониторинг и управление списками пользователей: просмотр состояний игр в списках, отслеживание популярности игр среди пользователей.

2. Анализ существующих аналогов

Для анализа существующих аналогов я выбрал два широко распространённых календаря: международный Google Calendar и разработанный российскими специалистами Яндекс.Календарь. Оба этих календаря обладают своими положительными и отрицательными сторонами. Рассмотрим каждый из них по подробней.

Яндекс Календарь — это бесплатный персональный информационный менеджер от компании Яндекс. Он позволяет планировать дела с различными приоритетами, устанавливать напоминания, создавать несколько календарей на разные темы, а также использовать его совместно с другими пользователями. Яндекс Календарь интегрирован с другими сервисами Яндекс.

Плюсы:

- Интеграция с экосистемой Яндекс обеспечивает синхронизацию с картами, погодой и афишей, что делает планирование более удобным;
- Возможность создания списков задач и настройки напоминаний.

Минусы:

- В сравнении с международными аналогами функциональность может быть ограниченной;
- Не всегда приходят push уведомления на телефон.

На рисунке 1 предоставлен внешний вид веб версии Яндекс календаря.

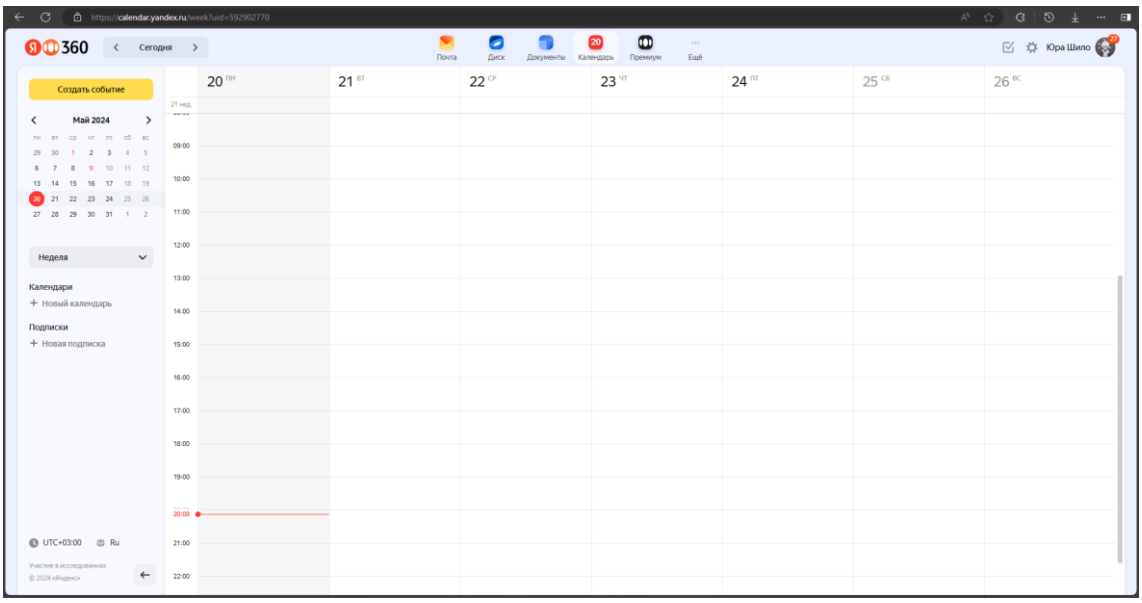

Рисунок 1. – Интерфейс веб версии Яндекс календаря

Google Календарь, разработанный компанией Google, предоставляет возможность планировать встречи, события и дела. Google Календарь позволяет задавать время встречи, создавать повторяющиеся мероприятия, устанавливать напоминания и приглашать других участников с уведомлениями по электронной почте. Его можно использовать для совместного планирования встреч и событий, а также синхронизировать с устройствами.

Плюсы:

- Возможность одновременного управления несколькими календарями, что идеально подходит для разделения личных и рабочих задач;
- Интеграция с Gmail и Google Tasks позволяет легко превращать письма в задачи и отслеживать их выполнение;
- Автоматическое добавление событий из электронной почты, таких как билеты на концерты или бронирование отелей, облегчает планирование.

Минусы:

- Некоторые продвинутые функции доступны только в рамках платных подписок Google Workspace, что может быть барьером для индивидуальных пользователей;

- Интеграция с корпоративными системами может быть затруднена без специализированных настроек.

На рисунке 2 предоставлен внешний вид веб версии Google календаря.

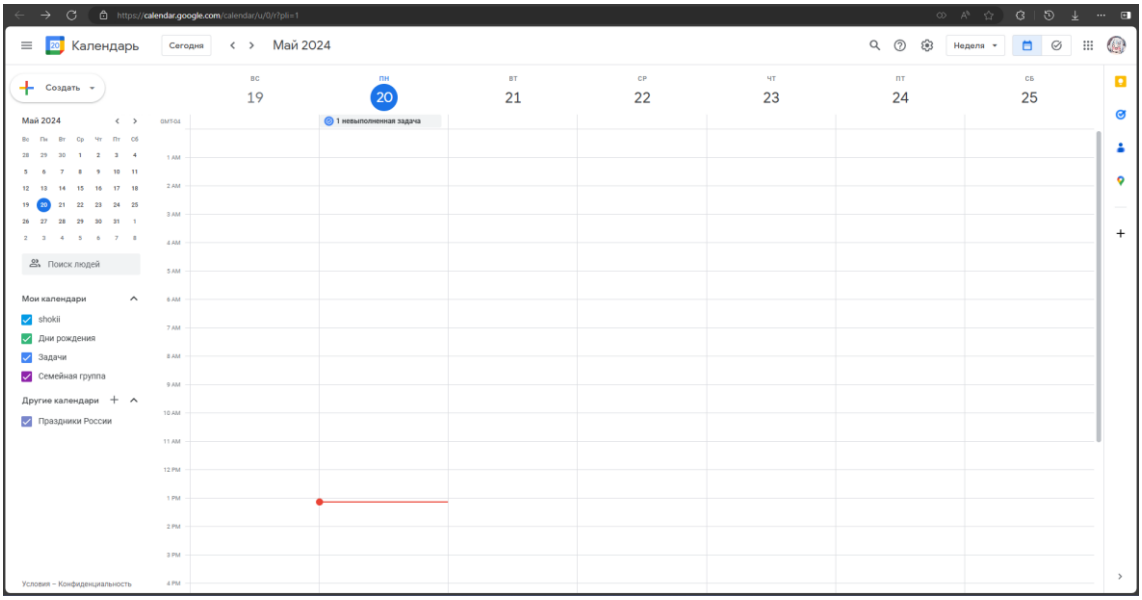

Рисунок 2. – Интерфейс веб версии Google календаря

Проведя анализ этих двух продуктов, я выяснил, что два этих календаря, не содержат функционал, который требуется в моем приложении. Но несмотря на это я приступил к написанию технического задания для моего календаря. Моя цель создать приложение, которое бы объединило лучшие качества обеих систем, стараясь исключать их недостатки.

3. Техническое задание

Назначение разработки

Целью разработки моего мобильного приложения состоит в том, чтобы обеспечить конечных пользователей сервисом, который помог бы им сохранять интересующие игры не на бумажном носителе, а в удобном мобильном приложении. Так же помочь пользователю вести список игр, в которые тот играет или еще собирается играть.

Требования к программе или программному изделию

Функциональные требования

- **Аутентификация:** Доступ к функционалу приложения только для зарегистрированных пользователей.
- **Просмотр игр**: Возможность просматривать список уже выпущенных и анонсированных игр.
- **Информация об играх**: Отображение детальной информации о каждой игре, включая название, логотип, дату выхода, издателя и разработчика.
- **Управление списками**: Возможность добавления игр в персональные списки пользователя.
- **Календарь релизов**: Отслеживание дат выхода игр с помощью календаря в приложении.

Требования к надежности

- **Роли пользователей**: Два типа пользователей - "Администратор" и "Обычный пользователь", с различными уровнями доступа и возможностями.

Условия эксплуатации

- **Интернет-соединение**: Необходимо устойчивое соединение с интернетом для работы с базой данных.

Требования к техническим средствам

- **Сервер**: Удалённое устройство для хранения данных и обработки запросов.
- **Клиент**: Устройство пользователя с операционной системой Android.

Требования к совместимости

- **Серверная часть**: Совместимость с Firebase для управления данными и аутентификации.
- **Клиентская часть**: Реализация на языке Java, совместимость с Android SDK и поддержка последних версий Android.

4. Архитектура

Для создания эффективной модели базы данных, которая будет служить основой для календаря игровых релизов, мы разработаем диаграмму IDEF0. Эта диаграмма поможет нам визуализировать и структурировать процессы взаимодействия данных и пользователей в системе.

На рисунке 3 представлен самый верхний уровень нашей схемы. Он иллюстрирует ключевые потоки информации и взаимодействия между различными компонентами системы:

Входы:

1. Информация об игре: включает в себя все данные об играх, такие как название, разработчик, издатель, описание и дата релиза;
2. Данные пользователя: личная информация, никнейм, фото профиля, списки ожидания и пройденных игр;
3. Текущее время сервера: используется для тайминга уведомлений и обновлений статусов игр.

Механизмы:

1. Участники: активные пользователи системы, которые ведут свои списки и взаимодействуют с календарем;
2. Сервер: обеспечивает обработку и хранение данных;
3. Модератор: отвечает за актуализацию данных.

Управляющие:

1. Время: определяет текущую дату и время;
2. Правила сравнения: используются для сравнения;
3. Правила регистрации игры и пользователя: определяют процесс добавления новых пользователей и игр в систему.

Выходы:

1. Информация о пользователе: обновленные данные о пользователях;

2. Уведомления о выходе игры: оповещения, отправляемые на устройства пользователей о скором релизе ожидаемых игр;

3. Сообщения об ошибках: система уведомит пользователя в случае обнаружения некорректно введенных данных или других проблем.

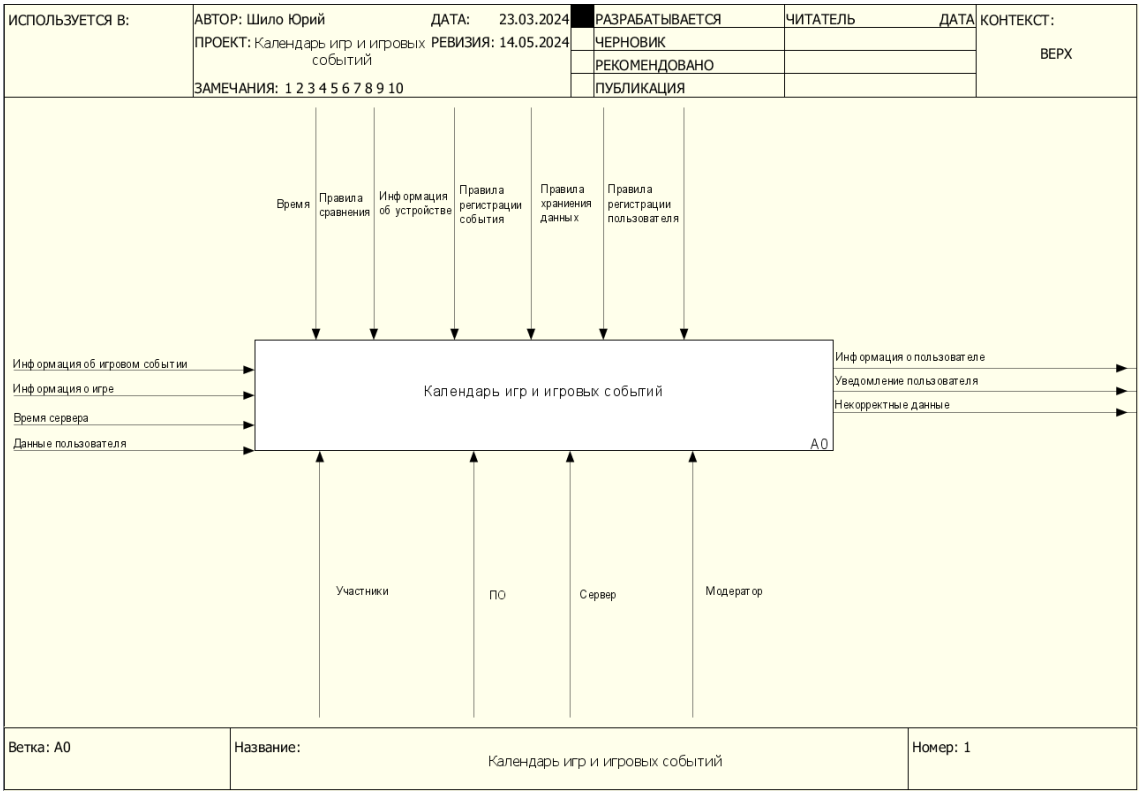

Рисунок 3. – Контекстная диаграмма «Календарь игровых событий» в методологии IDEF0.

Для более глубокого понимания нашей системы рассмотрим её на более низком уровне. На этом уровне будет демонстрироваться процесс регистрации пользователя и игры, а также механизм отправки уведомлений. Взаимодействие каждого блока предоставлена на рисунке 4.

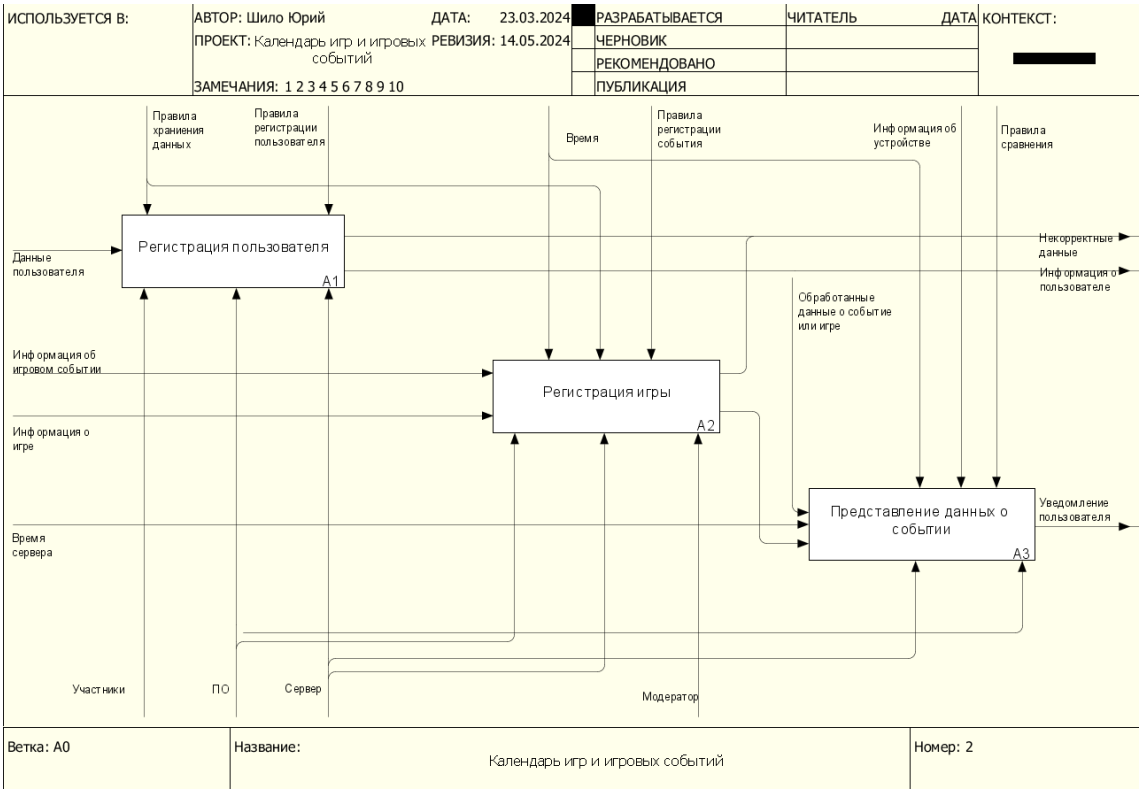

Рисунок 4. – Декомпозиция IDEF0 диаграммы.

Для того чтобы иметь более полное представление о реализуемой системе, мы применяем DFD (Data Flow Diagram) нотацию для нашего приложения. Это позволяет нам визуализировать и анализировать все потоки данных, проходящие через различные компоненты системы.

На рисунке 5 представлен самый первый слой реализуемой системы.

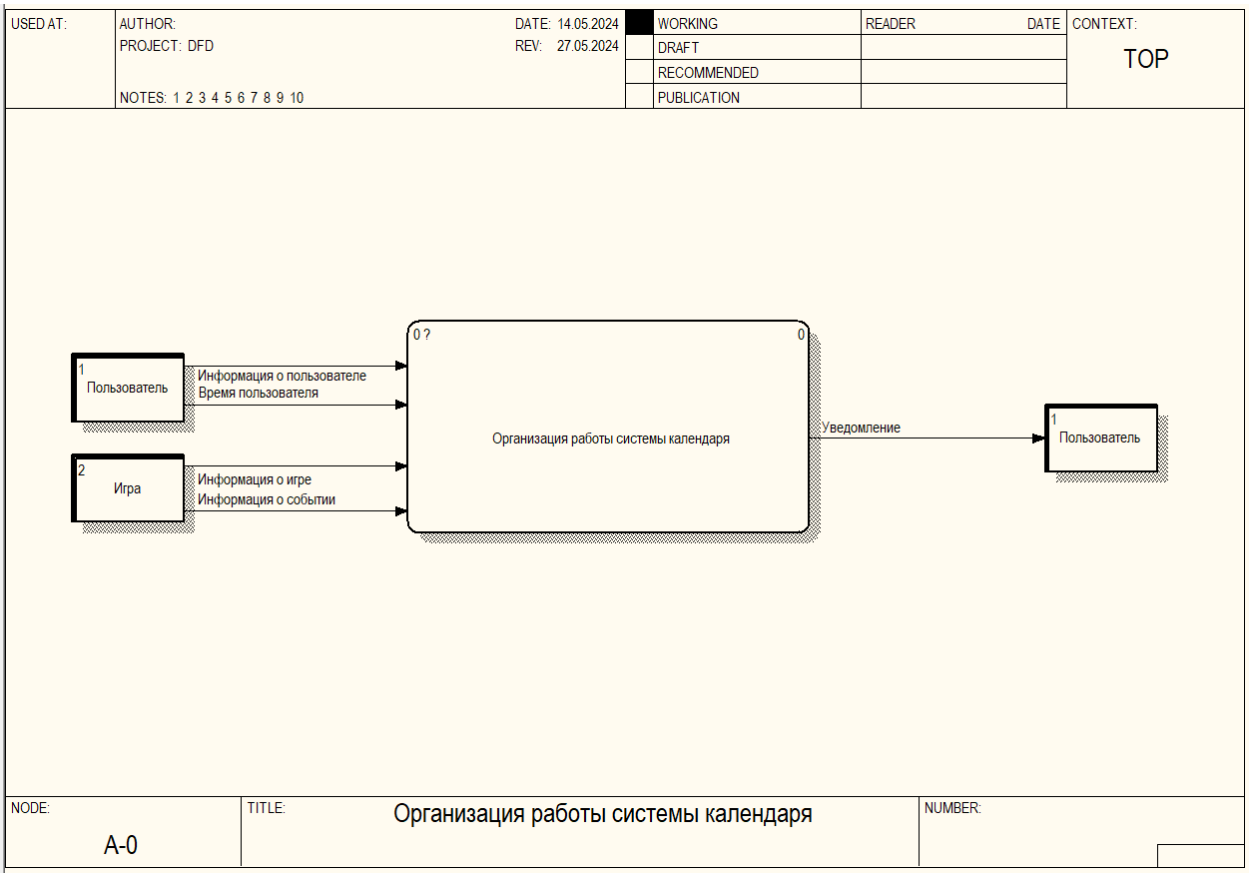

Рисунок 5. – Контекстная диаграмма «Организация работы системы календаря» в нотации DFD.

На рисунке 6, предоставлен более детальный уровень нашей системы, где каждый компонент и связь между ними рассматривается более тщательно. На этом уровне DFD мы можем увидеть, как взаимодействуют отдельные модули, какие конкретные данные они обменивают и какие преобразования с ними происходят.

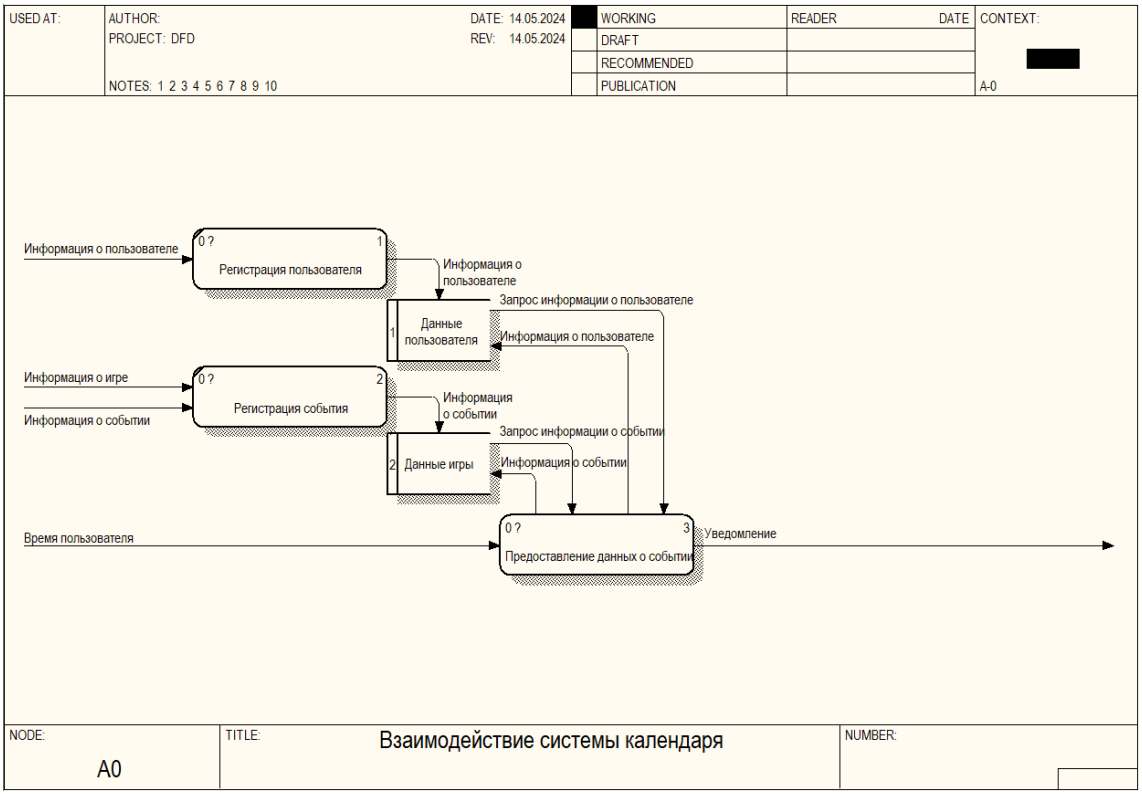

Рисунок 6. – Декомпозиция контекстной диаграммы «Организация работы системы календаря» в методологии DFD.

5. Описание и обоснование выбора ПО

В качестве платформы для разработки был выбран Android. Это решение было обусловлено широкой популярностью операционной системы, её открытостью и гибкостью, а также большим сообществом разработчиков. Android предоставляет разработчикам удобные инструменты для создания приложений, такие как Android Studio, различные библиотеки и API. Это позволяет создавать функциональные и адаптивные приложения, способные удовлетворить потребности широкого круга пользователей.

Рисунок 7. – Логотип операционной системы Android

Выбор языка программирования Java для разработки приложения на Android является классическим решением, так как Java была основным языком для этой платформы на протяжении многих лет. Java обладает мощными инструментами для управления памятью, исключениями и многопоточностью, что делает её идеальным выбором для создания надёжных и эффективных приложений.

Рисунок 8. – Логотип языка программирования Java

Для написания кода нашего приложения мной было использовано программа Android Studio. Android Studio — это официальная интегрированная среда разработки (IDE), предназначенная для разработки

приложений Android. Это мощный инструмент, который предоставляет все необходимые функции для проектирования, написания, тестирования и отладки приложений. Он включает в себя редактор кода, который поддерживает языки программирования Kotlin, Java и C/C++, а также обширный набор инструментов для профилирования производительности и анализа использования ресурсов приложения.

В Android Studio также имеется встроенный эмулятор, который был использован для части тестов нашего приложения. Эмулятор поддерживает различные конфигурации устройств и версии Android, что позволило протестировать мое мобильное приложение на широком спектре устройств.

Функция установки приложения на телефон напрямую очень сильно упростила процесс тестирования и отладки, позволив быстро и легко загружать новые версии приложения на физическое устройство. На рисунке 9 показан приветственный экран Android Studio.

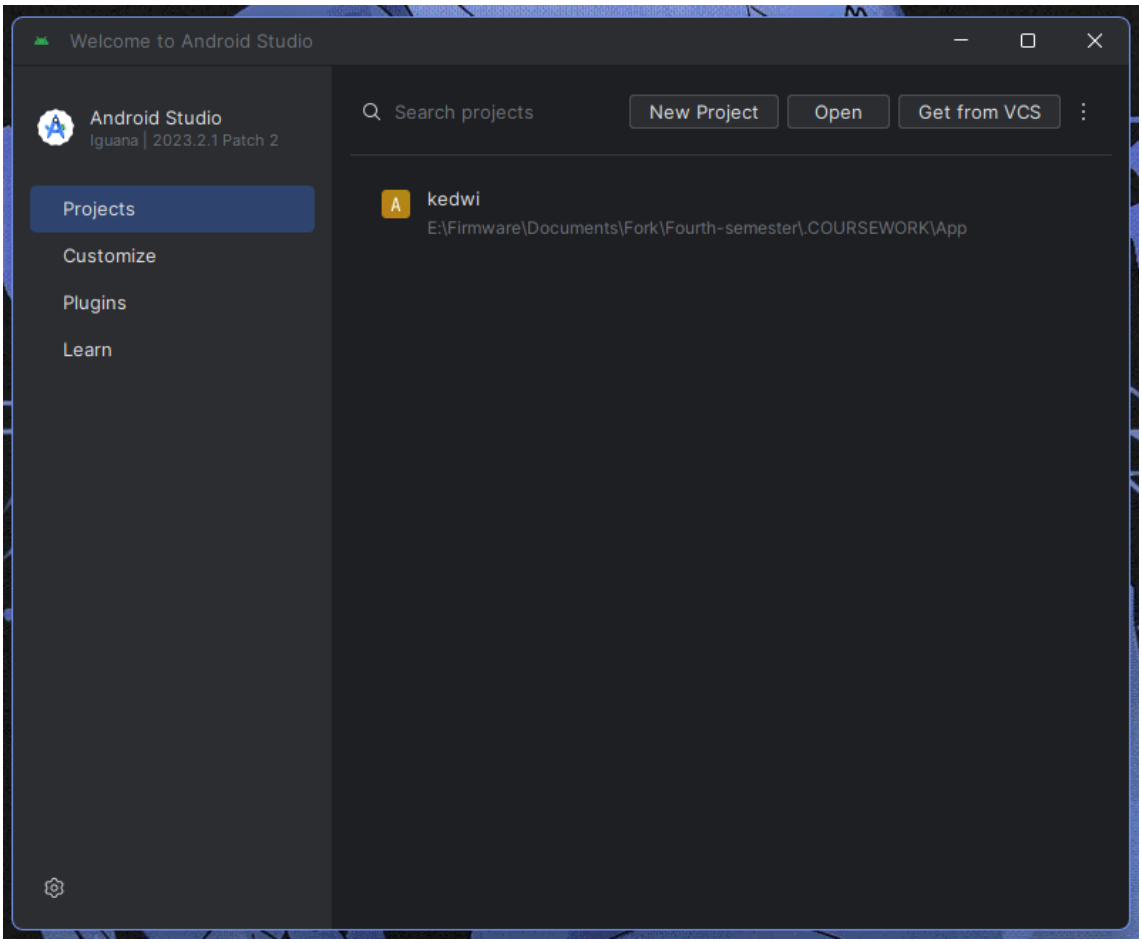

Рисунок 9. – Окно, запускаемое при входе в Android Studio.

Для создания уникальной иконки для нашего приложения, я выбрал инструмент графического дизайна – Adobe Illustrator. Это приложение от известной компании Adobe является стандартом в индустрии для создания векторной графики. С его помощью можно разрабатывать логотипы, иконки, эскизы, типографику и сложные иллюстрации.

Первое, что видит пользователь это иконка поэтому она должна быть запоминающейся. Именно поэтому я решил, что центральным элементом нашего логотипа станет кошка. Кошки пользуются большой любовью у людей по всему миру.

На рисунке 10, представлен итоговый вариант иконки.

Рисунок 10. – Логотип мобильного приложения.

Для создания красивого и функционального интерфейса нашего приложения, мы воспользовались программой Figma. Figma — это современный инструмент для совместного проектирования интерфейсов, который позволяет дизайнерам и разработчикам работать вместе в реальном времени. Это облачное приложение используется для создания, тестирования и развертывания дизайна интерфейса или продукта.

В моем проекте использовалась Figma для того, чтобы визуализировать идеи и создать дизайн, который будет выглядеть привлекательно. На рисунке 11 представлены некоторые из окон нашего приложения, дизайн которых был спроектирован в Figma.

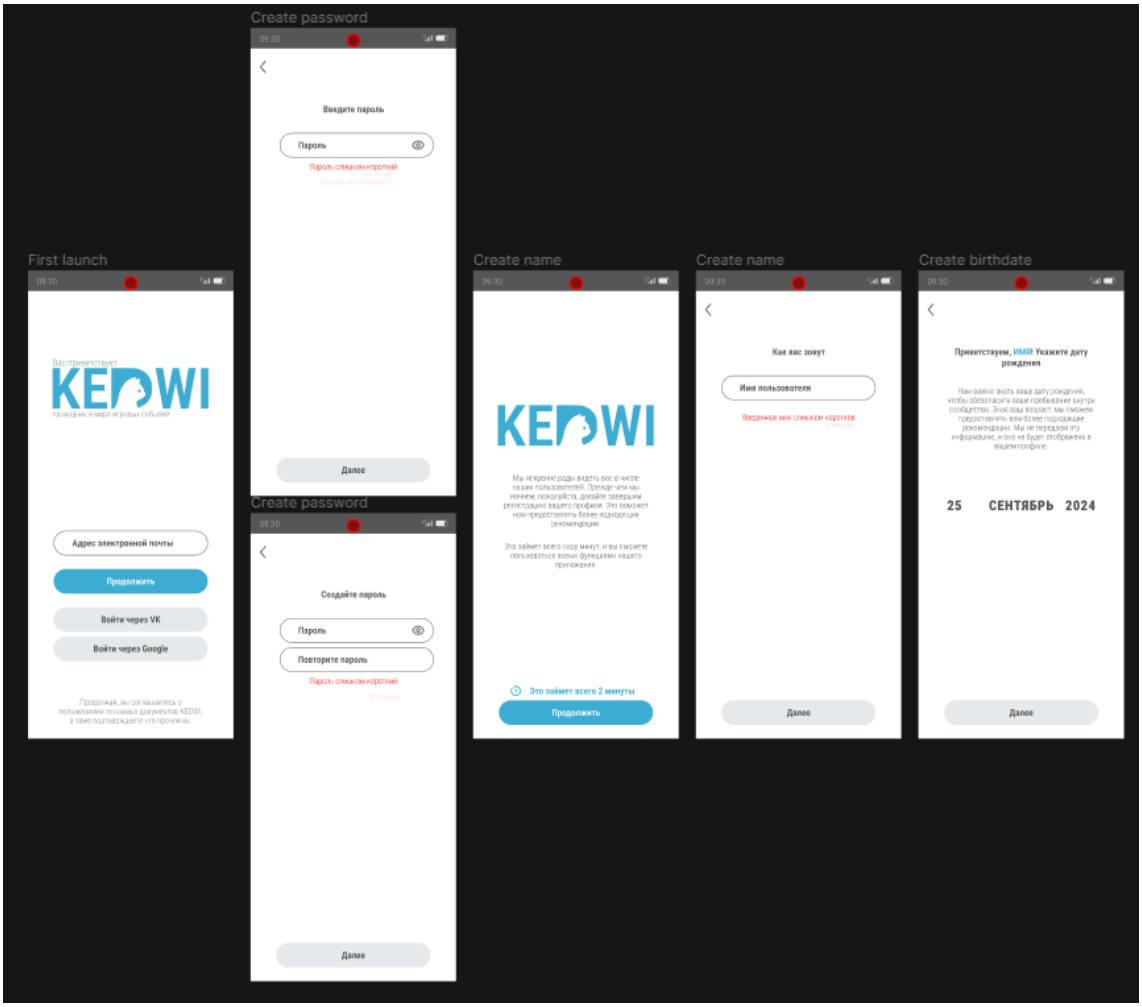

Рисунок 11. – Дизайн окон регистрации, построенный в Figma.

В качестве системы управления базами данных была использована Firebase. Firebase — это облачная база данных, которая позволяет хранить и синхронизировать данные в реальном времени между пользователями приложения. Это решение от компании Google, предоставляющее мощный набор инструментов для разработки кроссплатформенных приложений. Firebase не только упрощает процесс хранения данных, но и обеспечивает их безопасность, используя правила безопасности и валидацию данных на стороне сервера.

В нашем случае, Firebase используется для хранения всей необходимой информации о пользователях и играх. Для пользователей это такие данные, как никнейм, фотографию профиля, статус и списки, которые могут содержать избранные игры. Что касается информации об играх, то здесь хранятся название игры, дата выхода, разработчик, издатель, описание и рейтинг PEGI — система рейтинга, которая помогает понять, подходит ли игра для определенной возрастной категории.

На рисунке 12 показан интерфейс сайта с моей базой данных.

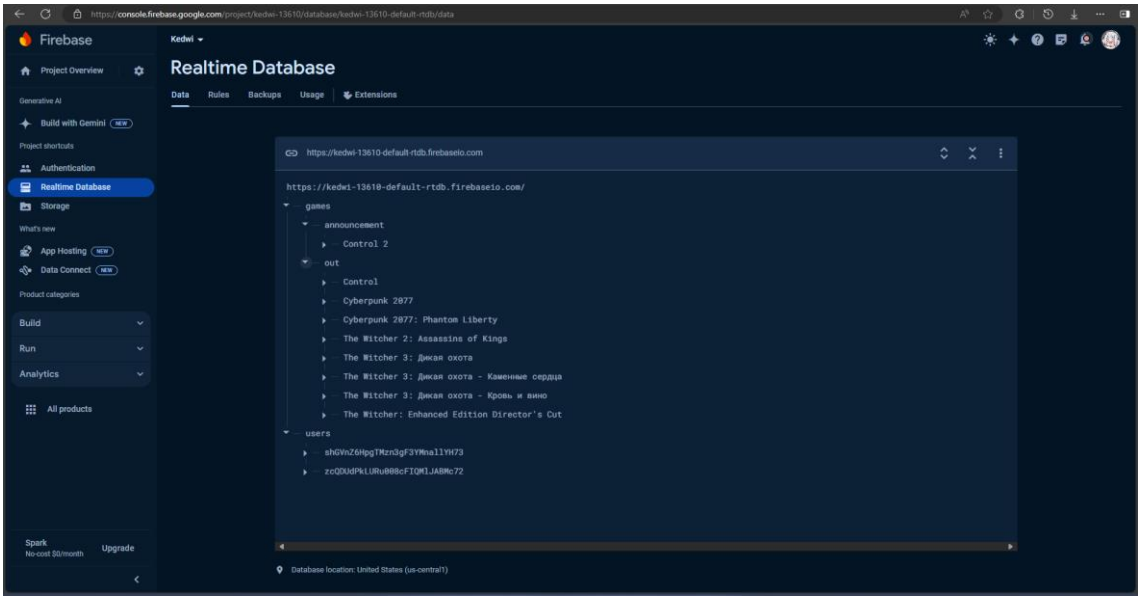

Рисунок 12. – Интерфейс веб сайта с моей базой данных.

6. Wireframe приложения

Перед тем как приступить к написанию разметки и кода нашего приложения я создал наброски в Figma. Wireframe — это схема с низким уровнем детализации, которая визуализирует структуру и содержание цифрового проекта. Wireframe показывает, как будут расположены все основные элементы продукта: навигация, карточки, текстовые блоки, иллюстрации и кнопки.

Первым шагом в процессе разработки моего мобильного приложения стало создание wireframe для процессов регистрации и авторизации пользователей. На рисунке 1 представлен первичный черновой макет, который демонстрирует расположение ключевых элементов интерфейса.

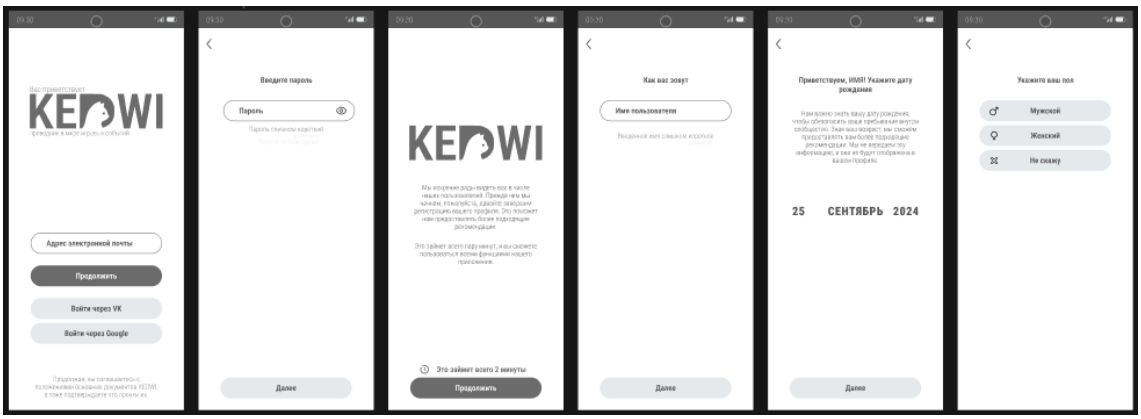

Рисунок 13. – Wireframe окно регистрации

После успешного создания wireframe для процессов регистрации и входа, я перешёл к разработке основных окон приложения. Основное окно было спроектировано, так чтобы пользователь мог просматривать списки недавно вышедших и скоро выходящих игр. Эта разметка послужила основой для дальнейшего дизайна пользовательских списков, где можно отслеживать интересующие игры.

Завершающим этапом стало создание wireframe для пользовательского профиля, где предусмотрены разделы для персональной информации, настроек и статистики. На Рисунке 2 представлены все перечисленные wireframe.

Рисунок 14. – Wireframe для пользовательских окон

7. Инструкция пользователя

После того как будет установлено и открыто приложение, пользователю будет предоставлено приветствует окно, в котором вам будет предложено поле для ввода электронной почты. После ввода электронной почты система проведет проверку, существует ли уже профиль с таким электронным адресом. Если профиль был найден, пользователь будет перенаправлены на страницу, где необходимо ввести пароль для входа в аккаунт.

Если же учетная запись с введенной электронной почтой не обнаружена, пользователю будет предложено зарегистрироваться, создав уникальный пароль. После этого, человека переносит на страницу, где я выражаю радость от того, что пользователь почти зарегистрировался в приложении, но для завершения профиля ему необходимо ввести дополнительные данные, такие как имя, дату рождения и пол.

После завершения регистрации, так же, как и после успешного входа в существующий аккаунт, пользователь будет перенаправлены на главный экран приложения. Где будет происходить последующие действия.

На главном экране нашего приложения, в нижней части расположена панель навигации. По нажатию на иконки, которой пользователь сможет легко переключаться между всеми существующими разделами.

На домашней странице расположены списки игр, как ожидаемых новинок, так и уже выпущенных игр. При нажатии на их логотипы пользователя перебрасывает на страницу, где описывается игра ее разработчик, издатель, дата выхода, описание и многое другое

На странице календаря пользователь сможет найти отслеживаемые игры, которые скоро выйдут и были добавлены в список отслеживаемых.

В разделе пользовательских списков пользователь сможет взаимодействовать с играми, которые были отмечены им как интересные, в процессе игры или уже пройденные.

В профиле отображается вся важная информация о пользователе: фото, никнейм, статус в сообществе, а также количество игр в каждом из пользовательских списков. Здесь же расположена кнопка для редактирования профиля, чтобы была возможность обновить данные в любое время.

8. Тестирование на физическом устройстве

Чтобы обеспечить максимальную надежность и эффективность, важно провести тщательное тестирование программного обеспечения. Поэтому я решил выполнить запуск приложения на реальном устройстве, что позволит не только проверить корректность работы программы в условиях, максимально приближенных к реальным, но и оценить её производительность и удобство использования конечными пользователями.

Для этих целей был выбран смартфон Infinix Note 30, работающий под управлением операционной системы Android версии 14.

При входе в приложении нас встречает окно ввода электронной почты. Пользователь вводит электронную почту и если пользователя с такой почтой нет, то нам предложат создать пароль. После ввода пароля первый этап регистрации пройден (рисунок 15).

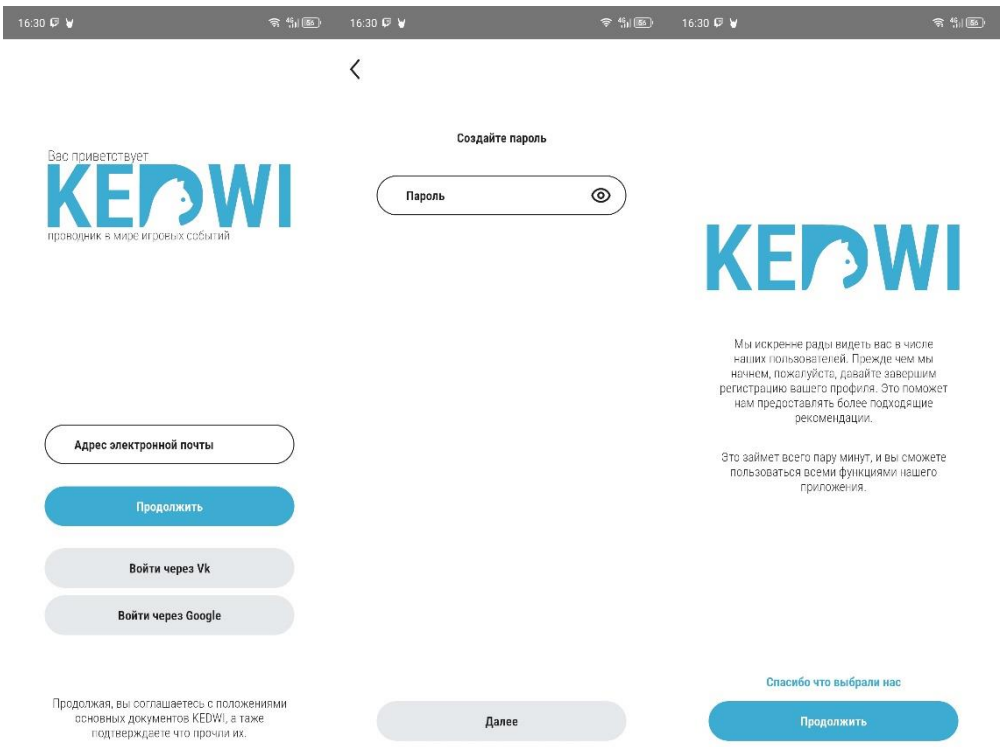

Рисунок 15. – Экраны регистрации пользователя

Что бы пользователь окончательно зарегистрировался в нашем приложении ему потребуется ввести никнейм, дату рождения и выбрать пол. Для выбора даты рождения у нас используется Datepicker. Все описанные окна показаны на рисунке 14.

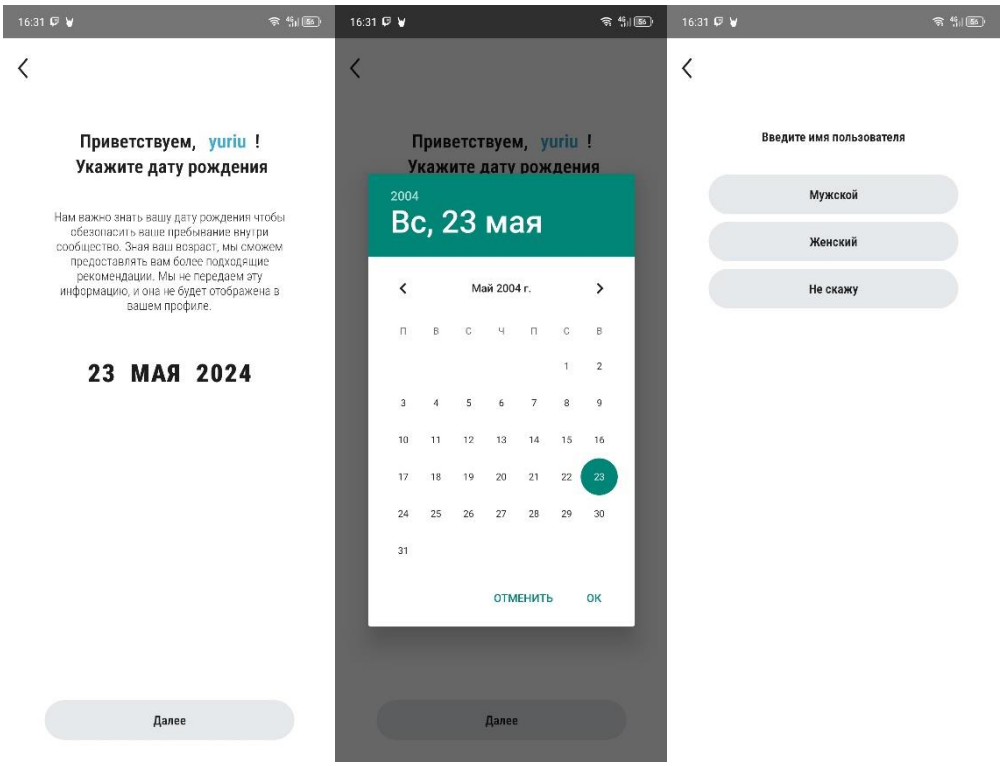

Рисунок 14. – Окна продолжения регистрации

Если пользователь добавляет игру из списка анонсированных в свой список, то мы сможем увидеть эту игру в нашем календаре на дате, когда игра выйдет. На рисунке 16 показано как это выглядит.

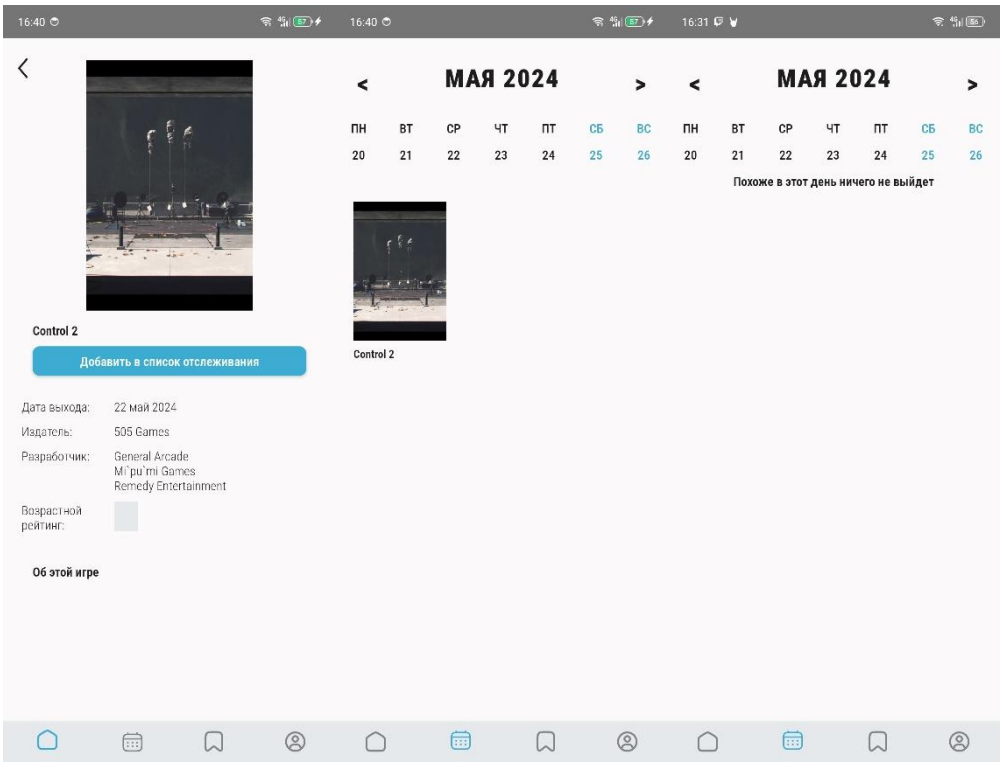

Рисунок 16. – Окна добавления игры из списка анонсированных

Если текущий пользователь администратор, то у него появляется возможность добавить игру в систему. Окно для добавления игры показано на рисунке 17.

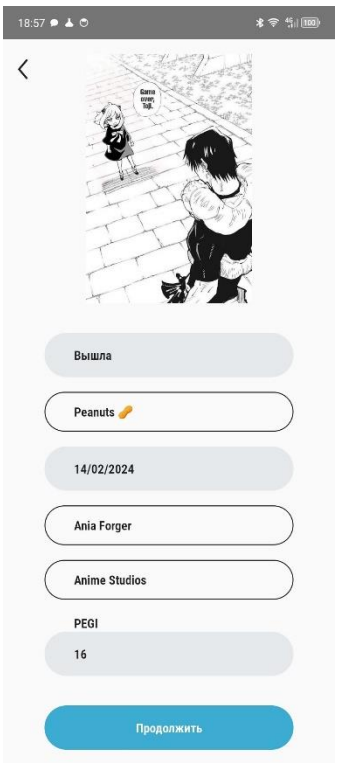

Рисунок 17. – Окно добавления игры

Все пользователи в приложении в настройках профиля смогут выйти из учетной записи, изменить фото профиля, изменить свой статус, изменить никнейм и пароль. На рисунке 18 показано как выглядят настройки и как выглядит изменение статуса. Для изменения статуса в моем случае используется AlertDialog.

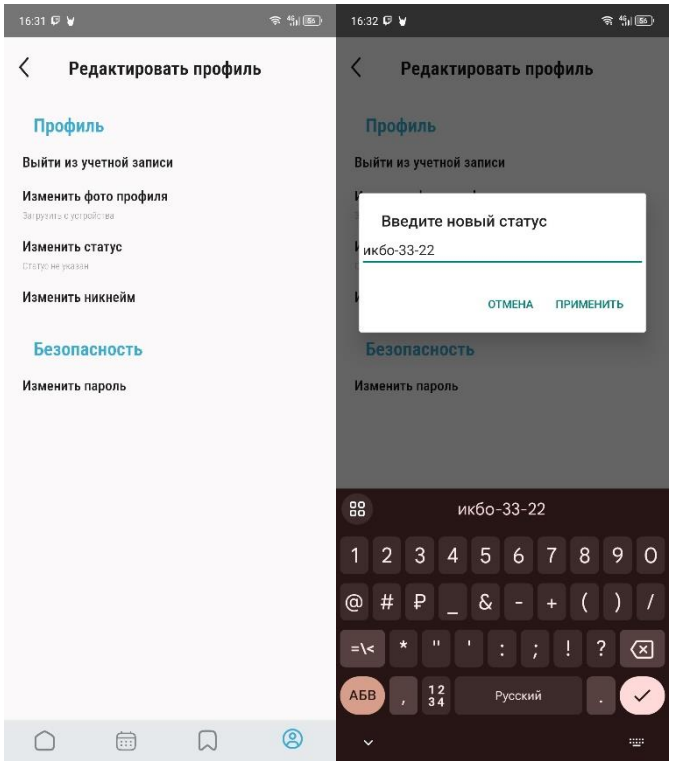

Рисунок 18. – Окно настройки и окно изменение статуса.

Для изменения фотографии нашего профиля был реализован кропер. Который позволяет пользователю обрезать фотографию прямо в приложении. На рисунке 19 показано как выглядит кропер и подтверждение его работы.

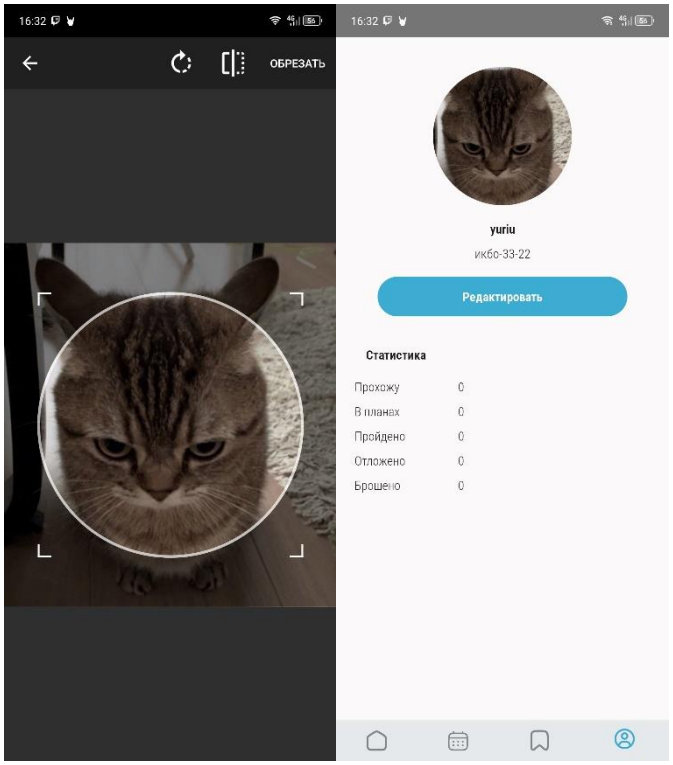

Рисунок 19. – Изменение пользовательского изображения

Заключение

В процессе выполнения данной курсовой работы было успешно разработано мобильное приложение “Календарь игровых новинок”. Этот процесс выключал в себя создание полностью рабочего приложения, связанного с внешней базой данных с применением современных технологий мобильной разработки. В результате был достигнут результат с интуитивно понятным и привлекательным интерфейсом.

Важную роль в успешной реализации проекта сыграл тщательный анализ предметной области. Этот анализ позволил выявить ключевые потребности пользователя и основные цели мобильного приложения, что в свою очередь способствовало эффективному проектированию интерфейса и реализации нужных функций. Каждая поставленная задача в рамках курсовой работы была успешно выполнена.

Ссылка на исходный код разработанного мобильного приложения на тему “Календарь игровых новинок”: https://github.com/shok1i/Fourth-semester/tree/main/.COURSEWORK

Список используемой литературы

1. StackOverflow / [Электронный ресурс] //: [сайт]. — URL: https://stackoverflow.com/ (дата обращения: 20.05.2024).
2. Android Tools / [Электронный ресурс] //: [сайт]. — URL: https://android-tools.ru/ (дата обращения: 20.05.2024).
3. Developer Android / [Электронный ресурс] //: [сайт]. — URL: https://developer.android.com/ (дата обращения: 20.05.2024).
4. Glide / [Электронный ресурс] //: [сайт]. — URL: https://bumptech.github.io/glide/ (дата обращения: 20.05.2024).
5. Geeksforgeeks / [Электронный ресурс] //: [сайт]. — URL: https://www.geeksforgeeks.org/ (дата обращения: 20.05.2024).
6. YouTube / [Электронный ресурс] //: [сайт]. — URL: https://www.youtube.com/ (дата обращения: 20.05.2024).
7. Firebase / [Электронный ресурс] //: [сайт]. — URL: https://firebase.google.com/docs?hl=en (дата обращения: 20.05.2024).
8. Github / [Электронный ресурс] //: URL: https://github.com/ (дата обращения 20.05.2024)
9. ГОСТ 7.32—2017 Система стандартов по информации, библиотечному и издательскому делу ОТЧЕТ О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ Структура и правила оформления — М.: Стандартинформ, 2017. — 32 с.
10. Клифтон Я. Проектирование пользовательского интерфейса в Android. 2-е изд. – М.: ДМК-Пресс, 2017. – 452 с.
11. Колисниченко Д.Н. Программирование для Android. Самоучитель. - 3-е изд. – СПб.: БХВ-Петербург, 2021. – 288 с.
12. Рудаков А.В. Технология разработки программных продуктов. Практикум: учеб. пособие для студ. - 4 -е изд. – М.: Издательский центр «Академия», 2014. – 192 с.

13. Эванс Б., Флэнаган Д. Java. Справочник разработчика. - 7-е изд. – СПб.: ООО «Диалектика», 2019. – 592 с.

Приложение

Листинг 1. AdminGameAdd

```
package com.shokii.kedwi;

import android.app.Activity;
import android.app.DatePickerDialog;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.widget.DatePicker;
import android.widget.Toast;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AlertDialog;
import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.shokii.kedwi.databinding.ActivityAdminGameAddBinding;
import com.theartofdev.edmodo.cropper.CropImage;
import com.theartofdev.edmodo.cropper.CropImageView;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.Calendar;

public class AdminGameAdd extends AppCompatActivity {
    ActivityAdminGameAddBinding activityAdminGameAddBinding;
    LocalDate currentDate;

    Uri uri;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        activityAdminGameAddBinding =
ActivityAdminGameAddBinding.inflate(getLayoutInflater());

        currentDate = LocalDate.now();
        String date =
currentDate.format(DateTimeFormatter.ofPattern("dd/MM/yyyy"));
        activityAdminGameAddBinding.gameDateBtn.setText(date);


        // Кнопки

activityAdminGameAddBinding.gameLogo.setOnClickListener(this::gameLog
o);

activityAdminGameAddBinding.backBtn.setOnClickListener(this::backBtn)
;
```

```
activityAdminGameAddBinding.gameStatusBtn.setOnClickListener(this::ch
angeGameStatus);

activityAdminGameAddBinding.gameDateBtn.setOnClickListener(this::game
DateBtn);

activityAdminGameAddBinding.gamePEGIBtn.setOnClickListener(this::game
PEGIBtn);

activityAdminGameAddBinding.addBtn.setOnClickListener(this::addBtn);

        setContentView(activityAdminGameAddBinding.getRoot());
    }

    private void gameLogo(View view) {
        CropImage.activity()
                .setAspectRatio(2, 3)
                .setCropShape(CropImageView.CropShape.RECTANGLE)
                .start((Activity) this);
    }
    @Override
    public void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE
&& data != null) {
            uri = CropImage.getActivityResult(data).getUri();
            activityAdminGameAddBinding.gameLogo.setImageURI(uri);
        }
    }

    private void changeGameStatus(View view) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Выберите один вариант");

        String[] options = {"Вышла", "Анонсирована"};
        builder.setItems(options, (dialog, which) ->
activityAdminGameAddBinding.gameStatusBtn.setText(options[which]));

        builder.show();
    }

    private void gameDateBtn(View view) {
        final Calendar calendar = Calendar.getInstance();

        int year = calendar.get(Calendar.YEAR), month =
calendar.get(Calendar.MONTH),
                day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new DatePickerDialog(
this, new DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year, int
monthOfYear, int dayOfMonth) {
                String month = String.valueOf(monthOfYear);
                if (monthOfYear < 10)
                    month = "0" + month;
```

```
activityAdminGameAddBinding.gameDateBtn.setText((dayOfMonth + "/" +
month + "/" + year).toString());
            }
        }, year, month, day);

datePickerDialog.getDatePicker().setMaxDate(System.currentTimeMillis(
));
        datePickerDialog.show();
    }

    private void gamePEGIBtn(View view) {
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("Выберите один вариант");

        String[] options = {"3", "7", "12", "16", "18"};
        builder.setItems(options, (dialog, which) ->
activityAdminGameAddBinding.gamePEGIBtn.setText(options[which]));

        builder.show();
    }

    private void backBtn(View view) {
        startActivity(new Intent(this, MainActivity.class));
        finish();
    }

    private void addBtn(View view) {
        String
                gameStatus =
activityAdminGameAddBinding.gameStatusBtn.getText().toString().equals
("Вышла") ? "out" : "announcement",
                gameName =
activityAdminGameAddBinding.gameNameText.getText().toString(),
                gameDate =
activityAdminGameAddBinding.gameDateBtn.getText().toString(),
                gameDeveloper =
activityAdminGameAddBinding.gameDeveloperText.getText().toString(),
                gamePublisher =
activityAdminGameAddBinding.gamePublisherText.getText().toString(),
                gamePEGI =
activityAdminGameAddBinding.gamePEGIBtn.getText().toString();

        if (gameName.isEmpty() || gameDeveloper.isEmpty() ||
gamePublisher.isEmpty()) {
            Toast.makeText(this, "Не все поля заполнены",
Toast.LENGTH_SHORT).show();
            return;
        }

        DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference().child("games").child(ga
meStatus).child(gameName);
        StorageReference storageRef =
FirebaseStorage.getInstance().getReference().child("games_covers");

        // cover
```

```
        storageRef.child(gameName + "_cover").putFile(uri);
        databaseRef.child("cover").setValue(gameName + "_cover");

        // date
        databaseRef.child("date").setValue(gameDate);

        // developers
        // => DEVELOPER 1

databaseRef.child("developers").child(gameDeveloper).setValue("flag")
;

        // publishers
        databaseRef.child("publishers").setValue(gamePublisher);

        // PEGI
        databaseRef.child("pegi").setValue(gamePEGI);

        startActivity(new Intent(this, MainActivity.class));
        finish();
    }
}
```

Листинг 2. BookmarksPage

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.viewpager.widget.ViewPager;

import com.google.android.material.tabs.TabLayout;
import com.shokii.kedwi.databinding.FragmentBookmarksPageBinding;


public class BookmarksPage extends Fragment {

    public BookmarksPage () {
        super(R.layout.fragment_bookmarks_page);
    }



    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        FragmentBookmarksPageBinding _binding =
FragmentBookmarksPageBinding.inflate(getLayoutInflater());

        TabLayout tabLayout = _binding.tabLayoutBookmarks;
        ViewPager viewPager = _binding.pagerBookmarks;
        tabLayout.setTabMode(TabLayout.MODE_SCROLLABLE);

// Создайте адаптер для ViewPager
```

```
        ViewPagerAdapterBookmarks pagerAdapter = new
ViewPagerAdapterBookmarks(getParentFragmentManager());
        viewPager.setAdapter(pagerAdapter);

// Свяжите TabLayout с ViewPager
        tabLayout.setupWithViewPager(viewPager);

        return _binding.getRoot();
    }
}
```

Листинг 3. BookmarksPageBlank

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import
com.shokii.kedwi.databinding.FragmentBookmarksPageBlankBinding;

import java.util.ArrayList;


public class BookmarksPageBlank extends Fragment {
    private String GAME_TYPE = "";
    FragmentBookmarksPageBlankBinding bookmarksPageBlank;
    FirebaseAuth mAuth = FirebaseAuth.getInstance();
    DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference();
    ArrayList<GameItem> list = new ArrayList<>();

    public BookmarksPageBlank(String game_type) {
        super(R.layout.fragment_bookmarks_page_blank);
        GAME_TYPE = game_type;
    }


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        bookmarksPageBlank =
FragmentBookmarksPageBlankBinding.inflate(getLayoutInflater());


        databaseRef.addValueEventListener(new ValueEventListener() {
            @Override
```

```
            public void onDataChange(@NonNull DataSnapshot snapshot)
{
                list = new ArrayList<>();
                DataSnapshot
                        USER_GAMES =
snapshot.child("users").child(mAuth.getUid()).child("game
statistic"),
                        GAMES = snapshot.child("games").child("out");

                for (DataSnapshot snap :
USER_GAMES.child(GAME_TYPE).getChildren()) {
                    String gameName = snap.getKey();
                    String imgSrc = "@null";
                    if
(GAMES.child(gameName).child("cover").getValue() != null)
                        imgSrc  =
GAMES.child(gameName).child("cover").getValue().toString();
                    String gameStatus = GAME_TYPE;

                    GameItem gameItem = new GameItem(gameName,
imgSrc, gameStatus);

                    list.add(gameItem);
                }

                GridViewAdapter adapter = new
GridViewAdapter(getContext(), list);

bookmarksPageBlank.gamesGridBookmarks.setAdapter(adapter);
            }


            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });

    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {

bookmarksPageBlank.gamesGridBookmarks.setOnItemClickListener(this::it
emClick);
        return bookmarksPageBlank.getRoot();
    }

    private void itemClick(AdapterView<?> adapterView, View view, int
i, long l) {
        GameItem item = list.get(i);

        Bundle bundle = new Bundle();
        bundle.putString("NAME", item.gameTitle);
        bundle.putString("IMG_SRC", item.imgSrc);
        bundle.putString("GAME_STATUS", "out");
        bundle.putString("USER_GAME_STATUS", item.gameStatus);
```

```
        bundle.putBoolean("PAGE", true);

        GameInfo gameInfo = new GameInfo();
        gameInfo.setArguments(bundle);

        if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, gameInfo).commit();
    }
}
```

Листинг 4. CalendarPage

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shokii.kedwi.databinding.FragmentCalendarPageBinding;

import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Locale;
import java.util.Objects;

public class CalendarPage extends Fragment {
    FragmentCalendarPageBinding binding;
    LocalDate currentDate;
    int currentDayOfWeek;
    TextView[] textViews = new TextView[7];

    DateTimeFormatter formatter = DateTimeFormatter.ofPattern("MMMM
yyyy", new Locale("ru"));

    FirebaseAuth mAuth = FirebaseAuth.getInstance();
    DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference();

    public CalendarPage() {
        super (R.layout.fragment_calendar_page);
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
        binding =
FragmentCalendarPageBinding.inflate(getLayoutInflater());

        currentDate = LocalDate.now();
        currentDayOfWeek = currentDate.getDayOfWeek().getValue() - 1;

        textViews = new TextView[]{ binding.d1, binding.d2,
binding.d3,
                binding.d4, binding.d5, binding.d6, binding.d7
        };

        binding.backWeek.setOnClickListener(this::backWeek);
        binding.nextWeek.setOnClickListener(this::nextWeek);

        selectedDate(currentDate, currentDayOfWeek);
    }

    private void updateTextViews() {
        for (int i = 0; i < textViews.length; i++)

textViews[i].setText(String.valueOf(currentDate.plusDays(i -
currentDayOfWeek).getDayOfMonth()));

        if (!textViews[2].getText().toString().isEmpty() &&
Integer.parseInt(textViews[2].getText().toString()) < 7)

binding.currentDate.setText(currentDate.format(formatter));
    }

    private void backWeek(View view) {
        currentDate = currentDate.minusWeeks(1);
        updateTextViews();
    }

    private void nextWeek(View view) {
        currentDate = currentDate.plusWeeks(1);
        updateTextViews();
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        binding.currentDate.setText(currentDate.format(formatter));
        updateTextViews();

        for (int i = 0; i < textViews.length; i++) {
            int finalI = i;
            textViews[i].setOnClickListener(view ->
selectedDate(currentDate, finalI));
        }

        return binding.getRoot();
    }

    private void selectedDate(LocalDate dataClicked, int i) {
        String selected = dataClicked.plusDays(i -
currentDayOfWeek).format(DateTimeFormatter.ofPattern("dd|MM|yyyy"));
```

```
        databaseRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{
                ArrayList<GameItem> list = new ArrayList<>();

                DataSnapshot user_GamesList =

snapshot.child("users").child(mAuth.getUid()).child("game
statistic").child("announcement");

                for (DataSnapshot dates :
user_GamesList.getChildren()) {
                    if (Objects.equals(dates.getKey(), selected)) {
                        for (DataSnapshot games :
dates.getChildren()) {
                            String gameName = games.getKey();

                            String imgSrc = "@null";
                            if
(snapshot.child("games").child("announcement").child(gameName).child(
"cover").getValue() != null)
                                imgSrc =
snapshot.child("games").child("announcement").child(gameName).child("
cover").getValue().toString();

                            GameItem gameItem = new
GameItem(gameName, imgSrc, "not played");
                            list.add(gameItem);
                        }
                    }
                }

                GridViewAdapter adapter = new
GridViewAdapter(getContext(), list);

                if (!list.isEmpty()) {
                    binding.gamesGrid.setAdapter(adapter);
                    binding.message.setVisibility(View.GONE);
                }
                else {
                    binding.gamesGrid.setAdapter(adapter);
                    binding.message.setVisibility(View.VISIBLE);
                }
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) { }
        });
    }
}
```

Листинг 5. EnterAccount

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
```

```
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shokii.kedwi.databinding.FragmentEnterAccountBinding;


public class EnterAccount extends Fragment {
    public EnterAccount() {
        super (R.layout.fragment_enter_account);
    }

    private FragmentEnterAccountBinding _binding;
    private FirebaseAuth _mAuth;
    private FirebaseDatabase _dataBase;
    private DatabaseReference _userRefs;
    private EnterPassword nextFragment = new EnterPassword();
    private Bundle _bundle = new Bundle();



    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        _mAuth = FirebaseAuth.getInstance();
        _dataBase = FirebaseDatabase.getInstance();
        _userRefs = _dataBase.getReference("users");


        _binding =
FragmentEnterAccountBinding.inflate(getLayoutInflater());


        _binding.loginContinue.setOnClickListener(this::EnterEmail);
        _binding.loginVk.setOnClickListener(this::InDev);
        _binding.loginGoogle.setOnClickListener(this::InDev);

        return _binding.getRoot();
    }

    public void EnterEmail(View v) {
        String email = _binding.loginText.getText().toString();

        _userRefs.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{
                boolean isEXIST = false;
```

```
                for (DataSnapshot userUID :snapshot.getChildren()) {
                    String databaseEmail =
userUID.child("email").getValue().toString();
                    if(databaseEmail.equals(email))
                        isEXIST = true;
                }

                _bundle.putString("EMAIL", email);
                _bundle.putBoolean("isEXIST", isEXIST);
                nextFragment.setArguments(_bundle);

                if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, nextFragment).commit();
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) { }
        });
    }

    public void InDev(View v) {
        Toast.makeText(getContext(), "Данная функция находиться в
разработке", Toast.LENGTH_SHORT).show();
    }
}
```

Листинг 6. EnterBirthdate

```
package com.shokii.kedwi;

import android.app.DatePickerDialog;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.DatePicker;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shokii.kedwi.databinding.FragmentEnterBirthdateBinding;

import java.text.SimpleDateFormat;
import java.util.Calendar;




public class EnterBirthdate extends Fragment {
    public EnterBirthdate() { super
(R.layout.fragment_enter_birthdate); }
```

```
    private FragmentEnterBirthdateBinding _binding;
    private FirebaseAuth _mAuth;
    private FirebaseDatabase _dataBase;
    private DatabaseReference _usersRefs, _curentUser;
    private String[] _monthMap = { "Январь", "Февраль", "Март",
"Апрель", "Май", "Июнь", "Июль", "Август", "Сентябрь", "Октябрь",
"Ноябрь", "Декабрь" };
    private int[] _pickDate = new int[3];


    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        _mAuth = FirebaseAuth.getInstance();
        _dataBase = FirebaseDatabase.getInstance();
        _usersRefs = _dataBase.getReference("users");

        _curentUser = _usersRefs.child(_mAuth.getUid());


        _binding =
FragmentEnterBirthdateBinding.inflate(getLayoutInflater());

        _curentUser.child("name").addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{

_binding.textHighlight.setText(snapshot.getValue().toString());
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
}
        });

        Calendar cal = Calendar.getInstance();
        SimpleDateFormat simpleDateFormat = new SimpleDateFormat("d
MMM  y");

_binding.datePicker.setText(simpleDateFormat.format(cal.getTime()));

        _binding.backBtn.setOnClickListener(this::Back);
        _binding.datePicker.setOnClickListener(this::dataPick);

_binding.birthdateContinue.setOnClickListener(this::Continue);

        return _binding.getRoot();
    }

    private void dataPick(View v) {
        final Calendar calendar = Calendar.getInstance();


        int year = calendar.get(Calendar.YEAR),
            month = calendar.get(Calendar.MONTH),
```

```
            day = calendar.get(Calendar.DAY_OF_MONTH);

        DatePickerDialog datePickerDialog = new DatePickerDialog(
getContext(), new DatePickerDialog.OnDateSetListener() {
            @Override
            public void onDateSet(DatePicker view, int year, int
monthOfYear, int dayOfMonth) {
                _pickDate[0] = dayOfMonth;
                _pickDate[1] = monthOfYear;
                _pickDate[2] = year;
                _binding.datePicker.setText(_pickDate[0] + "  " +
(_monthMap[_pickDate[1]]) + "  " + _pickDate[2]);
            }
        }, year, month, day);

datePickerDialog.getDatePicker().setMaxDate(System.currentTimeMillis(
));
        datePickerDialog.show();
    }

    private void Continue(View v) {
        _curentUser.child("birthdate").setValue(_pickDate[0] + "/" +
_pickDate[1] + "/" + _pickDate[2]);

        if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new EnterGender()).commit();
    }

    private void Back(View v) {
        if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new EnterName()).commit();
    }

}
```

Листинг 7. EnterGender

```
package com.shokii.kedwi;

import android.content.Intent;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.shokii.kedwi.databinding.FragmentEnterGenderBinding;
```

```
public class EnterGender extends Fragment {
    public EnterGender() {
        super (R.layout.fragment_enter_gender);
    }

    private FragmentEnterGenderBinding _binding;
    private FirebaseAuth _mAuth;
    private FirebaseDatabase _dataBase;
    private DatabaseReference _userRefs;



    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        _mAuth = FirebaseAuth.getInstance();
        _dataBase = FirebaseDatabase.getInstance();
        _userRefs = _dataBase.getReference("users");

        _binding =
FragmentEnterGenderBinding.inflate(getLayoutInflater());


        _binding.maleBtn.setOnClickListener(this::setGender);
        _binding.femaleBtn.setOnClickListener(this::setGender);
        _binding.wontSayBtn.setOnClickListener(this::setGender);

        _binding.backBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new EnterBirthdate()).commit();
            }
        });

        return _binding.getRoot();
    }

    public void setGender(View v) {
        String gender;
        if (v.getId() == _binding.maleBtn.getId())

_userRefs.child(_mAuth.getUid().toString()).child("gender").setValue(
"male");
        else if (v.getId() == _binding.femaleBtn.getId())

_userRefs.child(_mAuth.getUid().toString()).child("gender").setValue(
"female");
        else

_userRefs.child(_mAuth.getUid().toString()).child("gender").setValue(
"wontSay");

        startActivity(new Intent(getContext(), MainActivity.class));
    }
```

```
}
```

Листинг 8. EnterName

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.shokii.kedwi.databinding.FragmentEnterNameBinding;


public class EnterName extends Fragment {
    public EnterName () { super (R.layout.fragment_enter_name); }
    private FragmentEnterNameBinding _binding;

    private FirebaseAuth _mAuth;
    private FirebaseDatabase _dataBase;
    private DatabaseReference _userRefs;


    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        _mAuth = FirebaseAuth.getInstance();
        _dataBase = FirebaseDatabase.getInstance();
        _userRefs = _dataBase.getReference("users");

        _binding =
FragmentEnterNameBinding.inflate(getLayoutInflater());

        _binding.nameContinue.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {

_userRefs.child(_mAuth.getUid().toString()).child("name").setValue(_b
inding.nameText.getText().toString());
                if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new EnterBirthdate()).commit();
            }
        });

        _binding.backBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (getFragmentManager() != null)
```

```
getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new RegistrationContinue()).commit();
            }
        });

        return _binding.getRoot();
    }
}
```

Листинг 9. EnterPassword

```
package com.shokii.kedwi;

import android.content.Intent;
import android.os.Bundle;
import android.text.method.HideReturnsTransformationMethod;
import android.text.method.PasswordTransformationMethod;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import com.google.firebase.auth.AuthResult;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.shokii.kedwi.databinding.FragmentEnterPasswordBinding;


public class EnterPassword extends Fragment {
    private FragmentEnterPasswordBinding _binding;
    private Bundle _bundle;

    private FirebaseAuth _mAuth;
    private FirebaseDatabase _dataBase;
    private DatabaseReference _userRefs;



    public EnterPassword() {
        super (R.layout.fragment_enter_password);
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        _mAuth = FirebaseAuth.getInstance();
        _dataBase = FirebaseDatabase.getInstance();
        _userRefs = _dataBase.getReference().child("users");


        _binding =
FragmentEnterPasswordBinding.inflate(getLayoutInflater());
```

```
        _bundle = getArguments();

        String email = _bundle.getString("EMAIL");

        if (_bundle.getBoolean("isEXIST")) {
            _binding.passwordTitle.setText("Введите пароль");

            _binding.passwordContinue.setOnClickListener((view) ->
LogInWithEmailAndPassword (email));
        }
        else {
            _binding.passwordTitle.setText("Создайте пароль");

            _binding.passwordContinue.setOnClickListener((view) ->
RegistrationWithEmailAndPassword (email));
        }


        _binding.showUpBtn.setOnClickListener(new
View.OnClickListener() {
            boolean showPassword = true;

            @Override
            public void onClick(View v) {

                if (showPassword) {

_binding.passwordText.setTransformationMethod(HideReturnsTransformati
onMethod.getInstance());

_binding.showUpBtn.setBackgroundResource(R.drawable.hide_password_btn
);
                    showPassword = false;
                }
                else {

_binding.passwordText.setTransformationMethod(PasswordTransformationM
ethod.getInstance());

_binding.showUpBtn.setBackgroundResource(R.drawable.show_password_btn
);
                    showPassword = true;
                }
            }
        });

        _binding.backBtn.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new EnterAccount()).commit();
            }
        });

        return _binding.getRoot();
```

```
    }

    private void LogInWithEmailAndPassword (String email) {

        _mAuth.signInWithEmailAndPassword(email,
_binding.passwordText.getText().toString())
                .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                        if (task.isSuccessful()) {
                            startActivity(new Intent(getContext(),
MainActivity.class));
                        }
                        else {
                            Toast.makeText(getContext(), "LOGIN:
False", Toast.LENGTH_SHORT).show();
                        }
                    }

                });
    }

    private void RegistrationWithEmailAndPassword(String email) {
        _mAuth.createUserWithEmailAndPassword(email,
_binding.passwordText.getText().toString())
                .addOnCompleteListener(new
OnCompleteListener<AuthResult>() {
                    @Override
                    public void onComplete(@NonNull Task<AuthResult>
task) {
                        if (task.isSuccessful()) {
                            // Записываем в БД информацию
пользователя

_userRefs.child(_mAuth.getUid()).child("email").setValue(email);

_userRefs.child(_mAuth.getUid()).child("password").setValue(_binding.
passwordText.getText().toString());

_userRefs.child(_mAuth.getUid()).child("status").setValue("Статус не
указан");

                            if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new RegistrationContinue()).commit();
                        }
                        else {
                            Toast.makeText(getContext(),
"REGISTRATION: False", Toast.LENGTH_SHORT).show();
                        }
                    }
                });
    }
}
```

Листинг 10. GameInfo

```
package com.shokii.kedwi;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;

import com.bumptech.glide.Glide;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.shokii.kedwi.databinding.FragmentGameInfoBinding;

import java.util.ArrayList;
import java.util.Objects;

public class GameInfo extends Fragment {
    private FirebaseDatabase database;
    private FirebaseStorage storage;
    private StorageReference storageRef;
    private DatabaseReference gameRef;
    private FragmentGameInfoBinding binding;
    Bundle bundle = new Bundle();

    FirebaseAuth mAuth;
    DatabaseReference userGameRef;

    private int[] backgroundResources = {
            R.drawable.status_not_played,
            R.drawable.status_passing,
            R.drawable.status_planned,
            R.drawable.status_pass,
            R.drawable.status_postponed,
            R.drawable.status_abandoned,
            R.drawable.rounded_btn_game_sec,
            R.drawable.rounded_btn_game
    };

    String[] options = {"Не играл", "Прохожу", "В планах",
"Пройдено", "Отложено", "Брошено"};
    String[] temp = {"not played", "passing", "planned", "pass",
"postponed", "abandoned"};
```

```
    private String[] _monthMap = { "январь", "февраль", "март",
"апрель", "май", "июнь", "июль", "август", "сентябрь", "октябрь",
"ноябрь", "декабрь" };

    public GameInfo() {
        super(R.layout.fragment_game_info);
    }
    String gameName ;

    @Override
public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
    binding = FragmentGameInfoBinding.inflate(getLayoutInflater());
    bundle = getArguments();
    mAuth = FirebaseAuth.getInstance();
    userGameRef =
FirebaseDatabase.getInstance().getReference("users").child(mAuth.getU
id()).child("game statistic");
    gameName = bundle.getString("NAME");
    String imgSrc = bundle.getString("IMG_SRC");
    String gameStatus = bundle.getString("GAME_STATUS");
    String userGameStatus = bundle.getString("USER_GAME_STATUS");
    Button btn = binding.changeGameStatus;
    int i;
    for (i = 0; i < options.length - 1; i++) {
        if (userGameStatus.equals(temp[i])) break;
    }

btn.setBackground(getResources().getDrawable(backgroundResources[i]))
;
    btn.setTextColor(getResources().getColor(R.color.grey));
    if (i == 0)
btn.setTextColor(getResources().getColor(R.color.black));
    btn.setText(options[i]);
    if (gameStatus.equals("announcement")) {
        binding.changeGameStatus.setVisibility(View.GONE);
        binding.addAnocement.setVisibility(View.VISIBLE);
    }
    database = FirebaseDatabase.getInstance();
    gameRef =
database.getReference().child("games").child(gameStatus).child(gameNa
me);
    storage = FirebaseStorage.getInstance();
    storageRef = storage.getReference().child("games_covers");
    gameRef.addValueEventListener(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {

storageRef.child(imgSrc).getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
                @Override
                public void onSuccess(Uri uri) {
                    Glide.with(getContext())
                            .load(uri)
                            .optionalCenterCrop()
                            .into(binding.gameLogo);
                }
            });
```

```
            if
(!snapshot.child("date").getValue().toString().equals("")) {
                String[] date =
snapshot.child("date").getValue().toString().split("/");
                binding.dateDesc.setText(date[0] + " " +
_monthMap[Integer.parseInt(date[1]) - 1] + " " + date[2]);
            }
            else
                binding.dateDesc.setText("Дата будет объявлена
позже");
            ArrayList<String> temp = new ArrayList<>();
            for (DataSnapshot developers :
snapshot.child("developers").getChildren())
                temp.add(developers.getKey());
            for (int i = 0; i < temp.size() - 1; i++)
                binding.developerDesc.append(temp.get(i) + "\n");
            binding.developerDesc.append(temp.get(temp.size() - 1));
            if
(!snapshot.child("publishers").getValue().toString().equals(""))

binding.publisherDesc.setText(snapshot.child("publishers").getValue()
.toString());
            if
(!snapshot.child("pegi").getValue().toString().equals("")) {
                int pegiValue =
Integer.parseInt(snapshot.child("pegi").getValue().toString());
                int resourceId = getResources().getIdentifier("pegi"
+ pegiValue, "drawable", "com.shokii.kedwi");
                binding.pegiDesc.setImageResource(resourceId);
            }

            String OutDate = binding.dateDesc.getText().toString();
            String[] OutDateGame;

            if (OutDate.equals("Дата будет объявлена позже"))
                OutDate = "@null";
            else {
                OutDateGame = OutDate.split(" ");
                String day = OutDateGame[0];
                String month = " ";
                for (int j = 0; j < _monthMap.length; j++)
                    if (Objects.equals(OutDateGame[1], _monthMap[j]))
{
                        month = String.valueOf(j + 1);
                        if (month.length() == 1) {
                            month = "0" + month;
                        }
                    }
                String year = OutDateGame[2];
                OutDate = day + "|" + month + "|" + year;
            }
            String finalOutDate = OutDate;

userGameRef.child("announcement").addValueEventListener(new
ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot
snapshot) {
```

```
                    if
(snapshot.child(finalOutDate).child(gameName).getValue() == null)
                        binding.addAnocement.setText("Добавить в
список отслеживания");
                    else
                        binding.addAnocement.setText("Удалить из
списока отслеживания");
                }

                @Override
                public void onCancelled(@NonNull DatabaseError error)
{

                }
            });

        }
        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
    binding.backBtn.setOnClickListener(this::back);

binding.changeGameStatus.setOnClickListener(this::changeGameStatus);
    binding.gameName.setText(gameName);




    binding.addAnocement.setOnClickListener(this::addAnnouncement);
    return binding.getRoot();
}
    private void addAnnouncement (View view) {
        String OutDate = binding.dateDesc.getText().toString();
        String[] OutDateGame;

        if (OutDate.equals("Дата будет объявлена позже"))
            OutDate = "@null";
        else {
            OutDateGame = OutDate.split(" ");
            String day = OutDateGame[0];
            String month = " ";
            for (int j = 0; j < _monthMap.length; j++)
                if (Objects.equals(OutDateGame[1], _monthMap[j])) {
                    month = String.valueOf(j + 1);
                    if (month.length() == 1) {
                        month = "0" + month;
                    }
                }
            String year = OutDateGame[2];
            OutDate = day + "|" + month + "|" + year;
        }
        if (binding.addAnocement.getText().toString().charAt(0) ==
'Д') {
```

```
userGameRef.child("announcement").child(OutDate).child(gameName).setV
alue("flag");
            binding.addAnocement.setText("Удалить из списока
отслеживания");
        }
        else {

userGameRef.child("announcement").child(OutDate).child(gameName).remo
veValue();
            binding.addAnocement.setText("Добавить в список
отслеживания");
        }
    }
    private void changeGameStatus(View view) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
        builder.setTitle("Выберите один вариант");

        Button btn = binding.changeGameStatus;
        String search = btn.getText().toString();
        int i;
        for (i = 0; i < options.length; i++) {
            if (search.equals(options[i])) break;
        }
        String currentStatus = temp[i];
        String gameName = binding.gameName.getText().toString();

        builder.setItems(options, (dialog, which) -> {

btn.setBackground(getResources().getDrawable(backgroundResources[whic
h]));
            btn.setTextColor(getResources().getColor(R.color.grey));
            if (which == 0)
btn.setTextColor(getResources().getColor(R.color.black));
            btn.setText(options[which]);

            changeUserGameStatus(gameName, currentStatus,
temp[which]);
        });

        builder.show();
    }
    private void changeUserGameStatus(String gameName, String
prevStatus, String newStatus) {
        userGameRef.child(prevStatus).child(gameName).removeValue();

userGameRef.child(newStatus).child(gameName).setValue("flag");
    }
    private void back(View view) {
        startActivity(new Intent(getContext(), MainActivity.class));
//        if (bundle.getBoolean("PAGE"))
//
getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new BookmarksPage()).commit();
//        else
```

```
//
getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new HomePage()).commit();
    }
}
```

Листинг 11. GameItem

```
package com.shokii.kedwi;

public class GameItem {
    final String imgSrc, gameTitle, gameStatus;

    public GameItem(String  gameTitle, String imgSrc, String
gameStatus) {
        this.gameTitle = gameTitle; this.imgSrc = imgSrc;
this.gameStatus = gameStatus;
    }
}
```

Листинг 12. GridViewAdapter

```
package com.shokii.kedwi;

import android.content.Context;
import android.net.Uri;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;

import com.bumptech.glide.Glide;
import com.bumptech.glide.load.engine.DiskCacheStrategy;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;

import java.util.ArrayList;


public class GridViewAdapter extends ArrayAdapter<GameItem> {
    private FirebaseStorage storage;
    private StorageReference storageRef;

    String[] options = {"Не играл", "Прохожу", "В планах",
"Пройдено", "Отложено", "Брошено"};
    String[] temp = {"not played", "passing", "planned", "pass",
"postponed", "abandoned"};
    private int[] backgroundResources = {
            R.color.grey,
            R.color.passing,
            R.color.planned,
            R.color.pass,
            R.color.postponed,
```

```
            R.color.abandoned
    };

    public GridViewAdapter(@NonNull Context context,
ArrayList<GameItem> gameItems) {
        super(context, 0, gameItems);
    }


    @NonNull
    @Override
    public View getView(int position, @Nullable View convertView,
@NonNull ViewGroup parent) {
        storage = FirebaseStorage.getInstance();
        storageRef = storage.getReference().child("games_covers");

        View listView = convertView;
        if (listView == null)
            listView =
LayoutInflater.from(getContext()).inflate(R.layout.game_item, parent,
false);

        GameItem gameItem = getItem(position);

        ImageView gameLogo = (ImageView)
listView.findViewById(R.id.gameLogo);
        TextView gameText = (TextView)
listView.findViewById(R.id.gameText);
        TextView gameStatus = (TextView)
listView.findViewById(R.id.gameStatus);


storageRef.child(gameItem.imgSrc).getDownloadUrl().addOnSuccessListen
er(new OnSuccessListener<Uri>() {
            @Override
            public void onSuccess(Uri uri) {
                Glide.with(getContext())
                        .load(uri)
                        .diskCacheStrategy(DiskCacheStrategy.DATA)
                        .placeholder(R.drawable.game_logos)
                        .override(300, 450)
                        .optionalCenterCrop()
                        .into(gameLogo);
            }
        });

        gameText.setText(gameItem.gameTitle);

        int i;
        for (i = 0; i < options.length - 1; i++) {
            if (gameItem.gameStatus.equals(temp[i])) break;
        }
        if (i != 0) {

gameStatus.setBackground(getContext().getResources().getDrawable(back
groundResources[i]));
```

```
gameStatus.setTextColor(getContext().getResources().getColor(R.color.
grey));
            gameStatus.setText(options[i]);
        }

        return listView;
    }
}
```

Листинг 13. HomePage

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.viewpager.widget.ViewPager;

import com.google.android.material.tabs.TabLayout;
import com.shokii.kedwi.databinding.FragmentHomePageBinding;

public class HomePage extends Fragment {
    public HomePage () {
        super(R.layout.fragment_home_page);
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        FragmentHomePageBinding _binding =
FragmentHomePageBinding.inflate(getLayoutInflater());

        TabLayout tabLayout = _binding.tabLayout;
        ViewPager viewPager = _binding.pager;

        ViewPagerAdapter pagerAdapter = new
ViewPagerAdapter(getParentFragmentManager());
        viewPager.setAdapter(pagerAdapter);

        tabLayout.setupWithViewPager(viewPager);

        return _binding.getRoot();
    }
}
```

Листинг 14. HomePage_Blank

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
```

```
import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shokii.kedwi.databinding.FragmentHomePageBlankBinding;

import java.util.ArrayList;
import java.util.Objects;

public class HomePage_Blank extends Fragment {
    private String GAME_TYPE = "";
    FragmentHomePageBlankBinding bookmarksPageBlank;
    FirebaseAuth mAuth = FirebaseAuth.getInstance();
    DatabaseReference databaseRef =
FirebaseDatabase.getInstance().getReference();
    ArrayList<GameItem> list = new ArrayList<>();

    public HomePage_Blank(String game_type){
        super(R.layout.fragment_home_page_blank);
        GAME_TYPE = game_type;
    }


    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        bookmarksPageBlank =
FragmentHomePageBlankBinding.inflate(getLayoutInflater());


bookmarksPageBlank.gamesGrid.setOnItemClickListener(this::itemClick);

        databaseRef.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{
                list = new ArrayList<>();
                DataSnapshot
                        USER_GAMES =
snapshot.child("users").child(mAuth.getUid()).child("game
statistic"),
                        GAMES =
snapshot.child("games").child(GAME_TYPE);

                for (DataSnapshot snap : GAMES.getChildren()) {
                    String gameName = snap.getKey();
                    String imgSrc = "@null";
                    if (snap.child("cover").getValue() != null)
                        imgSrc  =
GAMES.child(gameName).child("cover").getValue().toString();
                    String gameStatus = "not played";
```

```
                    for (DataSnapshot users_list :
USER_GAMES.getChildren())
                        for (DataSnapshot game :
users_list.getChildren())
                            if (Objects.equals(gameName,
game.getKey()))
                                gameStatus = users_list.getKey();

                    GameItem gameItem = new GameItem(gameName,
imgSrc, gameStatus);
                    list.add(gameItem);
                }

                GridViewAdapter adapter = new
GridViewAdapter(getContext(), list);
                bookmarksPageBlank.gamesGrid.setAdapter(adapter);
            }


            @Override
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {

bookmarksPageBlank.gamesGrid.setOnItemClickListener(this::itemClick);
        return bookmarksPageBlank.getRoot();
    }

    private void itemClick(AdapterView<?> adapterView, View view, int
i, long l) {
        GameItem item = list.get(i);
        Bundle bundle = new Bundle();
        bundle.putString("NAME", item.gameTitle);
        bundle.putString("IMG_SRC", item.imgSrc);
        bundle.putString("GAME_STATUS", GAME_TYPE);
        bundle.putString("USER_GAME_STATUS", item.gameStatus);
        bundle.putBoolean("PAGE", false);

        GameInfo gameInfo = new GameInfo();
        gameInfo.setArguments(bundle);

        if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, gameInfo).commit();
    }
}
```

Листинг 15. Launch

```
package com.shokii.kedwi;

import android.content.Intent;
```

```
import android.os.Bundle;

import androidx.appcompat.app.AppCompatActivity;

import com.google.firebase.auth.FirebaseAuth;
import com.shokii.kedwi.databinding.ActivityLaunchBinding;

public class Launch extends AppCompatActivity {
    private ActivityLaunchBinding _binding;
    private FirebaseAuth _mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        _binding =
ActivityLaunchBinding.inflate(getLayoutInflater());
        setContentView(_binding.getRoot());

        _mAuth = FirebaseAuth.getInstance();
        //  Если пользователь уже входил на устройстве то пропускаем
вход/регистрацию
        if (_mAuth.getCurrentUser() != null) {
            startActivity(new Intent(this, MainActivity.class));
            finish();
        }

        getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container_view, new
EnterAccount()).commit();
//
//        getSupportFragmentManager().beginTransaction()
//                .add(R.id.fragment_container_view, new
HomePage()).commit();
    }

}
```

Листинг 16. CalendarPage

```
package com.shokii.kedwi;

import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;

import androidx.annotation.NonNull;
import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import androidx.fragment.app.Fragment;

import com.google.android.gms.tasks.OnCompleteListener;
import com.google.android.gms.tasks.Task;
import
com.google.android.material.bottomnavigation.BottomNavigationView;
import com.google.firebase.auth.FirebaseAuth;
```

```
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.google.firebase.storage.UploadTask;
import com.shokii.kedwi.databinding.ActivityMainBinding;
import com.theartofdev.edmodo.cropper.CropImage;

public class MainActivity extends AppCompatActivity /* implements
BottomNavigationView.OnNavigationItemSelectedListener */ {
    private ActivityMainBinding _binding;

    FirebaseAuth mAuth;
    FirebaseStorage storage;
    StorageReference storageRef;
    DatabaseReference userRefs;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        _binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(_binding.getRoot());

        mAuth = FirebaseAuth.getInstance();

        storage = FirebaseStorage.getInstance();
        storageRef = storage.getReference().child("user_image");

        _binding.bottomNav.setSelectedItemId(R.id.menu_home);


        HomePage homePage = new HomePage();
        CalendarPage calendarPage = new CalendarPage();
        BookmarksPage bookmarksPage = new BookmarksPage();
        UserPage userPage = new UserPage();

        _binding.bottomNav.setOnNavigationItemSelectedListener(new
BottomNavigationView.OnNavigationItemSelectedListener() {

            @Override
            public boolean onNavigationItemSelected(MenuItem item) {
                int curPage = _binding.bottomNav.getSelectedItemId();

                if (item.getItemId() == R.id.menu_home && curPage !=
R.id.menu_home) {
                    // menu_home
                    loadFragment(homePage);
                    return true;
                }
                if (item.getItemId() == R.id.menu_calendar && curPage
!= R.id.menu_calendar) {
                    // menu_calendar
                    loadFragment(calendarPage);
```

```
                    return true;
                }if (item.getItemId() == R.id.menu_bookmarks &&
curPage != R.id.menu_bookmarks) {
                    // menu_bookmarks
                    loadFragment(bookmarksPage);
                    return true;
                }if (item.getItemId() == R.id.menu_user && curPage !=
R.id.menu_user) {
                    // menu_user
                    loadFragment(userPage);
                    return true;
                }
                return false;
            }
        });

        userRefs =
FirebaseDatabase.getInstance().getReference("users").child(mAuth.getU
id());

        userRefs.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{
                if
(snapshot.child("email").getValue().toString().equals("admin@kedwi.co
m"))
                    _binding.adminAdd.setVisibility(View.VISIBLE);
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
}
        });

        _binding.adminAdd.setOnClickListener(this::adminAdd);

        getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container_view,
homePage).commit();
    }

    private void adminAdd(View view) {
        startActivity(new Intent(this, AdminGameAdd.class));
    }

    private void loadFragment(Fragment fragment) {
        getSupportFragmentManager().beginTransaction()
                .replace(R.id.fragment_container_view,
fragment).commit();
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode,
@Nullable Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == CropImage.CROP_IMAGE_ACTIVITY_REQUEST_CODE
&& data != null) {
```

```
            Uri uri = CropImage.getActivityResult(data).getUri();

storageRef.child(mAuth.getUid()).putFile(uri).addOnCompleteListener(n
ew OnCompleteListener<UploadTask.TaskSnapshot>() {
                @Override
                public void onComplete(@NonNull
Task<UploadTask.TaskSnapshot> task) {
                    Toast.makeText(MainActivity.this, "Фото
обновлено", Toast.LENGTH_SHORT).show();
                }
            });
        }
    }
}
```

Листинг 17. RegistrationContinue

```
package com.shokii.kedwi;

import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import
com.shokii.kedwi.databinding.FragmentRegistrationContinueBinding;


public class RegistrationContinue extends Fragment {
    public RegistrationContinue () {super
(R.layout.fragment_registration_continue);}
    private FragmentRegistrationContinueBinding _binding;

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        _binding =
FragmentRegistrationContinueBinding.inflate(getLayoutInflater());

        _binding.registrationContinue.setOnClickListener(new
View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, new EnterName()).commit();
            }
        });

        return _binding.getRoot();
    }
}
```

Листинг 18. Settings

```
package com.shokii.kedwi;
```

```
import android.app.Activity;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.text.InputType;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.EditText;

import androidx.annotation.NonNull;
import androidx.appcompat.app.AlertDialog;
import androidx.fragment.app.Fragment;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.shokii.kedwi.databinding.FragmentSettingsBinding;
import com.theartofdev.edmodo.cropper.CropImage;
import com.theartofdev.edmodo.cropper.CropImageView;

public class Settings extends Fragment {
    FragmentSettingsBinding SettingsBinding;

    public Settings() {
        super (R.layout.fragment_user_page);
    }

    FirebaseAuth mAuth;
    FirebaseDatabase dataBase;
    DatabaseReference userRefs;



    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        mAuth = FirebaseAuth.getInstance();
        dataBase = FirebaseDatabase.getInstance();
        userRefs = dataBase.getReference("users");


userRefs.child(mAuth.getUid().toString()).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{

SettingsBinding.changeUserStatusDescription.setText(snapshot.child("s
tatus").getValue().toString());
            }

            @Override
```

```
            public void onCancelled(@NonNull DatabaseError error) {

            }
        });

        SettingsBinding =
FragmentSettingsBinding.inflate(getLayoutInflater());
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        SettingsBinding.backBtn.setOnClickListener(this::back);
        SettingsBinding.out.setOnClickListener(this::logOut);


SettingsBinding.changeUserName.setOnClickListener(this::changeUserNam
e);

SettingsBinding.changeUserStatus.setOnClickListener(this::changeUserS
tatus);

SettingsBinding.changeUserPassword.setOnClickListener(this::changeUse
rPassword);


SettingsBinding.changeUserImage.setOnClickListener(this::changeUserIm
age);

        return SettingsBinding.getRoot();
    }

    private void changeUserImage(View view) {
        CropImage.activity()
                .setAspectRatio(1, 1)
                .setRequestedSize(360, 360)
                .setCropShape(CropImageView.CropShape.OVAL)
                .start((Activity) getContext());
    }

    private void changeUserPassword(View view) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
        builder.setTitle("Введите новый пароль");

        final EditText input = new EditText(getContext());
        input.setInputType(InputType.TYPE_CLASS_TEXT);
        builder.setView(input);

        builder.setPositiveButton("Применить", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

mAuth.getCurrentUser().updatePassword(input.getText().toString());

userRefs.child(mAuth.getUid()).child("password").setValue(input.getTe
xt().toString());
```

```
            }
        });

        builder.setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });

        builder.show();
    }
    private void changeUserStatus(View view) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
        builder.setTitle("Введите новый статус");

        final EditText input = new EditText(getContext());
        input.setInputType(InputType.TYPE_CLASS_TEXT);
        builder.setView(input);

        builder.setPositiveButton("Применить", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

userRefs.child(mAuth.getUid().toString()).child("status").setValue(in
put.getText().toString());
            }
        });

        builder.setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });

        builder.show();
    }
    private void changeUserName(View view) {
        AlertDialog.Builder builder = new
AlertDialog.Builder(getContext());
        builder.setTitle("Введите новое имя пользователя");

        final EditText input = new EditText(getContext());
        input.setInputType(InputType.TYPE_CLASS_TEXT);
        builder.setView(input);

        builder.setPositiveButton("Применить", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {

userRefs.child(mAuth.getUid().toString()).child("name").setValue(inpu
t.getText().toString());
```

```
            }
        });

        builder.setNegativeButton("Отмена", new
DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.cancel();
            }
        });

        builder.show();
    }
    private void back(View view) {
        UserPage userPageFragment = new UserPage();
        if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, userPageFragment).commit();
    }
    private void logOut(View view) {
        FirebaseAuth.getInstance().signOut();
        startActivity(new Intent(getContext(), Launch.class));
    }
}
```

Листинг 19. UserPage

```
package com.shokii.kedwi;

import android.net.Uri;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;

import com.bumptech.glide.Glide;
import com.bumptech.glide.load.engine.DiskCacheStrategy;
import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;
import com.google.firebase.storage.FirebaseStorage;
import com.google.firebase.storage.StorageReference;
import com.shokii.kedwi.databinding.FragmentUserPageBinding;

import java.util.Objects;


public class UserPage extends Fragment {
    FragmentUserPageBinding UserPageBinding;

    FirebaseAuth mAuth;
```

```
    FirebaseDatabase dataBase;
    DatabaseReference userRefs;

    FirebaseStorage storage;
    StorageReference storageRef;

    public UserPage() {
        super (R.layout.fragment_user_page);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        UserPageBinding =
FragmentUserPageBinding.inflate(getLayoutInflater());

        mAuth = FirebaseAuth.getInstance();
        dataBase = FirebaseDatabase.getInstance();
        userRefs = dataBase.getReference("users");

        storage = FirebaseStorage.getInstance();
        storageRef =
storage.getReference().child("user_image").child(mAuth.getUid());


userRefs.child(mAuth.getUid().toString()).addValueEventListener(new
ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot)
{

UserPageBinding.userName.setText(snapshot.child("name").getValue().to
String());

UserPageBinding.userText.setText(snapshot.child("status").getValue().
toString());

                DataSnapshot games = snapshot.child("game
statistic");

UserPageBinding.userGameGoingNUM.setText(Objects.requireNonNull(games
.child("passing").getChildrenCount()).toString());

UserPageBinding.userGamePlanNUM.setText(Objects.requireNonNull(games.
child("planned").getChildrenCount()).toString());

UserPageBinding.userGamePassNUM.setText(Objects.requireNonNull(games.
child("pass").getChildrenCount()).toString());

UserPageBinding.userGamePostponedNUM.setText(Objects.requireNonNull(g
ames.child("postponed").getChildrenCount()).toString());

UserPageBinding.userGameAbandonedNUM.setText(Objects.requireNonNull(g
ames.child("abandoned").getChildrenCount()).toString());
            }

            @Override
            public void onCancelled(@NonNull DatabaseError error) {
```

```
            }
        });

        storageRef.getDownloadUrl().addOnSuccessListener(new
OnSuccessListener<Uri>() {
            @Override
            public void onSuccess(Uri uri) {
                Glide.with(getContext())
                        .load(uri)
                        .diskCacheStrategy(DiskCacheStrategy.DATA)
                        .circleCrop()
                        .into(UserPageBinding.userImage);
            }
        });
    }

    @Override
    public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {


UserPageBinding.btn.setOnClickListener(this::startSettingFragment);



        return UserPageBinding.getRoot();
    }

    private void startSettingFragment(View view) {
        Settings settingsFragment = new Settings();
        if (getFragmentManager() != null)

getFragmentManager().beginTransaction().replace(R.id.fragment_contain
er_view, settingsFragment).commit();
    }
}
```

Листинг 20. ViewPagerAdapter

```
package com.shokii.kedwi;

import androidx.annotation.NonNull;
import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;

public class ViewPagerAdapter extends FragmentPagerAdapter {
    private static final int NUM_TABS = 2; // Количество вкладок
    private static final String[][] TABS_LABEL = {
            {"Вышедшие", "Анонсировано"},
            {"out", "announcement"}};

    public ViewPagerAdapter(@NonNull FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {
```

```
        return new HomePage_Blank(TABS_LABEL[1][position]);
    }

    @Override
    public int getCount() {
        return NUM_TABS;
    }

    @Override
    public CharSequence getPageTitle(int position) {
        return TABS_LABEL[0][position];
    }
}
```

Листинг 21. ViewPagerAdapterBookmarks

```
package com.shokii.kedwi;

import androidx.fragment.app.Fragment;
import androidx.fragment.app.FragmentManager;
import androidx.fragment.app.FragmentPagerAdapter;

public class ViewPagerAdapterBookmarks extends FragmentPagerAdapter {
    private static final int NUM_TABS = 5; // Количество вкладок
    private static final String[][] TABS_LABEL = {
            {"Прохожу", "В планах", "Пройдено", "Отложено",
"Брошено"},
            {"passing", "planned", "pass", "postponed",
"abandoned"}};

    public ViewPagerAdapterBookmarks(FragmentManager fm) {
        super(fm);
    }

    @Override
    public Fragment getItem(int position) {
        return new BookmarksPageBlank(TABS_LABEL[1][position]);
    }

    @Override
    public int getCount() {
        return NUM_TABS;
    }

    @Override
    public CharSequence getPageTitle(int position) {
        return TABS_LABEL[0][position];
    }
}
```