Частное учреждение образования

«Колледж бизнеса и права»

ОТЧЁТ

по производственной ТЕХНОЛОГИЧЕСКОЙ ПРАКТИКЕ

ОП Т.1910016

|  |  |
| --- | --- |
| Руководитель практики от предприятия | Шикунец Д. В. |
| Руководитель практики от колледжа | Коропа Е. Н. |
| Учащийся | Короленко Д. А. |

2025

СОДЕРЖАНИЕ

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.1910016

Разраб.

Короленко Д.А

Провер.

Коропа Е.Н.

Реценз.

Н. Контр.

Утверд.

*Отчет*

*по производственной*

*технологической практике*

Лит.

Листов

КБП

50

у

1 Организационно-функциональная структура предприятия и подразделения, характеристики основных видов деятельности 3

2 Должностные обязанности 5

3 Программное обеспечение, используемое на предприятии 6

4 Реализация индивидуального задания 7

4.1 Исследование предметной области 7

4.2 Проектирование модели 8

4.3 Организация данных 9

4.3.1 Структура организации данных 9

4.3.2 Структура json файла про матчи 10

4.3.3 Общение клиента и сервера 12

4.4 Концептуальный прототип 14

4.4.1 Главная страница 15

4.4.2 Страница матчей 16

4.4.3 Страница matchInfo 17

4.4.4 Страница Rankings 20

4.5 Реализация функций 22

4.5.1 Реализация раздела новостей 25

4.5.2 Реализация раздела расписания матчей 25

4.5.3 Реализация раздела расписания матчей 25

4.6 Функциональное тестирование 25

Выводы 28

Список информационных источников 29

Приложение А Текст программных модулей 30

Приложение Б Результаты работы программы 48

1. Организационно-функциональная структура предприятия и подразделения, характеристики основных видов деятельности

Общество с ограниченной ответственностью ООО «Пурпльхаб» является резидентом Парка высоких технологий. ООО «Пурпльхаб» является коммерческой организацией с иностранными инвестициями (100%).   
Основной вид деятельности компании - деятельность в области компьютерного программирования. Офис организации находится по адресу г. Минск, ул., Революционная 12.

В офисе работает 6 человека, которые занимаются разработкой программного обеспечения, а также 3 сотрудника – бухгалтера. Для выполнения своих рабочих задач у каждой имеется собственное устройство (ноутбук), а также они используют несколько программных средств, предназначенных для проектирования, моделирования и программирования:

Помимо использования вышеперечисленных программных средств, они также используют приложения для звонков и быстрого обмена информацией

Над реализацией проектов в организации работает 14 человек. Продолжительность рабочего дня у них зависит от запланированного объёма работы на выбранный период. Объём которой определяется исходя из возможностей и загруженности работников, финансирования. Указания о том, какие работы необходимо будет выполнить, работники получают от менеджера проекта в созданном для них чате в Telegram или приложении для отслеживания поставленных задач.

Выручка от реализации услуг (работ) по заказам на разработку программных продуктов от сторонних организаций (заказчиков) отражается на основании актов сдачи-приемки оказанных услуг, подписанных в конце месяца обеими сторонами. Услуги оказываются согласно заявкам на оказание услуг, составленных заказчиками, как правило, на 1 число каждого месяца

Для заказа услуг клиент оставляет заявку на сайте организации, заполняя форму. В заявку входят следующие его данные:

* имя;
* телефон;
* электронный адрес;
* сообщение (какие услуги клиент желает заказать).

Затем организация связывается с клиентом, договариваясь по поводу будущих работ.

Организация имеет двух главных специалистов, которые руководят разработкой проектов и их реализацией.

Основными задачами, которые выполняет организация, являются следующие виды деятельности:

* создание дизайна;
* создание документации к проекту;
* создание ПО для смартфонов на базе андроид и ios в том числе кроссплатформенных приложений;
* написание приложений с использвоанием технолгий Chromecast, AirPlay и других;
* тестирование;
* сопровождение программного обеспечения.

1. Должностные обязанности

Выполняемые задачи охватывали широкий спектр обязанностей, включая функции операторов ЭВМ, системных администраторов, графических дизайнеров, техников-программистов. Основные виды работ включали:

* поддержание функционирования программного обеспечения и физического оборудования организации, работа с системой управления контентом, на которой он работает;
* обучение у работника должностным обязанностям оператора ЭВМ;
* помощь работникам организации;
* выполнение индивидуального задания (разработка приложения);
* ­­техническое обслуживание и поддержка оборудования, используемого сотрудниками, а также специализированной техники для танцевальных занятий;
* участие в разработке цифровых продуктов, включая проектирование интерфейсов и создание графического контента для веб-приложений;
* выполнение специализированных задач, связанных с внедрением новых функций или доработкой существующего программного обеспечения для удовлетворения внутренних потребностей организации.

Такие задачи направлены на повышение эффективности работы компании, развитие цифровых продуктов и улучшение взаимодействия с клиентами.

1. Программное обеспечение, используемое на предприятии

На предприятии в основном используется следующее программное обеспечение Android Studio, Xcode, Git, Google Chrome, Figma

Android Studio – интегрированная среда разработки (IDE), созданная Google специально для разработки приложений на платформе Android. Программа используется для написания, тестирования, отладки и оптимизации Android-приложений. Android Studio поддерживает язык программирования Java, Kotlin и C++ и предоставляет такие инструменты, как эмулятор Android, систему анализа кода, редактор интерфейсов и средства тестирования.

Xcode – интегрированная среда разработки (IDE) от Apple, предназначенная для создания приложений и программного обеспечения для платформ iOS, macOS, watchOS и tvOS. Она включает инструменты для написания, тестирования, отладки и развертывания приложений, разработанных на Swift, Objective-C или C++.

Git – система контроля версий, которая используется для управления исходным кодом. Она помогает разработчикам отслеживать изменения, сохранять версии проекта и сотрудничать с другими участниками команды в режиме реального времени. Git не является инструментом для 3D-моделирования, а является важной частью процесса разработки программного обеспечения.

Google Chrome – браузер, разрабатываемый компанией Google на основе свободного браузера Chromium и движка Blink. В нём осуществляется поддержка работоспособности сайта организации.

Figma – это облачная платформа для проектирования интерфейсов, совместной работы над дизайном и создания интерактивных прототипов. Она предоставляет мощные инструменты для векторной графики, разработки пользовательских интерфейсов (UI) и взаимодействия (UX).

Реализация индивидуального задания

Исследование предметной области

Темой индивидуального задания является разработка мобильного приложения. Приложение будет посвящено спортивной тематике и предоставит пользователям доступ к актуальным новостям, видео и расписаниям матчей. Оно также будет включать функционал для просмотра статистики, результатов игр и персонализированных уведомлений о событиях любимых команд. Интуитивно понятный интерфейс обеспечит удобство использования, а современные технологии позволят приложению работать быстро и без сбоев.

Предметная область данной разработки связана с новостного приложения связана с управлением, анализом и предоставлением информации о спортивных событиях, командах, игроках и матчах. Приложение помогает пользователям следить за ходом соревнований, анализировать результаты и получать оперативные обновления.

Спорт является важной составляющей жизни современного общества. Большое количество людей занимается спортом или просто наблюдает за проведением спортивных мероприятий. Это создает потребность в быстром информировании пользователей о спортивных новостях. Также многие предпочитают мобильность в просмотре видео спортивного характера. В связи с этим проект фокусируется на предоставлении удобного инструмента для получения новостей и видео о спорте на мобильных устройствах.

Основным объектом в системе является матч. У каждого матча есть уникальный идентификатор, временные рамки, команды-участники и результаты. Команды обладают своими названиями, логотипами, составами игроков и статистикой. Игроки представляют собой членов команды, чья активность и действия непосредственно влияют на результаты матча.

Целевая аудитория приложения включает:

* спортивных болельщиков, активно следящих за играми, командами и соревнованиями;
* любителей аналитики и прогнозов, которые предпочитают изучать статистику и анализировать спортивные события;
* активных пользователей мобильных устройств, которым важно получать новости и видео в удобном формате, где бы они ни находились;
* молодежь и студентов, активно интересующихся спортом и предпочитающих мобильные приложения для удобства использования;
* профессиональных спортсменов и тренеров, которые могут использовать приложение для анализа матчей и просмотра актуальных событий.

Во время матча фиксируются события, такие как голы, замены или карточки. Эти события записываются с указанием времени, команды, игрока и типа события. После завершения матча данные используются для анализа и формирования статистики, например, рейтингов игроков, результатов команд или прогнозов на будущие соревнования.

Функционал приложения включает управление расписанием матчей, работу с профилями команд и игроков, обработку событий матча и предоставление аналитики. Пользователи могут изучать информацию о командах, смотреть лучшие моменты матчей, получать отчеты и прогнозы. Дополнительно приложение будет ориентировано на предоставление актуальных новостей и спортивных видеоматериалов, что удовлетворит потребность пользователей в мобильном доступе к информации.

В качестве целевых устройств выбрано направление мобильных устройств, так как на текущий момент это наиболее используемые и удобные устройства для просмотра информации. Таким образом, разработка приложения под Android позволит охватить широкую аудиторию пользователей и удовлетворить их запросы в спортивной тематике.

* 1. Проектирование модели

Для проектирования модели удобно использовать UML-диаграммы, в данном случае диаграмму вариантов использования, которая в наглядном виде отображает, какие варианты использования доступны клиенту, запустившему мобильное приложение.

Диаграмма прецедентов или диаграмма вариантов использования ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) Use case diagram) в UML – диаграмма, отражающая отношения между актёрами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне.

Прецедент – возможность моделируемой системы (часть её функциональности), благодаря которой пользователь может получить конкретный, измеримый и нужный ему результат. Прецедент соответствует отдельному сервису системы, определяет один из вариантов её использования и описывает типичный способ взаимодействия пользователя с системой. Варианты использования обычно применяются для спецификации внешних [требований](https://ru.wikipedia.org/wiki/%D0%A2%D1%80%D0%B5%D0%B1%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F_%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%BC%D1%83_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8E) к системе.

Основное назначение диаграммы — описание функциональности и поведения, позволяющее заказчику, конечному пользователю и разработчику совместно обсуждать проектируемую или существующую систему.

На рисунке 4.1 представлена диаграмма прецедентов, которая отображает способы взаимодействия пользователя с программой.

Мобильное приложение SportApp подразумевает различные способы взаимодействия, получение и отображение разного вида информации.

Основными способами использования приложения являются:

* просмотр последних спортивных новостей;
* просмотр трансляции;
* детальный анализ матча;
* просмотр турнирной таблицы.

Диаграмма use-case содержит в себе пользователя, который и имеет возможность использовать.

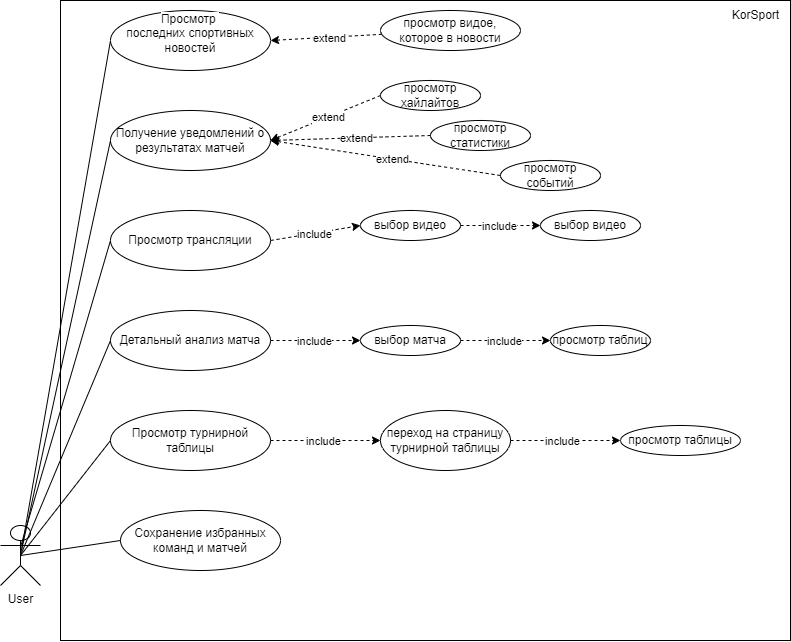


Рисунок 4.1 – Диаграмма вариантов использования

Организация данных

Структура организации данных

Организация данных очень важный процесс при создании системы такого типа. Ведь именно благодаря правильной структуре данные будут отображаться быстро и правильно. Важным условием является своевременность и актуальность полученных данных, для этого в разрабатываемой системе будет использоваться API, для получения необходимой и актуальной спортивной информации.

В приложении организована структура data классов, которые представляют классы, используемые для десериализации данных, которые приходят с внешнего API, в формате json.

В структуру входит папки, по которым и разделяются модельки для удобного использования. В структуре содержатся 2 основные папки: api, и domain.

Папка api содержит файлы, которые используются для получения данных. В структуру данной папки входят:

* папка entities;
* папка mappers;
* папка viewModels;
* файл SoccerRepository.

В папке entities хранятся data классы, которые представляют модели, приходящие из внешнего источника, далее модели разделяются по папкам на основании принадлежности их к разным сущностям. Структура папки entities, выглядит так:

* matches;
* matchReport;
* ranking.

В папке mappers хранятся файлы типа. kt, в которых описаны методы для взаимодействия с данными, которые приходят.

В папке viewModels «вью модели” – это сущности для сохранения состояний, которые «живут» от их создания до закрытия приложения.

Файл SoccerRepository является интерфейсом, который обеспечивает правильное и лаконичное передвижение данных между метода и функциями, данным принимает информацию, записанную в формате json и с при помощи различных инструментов преобразует данные в нужный формат.

JSON (JavaScript Object Notation) — это текстовый формат обмена данными, основанный на синтаксисе объектов JavaScript. Он используется для хранения и передачи структурированных данных между приложениями. JSON прост в чтении и написании как для человека, так и для машины.

* + 1. Структура json файла про матчи

Правильная организация данных в структуре json файла очень важна для быстрого доступа к необходимым полям объекта. Ниже приведен пример разбора файла, который хранит в себе информацию про определенный матч:

* competition\_id: Идентификатор турнира;
* season\_id: Идентификатор сезона;
* match\_id: Идентификатор матча;
* match\_unique\_code: Уникальный код;
* date: Дата матча;
* match\_day\_id: Идентификатор тура;
* match\_day\_name: Название тура;
* match\_day\_short\_name: Сокращение тура;
* date\_order: Дата проведения;
* team\_a\_id: ID команды A;
* team\_a\_name: Название команды;
* team\_a\_short\_name: Сокращённое название;
* team\_a\_acronym: Акроним команды;
* team\_b\_id: ID команды B;
* team\_b\_name: Название команды;
* team\_b\_short\_name: Сокращённое название;
* team\_b\_acronym: Акроним команды;
* goals\_team\_a: Голы команды A;
* goals\_team\_b: Голы команды B;
* stadium\_name: Название стадиона;
* stadium\_city: Город стадиона;
* match\_status: Статус матча;
* is\_abandoned: Матч отменён;
* is\_postponed: Матч перенесён;
* is\_suspended: Матч приостановлен;
* broadcaster: Транслятор матча;
* match\_start\_time: Время начала;
* match\_phase: Фаза матча;
* opta\_id: Opta идентификатор;
* is\_forfeit\_win: Техническая победа;
* minute: Текущая минута.

Таким образом эта структура, которая создается после сериализации объекта отображает все необходимые поля.

На рисунке 4.2 представлен пример содержимого json файла с расписанием матчей.

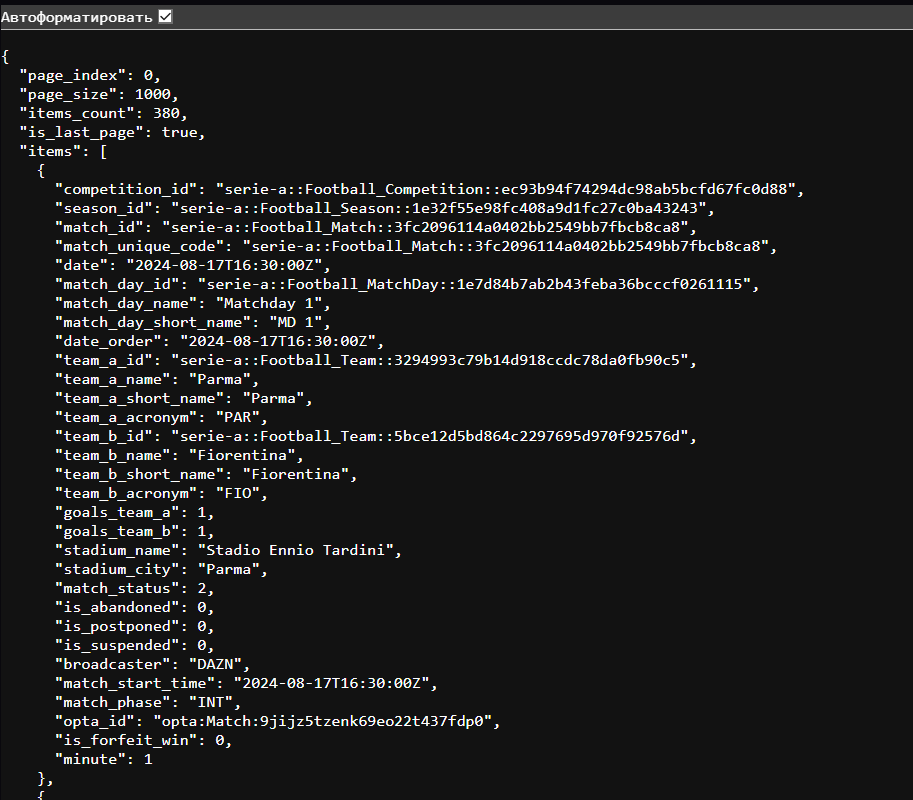


Рисунок 4.2

* + 1. Общение клиента и сервера

На диаграмме последовательности, рисунок 4.3 изображен процесс взаимодействия между четырьмя основными участниками передачи данных, а именно Парсер данных, Сервер, Клиентская часть приложения и Пользователь.

Парсер данных начинает свою работу, инициируя процесс парсинга данных с целевого веб-сайта. После завершения парсинга, парсер сохраняет результат в формате JSON-файла, который содержит структурированные данные с веб-ресурса.

Сервер запускает процесс обработки JSON-файла, полученного от парсера. В ходе обработки JSON-файла сервер извлекает данные и сохраняет их в базу данных.

Клиентская часть приложения инициирует запрос к серверу с целью получения последних новостей. Сервер обрабатывает запрос клиента, извлекая данные из базы данных. После обработки сервер отправляет последние новости в клиентскую часть приложения также в формате json.

Клиентская часть приложения получает данные от сервера и начинает их обработку для отображения. После обработки данных, клиентская часть формирует представление и отображает последние спортивные новости для пользователя. Пользователь видит обновлённые данные, которые были запрошены и обработаны системой.

На диаграмме также показаны действия пользователя: Пользователь взаимодействует с приложением, инициируя просмотр новостей. Приложение, в свою очередь, отправляет запрос к серверу для получения данных. Сервер передаёт данные приложению, которое их визуализирует для пользователя.

Процесс работы можно разделить на несколько этапов:

* этап сбора данных. Включает парсинг и сохранение в формате JSON;
* этап обработки данных на сервере. Включает извлечение, сохранение и предоставление данных по запросу;
* этап обработки на клиентской стороне. Включает получение, форматирование и отображение информации.

Диаграмма визуализирует синхронное взаимодействие между участниками системы через сообщения, отмеченные стрелками. Каждое сообщение соответствует определённой функции: запрос, передача данных или отображение. Диаграмма подчёркивает, что данные проходят три ключевых уровня обработки:

* парсер;
* сервер;
* клиент.

Взаимодействие завершается успешным отображением данных пользователю, обеспечивая удобный доступ к актуальным новостям.

Диаграмма демонстрирует строгую последовательность действий, начиная с автоматического парсинга данных и заканчивая отображением информации для конечного пользователя.

Парсер данных играет ключевую роль в извлечении свежей информации с веб-сайтов, формируя исходные данные для дальнейшей обработки.

JSON-файл служит стандартным форматом передачи данных между парсером и сервером, упрощая их обработку и интеграцию.

Сервер выполняет функцию центрального узла системы, обрабатывая данные, храня их в базе и передавая клиенту по запросу.

Взаимодействие между сервером и клиентской частью приложения организовано через запросы на получение последних новостей, что обеспечивает актуальность отображаемой информации.

Клиентская часть приложения отвечает за обработку данных, полученных от сервера, и их преобразование в удобный для пользователя интерфейс.

Пользователь взаимодействует только с клиентской частью приложения, не замечая сложных процессов обработки, происходящих на сервере и этапе парсинга.

Важным моментом является хранение данных в базе на сервере, что обеспечивает возможность повторного доступа без необходимости повторного парсинга.

Общая структура работы системы позволяет эффективно обновлять данные и предоставлять пользователю максимально актуальную информацию в кратчайшие сроки.

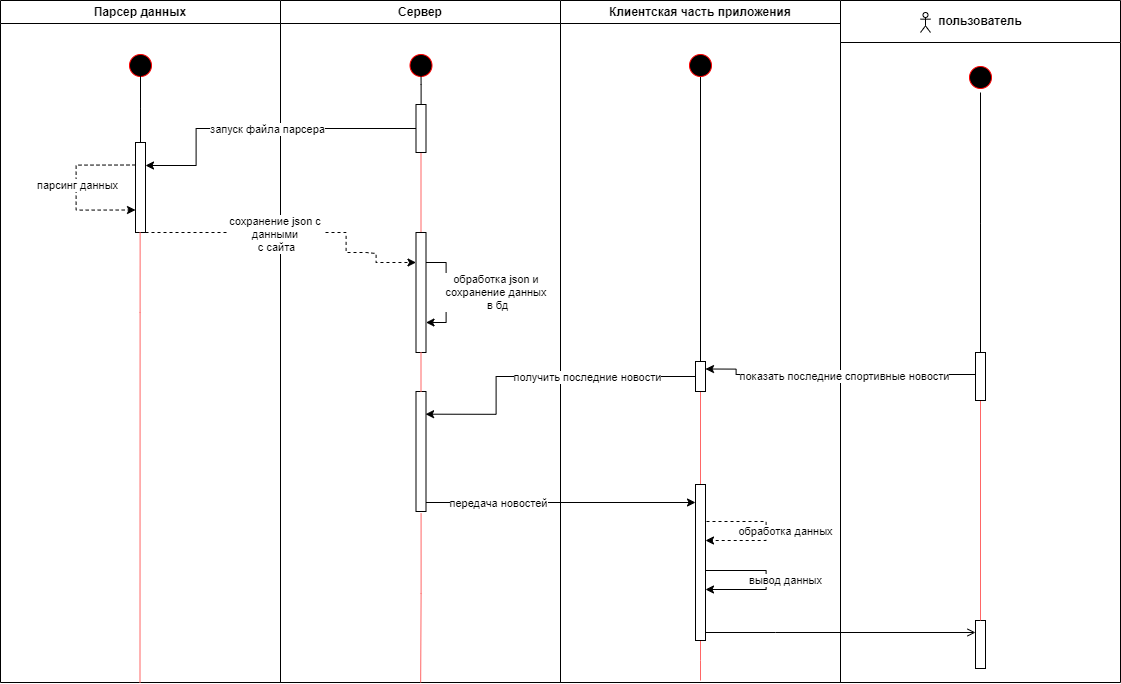


Рисунок 4.3 – Диаграмма последовательности

* 1. Концептуальный прототип

Концептуальный прототип представляет собой мобильное приложение, реализующее функционал, который предоставляет возможность пользователю, просмотра информации про спортивные новости, а также спортивную статистику в реальном времени.

Пользовательская часть состоит из множества экранов, каждый из которых имеет свое функциональное предназначение и отображает свою информацию.

Цветовая палитра имеет несколько цветов, а именно:

* белый;
* синий;
* черный.

В нижней части экрана мобильного устройства (каждого экрана приложения) располагается нижняя навигационная панель, которая предоставляет пользователю моментально переходить на выбранную страницу.

В нижнюю навигационную панель вынесены такие страницы как:

* главная;
* расписание матчей;
* страница с видео;
* профиль.

Каждая страница в навигационной панели имеет свою индивидуальную иконку, которая отображает ее смысловую нагрузку.

* + 1. Главная страница

На Главной странице акцентным объектом является блок с ближайшим/текущим/ближайшим прошедшим матчем. В данном блоке изображены логотипы играющих команд и акронимы (сокращенные названия), счет или время в зависимости от типа матча (ближайший или текущий/прошедший).

Ниже расположены два горизонтально скролящихся списка, один представляет из себя ленту новостей спорта разных стран и разного спорта, а второй видео моменты с трансляций.

На рисунке 4.4 представлен макет главной страницы.

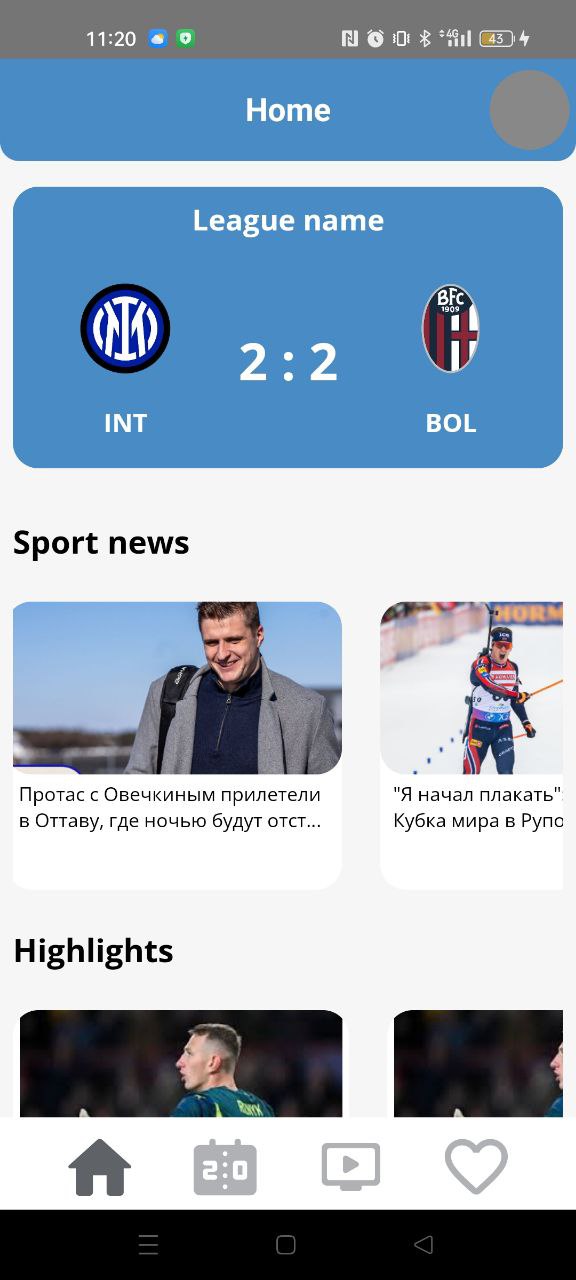


Рисунок 4.4 – Макет главной страницы

* + 1. Страница матчей

На странице матчей отображены два списка:

* горизонтальный список игровых дней;
* вертикальный список с расписанием матчей.

Игровой сезон состоит из 38 игровых серий по несколько дней подряд. Вертикальный состоит из карточек, в которых отображены логотипы играющих команд, времени начала матча или счета.

Каждая карточка может быть нажата для перехода на страницу информации про матч.

Макет страницы расписания матчей представлен на рисунке 4.5.

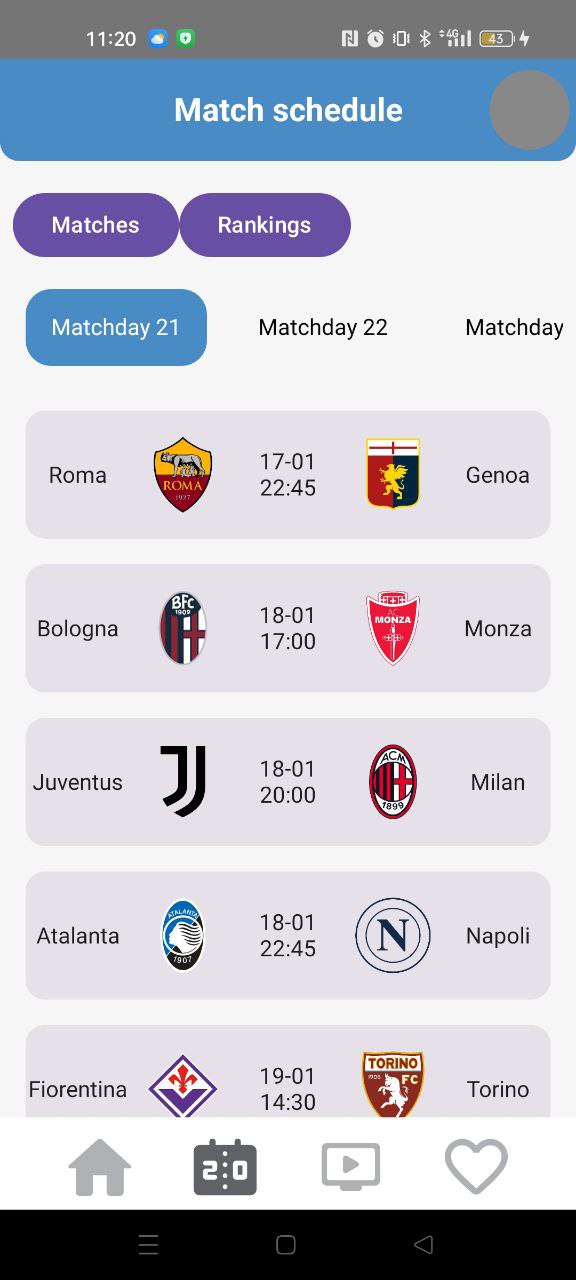


Рисунок 4.5 – Макет расписания матчей

* + 1. Страница matchInfo

Страница информации про матч представляет из себя дерево. В верхушке находится шапка, где представлена основная информации про команды, счет/дату и время начала, ниже идет ствол, в котором указано минуты матча, когда произошло игровое событие, справа и с слева от ствола указаны события. Каждая из сторон обозначает отношение игрового события к команде, с которой это произошло.

К игровым значимым событиям относятся:

* замена игрока;
* гол;
* желтая карточка;
* красная карточка;
* вторая желтая карточка;
* гол в свои ворота.

Макет страницы matchInfo представлен на рисунках 4.6 – 4.8.

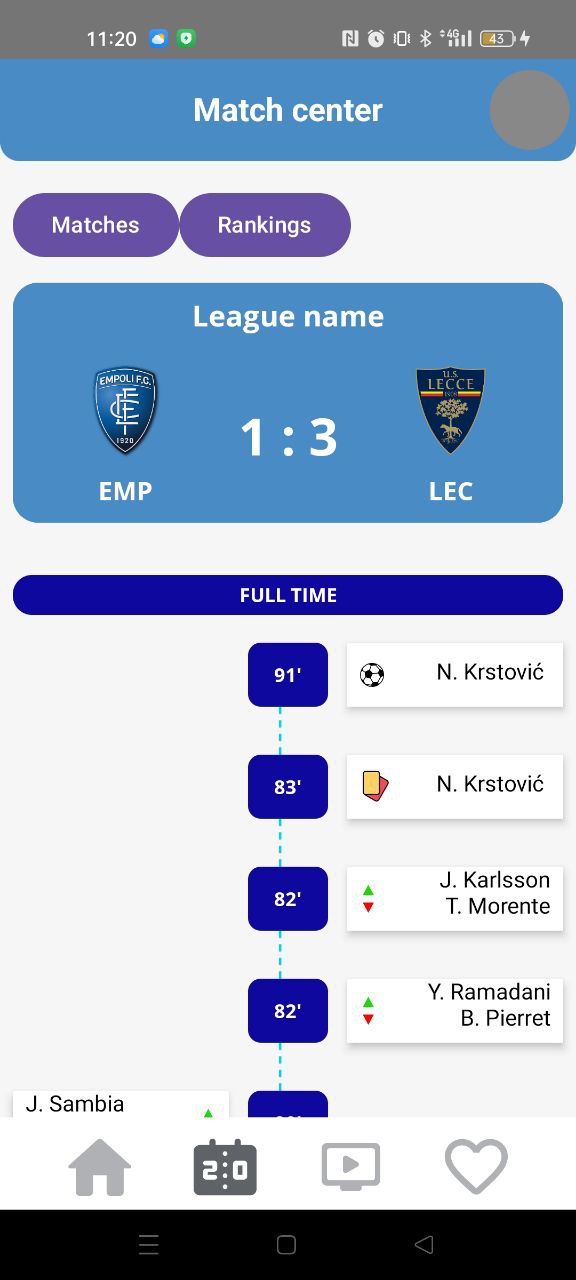


Рисунок 4.6 – Макет страницы матча (часть 1)

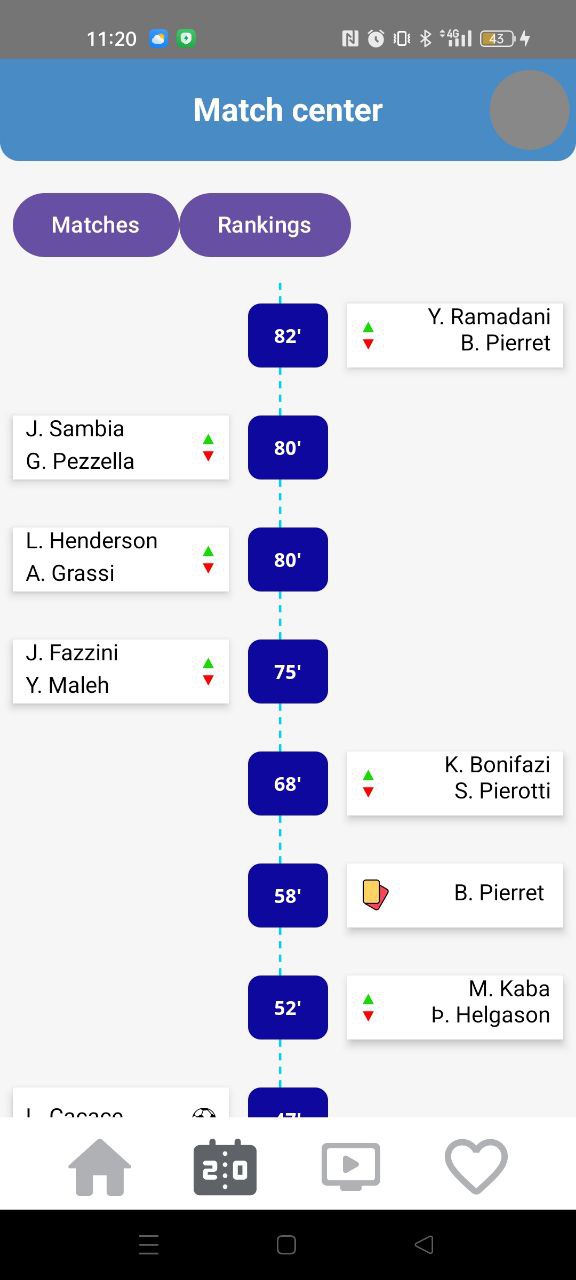


Рисунок 4.7 - Макет страницы матча (часть 2)

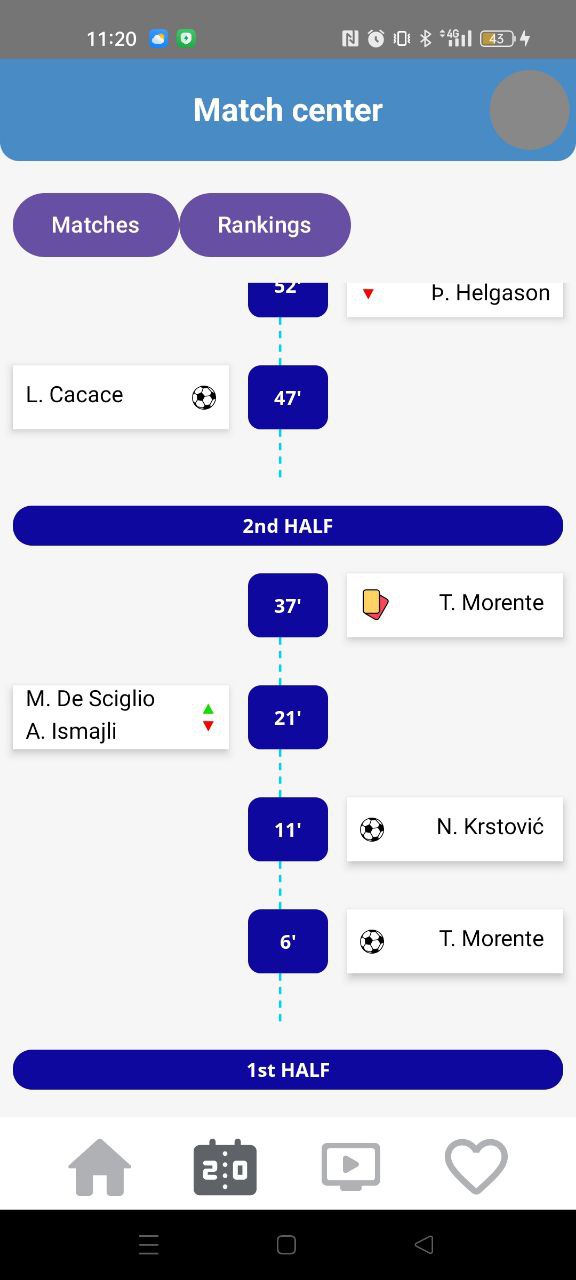


Рисунок 4.8 - Макет страницы матча (часть 3)

* + 1. Страница Rankings

На странице Rankings изображена таблица с командами, каждая карточка команды включает в себя такие данные как:

* номер команды в рейтинге;
* логотип команды;
* название команды/акроним;
* количество баллов, набранных командой;
* количество сыгранных игр;
* количество побед;
* количество игр, сыгранных в ничью;
* количество проигранных игр;
* разница между забитыми и пропущенными голами.

Также команды помечены разными цветными флажками, для наглядного понимания в какую лигу выходят команды, цвет флага обозначает:

* красный - команды выходит из турнира;
* зеленый - команда выходит в UEFA Conference League;
* оранжевый - команда выходит в UEFA Europa League;
* зеленый - команда выходит в UEFA Champions League.

Макет страницы Rankings представлен на рисунках 4.8 и 4.9.



Рисунок 4.8 - Макет страницы с таблицой лидеров (часть 1)



Рисунок 4.9 -Макет страницы с таблицой лидеров (часть 2)

* 1. Реализация функций

89

* + 1. Реализация раздела новостей

Раздел предоставляет пользователю актуальные спортивные новости, которые автоматически обновляются через сервер. Пользователь может читать статьи в полном объёме, просматривать изображения. Все новости сортируются по времени публикации, а в будущем также могут быть отфильтрованы по виду спорта. Лента новостей обновляется автоматически при пролистывании вправо. Для удобства доступна кнопка «Добавить в избранное».

* + 1. Реализация раздела расписания матчей

Этот раздел отображает расписание матчей с возможностью просмотра дате. Информация включает время начала, место проведения и ссылки на прямые трансляции. Реализована функция напоминания, которая уведомляет пользователя за 30 минут до начала матча. После завершения матчей в разделе появляются их результаты.

* + 1. Реализация раздела расписания матчей

Раздел предоставляет подробную статистику по турнирным таблицам, а также пользователь может просмотреть основные события матча, выбрав интересующий его матч. Данные получатся по средством запроса в формате json.

* 1. Функциональное тестирование

Тестирование работы мобильного приложения проводилась с использование устройства со следующими характеристиками:

* операционная система Android 13
* использование сетей LTE и Wi-Fi.
* оперативная память объёмом 6 ГБ;
* процессор Helio G95;
* экран 6.4 дюймов.

А также с использованием эмулятора, для проверки корректности

работы и совместимости с устройствами, которые имеют другие свойства.

Поскольку приложение про спортивные новости разрабатывается под платформу Android, его тестирование проводится на различных устройствах с этой операционной системой. Это позволяет убедиться в корректной работе всех функций приложения на разных версиях Android и устройствах с различными техническими характеристиками. В ходе тестирования особое внимание уделяется стабильности работы приложения, отображению информации в реальном времени, а также корректности отображения новостей и статистики. Тестирование включает проверку навигации между экранами, функциональности отображения матчей, работы с профилями команд и игроков. Кроме того, проводится проверка отображения иконок и правильности работы кнопок в нижней навигационной панели. Поскольку приложение ориентировано на оперативное предоставление спортивной информации, важной частью тестирования является проверка обновления данных о матчах и событиях. Все тест-кейсы выполняются последовательно, чтобы убедиться в отсутствии критических ошибок. На основе полученных данных производится доработка и оптимизация приложения для обеспечения лучшего пользовательского опыта.

Таблица 4.1 – Таблица описания тест-кейсов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предусловие: | | Запустить приложение | | |
| № | Описание | Шаги воспроизведения | Ожидаемый результат | Результат |
| 1 | Узнать на какой минуте и кем был забит гол | 1. Открыть страницу расписания матчей 2. Выбрать прошедший матч 3. Нажать на выбранный матч 4. Просмотреть дерево событий 5. Найти события «гол» 6. Посмотреть время на какой минуте забит гол | Список матчей и информация про матч успешно загружен и  отображен | Пройден. Результат представлен на рисунке  Б.1 приложения Б |
| 2 | Анализ таблицы лидеров и просмотр команд, которые вылетают | 1. Открыть страницу расписания матчей 2. Перейти на страницу таблицы лидеров 3. Проанализировать | Данные успешно загружены,  А 3 команды, которые набрали наименьшее количество очков помечены красным флагом | Пройден. Результат представлен на рисунке  Б.2 приложения Б |

Продолжение таблицы 4.1 (часть 1)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 1. таблицы 2. Пролистать вниз 3. Просмотреть последние команды, которые   проигрывают в сезоне |  |  |
| 3 | Аунтефикация пользователя | 1. Выйти из аккаунта, если до этого пользователь был зарегистрирован. 2. Ввести правильный email пользователя 3. Ввести правильный пароль пользователя 4. Нажать кнопки аунтефикации | Пользователь с таким email найден, а пароль правильный и пользователю даны права на использование приложения.  Осуществился переход на главную страницу | Пройден.  Пользователь перешел на главную страницу |
| 4 | Регистрация пользователя | Выйти из аккаунта, если до этого пользователь залогинился  На странице аунтефикации нажать на надпись «Don’t have account»  Ввести новый email пользователя,  Придумать и ввести пароль  Нажать кнопку регистрации | Пользователь успешно зарегистрирован  И его данные правильно занесены в базы данных | Пройден. Результат представлен на рисунке  Б.3  приложения Б |
| 5 | При запуске открывается экран с логотипом, слоганом и | Запуск приложения | При успешном запуске открывается сплеш-экран, логотип | Пройден. Результат представлен на рисунке |

Продолжение таблицы 4.1 (часть 2)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | логотипами партнеров |  | приложения и партнеров корректно отображаются, а слоган соответствует действительному | Б.4 приложения Бс |

Выводы

Во время выполнения технологической практики была изучена организационно-функциональная структура предприятия, особенности его деятельности и направления работы. В достаточном объёме исследована предметная область, а также программное обеспечение, используемое на предприятии. Были приобретены практические навыки в сопровождении и разработке программного обеспечения, что отражено в соответствующих разделах отчёта.

Индивидуальное задание было выполнено успешно. В процессе работы над заданием изучены методы парсинга данных с веб-сайтов, реализована работа с таблицами и данными в формате Microsoft Word. Было разработано мобильное приложение, которое реализует возможность следить за новостями спорта в реальном времени, а также просматривать различную статистику. Приложение, а также сервер были протестированы в различных условиях и показали стабильный результат при нормальных условиях. Полученные навыки и достигнутые результаты свидетельствуют о качественном освоении поставленных задач.

Список информационных источников

* 1. Базы данных. Проектирование и реализация / Р. Элмасри, Ш. Наватхе. – 6-е изд. – Москва: Вильямс, 2020. – 1232 с.
  2. Программирование на Kotlin для Android / П. Дж. Барнс. – Москва: ДМК Пресс, 2020. – 560 с.
  3. Руководство по Android Jetpack Compose / Дж. Гамильтон. – Санкт-Петербург: Питер, 2022. – 368 с.
  4. Базы данных. Учебник для вузов / В. Дейт. – 8-е изд. – Москва: Вильямс, 2021. – 1120 с.
  5. Основы разработки мобильных приложений для Android / А. Бреслав. – Москва: Бином, 2021. – 672 с.
  6. Kotlin. Полное руководство для разработчиков / М. Уайт. – Москва: БХВ-Петербург, 2021. – 720 с.
  7. Современные базы данных. Теория и практика / В. Филлипс. – Москва: Диалектика, 2019. – 928 с.
  8. Программирование на Kotlin. От основ до профессионала / Э. Сэдли, В. Черепанов. – Москва: Вильямс, 2021. – 576 с.
  9. Руководство по Android Jetpack Compose / [Электронный ресурс] – Режим доступа: <https://developer.android.com/jetpack/compose>. Дата доступа: 10.01.2025.
  10. Kotlin Documentation / [Электронный ресурс] – Режим доступа: https://kotlinlang.org/docs/home.html. Дата доступа: 15.01.2025.

# **Приложение А**

**(обязательное)**

# **Текст программных модулей**

package com.example.sportapp

import AppActivityViewModel

import BottomNavBar

import TopAppBar

import androidx.compose.foundation.layout.padding

import androidx.compose.material3.Scaffold

import androidx.compose.runtime.Composable

import androidx.compose.runtime.collectAsState

import androidx.compose.runtime.getValue

import androidx.compose.ui.Modifier

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.unit.dp

import androidx.navigation.NavHostController

import androidx.navigation.compose.NavHost

import androidx.navigation.compose.composable

import com.example.firebaseexample.pages.SignupPage

import com.example.sportapp.models.user.AuthViewModel

import com.example.sportapp.models.viewModels.MatchesActivitySoccerViewModel

import com.example.sportapp.models.viewModels.MatchesState

import com.example.sportapp.models.viewModels.NewsActivityViewModel

import com.example.sportapp.models.viewModels.NewsSate

import com.example.sportapp.models.viewModels.VideosState

import com.example.sportapp.models.viewModels.YoutubeActivityViewModel

import com.example.sportapp.pages.FirstPage

import com.example.sportapp.pages.HomePage

import com.example.sportapp.pages.LikePage

import com.example.sportapp.pages.LoginPage

import com.example.sportapp.pages.MatchesPage

import com.example.sportapp.pages.NewsPage

import com.example.sportapp.pages.VideoPage

import com.example.sportapp.shared.Loading

sealed class Screen(val route: String) {

data object Home : Screen("home")

data object Matches : Screen("matches")

data object Video : Screen("video")

data object Like : Screen("like")

data object News : Screen("news/{newsId}")

data object FirstPage : Screen("firstPage")

data object LoginPage : Screen("login")

data object SignupPage : Screen("signup")

data object Loading : Screen("loading")

}

@Composable

fun MyAppNavigation(

authViewModel: AuthViewModel,

newsState: NewsSate,

state: MatchesState,

videoState: VideosState,

newsViewModel: NewsActivityViewModel,

appActivity: AppActivityViewModel,

navController: NavHostController,

matchesViewModel: MatchesActivitySoccerViewModel,

videoViewModel: YoutubeActivityViewModel,

) {

val showBars by appActivity.showBars.collectAsState()

Scaffold(

containerColor = Color(0xFFF6F6F6),

bottomBar = {

if (showBars) {

BottomNavBar(navController = navController)

}

},

topBar = {

if (showBars) {

TopAppBar(appActivity, authViewModel, navController)

}

}

) { innerPadding ->

val paddings = if (showBars) 8.dp else 0.dp

NavHost(

navController = navController,

startDestination = Screen.FirstPage.route,

modifier = Modifier

.padding(innerPadding)

.padding(horizontal = paddings),

builder = {

composable(Screen.LoginPage.route) {

appActivity.changeShowBars(false)

LoginPage(navController, authViewModel)

}

composable(Screen.SignupPage.route) {

appActivity.changeShowBars(false)

SignupPage(navController, authViewModel)

}

composable(Screen.Loading.route) {

Loading()

}

composable(Screen.Home.route) {

appActivity.changeShowBars(true)

HomePage(

newsState = newsState,

state = state,

videoState = videoState,

newsViewModel = newsViewModel,

matchesViewModel = matchesViewModel,

videoViewModel = videoViewModel,

navController = navController,

authViewModel = authViewModel,

)

}

composable(Screen.FirstPage.route) {

FirstPage(navController)

}

composable(Screen.Matches.route) {

MatchesPage(matchesViewModel, state, appActivity)

}

composable(Screen.Video.route) { VideoPage(appActivity) }

composable(Screen.Like.route) { LikePage(appActivity) }

composable(Screen.News.route) { backStackEntry ->

val newsDateTime = backStackEntry.arguments?.getString("newsId")

NewsPage(

appActivity,

newsDateTime!!,

newsState,

navController,

newsViewModel

)

}

}

)

}

}package com.example.sportapp

import AppActivityViewModel

import android.annotation.SuppressLint

import android.os.Bundle

import androidx.activity.ComponentActivity

import androidx.activity.compose.setContent

import androidx.activity.viewModels

import androidx.compose.runtime.collectAsState

import androidx.compose.runtime.getValue

import androidx.lifecycle.viewmodel.compose.viewModel

import androidx.navigation.compose.rememberNavController

import com.example.sportapp.models.user.AuthViewModel

import com.example.sportapp.models.viewModels.MatchesActivitySoccerViewModel

import com.example.sportapp.models.viewModels.NewsActivityViewModel

import com.example.sportapp.models.viewModels.YoutubeActivityViewModel

class mainActivity : ComponentActivity() {

@SuppressLint("CoroutineCreationDuringComposition", "StateFlowValueCalledInComposition")

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContent {

val appActivity: AppActivityViewModel = viewModel()

val newsViewModel: NewsActivityViewModel = viewModel()

val newsState by newsViewModel.getState().collectAsState()

val matchesViewModel: MatchesActivitySoccerViewModel = viewModel()

val state by matchesViewModel.getState().collectAsState()

val videoViewModel: YoutubeActivityViewModel = viewModel()

val videoState by videoViewModel.getState().collectAsState()

val navController = rememberNavController()

val authViewModel: AuthViewModel by viewModels()

MyAppNavigation(

authViewModel = authViewModel,

newsState = newsState,

state = state,

videoState = videoState,

newsViewModel = newsViewModel,

appActivity = appActivity,

navController = navController,

matchesViewModel = matchesViewModel,

videoViewModel = videoViewModel

)

}

}

}

import androidx.compose.animation.core.animateDpAsState

import androidx.compose.animation.core.tween

import androidx.compose.foundation.background

import androidx.compose.foundation.layout.Box

import androidx.compose.foundation.layout.Column

import androidx.compose.foundation.layout.fillMaxWidth

import androidx.compose.foundation.layout.height

import androidx.compose.foundation.lazy.LazyColumn

import androidx.compose.foundation.lazy.rememberLazyListState

import androidx.compose.foundation.shape.RoundedCornerShape

import androidx.compose.runtime.Composable

import androidx.compose.runtime.LaunchedEffect

import androidx.compose.runtime.derivedStateOf

import androidx.compose.runtime.getValue

import androidx.compose.runtime.mutableStateOf

import androidx.compose.runtime.remember

import androidx.compose.runtime.setValue

import androidx.compose.ui.Modifier

import androidx.compose.ui.draw.clip

import androidx.compose.ui.graphics.Color

import androidx.compose.ui.unit.dp

import com.example.sportapp.models.soccer.api.domain.EventEntity

import com.example.sportapp.models.soccer.api.domain.MatchEntity

import com.example.sportapp.ui.theme.Blue100

import com.example.sportapp.widgets.home.CurrentMatch

import com.example.sportapp.widgets.matchInfo.MatchEventItem

import com.example.sportapp.widgets.matchInfo.PhaseNameLine

@Composable

fun MatchInfoContent(eventsList: List<EventEntity>, match: MatchEntity) {

val lazyListState = rememberLazyListState()

val isAtTop = remember {

derivedStateOf { lazyListState.firstVisibleItemIndex == 0 && lazyListState.firstVisibleItemScrollOffset == 0 }

}

var currentHeight by remember { mutableStateOf(200.dp) }

var currentSize by remember { mutableStateOf(100.dp) }

LaunchedEffect(isAtTop.value) {

currentHeight = if (isAtTop.value) 150.dp else 70.dp

currentSize = if (isAtTop.value) 100.dp else 10.dp

}

val boxHeight by animateDpAsState(

targetValue = currentHeight,

animationSpec = if (currentHeight == 70.dp) {

tween(durationMillis = 4000)

} else {

tween(durationMillis = 4000)

}

)

LazyColumn(state = lazyListState) {

item {

Box(

modifier = Modifier

.height(boxHeight)

.clip(shape = RoundedCornerShape(16.dp))

.background(color = Color.Transparent)

.fillMaxWidth()

) {

Column {

Box(

modifier = Modifier

.height(boxHeight)

.clip(shape = RoundedCornerShape(16.dp))

.background(color = Blue100)

.fillMaxWidth()

) {

CurrentMatch(match)

}

}

}

Box(

Modifier

.fillMaxWidth()

.height(15.dp)

)

}

items(eventsList.size) { index ->

if (index == eventsList.size - 1) PhaseNameLine("1st HALF")

else {

MatchEventItem(eventsList[index], match)

}

}

}

}

package com.example.sportapp.widgets.matchInfo.matchCard.specialEventsCards

import androidx.compose.runtime.Composable

import com.example.sportapp.models.soccer.api.domain.EventEntity

import com.example.sportapp.widgets.matchInfo.matchCard.specialEventsCards.playerSubstitulation.LeftSubstitulation

import com.example.sportapp.widgets.matchInfo.matchCard.specialEventsCards.playerSubstitulation.RightSubstitulation

@Composable

fun PlayerSubstitution(event: EventEntity, side: Boolean) {

if (side) {

LeftSubstitulation(event)

} else {

RightSubstitulation(event)

}

}package com.example.sportapp.widgets.matchInfo.matchCard.specialEventsCards

import androidx.compose.runtime.Composable

import androidx.compose.ui.unit.dp

import com.example.sportapp.R

import com.example.sportapp.models.soccer.api.domain.EventEntity

@Composable

fun GoalCard (event: EventEntity, side: Boolean) {

if (side) {

LeftLeaf(

event = event,

iconId = R.drawable.ball,

iconSize = 15.dp

)

}

else {

RightLeaf(

event = event,

iconId = R.drawable.ball,

iconSize = 15.dp

)

}

}package com.example.sportapp.widgets.matches.rankigs.rankingsParts

import androidx.compose.foundation.layout.Arrangement

import androidx.compose.foundation.layout.Row

import androidx.compose.foundation.layout.fillMaxWidth

import androidx.compose.material3.Text

import androidx.compose.runtime.Composable

import androidx.compose.ui.Modifier

import com.example.sportapp.models.soccer.api.domain.RankingEntity

@Composable

fun ScorePart (modifier: Modifier,item: RankingEntity) {

Row(

modifier = modifier

.fillMaxWidth(),

horizontalArrangement = Arrangement.SpaceEvenly

) {

Text(text = item.points.toString())

Text(text = item.gamesPlayed .toString())

Text(text = item.won.toString())

Text(text = item.draws.toString())

Text(text = item.lost.toString())

Text(text = (item.goalsMade - item.goalsConceeded).toString())

}

}

package com.example.sportapp.widgets.home.currentMatchFeatures

import androidx.compose.foundation.Image

import androidx.compose.foundation.layout.Arrangement

import androidx.compose.foundation.layout.Column

import androidx.compose.foundation.layout.Row

import androidx.compose.foundation.layout.fillMaxHeight

import androidx.compose.foundation.layout.fillMaxSize

import androidx.compose.foundation.layout.size

import androidx.compose.foundation.layout.width

import androidx.compose.material3.Text

import androidx.compose.runtime.Composable

import androidx.compose.ui.Alignment

import androidx.compose.ui.Modifier

import androidx.compose.ui.text.TextStyle

import androidx.compose.ui.text.style.TextAlign

import androidx.compose.ui.text.style.TextOverflow

import androidx.compose.ui.unit.dp

import coil.compose.rememberAsyncImagePainter

import com.example.sportapp.models.soccer.api.domain.MatchEntity

import com.example.sportapp.ui.theme.style12

import com.example.sportapp.ui.theme.style4

import com.example.sportapp.ui.theme.style5

import com.example.sportapp.ui.theme.style6

import java.time.ZonedDateTime

import java.time.format.DateTimeFormatter

@Composable

fun MiddleLine(

nearestMatch: MatchEntity

) {

val painterLogoA = rememberAsyncImagePainter(nearestMatch.logoUrlA)

val painterLogoB = rememberAsyncImagePainter(nearestMatch.logoUrlB)

val currentTime = ZonedDateTime.now()

val currentMatchCardText: String

val mainCardTextStyle: TextStyle

if (nearestMatch.localDateTimeMatchStart.isBefore(currentTime) || nearestMatch.matchStatus == 1){

currentMatchCardText = "${nearestMatch.goalsTeamA} : ${nearestMatch.goalsTeamB}"

mainCardTextStyle = style5

}

else {

val dateFormatter = DateTimeFormatter.ofPattern("dd MMMM")

val timeFormatter = DateTimeFormatter.ofPattern("HH:mm")

val formattedDate = nearestMatch.localDateTimeMatchStart.format(dateFormatter)

val formattedTime = nearestMatch.localDateTimeMatchStart.format(timeFormatter)

currentMatchCardText = "$formattedTime\n$formattedDate"

mainCardTextStyle = style6

}

Row(

modifier = Modifier.fillMaxSize(),

horizontalArrangement = Arrangement.SpaceEvenly,

verticalAlignment = Alignment.CenterVertically

) {

Column(

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.SpaceEvenly,

modifier = Modifier.fillMaxHeight()

) {

Image(

painter = painterLogoA,

contentDescription = "",

modifier = Modifier

.size(60.dp)

)

Text(

text = nearestMatch.teamAAcronym,

maxLines = 2,

overflow = TextOverflow.Ellipsis,

modifier = Modifier.width(100.dp),

textAlign = TextAlign.Center,

style = style4

)

}

if (nearestMatch.matchStatus == 1){

Column (

verticalArrangement = Arrangement.Center,

horizontalAlignment = Alignment.CenterHorizontally,

) {

Text(

text = currentMatchCardText,

style = mainCardTextStyle,

)

Text(

text = nearestMatch.minute.toString() + "'",

style = style12,

)

}

}

else {

Text(

text = currentMatchCardText,

modifier = Modifier.align(Alignment.CenterVertically),

style = mainCardTextStyle

)

}

Column(

horizontalAlignment = Alignment.CenterHorizontally,

verticalArrangement = Arrangement.SpaceEvenly,

modifier = Modifier.fillMaxHeight()

) {

Image(

painter = painterLogoB,

contentDescription = "",

modifier = Modifier

.size(60.dp),

)

Text(

text = nearestMatch.teamBAcronym,

maxLines = 2,

overflow = TextOverflow.Ellipsis,

textAlign = TextAlign.Center,

modifier = Modifier.width(100.dp),

style = style4

)

}

}

}

# **Приложение Б**

**(справочное)**

# **Результаты работы программы**

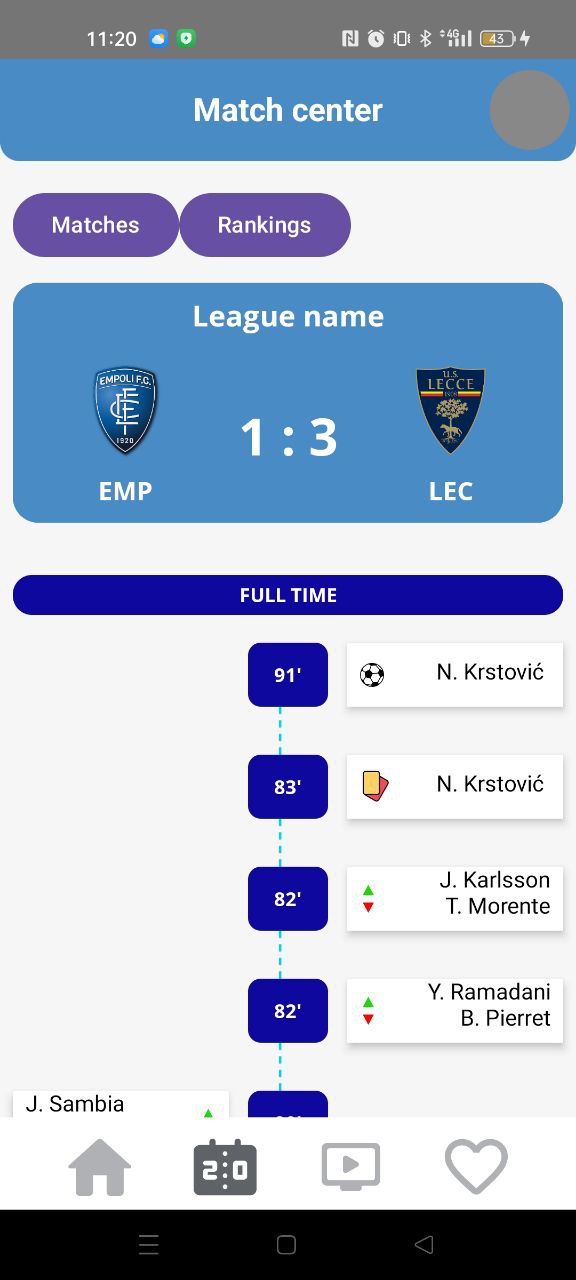


Рисунок Б.1 – Результат просмотра на какой минуте был забит гол



Рисунок Б.2 – Результат просмотра турнирной таблицы и 3 команды, которые вылетают

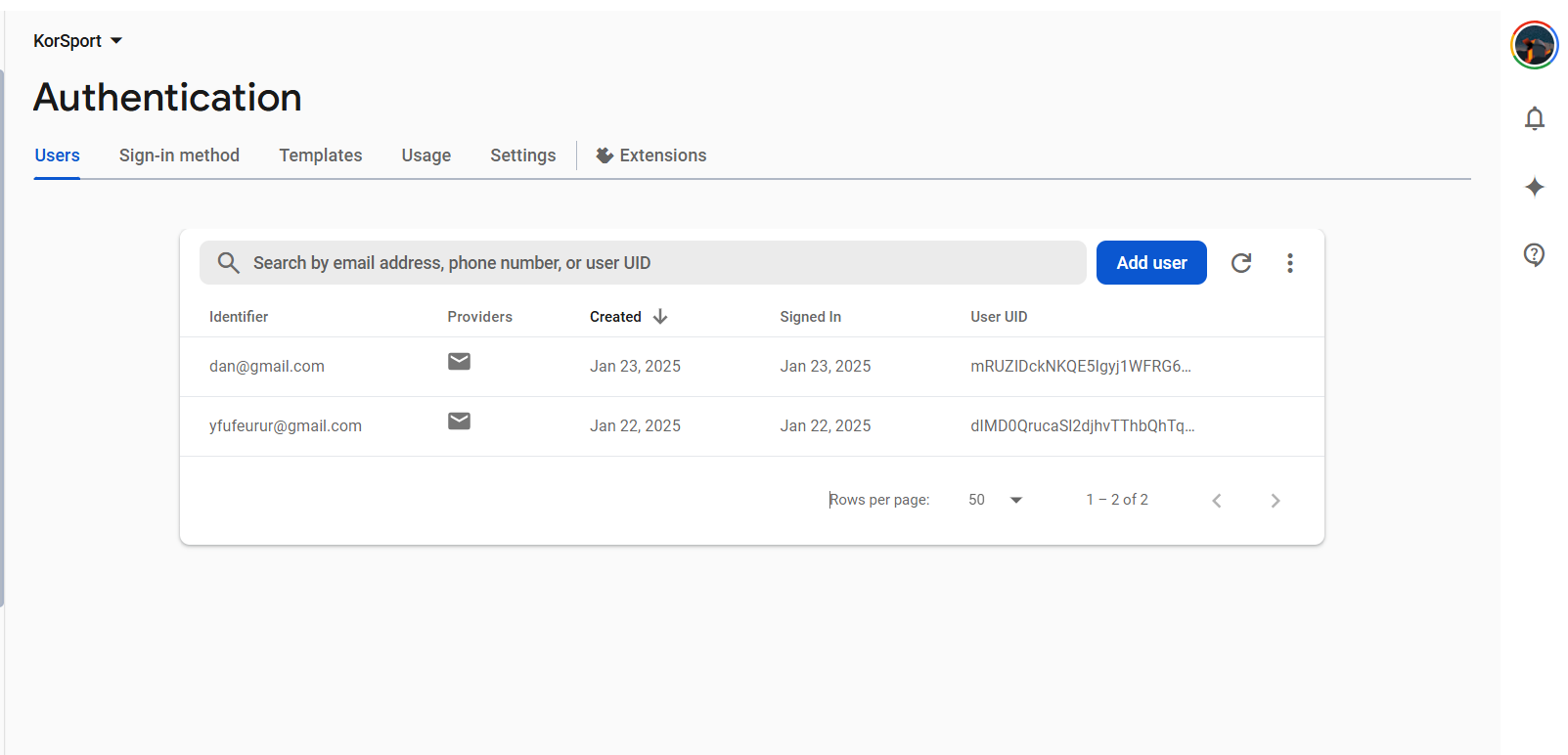


Рисунок Б.3 – Результат регистрации и пользователя



Рисунок Б.4 – Сплеш экран