Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Объектно-ориентированное программирование

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**СИСТЕМА ТРЕКИНГА РАБОЧЕГО ВРЕМЕНИ СОТРУДНИКОВ**

БГУИР КП 1-40 04 01

Студент А. В. Шпаковский

Руководитель Е. В. Тушинская

Нормоконтролер Е. В. Тушинская

Минск 2024

**CОДЕРЖАНИЕ**

[Введение 5](#_Toc168503922)

[1 Анализ предметной области 6](#_Toc168503923)

[1.1 Обзор аналогов 6](#_Toc168503924)

[1.2 Постановка задачи 10](#_Toc168503925)

[2. Проектирование программного средства 11](#_Toc168503926)

[2.1 Общая информация 11](#_Toc168503927)

[2.2 Разработка функциональности программного средства 11](#_Toc168503928)

[2.3 Архитектура программного средства 12](#_Toc168503929)

[2.4 Технологии программирования, используемые для решения поставленных задач 13](#_Toc168503930)

[3. Разработка программного средства 15](#_Toc168503931)

[3.1 Визуальное представление приложения 15](#_Toc168503932)

[3.2 Анализ необходимых средств для отображения графики 26](#_Toc168503933)

[3.3 Разработка программного кода 26](#_Toc168503934)

[4. Проверка работоспособности приложения 28](#_Toc168503935)

[5 Руководство пользователя 30](#_Toc168503936)

[Заключение 31](#_Toc168503937)

[Список использованных источников 32](#_Toc168503938)

[Приложение А (обязательное) Исходный код 33](#_Toc168503939)

[Приложение Б (обязательное) Схемы алгоритмов программного средства 37](#_Toc168503940)

# ВВЕДЕНИЕ

В современном деловом мире, где эффективность и продуктивность играют ключевую роль, важно иметь инструменты для оптимизации рабочего процесса и управления временем сотрудников. Одним из таких инструментов является трекинг рабочего времени. Это система, позволяющая отслеживать, анализировать и оптимизировать использование рабочего времени сотрудниками, что в конечном итоге способствует повышению общей эффективности компании.

В основе трекинга рабочего времени лежит идея систематического сбора данных о том, как сотрудники распределяют свое время в течение рабочего дня. Это может включать в себя учет времени, затраченного на выполнение конкретных задач, проектов, встреч, а также анализ перерывов и простоев. Такая информация позволяет руководителям и сотрудникам получить детальное представление о рабочем процессе, выявить узкие места, оптимизировать распределение задач и повысить общую эффективность работы.

Мобильные приложения для трекинга рабочего времени стали незаменимым инструментом для современных компаний. Они предоставляют удобный и эффективный способ сбора данных о рабочем времени сотрудников в режиме реального времени. Такие приложения предлагают широкий спектр функций, включая учет рабочего времени, создание отчетов, анализ продуктивности, интеграцию с другими бизнес-инструментами и многое другое.

Преимущества использования мобильных приложений для трекинга рабочего времени очевидны. Во-первых, они обеспечивают точность и объективность данных, устраняя необходимость ручного учета времени. Во-вторых, они позволяют сотрудникам самостоятельно контролировать свое рабочее время и повышать свою продуктивность. В-третьих, они предоставляют руководителям ценную информацию для принятия обоснованных решений по оптимизации рабочего процесса, распределению ресурсов и повышению общей эффективности компании.

Мобильное приложение «Attendify», разработанное в рамках данного курсового проекта, представляет собой удобное средство для трекинга рабочего времени сотрудников. Приложение позволяет сотрудникам легко регистрировать свое рабочее время и отслеживать запросы на отпуск, а руководителям – получать данные для анализа и оптимизации рабочего процесса.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

* 1. **Обзор аналогов**

На рынке приложений для трекинга рабочего времени FactoTime выделяется как комплексное решение, разработанное специально для малого и среднего бизнеса. Оно предлагает широкий спектр функций, объединяя в себе возможности учета рабочего времени, управления персоналом и расчета заработной платы.

FactoTime позволяет сотрудникам отмечать свой приход и уход с помощью мобильного устройства, фиксировать перерывы и отслеживать время, затраченное на выполнение конкретных задач. Геолокация помогает контролировать местоположение сотрудников, что особенно полезно для компаний с выездными работниками.

Приложение также упрощает управление отпусками и отсутствием, позволяя сотрудникам отправлять запросы и руководителям – утверждать или отклонять их. Автоматический расчет заработной платы на основе отработанного времени и установленных ставок экономит время и снижает риск ошибок.

FactoTime предоставляет подробные отчеты о рабочем времени, производительности и затратах на оплату труда, помогая руководителям принимать обоснованные решения. Гибкая система настроек позволяет адаптировать приложение под особенности конкретной компании.

Однако, несмотря на свои преимущества, FactoTime может показаться излишне сложным для небольших команд или индивидуальных предпринимателей. Некоторые пользователи отмечают, что интерфейс приложения не всегда интуитивно понятен, а некоторые функции могут быть избыточными для простых сценариев использования.

В целом, FactoTime – это мощный инструмент для комплексного управления рабочим временем и персоналом, который может быть особенно полезен для компаний, нуждающихся в расширенной функциональности и детальной аналитике.

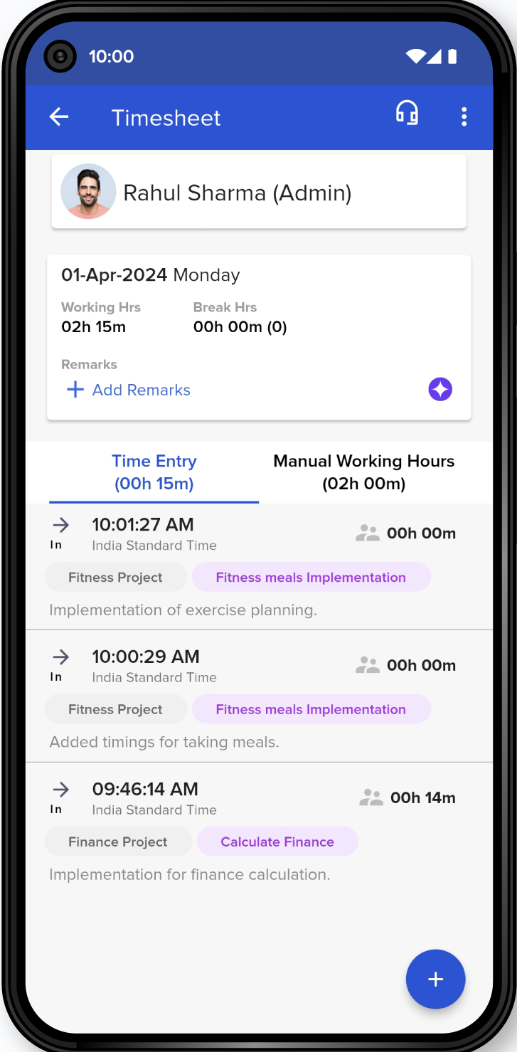


Рисунок 1.1 – Табель рабочего времени «FactoTime»

В то время как FactoTime предлагает широкий спектр возможностей, другие приложения фокусируются на более узких, но не менее важных аспектах трекинга рабочего времени. Time Doctor, например, выделяется своими расширенными функциями мониторинга активности сотрудников. Он позволяет не только отслеживать время, затраченное на задачи, но и делать периодические скриншоты экрана, отслеживать посещаемые веб-сайты и используемые приложения. Это дает руководителям более детальное представление о том, как сотрудники используют свое рабочее время, и помогает выявить потенциальные проблемы с продуктивностью.

Time Doctor также предлагает функции управления проектами, позволяя создавать задачи, назначать их сотрудникам и отслеживать прогресс. Подробные отчеты о рабочем времени, производительности и использовании приложений помогают руководителям принимать обоснованные решения по оптимизации рабочего процесса. Интеграция с популярными инструментами, такими как Asana, Trello, Slack и Jira, упрощает управление проектами и коммуникацию в команде.

Однако, Time Doctor может не подойти для компаний, которые ценят конфиденциальность сотрудников превыше всего. Функции мониторинга активности, такие как скриншоты экрана, могут вызывать дискомфорт у некоторых сотрудников и создавать ощущение недоверия. Кроме того, Time Doctor является платным приложением, и его стоимость может быть значительной для небольших компаний.

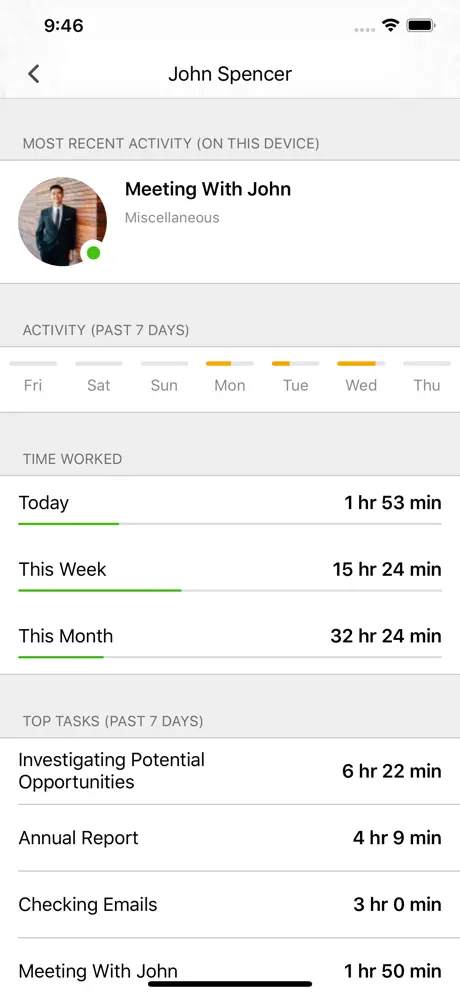


Рисунок 1.2 ­– Страница активности «TimeDoctor»

Harvest – это популярное приложение для учета рабочего времени, которое отличается простотой использования и гибкостью настроек. Оно позволяет сотрудникам легко отслеживать время, затраченное на проекты и задачи, используя таймер или вводя данные вручную. Harvest также предоставляет удобные функции для создания отчетов о проделанной работе, что упрощает процесс выставления счетов клиентам и управления бюджетами проектов.

Одним из ключевых преимуществ Harvest является его интеграция с популярными инструментами управления проектами, такими как Asana, Trello и Basecamp. Это позволяет автоматически импортировать задачи и проекты в Harvest, что экономит время и упрощает процесс отслеживания времени.

Harvest также предоставляет руководителям мощные инструменты для анализа данных о рабочем времени. Они могут просматривать отчеты о затраченном времени по проектам, задачам, сотрудникам и датам, а также сравнивать запланированное и фактическое время выполнения задач. Это помогает выявлять узкие места в рабочем процессе, оптимизировать использование ресурсов и принимать обоснованные решения по управлению проектами.

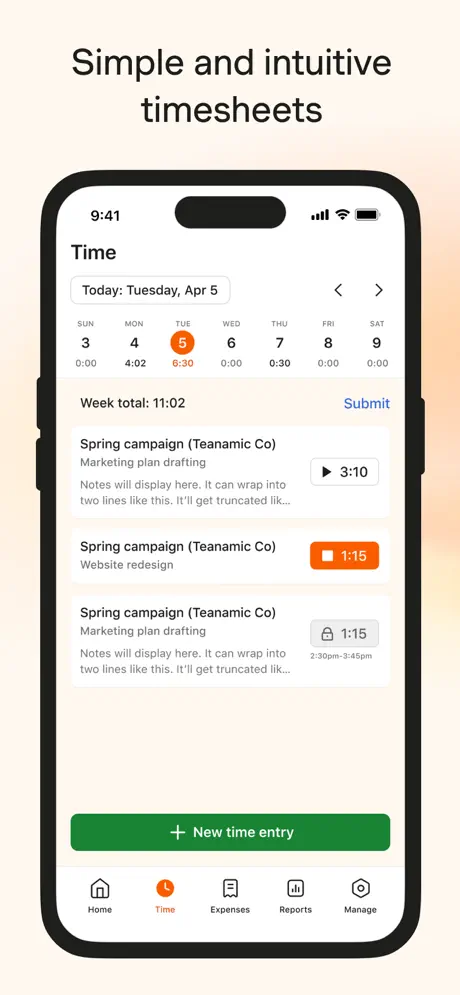


Рисунок 1.3 – Экран отслеживания времени «Harvest»

Проанализировав аналоги, можно заметить, что многие приложения для учета рабочего времени предлагают широкий спектр функций, от простого отслеживания времени до комплексного управления персоналом и расчета заработной платы. Некоторые, как FactoTime, ориентированы на малый и средний бизнес, предоставляя расширенную функциональность и детальную аналитику. Другие, как Time Doctor, делают акцент на мониторинге активности сотрудников, предлагая функции отслеживания посещаемых веб-сайтов, используемых приложений и даже скриншоты экрана. Harvest же выделяется своей простотой и гибкостью, предлагая удобные инструменты для отслеживания времени и интеграцию с популярными инструментами управления проектами.

В отличие от аналогов, «Attendify» делает акцент на удобстве и простоте использования для сотрудников и руководителей. Приложение позволяет сотрудникам легко регистрировать свой приход и уход, отслеживать историю посещаемости, создавать запросы на отпуск и просматривать их статус. Руководители же получают доступ к данным о рабочем времени сотрудников, что помогает принимать обоснованные решения по оптимизации рабочего процесса и распределению ресурсов.

## 

## 1.2 Постановка задачи

В рамках данного курсового проекта была поставлена задача: разработать мобильное приложение «Attendify».

Проанализировав конкретно поставленную задачу, был выделен следующий ряд задач:

* реализовать создание аккаунтов сотрудников администратором;
* реализовать возможность как выхода пользователя из учетной записи, так и полного ее удаления;
* реализовать отслеживание и отображение учета рабочего времени;
* реализовать возможность создания запросов сотрудниками и рассмотрения данных запросов администратором.

Разработав данный набор задач, можно перейти непосредственно к проектированию программного средства. Этот этап включает в себя определение архитектуры приложения, выбор технологий и инструментов разработки, а также создание детального плана реализации функциональности. Особое внимание будет уделено проектированию пользовательского интерфейса, который должен быть интуитивно понятным и удобным для пользователей. Кроме того, на этом этапе будут разработаны прототипы основных экранов приложения, что позволит визуализировать его будущий облик и внести необходимые корректировки еще до начала программирования.

# 2. ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

## 2.1 Общая информация

Для разработки данного приложения было выбрано сочетание React Native и Visual Studio Code. React Native, фреймворк от Facebook, позволяет создавать нативные мобильные приложения с использованием JavaScript и React, что обеспечивает высокую производительность и кроссплатформенность. Visual Studio Code, мощный и гибкий редактор кода от Microsoft, стал популярным инструментом среди разработчиков благодаря своей расширяемости и удобству использования.

Visual Studio Code предлагает обширную экосистему расширений, позволяющую настроить среду разработки под конкретные потребности. Для React Native доступны расширения, обеспечивающие подсветку синтаксиса, автодополнение кода, отладку приложений, интеграцию с эмуляторами и реальными устройствами, а также множество других полезных функций. Это делает Visual Studio Code идеальным инструментом для разработки React Native приложений, обеспечивая высокую продуктивность и комфорт в процессе работы.

React Native, основанный на React, позволяет использовать декларативный подход к построению пользовательского интерфейса, что упрощает разработку и поддержку приложений. Кроме того, React Native предоставляет доступ к нативным компонентам платформы, что позволяет создавать приложения, выглядящие и функционирующие как нативные. Благодаря этому, React Native стал популярным выбором для разработки кроссплатформенных мобильных приложений.

React Native продолжает активно развиваться, привнося новые возможности и улучшения. В последних обновлениях было уделено внимание производительности, добавлена поддержка новых платформ, улучшена интеграция с нативными компонентами и многое другое. Фреймворк поддерживается большим сообществом разработчиков и активно используется в крупных компаниях, что говорит о его надежности и перспективности.

## **2.2** **Разработка функциональности программного средства**

Проанализировав требования к проектируемой программной системе, были выделены возможности пользователя, представленные в виде диаграмм прецедентов на рисунке 2.1 и рисунке 2.2.

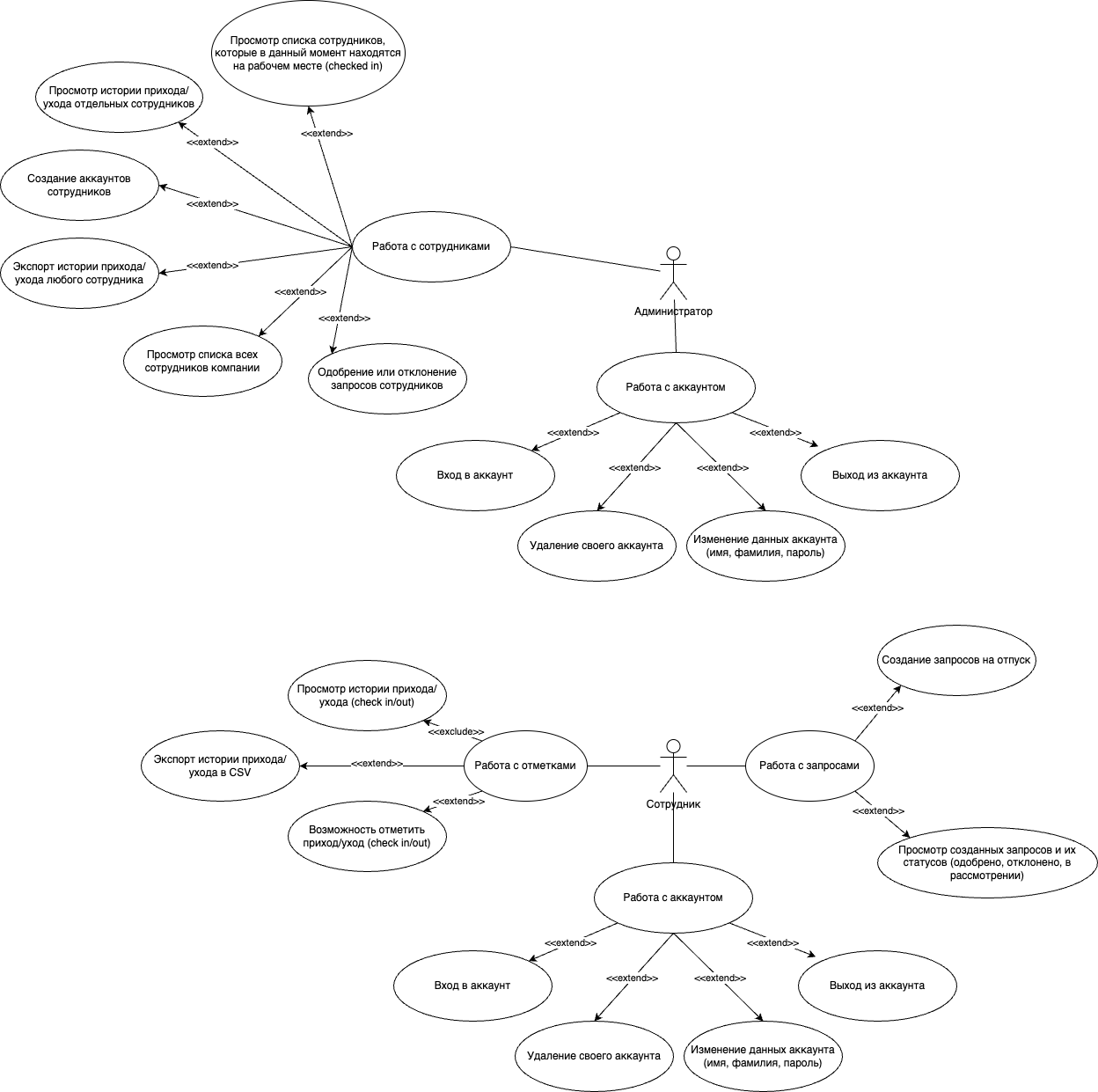


Рисунок 2.1 – Диаграмма вариантов использования

На основе представленной выше диаграммы, можно перейти, непосредственно, к определению архитектуры программного средства.

## 2.3 Архитектура программного средства

В основе архитектуры приложения лежит чистая архитектура, которая обеспечивает четкое разделение ответственности между различными слоями и способствует созданию независимых, легко тестируемых компонентов. Каждый функциональный модуль, такой как учет рабочего времени, управление запросами на отпуск или управление пользователями, рассматривается как отдельный компонент, состоящий из трех основных слоев.

Слой представления отвечает за отображение пользовательского интерфейса и взаимодействие с пользователем. Здесь находятся экраны приложения, компоненты UI и логика, связанная с отображением данных и обработкой пользовательского ввода. Слой бизнес-логики содержит основную логику приложения, реализующую бизнес-правила и операции. Слой данных отвечает за взаимодействие с источниками данных, такими как базы данных, API или локальное хранилище. В этом слое находятся репозитории, которые предоставляют методы для доступа к данным и абстрагируют детали реализации конкретного источника данных.

Такое разделение на слои позволяет достичь нескольких важных преимуществ. Во-первых, каждый слой может быть изменен или заменен без влияния на другие слои. Например, можно изменить способ хранения данных (например, перейти с Firebase на другую базу данных), не затрагивая бизнес-логику и пользовательский интерфейс. Во-вторых, каждый слой можно тестировать независимо от других слоев, что упрощает процесс тестирования и обеспечивает более высокое качество кода. В-третьих, приложение легко расширяется за счет добавления новых функциональных модулей, которые взаимодействуют с существующими модулями через четко определенные интерфейсы.

Использование чистой архитектуры позволяет создать гибкое, масштабируемое и легко поддерживаемое приложение, которое соответствует современным стандартам разработки программного обеспечения.

## 2.4 Технологии программирования, используемые для решения поставленных задач

Для приложения «Attendify» была выбрана клиент-серверная архитектура, которая обеспечивает четкое разделение между интерфейсом пользователя и логикой обработки данных. В качестве серверной части используется Firebase, мощная облачная платформа от Google, предоставляющая широкий спектр инструментов и сервисов для разработки мобильных и веб-приложений.

Firebase Authentication, компонент Firebase, отвечает за безопасную аутентификацию пользователей в «Attendify». Он позволяет реализовать вход с использованием электронной почты и пароля, а также предоставляет возможность интеграции с другими популярными провайдерами аутентификации, такими как Google и Facebook. Это обеспечивает удобство и безопасность для пользователей, позволяя им выбирать наиболее предпочтительный способ входа.

Для хранения данных о пользователях, их посещаемости и запросах на отпуск используется Firestore, NoSQL база данных, также входящая в состав Firebase. Firestore обладает высокой масштабируемостью и гибкостью, что позволяет легко адаптировать приложение к растущим потребностям бизнеса. Кроме того, Firestore поддерживает работу в реальном времени, благодаря чему изменения данных мгновенно синхронизируются на всех подключенных устройствах. Это особенно важно для приложения учета рабочего времени, где актуальность данных имеет решающее значение.

Клиентская часть приложения реализована с использованием React Native, кроссплатформенного фреймворка от Meta (Facebook). React Native позволяет создавать нативные мобильные приложения с использованием JavaScript и React, что значительно ускоряет процесс разработки и позволяет использовать единый код для Android и iOS. Это не только экономит время и ресурсы, но и обеспечивает согласованность пользовательского интерфейса на разных платформах.

Выбор Firebase и React Native для разработки «Attendify» обусловлен их высокой надежностью, масштабируемостью и удобством интеграции. Firebase предоставляет готовую инфраструктуру для хранения и управления данными, а также мощные инструменты для работы в реальном времени, что значительно упрощает разработку и поддержку приложения. React Native, в свою очередь, позволяет создавать высокопроизводительные и удобные мобильные приложения с использованием знакомых веб-технологий.

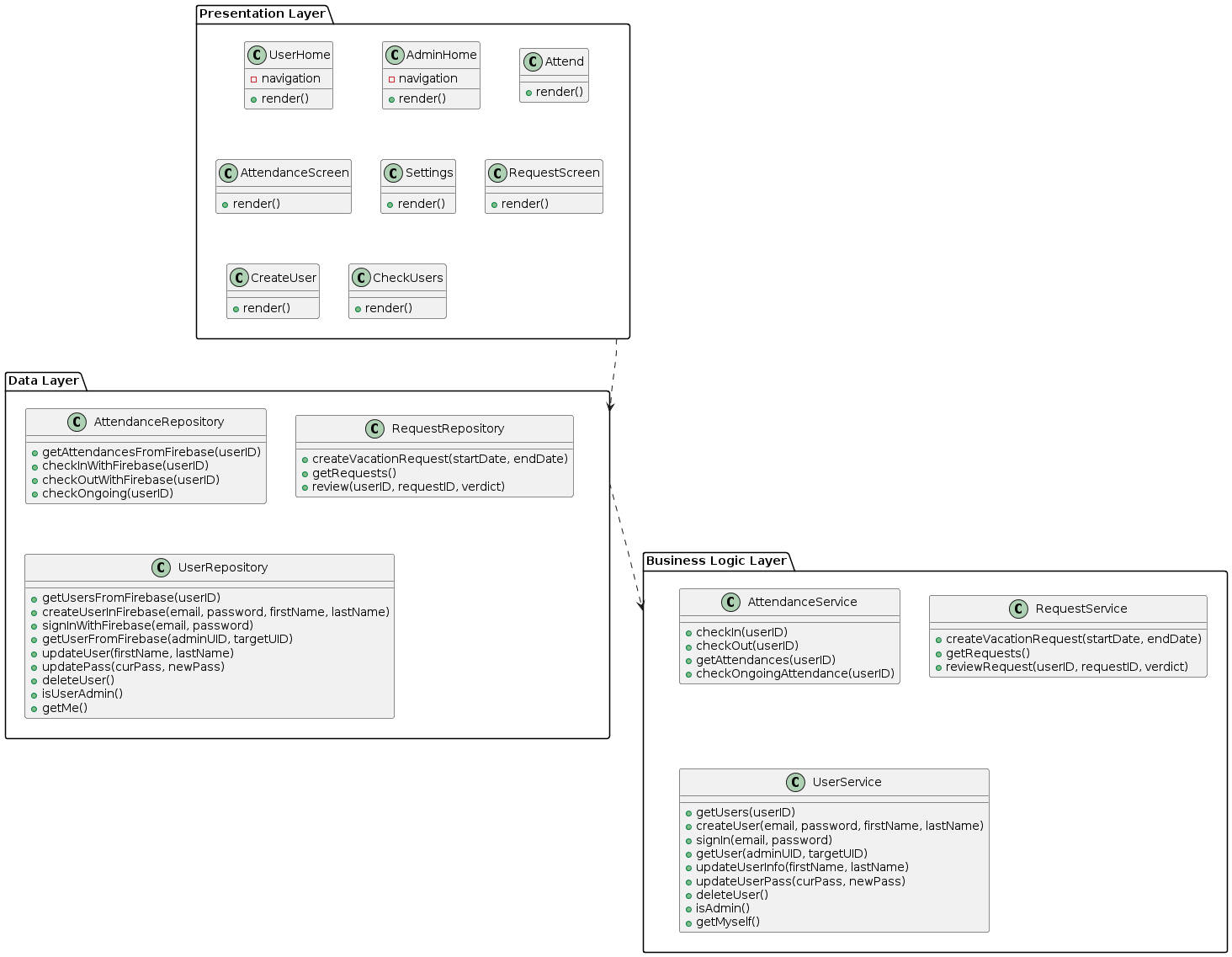


Рисунок 2.3 – Диаграмма классов приложения

# 3. РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА

## 3.1 Визуальное представление приложения

Разработка интуитивно понятного интерфейса – это ключевая задача, которая обеспечивает удобство использования приложения и позволяет пользователям быстро освоиться без необходимости обращаться к инструкциям. Достичь этого можно с помощью различных подходов, таких как использование знакомых иконок, интуитивной навигации и продуманной организации элементов на экране.

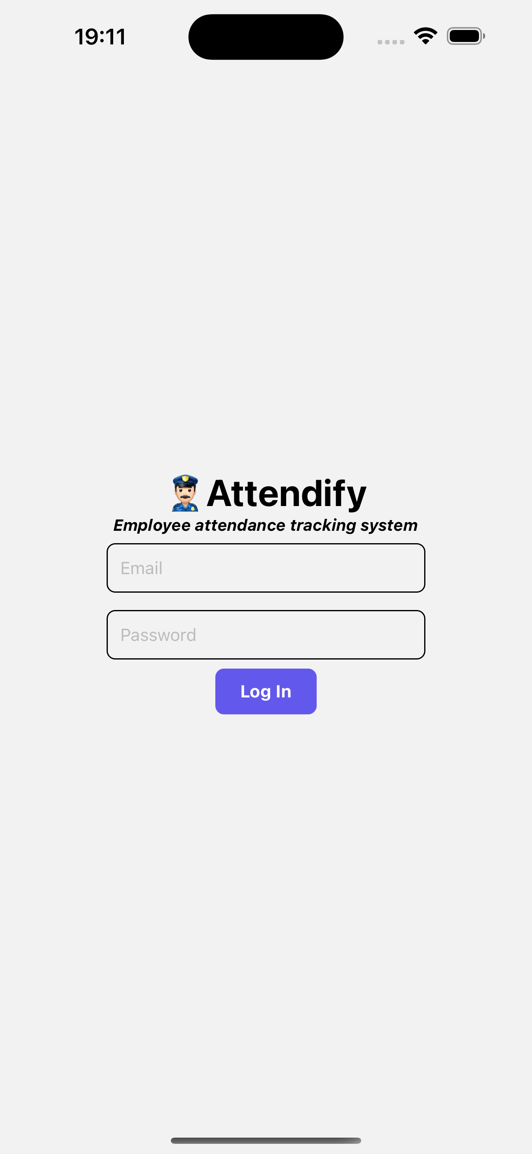


Рисунок 3.1 – Экран авторизации

Экран входа в систему, представленный на рисунке 3.1, встречает пользователя чистым и интуитивно понятным дизайном, который сразу фокусирует внимание на основных элементах, необходимых для авторизации. В середине экрана расположены два текстовых поля: одно для ввода адреса электронной почты, а другое – для ввода пароля. Эти поля четко обозначены и предоставляют достаточно места для удобного ввода данных. Ниже находится кнопка «Log In», выполненная в фирменных цветах приложения. При нажатии на эту кнопку, «Attendify» проверяет введенные данные и, в случае успеха, предоставляет доступ к функциональности приложения. Однако, если пользователь ввел неверный адрес электронной почты или пароль, приложение выводит сообщение об ошибке, предлагая повторить попытку.

Стоит учесть, что «Attendify» не предусматривает самостоятельной регистрации пользователей. Вместо этого, создание аккаунтов для сотрудников осуществляется администратором непосредственно через приложение, что обеспечивает централизованный контроль и управление учетными записями. Создание аккаунта администратора, в свою очередь, требует ручного вмешательства через Firebase, облачную платформу, на которой базируется приложение.

После успешного входа, пользователь попадает на один из двух экранов, в зависимости от своей роли в системе. Сотрудники перенаправляются на главный экран, где им предоставляется доступ к основным функциям приложения, таким как регистрация прихода и ухода, просмотр истории посещаемости, создание запросов на отпуск и управление настройками аккаунта. Для удобной навигации между этими функциями используется панель для навигации, расположенная в нижней части экрана.

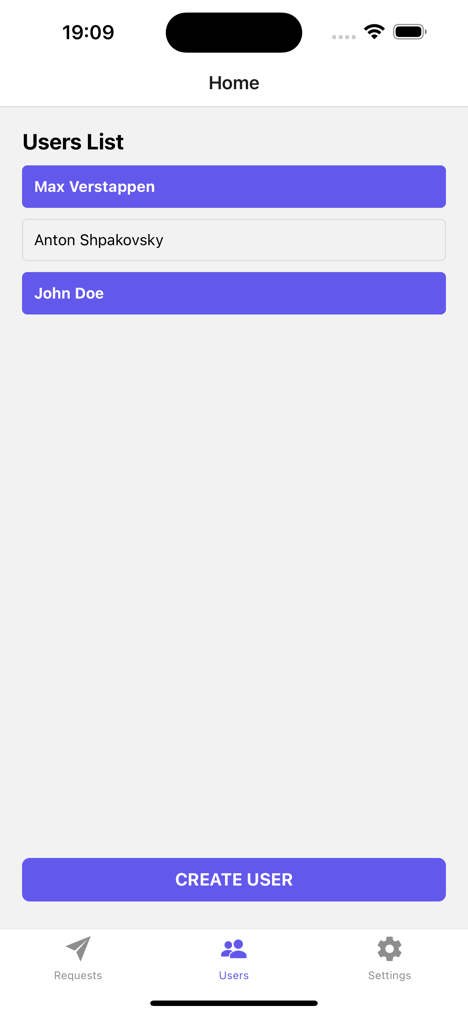


Рисунок 3.2 – Домашний экран администратора

При успешном входе, администраторы же попадают на панель управления, изображенная на рисунке 3.2, которая предоставляет им обзор всех сотрудников. Здесь администратор может видеть кто в данный момент отметился и работает по характерной обводке имени сотрудника в фирменном цвете, просматривать и экспортировать историю посещаемости каждого сотрудника, создавать новые учетные записи для сотрудников, а также обрабатывать запросы на отпуск. Как и в случае с сотрудниками, для навигации по панели администратора используется панель для навигации, расположенная внизу экрана.

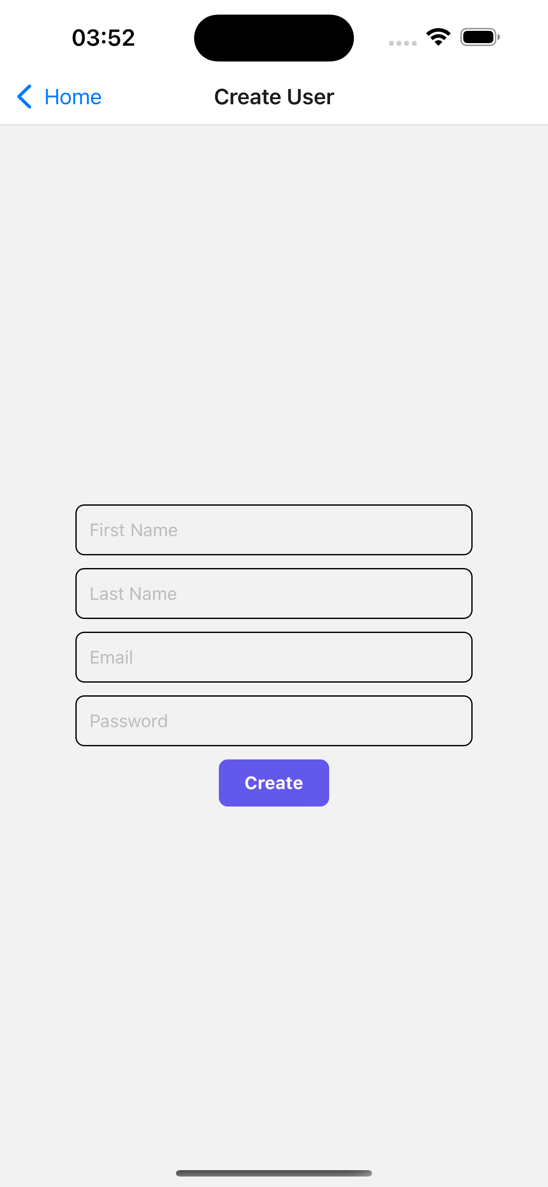


Рисунок 3.3 – Экран создания аккаунта сотрудника

При нажатии на кнопку «Create User», администратор попадает на следующий экран, представленный на рисунке 3.3, где он имеет возможность создать аккаунт для своего сотрудника. Данный экран является крайне минималистичным и интуитивно понятным. Администратору необходимо заполнить несколько полей, включая имя, фамилию, адрес электронной почты и пароль будущего сотрудника. После ввода всех необходимых данных и нажатия кнопки «Create», приложение выполняет проверку корректности данных и, в случае успеха, создает новый аккаунт сотрудника. Об успешном создании аккаунта администратор уведомляется с помощью всплывающего сообщения. После этого приложение автоматически возвращает администратора на предыдущий экран, где отображается список всех сотрудников.

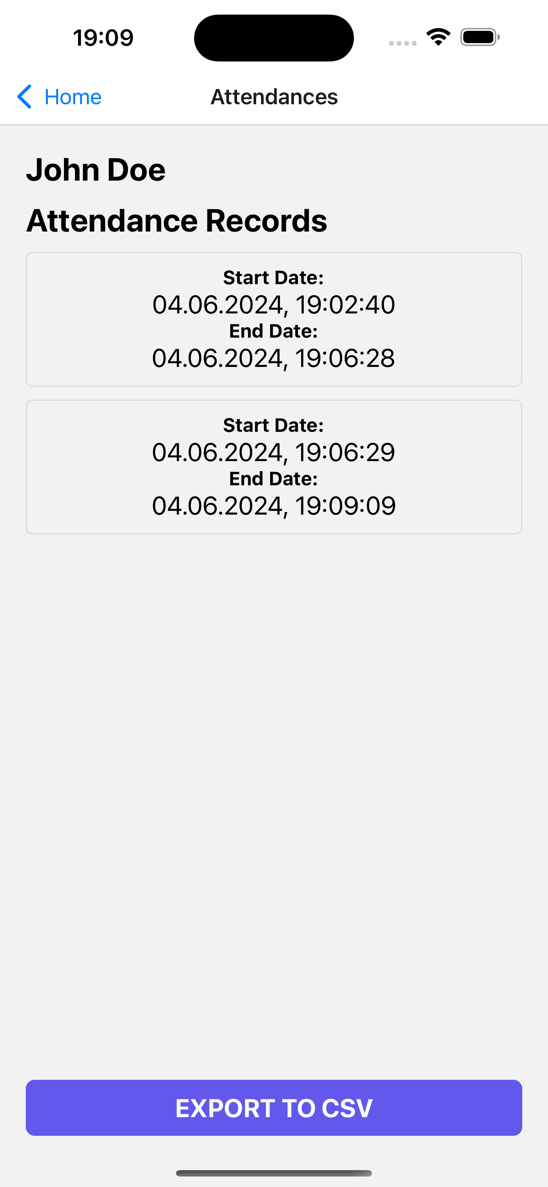


Рисунок 3.4 – Экран истории отметок конкретного сотрудника

На рисунке 3.4 представлен экран, где администратор имеет возможность просмотреть всю историю отметок конкретного сотрудника. Вверху экрана отображается имя сотрудника, а ниже – список его посещений с указанием даты и времени начала и окончания каждого посещения в формате ДД.ММ.ГГГГ. Внизу экрана расположена кнопка «EXPORT TO CSV», позволяющая экспортировать историю посещений в файл CSV.

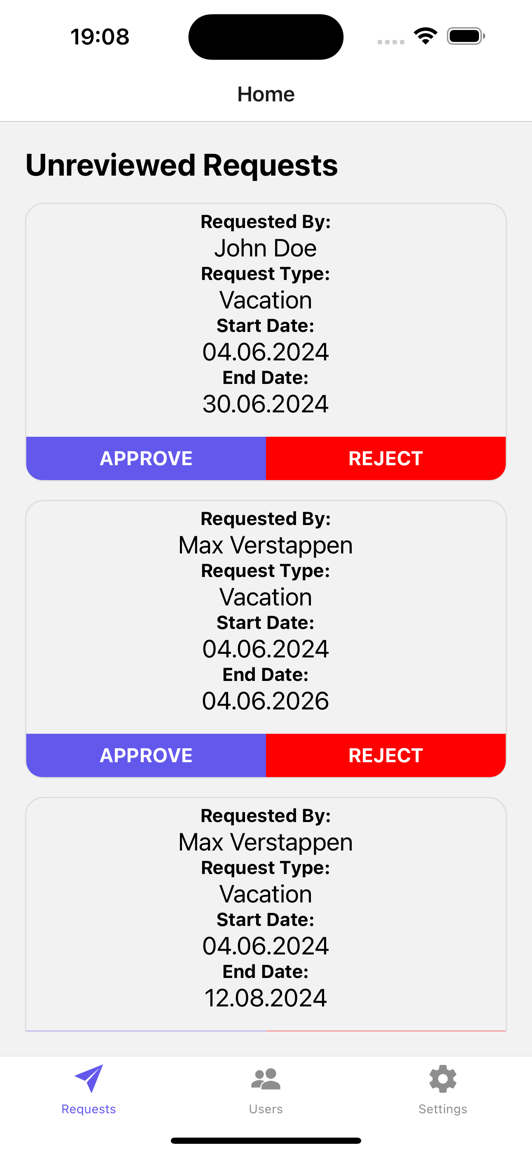


Рисунок 3.5 – Экран рассмотрения запросов сотрудников

На рисунке 3.5 показан экран, который отображает список запросов от сотрудников, ожидающих рассмотрения, каждый из которых представлен в виде отдельного блока с информацией об отправителе, типе запроса, дате начала и окончания отпуска. Под каждым запросом расположены две кнопки: «APPROVE» для одобрения и «REJECT» для отклонения.

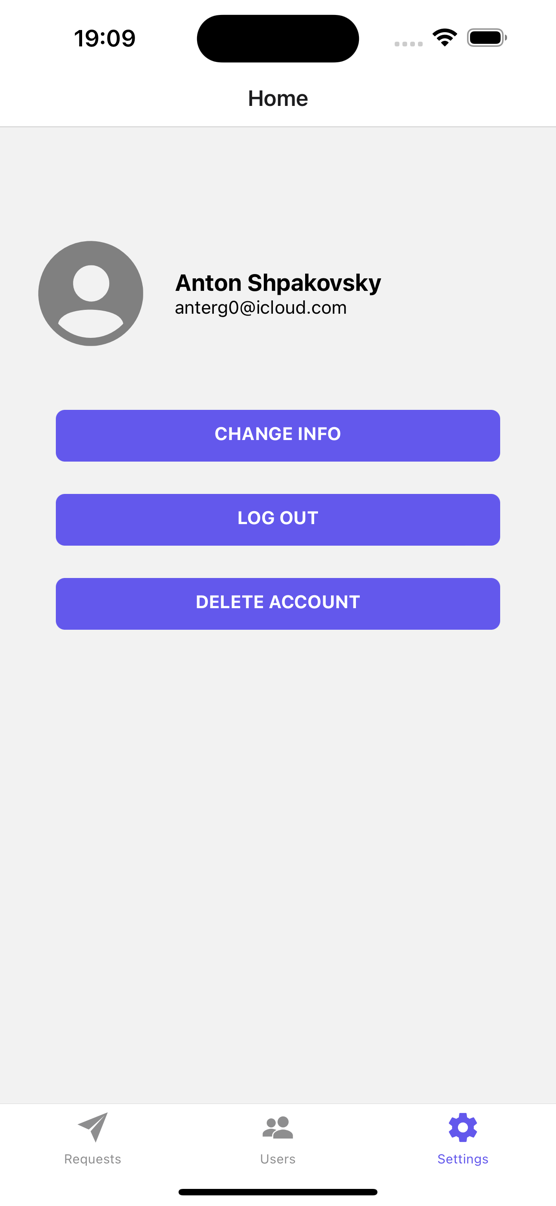


Рисунок 3.6 – Экран настроек

Экран настроек, изображенный на рисунке 3.6, пользователя в приложении «Attendify» отображает аватар, имя и адрес электронной почты пользователя. Ниже расположены три кнопки: «CHANGE INFO» для редактирования личной информации, «LOG OUT» для выхода из аккаунта и «DELETE ACCOUNT» для удаления аккаунта.

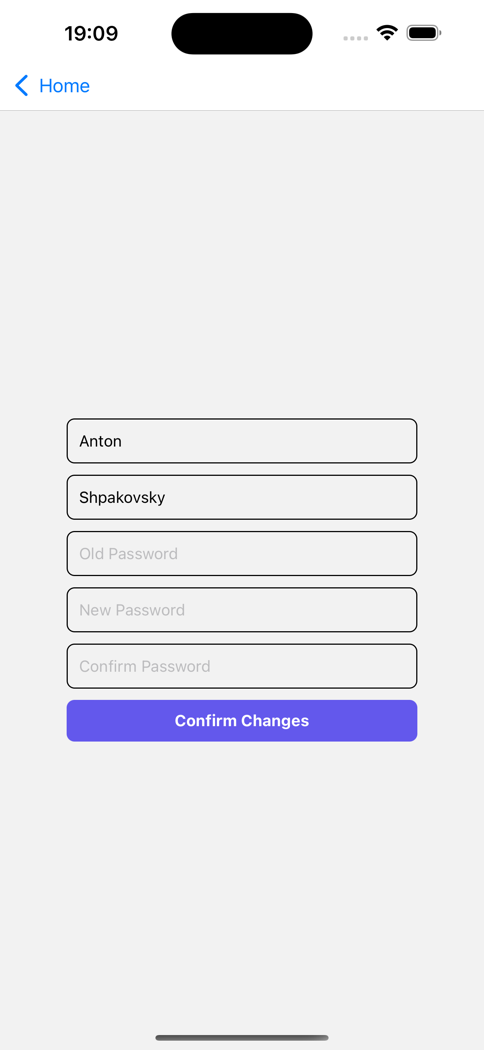


Рисунок 3.7 – Экран изменения личной информации

При нажатии на кнопку «CHANGE INFO» на экране настроек, пользователь попадает на экран, изображенный на рисунке 3.7. Здесь он имеет возможность изменить своё имя, фамилию, а также пароль. Для смены пароля стоит просто ввести старый пароль, после ввести новый и подтвердить правильный ввод нового пароля, введя его снова в последнее текстовое поле. После нажатия на кнопку «CONFIRM CHANGES», в зависимости от того, какие поля были изменены, произойдёт смена информации, а после всплывёт сообщение об успешной смене данных и пользователя вернет на прошлый экран.

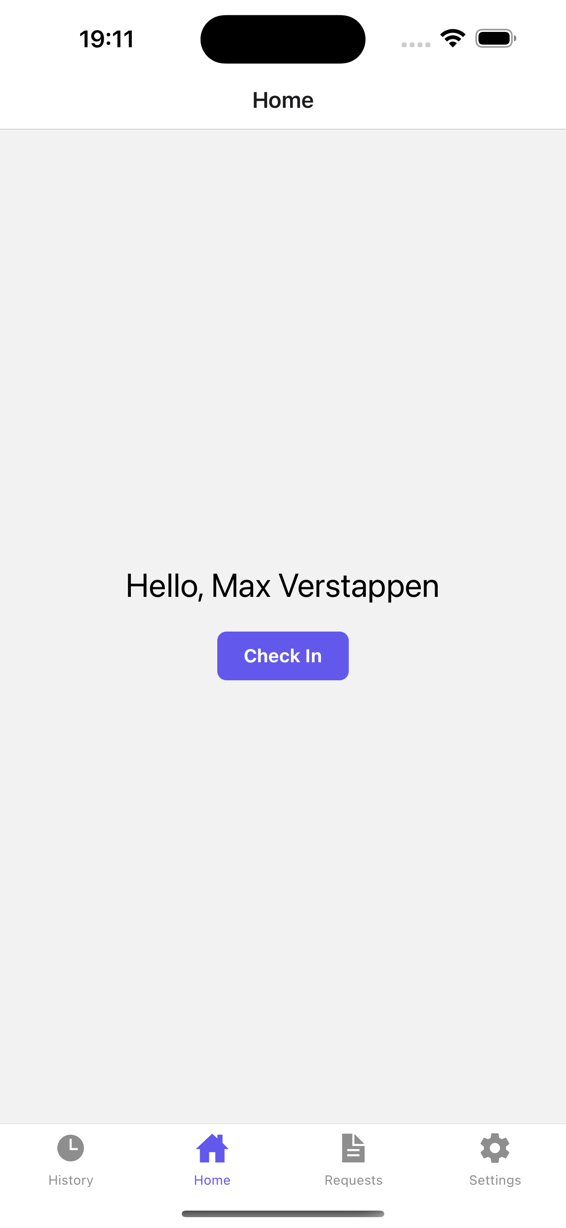


Рисунок 3.8 – Главный экран сотрудника

При успешном входе в свой аккаунт, сотрудник попадает на экран, изображенный на рисунке 3.8, где он может отметить своё присутствие. Внизу экрана расположена панель навигации, с помощью которой, пользователь может просмотреть свою историю посещений, свои запросы, а также может просмотреть настройки.

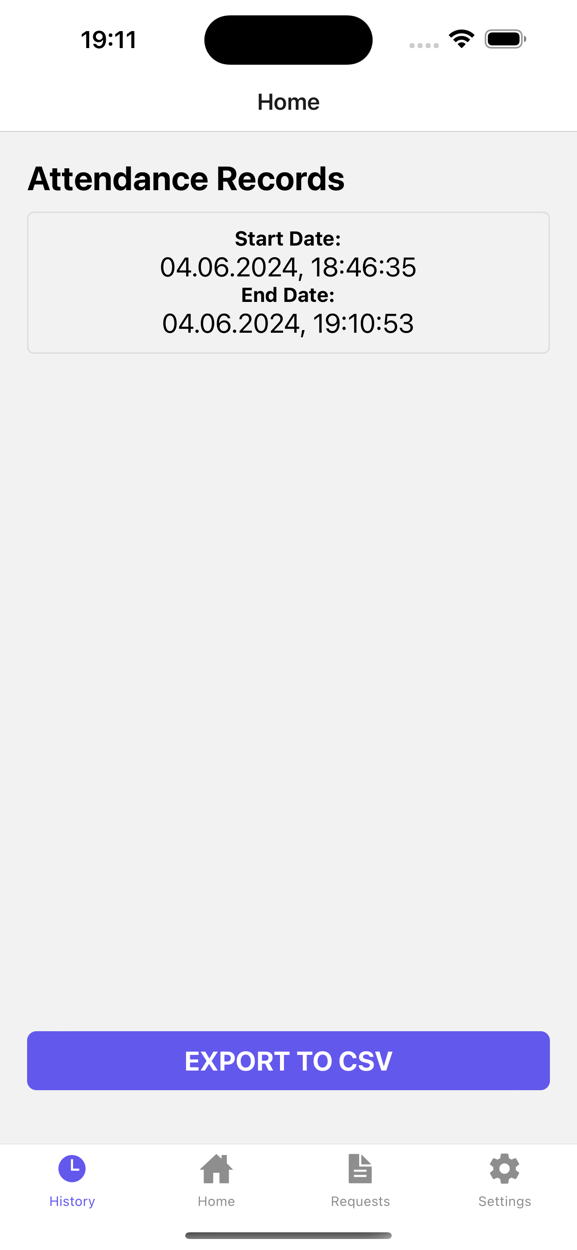


Рисунок 3.9 – Экран истории отметок

Экран, изображенный на рисунке 3.9, ничем не отличается от экрана, который может просматривать администратор. Единственное отличие заключается в том, что вверху экрана не указано имя сотрудника. Аналогично на этом экране можно просмотреть свою историю отметок, а также экспортировать всю историю в CSV файл.

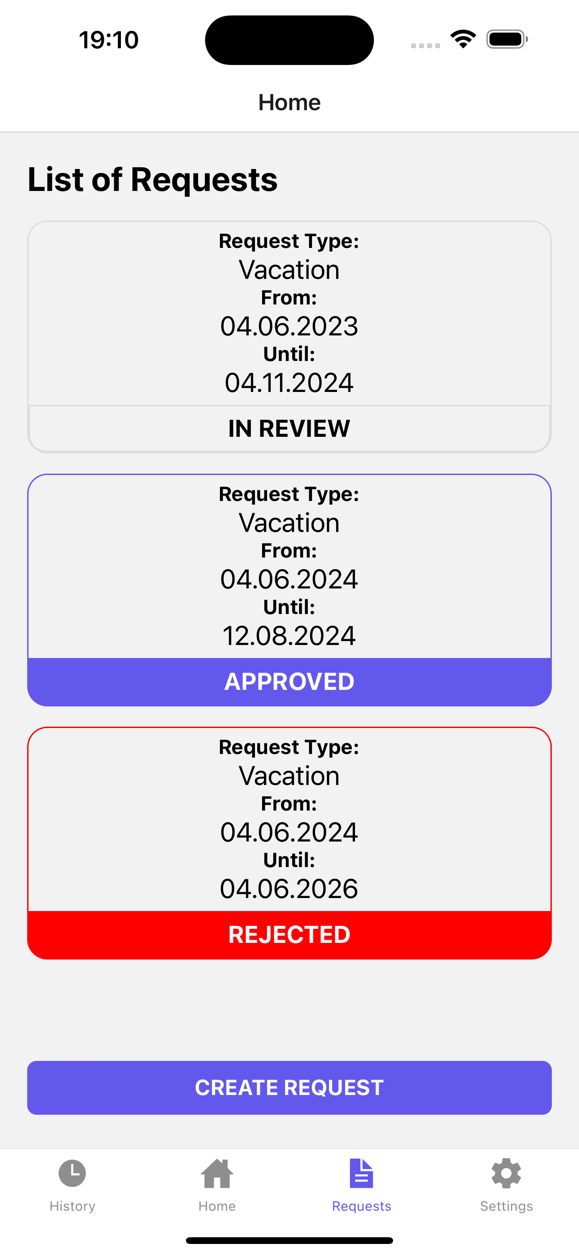


Рисунок 3.10 – Экран запросов сотрудника

Перейдём к экрану запросов сотрудника. На рисунке 3.10 изображен экран, отображающий историю запросов пользователя, включая тип запроса (в данном случае все запросы относятся к типу «Vacation»), даты начала и окончания, а также статус каждого запроса («In Review», «Approved» или «Rejected»). Внизу экрана расположена кнопка «CREATE REQUEST», позволяющая перейти на другой экран, где пользователь может создать новый запрос.

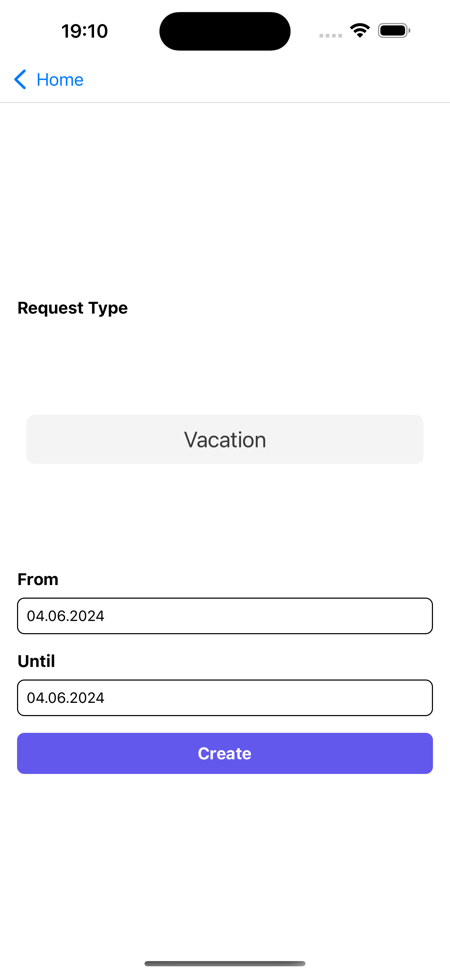


Рисунок 3.11 – Экран создания запроса

На рисунке 3.11 изображен экран создания запроса, который позволяет сотруднику создать запрос определенного типа, в последствии который будет рассмотрен одним из администраторов. На данном экране присутствует выбор типа запроса, а последующие элементы зависят от самого типа запроса. В данной ситуации, для типа запроса «Vacation» предлагается выбрать дату начала отпуска, а также дату окончания. При нажатии на кнопку «Create» будет создан запрос и пользователь будет возвращен на предыдущий экран.

## 3.2 Анализ необходимых средств для отображения графики

В React Native все элементы пользовательского интерфейса строятся из компонентов. Это фундаментальная концепция фреймворка, позволяющая создавать гибкие и динамичные интерфейсы для мобильных приложений. Компоненты в React Native представляют собой независимые, многократно используемые модули, которые определяют внешний вид и поведение части интерфейса. Они могут быть как простыми, например, текстовые элементы или кнопки, так и сложными, состоящими из множества других компонентов, образуя иерархическую структуру.

Компоненты в React Native могут быть функциональными или классовыми. Функциональные компоненты – это простые JavaScript-функции, которые принимают пропсы (свойства) и возвращают элементы React, описывающие часть пользовательского интерфейса. Классовые компоненты, в свою очередь, являются классами JavaScript, которые расширяют базовый класс Component и имеют более сложную структуру, включая методы жизненного цикла и состояние.

Типичные компоненты, используемые в React Native приложениях, включают View, Text, Image, TextInput, TouchableOpacity и другие. Компонент View является основным строительным блоком для создания макетов и контейнеров для других компонентов. Text используется для отображения текста, Image для отображения изображений, TextInput для ввода текста пользователем, а TouchableOpacity для создания интерактивных кнопок и областей.

React Native также предоставляет множество встроенных API и компонентов для работы с различными аспектами мобильных приложений, такими как навигация, анимация, доступ к камере и геолокации, работа с сетью и многое другое. Кроме того, существует огромное количество сторонних библиотек и компонентов, которые расширяют возможности React Native и упрощают разработку сложных приложений.

В данном проекте для обеспечения визуальной привлекательности и единообразия интерфейса была использована библиотека Ionicons. Она предоставляет богатый набор качественных иконок, которые легко интегрируются в компоненты React Native, позволяя создавать интуитивно понятный и эстетически приятный пользовательский интерфейс.

## 3.3 Разработка программного кода

Данный курсовой проект состоит из множества файлов, каждый из которых выполняет определенную роль в архитектуре и функциональности приложения. Основные файлы включают:

* index.tsx (UserHome и AdminHome)
* requestService.tsx
* attendanceService.tsx
* userService.tsx

Начнем с index.tsx (UserHome и AdminHome). Эти файлы представляют собой корневые компоненты для двух основных интерфейсов приложения: интерфейса пользователя и интерфейса администратора. Они определяют структуру навигации по вкладкам, позволяя пользователям переключаться между различными экранами приложения. Каждый экран представлен отдельным компонентом React Native и отвечает за отображение информации и взаимодействие с пользователем.

Файл requestService.tsx содержит бизнес-логику, связанную с обработкой запросов. Функции createVacationRequest, reviewRequest и getRequests отвечают за создание новых запросов, их рассмотрение (одобрение или отклонение) администратором, а также за получение списка запросов для пользователя или администратора.

В файле attendanceService.tsx реализована основная логика учета рабочего времени сотрудников. Функции checkIn и checkOut регистрируют приход и уход сотрудников, обновляя их статус в базе данных Firestore. Функция getAttendances позволяет получать историю посещений конкретного сотрудника, а checkOngoingAttendance проверяет, отмечен ли сотрудник в данный момент на работе.

И наконец, файл userService.tsx отвечает за управление пользователями, включая аутентификацию, создание, обновление и удаление учетных записей. Функции signIn и createUser реализуют вход и создание пользователей соответственно, используя Firebase Authentication. Функции updateUser, updatePass и delUser позволяют изменять данные профиля и удалять учетные записи. Функции getUsers и getUser используются для получения информации о пользователях, а isAdmin проверяет, является ли текущий пользователь администратором.

# 4. ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРИЛОЖЕНИЯ

Осуществлялось функциональное тестирование; отчёт о проведённом тестировании представлен в таблице 4.1.

Таблица 4.1 – Тестирование программного средства

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 1 | Создание аккаунта сотрудника | Создание нового аккаунта сотрудника и возврат на предыдущий экран | Соответствует заданному результату |
| 2 | Попытка войти в аккаунт, неправильно указав формат электронной почты | Всплывающее сообщение, уведомляющее о неверном формате электронной почты | Соответствует заданному результату |
| 3 | Попытка создания запроса об отпуске, где дата начала установлена после даты окончания | Всплывающее сообщение, уведомляющее о неправильном вводе даты начала | Соответствует заданному результату |
| 4 | Нажатие кнопки удаления аккаунта | Успешное удаление аккаунта и перенос на страницу входа | Соответствует заданному результату |
| 5 | Выход с аккаунта | Выход с аккаунта и последующее перенаправление на страницу с авторизацией | Соответствует заданному результату |
| 6 | Смена имени и фамилии | Имя и фамилия пользователя успешно изменятся | Соответствует заданному результату |
| 7 | Смена пароля | Пароль успешно сменится | Соответствует заданному результату |
| 9 | Экспорт истории отметок в файл CSV | Появление опций приложений для отправки CSV файла | Соответствует заданному результату |

Продолжение таблицы 4.1

|  |  |  |  |
| --- | --- | --- | --- |
| № | Тестируемая функция | Ожидаемый результат | Полученный результат |
| 8 | Попытка ввода подтверждения пароля, не соответствующее новому паролю | Появление надписи о несоответствии паролей | Соответствует заданному результату |
| 10 | Одобрение запроса | Запрос одобрен и сотрудник сможет увидеть обновленный статус запроса | Соответствует заданному результату |

Итоговое комплексное тестирование приложения, проведенное в соответствии с утвержденным планом, подтвердило полное отсутствие критических ошибок, неисправностей и непредвиденных сбоев в его работе. Приложение успешно прошло все предусмотренные тестовые сценарии, включая нагрузочное тестирование, тестирование безопасности и тестирование удобства использования, демонстрируя высокую стабильность, надежность и корректность функционирования в различных, в том числе экстремальных, условиях эксплуатации.

Полученные результаты свидетельствуют о высоком качестве разработки, строгом соблюдении стандартов кодирования и тщательном подходе к тестированию на всех этапах жизненного цикла проекта. Приложение полностью соответствует заявленным функциональным требованиям и готово к развертыванию и использованию в производственной среде.

В целом, результаты итогового тестирования подтверждают, что приложение является надежным, безопасным и эффективным инструментом, способным удовлетворить потребности пользователей и принести значительную пользу бизнесу.

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

Для начала работы с приложением «Attendify», его необходимо установить на ваше мобильное устройство и запустить. При первом запуске вы увидите экран входа, где необходимо ввести адрес электронной почты и пароль, предоставленные вашим администратором.

После успешного входа в аккаунт, функциональность, доступная вам в приложении, будет зависеть от ваших полномочий. Если вы являетесь администратором, вы попадете на панель управления, где сможете просматривать историю посещений всех сотрудников, создавать новые учетные записи для сотрудников, а также одобрять или отклонять их запросы на отпуск.

Если же вы являетесь сотрудником, то после входа вы окажетесь на главной странице, где сможете отмечать свое присутствие на работе, просматривать историю своих отметок, а также создавать и отслеживать статус своих запросов на отпуск.

В обоих случаях, вам будет доступен раздел настроек, где вы сможете просмотреть свою личную информацию, такую как полное имя и адрес электронной почты. В настройках вы также сможете перейти на экран редактирования профиля, где можно изменить свой пароль или имя и фамилию. Для изменения пароля вам потребуется подтвердить свой старый пароль в специальном поле.

Кроме того, в настройках вы найдете опции для выхода из аккаунта и удаления аккаунта. Важно отметить, что для удаления аккаунта потребуется подтверждение, чтобы предотвратить случайное удаление.

# ЗАКЛЮЧЕНИЕ

В рамках данного проекта было разработано мобильное приложение «Attendify» для учета рабочего времени сотрудников, которое полностью соответствует поставленным задачам и требованиям. Приложение демонстрирует высокий уровень качества и функциональности, предоставляя удобные инструменты как для сотрудников, так и для администраторов.

Однако, «Attendify» имеет значительный потенциал для дальнейшего развития. Одним из возможных направлений развития является добавление функциональности для учета рабочего времени на конкретных задачах или проектах, что позволит более детально анализировать эффективность работы сотрудников. Также можно рассмотреть возможность отметки сотрудников используя такую технологию как NFC.

Другим перспективным направлением развития является расширение функциональности приложения для администраторов. Например, можно добавить возможность просмотра статистики по отделам или группам сотрудников, а также инструменты для более гибкого управления правами доступа.

Приложение «Attendify» уже сейчас обладает достаточным функционалом для эффективного учета рабочего времени сотрудников и может быть успешно внедрено в компаниях небольших размеров. Дальнейшее развитие приложения позволит расширить его возможности и сделать его еще более полезным инструментом для управления персоналом и оптимизации рабочего процесса.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] React Native Documentation [Электронный ресурс]. – Режим доступа: https://reactnative.dev/docs/getting-started. – Дата доступа: 20.04.2024.

[2] Firebase Documentation [Электронный ресурс]. – Режим доступа: https://firebase.google.com/docs. – Дата доступа: 20.04.2024.

[3] React Navigation Documentation [Электронный ресурс]. – Режим доступа: https://reactnavigation.org/docs/getting-started. – Дата доступа: 14.05.2024.

[4] Expo Documentation [Электронный ресурс]. – Режим доступа: https://docs.expo.dev/. – Дата доступа: 25.04.2024.

ПРИЛОЖЕНИЕ А  
(обязательное)  
Исходный код

import { initializeApp } from "firebase/app";

import { getAuth, createUserWithEmailAndPassword, signInWithEmailAndPassword, reauthenticateWithCredential, updatePassword, EmailAuthProvider, deleteUser, initializeAuth, updateProfile } from "firebase/auth";

import { collection, deleteDoc, doc, getDoc, getDocs, getFirestore, setDoc, updateDoc } from 'firebase/firestore';

import { Alert } from "react-native";

const firebaseConfig = {

apiKey: "AIzaSyDfyM\_pSJ79bBQywoEzB5Le6eZUWrFplvc",

authDomain: "attendify-e4203.firebaseapp.com",

projectId: "attendify-e4203",

storageBucket: "attendify-e4203.appspot.com",

messagingSenderId: "675361693235",

appId: "1:675361693235:web:5d2afdfb34b31496664d46",

measurementId: "G-N3MJYQHKM2"

};

const app = initializeApp(firebaseConfig);

const auth = getAuth(app);

const db = getFirestore(app);

export const getUser = async (adminUID, targetUID) => {

const userRef = doc(db,'users',adminUID);

const userSnap = await getDoc(userRef);

if (userSnap.data().type === 'admin') {

const targetRef = doc(db,'users',targetUID);

const targetSnap = await getDoc(targetRef);

return targetSnap.data();

} else {

console.error('You are not an admin.');

throw new Error('You are not an admin.')

}

};

export const updateUserInfo = async (firstName, lastName) => {

try {

const userRef = doc(db,'users',auth.currentUser.uid);

await updateDoc(userRef, {

firstName: firstName,

lastName: lastName,

});

await updateProfile(auth.currentUser, { displayName: `${firstName} ${lastName}` });

} catch (error) {

console.error(error);

throw error;

}

};

export const updateUserPass = async (curPass, newPass) => {

try {

const creds = EmailAuthProvider.credential(auth.currentUser.email,curPass);

await reauthenticateWithCredential(auth.currentUser, creds).then(() => {

updatePassword(auth.currentUser, newPass);

});

} catch (error) {

console.error(error);

throw error;

}

};

export const delUser = async () => {

try {

const userID = auth.currentUser.uid;

await deleteDoc(doc(db, 'users', userID));

await deleteUser(auth.currentUser);

} catch (error) {

throw error;

}

};

export const getUsers = async (userID) => {

const userRef = doc(db, 'users', userID);

const userSnap = await getDoc(userRef);

if (userSnap.data().type === 'admin') {

const usersRef = collection(db, 'users');

const snapshot = await getDocs(usersRef);

const userList = snapshot.docs.map(doc => ({ id: doc.id, ...doc.data() }));

return userList;

} else {

Alert.alert('Error', 'How did we even get here?');

return null;

}

};

export const createUser = async (email, password, firstName, lastName) => {

try {

const db = getFirestore();

const secondaryApp = initializeApp(firebaseConfig, "Secondary");

const { user } = await createUserWithEmailAndPassword(

getAuth(secondaryApp),

email,

password

);

await updateProfile(user, {

displayName: `${firstName} ${lastName}`,

});

const usersCollection = collection(db, 'users');

const newUser = {

firstName,

lastName,

type: 'user',

};

const newDoc = doc(usersCollection, user.uid);

await setDoc(newDoc, newUser);

console.log('User successfully created!', user.uid);

} catch (error) {

console.error('Error creating account:', error);

throw error;

}

};

export const signIn = async (email, password) => {

try {

const userCredential = await signInWithEmailAndPassword(auth, email, password);

const user = userCredential.user;

const userDocRef = doc(db,'users',user.uid);

const userDocSnap = await getDoc(userDocRef);

if (userDocSnap.exists()) {

const UserData = userDocSnap.data();

console.log('User db data:', UserData);

console.log('User auth data: ', user);

return UserData;

} else {

console.error('User data not found in Firestore');

}

} catch (error) {

console.error('Error signing in user:', error.code);

let errorMessage = '';

switch (error.code) {

case 'auth/user-not-found':

errorMessage = 'User not found. Please check your email or sign up.';

break;

case 'auth/wrong-password':

errorMessage = 'Incorrect password. Please try again.';

break;

case 'auth/invalid-email':

errorMessage = 'Invalid email format. Please enter a valid email address.';

break;

case 'auth/email-already-in-use':

errorMessage = 'Email is already in use. Please use a different email or sign in.';

break;

case 'auth/too-many-requests':

errorMessage = 'Too many unsuccessful login attempts. Please try again later.';

break;

case 'auth/network-request-failed':

errorMessage = 'Network error. Please check your internet connection.';

break;

case 'auth/invalid-credential':

errorMessage = 'Invalid credentials.'

break;

default:

errorMessage = 'An error occurred. Please try again later.';

}

Alert.alert('Authentication Error', errorMessage);

}

};

export const isAdmin = async () => {

const userRef = doc(db, 'users', auth.currentUser.uid);

const userSnap = await getDoc(userRef);

if (userSnap.data().type === 'admin') return true;

return false;

};

export const getMyself = async () => {

const userRef = doc(db, 'users', auth.currentUser.uid);

const userSnap = await getDoc(userRef);

return userSnap.data();

};

# ПРИЛОЖЕНИЕ Б (обязательное) Схемы алгоритмов программного средства

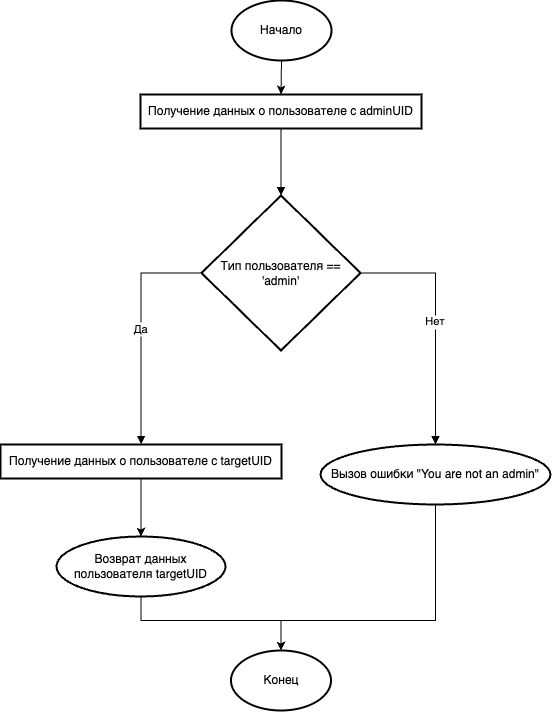


Рисунок Б.1 – Блок-схема алгоритма получения данных пользователя

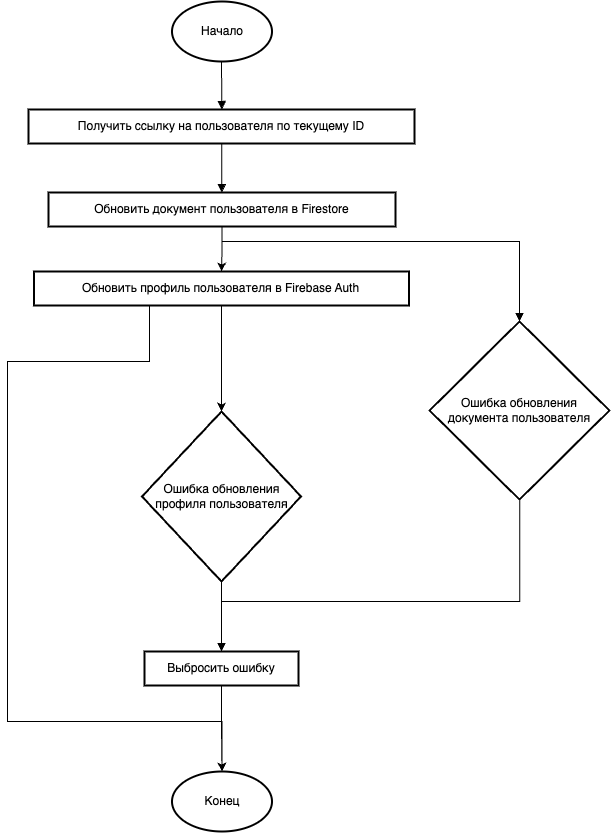


Рисунок Б.2 – Блок-схема обновления имени и фамилии пользователя

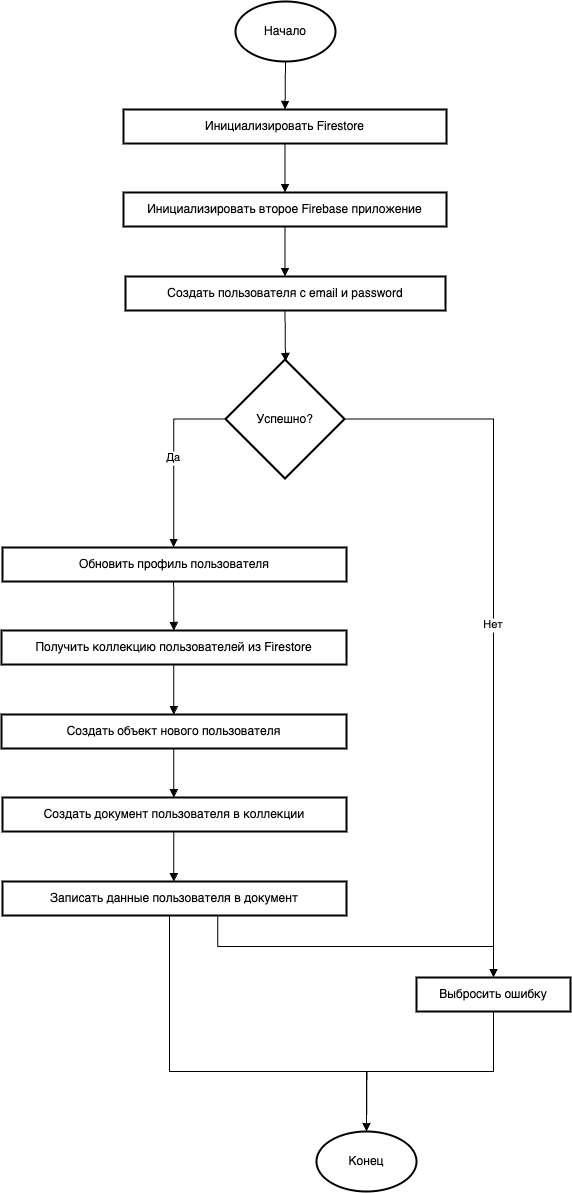


Рисунок Б.3 – Блок-схема алгоритма создания нового аккаунта

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | Наименование | | | | Дополнительные сведения | | |
|  | | | | Текстовые документы | | | |  | | |
|  | | | |  | | | |  | | |
| БГУИР КП 1-40 04 01 | | | | Пояснительная записка | | | | 40 с. | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | | Графические документы | | | |  | | |
| ГУИР 253505 014 СА | | | | Программное средство для тестирования. Схемы алгоритмов | | | | Формат А4 | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  | | | |  | | | |  | | |
|  |  |  |  |  | БГУИР КП 1-40 04 01 | | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
| Изм. | Л. | № докум. | Подп. | Дата | Система трекинга рабочего времени сотрудников  Ведомость курсового проекта |  | | | Лист | Листов |
| Разраб. | | *Шпаковский А.В.* |  | 05.06.24 | Т |  |  | 40 | 40 |
| Пров. | | *Тушинская Е.В.* |  | 05.06.24 | Кафедра Информатики гр. 253505 | | | | |
|  | |  |  |  |