МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
«Тольяттинский государственный университет»

Кафедра «Прикладная математика и информатика»
(наименование)

02.03.03 «Математическое обеспечение и администрирование информационных систем»
(код и наименование направления подготовки / специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка мобильного приложения для управления сотрудниками компании: увольнение и приём на работу, анализ рабочего времени по категориям»

Обучающийся Д.Р. Абдуллин
(Инициалы Фамилия) (личная подпись)

Руководитель М.А. Тренина
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант к.п.н., доцент С.А. Гудкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2025

Аннотация

Тема бакалаврской работы: «Приложение для управления сотрудниками компании: увольнение и приём на работу, анализ рабочего времени по категориям».

Цель данной работы заключается в создании многофункционального программного решения, которое автоматизирует процессы управления персоналом, включая процедуры приема и увольнения сотрудников, а также предоставляет средства для анализа рабочего времени по различным категориям.

Актуальность проекта объясняется необходимостью повышения эффективности управления людскими ресурсами в современных компаниях, что особенно важно в условиях усиленной конкуренции и динамичного делового окружения. Структура работы включает введение, три основные главы, заключение и приложения.

В первой разделе уделено внимание анализу проблем, связанных с традиционными методами управления персоналом, и проведению сравнительного анализа существующих решений. Также обосновывается выбор платформы и технологий, которые будут использованы для разработки приложения.

Второй раздел сосредоточен на проектировании системы управления кадрами. Здесь рассматриваются формирование требований к функциональности приложения, выбор подходящих технологий для логического моделирования, а также разработка модели данных. Описывается архитектура программного продукта и его потенциальная интеграция с другими системами компании.

Третий раздел посвящен процессу реализации приложения. В ней подробно описывается разработка пользовательского интерфейса, создание модулей для управления персоналом и анализа рабочего времени.

Abstract

The title of the graduation work is: "Develop application for company employee management: hiring, dismissal, and working time analysis by categories."

The graduation work is devoted to the development of a software solution for automating HR processes in companies, including employee hiring and dismissal procedures, as well as tracking and analyzing working hours across different categories. The goal is to optimize personnel management, reduce administrative workload, and improve the accuracy of workforce analytics.

The main challenge addressed in this research is the inefficiency of manual employee management processes and the lack of integrated tools for comprehensive working time analysis in many organizations.

The first chapter explores the relevance of automating HR tasks, examining existing problems in traditional personnel management methods. It includes an analysis of current HR software solutions, justification for developing a specialized application, and an overview of key functional requirements.

The second chapter focuses on the system design, covering database structure, user roles, and business process automation for hiring and dismissal workflows. The technology stack selection is explained, with emphasis on performance, security, and compatibility with corporate IT infrastructure.

The third chapter describes the practical implementation of the application. It details the development of core modules: employee records management, time tracking, and analytical reporting. The chapter also discusses interface design principles, data processing methods, and testing procedures to ensure system reliability.

In conclusion, the work demonstrates that the developed application provides an effective solution for streamlining HR operations and enhancing workforce productivity analysis. The results can be applied in various business environments to improve personnel management efficiency. The project's outcomes contribute to the broader field of digital HR management systems.

Содержание

Введение

В современном деловом мире эффективное управление персоналом является ключевым фактором успеха любой компании. С ростом конкуренции и увеличением объема данных, связанных с управлением человеческими ресурсами, традиционные методы управления становятся все менее эффективными. Компании стремятся внедрять инновационные решения для автоматизации процессов, связанных с приемом на работу, увольнением сотрудников и анализом их рабочего времени.

Цифровизация бизнес-процессов позволяет значительно улучшить управление персоналом, делая его более прозрачным и оперативным. Приложения для управления сотрудниками становятся незаменимыми инструментами для современных организаций, предлагая автоматизацию рутинных задач и предоставляя возможность более глубокого анализа данных. Это позволяет не только сократить затраты времени и ресурсов, но и повысить общую эффективность работы команды.

Настоящая работа посвящена разработке приложения для управления сотрудниками компании, которое автоматизирует процессы приема и увольнения, а также предоставляет инструменты для анализа рабочего времени по различным категориям. Особое внимание уделяется интеграции функциональных возможностей, которые обеспечивают легкость в использовании и адаптацию под потребности конкретной организации.

Актуальность данной работы заключается в необходимости повышения эффективности управления персоналом в современных компаниях. В условиях усиленной конкуренции и динамично изменяющейся деловой среды традиционные методы управления становятся недостаточно эффективными. Автоматизация процессов приема и увольнения сотрудников, а также анализ рабочего времени с помощью специализированного приложения позволяет не только сократить затраты времени и ресурсов, но и улучшить качество управления человеческими ресурсами. Это, в свою очередь, способствует

повышению производительности труда, улучшению планирования и повышению удовлетворенности сотрудников, что делает разработку и внедрение такого приложения крайне актуальной задачей для компаний, стремящихся к повышению своей конкурентоспособности.

Для выполнения задачи по разработке приложения для управления сотрудниками компании были сделаны такие пункты:

- исследование текущих решений и потребностей: Проведено тщательное изучение современных методов и программных инструментов для управления персоналом. Это позволило выявить основные требования и функциональные возможности, которые следует интегрировать в новое приложение;
- создание интерфейса и архитектуры: Разработана архитектура приложения, учитывающая будущие возможности масштабирования и интеграции с другими системами компании. Спроектирован пользовательский интерфейс, ориентированный на удобство и простоту использования;
- разработка функциональности: Созданы и интегрированы модули, обеспечивающие автоматизацию процессов приема и увольнения сотрудников, а также инструменты для анализа рабочего времени по различным категориям;
- тестирование: Выполнено тестирование приложения для оценки его функционала и пользовательского опыта. На основании результатов тестирования выполнена оптимизация, направленная на повышение эффективности и надежности работы приложения.

Работа состоит из трех разделов, в них рассмотрены этапы разработки, тестирования и оптимизации приложения.

1 Виды мобильных приложений и актуальность разработки

1.1 Проблемы управления персоналом в традиционных системах

Управление персоналом в традиционных системах сталкивается с рядом проблем, которые снижают эффективность рабочих процессов и могут негативно сказываться на общей производительности компании. Одной из основных трудностей является зависимость от ручных процессов и бумажной документации. Многие компании до сих пор полагаются на такие методы для управления персоналом, включая ведение табелей учета рабочего времени и кадровые изменения, что не только занимает много времени, но и подвержено ошибкам.

Еще одной проблемой является отсутствие оперативного доступа к данным. В традиционных системах информация о сотрудниках часто хранится в разрозненных местах, что затрудняет быстрый доступ к данным и их анализ. Это может приводить к задержкам в принятии решений и неэффективному управлению ресурсами.

Низкая прозрачность процессов также является значительной проблемой. Недостаток прозрачности в кадровых процессах может вызывать недопонимания и недовольство среди сотрудников, особенно в вопросах, связанных с начислением заработной платы, отпусками и переработками.

Кроме того, традиционные системы часто не обладают инструментами для глубокого анализа данных, что затрудняет оценку эффективности сотрудников и выявление областей для улучшения. Это ограничивает возможности компании использовать данные для стратегического планирования и оптимизации процессов.

Наконец, в условиях быстрого изменения бизнес-среды компании, использующие устаревшие системы, могут испытывать трудности с адаптацией к новым требованиям и стандартам управления персоналом. Решение этих проблем посредством внедрения современных технологий и

автоматизированных систем управления персоналом может существенно улучшить эффективность работы компании и повысить удовлетворенность сотрудников.

Разработка приложения для управления сотрудниками компании способствует значительному улучшению эффективности и прозрачности процессов управления персоналом.

Во-первых, такое приложение автоматизирует рутинные операции, такие как прием и увольнение сотрудников, что позволяет значительно сократить время и ресурсы, затрачиваемые на эти процедуры. Это, в свою очередь, освобождает сотрудников отдела кадров для выполнения более стратегических задач.

Во-вторых, приложение предоставляет инструменты для детального анализа рабочего времени, что помогает выявить неэффективности и оптимизировать распределение трудовых ресурсов. Это способствует повышению производительности и снижению затрат, связанных с переработками и нерациональным использованием рабочего времени.

Кроме того, централизованный доступ к информации о сотрудниках улучшает коммуникацию и координацию внутри компании. Это облегчает процесс принятия решений и повышает прозрачность кадровых процессов, что может привести к увеличению доверия и удовлетворенности сотрудников.

Наконец, внедрение такого приложения помогает компании адаптироваться к быстро меняющимся условиям рынка и требованиям законодательства, обеспечивая гибкость и готовность к изменениям. Это делает компанию более конкурентоспособной и способной эффективно реагировать на новые вызовы в нашем мире, поэтому эта система невероятно важна для нас чтобы максимизировать эффективность работы и минимизировать риски в условиях постоянных изменений.

1.2 Сравнительный анализ видов мобильных приложений

В нашем мире мобильные приложения стали неотъемлемой частью повседневной жизни, играя важную роль в различных сферах, от коммуникации и развлечений до бизнеса и образования. С развитием технологий разработка мобильных приложений стала более доступной, предлагая разработчикам широкий выбор из способов их создания. Давайте рассмотрим виды мобильных приложений.

Нативные приложения: они предполагают создание приложений под конкретную операционную систему с использованием специализированных языков и инструментов. Для iOS основными инструментами являются Swift и Objective-C в среде Xcode, а для Android Kotlin и Java в Android Studio. Преимущество нативного подхода заключается в высокой производительности, полном доступе к функциям устройства и соответствию гайдлайнам операционной системы. Однако разработка требует отдельных ресурсов для каждой платформы, что увеличивает временные и финансовые затраты.

Кроссплатформенные приложения: позволяют создавать приложения для нескольких операционных систем из единой кодовой базы. К ним относятся React Native (на базе JavaScript/TypeScript), Flutter (с использованием языка Dart) и Xamarin (на основе C#). Эти технологии сокращают время разработки и упрощают поддержку, но могут ограничивать доступ к некоторым функциям ОС. Например, Flutter обеспечивает высокую кастомизацию интерфейса благодаря собственному движку рендеринга, а React Native интегрирует нативные компоненты для улучшения производительности.

Кроссплатформенные приложения позволяют создавать приложения для нескольких операционных систем из единой кодовой базы. К ним относятся React Native (на базе JavaScript/TypeScript), Flutter (с использованием языка Dart) и Xamarin (на основе C#). Эти технологии

сокращают время разработки и упрощают поддержку, но могут ограничивать доступ к некоторым функциям ОС. Например, Flutter обеспечивает высокую кастомизацию интерфейса благодаря собственному движку рендеринга, а React Native интегрирует нативные компоненты для улучшения производительности.

Гибридные приложения: сочетают веб-технологии (HTML, CSS, JavaScript) с нативной оболочкой, например, через WebView. Решения вроде Ionic и Apache Cordova подходят для разработчиков с опытом веб-программирования, так как требуют минимального обучения. Однако такие приложения уступают в производительности нативным и кросс-платформенным аналогам, а также имеют ограниченный доступ к аппаратным функциям устройства. Гибридный подход часто применяется для создания MVP или внутренних корпоративных инструментов.

Progressive Web Apps (PWA): это веб-приложения, расширенные функциями нативных приложений, такими как офлайн-режим, push-уведомления и установка на домашний экран. Как отмечает М. Джексон, «Использование современных фреймворков позволяет создавать веб-приложения, которые по функциональности и удобству приближаются к нативным приложениям, обеспечивая кросс-платформенность и снижение затрат на разработку и поддержку» [6]. Технологии вроде Next.js, Angular или Gatsby позволяют реализовать PWA, обеспечивая кросс-платформенность и низкие затраты на поддержку. Однако на iOS некоторые функции, например, push-уведомления, работают ограниченно, а эффективность приложения зависит от браузера и качества интернет-соединения.

Low-Code/No-Code системы, такие как AppSheet или OutSystems, ориентированы на визуальную разработку приложений без глубокого погружения в программирование. Они подходят для быстрого прототипирования или создания простых бизнес-решений, но обладают ограниченной гибкостью и сложны для масштабирования. Такие системы актуальны для задач с минимальной кастомизацией, где скорость разработки

приоритетнее сложной функциональности. Давайте их сравним, для этого была подготовлена таблица 1.

Таблица 1 – Сравнительный анализ видов мобильных приложений

Критерий	Нативные	Кроссплатформенные	Гибридные	PWA	Low-Code
Производительность	Очень высокая	Высокая	Средняя	Средняя	Низкая
Кроссплатформенность	Нет	Высокая	Высокая	Высокая	Высокая
Стоимость разработки	Высокая	Средняя	Низкая	Низкая	Средняя
Доступ к функциям ОС	Полный доступ	Низкий	Ограниченный доступ	Ограниченный доступ	Минимальный доступ
Офлайн-режим	Полный доступ	Частичная поддержка	Отсутствует	Полная поддержка	Частичная поддержка

На основе данных из таблицы я выбрал PWA (Progressive Web Apps) как платформу для разработки моего проекта. Это решение обладает несколькими ключевыми преимуществами, которые соответствуют моим требованиям.

Во-первых, PWA обеспечивает высокий уровень кроссплатформенности. Приложения, созданные с использованием этой технологии, могут работать на любом устройстве с браузером, что исключает необходимость разработки отдельных версий для разных операционных систем. Это значительно сокращает время и затраты на разработку и поддержку.

Во-вторых, простота использования и развертывания PWA позволяет быстро адаптироваться к изменениям и внедрять обновления. Это особенно важно в динамичной среде, где требуется оперативное реагирование на обратную связь пользователей и изменения в функциональности.

Кроме того, PWA поддерживает офлайн-режим и может использовать такие функции, как push-уведомления и установка на главный экран устройства, что приближает пользовательский опыт к нативным

приложениям. Это делает PWA привлекательным выбором для создания приложений с широким охватом аудитории, особенно в условиях ограниченного доступа к сети.

Таким образом, выбор PWA обоснован необходимостью создания доступного, гибкого и экономичного решения, которое сможет удовлетворить потребности пользователей и обеспечит широкий охват на различных устройствах.

1.3 Выбор среды разработки

Среда разработки является инструментом в руках разработчика, она позволяет увеличить скорость, качество и безопасность приложений. Конкретно для разработки веб-приложений существует множество сред разработки, каждая из которых имеет свои особенности, предназначение, преимущества и недостатки. Как отмечает Д. Браун, «Использование подходящих инструментов разработки является ключевым фактором для достижения высокой производительности, качества и масштабируемости веб-приложений» [1]. Рассмотрим несколько популярных инструментов, включая Visual Studio Code, и проведем их сравнение.

JetBrains WebStorm: мощная среда разработки, выпущенная в 2010 году, специально предназначенная для JavaScript и связанных технологий. WebStorm предлагает широкий набор функций, включая интеллектуальное автодополнение кода, встроенную отладку и поддержку интеграции с популярными фреймворками и библиотеками. Однако WebStorm является платным продуктом, и его стоимость может стать препятствием для некоторых разработчиков или небольших команд.

Visual Studio Code (VS Code): редактор кода от Microsoft, выпущенный в 2015 году. VS Code быстро стал одним из самых популярных инструментов среди разработчиков благодаря своей бесплатной модели распространения, легкости и богатому набору функций. Как указано в официальной

документации, «Visual Studio Code сочетает в себе простоту редактора исходного кода с мощными инструментами разработчика, такими как отладка, встроенная поддержка Git и богатая экосистема расширений» [25]. Он поддерживает множество расширений для работы с различными языками и технологиями, включая PWA. VS Code предлагает интеграцию с системами контроля версий, такими как Git, и поддерживает отладку, что делает его мощным инструментом для разработки.

Xamarin: среда разработки, созданная в 2011 году и позже приобретенная Microsoft. Она позволяет разрабатывать кроссплатформенные мобильные приложения с использованием C# и .NET, предоставляя доступ к нативным API платформ iOS и Android. Основным преимуществом Xamarin является возможность повторного использования значительной части кода для разных платформ, что сокращает время и затраты на разработку. Однако, несмотря на свои достоинства, Xamarin может создавать более тяжелые приложения по сравнению с нативными аналогами, что может повлиять на производительность и размер приложения.

Eclipse: одна из старейших интегрированных сред разработки, выпущенная в 2001 году. Она изначально создавалась для разработки на Java, но благодаря своей модульной архитектуре поддерживает множество языков программирования и технологий через плагины. Eclipse широко используется в разработке Android-приложений, особенно до появления Android Studio. Основные плюсы Eclipse его высокая настраиваемость и обширная экосистема плагинов. Однако по сравнению с более современными средами, такими как VS Code или Android Studio, Eclipse может показаться менее интуитивным и требовать больше времени на настройку. Несмотря на это, данная среда по-прежнему остается востребованной в проектах, где важна стабильность и гибкая конфигурация рабочего окружения, также он поддерживает не только приложения на Java, но и на других языках.

Для более удобного сравнения представлена таблица 2.

Таблица 2 – Анализ сред разработки под ОС Android

Среда разработки	Основное предназначение	Преимущества	Недостатки
Visual Studio Code	Универсальная среда для веб-разработки	Бесплатная, легкая, поддержка множества расширений, интеграция с Git, высокая производительность	Может требовать настройки для использования всех функций
JetBrains WebStorm	Специализированная среда для JavaScript	Интеллектуальное автодополнение, встроенная отладка, поддержка фреймворков	Платная, что может быть ограничением для некоторых команд
Xamarin	Кроссплатформенная разработка мобильных приложений	Возможность повторного использования кода, доступ к нативным API	Более тяжелые приложения, может влиять на производительность
Eclipse	Разработка на Java и других языках через плагины	Высокая настраиваемость, обширная экосистема плагинов	Менее интуитивный интерфейс, требует больше времени на настройку

Сравнивая эти среды разработки, можно отметить, что WebStorm предлагает оптимальное сочетание функциональности, производительности и доступности. Как отмечает Р. Хартман, «Выбор подходящего инструмента разработки должен основываться на его функциональности, производительности и способности легко интегрироваться в существующую инфраструктуру проекта» [20]. WebStorm легок и быстр, что позволяет эффективно работать даже на менее мощных машинах. Богатая экосистема расширений делает его универсальным инструментом для различных задач. Кроме того, интеграция с Git и возможность настройки под специфические нужды проекта делают WebStorm удобным решением для разработчиков обеспечивающим безопасность и стабильность.

1.4 Особенности и специфика разработки

Разработка прогрессивного веб-приложения (PWA) имеет ряд особенностей, связанных как с выбором технологий, так и с требованиями к функциональности. Как отмечает Л. Марк, «При разработке современных веб-приложений важно учитывать не только выбор подходящих технологий, но и требования к функциональности и удобству пользователей» [13].

Например, кэширование играет ключевую роль в поддержании офлайн-работы. С помощью Cache API приложение кэширует сетевые запросы и их ответы, что позволяет загружать ресурсы из кэша и обеспечивать доступность приложения без подключения к интернету. Service Workers, работающие в фоновом режиме, управляют сетевыми запросами и кэшированием, перехватывая их и обслуживая из кэша в случае отсутствия соединения. Это также позволяет обновлять кэшированные ресурсы, когда соединение восстанавливается, обеспечивая пользователям доступ к актуальным данным.

Фоновая синхронизация является ещё одной важной особенностью, позволяющей откладывать сетевые операции до момента восстановления интернет-соединения. Это означает, что изменения, внесённые в офлайн-режиме, могут быть автоматически отправлены на сервер, как только подключение будет восстановлено, что гарантирует целостность данных. При этом важно предусмотреть механизмы разрешения конфликтов, которые могут возникать при одновременном изменении одних и тех же данных разными пользователями или устройствами. Это может включать в себя использование версионности данных или разработку стратегий слияния изменений. Все эти технологии делают приложение более надежным и удобным для пользователей, обеспечивая его функциональность в любое время и в любом месте.

Также автоматические обновления позволяют пользователям всегда иметь доступ к последней версии приложения без необходимости вручную обновлять его, как это бывает с традиционными нативными приложениями.

Когда пользователь впервые посещает PWA, Service Worker устанавливается в фоновом режиме. Этот Service Worker отвечает за кэширование необходимых ресурсов приложения, таких как HTML, CSS, JavaScript и изображения. Впоследствии, при каждом посещении приложения, Service Worker проверяет наличие обновлений на сервере. Если он обнаруживает изменения в ресурсах, он автоматически загружает и кэширует новые версии, чтобы они были готовы к использованию при следующем запуске приложения. Этот процесс обновления обычно происходит незаметно для пользователя, обеспечивая плавный и непрерывный опыт. Однако разработчики могут настроить PWA так, чтобы уведомлять пользователей о наличии обновлений и предлагать перезагрузить приложение для применения последних изменений.

Ограниченный доступ к нативным функциям PWA ограничены по сравнению с нативными приложениями. Например, в фоновом режиме приложение будет работать только при нахождении в списках открытых приложений. Однако они могут использовать такие функции, как камера, геолокация и push-уведомления, если пользователь дает соответствующее разрешение.

Процесс установки прогрессивных веб-приложений (PWA) на устройство позволяет пользователям добавлять их на домашний экран, как нативные приложения. После установки PWA запускается в отдельном окне без элементов браузера, обеспечивая более нативный пользовательский опыт. Удаление PWA с устройства просто и похоже на удаление любого другого приложения. Весь цикл изображен на рисунке 1.

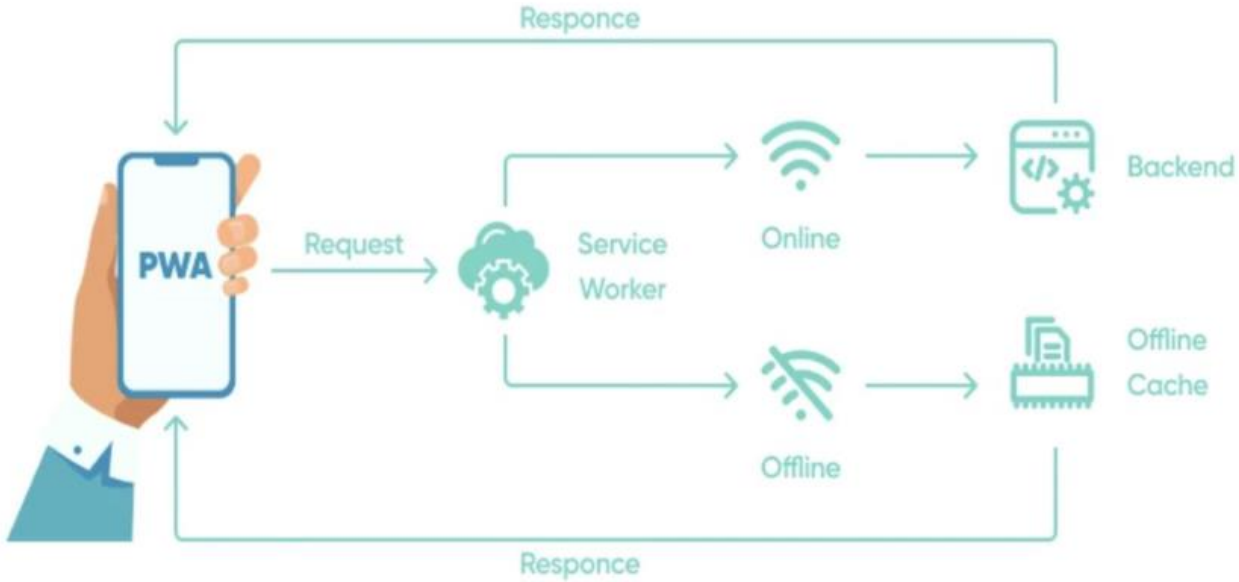

Рисунок 1 – Представление кэширования

Жизненный цикл прогрессивного веб-приложения (PWA) начинается с его открытия пользователем через браузер или иконку на домашнем экране. При запуске загружается стартовая страница, после чего приложение переходит в активный режим использования. Внутренние ссылки, например переход между разделами, обрабатываются без полной перезагрузки страницы, что обеспечивает плавный пользовательский опыт. Внешние ссылки, особенно те, что открываются в новых вкладках, могут временно переключать внимание пользователя, но PWA продолжает работать в фоновом режиме благодаря Service Workers, отвечающим за кэширование данных и офлайн-функциональность.

Работа приложения может быть прервана системными событиями или действиями пользователя. Например, блокировка телефона, входящий звонок или системные уведомления временно приостанавливают выполнение задач. Если пользователь закрывает браузер или переключается на другое приложение, PWA переходит в фоновый режим. Однако через примерно 5 секунд неактивности система, руководствуясь политиками энергосбережения (например, RMSI), может заморозить или завершить процесс приложения. Это

приводит к остановке фоновых задач, таких как синхронизация данных или обновление кэша.

Возврат пользователя к приложению происходит в нескольких сценариях: разблокировка телефона, завершение звонка или закрытие уведомлений. Если процесс PWA не был завершен, приложение возобновляет работу с последнего состояния. В случае завершения процесса система перезагружает приложение, восстанавливая данные из кэша или загружая актуальную версию при наличии интернета. Повторный запуск через иконку на домашнем экране или вкладку браузера также инициирует загрузку, при этом офлайн-режим обеспечивает доступ к ранее сохраненным данным.

Завершение работы происходит, когда пользователь явно закрывает приложение через меню многозадачности или систему принудительно освобождает ресурсы. На этом этапе все фоновые процессы останавливаются, а данные сохраняются для последующего использования. Процесс целиком отображен на рисунке 2.

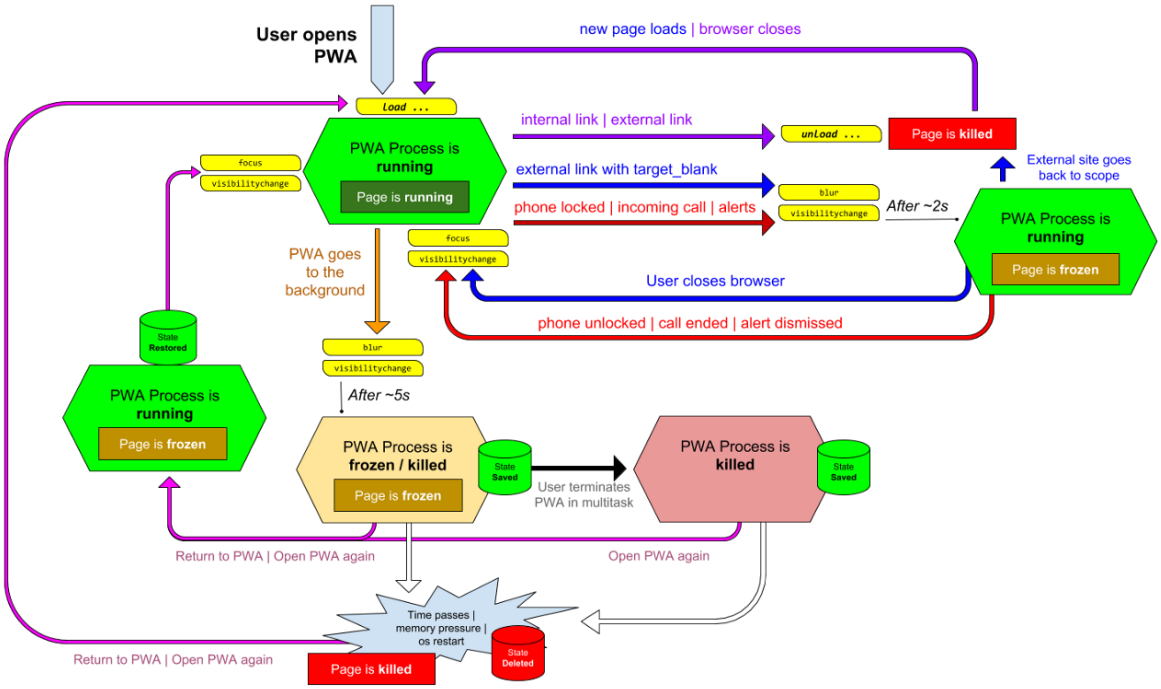

Рисунок 2 – Схема жизненного цикла приложения PWA

Вывод по 1 разделу.

В первом разделе мы подробно рассмотрели концепцию разработки приложения для управления сотрудниками компании с использованием прогрессивного веб-приложения (PWA) на базе Next.js. Мы определили актуальность автоматизации процессов управления персоналом, таких как прием на работу, увольнение и анализ рабочего времени, в контексте современного бизнеса. Были изучены существующие решения на рынке, их преимущества и недостатки, что позволило выделить ключевые функции и особенности, которые необходимо реализовать в нашем приложении.

Анализ платформы PWA показал, что она предоставляет значительные преимущества, включая возможность работы в офлайн-режиме и установку на устройства пользователей, что делает ее идеальной для создания доступных и эффективных приложений. Выбор Next.js обоснован его способностью обеспечивать высокую производительность и SEO-оптимизацию благодаря серверному рендерингу и статической генерации страниц. Среда разработки Visual Studio Code была выбрана за ее легкость, мощный набор инструментов и поддержку множества расширений, которые облегчают процесс разработки. Для управления состоянием приложения был интегрирован Redux Toolkit, обеспечивающий предсказуемость данных и упрощённую архитектуру хранилища.

Мы заложили теоретическую основу для дальнейшей разработки приложения, определили технологическую платформу и инструменты, которые будут использоваться, и сформировали четкое понимание задач, которые предстоит решить в процессе реализации проекта. Эти выводы создают прочную основу для перехода к практической части работы, где будет рассмотрена непосредственная разработка и тестирование приложения.

2 Логическое моделирование и проектирование системы

2.1 Формирование требований к системе

В процессе разработки приложения для управления сотрудниками компании ключевым этапом является формирование требований к системе. Этот этап играет решающую роль, поскольку от него зависит, насколько эффективно приложение будет удовлетворять потребности бизнеса и пользователей. Как отмечает Дж. Смит, «Эффективное определение требований является ключевым фактором успеха информационной системы, поскольку именно на этом этапе закладывается основа для удовлетворения потребностей пользователей и достижения целей бизнеса» [17]. Давайте применим модель FURPS+ к процессу формирования требований для приложения управления сотрудниками компании. Конечно, давайте добавим английские термины к каждому аспекту модели FURPS+.

Функциональность (Functionality) приложения должна включать основные процессы управления персоналом, такие как прием на работу (hiring), увольнение сотрудников (termination) и анализ рабочего времени (time tracking). Эти функции должны быть реализованы через интуитивно понятный интерфейс (user-friendly interface), который позволяет администраторам легко добавлять, изменять и удалять данные о сотрудниках. Анализ рабочего времени должен поддерживать возможность классификации по различным категориям (categorical analysis), что поможет в оптимизации использования трудовых ресурсов.

Удобство использования (Usability) подразумевает, что интерфейс приложения должен быть простым и интуитивно понятным, обеспечивая пользователям легкий доступ к основным функциям.

Надежность (Reliability) приложения должна обеспечивать минимизацию ошибок и сбоев (minimization of errors and failures), что особенно важно для защиты и обработки данных сотрудников. Это требует наличия

механизмов резервного копирования данных (data backup) и восстановления после сбоев (disaster recovery), чтобы избежать потери информации.

Производительность (Performance) важна для обеспечения быстрой загрузки страниц (fast page loading) и минимального времени отклика (minimal response time). Приложение должно быть способно обрабатывать запросы большого количества пользователей и данных без снижения скорости работы, что критично для эффективного управления персоналом.

Поддержка (supportability) приложения должна быть лёгкой и масштабируемой, предусматривая оперативное добавление новых функций и интеграцию с другими корпоративными системами; ведь, как подчёркивают К. Вигерс и Дж. Битти, «поддерживаемость системы определяется тем, насколько легко её можно модифицировать и расширять по мере появления новых бизнес-потребностей» [4].

В рамках дополнительного аспекта безопасности (Security) данных сотрудников необходимо внедрить механизмы аутентификации (authentication) и авторизации (authorization) для защиты информации от несанкционированного доступа. Как подчёркивает Т. Вебер, «Безопасность веб-приложения предполагает обязательное внедрение строгих механизмов аутентификации и авторизации, чтобы защитить данные от несанкционированного доступа» [3].

Использование модели FURPS+ предоставляет возможность тщательно структурировать и анализировать требования к приложению, обеспечивая всесторонний подход (comprehensive approach) к его разработке и внедрению, что в конечном итоге способствует созданию качественного продукта, удовлетворяющего потребности бизнеса и пользователей.

В модели FURPS+ знак "+" обозначает дополнительные характеристики, которые не входят в основные пять категорий (Functionality, Usability, Reliability, Performance, Supportability). Эти характеристики могут охватывать различные аспекты, такие как:

– security (безопасность) включает в себя меры по обеспечению

безопасности и конфиденциальности информации;

- compatibility (совместимость) возможность приложения функционировать на разнообразных устройствах;
- maintainability (поддерживаемость) простота, с которой приложение может быть обновлено, исправлено и дополнено новым функционалом;
- scalability (масштабируемость) способность приложения обрабатывать увеличивающееся количество пользователей или данных без ухудшения производительностиж
- localization (локализация) возможность адаптации приложения для разных языков и регионов;
- regulatory compliance (соответствие нормативным требованиям) учет и соблюдение правовых и нормативных требований, связанных с использованием приложения.

Применение модели FURPS+ позволяет детально структурировать и анализировать требования к приложению, обеспечивая всесторонний подход к его разработке и внедрению, что в конечном итоге способствует созданию качественного продукта, удовлетворяющего потребности бизнеса и пользователей.

В данный момент требования к программному обеспечению обычно классифицируют на два основных типа.

Функциональные требования описывают набор задач и возможностей, которые система должна реализовать. Они определяют, какие действия должно выполнять приложение для удовлетворения потребностей пользователей и бизнес-целей. Эти требования включают в себя описание того, как система должна функционировать в различных ситуациях, спецификацию входных и выходных данных, а также взаимодействие с другими системами и пользователями.

Нефункциональные требования, в свою очередь, описывают характеристики и качества системы, которые не связаны непосредственно с ее

функциональностью.

Требования к приложению представлены в таблицах 3 и 4.

Таблица 3 – Функциональные требования к приложению

Тип требований	Описание
Управление сотрудниками	Возможность добавления, редактирования и удаления данных о сотрудниках.
Отчеты	Генерация отчетов по рабочему времени, включая классификацию по категориям.
Интеграция	Интеграция с существующими системами управления персоналом и другими корпоративными инструментами.
Авторизация	Система регистрации и авторизации пользователей для управления доступом к функциям приложения.

Таблица 4 – Нефункциональные требования к приложению

Тип требований	Описание
Производительность	Высокая производительность при обработке большого объема данных и минимальное время отклика.
Надежность	Устойчивость к сбоям и наличие механизмов резервного копирования данных и восстановления после сбоев.
Безопасность	Защита данных от несанкционированного доступа, включая шифрование и аутентификацию пользователей.
Удобство использования	Интуитивно понятный интерфейс с логичной навигацией и ясными инструкциями.
Масштабируемость	Возможность обработки увеличивающегося количества пользователей и данных без ухудшения производительности.
Поддерживаемость	Легкость обновления и расширения функциональности, включая наличие документации и структурированного кода.
Совместимость	Работа на различных устройствах и браузерах для обеспечения широкой доступности.

Эти таблицы предлагают структурированный обзор требований, необходимых для разработки приложения, чтобы оно соответствовало ожиданиям бизнеса и пользователей.

2.2 Обзор и анализ существующих мобильных приложений по управлению сотрудниками

Рынок мобильных приложений для управления сотрудниками активно развивается, предлагая решения для различных бизнес-задач: от учета рабочего времени до комплексного HR-менеджмента. Ниже представлен анализ ключевых категорий и популярных приложений.

Одним из популярных решений на рынке является приложение BambooHR. Это облачное приложение предлагает широкий спектр функций для управления персоналом, включая учет рабочего времени, управление отпусками и кадрами, а также функции для оценки производительности. Преимуществом BambooHR является его интуитивно понятный интерфейс и возможность интеграции с другими системами.

Еще одно известное приложение – Zenefits. Оно предоставляет полный набор инструментов для управления персоналом, включая расчеты заработной платы, управление льготами и соответствие нормативным требованиям. Zenefits интегрируется с множеством других бизнес-приложений, что делает его удобным для комплексного управления бизнес-процессами.

Gusto – это еще одно популярное приложение, ориентированное на малый и средний бизнес. Оно предлагает удобные инструменты для управления зарплатами, налогами и льготами сотрудников.

Sapling – это приложение для управления персоналом, которое фокусируется на адаптации новых сотрудников и управлении жизненным циклом сотрудников в компании. Оно предлагает функции для автоматизации процессов найма, обучения и оценки производительности.

Давайте представим анализ популярных приложений для управления сотрудниками на основе установленных критериев. Результаты анализа представлены в таблице 5.

Таблица 5 – Сравнение мобильных обучающих систем

Критерий	BambooHR	Zenefits	Gusto	Sapling
Функциональность	+	+	+	+
Удобство использования	+	-	+	+
Надежность	+	+	+	+
Производительность	+	+	-	+
Поддержка	+	+	+	+
Безопасность	+	+	+	+
Масштабируемость	+	+	-	+
Стоимость	-	-	+	-

Изучив таблицу, можно заключить, что большинство мобильных приложений в значительной мере удовлетворяют предъявленным требованиям. Однако значительные ограничения наблюдаются в двух областях: наличие бесплатной версии программного обеспечения и поддержка работы в офлайн-режиме. Анализ показал, что многие из исследованных платформ не полностью соответствуют всем критериям.

В связи с этим было принято решение разработать собственное мобильное приложение, которое будет включать все необходимые функции. Это приложение сможет работать в изолированных сетях и предоставлять офлайн-доступ, что позволит пользователям продолжать обучение без зависимости от интернет-соединения. Кроме того, приложение будет постоянно обновляться и адаптироваться под изменяющиеся нужды пользователей, обеспечивая долгосрочную актуальность и пользу.

2.3 Выбор технологии логического моделирования мобильного приложения

Логическое моделирование – это процесс создания абстрактного представления системы, которое описывает, как она будет функционировать, без учета технических деталей реализации. Этот подход используется для понимания и документирования структуры данных, процессов и взаимодействий внутри системы. Целью логического моделирования является создание четкой и понятной модели, которая может быть использована для разработки, анализа и улучшения системы.

В контексте разработки программного обеспечения логическое моделирование помогает разработчикам и аналитикам определить требования и ожидания к системе на более высоком уровне. Как подчёркивает Р. С. Прессман, «создание абстрактной модели позволяет зафиксировать суть требований, прежде чем переходить к конкретным техническим решениям» [16]. Это важный этап, который предшествует физическому моделированию, где внимание уделяется уже конкретным технологиям и архитектуре.

Некоторые из ключевых аспектов логического моделирования включают определение сущностей и их взаимосвязей. Это включает в себя идентификацию основных объектов, с которыми будет работать система, и описание их связей. Например, в системе управления персоналом сущностями могут быть сотрудники, отделы и проекты. «Объектно-ориентированный подход помогает управлять сложностью путём декомпозиции системы на объекты и взаимосвязи между ними» [5]. Например, в системе управления персоналом сущностями могут быть сотрудники, отделы и проекты.

Логическое моделирование также помогает описывать процессы и потоки данных, визуализируя, как информация передаётся и обрабатывается внутри системы, а вместе с тем фиксируя поддерживаемые бизнес-процессы. Такая наглядность упрощает восприятие архитектуры всеми участниками

проекта; как справедливо замечает С. Круг, «если вы заставляете пользователя задумываться о том, что происходит, значит вы уже сделали что-то не так» [9].

Создание диаграмм и моделей является ещё одним важным аспектом проектирования. Как подчёркивает К. Ларман, визуальные абстракции позволяют «делать явными ключевые концепции системы и служат общим языком для команды» [10]. Различные диаграммы, такие как диаграммы классов UML или ER-диаграммы, используются для представления структуры и поведения системы, помогая лучше понять сложные решения и поддерживать согласованность в коллективе разработчиков.

Логическое моделирование является важным шагом в разработке сложных систем, так как оно позволяет выявить потенциальные проблемы и улучшить дизайн системы до начала ее реализации. Это также облегчает коммуникацию между участниками проекта, включая разработчиков, аналитиков и заинтересованные стороны, обеспечивая общее понимание того, как должна работать система.

Выбор технологии логического моделирования мобильного приложения играет ключевую роль в разработке, так как это определяет, как данные и процессы будут структурированы и взаимодействовать внутри системы. Логическое моделирование помогает создать четкую архитектуру приложения, что упрощает разработку и последующую поддержку. Рассмотрим несколько подходов и технологий, которые можно использовать для логического моделирования мобильных приложений.

Unified Modeling Language (UML) является одной из наиболее популярных технологий для логического моделирования. UML предлагает широкий набор диаграмм, таких как диаграммы классов, диаграммы последовательностей и диаграммы состояний, которые позволяют визуализировать структуру и поведение системы. UML помогает разработчикам четко определить взаимодействие между компонентами приложения и может быть особенно полезен для сложных систем.

Entity-Relationship (ER) диаграммы – это еще один подход, который фокусируется на моделировании данных и их взаимосвязей. ER-диаграммы хорошо подходят для проектирования баз данных, так как они помогают определить, как данные будут храниться и связаны между собой. Как подчёркивает К. Льюис, «Грамотное проектирование интерфейса требует понимания того, как данные представлены и взаимосвязаны, поскольку это напрямую влияет на удобство и эффективность работы пользователя» [11]. Это особенно важно для приложений, где управление данными является критически важным аспектом.

Business Process Model and Notation (BPMN) применяется для визуального представления и описания бизнес-процессов, которые будут автоматизированы в приложении. BPMN помогает разработчикам понять и визуализировать рабочие процессы и взаимодействия, что может быть полезно для приложений, связанных с управлением бизнес-процессами.

Для мобильных приложений, которые требуют гибкости и быстрого прототипирования, лучше все подходит UML диаграмма, потому что диаграммы вариантов использования позволяют в сжатой и наглядной форме зафиксировать все пользовательские сценарии без лишних технических деталей.

В контексте системы управления сотрудниками три ключевые роли, на которые следует обратить внимание, играют важную роль в обеспечении эффективности и успешности сервиса.

Есть сотрудники, которые являются основными пользователями системы. Им необходимо предоставить удобный доступ к функциям, таким как просмотр и обновление личной информации, запрос отпусков, а также отслеживание своего рабочего времени и задач. Удобный и интуитивно понятный интерфейс важен для повышения их удовлетворенности и производительности.

Также администраторы, которые используют систему для управления персоналом. Их задачи включают в себя прием и увольнение сотрудников,

назначение задач, анализ рабочего времени и производительности. Для них система должна предоставлять инструменты для эффективного управления и мониторинга, а также возможность создания отчетов и аналитики для принятия обоснованных решений.

За поддержание актуальности кадровых данных и соблюдение нормативных требований отвечает HR-специалист. Ему необходим доступ к функционалу для ведения учета кадровых данных, просмотра учета времени, а также для выполнения других административных задач. Система должна обеспечивать надежную защиту данных и возможность интеграции с другими корпоративными системами.

Сводный перечень всех сценариев, распределённых между сотрудником, администратором и HR-специалистом приведены в таблице 6.

Таблица 6 – Описание вариантов использования

Прецедент	Краткое описание
Добавление нового сотрудника	HR регистрирует нового сотрудника в системе
Удаление сотрудника	HR удаляет данные сотрудника после увольнения
Редактирование сотрудника	HR обновляет свою личную информацию в профиле
Просмотр отчета времени	HR смотрит отчет по времени сотрудника
Создание отчета времени	Сотрудник создает отчет времени
Удаления отчета времени	Сотрудник удаляет отчет времени
Редактирование отчета времени	Сотрудник редактирует отчет времени

Эта таблица служит связующим звеном с диаграммой вариантов использования (рисунок 3).

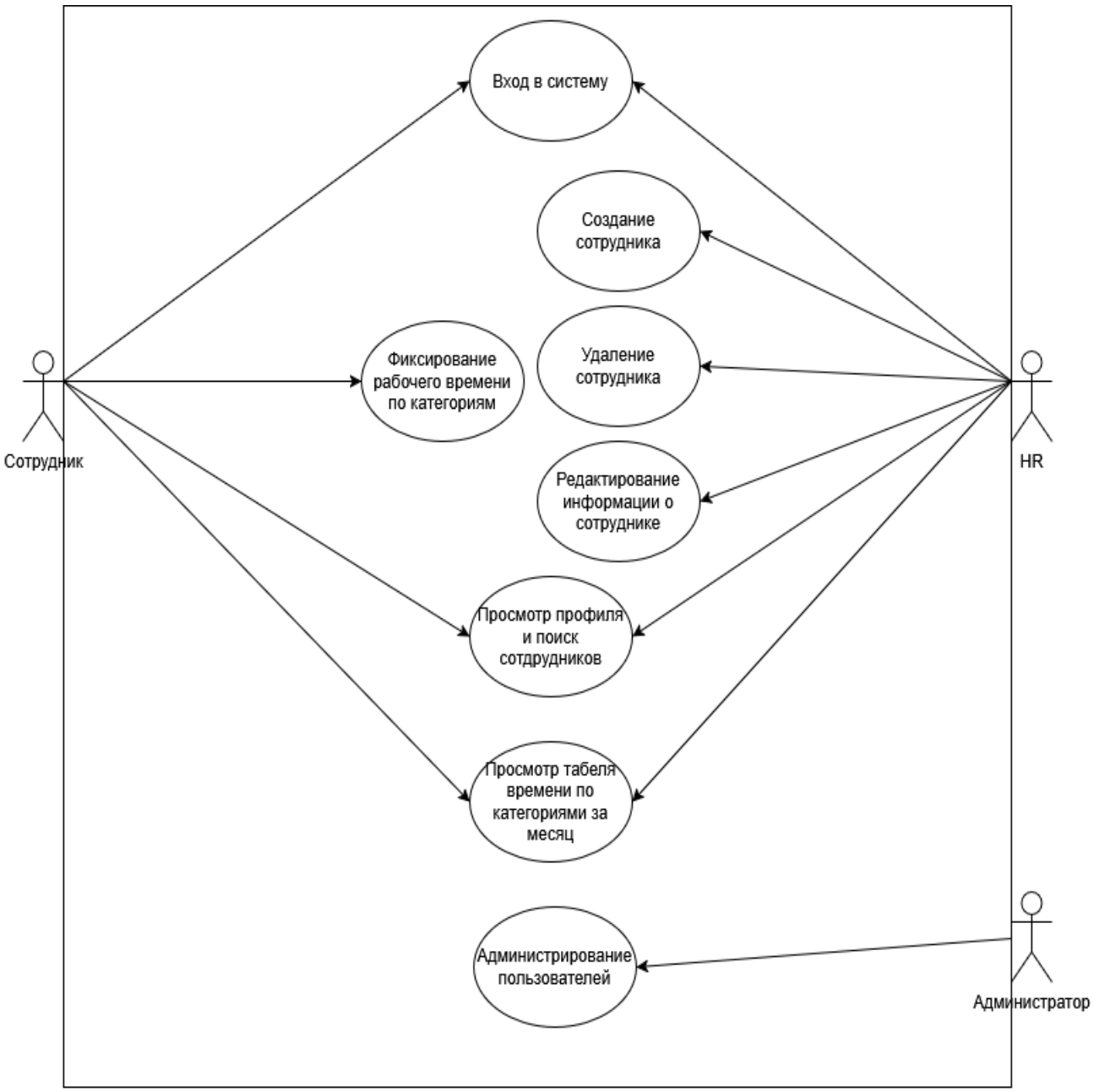

Рисунок 3 – Диаграмма вариантов использования

На диаграмме вариантов использования системы управления сотрудниками представлены основные взаимодействия между пользователями и системой. В верхней части диаграммы расположены актеры, такие как сотрудники, менеджеры, HR-специалисты и администраторы, каждый из которых обозначен специфической иконкой и подписью.

В центре диаграммы находятся овальные фигуры, обозначающие функции системы. К ним относятся «Добавление нового сотрудника»,

«Увольнение сотрудника», «Обновление информации».

Диаграмма предоставляет визуальное представление о том, как различные пользователи взаимодействуют с системой, помогая понять ключевые процессы и роли в управлении сотрудниками.

2.4 Проектирование модели данных

«Чтобы приступить к проектированию модели данных, важно создать как концептуальную, так и логическую модели» [7]. «Концептуальная модель базы данных представляет собой визуальную диаграмму, выполненную с использованием стандартных обозначений, которая детально демонстрирует взаимосвязи между объектами и их характеристиками» [5]. Для начала разработки модели данных необходимо определить ключевые сущности, такие как задачи, рассылка уведомлений и выполненные задачи. Концептуальная схема базы данных представлена ниже (рисунок 4).

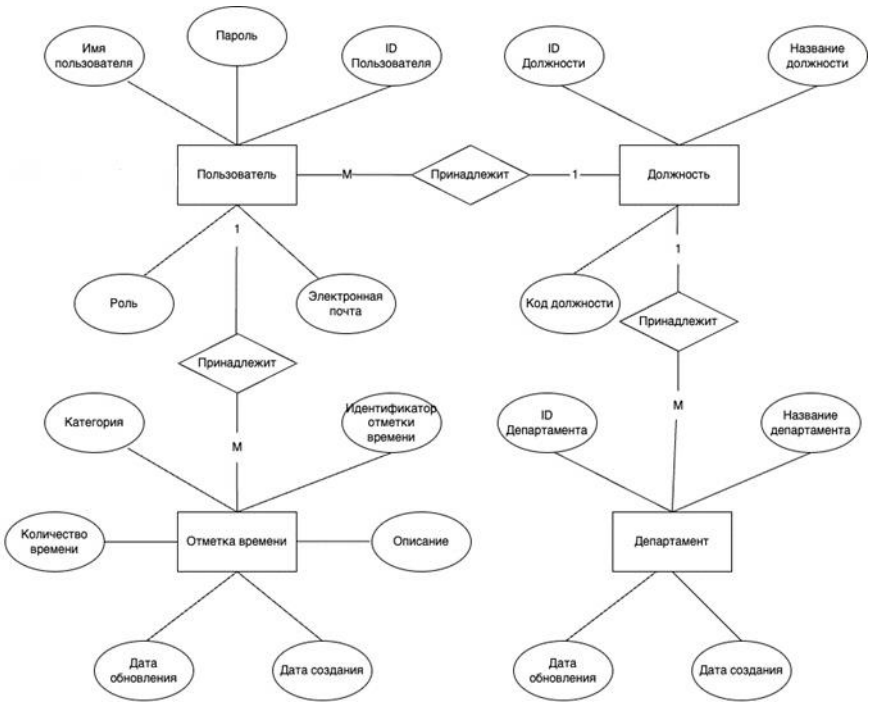

Рисунок 4 – Концептуальная модель данных

Модель включает пять ключевых сущностей: «Пользователь»,

«Должность», «Департамент», «Отметка времени» и «Роль», а также атрибуты и связи между ними, отражающие структуру данных и логику взаимодействия объектов внутри системы.

Сущность «Пользователь» описывает участников системы и содержит такие атрибуты, как идентификатор, имя пользователя, пароль и адрес электронной почты. Каждому пользователю соответствует одна роль, которая определяет его уровень доступа или функциональные обязанности. Пользователь может иметь множество временных отметок, что отражено связью «один ко многим» с сущностью «Отметка времени». Также пользователь относится к конкретной должности, что устанавливает связь с соответствующей сущностью.

Сущность «Должность» содержит информацию о наименовании и коде должности, а также уникальном идентификаторе. Каждая должность может быть закреплена за несколькими пользователями, однако принадлежит только одному департаменту. Эта иерархическая структура отражается в модели посредством связи «многие к одному» между сущностями «Должность» и «Департамент».

Сущность «Департамент» представляет организационные подразделения компании. Она включает идентификатор, название, а также дату создания и дату последнего обновления записи. Данная сущность является родительской по отношению к «Должности» и обеспечивает группировку сотрудников по функциональной принадлежности.

Сущность «Отметка времени» фиксирует действия пользователей во времени. В ее структуру входят такие параметры, как идентификатор отметки, категория, описание, количество затраченного времени, дата создания и дата обновления. Каждая временная отметка привязана к одному пользователю, что отражено соответствующей связью.

Эта концептуальная модель данных предоставляет общее представление о структуре данных и их взаимосвязях в системе управления сотрудниками, что помогает в дальнейшем проектировании и реализации приложения.

2.5 Архитектура программного продукта

«Мобильное приложение будет разработано с учетом архитектуры Architecture Components, которая была анонсирована на конференции Google I/O в 2017 году. Архитектура приложения будет включать в себя новые библиотеки, которые помогут разработать надежное и тестируемое приложение. Этот набор библиотек поможет решить общие проблемы изменения конфигурации, утечки памяти» [21]. «Компоненты архитектуры могут использоваться как в совокупности, так и независимо друг от друга». [8]. Схема Architecture Components представлена ниже (рисунок 5).

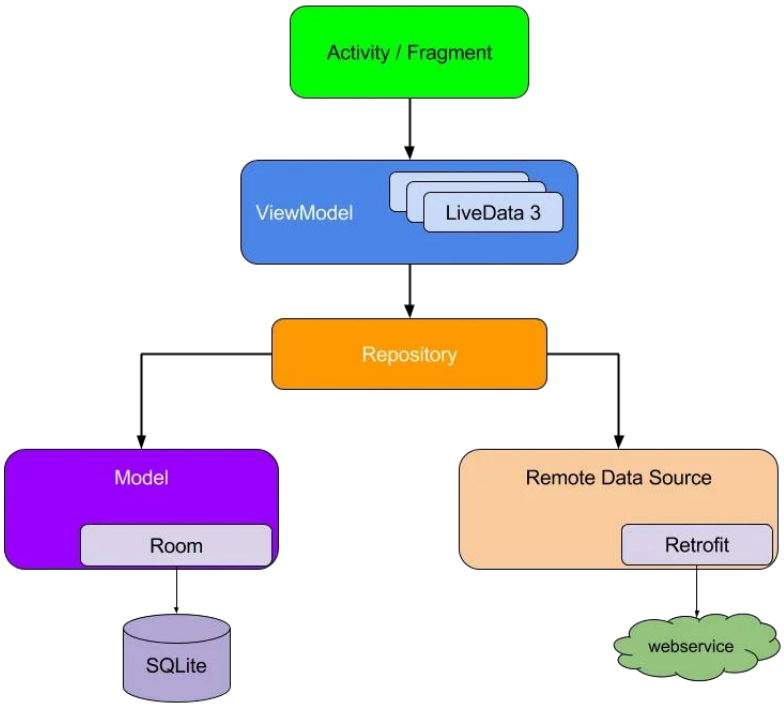

Рисунок 5 – Диаграмма компонентов архитектуры.

После выбора архитектурного решения для мобильного приложения важно перейти к визуализации классов и их связей с помощью модели структуры классов. Этот подход помогает воспроизвести структуру классов PWA приложения (рисунок 6).

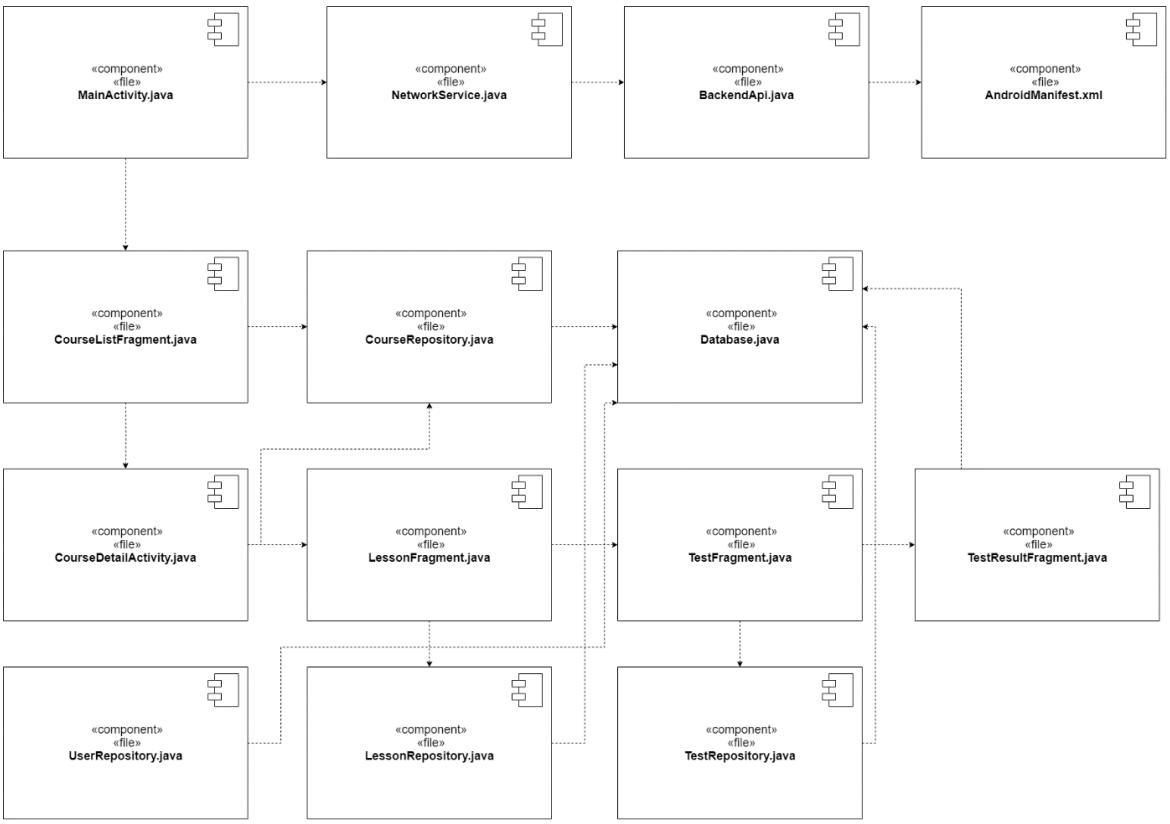

Рисунок 6 – Диаграмма классов PWA приложения

Рисунок 7 иллюстрирует диаграмму развертывания, на которой показан элемент «docker_host», который выполняет запуск и контроль контейнера. Как указано в официальной документации Docker, «Контейнеры Docker – это легковесные и переносимые единицы программного обеспечения, которые упаковывают приложение и его зависимости для удобного и быстрого развертывания в различных средах» [20]. Инфраструктура включает две основные компоненты: «service_net», она обеспечивает коммуникацию между другими такими же сервисами, и «postgres_volume», служит местом для данных БД. Внутри «service_net» существуют три контейнера: «hr-erp-app», он содержит в себе «hr-erp-app.js». Этот контейнер содержит в себе PWA приложение; «hr-erp-service», который содержит «hr-erp-service.jar». Этот контейнер представляет собой RESTful-сервис, обеспечивающий обработку запросов приложения; «postgres», является базой данных Postgres [22].

На правой стороне диаграммы представлена клиентская часть, обозначенная как «Мобильное устройство». Это устройство взаимодействует

с приложением через протокол HTTPS, используя порт 80 для передачи данных. На мобильном устройстве информация отображается в формате HTML, что обеспечивает пользователю интуитивный и удобный доступ к функциональным возможностям приложения. Как отмечает И. Соммервилл, «проектирование пользовательского интерфейса должно быть ориентировано на потребности пользователя и учитывать особенности устройства [18].

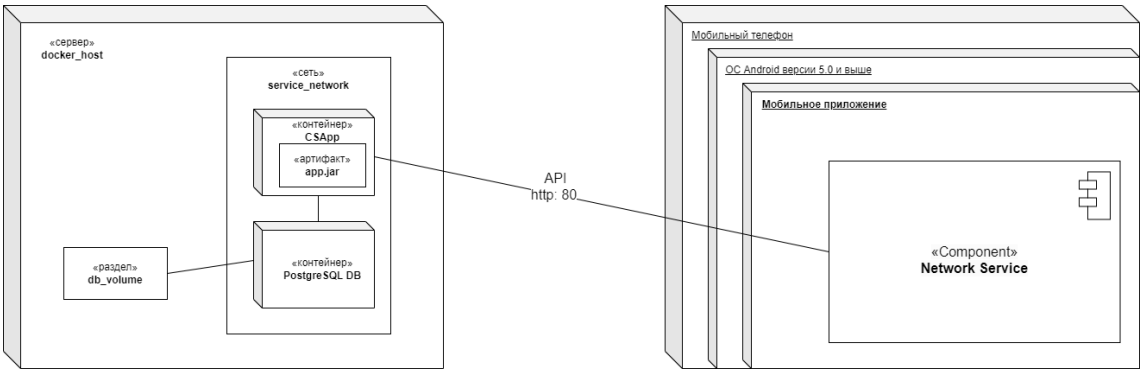

Рисунок 7 – Схема мобильного приложения и серверной части

Завершение проектирования диаграммы развертывания знаменует окончание этапа разработки мобильного приложения, открывая путь к его реализации и последующему тестированию, а также к созданию rest-сервиса. «Паттерны проектирования делают проектные решения явными, а не подразумеваемыми». [19] Уделение внимания всем аспектам, отраженным в диаграмме, является критически важным для снижения рисков возникновения проблем в процессе разработки и для обеспечения надежной работы системы.

Вывод по 2 разделу.

Во втором разделе был проведен всесторонний анализ и проектирование системы управления сотрудниками, что стало важным шагом в разработке приложения. Мы начали с формирования требований к системе, определив ключевые функциональные и нефункциональные аспекты, которые должны быть учтены для удовлетворения потребностей бизнеса и пользователей. Это

включало такие функции, как прием и увольнение сотрудников, а также анализ рабочего времени по категориям.

Далее было уделено внимание выбору технологии логического моделирования, что позволило создать концептуальную модель данных. Эта модель структурировала основные сущности и связи между ними, такие как сотрудники, отделы, задачи и рабочее время, обеспечивая целостность и согласованность данных.

Затем мы рассмотрели проектирование модели данных, что включало определение атрибутов и взаимосвязей между сущностями. Это дало возможность создать прочную основу для хранения и управления данными, необходимыми для работы приложения.

Также была разработана диаграмма развертывания, которая визуализировала, как различные компоненты системы будут взаимодействовать и как они распределены по инфраструктуре. Это позволило учесть все технические детали и подготовить основу для реализации и тестирования приложения.

В результате, второй раздел заложил фундамент для дальнейших этапов разработки, обеспечив четкое понимание структуры и архитектуры системы, что способствует успешной реализации проекта.

3 Реализация многофункционального мобильного обучающего приложения

3.1 Выбор технологии разработки приложения

Для разработки прогрессивного веб-приложения (PWA) было решено использовать «Next.js». Он имеет множество возможностей для разработки современных веб-приложений. Как отмечает А. Петров, «Next.js предоставляет разработчикам мощный набор инструментов, включая серверный рендеринг и статическую генерацию страниц, что положительно сказывается на производительности и SEO веб-приложений» [15]. Благодаря встроенной маршрутизации и возможности создания API-роутов, Next.js обеспечивает разработчикам гибкость в реализации разнообразных функций. Для контроля изменений будем использовать git. Git является системой контроля управления версиями. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

Чтобы протестировать наше приложение, нужно понимать какие браузеры являются популярными, ведь именно ими в большинстве случаев будут пользоваться сотрудники. Статистика использования различных браузеров, актуальная на ноябрь 2024 года, представлена ниже (рисунок 8).

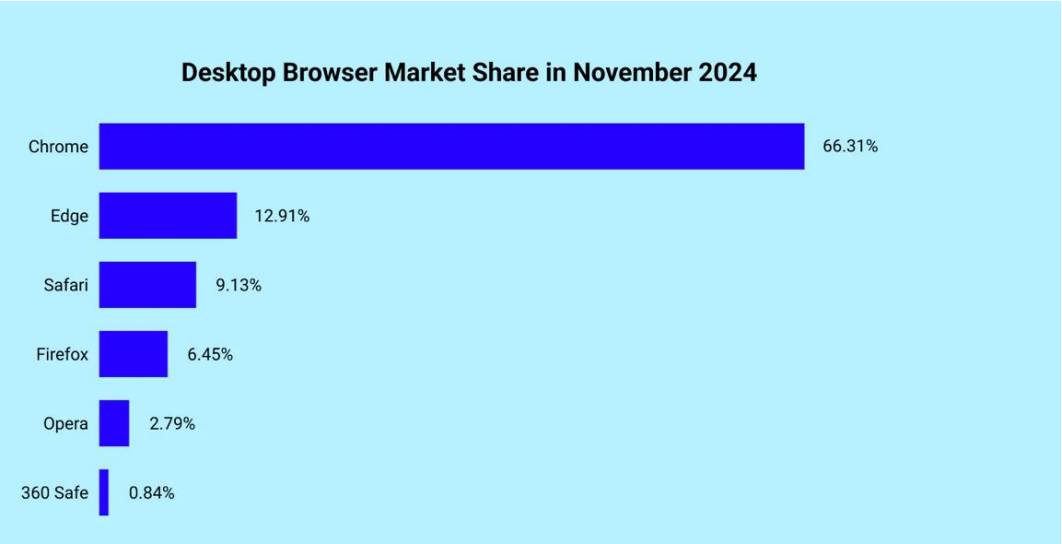

Рисунок 8 – Распределение браузеров по состоянию на ноябрь 2024 года

Система будет адаптирована для обеспечения совместимости и оптимальной работы на различных устройствах и в разных браузерах. Это включает адаптацию интерфейса для корректного отображения на экранах разных размеров, а также оптимизацию производительности для стабильной работы на устройствах с различными техническими характеристиками. Кроме того, система может быть настроена для поддержки нескольких языков, чтобы удовлетворить потребности пользователей из разных регионов.

Также нужны регулярные проверки, которые включают в себя проведение тестов на уязвимости, таких как сканирование на наличие SQL-инъекций и межсайтового скриптинга (XSS). Пентесты (тесты на проникновение) помогают выявить слабые места в системе защиты, имитируя действия злоумышленников. Проведение стресс-тестирования для оценки устойчивости системы к DDoS-атакам. Эти меры тестирования, вкупе с регулярными обновлениями и обучением сотрудников, обеспечивают надежную защиту данных и стабильную работу системы.

Такой подход позволяет гарантировать, что приложение будет работать стабильно и предоставлять качественный пользовательский опыт.

3.2 Создание проекта в Web Storm

Для разработки приложения, на платформе Next.js, надо начать установку необходимых инструментов, таких как Node.js и npm, а также среды разработки, WebStorm. Далее создается новый проект с помощью команды «npx create-next-app». На этапе инициализации проекта задается его название, что является важным шагом, так как имя проекта будет использоваться в конфигурациях и структуре папок. Далее определяются ключевые параметры, такие как тип проекта (например, с использованием TypeScript) и необходимость подключения дополнительных библиотек, таких как Tailwind CSS для стилизации или ESLint для проверки кода. Еще необходимо добавить «manifest.xml», для того чтобы была поддержка PWA приложения.

Как отмечает С. Макконнелл, «эффективный интерфейс начинается с учёта контекста использования и ограничений устройств, на которых он будет работать» [12]. При создании интерфейсов важно учитывать целевые устройства, на которых будет использоваться прогрессивное веб-приложение смартфоны, планшеты, ноутбуки или настольные компьютеры. Выбор устройств влияет на проектирование интерфейса, например на адаптацию макетов под различные размеры экранов и использование медиа-запросов для обеспечения отзывчивости.

Процесс выбора и настройки параметров проекта представлен ниже (рисунок 9).

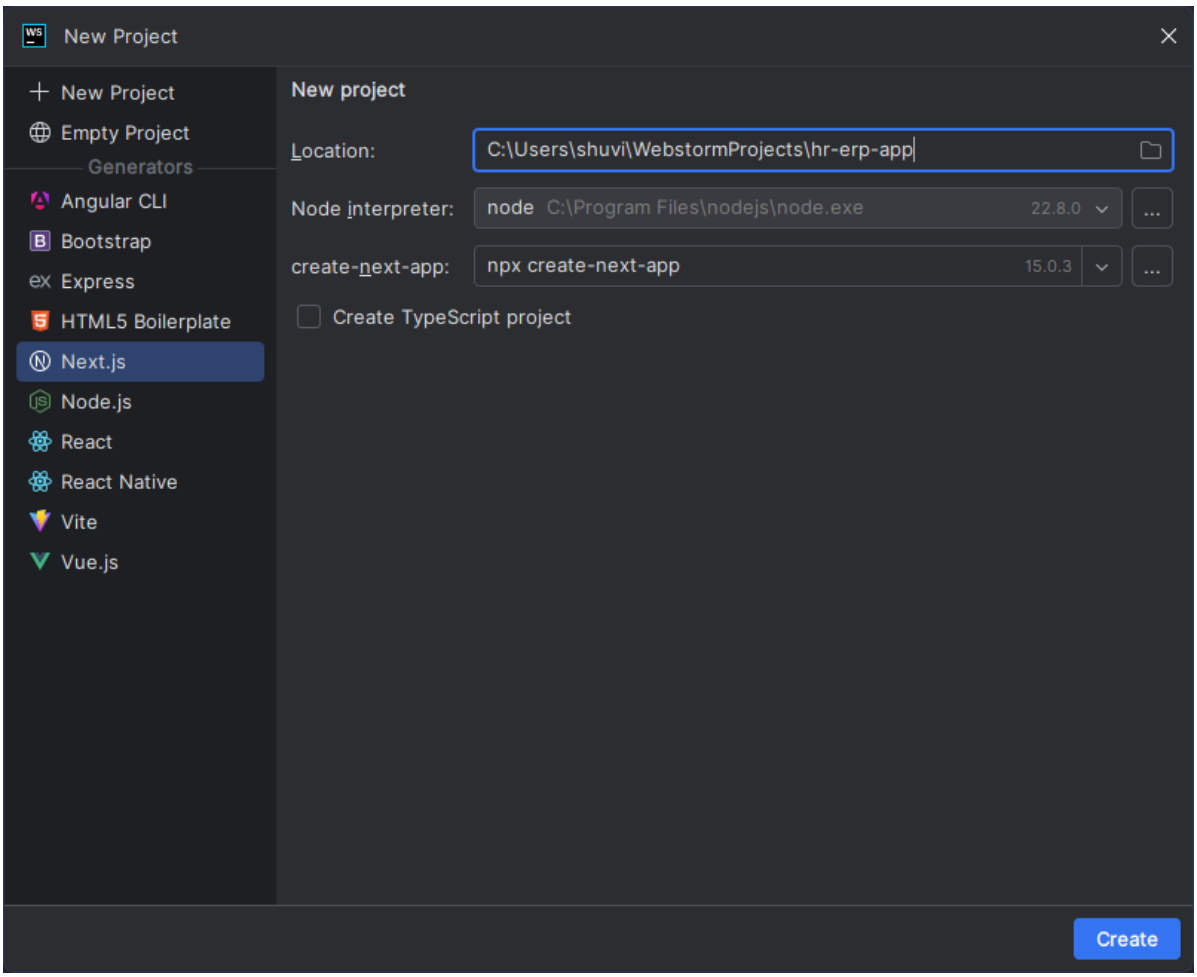

Рисунок 9 – Окно настройки «next.js» приложения

После создания проекта запускается окно разработки приложения (рисунок 10).

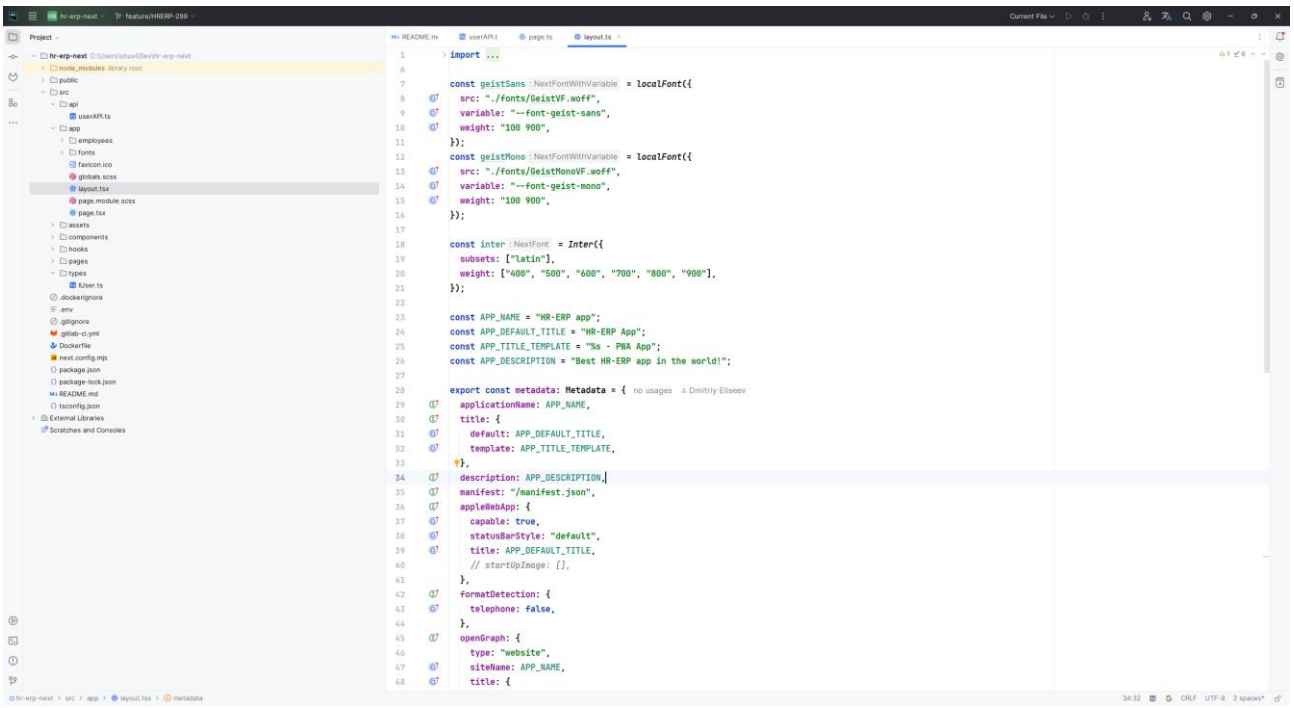

Рисунок 10 – Вид после создания проекта

Слева мы можем увидеть нашу структура проекта и файлы конфигурации (рисунок 11).

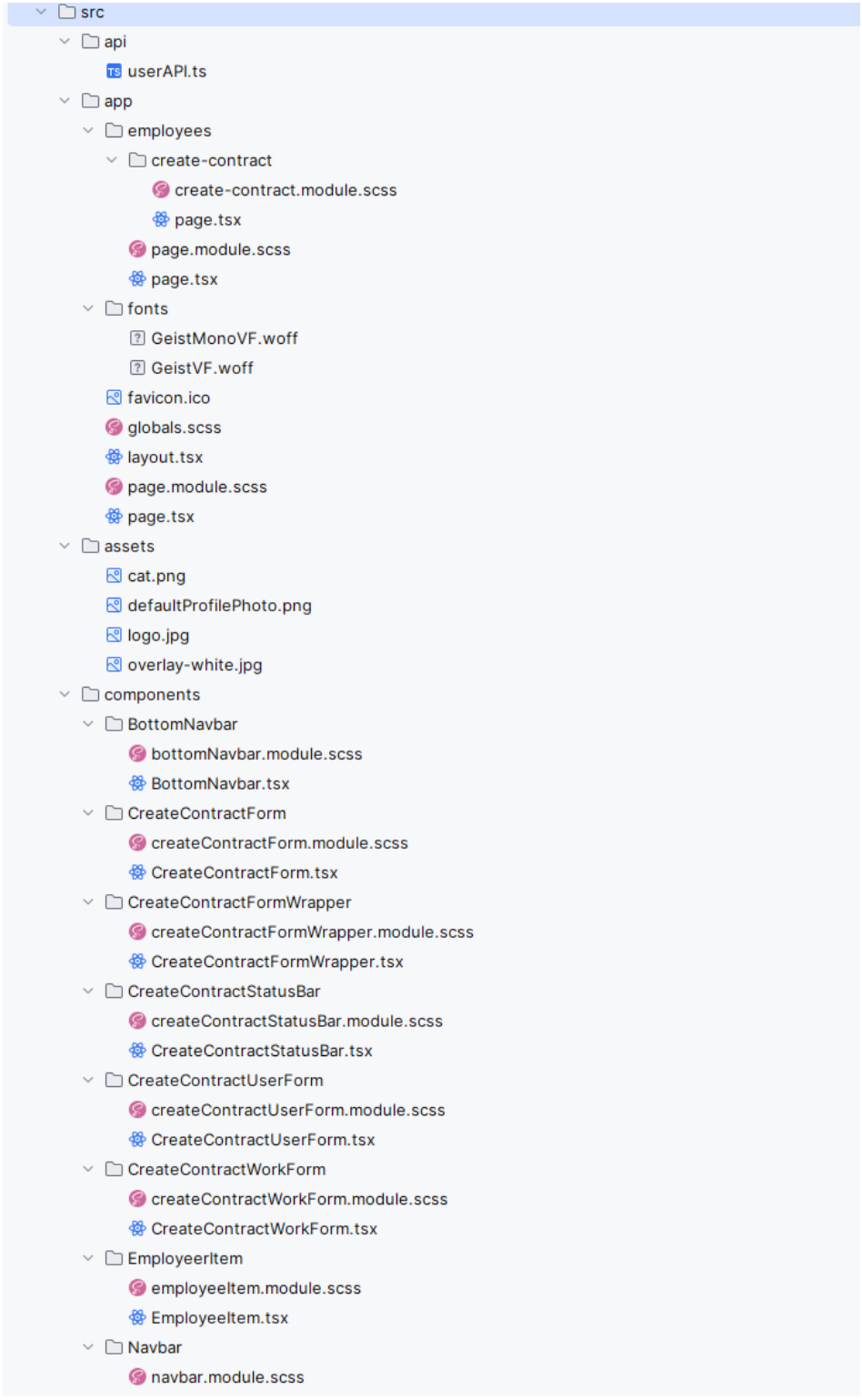

Рисунок 11 – Файлы конфигурации и структура проекта

Основной файл конфигурации next.config.js, который находится в корне проекта. Как указано в официальной документации Next.js, «Файл next.config.js позволяет настраивать приложение Next.js с помощью

расширенных параметров, таких как оптимизация сборки, настройка маршрутизации и интеграция плагинов» [23]. Этот файл содержит настройки для оптимизации сборки, конфигурации плагинов и других параметров, таких как обработка маршрутов или настройка статического контента.

В папке «page» находятся все страницы приложения, где каждая страница соответствует отдельному URL. Например, файл «index.js» будет точкой входа в приложение, отображая главную страницу, а другие файлы в этой директории определяют дополнительные страницы, например, о проекте или контакты.

Ресурсы приложения, такие как изображения, шрифты и стили, хранятся в папке «public». Эта директория используется для хранения статических файлов, которые доступны для клиента. Например, изображения или другие файлы, используемые в интерфейсе, будут храниться в папке «public/images».

Для управления зависимостями и сборки проекта используется файл «package.json», в котором указываются все необходимые библиотеки и инструменты. В «Next.js» также можно использовать файлы конфигурации для настройки различных инструментов и оптимизаций, например, для работы с типами данных или стилизацией с помощью таких библиотек как «Tailwind CSS» или «Styled Components». Исходный код проекта представлен в приложении А.

Такая структура проекта обеспечивает удобство при разработке, поддержке и улучшении функционала.

3.3 Описание функциональности приложения

Когда мы запускаем приложение, мы наблюдаем экран авторизации (рисунок 12), если мы введем корректные данные, то увидим главный интерфейс приложения (рисунок 13).

Рисунок 12 – Окно входа в приложение

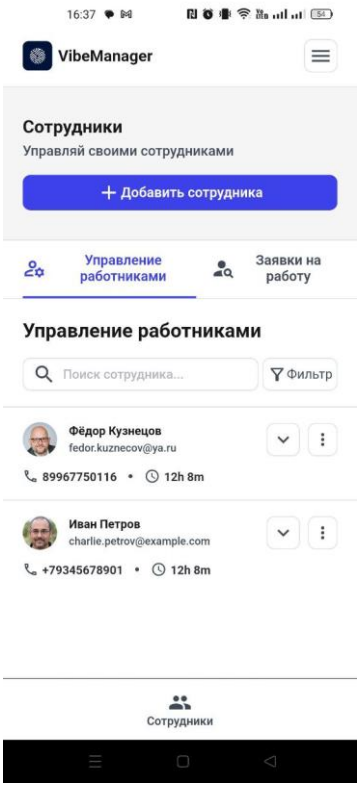

Рисунок 13 – Главная страница

В верхней части интерфейса размещены логотип компании, название приложения, а также кнопка вызова бокового меню, выполненная в виде иконки «гамбургер». Ниже расположен заголовок «Сотрудники» с подзаголовком «Управляй своими сотрудниками», рядом с которым находится кнопка «Добавить сотрудника», визуально выделенная синим цветом для акцентирования внимания пользователя.

В центральной части экрана представлена навигационная панель с вкладками «Управление работниками» и «Заявки на работу». Активной является вкладка «Управление работниками», что визуализировано с помощью синей линии под заголовком. Для облегчения поиска предусмотрено текстовое поле с подписью «Поиск сотрудника» и кнопка фильтрации, позволяющая уточнить отображаемые данные.

Основное содержимое экрана составляет список сотрудников. Для каждого отображается имя, фамилия, адрес электронной почты, номер телефона, время последнего действия, а также набор иконок, предназначенных для вызова дополнительных функций. В нижней части интерфейса находится панель навигации, где активной вкладкой является «Сотрудники».

Интерфейс отличается лаконичным дизайном, удобством взаимодействия и визуальной последовательностью элементов, что способствует комфортной работе пользователя с системой.

При активации кнопки вниз рядом с сотрудником можно увидеть чуть более подробную информацию о сотруднике (рисунок 14).

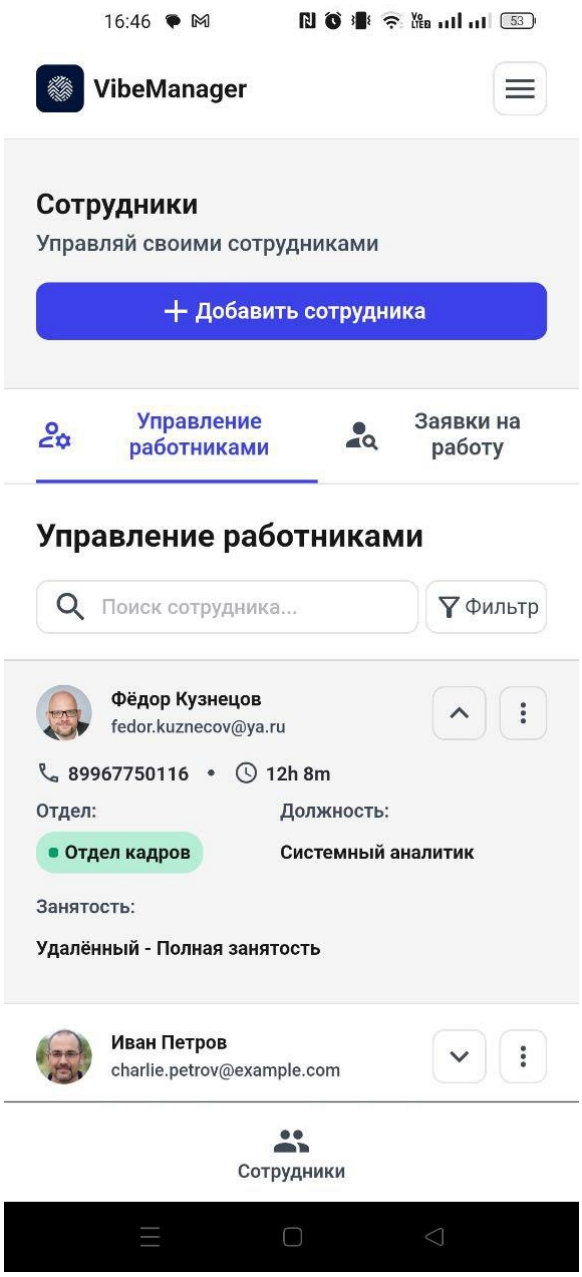

Рисунок 14 – Дополнительная информация о сотруднике

При нажатии на троеточие можно увидеть всплывающее меню с выбором просмотра профиля или увольнения сотрудника (рисунок 15).

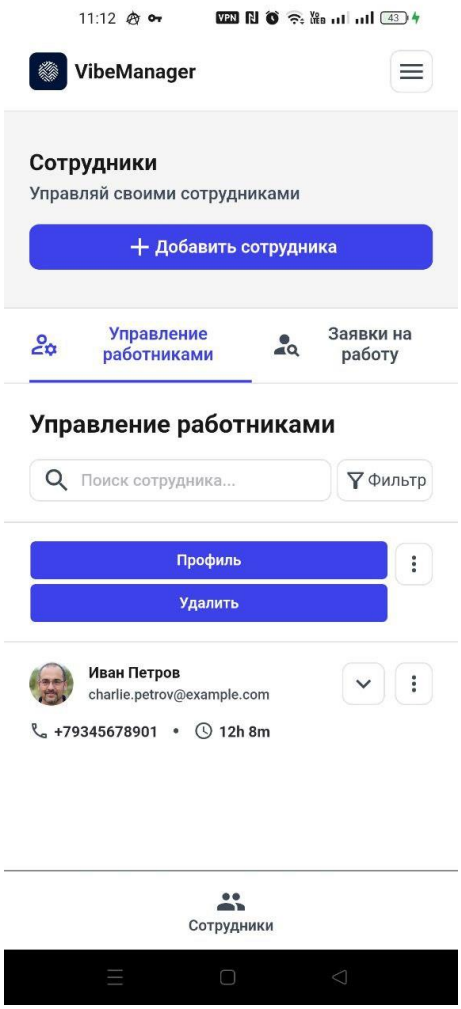

Рисунок 15 – Взаимодействие с сотрудником

При нажатии на профиль открывается дополнительная информация о сотруднике (рисунок 16).

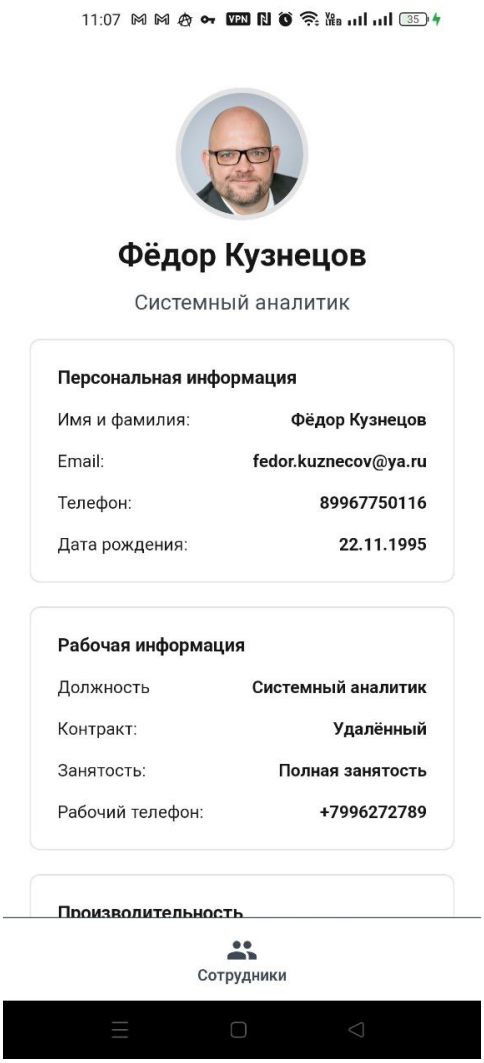

Рисунок 16 – Профиль сотрудника

Теперь, помимо имени, электронной почты, номера телефона и времени последнего действия, отображаются следующие данные:

– отдел, к которому относится сотрудник, с визуальным выделением в виде цветной метки;

– должность сотрудника;

– тип занятости с указанием формата работы;

– производительность.

Теоретическая часть системы управления сотрудниками реализована в виде набора HTML-страниц, которые предоставляются через серверную часть (backend).

В контексте разработки Progressive Web Application (PWA) на базе

Next.js использование современных веб-технологий, таких как HTML, CSS и JavaScript, позволяет создать высокоинтерактивный интерфейс, доступный как на десктопах, так и на мобильных устройствах. Как отмечается в документации по PWA, «Прогрессивные веб-приложения (PWA) используют современные веб-технологии для обеспечения быстрого, надежного и интерактивного пользовательского опыта, в том числе и в оффлайн-режиме благодаря использованию Service Worker» [21]. Преимущество такого подхода заключается в том, что веб-страницы загружаются динамически, обеспечивая возможность оффлайн-доступа благодаря кэшированию через Service Worker.

Функционал тестирования и взаимодействия с пользователем может быть реализован с использованием интерактивных компонентов Next.js и интеграцией с API backend. Это обеспечивает не только отображение материала, но и двусторонний обмен данными между клиентской частью и сервером. Как отмечается в официальной документации по Spring Boot, «Spring Boot упрощает создание автономных приложений, предоставляя механизмы для быстрой разработки REST API и интеграции с клиентскими приложениями» [24]. Например, пользователь может изучить теоретический материал, после чего пройти тестирование прямо в приложении.

Поддержка JavaScript в рамках PWA позволяет использовать гибкие методы взаимодействия, такие как обработка событий, динамическая подгрузка данных и работа с локальным хранилищем. Благодаря этим возможностям система становится интуитивно понятной и удобной для пользователей, предлагая функциональность, сопоставимую с нативными приложениями.

А теперь можем добавить сотрудника.

Окно добавления сотрудника представлено ниже (рисунок 17).

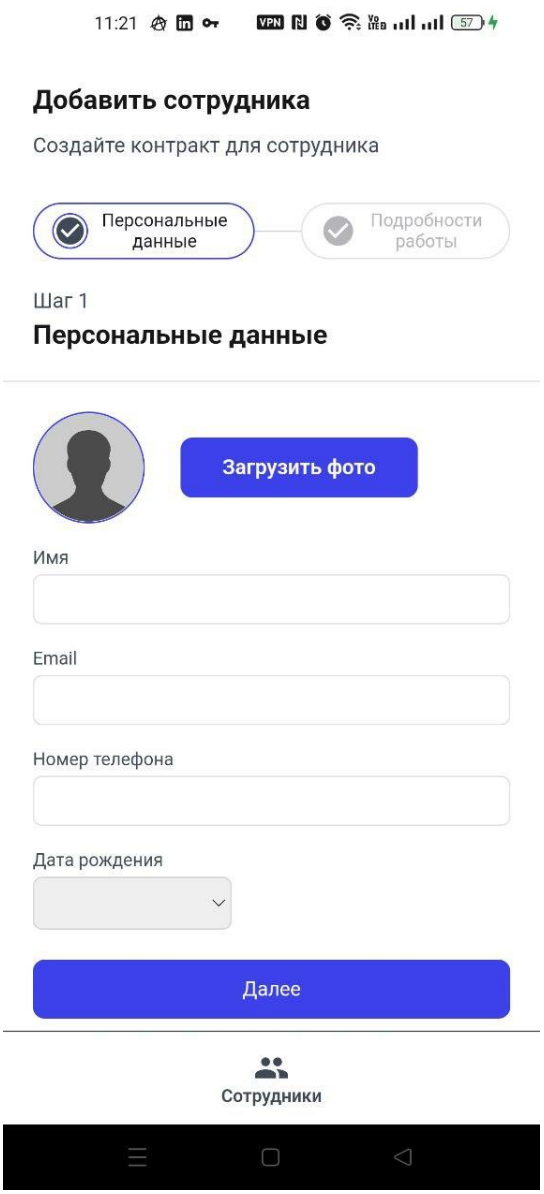

Рисунок 17 – Экран добавления сотрудника

Подробности работы представлены на втором шаге (рисунок 18).

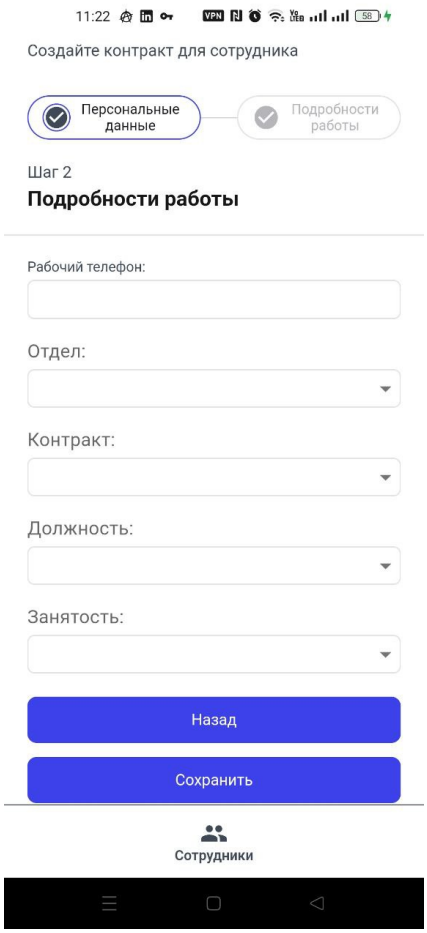

Рисунок 18 – Форма указания подробностей работы

В итоге получается «Мобильное приложение для управления сотрудниками компании: увольнение и приём на работу, анализ рабочего времени по категориям».

3.4 Тестирование мобильного приложения

Для подтверждения перечисленных выше характеристик и выявления возможных недостатков, было проведено тестирование мобильного приложения на различных устройствах и при разных сценариях использования. Основной целью данного этапа являлась проверка стабильности работы, удобства интерфейса, адаптивности дизайна и общей производительности приложения при работе на мобильных устройствах с разными размерами экранов и разрешениями.

Как подчёркивает Р. Мартин, «качественный продукт проявляется в предсказуемом поведении системы при любых условиях эксплуатации» [14]. В рамках тестирования особое внимание уделялось: удобству взаимодействия с элементами интерфейса на сенсорных экранах, корректности отображения данных и компонентов в горизонтальной и вертикальной ориентациях, а также стабильности приложения при нестабильном интернет-соединении. Ниже приведены результаты испытаний и примеры экранов, демонстрирующие адаптивность и удобство использования мобильного приложения на различных устройствах.

На изображении ниже показан экран приложения для добавления нового сотрудника и добавления отчета времени. В этом разделе пользователь может заполнить персональные данные сотрудника. Тестирование форм представлено ниже (рисунок 19 и рисунок 20).

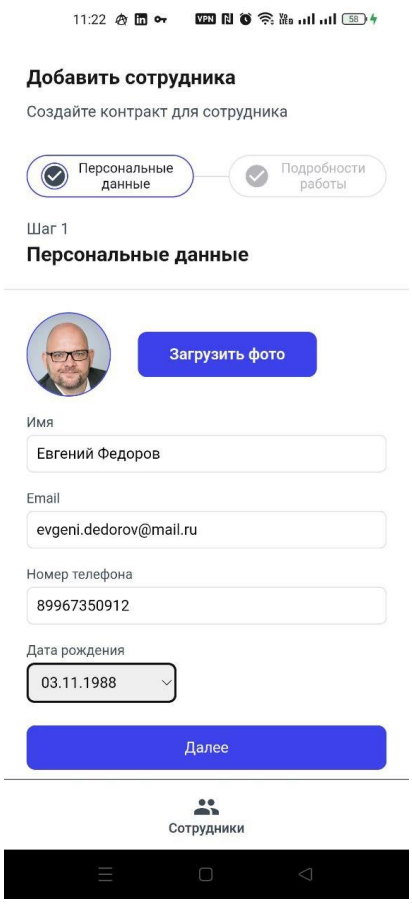

Рисунок 19 – Тестирование создания нового сотрудника

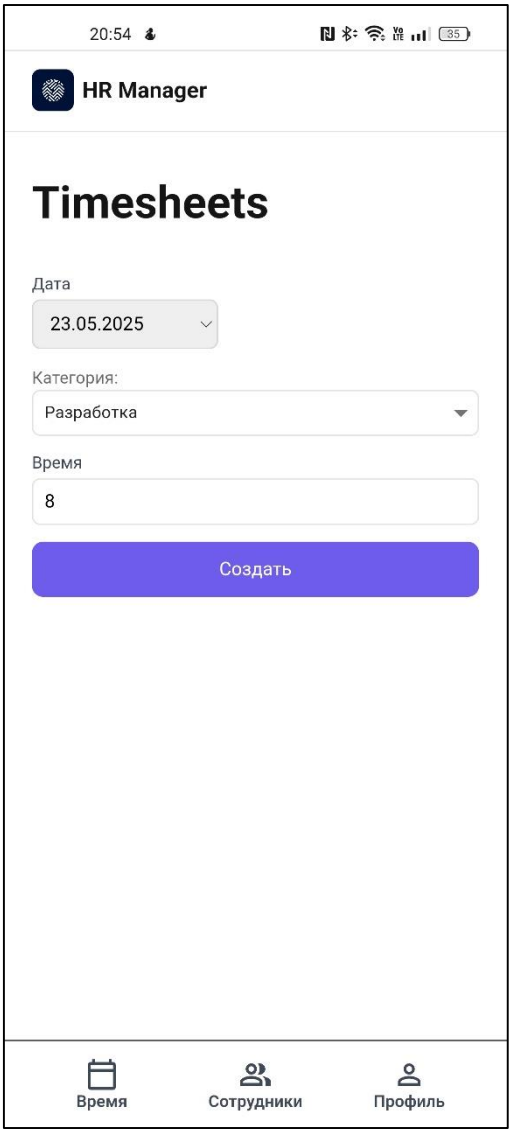

Рисунок 20 – Тестирование создания нового отчета времени

В итоге приложение обеспечивает высокую эффективность, надежно работает и быстро обрабатывает данные, что критично для управления сотрудниками в реальном времени. Использование PWA гарантирует бесперебойную работу приложения, даже при нестабильном интернет-соединении. Гибкость системы позволяет легко адаптировать приложение под изменения в потребностях бизнеса, быстро внося улучшения и обновления.

Помимо этого, приложение обладает возможностью расширения, что позволяет добавлять новые функциональные возможности, такие как отчеты, интеграции с другими сервисами и новые инструменты для сотрудников. Модульная структура и использование компонентов на базе Next.js делают приложение удобным для повторного использования кода, что значительно

ускоряет процесс разработки и облегчает поддержку приложения. Это позволяет быстро реагировать на изменения в требованиях компании и оптимизировать процесс разработки.

Таким образом, разработанное мобильное приложение для управления сотрудниками компании не только обеспечивает высокую эффективность и надежность работы, но и адаптируется к изменениям в потребностях бизнеса, что делает его ценным инструментом для управления персоналом в реальном времени.

Вывод по 3 разделу.

Разработка многофункционального мобильного приложения для управления сотрудниками была проведена с использованием современных технологий и инструментов, что позволило создать удобное и эффективное решение для работы с персоналом. Процесс разработки включал выбор технологий, создание проекта в WebStorm, проектирование функциональности и тестирование приложения. В результате были получены положительные результаты, особенно в области адаптивности и производительности приложения.

Заключение

В ходе выполнения была поставлена задача создать кросс-платформенную мобильную систему управления сотрудниками, предоставляющую полный цикл кадрового учета от регистрации персональных данных до мониторинга рабочего времени

Разработанная архитектура позволила четко разделить пользовательский интерфейс и бизнес-логику, упростить тестирование и дальнейшее сопровождение. При этом слоистый подход с чёткими границами ответственности каждого модуля обеспечил слабую связанность компонентов, что облегчает внедрение новых функций без затрагивания существующего кода. Использование унифицированных интерфейсов для взаимодействия между слоями повысило гибкость системы и упростило замену отдельных сервисов при изменении требований или технологий. Благодаря такому построению архитектуры приложение сохраняет масштабируемость, допускает параллельную работу нескольких команд разработки и снижает затраты на сопровождение в долгосрочной перспективе.

Особое внимание уделено пользовательскому опыту: проведены несколько циклов тестирования, благодаря чему устранены большинство проблем навигации и повышена интуитивная понятность интерфейса.

Таким образом, все функциональные и нефункциональные требования были полностью выполнены. Созданное приложение демонстрирует высокую надежность, удобство использования и потенциальную экономию времени для сотрудников кадровой службы, а также повышение прозрачности процессов для руководства. Перспективы дальнейшего развития включают интеграцию с корпоративными аналитическими платформами, расширение отчётности, внедрение пуш-уведомлений и подключение модулей самообучения сотрудников, что дополнительно повысит ценность системы для компании.

Список используемой литературы и используемых источников

1. Браун Д. Масштабируемые веб-приложения. М. : Вильямс, 2020. 432 с.
2. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений. М. : Мир, 1998. 528 с.
3. Вебер Т. Безопасность в веб-приложениях. СПб. : Питер, 2021. 288 с.
4. Вигерс К., Битти Дж. Разработка требований к программному обеспечению. 3-е изд. СПб. : Питер, 2014. 592 с.
5. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приёмы объектно-ориентированного проектирования. Паттерны проектирования. М. : Мир, 2000. 416 с.
6. Джексон М. Архитектура современных веб-приложений. М. : ДМК Пресс, 2017. 400 с.
7. Дубейковский В. И. Практика функционального моделирования с AllFusion Process Modeler. М. : Диалог-МИФИ, 2021. 464 с.
8. Кнут Д. Э. Искусство программирования. Т. 1: Основные алгоритмы. М. : Вильямс, 2002. 720 с.
9. Круг С. Не заставляйте меня думать. СПб. : Питер, 2020. 216 с.
10. Ларман К. Применение UML и шаблонов: введение в проектирование ПО и процесс разработки. М. : Вильямс, 2004. 768 с.
11. Льюис К. Принципы проектирования пользовательского интерфейса. СПб. : Питер, 2016. 304 с.
12. Макконнелл С. Совершенный код. СПб. : Питер, 2019. 912 с.
13. Марк Л. Основы веб-программирования. СПб. : Наука и техника, 2015. 320 с.
14. Мартин Р. Чистый код. Создание, анализ и рефакторинг. СПб. : Питер, 2012. 464 с.

15. Петров А. Разработка на JavaScript для начинающих. М. : БХВ-Петербург, 2019. 256 с.

16. Прессман Р. С. Разработка программного обеспечения. СПб. : Питер, 2015. 912 с.

17. Смит Дж. Разработка и внедрение информационных систем. М. : ДМК Пресс, 2010. 384 с.

18. Соммервилл И. Инженерия программного обеспечения. М. : Вильямс, 2016. 832 с.

19. Фаулер М. Шаблоны корпоративных приложений. СПб. : Питер, 2003. 560 с.

20. Хартман Р. Управление проектами в IT. М. : Альпина Паблишер, 2018. 352 с.

21. Next.js Documentation [Electronic resource]. URL: https://nextjs.org/docs (accessed: 01.10.2023).

22. PostgreSQL Documentation [Electronic resource]. URL: https://www.postgresql.org/docs/ (accessed: 01.10.2023).

23. Progressive Web Apps Documentation [Electronic resource]. URL: https://web.dev/progressive-web-apps/ (accessed: 01.10.2023).

24. Spring Boot Documentation [Electronic resource]. URL: https://spring.io/projects/spring-boot (accessed: 01.10.2023).

25. Visual Studio Code Documentation [Electronic resource]. URL: https://code.visualstudio.com/docs (accessed: 01.10.2023).

Приложение А

Ссылка на репозиторий с исходным кодом

https://gitlab.com/hr-erp