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)

Мобильные и сетевые технологии
(направленность (профиль))

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему Разработка мобильного приложения для сбора заявок на информационное освещение мероприятия

Обучающийся Д.В. Данилов
(Инициалы Фамилия) (личная подпись)

Руководитель канд. техн. наук Н.В. Хрипунов
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Консультант канд. пед. наук, доцент С.А. Гудкова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2024

Аннотация

В рамках бакалаврской работы «Разработка мобильного приложения для сбора заявок на информационное освещение мероприятия» была проведена обширная исследовательская работа. Объем работы составил 57 страниц, 35 иллюстраций и 6 таблиц. Выпускная работа состоит из введения, трёх глав, заключения и списка литературы.

Бакалаврская работа связана с разработкой мобильного приложения, предназначенного для упрощения процесса подачи заявок на информационное освещение мероприятий. Цель исследования заключается в создании функционального мобильного приложения, способного эффективно собирать и обрабатывать заявки. Для достижения этой цели были поставлены следующие задачи: анализ текущего бизнес-процесса в организации, разработка требований к приложению, изучение аналогичных решений на рынке, выбор платформы и инструментов разработки, проектирование структуры данных, выбор архитектуры программного решения.

В ходе нашего исследования мы создали уникальное мобильное приложение, которое облегчает процесс создания и отслеживания заявок для пользователей. Это приложение было спроектировано с учетом всех требований и успешно интегрировано в рабочий процесс сотрудников медиахолдинга, повышая эффективность обслуживания клиентов.

Мы приступили к этому проекту с анализа текущих бизнес-процессов, разработали полный набор требований к приложению, изучили существующие аналоги и выбрали оптимальные платформы и инструменты для его создания. Мы также разработали модель данных, обеспечивающую эффективное взаимодействие с приложением.

В заключительной стадии мы успешно представили разработанное мобильное приложение и продемонстрировали результаты функционального тестирования, подтверждающие надежную работу всех его основных функций.

Abstract

The title of the graduation work is «Development of a mobile application to collect requests for information coverage of the event».

The graduation work consists of an explanatory note on 57 pages, introduction, including 35 figures, 6 tables, the list of 20 references including 6 foreign sources.

The key issue of the graduation work is the development of a mobile application designed to simplify the process of submitting and collecting requests for information coverage of an event.

The graduation work may be divided into several logically connected parts which are analysis of the current business process in the organization, development of requirements for the application, study of similar solutions on the market, selection of platform and development tools, design of data structure, selection of software solution architecture.

The graduation work describes in details the analysis of current business processes, the development of a complete set of requirements for the application, the study of existing analogues and the selection of optimal platforms and tools for its creation. We also developed a data model to ensure efficient interaction with the application.

In conclusion we'd like to stress we successfully presented the developed mobile application and demonstrated the results of functional testing, confirming the reliable operation of all its main functions.

Оглавление

Введение

Ключевую роль в успешной деятельности подразделений играет эффективное управление бизнес-процессами. Важная деталь этого процесса - сбор заявок внутри подразделения. Качественное выполнение заявок становится решающим фактором для обеспечения работы подразделения. Но нынешние подходы к сбору заявок могут оказаться безрезультатными в использовании. Поэтому разработка мобильного приложения, которое поможет усовершенствовать процесс сбора заявок для медиахолдинга, становится важной задачей.

Целью дипломной работы является создание мобильного приложения для сбора заявок на информационное освещение мероприятия, которое поможет усовершенствовать процесс обработки заявок.

Для начала проанализируем бизнес-процесс. Для описания процессов будет описано само подразделение и инструменты. После чего осуществим имеющийся бизнес-процесс в виде диаграмм.

На базе проведенного анализа будут заявлены требования к мобильному приложению для сбора заявок на информационное освещение мероприятия. Проведется обзор и анализ похожих мобильных приложений для сбора заявок на информационное освещение мероприятия с целью выявления их преимуществ и недостатков.

На базе сформулированных требований и анализа будет определена задача по разработке мобильного приложения для сбора заявок. В рамках этой задачи будет выполнена разработка приложения, в которую входит: поиск и выбор платформы для реализации, инструменты для разработки и проектирование. Далее будет разработана архитектура приложения.

Разработку мобильного приложения выполним на основе выбранной платформы и инструментов. Далее протестируем приложение и сформируем описание нашего продукта. Разработанное мобильное приложение позволит качественно обрабатывать заявки в медиахолдинге.

Глава 1 Анализ бизнес-процессов предприятия

1.1 Краткая характеристика предприятия

Центр гуманитарных технологий и медиакоммуникаций «Молодежный медиахолдинг «Есть talk!»» – структурное подразделение Тольяттинского государственного университета, которое занимается информационным сопровождением деятельности университета. Он был основан в 2016 году. Центр состоит из: редакции сайтов, редакции газет, телевизионной редакции, радиостудии и пресс-службы [6]. Как говорил Павел Валерьевич Дуров: «Что такое университет? Это же раздробленная структура с удельными княжествами». На рисунке 1 показан состав Центра.

Рисунок 1 – Состав ЦГТиМ

Мобильное приложение нужно сделать для Центра гуманитарных технологий и медиакоммуникаций «Молодёжный медиахолдинг «Есть talk!»», который находится по адресу: Самарская область, город Тольятти, улица Белорусская 14.

Организационная структура подразделения изображена на рисунке 2.

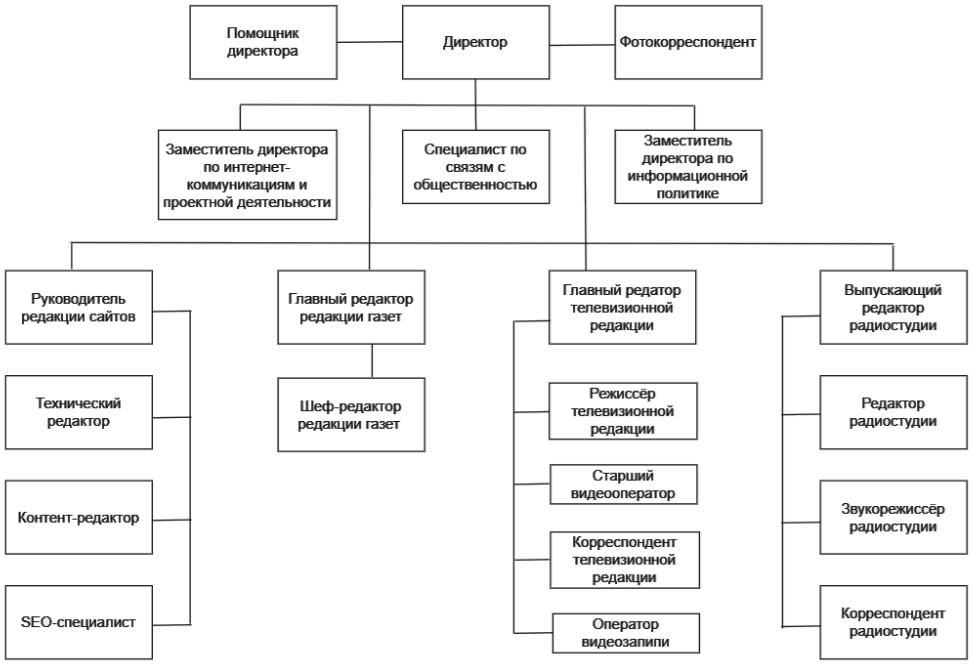

Рисунок 2 – Организационная структура ЦГТиМ

Медиахолдинг занимается следующими задачами [6]:

– информационное и имиджевое сопровождение деятельности университета для внутренних и внешних потребителей;

– поддержка актуального состояния группы сайтов ТГУ;

– «упаковка» контента в рамках проекта «Росдистант»;

– реальное практико-ориентированное обучение через профессиональную практическую деятельность студентов-журналистов;

– присутствие университета в социальных сетях;

– медиасопровождение деятельности по привлечению абитуриентов и слушателей с применением современных маркетинговых технологий (включая интернет-технологии).

Говорят, что несчастие хорошая школа; может быть. Но счастие есть лучший университет. Оно довершает воспитание души, способной к доброму и прекрасному, - так писал Александр Сергеевич Пушкин в письме Нащокину.

Центр гуманитарных технологий и медиакоммуникаций «Молодёжный

медиахолдинг «Есть talk!»» – ключевое подразделение, формирующее информационную политику университета.

1.2 Выбор CASE-средств для описания бизнес-процессов

Для разработки мобильного приложения, важно еще на этапе разработки грамотно выбрать CASE-средства. Они позволяют не только дескриптивно описывать бизнес-модели, но и анализировать полученные в результате описания данные.

Среди массива разнообразных CASE-средств можно выделить: Bizagi, Lucidchart, Visual Paradigm, Ramus и другие. В рамках настоящего исследования несколько следующих программ разобраны подробно.

Bizagi Studio – инструмент для моделирования бизнес-процессов, который поддерживает стандарт BPMN 2.0. Он обеспечивают возможность создания и анализа сложных процессов для повышения эффективности работы организации.

Lucidchart – онлайн-платформа для создания различных видов диаграмм, включая модели бизнес-процессов в соответствии со стандартом BPMN 2.0. Lucidchart помогает визуализировать бизнес-процессы и делает их более понятными для сотрудников и заинтересованных сторон.

Visual Paradigm – инструмент для разработки моделей бизнес-процессов, анализа требований и создания программного обеспечения. С помощью Visual Paradigm пользователи могут создавать качественное программное обеспечение на основе детального анализа и моделирования бизнес-процессов.

Ramus – программа позволяет создавать визуальные диаграммы для наглядного отображения различных бизнес-процессов. Поддерживает сразу две популярных методологии: DFD и IDEF0 [14].

Для того, чтобы сделать объективный и обоснованный выбор в пользу конкретного CASE-средства для данной выпускной квалификационной

работы, был проведен сравнительный анализ представленных выше программ. Рассмотрим ниже.

Функциональность:

CASE-средство Bizagi примечательно тем, что позволяет не только анализировать, но и визуализировать и автоматизировать бизнес-процессы. Кроме того, средство предоставляет функционал для моделирования данных, в том числе в рамках совместной работы над проектом.

Lucidchart в свою очередь имеет широкий спектр инструментов для решения разнообразных задач. Среди которых: создание диаграмм разных типов, включая организационные диаграммы, UML-диаграммы, потоковые диаграммы и другие. Может быть интегрировано в самые популярные облачные сервисы и самый распространенный офисный софт. Как и Bizagi, позволяет работать над проектом совместно.

Visual Paradigm аналогично Lucidchart позволяет создавать UML- и ER-диаграммы, а также ряд других. Анализирует требования; в программе заложены функции моделирования бизнес-процессов, а также разработки ПО.

Ramus помимо UML-диаграмм и других типов диаграмм, может экспортировать диаграммы в широкой палитре форматов, снабжена интуитивно понятным user friendly интерфейсом.

Таким образом, в результате анализа можно сделать вывод, что Bizagi предназначен в основном для моделирования бизнес-процессов, Lucidchart - для широкого спектра типов диаграмм с акцентом на коллаборацию, Visual Paradigm - для моделирования бизнес-процессов и разработки программного обеспечения, Ramus - для создания UML-диаграмм с простым интерфейсом.

Интерфейс и удобство пользования:

у Bizagi интуитивно понятный интерфейс позволяет легко создавать и моделировать бизнес-процессы с помощью встроенных шаблонов и «упаковывать» их в диаграммы. Программа, однако, рассчитана на опытного пользователя.

За счет адаптированного и интуитивно понятного интерфейса Lucidchart

может создавать диаграммы различного типа. Поддерживает коллаборацию и возможность работы в реальном времени, что делает его удобным для совместной работы над проектами.

Visual Paradigm предоставляет профессиональный и функциональный интерфейс, который может показаться сложным для новичков, но обладает большим количеством возможностей. Имеет множество инструментов для моделирования бизнес-процессов и разработки программного обеспечения. Удобство использования зависит от уровня знаний пользователя в области моделирования.

Ramus имеет простой интерфейс. Предоставляет базовый набор инструментов для создания UML-диаграмм. Удобство использования зависит от потребностей пользователя в создании диаграмм.

Таким образом, Lucidchart выделяется своим интуитивно понятным интерфейсом, который делает его удобным для пользователей всех уровней. Bizagi также имеет удобный интерфейс для работы с бизнес-процессами. Visual Paradigm может показаться сложным для новичков, но обладает большим функционалом. Ramus предоставляет базовые возможности с простым интерфейсом.

Интеграция с другими приложениями:

Bizagi предоставляет возможность интеграции с различными системами и приложениями через API; поддерживает интеграцию с популярными платформами, такими как Salesforce, Microsoft Office, SAP и другими; обеспечивает возможность обмена данными между бизнес-процессами и другими приложениями.

Lucidchart позволяет интегрироваться с популярными сервисами и приложениями, такими как Google Drive, Microsoft Office, Slack и другими; обладает API для создания собственных интеграций; поддерживает экспорт и импорт диаграмм в различные форматы для работы с другими приложениями.

Visual Paradigm предоставляет возможность интеграции с различными средами разработки программного обеспечения, такими как Eclipse, NetBeans,

Visual Studio и другими; обладает API для создания собственных интеграций и расширений; поддерживает экспорт диаграмм в различные форматы для работы с другими приложениями.

Ramus не обладает широким набором интеграций с другими приложениями из коробки; возможно, требуется использование дополнительных инструментов или ручной работы для интеграции с другими приложениями; ограниченные возможности по обмену данными с другими системами.

Исходя из этого сравнения, Bizagi, Lucidchart и Visual Paradigm выделяются своей широкой поддержкой интеграции с различными приложениями и платформами. Ramus ограничен в интеграционных возможностях и может потребовать дополнительных усилий для работы с другими приложениями.

Стоимость:

Bizagi предлагает различные тарифные планы, включая бесплатный план для небольших команд и стартапов; платные планы Bizagi начинаются от $10-20 в месяц на пользователя, в зависимости от функциональности и возможностей, включенных в план.

Lucidchart также предлагает бесплатный план с ограниченным функционалом; платные планы Lucidchart начинаются от $7.95 в месяц на пользователя для индивидуальных пользователей и от $9.95 для команд.

Visual Paradigm имеет различные тарифные планы, включая бесплатный план для некоммерческого использования; платные планы Visual Paradigm начинаются от $99 в год на пользователя для студентов и преподавателей, а для коммерческого использования цены могут быть выше.

Ramus является бесплатным open-source инструментом, доступным для скачивания и использования без дополнительной оплаты.

Исходя из этого сравнения, Ramus выделяется как бесплатный инструмент без необходимости дополнительной оплаты. Bizagi, Lucidchart и Visual Paradigm предлагают различные тарифные планы с разной стоимостью

в зависимости от функциональности и возможностей, предоставляемых каждым инструментом.

Возможности совместной работы:

Bizagi предоставляет возможность совместной работы в реальном времени для команд, позволяя пользователям работать над проектами одновременно; пользователи могут делиться доступом к проектам, комментировать и обсуждать изменения, а также отслеживать историю версий.

Lucidchart также поддерживает совместную работу в реальном времени, позволяя пользователям совместно создавать и редактировать диаграммы и схемы; пользователи могут приглашать коллег на проекты, делиться доступом к документам и обсуждать изменения через комментарии.

Visual Paradigm предлагает функциональность совместной работы для команд, позволяя пользователям работать над проектами в реальном времени; пользователи могут совместно создавать диаграммы, модели и документацию, а также обмениваться комментариями и идеями.

Ramus как open-source инструмент, не предоставляет встроенных функций совместной работы в реальном времени; однако пользователи могут обмениваться файлами и документацией для совместной работы через другие средства связи.

Исходя из этого сравнения, Bizagi, Lucidchart и Visual Paradigm предлагают возможности совместной работы в реальном времени для команд. Ramus не имеет встроенных функций совместной работы, но пользователи могут обмениваться файлами для коллективной работы. Далее рассмотрим свойства всех инструментов в виде таблицы, так как таблица является удобным способом структурирования данных, и она используется как раз для анализа данных, а также чтобы наглядно увидеть плюсы и минусы каждого из них, и затем выберем CASE-инструмент для использования в проектировании модели бизнес-процессов.

Свойства ПО представлены в таблице 1.

Таблица 1 – Свойства ПО

	Bizagi	Lucidchart	Visual Paradigm	Ramus
Функциональность	-	+	+	-
Интерфейс и удобство использования	+	+	+	+
Интеграция с другими приложениями	+	+	+	-
Стоимость	-	-	-	+
Возможности совместной работы	+	+	+	-

Таким образом, для разработки мобильного приложения для сбора заявок на информационное освещение мероприятия, использование CASE-инструмента Ramus будет оптимальным решением по соотношению удобства и функциональности.

1.3 Анализ модели бизнес-процесса

Приступим к анализу имеющейся модели бизнес-процесса для сбора заявок на информационное освещение мероприятия. Так, инициатор заявки подаёт её на специальной странице на сайте. Сотрудник медиахолдинга рассматривает заявку, в случае если она соответствует информационной политике университета и грамотно заполнена, сотрудник принимает заявку и начинает её выполнять.

Контекстная диаграмма процесса сбора заявок показана на рисунке 3.

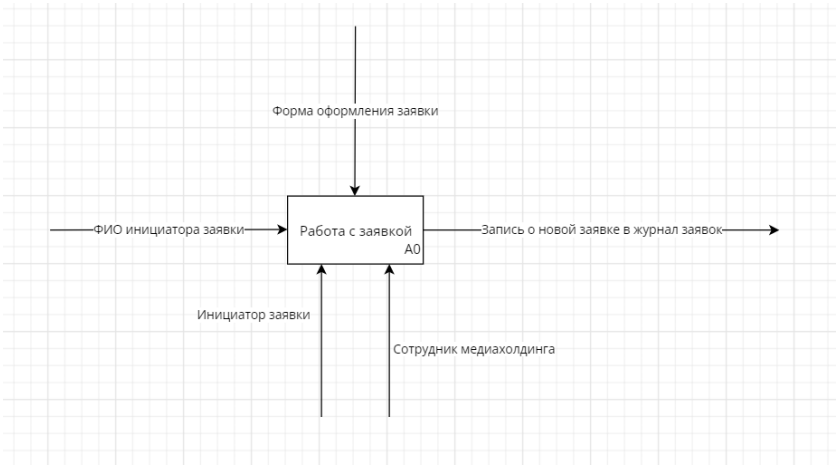

Рисунок 3 – Контекстная диаграмма процесса сбора заявок

Декомпозиция блока A0 показана на рисунке 4.

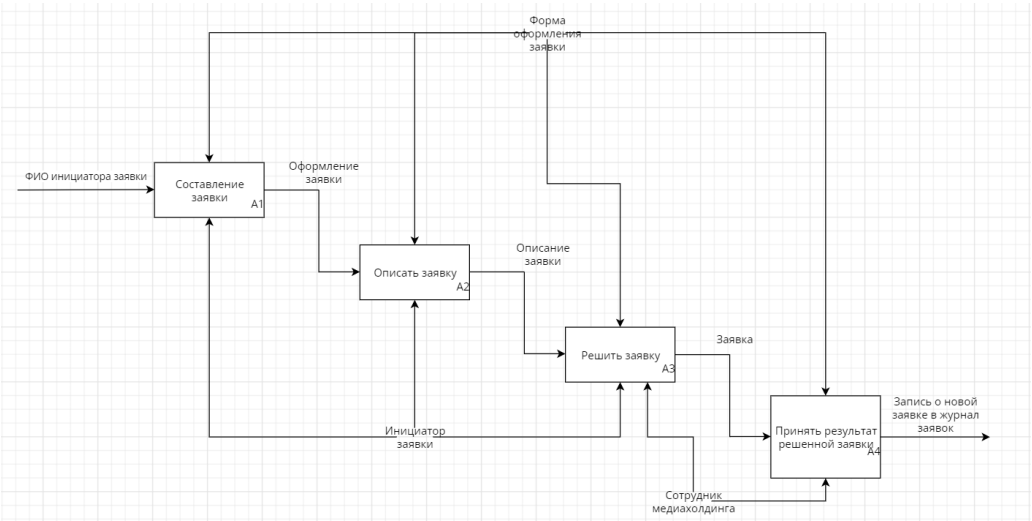

Рисунок 4 – Декомпозиция блока A0

Декомпозиции блоков A1 и A2 показаны на рисунках 5 и 6 соответственно.

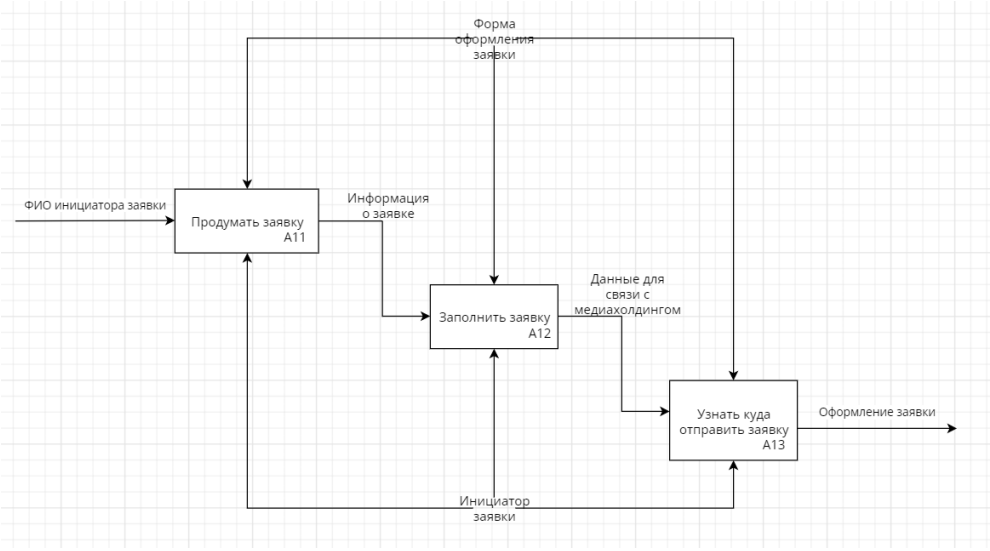

Рисунок 5 – Декомпозиция блока A1

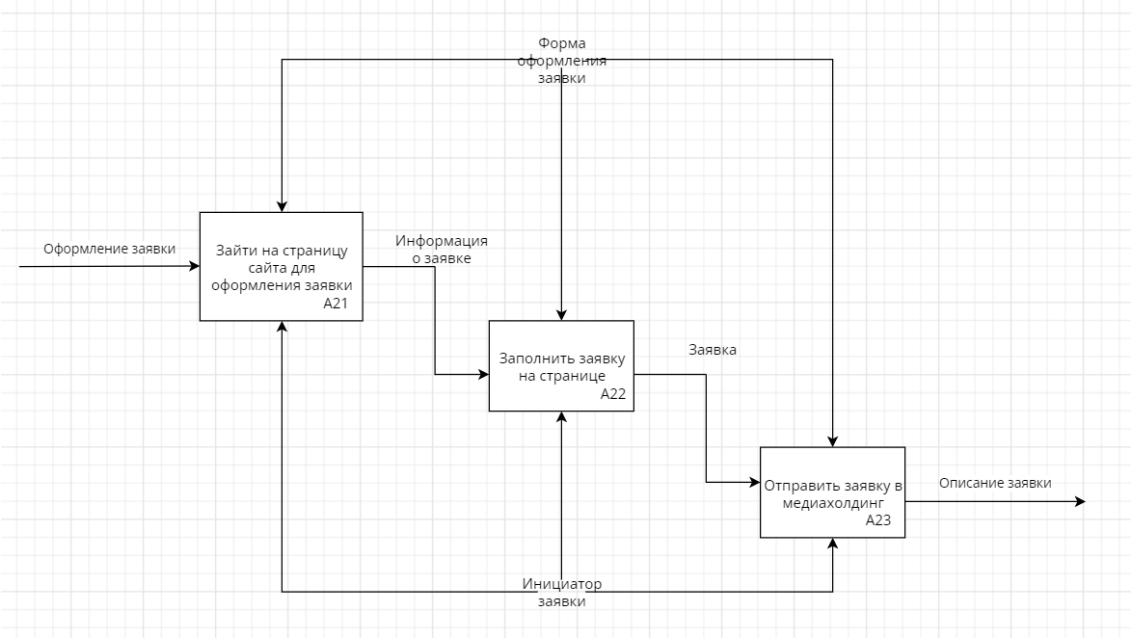

Рисунок 6 – Декомпозиция блока A2

Покажем DFD-диаграмму на рисунке 7.

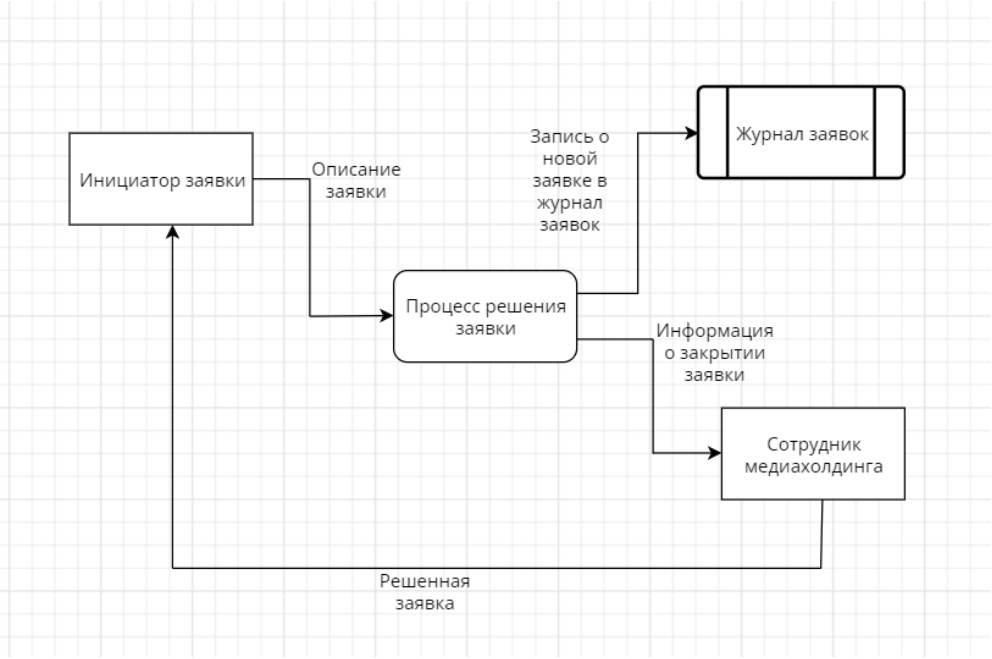

Рисунок 7 – DFD-диаграмма процесса выполнения заявки

Проанализировав процессы, были выявлены изъяны в нынешней системе работы по сбору заявок на информационное освещение мероприятия. В частности, на данный момент отсутствует возможность отследить статус заявки. Кроме того, процесс оформления заявки заказчиком занимал много времени. Предлагаемое в рамках настоящего исследования решение выявленных проблем – создание и внедрение мобильного приложения, с помощью которого будет вестись сбор заявок на информационное сопровождение мероприятия.

1.4 Обзор и анализ аналогов мобильных приложений для сбора заявок на информационное освещение мероприятия

Для выполнения задачи сбора заявок на информационное освещение мероприятия в результате исследования не было найдено приложений для мобильных устройств аналогичной или близкой тематикой. Установлено, однако, что существуют приложения с похожим функционалом.

Рассмотрим и проанализируем несколько аналогов.

Freshdesk – полноценное приложение для управления собранными заявками, оно предоставляет комплексный набор функций для работы с заявками от создания до размещения итогов. С помощью Freshdesk пользователь может создавать, просматривать и обновлять заявки в режиме реального времени; настраивать и автоматизировать рабочие процессы, отслеживать все необходимые KPI и оценивать эффективность служб; интегрироваться с другими приложениями, расширяя функционал.

Zendesk популярное приложение для управления заявками, проектами и осуществления техподдержки. С помощью этого софта можно работать как с внешними, так и с внутренними запросами. Кроме того, с помощью интерфейса этого приложения можно создавать задачи и назначать исполнителей, а также отслеживать прогресс выполнения. Пользователь может интегрировать эту программу в различные каналы коммуникации клиентов, такие как (электронная почта, социальные сети, смартфон); реализована расширяемость за счет настраиваемых плагинов.

Jira Service Desk – комплексная платформа управления заявками и проектами, это мощное решение, предназначенное для комплексного управления заявками технической поддержки и проектами. Оно предоставляет широкий спектр функций, помогая организациям повышать эффективность работы и улучшать качество обслуживания клиентов. Управление заявками Jira Service Desk позволяет создавать, просматривать и отслеживать заявки в режиме реального времени. Пользователи могут: создавать заявки через различные каналы, такие как электронная почта, веб-портал или мобильное приложение. Настраивать поля заявки и рабочие процессы в соответствии с потребностями компании. Присваивать заявки специалистам, отслеживать их статус и устанавливать сроки выполнения. Получать оповещения о новых или обновленных заявках, что позволяет оперативно реагировать на запросы клиентов.

Помимо управления заявками, Jira Service Desk также предлагает

возможности управления проектами. Пользователи могут: создавать и управлять проектами на основе kanban-досок или диаграмм Ганта. Назначать задачи членам команды, устанавливать зависимости и отслеживать прогресс. Использовать пользовательские поля и рабочие процессы для настройки проекта в соответствии с конкретными потребностями. Получать отчеты о ходе выполнения проекта, что позволяет контролировать фактический и запланированный прогресс.

Проанализированные по отдельности приложения можно сравнить по требованиям FURPS+.

FURPS+ – метод классификации требований к программному обеспечению (ПО), учитывающий возможности ПО, стабильность работы, производительность, удобство использования, а также потенциал обслуживания и развития ПО [16].

Значок + (плюс) в формуле FURPS+ обозначает дополнительные характеристики, среди которых чаще всего встречаются важные аспекты дизайна, интерфейса и итоговой реализации продукта.

Анализ мобильных приложений учёта заявок по требованиям FURPS+ показал следующие результаты:

– Freshdesk обеспечивает удобство создания и отслеживания заявок;

– Zendesk предлагает удобный интерфейс и простое использование;

– Jira Service Desk имеет простой интерфейс и удобное использование.

Надежность и стабильность работы:

– Freshdesk обеспечивает стабильность;

– Zendesk регулярно улучшается;

– Jira Service Desk имеет высокую надежность.

С точки зрения производительности:

– У Freshdesk высокая скорость работы и эффективность;

– Zendesk позволяет быстро управлять заявками;

– Jira Service Desk имеет высокую скорость работы.

User friendly:

– Эти приложения дают хороший функционал поддержки.

Дополнительные параметры включают цену и способность обновления приложений, а также внедрение нового функционала в соответствии с потребностями компании.

Изученная информация представлена в таблице 2.

Таблица 2 – Сравнительный анализ приложений с требованиями FURPS+

	Freshdesk	Zendesk	Jira Service Desk
Functionaluty	+	+	+
Usability	+	+	+
Reliability	+	+	+
Perfomance	+	+	+
Supportability	+	+	+
Дополнительные характеристики	-	-	-

Таким образом, при разработке мобильного приложения нужно обратить внимание на все аспекты, описанные выше.

Основной функционал по сбору заявок в достаточной мере реализован во всех рассмотренных приложениях. Однако различаются в дополнительных возможностях.

При разработке мобильного приложения для сбора заявок на информационное освещение мероприятия необходимо учитывать положительные и отрицательные стороны похожих приложений и тем самым сформулировать требования к приложению.

1.5 Разработка требований к мобильному приложению сбора заявок на информационное освещение мероприятия

С помощью проанализированного бизнес-процесса и изучения аналогов были определены следующие условия к мобильному приложению:

Необходимо создать функционал:

– регистрация пользователей;

– создание заявок с возможностью ввода необходимой информации;

– просмотр списка всех пользователей;

– просмотр всех заявок, включая смену статуса в личном кабинете администратора;

– просмотр всех своих заявок, включая статус в личном кабинете инициатора.

Приложение должно обладать интуитивно понятным интерфейсом для удобства использования, обеспечивать быструю и стабильную работу и защиту конфиденциальности данных пользователей.

Этот функционал лёг в основу диаграммы вариантов использования мобильного приложения для сбора заявок на информационное освещение мероприятия, которая показана на рисунке 9.

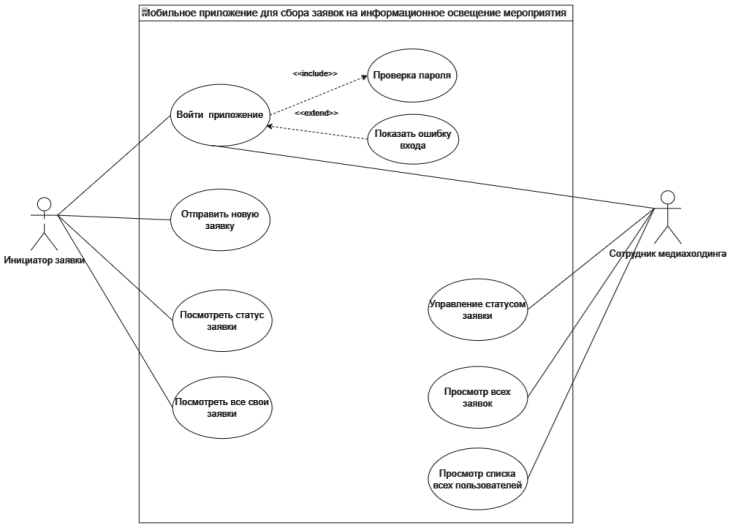

Рисунок 8 – Диаграмма вариантов использования

На диаграмме вариантов использования представлена графическая модель, которая показывает нам, как пользователи используют приложение.

На графике представлены два участника: сотрудник медиахолдинга и инициатор заявки.

1.6 Постановка задачи на разработку мобильного приложения для сбора заявок

Для улучшения процесса составления заявки необходимо внести изменения в бизнес-процесс [10]. Использовать мобильное приложение или веб-приложение – один из потенциальных способов модернизации бизнес-процесса. Однако мобильное приложение может быть более удобным для пользователей, так как они могут запускать его с мобильных устройств, что делает процесс подачи заявки на информационное освещение мероприятия более доступным.

Инициатору заявок после внедрения функции создания заявок в мобильное приложение упростит подачу заявок и оптимизирует работу сотрудника медиахолдинга.

Контекстная диаграмма показана на рисунке 9.

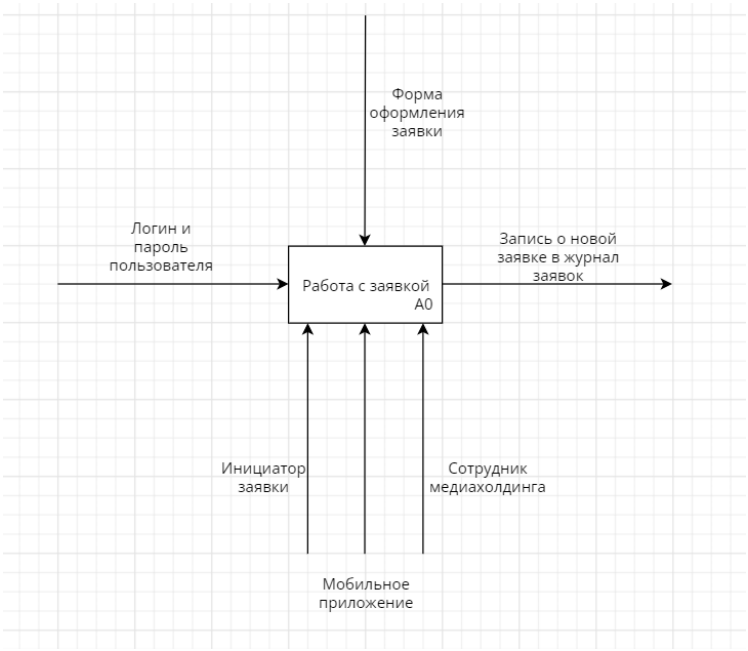

Рисунок 9 – Контекстная диаграмма процесса сбора заявок через мобильное приложение

Декомпозиция блока А0 представлена на рисунке 10.

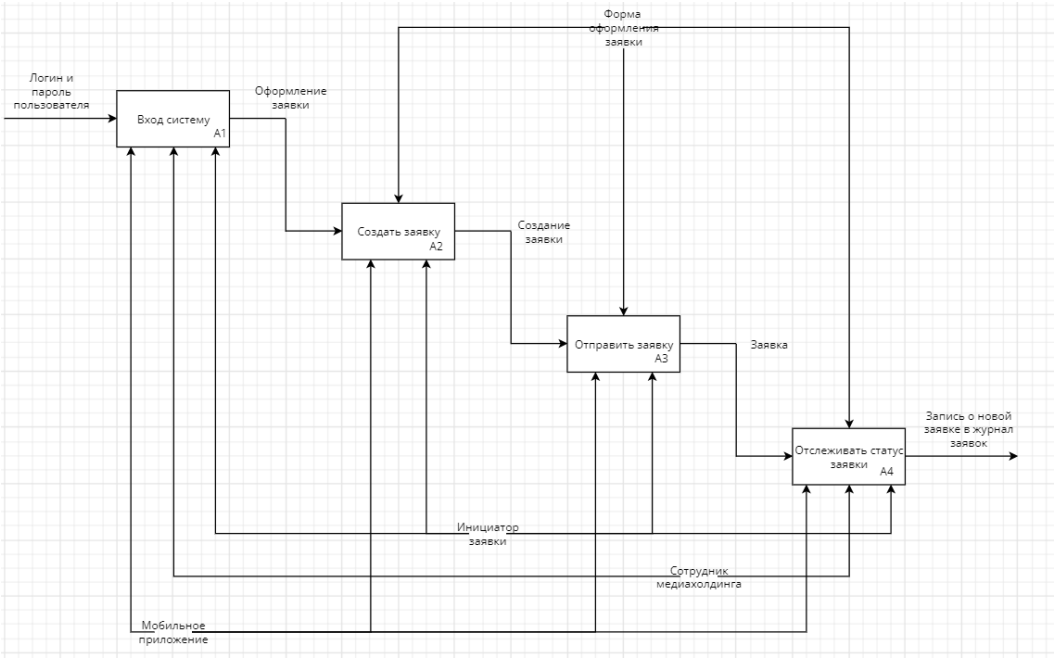

Рисунок 10 – Декомпозиция блока А0

Декомпозиции блоков А2 и А3 представлены на рисунках 11 и 12 соответственно.

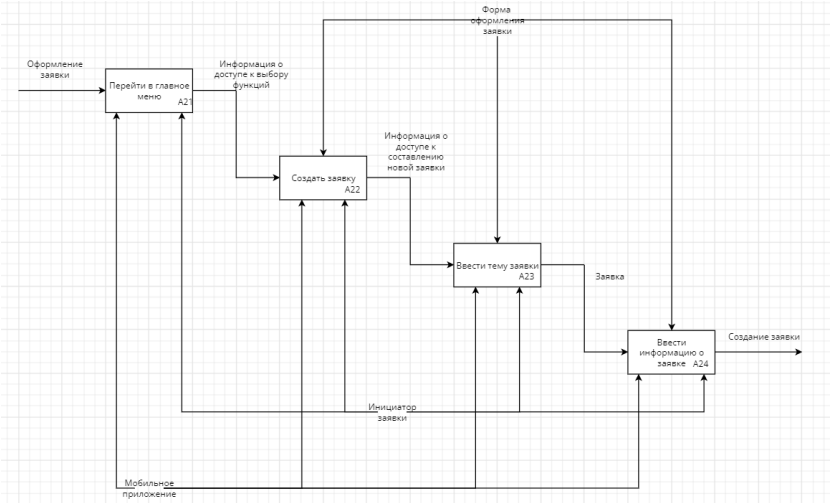

Рисунок 11 – Декомпозиция блока А2

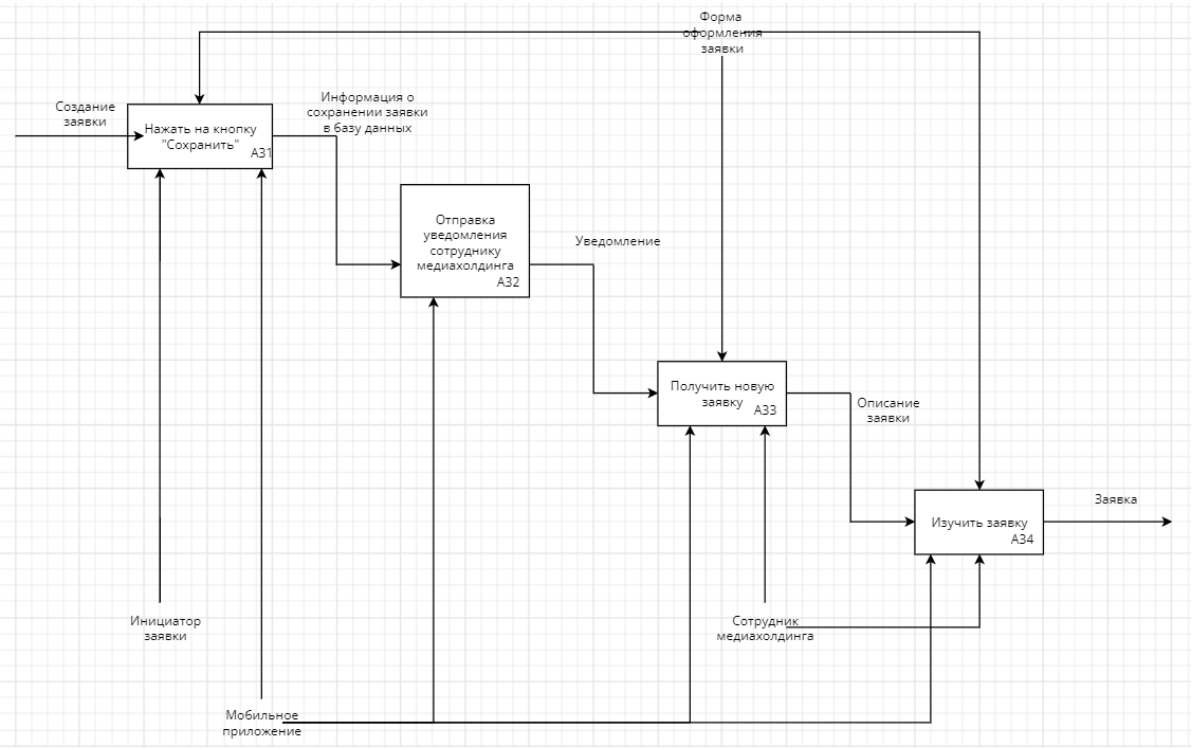

Рисунок 12 – Декомпозиция блока А3

На рисунке 13 показана DFD-диаграмма.

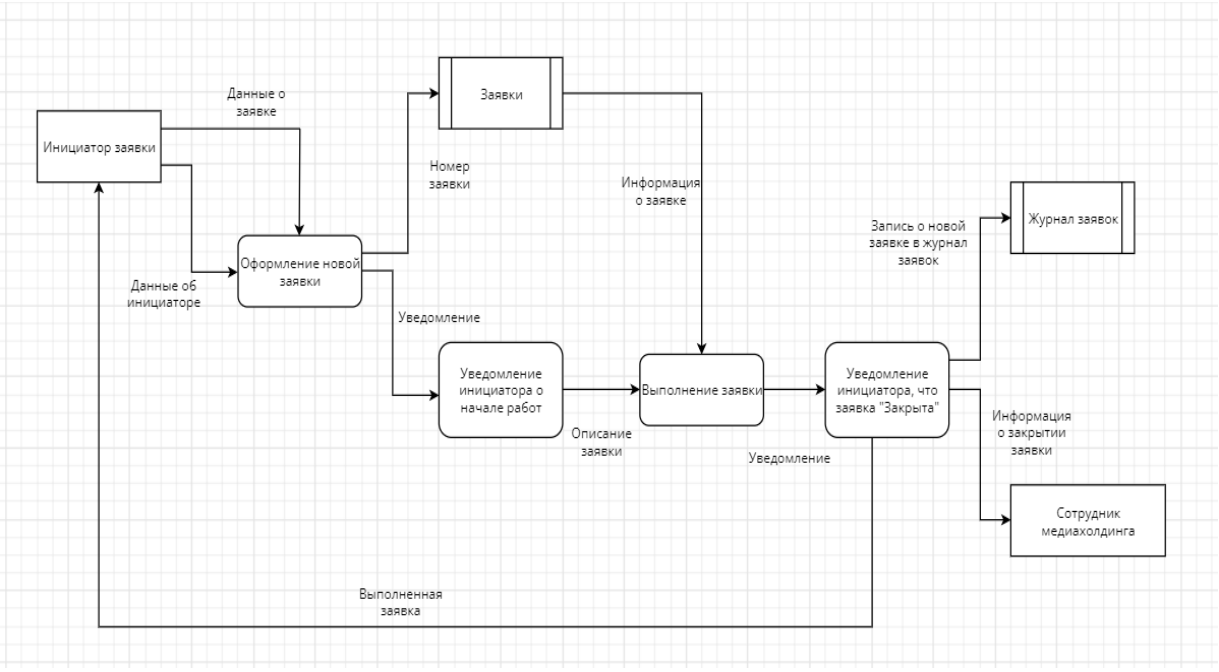

Рисунок 13 – DFD-диаграмма визуального взаимодействия

Итоговую постановку задачи можно сформулировать следующим образом. Необходимо создать мобильное приложение сбора заявок на информационное освещение мероприятия, которое позволит сотрудникам медиахолдинга легко их обрабатывать, а инициаторам заявок быстро и удобно создавать, и отправлять, а также отслеживать статус заявки.

Инициаторам в приложении следует предоставить возможность подавать заявки через удобный мобильный интерфейс, указывая содержание заявки. После отправки заявки они должны видеть статус выполнения и изменениях в ней. Кроме того, они должны иметь доступ к просмотру всех своих заявок.

Сотрудники медиахолдинга должны работать в комфортном интерфейсе. Они могут иметь возможность смотреть все заявки и менять их статусы.

Еще одним значимым моментом в процессе разработки данного приложения является обеспечение безопасности личных данных инициаторов заявок и сотрудников медиахолдинга, а также сохранность информации о заявках. Для этого требуется применение современных технологий шифрования данных, систем аутентификации и авторизации пользователей.

Для эффективной отправки заявок в медиахолдинг, пользовательский интерфейс приложения должен быть интуитивно понятным и удобным, обеспечивая быструю и безпроблемную процедуру подачи заявок. Приложение должно быть адаптировано для использования на мобильных гаджетах с операционной платформой Android.

Данное приложение улучшит взаимодействие пользователей, так как инициаторы заявок смогут оформить заявку с любого мобильного устройства, а также будут тратить меньше времени на оформление заявки. Также в любом удобном месте, например: улица, дом, работа или даже ресторан, инициаторы могут подавать заявки, а администратор их обрабатывать. Легко, быстро и доступно.

Выводы по главе 1

В первой главе приведена краткая характеристика подразделения и определены задачи медиахолдинга. Для концептуальной разработки было выбрано CASE-средство. На основе этого была изложена действующая структура бизнес-процесса. Далее было проведено исследование аналоговых мобильных приложений, сформулированы необходимые критерии для мобильного приложения и представлена усовершенствованная схема бизнес-процесса. В финальной части этой главы были определены ключевые задачи, предстоящие перед разработкой мобильного приложения, которое улучшит взаимодействие пользователей.

Глава 2 Проектирование мобильного приложения для сбора заявок на информационное освещение мероприятия

2.1 Выбор платформы реализации мобильного приложения

Для выбора мобильной операционной системы нам нужно учитывать несколько характеристик: функциональность, простота использования, безопасность, стабильность, популярность, доступность.

Сейчас на рынке есть много мобильных операционных систем. Давайте рассмотрим самые популярные из них.

Android – операционная система, разработанная компанией Google. На конец апреля 2024 года доля Android на рынке составляет 71,31% [20].

iOS – операционная система компании Apple для устройств iPhone и iPad [18]. Ее доля рынка составляет 27,95% к концу апреля 2024 г. iOS безопасна и надежна.

Windows Phone – операционная система от Microsoft. На конец апреля 2024 года ее доля на рынке составила 0,02%.

На рисунке 14 представлен график популярности мобильных операционных систем в 2024 году [19].

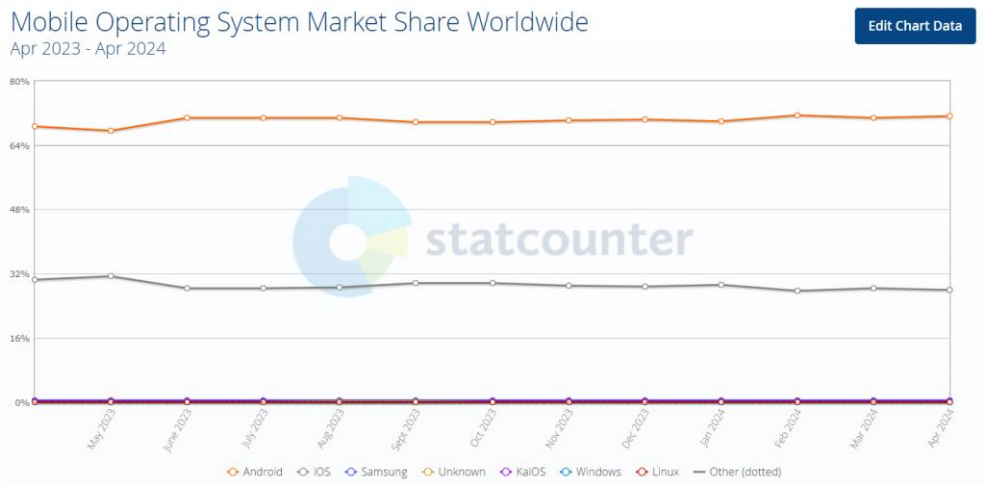

Рисунок 14 – График популярности мобильных операционных систем

Проведем и изучим анализ мобильных операционных систем [7].

Функциональность:

Android предлагает обширный набор функций для полного погружения в разработку приложений.

iOS также предлагает широкий набор функций, но более жестко контролирует доступ разработчиков к определенным системным функциям и инструментам.

Windows Phone в целом схожа с вышеперечисленными ОС, но не так широко распространена среди пользователей и имеет некоторые ограничения на доступ к приложениям.

Удобство использования:

Интерфейс Android интуитивно понятен и знаком пользователям, поскольку продукты Google широко используются.

iOS имеет простой и интуитивно понятный интерфейс.

Windows Phone имеет простой интерфейс, но поскольку эта операционная система не популярна среди пользователей, как Android и iOS, пользователям может показаться сложным ее использование.

Безопасность:

Android обеспечивает высокий уровень безопасности благодаря передовым технологиям.

iOS обеспечивает высокий уровень безопасности благодаря стандартам Apple и строгому контролю за приложениями.

Windows Phone, хотя и предоставляет некоторые механизмы безопасности, из-за ограниченной популярности и доступности сторонних приложений, становится более уязвимым перед атаками и вирусами.

Стабильность:

Стабильность работы устройств на базе Android зависит от производителя и модели. И благодаря постоянному развитию платформы и совершенствованию аппаратного и программного обеспечения стабильность устройств Android постоянно повышается.

iOS славится своей надежностью и отличной работоспособностью на всех гаджетах от Apple. Благодаря закрытой экосистеме и тщательной оптимизации операционной системы для каждого устройства, iOS часто превосходит их по стабильности и производительности.

Windows Phone больше активно не поддерживается Microsoft, обновления и улучшения производительности ограничены.

Популярность и доступность:

Android - самая распространенная мобильная операционная система в мире (доля 71,31 %). Ее повсеместное распространение обеспечивает потенциальную доступность приложений, разработанных на множестве устройств разных производителей и ценовых категорий.

iOS очень популярна среди пользователей (27,95 %). Однако доступность iOS ограничена только устройствами Apple, что сужает потенциальную аудиторию пользователей разрабатываемого приложения.

Windows Phone гораздо менее популярна по сравнению с Android и iOS (0,02 %). Поэтому столь малый сегмент пользователей может препятствовать широкому распространению приложения и создавать неудобства для пользователей.

Далее рассмотрим плюсы и минусы операционных систем и приведем их в таблице, чтобы было понятнее какая операционная система самая популярная, с большим функционалом, стабильная, удобная для пользования и безопасная. Таблица является удобным способом структурирования данных. Таблица представляет собой распределение данных по однотипным строкам и столбцам. Таблицы широко используются в различных исследованиях и анализе данных. Таблицы также встречаются в СМИ и в рукописных материалах. Затем сделаем вывод и выберем операционную систему, для которой будем разрабатывать мобильное приложение для сбора заявок на информационное освещение мероприятия.

Сравнение мобильных операционных систем показано в таблице 3.

Таблица 3 – Сравнение мобильных операционных систем

	Android	IOS	Windows Phone
Функциональность	+	+	-
Удобство использования	+	+	+
Безопасность	+	+	+
Стабильность	+	+	+
Популярность и доступность	+	-	-

Таким образом, можно сделать вывод, что операционная система Android является наиболее гибкой и открытой для разработчиков и конечных пользователей приложений операционной системы. Это позволяет разработчикам свободнее работать с прикладным программным обеспечением. Также Android поддерживает большое количество устройств от разных производителей.

2.2 Выбор средств разработки

Android Studio выбрана средой разработки для создания мобильного приложения для сбора заявок на информационное освещение мероприятие по нескольким причинам:

Во-первых, Android Studio является официальной средой разработки для операционной системы Android, предоставляя разработчикам широкий спектр инструментов и ресурсов [15].

Во-вторых, Android Studio обладает интуитивно понятным интерфейсом. Простая навигация и встроенные инструменты помогают повысить эффективность работы над приложением.

В-третьих, Android Studio предоставляет возможность эмулировать устройства для тестирования приложений, работает с визуальными редакторами пользовательского интерфейса и обладает функционалом отладки.

В-четвертых, данная платформа полностью поддерживает язык

программирования Java, который является основным языком для создания приложений под операционную систему Android. Java - популярный язык среди разработчиков, имеющий обширное сообщество и множество ресурсов, что значительно упрощает процесс разработки приложений.

Платформа интегрируется со множеством инструментов разработки, включая Firebase. Это ускоряет разработку мобильного приложения для сбора заявок, делая выбор Android Studio логичным и обоснованным.

Выбор языка программирования Java для разработки был обусловлен рядом причин, выявленных в ходе анализа [1]:

– Android Studio, выбранная в качестве среды разработки, полностью совместима с языком программирования Java.
– Разработчик приложения уже имеет опыт работы с языком программирования Java. Это ускорит разработку приложения, освоить различные инструменты и фреймворки, а также закрепить навыки на практике.
– Поскольку язык программирования Java широко распространен и давно и активно используется программистами по всему миру, в том числе и в России, по Java существует обширная база вспомогательных ресурсов.
– Java - один из самых стабильных и безопасных языков программирования, имеющий долгую историю использования в самых разных проектах [4].

Из множества средств разработки, доступных для создания мобильного приложения для сбора заявок, был выбран Firebase [17]. Использование Firebase позволяет сосредоточиться на разработке самого приложения. Совмещение Firebase с ОС Android осуществляется без труда, предоставляя удобные возможности для интеграции с мобильными приложениями. SDK Firebase содержит готовые к использованию API и компоненты, которые упрощают работу с облачными сервисами Firebase.

Технология Firebase Realtime Database обеспечивает базу данных в

реальном времени. С ее помощью можно создавать приложения, которые сразу реагируют на изменения данных, а затем синхронизируются между устройствами и платформами.

Firebase имеет богатую документацию, обучающие ресурсы, что позволяет легко освоить работу системы.

2.3 Проектирование модели данных

Создание внешней модели, которая описывает объект независимо от его физических характеристик, известно как инфологическое проектирование. Основная цель этого процесса - обеспечить эффективные и понятные методы сбора и представления данных для последующего хранения в базе данных [11].

Внешние модели могут быть разработаны различными способами, например, с использованием семантических сетей, а также языка инфологического моделирования или ER-схем. Однако наиболее распространенным подходом является моделирование сущностей и отношений. Он состоит из сущности, то есть различимого и наблюдаемого объекта, о котором в базе данных содержится информация. Атрибут, который, в свою очередь, отображает характеристики названного объекта. Отношение - взаимосвязь между двумя или более объектами, которая отражает их взаимозависимость.

Концептуальная модель показывает характеристики и взаимосвязи сущностей без физической реализации данных. Затем она необходима для дальнейшего проектирования и реализации базы данных.

На рисунке 15 представлена концептуальная модель.

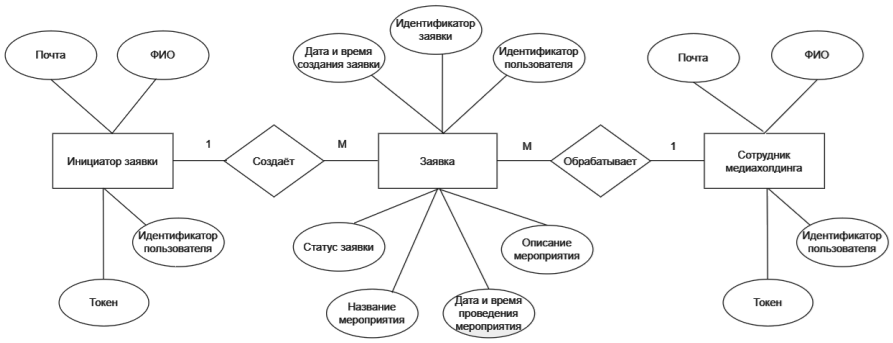

Рисунок 15 – Концептуальная модель данных

Объекты и атрибуты определены на основе изучения предметной области.

Логическая модель данных демонстрирует логические связи между информационными компонентами. Логическая модель данных является расширением концептуальной модели. Она представляет собой будущую информационную систему. Основными элементами являются сущности, атрибуты, ключи и взаимосвязи, которые представляют бизнес-информацию и определяют бизнес-правила [9].

Логическая модель данных изображена на рисунке 16.

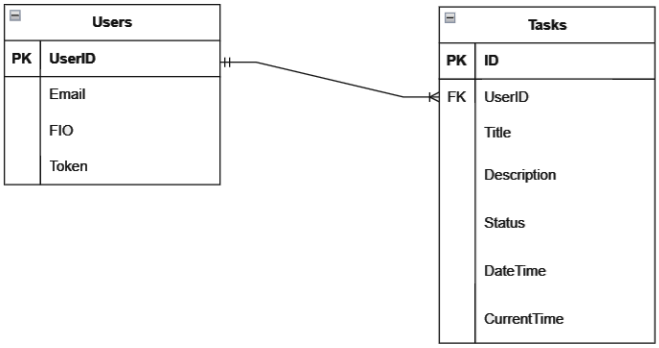

Рисунок 16 – Логическая модель данных

В таблице 4 описаны связи, представленные на рисунке 16.

Таблица 4 – Связи данных

Ключ связи	Таблица	Подчинённая таблица	Связь
userID (уникальный идентификатор пользователя)	users (пользователи)	tasks (Заявки)	1:M

Для разработки этого приложения мы построили физическую модель данных на основе логической модели данных. Построенная модель предоставляет информацию, используя в качестве базы данных Firebase Realtime Database. Физическая модель данных показана на рисунке 17.

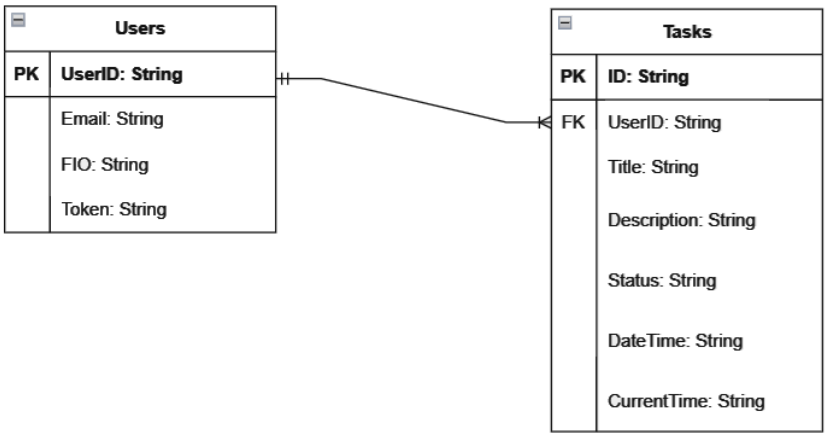

Рисунок 17 – Физическая модель данных

Далее рассмотрим структуру таблиц для пользователей и заявок, чтобы было понятно какие будут поля, тип полей, и какая в них будет содержаться информация.

Данная информация заполнит нашу базу данных, впоследствии чего мы будем видеть корректно информацию, отправленной заявки.

Дополнительная информация будет полезна для более точного анализа и обработки данных, что поможет нам лучше понять контекст заявки.

В таблице 5 представлена структура таблиц.

Таблица 5 – Структура таблиц

Номер	Название поля	Тип поля	Описание
Users (пользователи)			
1	userID	String	ID пользователя (уникальный идентификатор)
2	Email	String	Почта (строка, содержащая адрес электронной почты)
3	FIO	String	ФИО (строка, содержащая ФИО пользователя)
4	Token	String	Токен устройства (строка, содержащая уникальный идентификатор устройства)
Tasks (заявки)			
1	id	String	ID заявки (уникальный идентификатор)
2	userID	String	ID пользователя (идентификатор, связанный с данной заявкой)
3	title	String	Название мероприятия (строка, содержащая название мероприятия)
4	description	String	Описание заявки (строка, содержащая описание заявки, связанная с данной заявкой)
5	dateTime	String	Дата и время проведения мероприятия (строка, содержащая дату и время проведения мероприятия)
6	status	String	Статус заявки (строка, указывающая на текущий статус заявки)
7	currentTime	String	Время отправления заявки (дата и время, когда заявка была отправлена)

Физическая модель данных определяет, как хранится и обрабатывается информация в конкретной СУБД, такой как Firebase Realtime Database [12].

2.4 Архитектура программного продукта

Виды архитектуры мобильных приложений, представлены ниже [3].

MVC (Model-View-Controller) – архитектура, которая разбивает приложение на три компонента: модель, представление и контроллер. За счет модульности и возможности повторного использования кода, MVC обеспечивает четкое разграничение обязанностей.

MVP (Model-View-Presenter) основан на архитектуре MVC, однако есть небольшие изменения. Управление взаимодействием между моделью и

представлением берет на себя Presenter. В отличие от MVC, где управление распределено между компонентами. Шаблон MVP упрощает процесс тестирования и обеспечивает более четкое разделение логики и представления, что способствует повышению качества и эффективности разработки.

Архитектура MVVM (Model-View-ViewModel) основывается на принципах MVC, но добавляет еще один компонент – ViewModel. Этот компонент выполняет роль посредника между моделью и представлением, обеспечивая передачу данных и методов для их обработки. MVVM способствует более четкому разграничению пользовательского интерфейса и бизнес-логики, что упрощает процесс тестирования и сопровождения кода.

Reactive Architecture – концепция архитектуры, опирающаяся на принципы реактивного программирования, основанные на обработке данных в потоках и отклике на события. Этот подход позволяет разрабатывать приложения, способные гибко преобразовываться и масштабироваться.

Еще один метод организации архитектуры предполагает, что размещение кода и компонентов приложения может соответствовать стандартным практикам разработки для платформы Android, но без прямого использования определенного архитектурного шаблона. Этот подход известен как «Основанный на стандартных компонентах Android».

В диаграмме развёртывания изображены важные компоненты, необходимые для качественной работы мобильного приложения.

Firebase используется в качестве облачной платформы для хранения данных. С его помощью можно через API взаимодействовать с базой данных и пользоваться облачными функциями.

Диаграмма развёртывания изображена на рисунке 18.

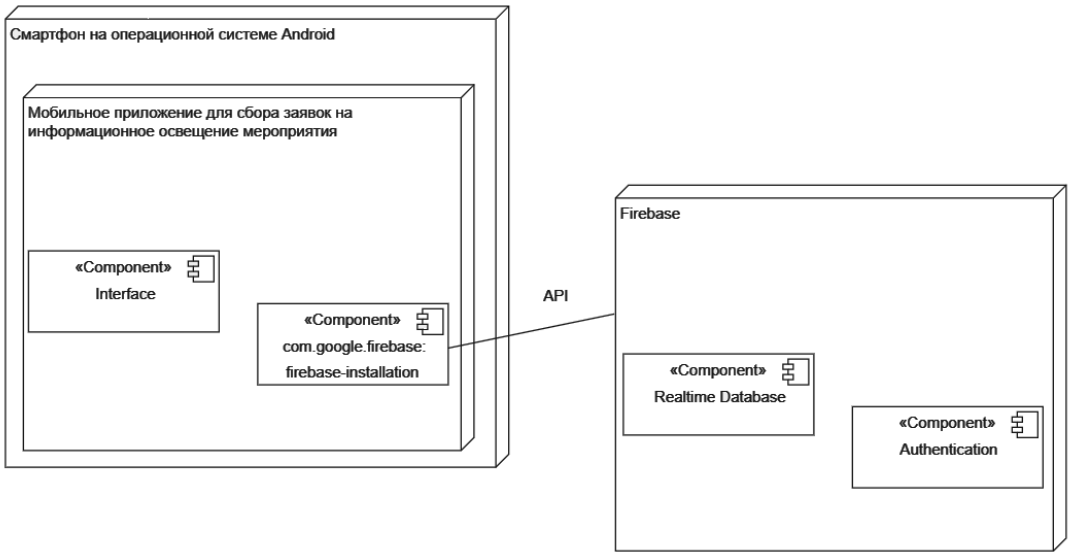

Рисунок 18 – Диаграмма развёртывания

Именно устойчивое и правильное взаимодействие всех обозначенных на диаграмме развертывания элементов, позволяет приложению эффективно работать.

Диаграмма развертывания даёт нам понять, как клиентские устройства взаимодействуют с базой данных, которая в свою очередь является основой для работы мобильного приложения.

Чтобы представить организацию компонентов системы и связей между ними используются диаграммы компонентов. Они показывают общее представление обо всех компонентах системы [2].

Диаграмма компонентов представлена на рисунке 19.

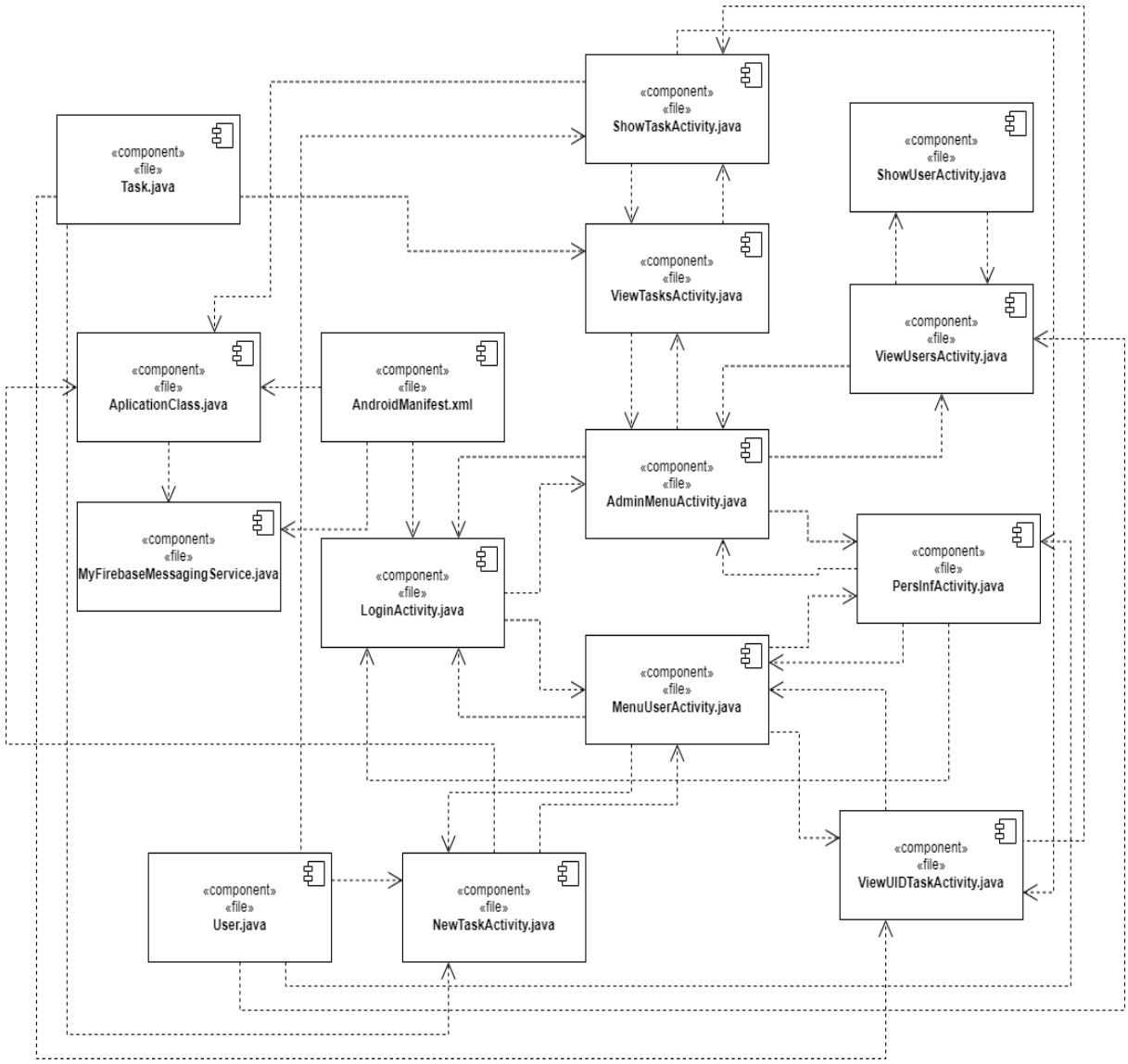

Рисунок 19 – Диаграмма компонентов

Диаграммы компонентов и развертывания помогают выделить ключевые компоненты и показать их взаимосвязи, что способствует лучшему пониманию работы приложения и организации.

Выводы по главе 2

В главе 2 представлена компаративистика мобильных операционных систем, в ходе сравнения мобильные операционные системы оценивались по нескольким критериям. В результате обобщения полученных выводов и проведенной аналитической работы, выбрана операционная система Android,

как платформа для создания мобильного приложения.

Были описаны инструменты, использованные для создания мобильного приложения, включая Android Studio, Firebase и язык программирования Java. После обоснования выбора этих инструментов были разработаны модели данных: концептуальная, логическая и физическая. Были изучены архитектуры программных продуктов и была выбрана наиболее подходящая структура.

В заключительной части второй главы представлены диаграмма развертывания и диаграмма компонентов, которые помогают выделить ключевые компоненты и показать их взаимосвязи, что способствует лучшему пониманию его работы и организации. И с помощью этих данных начнем разработку приложения для сбора заявок.

Глава 3 Реализация мобильного приложения для сбора заявок на информационное освещение мероприятия

3.1 Краткое описание разработанного решения

Предложенное решение – мобильное приложение, которое предназначено для сбора заявок на информационное освещение мероприятий. Приложение разработано с использованием языка программирования Java для операционной системы Android, используя Android Studio и Firebase.

Приложение позволяет инициаторам создавать новые заявки на информационное освещение мероприятия и отслеживать их статус. Сотрудники медиахолдинга могут обрабатывать полученные заявки. Приложение предоставляет удобный интерфейс для инициаторов заявок, что облегчает заполнение информации о мероприятии и отслеживание его статуса.

Разработанное решение тесно интегрировано с облачной платформой Firebase от Google. Firebase используется для хранения данных пользователей, включая информацию о заявках.

Данное решение обеспечивает подразделению эффективный механизм для сбора заявок, улучшает коммуникацию между инициаторами и сотрудниками медиахолдинга, а также обеспечивает более оперативное реагирование на заявки.

3.2 Реализация аутентификации и авторизации

Для обеспечения безопасности и контроля доступа к функциям приложения была разработана система аутентификации и авторизации [8]. Основной метод аутентификации основан на использовании электронной почты, что позволяет пользователям входить в приложение.

В качестве инструмента для реализации процессов аутентификации и авторизации была интегрирована платформа Firebase Authentication.

На рисунке 20 показана панель управления зарегистрированными пользователями.

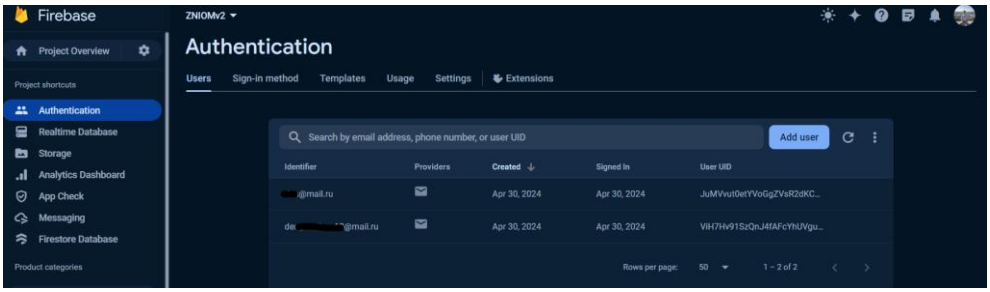

Рисунок 20 – Панель управления зарегистрированными пользователями

Для регистрации пользователю предлагается ввести свою электронную почту, пароль и Ф.И.О. Затем требуется войти в систему, указав свою электронную почту и пароль. Далее эти данные проверяются с помощью сервиса Firebase Authentication. При успешной авторизации пользователю открывается доступ ко всем основным функциям приложения.

На рисунке 21 представлен экран регистрации пользователя, а на рисунке 22 экран входа пользователя.

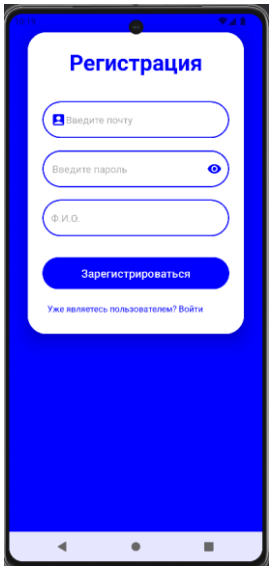

Рисунок 21 – Интерфейс регистрации

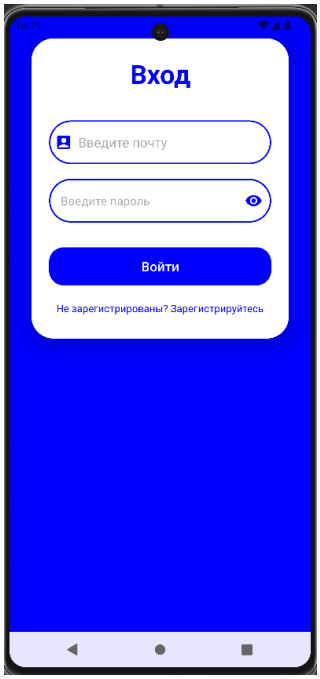

Рисунок 22 – Интерфейс входа

Код регистрации пользователи приведен в приложении А.

На рисунках 23 и 24 показаны меню администратора и пользователя соответственно.

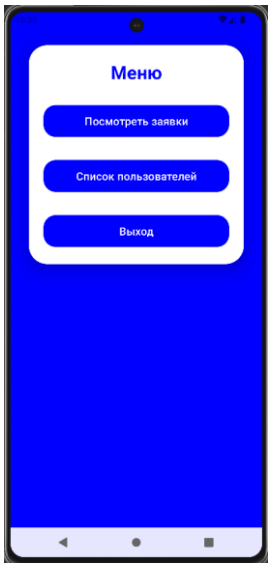

Рисунок 23 – Интерфейс меню администратора

Рисунок 24 – Интерфейс меню пользователя

Обеспечение безопасности данных пользователей, предотвращение несанкционированного доступа и гарантирование конфиденциальности информации осуществляются через внедрение процессов регистрации и аутентификации в приложении.

На рисунке 25 представлен фрагмент кода регистрации нового пользователя.

```
private boolean validateInput(String user, String pass, String fio) {
    boolean isValid = true;

    if (user.isEmpty()) {
        signupEmail.setError("Введите почту");
        isValid = false;
    } else {
        signupEmail.setError(null);
    }

    if (pass.isEmpty()) {
        signupPassword.setError("Введите пароль");
        isValid = false;
    } else if (pass.length() < 8) {
        signupPassword.setError("Пароль должен содержать минимум 8 символов");
        isValid = false;
    } else {
        signupPassword.setError(null);
    }

    if (fio.isEmpty()) {
        signupFio.setError("Введите Ф.И.О.");
        isValid = false;
    } else {
        signupFio.setError(null);
    }

    return isValid;
}
```

Рисунок 25 – Часть кода регистрации нового пользователя

Метод объявлен с модификатором доступа private, что означает, что он может быть вызван только внутри класса, в котором объявлен. Он принимает три строки: user, pass и fio, и возвращает значение типа boolean, указывающее на то, являются ли данные валидными. Переменная isValid инициализируется значением true. Эта переменная будет использоваться для отслеживания валидности всех вводимых данных. Если строка user пустая, устанавливается ошибка для signupEmail с сообщением «Введите почту», и переменная isValid устанавливается в false. В противном случае ошибка сбрасывается. Если строка pass пустая, устанавливается ошибка для signupPassword с сообщением «Введите пароль», и переменная isValid устанавливается в false. Если длина пароля меньше 8 символов, устанавливается ошибка «Пароль должен содержать минимум 8 символов», и переменная isValid устанавливается в false. В противном случае ошибка сбрасывается. Если строка fio пустая, устанавливается ошибка для signupFio с сообщением «Введите Ф.И.О.», и переменная isValid устанавливается в false. В противном случае ошибка сбрасывается. В конце метод возвращает значение isValid, которое указывает, прошли ли все проверки. Код регистрации нового пользователя указан в

приложении А.

На рисунке 26 изображён фрагмент кода входа в систему существующего пользователя.

Рисунок 26 – Фрагмент кода входа в систему существующего пользователя

Метод validateInput объявлен с модификатором доступа private, что означает, что он может быть вызван только внутри класса, в котором объявлен. Он принимает две строки: email и password, и возвращает значение типа boolean, указывающее на то, являются ли данные валидными. Если строка email пустая, устанавливается ошибка для loginEmail с сообщением «Введите почту», и метод возвращает false, указывая на недействительные данные. Если email не соответствует шаблону действительного email, устанавливается ошибка для loginEmail с сообщением «Пожалуйста введите действительную почту», и метод возвращает false. Если строка password пустая, устанавливается ошибка для loginPassword с сообщением «Введите пароль», и метод возвращает false. Если все проверки пройдены успешно, метод возвращает true, указывая на валидные данные.

Метод signIn объявлен с модификатором доступа private, что означает, что он может быть вызван только внутри класса, в котором объявлен. Он принимает две строки: email и password. Метод использует объект auth для выполнения входа с помощью email и пароля. Если вход успешен (addOnSuccessListener), вызывается метод handleLoginSuccess. Если вход

неудачен (addOnFailureListener), отображается сообщение об ошибке с текстом ошибки.

– auth.signInWithEmailAndPassword(email, password): метод, выполняющий вход с указанными email и паролем.
– addOnSuccessListener(this::handleLoginSuccess): метод, вызываемый в случае успешного входа. Он ссылается на метод handleLoginSuccess текущего объекта (this).
– addOnFailureListener(e -> Toast.makeText(LoginActivity.this, «Ошибка:» + e.getMessage(), Toast.LENGTH_SHORT).show()): метод, вызываемый в случае ошибки входа. Он отображает всплывающее сообщение (Toast) с текстом ошибки.

На рисунке 27 изображён фрагмент кода перехода в меню инициатора или администратора.

```
private void handleLoginSuccess(AuthResult authResult) {
    FirebaseUser currentUser = auth.getCurrentUser();
    assert currentUser != null;
    String userName = currentUser.getEmail();
    String admin = "den_danilov_13@mail.ru";
    Class<?> destinationClass = userName.equals(admin) ? MainActivityAdmin.class : MainActivity.class;
    startActivity(new Intent( packageContext: this, destinationClass));
    Toast.makeText( context: this,  text: "Успешный вход", Toast.LENGTH_SHORT).show();
    finish();
}
```

Рисунок 27 – Фрагмент кода перехода в меню инициатора заявки или администратора

Метод объявлен с модификатором доступа private, что означает, что он может быть вызван только внутри класса, в котором объявлен. Он принимает один параметр authResult типа AuthResult, который содержит результат аутентификации. Метод получает текущего аутентифицированного пользователя с помощью auth.getCurrentUser(). Утверждение assert currentUser != null используется, чтобы гарантировать, что текущий пользователь не является null. Из текущего пользователя извлекается его email и сохраняется в переменную userName. Email администратора жестко закодирован в

переменную admin. Используется тернарный оператор для определения класса назначения. Если email текущего пользователя совпадает с email администратора, то destinationClass устанавливается как MainActivityAdmin.class. В противном случае, destinationClass устанавливается как MainActivity.class. Создаётся намерение (Intent) для запуска соответствующей активности (экрана) на основе класса назначения и вызывается метод startActivity, чтобы начать новую активность. Отображается всплывающее сообщение (Toast) с текстом «Успешный вход», информирующее пользователя о том, что вход в систему выполнен успешно. Завершается текущая активность, что означает, что пользователь не сможет вернуться к экрану входа, нажав кнопку «Назад».

3.3 Реализация личного кабинета пользователя

В созданном приложении был добавлен функционал личного кабинета для пользователей, позволяющий просматривать персональные данные.

«Личный кабинет (ЛК) – персональная страница сайта или раздел мобильного приложения с личными данными пользователя» [5].

После входа в приложение пользователь получает доступ к меню, соответствующему его роли в системе, где он может воспользоваться различными функциями, включая просмотр своего личного кабинета. В личном кабинете пользователь может увидеть свою электронную почту и Ф.И.О.

На рисунке 28 изображён экран личного кабинета.

Рисунок 28 – Интерфейс личного кабинета

На рисунке 29 представлена информация о пользователе в базе данных.

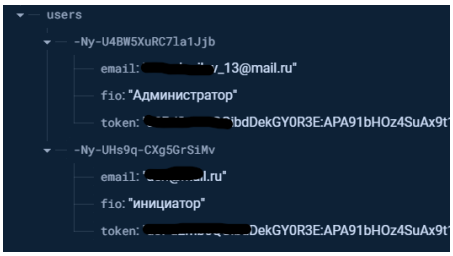

Рисунок 29 – Вид информации о пользователе в базе данных

3.4 Реализация отправки и просмотра заявок

Чтобы функции отправки и просмотра заявок отобразить были выполнены следующие этапы:

Отправка заявки:

– создана форма, позволяющая пользователю ввести информацию о заявке. На рисунке 30 представлен интерфейс для создания новой заявки;

– данные о заявке сохраняются в базе данных Firebase;

– внедрен механизм, автоматически генерирующий время отправки заявки и устанавливающий начальный статус «Открыта» для каждой новой заявки.

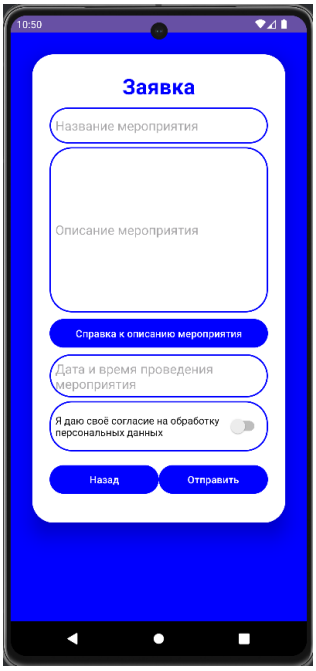

Рисунок 30 – Интерфейс создания новой заявки

Просмотр заявок:

– разработаны интерфейсы для просмотра заявок администраторами и инициаторами. На рисунках 31 и 32 представлены экраны соответственно;

– заявки извлекаются из базы данных Firebase и отображаются в списке с элементами, такими как название мероприятия, описание, дата и время проведения, а также текущий статус;

– администратор может изменять статус заявки, например, устанавливать её как «В работе» или «Закрыта».

Рисунок 31 – Интерфейс заявок для администратора

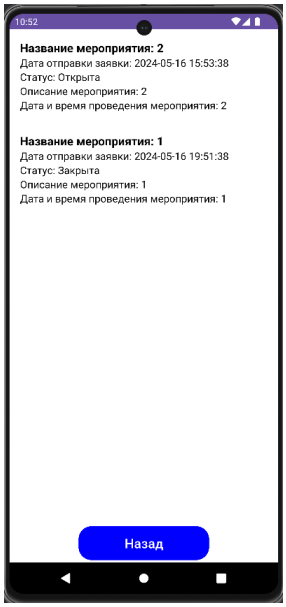

Рисунок 32 – Интерфейс заявок для пользователя

Таким образом, реализация функции отправки и просмотра заявок включает опцию изменения их статуса.

На рисунках 33 и 34 представлены фрагменты кода создания новой заявки.

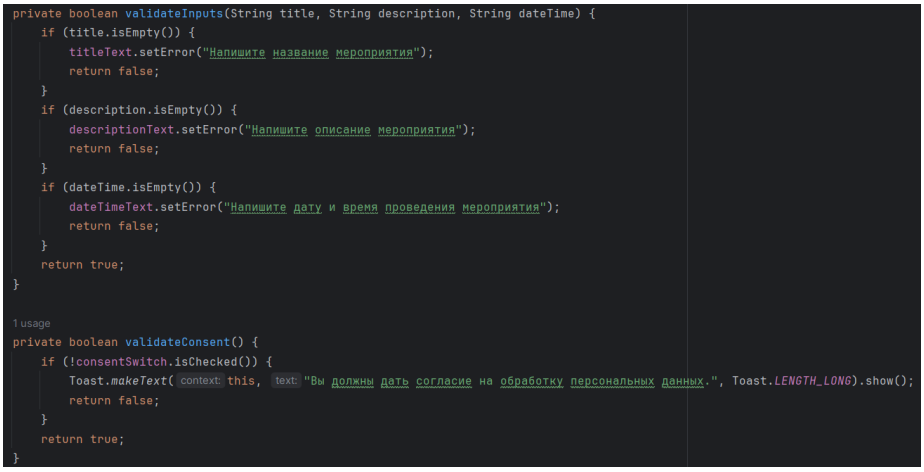

Рисунок 33 – Фрагмент кода создания новой заявки

Метод validateInputs объявлен с модификатором доступа private, что означает, что он может быть вызван только внутри класса, в котором объявлен. Он принимает три строки: title, description и dateTime, и возвращает значение типа boolean, указывающее на то, являются ли данные валидными. Если строка title пустая, устанавливается ошибка для titleText с сообщением «Напишите название мероприятия», и метод возвращает false, указывая на недействительные данные. Если строка description пустая, устанавливается ошибка для descriptionText с сообщением «Напишите описание мероприятия», и метод возвращает false. Если строка dateTime пустая, устанавливается ошибка для dateTimeText с сообщением «Напишите дату и время проведения мероприятия», и метод возвращает false. Если все проверки пройдены успешно, метод возвращает true, указывая на валидные данные.

Метод validateConsent объявлен с модификатором доступа private, что означает, что он может быть вызван только внутри класса, в котором объявлен. Он не принимает параметров и возвращает значение типа boolean, указывающее на наличие согласия. Если переключатель согласия (consentSwitch) не установлен (не включён), отображается всплывающее сообщение (Toast) с текстом «Вы должны дать согласие на обработку

персональных данных.», и метод возвращает false, указывая на отсутствие согласия. Если переключатель согласия установлен (включён), метод возвращает true, указывая на наличие согласия.

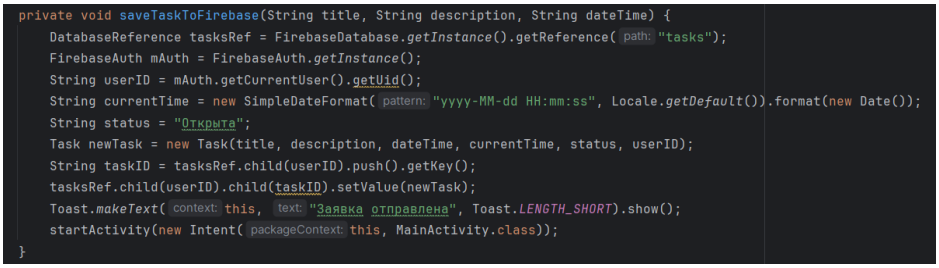

Рисунок 34 – Фрагмент кода создания новой заявки

Метод объявлен с модификатором доступа private, что означает, что он может быть вызван только внутри класса, в котором объявлен. Он принимает три строки: title, description и dateTime, представляющие информацию о заявке. Получается ссылка на узел tasks в базе данных Firebase. Используется объект FirebaseAuth для получения текущего аутентифицированного пользователя. Затем извлекается уникальный идентификатор пользователя (userID). Текущее время форматируется в строку вида «yyyy-MM-dd HH:mm:ss» и сохраняется в переменную currentTime. Устанавливается статус заявки как «Открыта» и сохраняется в переменную status. Создаётся новый объект заявки Task с переданными параметрами и дополнительной информацией (текущим временем, статусом и идентификатором пользователя). Генерируется уникальный идентификатор заявки (taskID) с помощью метода push(), который создаёт уникальный ключ в Firebase. Задача сохраняется в базу данных по пути tasks/userID/taskID. Отображается всплывающее сообщение (Toast) с текстом «Заявка отправлена», информирующее пользователя о том, что заявка успешно сохранена. Создаётся намерение (Intent) для перехода на MainActivity и запускается новая активность.

На рисунке 35 представлен фрагмент кода отображения списка заявок инициатора заявки.

Рисунок 35 – Фрагмент кода отображения списка заявок инициатора заявки

Метод onDataChange вызывается, когда данные в указанном узле базы данных Firebase изменяются. Он принимает параметр dataSnapshot, представляющий снимок текущих данных. Этот цикл for перебирает все дочерние элементы dataSnapshot, каждый из которых представляет собой отдельную задачу. Из каждого taskSnapshot извлекаются значения полей задачи: title, currentTime, status, description и dateTime. Используется метод getValue для получения значений в виде строк (String.class). Если все извлечённые значения не равны null, продолжается обработка задачи. Создаётся новое представление задачи taskView, используя макет R.layout.item_task. Затем извлечённые данные задачи устанавливаются в соответствующие текстовые поля этого представления: название мероприятия, дата отправки заявки, статус, описание мероприятия, дата и время проведения мероприятия. Новое представление задачи добавляется в макет user_tasks_layout. Параметр 0 указывает на то, что представление добавляется в начало списка, то есть оно будет первым в отображаемом списке задач.

Этот метод обеспечивает динамическое создание и отображение списка

задач, полученных из базы данных Firebase, в пользовательском интерфейсе.

3.5 Тестирование мобильного приложения

Тестирование проводилось на различных устройствах. В таблице ниже приведены устройства и версии.

Таблица 6 – Устройства, на которых проводилось тестирование

Устройство	Версия
Honor 70	Android 13
Huawei P30	Android 12
Huawei P30 lite	Android 12

Проверку функциональности мобильного приложения проверим ниже [13].

Аутентификация пользователей:

– верификация успешного входа с зарегистрированным аккаунтом;

– тестирование обработки ошибок при вводе некорректных учетных данных;

– проверка создания и регистрации нового пользователя;

– проверка доступности функций в зависимости от прав доступа пользователя.

Создание и отправка заявок:

– проверка создания новой заявки с заполнением всех обязательных полей (название, описание, дата и время мероприятия);

– проверка сохранения заявки в базе данных.

Просмотр информации о пользователе:

– проверка возможности просмотра почты пользователя;

– проверка возможности просмотра Ф.И.О. пользователя.

Обновление статуса заявок:

– проверка изменения статуса заявки на «В работе» и «Закрыта»;

– проверка обновления статуса в базе данных.

Обработка ошибок и исключительных ситуаций:

– проверка корректного реагирования на неправильный ввод почты и пароля;

– проверка корректного реагирования на пустые обязательные поля.

По результатам тестирования выявлено, что все ключевые функции приложения работают корректно и соответствуют заявленным требованиям.

Выводы по главе 3

В третьей главе представлено краткое описание разработанного решения. Подробно рассмотрена реализация ключевых функций приложения. Функциональное тестирование продемонстрировало, что основные функции приложения работают корректно и соответствуют требованиям.

Заключение

В рамках данного исследования мы поставили перед собой задачу разработки уникального мобильного приложения, предназначенного для сбора заявок на информационное освещение мероприятия с целью оптимизации и улучшения процесса обработки запросов. Учитывая актуальность данной проблемы и важность повышения эффективности и удовлетворенности сотрудников, разработка такого приложения приобретает значимость в практике.

В ходе исследования был проведен анализ бизнес-процессов подразделения, включая описание его структуры и выбор инструментов CASE для моделирования. Также был выполнен анализ текущих моделей бизнес-процессов с использованием диаграмм, что позволило глубже понять их характеристики и определить области для улучшения.

На основании данного анализа были выработаны требования к мобильному приложению. Мы также осуществили обзор и анализ существующих мобильных приложений для сбора заявок, что позволило выявить их преимущества и недостатки.

С учетом определенных требований и проведенного анализа была начата разработка мобильного приложения. Разработан дизайн приложения, выбрана платформа и инструменты для разработки, а также создана модель данных. Архитектура мобильного приложения основана на стандартных компонентах Android.

Внедрение приложения было осуществлено с применением выбранной платформы и инструментов. По завершении разработки было проведено тестирование, чтобы удостовериться в его функциональности и соответствии заданным требованиям.

В результате работы мы достигли поставленной цели - разработали мобильное приложение для сбора заявок на информационное освещение мероприятия.

Список используемой литературы

1. Афанасьев А. В. Актуальность языка программирования Java в разработке мобильных приложений в 2020 году //Информационные технологии в процессе подготовки современного специалиста. – 2020. – С. 25-29.
2. Афанасьев А. Н., Войт Н. Н. Разработка компонентно-сервисной платформы обучения: диаграммы классов программного компонента сценария на UML-языке //Вестник Ульяновского государственного технического университета. – 2012. – №. 2 (58). – С. 32-36.
3. Бежик А. А., Мажей Я. В. Архитектура приложений. Чем является? Для чего используется? Основные виды и критерии хорошей архитектуры //Столыпинский вестник. – 2022. – Т. 4. – №. 9. – С. 4998-5008.
4. Вязовик Н., Жилин Е. Программирование на Java //М.: Интуит. ру. – 2003.
5. Для чего создавать личный кабинет? [Электронный ресурс] / Режим доступа: URL: https://mobile.i-neti.ru/blog/dlya-chego-sozdavat-lichnyj-kabinet (дата обращения 06.03.2024).
6. Есть talk! – это. [Электронный ресурс] / Режим доступа: URL: https://talk-on.ru/o-nas/detail.php?ELEMENT_ID=166 (дата обращения 25.02.2024).
7. Зиятдинова А., Староверова Н. А. Аналитический обзор и сравнение возможностей операционных систем для мобильных устройств. – 2015. – №. 9-2. – С. 227-231. 78
8. Иванов В. В., Лубова Е. С., Черкасов Д. Ю. Аутентификация и авторизация. – 2017. – №. 2 (84). – С. 31-33.
9. Микляев И. А., Ундозерова А. Н., Кудаева М. В. Универсальная логическая модель базы данных //Arctic Environmental Research. – 2010. – №. 1. – С. 93-98.

10. Оптимизация бизнес-процессов предприятия. [Электронный ресурс] / Режим доступа: URL: https://piter-consult.ru/home/Articles/Simply-about-the-difficult/Let-entrust-business-processes.html (дата обращения 25.02.2024).

11. Подольская О. В., Коржов С. А. Инфологическое проектирование //International innovation research. – 2017. – С. 11-13.

12. Тарасов С. В. СУБД для программиста. Базы данных изнутри. – 2015.

13. Юрченко А. Н. Функциональное тестирование как одна из методологий тестирования программного обеспечения //Вестник современных исследований. – 2018. – №. 1.1. – С. 155-156.

14. Ramus. [Электронный ресурс] / Режим доступа: URL: https://softrare.space/ru/windows/ramus/ (дата обращения 26.02.2024).

15. Android Studio. [Электронный ресурс] / Режим доступа: URL: https://developer.android.com/studio/intro (дата обращения 06.03.2024).

16. Eeles P. Capturing architectural requirements //IBM Rational developer works. – 2005.

17. Firebase for Android. [Электронный ресурс] / Режим доступа: URL: https://firebase.google.com/docs/android/setup?authuser=0&hl=en (дата обращения 26.02.2024).

18. Goadrich M. H., Rogers M. P. Smart smartphone development: iOS versus Android //Proceedings of the 42nd ACM technical symposium on Computer science education. – 2011. – С. 607-612.

19. Mobile Operating System Market Share. [Электронный ресурс] / Режим доступа: URL: https://gs.statcounter.com/os-market-share/mobile/worldwide (дата обращения 01.05.2024).

20. Nimodia C., Deshmukh H. R. Android operating system //Software Engineering. – 2012. – Т. 3. – №. 1. – С. 10.

Приложение А

Код регистрации нового пользователя

```
public class SignUpActivity extends AppCompatActivity {

    private FirebaseAuth auth;
    private EditText signupEmail, signupFio;
    private TextInputEditText signupPassword;
    private TextInputLayout passwordTextInputLayout;
    private Button signupButton;
    private TextView loginRedirectText;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        EdgeToEdge.enable(this);
        setContentView(R.layout.activity_sign_up);
        initViews();
        setupListeners();
    }

    private void initViews() {
        auth = FirebaseAuth.getInstance();
        signupEmail = findViewById(R.id.signup_email);
        signupPassword = findViewById(R.id.signup_password);
        signupFio = findViewById(R.id.signup_fio);
        passwordTextInputLayout = findViewById(R.id.passwordTextInputLayout);
        signupButton = findViewById(R.id.signup_button);
        loginRedirectText = findViewById(R.id.loginRedirectText);
    }
```

```
    private void setupListeners() {
        passwordTextInputLayout.setEndIconOnClickListener(v ->
togglePasswordVisibility());
        signupButton.setOnClickListener(this::attemptSignUp);
        loginRedirectText.setOnClickListener(v -> startActivity(new Intent(this,
LoginActivity.class)));
    }

    private void togglePasswordVisibility() {
        if
(signupPassword.getTransformationMethod().equals(PasswordTransformationMet
hod.getInstance())) {

signupPassword.setTransformationMethod(HideReturnsTransformationMethod.get
Instance());
        } else {

signupPassword.setTransformationMethod(PasswordTransformationMethod.getIns
tance());
        }
        signupPassword.setSelection(signupPassword.getText().length());
    }

    private void attemptSignUp(View v) {
        String user = signupEmail.getText().toString().trim();
        String pass = signupPassword.getText().toString().trim();
        String fio = signupFio.getText().toString().trim();
```

```
  if (validateInput(user, pass, fio)) {

FirebaseMessaging.getInstance().getToken().addOnCompleteListener(this::handle
SignUp);
      }
  }

  private boolean validateInput(String user, String pass, String fio) {
      boolean isValid = true;

      if (user.isEmpty()) {
          signupEmail.setError("Введите почту");
          isValid = false;
      } else {
          signupEmail.setError(null);
      }

      if (pass.isEmpty()) {
          signupPassword.setError("Введите пароль");
          isValid = false;
      } else if (pass.length() < 8) {
          signupPassword.setError("Пароль должен содержать минимум 8
символов");
          isValid = false;
      } else {
          signupPassword.setError(null);
      }
```

```
    if (fio.isEmpty()) {
            signupFio.setError("Введите Ф.И.О.");
            isValid = false;
        } else {
            signupFio.setError(null);
        }

        return isValid;
    }

    private void handleSignUp(Task<String> task) {
        if (task.isSuccessful()) {
            String token = task.getResult();
            DatabaseReference usersRef =
FirebaseDatabase.getInstance().getReference("users");
            String email = signupEmail.getText().toString().trim();
            String fio = signupFio.getText().toString().trim();
            String userId = FirebaseAuth.getInstance().getCurrentUser().getUid();

            User newUser = new User(email, fio, token);

            // Список адресов электронной почты администраторов
            List<String> adminEmails = Arrays.asList("den_danilov_13@mail.ru",
"another_admin@example.com", "yet_another_admin@example.com");

            // Проверяем, является ли пользователь администратором
            if (adminEmails.contains(email)) {
                usersRef.child("admin").child(userId).setValue(newUser);
```

```
} else {
            usersRef.child("iniciator").child(userId).setValue(newUser);
        }

        auth.createUserWithEmailAndPassword(email,
signupPassword.getText().toString().trim())
                .addOnCompleteListener(this::handleAuthResult);
    }
  }

  private void handleAuthResult(Task<AuthResult> task) {
    if (task.isSuccessful()) {
        Toast.makeText(SignUpActivity.this, "Регистрация успешна",
Toast.LENGTH_SHORT).show();
        startActivity(new Intent(SignUpActivity.this, LoginActivity.class));
    } else {
        Toast.makeText(SignUpActivity.this, "Регистрация не удалась: " +
task.getException().getMessage(), Toast.LENGTH_LONG).show();
    }
  }

  public static class User {
    public String email;
    public String fio;
    public String token;

    public User(String email, String fio, String token) {
        this.email = email;
```

```
            this.fio = fio;
             this.token = token;
        }
    }
}
```