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

Институт Математики, физики и информационных технологий
(наименование института полностью)

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 «Планировщик задач»

Обучающийся Е.Д. Говорова
(Инициалы Фамилия) (личная подпись)

Руководитель канд. пед. наук, доцент, О.А. Крайнова
(ученая степень (при наличии), ученое звание (при наличии), Инициалы Фамилия)

Тольятти 2023

Аннотация

Выпускная работа представлена в виде пояснительной записки объемом 48 страниц, включает введение на двух страницах, 30 рисунков, 3 таблиц и список из 20 источников, в том числе 9 на английском языке.

Объектом исследования является процесс планирования задач, а предметом исследования – автоматизация процесса планирования задач. Целью бакалаврской работы является разработка данного мобильного приложения.

Актуальность работы обусловлена отсутствием в МАУ «МФЦ» мобильного приложения, которое бы позволяло планировать задачи и напоминать пользователю о них. Работа структурирована следующим образом: введение, 3 главы, заключение и приложение.

В первой главе производится анализ бизнес-процессов предприятия, а именно в разделе 1.4 разработаны требования к мобильному приложению «Планировщик задач», в разделе 1.5 проведен обзор и анализ аналогов мобильных приложений для планирования задач, в разделе 1.6 сформулирована задача на разработку мобильного приложения «Планировщик задач».

Во второй главе осуществляется проектирование мобильного приложения «Планировщик задач», для разработки системы выбрана платформа Android Studio. Анализ языков программирования и интегрированной среды разработки приложений привел к выводу, что наиболее подходящим выбором является язык программирования Java в сочетании с Android Studio.

В третьей главе описывается реализация мобильного приложения «Планировщик задач», в разделе 3.2 описана реализация создания задачи, в разделе 3.3 описана реализация оповещения пользователя, в разделе 3.4 проведено тестирование мобильного приложения «Планировщик задач».

Abstract

The graduation work consists of an explanatory note on 48 pages, including 30 figures, 3 tables, the list of 20 references including 9 foreign sources. The paper pays attention to both theoretical and practical aspects of mobile application development, as well as a study of existing solutions in this field.

The object of the research is the task planning process, and the subject of the research is the automation of the task planning process. The aim of the bachelor's thesis is to develop this mobile application.

The urgency of the work is caused by absence of mobile application in MFC which would allow scheduling tasks and reminding user about them. The paper is structured as follows: introduction, 3 chapters, conclusion and appendix.

Chapter one contains analysis of the business processes in the enterprise, namely in section 1.4 the requirements for the mobile application "Task Scheduler" were developed, in section 1.5 the review and analysis of mobile applications for task scheduling, in section 1.6 the problem of developing the mobile application "Task Scheduler" was formulated.

In the second chapter, the design of the mobile application "Task Scheduler" is carried out, the Android Studio platform is chosen for the development of the system. The analysis of programming languages and integrated application development environment led to the conclusion that Java programming language in combination with Android Studio is the most suitable choice.

The third chapter describes the implementation of the mobile application "Task Scheduler", section 3.2 describes the task creation implementation, section 3.3 describes the user notification implementation, section 3.4 tests the mobile application "Task Scheduler".

ОГЛАВЛЕНИЕ

Введение

В настоящее время мобильные приложения стали неотъемлемой частью нашей повседневной жизни. Они значительно упрощают и улучшают наши возможности в различных сферах, включая планирование и организацию задач. Отслеживание и управление задачами является важным аспектом достижения целей и повышения продуктивности.

Актуальность данной бакалаврской работы обусловлена необходимостью улучшения планирования задач в современной информационной среде. Мобильные приложения, обладающие функциональностью планировщика задач, становятся все более популярными и востребованными. Однако в МАУ «МФЦ» отсутствует специализированное мобильное приложение, что ограничивает возможности работников в эффективной организации своего рабочего времени.

Объектом исследования данной работы является процесс планирования задач, а предметом исследования - автоматизация процесса планирования задач.

Целью данного дипломного проекта является разработка мобильного приложения «Планировщик задач», которое позволит пользователям эффективно управлять своим временем, планировать и отслеживать выполнение задач на своих мобильных устройствах.

Для достижения поставленной цели будут выполнены следующие задачи:

- Анализ бизнес-процессов предприятия: проведение обзора особенностей процесса планирования в МАУ «МФЦ» и выявление требований к мобильному приложению для планирования задач;
- Проектирование мобильного приложения: выбор средств разработки, разработка модели данных и определение архитектуры программного продукта;

– Реализация мобильного приложения: создание функциональности, включающей создание задач, оповещение пользователей;
– Тестирование мобильного приложения: проведение тестовых сценариев для проверки корректной работы основных функций приложения.

В рамках работы предусмотрены три основные главы.

В первой главе осуществляется анализ существующих приложений, используемых для планирования задач. Проводится анализ существующих аналогов мобильных приложений для планирования задач и их функциональных возможностей. Также проводится сравнение приложений, что поможет определить основные требования и функции, которые должны быть реализованы в разрабатываемом приложении. Выбирается платформа для реализации приложения и создаётся диаграммы вариантов использования для наглядного представления взаимодействия пользователя с мобильным приложением.

Во второй главе происходит проектирование разрабатываемого мобильного приложения, выбор оптимальных средств разработки и архитектуры программного продукта. Также разрабатываются логическая и физическая модели данных, которые определяют структуру и хранение информации в приложении.

В третьей главе осуществляется реализация мобильного приложения. Демонстрируется реализация функций создания задачи и оповещения пользователя, которые обеспечивают удобное взаимодействие пользователя с приложением. Проводится демонстрация работы приложения, а также процесс тестирования для проверки его функциональности, стабильности и соответствия поставленным требованиям.

Данная работа имеет практическую значимость для МАУ «МФЦ», которая заключается в возможности использования разработанного приложения для улучшения планирования и оптимизации рабочего процесса.

Глава 1 Анализ бизнес-процессов предприятия

1.1 Краткая характеристика предприятия

МФЦ (Многофункциональный центр предоставления государственных и муниципальных услуг) - это учреждение, где граждане и организации могут получать государственные и муниципальные услуги, а также осуществлять прием, регистрацию и выдачу документов. Концепция административной реформы и план мероприятий по ее проведению в России были приняты 25 октября 2005 года с целью повышения доступности и качества государственных услуг. Реализация проектов по созданию МФЦ началась во втором полугодии 2007 года и к 1 декабря 2008 года в России уже функционировали 25 МФЦ в 16 регионах. К концу 2011 года МФЦ были насчитывались в 62 регионах страны. На начало 2016 года в России функционировало 2,7 тыс. центров и 10,1 тыс. офисов государственных и муниципальных услуг [10].

МФЦ обеспечивают гражданам быстрый и комфортный доступ к услугам, а также предоставляют возможность получать несколько взаимосвязанных услуг одновременно, а также предоставляют достоверную и актуальную информацию, необходимую для получения услуг.

Организационная структура МФЦ представлена на рисунке 1 и основывается на директоре МФЦ. Директор отвечает за организацию работы МФЦ, включая взаимодействие с органами власти, предоставляющими государственные и муниципальные услуги через МФЦ. Он также обеспечивает соблюдение законодательства при оказании услуг. На данный момент действующим директором МФЦ города Тольятти является Елена Владимировна Рослякова.

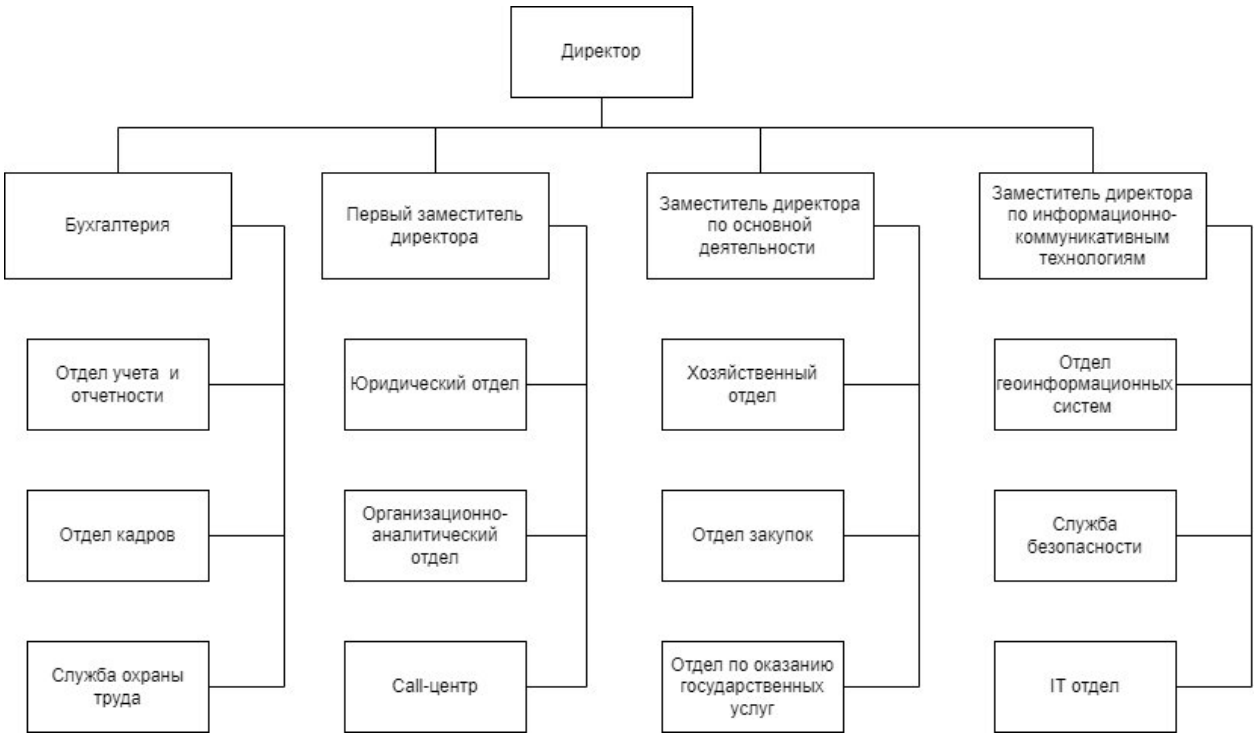

Рисунок 1 - Организационная структура МФЦ

Бухгалтерия предприятия занимается сбором данных об имуществе и обязательствах и включает в себя несколько отделов, таких как отдел учета и отчетности, отдел кадров и службу охраны труда.

Первый заместитель директора управляет несколькими отделами, включающие в себя юридический отдел, организационно-аналитический отдел и call-центр.

Заместитель директора по основной деятельности руководит хозяйственным отделом, отделом закупок и отделом по оказанию государственных услуг.

Заместитель директора по информационно-коммуникативным технологиям отвечает за работу отдела геоинформационных систем, службы безопасности и IT-отдела.

Главная цель отдела геоинформационных систем (ОГИС) заключается в обеспечении органов государственной власти, органов местного самоуправления и юридических лиц географически привязанной

информацией, необходимой для выполнения функций, предусмотренных законодательством.

ОГИС выполняет следующие функции:

– Участие в разработке и реализации муниципальных программ в области информатизации, включая программу развития ГИС округа;
– Внедрение географической информационной системы;
– Формирование базовых слоев и таблиц ГИС;
– Координация работ по созданию специализированных отраслевых ГИС, а также контроль за их совместимостью, полнотой и актуальностью.

На данный момент МФЦ существует уже 21 год и успешно предоставляет услуги гражданам.

1.2 Выбор CASE-средств для описания бизнес-процессов

Для проектирования необходимо выбрать CASE-средства (Computer Aided System/Software Engineering) для описания бизнес-процессов, этап проектирования один из важных этапов разработки мобильного приложения. Одним из таких CASE-средств является Ramus 2.0 – это программное обеспечение, которое представляет собой компьютерный инструмент для анализа, моделирования и разработки систем. Ramus поддерживает методики моделирования бизнес-процессов IDEF0 (Integration Definition for Function Modeling) и DFD (data flow diagrams), а также обладает рядом дополнительных возможностей, которые призваны удовлетворять потребности коллектива проектировщиков систем управления компаниями [5].

Еще одним CASE-средством является Microsoft Visio - векторный графический редактор, редактор диаграмм и блок-схем для Windows. Все версии поддерживают традиционные функции MS Office Word и Excel, к примеру настройку шрифта и цвета, а также позволяют импортировать данные

напрямую из MS Excel и Access. Visio также располагает библиотекой фигур и шаблонов, для создания разных типов схем [2].

Еще одним вариантом является Diagrams.net - это бесплатное приложение для построения диаграмм, позволяющее пользователям создавать схемы, модели и диаграммы в веб-браузере и обмениваться ими. Оно совместимо с Google Suite и Dropbox, а также глубоко интегрировано с продуктами Atlassian Confluence и Jira. Diagrams.net позволяет работать с диаграммами в автономном режиме и сохранять их локально с помощью настольной программы для macOS, Windows и Linux. Пользователи могут отслеживать и восстанавливать изменения, импортировать и экспортировать различные форматы, а также автоматически публиковать работы и делиться ими. Приложение также интегрируется с облачными сервисами для хранения данных, таких как Dropbox, OneDrive, Google Drive, GitHub и GitLab.com [13].

Чтобы выбрать подходящее CASE-средств были выбраны такие критерии для сравнительной оценки средств, как: совместная работа пользователей, импорт и экспорт в различные форматы, возможность работать в разных нотациях, совместная работа с другими программами, бесплатная программа.

Как можно видеть в таблице 1 наиболее подходящим является средство Diagrams.net, которое соответствует всем критериям для сравнительной оценки CASE-средств для описания бизнес-процессов.

Таблица 1 - Критерии для сравнительной оценки CASE-средств.

	Ramus 2.0	Microsoft Visio	Diagrams.net
Совместная работа пользователей	+	+	+
Импорт и экспорт в различные форматы	+	+	+

Продолжение таблицы 1

	Ramus 2.0	Microsoft Visio	Diagrams.net
Возможность работать в разных нотациях	-	+	+
Совместная работа с другими программами	-	+	+
Импорт и экспорт в различные форматы	+	-	+

В таблице показано, что Ramus 2.0 не соответствует таким критериям, как возможность работать в разных нотациях и совместная работа с другими программами. CASE-средств Microsoft Visio не соответствует только одному критерию - бесплатная программа, а Diagrams.net соответствует всем критериям.

1.3 Анализ модели бизнес-процесса

Для того, чтобы в данное время работник МФЦ мог планировать свою деятельность, ему необходимо самому искать, где записывать задачи и в какой форме. Контекстная диаграмма данного процесса представлена на рисунке 2.

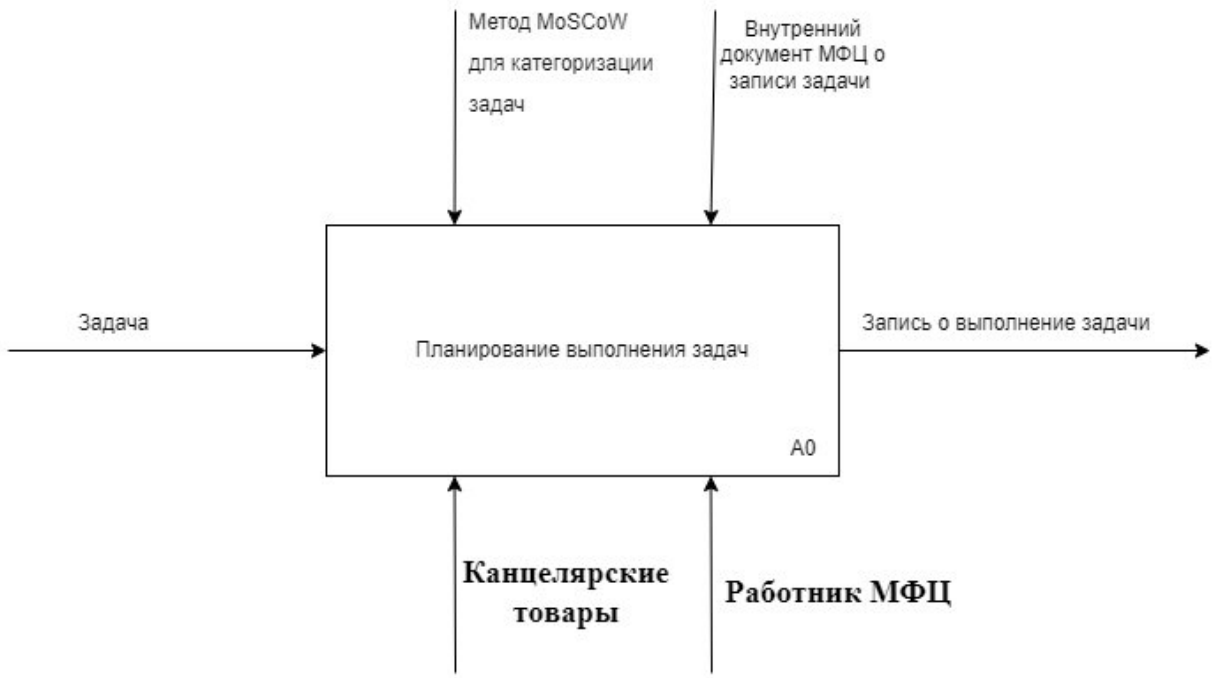

Рисунок 2 - Контекстная диаграмма процесса

Для наглядности и простоты понимания декомпозиция блока А0 представлена на рисунке 3.

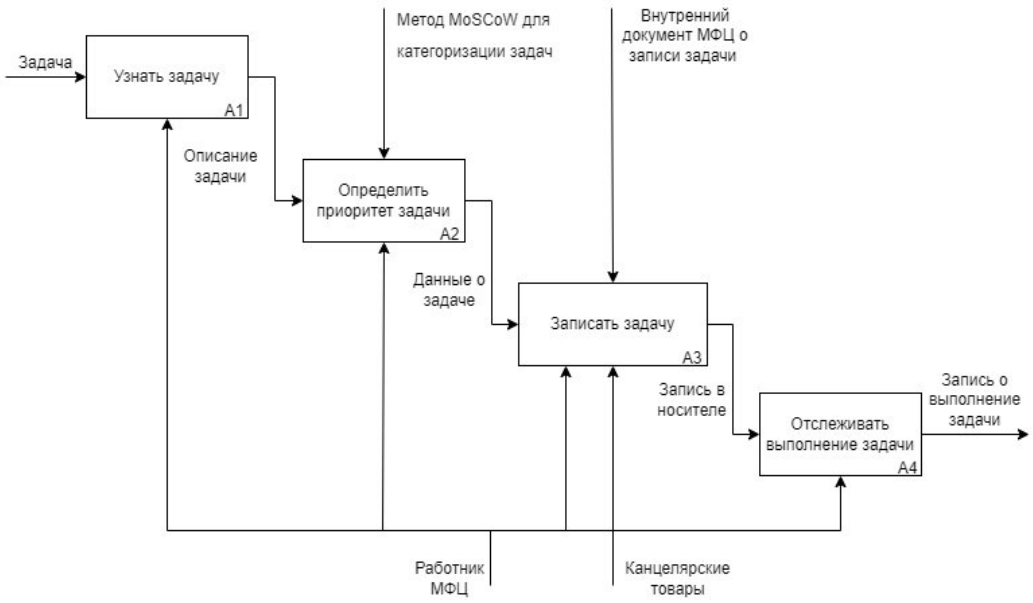

Рисунок 3 - Декомпозиция блока А0 функциональной модели «Планирование выполнения задач»

Сегодня методология MoSCoW широко известна и применяется в различных областях управления по всему миру. Акроним MoSCoW состоит из согласных букв, которые представляют приоритетность задач и требований:

– M (Must) - задачи и требования с самым высоким приоритетом, которые должны быть реализованы в первую очередь. Эти элементы являются неотъемлемой частью продукта, и без их выполнения релиз будет невозможен;
– S (Should) - важные требования, но с немного более низким приоритетом. Хотя они не имеют решающего значения, их выполнение все же является обязательным;
– C (Could) - желательные требования и задачи, которые могут быть включены в релиз, если это возможно. Они не являются критическими, но их выполнение предпочтительно;
– W (Would) - наименее критичные требования, которые могут быть отложены или проигнорированы до будущих релизов. Их выполнение не является обязательным в текущем этапе разработки.

Методология MoSCoW обеспечивает систематическую классификацию требований и задач по их приоритетности, позволяя определить, какие элементы являются наиболее важными и необходимыми [18].

Чтобы ещё лучше понять процесс, выполним декомпозицию блоков A3 и A4, которые представлены на рисунках 4 и 5 соответственно.

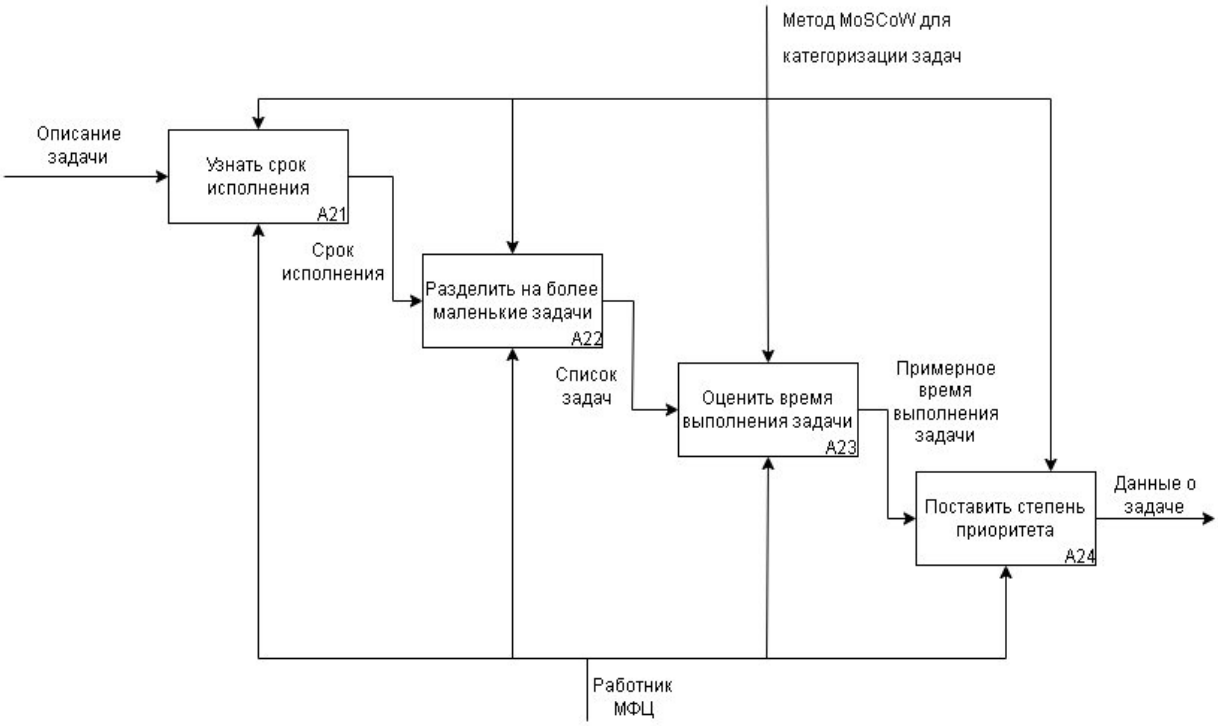

Рисунок 4 - Декомпозиция блока A2

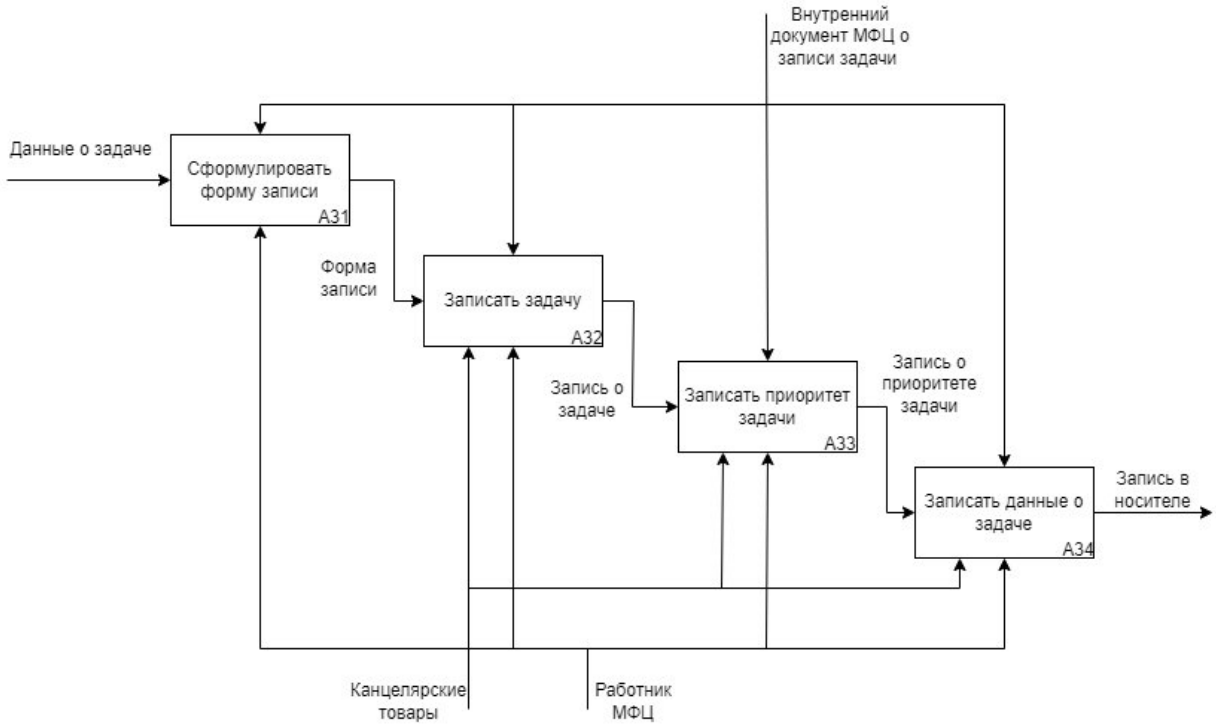

Рисунок 5 - Декомпозиция блока A3

С помощью DFD-диаграммы визуально покажем процесс с точки зрения данных на рисунке 6.

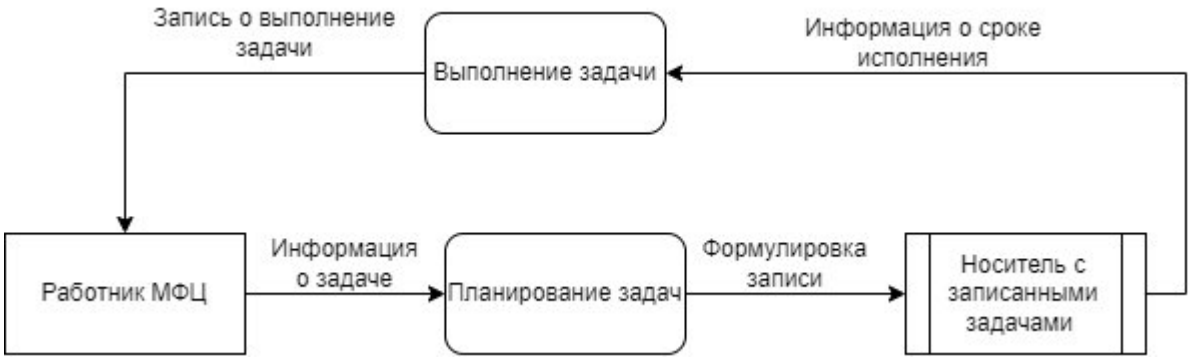

Рисунок 6 - DFD-диаграмма процесса планирования задач

В результате были рассмотрена контекстная диаграмма и её декомпозиции, а также DFD-диаграмма данных.

1.4 Обзор и анализ аналогов мобильных приложений для планирования задач

Рынок мобильных приложений достаточно богат, рассмотрим некоторые мобильные приложения для планирования задач из таких магазинов приложений как Google Play, App Store.

Google Play - это цифровая платформа, разработанная компанией Google, которая предоставляет доступ к широкому выбору приложений, игр, фильмов, музыки, книг и другого контента для устройств на операционной системе Android. Она является официальным магазином приложений для Android, предлагая пользователям возможность просматривать, загружать и устанавливать разнообразные программы и игры на свои смартфоны, планшеты и другие устройства. Пользователи могут просматривать описания, оценки и отзывы о приложениях, чтобы принять информированное решение

перед их установкой. Кроме приложений, Google Play предлагает также доступ к фильмам, телешоу, музыке и электронным книгам. Пользователи могут покупать или арендовать фильмы и телешоу, стримить музыку или приобретать отдельные треки, а также читать электронные книги на своих устройствах. Google Play является основным источником контента для миллионов пользователей Android-устройств по всему миру и предоставляет удобный способ получить доступ к разнообразному контенту для удовлетворения различных потребностей и интересов.

App Store - официальный магазин приложений, который предоставляет доступ к огромному выбору приложений для устройств на операционной системе iOS, таких как iPhone, iPad и iPod Touch. Данный магазин предлагает широкий спектр приложений, которые охватывают различные категории, включая социальные сети, игры, развлечения, образование, здоровье, фотографию, финансы и многое другое. Каждое приложение проходит процесс ручной модерации и проверки со стороны Apple, чтобы обеспечить безопасность, качество и соответствие стандартам перед его публикацией в App Store. Пользователи могут просматривать описания приложений, смотреть скриншоты и видеообзоры, читать отзывы и оценки других пользователей, чтобы принять информированное решение перед загрузкой и установкой приложения. Многие приложения предлагаются как бесплатные, так и платные версии, и пользователи могут совершать покупки внутри приложений для разблокировки дополнительных функций или контента.

В процессе сбора данных для решения задачи автоматизации планирования деятельности был проанализирован рынок программных продуктов, способных решить поставленную задачу. В качестве критериев сравнения были выбраны следующие:

- Наличие русского языка в интерфейсе;
- Наличие функции напоминания;
- Возможность добавлять описание задач;

– Стабильная работа приложения.

Такой критерий, как наличие русского языка в интерфейсе показывает, что в мобильном приложение предусмотрен русский язык, как язык интерфейса. Критерий наличие функции напоминания отвечает за возможность присылать мобильным приложением уведомления на телефон, которое содержит информацию о задаче и напоминает пользователю, что нужно выполнить эту задачу. Возможность добавлять описание задач важный критерий, потому что задачи бывают с объемным описанием, которое нужно записать, для того чтобы пользователь не забывал, что именно нужно сделать, необходимо описание задачи. Критерий стабильная работа приложения показывает, что приложение работает без ошибок и стабильно, то есть без сбоев.

Any.do в Google Play, App Store – простой планировщик задач, который включает в себя список дел, календарь, почтовый ящик, блокнот для заметок, контрольный список, список задач, приложение для напоминаний. У приложения присутствуют такие функции, как: присваивание задачам приоритеты, возможность комментирования задач, возможность прикреплять файлы, разделение задач на подзадачи и возможность совместной работы.

ToDoist в Google Play – это планировщик заданий, доступный на различных платформах, включая Android, iOS, Windows, а также имеющий веб-версию. Основной функционал ToDoist: добавление задач, добавление в раздел «Избранное», создание собственных разделов и подразделов, расстановка приоритетов для задач, получение уведомлений.

Wunderlist в Google Play, App Store – планировщик задач и дел, разработанный немецкой командой программистов, который включает в себя такие функции, как: составление списков и заданий, синхронизация созданных списков и напоминаний, а также возможность напоминания задач.

Weeek в Google Play – планировщик с обширным функционалом и большим количеством инструментов. В приложение предусмотрены

следующие инструменты: канбан-доски, недельный календарь, управление задачами, гибкие уведомления и таймер задач.

Singularity app в Google Play – кроссплатформенный планировщик задач и дел, который содержит все основные инструменты для управления делами: задачи, проекты, планы и чек-листы. Из интересных возможностей: голосовой ввод задач в мобильном приложении и встроенный функционал популярного инструмента тайм-менеджмента «pomodoro».

To Do Reminder with Alarm в Google Play – простое приложение для управления задачами. Его можно рассматривать скорее, как менеджер напоминаний, чем как планировщик задач. Наиболее полезные функции: голосовой ввод Google, возможность настроить напоминание за определенное время до начала события и добавление дней рождения друзей.

Все вышеописанные программные продукты выполняют возложенные на них функции, но также приложения имеют ряд недостатков. В связи с этим был произведен сравнительный анализ мобильных приложений, результаты анализа представлены в таблице 2.

Таблица 2 – Сравнительный анализ мобильных приложений

Название приложения	Наличие русского языка в интерфейсе	Наличие функции напоминания	Возможность добавлять описание задач	Стабильная работа приложения
Any.do	+	+	-	+
ToDoist	+	-	+	+
Wunderlist	+	+	-	+
Weeek	+	+	+	-
Singularity app	+	+	-	+
To Do Reminder with Alarm	-	+	+	+

В таблице показано, что не одно из приложений не соответствует всем критериям, это вызывает необходимость в создании нового приложения, которое будет соответствовать всем критериям.

1.5 Разработка требований к мобильному приложению «Планировщик задач»

Мобильное приложение «Планировщик задач» должно соответствовать функциональным требованиям FURPS. Сокращение FURPS расшифровывается так:

1) Functionality, функциональность;
2) Usability, удобство использования;
3) Reliability, надежность;
4) Performance, производительность;
5) Supportability, поддерживаемость;

Функциональные требования содержат основные свойства/функции приложения [17]. В приложении «Планировщик задач», к примеру безопасность реализована созданной в самом приложении базой данных, что позволяет избежать несанкционированного доступа к определенным ресурсам информации.

Удобство использования заключается в подсказках для пользователя, какую информацию нужно вводить, также интерфейс содержит минимальный набор функций, поэтому приложение «Планировщик задач» обладает дружественным интерфейсом.

Надежность приложения проявляется в возможности использовать его 24 часа в сутки 7 дней в неделю, приложение работает постоянно и у пользователя в любой момент времени есть доступ к нему.

К производительности системы относятся скорость работы, время отклика программы, потребление ресурсов. Приложение «Планировщик задач» обладает минимальным временем отклика и потреблением ресурсов, а также максимальной скоростью работы приложения.

Поддерживаемость заключается в возможности установить приложение на любое мобильное устройство с операционной системой Android, а также в возможности быстро установить обновление.

1.6 Постановка задачи на разработку мобильного приложения «Планировщик задач»

Чтобы ускорить процесс планирования задач и добавить возможность напоминания о задаче, необходимо модифицировать бизнес-процесс, добавив мобильное приложение «Планировщик задач». Это поможет оптимизировать процесс планирования задач и сохранит время работника, также задачи не потеряются на бумажных носителях, потому что собраны в одном месте. Контекстная диаграмма процесса с программой представлена на рисунке 7.

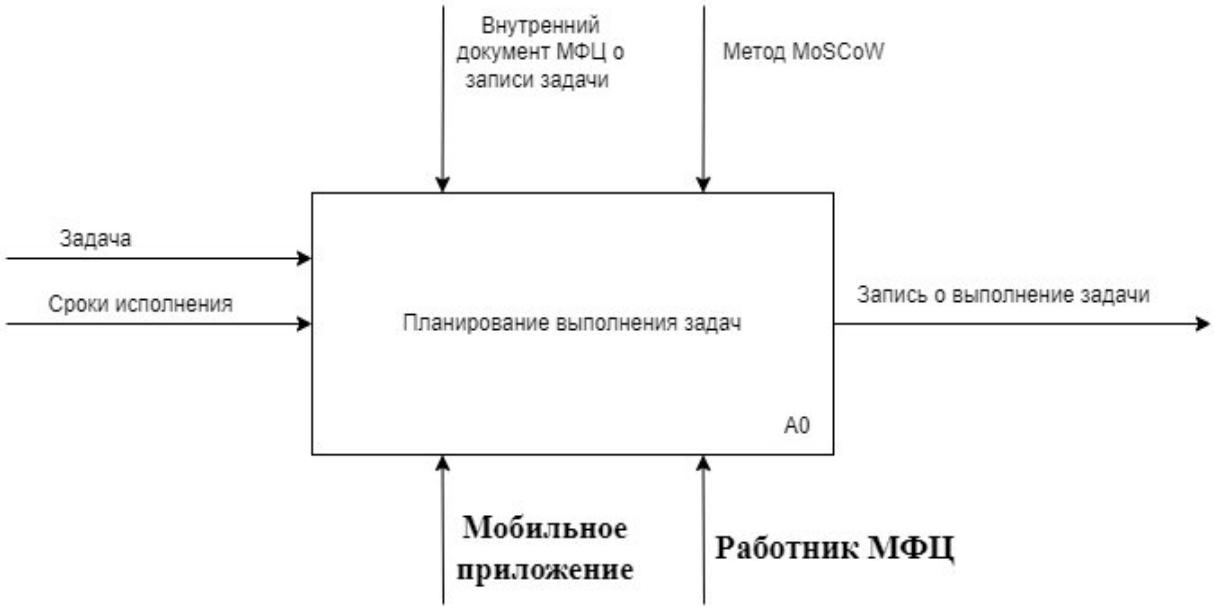

Рисунок 7 - Контекстная диаграмма процесса с программой

Для лучшего понимания и восприятия декомпозиция блока A0 представлена на рисунке 8. В мобильное приложение будут встроены основы метода MoSCoW.

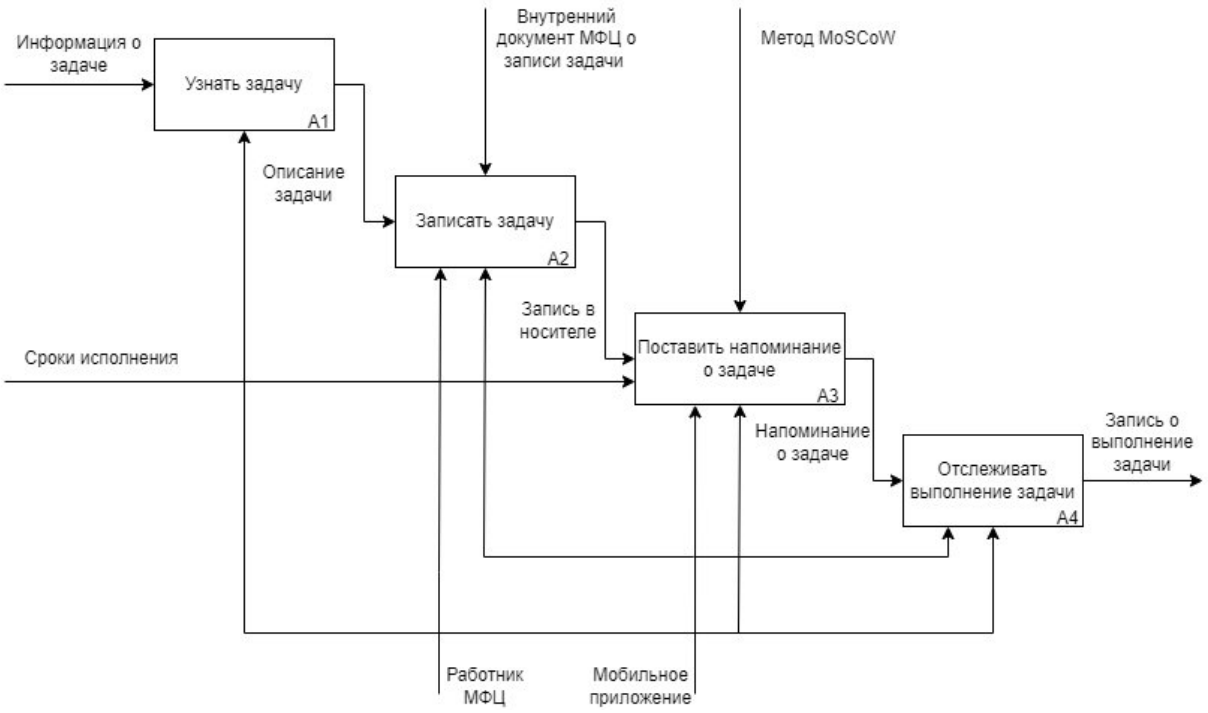

Рисунок 8 - Декомпозиция блока A0 функциональной модели «Планирование выполнения задач»

Для наглядности и простоты понимания, выполним декомпозицию блоков A2 и A3, которые представлены на рисунках 9 и 10 соответственно.

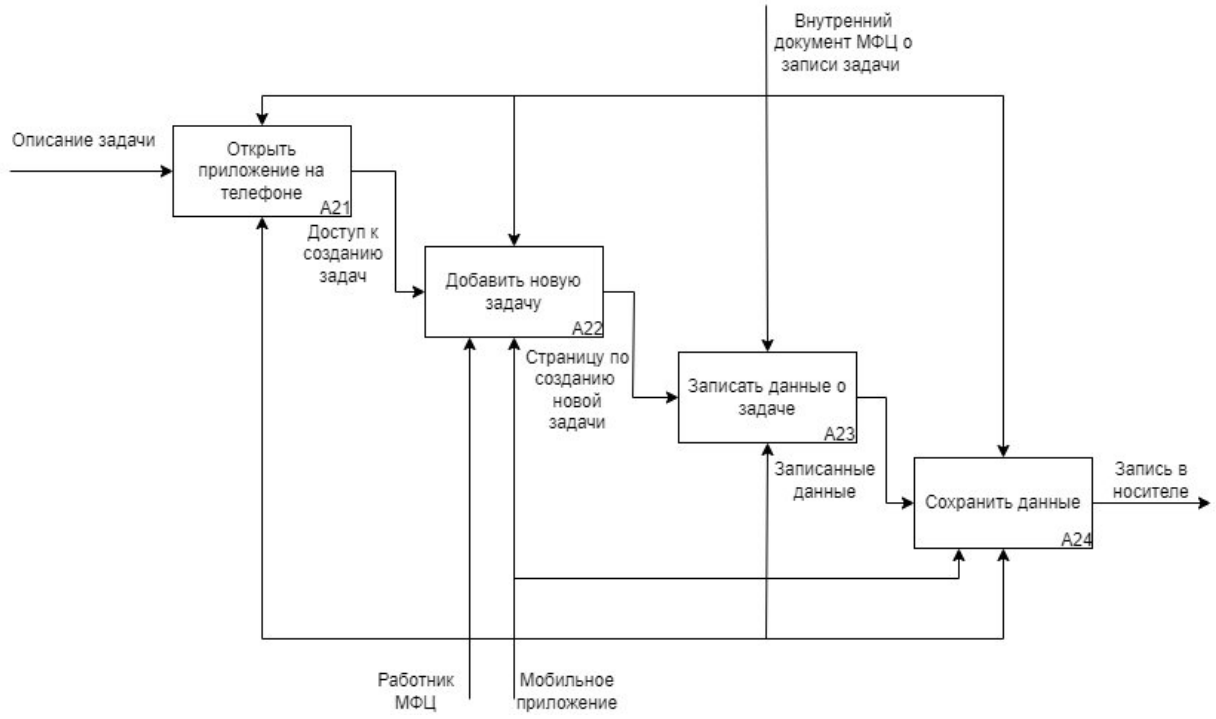

Рисунок 9 - Декомпозиция блока A2

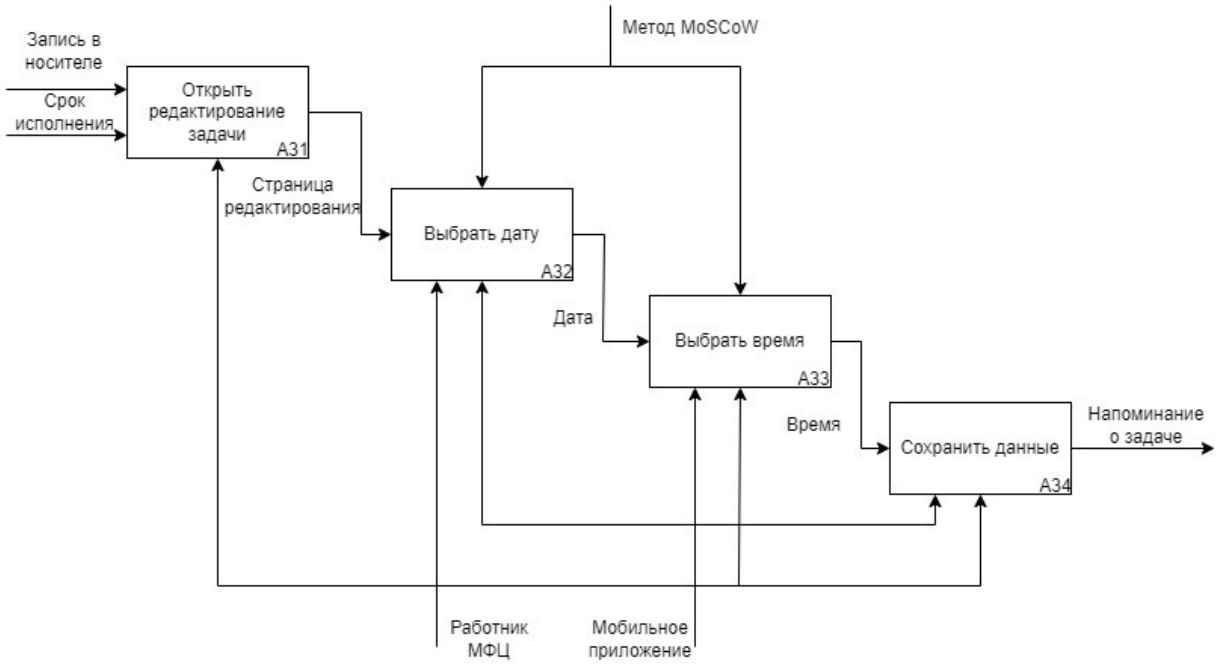

Рисунок 10 - Декомпозиция блока A3

Благодаря DFD диаграмме, покажем визуальное взаимодействие данных в процессе на рисунке 11.

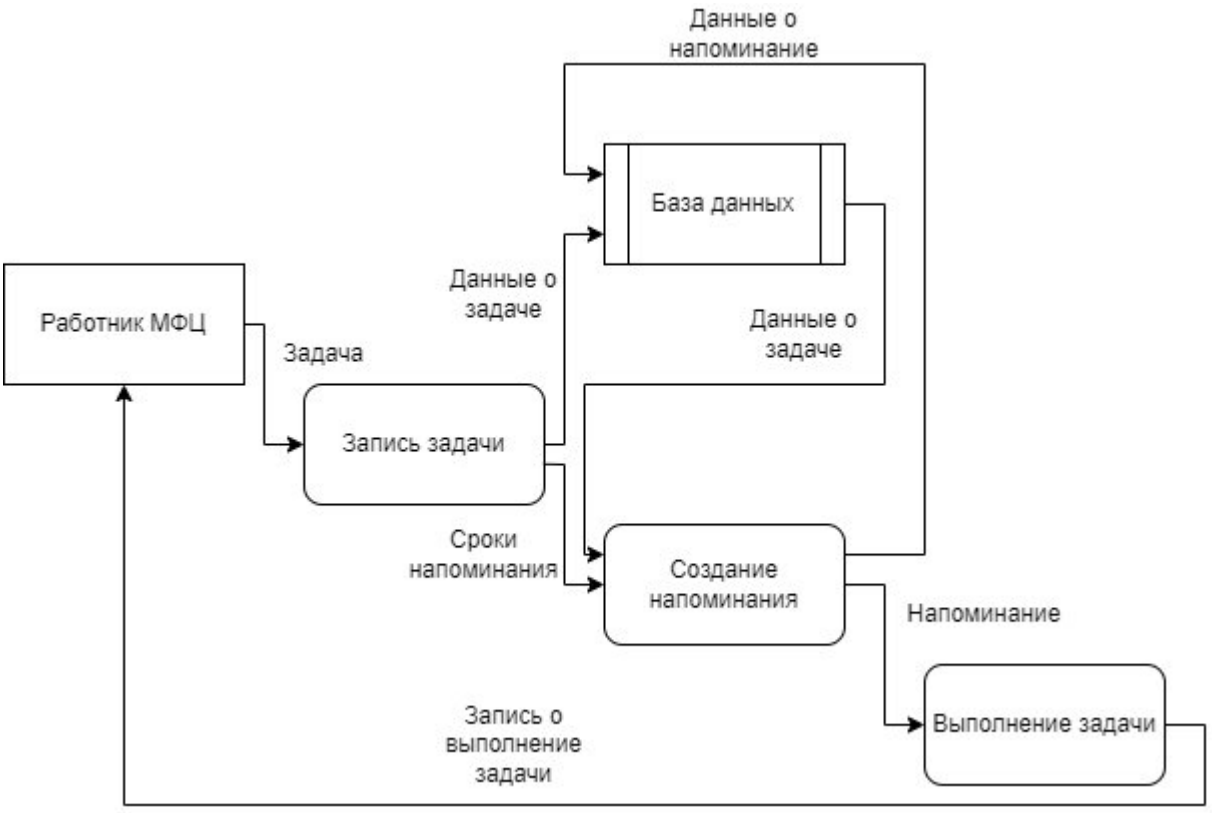

Рисунок 11 - DFD-диаграмма процесса планирования задач

Также при создании приложения важно ответить на вопрос «кто будет пользовать этим приложением?». Для этого используют диаграмму вариантов использования, которая представлена на рисунке 12. Данная диаграмма описывает функционал разрабатываемой программной системы доступный пользователю.

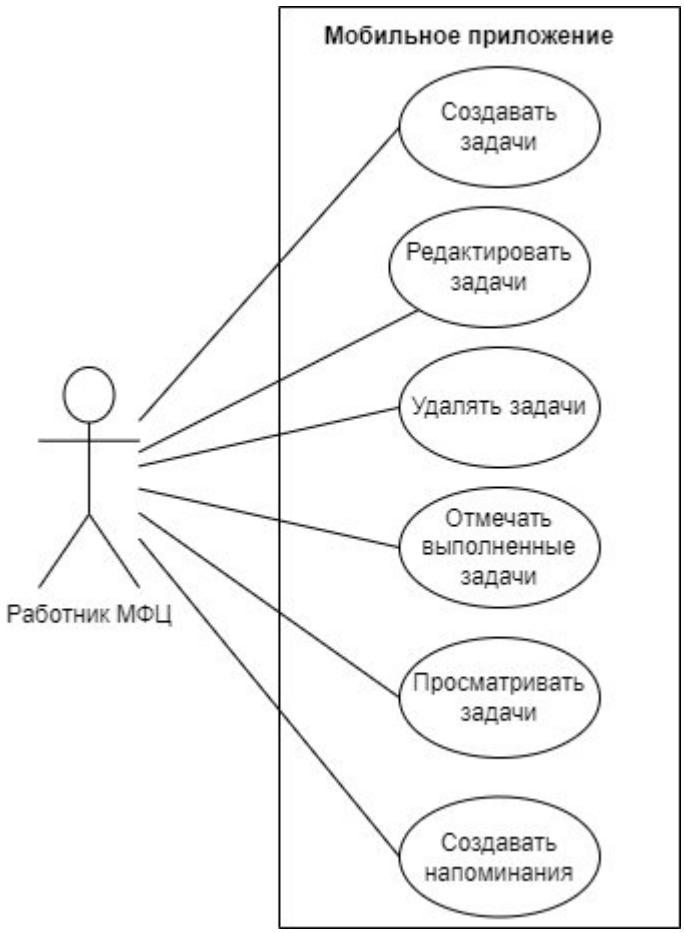

Рисунок 12 - Диаграмма вариантов использования

Целью работы является разработка мобильного приложения «Планировщик задач» для оптимизации процесса планирования задач, в котором работник МФЦ сможет отслеживать и управлять задачами. Приложение должно отвечать всем требованиям, выдвинутым в разделе 1.4 по системе классификации требований FURPS.

Выводы по первой главе

По итогу в этой главе, была дана краткая характеристика предприятия МАУ «МФЦ», был произведен анализ бизнес-процессов предприятия, разработаны требования к мобильному приложению, произведен обзор и анализ рынка мобильных приложений для планирования задач.

Глава 2 Проектирование мобильного приложения «Планировщик задач»

2.1 Выбор платформы реализации мобильного приложения

По статистике использования мобильных ОС в России по данным Яндекс.Метрики, на данный момент мобильных устройств с операционной системой Android в России 2 793 032 854, что составляет 77,59 % от общего количества. В свою очередь мобильных устройств с операционной системой iOS в России 805 891 960, что составляет 22,39 % от общего количества. А мобильных устройств с операционной системой Windows в России 333 005, что составляет меньше 0,01 % от общего количества. Также на сайте Яндекс.Радар приведены данные об остальных операционных системах, количество устройств с которыми составляет 585 362, то есть 0,02% от общего количества [8]. На рисунке 13 показан график соотношения мобильных устройств с разными операционными системами с Марта 2022 года до Марта 2023 года.

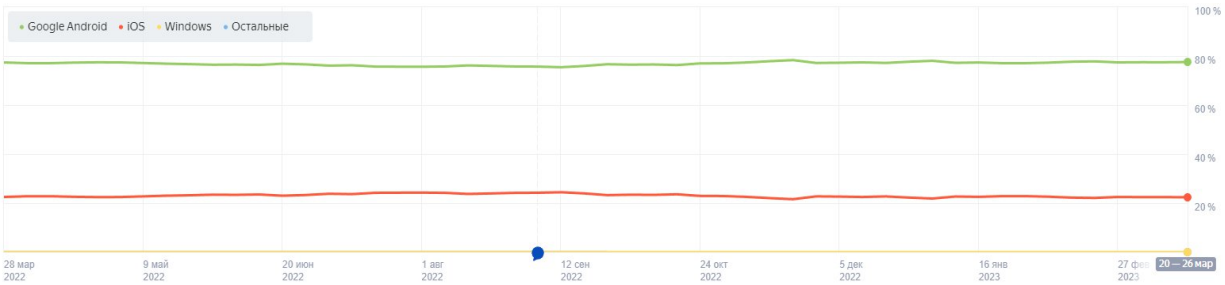

Рисунок 13 – График соотношения мобильных устройств с разными операционными системами за 2022 год и 2023 год

Также сайт позволяет просмотреть график с Апреля 2015 года до Марта 2023 года и сравнить как менялось соотношение мобильных устройств с

разными операционными системами, данный график можно увидеть на рисунке 14.

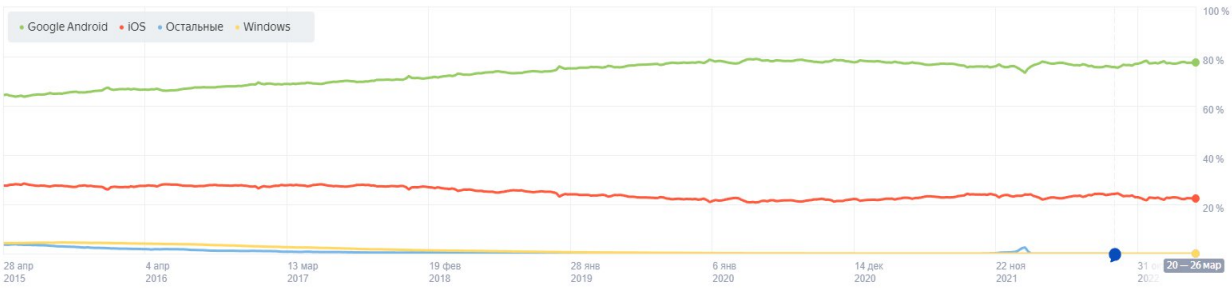

Рисунок 14 – График соотношения мобильных устройств с разными операционными системами за 2015 год и 2023 год

На рисунках 13 и 14 можно видеть, что устройств с операционной системой iOS где-то в 2-3 раза меньше, чем устройств с Android. Опираясь на данные с сайта Яндекс.Метрики, можно сказать, что самой распространённой операционной системой для мобильных устройств является Android. Таким образом, платформой реализации была выбрана операционная система Android, поскольку она имеет наибольшее распространение на мобильных устройствах пользователей в России.

2.2 Выбор средств разработки

Средства разработки программного обеспечения – совокупность приемов, методов, методик, а также набор инструментальных программ, используемых для создания программного кода.

Android Studio – официальная интегрированная среда разработки для операционной системы Android, созданная на базе программного обеспечения JetBrains IntelliJ IDEA и разработанная специально для разработки под Android, поддерживает языки программирования Java и Kotlin [14].

Преимущества Android Studio:

- Среда разработки поддерживает работу с несколькими языками, например C/C++, Java, Kotlin;
- Тестирование корректности работы приложений происходит непосредственно в эмуляторе;
- Большая библиотека с готовыми шаблонами и компонентами для разработки ПО;
- Поддержка системы контроля версий;
- Скорость сборки приложения.

Так как среда разработки поддерживает несколько языков, то стоит определиться на каком будет написано мобильное приложение. Рассмотрим оба языка и сравним их преимущества и недостатки. У языка Java есть ряд преимуществ, таких как большое количество библиотек, решений и готовых модулей, при этом стоит отметить, что в коде на Java используются большие и длинные конструкции, это не всегда будет удобно [9]. Язык Kotlin является более новым и код на нем получается меньшего объёма, так как он не имеет лишних функций, избыточных модулей [11]. Но при этом у языка меньше библиотек и фреймворков, чем у Java. В результате сравнения более подходящим языков является Java, потому что у Kotlin есть серьезный недостаток в виде малого количества библиотек.

Для создания базы данных можно использовать библиотеку Room. Данная библиотека представляет собой обертку для работы с базой данных SQLite, которая является системой управления реляционными базами данных [20]. Компактная встраиваемая СУБД SQLite не поддерживает модель «клиент-сервер», она встроена в конечную программу [12]. Библиотека имеет три основных компонента: Entity, Dao, Database. Аннотацией Entity необходимо помечать объекты, которые будут храниться в базе данных, объект Dao содержит описание методов для работы с базой данных, аннотация Database помечает основной класс по работе с базой данных [19].

Таким образом, средой разработки была выбрана Android Studio, языком программирования был выбран Java, а для реализации базы данных в мобильном приложение была выбрана библиотека Room.

2.3 Проектирование модели данных

Для того, чтобы спроектировать модель данных необходимо построить концептуальную и логическую модель данных. Концептуальная модель базы данных - это наглядная диаграмма, нарисованная в принятых обозначениях и подробно показывающая связь между объектами и их характеристиками [3]. Чтобы начать проектирование определим сущности, такие как задачи, рассылка напоминаний, поставленные задачи. Концептуальная модель базы данных показана на рисунке 15.

Рисунок 15 - Концептуальная модель данных

Как можно видеть каждая сущность имеет свои атрибуты, так задачи включает атрибуты ID задачи, название, описание, а сущность рассылка напоминаний содержит дату, время и ID задачи. Ещё одна сущность поставленные задачи имеет атрибуты дата создания, статус выполнения, ID задачи. На основании концептуальной модель данных строиться логическая модель данных, которая представлена на рисунке 16. При создании будет использоваться методология IDEF1X.

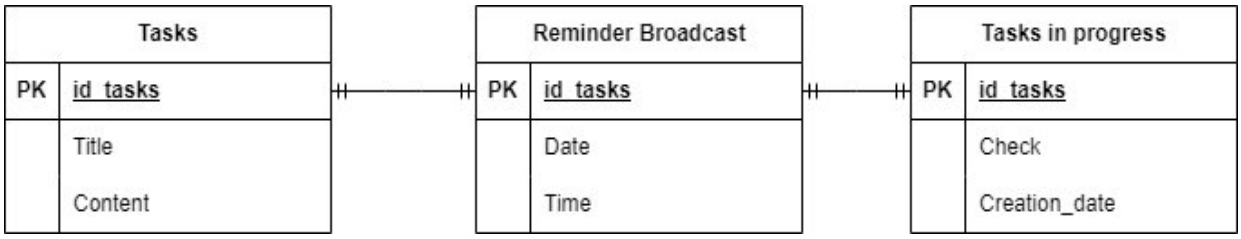

Рисунок 16 - Логическая модель данных

Логическая модель данных представляет структуру базы данных с учетом принимаемой модели данных [6]. Посмотрев, на рисунок логической модели данных, становится понятно, что сущность tasks содержит такие атрибуты, как id_tasks, который является первичным ключом, title, content. В тоже время сущность reminder broadcast имеет атрибуты id_tasks, выполняющего роль первичного ключа, date, time. А сущность tasks in progress включает в себя id_tasks, который необходим в качестве первичного ключа, check, creation_date. Логическая модель данных завершает проектирование модели базы данных, из этого можно сделать вывод о том, что были описаны все необходимые объекты для реализации базы данных мобильного приложения «Планировщик задач».

2.4 Архитектура программного продукта

Мобильное приложение «Планировщик задач» будет реализовано в архитектуре Architecture Components. На Google I/O 2017 года команда Android анонсировала Architecture Components и рекомендации по созданию приложений для Android. Архитектура включает в себя новые коллекции библиотек, которые помогут разрабатывать надежные, тестируемые и поддерживаемые приложения. Этот набор библиотек поможет решить общие проблемы изменения конфигурации, утечки памяти [15]. На рисунке 17 представлена схема Architecture Components [16]. Компоненты архитектуры можно использовать как вместе, так и по отдельности.

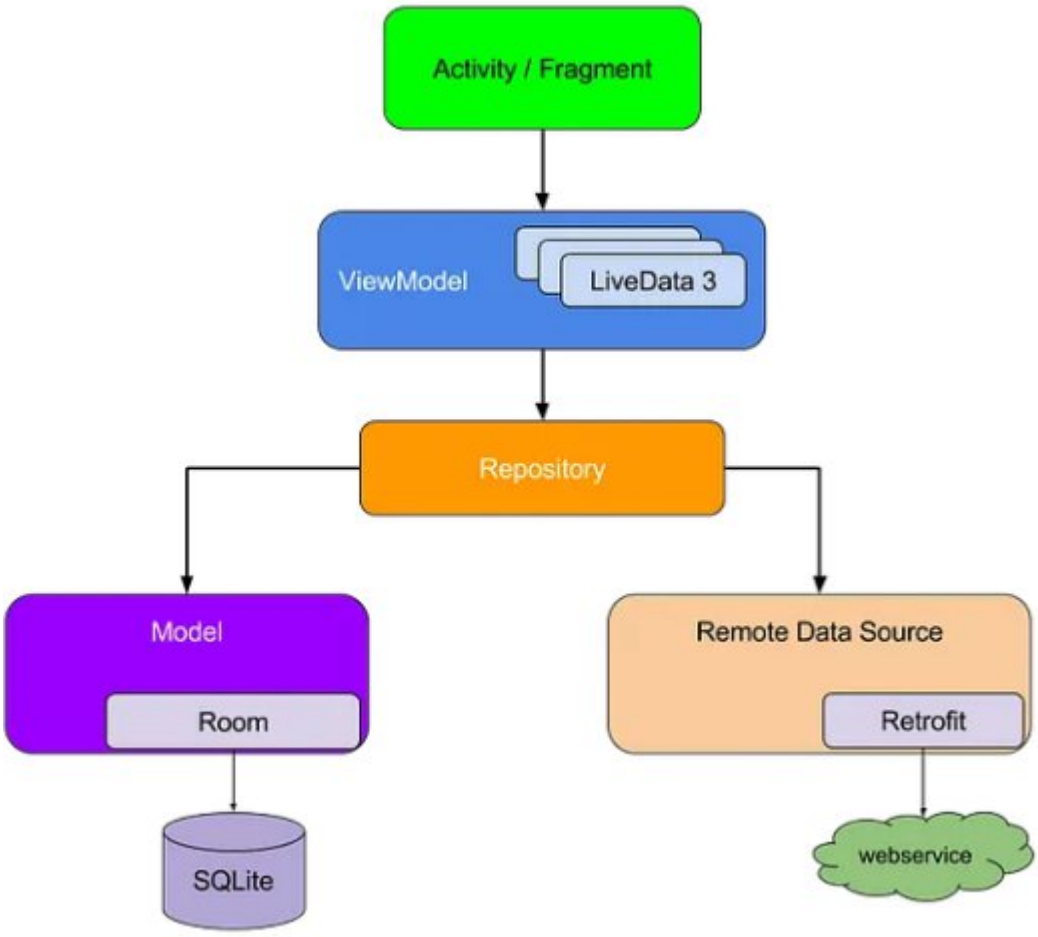

Рисунок 17 - Схема Architecture Components

После выбора архитектуры мобильного приложения можно показать классы и их взаимосвязи с помощью диаграммы классов, которая дает упрощенное представление о сложной системе. На рисунке 18 показана диаграмма классов мобильного приложения «Планировщик задач».

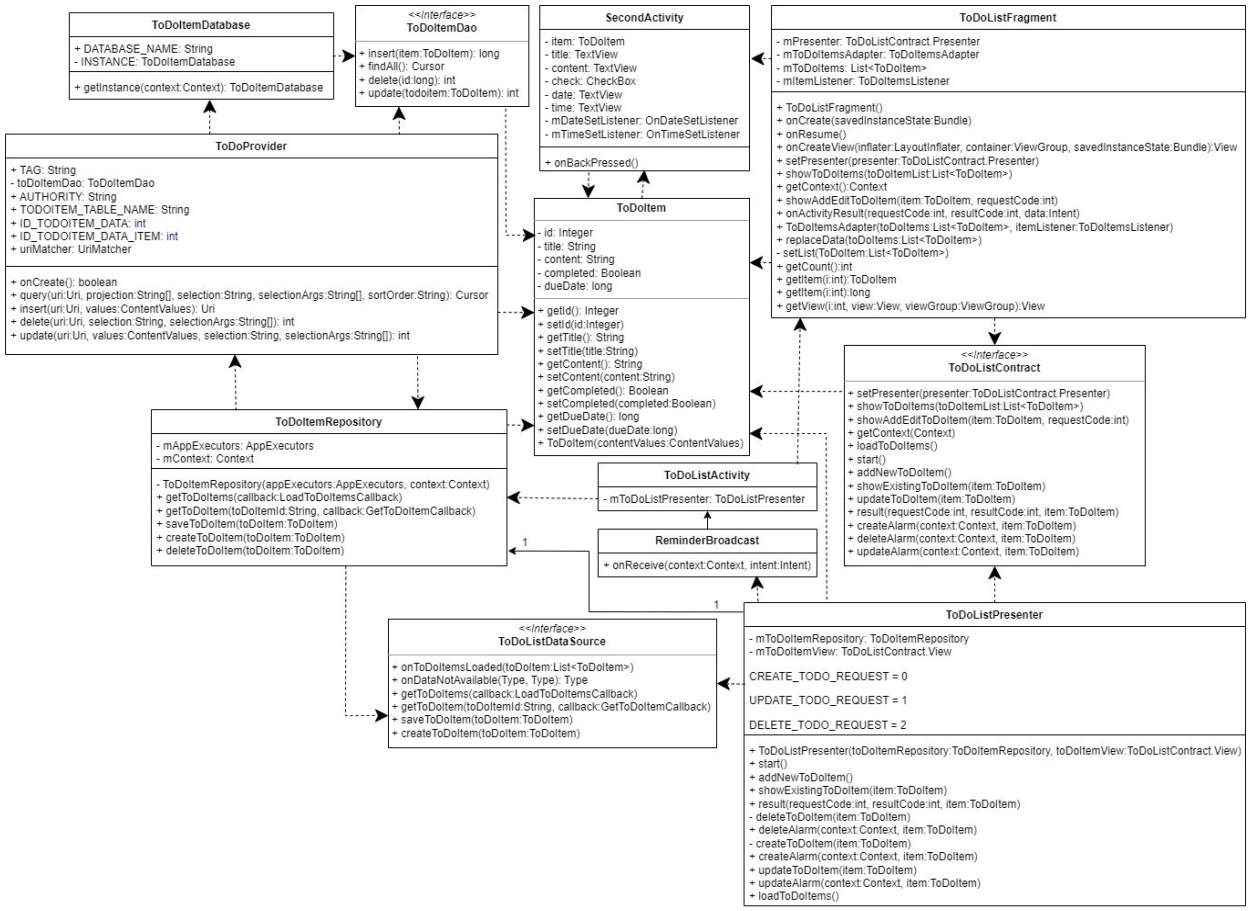

Рисунок 18 – Диаграмма классов мобильного приложения «Планировщик задач»

С помощью диаграммы развертывания, показано физическое развертывание артефактов на узлах системы, диаграмма продемонстрирована на рисунке 19. Узлы – это некоторый тип вычислительного устройства или среда выполнения [7], в данном случае в качестве узлов выступают мобильный телефон и мобильного приложения «Планировщик задач». Артефакт – это

конкретные элементы, которые вызваны процессом разработки, в данном случае артефактом является Room Database.

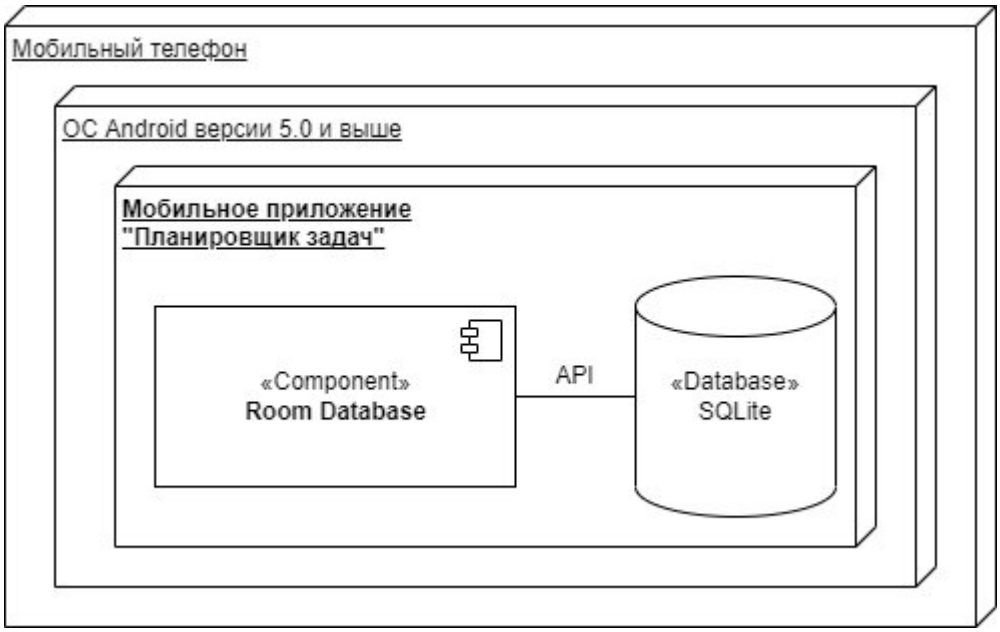

Рисунок 19 - Диаграмма развертывания мобильного приложения «Планировщик задач»

Диаграмма развертывания заканчивает проектирование мобильного приложения «Планировщик задач», теперь можно перейти к реализации и последующему тестированию приложения.

Выводы по второй главе

В результате в этой главе, было описано проектирование мобильного приложения «Планировщик задач», включающее выбор платформы и средств разработки, проектирование модели данных и определение архитектуры программного продукта. Эти результаты являются важным шагом в создании функционального и эффективного приложения, соответствующего требованиям и потребностям пользователей.

Глава 3 Реализация мобильного приложения «Планировщик задач»

3.1 Краткое описание разработанного решения

В результате проектирования и реализации мобильного приложения «Планировщик задач» было разработано девять классов и три интерфейса. Также с классами связаны файлы xml, например класс ToDoListFragment с fragment_to_do_list.xml. Класс хранятся в двух папках, в папке data находятся такие классы, как:

- ToDoItem - реализует serializable для передачи между интентами;
- ToDoItemDao – интерфейс, с методами записи в базу данных, выбора записи из базы данных, удаления записи по id, обновление записи;
- ToDoItemDatabase – создание базы данных;
- ToDoItemRepository – реализует интерфейс ToDoDataSource;
- ToDoListDataSource – интерфейс, который реализует функции обратного вызова в зависимости от результата функций в интерфейсах;
- ToDoProvider – реализует «Поставщика содержимого» (Content Provider), который реализует интерфейс ToDoItemDao.

А в папке todolistactivity находятся такие классы, как:

- ReminderBroadcast – реализует создание напоминаний;
- SecondActivity – реализует создание новой задачи или обновление уже созданной задачи, а также за отображение информации о задачи;
- ToDoListActivity – класс, с которого начинается работа приложения;
- ToDoListContract – интерфейс, который содержит в себе два интерфейса View и Presenter;
- ToDoListFragment – реализует интерфейс View из ToDoListContract;

– ToDoListPresenter – реализует интерфейс Presenter из ToDoListContract.

Для лучшего понимания на рисунке 20 показана структура классов мобильного приложения «Планировщик задач».

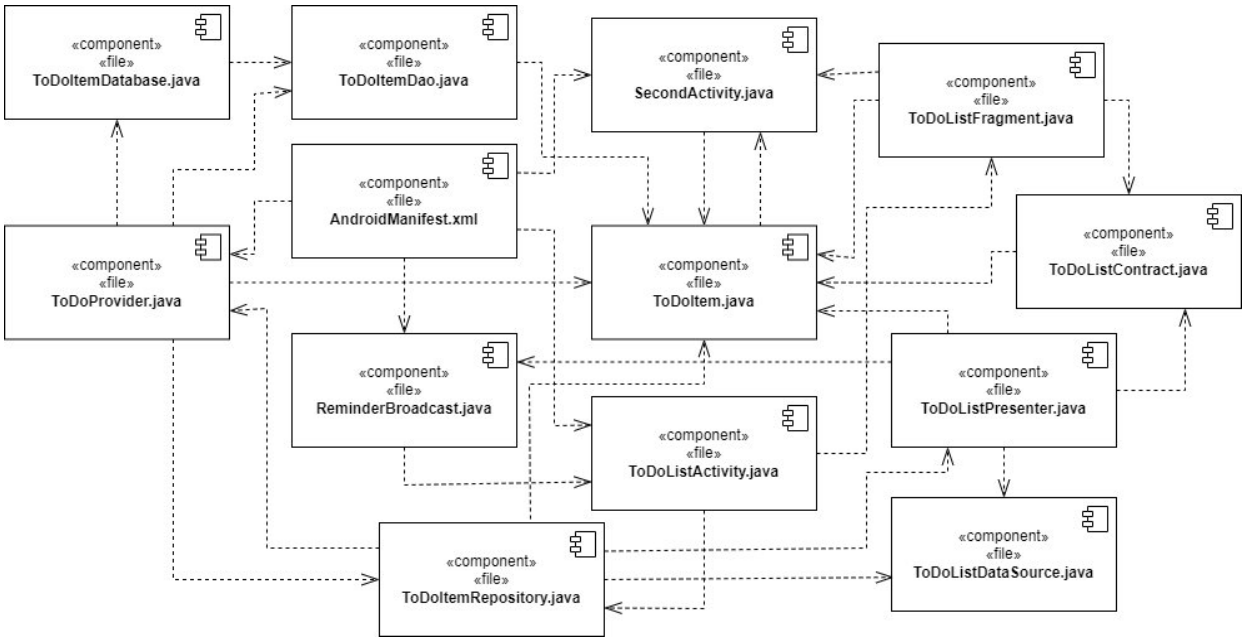

Рисунок 20 – Структура классов мобильного приложения «Планировщик задач»

Таким образом было разработано мобильное приложение «Планировщик задач».

3.2 Реализация создания задачи

Процесс создания задачи реализован в классе SecondActivity, также этот класс отвечает за обновление информации уже созданной задачи и отображении информации. Данный класс связан с файлом activity_add_edit_to_do_item.xml и всеми его элементами, это показано на рисунке 21.

```
setContentView(R.layout.activity_add_edit_to_do_item);
title = findViewById(R.id.etItemTitle);
content = findViewById(R.id.etItemContent);
check = findViewById(R.id.checkBox);
date = findViewById(R.id.textView3);
time = findViewById(R.id.textView4);
Button save = findViewById(R.id.btnSaveToDoItem);
ImageButton delete = findViewById(R.id.imageButton);
```

Рисунок 21 – Связь между файлами SecondActivity.java и activity_add_edit_to_do_item.xml

При создании новой задачи имеется возможность установить дату и время, когда нужно прислать напоминание. Эта функция реализована с помощью слушателя – обработчик события, например он реагирует, когда нажимают на кнопку и выполняет определенные действия [4]. Слушатели реализованы для того, чтобы ввести дату (рисунок 22) и время (рисунок 23) напоминания.

```
date.setOnClickListener((v) → {
        Calendar cal = Calendar.getInstance();
        int day = cal.get(Calendar.DAY_OF_MONTH);
        int month = cal.get(Calendar.MONTH);
        int year = cal.get(Calendar.YEAR);
        DatePickerDialog dialog = new DatePickerDialog(
                context: SecondActivity.this,
                android.R.style.Theme_Holo_Light_DarkActionBar,
                mDateSetListener,
                year, month, day);
        dialog.getWindow().setBackgroundDrawable(new ColorDrawable(Color.TRANSPARENT));
        dialog.show();
});
mDateSetListener = (OnDateSetListener) (datePicker, year, month, day) → {
        calendar.set(Calendar.YEAR, year);
        calendar.set(Calendar.MONTH, month);
        calendar.set(Calendar.DAY_OF_MONTH, day);
        String picked_date = (month+1) + "/" + day + "/" + year;
        Log.d( tag: "SecondActivity",picked_date);
        date.setText(picked_date);
};
```

Рисунок 22 – Реализация слушателя для ввода даты напоминания

```
time.setOnClickListener((v) → {
        Calendar cal = Calendar.getInstance();
        int hour = cal.get(Calendar.HOUR_OF_DAY);
        int minute = cal.get(Calendar.MINUTE);
        TimePickerDialog timePickerDialog = new TimePickerDialog(
                context: SecondActivity.this,
                mTimeSetListener,
                hour, minute,
                android.text.format.DateFormat.is24HourFormat( context: SecondActivity.this));
        timePickerDialog.show();
});
mTimeSetListener = (OnTimeSetListener) (timePicker, hour, minute) → {
        time.setText(hour + ":" + minute);
        calendar.set(Calendar.HOUR_OF_DAY, hour);
        calendar.set(Calendar.MINUTE, minute);
};
```

Рисунок 23 - Реализация слушателя для ввода времени напоминания

Помимо этого, слушатели реализуют функцию удаления и функцию сохранения задачи. При удалении создаётся предупреждение, в котором у пользователя спрашивают, точно ли он хочет удалить задачу. Если пользователь подтверждает свои намерения, то задача удаляется, в обратном случае ничего не происходит. При сохранении данных о задаче записываем название и описание задачи в переменные, после этого получаем дату и время для напоминания, затем создаем новое намерение вернутся к главной странице приложения. Реализации слушателей для удаления и сохранения задачи, показаны на рисунках 24 и 25 соответственно.

```
delete.setOnClickListener((view) → {
        AlertDialog.Builder builder = new AlertDialog.Builder( context: SecondActivity.this);
        builder.setCancelable(true);
        builder.setTitle("Удаление задачи");
        builder.setMessage("Вы уверены, что хотите удалить эту задачу?");
        builder.setPositiveButton( text: "Подтвердить",
                (dialog, which) → {
                        Intent returnIntent = new Intent();
                        returnIntent.putExtra( name: "ToDoItem", item);
                        returnIntent.putExtra( name: "DELETE",  value: true);
                        setResult(Activity.RESULT_OK, returnIntent);
                        finish();
                });
        builder.setNegativeButton(android.R.string.cancel, new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
            }
        });
        AlertDialog dialog = builder.create();
        dialog.show();
});
```

Рисунок 24 - Реализация слушателя для удаления задачи

```
    save.setOnClickListener((v) -> {
            item.setTitle("" + title.getText());
            item.setContent("" + content.getText());
            if(check.isChecked()){item.setCompleted(check.isChecked());}
            else{item.setCompleted(false);}
            Log.d( tag: "SecondActivity", String.valueOf(calendar.getTime()));
            item.setDueDate(calendar.getTimeInMillis());
            Intent returnIntent = new Intent();
            returnIntent.putExtra( name: "ToDoItem", item);
            setResult(Activity.RESULT_OK, returnIntent);
            finish();
    });
}
```

Рисунок 25 - Реализация слушателя для сохранения задачи

С помощью интерфейса ToDoItemDao и класса ToDoProvider (Приложение А) происходит запись данных о задачи в базу данных. База данных создается классом ToDoItemDatabase с помощью RoomDatabase, реализация создания базы данных показана на рисунке 26.

```
@Database(entities = {ToDoItem.class}, version = 1, exportSchema = false)
public abstract class ToDoItemDatabase extends RoomDatabase {
    1 usage
    public static final String DATABASE_NAME = "todo_db";
    3 usages
    private static ToDoItemDatabase INSTANCE;
    1 usage
    public static ToDoItemDatabase getInstance(Context context){
        if(INSTANCE == null){
        INSTANCE = Room.databaseBuilder(context,ToDoItemDatabase.class,DATABASE_NAME).build();
        }
        return INSTANCE;
    }
    1 usage  1 implementation
    public abstract ToDoItemDao getToDoItemDao();
}
```

Рисунок 26 - Реализация создания базы данных в классе ToDoItemDatabase

Покажем физическую модель базы данных, целью которой является преобразование логической схемы с учетом синтаксиса, семантики и возможностей выбранной целевой СУБД [1], на рисунке 27.

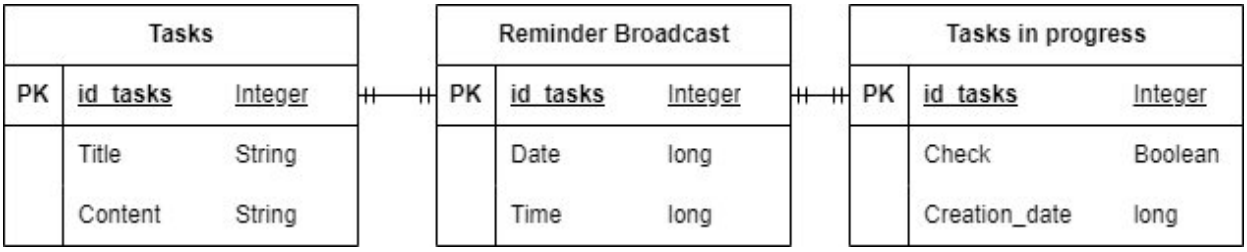

Рисунок 27 - Физическая модель данных

Атрибуты Title, Content являются текстовыми значениям, атрибуты Date, Time, Creation_date имеет 64-битный тип со знаком. Первичные ключи имеют целочисленный тип данных, а атрибут Check имеет логический тип данных. В результате получилось разработать физическую модель, основанную на логической модели данных мобильного приложения «Планировщик задач».

3.3 Реализация оповещения пользователя

Напоминания для пользователей реализованы классом ReminderBroadcast, который расширяет базовый класс BroadcastReceiver. BroadcastReceiver базовый класс, в котором происходит получение и обработка сообщений, посылаемых приложением с помощью вызова метода sendBroadcast. Зарегистрируем класс ReminderBroadcast в файле AndroidManifest в секции application создадим секцию receiver и укажем класс, также поставим атрибут android:exported=«true», чтобы пользователь смог принимать широковещательные сообщения вне области приложения. Помимо

этого, укажем фильтр намерений в виде строки, чтобы определить, какие сообщения приёмник должен прослушивать. Регистрация файла в AndroidManifest показана на рисунке 28.

```
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    <receiver android:name=".todolistactivity.ReminderBroadcast"></receiver>
</application>
```

Рисунок 28 - Статическая регистрации ReminderBroadcast

После этого создадим строителя уведомлений с деталями задачи (рисунок 29), он также понадобится для создания TaskStackBuilder, чтобы получать несколько уведомлений, а не одно. Реализация TaskStackBuilder представлена на рисунке 30.

```
NotificationCompat.Builder builder = new NotificationCompat.Builder(context, channelId: "notifyUser")
        .setSmallIcon(R.drawable.small_icon)
        .setContentTitle(title)
        .setContentText(content)
        .setChannelId("notifyUser");
Intent resultIntent = new Intent(context,ToDoListActivity.class);
```

Рисунок 29 – Создание строителя уведомлений

```
TaskStackBuilder stackBuilder = TaskStackBuilder.create(context);
stackBuilder.addParentStack(ToDoListActivity.class);
stackBuilder.addNextIntent(resultIntent);
PendingIntent resultPendingIntent = stackBuilder.getPendingIntent(id,PendingIntent.FLAG_UPDATE_CURRENT);
builder.setContentIntent(resultPendingIntent);
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O)
{
    String channelId = "notifyUser";
    NotificationChannel channel = new NotificationChannel(
            channelId,
            name: "ReminderBroadcast",
            NotificationManager.IMPORTANCE_HIGH);
    notificationManager.createNotificationChannel(channel);
    builder.setChannelId(channelId);
}
notificationManager.notify(id, builder.build());
```

Рисунок 30 - Реализация TaskStackBuilder

Таким образом был создан класс ReminderBroadcast, который используется для создания уведомлений.

3.4. Тестирование мобильного приложения «Планировщик задач»

Тестирование является важной частью процесса разработки и позволяет обнаружить и исправить возможные ошибки, гарантируя качество работы приложения. Цель проведения тестирования мобильного приложения заключается в обнаружении ошибок в его работе.

Функциональное тестирование - это проверка программного обеспечения на правильное выполнение разработанных функций. Оно направлено на проверку выполнения задачи приложением в различных условиях, важных для пользователей.

Для функционального тестирования проведем четыре теста. Для проведения тестов необходимо определить следующие критерии: цель, ожидаемый результат, входные данные, процедура тестирования, полученный

результат. В таблице 3 представлены результаты тестирования некоторых основных функций мобильного приложения.

Таблица 3 – Тестирование основных функций приложения

№	Цель	Ожидаемый результат	Входные данные	Процедура тестирования	Полученный результат
1	Проверка сохранения данных о задачи в базу данных	Запись о задачи на главном экране приложения	Пользователь находится в окне создания задачи	Пользователь вводит название задачи и её описание в соответствующие поля и нажимает кнопку «Сохранить».	Соответствует ожидаемому результату
2	Проверка функции напоминания о задаче	На телефон придет уведомление с названием задачи и её описанием в заранее выбранную дату и время	Пользователь находится в окне создания задачи	Пользователь вводит дату и время, когда должно быть получено напоминание в соответствующие поля и нажимает кнопку «Сохранить».	Напоминание было отправлено на телефон
3	Проверка функций о выполнении задачи	После выбора функции отметки о выполнении задачи на главном экране приложения задача будет отображаться зеленым цветом	Пользователь находится в окне создания задачи	Пользователь нажимает кнопку «Сделано?», после чего нажимает на кнопку «Сохранить» и возвращается на главный экран приложения, где название и описание задачи становиться написано зеленым цветом	Соответствует ожидаемому результату
	Проверка функций удаления задачи	После выбора функции удаления задачи на главном экране приложения задача будет удалена	Пользователь находится в окне создания задачи	При нажатии кнопки «Удалить?» пользователь видеть появившееся окно, в котором уточняется точно ли пользователь	Соответствует ожидаемому результату

Продолжение таблицы 3

№	Цель	Ожидаемый результат	Входные данные	Процедура тестирования	Полученный результат
				хочет удалить задачу, в случае подтверждения задача и все данные о ней удаляются, в случае отмены действия ничего не происходит	

В первой строке проверяется сохраняются ли данные и задачи в базе данных. Для этого создаётся новая задача, заполняются данные и проверяется отображается ли задача в списке задач на главном экране. Запись о задаче была добавлена на главный экран приложения, значит сохранение данных прошло успешно. Из этого можно сделать вывод данные о задачи сохраняются в базу данных.

В второй строке проверяется функция напоминания. Чтобы это проверить в задаче следует выбрать время и дату напоминания. Как можно видеть функция напоминания сработала и уведомление было отправлено в назначенный срок. Из этого следует, что функция напоминания работает исправно.

В третьей строке проверяется функция отметки о выполнении задачи, для проверки этой функций в уже созданной задаче необходимо нажать кнопку «Сделано?». Полученный результат совпадает с ожидаемым, и задача становится зеленого цвета. Таким образом функция отметки о выполнении задачи работают исправно.

В четвертой строке проверяется функция удаления, чтобы проверить работу этой функций в уже созданной задаче нажмем кнопку «Удалить?». Как и ожидалась появилось всплывающее окно, при подтверждении в котором

происходит удаление задачи. Можно сделать вывод, что функция удаления задачи работают исправно.

Тестирование основных функций мобильного приложения для планирования показало успешное выполнение задач. Это говорит о правильном функционировании приложения и его соответствии ожиданиям пользователей.

Выводы по третьей главе

В результате третья глава представляет реализацию мобильного приложения «Планировщик задач», включающую разработку основного функционала, в том числе создания задачи и оповещения пользователя, а также проведение тестирования для проверки работоспособности и качества приложения. Эти результаты подтверждают успешную реализацию поставленных задач и функциональности приложения.

Заключение

В процессе выполнения бакалаврской работы было создано мобильное приложение «Планировщик задач» на операционной системе Android. Целью работы было создание удобного и эффективного инструмента для работников МФЦ, позволяющего планировать и управлять своими задачами.

В ходе выполнения работы были достигнуты следующие результаты:

- Проведен анализ бизнес-процессов предприятия, в результате которого были выявлены требования к функциональности и особенностям мобильного приложения для планирования задач;
- Произведено проектирование мобильного приложения, включающее выбор средств разработки, разработку модели данных и определение архитектуры программного продукта;
- Была выполнена реализация мобильного приложения, включающая создание основных функций, таких как создание задач, оповещение пользователей;
- Проведено тестирование мобильного приложения, в результате которого была проверена корректность работы основных функций и убедились в соответствии результатов с ожидаемыми.

Результатом работы стало полноценное мобильное приложение «Планировщик задач», которое предоставляет пользователям возможность эффективно организовывать свое время, планировать и контролировать выполнение задач. Приложение предоставляет удобный и интуитивно понятный интерфейс, позволяющий пользователям легко освоить все его функции.

В ходе работы были решены поставленные задачи, достигнута поставленная цель, и разработанное приложение полностью соответствует требованиям и ожиданиям пользователей. Однако, в дальнейшем развитии мобильного приложения «Планировщик задач» есть потенциал для внесения

дополнительных улучшений и функциональных возможностей. Некоторые из них могут включать:

- Персонализация и настройки. Разработать функционал, позволяющий пользователям настраивать интерфейс и управлять предпочтениями отображения задач и расписания. Включение возможности выбора темы, настройки напоминаний и уведомлений может значительно улучшить пользовательский опыт;
- Аналитика и отчетность. Добавить возможность анализировать статистику выполнения задач, уровень продуктивности и эффективности использования времени. Создание отчетов и графиков поможет пользователям лучше понять свои рабочие привычки и сделать наиболее информированные решения.

Однако, при разработке мобильных приложений необходимо учитывать индивидуальные потребности и предпочтения пользователей. Дальнейшая работа должна быть направлена на постоянное обновление и улучшение приложения, внедрение новых функций и возможностей, а также обратную связь с пользователями для учета их потребностей.

Список используемой литературы

1. Аканов А. Д., Сагындыков К. М. ФИЗИЧЕСКАЯ МОДЕЛЬ ДАННЫХ //Под общей редакцией Курманбаевой Ш. А. Выпускающий редактор Матаева МХ Редакционная коллегия. – 2018. – С. 101.
2. Андиева Е. Ю., Сидоренко В. С. АНАЛИЗ КРИТЕРИЕВ ВЫБОРА CASE–СРЕДСТВ //Редакционная коллегия. – 2015. – С. 164.
3. Белоусов А. В., Толкачева О. С., Баева В. Г. Разработка концептуальной модели базы данных информационно-справочных СППР //Системный анализ в проектировании и управлении: сборник научных трудов XXI Международной научнопрактической конференции. – 2017. – С. 298.
4. Болхудере Е. И. СРАВНЕНИЕ СИСТЕМ РАЗРАБОТКИ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ //XXIII Всероссийская студенческая научно-практическая конференция Нижневартовского государственного университета. – 2021. – С. 91-97.
5. Гузилов А. В., Мышенков К. С., Симонов М. Ф. АНАЛИЗ КАЧЕСТВА CASE-СРЕДСТВ ДЛЯ ФУНКЦИОНАЛЬНОГО МОДЕЛИРОВАНИЯ СИСТЕМ //Информационно-аналитические и интеллектуальные системы для производства и социальной сферы. – 2021. – С. 18-29.
6. Дорохова А. М., Шацкий В. А., Картечина Н. В. Создание логической и физической модели базы данных //Наука и Образование. – 2020. – Т. 3. – №. 4. – С. 7.
7. Дубаков С. А., Силич В. А. Использование набора диаграмм UML для построения моделей производительности //Известия Томского политехнического университета. Инжиниринг георесурсов. – 2005. – Т. 308. – №. 3. – С. 154-158.
8. Мобильнные ОС в России. [Электронный ресурс] / Режим доступа: URL: https://radar.yandex.ru/mobile (дата обращения 26.03.2023).

9. Программирование на Java / Патрик Нимейер, Дэниэл Леук ; [пер. с англ. М. А. Райтмана]. — Москва: Эксмо, 2014. — 1216 с.

10. Портал МФЦ Самарской области. [Электронный ресурс] / Режим доступа: URL: https://mfc63.samregion.ru/#city (дата обращения 02.02.2023).

11. Солонько М. К. Язык программирования KOTLIN //Вестник науки и образования. – 2020. – №. 7-1 (85). – С. 25-27.

12. Architecture of SQLite. [Электронный ресурс] / Режим доступа: URL: https://www.sqlite.org/arch.html (дата обращения 11.04.2023).

13. Diagrams.net. [Электронный ресурс] / Режим доступа: URL: https://en.wikipedia.org/wiki/Diagrams.net (дата обращения 25.02.2023).

14. Download Android Studio and SDK tools. [Электронный ресурс] / Режим доступа: URL: https://developer.android.com/studio/intro (дата обращения 10.04.2023).

15. Guide to app architecture. [Электронный ресурс] / Режим доступа: URL: https://developer.android.com/topic/architecture (дата обращения 21.04.2023).

16. Introduction to Android Architecture Components. [Электронный ресурс] / Режим доступа: URL: https://medium.com/android-news/introduction-to-android-architecture-components-22b8c84f0b9d (дата обращения 20.04.2023)

17. Janošcová R. Evaluation of software quality //IMEA 2012. – 2012. – С. 24.

18. MoSCoW method. [Электронный ресурс] / Режим доступа: URL: https://en.wikipedia.org/wiki/MoSCoW_method (дата обращения 20.02.2023)

19. Room. [Электронный ресурс] / Режим доступа: URL: https://developer.android.com/jetpack/androidx/releases/room (дата обращения 13.04.2023).

20. SQLite [Электронный ресурс] / Режим доступа: URL: https://en.wikipedia.org/wiki/SQLite (дата обращения 12.04.2023).

Приложение А

Пример описания класса ToDoProvider

```
package edu.csce4623.ahnelson.todomvp3.data;
import android.content.ContentProvider;
import android.content.ContentUris;
import android.content.ContentValues;
import android.content.UriMatcher;
import android.database.Cursor;
import android.net.Uri;
import android.util.Log;

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

public class ToDoProvider extends ContentProvider {

    public static final String TAG = ToDoProvider.class.getName();
    private ToDoItemDao toDoItemDao;

    public static final String AUTHORITY =
"edu.csce4623.ahnelson.todomvp3.data";
    public static final String TODOITEM_TABLE_NAME = "todoitems";

    public static final int ID_TODOITEM_DATA = 1;
    public static final int ID_TODOITEM_DATA_ITEM = 2;

    //URI matcher for switch statements on calls to the provider
    public static final UriMatcher uriMatcher = new
UriMatcher(UriMatcher.NO_MATCH);
```

```
        //Definition of two patterns (table or table + ID)
        static{

uriMatcher.addURI(AUTHORITY,TODOITEM_TABLE_NAME,ID_TODOITEM_DATA);

uriMatcher.addURI(AUTHORITY,TODOITEM_TABLE_NAME+"/*",ID_TODOITEM_DATA_ITEM);
        }

        @Override
        public boolean onCreate() {
            toDoItemDao = ToDoItemDatabase.getInstance(getContext()).getToDoItemDao();
            return false;
        }

        @Nullable
        @Override
        public Cursor query(@NonNull Uri uri, @Nullable String[] projection, @Nullable String selection,
                            @Nullable String[] selectionArgs, @Nullable String sortOrder) {
            Log.d(TAG,"query");
            Cursor cursor;
            switch (uriMatcher.match(uri)){
                case ID_TODOITEM_DATA:
                    cursor = toDoItemDao.findAll();
                    if(getContext() != null){
```

```
                cursor.setNotificationUri(getContext()
                        .getContentResolver(),uri);
                return cursor;
            }
            break;
        default:
            throw new IllegalArgumentException("Unknown URI: " + uri);
    }
    return null;
}

@Nullable
@Override
public String getType(@NonNull Uri uri){
    return null;
}

@Nullable
@Override
public Uri insert(@NonNull Uri uri, @Nullable ContentValues values) {
    Log.d(TAG,"insert");
    switch (uriMatcher.match(uri)){
        case ID_TODOITEM_DATA:
            if(getContext() != null){
                long id =
toDoItemDao.insert(ToDoItem.fromContentValues(values));
                if (id != 0) {
                    getContext().getContentResolver().notifyChange(uri,null);
                    return ContentUris.withAppendedId(uri,id);
```

```
                }
            }
            case ID_TODOITEM_DATA_ITEM:
                throw new IllegalArgumentException("Invalid URI: Insert failed "
+ uri);
            default:
                throw new IllegalArgumentException("Unknown URI: " + uri);
        }
    }

    public ToDoProvider() {
    }

    @Override
    public int delete(Uri uri, String selection, String[] selectionArgs) {
        Log.d(TAG, "delete");
        switch (uriMatcher.match(uri)){
            case ID_TODOITEM_DATA:
            throw new IllegalArgumentException("Invalid uri: cannot delete");
            case ID_TODOITEM_DATA_ITEM:
                if(getContext() != null){
                    //Delete item in the DAO at a given ID
                    int count = toDoItemDao.delete(ContentUris.parseId(uri));
                    getContext().getContentResolver().notifyChange(uri,null);
                    return count;
                }
            default:
                throw new IllegalArgumentException("Unknown URI: " + uri);
        }
```

```
        }

        @Override
        public int update(@NonNull Uri uri, @Nullable ContentValues values,
@Nullable String selection,
                          @Nullable String[] selectionArgs) {

            Log.d(TAG,"update");
            Log.d(TAG,uri.toString());
            switch (uriMatcher.match(uri)){
                case ID_TODOITEM_DATA:
                    if(getContext() != null){
                        //Update DAO from given ToDoItem
                        int                                count                                =
toDoItemDao.update(ToDoItem.fromContentValues(values));
                        if(count != 0){
                            getContext().getContentResolver().notifyChange(uri,null);
                            return count;
                        }
                    }
                    break;
                case ID_TODOITEM_DATA_ITEM:
                throw new IllegalArgumentException("Invalid URI: cannot update");
                default:
                    throw new IllegalArgumentException("Unknwon URI: " + uri);
            }
            return 0;
        }
    }
```