Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Южно-Уральский государственный университет
(национальный исследовательский университет)»
Институт открытого и дистанционного образования
Кафедра Техника, технологии и строительство

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой,
к.т.н., доцент
_____К.М. Виноградов
_____2020 г.

Разработка мобильного приложения для планирования активности людей с умеренными расстройствами памяти

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
ЮУрГУ–09.03.01.2020.886 ПЗ ВКР

Руководитель работы,
к.э.н., доцент
_____А.Г. Калачева
_____2020 г.

Автор работы
студент группы ДО – 525
_____Д.Г. Митюрева
_____2020 г.

Нормоконтролер,
преподаватель
_____О.С. Микерина
_____2020 г.

Челябинск 2020

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Институт открытого и дистанционного образования
Кафедра Техника, технологии и строительство
Направление 09.03.01 «Информатика и вычислительная техника»

УТВЕРЖДАЮ
Заведующий кафедрой
_____ К.М. Виноградов
_____2020 г.

ЗАДАНИЕ

на выпускную квалификационную работу студента
Митюревой Дарьи Глебовны
Группа ДО-525

1 Тема работы: Разработка мобильного приложения для планирования активности людей с умеренными расстройствами памяти

утверждена приказом по университету от «24» апреля 2020 г. №627

2 Срок сдачи студентом законченной работы 10.06.2020

3 Исходные данные к работе

3.1 Требования к мобильному приложению для планирования активности людей с умеренными расстройствами памяти

3.2 Платформа: ОС Android

3.3 Документация ОС Android

3.4 Методические рекомендации WCAG по обеспечению доступности

4 Содержание расчетно-пояснительной записки (перечень подлежащих разработке вопросов)

4.1 Анализ процессов создания и применения мобильных приложений

4.1.1 Принципы разработки и функционирования мобильных приложений

4.1.2 Компоненты и работа приложений в ОС Android

4.1.3 Социальные функции и значение мобильных устройств и приложений

4.1.4 Использование мобильных приложений при когнитивных расстройствах

4.2 Сравнительный анализ отечественных и зарубежных технологий

4.2.1 Обзор существующих мобильных приложений для планирования действий

4.2.2 Обзор технологий создания мобильных приложений

4.3 Реализация мобильного приложения для планирования активности

4.3.1 Архитектура приложения и база данных

4.3.2 Модули работы с данными и службами ОС Android

4.3.3 Модули представления

4.3.4 Разработка и тестирование пользовательского интерфейса

5 Перечень графического материала

5.1 Пояснительная записка на 56 листах формата А4

5.2 Презентация на 14 слайдах

Дата выдачи задания 24.03.2020

Руководитель _____А.Г. Калачева
(подпись)

Задание принял к исполнению _____Д.Г. Митюрева
(подпись)

КАЛЕНДАРНЫЙ ПЛАН

Наименование этапов выпускной квалификационной работе	Срок выполнения этапов работы	Отметка руководителя о выполнении
Введение		
Анализ процессов создания и применения мобильных приложений		
Принципы разработки и функционирования мобильных приложений		
Компоненты и работа приложений в ОС Android		
Социальные функции и значение мобильных устройств и приложений		
Использование мобильных приложений при когнитивных расстройствах		
Сравнительный анализ отечественных и зарубежных технологий		
Обзор существующих мобильных приложений для планирования действий		
Обзор технологий создания мобильных приложений		
Реализация мобильного приложения для планирования активности		
Архитектура приложения и базы данных		
Модули работы с данными и службами ОС Android		
Модули представления		
Разработка и тестирование пользовательского интерфейса		
Заключение		

Зав. кафедрой _____ К.М. Виноградов
(подпись)

Руководитель работы _____ А.Г. Калачева
(подпись)

Студент_____ Д.Г. Митюрева
(подпись)

АННОТАЦИЯ

Митюрева, Д.Г. Разработка мобильного приложения для планирования активности людей с умеренными расстройствами памяти. – Челябинск: ФГАОУ ВО «ЮУрГУ (НИУ)», ИОДО; 2020, 56 с., 23 илл., 1 таблица, библиогр. список – 40 наим., 1 прил.

Работа посвящена разработке мобильного приложения для планирования активности людей с умеренными расстройствами памяти, актуальность которого состоит в интеграции специальных средств обеспечения доступности платформы, путем оптимизации приложения для работы с ними. Рассмотрены основные принципы создания и функционирования мобильных приложений. Изучены международные и отечественные рекомендации по созданию доступных приложений и информации, представленной в электронно-цифровой форме. Выполнен сравнительный анализ существующих отечественных и зарубежных аналогов создаваемого приложения, а также платформ и средств разработки.

В разделе программной реализации представлены практические результаты проведенной работы: описана структура созданного приложения для планирования ежедневной деятельности людей с умеренными расстройствами памяти на базе ОС Android, а также описаны методы, примененные для оптимизации работы программы со встроенными специальными возможностями платформы, обеспечивающими доступность системы и приложений.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ 6
1 АНАЛИЗ ПРОЦЕССОВ СОЗДАНИЯ И ПРИМЕНЕНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ
1.1 Принципы разработки и функционирования мобильных приложений 8
1.2 Компоненты и работа приложений в ОС Android 11
1.3 Социальные функции и значение мобильных устройств и приложений 13
1.4 Использование мобильных приложений при когнитивных расстройствах .. 16
2 СРАВНИТЕЛЬНЫЙ АНАЛИЗ ОТЕЧЕСТВЕННЫХ И ЗАРУБЕЖНЫХ ТЕХНОЛОГИЙ
2.1 Обзор существующих мобильных приложений для планирования действий 20
2.2 Обзор технологий создания мобильных приложений 22
3 РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПЛАНИРОВАНИЯ АКТИВНОСТИ
3.1 Архитектура приложения и базы данных 29
3.2 Модули работы с данными и службами ОС Android 32
3.3 Модули представления 37
3.4 Разработка и тестирование пользовательского интерфейса 42
ЗАКЛЮЧЕНИЕ 50
БИБЛИОГРАФИЧЕСКИЙ СПИСОК 51
ПРИЛОЖЕНИЯ
ПРИЛОЖЕНИЕ А Фрагменты кода приложения 54

ВВЕДЕНИЕ

Актуальность темы. Современные информационные технологии обладают рядом функций, способных предоставить людям с ограниченными возможностями доступ к различным информационным средам и услугам. Примером таких функций может быть использование SMS и других видов текстовых сообщений для общения людьми с нарушениями речи и слуха.

Достаточно часто при некоторых заболеваниях, вызывающих нарушение памяти, люди способны самостоятельно ухаживать за собой, благодаря изменению образа жизни. В этом случае мобильные приложения могут помочь с организацией ежедневной активности таких людей.

Растущий спрос на приложения и устройства, реализующие несколько способов коммуникации с ними, как у людей с какими-либо нарушениями, так и без, стал стимулом, который привел к распространению такого подхода у создателей операционных систем, приложений и устройств. Так, например, голосовое управление устройством, может помочь любому человеку, у которого заняты руки, будь это из-за готовки или вождения автомобиля, настройки контрастности могут быть полезны в случае яркого света или бликов на фоне.

Целью выпускной квалификационной работы является создание Android приложения для планирования ежедневной деятельности людей с умеренными расстройствами памяти.

Задачи выпускной квалификационной работы:

– проанализировать теоретические основы разработки и функционирования мобильных приложений;

– изучить опыт отечественных и зарубежных разработчиков мобильных приложений;

– изучить методические рекомендации международных организаций к созданию доступных для использования интерфейсов и приложений;

– изучить методы обеспечения доступности приложения и их тестирование;

– создать мобильное приложение;

– изучить особенности интеграции специальных возможностей операционной системы в работу приложения.

Объектом выпускной квалификационной работы является процесс планирования активности людей с умеренными расстройствами памяти.

Предметом выпускной квалификационной работы является автоматизация процесса планирования активности людей с помощью Android-приложения.

Практическая значимость выпускной квалификационной работы состоит в предложении нового приложения для планирования деятельности, созданного для специфической аудитории, согласно с требованиями международных консорциумов к оформлению таких приложений, и оптимизированного для работы со встроенными системными специальными возможностями обеспечения доступности.

Структура выпускной квалификационной работы состоит из введения, трех разделов, заключения и библиографического списка. Раздел 1 посвящен теоретическому обоснованию и анализу темы исследования, описываются процессы создания, применения и функционирования мобильных приложений, их социальные функции и возможности, методические рекомендации по обеспечению доступности. Раздел 2 посвящен анализу и обоснованию проектных решений и выбора технологий, на основании которых ведется разработка проекта. Раздел 3 посвящен практическому решению поставленной задачи: планированию архитектуры приложения, его созданию, тестированию и оптимизации интерфейса.

Объем выпускной квалификационной работы составляет 56 страниц машинописного текста и содержит 23 иллюстрации, 1 таблицу, библиографический список из 40 наименований и 1 приложение.

1 АНАЛИЗ ПРОЦЕССОВ СОЗДАНИЯ И ПРИМЕНЕНИЯ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ

1.1 Принципы разработки и функционирования мобильных приложений

Мобильное приложение – это программное обеспечение, предназначенное для работы на мобильных устройствах, разработанное для конкретных платформ и под их функциональные возможности. Такие приложения могут быть как предустановлены на устройстве, так и загружены на него из магазинов приложений или других платформ распространения программного обеспечения.

Особенностью мобильных платформ относительно стационарных является их малый размер, высокая плотность размещения информации (количество пикселей на дюйм) и взаимодействие с пользователем с помощью сенсорного экрана. Кроме этого, большинство мобильных устройств имеют сравнительно медленный процессор и работают от аккумулятора, поэтому оптимизация кода особенно важна.

Существует три основных вида мобильных приложений: веб-приложения, гибридные приложения и нативные приложения. Каждый из этих видов, изображенных на рисунке 1, определяет свой подход к созданию приложения и свой набор используемых при этом инструментов [13].

Рисунок 1 – Виды мобильных приложений

Веб-приложения работают на устройстве через веб-браузер, но могут предоставить пользователю опцию запуска приложения через ярлык-закладку на рабочем столе. Такие приложения создаются при помощи HTML, CSS и JavaScript, кроссплатформенны и почти не занимают места на устройстве.

Несмотря на то, что иногда они сделаны таким образом чтобы внешне имитировать нативные приложения, код таких приложений пишется не под конкретную платформу, а сами приложения имеют доступ к гораздо меньшему количеству функций API и устройства, низкую производительность и требуют подключения к сети.

Нативные приложения – самый распространенный тип приложений. Они созданы для конкретных платформ на конкретных для платформы языках – Swift или Objective C для iOS или Java и Kotlin для Android. Такие приложения устанавливаются непосредственно на устройство пользователя и обычно загружаются на него через конкретный для платформы магазин приложений – Play Store для Google и Apple App Store для iOS – обновление приложений происходит через эти же платформы. Нативные приложения работают быстрее всех остальных типов приложений и имеют доступ ко всем функциям системы и устройства, но при этом не могут быть запущены ни на каких других платформах, кроме тех, для которых они были разработаны.

Гибридные приложения создаются так, чтобы выглядеть и использоваться как нативные приложения. Они разрабатываются при помощи HTML, CSS и JavaScript, но работают внутри нативной оболочки, приложение может совмещать использование WebView и нативных объектов. Гибридные приложения так же, как и нативные можно найти в магазинах приложений и они, благодаря нативной оболочке, имеют доступ к функциям системы, которые были бы не доступны обычному веб-приложению. Тем не менее такие приложения имеют достаточно низкую скорость работы, могут нестабильно работать с другими функциями мобильного устройства.

Архитектура мобильного приложения варьируется от проекта к проекту, но обычно строится по принципу разделения ответственностей и может включать в себя, как показано на рисунке 2, несколько базовых уровней типичных для приложений с многоуровневой архитектурой [34].

Слой представления содержит компоненты пользовательского интерфейса, а также компоненты, обрабатывающие их. Слой бизнес-логики определяет логику работы приложения, его основные функции, зависимости поведения объектов предметной области. Слой доступа к данным состоит из комбинации утилит, компонентов и сервисов для доступа к данным, необходимым для работы приложения.

При разработке логики взаимодействия слоев приложения обычно используют шаблоны, самыми распространенными из которых являются MVC, MVP, MVVM и VIPER [10].

Первые три модели предполагают отнесение объектов приложения к одной из трех категорий:

– Model – предоставляет логику работы с данными и описание данных, необходимых для работы приложения.

– View – графический интерфейс пользователя.

– Controller/Presenter/ViewModel – компонент обеспечивающий взаимодействие между моделью и представлением, ответственный за изменение модели в ответ на действия пользователя в представлении и изменение представления в соответствии с обновлением модели.

VIPER (View-Interactor-Presenter-Entity-Router) в свою очередь является реализацией принципов чистой архитектуры для iOS, где interactor содержит в себе бизнес-логику приложения, presenter логику подготовки содержимого для отображения, router – навигационную логику приложения, а entity — это объекты, которыми управляет interactor.

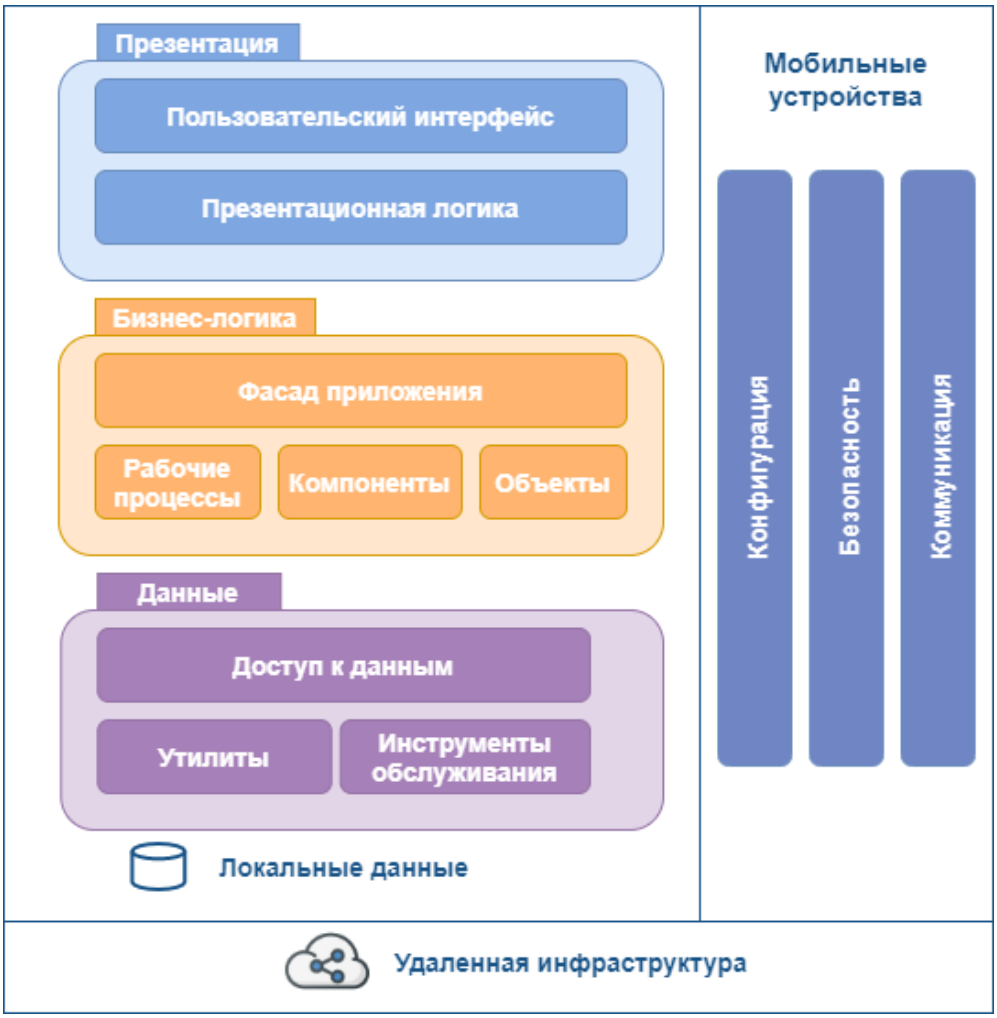

Рисунок 2 – Архитектура мобильного приложения

Слой представления содержит компоненты пользовательского интерфейса, а также компоненты, обрабатывающие их. Слой бизнес-логики определяет логику работы приложения, его основные функции, зависимости поведения объектов предметной области.

Слой доступа к данным состоит из комбинации утилит, компонентов и сервисов для доступа к данным, необходимым для работы приложения.

При разработке логики взаимодействия слоев приложения обычно используют шаблоны, самыми распространенными из которых являются MVC, MVP, MVVM и VIPER [10].

Первые три модели предполагают отнесение объектов приложения к одной из трех категорий:

– Model – предоставляет логику работы с данными и описание данных, необходимых для работы приложения.

– View – графический интерфейс пользователя.

– Controller/Presenter/ViewModel – компонент обеспечивающий взаимодействие между моделью и представлением, ответственный за изменение модели в ответ на действия пользователя в представлении и изменение представления в соответствии с обновлением модели.

VIPER (View-Interactor-Presenter-Entity-Router) в свою очередь является реализацией принципов чистой архитектуры для iOS, где interactor содержит в себе бизнес-логику приложения, presenter логику подготовки содержимого для отображения, router – навигационную логику приложения, а entity — это объекты, которыми управляет interactor.

1.2 Компоненты и работа приложений в ОС Android

При разработке приложения необходимо учитывать структуру компонентов, из которых состоит любое приложение Android. Каждый компонент – это отдельный способ общения приложения с системой и отдельная точка входа в приложение. Взаимодействие всех этих компонентов обеспечивает работу приложения, а каждый из них является самостоятельной структурной единицей и выполняет свою конкретную функцию. Компонент также имеет собственный жизненный цикл, определяющий как он создается и уничтожается. Всего существует четыре типа компонентов: операции, службы, поставщики контента и приемники широковещательных сообщений [37].

Activity (операции) – предоставляют интерфейс взаимодействия пользователю. Обычно одна из операций в приложении является главной и предлагается пользователю при запуске приложения. Дизайн операционной системы позволяет одновременно запускать множество операций, само приложение может переключаться между ними во время своего выполнения. Каждый раз, когда запускается новая операция, предыдущая операция останавливается, однако система сохраняет ее в стеке.

Services (службы) – работают в фоновом режиме и не содержат пользовательского интерфейса. Эти компоненты могут продолжать работу даже в случае переключения пользователем на другое приложение и даже если приложение не запущено, например для запуска оповещений.

Служба также может быть запущена другим компонентом, который затем будет взаимодействовать с ней.

Content Provider (поставщик контента) – управляет данными, которые могут храниться в файловой системе, базах данных или в сетевых хранилищах, к которым может получить доступ приложение. Они инкапсулируют данные и предоставляют механизмы обеспечения их безопасности. С помощью этого компонента приложение может осуществлять обмен данными с другими приложениями.

Broadcast Receiver (Приемник широковещательных сообщений) – предоставляет приложениям данные вне пользовательского взаимодействия. Этот компонент позволяет приложениям реагировать на объявления, которые распространяются по всей системе или самой системой, такие как выключение экрана или низкий заряд аккумулятора.

Любое приложение Android может запускать компоненты других приложений. Поскольку у каждого приложения есть свои права доступа, и система ограничивает доступ одного приложения к другим, такой запуск не может происходить напрямую. Для этого система использует так называемый Intent (намерение) – асинхронное сообщение, которое связывает операции, службы и приемники разных приложений во время их выполнения [9].

При таком вызове система запускает отдельный процесс требуемого приложения и экземпляры классов, необходимых для работы нужного компонента. Именно поэтому в приложениях Android нет единой точки входа в виде функции main.

Кроме этого, важной частью приложения является ее файл манифеста. Именно из этого файла система узнает о наличии компонентов в программе, поэтому они должны быть объявлены в нем. В нем же указываются необходимые программе разрешения, аппаратные функции, требования к API системы.

При использовании в приложении различных ресурсов, таких как изображение или видео, SDK для каждого из них задает уникальный целочисленный идентификатор, на который и нужно ссылаться во время работы приложения. Благодаря такому подходу Android реализует возможность использования альтернативных ресурсов для разных конфигураций устройств.

Еще одной важной концепцией в работе приложения в системе Android является жизненный цикл операции. Управление этим циклом происходит с помощью методов обратного вызова, сигнализирующих создание, приостановление, возобновление и остановку процесса. Всего операции предоставляется шесть таких методов, переопределяя которые, можно напрямую управлять жизненным циклом операций.

Правильная имплементация этих методов важна, чтобы избежать сбоев в работе приложения, например в случае получения пользователем телефонного вызова или переключении на другое приложение. Она также позволяет избежать лишней траты ресурсов системы, когда приложение не активно, избежать утраты данных. Поэтому важно понимать, когда система использует эти вызовы [7].

Так метод onCreate(), вызываемый при создании активности системой, присутствует в каждой операции и фрагменте, так как через его переопределение назначается вывод макета соответствующего этим компонентам на экран. При этом метод onDestroy() вызывается не только в случае завершения приложения, но и в случае временного уничтожения активности, при изменении конфигурации, таком как, например, изменение ориентации при повороте экрана [31].

Жизненный цикл операции изображен на рисунке 3.

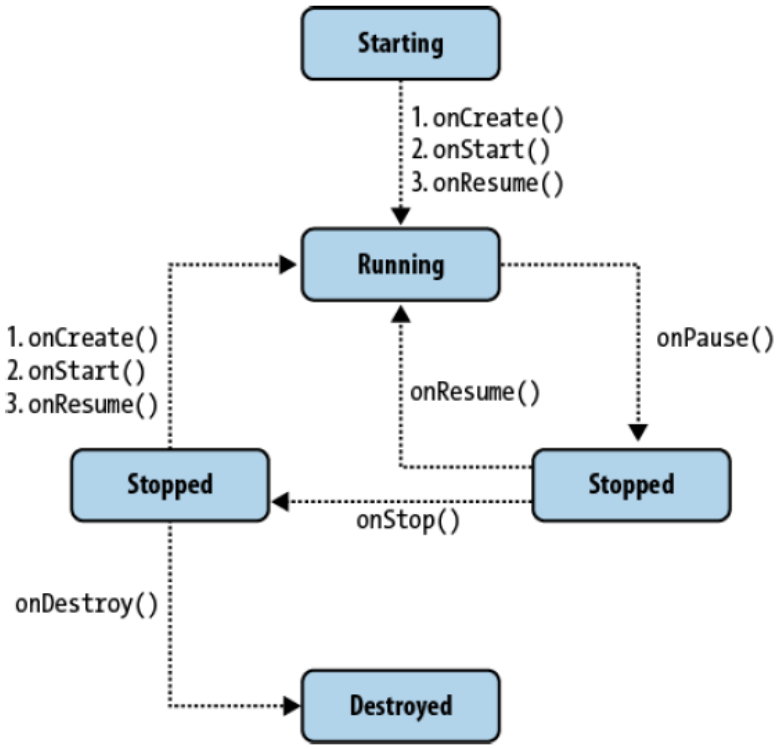

Рисунок 3 – Жизненный цикл операции [24]

1.3 Социальные функции и значение мобильных устройств и приложений

Доступ к информации людей с ограниченными возможностями является не менее значимой проблемой чем развитие доступной инфраструктуры для таких людей. Более 50% мирового интернет-трафика приходится на мобильные устройства. Их распространенность и относительная доступность могут позволить решить некоторые проблемы людей с ограниченными возможностями.

Инициатива создания сервисных систем для социального сектора в настоящее время исходит от самых разнообразных организаций: коммерческих, некоммерческих, частных и государственных. Разрабатываемые приложения позволяют получить с помощью смартфона или другого мобильного устройства самый различный спектр услуг от цифровой регистратуры, до получения результатов анализов онлайн [14].

Современные информационные технологии обладают рядом функций, способных предоставить людям с ограниченными возможностями доступ к различным информационным средам и услугам. В таблице 1 приведены примеры таких функций и решаемых ими ограничений. Примером таких функций может быть использование SMS для общения людьми с нарушениями речи и слуха.

Растущий спрос на приложения и устройства, реализующие несколько способов коммуникации с ними, как у людей с какими-либо нарушениями, так и без, стал стимулом, который привел к распространению такого подхода у создателей операционных систем, приложений и устройств [22].

Таблица 1 – Виды нарушений и соответствующие технологические решения

Вид нарушения	Примеры ограничений	Виды доступного технологического решения
Нарушения зрения	Чтение, навигация в пространстве (в случае текстового исполнения вывесок и указателей)	Синтез речи Распознавание речи Увеличение текста и экранного изображения Аудио-аутентификация GPS-навигация Изменение настроек яркости и цветовых схем интерфейса Дисплей Брайля
Нарушения слуха	Восприятие звуковых уведомлений и вербальной коммуникации	Субтитры для видео и телепрограмм SMS и текстовые сообщения Вибрация вместо звукового оповещения
Психосоциальные нарушения	Сложности в принятии решений и реагировании на инструкции, в запоминании, коммуникации.	Средства онлайн-коммуникации, документации Средства организации и помощи в запоминании
Нарушения речи	Вербальная коммуникация	SMS и текстовые сообщения Синтез речи

Окончание таблицы 1

Вид нарушения	Примеры ограничений	Виды доступного технологического решения
Нарушения моторики и двигательные нарушения	Ограничения передвижения и использования письменных принадлежностей	Распознавание речи Специальные и экранные клавиатуры Распознавание интуитивных жестов и взгляда Дистанционные и онлайн-системы доступа к сервисам
Когнитивные нарушения	Сложности в запоминании, следовании инструкциям, коммуникации. Часто появляется вместе с другими нарушениями	Синтез речи Средства организации и помощи в запоминании GPS-навигация Использование мультимедийных ресурсов для помощи в запоминании
Нарушения речи	Вербальная коммуникация	SMS и текстовые сообщения Синтез речи

Очевидно, есть множество видов решений, способных помочь оптимизировать устройства и приложения для людей с различными ограничениями. Кроме этого, довольно часто одному человеку необходимо несколько видов таких решений одновременно, но благодаря возможности установки многих приложений сразу и подключению к веб-сервисам мобильные технологии могут этого достичь.

К счастью, большинство таких решений в настоящее время являются частью предлагаемого операционной системой функционала. Такие технологии обозначаются собирательным термином «ассистивные технологии», предназначенные для поддержания на прежнем уровне или повышения функциональных возможностей и автономности людей.

В случае мобильных устройств, количество и исполнение таких функций варьируется от одной операционной системы к другой. Android для их предоставления использует Android Accessibility Suite – коллекцию приложений помогающую людям с некоторыми нарушениями использовать устройство, благодаря имплементации различных способов общения с устройством и приложениями.

В Android Accessibility Suite входят:

– TalkBack – программа, помогающая людям с нарушениями зрения, озвучивает все значки, кнопки и другие элементы на экране, когда пользователь прикасается к ним. Она также включает в себя режим чтения текста на экране;

– Switch Access – позволяет использовать подключаемые кнопки или клавиатуры вместо тачскрина;

– Select to Speak – озвучивает выделенный текст на экране и при наведении на него камеры;

– При нарушениях моторики, полезным может оказаться Assistant/Accessibility menu, позволяющий использовать экранные кнопки и жесты, вместо кнопок устройства, для настройки звука, навигации и других функций.

Пользователь Android также может использовать голосовой ввод, для отправки текста в любое приложение. Для организационных решений в системе присутствует встроенный календарь.

1.4 Использование мобильных приложений при когнитивных расстройствах

Демографические изменения приводят к повсеместному старению населения, особенно в развитых странах. Так по данным Global Age Watch Index 2017 в Европе процент людей старше 60-ти лет составил 29,7% населения [25]. В России доля людей в возрасте разнится по регионам и сдерживается достаточно высокой смертностью, но в среднем составляет порядка 18%.

Тем не менее существующий тренд сопровождает также некоторое количество возрастных заболеваний, таких как деменция. По данным Всемирной Организации Здравоохранения более 20% людей в возрасте старше 60 лет страдают деменцией или другими видами неврологических ментальных нарушений.

Между тем множество других заболеваний могут вызывать нарушения когнитивных функций: болезнь Альцгеймера, Паркинсона, аутоиммунные и воспалительные энцефалопатии, травмы, опухоли. Нарушение памяти могут быть вызваны также нарушениями сна, депрессиями и тревожными расстройствами.

Тем не менее достаточно часто при некоторых из этих заболеваний, либо на их ранних стадиях, люди с когнитивными нарушениями способны самостоятельно ухаживать за собой при достаточно небольшой внешней поддержке или даже ее отсутствии, благодаря изменению образа жизни, в котором некоторые приложения могут помочь как больным так и их попечителям. В задачи таких приложений могут входить, например, планирование активности, напоминание о приеме лекарств и коллекции изображений предметов и людей, которых необходимо запомнить [27].

Согласно оценке HealthWorks Collective в 2018 году в магазинах приложений для разных мобильных устройств можно найти порядка 97,000 приложений, тематикой которых являются здоровье и фитнесс [40].

Такие приложения могут помогать человеку отслеживать информацию о своём здоровье, находить и следовать рекомендациям по физическим упражнениям, напоминать о приеме лекарств, планировать питание и даже связываться со своим врачем.

Согласно исследованию BioMed Research International в 2019-ом году, проводившегося для оценки четырех главных когнитивных функций – языковой, визуальной, визуально-пространственная ориентации, памяти и решения проблем – приложения для когнитивных упражнений и повседневной активности, куда вошли также и приложения планирования, могут помочь пациентам с деменцией лучше справляться с ежедневной деятельностью. Оно также показало, что приложения с более простым дизайном операций оказались для пациентов более удобными, а обилие функционала и сложный лексикон, наоборот, сказались на применимости приложения отрицательно [30].

Хотя клинических испытаний доказывающих эффективность использования мобильных приложений при деменции и других заболеваниях, вызывающих нарушения памяти, очень мало, исследования демонстрируют, что применение таких технологий способно обеспечить повышение уровня жизни, а так же облегчить симптомы депрессии и стимулировать когнитивные функции.

Таким образом результаты исследования показывают, что применение мобильных технологий для помощи людям с когнитивными нарушениями может облегчить жизнь как людям, страдающим нарушениями, так и их попечителям. Но при создании таких приложений необходимо учитывать, что обилие функционала может запутать пользователя.

Кроме этого, многие когнитивные расстройства, такие как, например, рассеянный склероз, которое в виду специфики заболевания может поразить несколько различных отделов нервной системы одновременно, чаще всего сопровождаются каким-либо сопутствующим видом нарушений – это могут быть как визуальные или слуховые нарушения, так и нарушения моторики. Поэтому при создании приложения необходимо максимально использовать предоставляемые системой возможности, чтобы обеспечить удобство применения для целевой аудитории.

Еще одной важной концепцией в плане предоставления средств и информации людей с разными нарушениями является доступность.

Многие законы о доступности цифрового контента во всём мире основаны на критериях выполнения Web Content Accessibility Guidelines (WCAG) [39], разработанных W3C (Консорциум Всемирной паутины) в рамках Web Accessibility Initiative. Не исключение и «ГОСТ Р 52872–2019» [6], тоже разработанный на основе WCAG.

В случае Android разработки основным источником информации по оформлению приложения является Material design guidelines [28]. Material design – это стиль графического дизайна интерфейсов программного обеспечения и приложений, разработанный компанией Google в 2014-ом году.

Его целью является создание фундамента для разработки настраиваемого интерфейса, объединяющего принципы взаимодействия с пользователем на всех платформах.

Методические рекомендации этого стиля включают в себя не только лучшие практики оформления типографии, макетов операций и цветовых схем приложения, но и обширный обзор принципов включения специальных возможностей в рамках Material Design Accessibility Guidelines, в том числе для обеспечения интеграции ассистивных технологий платформы в работу приложения или его оптимизации для их использования.

Применение таких рекомендаций повышает удобство использования системы не только для людей с ограниченными возможностями и временными нарушениями (такими как, например, перелом руки), но и в целом улучшает качество и удобство интерфейса для любого пользователя.

Согласно этим принципам при разработке приложения необходим «чистый» дизайн интерфейса – это означает что разные части приложения должны быть легко идентифицируемыми и отличаемыми. Дизайн должен быть легко понятным и ясным – элементы должны находиться в предсказуемых местах, а их маркировки должны находиться в непосредственной близости к ним.

Для достижения такой цели дизайн приложения должен иметь подходящие цвета, контрастности и типографические характеристики. Конкретные рекомендации можно найти в стандартах W3C, но Google предоставляет разработчикам набор утилит, таких как ColorTool, позволяющих не только выбрать подходящие цвета, но и оценить читаемость текстов программы.

Кроме этого, элементы программы должны иметь определенную иерархию – важные элементы интерфейса должны располагаться в самом верху и в самом низу экрана, стандартным расположением информации будет ее группировка сверху вниз по степени важности, родственные элементы должны находиться рядом друг с другом. Особое внимание следует также уделить размерам и расположению интерактивных элементов. Для большинства платформ рекомендуемый минимальный размер сенсорной области элементов – 48dp (независящих от плотности пикселей). Минимальным расстояние между интерактивными элементами считают 8dp.

Отдельное внимание следует уделить последнему принципу – определенности. Этот принцип означает, что при разработке приложения, рекомендуется имплементировать поддержку ранее рассмотренных ассистивных технологий. Оптимизация приложения для работы с уже встроенными в систему специальными возможностями, такими как средства чтения и увеличения содержимого экрана, требует понимания базовых принципов их работы.

Например, TalkBack может произносить текст приложения, но так как не все элементы интерфейса могут иметь текстовые поля, необходимо в тексте программы предоставить таким элементам специальную маркировку.

Все эти принципы позволяют построить приложение подходящее и удобное для использования людьми со зрительными нарушениями, которые могут, например, видеть только часть операции из-за использования экранного увеличителя. А также людьми с когнитивными расстройствами, которым необходима простота, понятность и логичная согласованность интерфейса.

Для проверки приложения на доступность Google предоставляет Accessibility Scanner. Accessibility Scanner сканирует приложение и предлагает улучшения, связанные с контрастностью, оформлением интерактивных элементов, описанием контента и так далее. Для этого необходимо установить и приложение, и сканер на смартфон/эмулятор и активировать сканер в настройках системы. После выдачи ему необходимых разрешений, сканер сможет выделять зоны, которые он считает проблемными, и указывать на классы/ресурсы в коде, которым, возможно, необходимо редактирование.

Выводы по разделу один

В результате анализа процессов создания мобильных приложений были рассмотрены типы мобильных приложений и виды их архитектур, а так же особенности их проектирования для устройств на основе операционной системы Android, такие как жизненный цикл операций и принципы функционирования основных компоненты системы: операций, служб, поставщиков контента и приемников широковещательных сообщений. Рассмотренные особенности применения мобильных приложений целевой аудиторией выявили необходимые требования для создаваемого приложения, реализация которых основана на принципах обеспечения доступности цифрового контента и проведенных исследованиях, выявивших дополнительные требования к структуре и интерфейсу приложения, такие как простота дизайна, минимализация функционала и упрощение используемого лексикона.

2 СРАВНИТЕЛЬНЫЙ АНАЛИЗ ОТЕЧЕСТВЕННЫХ И ЗАРУБЕЖНЫХ ТЕХНОЛОГИЙ

2.1 Обзор существующих мобильных приложений для планирования действий

В список основных функций приложения для планирования активности людей с умеренными расстройствами памяти должны входить:

– ежедневные повторяющиеся задания;
– напоминания по лекарствам и внешний вид лекарства;
– отслеживание общего прогресса выполнения заданий за день;
– элемент формального вознаграждения за этот прогресс.

При этом приложение должно отвечать всем требованиям доступности специфическим для целевой аудитории и специфической задачи приложения: простой дизайн, минимум функционала, соответствие Material Design Accessibility Guidelines.

Поиск аналогичных приложений велся по ключевым словам: задания, органайзер, ежедневник, планер. Среди найденных приложений многие имеют часть требуемых функций или какой-либо способ их достижения, но не отвечают специфическим требованиям. В качестве примеров для сравнения выбраны приложения День за днём органайзер PRO, Google Tasks и Habitica.

День за днём органайзер PRO, интерфейс которого изображен на рисунке 4.

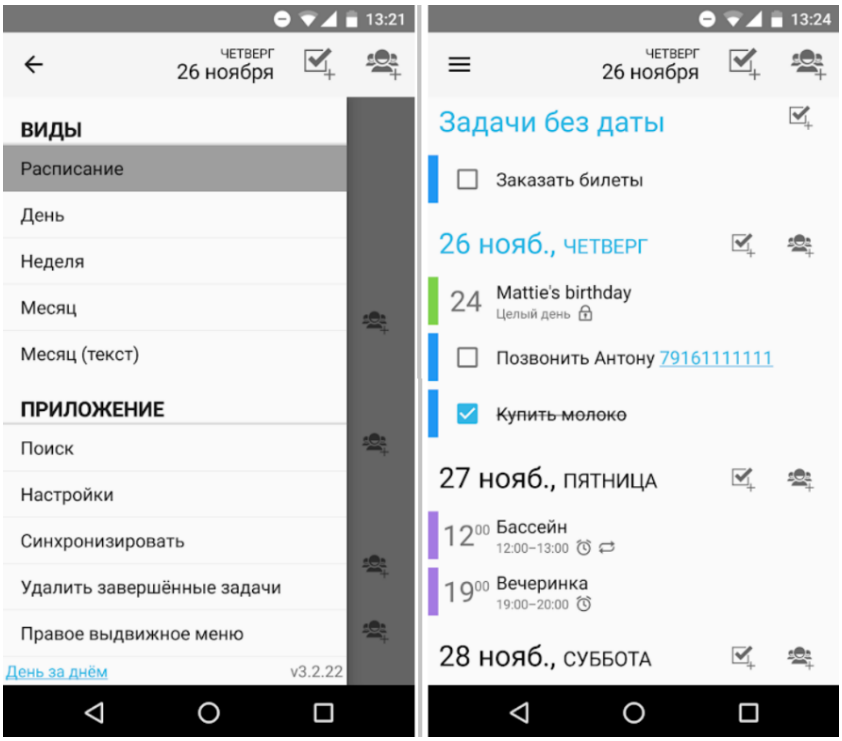

Рисунок 4 – День за днём органайзер PRO

Кроме требуемого планировщика заданий предлагает так же календарь, голосовой ввод и виджет, но существенно ограничивает функционал бесплатной версии приложения.

Относительно требуемой задачи, имеет слишком много функций: среди них выбор вида отображения задач, отображение дней рождений контактов, календарь, возможность использования нескольких профилей, приглашать и просматривать гостей в мероприятиях. Нет возможности добавить изображение для лекарства, отсутствует индикатор прогресса и элемент вознаграждения за выполнение задач, необходимые при проблемах самоорганизации и концентрации.

Google Tasks отвечает всем требованиям к интерфейсу, прост и понятен в использовании, можно установить период повторения задания на необходимое ежедневное, как показано на рисунке 5, отсутствует премиум версия, поэтому весь функционал доступен сразу, доступна так же синхронизация через аккаунт Google. Нет возможности добавить изображение для напоминания про лекарство, отсутствует индикатор прогресса.

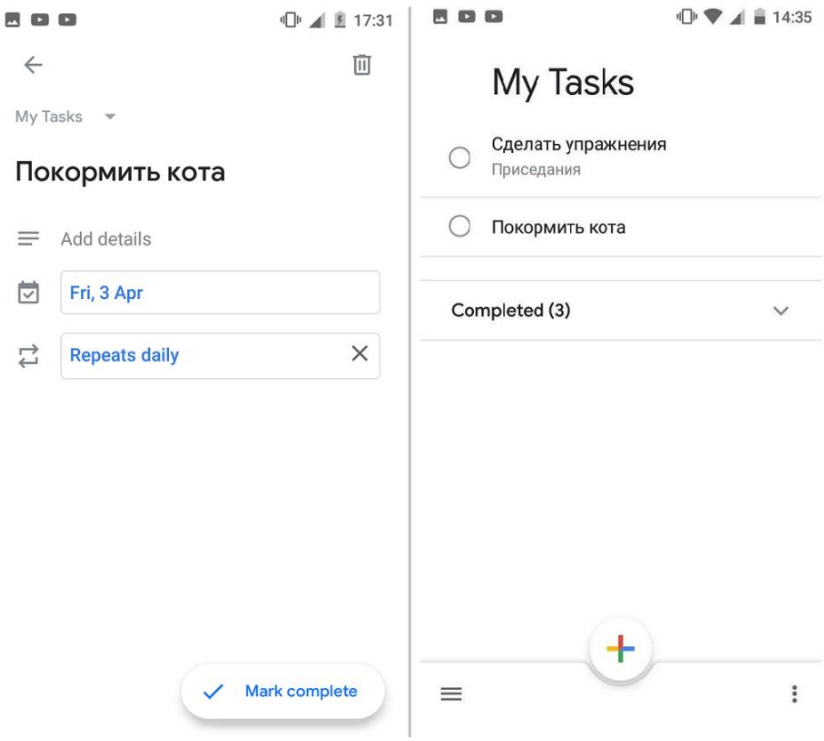

Рисунок 5 – Google Tasks

Поскольку во всех приложения данного формата в основном отсутствуют именно эти функциональные требования, в качестве примера приложения с отслеживанием прогресса и реализацией формального вознаграждения за выполнения заданий отдельно приведена Habitica, интерфейс которой изображен на рисунке 6.

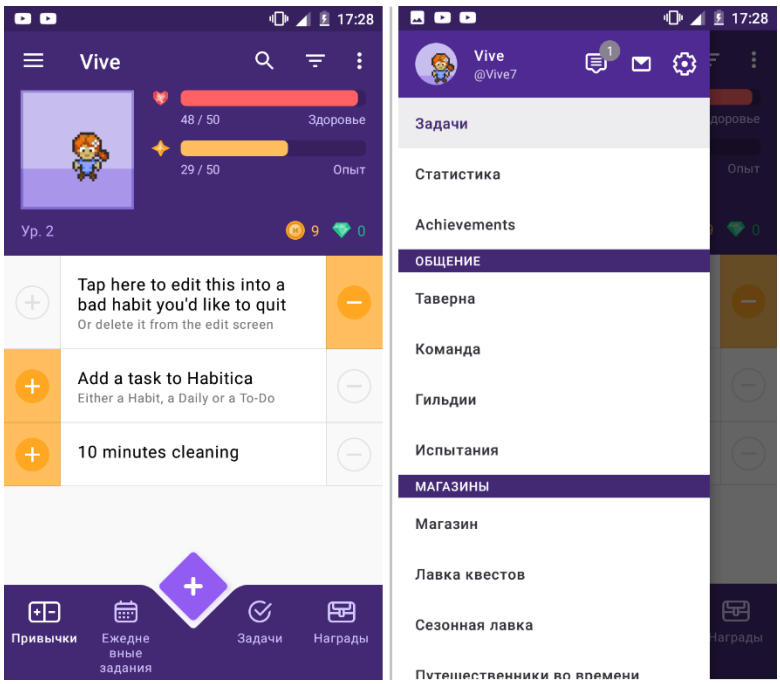

Рисунок 6 – Habitica

Habitica также обеспечивает пользователя возможностью добавлять ежедневно повторяющиеся задания и задачи с неустановленной датой, элемент формального вознаграждения за выполнение заданий, как в кратковременном так и в долговременном смысле, в виде имплементированных RPG-элементов, предоставляет платформу для общения. Однако, в рамках поставленной задачи, излишняя геймификация является отрицательной чертой, точно так же, как и ее огромное количество функций.

Таким образом рассмотренные приложения не отвечают в полной мере решению поставленных в работе задач, поэтому принято решение об актуальности создания собственного мобильного приложения, полностью отвечающего специфическим требованиям к оформлению и функционалу.

2.2 Обзор технологий создания мобильных приложений

Одной из главных задач при разработке нативного приложения является выбор целевой платформы, нак который влияют множество факторов, таких как целевая аудитория, распространенность платформы, знание языка и средств разработки, подходящих для данной ОС. Согласно рисунку 7, на сегодняшний день Android сохраняет лидерские позиции по количеству пользователей в мире с большим отрывом от других участников рынка. Так доля Android в 2020 году составила 72.26 %, а iOS 27.03 %. Для России эти цифры составляют в 2020 году 73.97 % и 25.52 % соответственно [33].

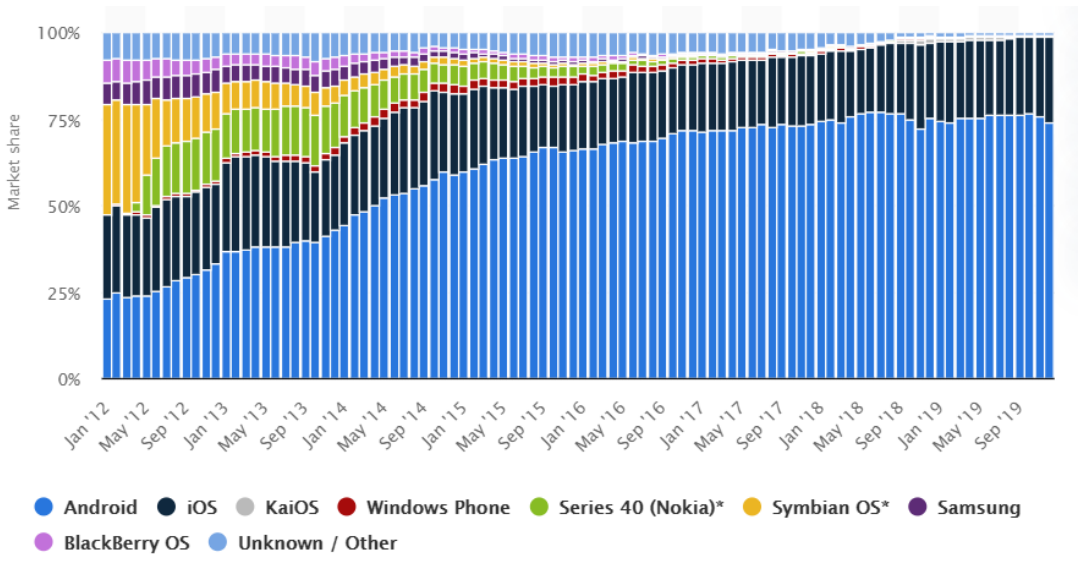

Рисунок 7 – Диаграмма соотношения мобильных операционных систем на рынке смартфонов

Таким образом, с учетом того, что оставшаяся рыночная доля равна всего 0.71 %, при этом основными платформами в которой являются KaiOS, предназначенная для кнопочных мобильных телефонов и Windows Phone, поддержка которой прекращена в декабре 2019 года, в реальности можно говорить о выборе между двумя лидирующими операционными системами.

Кроме объективной распространённости устройств с Android причиной выбора данной операционной системы является ее относительная ценовая доступность по сравнению с устройствами iOS ориентированными в основном на аудиторию с премиальным уровнем дохода, что противоречит исходной цели создания приложения, обладающего максимальной доступностью.

Android – это комплексная платформа для мобильных устройств с открытым исходным кодом, разработанная на основе ядра Linux и виртуальной машины Java. Эта операционная система разрабатывалась компанией Android, Inc., которую Google приобрела в 2005-ом году. Первый смартфон, работающий на Android, был выпущен уже в 2008-ом. В настоящий момент последней версией ОС Android является Android 10.0, поддерживающая в том числе и устройства со складными экранами.

Исходный код платформы, известный как Android Open Source Project, выпущен под лицензией Apache 2.0, которая разрешает использование программного обеспечения для любых целей, а также свободное изменение и распространение изменённых копии (за исключением названия). Такой подход позволяет разработчикам изучать работу операционной системы, участвовать в ее изменении и доработке, использовать ее код для создания новых программ, а производителям легко портировать Android на конкретные устройства.

Комплексность означает, что Android – это полный программный стек для мобильных устройств. Для разработчика приложений платформа предоставляет необходимые системы и фреймворки для разработки. Разработка для Android не требует наличия у разработчика необходимого устройства, для этого нужен только Android SDK (Software Development Kit).

Для потребителя это означает, что устройства, работающие на Android, можно использовать сразу после покупки – для этого не требуется никакой дополнительной сборки или настроек. Для производителей мобильных устройств Android может потребовать только разработки некоторых драйверов для конкретных аппаратных средств.

Поскольку Android был разработан для работы на мобильных устройствах, его работа не привязана к конкретному разрешению, размеру экрана и процессору, учтены особенности работы таких устройств, такие как использование аккумуляторов в качестве источника питания и соответствующие этому ограничения энергопотребления.

Android разработан на основе ядра Linux, поэтому наследует некоторые его особенности. Одной из таких особенностей является абстракция от аппаратных средств – при разработке приложений разработчику не нужно учитывать их особенности. Такие функции Linux как управление памятью, энергопотреблением и сетевыми соединениями так же используются операционной системой Android.

Android, как и Linux, является высокозащищённой системой – каждое приложение запускается как отдельный процесс, разрешения для которого выставлены Linux-системой. Система назначает каждому из приложений собственный идентификатор пользователя и определяет полномочия для всех файлов этого приложения, а у каждого процесса имеется своя виртуальная машина, которая выполняет код приложения изолированно от других приложений. Это означает, не только то, что оно не может помешать работе других приложений, но и имеет доступ только компонентам системы, необходимым ему для работы. Однако при установке оно также может запросить разрешение на доступ к данным на устройстве.

Android также использует набор нативных библиотек для обеспечения работы приложений, таких как OpenGL для 2D и 3D графики, SQLite для баз данных, браузерный движок Webkit и другие. Для работы в составе Android Google также разработала свой вариант стандартной библиотеки языка Си – Bionic, для работы на процессорах с низкими частотами, использующий меньше ресурсов, но не реализующий некоторые функции полноразмерной библиотеки.

Приложения для Android имеют несколько точек входа – то есть состоят из отдельных компонентов, которые могут быть вызваны независимо друг от друга. Например, некоторый компонент может выполнятся в фоновом режиме, пока другой предоставляет интерфейс пользователю, или вызываться отдельно от остальной программы, например при использовании оповещений.

Существует так же специальный объект Intent, позволяющий запустить один компонент из этого или другого приложения. Он может так же принимать различную информацию в состав своего пакета, для ее пересылки.

Выбор языка программирования для создания приложения также основан на особенностях операционной системы Android, так как создается нативное приложение, то использование web-ориентированных языков HTML, CSS и JavaScript не является уместным. Вся официальная документация к данной операционной системе предоставляется на двух языка: Java и полностью совместимом с ним Kotlin.

Java считается одним из лучших языков для разработки приложений. Индикатор популярности языков программирования TIOBE index в 2020 году показал, что Java продолжает лидировать в этом списке, тогда как Kotlin оказался только на 30-ом месте [38]. В то же время Kotlin всё еще считается относительно новым языком программирования несмотря на то, что Google официально признала его приоритетным языком Android-разработки в 2019 году. Так опрос Jexenter в 2018 году, поместил Kotlin на пятое место среди технологических трендов, хотя Java 9 оказался на первом месте и там. Основными преимуществами Kotlin являются улучшенная производительность, легкость его изучения и его полная совместимость с Java – то есть даже если у приложения уже есть большая кодовая база на Java, эти методы можно вызывать из Kotlin.

В то же время у Kotlin всё еще достаточно небольшое сообщество разработчиков, поэтому при изучении разработки на Android, многие вопросы могут остаться без ответов. Java в свою очередь, хоть и считается языком сложнее и медленнее, такого недостатка лишен, и в свою очередь не ограничен разработкой только Android приложений и всё еще остается значимым благодаря своей кроссплатформенности и огромному количеству готовых библиотек и SDK для облегчения процесса разработки [1, 2].

Kotlin является относительно новым языком, созданным на основе Java, он не только похож на него, но и использует Java-библиотеки, поэтому в случае необходимости обновления навыков при изменении трендов в области языков программирования, его изучение при знании Java составит не так много труда. Именно поэтому для реализации приложения в данной работе выбран язык программирования Java.

Язык Java является полностью объектно-ориентированным. Это означает, что любая программа, написанная на языке Java, должна поддерживать парадигму объектно-ориентированного программирования [29].

Программа на Java компилируется для Виртуальной Машины Java. Результатом этой компиляции является байт-код, который в свою очередь может выполняться на любой операционной системе при условии наличия там Java Runtime Environment – минимальной реализации виртуальной машины, необходимой для исполнения Java-приложений, которая интерпретирует байт-код в реальный машинный код конкретной системы [15, 16].

В системе Android полученный байт-код компилируется еще раз в собственный формат байт-кода. Android использует собственную виртуальную машину для запуска кода – ART (Android Runtime), заменившую Dalvik в версии Android 5.0. Их архитектура и различия представлены на рисунке 8.

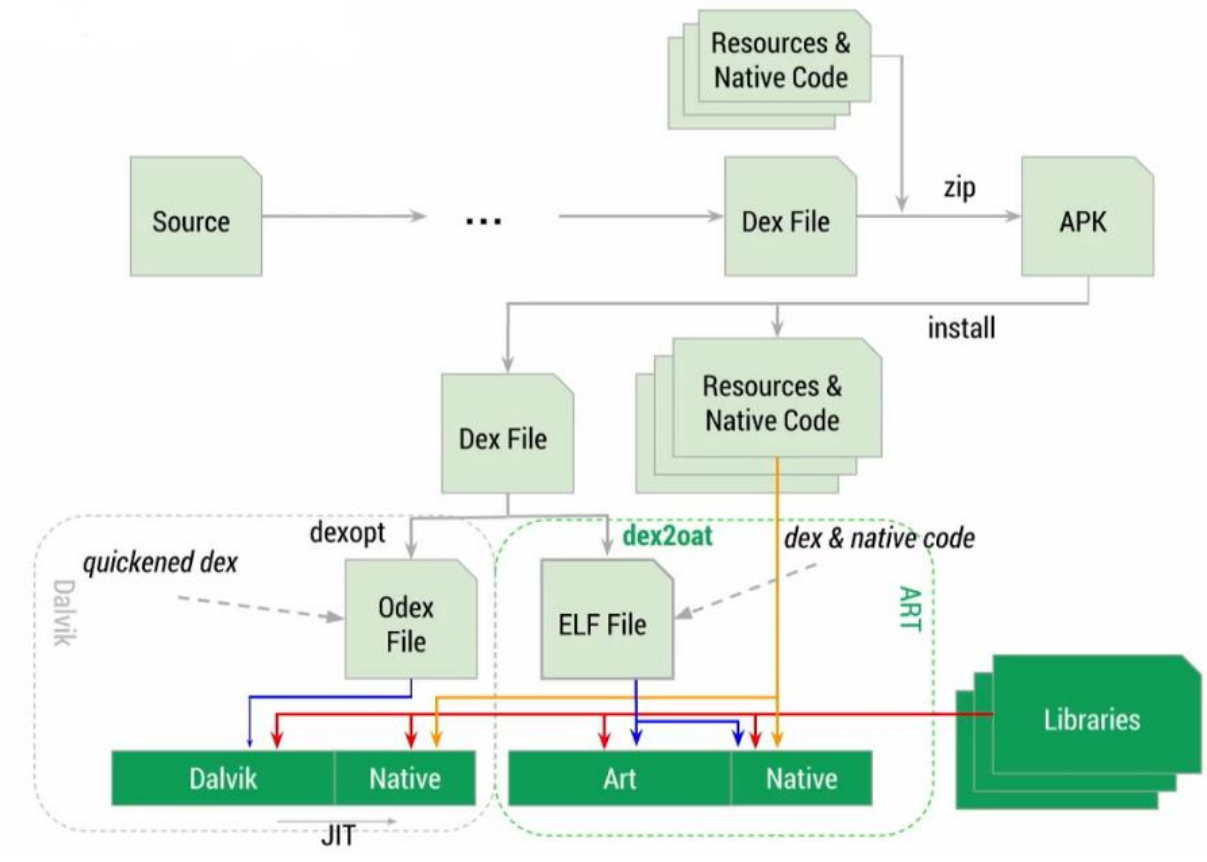

Рисунок 8 – Архитектуры Dalvik и ART [21]

В отличии от предыдущей версии виртуальной машины, использующей JIT-компиляцию (Just-in time) совершаемую во время запуска программы, ART выполняет AOT-компиляцию (ahead-of-time) приложения во время его установки, что увеличивает скорость работы программ и их энергоэффективность ценой увеличения времени загрузки устройства и установки приложения. В версии Android 7.0 представлен также и JIT-компилятор, что далее увеличивает производительность приложений при запуске.

При выборе технологии хранения данных необходимо оценить масштаб и структуру приложения, типы хранимых данных. Популярными решениями для Android приложений в этой сфере можно назвать MySQL, PostgreSQL и SQLite.

MySQL – это реляционная база данных, для хранения данных в таблицах которой используются различные движки, которые отличаются между собой методами считывания и записи данных на диск. PostgreSQL – объектно реляционная база данных.

Это означает, что все таблицы в этой БД являются объектами, поэтому она поддерживает такие объектно-ориентированные функции как наследование или перегрузка методов.

Для решения задачи хранения данных в Android обычно используется SQLite. Эта база данных является встраиваемой и включена в системы Android по умолчанию. В то время как для большинства баз данных необходим процесс сервера базы данных, SQLite использует для записи обычный файл [32].

Так как приложение не подразумевает хранения больших объемов информации, синхронизации с другими устройствами и доступа нескольких пользователей, то в качестве используемой базы данных можно не рассматривать серверные БД такие как MySQL и использовать встроенное системное решение, поэтому для данного проекта был выбран SQLite.

Движок SQLite представляет собой не отдельно работающий процесс, а библиотеку, в качестве протокола обмена с которой используются вызовы ее функций. Такая база данных является оптимальной для мобильных устройств, она не расходует процессорное время, пока не используется, чтобы избежать разрядки аккумулятора.

SQLite позволяет нескольким процессам одновременно читать данные из одной базы, но запись можно осуществлять только в том случае, если никаких других запросов в данный момент не обслуживается; в противном случае в программу возвращается код ошибки.

Android автоматически создает для приложения директорию для базы данных и файла журнала. В файле журнала хранится информация обо всех изменениях, внесенных в базу данных. Если в работе с данными возникает проблема, Android использует данные журнала для отката последних изменений.

Для создания приложения на языке Java доступно множество сред разработки. Одними из самых популярных при этом являются IntelliJ IDEA, Eclipse и, в случае Android-разработки, Android Studio.

Eclipse была разработана компанией IBM в 2001-ом году, эта среда разработки полностью бесплатна, имеет открытый исходный код, написанный преимущественно на Java. Благодаря этому эта IDE обладает портативностью, позволяющей ей работать на Linux, Mac OS X, Solaris и Windows. Однако основным преимуществом Eclipse является модульная архитектура, позволяющая использовать ее не только с Java. В настоящее время для Eclipse написано более 1700 плагинов, как официальных, так и неофициальных. Плагины Eclipse, поддерживают более 100 языков программирования и почти 200 фреймворков для разработки приложений. Но именно из-за этого эту IDE сложно настроить, при отсутствии опыта ее использования.

IntelliJ IDEA, созданная JetBrains в 2001 году, имеет две официальные версии – коммерческую Ultimate edition и бесплатную Cummunity Edition, с разным доступом к некоторым дополнительным функционалам.

Последняя предназначена именно для Java и Android разработки, поддерживает Java, Kotlin, Groovy и Scala; Android; Maven, Gradle и SBT; работает с системами контроля версий Git, SVN, Mercurial и CVS.

IDEA также предлагает «умное автодополнение» показывающее список наиболее релевантных символов, применимых в данном контексте. Список символов зависит не только от общепринятого контекста, но и от того насколько часто разработчик использует те или иные операторы. Кроме того, в случае со статическими членами или константами IDEA автоматически добавляет любые необходимые операторы импорта.

Android Studio, в отличии от Eclipse и IDEA является узконаправленной IDE созданной Google совместно с JetBrains специально для Android Разработки. Android Studio основана на IntelliJ IDEA, поэтому также использует умное автодополнение. Кроме этого, она предоставляет множество специфичных функций, таких как эмулятор, возможность предпросмотра приложения на разных конфигурациях экрана, Instant run, позволяющий отправлять обновления в запущенное приложение. Всё это и легкость ее использования для Android разработки и является причиной выбора этой IDE для использования в данном проекте.

Выводы по разделу два

Таким образом, в результате рассмотрения аналогов приложения, было выявлено, что они не соответствуют или частично не соответствуют требованиям предъявленым к функционалу и оформлению приложения, выявленным в результате анализа в разделе один данной работы. На основе сравнительного анализа программных средств и технологий создания мобильных приложений, было принято решение о разработке собственного мобильного приложения для платформы Android на основе Java, SQLite и Android Studio.

3 РЕАЛИЗАЦИЯ МОБИЛЬНОГО ПРИЛОЖЕНИЯ ДЛЯ ПЛАНИРОВАНИЯ АКТИВНОСТИ

3.1 Архитектура приложения и базы данных

В данной работе приложение реализовано по принципу разделения ответственностей и с использованием MVVM-архитектуры, схема которой представлена на рисунке 9. Такой шаблон используется для разделения бизнес-логики приложения и её представления, что позволяет изменять их отдельно друг от друга [20].

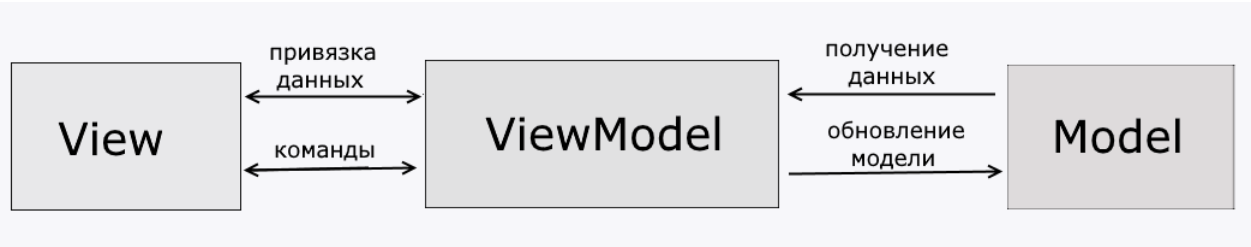

Рисунок 9 – Шаблон MVVM

View – это представление, определяющее визуальный интерфейс, через который пользователь взаимодействует с приложением. Model описывает содержащиеся в приложении данные, предоставляет логику работы с ними, при этом в рамках этого шаблона проектирования она не содержит логики коммуникации с представлением и интерфейсом. Для этого используется ViewModel – абстракция представления, связывающая его с моделью.

В данной работе приложение реализуется в рамках системы Android, для которой Google предоставляет определенные архитектурные компоненты, для реализации подобных моделей разделения доступа к данным. Среди них классы LiveData и ViewModel, а также библиотека Room [17].

LiveData – это класс хранилище данных, работающий про принципу паттерна Observer. Он может не только хранить объекты, но и предоставляет механизм получения объектов из него для своих подписчиков.

При этом класс сам определяет активен запрашивающий данные подписчик или нет и отправляет данные только в первом случае. Списки из базы данных, необходимые для работы приложения будут хранится именно в таких классах.

Класс ViewModel позволяет операциям и фрагментам сохранять необходимые им элементы интерфейса при изменениях конфигурации, таких как, например, поворот экрана. Таким образом при пересоздании интерфейса, необходимая информация, например списки заданий, будет сохранена в этой модели и интерфейсу не придется запрашивать ее снова.

Room предоставляет уровень абстракции для базы данных и имеет три основных компонента: Entity, Dao и Database.

Эта библиотека облегчает работу с объектами базы данных SQLite и проверяя SQL-запросы во время компиляции, что позволяет уменьшить объем стандартного кода. С помощью методов этой библиотеки так же будут настроены правила миграции базы данных и наполнения при ее создании.

Рекомендуемая Google архитектура приложения, в случае использования архитектурных компонентов, представлена на рисунке 10.

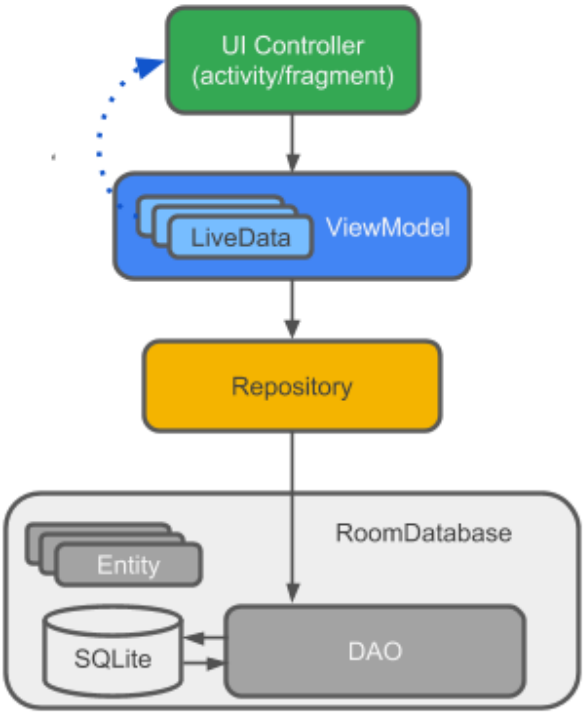

Рисунок 10 – Рекомендуемая архитектура приложения [26]

Основные компоненты архитектуры:

– база данных SQLite, в которой хранятся данные на устройстве;

– библиотека Room используется как уровень абстракции базы данных для ее создания и обслуживания;

– аннотируемый класс Entity, описывающий таблицу базы данных, и предоставляющий интерфейс для получения и установления значений объекта;

– интерфейс DAO (Data Access Object), который хранит SQL-запросы в виде функций;

– модуль Repository непосредственно взаимодействует с данными и предоставляет API, который остальная программа может использовать для доступа к данным;

– ViewModel обеспечивает коммуникацию между репозиторием и интерфейсом программы;

– класс, использующий шаблон «наблюдатель» [4] LiveData, содержащий последнюю версию данных программы, оповещающий наблюдателей об изменении данных;

– пользовательский интерфейс программы, предоставляемый фрагментом или операцией.

Именно данные рекомендации и шаблон проектирования были использованы при создании программы.

Приложение для Android OS состоит из активностей, фрагментов и вспомогательных классов, каждому из которых соответствует .java файл, в котором они реализованы. Каждой активности и фрагменту при этом назначен свой .xml файл, содержащий их описание: расположение и оформление визуализируемых объектов и элементов интерфейса.

Проектируемое приложение предоставляет возможность как планировать ежедневные действия, так и следить за приемом лекарств самостоятельно. Архитектура приложения представлена на рисунке 11.

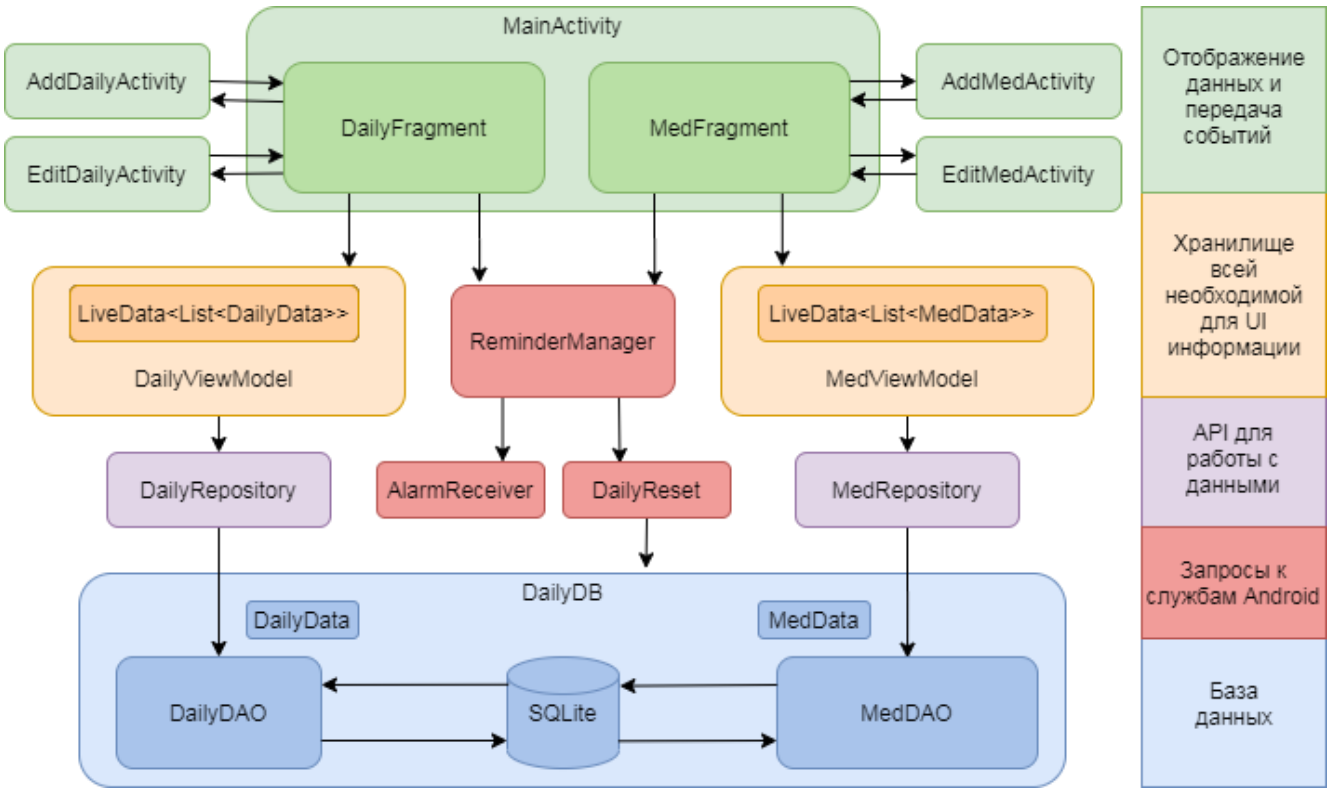

Рисунок 11 – Архитектура приложения

Основные различия от шаблонной архитектуры вызваны тем, что программа использует две таблицы базы данных SQLite: одну для хранения ежедневных заданий, другую для лекарств, и два соответствующих этим таблицам фрагмента приложения. А также использует системную службу AlarmService для запуска оповещений и ежедневного обновления статуса элементов базы данных в системе Android.

База данных и содержащиеся в ней таблицы представлены на рисунке 12.

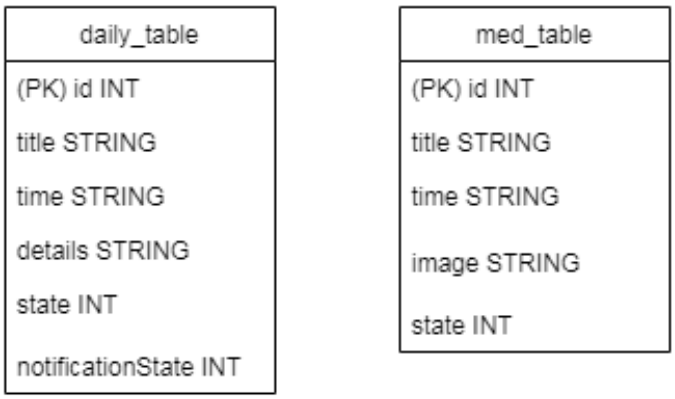

Рисунок 12 – База данных приложения DailyDB

Обе таблицы почти идентичны и хранят идентификационный номер, название, время и переменную state, используемую в данном приложении, как способ регистрации выполнения задания/приема лекарства, где 1 – активное, невыполненное задание или лекарство, а 2 – неактивное, то есть выполненное задание/лекарство. Различием таблиц является четвертый столбец, который в рамках ежедневных заданий хранит детали задания, а в рамках лекарств хранит изображение лекарства в виде строки с названием файла изображения, и столбец, хранящий состояние оповещения для задания [3].

3.2 Модули работы с данными и службами ОС Android

В данном разделе представлено описание трех архитектурных категорий программы: базы данных, запросов к службам ОС Android и API для работы с данными.

Для описания работы модулей программы в дальнейшем будут использованы UML-диаграммы входящих в них классов [11, 35]. В этой и следующей главе сначала будут рассмотрены классы, отвечающие за хранение данных о ежедневных заданиях и функционирование части программы, ответственной за их обработку и отображение. А после этого показаны различия в зеркальных классах хранения и обработки данных о лекарствах, возникающие в силу отличий в части их представления, структуре таблиц базы данных и соответственных методах их обработки в силу этих различий.

Приложение работает с реляционной базой данных SQLite, которая является встраиваемой, поэтому не представляет собой отдельный процесс, с которым программа должна взаимодействовать, а становится составной частью программы, так как ее движок представляет собой библиотеку, с которой программа компонуется.

В первую очередь диаграмма, изображенная на рисунке 13, показывает классы работы с базой данных и службами оповещения и сигнализации ОС Android.

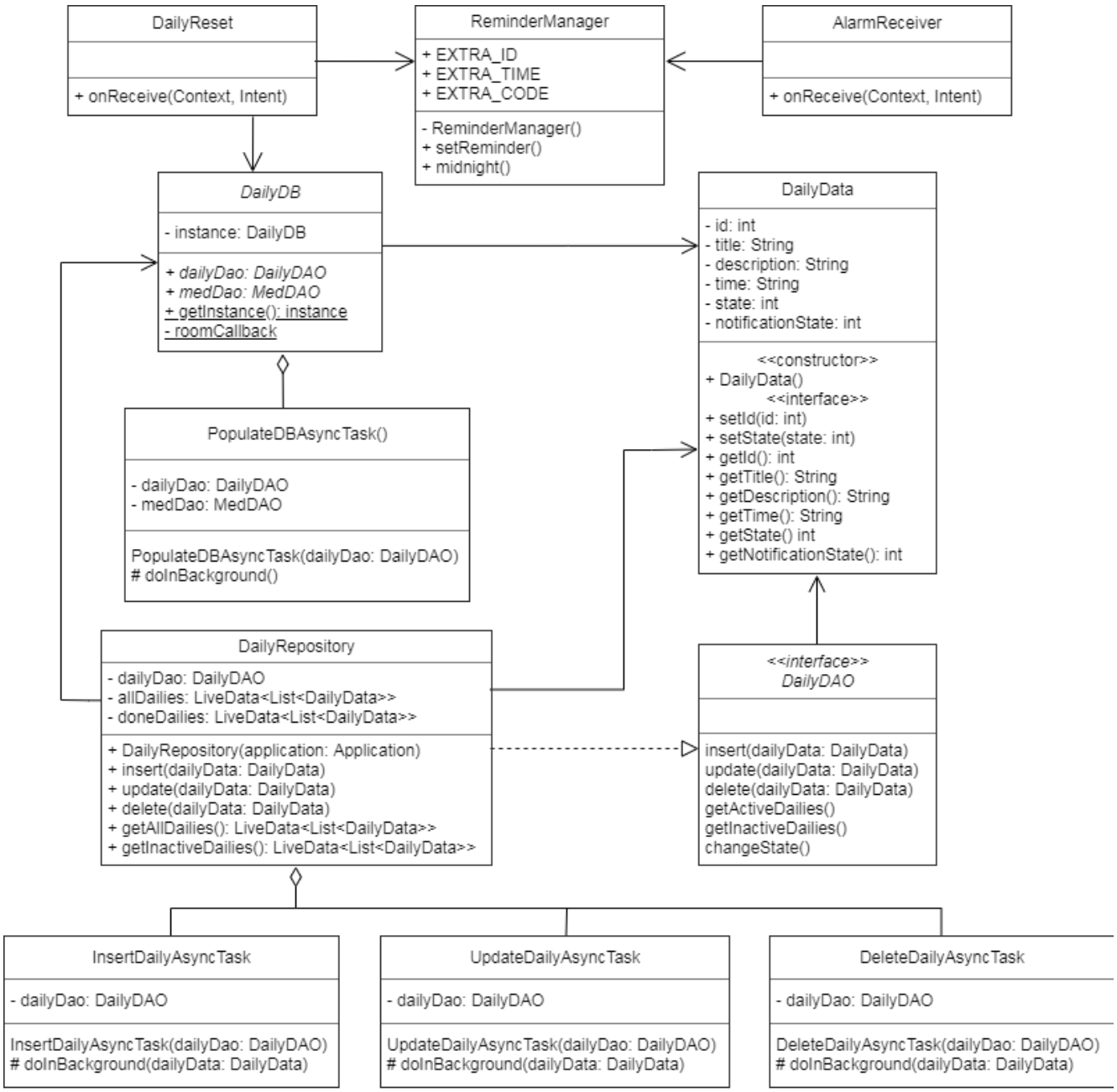

Рисунок 13 – Модули работы с базой данных и таблицей ежедневных заданий

Создание и дальнейшая работа с базой данных используют Android библиотеку Room, поэтому объекты базы данных создаются в аннотируемых классах как Room Entity. Класс DailyData представляет собой объект базы данных, из которого Room создает таблицу базы данных. Здесь специальные аннотации с соответствующими параметрами перед объявлением переменной, указывают библиотеке особые детали создания таблицы в SQLite.

Например, @Primary key назначает столбец ключевым, @Entity(tableName="daily_table") определяет название таблицы, а @ColumnInfo (defaultValue = "1") присваивает столбцу значения по умолчанию.

Кроме этого, класс предоставляет методы установления и получения значений строк таблицы, а также конструктор для создания записей в ней.

Класс DailyDAO содержит пустые методы insert, delete и update, так как библиотека Room автоматически генерирует код для таких методов, в случае использования соответствующих аннотаций. В нем так же представлены методы запросов к базе данных getActiveDailies, getInactiveDailies, и changestate с аннотацией @Query, используемые для наполнения списков активных и выполненных заданий и изменения столбца state, для возвращения всем заданиям статуса активных в начале нового дня, реализованные в виде SQL-запросов [5, 12].

Класс DailyRepository предоставляет интерфейс для взаимодействия презентационного слоя с данными. Необходимые для работы приложения списки в нем так же объявляются, как хранящиеся в LiveData-объектах, что позволяет установить наблюдение за их изменениями. В первой половине класса создается репозиторий необходимых данных из экземпляра базы данных DailyDB и переопределяются методы из DailyDAO, предоставляющие доступ к этим данным для ViewModel. Библиотека Room по умолчанию не позволяет совершать запросы к базе с основного потока, чтобы избежать долгой загрузки ответа интерфейса. Поэтому этот класс так же является агрегатором трех статических классов [12, 23], наследующихся от AsyncTask, которые создают асинхронные операции для методов обновления, удаления и создания объектов в базе данных. AsyncTask используется для перемещения операций в фоновый поток и обеспечивает синхронизацию обработчиков событий с графическим потоком, что позволяет обновлять элементы пользовательского интерфейса для вывода результатов, когда задача завершена. Основные операции здесь производятся в переопределенном методе doInBackground, а полученные данные передаются в качестве параметров методу execute() при вызове операции в основном классе.

Однако не все методы в репозитории требуют отдельного создания асинхронных операций. Исключением являются запросы возвращающие LiveData, так как обновление этого класса по умолчанию происходит через фоновый поток, поэтому для методов возвращающих списки типа LiveData<List<DailyData>> асинхронные задачи не определяются.

DailyDB – абстрактный класс базы данных, так же аннотируемый Room. Здесь перед определением класса указываюся параметры базы данных, такие как содержащиеся в ней таблицы и версия базы данных. В этом классе так же создается пустой метод доступа к DAO, генерируемый Room автоматически. Экземпляр базы данных, если такового еще нет, создается синхронизированным методом, что означает, что доступ к нему одновременно может иметь только один поток, обеспечивающий создание единственного экземпляра. В противном случае этот же метод возвращает экземпляр базы данных.

Для создания базы данных используется DatabaseBuilder из состава Room, позволяющий с помощью специальных методов определить правила миграции данных.

В данном классе используется fallbackToDestructiveMigration, который в случае изменения структуры базы данных пересоздает ее, уничтожая старые данные, а так же addCallback, использующийся для наполнения базы данных примерами заданий при создании.

ReminderManager – вспомогательный класс, работающий со службой сигнализации ОС Android. Для этого он использует класс AlarmManager, предоставляющий доступ к этому системному сервису [36]. Внутри класса определен пустой закрытый статический конструктор, и два метода для работы с AlarmManager. Метод setReminder используется для сбора информации и установления времени оповещений о заданиях, а midnight для обновления значений state в базе данных.

Для настройки сигнализации необходимо использовать так называемый PendingIntent, который включает в себя контекст, ключ запроса, объект-intent и определенный флаг.

Так в методе setReminder в качестве ключа используется идентификатор ежедневного задания, чтобы различать PendingIntent заданий между собой и при необходимости редактировать их. Intent включает в себя названия и идентификатор задания для их использования в классе-приемнике, флаг FLAG_UPDATE_CURRENT используется для обновления PendingIntent в случае поступления одинакового ключа. В зависимости от переданного в метод кода alarmCode в конце настройки либо устанавливается точная сигнализация, выводящая устройство из спящего режима для передачи сервису PendingIntent в определенное время каждый день с помощью метода setRepeating, либо отменяется и PendingIntent и созданный с его помощью AlarmManager. Для отправки PendingIntent используется метод getBroadcast, отправляющий его в виде широковещательного сообщения. Использование этого метода позволяет определить необходимые для обработки поступивших данных классы, как приемники широковещательных сообщений, зарегистрировав их в манифест-файле программы.

Данные, поступившие во вспомогательный класс ReminderManager, далее используются в зависимости от вызванного метода в классах-приемниках AlarmReceiver и DailyReset. AlarmReceiver, использует метод onReceive, доступный приемникам широковещательных сообщений, для передачи полученной информации в конструктор оповещений NotificationManager. Тот в свою очередь принимает название задания для отображения его в тексте оповещения и его идентификатор, для настройки оповещения. Класс-приемник DailyReset использует DailyDao и аналогичный интерфейс таблицы лекарств для вызова метода changeState на имеющийся экземпляр базы данных, активируя весь список заданий.

Далее рассмотрены классы, обеспечивающие методы для работы с таблицей лекарств из базы данных

Функциональными отличиями этой части программы при работе с базой лекарств является отсутствие одного столбца базы данных и наличие в DAO метода, возвращающего количество строк с невыполненными заданиями.

Вместе с соответствующим ему методом в репозитории он необходим для обеспечения работы индикатора ежедневного прогресса, из-за различий в оформлении фрагментов. Изображения для лекарств, аналогично зеркальному полю description в таблице ежедневных заданий, хранятся в текстовом формате String в виде имени файла изображения в ресурсах программы, поэтому в данной части программы не вызывают существенных изменений кода. Таблица MedData так же является объектом базы DailyDB, работающим с ней через свой собственный интерфейс MedDAO.

UML-диаграмма соответствующих классов приведена на рисунке 14.

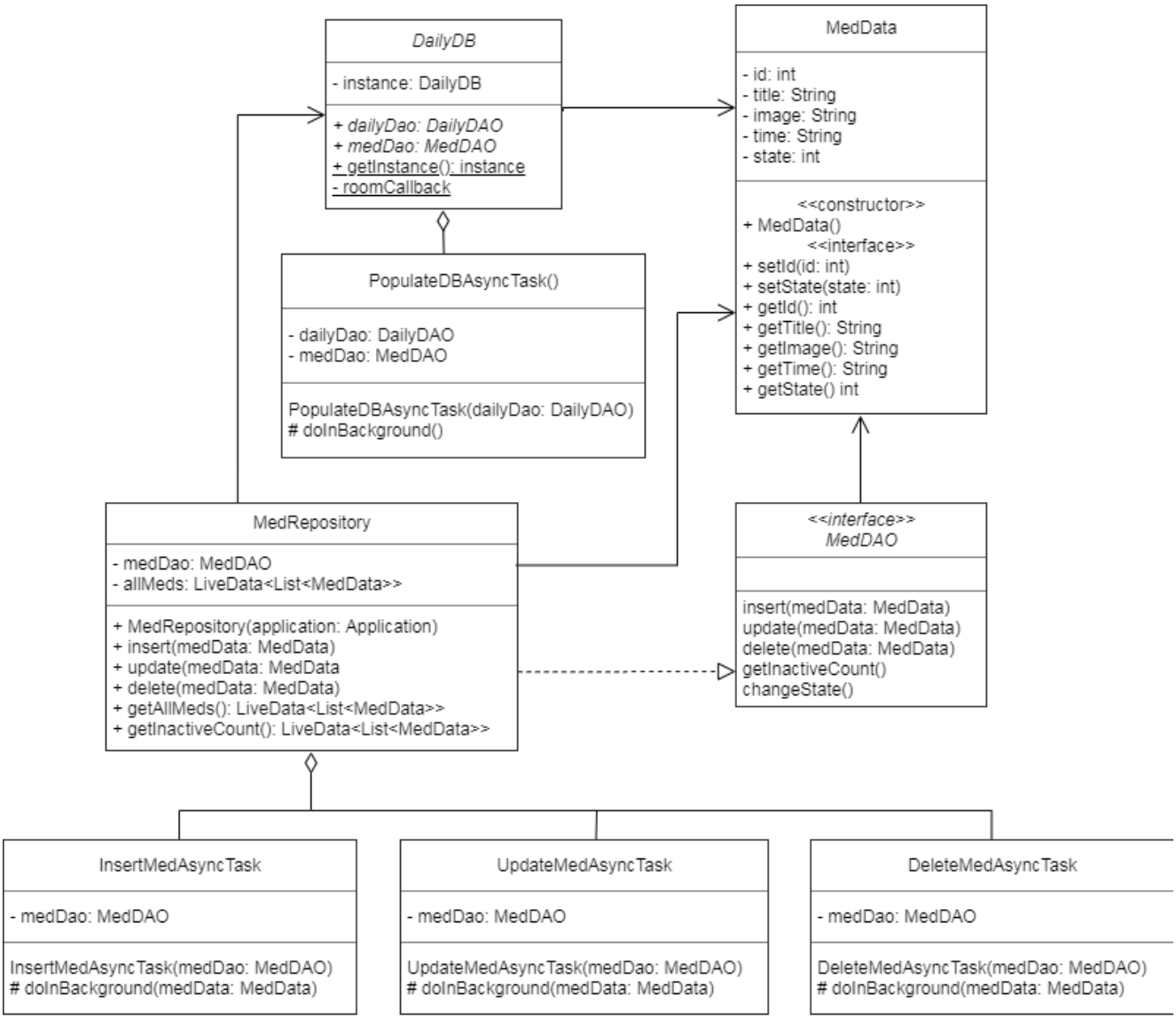

Рисунок 14 – Модули работы с базой данных и таблицей лекарств

3.3 Модули представления

В данном разделе представлено описание двух архитектурных категорий программы: отображение данных и передача событий и хранилище данных для UI.

На рисунке 15 изображена UML-диаграмма классов ответственных за обработку и представление информации из таблицы ежедневных заданий, ответ на взаимодействие с ней, а также связь этих модулей с ранее рассмотренными классами части программы, обеспечивающей работу с базой данных и сервисами ОС Android.

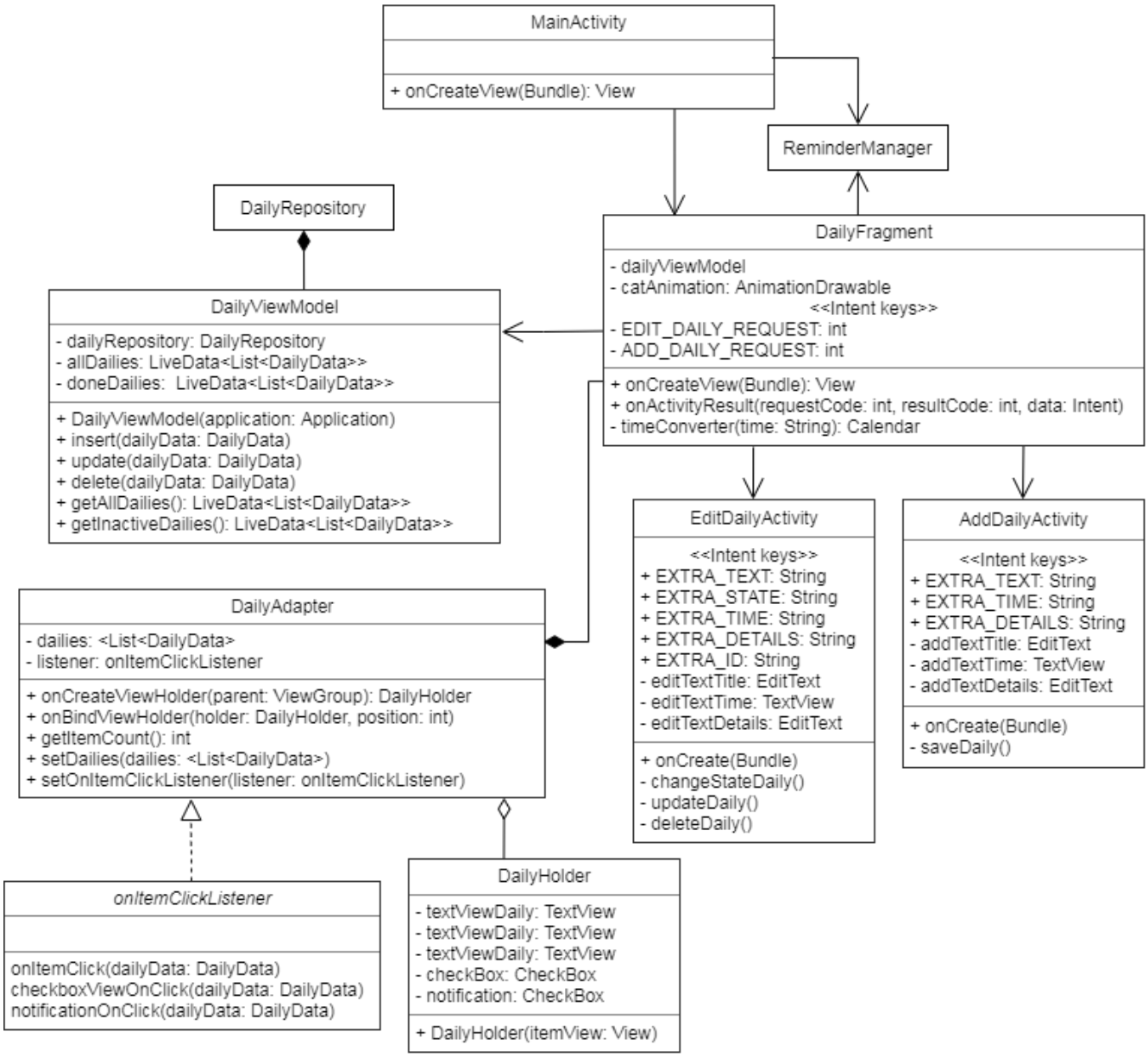

Рисунок 15 – Модули представления ежедневных заданий

DailyViewModel, согласно архитектуре, является связующим звеном между репозиторием и интерфейсом программы.

В этом классе создается экземпляр DailyRepository, LiveData-объекты, хранящие переданные оттуда списки. DailyViewModel так же реализует методы работы с этими данными, используемые фрагментами и операциями программы через репозиторий: создание, обновление и удаление записи, а также методы для наполнения списков активных и неактивных заданий.

DailyAdapter – отвечает за наполнение двух RecyclerView-списков активных и неактивных заданий. Элементы списка в нем представляют из себя сгруппированные view-элементы, каждый из который является экземпляром класса ViewHolder. DailyHolder является наследником этого класса, в нем идентифицируются необходимые элементы интерфейса и обработчики событий. В случае ежедневных заданий это текстовые поля для названия, деталей и времени задания, и два чекбокса: для управления оповещениями и статусом задания. Для последних инициируются обработчики, возвращающие во фрагмент позицию задания в списке и статус чекбокса. Для нажатия на сам элемент списка также устанавливается обработчик, для последующего открытия операции редактирования задания. Для наполнения RecyclerView используется отдельный xml-файл, определяющий конструкцию каждого элемента списка.

Особенностью RecyclerView является то, что он создает столько элементов соответствующего ему списка, сколько необходимо для отображения присутствующей на экране части данных и несколько дополнительных элементов, для оптимизации прокрутки этого списка [18]. Каждому элементу RecyclerView присваивается позиция в списке, к каждой из позиций в переопределенном методе onBindViewHolder привязывается элемент базы данных, информация из которого используется для заполнения текстовых полей каждого объекта ViewHolder.

В этот класс так же входят методы getItemCount, возвращающий размер списка заданий, и setDailies, используемый при назначении адаптера во фрагменте для заполнения RecyclerView согласно логике адаптера.

AddDailyActivity – класс операции добавления нового задания. В состав операции входят два поля редактирования текста для названия и деталей задания, одно текстовое поле, содержащее время задания и кнопки сохранения задания и отмены добавления. Полю времени задания назначен обработчик, при нажатии на него вызывающий стандартный TimePicker-диалог из состава Android и передающий выбранное значение обратно в текстовое поле. Здесь данные из интерфейса считываются и, в случае вызова метода saveDaily, создается Intent-объект data, содержащий дополнительные данные в составе своего набора с помощью метода putExtra. После этого данные возвращаются в DailyFragment через метод setResult, который принимает в себя data-объект и код результата выполнения операции.

За редактирование, удаление и смену флага активности задания отвечает операция EditDailyActivity. Она так же, как и AddDailyActivity, передает данные во фрагмент через Intent.

Каждый из методов, через который инициирована передача информации deleteDaily, updateDaile или changeStateDaily, возвращает свой код результата, так как по нему во фрагменте определяется цель передачи информации и необходимые действия с данными. В этой операции так же имеются два поля редактирования текста для информации о задании и текстовый элемент, вызывающий установщик времени. Но, кроме этого, здесь присутствует кнопка смены статуса задания на выполненное или возвращения его в список активных заданий и кнопка удаления задания, вызывающая перед действием диалог подтверждения.

DailyFragment – фрагмент основной операции, отвечающий за отображение списков заданий, индикаторов ежедневного прогресса о обрабатывающий данные, поступающие из вызываемых из него операций.

Метод timeConverter является вспомогательным методом, обеспечивающим конвертацию времени из типа данных String в Calendar для регистрации оповещения и методов обработки данных из AddDailyActivity и EditDailyActivity.

Он делит полученный текст на токены, разделением по знаку двоеточия и совершает необходимые настройки объекта calendar, прежде чем вернуть его значение в вызвавший его метод.

При создании фрагмента в методе onCreate происходит привязка адаптеров к RecyclerView-спискам, информации из ViewModel и ее наблюдателей к соответствующим этим спискам адаптерам, описание логики работы индикаторов прогресса и интерактивных кнопок элементов списка, таких как флажок активности задания и активации оповещения, а так же логика кнопки добавления задания из MainActivity.

Так один из RecyclerView-списков наполняется активными заданиями, с помощью методов setDailies из адаптера и getAllDailies из DailyViewModel, а второй выполненными заданиями с помощью getInactiveDailies. В это же время происходит привязка наблюдателя DailyViewModel к каждому из адаптеров, для оповещения модели об изменениях в данных и соответствующего обновления представления. Перед этим создается ViewModelProvider для этого фрагмента, который хранит ViewModel, пока фрагмента DailyFragment является активным.

Здесь же инициализируются обработчики событий из адаптера для каждого RecyclerView. При нажатии на сам элемент списка создается Intent, в который передаются данные об объекте dailyData согласно позиции элемента в списке. После этого запускается операция EditDailyActivity, с помощью метода startActivityForResult, в которую передается собранный Intent и код запроса.

При нажатии на флажок статуса задания меняется поле state соответствующего объекта, после чего вызывается метод обновления ViewModel и активируется проигрывание анимации в шапке программы.

При нажатии на флажок оповещения, фрагмент запрашивает идентификатор задания, обновляет поле notificationState соответствующего объекта, вызывает метод обновления ViewModel и передает необходимые данные в метод setReminder из класса ReminderManager с соответствующим кодом alarmCode.

Далее обработчик назначается текстовому полю, нажатие на которое отвечает за смену видимости списка неактивных заданий, а также настраивается логика работы шапки фрагмента, содержащей индикатор прогресса с соответствующим ему текстовым полем и поле с изображением-анимацией. Здесь метод адаптеров getItemCount используется для установления значений общего количества заданий и количества выполненных заданий. А полю изображения присваивается созданная из набора изображений анимация.

Кроме этого, в каждом фрагменте реализована своя логика для элемента FloatingActionButton из MainActivity приложения. В случае DailyFragment с помощью метода startActivityForResult запускается операция AddDailyActivity.

Метод onActivityResult ответственен за обработку данных, поступивших из операций Add/EditDailyActivity. Если кодом запроса операции был ADD_DAILY_REQUEST, то из переданного активностью Intent собираются данные, с помощью которых создается новый объект DailyData, после чего на объявленную во фрагменте DailyViewModel вызывается метод insert, принимающий эти данные и добавляющий их в таблицу через методы репозитория.

Операция редактирования задания может возвращает код операции EDIT_DAILY_REQUEST и три возможных кода результатов. В случае простого редактирования задания, метод так же собирает необходимую информацию из предоставленного Intent и создает новый объект DailyData, но присваивает ему переданный из операции идентификатор, тем самым заменяя объект. После этого так же вызывается метод обновления ViewModel а оповещение деактивируется, с помощью метода setReminder.

Другой код результата сигнализирует о необходимости проведения операции удаления задания. Эта операция реализована аналогично редактированию, но вызывает метод удаления из модели представления и деактивирует вызов сервиса сигнализации и отправки PendingIntent из класса ReminderManager.

Последним вариантом результата выполнения операций является код смены статуса задания, который также обновляет ViewModel, но перед этим использует методы получения поля state объекта DailyData для смены статуса на противоположный, что перемещает выбранный элемент из списка активных заданий в противоположный и наоборот.

Основные различия в соответствующих лекарствам классах вызваны особенностями оформления их списка и наличием выбора изображения лекарства. Так, например, адаптер получает необходимое изображение из ресурсов программы с помощью метода, получающего идентификатор изображения из его названия. Сам фрагмент же использует для отображения элементов один единственный список, форматируя его с помощью сортировки выборки из базы данных и смене формления элемента при изменении его статуса. UML-диаграмма классов лекарств изображена на рисунке 16.

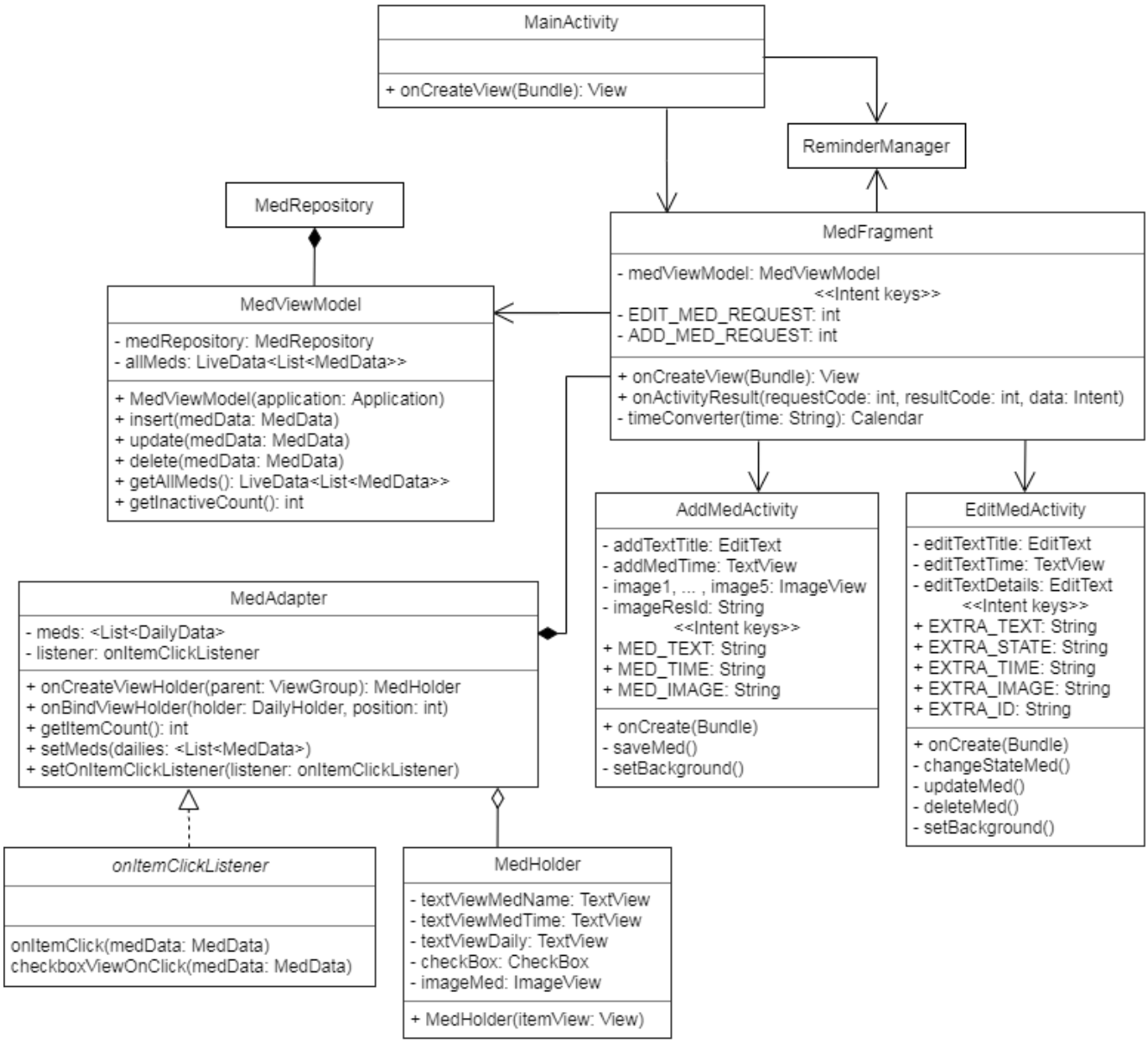

Рисунок 16 – Модули представления лекарств

Поэтому MedAdapter имеет несколько отличий от соответствующего адаптера для ежедневных заданий. Он не имеет обработчик событий активации оповещения, так как оповещения для лекарств всегда включены по умолчанию, а поле изображения генерируется из строки с названием лекарства, методом преображения его в идентификатор. Кроме этого, поскольку в данном случае адаптер назначается только один раз, так как во фрагменте лекарств присутствует только один RecyclerView, адаптер так же оказывается ответственен за обеспечение различия между принятыми и непринятыми лекарствами. В зависимости от статуса лекарства, MedAdapter меняет цвет поля времени приема лекарства и текста с его названием, а также прячет изображение лекарства. Эта же логика, кроме аналогичной заданиям передачи позиции элемента в списке, прописана в обработчике нажатия на флажок статуса лекарства.

В операциях удаления и редактирования лекарства название хранится в виде переменной типа String и при нажатии на изображение в списке их выбора, на изображение устанавливается фон в виде рамки, для индикации выбора пользователя, а переменная обновляется для последующей передачи во фрагмент.

В случае обоих фрагментов программы, MainActivity не предоставляет никаких методов, кроме метода построения макета приложения с общими для обоих фрагментов нижним меню навигации и кнопки добавления задания или лекарства, логика который определена в каждом из них. MainActivity также ответственна за вызов метода midnight из класса ReminderManager, ежедневно обновляющего статус заданий.

3.4 Разработка и тестирование пользовательского интерфейса

При разработке пользовательского интерфейса приложения необходимо учитывать рекомендации Material Design Guidelines, включающие в себя требования Консорциума Всемирной паутины о доступности цифрового контента.

В рамках проектирования и тестирования интерфейса, так же возможно использование утилит, предоставляемых компанией Google для разработчиков ОС Android. Так при выборе цветовой гаммы интерфейса и оптимизации цвета текста используется утилита Color Tool, изображенная на рисунке 17.

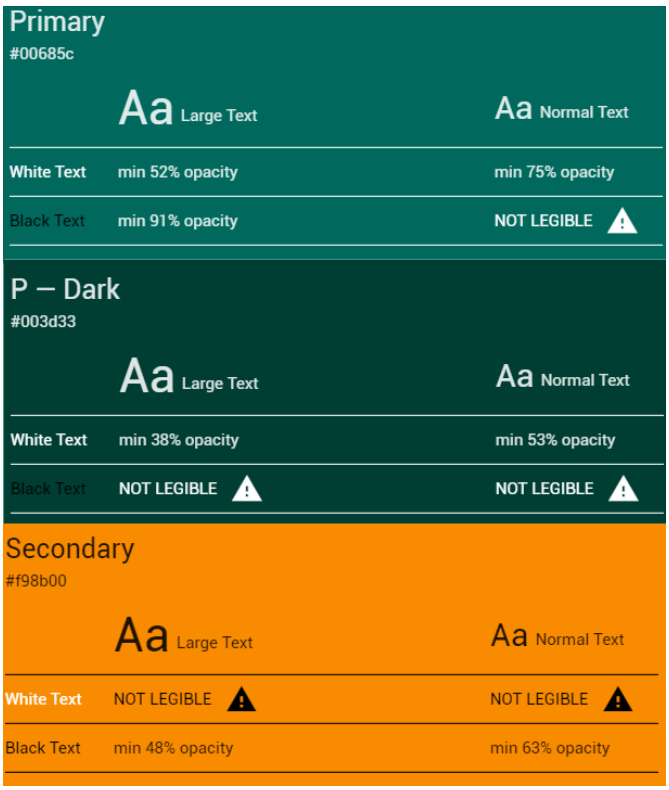

Рисунок 17 – Результат анализа утилитой Color Tool

Утилита рекомендует цвет текста, относительно цветовой схемы приложения и размеров символов. Соответственно для основного цвета при нормальном размере текста, утилита рекомендует использовать белый цвет текста с минимальной прозрачностью 75 %, а для вторичного черный, с минимальной прозрачностью 48 %. С помощью этой же утилиты можно так же подобрать собственный цвет текста для маркировки неактивного элемента меню и других элементов интерфейса.

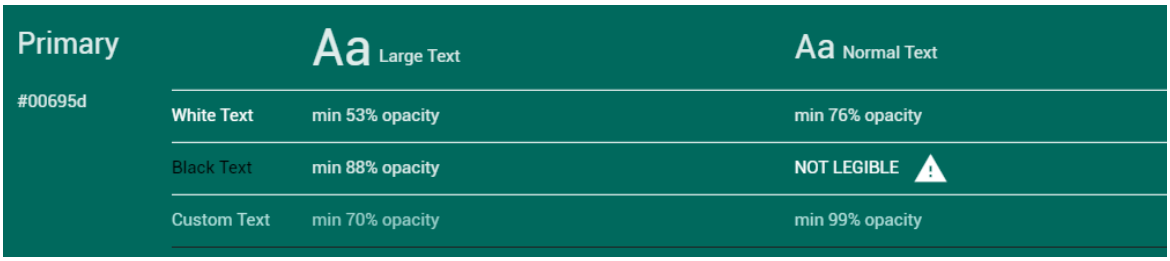

Рисунок 18 – Выбор дополнительного цвета текста и иконок меню

Основная операция программы содержит в себе два фрагмента, изображенных на рисунке 19, навигация между которыми осуществляется с помощью меню внизу экрана и кнопку добавления задания или лекарства в список.

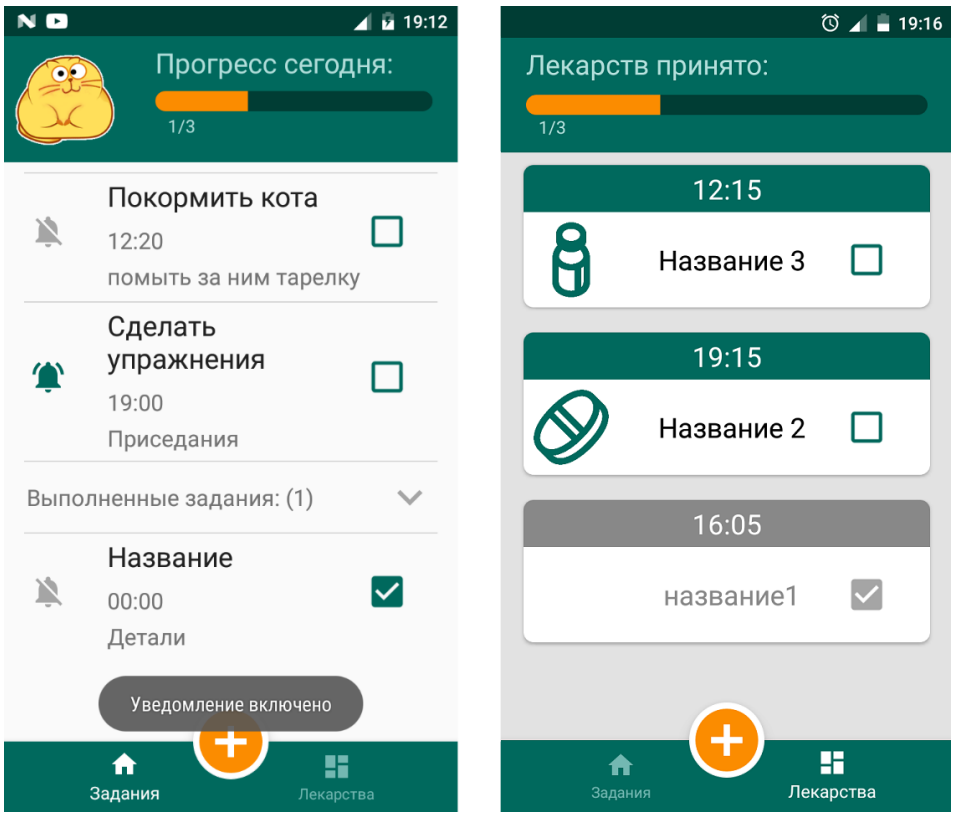

Рисунок 19 – Вид фрагментов меню

Фрагмент, предоставляющий информацию о лекарствах, содержит в себе один RecyclerView-список и индикатор прогресса их приема сверху экрана, с сопровождающим его текстовым элементом, предоставляющим более конкретную информацию. В каждом элементе этого списка, представляющем собой CardView, содержится основная информация о лекарстве: его название, время приема и изображение внешнего вида лекарства, а также интерактивный флажок для изменения его состояния.

При нажатии на элемент списка и кнопку добавления записи вызываются экраны редактирования и добавления лекарства соответственно, они изображены на рисунке 20. Экран редактирования позволяет также удалить задание, соответствующей кнопкой в правом верхнем углу экрана, или изменить его состояние, по нажатию кнопки «Выполнить».

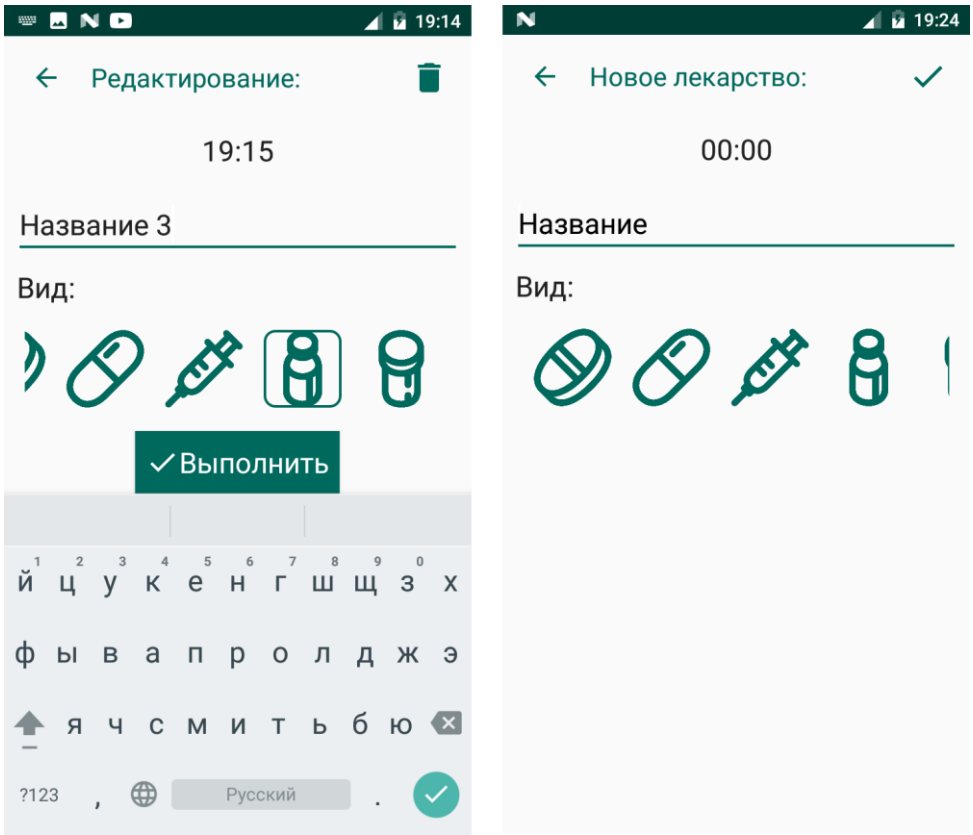

Рисунок 20 – Экраны редактирования и добавления лекарств

Оба экрана содержат поля редактирования текста, текстовое поле, вызывающее диалог TimePicker системы Android для выбора времени и список доступного для установки внешнего вида лекарств, реализованного в виде горизонтального ScrollView, снабженного механизмом прокрутки элементов, чтобы обеспечить их видимость на разных размерах экрана.

Выбор элемента из этого списка также снабжен индикатором в виде рамки, вокруг него. Подтвердить завершение создания записи можно кнопкой в правом верхнем углу экрана.

В противном случае информация сбрасывается и при нажатии кнопки назад проихсодит возвращение на экран фрагмента.

Похожим образом реализованы экраны добавления и редактирования ежедневного задания, вызываемые из первого фрагмента. Единственным различием здесь является наличие поля редактирования текста вместо списка изображений. Внешний вид этих экранов показан на рисунке 21.

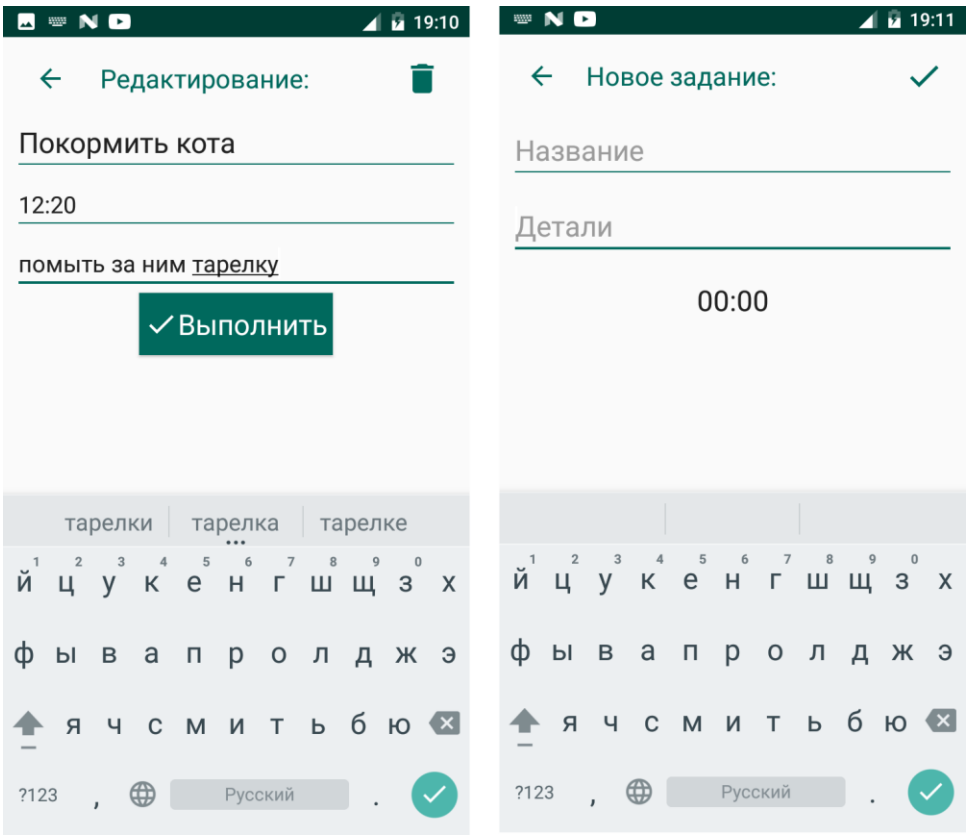

Рисунок 21 – Экраны редактирования и добавления задания

Кроме приведенных выше экранов в приложении так же присутствуют диалоговые окна. У них отсутствует реализация в виде xml-файла макета их внешнего вида. Вместо этого их вызов и настройка производятся внутри программного кода соответствующего фрагмента или операции с помощью конструкторов, таких как, например, AlertDialog.Builder, согласно цветовой теме приложения.

Обе версии этих экранов приведены на рисунке 22. Первое изображение – диалог с утилитой TimePicker из стандартных ресурсов ОС Android, предназначена для выбора времени в меню добавления и редактирования лекарства или задания.

Второе – диалог подтверждения удаления лекарства или задания из списка, вызываемый при нажатии кнопки в верхнем правом углу экрана.

На нем так же видна иконка оповещения этого приложения для данного задания в строке состояния.

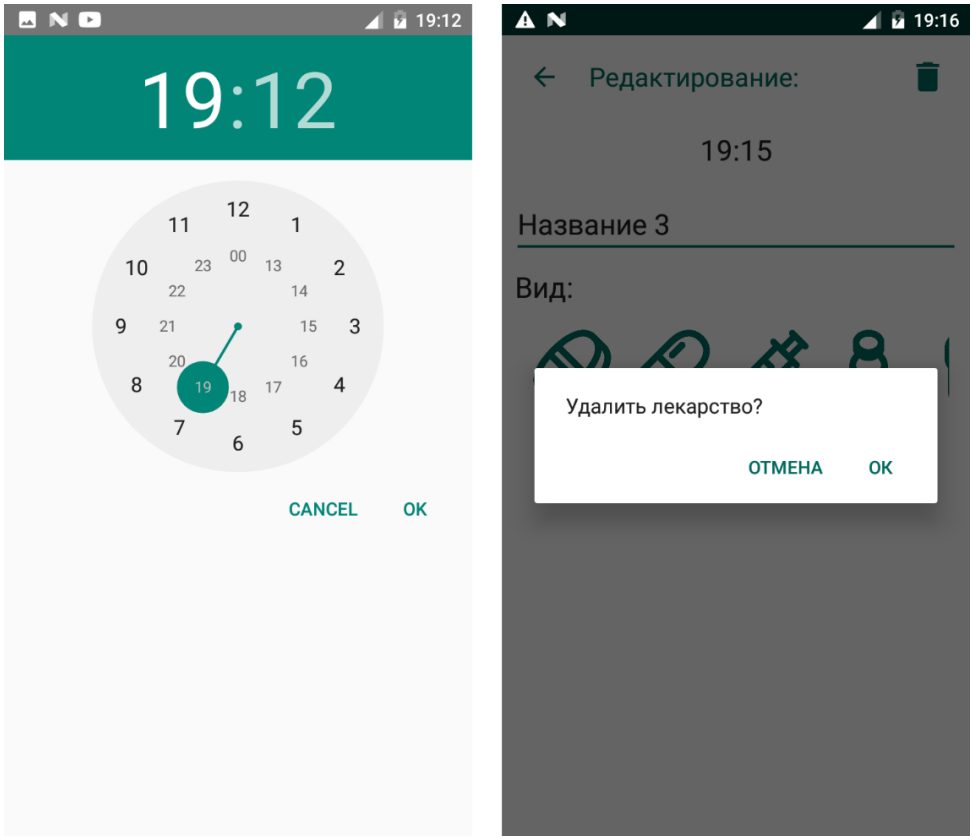

Рисунок 22 – Диалоговые экраны

Оптимизация приложения, однако, включает в себя и интеграцию возможностей Android Accessibility Suite, включающую в себя, например, функцию чтения с экрана, что требует дополнительных настроек как в .xml так и в .java файлах приложения [19].

TalkBack позволяет людям со зрительными нарушениями осуществлять навигацию в приложении посредством касания. Одна из функций, например, позволяет прослушивать описание всего, к чему пользователь прикасается на экране. Для должного функционирования этой возможности, необходимо предоставлять описательные маркировки для необходимого контента на экране. Так для полей редактирования текста существует атрибут «android:hint», который как предоставляет подсказку в поле ввода, так и передает содержимое этого атрибута функциям TalkBack. Для элементов-изображений и кнопок с изображениями вместо текста, следует использовать атрибут «android:contentDescription».

Так на примере рисунка на главном экране в xml-описание этого элемента с помощью функции android:contentDescription, следует включить ссылку на ресурс программы содержащий текст «Радостный котя».

Еще одним важным атрибутом для интерактивных элементов-изображений является android:focusable, он необходим при использовании пользователем клавиатуры. В данном случае, так как картинка имеет исключительно декоративно-мотивационную цель, этот атрибут можно упустить.

В случае динамического изменения содержания элементов, предусмотрен также программный подход к изменению атрибута описания элемента с помощью метода setContentDescription. Так в зависимости от выбранного пользователем внешнего вида лекарства, MedAdapter может присваивать этому элементу соответствующее описание, например при выборе первого варианта изображения, описание должно гласить «плоская таблетка с одним вырезом в центре».

При чтении содержимого экрана с помощью специальных возможностей можно группировать контент в группы, для удобства восприятия. Так например в музыкальном приложении лейбл «Название» и текстовое поле с названием песни можно объединить в одну группу, при наведении на которую TalkBack или Select to Speak будет озвучивать их парой. В случае данного приложения логичнее будет группировка всего содержимого экземпляра списка, содержащего задание, детали задания и время выполнения. Для этого необходимо добавить атрибут android:focusable со значение true в шапку всего содержащего их макета, использующегося для заполнения RecyclerView.

Изменения динамических полей в приложении так же озвучивается с помощью TalkBack. Например, одним из таких элементов на главном экране является флажок напротив заданий, сигнализирующий о его выполнении. Так как это стандартный элемент библиотеки Android, то при активации этого динамического элемента TalkBack будет оповещать пользователя о смене его состояния стандартным сообщением.

Для активации этой функции необходимо добавить атрибут android:accessibilityLiveRegion и установить его значение как polite, что означает, что программа не будет прерывать текущие сообщения, для оповещения о смене состояния элемента.

Material Design предполагает, что любой интерактивный элемент на экране должен иметь поле размером минимум 48dp, вне зависимости от размеров самого элемента. Для достижения такого результата можно не только вручную настраивать атрибут android:padding в соответствии с размерами элемента, но и использовать атрибуты, устанавливающие минимальный размер элемента, которые автоматически подгоняют этот атрибут до необходимого размера, например android:minWidth.

Для проверки приложения на доступность Google предоставляет Accessibility Scanner. Эта утилита сканирует приложение и предлагает улучшения, связанные с контрастностью, оформлением интерактивных элементов и описанием контента.

Для этого необходимо установить и приложение, и сканер на смартфон/эмулятор и активировать сканер в настройках системы.

После выдачи ему необходимых разрешений сканер сможет обводить зоны, которые он считает проблемными и указывать классы, в которых находится соответсвующий код.

В случае элемента 1, сканер предлагает дать описание списку-контейнеру, содержащему элементы-задания. Но так как приложение подразумевает в себе наличие списка и каждый отдельный элемент имеет собственное описание, описание контейнера списка можно считать излишним.

Для проверки элемента 2 проведем поиск по совпадению строки во всем коде приложения. Результат показывает, что совпадения найдены в двух файлах: mobile_navigation.xml и bottom_nav_menu.xml. Второй файл является макетом нижнего меню приложения, содержащим настройки и текст интерфейса нижнего меню приложения, а mobile_navigation.xml – это файл-граф, содержащий всю необходимую информацию о навигации и переходах в одном файле и обеспечивающий корректное поведение навигации, а так же настройку анимаций переходов.

Так как это совпадение в данной реализации навигационного меню необходимо для его функционирования, то редактирование этих элементов тоже не требуется. Результаты сканирования приложения, приведены на рисунке 23.

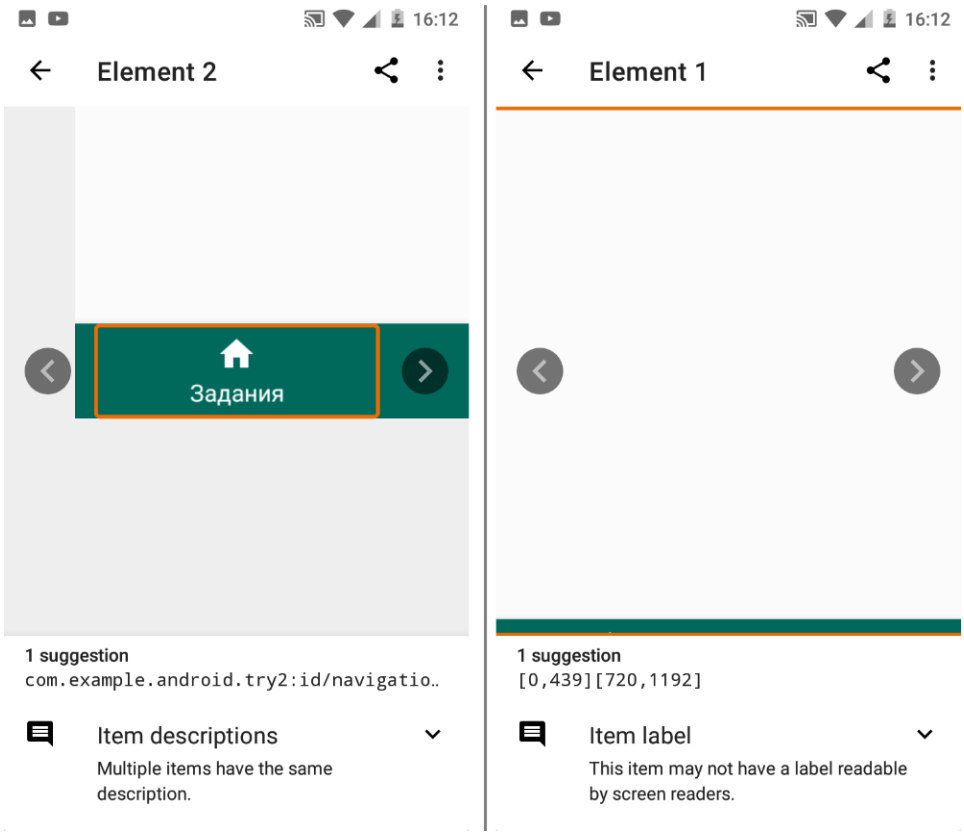

Рисунок 23 – Предложения по оптимизации Accessibility Scanner

Возможными дальнейшими путями развития приложения являются реализация перевода на другие языки, для чего в любых полях, содержащих текст, согласно рекомендациям системы использовались ссылки на ресурс, содержащий строку, а не строго указанный текст. Архитектура приложения так же предполагает простую интеграцию базы данных в облаке в, случае необходимости реализации двухкомпонентной базы данных, благодаря разделению логики представления и хранения данных, обеспечивающих оптимальное решение для масштабирования приложения и подключения дополнительных таблиц или баз данных.

Выводы по разделу три

Таким образом, созданное приложение для планирования деятельности предоставляет необходимый функционал для составления списка ежедневных заданий и планирования приема лекарств, предоставляет систему оповещений и отслеживания прогресса выполнения заданий. Тестирование приложения с использованием предоставляемых платформой средств выявило, что программа оптимизирована для использования функций Accessibility Suite, предоставляющих набор ассистивных технологий для пользователей платформы, и отвечает требованиям доступности цифрового контента.

ЗАКЛЮЧЕНИЕ

Современные мобильные платформы предоставляют пользователям множество видов взаимодействия с системой и приложениями. Использование этих методов может помочь людям с разными видами ограничений как пользоваться различными устройствами, так и иметь доступ к предоставляемой ими цифровой информации. Эти устройства и информация в свою очередь могут помочь каждому в решении разнообразных задач, в том числе задачи планирования деятельности.

В процессе подготовки дипломной работы были проанализированы теоретические сведения о разработке мобильных приложений и их функционировании, изучены методические рекомендации WCAG к обеспечению доступности цифровой информации в целом и конкретные рекомендации платформы Android для создания доступного интерфейса и интеграции в приложение специальных возможностей, поддерживаемых системой.

Проанализированы отечественные и зарубежные аналоги создаваемого приложения и технологии создания мобильных приложений. Было решено создать приложение на основе операционной системы Android, с помощью объектно-ориентированного языка Java, системы управления базами данных SQLite и интегрированной среды разработки Android Studio.

Изучено применение основных архитектурных паттернов создания мобильного приложения и архитектурные компоненты ОС Android, способствующие реализации этих архитектур. Разработана локальная база данных для приложения и система классов, приведенная в работе виде UML-диаграмм. Созданы и оптимизированны, согласно требованиям платформы ОС Android и стандартам WCAG, макеты пользовательского интерфейса.

В результате работы было создано и протестированно, с помощью предлагаемых платформой инструментов, приложение, призванное помочь планированию деятельности людей с умеренными расстройствами памяти. Также была проведена оптимизация кода программы, для создания приложения, отвечающего поставленным требованиям к доступности и интеграции специальных возможностей системы.

Задачи выпускной квалификационной работы решены. Поставленные цели достигнуты.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1 Блинов, И. Н. Java. Методы программирования: уч-мет. пособие / И. Н. Блинов, В. С. Романчик. – Минск: Четыре четверти, 2013. – 896 с.

2 Блох, Д. Java. Эффективное программирование / Д. Блох. – М: Вильямс, 2019. – 464 с.

3 Вербовецкий, А. А. Основы проектирования баз данных / А. А. Вербовецкий. – М.: Радио и связь, 2000. – 85 с.

4 Гамма, Э. Приемы объектно-ориентированного проектирования: Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Д. Влиссидес; пер. с англ. А. Слинкина. – СПб. и др.: Питер, 2008. – 366 с.

5 ГОСТ Р ИСО/МЭК 9075-93: Информационная технология. Язык баз данных SQL с расширением целостности: введ. в действие с 01.07.94 / Техн. ком. по стандартизации ТК 22 «Информ. Технология». – М.: Госстандарт России, 1993. – 106 с.

6 ГОСТ Р 52872-2019: Интернет-ресурсы и другая информация, представленная в электронно-цифровой форме. Приложения для стационарных и мобильных устройств, иные пользовательские интерфейсы. Требования доступности для людей с инвалидностью и других лиц с ограничениями жизнедеятельности (с Поправкой): введ. в действие с 01.04.20 [Текст] / Техн. ком. по стандартизации ТК 22 «Информ. Технология». – М.: Госстандарт России, 2020. – 37 с.

7 Гриффитс, Д. Head First. Программирование для Android. 2-е изд. / Дэвид Гриффитс, Дон Гриффитс. – СПб.: Питер, 2018. – 912 с.

8 Иванова, Г.С. Технология программирования: Учебник для вузов / Г.С. Иванова. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 320 с.

9 Леоненков, А. В. Самоучитель UML. 2-е изд., перераб. и доп. / А. В. Леоненков. – СПб: БХВ-Петерберг, 2004. – 432 с.

10 Молинаро, Э. SQL. Сборник рецептов. / Э. Молинаро. – СПб: Символ Плюс, 2009. – 672 с.

11 Объекты Intent и фильтры объектов Intent [Электронный ресурс]. – URL: https://developer.android.com/guide/components/intents-filters

12 Пайлон, Д. UML 2 для программистов. / Д. Пайлон, Н. Питмен – СПб: Питер, 2012. – 240 с.

13 Соколова, В. В. Вычислительная техника и информационные технологии. Разработка мобильных приложений: учебное пособие для прикладного бакалавриата / В. В. Соколова. – Москва: Юрайт, 2019. — 175 с.

14 Студеникин, Н. В. Влияние цифровых технологий на социальные услуги: мировой опыт и перспективы в России / Н. В. Студеникин. // Известия Тульского государственного университета. Гуманитарные науки. – 2018. – №1 – С. 32–36.

15 Сьерра, К. Head First Java, Изучаем Java / К. Сьерра, Б. Бэйтс / М: Эксмо 2012. – 720 с.

16 Шилдт, Г. Java 8. Руководство для начинающих / Г. Шилдт. – М.: Вильямс, 2015. – 720 с.
17 Android Jetpack Components. Architecture. 2020 [Электронный ресурс]. – URL: https://developer.android.com/jetpack#architecture-components
18 Android Runtime Improvements 2020 [Электронный ресурс]. – URL: https://medium.com/mindorks/android-runtime-improvements-e69bf7c1d10c
19 Basic Android Accessibility 2020 [Электронный ресурс]. – URL: https://codelabs.developers.google.com/codelabs/basic-android-accessibility
20 Cheng, Y. Advanced Android App Architecture / Y. Cheng, A. O. Dominguez. – Virginia: Razeware LLC, 2019. – 273 с.
21 Create a List with RecyclerView. 2020 [Электронный ресурс]. – URL: https://developer.android.com/guide/topics/ui/layout/recyclerview
22 Deepti S. R. Bridging the disability divide through digital technologies / S. R. Deepti. – Washington: World Development Report: Digital Dividends, 2016 – 35 с.
23 Fowler, M. UML Distilled: A Brief Guide to the Standard Object Modeling Language / M. Fowler. – Massachusetts: Addison-Wesley Professional, 2018 – 191 с.
24 Gargenta, M. Learning Android / M. Gargenta, M. Nakamura. – CA.: O'Reilly Media, 2014. – 268 с.
25 Global AgeWatch Insights 2018: Report, summary and country profiles [Электронный ресурс]. – URL: http://globalagewatch.org/reports/global-agewatch-insights-2018-report-summary-and-country-profiles/
26 Google developer training: Working with Architecture Components [Электронный ресурс]. – URL: https://google-developer-training.github.io/android-developer-advanced-course-practicals/unit-6-working-with-architecture-components/lesson-14-room,-livedata,-viewmodel/14-1-a-room-livedata-viewmodel/14-1-a-room-livedata-viewmodel.html
27 Klimova, B. Mobile Phone Apps in the Management and Assessment of Mild Cognitive Impairment and/or Mild-to-Moderate Dementia/ B. Klimova // Frontiers in Human Neuroscience. – 2017. – №11. – С. 461.
28 Material Design Accessibility Guidelines 2020 [Электронный ресурс]. – URL: https://material.io/design/usability/accessibility.html
29 McLaughlin, B. Head First Object-Oriented Analysis and Design / B. McLaughlin, G. Pollice, D. West – CA.: O'Reilly Media, 2007. – 600 с.
30 Mehmood, Y. Mobile-Health Applications for the Efficient Delivery of Health Care Facility to People with Dementia (PwD) and Support to Their Carers: A Survey / Y. Mehmood // BioMed Research International. – 2019. – №2019. – С. 1-26.
31 Meier, R. Professional Android / R. Meier, I. Lake. – Birmingham: Wrox Press, 2018. – 434 с.
32 Mobile App Architecture [Электронный ресурс]. – URL: https://dzone.com/articles/everything-you-need-to-know-about-mobile-app-archi
33 Mobile Operating System Market Share Worldwide 2020 [Электронный ресурс]. – URL: https://gs.statcounter.com/os-market-share/mobile/worldwide

34 Owens, M. The Definitive Guide to SQLite / M. Owens – NY: Apress, 2006. – 440 с.

35 Page-Jones, M. Fundamentals of Object-oriented Design in UML / – Massachusetts: Addison-Wesley Professional, 1999 – 480 с.

36 Phillips, B. Android Programming 4th Ed / B. Phillips, C. Stewart, K. Marsicano. – Georgia: Big Nerd Ranch Guide, 2019. – 624 с.

37 Schedule repeating alarms. 2020 [Электронный ресурс]. – URL: https://developer.android.com/training/scheduling/alarms

38 TIOBE Index for April 2020 [Электронный ресурс]. – URL: https://www.tiobe.com/tiobe-index/

39 Web Content Accessibility Guidlelines 2.1 2020 [Электронный ресурс]. – URL: https://www.w3.org/WAI/standards-guidelines/wcag/

40 What You Need to Know About Mobile Health Apps in 2018 [Электронный ресурс]. – URL: https://www.healthworkscollective.com/what-you-need-to-know-about-mobile-health-apps

ПРИЛОЖЕНИЕ А
Фрагменты кода приложения

Методы запросов к базе данных MedData.

```
@Query("SELECT * FROM MED_TABLE ORDER BY state ASC, time(time) ASC")
LiveData<List<MedData>> getAllMeds();
@Query("SELECT * FROM MED_TABLE WHERE state=2")
LiveData<List<MedData>> getInactiveMeds();
@Query("SELECT * FROM MED_TABLE WHERE ID = :id ")
MedData findMedById(int id);
@Query("UPDATE med_table SET state = 1")
void changestate();
@Query("SELECT COUNT(*) FROM MED_TABLE WHERE state=2")
int getInactiveCount();
```

Настройка пробуждения устройства для запуска оповещений с помощью сервиса AlarmManager.

```
AlarmManager alarmManager = (AlarmManager) context
   .getSystemService(Context.ALARM_SERVICE);
Intent i = new Intent(context, AlarmReceiver.class);
i.putExtra(EXTRA_TEXT, title);
i.putExtra(EXTRA_ID, dailyId);
Calendar timeNow = Calendar.getInstance();
if(when.before(timeNow)){
   when.add(Calendar.DATE, 1);
}
PendingIntent pi = PendingIntent.getBroadcast(context, dailyId, i,
       PendingIntent.FLAG_UPDATE_CURRENT);
alarmManager.setRepeating(AlarmManager.RTC_WAKEUP,
       when.getTimeInMillis(), AlarmManager.INTERVAL_DAY, pi);
```

Далее приведены методы, ответственные за обработку операции редактирования записи пользователем в приложении, в классах разных архитектурных слоев программы.

Класс экрана редактирования записи EditDailyActivity собирает информацию и посылает ее во фрагмент.

```
private void updateDaily() {
   //сбор информации
   String title =  editTextTitle.getText().toString();
   String time = editTextTime.getText().toString();
   String details = editTextDetails.getText().toString();
   int state = getIntent().getIntExtra(EXTRA_STATE, 1);
   if (title.trim().isEmpty() || time.trim().isEmpty() || details.trim()
         .isEmpty()) {
      Toast.makeText(this, "Пожалуйста, введите задание, " +
```

```
            "время и детали", Toast.LENGTH_SHORT).show();
        return;
    }
    //передача в DailyFragment через intent
    Intent data = new Intent();
    data.putExtra(EXTRA_TEXT, title);
    data.putExtra(EXTRA_TIME, time);
    data.putExtra(EXTRA_DETAILS, details);
    data.putExtra(EXTRA_STATE, state);
    int id = getIntent().getIntExtra(EXTRA_ID, -1);
    if (id != -1) {
        data.putExtra(EXTRA_ID, id);
    }
    setResult(RESULT_OK, data);
    finish();
}
```

Фрагмент DailyFragment получает информацию, создает новый объект таблицы и присваивает ему уже имеющийся идентификатор. После этого передает запрос на обновление в DailyViewModel.

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == EDIT_DAILY_REQUEST && resultCode == RESULT_OK) {
        int id = data.getIntExtra(EditDailyActivity.EXTRA_ID, -1);
        String title = data.getStringExtra(EditDailyActivity.EXTRA_TEXT);
        String time = data.getStringExtra(EditDailyActivity .EXTRA_TIME);
        String details = data.getStringExtra(EditDailyActivity.EXTRA_DETAILS);
        int state = data.getIntExtra(EditDailyActivity.EXTRA_STATE, 1);
        DailyData dailyData = new DailyData(title, details, time,state, 2);
        dailyData.setId(id);
        dailyViewModel.update(dailyData);
        Calendar calendar = timeConverter(time);
        ReminderManager.setReminder(getActivity().getApplicationContext(),
            id, title, calendar, 2);
```

Класс DailyViewModel в свою очередь работает с экземпляром репозитория данных.

```
public void update(DailyData dailyData) {
    dailyRepository.update(dailyData);
}
```

В классе DailyRepository в свою очередь определен метод обновления таблицы, создающий асинхронное задание для выполнения операции.

```
public void update(DailyData dailyData) {
    dailyRepository.update(dailyData);
}
private static class UpdateDailyAsyncTask extends AsyncTask<DailyData, Void, Void> {
    private DailyDao dailyDao;
    private UpdateDailyAsyncTask(DailyDao dailyDao) {
        this.dailyDao = dailyDao;

    @Override
    protected Void doInBackground(DailyData... dailyData) {
        dailyDao.update(dailyData[0]);
        return null;
}
```

В свою очередь в интерфейсе DailyDAO метод обновления таблицы генерируется библиотекой Room автоматически с помощью использования аннотации @Update.

```
@Update
void update(DailyData dailyData);
```