**МИНОБРНАУКИ РОССИИ**

**Федеральное государственное автономное   
образовательное учреждение высшего образования**

**«Южный федеральный университет»**

**Институт высоких технологий и пьезотехники Кафедра прикладной информатики и инноватики**

**Яминов Дамир Маратович**

**Приложение** **учета лекарственных средств в аптечке пользователя**

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

**по направлению 09.03.03 – Прикладная информатика**

**Научный руководитель – старший преподаватель**

**Яценко Дмитрий Владимирович**

**Ростов-на-Дону – 2022**

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ**

**ИНСТИТУТ ВЫСОКИХ ТЕХНОЛОГИЙ И ПЬЕЗОТЕХНИКИ**

**Кафедра прикладной информатики и инноватики**

**З А Д А Н И Е**

**на выпускную квалификационную работу**

**Студент** гр. 4–7 Яминов Д.М.

1. **Тема:** приложение учета лекарственных средств в аптечке пользователя
2. **Срок сдачи законченной работы:** 06.06.2022
3. **Исходные данные:**

Государственные стандарты:

1. ГОСТ 7.1–2003. Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание. Общие требования и правила составления;
2. ГОСТ 7.82–2001 Система стандартов по информации, библиотечному и издательскому делу. Библиографическая запись. Библиографическое описание электронных ресурсов. Общие требования и правила составления;
3. ГОСТ 19.001–77. Единая система программной документации (ЕСПД). Общие положения;
4. ГОСТ 19.781–90. Обеспечение систем обработки информации программное. Термины и определения;
5. ГОСТ 19.701–90 (ИСО 5807–85). Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения;
6. ГОСТ 19.103–77. Единая система программной документации. Обозначения программ и программных документов;
7. ГОСТ 19.101–77. Единая система программной документации. Виды программ и программных документов;
8. ГОСТ 19.106–78. Единая система программной документации. Требования к программным документам, выполненным печатным способом;
9. ГОСТ 19.104–78. Единая система программной документации. Основные надписи;
10. ГОСТ 19.301–79. Единая система программной документации. Программа и методика испытаний. Требования к содержанию и оформлению;
11. ГОСТ 19.402–78. Единая система программной документации. Описание программы;
12. ГОСТ 19.401–78. Единая система программной документации. Текст программы. Требования к содержанию и оформлению;
13. ГОСТ 19.504–79. Единая система программной документации. Руководство программиста. Требования к содержанию и оформлению;
14. ГОСТ 19.505–79. Единая система программной документации. Руководство оператора. Требования к содержанию и оформлению;
15. ГОСТ 19.202–78. Единая система программной документации. Спецификация. Требования к содержанию и оформлению.

Основное требование – разработать мобильное приложение, реализующее функции поиска и учета лекарственных средств, с возможностью добавления уведомлений и заметок

1. **Перечень вопросов, подлежащих разработке:**
   1. Анализ предметной области.
   2. Обоснование технических решений.
   3. Проектирование элементов системы.
   4. Реализация компонентов системы.
   5. Разработка методики тестирования.
   6. Тестирование системы.
2. **Перечень графического материала:**

Подготовка графических материалов для презентации работы.

1. **Консультанты по работе:**
2. **Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_** Яценко Д. В.

*Подпись ФИО*

1. **Задание принято к исполнению**

\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_

*Дата Подпись студента*

АННОТАЦИЯ

Результатом выпускной квалификационной работы является мобильное приложение под ОС Android – “ приложение учета лекарственных средств в аптечке пользователя ”.

Данное приложение позволяет искать лекарственные препараты по названию и добавлять их в локальное хранилище телефона, также реализована возможность добавлять напоминания, которые в назначенное время выведут pop-up уведомление, в дополнение есть возможность сохранять заметки о приеме лекарств.

В ходе выполнения работы были выбраны методология разработки и язык программирования, проведен анализ существующих решений, их преимуществ и недостатков. На основе анализа были составлены требования для данного проекта и, учитывая данные требования, разработаны UML диаграммы классов и вариантов использования и, используя их в качестве отправной точки, был спроектирован макет пользовательского интерфейса и выбраны средства для разработки и технологии.

В результате было создано мобильное приложение, отвечающее заявленным требованиям.

Ключевые слова: Java, Android studio, SQLite, Figma, Jsoup, Room, LiveData, Junit, Apptim

# Annotation

The result of the final qualifying work is a mobile application for Android OS – " application for accounting of medicines in the user's first aid kit ”.

This application allows you to search for medicines by name and add them to the phone's local storage, it is also possible to add reminders that will pop-up a notification at the appointed time, in addition, it is possible to save notes about taking medications.

In the course of the work, a development methodology and a programming language were selected, an analysis of existing solutions, their advantages and disadvantages were carried out. Based on the analysis, the requirements for this project were compiled and, considering these requirements, UML diagrams of classes and use cases were developed and using them as a starting point, a user interface layout was designed and were selected development tools and technologies.

As a result, a mobile application was created that meets the stated requirements.

Keywords: Java, Android studio, SQLite, Figma, Jsoup, Room, LiveData, Junit, Apptim

Содержание

[Введение 9](#_Toc97654418)

[1. Постановка задачи 10](#_Toc97654419)

[1.1Словесная постановка задачи 10](#_Toc97654420)

[1.2Формальная постановка задачи 10](#_Toc97654421)

[1.3 Анализ существующих решений 10](#_Toc97654422)

[1.4 Функциональные свойства приложения 17](#_Toc97654423)

[1.5 Требования к системе 18](#_Toc97654424)

[1.6 Выводы и результаты по главе 1 18](#_Toc97654425)

[2. Проектирование программного продукта 19](#_Toc97654426)

[2.1 Декомпозиция задачи 19](#_Toc97654427)

[2.2 Методология проектирования 19](#_Toc97654428)

[2.3 Обоснование используемых средств разработки 20](#_Toc97654429)

[2.3.1 Выбор языка программирования 20](#_Toc97654430)

[2.3.2 Выбор интегрированной среды разработки 21](#_Toc97654431)

[2.3.3 Выбор СУБД 22](#_Toc97654432)

[2.3.4 Выбор дополнительных библиотек 23](#_Toc97654433)

[2.4 UML-моделирование 29](#_Toc97654434)

[2.4.1 Диаграмма вариантов использования 30](#_Toc97654435)

[2.4.2 Диаграмма классов 35](#_Toc97654436)

[2.5 Проектирование пользовательского интерфейса 36](#_Toc97654437)

[2.6 Выводы и результаты по главе 2 36](#_Toc97654438)

[3. Реализация приложения 38](#_Toc97654439)

[3.1 Реализация базы данных 38](#_Toc97654440)

[3.2 Реализация основных функций 39](#_Toc97654441)

[3.3 Выводы и результаты по главе 3 42](#_Toc97654442)

[4. Тестирование 43](#_Toc97654443)

[4.1 Модульное тестирование 43](#_Toc97654444)

[4.2 Тестирование производительности 45](#_Toc97654445)

[4.3 Выводы и результаты по главе 4 47](#_Toc97654446)

[5. Безопасность жизнедеятельности 48](#_Toc97654447)

[5.1 Разработка комплекса мер защиты и профилактики 48](#_Toc97654448)

[5.2 Выводы и результаты по главе 5 50](#_Toc97654449)

[6. Экономическое обоснование 51](#_Toc97654450)

[6.1 Определение стоимости 51](#_Toc97654451)

[6.2 Вычисление окупаемости 52](#_Toc97654452)

[6.3 Расчет окупаемости через расширенную версию 52](#_Toc97654453)

[6.4 Расчет окупаемости через рекламные баннеры 53](#_Toc97654454)

[6.5 Выводы и результаты по главе 6 55](#_Toc97654455)

[Заключение 56](#_Toc97654456)

[Список используемых источников 57](#_Toc97654457)

# Введение

Многие медицинские организации рекомендуют, чтобы в каждом доме находилась аптечка, в которой не менее 10 различных лекарственных наименований [1]. А если в семье есть дети или люди с хроническими заболеваниями, то количество лекарств неуклонно возрастает. Запоминать все названия, сроки годности и хотя бы краткие описания каждого лекарства становится очень сложно. В добавок к этому лекарства в основном хранятся в темных труднодоступных местах, это обусловлено тем, что на всех препаратах в условиях хранения прописаны данные ограничения, а значит проверка срока годности бывает довольно проблематичной, и в большинстве случаев о лекарствах вспоминают только тогда, когда они понадобятся [2]. Из этого следует, что к этому времени у лекарств может истечь срок годности.

Но мы живем в цифровом мире, количество электроники возрастает, а сама техника становится дешевле, мощнее и удобнее. Именно поэтому, сейчас практически невозможно найти человека, у которого нет смартфона [3]. А при таких условиях можно очень просто перенести на вычислительное устройство рутинные задачи, например: запоминать всю необходимую информацию о лекарствах, их сроках годности, времени приема и уведомлять пользователя при необходимости.

Объектом данной выпускной квалификационной работы является приложение для учета лекарств в аптечке пользователя

# 1. Постановка задачи

## Словесная постановка задачи

Объект выпускной квалификационной работы — мобильное приложение учета лекарственных средств в аптечке пользователя. Приложение будет предоставлять пользователю возможность добавления препарата по его названию. В интернете будет произведен поиск данного лекарства, для того чтобы иметь возможность вместе с названием загрузить инструкцию к применению в память телефона.

Вместе с добавлением лекарственного средства в локальное хранилище будет загружен и срок годности, при истечении которого пользователь получит push-уведомление. Также пользователь может создавать напоминания о приеме лекарств, которые он также будет получать через push-уведомления.

## Формальная постановка задачи

Учитывая предметную область и словесную постановку задачи, нужно создать мобильное приложение, реализующее возможность поиска лекарства и инструкции к применению в интернете, сохранения его на телефоне пользователя, а также возможность указания срока годности и создания напоминаний. Для этого необходимы следующие объекты:

* Препарат — лекарственное средство. Пользователь вводит строку (название препарата), а дальнейшее описание находится с помощью интернета, после чего пользователь может добавить срок годности
* Напоминание — уведомление, которое создает пользователь, чтобы не забыть принять препарат в нужное время

## 1.3 Анализ существующих решений

Данное приложение не предоставляет каких-либо уникальных решений, на рынке уже есть похожие проекты.

Например, «Домашняя Аптечка: учет медикаментов» [4], которая представлена на рисунке 1. Оно позволяет записывать название таблетки, указывать его категорию и срок годности, об окончании которого приложение оповестит пользователя. Но в нем нет краткой информации об лекарстве и, чтобы прочитать, например, показания к применению, нужно будет искать его в инструкции к препарату, что может быть неудобно. Также в отзывах пользователи отмечают неудобную работу с уведомлениями.

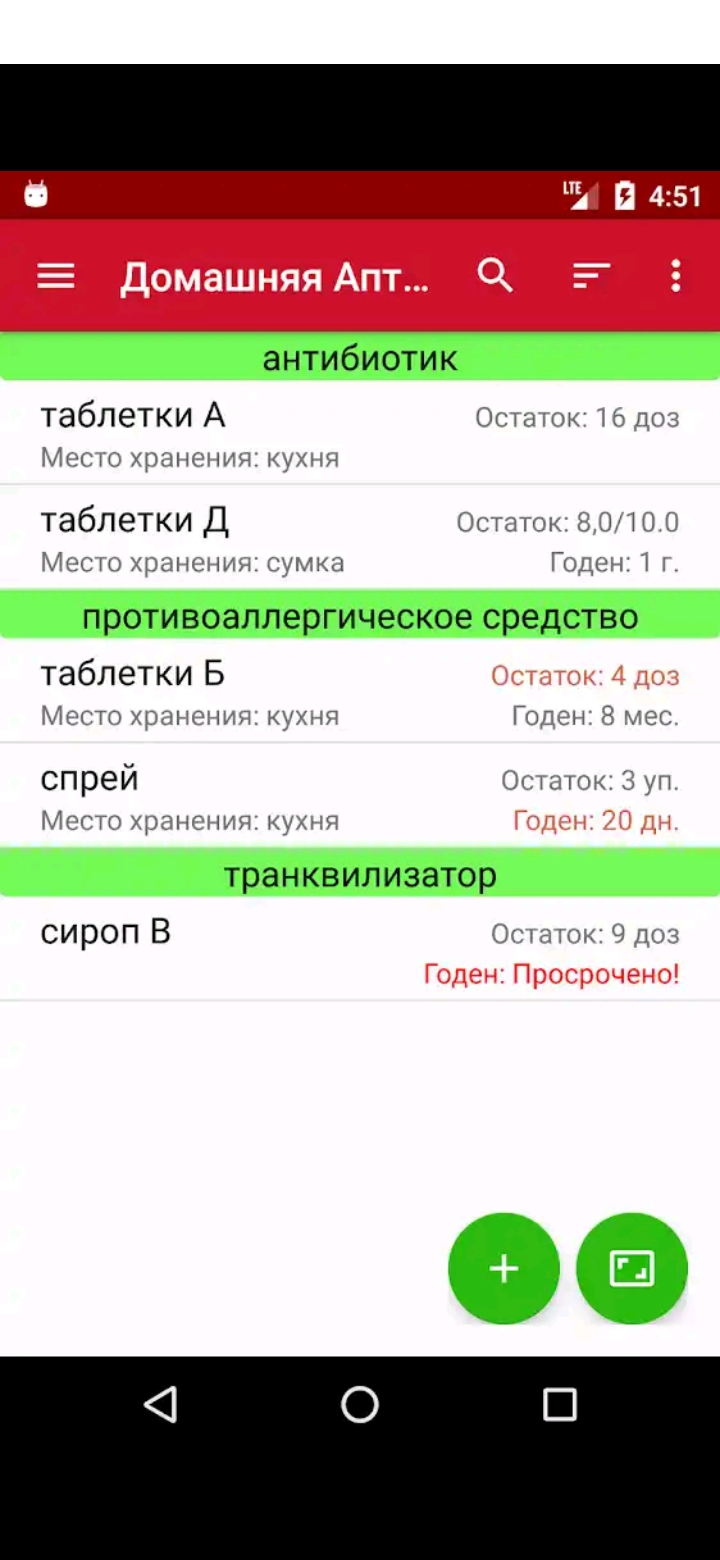


Рисунок 1 — Внешний вид мобильного приложения «Домашняя Аптечка: учет медикаментов»

Следующим аналогом представлен на рисунке 2 — приложение «Аптечка» [5]. Его функции предоставляют возможность поиска инструкции к лекарствам, а также добавления лекарства в избранное. Минусами, которые отмечают пользователи, являются невозможность удалить лекарства из избранного, а также скудный список доступных лекарств, в котором нет многих современных препаратов.

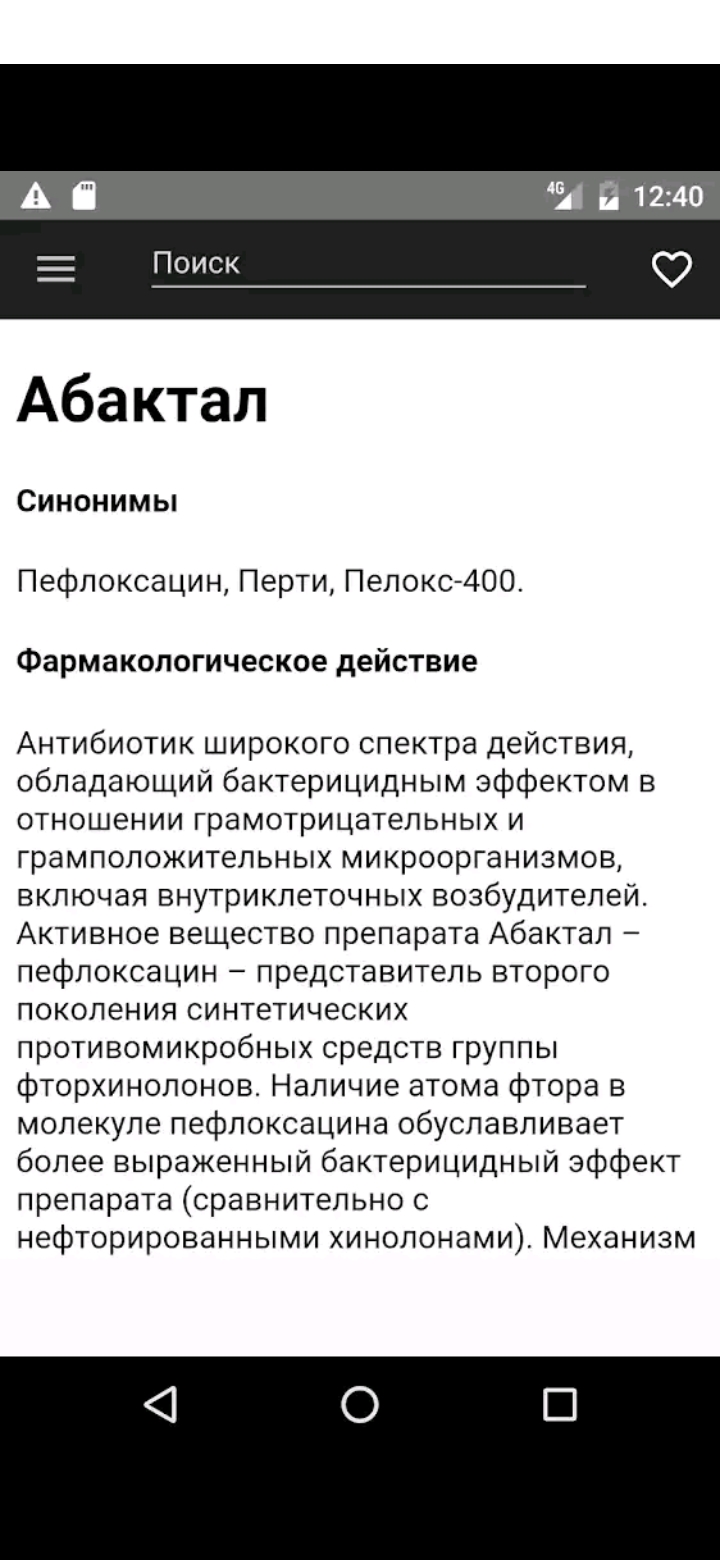


Рисунок 2 — Внешний вид мобильного приложения «Аптечка»

Еще один пример мобильного приложения с похожими функциями — это «Моя аптечка + Аналоги лекарств» [6], которое представлено на рисунке 3. Функциональными особенностями являются:

* возможность добавления лекарств
* учитывание срока годности
* расписание приема лекарств.

Но, как отмечают пользователи в отзывах, приложение требует обязательную регистрацию, невозможность доступа к приложению без подключения интернета, невозможность удаления лекарства из добавленных в аптечку.

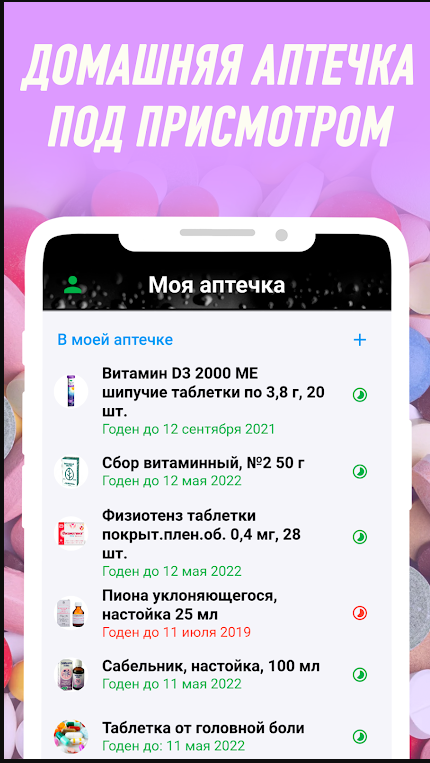


Рисунок 3 — Внешний вид мобильного приложения «Моя аптечка + Аналоги лекарств»

Таблица 1 — Сводная таблица для аналогов

| Названия  Функции | Домашняя Аптечка: учет медикаментов | Аптечка | Моя аптечка + Аналоги лекарств |
| --- | --- | --- | --- |
| Возможность добавления лекарств в локальное хранилище | + | + | + |
| Возможность добавления инструкции к применению | - | + | - |
| Учет срока годности | + | - | + |
| Возможность создания уведомлений | Платная функция (319 руб.) | - | + |
| Возможность удаления лекарств из локального хранилища | + | - | + |
| Требования к постоянному подключению к интернету | - | - | + |
| Оффлайн доступ к локальному хранилищу | + | + | - |
| Размер приложения | 4,22МБ | 22,98МБ | 9,68МБ |
| Количество скачиваний | 10тыс.+ | 1млн+ | 500+ |
| Платный контент | + | - | - |
| рейтинг | 4,4 | 4,4 | 2,4 |

Таким образом, во всех примерах нет достаточного функционала для пользователя, в то время как разрабатываемое приложение объединяет в себе функции удобного лекарственного справочника, а также является хранилищем лекарств самого пользователя

## 1.4 Функциональные свойства приложения

Разрабатываемое мобильное приложение должно быть простым как в интерфейсном, так и в функциональном плане, но в то же время содержать все необходимые функции.

Такие как:

* Поиск лекарства по названию
* Сохранение и удаление препарата в локальном хранилище с его описанием
* Возможность добавления срока годности и учет данного параметра
* Возможность создания, изменения и удаления уведомлений о приеме лекарства

## 1.5 Требования к системе

Для корректной работы приложения система должна удовлетворять следующим требованиям

* Операционная система: Android с версией sdk не ниже 23, так как 94,1% существующих устройств поддерживают эту версию sdk [7], а средства разработки работают более стабильно, начиная с данной версии
* Подключение к сети Интернет. Только на время поиска и добавления лекарства в локальное хранилище. Все остальные функции будут работать без доступа в интернет

## 1.6 Выводы и результаты по главе 1

В этой главе были описаны словесная и формальная постановка задачи, был проведен сравнительный анализ аналогов, которые присутствуют на рынке. Составлены функциональные требования к приложению, а также требования к системе, на которой будет работать приложение.

# 2. Проектирование программного продукта

## 2.1 Декомпозиция задачи

Данную задачу можно разбить на несколько подзадач:

1) выбор методологии проектирования

2) выбор средств реализации и обоснование их необходимости

3) разработка UML-диаграмм, таких как диаграмма вариантов использования, диаграмма классов

4) разработка пользовательского интерфейса

## 2.2 Методология проектирования

Для разработки данного проекта был использован объектно-ориентированный метод анализа и проектирования [8].

Объектно-ориентированный подход подразумевает, что перед написанием непосредственно кода нужно завершить этапы анализа и проектирования. Учитывая это можно выделить 3 ключевых понятия:

* OOA (object oriented analysis) - объектно-ориентированный анализ, при котором требования рассматриваются с точки зрения классов и объектов. Класс представляет собой шаблон для создания объектов, предоставляя начальные значений для состояний и имплементацию поведения функций и методов
* OOD (object oriented design) - объектно-ориентированное проектирование, представляющее собой декомпозицию задачи, а также различные приемы представления логической модели системы
* OOP (object oriented programming) - объектно-ориентированное программирование, которая основана на представлении программы в виде классов, которые образуют иерархию с помощью наследования, и объектов, являющиеся экземплярами классов

Концепциями ООАП являются:

* Объекты/Классы. Ассоциация структуры данных с методами или функциями, которые работают с данными
* Инкапсуляция. Принцип защиты некоторый компонентов объекта от внешних сущностей
* Наследование. Возможность класса расширять или переопределять функциональность другого класса
* Абстракция. Характеристика сущности, отличающая ее от других сущностей. Применяется для определения классов и объектов
* Полиморфизм. Позволяет заменить объект на подобъекты, а также дает возможность объекту-переменной содержать не только этот объект, но все его подобъекты

## 2.3 Обоснование используемых средств разработки

### 2.3.1 Выбор языка программирования

Для разработки Android приложения существуют два основных языка программирования: Kotlin и Java. Преимущества Kotlin в том, что он объединяет в себе функциональный и объектно-ориентированный подход, также он не требует определять типы данных для каждой переменной. И в дополнении к этому он совместим с кодом, написанным на Java.

Но главный недостаток этого языка – это недостаточная распространенность, из-за чего бывает сложно найти гайды или помощь с решением конкретной проблемы.

Поэтому в качестве языка программирования был выбран Java. Этот объектно-ориентированный язык был разработан для бизнес-приложений общего пользования.

Некоторыми из преимуществ, которые сделали Java популярной, являются функции безопасности и тот факт, что он архитектурно нейтральный. Это означает, что написанная программа будет выполняться на любом устройстве, вне зависимости от операционной системы, потому что он не запускает инструкции непосредственно на компьютере, а работает на гипотетическом компьютере, называемая Java Virtual Machine (JVM, виртуальная машина Java) [9].

Безопасность программ основано на том, что компания Sun Microsystems опубликовала спецификацию и код виртуальной машины. Данный шаг значительно ускорил выявление и устранение ошибок [10]. Для разработки на android платформе Java обладает рядом преимуществ по сравнению с другими языками.

Первое — это портируемость, благодаря JVM.

Второе, огромное сообщество, которое написало много различных библиотек под любые нужды и готовое оказывать помощь по любой проблеме [11].

Третье, большое количество обучающего материала, как платного, так и бесплатного.

Четвертое, синтаксис Java прост и ясен, что значительно уменьшает вероятность возникновения ошибок

### 2.3.2 Выбор интегрированной среды разработки

Для разработки программы под операционную систему android на языке Java существуют 3 среды разработки:

* Eclipse
* IntelliJ IDEA
* Android Studio

Eclipse — популярная IDE (Integrated Development Environment. Интегрированная Среда Разработки) для Java. Основным преимущество является наличие Eclipse Marketplace. Благодаря ему можно очень гибко настроить IDE скачивая или создавая плагины. Данные плагины представляют собой конструкторы графического интерфейса, инструменты для моделирования и тестирования, построения диаграмм и создания отчетов [12].

IntelliJ IDEA — IDE разработанная на Java от компании JetBrains. Главной особенностью является «глубокий интеллект», который индексирует написанный код и предоставляет соответствующие предложения в каждом контексте, такие как мгновенное и умное завершение кода, анализ кода на лету и надежные инструменты рефакторинга [13].

Android Studio — среда разработки, созданная совместными усилиями Google и JetBrains. Переняв у IntelliJ IDEA всё лучшее, он также обладает своими уникальными особенностями, которые делают данное IDE лучшим для андроид-разработки.

Во-первых, это редактор визуального макета, который позволяет писать xml код для экранов приложения, управляя визуальными объектами.

Во-вторых, анализатор apk, который позволит уменьшить вес приложения, просканировав файлы манифеста, ресурсов.

И самая главная черта, которая делает Android Studio такой популярной для разработок приложений по ОС Android, — это эмулятор. Он позволяет быстро устанавливать и запускать приложения на разных версиях андроид, моделировать различные конфигурации и функции, включая ARCore, платформу для создания дополненной реальности [14].

К тому же благодаря встроенному эмулятору удобнее проводить дебаг проекта, то есть находить ошибки и баги.

### 2.3.3 Выбор СУБД

Для того чтобы пользователь мог просматривать информацию о лекарствах вне зависимости от того, есть ли у него интернет-подключение или нет, нужно хранить данные локально. Есть 2 способа хранения данных:

* Хранение данных в файле в форматах, типа XML, JSON, CSV. Но в таком случае могут появиться ошибки при записи, изменении, удалении данных. Также при большом количестве данных скорость их обработки оставляет желать лучшего
* СУБД. В отличии от предыдущего варианта при использовании СУБД можно не беспокоиться о надежности и производительности.

Самыми популярными СУБД является реляционные, которыми можно управлять с помощью языка SQL. SQL — это язык для доступа к данным в реляционных базах данных, управляемых СУБД [15].

В Android уже есть встроенная СУБД — SQLite. SQLite — это библиотека, написанная на языке C, которая реализует небольшой, быстрый, полнофункциональный механизм базы данных SQL [16].

Особенностями SQLite являются:

* Стабильность. Транзакции завершаться без сбоев даже если сбой произошел в системе или в питании
* Простота. Отсутствие конфигурации, а также хранение всей базы данных в одном кроссплатформенном файле
* Самодостаточность. Для ее работы не требуется загружать дополнительные библиотеки
* Кроссплатформенность. Помимо Android SQLite доступна на всех популярных ОС

Приняв во внимание все альтернативы хранения информации и оценив плюсы встроенной СУБД, было принято решение использовать SQLite для хранения информации о лекарствах и напоминаниях.

### 2.3.4 Выбор дополнительных библиотек

Во избежание написания большого количества кода, а также проблем и ошибок, возникающих из-за этого, были использованы дополнительные библиотеки для следующих направлений разработки: создание ui, чтение html кода, создание уведомлений, а уменьшение количества sql кода.

Для создания интерфейса использовался Material Design — это система дизайна от Google, чтобы помочь создавать высококачественные цифровые приложение для Android. Он вдохновлен физическим миром и его текстурами, в том числе, как они отражают свет и отбрасывают тени.

Помимо рекомендаций к оформлению Material Design дает доступ к components, которые представляют собой строительные блоки для создания пользовательского интерфейса и включают встроенную систему для передачи состояний фокуса, выбора, активации, перетаскивания и т. д. [17].

Данные компоненты удовлетворяют ряд требований интерфейса, такие как:

* Отображение: размещение и организация содержимого с помощью таких компонентов, как cards, lists, sheets
* Навигация: позволяют пользователю перемещаться по продукту, используя navigation drawers, tabs
* Действия: добавление возможности пользователю выполнять задачи, нажимая на floating action button
* Ввод: позволяет пользователю вводить информацию или делать выбор в text fields, chips, selection controls
* Коммуникация: добавляет возможность оповещать пользователя о ключевой информации и вызывать другие сообщения благодаря snackbars, banners, dialogs

Оформление в material design осуществляется с помощью material theming, который позволяет настроить UI в соответствии с внешним видом бренда с помощью встроенной поддержки и инструкцией по настройке цветов, стилей типографики и угловых форм:

* Цвет: цветовая система – это организованный подход к применению цвета в UI. Глобальные цветовые стили имеют семантические имена и определенное использование в компонентах – первичные, вторичные (фирменные цвета), поверхности и фоны
* Типография: предоставляет 13 стилей для оформления всего, начиная с заголовков заканчивая основным текстом и подписями. Каждый стиль имеет четкое значение и предполагаемое применение в интерфейсе.
* Форма: применение различных форм может помочь привлечь внимание или идентифицировать компоненты, сообщить об их состоянии. Все компоненты Material Design группируются по категориям форм в зависимости от их размера (маленькие, средние, большие). Также можно создавать собственные стили фигур с помощью инструмента настройки форм

Для парсинга данных с веб-страницы были найдены 4 библиотеки:

* Jsoup: Java HTML Parser
* TagSoup
* Htmlcleaner
* Nexage SourceKit-MRAID For Android

После поверхностного сравнения данных библиотек выбор остановился на двух: jsoup и htmlcleaner ввиду того, что примеры кода выглядели лаконичными, а способ установки и использования более простым по сравнению с TagSoup и Nexage SourceKit-MRAID For Android.

После пробного использования окончательно был выбран jsoup, так как после замеров времени он показывал результат лучше, чем Htmlcleaner.

Jsoup – это библиотека для работы с HTML. Он предоставляет удобный API для получения URL-адресов, извлечения и обработки данных. Главной особенностью является то, что Jsoup реализует спецификацию WHATWG HTML5 [18] и анализирует страницы в той же модели DOM, что и современные браузеры.

Благодаря этому он обладает рядом преимуществ:

• Анализ HTML из URL, файла или строки

• Поиск и извлечение данных, используя CSS-селекторы и обход дерева DOM

• Манипулирование HTML-элементами, атрибутами и текстом

Jsoup предназначен для работы со всеми разновидностями HTML, которые встречаются в интернете; от чистых и валидированных до недействительных тегов он сможет создать правильное дерево синтаксического анализа [19].

Для создания уведомлений был имплементирован NotifyMe [20]. В отличии от ручного внедрения pop-up уведомлений, данная библиотека позволяет реализовать их проще и эффективнее благодаря RRULE.

RRULE определяет шаблон или правило повторения для задач, записей журнала и событий, благодаря нему можно легко установить задержку или время, когда уведомление появится, частоту, максимальное кол-во вызовов.

Использовать «чистый» sql-код неудобно, и в перспективе может вызвать большое количество ошибок. Для решения данной проблемы существуют ORM (Object-Relational Mapping, объектно-реляционное отображение) библиотеки.

Они предоставляют дополнительный уровень абстракции над sql, чтобы работать с базой данных на уровне классов и интерфейсов, а не таблиц и запросов. Для базы данных SQLite было создано множество ORM библиотек: OrmLite, SugarORM, GreenDAO, ActiveAndroid, Room.

Каждая представленная библиотека обладает своими минусами, плюсами и особенностями. Для выбора конкретной библиотеки был проведен сравнительный анализ, результаты которого изображены на таблице 2.

Таблица 2 — Сравнение различных ORM библиотек

| Сравнение  Название | OrmLite | SugarORM | GreenDAO | ActiveAndroid | Room |
| --- | --- | --- | --- | --- | --- |
| Инициализация сущностей | С помощью аннотаций | Наследование от класса | С помощью аннотаций | С помощью аннотаций | С помощью аннотаций |
| Добавление конфигурации | С помощью аннотаций | Добавление тегов в Android Manifest.xml | С помощью аннотаций | Добавление тегов в Android Manifest.xml, инициализация в главной активности | С помощью аннотаций |
| Скорость чтения (msec) | 238 | 3139 | 146 | 470 | 192 |
| Скорость записи (msec) | 1339 | 5299 | 731 | 4431 | 1035 |

Для целей проекта была выбрана библиотека Room по двум причинам: инициализация и конфигурирование происходит с помощью аннотаций, которые являются удобными и довольно наглядными, а также скорость выполнения запросов достаточно высокая для выбранной предметной области, в дополнении к этому у данной библиотеки хорошо написанная документация, которая позволяет быстро понять принципы работы с ней с нуля.

Room обеспечивает уровень абстракции над SQLite для более гибкого доступа к базе данных без потери мощной функциональности SQLite [21].

В частности, Room предоставляет следующие преимущества:

* Проверка SQL-запросов во время компиляции.
* Удобные аннотации, которые сводят к минимуму повторяющийся и подверженный ошибкам шаблонный код.
* Оптимизированные пути миграции базы данных.

У Room есть 3 основных компонента:

* Класс базы данных (Database class), который содержит базу данных и служит основной точкой доступа для базового подключения к постоянным данным приложения
* Сущности данных (Data entities), которые представляют таблицы в базе данных приложения
* Объекты доступа к данным (DAOs; Data access objects), реализующее методы, которое приложение может использовать для вставки, обновления, удаления информации из базы данных, а также для составления других запросов

Работа библиотеки представлена следующим образом. Класс базы данных предоставляет приложению объекты доступа к данным, которые с ней связаны. В свою очередь, приложение использует эти объекты для извлечения данных в качестве экземпляров объектов сущностей данных. Также приложение может использовать объекты этих сущностей для обновления или добавления новых строк в базе данных.

Ниже приведен рисунок 4, в котором показана взаимосвязь между различными компонентами Room.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 4 – Схема архитектуры библиотеки Room

## 2.4 UML-моделирование

UML (Unified Modeling Language; Унифицированный язык моделирования) представляет собой семейство графических обозначений, поддерживаемых единой метамоделью, которые помогают в описании и проектировании программных систем, в частности программных систем, построенных с использованием ООАП [22].

Для построения UML моделей было использовано приложение Umbrello [23]. Umbrello дает возможность создавать диаграммы программного обеспечения в стандартном формате для документирования или проектирования структуры программ. Многие функции данного приложения зависят от стандарта UML. Для этого проекта был использован стандарт версии 2.5

### 2.4.1 Диаграмма вариантов использования

Для того чтобы правильно описать, что такое диаграмма прецендентов или диаграмма вариантов использования (use cases diagram), нужно описать, что такое сценарий.

Сценарий – это последовательность шагов, описывающих взаимодействие между пользователем и системой. Варианты использования или преценденты – это набор сценариев, связанных вместе общей целью пользователя. А диаграмма прецендентов визуализирует данные варианты использования.

Основными компонентами диаграммы являются:

* Актер – это роль, которую пользователь играет по отношению к системе
* Варианты использования – набор сценариев
* Отношения – связи между актерами и прецендентами или между прецендентами и прецендентами. Основными отношениями являются:
  + Отношение ассоциации, отражающий возможность использования прецендента актером
  + Отношение включения указывает на то, что один прецендент в определенной точке включает в себя другой прецендент в качестве составного компонента
  + Отношение расширения – присоединение одного варианта к другому

Ниже, на рисунке 5 и в таблице 3, приведена диаграмма прецендентов, показывающая работу программа

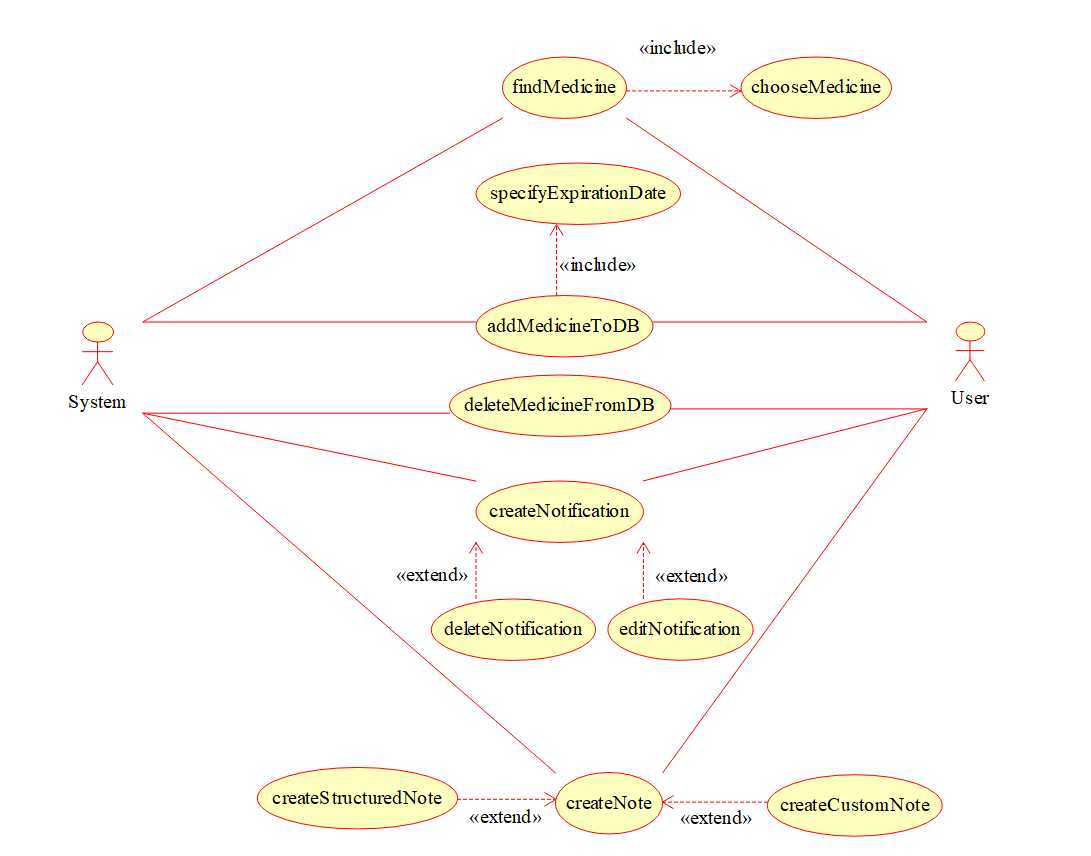


Рисунок 5 – Диаграмма вариантов использования

Таблица 3 — Детализация диаграммы варианта использования

| Вариант использования | Детализация |
| --- | --- |
| findMedicine (найти лекарство) | Найти лекарство на сайте  **Входные параметры:** название лекарства в виде строки  **Выходные данные:** URL адрес найденных лекарств |
| chooseMedicine (выбрать лекарство) | Выбор конкретного лекарства из списка  **Входные параметры:** список лекарств, которые мы получили от метода findMedicine  **Выходные данные:** выбранное лекарство |
| addMedicineToDB (добавить лекарство в базу данных) | Добавление лекарства в локальное хранилище  **Входные параметры:** найденное или созданное пользователем лекарство  **Выходные данные:** статус операции — успешно/не успешно |
| specifyExpirationDate (указать срок годности) | Указание срока годности лекарства  **Входные параметры:** выбранное лекарство, которое получили от addMedicineToDB  **Выходные данные:** статус операции — успешно/не успешно |
| deleteMedicineFromDB (удалить лекарство из базы данных) | Удаление лекарства из локального хранилища  **Входные параметры:** выбранное лекарство  **Выходные данные:** статус операции — успешно/не успешно |
| createNotification (создать напоминание) | Создание напоминания. После создания при нажатии на кнопку «удалить напоминание» будет вызван метод deleteNotification. При нажатии на само уведомление в списке будет вызван метод editNotification  **Входные параметры:** название лекарства, которое нужно принять; количество лекарства для приема; время приема лекарства  **Выходные данные:** статус операции — успешно/не успешно |
| deleteNotification (удалить напоминание) | Удаление напоминания  **Входные параметры:** созданное напоминание  **Выходные данные:** статус операции — успешно/не успешно |
| editNotification (редактировать напоминание) | Изменение созданного напоминания  **Входные параметры:** созданное напоминание  **Выходные данные:** статус операции — успешно/не успешно |
| createNote (создание заметки) | Создание заметки о приеме лекарства  **Входные параметры:**  Название лекарства, частота сердцебиения, артериальное давление  Если пользователь нажмет на кнопку «Создать структурированную заметку», то вызовется метод createStructuredNote. Если на «Создать заметку», то вызовется метод createCustomNote  **Выходные данные:**  статус операции — успешно/не успешно |
| createStructuredNote (создание заметки с определенной структурой) | Создание структурированной заметки о состоянии здоровья  **Входные параметры:**  Оценки одышки, потоотделения, головокружения, а также краткое описание состояния здоровья  **Выходные данные:**  статус операции — успешно/не успешно |
| createCustomNote (создание заметки без структуры) | Создание простой заметки  **Входные параметры:**  Краткое дополнительное описание  **Выходные данные:**  статус операции — успешно/не успешно |

### 2.4.2 Диаграмма классов

Диаграмма классов описывает структуру программы, показывая все сущности и их атрибуты, а также отношения между ними.

Как видно на рисунке 6 классы на диаграмме представлены в виде прямоугольников, которые содержат три раздела:

* верхний – название класса
* средний содержит атрибуты класса в виде <модификатор доступа> <название атрибута> : <тип атрибута>
* нижний раздел нужен для методов, которые реализует данный класс. На рисунке ниже представлена диаграмма классов для сущностей. Они не могут иметь функциональных методов, из-за этого нижний раздел – пустой.

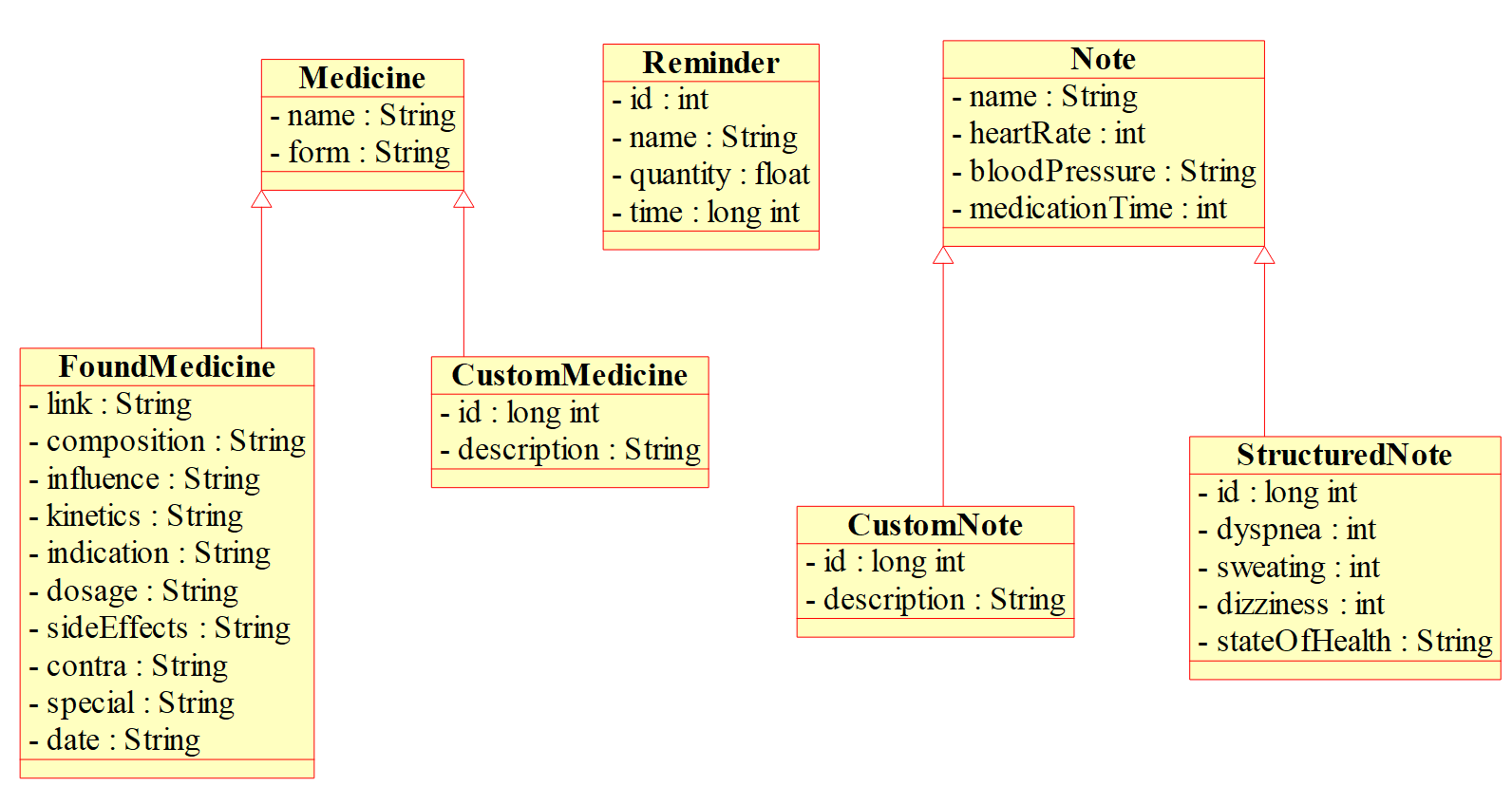


Рисунок 6 – Диаграмма классов

## 2.5 Проектирование пользовательского интерфейса

Пользовательский интерфейс проектировался в Figma – это редактор векторной графики и инструмент для создания прототипов. Данный онлайн сервис позволяет удобно разрабатывать интерфейс приложения благодаря большому сообществу, которое может выкладывать свои наработки или отдельные элементы в открытый доступ, где каждый может воспользоваться данными прототипами.

Пользовательский интерфейс разрабатывался в соответствии с диаграммой вариантов использования, которая был описана выше.

Из той диаграммы можно выделить 4 основные функции – это:

1. поиск лекарства
2. добавление и удаление лекарства
3. добавление, изменение и удаление уведомлений
4. добавление, изменение и удаление заметок

Относительно данных основных функций был разработан интерфейс приложения, представленный на рисунке 7.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 7 – интерфейсы основных экранов

## 2.6 Выводы и результаты по главе 2

В данной главе были рассмотрены и выбраны методология проектирования, инструментальные средства разработки, также была нарисована диаграмма вариантов использования и спроектирован пользовательский интерфейс. Всё это позволит перейти к следующему этапу: реализации программного продукта

# 3. Реализация приложения

## 3.1 Реализация базы данных

Для сохранения данных в памяти телефона была использована база данных SQLite, а взаимодействие с данной СУБД происходило с помощью библиотеки Room. Для использования данной библиотеки необходимо наличие трех компонентов:

* Entity – класс, которому будет сопоставлена таблица в БД. На листинге 1 приведен один из таких классов
* DAO – интерфейс, методы которого приложение будет использовать для запросов к БД. Листинг 2 показывает пример данного интерфейса
* Database – абстрактный класс, в котором определены entity и dao. В листинге 3 показан данный абстрактный класс.

**Листинг 1 – Аннотация @Entity для класса Medicine**

@Entity

public class Medicine {

@PrimaryKey

@NonNull

String link;

Аннотация @PrimaryKey указывает, какое поле класса должно быть первичным ключом. Первичный ключ нужен для идентификации каждой строки в таблице для возможности получения определенной строки, чтобы в дальнейшем она могла быть изучена и/или изменена. И так как первичный ключ не может быть пустым (Null), то дополнительно нужно указать аннотацию @NonNull.

**Листинг 2 – Интерфейс MedicineDao**

@Dao

public interface MedicineDao {

@SuppressWarnings(RoomWarnings.CURSOR\_MISMATCH)

@Query("SELECT name, link, form FROM medicine")

List<Medicine> getAll();

@Query("SELECT \* FROM medicine where link = :link")

Medicine getByLink(String link);

@Insert

void insert(Medicine medicine);

@Delete

void delete(Medicine medicine);

}

В библиотеке Room есть аннотации со встроенными запросами, например @Insert, @Delete. Для использования других запросов нужно прописать аннотацию @Query, в которой пишется SQL запрос.

**Листинг 3 – Абстрактный класс AppDatabase**

@Database(entities = {FoundMedicine.class, CustomMedicine.class, StructuredNote.class, CustomNote.class, Reminder.class}, version = 4)

public abstract class AppDatabase extends RoomDatabase{

public abstract FoundMedicineDao foundMedicineDao();

public abstract CustomMedicineDao customMedicineDao();

public abstract StructuredNoteDao structuredNoteDao();

public abstract CustomNoteDao customNoteDao();

public abstract ReminderDao reminderDao();

}

В аннотации @Database указываются все Entity классы, которые представляют таблицы в БД, а также версию БД, которую нужно обновлять каждый раз при внесении изменений в Entity или добавлений новых данных в этот класс. Внутрь класса вписываются все DAO классы, которые используются для доступа к таблицам в БД.

## 3.2 Реализация основных функций

Для получения данных о лекарстве с сайта используется Jsoup. Он позволяет подключиться к веб-странице, получить оттуда весь HTML-код в формате Document, после чего используя методы из данной библиотеки получить необходимую информацию в формате List<String>.

Листинг 4 показывает использование данной библиотеки.

**Листинг 4 – Использование Jsoup**

try {

doc = Jsoup.connect(String.format(Constants.URL, url)).get();

} catch (IOException e) {

e.printStackTrace();

}

String[] names = doc.select(Constants.URL\_NAME) .eachText().toArray(new String[0]);

String[] links = doc.select(Constants.URL\_NAME) .eachAttr(Constants.URL\_LINK).toArray(new String[0]);

String[] forms = doc.select(Constants.URL\_FORM) .eachText().toArray(new String[0]);

Для вывода списка лекарств, напоминаний и заметок на экран использовались Adapter с ListView. Дополнительно к этому полученные данные из БД оборачивались в LiveData.

LiveData – это класс-наблюдатель (observer) держателя данных. В отличие от обычных, LiveData учитывает жизненный цикл других компонентов приложения, таких как Activity, Fragment или Services. Это гарантирует, что LiveData обновляет только те данные компонентов приложения, которые находятся в активном состоянии жизненного цикла [24].

Пример использования данных возможностей приведен в листинге 5.

**Листинг 5 – Получение данных и вывод на экран**

dashboardViewModel.getAllFoundMedicines().observe(getViewLifecycleOwner(), foundMedicines -> {

MedicineAdapter adapter = new MedicineAdapter(requireContext(), foundMedicines.toArray(new FoundMedicine[0]));

lv.setAdapter(adapter);

lv.setOnItemClickListener((parent,view,position,id) -> {

Intent intent = dashboardViewModel.setIntent(root.getContext(), adapter, position);

startActivity(intent);

});

});

Для создания пуш уведомлений использовалась библиотека NotifyMe. Благодаря данной библиотеки можно с легкостью настроить уведомления, а также задать множество различных параметров для вывода.

Пример создания уведомлений представлен на листинге 6.

**Листинг 6 – создание NotifyMe**

private Calendar setNotifyMe(NotifyMe.Builder notifyMe, TimePicker t){

Calendar calendar = Calendar.getInstance();

calendar.set(Calendar.HOUR\_OF\_DAY, t.getHour());

calendar.set(Calendar.MINUTE, t.getMinute());

notifyMe.title(Constants.TITLE);

notifyMe.content(String.format(Constants.MESSAGE, reminder.getName(), reminder.getQuantity()));

notifyMe.time(calendar);

notifyMe.rrule(Constants.FREQUENCY);

return calendar;

}

Для создания интерфейса был выбран шаблон Bottom Navigation Activity. Он представляет стандартную нижнюю панель навигации для приложения. Это реализация нижней навигации, представленной в Material Design. Нижние панели навигации позволяют пользователям легко исследовать и переключаться между представлениями верхнего уровня одним касанием. Их следует использовать, когда приложение имеет от трех до пяти мест назначения верхнего уровня [25].

Данный шаблон использует фрагменты для представления интерфейса, поэтому в MainActivity их нужно определить как показано в листинге 7.

**Листинг 7 – код класса MainActivity**

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

BottomNavigationView navView = findViewById(R.id.nav\_view);

AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(R.id.navigation\_home, R.id.navigation\_dashboard, R.id.navigation\_notifications, R.id.navigation\_note).build();

NavController navController = Navigation.findNavController(this, R.id.nav\_host\_fragment);

NavigationUI.setupActionBarWithNavController(this, navController, appBarConfiguration);

NavigationUI.setupWithNavController(navView, navController);

}

}

## 3.3 Выводы и результаты по главе 3

В данной главе описана реализация мобильного приложения от базы данных и взаимодействия с ней до обработки действий пользователя и функция взаимодействия данных с интерфейсом.

# 4. Тестирование

Одним из главных этапов разработки любого приложения является тестирование. Сегодня справедливо сказать, что все разработчики изучают и используют методологию модульного тестирования (unit-тесты) на ранних стадиях, и она стала основополагающей для успешной разработки программного обеспечения [26].

К типам тестирования относят юнит тестирование, интеграционные и системные тесты, а также тестирование графического интерфейса. Тестирование мобильных приложений – это процесс, с помощью которого прикладное ПО, разработанное для портативных мобильных устройств, проверяется на его функциональность, удобство использования и совместимость [27].

Для данного приложения использовались 2 вида тестирования:

* Модульное тестирование – метод тестирования, с помощью которого тестируются отдельные модули, чтобы определить, есть ли какие-либо проблемы с ними. Речь идет о корректности работы автономных модулей. Основная цель - изолировать каждую единицу системы для выявления, анализа и исправления дефектов.
* Тестирование производительности – это общее название для тестов, которые проверяют как система ведет себя и работает. Оно исследует быстродействие, стабильность, масштабируемость, надежность, скорость и использование ресурсов программного обеспечения.

## 4.1 Модульное тестирование

Есть множество библиотек для написания модульных тестов на java. Среди самых популярных – это Junit и TestNG.

Junit – один из самых популярных фреймворков, который используется для тестирования java кода.

Его преимущество состоит в первую очередь в простоте, для написания тестов достаточно просто создать класс и пометить один из методов аннотацией @Test. Вся конфигурация порядков вызова методов и определенных условий для вызова методов тоже определяются с помощью аннотаций, таких как: @BeforeAll, @BeforeEach, @AfterEach, @AfterAll.

Среди недостатков можно отметить невозможность проведения независимых тестов, методы которого обращаются к внешним данным, например базе данных или определенному сервису.

Для решения этой проблемы существует библиотека Mockito, которая позволяет заменять реальные объекты на заглушки с дополнительной логикой.

Далее прямой конкурент Junit – TestNG. Он обладает почти всеми возможностями Junit, но также предоставляет больше возможностей в параметрическом тестировании и тестировании зависимостей. Он в основном используется для функционального и высокоуровневого тестирования, а также для комплексных интеграционных тестов.

Ввиду того что для данного программного продукта не нужны столь точно настроенные тесты, то использование TestNG только усложнит проект, но не добавит заметных преимуществ по сравнению с Junit.

Поэтому в качестве фреймворка для тестирования разрабатываемого приложения был выбран Junit.

Для тестирования использовалась 4-я версия Junit [28]. Тестировались основные CRUD операции, т. е. создание, чтение, редактирование и удаление для каждой сущности. Как видно из результатов на рисунке 8. Все тесты успешно выполнились.

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 8 – Результаты выполнения тестов

## 4.2 Тестирование производительности

Основные параметры, которые нужно проверять при тестировании производительности – это [29]:

* Запуск приложения. На запуск приложения не должно уходить больше трех секунд, иначе пользователь может устать ждать и удалит мобильное приложение.
* Загрузка процессора. Чем меньше загружен процессор, тем производительнее будет приложение, а заряд батарее будет меньше расходоваться.
* Потребление оперативной памяти. До сих пор существуют мобильные телефоны с оперативной памятью в 1 ГБ, половину из которых занимают другие приложения и процессы. Поэтому оптимизация потребления памяти является важным шагом для создания хорошего приложения.
* Частота кадров. Хоть частота кадров в основном важна для игр, но и для обычного приложения fps в диапазоне 20–30 является важным качеством.

Существует множество сторонних программ для тестирования производительности мобильного приложения.

Например, такие программы как: Akamai CloudTest, Gatling, JMeter служат для проверки производительности системы при сильной нагрузке. Так как создаваемое мобильное приложение обладает в основном оффлайн функциями, а также не взаимодействует с другими приложениями в телефоне, то нагрузочное тестирование практически ничего не скажет о производительности приложения.

Поэтому программы данного типа не подойдут для тестирования в разрабатываемом проекте.

Программы по типу: Eggplant, HeadSpin используют искусственный интеллект для симуляции действия пользователя. Главный их минус заключается в сложной, глобальной настройке тестов. Поэтому использование таких программ слишком усложняет проект по отношению к пользе, которую могут дать результаты тестирования.

Третий тип программ мониторит производительность мобильного телефона в то время, как разработчик самостоятельно выполняет действия, которые может выполнять пользователь. К таким программам относятся: Experitest, Apptim, а также встроенный в IDE Android Studio profiler. Данный тип подходит для создаваемого приложения, в виду того, что они просты в настройке и эксплуатации, а также быстро могут показать в каких местах приложение может быть неудобным для использования.

Для тестирования производительности была использована программа “Apptim” [30]. Она производит мониторинг системы, на которой запущено приложение, и оценивает производительность, генерируя отчет об этом, который можно сравнить с другими отчетами для того, чтобы узнать, насколько изменилась производительность системы после того, как вышла новая версия.

После тестирования данная программа предоставляет краткую информацию, как на рисунке 9, в которой можно увидеть, время запуска, среднее использование центрального процессора и оперативной памяти, а также средний ФПС работы.

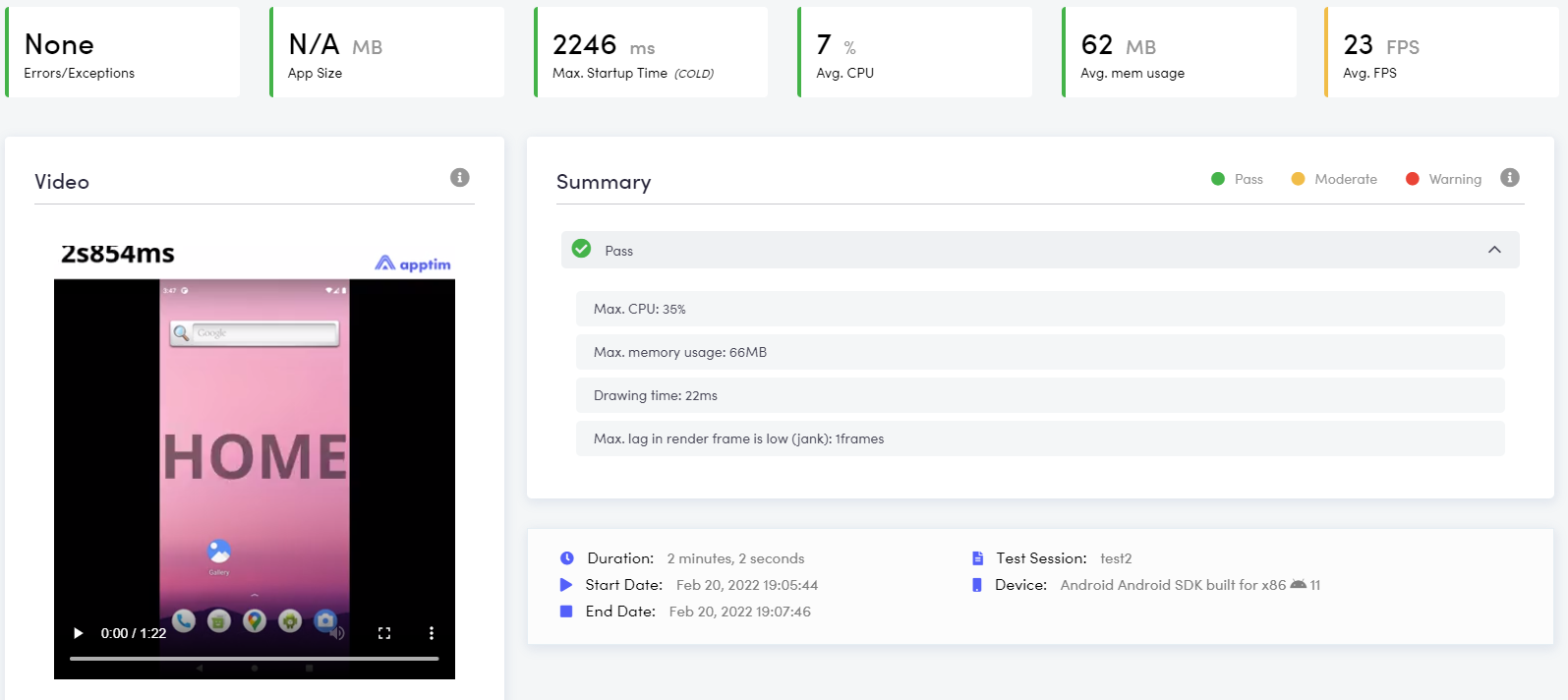


Рисунок 9 – краткая информация производительности системы

По представленным данным можно видеть, что основные параметры разрабатываемого продукта, на которые могут обращать внимание пользователи, в пределах допустимых значений, а также нет критичных ошибок.

## 4.3 Выводы и результаты по главе 4

В данной главе было описано тестирование как с использование сторонних приложений, так и со встроенными функциями выбранной IDE, а также показаны результаты тестов данного продукта по двум видам тестирования.

# 5. Безопасность жизнедеятельности

Внедрение технологий и автоматизация бизнес-процессов как напрямую, так и косвенно позволяют увеличить экономическую эффективность предприятия за счет того, что сотрудники способны обрабатывать большее количество информации, быстрее производить расчеты, лучше подготавливать регламентированную отчетность. Но несмотря на то, что пользователи персональных компьютеров в целом и программисты в частности работа относительно спокойная и легкая, из-за того, что данная работа малоподвижная и сидячая у людей данной специальности есть риск развития серьезных заболеваний, связанных в основном с эргономическими факторами [31].

К таким факторам относятся:

* Сидячее положение в течение длительного времени
* Утомление глаз, нагрузка на зрение
* Нагрузка на суставы кистей
* Умственное перенапряжение, вызванное информационной нагрузкой

Данные факторы могут вызвать такие заболевания, как остеохондроз, артрит, радикулит ожирение, близорукость, гиподинамия, синдром запястного канала, варикоз, геморрой, рассеянный склероз, анемия, головокружения, мигрени, заболевания позвоночника, глазные заболевания и заболевания сердечно-сосудистой системы.

## 5.1 Разработка комплекса мер защиты и профилактики

Так как разработка данного программного продукта подразумевает частое и длительное использование персонального компьютера, то, для того чтобы риск развития заболеваний был минимален, необходимо придерживаться определенных правил использования компьютера. Роспотребнадзор предлагает следующие рекомендации [32]:

* Чтобы избежать проблем, связанных со зрением, нужно позаботиться о том, чтобы монитор стоял не ближе, чем в 45 сантиметрах от глаз, а верхняя точка монитора должна быть не ниже уровня глаз. Экран монитора не должен быть повернут в сторону окна. В темное время суток работать с компьютером, где единственным источником света является монитор, нельзя. Избежать неприятных ощущений, связанных с глазами, возможно, если раз в полчаса отвлекать взгляд от экрана делать гимнастику для глаз или, хотя бы, просто моргать, смотреть вдаль.
* Для профилактики от синдрома запястного канала рекомендуется регулярно делать легкую зарядку для рук, а также во время работы за компьютером пользоваться специально разработанными подушечками
* Температура, относительная влажность и скорость движения воздуха на рабочих местах, а также уровни положительных и отрицательных аэроионов и химических веществ в воздухе помещений, должны соответствовать действующим санитарным нормам микроклимата производственных помещений [33].
* Особую значимость для предупреждения перенапряжения имеют рациональные режимы труда и отдыха, установленные в соответствии с характером и условиями труда, динамикой функционального состояния работников. Рациональный режим, помимо перерыва на обед, должен включать регламентированные перерывы, общая продолжительность которых должна быть не менее 2–3 перерывов по 15 минут каждый.
* Перерывы следует заполнять производственной гимнастикой, направленной на расслабление основных работающих мышц, проведение самомассажа рук.

## 5.2 Выводы и результаты по главе 5

В данной главе рассмотрены основные заболевания, связанные с длительным использование персонального компьютера, а также разработан комплекс мер для профилактики.

# 6. Экономическое обоснование

Использованный стек технологий бесплатен для разработчика и не требует вложений и инвестиций в будущем.

## 6.1 Определение стоимости

Основной стоимостью продукта является время, затраченное на разработку. Учитывая декомпозицию задачи, описанной во второй главе, можно составить таблицу с основным этапами разработки, а также затраченным временем на выполнение. Данные приведены в таблице 4.

Таблица 4 – Время выполнения

| Этап разработки | Затраченное время (часы) |
| --- | --- |
| Анализ предметной области | 35 |
| Поиск существующих решений | 6 |
| Формирование основных задач | 41 |
| Выбор методологии проектирования | 5 |
| Выбор средств реализации | 16 |
| Разработка UML-диаграмм | 29 |
| Проектирование пользовательского интерфейса | 38 |
| Создание базы данных | 12 |
| Разработка основных функций | 111 |
| Тестирование | 19 |
| Итого | 312 |

Исходя из таблицы, на разработку данного приложения было потрачено 39 дней. Согласно статье 91 Трудового Кодекса Российской Федерации предел нормального рабочего времени в 40 часов в неделю [34], при условии, что рабочая неделя составляет 5 дней, то средний рабочий день составляет 8 часов. Соответственно в пересчете на дни на создание программного продукта было потрачено: 39 дней.

Для оценки стоимости продукта нужно узнать заработную плату Android разработчика по Ростову-на-дону. По данным сайта hh.ru, одного из крупнейших компаний по поиску работы [35], минимальная зарплата составляет 50000 рублей [36]. При работе 5 дней в неделю среднее количество дней в месяце составляет: 21 день. Из этого выходит, что оплата одного рабочего дня составляет:50000/21 ≈ 2380,95 руб./день.

Таким образом примерные денежные затраты на реализацию данного продукта составляют: 2380,95\*39 ≈ 93000 рублей.

Это и будет являться себестоимостью продукта. Оценив себестоимость можно приступить к вычислению окупаемости продукта

## 6.2 Вычисление окупаемости

Монетизировать приложение можно двумя способами.

Первое, добавить платную расширенную версию с дополнительным функционалом. При таком подходе нужно рассчитать стоимость создания дополнительного функционала, а также приблизительно понять, какое количество загрузок будет у данного приложения и сколько людей будут готовы купить расширенную версию.

Второе, вставлять рекламные баннеры в экраны приложений. Для этого нужно узнать, сколько раз пользователь будет заходить в приложение и переключать экраны и какую цену готовы предложить рекламодатели за рекламное место.

## 6.3 Расчет окупаемости через расширенную версию

В качестве дополнительного функционала можно ввести такие особенности, как:

* Возможность кастомизации вывода информации о лекарстве, то есть сокрытие некоторых пунктов, перестановка
* Возможность добавление группировки лекарств по различным характеристикам, такими как: по способу принятия лекарства (мазь, таблетки, капли), по месту хранения лекарств (аптечка дома, в машине)
* Добавление более удобной кастомизации уведомлений, например: возможность создания уведомлений, которые будут вызваны более одного раза в день (утром (8:00), днем (14:00), вечером (20:00))
* Возможность бекапа сохраненных лекарств с облака

Каждое из данных функциональных особенностей будет затрагивать переработку/добавление 10–15% кода, соответственно при грубом подсчете в среднем на добавление одной функциональности будет уходить 5 дней или одна рабочая неделя.

Из этого следует что на добавление всего дополнительного функционала уйдет 1 месяц, то есть затраты на разработку увеличатся на 50000 рублей и будут составлять: 93000 + 50000 = 143000 рублей.

Используя данные таблицы 1, возьмем среднее количество скачиваний, которое составляет 350000. Среди аналогов есть только одно приложение с платными функциями, и ее стоимость составляет 319 рублей. Взяв ценник ниже данной отметки в 299 рублей, потребуется: 143000/299 ≈ 480 человек.

Если 480 человек или 0,1% от пользователей купят расширенную версию, то тогда данный программный продукт окупит свою стоимость.

## 6.4 Расчет окупаемости через рекламные баннеры

В зависимости от вида рекламы, их стоимость будет разниться. В настоящее время существуют такие виды реклам, как:

* Баннерная – небольшие блоки изображений или текста. Данный вид рекламы можно вставлять вверху, внизу экрана тем самым не раздражая пользователей, а переходить по рекламным баннерам будут, в основном, заинтересованные пользователи, что является также плюсом для рекламодателей. Такой вид рекламы хоть и является самым дешевым, но, учитывая плюсы, является предпочтительным
* Нативная – встроенные рекламные блоки, которые маскируются под дизайн приложения. Ввиду специфики данного вида (подойдет, в основном, для приложений с новостной лентой или аналогами) данный вид рекламы не подойдет для данного проекта.
* Interstitial (межстраничная) – полноэкранная реклама, отображающаяся в промежутке между запросом информации пользователем и ее получением (например, при переходе на следующий экран). Несмотря на очевидный плюс того, что пользователь обязательно его увидим, данный вид рекламы является самым раздражительным и может снизить количество активных пользователей приложения.

Рассмотрев основные виды рекламы, наиболее предпочтительной является баннерная реклама, поэтому далее будет рассмотрен только данный способ монетизации.

Есть 2 способа размещать рекламу в своем приложении: самому искать рекламодателей, но это сложный процесс, который может не окупиться, либо обратиться в одно из рекламных сетей, которые сами будут выбирать, какую рекламу вставить. В данном примере будет рассматриваться второй случай.

Для расчета дохода от размещения рекламы был выбран Google AdMob [37]. Чтобы рассчитать стоимость, нужно выбрать местоположение пользователей приложения и категорию рекламного контента.

Данном приложением ввиду того, что оно полностью на русском языке будут пользоваться люди из России и стран СНГ, категория – медицина из-за тематики проекта.

Google AdMob позволяет рассчитать годовой доход, основываясь на количестве просмотров приложения в месяц. Среднее предполагаемое количество пользователей равно 350000. Количество открытий приложения в день зависит от количества лекарств для приема. Во время болезни обычно выписывается от двух до четырех препаратов, поэтому возьмем среднее равное трем. По данным федеральной службы государственной статистики (Росстат) в год регистрируется заболеваний у приблизительно 250000 человек [38], что составляет примерно 0,17% от всего населения страны.

Соответственно количество заболевших людей с данным приложением равно: 350000\*0,17% ≈ 61000. При количестве открытий приложения равным трем, количество просмотра рекламы составляет 61000\*3 = 183000.

Учитывая данное значение годовой доход от баннерной рекламы в приложении составит: 994$. По курсу на момент написания работы это составляет 99400 рублей.

При стоимости приложения в 93000, он окупит себя через (93000/99400)\*12 ≈ 11 месяцев

## 6.5 Выводы и результаты по главе 6

В данной главе был описан комплекс работ по разработке программы с оценкой их продолжительности, также оценены затраты на реализацию. Рассмотрены два варианта окупаемости приложения и произведены расчеты для этих вариантов.

# Заключение

Результатом выполнения выпускной квалификационной работы является мобильное приложение учета лекарственных средств в аптечке пользователя. Реализованы все поставленные задачи. Пользователь может найти препарат в интернете по названию или создать свои заметки о препарате, указав название, форм-фактор и срок годности. Также он может создавать и напоминания, которые в назначенное время будут всплывать в виде пуш-уведомления. В дополнение пользователь может писать заметки о состоянии здоровья в структурированном или произвольном формате.

Разработанное приложение соответствует всем требованиям к выпускной квалификационной работе. В процессе выполнения был произведен анализ предметной области и аналогов на рынке. Выбраны средства проектирования и разработки, в последствии, основываясь на данные проектирования, был разработан программный продукт и проведены тесты для проверки корректности выполняемых функций.

# Список используемых источников

1. Домашняя аптечка. URL: <https://mos03.ru/rubrics/childrens-doctor-advice/domashnyaya-aptechka/> (Дата обращения 01.04.2022)
2. Исследование Rambler&Co и СБЕР ЕАПТЕКИ: 26% россиян обновляют домашнюю аптечку каждые полгода. URL: <https://press.sber.ru/publications/issledovanie-rambler-co-i-sber-eapteki-26-rossiian-obnovliaiut-domashniuiu-aptechku-kazhdye-polgoda> (Дата обращения 03.04.2022)
3. Медиапотребление в России – 2020. URL: <https://www2.deloitte.com/content/dam/Deloitte/ru/Documents/technology-media-telecommunications/russian/media-consumption-russia-2020.pdf> (Дата обращения 03.04.2022)
4. Домашняя Аптечка: учёт медикаментов. URL: <https://play.google.com/store/apps/details?id=com.vastuf.medicinechest&hl=ru&gl=US> (Дата обращения 03.04.2022)
5. Аптечка. URL: <https://play.google.com/store/apps/details?id=com.veos.pharm&hl=ru&gl=US> (Дата обращения 03.04.2022)
6. Моя аптечка + Аналоги лекарств. URL: <https://play.google.com/store/apps/details?id=com.overinet.apteka> (Дата обращения 03.04.2022)
7. Google's latest Android version distribution numbers show 11 in dead heat with 10. URL: <https://www.androidpolice.com/googles-latest-android-version-distribution-numbers-show-11-in-dead-heat-with-10/> (Дата обращения 13.04.2022)
8. Grady Booch, Robert Maksimchuk, Michael Engle, Jim Conallen, Kelli Houston, Bobbi Young Ph.D. Object-Oriented Analysis and Design with Applications, Pearson Education, 2007. 721с.
9. Joyce Farrell. Java programming, Cengage Learning, 2011. 912c.
10. Язык программирования Java: особенности и преимущества. URL: <https://oracle-patches.com/coding/язык-программирования-java-особенности-и-преимущества> (Дата обращения 13.04.2022)
11. Java Community Home. URL: <https://www.oracle.com/java/technologies/javacommunity.html> (Дата обращения 13.04.2022)
12. Eclipse desktop & web IDEs. URL: <https://www.eclipse.org/ide/> (Дата обращения 13.04.2022)
13. IntelliJ IDEA: The Capable & Ergonomic Java IDE by JetBrains. URL: <https://www.jetbrains.com/idea/> (Дата обращения 13.04.2022)
14. Download Android Studio and SDK tools. URL: <https://developer.android.com/studio> (Дата обращения 13.04.2022)
15. Jim Melton, Alan R. Simon. Understanding the New SQL: A Complete Guide, Morgan Kaufmann, 1993. 536с.
16. SQLite Home Page. URL: <https://www.sqlite.org/index.html> (Дата обращения 16.04.2022)
17. Introduction – Material Design. URL: <https://material.io/design/introduction> (Дата обращения 28.04.2022)
18. HTML. Living Standard. URL: <https://html.spec.whatwg.org/> (Дата обращения 28.04.2022)
19. jsoup: Java HTML Parser. URL: <https://jsoup.org/> (Дата обращения 28.04.2022)
20. NotifyMe. URL: <https://github.com/jakebonk/NotifyMe> (Дата обращения 28.04.2022)
21. Save data in a local database using Room. URL: <https://developer.android.com/training/data-storage/room> (Дата обращения 28.04.2022)
22. Martin Fowler. UML Distilled: A Brief Guide to the Standard Object Modeling Language, Addison-Wesley Professional, 2004. 175с.
23. Welcome to Umbrello - The UML Modeller. URL: <https://umbrello.kde.org/> (Дата обращения 05.05.2022)
24. LiveData Overview. URL: <https://developer.android.com/topic/libraries/architecture/livedata> (Дата обращения 12.05.2022)
25. BottomNavigationView. URL: <https://developer.android.com/reference/com/google/android/material/bottomnavigation/BottomNavigationView> (Дата обращения 12.05.2022)
26. Jason Porter, Alex Soto, Andrew Gumbrecht. Testing Java Microservices: Using Arquillian, Hoverfly, AssertJ, JUnit, Selenium, and Mockito, Simon and Schuster, 2018. 296 с.
27. Daniel Knott. Hands-on Mobile App Testing: A Guide for Mobile Testers and Anyone Involved in the Mobile App Business, Addison-Wesley, 2015. 230 с.
28. Alan J. Richardson. Java for Testers: Learn Java Fundamentals Fast, Compendium Developments, 2015. 436 с.
29. Brian Fling. Mobile Design and Development: Practical concepts and techniques for creating mobile sites and web apps, "O'Reilly Media, Inc.", 2009. 336 с.
30. Fully test your mobile app and improve performance at minimum effort. URL: <https://www.apptim.com/> (Дата обращения 20.05.2022)
31. Профессиональные заболевания программистов и их профилактика. URL: <https://pandia.ru/text/79/096/81685.php> (Дата обращения 24.05.2022)
32. ВНИМАНИЕ ПОТРЕБИТЕЛЯ: Профилактика профессиональных заболеваний сотрудников офиса. URL: <https://www.rospotrebnadzor.ru/about/info/news/news_details.php?ELEMENT_ID=13777> (Дата обращения 24.05.2022)
33. НИИ медицины труда РАМН (Афанасьева Р. Ф., Репин Г. Н., Михайлова Н. С., Бессонова Н. А., Бурмистрова О. В., Лосик Т. К.); Московский НИИ гигиены им. Ф.Ф.Эрисмана (Устюшин Б.В.); при участии Санкт-Петербургского НИИ гигиены труда и профзаболеваний (Синицина Е.В., Чащин В. П.); Госкомсанэпиднадзор России (Лыткин Б. Г., Кучеренко А.И.). САНИТАРНЫЕ ПРАВИЛА И НОРМЫ, СанПиН 2.2.4.548–96
34. Трудовой кодекс Российской Федерации. Москва: Эксмо, 2022. 256с.
35. Рейтинг топ веб-сайтов категории Работа и трудоустройство. URL: <https://www.similarweb.com/ru/top-websites/category/jobs-and-career/jobs-and-employment/> (Дата обращения 28.05.2022)
36. Вакансия «Android developer». URL: <https://rostov.hh.ru/search/vacancy?clusters=true&area=76&ored_clusters=true&order_by=salary_asc&enable_snippets=true&salary=&text=Android+developer&from=suggest_post&page=1&hhtmFrom=vacancy_search_list> (Дата обращения 28.05.2022)
37. Google AdMob – монетизация мобильных приложений. URL: <https://admob.google.com/intl/ru/home/> (Дата обращения 28.05.2022)
38. Л.И. Агеева, Г.А. Александрова, Н.М. Зайченко, Г.Н. Кириллова, С.А. Леонов, Е.В. Огрызко, И.А. Титова, Т.Л. Харькова, В.Ж. Чумарина, Е.М. Шубочкина Здравоохранение в России. 2017: Стат.сб./Росстат. - М., З-46 2017. – 170 с.