

**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук
Департамент программной инженерии

УТВЕРЖДАЮ
Академический руководитель
образовательной программы
«Программная инженерия»,
профессор департамента программной
инженерии, канд. техн. наук

_____ В.В. Шилов
«___» _____ 2021 г.

Выпускная квалификационная работа


на тему **«Клиент-серверное приложение органайзер лекарств»**

по направлению подготовки 09.03.04 «Программная инженерия»

Научный руководитель
НИУ ВШЭ, Доцент
Е.Ю. Песоцкая

_____ 

Выполнил
студент группы БПИ172
4 курса бакалавриата
образовательной программы
«Программная инженерия»
Д.Р. Долгошеев

_____ 

Реферат

В наше время необходимо позаботиться о здоровье человека и максимально упростить этот процесс. Многие люди устанавливают разные приложения, которые могут помочь им решить эту проблему. Решение, описанное в этой статье, может помочь людям организовать свое время и немного облегчить себе жизнь.

Итак, в этом документе описывается приложение для Android, которое может помочь людям более эффективно заботиться о своем здоровье и организовать прием лекарств в несколько кликов, используя смартфон.

Работа содержит 59 страниц, 4 главы, 41 рисунок, 41 источников.

Ключевые слова: препараты; здравоохранение; Android; приложение; клиент-сервер; органайзер;

Abstract

Nowadays, it is essential to take care of man's health and to simplify this process as much as possible. Many people are installing different applications that can help them to solve this problem. The solution described in this paper can help people organize their time and make their lives a little bit easier.

So, this paper describes the android application, which can help people take care of their health more efficiently and organize their drug intake with just a few clicks using their smartphone. Also, private hospitals can use this application to simplify the process of engagement between a doctor and a patient.

The work contains 59 pages, 4 chapters, 41 figures, 41 sources.

Keywords: drugs; healthcare; android; application; client-server; organizer;

Основные определения, термины и сокращения

Клиент-серверное приложение – архитектура приложения, в которой одна часть программы – клиент запрашивает выполнение неких функций у другой программы – сервера.

Android – это операционная система для различных устройств основанная на ядре Linux.

Препарат – сущность, представляющая препарат, имеет название, время и дату начала и конца приема, Bar code, группу пользователей и другие свойства. Имеет связь с сущностями диагноза и пользователя

Диагноз – сущность, представляющая диагноз врача, имеет название и список препаратов для лечения

Пациент – конечный пользователь системы, пациент имеет возможность, просматривать препараты и добавлять их, только для своего аккаунта

Врач – конечный пользователь системы, который имеет возможность, добавлять диагнозы для любого пациента, зарегистрированного в системе

REST API – структура, которая подразумевает доступность различных методов при помощи, соответствующих HTTP запросов.

JSON – текстовое представление, формата данных для передачи на сервер и обратно, данный формат является общепринятым.

JSON WEB TOKEN (JWT) – это открытый стандарт (RFC 7519) для создания токенов доступа, основанный на формате JSON. Как правило, используется для передачи данных для аутентификации в клиент-серверных приложениях.

Java – строго типизированный объектно-ориентированный язык программирования общего назначения, разработанный компанией Sun Microsystems

Spring Framework – Один из самых популярных Фреймворков для реализации серверной части приложения на языке Java

QR-код – машиночитаемый стандартизированный код, содержащий информацию о связанном с ним объекте.

Bar-код – графическая информация наносимая на упаковку изделия или на само изделие, которая предоставляет возможность считывать информацию об изделии при помощи электронных устройств

Группа пользователей – представляет собой атрибут Препарата, который хранит имя человека, которому необходимо принять препарат. Например бабушка, дедушка, дочка и т.д.

Аптечка – представляет собой список Препаратов, имя пользователя, которому принадлежит и некоторые другие атрибуты

Endpoint – Это конечная точка маршрута или цепи взаимодействия клиента с сервером, принимающая и возвращающая необходимые данные

Оглавление

Введение.....	8
Глава 1. Обоснование необходимости разработки клиент–серверного приложения.....	10
1.1. Обзор существующих решений.....	10
1.2. Сравнение разрабатываемого продукта с известными аналогами	12
1.3. Функциональные требования к клиент – серверному приложению	13
Выводы по главе	15
Глава 2. Архитектура клиент – серверного приложения и технологии его разработки	16
2.1. Выбор архитектуры Android приложения.....	16
2.2. Выбор архитектуры серверной части приложения	17
2.3. Выбор архитектуры клиент – серверного приложения.....	18
2.4. Выбор языка программирования для клиент – серверного приложения	19
2.5. Выбор среды разработки	21
2.6. Выбор системы контроля версий	21
2.7. Выбор виртуальной машины и инструментов для работы с ней	21
2.8. Применение паттернов проектирования	23
2.9. Применение сторонних библиотек и модулей.....	24
2.10. Выбор инструментов для тестирования работы серверной части	24
Выводы по главе	25
Глава 3. Реализация приложения	26
3.1. Описание моделей серверной части и Android приложения	26
3.2. Описание структуры базы данных.....	28
3.3. Описание структуры и работы Android приложения	28
3.4.1. Диаграмма классов Android приложения.....	28
3.3.3. Описание структуры Android приложения	29
3.4. Описание структуры и работы серверной части приложения.....	32
3.5.1. Диаграмма классов серверной части приложения	32
3.5.2. Описание структуры серверной части проекта.....	33
3.5.3. Описание авторизации пользователей и структуры JWT	35
3.5.4. Описание запросов серверной части приложения	36
3.4.5. Модуль для проверки препарата на оригинальность.....	40
3.5. Выполненный объем работы	41
3.6. Ссылка на исходный код клиент – серверного приложения	42
Выводы по главе	42
Глава 4. Описание пользовательского интерфейса мобильного приложения.....	43
4.1. Главный экран – лента препаратов	43
4.2. Окно для добавления препарата.....	44
4.3. Экран регистрации.....	45
4.4. Экран авторизации.....	46

4.5.	Сканер штрих кодов	47
4.6.	Проверка препарата на оригинальность	48
4.7.	Уведомления	49
4.8.	Поиск препаратов	50
4.9.	Меню	51
4.10.	Карточка для просмотра информации о препарате	52
4.11.	Профиль	53
	Выводы по главе	54
	Заключение	55
	Список использованных источников	56
	Приложение А. Техническое задание	59
	Приложение Б. Руководство оператора	59
	Приложение В. Программа и методика испытаний.....	59
	Приложение Г. Текст программы	59

Введение

В настоящее время мы даже не можем представить свою жизнь без использования технологий. Именно поэтому было бы неправильно не использовать все преимущества, которые они нам дают, особенно когда они способны сделать нашу жизнь намного комфортнее.

По статистике, люди в России болеют ОРВИ в среднем два раза в год. Также более сорока пяти процентов людей в нашей стране страдают хроническими заболеваниями [1].

Более того, из-за пандемии в 2019 году и вируса «COVID-19». Данный вирус вызывает серьезные последствия для здоровья как вовремя течения болезни, так и после выздоровления пациента. Чтобы не ухудшить свое здоровье, люди должны принимать много лекарств, зачастую очень сложно держать информацию о том какие препараты необходимо принять и когда именно [2].

По мимо этого, некоторые люди, страдающие хроническими заболеваниями, должны принимать лекарства, чтобы предотвратить прогрессирования болезни. Например, людям с диабетом требуется инсулин в среднем два раза в день, если они забудут использовать инсулин, это может быть чрезвычайно опасно для их здоровья.

Кроме того, некоторые люди слишком заняты своей работой, чтобы думать о приеме лекарственных препаратов.

Вот почему я решил сделать приложение, которое поможет людям, следить за приемом лекарственных препаратов, а также отслеживать срок годности уже купленных препаратов и их оставшееся количество. Это приложение поможет многим людям с хроническими заболеваниями предотвратить тяжелые последствия для их здоровья и поможет людям, которые слишком заняты, чтобы отслеживать время приема препаратов.

Цель выпускной квалификационной работы – разработка сервера, на языке Java с использованием Spring Framework, а также мобильного клиента для устройств на базе Android для простого просмотра препаратов, имеющихся у пациентов в наличие, проверки оригинальности препаратов при покупке, а также получения уведомлений о приеме препарата.

Цель данного документа – описать техническую часть работы проекта, описать проблемы, выявленные при разработке и рассмотреть их решения

Для успешной реализации клиент – серверного приложения и написания технической документации были поставлены следующие задачи, а именно:

1. Анализ существующих решений, с целью выявления новых функций, не предложенных на рынке в схожих приложениях;
2. Выявление технических требований к разрабатываемому продукту;
3. Выбор технологий и инструментов реализации;
4. Проектирование архитектуры базы данных;

5. Проектирование архитектуры серверной и клиентской части приложения;
6. Реализация серверной и клиентской части приложения;
7. Развертка сервера на виртуальной машине
8. Написание технической документации

Данный документ содержит 4 главы.

В первой главе текущего документа, проанализирован рынок, подробно рассмотрены решения, уже предложенные на рынке, выявлены их сильные и слабые стороны, а также были выявлены технические требования к разрабатываемому продукту.

Во второй главе подробно рассмотрены инструменты реализации, обоснован выбор в пользу языка программирования и используемых технологий, а также архитектурных решений.

В третьей главе разобраны особенности реализации, структура проекта, а также рассчитан объем выполненной работы.

В четвертой главе представлены экраны пользовательского интерфейса при различных сценариях использования.

Глава 1. Обоснование необходимости разработки клиент–серверного приложения

1.1. Обзор существующих решений

Так как клиент разработан, для устройств на платформе Android, то рассмотрим официальный магазин приложений Google Play. В нем можно найти большое количество различных приложений со схожим функционалом. Для обзора были выбраны наиболее популярные приложения, которые наиболее близки к разрабатываемому продукту и соответствуют предметной области работы.

Напоминание о таблетках и тикер лекарств (MyTherapy) [3]

Приложение лишено возможности добавлять препараты по QR/Bar коду. Соответственно, возможность проверить препарат на оригинальность отсутствует. Пользователю предлагается вводить данные вручную.

Количество скачиваний: более 1 000 000

Преимущества:

- 1) наличие календаря принятых препаратов, где пользователь отмечает выпил ли он тот или иной препарат;
- 2) возможность отслеживать свое самочувствие;
- 3) возможность выгрузить данные о здоровье в отчет и импортировать в формат .xls;
- 4) бесплатное.

Недостатки:

- 1) нет возможности быстрого добавления препаратов по Bar коду;
- 2) нет групп пользователей;
- 3) нет возможности проверить препарат на оригинальность.
- 4) нет мониторинга срока годности препарата
- 5) нет аптечки

Напоминание и трекер таблеток (Medisafe) [4]

Приложение предлагает напоминания о приеме лекарств, добавление контактов врача, группы пользователей, аптечку. Как таков серьезных минусов приложение не имеет, единственными недочетами, на мой взгляд, является отсутствие возможности отслеживать количество оставшихся препаратов в аптечке и отсутствие возможности быстрого добавления препаратов

Количество скачиваний: более 1 000 000

Преимущества:

- 1) весь функционал размещен компактно и интуитивно понятен;
- 2) группы пользователей;

- 3) аптечка;
- 4) возможность добавить контакты врачей;
- 5) бесплатное.

Недостатки:

- 1) нет проверки препарата на оригинальность;
- 2) нет возможности добавления препарата по Bar коду;
- 3) не отслеживается количество оставшихся таблеток.
- 4) нет мониторинга срока годности препарата

Мои таблетки Напоминание и трекер о лекарствах (Mobile Creatures) [5]

Приложение предлагает календарь с препаратами и напоминаниями о их приеме. Приложение обладает минимальным функционалом, существует возможность добавить препарат, добавить фото, установить время приема и получать уведомления. Очень удобная реализация UI приема препаратов в виде календаря.

Количество скачиваний: более 100 000

Преимущества:

- 1) возможность указать время приема препарата относительно приема пищи;
- 2) возможность добавить свою фотографию препарата;
- 3) мониторинг и учет выпитых препаратов;
- 4) реализовано в виде календаря;
- 5) бесплатное.

Недостатки:

- 1) нет групп пользователей;
- 2) нет возможности добавления по Bar коду;
- 3) нет проверки на оригинальность;
- 4) нет аптечки.
- 5) нет мониторинга срока годности препарата

Мои таблетки (level38) [6]

Приложение представлено в виде календаря, с возможностью добавления препаратов и получения уведомлений. Главным минусом является отсутствие какого-либо дополнительного функционала, по мимо того, что описан выше.

Количество скачиваний: более 100 000

Преимущества:

- 1) реализация в виде календаря;
- 2) бесплатное;

Недостатки:

- 1) нет добавления препаратов по Bar коду;
- 2) нет проверки препаратов на оригинальность;
- 3) нет аптечки;
- 4) нет групп пользователей.
- 5) нет мониторинга срока годности препарата

1.2. Сравнение разрабатываемого продукта с известными аналогами

Для проведения детального сравнения было решено выделено несколько ключевых свойств, а именно:

- 1) цена;
- 2) возможность добавить препарат и получить уведомление о его приеме;
- 3) проверка препарата на оригинальность;
- 4) быстрое добавление препаратов при помощи Bar кода;
- 5) группы пользователей;
- 6) выгрузка отчета о самочувствие;
- 7) аптечка;
- 8) отслеживание количества оставшегося препарата;
- 9) мониторинг срока годности препарата;
- 10) проверка препарата на совместимость по действующим компонентам.

Результаты данного сравнения представлены в таблице 1.

Таблица 1. Сравнение решений

Название приложения	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
MyTherapy – Напоминание о таблетках и тикер лекарств	free	+	–	–	–	+	–	–	–	–
Medisafe – Напоминание и трекер таблеток	free*	+	–	–	+	–	+	+	+	–
Mobile Creatures – Мои таблетки Напоминание и трекер о лекарствах	free*	+	–	–	–	–	–	–	–	–
level38 – Мои таблетки	free*	+	–	–	–	–	–	–	–	–
Разрабатываемое приложение	free	+	+	+	+	–	+	+	+	–

free – приложение является полностью бесплатным

free* – базовый функционал приложения является бесплатным, расширенный доступный по подписке

Как мы можем заметить, на основании сравнения, основным конкурентом предложенного приложения, является приложение “Напоминание и тикер таблеток” от “Medisafe”, основным недостатками данного приложения являются: отсутствие возможности быстрого добавления препарата, проверки препаратов на оригинальность. По моему мнению это является важным функционалом, т.к. сокращает время ввода данных пользователем, тем самым экономя его время. Более того во время считывания штрих кода, данные отправляются в валидатор и пользователь будет уведомлен о том, что препарат является оригиналом или репликой.

Фокус при разработке мобильного приложения в рамках данной работы обозначен на возможности мониторинга препаратов, имеющихся в наличие у пользователя, быстрого добавление препаратов и одновременной проверке оригинальности, а также на создании пользователем неограниченного количества групп пользователей с одного устройства

1.3. Функциональные требования к клиент – серверному приложению

Для серверной части выделены и определены следующие функциональные требования:

- Сервер должен описывать все сущности хранящихся в базе данных
- Сервер должен имплементировать авторизацию при помощи технологии JWT
- Сервер должен иметь модуль для проверки оригинальности препарата
- Сервер должен сохранять данные в базу данных
- Сервер должен содержать следующие endpoint-ы:
 1. аутентификация пользователя;
 2. регистрация пользователя;
 3. получение аптечки пользователя;
 4. проверка препарата на оригинальность;
 5. получение списка всех зарегистрированных пользователей;
 6. добавление препарата в аптечку пользователя;
 7. добавление препарата в базу данных;
 8. тестовый, для проверки работоспособности сервера;
 9. получение препарата по штрих коду;
 10. добавление времени приема препарата;
 11. получения полной информации о пользователе, за исключением пароля.
 12. изменение пароля пользователя
 13. удаление информации о пользователе
 14. удаление препарата из базы данных
- Для каждого пользователя должна существовать возможность изменения время сессии

- Сервер должен содержать конфигурацию, для определения прав доступа ролей и endpoint – ов
- Сервер должен хранить и передавать пароли пользователей в зашифрованном виде
- Сервер должен быть развернут на удаленной виртуальной машине и иметь открытый порт для отправки запросов
- Сервер должен генерировать уведомления для отправки клиенту

Для мобильного Android приложения выделены и определены следующие функциональные требования:

- Приложение должно предоставлять возможность авторизации для пациентов;
- Приложение должно предоставлять возможность регистрации для пациентов;
- Приложение должно предоставлять пациенту возможность добавления препарата;
- Приложение должно предоставлять пациенту возможность отсканировать штрих код и заполнить необходимые поля автоматически;
- Приложение должно иметь возможность отправлять штрих код для проверки оригинальности и отображать пользователю результат проверки;
- Приложение должно отправлять пользователю push–уведомления: о времени приема препарата, о том, что таблетки заканчиваются, о том, что у препарата истек срок годности;
- Приложение должно содержать список препаратов добавленных пользователей;
- При нажатии на элемент списка препаратов приложение должно предоставлять всю доступную информацию о препарате;
- Приложение должно иметь возможность выхода из аккаунта;
- Приложение должно иметь возможность отфильтровать аптечку по значению групп пользователей.

Для СУБД выделены и определены следующие функциональные требования:

- Любая сущность хранящиеся в базе данных должна иметь следующие поля: id-уникальный номер объекта, created – дата создания объекта, updated – дата изменения объекта, status – статус объекта (ACTIVE, NOT_ACTIVE, DELETED)
- База данных должна быть в третьей нормальной форме
- Поле barcode таблицы с препаратами должно иметь уникальное значение для каждого элемента

Выводы по главе

В данной главе описаны уже существующие на рынке решения, рассмотрены положительные и отрицательные стороны данных решений, а также выявлены ключевые особенности предложенного приложения. Помимо этого были выявлены и описаны функциональные требования к клиенту, серверу и базе данных.

Следующая глава будет описывать выбранную архитектуру, технологии и библиотеки.

Глава 2. Архитектура клиент – серверного приложения и технологии его разработки

2.1. Выбор архитектуры Android приложения

Для клиентской части программного продукта, разработанного в рамках данной выпускной квалификационной работы, использовалась архитектура Model-View-Controller [7] (рис. 1), сокращённо MVC.

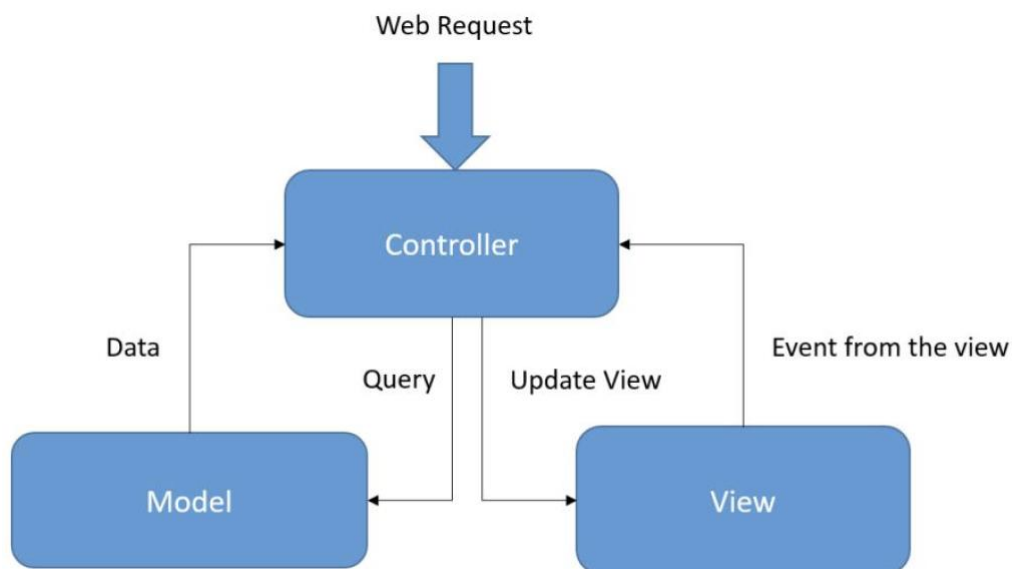


Рис. 1. Архитектура Model-View-Controller

Данное архитектурное решение, имеет достаточно понятную концепцию: Model отвечает за представление данных и изменяет свое состояние, основываясь на командах Controller-a, View отвечает за представление данных пользователю, Controller обрабатывает действия пользователя, и уведомляет Model о необходимости внести изменения. Основным преимуществом предложенной архитектуры, является обособленность данных и представления, что позволяет использовать одни и те же компоненты, для различных задач.

Обосновать выбор в пользу Model-View-Controller позволяют сразу несколько аргументов:

- Данная архитектура является стандартом в разработке мобильных приложений;
- Простой концепт архитектуры;
- Высокая скорость разработки;
- Хорошая читаемость кода, что позволяет стороннему разработчику быстро разобраться в модулях приложения;
- Данная архитектура является достаточно гибкой, что позволяет добавлять необходимый функционал, достаточно быстро;

- В связи с пунктами описанными выше очень легко и быстро возможно внести изменения в работу приложения.

Основной причиной выбора данного шаблона, стала хорошая читаемость кода, хорошая декомпозиция, благодаря чему функционал добавляется крайне быстро, простой концепт, а также распространённость данной архитектуры среди разработчиков мобильных приложений.

2.2. Выбор архитектуры серверной части приложения

В качестве архитектурного решения для серверной части программного продукта, разработанного в рамках данной выпускной квалификационной работы, использовалась архитектура Representational State Transfer [8], сокращенно REST (Рис. 2).

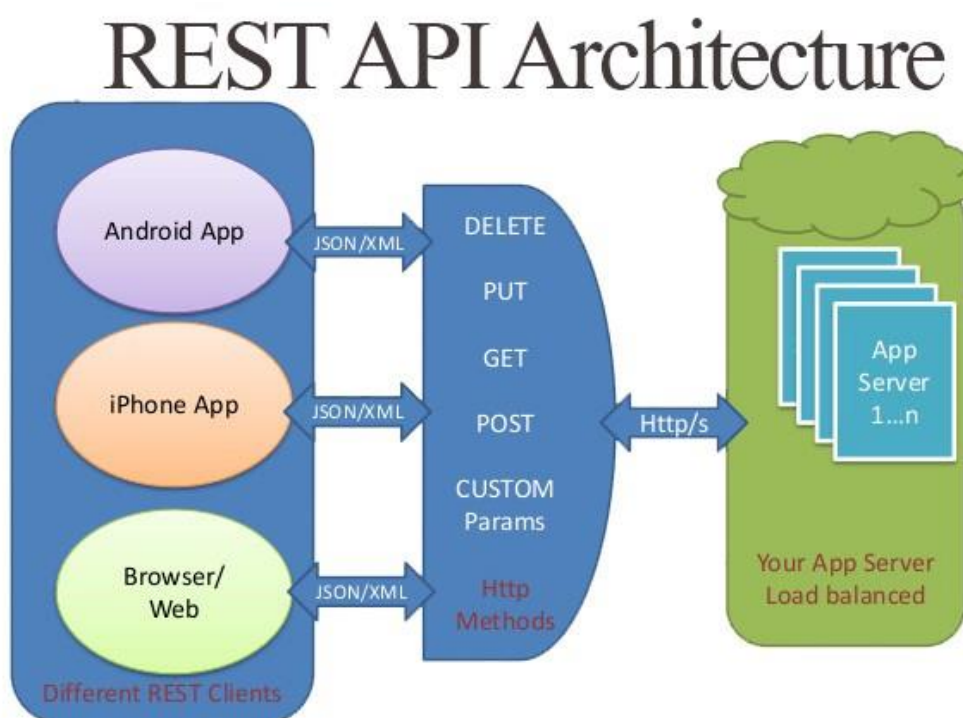


Рис. 2. Архитектура Representational State Transfer

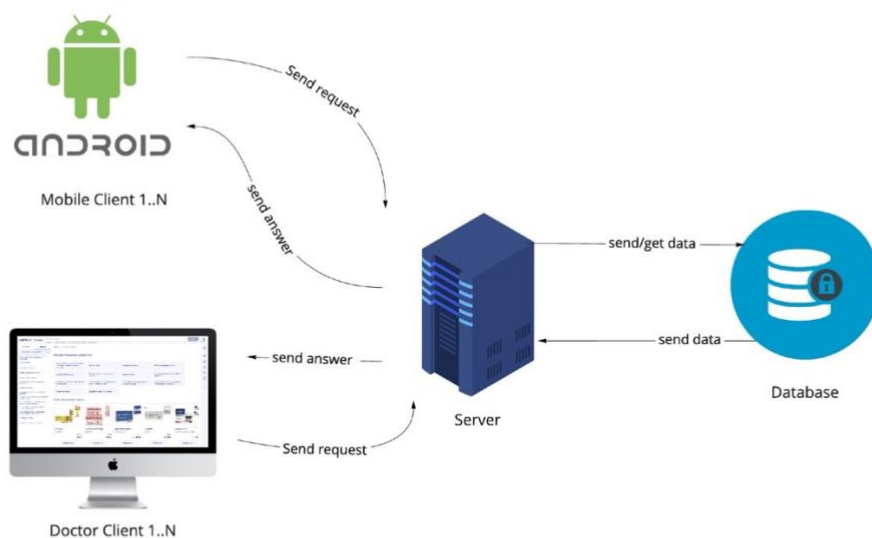
Выбор данной архитектуры был очевиден, так как веб сервер потенциально должен уметь обрабатывать запросы с различных устройств, а именно мобильных клиентов и Web клиентов. REST как раз является технологией, которая поддерживается на подавляющем числе устройств, поскольку для работы с ним система должна иметь возможность отправлять и принимать HTTP запросы а также обрабатывать JSON объекты. По мимо этого каждый метод определен строго заданной URL, что делает REST менее ресурсоемким, по сравнению с его аналогами.

Подводя итоги, в пользу REST архитектуры выступило 4 очень важных фактора, а именно:

- практически любая система способна работать с REST сервером;
- REST является менее ресурсоемким чем его аналоги;
- REST является очень простым интерфейсом и не требует дополнительных внутренних надстроек;
- осведомленность о данной архитектуре среди подавляющего числа backend разработчиков.

2.3. Выбор архитектуры клиент – серверного приложения

В качестве архитектуры приложения была выбрана клиент – серверная архитектура (рис. 3), так как приложение подразумевает по собой серверную часть, базу данных, множество Android клиентов и потенциальное добавление Web клиента для врача. Все данные обрабатываются сервером, клиенту остается лишь получить и отобразить уже готовые данные



miro

Рис. 3. Схема клиент – серверной архитектуры

В пользу выбора клиент-серверной архитектуры выступило несколько ключевых факторов:

- сервер мощнее устройства с которого уходят запросы, следовательно приложение будет в разы производительней;
- нет дублирования кода, в отличие от случая если бы сервера не было;

- данные защищены, так как прямой доступа у пользователя к персональным данным отсутствует, он ограничен лишь списком доступных ему endpoint-ов, для различных ролей.

2.4. Выбор языка программирования для клиент – серверного приложения

Для разработки приложений для Android доступны два языка программирования: **Java** [9] и **Kotlin** [10].

Kotlin и **Java** – это в первую очередь взаимодополняющие языки. Несомненно, некоторые функции лучше реализованы в Java, а некоторые в Kotlin. Несмотря на это, оба языка полностью совместимы. Kotlin использует JVM экосистему и библиотеку, это позволяет просто пользоваться Java методами и классами. Ниже представлена табличка с основными преимуществами и недостатками обоих языков программирования.

Таблица 2. Результаты сравнения Java и Kotlin

Свойство	Kotlin	Java
Корутины и дополнительные потоки Пояснение: Android является однопоточным из-за чего в случае вызова, например метода для запроса данных у сервера, блокируется основной поток, не давая возможности использовать пользовательский интерфейс.	В языке Kotlin предоставляется возможность создавать корутины и сопрограммы, которые позволяют использовать меньше памяти в сравнении с обычным потоком.	Java же создает фоновый поток из основного интенсивного потока. Но грамотное управление данными потоками усложняет процесс разработки.
Простое создание классов данных	Есть возможность создания классов, в которой при помощи специальной нотации data будут заданы поля для хранения в базе данных, геттеры, сеттеры, конструкторы и остальные методы класса Object языка Java.	В Java есть возможность подключить библиотеку Lombok и нотацию data и использовать ее. Без подключения библиотеки это не предоставляется возможным.
Расширение классов	Есть	Есть

Лямбда-выражения	Есть	Есть
Поддержка более одного конструктора	В дополнение к первичному конструктору - классы Kotlin, могут обладать одним или несколькими второстепенными конструкторами, путем добавления в объявления класса второстепенных конструкторов.	Не реализована
Отсутствие необходимости выявления исключений	Условия для проверяемых исключений в коде Kotlin отсутствуют, благодаря чему объявлять или отлавливать исключения больше нет необходимости.	Необходимо обрабатывать исключения вручную, в данном случае это скорее плюс
Неявные преобразования	В Kotlin не реализовано, он требует выполнить именно явное преобразование.	В Java имеется поддержка неявных преобразований, которые позволяют меньшим типам преобразовываться в большие.
Статические элементы	Не имплементированы	Имплементированы
Тернарный оператор	Не имплементирован	Тернарный оператор в Java выполняет функции базового оператора if, и содержит условие оценивающееся как false или true.
Подстановочные знаки	Не имплементировано	Имплементировано
Сообщество	Трудно будет найти ответы на все интересующие вопросы, в связи с тем, что язык относительно новый и разработчиков, пишущих программы для Android на нем не так много.	Java достаточно старый язык, его знает практически любой программист пишущий под Android и к тому же, по нему есть множество официальной документации. Что значительно упрощает процесс изучения этого языка.
Компетентность	Очень низкая	Очень высокая

Для разработки клиентской и серверной части приложения требовалось выбрать язык на котором будет написаны обе части приложения. В таблице 2 приведено сравнение обоих языков по различным параметрам.

Основной причина выбора языка Java, в качестве языка разработки для мобильного приложения, стало наличие опыта разработки на данном языке, отсутствие серьезных

преимуществ у языка Kotlin, а также наличие большого сообщества у языка Java, что помогает решить различные проблемы, возникающие по мере разработки, гораздо быстрее.

В качестве языка программирования для серверной части также был выбран язык Java, по причинам, которые описаны выше. Помимо этого обмен данными между приложениями значительно упрощен, так как данные имеют одинаковые типы, например класс Date, что позволяло сериализовать данные без дополнительной обработки.

2.5. Выбор среды разработки

Для разработки серверной части приложения использовалась последняя версия IntelliJ IDEA Ultimate [11] от компании JetBrains на момент написания данного текста это версия 2021.1. Выбор был сделан в пользу данной среды разработки, так как она является самой популярной и имеет большое сообщество, следовательно проще найти ответы на все интересующие вопросы.

Для разработки клиентской части приложения использовалась последняя версия Android Studio [12] от компании Google на момент написания данного текста это версия 4.1.3. Выбор был сделан в пользу данной среды разработки, так как она является официальной средой для разработки приложений на Android, также Android Studio сделана на основе IntelliJ IDEA, что упрощает взаимодействие с ней, в связи с наличием опыта работы в средах разработки от JetBrains. Помимо этого Android Studio имеет большое сообщество, следовательно проще найти ответы на все интересующие вопросы.

2.6. Выбор системы контроля версий

В качестве системы контроля версий использовался GitHub. Выбор был сделан в пользу GitHub [13], поскольку существует возможность автоматического запуска тестов при деплое – сервис GitHub Actions [14].

2.7. Выбор виртуальной машины и инструментов для работы с ней

Виртуальная машина необходима для того, чтобы развернуть сервер и иметь возможность удаленного доступа со сторонних устройств, в нашем случае это мобильный клиент. Для целей разработки и тестирования приложения подходит абсолютно любая виртуальная машина, на операционной системе Linux [15] имеющая более 1 центрального процессора и статический IP адрес. В качестве виртуальной машины был выбран сервис Azure VM [16] (Рис. 4), так как они предоставляют кредиты в размере 100\$ для учеников НИУ ВШЭ [17] и удовлетворяют вышеперечисленным условиям

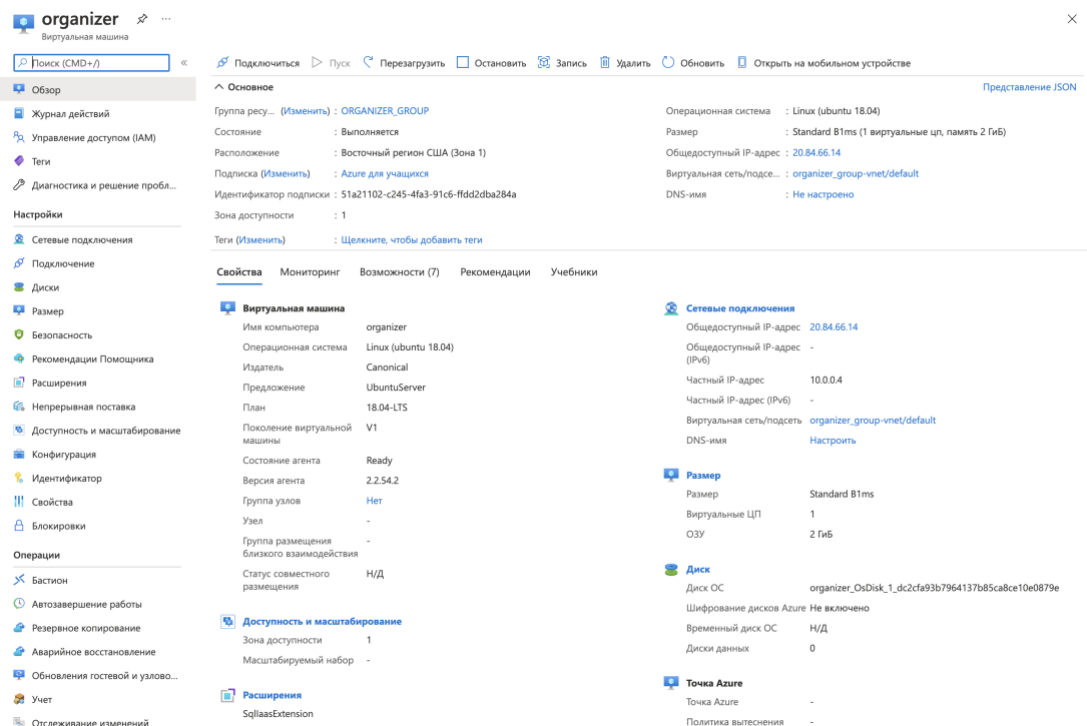


Рис. 4. Azure VM

Подключение и настройки виртуальной машины: установка библиотек, запуск приложения в режиме сервиса, работа с портами происходила через терминал. Подключение происходит через терминал по публичному ssh ключу [18] (Рис. 5.)

```
drdolgosheev@MacBook-Pro-Dmitrij Azure % ssh -i organizer_key.pem drdolgosheev@20.84.66.14
Welcome to Ubuntu 18.04.5 LTS (GNU/Linux 5.4.0-1039-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Sat Apr 24 11:24:22 UTC 2021

System load:  0.0          Processes:    120
Usage of /:   31.0% of 28.90GB Users logged in:  0
Memory usage: 76%         IP address for eth0: 10.0.0.4
Swap usage:   0%

 * Pure upstream Kubernetes 1.21, smallest, simplest cluster ops!

    https://microk8s.io/

 * Canonical Livepatch is available for installation.
   - Reduce system reboots and improve kernel security. Activate at:
     https://ubuntu.com/livepatch

37 packages can be updated.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

*** System restart required ***
Last login: Wed Apr 14 14:09:52 2021 from 81.200.23.142
drdolgosheev@organizer:~$
```

Рис. 5. Подключение к VM

Загрузка необходимых файлов на удалённую рабочую машину происходила через инструмент FileZilla [19] (Рис. 6)

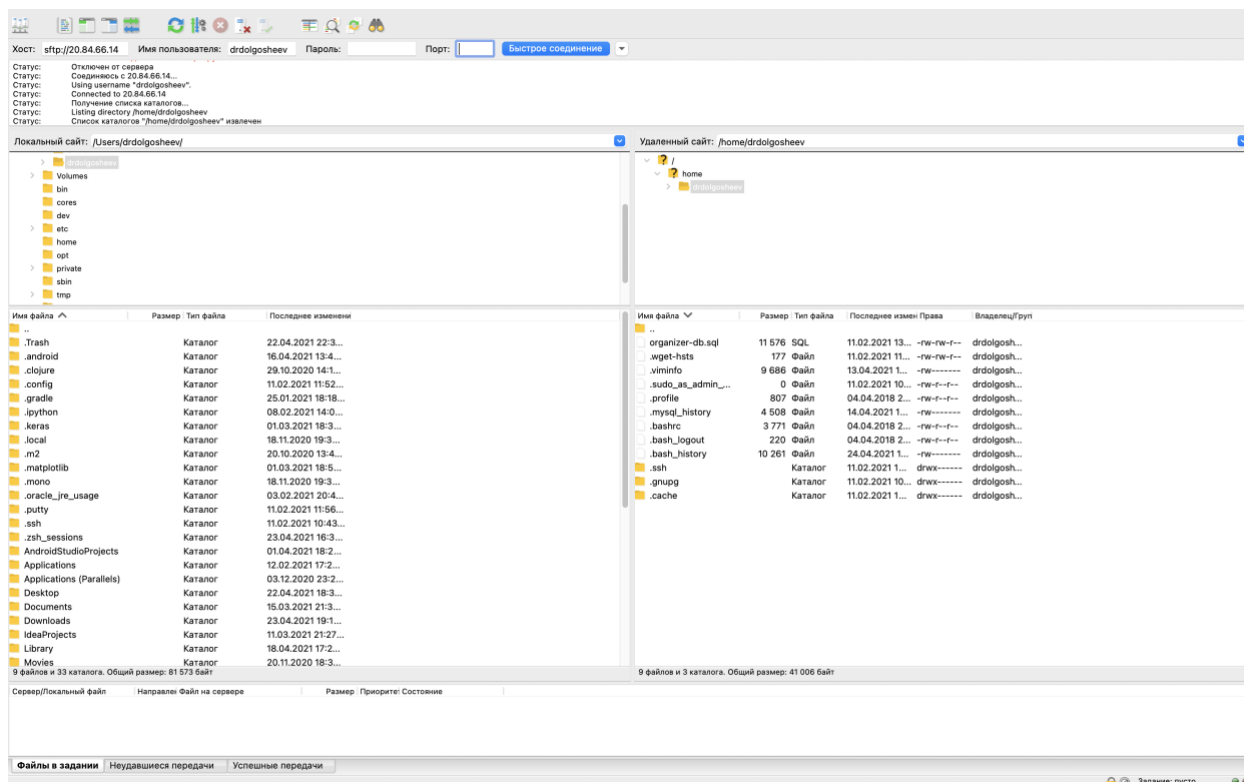


Рис. 6. FileZilla

2.8. Применение паттернов проектирования

Для ускорения и оптимизации разработки в рамках выпускной квалификационной работы использовались различные паттерны:

- Singleton [20] – паттерн, относящийся к порождающим, он гарантирует что у класса будет только один объект и доступ к этому классу будет глобальный. В предложенном проекте данный паттерн применялся для класса инициализации сервера, так как мы должны гарантировать что точка входа всех данных в базу данных будет только одна, что помогает избежать конфликтов с сохранением данных, пришедших от клиентов.
- Factory [21] – паттерн, относящийся к порождающим, он предоставляет возможность подклассам интерфейс для создания экземпляров некоторых классов. Это позволяет манипулировать абстрактными объектами на более высоком уровне, а не использовать специфичные классы, что сокращает размер кода и повышает читаемость. В предложенном проекте, данный паттерн использовался в классе генерирующем токен JWT из класса пользователя, внося всю необходимую информацию в токен.
- Prototype [22] – паттерн, относящийся к порождающим, он позволяет копировать объекты любой сложности без привязки к их конкретным классам. В предложенном проекте данный паттерн использовался для качественного сокращения строк кода и повышения его читаемости, а именно все сущности дополняли базовую сущность, содержащую

уникальный идентификатор объекта, статус объекта, дату последнего обновления и дату последнего создания.

2.9. Применение сторонних библиотек и модулей

Для сервера использовались множества сторонних библиотек, являющихся стандартом в разработке серверов на Java а именно:

- библиотека Java Spring Boot [23] - это набор уже настроенных модулей, которые работают в Spring Framework [24]. Эти модули упрощают конфигурацию приложения, написанного на Spring. В предложенном приложении использовались различные нотации, а также sl4j logger [25] – представляющий собой инструмент для логирования.
- Java Spring Security [26] – для упрощения процесса создания авторизации, регистрации на основе JWT.
- библиотека от Google[27] – gson [28], которая представляет собой парсер данных формата JSON.
- для использования специальных нотаций использовался Lombok [29], это помогло уменьшить количество строк кода, а также повысить читаемость кода.
- для подключения базы данных использовался MySql Connector [30]

В клиентской части приложения использовалась:

- okhttp[31] – для отправки запросов на сервер
- FireStorage[32] – сервис от Google, использовался для хранения изображений
- библиотека от Google – gson, которая представляет собой парсер данных формата JSON.
- Модуль для сканирования QR/BAR кодов от стороннего неизвестного разработчика

2.10. Выбор инструментов для тестирования работы серверной части

В качестве инструментов для тестирования использовались Unit тесты, инструмент Junit4, а также ручное тестирование, при помощи инструмента Postman [33] (Рис. 7).

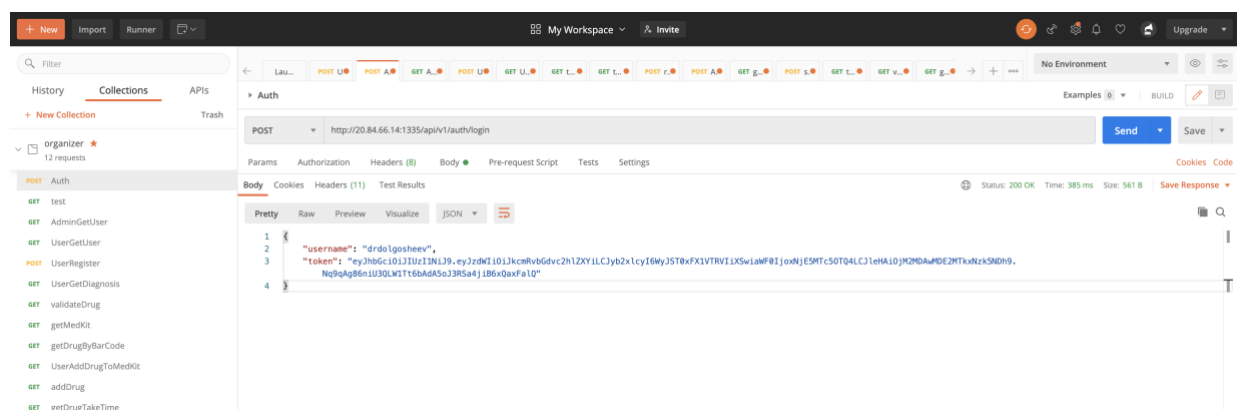


Рис. 7. Postman

Выводы по главе

В данной главе подробно рассказано про используемые архитектурные решения, паттерны проектирования, обоснован выбор в пользу выбранного языка программирования, описаны инструменты тестирования, а также остальные различные технологии, которые были использованы в процессе разработки.

Глава 3. Реализация приложения

3.1. Описание моделей серверной части и Android приложения

Серверная часть приложения, для хранения данных в базе данных используют следующие модели:

- BaseEntity – сущность от которой наследуются все остальные сущности, используемые в приложении, содержит уникальный номер сущности, дату создания, дату изменения, статус (удалена, не активна, активна);
- DateDrugs – сущность для хранения списка дат приема препарата пользователем;
- Diagnosis – сущность для хранения диагнозов пользователей, сущность диагноза содержит: название, описание, список препаратов, имя пользователя, которому назначен диагноз, и врач который поставил данный диагноз, а также список препаратов необходимых для приема;
- Doctor – сущность описывает модель врача и содержит: имя пользователя в системе, имя, фамилию, электронную почту и пароль;
- Drug – сущность описывающая модель препарата. Данная модель содержит все необходимые поля описывающие препарат, в частности: список пользователей которые добавили данный препарат, дату завершения срока годности, количество таблеток в упаковке и бар код;
- DrugType – представляет собой класс типа нумераторов, содержащий группы препаратов;
- RoleType – сущность представляющая собой список ролей, например ROLE_ADMIN, ROLE_USER. Используются для предоставления пользователям различных прав доступа;
- User – сущность описывающая модель пользователя, содержит: логин, пароль, имя, фамилию, электронную почту, а также поля для связи с другими таблицами.

Для передачи данных с клиента на сервер и наоборот используются специальные структуры для передачи данных (Data Transfer Objects):

- AuthenticationRequestDto – Необходим для авторизации, содержит логин и пароль пользователя;
- BarcodeDto – необходим для выполнения запросов связанных с поиском, изменением, удалением препарата в аптечке пользователя по штрих коду;
- BooleanDto – необходим для передачи ответов от сервера, которые в качестве ответа используют единственное поле типа Boolean;
- ChangePasswordDto – необходим для передачи и получения данных связанных с процедурой смены пароля;

- DrugForShare – содержит полную информацию о препарате, для передачи клиенту, при запросе;
- GetMedKitDto – необходим для корректной передачи списка препаратов добавленных пользователем в формате List of JSON's;
- IntegerDto – необходим для передачи данных клиенту, содержит единственное поле number типа Integer;
- RegisterUserRequestDto – необходим для отправки клиента, обработки сервером процесса регистрации нового пользователя;
- ResponseLoginDto – необходим для отправки клиенту токена, после успешной авторизации.
- StringDto – необходим для отправки сервером ответов, содержащих строчное значение;
- TakeTimeDto – необходим для отправки списка даты и времени приема препарата;
- AddDrugDto – необходим для добавления препаратов напрямую в базу данных;
- AddDrugToMedKitDto – необходим для добавления препарата в аптечку пользователя.

3.2. Описание структуры базы данных

База данных была, представляет собой реляционную базу данных и находится в третьей нормальной форме. Построена UML диаграмма (Рис. 8), для простоты визуального восприятия

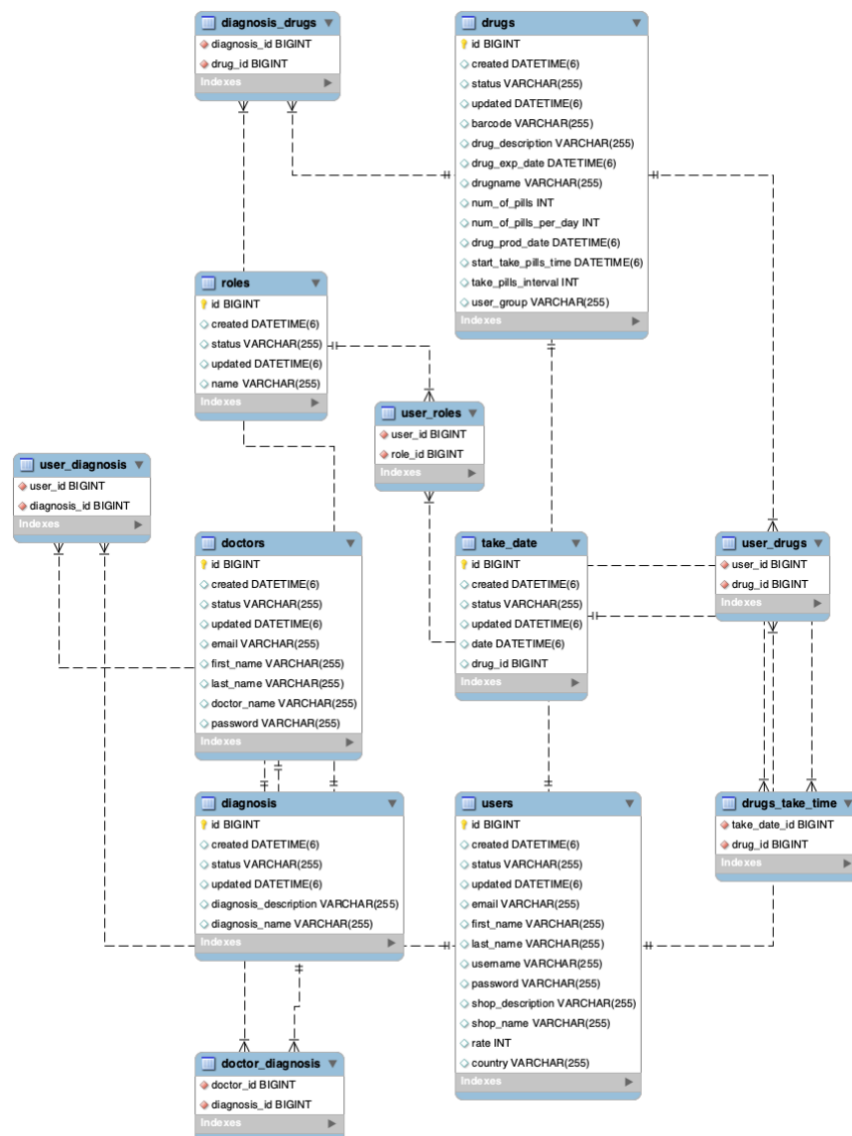


Рис. 8. UML диаграмма базы данных

База данных включает в себя таблицы-сущности: User, Roles, Doctors, Diagnosis, Drugs, а также таблицы для связи данных сущностей, это необходимо для соблюдения третьей нормальной формы, что позволяет устранить из базы данных избыточность и в последствии помогает избежать проблем с логически ошибочными результатами выборки или изменения данных.

3.3. Описание структуры и работы Android приложения

3.4.1. Диаграмма классов Android приложения

Из-за декомпозиции классов диаграмма получилось слишком громоздкой (Рис. 9), в связи с чем не предоставляется возможным, изучить ее подробно в данном документе, по этой причине данная диаграмма сохранена в корневой папке репозитория с клиентской частью проекта.

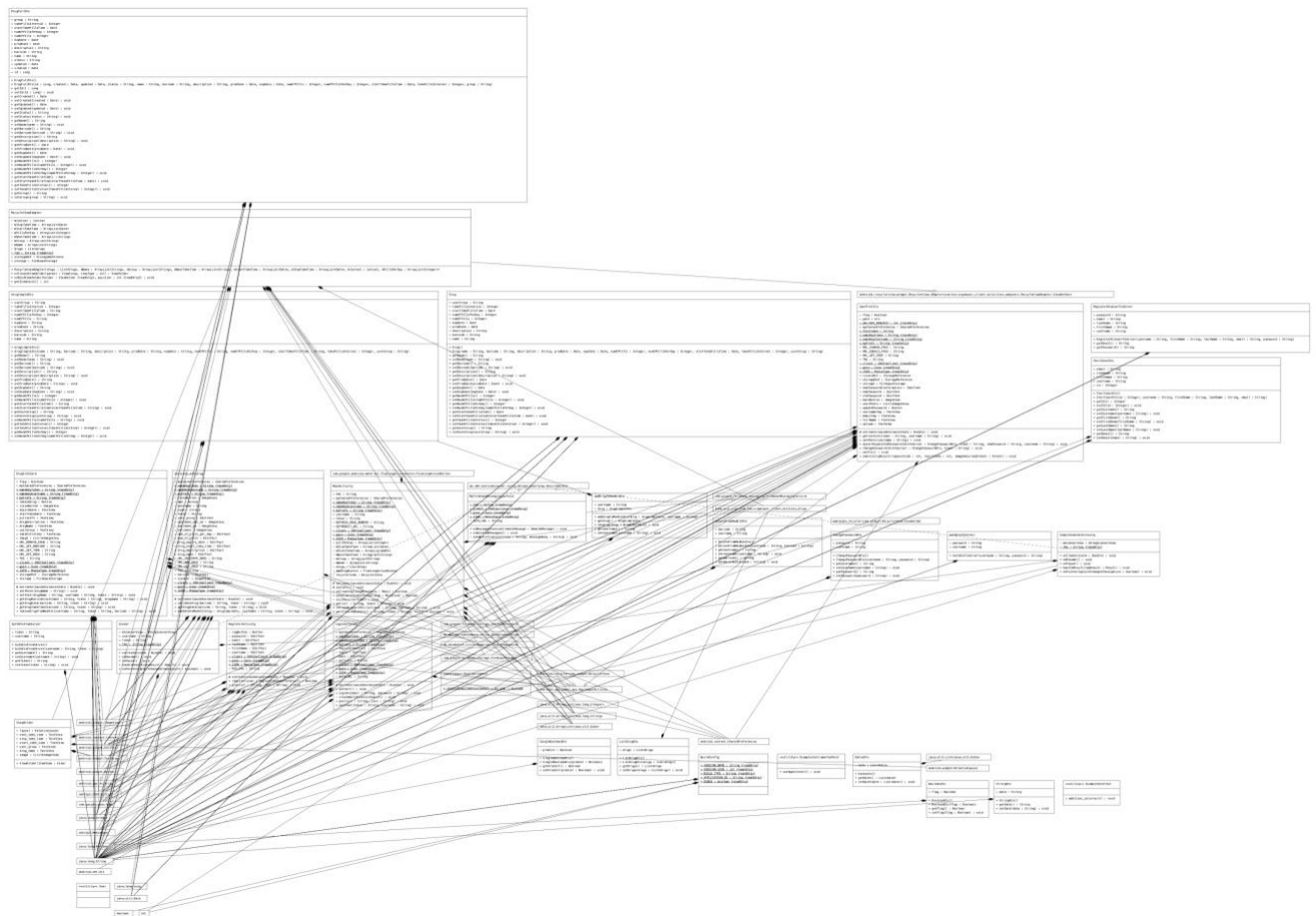


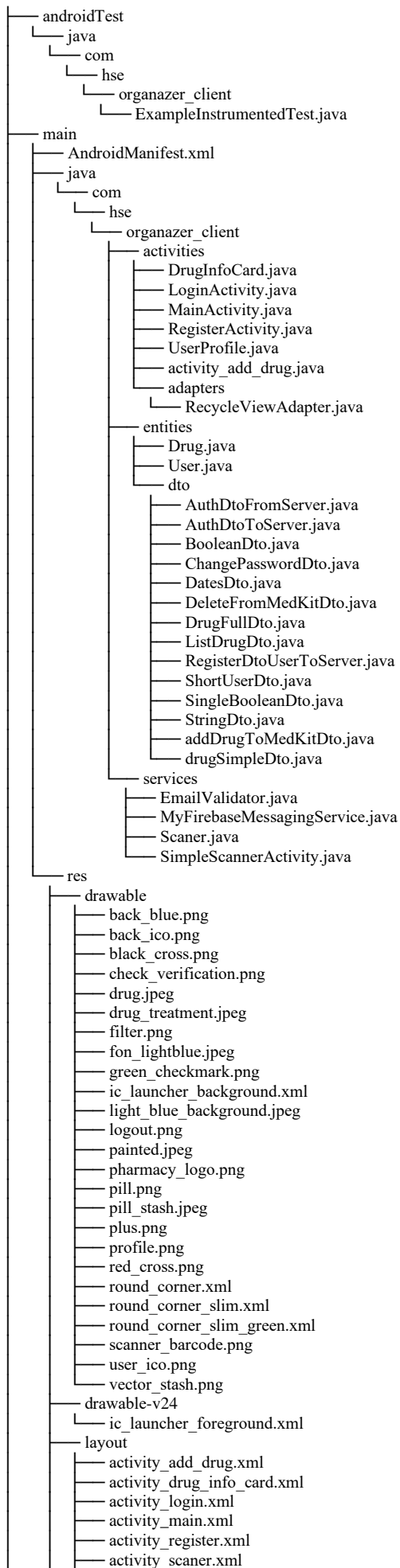
Рис. 9. Диаграмма классов клиентской части

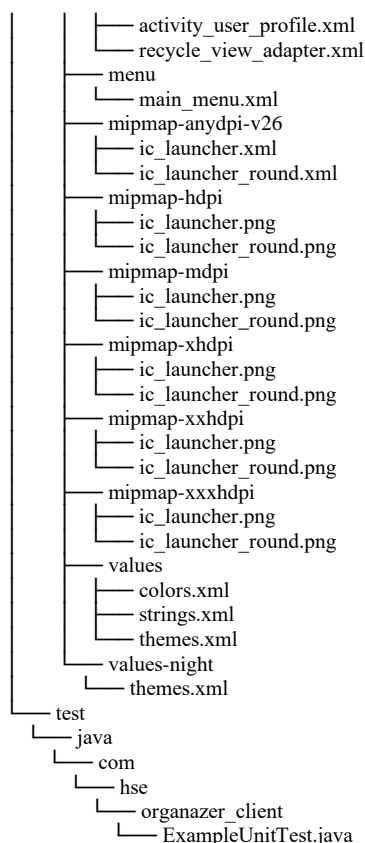
3.3.3. Описание структуры Android приложения

В корневой папке директории расположены следующие файлы:

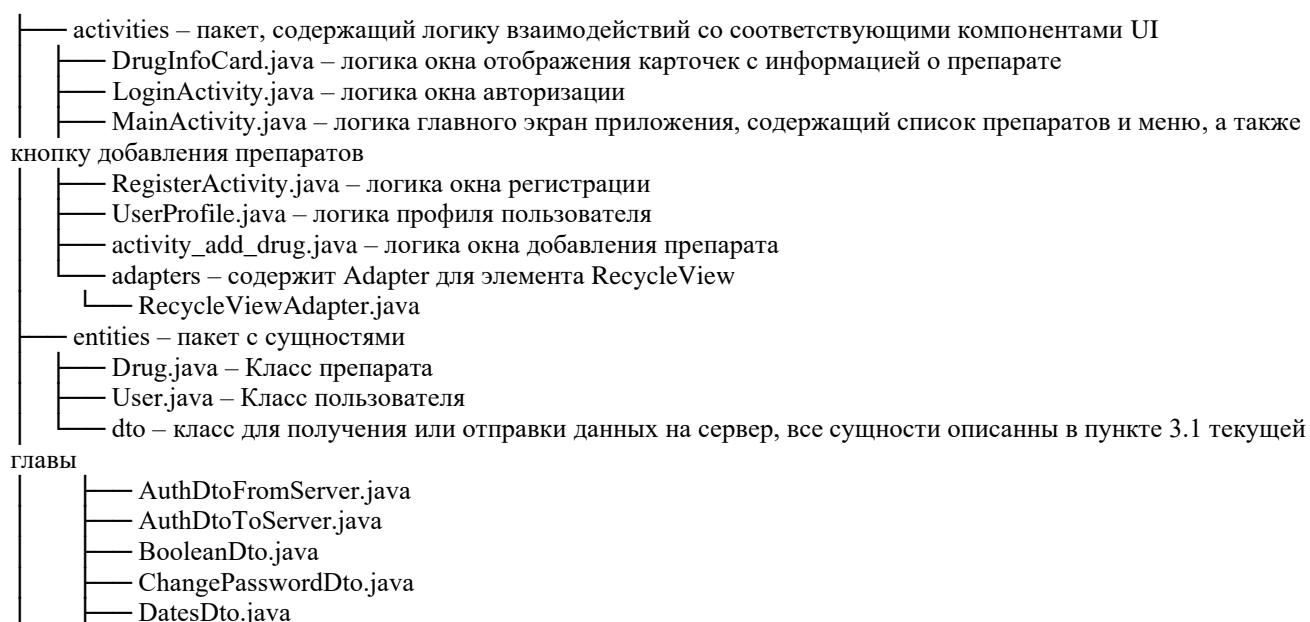
- HELP.md – информация о настройке проекта, для первого запуска
- gradle – директория для управление зависимостями проекта
- gradlew.bat – исполняемый файл для загрузки зависимостей
- app – папка содержащая код проекта
- README.md – описание проекта
- build.gradle - исполняемый файл для загрузки зависимостей и настройки конфигураций проекта
- gradlew - исполняемый файл для загрузки зависимостей
- settings.gradle – файл для конфигураций запуска gradlew.bat

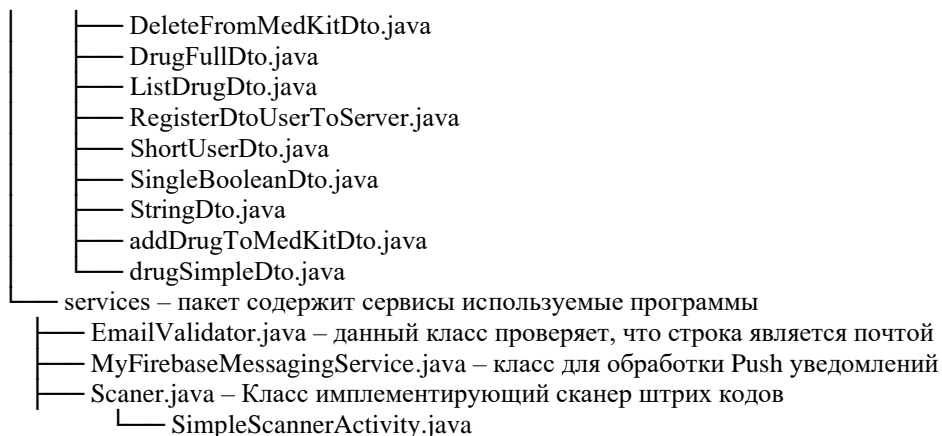
Далее рассмотрим структуру папки app/src/, содержащей все ключевые файлы приложения





Как мы видим, директория `src` содержит четыре основных пакета. Пакеты `androidTest` и `test` дынные пакеты содержат файлы необходимые для тестирования приложения. Пакет `res` содержит данные, отвечающие за графический интерфейс: файлы верстки, изображения, темы, стили и т. д. Пакет `main` содержит файл `AndroidManifest.xml`, он отвечает за конфигурацию приложения, настройки активности и разрешений приложения. Также данный пакет содержит директорию `main/java/com/hse/organizer_client`, данный пакет содержит Java классы, отвечающие за логику, получения отправки и обработки данных, введенным пользователем, рассмотрим данную директорию поподробней.





3.4. Описание структуры и работы серверной части приложения

3.5.1. Диаграмма классов серверной части приложения

Из-за декомпозиции классов диаграмма получилось слишком громоздкой, в связи с чем не предоставляется возможным, изучить ее подробно в данном документе, по этой причине данная диаграмма сохранена в корневой папке репозитория с серверной частью проекта. Так же представлен фрагмент (Рис. 10), дающий понимания о том что изображено на диаграмме (Рис. 11)



Рис. 10. Диаграмма классов серверной части

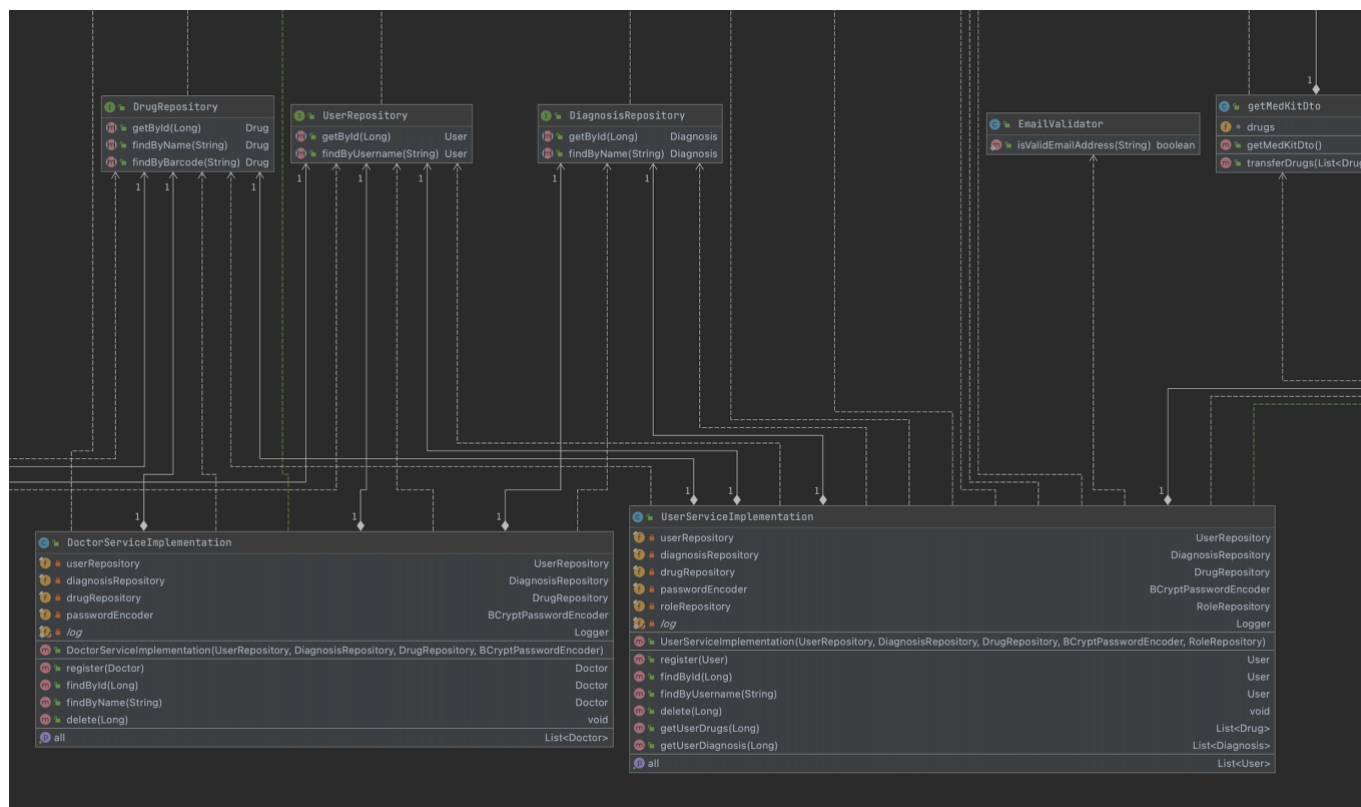


Рис 11. Фрагмент диаграммы классов серверной части

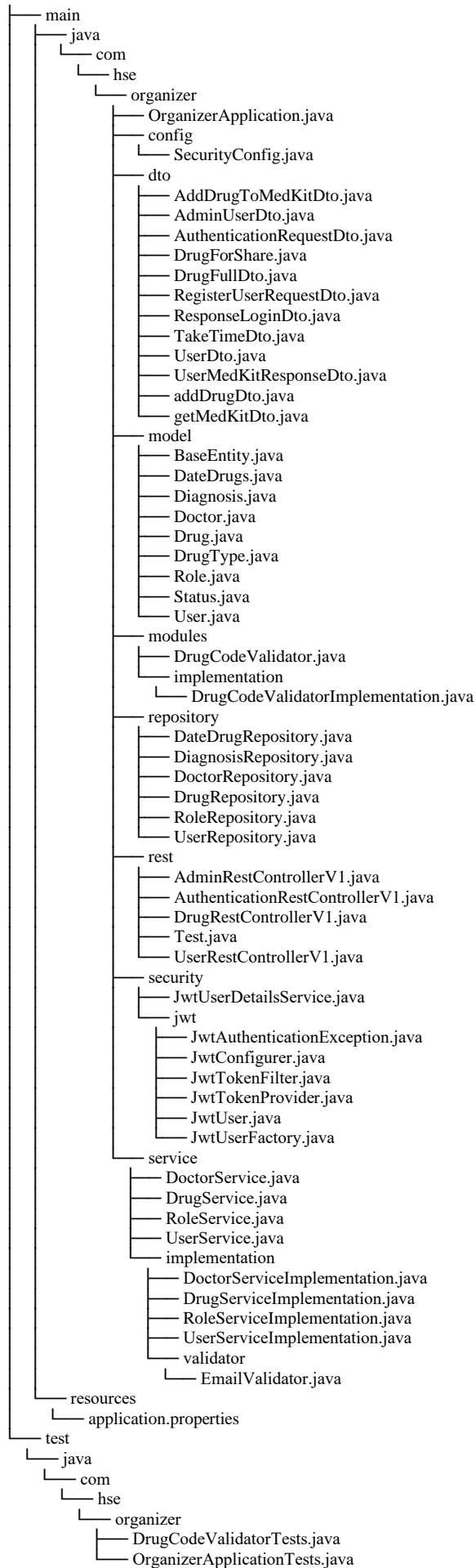
Для генерации диаграмм данных использовалась встроенный в IntelliJ IDEA плагин, а именно UML [34].

3.5.2. Описание структуры серверной части проекта

В корневой паке директории расположены следующие файлы:

- HELP.md – информация о настройке проекта, для первого запуска
- build – служебная директория с файлами создающимися после сборки проекта
- gradle – директория для управление зависимостями проекта
- gradlew.bat – исполняемый файл для загрузки зависимостей
- organizer-server-uml.png – диаграмма классов
- src – папка содержащая код проекта
- README.md – описание проекта
- build.gradle - исполняемый файл для загрузки зависимостей и настройки конфигураций проекта
- gradlew - исполняемый файл для загрузки зависимостей
- lib – папка со сторонними библиотеками проекта
- settings.gradle – файл для конфигураций запуска gradlew.bat

Далее рассмотрим структуру папки src, содержащей все написанные мною классы



Как мы можем видеть из диаграммы в директории `src` содержится 2 пакета, а именно `main` и `test`, в пакете `test` – содержаться все Unit тесты для приложения. В пакете `main` – содержаться исполняемые Java классы.

В директории `main`, содержатся 2 папки: `resources` и `java`. Папка `resources` содержит один единственный файл, а именно `application.properties`, данный файл содержит настройки для подключения базы данных, константное время жизни JWT токена, а также кодовое слово, для его расшифровки.

В директории `main` содержится директории `com > hse > organizer` в ней располагается файл для запуска сервера `OrganizerApplication`, а также директории с кодом:

- `config` – в ней содержится файл, который описывает возможности перехода по различным endpoint-ам: не авторизованных пользователей, пользователей имеющих `ROLE_USER`, а также имеющий роль `ROLE_ADMIN`;
- `dto` – в этой директории содержатся сериализуемые оболочки для отправки данных на сервер, и получения данных от клиента;
- `model` – в этой директории находятся все сущности хранящиеся в базе данных, подробнее с сущностями можно ознакомиться в главе 3.1. текущего документа;
- `modules` – данный пакет содержит интерфейс и имплементацию валидатора препаратов;
- `repository` – данный пакет содержит CRUD [35] классы – репозитории для различных сущностей, с их помощью мы взаимодействуем с данными хранящимися в базе данных;
- `rest` – данный пакет содержит описание всех доступных endpoint-ов;
- `security` – классы находящиеся в данном пакете, реализуют аутентификацию пользователей, с использованием технологии JWT;
- `service` – данный пакет содержит интерфейсы и их имплементации, описывающие всю логику взаимодействия с базой данных, а именно обработку данных, сохранение данных, логирование, удаление данных, изменение данных.

3.5.3. Описание авторизации пользователей и структуры JWT

Авторизация пользователей происходит при помощи технологии JWT, она позволяет:

- Конфигурировать содержимое токена
- Регулировать время жизни токена авторизации
- Позволяет передавать данные в зашифрованном виде

Выбор был сделан, в пользу технологии JWT, а не сторонних библиотек с готовой авторизацией, так как это повышает читаемость кода, упрощается кастомизацию передаваемых данных и позволяет сторонним разработчикам быстрее разобраться и перенастроить авторизацию при необходимости. Ниже мы рассмотрим структуру JWT токенов в предложенном приложении. (рис 12)

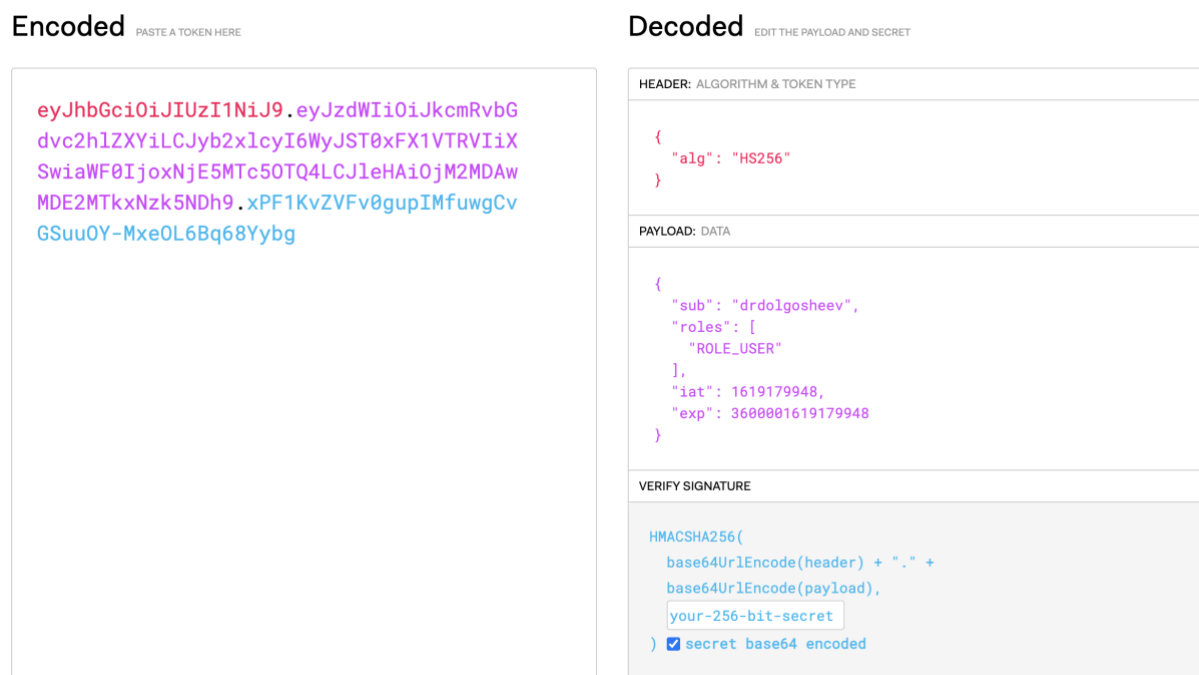


Рис. 12. Структура JWT токена

Заголовок токена содержит, алгоритм шифрования. Кодовое слово для декодирования токена храниться в файле `application.properties`, что исключает возможность расшифровки токена сторонними лицами В поле дата токена находится, имя пользователя, список ролей, а также время жизни токена в миллисекундах, когда у токена заканчивается время жизни, используя его пользователь не сможет получать данные от сервера, а также обновлять данные. Для продолжения работы пользователю необходимо получить новый токен, посредством авторизации. Добавления пользователя с ролью отличной от `ROLE_USER`, доступно только из терминала или вручную при помощи SQL консоли [36], по умолчанию зарегистрированному через мобильный клиент пользователю присваивается `ROLE_USER`.

3.5.4. Описание запросов серверной части приложения

Сервер представляет собой REST сервис, соответственно работает на основе HTTP и поддерживает следующие типы запросов: GET, POST, DELETE, PUT.

Запросы содержат заголовки [37] – в нашем случае `Authorization: Bearer_ + token`, где `token` это уникальный токен сессии пользователя, а также некоторые стандартные заголовки (Рис. 13)


Params	Authorization	Headers (7)	Body	Pre-request Script	Tests	Settings
Headers  Hide auto-generated headers						
KEY	VALUE					
<input checked="" type="checkbox"/> Postman-Token ⓘ	<calculated when request is sent>					
<input checked="" type="checkbox"/> Host ⓘ	<calculated when request is sent>					
<input checked="" type="checkbox"/> User-Agent ⓘ	PostmanRuntime/7.26.5					
<input checked="" type="checkbox"/> Accept ⓘ	*/*					
<input checked="" type="checkbox"/> Accept-Encoding ⓘ	gzip, deflate, br					
<input checked="" type="checkbox"/> Connection ⓘ	keep-alive					
<input checked="" type="checkbox"/> Authorization	Bearer_eyJhbGciOiJIUzI1NiIsInR5cGE6YWV0IjklcmRvbGdvc2hlZXkiLCJyb2xlcyl6WyJST0x0FX1VTRVliXSwiaWF0IjoxNjE4NzQ5N					

Рис. 13. Заголовки запроса

По мимо этого каждый запрос имеет HTTP status [38] – что отображает код выполнения запроса (Рис. 14)

Hits by Response Code		
Code 200 - OK	15.45%	7527
Code 206 - Partial Content	0.13%	64
Code 301 - Moved Permanently	0.03%	14
Code 302 - Found	0.18%	90
Code 304 - Not Modified	83.46%	40656
Code 403 - Forbidden	0.00%	1
Code 404 - Not Found	0.72%	353
Code 500 - Internal Server Error	0.02%	10

Рис. 14. Коды http ответов

Также каждый запрос содержит тело (Рис. 15) в формате JSON, структура всех возможных объектов для обработки сервером описаны в пакете DTO – Data Transfer Objects [39]

```

1  {
2      "username": "drdolgosheev1",
3      "firstName": "Dmitriy",
4      "lastName": "Dolgosheev",
5      "email": "test@mail.ru",
6      "password": "test"
7  }
```

Рис. 15. Пример тела запроса

Если запрос возвращает список объектов, то каждый объект выделен в квадратный скобках или фигурных скобках (Рис. 16)

```

{
  "name": "Peach oil",
  "id": 17,
  "barcode": "4627074841092",
  "description": "Peach oil is a viscous liquid obtained by cold pressing the seeds that are inside the seeds of the fruit of plants, followed by filtration.",
  "prodDate": "2021-04-10T00:00:00.000+00:00",
  "expDate": "2021-03-10T00:00:00.000+00:00",
  "numOfPills": 24,
  "numOfPillsPerDay": 3,
  "startTakePillsTime": "2020-03-10T00:00:00.000+00:00",
  "takePillsInterval": 3,
  "userGroup": "Mother"
},
{
  "name": "Panavir",
  "id": 18,
  "barcode": "4605370002485",
  "description": "Antiviral and immunomodulatory agent of plant origin, polysaccharides of Solanum tuberosum shoots. The main active ingredient is a hexose glycoside, consisting of glucose, rhamnose, arabinose, mannose, xylose, galactose, uronic acids.",
  "prodDate": "2021-04-10T00:00:00.000+00:00",
  "expDate": "2020-03-10T00:00:00.000+00:00",
  "numOfPills": 100,
  "numOfPillsPerDay": 5,
  "startTakePillsTime": "2021-03-10T00:00:00.000+00:00",
  "takePillsInterval": 5,
  "userGroup": "Me"
},
{
  "name": "Nurofen express",
  "id": 19,
  "barcode": "5000158106123",
  "description": "NSAIDs, a derivative of phenylpropionic acid. It has anti-inflammatory, analgesic and antipyretic effects.",
  "prodDate": "2020-04-10T00:00:00.000+00:00",
  "expDate": "2023-03-10T00:00:00.000+00:00",
  "numOfPills": 16,
  "numOfPillsPerDay": 1,
  "startTakePillsTime": "2020-03-10T00:00:00.000+00:00",
  "takePillsInterval": 1,
  "userGroup": "Me"
}
}

```

Рис. 16. Список объектов

Ниже приведены все доступные API сервера, в качестве заголовков для всех запросов необходим заголовок Authorization, для всех endpoint-ов кроме: login, register, test.

Таблица 4. Методы REST API

URL	Тип	Тело запроса (POST)	Аргументы URL (GET)	Назначение	Возвращает
/api/v1/auth/login	POST	AuthUserDto	—	авторизация пользователя	http status, message, body: username, token
/api/v1/auth/register	POST	RegUserDto	—	регистрация пациента	http status, message, body: User
/test	GET	—	—	Проверка работоспособности сервера	http status, message, body: String = "test"
/api/v1/admin/users/{id}	GET	—	{id} – id пользователя	Получение информации о пользователе администратором	http status, message, body: User
/api/v1/user/get/diagnosis/{id}	GET	—	{id} – id диагноза	Получение диагноза по id пользователя	http status, message, body: Diagnosis
/api/v1/drug/validate/{barcode}	GET	—	{barcode} – штрих код препарата	Проверка препарата на оригинальность	http status, message, body: Boolean
/api/v1/user/get/medKit/{username}	GET	—	{username} – имя пользователя	Возвращает список препаратов,	http status, message, body: List<DrugDto>

				добавленных пользователем	
/api/v1/drug/getDrugByBarcode/{barcode}	GET	–	{barcode} – штрих код препарата	Возвращает препарат	http status, message, body: List<DrugDto>
/api/v1/drug/addDrugToMedKit	POST	UserAddToMedKitDto: String username, Drug drug	–	Добавляет препарат в аптечку пользователя	http status, message, body: String = “OK”
/api/v1/drug/addDrug	POST	DrugDto	–	Добавляет препарат в базу данных	http status, message, body: String = “OK”
/api/v1/drug/getDrugsTakeTime/{barcode}	GET	–	{barcode} – штрих код препарата	Возвращает время приема препаратов	http status, message, body: List<Date>
/api/v1/user/get/diagnosis/{username}	GET	–	{username} – имя пользователя	Получения списка диагнозов по id пользователя	http status, message, body: List<Diagnosis>
/api/v1/drug/recountNumberOfPills/{barcode}	GET	–	{barcode} – штрих код препарата	Пересчитывает количество оставшихся таблеток	http status, message, body: Integer number
/api/v1/user/deleteDrugFromMedKit	POST	DeleteFromMedKitDto: String username, String barcode	–	Удаляет препарат из аптечки пользователя	http status, message, body: Boolean flag
/api/v1/drug/getBarcodeByName/{drugname}	GET	–	{drugname} – имя препарата	Возвращает препарат по имени	http status, message, body: DrugDto dto
/api/v1/user/checkPasswordEquals	POST	ChangePasswordDto: String username, String password	–	Проверяет совпадение заданного пользователем пароля с текущим, необходимо для смены пароля	http status, message, body: Boolean flag
/api/v1/user/changePassword	POST	ChangePasswordDto: String username, String password	–	Шифрует и сохраняет пароль в базу данных	http status, message, body: Boolean flag

/api/v1/user/getUser/{username}	GET	{username} – имя пользователя	–	Ищет и возвращает сжатую информацию по пользователя по полю username	http status, message, body: ShortUserDto dto
---------------------------------	-----	-------------------------------------	---	---	--

3.4.5. Модуль для проверки препарата на оригинальность

Серверная часть приложения содержит модуль для проверки препарата на оригинальность при помощи штрих кода (Рис. 17), текстовое описание алгоритма было взято с сайта Рос. Потреб. [40] (Рис. 18) и имплементировано на языке Java



Рис. 17. Структура штрих кода

Последняя цифра служит для проверки подлинности штрихкода, это так называемая контрольная сумма. Чтобы проверить штрихкод на подлинность, нужно выполнить ряд арифметических операций:

1. Сложить все цифры, стоящие в штрихкоде на четных местах и умножить это число на 3;
2. Сложить все цифры, стоящие на нечетных местах кроме последней цифры (контрольной суммы);
3. Далее нужно сложить результаты (1) и (2) и отбросить десятки, т.е. оставить от полученной суммы последнюю цифру;
4. Вычесть из 10 результат (3) и сравнить его с контрольной суммой. Если значения совпадают - все в порядке, иначе штрихкод поддельный, либо контрольная сумма вычислена неверно.

Рис. 18. Текстовое описание алгоритма проверки оригинальности препарата

Также существуют готовые сервисы для проверки штрих кодов препаратов на оригинальность (Рис. 19), выбор был сделан в пользу имплементации алгоритма на сервере предложенного приложения так как:

- Большинство из них не имеют API;
- Нет знания о том, какой мощностью обладает тот или иной хостинг сайта, следовательно выдержит ли он большую нагрузку, в виде множественных запросов к одному endpoint-у, неизвестно;

РАСШИФРОВКА ШТРИХКОДА

Данный сервис позволяет узнать страну-производителя товара по его штрихкоду (EAN-13) и проверить штрихкод на подлинность.

Введите 13-значный код штрихкода:

Проверить

Рис. 19. Сторонний сервис для проверки препарата на оригинальность

3.5. Выполненный объем работы

При помощи утилиты Cloc [41], было получено общее количество строк в директории src, серверной и клиентской части проекта (Рис. 20 и Рис. 21). Данная директория не содержит автоматически сгенерированных файлов и сторонних или стандартных библиотек.

```
[drdolgosheev@MacBook-Pro-Dmitrij src % cloc .
  60 text files.
  60 unique files.
  1 file ignored.

github.com/AlDanial/cloc v 1.88 T=0.09 s (653.2 files/s, 35640.8 lines/s)
-----
Language             files            blank          comment           code
-----
Java                  59              634            386             2199
-----
SUM:                  59              634            386             2199
-----
```

Рис. 20. Вычисление строк кода в серверной части

```
[drdolgosheev@MacBook-Pro-Dmitrij src % cloc .
  50 text files.
  49 unique files.
  1 file ignored.

github.com/AlDanial/cloc v 1.88 T=0.08 s (582.9 files/s, 46834.4 lines/s)
-----
Language             files            blank          comment           code
-----
Java                  29              411            96             2161
XML                   20              93             15             1161
-----
SUM:                  49              504            111             3322
-----
```

Рис. 21. Вычисление строк кода в клиентской части

- **Java** – 88 файлов, 4360 строк кода;
- **XML** – сюда входят преимущественно файлы разметки – 20 файлов, 1161 строка кода;

Стоит отметить, что комментарии и пробелы также не учитываются. Если учитывать комментарии и пробелы выходит суммарно 7156 строки.

3.6. Ссылки на исходный код клиент – серверного приложения

Ссылка на репозиторий с исходным кодом мобильного Android приложения:
<https://github.com/drdolgosheev/organizer-client>.

Ссылка на репозиторий с исходным кодом серверной части приложения:
<https://github.com/drdolgosheev/organizer-server>

Выводы по главе

В данной главе рассмотрены особенности реализации сервера и клиента, структура базы данных, подсчитано число файлов и строк кода в проекте, рассмотрена файловая структура проекта. Также указаны ссылки на репозитории в GitHub содержащие исходный код.

Глава 4. Описание пользовательского интерфейса мобильного приложения

4.1. Главный экран – лента препаратов

Главный экран содержит список добавленных препаратов и кнопку для добавления новых препаратов. В каждой карточке препарата можно увидеть следующую информацию: название, группу пользователя, ближайшее время приема, дату начала приема, и дату окончания срока годности препарата (Рис 22).



Рис. 22. Главный экран, список препаратов

4.2. Окно для добавления препарата

В окне для добавления препарата можно увидеть следующие элементы: текстовые поля для: названия препарата, описания препарата, даты начала приема в формате dd.mm.yyyy, дату окончания срока годности в аналогичном формате, поле для количества таблеток в упаковке, количество приемов в день, группы пользователей и штрих кода, кнопку для сканирования штрих кода при помощи камеры, кнопку для отправки номера штрих кода на сервер и его валидации, кнопка добавления препарата в аптечку (Рис. 23).

The screenshot displays a mobile application interface titled "Drug intake organizer". The main form is titled "Add drug" and contains the following fields and controls:

- Drug name:** A text input field with the placeholder "Drug name".
- Description:** A larger text input field with the placeholder "Description".
- Start take date:** A date picker showing "10.03.2000".
- Expire date:** A date picker showing "10.03.2004".
- Number of pills on box:** A numeric input field showing "30".
- Number of pills per day:** A numeric input field showing "4".
- User group:** A text input field showing "Mother".
- Bar code numbers:** A text input field showing "3282770072815".
- Barcode icon:** A small icon representing a barcode.
- Validate:** A button with a checkmark icon and the text "Validate".
- Result:** A label next to the "Validate" button.
- ADD:** A green button at the bottom of the form.

The form is overlaid on a background showing a list of existing drugs with columns for "Start take date" and "Expire date". The bottom of the screen shows a standard Android navigation bar with icons for home, back, and recent apps.

Рис. 23. Окно для добавления препарата

4.3. Экран регистрации

В окне для добавления препарата можно увидеть следующие элементы: текстовые поля для: имени, фамилии, псевдонима, электронной почты, пароля и подтверждения пароля, а также кнопка для отправки данных на сервер (Рис. 24).

17:56 B. ... 33%

Drug intake organizer

Name

Surname

Username

Email

Password

Confirm password

REGISTER

Рис. 24. Окно регистрации

4.4. Экран авторизации

Экран содержит поля для ввода псевдонима пользователя, а также пароля (Рис 25). При отсутствии текста в одном из полей, запрос на сервер отправлен не будет (Рис 26). При некорректном вводе логина или пароля, сервер вернет ответ с соответствующей ошибкой (Рис 27).

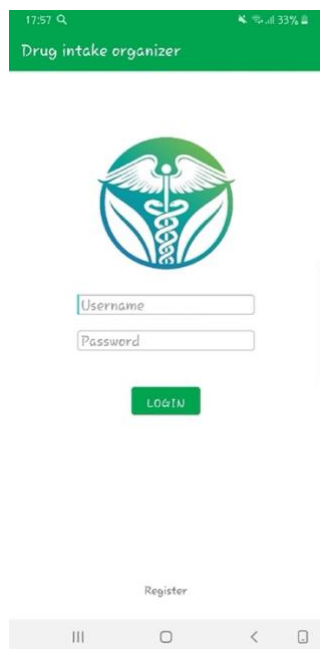


Рис. 25. Окно авторизации

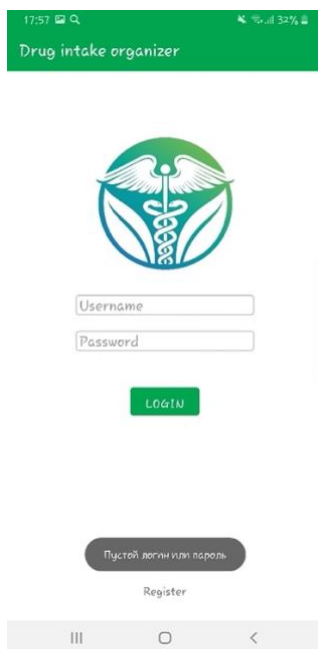


Рис. 26. Пустой логин или пароль

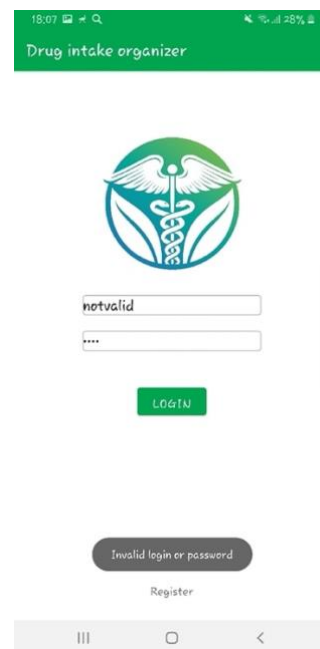


Рис. 27. Некорректный логин или пароль

4.5. Сканер штрих кодов

При наведении на штрих код, окно автоматически закроется и передаст данные о типе кодировки и цифрах находящимся под штрих кодом клиенту (Рис. 28).

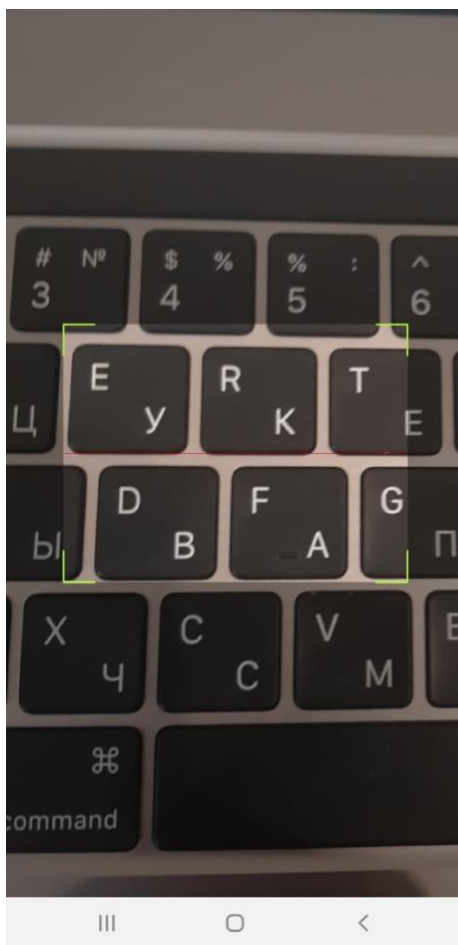


Рис. 28. сканер штрих кодов

4.6. Проверка препарата на оригинальность

После вызова сканера данные о кодировке сохраняются внутри программы, а код под штрих кодом заполняется автоматически, при необходимости пользователь может вписать данные вручную или изменить их. После нажатия кнопки “Validate” запрос уходит на сервер, в случае если препарат является оригинальным, и он есть в базе данных, поля: имя препарата, описания препарата и количество таблеток заполняется автоматически (Рис. 29). В случае если препарат не оригинальный, то выводится соответствующее сообщение (Рис. 30)

The screenshot shows the 'Drug intake organizer' app interface. At the top, there's a green header with the text 'Drug intake organizer'. Below it, the drug name 'Nurofen express' is entered in a text field. A description box contains the text: 'NSAIDs, a derivative of phenylpropionic acid. It has anti-inflammatory, analgesic and antipyretic effects.' Below the description, there are two date pickers: 'Start take date' set to '10.03.2000' and 'Expire date' set to '10.03.2004'. There are also two numeric input fields: 'Number of pills on box' set to '16' and 'Number of pills per day' set to '4'. The 'User group' is set to 'Mother'. A 'Bar code numbers' field contains '5000158106123'. At the bottom, there is a 'Validate' button with a checkmark icon. Below the button, the text 'Result: ✓' is displayed, and a dark grey button with the text 'Drug is original' is visible.

Рис. 29. Препарат оригинальный

The screenshot shows the 'Drug intake organizer' app interface. At the top, there's a green header with the text 'Drug intake organizer'. Below it, the drug name 'Nurofen express' is entered in a text field. A description box is empty. Below the description, there are two date pickers: 'Start take date' set to '10.03.2000' and 'Expire date' set to '10.03.2004'. There are also two numeric input fields: 'Number of pills on box' set to '30' and 'Number of pills per day' set to '4'. The 'User group' is set to 'Mother'. A 'Bar code numbers' field contains '5000158106122'. At the bottom, there is a 'Validate' button with a checkmark icon. Below the button, the text 'Result: ✗' is displayed, and a dark grey button with the text 'Drug is NOT original' is visible.

Рис. 30. Препарат не оригинальный

4.7. Уведомления

Сервер высылает уведомление о необходимости приема препарата или о необходимости купить новую упаковку препарата, а клиент получает, обрабатывает и отображает их, в строке с логотипом миниатюры (Рис. 31), и при открытии диспетчера уведомлений, с текстом, заголовком, именем приложения, большой иконкой (Рис. 32).



Рис. 31. Уведомления миниатюра

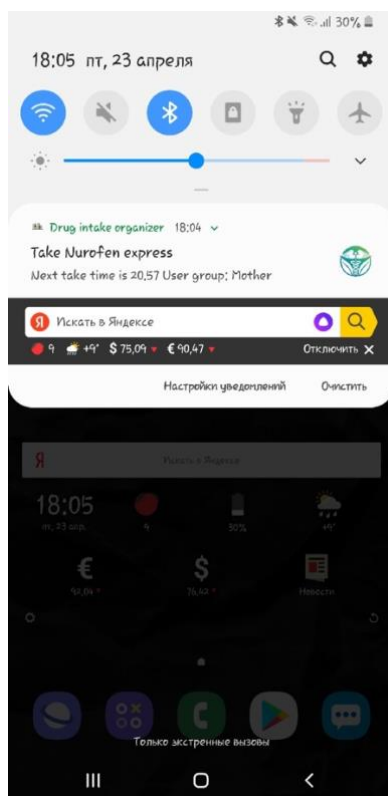


Рис. 32. Уведомление большое

4.8. Поиск препаратов

Поиск препаратов осуществляется при помощи поисковой строки (Рис. 33), пользователь может найти группу препаратов, принадлежащих конкретному члену семьи (Рис. 34). Либо найти препарат по названию (Рис. 35)

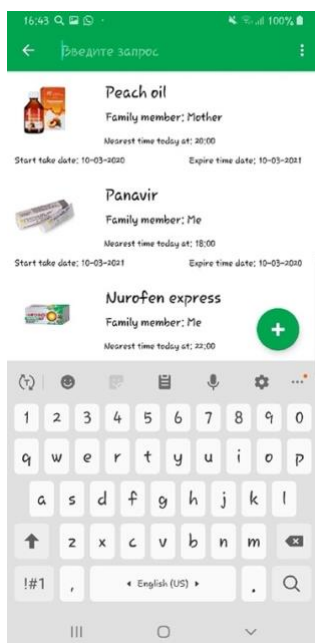


Рис. 33. Поисковая строка



Рис. 34. Поиск по члену семьи



Рис. 35. Поиск по названию препарата

4.9. Меню

Главный экран приложения содержит кнопку, для открытия меню и содержит поля для выхода из аккаунта и перехода в профиль пользователя (Рис. 36).



Рис. 36. Меню

4.10. Карточка для просмотра информации о препарате

Главный экран приложения содержит список препаратов, при нажатии на карточку появляется окно с информацией о соответствующем препарате также есть возможность удалить его из аптечки (Рис. 37).

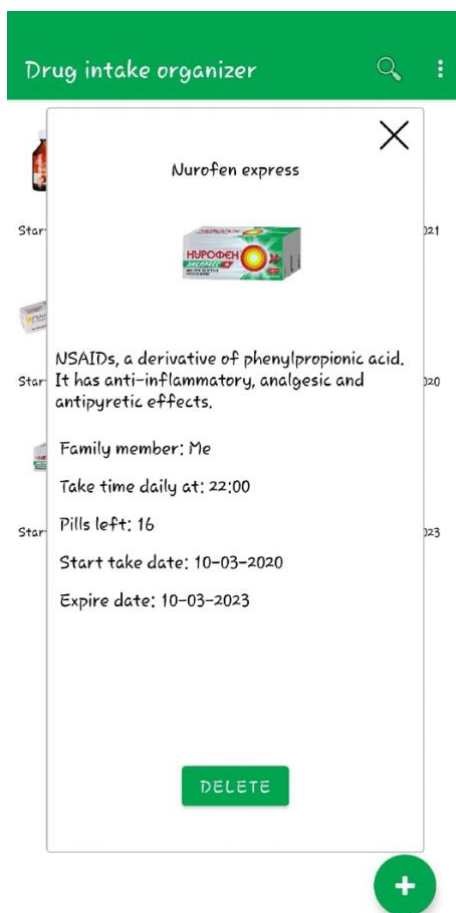


Рис. 37. Информация о препарате

4.11. Профиль

В профиле пользователь может ознакомиться с информацией о своей почте, имени, фамилии, имени пользователя, установить изображение профиля, а также поменять свой текущий пароль. (Рис. 38). Если старый пароль указан неверно, пользователь получит соответствующее предупреждение (Рис. 39). Если пароль в поле для подтверждения не совпадает, с указным ранее, пользователь получит предупреждение (Рис. 40). В случае если пароль обновлен успешно, пользователь будет уведомлен об этом (Рис. 41).

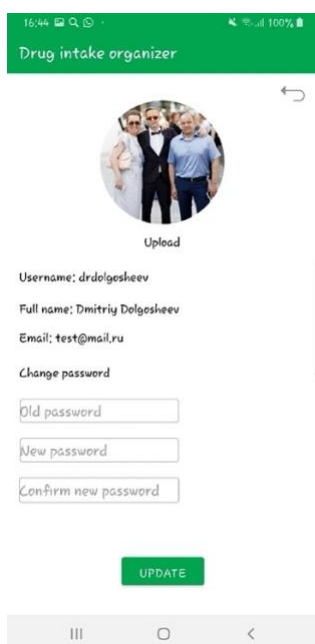


Рис. 38. Профиль



Рис. 39. Некорректный
пароль

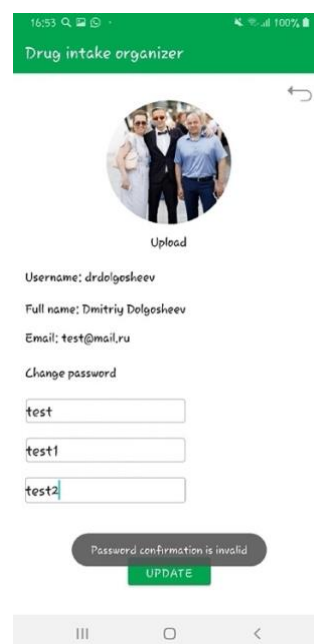


Рис. 40. Пароли не
совпадают



Рис. 41. Пароль изменен

Выводы по главе

В данной главе подробно рассмотрены все экраны пользовательского интерфейса при различных сценариях использования.

Заключение

Результатом данной выпускной квалификационной работы является разработанное клиент – серверное приложение для упрощения приема лекарственных препаратов.

В первой главе данного документа проанализирован рынок, подробно разобраны конкуренты, выявлены ключевые особенности предложенного приложения, а также были выявлены функциональные требования

Во второй главе обосновывается выбор того или иного архитектурного решения, описываются используемые паттерны проектирования, обосновывается выбор языка программирования и описываются все сторонние программы и библиотеки, которые были использованы при создании предложенного приложения.

В третьей главе описаны особенности реализации как Android приложения, так и серверной части приложения.

В четвертой главе рассмотрен пользовательский интерфейс Android приложения, при различных сценариях использования.

По результатам разработки программного продукта в рамках Выпускной Квалификационной Работы были получены следующие результаты:

1. Выделены функциональные требования к разрабатываемому продукту, составлено техническое задание;
2. Проанализированы и выбраны наиболее подходящие технологии, для реализации приложения
3. Имплементирован сервер, подключены все необходимые зависимости, подключена база данных;
4. Сервер развернут на удаленной машине, а также виртуальная машина настроена, для получения запросов из внешних источников
5. Разработан и протестирован код мобильного приложения;
6. Написана вся необходимая техническая документация, а именно: Project Proposal, текст Выпускной Квалификационной Работы, Руководство Оператора, Программа и Методика Испытаний, а также Текст Программы.

В качестве дальнейшего развития проекта предполагается имплементация Web UI для врача, улучшения мобильного клиента, добавление фильтров, совместимостей препаратов, подключение платного API для получения актуальных данных для всех препаратов, разработка IOS приложения, а также добавления карты с аптеками в которых можно купить необходимые препараты.

Список использованных источников

1. Минздрав: 45% россиян имеют хронические заболевания [Электронный ресурс] // <https://aif.ru/> URL: <https://aif.ru/society/healthcare/1414177> (Дата обращения: 18.04.2021)
2. Названы все осложнения после коронавируса [Электронный ресурс] // <https://www.mk.ru/> URL: <https://www.mk.ru/social/2020/06/26/nazvany-vse-oslozhneniya-posle-koronavirusa.html> (Дата обращения: 18.04.2021)
3. Напоминание о таблетках и трекер лекарств [Электронный ресурс] // <https://play.google.com/> URL: <https://play.google.com/store/apps/details?id=eu.smartpatient.mytherapy&hl=ru&gl=US> (Дата обращения: 18.04.2021)
4. Напоминания и трекер таблеток [Электронный ресурс] // <https://play.google.com/> URL: <https://play.google.com/store/apps/details?id=com.medisafe.android.client&hl=ru&gl=US> (Дата обращения: 18.04.2021)
5. Мои таблетки Напоминание и трекер о лекарствах [Электронный ресурс] // <https://play.google.com/> URL: <https://play.google.com/store/apps/details?id=mobilecreatures.pillstime&hl=ru&gl=US> (Дата обращения: 18.04.2021)
6. Мои Таблетки [Электронный ресурс] // <https://play.google.com/> URL: <https://play.google.com/store/apps/details?id=com.devsoldiers.calendar.pills.limit&hl=ru&gl=US> (Дата обращения: 18.04.2021)
7. Model-View-Controller [Электронный ресурс] // <https://ru.wikipedia.org/> URL: <https://ru.wikipedia.org/wiki/Model-View-Controller> (Дата обращения: 18.04.2021)
8. Representational state transfer [Электронный ресурс] // <https://habr.com/> URL: <https://habr.com/ru/post/38730/> (Дата обращения: 18.04.2021)
9. Java [Электронный ресурс] // <https://ru.wikipedia.org/> URL: <https://ru.wikipedia.org/wiki/Java> (Дата обращения: 18.04.2021)
10. Kotlin [Электронный ресурс] // <https://ru.wikipedia.org/> URL: <https://ru.wikipedia.org/wiki/Kotlin> (Дата обращения: 18.04.2021)
11. IntelliJ IDEA Ultimate [Электронный ресурс] // <https://www.jetbrains.com/> URL: <https://www.jetbrains.com/ru-ru/idea/> (Дата обращения: 18.04.2021)
12. Android Studio [Электронный ресурс] // <https://developer.android.com/> URL: <https://developer.android.com/studio?hl=ru> (Дата обращения: 18.04.2021)
13. GitHub [Электронный ресурс] // <https://ru.wikipedia.org/> URL: <https://ru.wikipedia.org/wiki/GitHub> (Дата обращения: 18.04.2021)
14. GitHub Actions [Электронный ресурс] // <https://github.com/> URL: <https://github.com/features/actions> (Дата обращения: 18.04.2021)

15. Linux [Электронный ресурс] // <https://ru.wikipedia.org/> URL: <https://ru.wikipedia.org/wiki/Linux> (Дата обращения: 18.04.2021)
16. Azure Virtual Machines [Электронный ресурс] // <https://azure.microsoft.com/> URL: <https://azure.microsoft.com/en-us/services/virtual-machines/> (Дата обращения: 18.04.2021)
17. НИУ ВШЭ [Электронный ресурс] // <https://www.hse.ru/> URL: <https://www.hse.ru/> (Дата обращения: 18.04.2021)
18. SSH ключ [Электронный ресурс] // <https://losst.ru/> URL: <https://losst.ru/avtorizatsiya-po-klyuchu-ssh> (Дата обращения: 18.04.2021)
19. FileZilla [Электронный ресурс] // <https://www.filezilla.ru/> URL: <https://www.filezilla.ru/> (Дата обращения: 18.04.2021)
20. Singleton [Электронный ресурс] // <https://refactoring.guru/> URL: <https://refactoring.guru/ru/design-patterns/singleton> (Дата обращения: 18.04.2021)
21. Factory [Электронный ресурс] // <https://refactoring.guru/> URL: <https://refactoring.guru/ru/design-patterns/factory-method> (Дата обращения: 18.04.2021)
22. Prototype [Электронный ресурс] // <https://refactoring.guru/> URL: <https://refactoring.guru/ru/design-patterns/prototype> (Дата обращения: 18.04.2021)
23. Java Spring Boot [Электронный ресурс] // <https://spring.io/> URL: <https://spring.io/projects/spring-boot> (Дата обращения: 18.04.2021)
24. Spring Framework [Электронный ресурс] // <https://spring.io/> URL: <https://spring.io/projects/spring-framework> (Дата обращения: 18.04.2021)
25. Sl4j logger [Электронный ресурс] // <http://www.slf4j.org/> URL: <http://www.slf4j.org/apidocs/org/slf4j/Logger.html> (Дата обращения: 18.04.2021)
26. Spring Security [Электронный ресурс] // <https://spring.io/> URL: <https://spring-projects.ru/guides/securing-web/> (Дата обращения: 18.04.2021)
27. Google [Электронный ресурс] // <https://ru.wikipedia.org/> URL: [https://ru.wikipedia.org/wiki/Google_\(компания\)](https://ru.wikipedia.org/wiki/Google_(компания)) (Дата обращения: 18.04.2021)
28. Gson [Электронный ресурс] // <https://github.com/> URL: <https://github.com/google/gson> (Дата обращения: 18.04.2021)
29. Lombok [Электронный ресурс] // <https://projectlombok.org/> URL: <https://projectlombok.org/> (Дата обращения: 18.04.2021)
30. MySQL Connector [Электронный ресурс] // <https://www.mysql.com/> URL: <https://www.mysql.com/products/connector/> (Дата обращения: 18.04.2021)
31. OkHttp [Электронный ресурс] // <https://square.github.io/> URL: <https://square.github.io/okhttp/> (Дата обращения: 18.04.2021)
32. FireStorage [Электронный ресурс] // <https://firebase.google.com/> URL: <https://firebase.google.com/docs/storage?hl=sr> (Дата обращения: 18.04.2021)

33. Postman [Электронный ресурс] // <https://www.postman.com/> URL: <https://www.postman.com/>
(Дата обращения: 18.04.2021)
34. UML [Электронный ресурс] // <https://ru.wikipedia.org/> URL:
[https://ru.wikipedia.org/wiki/Диаграмма_\(UML\)](https://ru.wikipedia.org/wiki/Диаграмма_(UML)) (Дата обращения: 18.04.2021)
35. CRUD [Электронный ресурс] // <https://ru.wikipedia.org/> URL:
<https://ru.wikipedia.org/wiki/CRUD> (Дата обращения: 18.04.2021)
36. MySQL консоль [Электронный ресурс] // <http://lifeexample.ru/> URL:
<http://lifeexample.ru/razrabotka-i-optimizacia-saita/mysql-konsol.html> (Дата обращения:
18.04.2021)
37. Заголовки HTTP запросов [Электронный ресурс] // <https://ru.wikipedia.org/> URL:
https://ru.wikipedia.org/wiki/Список_заголовков_HTTP (Дата обращения: 18.04.2021)
38. Список кодов состояния HTTP [Электронный ресурс] // <https://ru.wikipedia.org/> URL:
https://ru.wikipedia.org/wiki/Список_кодов_состояния_HTTP (Дата обращения: 18.04.2021)
39. Data Transfer Object [Электронный ресурс] // <https://habr.com/> URL:
<https://habr.com/ru/post/513072/> (Дата обращения: 18.04.2021)
40. Рос. Потреб [Электронный ресурс] // <http://ros-potreb.ru/> URL: <http://ros-potreb.ru/services/5.html> (Дата обращения: 18.04.2021)
41. Слос [Электронный ресурс] // <https://github.com/> URL: <https://github.com/AIDanial/cloc> (Дата обращения: 18.04.2021)

Приложение А. Техническое задание

Приложение Б. Руководство оператора

Приложение В. Программа и методика испытаний

Приложение Г. Текст программы