



Лист с заданием



## **Резюме**

Тема дипломной работы «Программный комплекс для автоматизации учета работы врача-анестезиолога».

В ходе выполнения дипломной работы было разработано программный комплекс, которое помогает врачам-анестезиологам в формировании удобного, многофункционального и эффективного рабочего места.

Объектом разработки является программное средство, предназначенное для автоматизации формирования отчетов об обследовании пациентов.

Поставленная задача была успешно выполнена в полном объеме.

## **Резюме**

Тема дипломной работы «Программный комплекс для автоматизации учета работы врача-анестезиолога».

У ходзе викання дипломной работы было разработано программный комплекс, який допомагає лікарям-анестезіологам у формуванні зручного, шматфункціонального і ефективного робочого місця.

Об'єктом розробки є програмне засередство, призначене для автоматизації формування звітів про обстеження пацієнтів.

Поставлена задача була успішно виконана у повному об'ємі.

## **Summary**

The theme of the thesis is "A software package for automating the work of an anesthetist."

In the course of the thesis, a software package was developed that helps anesthetists in creating a convenient, multifunctional and effective workplace.

The object of development is a software tool designed to automate the generation of patient examination reports.

The task was successfully completed in full.

# СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ .....</b>	<b>7</b>
<b>1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ .....</b>	<b>8</b>
1.1 История развития медицинских информационных систем .....	8
1.2 Аналоги в Беларуси.....	9
1.3 Выбор языка программирования.....	11
1.4 Технология Spring .....	12
1.5 MySQL .....	15
<b>2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ.....</b>	<b>16</b>
2.1 Общая структура приложения.....	16
2.2 Функциональная схема .....	24
2.3 База данных .....	26
<b>3 Разработка программного обеспечения.....</b>	<b>33</b>
3.1 Серверная часть приложения .....	33
3.2 Android приложение.....	40
<b>4 ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ .....</b>	<b>43</b>
4.1 Ручное тестирование программного комплекса в режиме админ .....	43
4.2 Ручное тестирование программного комплекса в режиме доктор .....	47
4.3 Ручное тестирование программного комплекса в режиме медсестры .....	53
4.4 Ручное тестирование android приложения.....	55
4.5 Модульное тестирование работы сложных алгоритмов программного комплекса .....	57
<b>5 Экономическое обоснование дипломной работы .....</b>	<b>61</b>
5.1 Расчет трудоемкости работ по разработке программного обеспечения ....	61
5.2 Расчет затрат на разработку программного продукта.....	67
5.3 Формирование цены создания программного обеспечения .....	76
5.4 Расчет эффекта от внедрения программного обеспечения .....	78
<b>6 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ.....</b>	<b>81</b>
6.1 Понятие и содержание гигиены труда и производственной санитарии.....	81
6.2 Классификация вредных и опасных производственных факторов .....	81
6.3 Методы профилактики профессиональных заболеваний .....	83
6.4 Медицинские предварительные и периодические осмотры работников...	83
6.5 Санитарно-бытовое обеспечение работников. Оборудование санитарно-бытовых помещений, их размещение .....	85
<b>7 РЕСУРСО- И ЭНЕРГОСБЕРЕЖЕНИЕ ПРИ ВНЕДРЕНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ .....</b>	<b>87</b>
7.1 Основные понятия в области ресурсо- и энергосбережения .....	87
7.2 Регулирование вопросов, связанных с внедрением программного обеспечения и последующего ресурсосбережения .....	87

7.2 Экономия энергоресурсов в результате внедрения программного обеспечения .....	88
<b>ЗАКЛЮЧЕНИЕ</b> .....	90
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b> .....	91

## ВВЕДЕНИЕ

В последние десятилетия результативность лечебно-диагностического процесса, эффективность использования ресурсов в нем задействованных во многом определяются своевременностью и качеством решений, принимаемых врачом. Врач всегда был и остается главным звеном всей системы здравоохранения. Именно на уровне врача - сосредоточен весь конгломерат мер по реализации профессионального врачебного процесса, экономики медицинского учреждения и организационных форм оказания этапной медицинской помощи пациентам. Логичным и обоснованным представляется убеждение в том, что врач XXI века - это профессионал, владеющий всеми методами современной профилактики, диагностики и лечения болезней и вооруженный для этого современными медицинскими технологиями.

Объем профессиональных знаний, необходимых для успешной работы врача, значителен. Сейчас же он увеличился многократно и продолжает постоянно расти. Поэтому актуальность усиления информационной поддержки профессиональной врачебной деятельности, медицинских работников в целом обусловлена не только потребностью в повышении качества оказываемой медицинской помощи населению, но и необходимостью оптимизации используемого для этого потенциала лечебного учреждения.

Эффективное решение этой проблемы возможно только посредством новых подходов, через систему информатизации основных направлений деятельности учреждений здравоохранения и труда медицинских работников, путем реорганизации системы управления ресурсами здравоохранения (кадры, финансы, материально-техническая база, лечебно-диагностический процесс, обеспечение лекарственными средствами и изделиями медицинского назначения и др.) на основе системы медицинских и информационных технологий.

Поэтому проблема создания автоматизированных рабочих мест (АРМ) врачей и, в частности, врачей общей практики давно уже перешла из плоскости теоретических рассуждений в плоскость практических действий, издания распорядительных актов, разработки проектно-технической документации по созданию и внедрению конкретных предметно-ориентированных медицинских АРМ, как составляющих элементов, автоматизированных медицинских информационных систем лечебно-профилактических учреждений. Развитие подобных систем, имеющих десятки, а в ряде медицинских организаций сотни АРМ персонала ЛПУ - это не планы, это реальность современного здравоохранения.

# 1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

## 1.1 История развития медицинских информационных систем

История развития Медицинских Информационных Систем (МИС) началась еще в 50-х годах XX века в Соединенных Штатах Америки, когда на рынке появились универсальные компьютеры многоцелевого назначения. Первым проектом такой информационной системы был MEDINET, разработанный фирмой “General Electric” [1].

В XIX веке здравоохранение в США представляло собой благотворительную общественную программу помощи больным и нуждающимся людям и состояло из изолированных служб, которые оказывали пациентам эпизодическую, неотложную помощь. Необходимость в обмене информацией между врачами или клиниками была невысокой. Век положил начало бурному развитию и расширению отрасли здравоохранения. Благодаря федеральному финансированию и ряду законов населению расширился доступ к медицинскому обслуживанию, а учреждениям гарантировалось достаточное возмещение затрат на медицинские услуги. В течении всего этого периода истории болезни велись в бумажной форме и были ориентированы на нужды конкретного медицинского учреждения.

Однако, благодаря гарантированному и стабильному притоку денежных средств на оказание медицинской помощи, специалисты и учреждения могли развивать новаторские подходы: применять новые лекарственные вещества, передовые хирургические методы, современные приемы диагностики.

Один из таких новаторских подходов стало применение компьютерных технологий в клинической практике. В 70-х годах XX века развитие МИС разделилось на два основных направления. Один из них - наличие одного мощного компьютера (сервера) и терминалов на рабочих станциях, которые включали в себя только монитор и клавиатуру. Другая часть разработок базировалась на создании распределенных систем, которые поддерживали раздельную реализацию специализированных приложений с помощью самостоятельных компьютеров. В обоих направлениях действовал принцип единой базы данных, в которой хранится вся информация о пациентах [1].

На ранней стадии развития усилия были направлены на развитие больничных информационных систем и программных средств поддержки принятия решений.

Больничные информационные системы (БИС) представляют собой большие автоматизированные базы данных, которые используются для хранения информации медицинского и административного характера. В начале развития, системы были ориентированы на передачу инструкций по оказанию неотложной помощи и получению данных от вспомогательных служб, например, лабораторий или аптек.

В конце 80-х годов остро встала проблема адекватности принимаемых медиками решений и эффективности лечения. Это послужило стимулом для



появления такого направления как анализ исходов лечения. Базой для этих перемен стало объединение медицинских учреждений различных видов в комплексные сети. Вместе с объединением самих учреждений возникла необходимость объединить информационные базы и разрешить к ним совместный доступ.

Массовое распространение информационных технологий в СССР и зарубежных странах приходится на середину 70-х годов XX века, что связано с появлением персональных ЭВМ.

Больницы и их отделения получили возможность приобрести собственные компьютеры, для самостоятельной разработки требуемых прикладных систем. В начале 80-х годов XX века этой возможностью пользовались многие крупные лечебные учреждения. Итогом стали разработки плохо тиражируемых и трудно развиваемых систем.

Главным толчком для разработки программного обеспечения нового поколения стало появление в конце 80-х - начале 90-х годов XX века персональных компьютеров. Это давало возможность пользоваться компьютером всем работникам здравоохранения [1].

В 90-е годы велось широкое обсуждение достоинств интегрированных информационных систем. Их преимущество заключалось в снижении повторяемости и избыточности информации, точности и доступности данных. Такие системы способны облегчить интеллектуальную работу врачей, повысить качество принятия решений в клинической практике.

В настоящее время существуют различные фирмы, производящие МИС, включающие в себя не только АРМ врачей по профилю, но и системы административного и бухгалтерского учета. Среди них наибольшее распространение получили МЕДИАЛОГ, Эверест, Интерин, Барс, Гиппократ, Авицена, Амулет, Дока +.

## **1.2 Аналоги в Беларуси**

Как таковых аналогов в Беларуси не существуют, поэтому мы будем рассматривать аналоги на основе медицинских систем, которых имеется большое количество. Самые популярные в Беларуси: «Клиника», «Стационар» и «4D Client».

Все медицинские системы имеют схожую структуру, а именно она состоит из нескольких частей, что и образует медицинскую систему, поэтому рассмотрим одну из популярных систем, такую как «Стационар».

Данный продукт разрабатывает компания ЗАО «МАПСОФТ» – белорусская компания-разработчик программного обеспечения. Компания разрабатывает и внедряет современные программы для бюджетных организаций Республики Беларусь. Данной программой пользуются более 200 медицинских учреждений, более 4500 специалистов, и обслуживают более 4 миллионов населения [2].

«Стационар» состоит из 14 частей:

- 1) **приемный покой** – программа предназначена для регистрации поступивших пациентов;
- 2) **врач отделения** – программа предназначена для ведения врачом электронной истории болезни пациента;
- 3) **заведующий отделением** – программа предназначена для контроля и анализа лечения пациентов в отделении;
- 4) **старшая медсестра** – программа предназначена для учета движения пациентов, медикаментов и коечного фонда в отделениях;
- 5) **постовая медсестра** – программа предназначена для учета движения пациентов и коечного фонда в отделениях, регистрации и контроля выполнения медикаментозных назначений;
- 6) **аптека** – программа предназначена для учета медикаментов в аптеке учреждений здравоохранения Республики Беларусь;
- 7) **лаборатория** – программа предназначена для учета лабораторных исследований и полученных результатов;
- 8) **лучевая диагностика** – программа предназначена для учета ультразвуковых исследований и полученных результатов;
- 9) **функциональная диагностика** – программа предназначена для учета функциональных исследований и полученных результатов;
- 10) **родовспоможение** – программа предназначена для ведения электронных медицинских карт рожениц, новорожденных и журнала родов;
- 11) **статистика стационара** – программа предназначена для автоматизации статистического учета в стационаре;
- 12) **платные услуги** – программа предназначена для подготовки договоров и актов на платные услуги, оказываемые учреждением здравоохранения;
- 13) **главный врач** – программа предназначена для мониторинга и контроля работы стационара;
- 14) **администратор** – программа предназначена для администрирования работы пользователей.

Рассмотрим подробнее 2 системы: «врач отделения», «постовая медсестра».

Система «врач отделения» имеет следующие возможности [2]:

- 1) ведение медицинских карт пациентов в отделении;
- 2) формирование дневниковых записей в электронной истории болезни на основе шаблонов (соответствуют приказу Министерства здравоохранения №1050 от 18.11.2008г.);
- 3) ввод медикаментозных назначений пациенту отделения;
- 4) формирование и печать листка назначения;
- 5) ввод данных об оказанных медицинских услугах и выполненных операциях;
- 6) формирование выписного эпикриза;
- 7) формирование оперативных и аналитических отчетов: медицинская карта стационарного пациента, статистическая карта, форма 066/у, журнал хирургических операций и др.

Данная программа позволяет пользователям вести карту пациента, но она предназначена для общего пользования, а не для конкретных задач отдельных врачей. Каждого отдельного врача интересует отдельные показатели пациента и для них существуют разные виды карт пациентов, что является существенным недостатком данной системы.

Система «постовая медсестра» имеет следующие возможности [2]:

- 1) регистрация информации по движению пациентов и коечного фонда в отделениях (форма 007/у);
- 2) формирование направлений на назначенные пациентам диагностические исследования;
- 3) выполнение медикаментозных назначений пациентам;
- 4) формирование оперативных и аналитических отчетов: листок учета движения пациентов и коечного фонда, списки пациентов с распределением по палатам, назначения по палатам, порционное требование, листки назначений и др.

Данная программа имеет возможность выполнять медикаментозных назначений пациентам, но не имеет возможность отслеживать прием данных назначений, что является недостатком данной системы.

Плюсом данной системы является то, что все эти программы являются одной системой, а значит, мы имеем возможность добавить нашу программу в эту систему, чтобы улучшить ее, что является огромным плюсом как нашей программе, так и их системе.

Минусом данной программы является то, что пользоваться ее можно только на компьютерах медицинских учреждений. Что является минусом, когда нужно просмотреть карточку пациента здесь и сейчас.

### 1.3 Выбор языка программирования

Язык Java разработан компанией Sun Microsystems, создателем которого был Джеймс Гослинг, и выпущен в 1995 году в качестве основных компонентов компании Sun Microsystems – Java платформ (Java 1.0 [J2SE]).

По состоянию на 2018 год последней версией Java Standard Edition является 10 (J2SE). С развитием Java, и её широкой популярностью, несколько конфигураций были построены для различных типов платформ. Например, J2EE – приложения для предприятий, J2ME – для мобильных приложений.

Sun Microsystems переименовала прежнюю версию J2 и ввела новые: Java SE, Java EE и Java ME. Введение в программирование Java различных версий подтверждало знаменитый слоган компании **«Напиши один раз, запускай везде»**.

Объектно-ориентированный: в Java все является объектом. Дополнение может быть легко расширено, так как он основан на объектной модели.

**Платформонезависимый:** в отличие от многих других языков, включая C и C++, Java, когда был создан, он не компилировался в платформе конкретной машины, а в независимом от платформы байт-коде. Этот байт код

распространяется через интернет и интерпретируется в Java Virtual Machine (JVM), на которой он в настоящее время работает.

**Простой:** процессы изучения и введение в язык программирования Java остаются простыми. Если Вы понимаете основные концепции объектно-ориентированного программирования, то он будет прост для Вас в освоении.

**Безопасным:** методы проверки подлинности основаны на шифровании с открытым ключом.

**Архитектурно-нейтральным:** компилятор генерирует архитектурно-нейтральные объекты формата файла, что делает скомпилированный код исполняемым на многих процессорах, с наличием системы Java Runtime.

**Портативный:** архитектурно-нейтральный и не имеющий зависимости от реализации аспектов спецификаций – все это делает Java портативным. Компилятор в Java написан на ANSI C с чистой переносимостью, который является подмножеством POSIX.

**Прочный:** выполняет усилия, чтобы устранить ошибки в различных ситуациях, делая упор в основном на время компиляции, проверку ошибок и проверку во время выполнения.

**Многопоточный:** функции многопоточности, можно писать программы, которые могут выполнять множество задач одновременно. Введение в язык Java этой конструктивной особенности позволяет разработчикам создавать отлаженные интерактивные приложения.

**Интерпретированный:** Java байт-код переводится на лету в машинные инструкции и нигде не сохраняется. Делая процесс более быстрым и аналитическим, поскольку связывание происходит как дополнительное с небольшим весом процесса.

**Высокопроизводительный:** введение Just-In-Time компилятора, позволило получить высокую производительность.

**Распространенный:** предназначен для распределенной среды интернета.

**Динамический:** программирование на Java считается более динамичным, чем на C или C++, так как он предназначен для адаптации к меняющимся условиям. Программы могут выполнять обширное количество во время обработки информации, которая может быть использована для проверки и разрешения доступа к объектам на время выполнения [4].

## 1.4 Технология Spring

Spring обеспечивает решения многих задач, с которыми сталкиваются Java-разработчики и организации, которые хотят создать информационную систему, основанную на платформе Java. Из-за широкой функциональности трудно определить наиболее значимые структурные элементы, из которых он состоит. Spring не всецело связан с платформой Java Enterprise, несмотря на его масштабную интеграцию с ней, что является важной причиной его популярности.

Spring, вероятно, наиболее известен как источник расширений (features), нужных для эффективной разработки сложных бизнес-приложений вне

тяжеловесных программных моделей, которые исторически были доминирующими в промышленности. Ещё одно его достоинство в том, что он ввел ранее неиспользуемые функциональные возможности в сегодняшние господствующие методы разработки, даже вне платформы Java.

Этот фреймворк предлагает последовательную модель и делает её применимой к большинству типов приложений, которые уже созданы на основе платформы Java. Считается, что Spring реализует модель разработки, основанную на лучших стандартах индустрии, и делает её доступной во многих областях Java [5].

Spring может быть рассмотрен как коллекция меньших фреймворков или фреймворков во фреймворке. Большинство этих фреймворков может работать независимо друг от друга, однако они обеспечивают большую функциональность при совместном их использовании. Эти фреймворки делятся на структурные элементы типовых комплексных приложений:

**1) inversion of Control-контейнер:** конфигурирование компонентов приложений и управление жизненным циклом Java-объектов;

**2) фреймворк доступа к данным:** работает с системами управления реляционными базами данных на Java-платформе, используя JDBC- и ORM-средства и обеспечивая решения задач, которые повторяются в большом числе Java-based environments;

**3) фреймворк MVC:** каркас, основанный на HTTP и сервлетах, предоставляющий множество возможностей для расширения и настройки (customization);

**4) фреймворк аутентификации и авторизации:** конфигурируемый инструментальный процесс аутентификации и авторизации, поддерживающий много популярных и ставших индустриальными стандартами протоколов, инструментов, практик через дочерний проект Spring Security (ранее известный как Aсegi) [5].

Центральной частью Spring является контейнер Inversion of Control, который предоставляет средства конфигурирования и управления объектами Java с помощью рефлексии. Контейнер отвечает за управление жизненным циклом объекта: создание объектов, вызов методов инициализации и конфигурирование объектов путём связывания их между собой.

Объекты, создаваемые контейнером, также называются управляемыми объектами (beans). Обычно конфигурирование контейнера осуществляется путём загрузки XML-файлов, содержащих определение bean'ов и предоставляющих информацию, необходимую для создания bean'ов.

Объекты могут быть получены одним из двух способов:

Поиск зависимости – шаблон проектирования, в котором вызывающий объект запрашивает у объекта-контейнера экземпляр объекта с определённым именем или определённого типа.

Внедрение зависимости – шаблон проектирования, в котором контейнер передает экземпляры объектов по их имени другим объектам с помощью конструктора, свойства или фабричного метода [5].

Spring имеет собственную MVC-платформу веб-приложений, которая не была первоначально запланирована. Разработчики Spring решили написать её как реакцию на то, что они восприняли как неудачность конструкции (тогда) популярного Apache Struts, а также других доступных веб-фреймворков. В частности, по их мнению, было недостаточным разделение между слоями представления и обработки запросов, а также между слоем обработки запросов и моделью.

Класс `DispatcherServlet` является основным контроллером фреймворка и отвечает за делегирование управления различным интерфейсам, на всех этапах выполнения HTTP-запроса. Об этих интерфейсах следует сказать более подробно.

Как и Struts, Spring MVC является фреймворком, ориентированным на запросы. В нем определены стратегические интерфейсы для всех функций современной запросно-ориентированной системы. Цель каждого интерфейса — быть простым и ясным, чтобы пользователям было легко его заново имплементировать, если они того пожелают. MVC прокладывает путь к более чистому front-end-коду. Все интерфейсы тесно связаны с Servlet API. Эта связь рассматривается некоторыми как неспособность разработчиков Spring предложить для веб-приложений абстракцию более высокого уровня. Однако эта связь оставляет особенности Servlet API доступными для разработчиков, облегчая все же работу с ним. Наиболее важные интерфейсы, определенные Spring MVC, перечислены ниже:

1. `HandlerMapping`: выбор класса и его метода, которые должны обработать данный входящий запрос на основе любого внутреннего или внешнего для этого запроса атрибута или состояния.

2. `HandlerAdapter`: вызов и выполнение выбранного метода обработки входящего запроса.

3. `Controller`: включен между Моделью (Model) и Представлением (View). Управляет процессом преобразования входящих запросов в адекватные ответы. Действует как ворота, направляющие всю поступающую информацию. Переключает поток информации из модели в представление и обратно.

4. `View`: ответственно за возвращение ответа клиенту в виде текстов и изображений. Некоторые запросы могут идти прямо во View, не заходя в Model; другие проходят через все три слоя.

5. `ViewResolver`: выбор, какое именно View должно быть показано клиенту.

6. `HandlerInterceptor`: перехват входящих запросов. Сопоставим, но не эквивалентен сервлет-фильтрам (использование не является обязательным и не контролируется `DispatcherServlet`-ом).

7. `LocaleResolver`: получение и, возможно, сохранение локальных настроек (язык, страна, часовой пояс) пользователя.

8. `MultipartResolver`: обеспечивает Upload — загрузку на сервер локальных файлов клиента.

Spring MVC предоставляет разработчику следующие возможности:

1. Ясное и прозрачное разделение между слоями в MVC и запросах.

2. Стратегия интерфейсов — каждый интерфейс делает только свою часть работы.

3. Интерфейс всегда может быть заменен альтернативной реализацией.

4. Интерфейсы тесно связаны с Servlet API.

5. Высокий уровень абстракции для веб-приложений.

В веб-приложениях можно использовать различные части Spring, а не только Spring MVC [5].

Spring предоставляет свой слой доступа к базам данных и поддерживает все популярные СУБД.

JDBC, iBatis / MyBatis, Hibernate, JDO, JPA, Oracle TopLink, Apache OJB, Apache Cayenne и т. д.

Для всех этих фреймворков Spring предоставляет такие особенности:

1. Управление ресурсами — автоматическое получение и освобождение ресурсов базы данных

2. Обработка исключений — перевод исключений при доступе к данным в исключения Spring-a

3. Транзакционность — прозрачные транзакции в операциях с данными

4. Распаковка ресурсов — получение объектов базы данных из пула соединений.

5. Абстракция для обработки BLOB и CLOB [5].

## 1.5 MySQL

*MySQL* — это популярная СУБД. Она не предназначена для работы с большими объемами информации, но ее применение отлично подходит для интернет-сайтов, как небольших, так и достаточно крупных.

*MySQL* отличается хорошей скоростью работы, надежностью, гибкостью. Работа с ней, как правило, не вызывает больших трудностей.

Данная СУБД более компактная, чем аналоги, как *MsSQL* или *Oracle*, бесплатная, имеет многочисленные средства для восстановления таблиц, имеет *API* через который очень быстро взаимодействует с приложением, имеет встроенный сервер — т. е. можно включить в приложение и не устанавливать сервер отдельно от программы.

Наиболее быстрый тип таблиц *MyISAM*, который используется в *MySQL* имеет не высокую надёжность, по сравнению с *Oracle* и *MS SQL Server*, при достижении таблиц размера более *10Gb*. Таблицы типа *MyISAM* не поддерживают транзакции, зато поддерживают полнотекстовый поиск, а *InnoDB* поддерживают транзакции, но не поддерживают полнотекстовый поиск. Поддерживает не все элементы *SQL*-стандарта, однако данная ситуация исправляется в данный момент [6].

## 2 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

### 2.1 Общая структура приложения

2.1.1 По поставленной задаче было создано *WEB*-приложение. Приложение создано на клиент-серверном составляющем. Данное приложение имеет СУБД на основе *MySQL*. Сервер написан на *Java* фреймворке *Spring*. Клиент используемый в данном приложении, использует *JavaScript* фреймворк. Также было разработано android приложение, которое позволяет получать быстрый доступ к документам. Android общается с сервером, посылая ему запросы и обрабатывая ответы. Доступ к android приложению имеют не все пользователи данной системы.

Общая схема данного приложения представлена на рисунке 2.1.

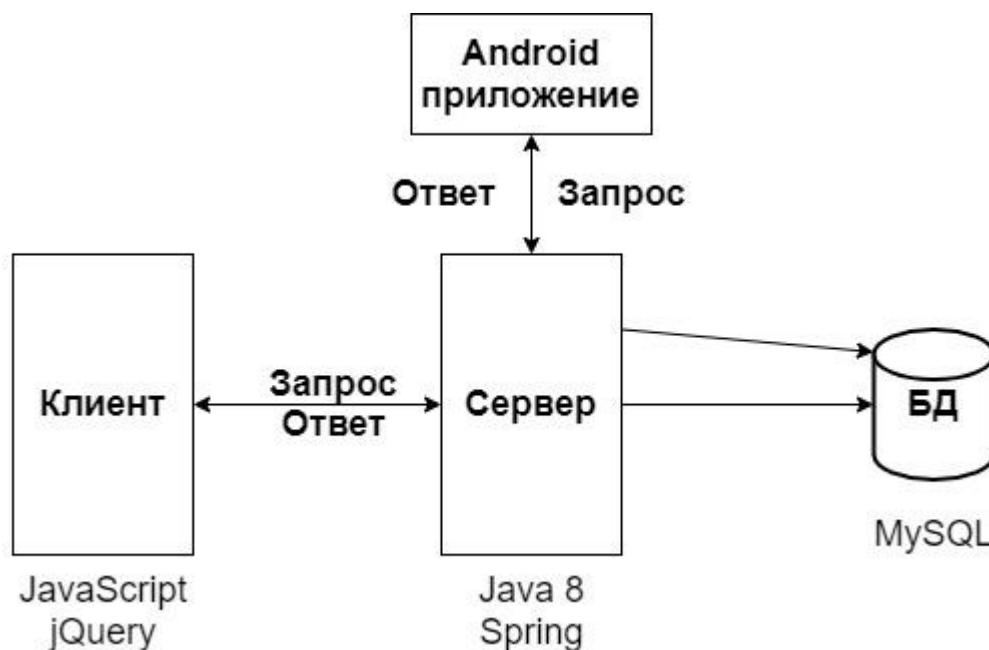


Рисунок 2.1 – Общая схема приложения

Примечание к рисунку 2.1: для каждой операции требуется авторизация пользователя в системе. Если пользователь не авторизован, ему будет предложено произвести регистрацию. Представленная диаграмма предполагает, что пользователь прошёл авторизацию.

Приведём ниже примеры нескольких сценариев, моделирующих ситуации, для каждой из ролей.

Доктор, добавление диагнозов поступившему пациенту:

- доктор переходит к списку пациентов и выбирает пациента;
- доктор переходит на страницу выбранного пациента;
- доктор нажимает кнопку «Диагнозы» и переходит на страницу диагнозов данного пациента;



- доктор нажимает кнопку «Добавить диагноз» и переходит на страницу добавления диагноза;

- доктор выбирает диагноз из справочника, который заполнен на сервере, если такой диагноз уже есть у данного пациента, то в данном справочнике он уже не отображается;

- повторять предыдущий шаг столько раз, сколько диагнозов у пациента.

Доктор, добавление медикаментов для лечения поступившему пациенту:

- доктор переходит к списку пациентов и выбирает пациента;

- доктор переходит на страницу выбранного пациента;

- доктор нажимает кнопку «Медикаменты» и переходит на страницу назначенных медикаментов данному пациенту;

- доктор нажимает кнопку «Добавить медикамент» и переходит на страницу добавления медикаментов;

- доктор выбирает медикамент из справочника, который заполнен на сервере, если такой медикамент уже есть у данного пациента, то в данном справочнике он уже не отображается, а также заполняет поля по дозе и частоте употребления данного медикамента;

- повторять предыдущий шаг столько раз, сколько медикаментов назначено у пациента.

Медсестра, добавление отчета о приеме лекарства, выбранному пациенту:

- медсестра переходит к списку пациентов и выбирает пациента;

- медсестра переходит на страницу медикаментов данного пациента;

- медсестра выбирает медикамент;

- медсестра нажимает кнопку «Добавить применение» и переходит на страницу добавления;

- медсестра нажимает кнопку «Добавить» и в базу сохраняется принятие данного медикамента выбранным пациентом;

Доктор, сценарий планового осмотра и составление документа о данном осмотре:

- доктор переходит к списку пациентов и выбирает пациента, которому он будет проводить плановый осмотр;

- доктор переходит на страницу выбранного пациента;

- доктор создает новое посещение, указав дату, описание посещения и примечания;

- доктор выбрал новое посещение и перешел на страницу документов;

- доктор добавил новый документ, который он должен заполнить;

- доктор перешёл на страницу документа, заполнил и сохранил документ.

2.1.2 Структурная схема – это совокупность элементарных блоков (модулей системы) и отношений между ними, один из видов графической модели. Разделение на структурные блоки и определение полной структурной схемы на этапе проектирования приложения способствует ускорению процесса разработки программного комплекса, путем сведения к минимуму переключения внимания разработчика между задачами разного рода, что повышает его эффективность производительности. Помимо этого, выделение структурной

схемы позволяет снизить порог опыта, необходимого разработчику для реализации продукта, так как данная операция способна помочь разработчику с меньшим опытом совершать меньше ошибок при разработке, чем если бы он выполнял планирование непосредственно в процессе написания программного кода.

Структурная схема разрабатываемого приложения представлена на рисунке 2.2.

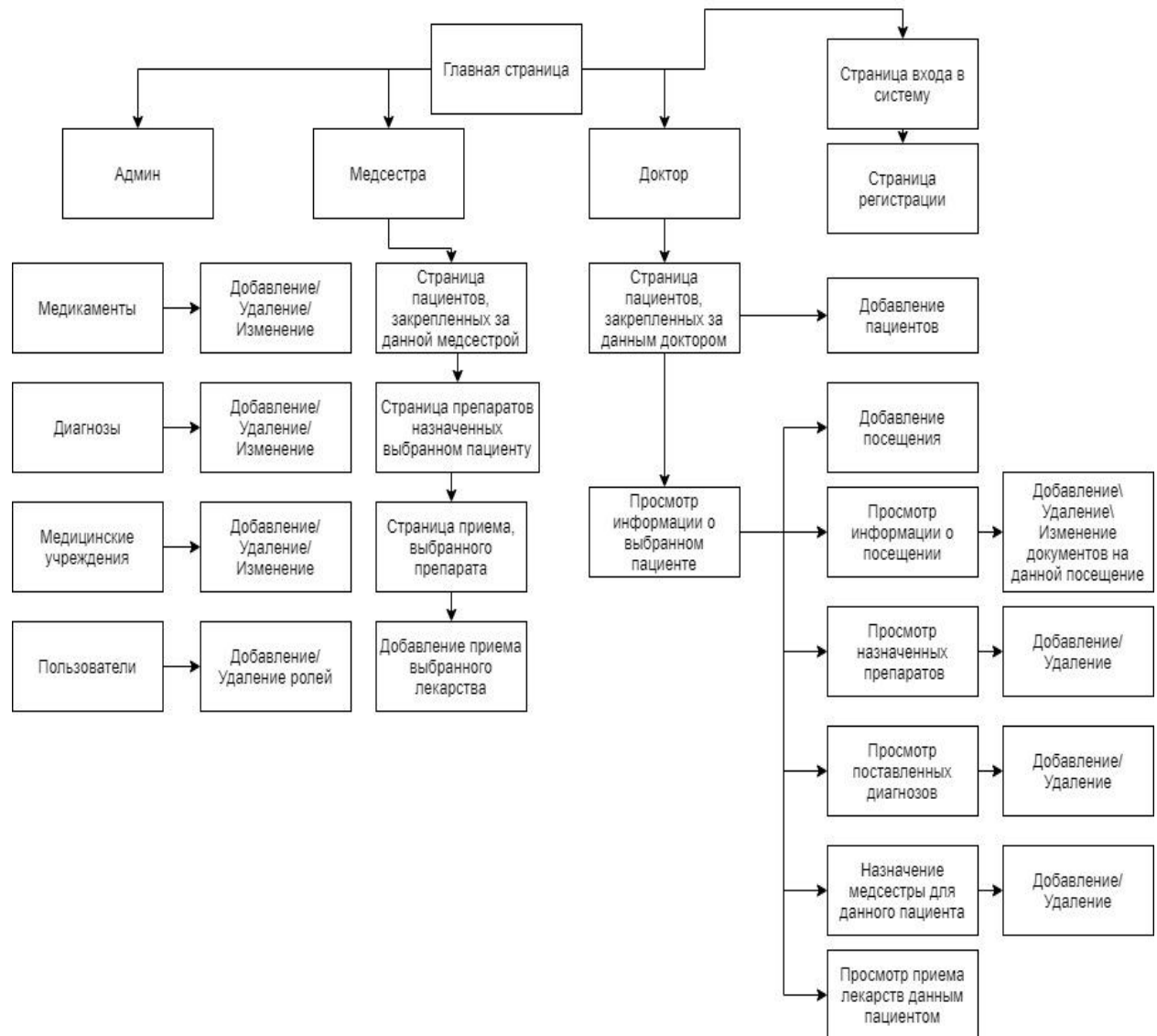


Рисунок 2.2 – Структурная схема разрабатываемого приложения

2.1.3 Для обеспечения безопасности было использовано решение, которое предоставляет фреймворк Spring, а именно Spring Security. Spring Security – это Java/JavaEE framework, предоставляющий механизмы построения систем аутентификации и авторизации, а также другие возможности обеспечения безопасности для корпоративных приложений, созданных с помощью Spring Framework. Ключевые объекты контекста Spring Security:

– SecurityContextHolder, в нем содержится информация о текущем контексте безопасности приложения, который включает в себя подробную

информацию о пользователе(Principal) работающем в настоящее время с приложением. По умолчанию SecurityContextHolder использует ThreadLocal для хранения такой информации, что означает, что контекст безопасности всегда доступен для методов исполняющихся в том же самом потоке;

- SecurityContext, содержит объект Authentication и в случае необходимости информацию системы безопасности, связанную с запросом от пользователя;

- GrantedAuthority отражает разрешения выданные пользователю в масштабе всего приложения, такие разрешения (как правило называются «роли»), например ROLE\_ANONYMOUS, ROLE\_USER, ROLE\_ADMIN (в нашей системе есть 3 роли ROLE\_ADMIN, ROLE\_DOCTOR, ROLE\_NURSE);

- UserDetails предоставляет необходимую информацию для построения объекта Authentication из DAO объектов приложения или других источников данных системы безопасности. Объект UserDetails содержит имя пользователя, пароль, флаги: isAccountNonExpired, isAccountNonLocked, isCredentialsNonExpired, isEnabled и Collection — прав (ролей) пользователя;

- UserDetailsService, используется чтобы создать UserDetails объект путем реализации единственного метода этого интерфейса UserDetails loadUserByUsername(String username) throws UsernameNotFoundException; Позволяет получить из источника данных объект пользователя и сформировать из него объект UserDetails который будет использоваться контекстом Spring Security;

Аутентификация:

- пользователю будет предложено войти в систему предоставив имя (логин или email) и пароль. Имя пользователя и пароль объединяются в экземпляр класса UsernamePasswordAuthenticationToken(экземпляр интерфейса Authentication) после чего он передается экземпляру AuthenticationManager для проверки;

- в случае если пароль не соответствует имени пользователя будет выброшено исключение BadCredentialsException с сообщением “Bad Credentials”;

- если аутентификация прошла успешно возвращает полностью заполненный экземпляр Authentication.

Для безопасности учетных данных пользователей пароли шифруются при помощи библиотеки BCryptPasswordEncoder и в базе данных уже хранятся зашифрованные пароли.

2.1.4 В основе архитектуры приложения лежит использование шаблона проектирования MVC (от англ. – Model-View- Controller, рис. 2.3).

# Model-View-Controller

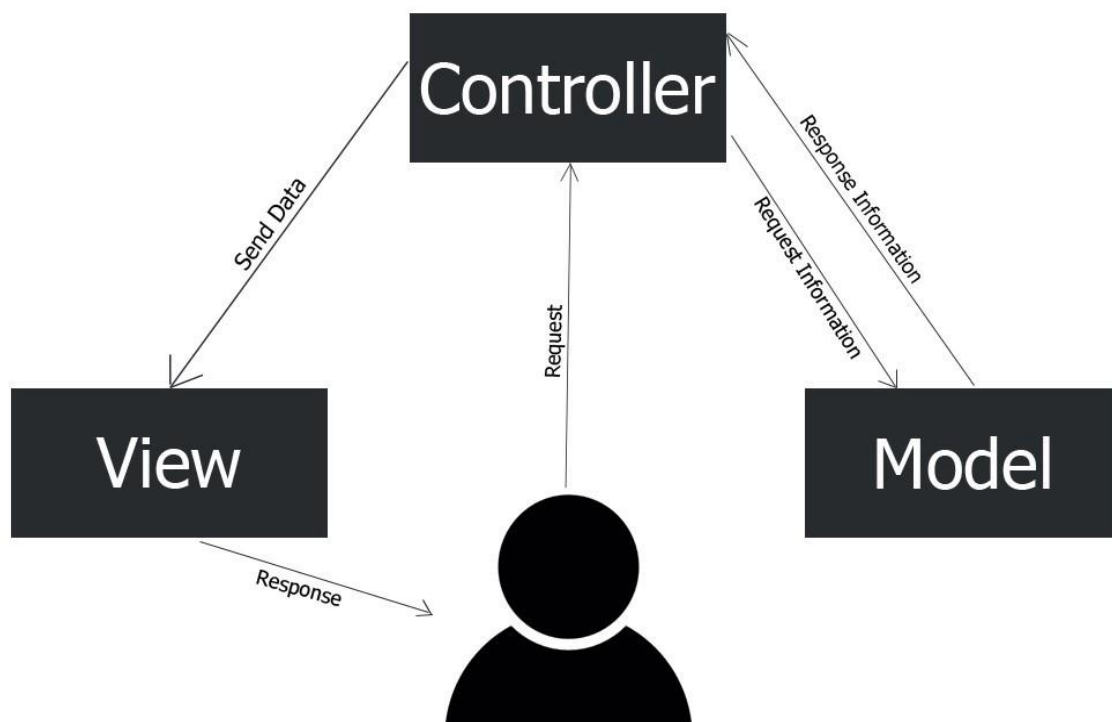


Рисунок 2.3 – Паттерн проектирования MVC

MVC – это паттерн проектирования веб-приложений, который включает в себя несколько более мелких шаблонов. При использовании MVC на три отдельных компонента разделены модель данных приложения, пользовательский интерфейс и логика взаимодействия пользователя с системой, благодаря чему модификация одного из этих компонентов оказывает минимальное воздействие на остальные или не оказывает его вовсе.

Основная цель применения MVC состоит в разделении данных и бизнес-логики от визуализации. За счет такого разделения повышается возможность повторного использования программного кода: например, добавить представление данных какого-либо существующего маршрута не только в виде HTML, но и в форматах JSON, XML, PDF, XLSX становится очень просто и не требует изменений слоя бизнес-логики исходного маршрута. Также упрощается и сопровождение программного кода: внесение изменений во внешний вид, например, не отражаются на бизнес-логике, а изменения бизнес-логики не затрагивают визуализацию.

Концепция MVC разделяет данные, представление и обработку действий пользователя на компоненты:

1. **Модель / Model** – предоставляет собой объектную модель некой предметной области, включает в себя данные и методы работы с этими данными, реагирует на запросы из контроллера, возвращая данные и/или изменяя своё состояние. При этом модель не содержит в себе информации о способах

визуализации данных или форматах их представления, а также не взаимодействует с пользователем напрямую.

**2. Представление / View** – отвечает за отображение информации (визуализацию). Одни и те же данные могут представляться различными способами и в различных форматах. Например, коллекцию объектов при помощи разных представлений можно представить на уровне пользовательского интерфейса как в табличном виде, так и списком; на уровне API можно экспортировать данные как в JSON, так в XML или XSLX.

**3. Контроллер / Controller** – обеспечивает связь между пользователем и системой, использует модель и представление для реализации необходимой реакции на действия пользователя. Как правило, на уровне контроллера осуществляется фильтрация полученных данных и авторизация – проверяются права пользователя на выполнение действий или получение информации.

2.1.5 В процессе проектирования были выделены основные доменные классы Entity, объекты которых соответствуют сущностям, хранимым в базе данных (от англ. Entity – «Сущность»):

- User – класс пользователя (хранит данные профиля учётной записи). Имеет отношение к классу доктор Doctor один-к-одному и к классу роль Role многие-ко-многим;

- Role – класс роли пользователей, хранит имя роли и имеет отношение многие-ко-многим с классом пользователя User;

- PersonalInformation – класс личной информации, содержит информацию об личной информации человека: имя, фамилия, отчество, возраст, адрес, телефон. Имеет отношение один-к-одному с классом доктор Doctor. Также содержит отношение один-к-одному с классом пациента Patient, который хранит информацию о пациенте;

- Doctor – класс хранит информации о докторе, который зарегистрирован в системе. Имеет отношение один-к-одному с классом личной информации PersonalInformation. Также содержит отношение один-к-одному с классом пользователь User. Имеет отношение один-ко-многим с классом пациент Patient. Имеет отношение один-ко-многим с классом PatientNurse. Имеет отношение многие-к-одному с классом медицинское учреждение MedicalInstitution. Имеет отношение один-ко-многим с классом посещения Visit. Имеет отношение один-ко-многим с классом приема лекарств Medication;

- Diagnosis – класс хранит информацию о диагнозах, а именно их названия. Имеет отношение один-ко-многим с классом PatientDiagnosis;

- Document – класс хранит информацию о документах, которые могут создаваться в системе (название документа, путь к html шаблону, путь к word шаблону). Имеет отношение один-ко-многим с классом VisitReportDocument;

- MedicalInstitution – класс хранит информацию о медицинских учреждениях, к которым привязываются пользователи системы. Имеет отношение один-ко-многим с классом доктора Doctor;

– Medicament – класс хранит информацию о названиях медикаментах, которые присутствуют в системе. Имеет отношение один-ко-многим с классом PatientMedicament;

– Medication – класс хранит информацию о приеме лекарств данным пациентом. Имеет связи многие-ко-одному с классами доктор Doctor, пациент Patient и PatientMedicament;

– Patient – класс хранит информацию о пациенте, который был создан в системе. Имеет связь один-к-одному с классом личная информация PersonalInformation. Также имеет связи один-ко-многим с классами PatientDiagnosis, посещения Visit, Medication, PatientMedicament, PatientNurse. Имеет связь многие-к-одному с классом доктор Doctor;

– PatientDiagnosis – класс хранит отношение пациентов к их диагнозам, а также стадию диагноза и название стадии. Имеет отношения многие-к-одному с классами диагноз Diagnosis и пациент Patient;

– PatientMedicament – класс хранит отношение пациентов к их медикаментам, а также дозу медикамента и частоту употребления. Имеет отношение один-ко-многим с классом Medication. А также отношение многие-к-одному с классами Medicament, Patient;

– PatientNurse – класс хранит связь пациентов с назначенными медсестрами. Имеет связи многие-к-одному с классами Doctor и Patient;

– Visit – класс хранит информацию о посещении пациента доктором. Имеет отношение один-к-одному с классом VisitReport. А также имеет отношения многие-к-одному с классами Patient, Doctor;

– VisitReport – класс хранит отчет о посещении пациента доктором. Имеет отношение один-к-одному с классом Visit. А также имеет отношение один-ко-многим с классом VisitReportDocument;

– VisitReportDocument – класс хранит информацию о документе, который создал доктор для пациента. Имеет связи многие-к-одному с классами Document, VisitReport;

– Roles – перечисление хранит названия ролей в данной системе.

2.1.6 При создании Android-приложений наиболее востребовано и оправдано использование библиотек, фреймворков и обращение к готовым API для уменьшения срока разработки приложения.

В основе архитектуры активностей и фрагментов приложения лежит использование шаблона проектирования MVVM (от англ. – Model-View-ViewModel). Использование данного архитектурного шаблона позволяет данным активности и её операциям быть менее подверженными потерям в результате изменения состояния жизненного цикла активности [8]. Пример схемы работы архитектуры MVVM в Android приложении представлен на рис. 2.4.

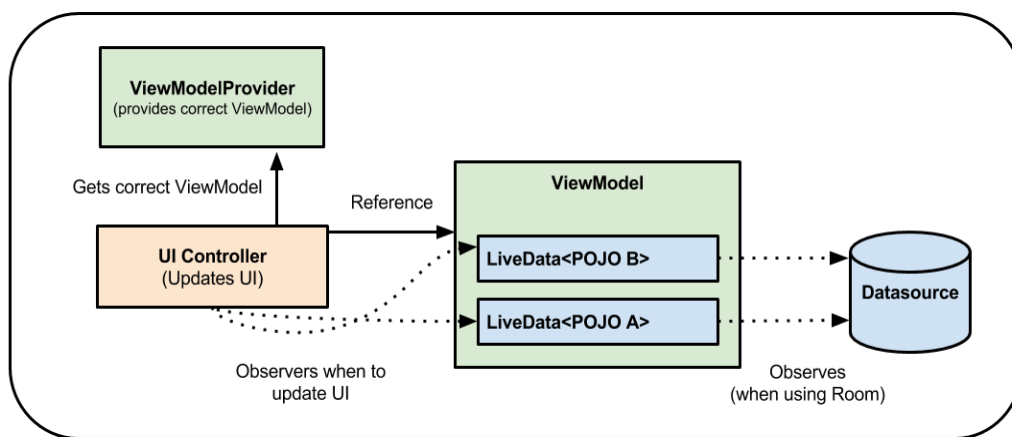


Рисунок 2.4 – Пример схемы работы архитектуры MVVM в Android приложении

ViewModel (в контексте используемого набора библиотек Android Architecture Components) – это объекты, которые предоставляют данные для UI компонентов, таких как активность или фрагмент. Они независимы от жизненного цикла LifecycleOwner’a (т.н. владельца жизненного цикла). Реализация класса такого объекта осуществляется путём наследования от класса ViewModel. Компонент (активность или фрагмент) в процессе работы запрашивает соответствующий ему ViewModel-объект у ViewModelProvider и, если объект не существует в памяти, создаёт объект, иначе берёт объект из памяти. Жизненный цикл объекта ViewModel (и, как следствие, его наследников) приведён на рис. 2.5.

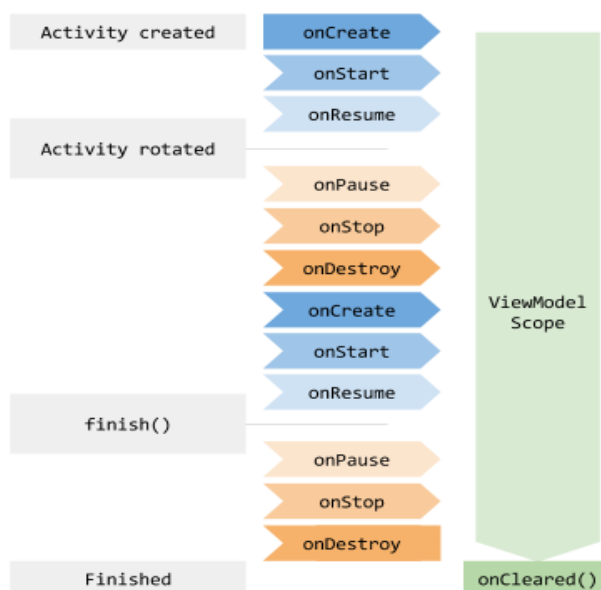


Рисунок 2.5 – Жизненный цикл объекта ViewModel из библиотеки Android Architecture Components в Android приложении

Пояснение к рисунку 2.5: слева направо расположены следующие логические части: состояние компонента, жизненный цикл LifecycleOwner-компонента и жизненный цикл объекта ViewModel. Как можно видеть из данного рисунка, объект ViewModel уничтожается только после уничтожения LifecycleOwner-объекта.

Все экраны приложения (или их составляющие) содержат в себе функционал привязки данных (т.н. Data Binding). Данный функционал позволяет обращаться к элементам экрана из кода более лаконичным и удобным образом, устанавливать значения переменных с использованием кода макета. Данный подход позволяет избавиться от т.н. boilerplate-кода и устанавливать значения переменных наиболее лёгким способом для программиста. Такой подход используется в местах, где нужно показывать данные.

Передача данных между активностями осуществляется с помощью механизма передачи аргументов при инициализации т.н. Intent-а (в русской локализации – «Намерение») на запуск некоторой активности. Intent – это абстрактное определение намерения на запуск активности. При инициализации объекта Intent есть возможность установить аргументы в формате «Ключ-значение». С помощью данного механизма и осуществляется передача аргументов.

Взаимодействие с сервером осуществляется при помощи класс HttpClient, который находится в библиотеке Apache. Для выполнения запроса необходимо в дополнение создать класс HttpGet для отправки GET запроса или HttpPost для отправки POST запроса. Конструктор классов HttpGet и HttpPost принимают URL. После создания одного из объектов, вызвав метод execute у класса HttpClient и передав в параметр HttpGet или HttpPost, мы получаем объект класса HttpResponse, где мы можем узнать ответ от сервера. Для выполнения запросов на сервер реализованы специальные классы, которые наследуются от AsyncTask. Сделано это с той целью, чтобы остальные функции android не ждали ответа от сервера и функционировали независимо от этого.

В android-приложении есть функция скачивания документов, для этой цели был использован класс DownloadManager – стандартный класс android, который позволяет скачать файл в папку «Downloads» телефона.

## **2.2 Функциональная схема**

На рисунке 2.5 представлена функциональная схема, составленная для данной системы. Функциональная схема необходима для того, чтобы показать взаимодействия между пользователями системы.



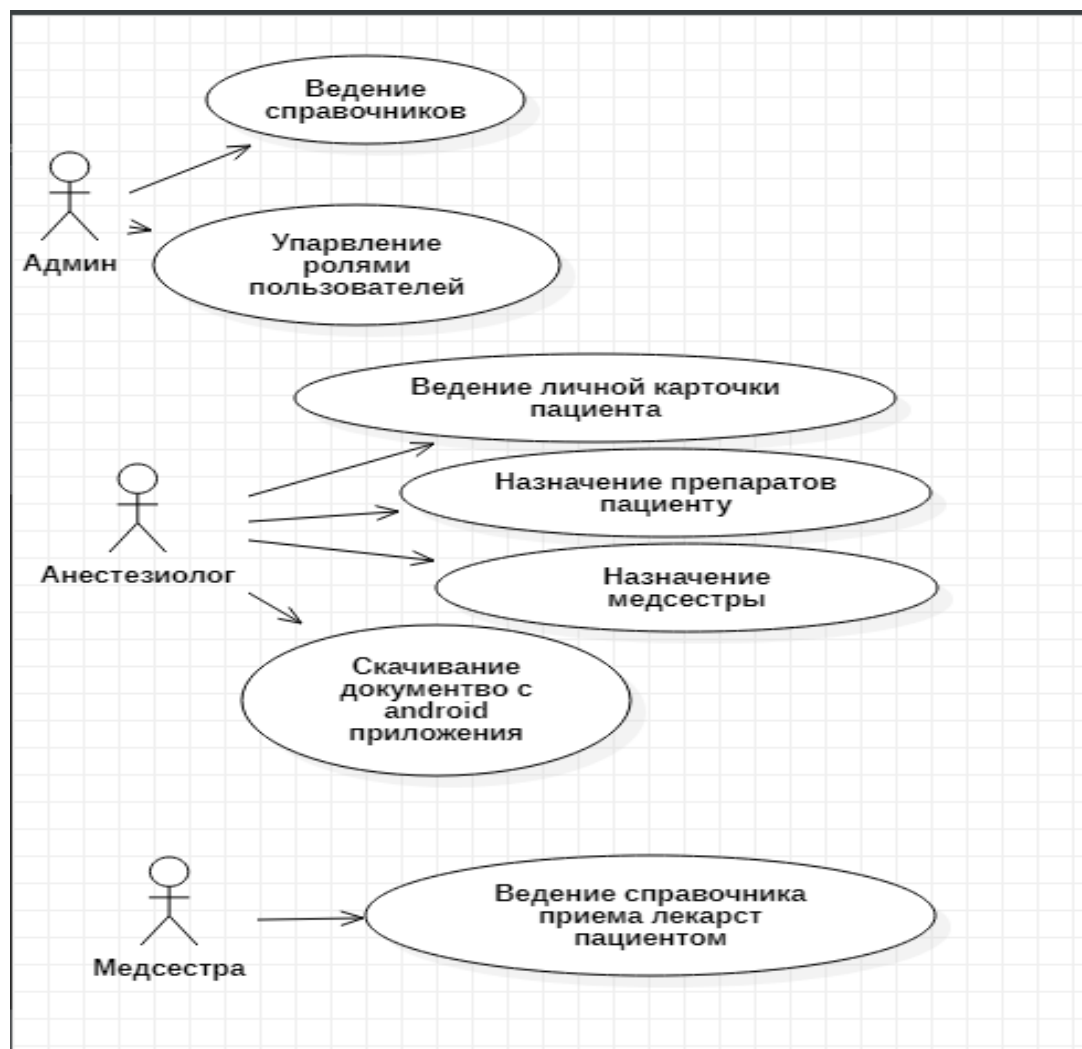


Рисунок 2.5 – Функциональная схема

В данной системе есть 3 роли, которые могут взаимодействовать с системой.

У админа есть возможно ведения нескольких справочников, которые нужны системе, а также другим членам системы (пример медицинские учреждения, болезни, препараты и т.д.). Также имеет возможность распоряжаться правами пользователей, которые находятся в данной системе. Смотреть статистику по пациенту (количество человек на лечении, количество выписанных пациентов за определенный срок и т.д.). И самая основная функция – это предоставление возможности регистрации новым пользователям.

Анестезиолог – это ключевая роль данной системы. Данный врач не только делает анестезии при операции, а также наблюдает их и после операции, поэтому ему нужно ввести так называемую карточку пациента, которая состоит из множества пунктов. Анестезиолог прикрепляется к определенным пациентом, что позволяет им ввести карточку данного пациента, делая заметки и следя за состоянием пациента. Может назначить препараты, которые необходимы для восстановления пациента и чтоб он их принимал, анестезиолог имеет возможность назначит медсестру, которая будет в дальнейшем ответственна за

прием лекарств данного пациента. Что позволит ему смотреть время и дату приема лекарств.

Медсестра – как описано выше, для восстановления пациента, ему нужно принимать определённые препараты определённых доз. Медсестра имеет возможность просматривать нужные препараты для данного пациента, а также их дозы, после приема, она отчитывается в системе, чтоб в дальнейшем анестезиолог имел возможность смотреть статистику приема препаратов.

## **2.3 База данных**

Одна из главных задач множества существующих приложений – хранение данных в каком-либо виде. Базы данных – неотъемлемая часть работы подавляющего большинства программного обеспечения, существующего на сегодняшний день. Использование баз данных является достаточно эффективным способом хранить данные.

Как уже упоминалось ранее, на этапе проектирования схемы данных было установлено, что проектируемые доменные сущности имеют множественные отношения друг между другом. Таким образом предпочтение в выборе СУБД было отдано реляционной базе данных MySQL.

Была спроектирована база данных, состоящая из 12 таблиц, и данное решение полностью удовлетворяет потребности приложения.

Для заполнения базы данных были использованы архивные данные, а также данные, внесённые в базу данных во время тестирования и работы с программой. Структура базы данных представлена на рисунке 2.5.

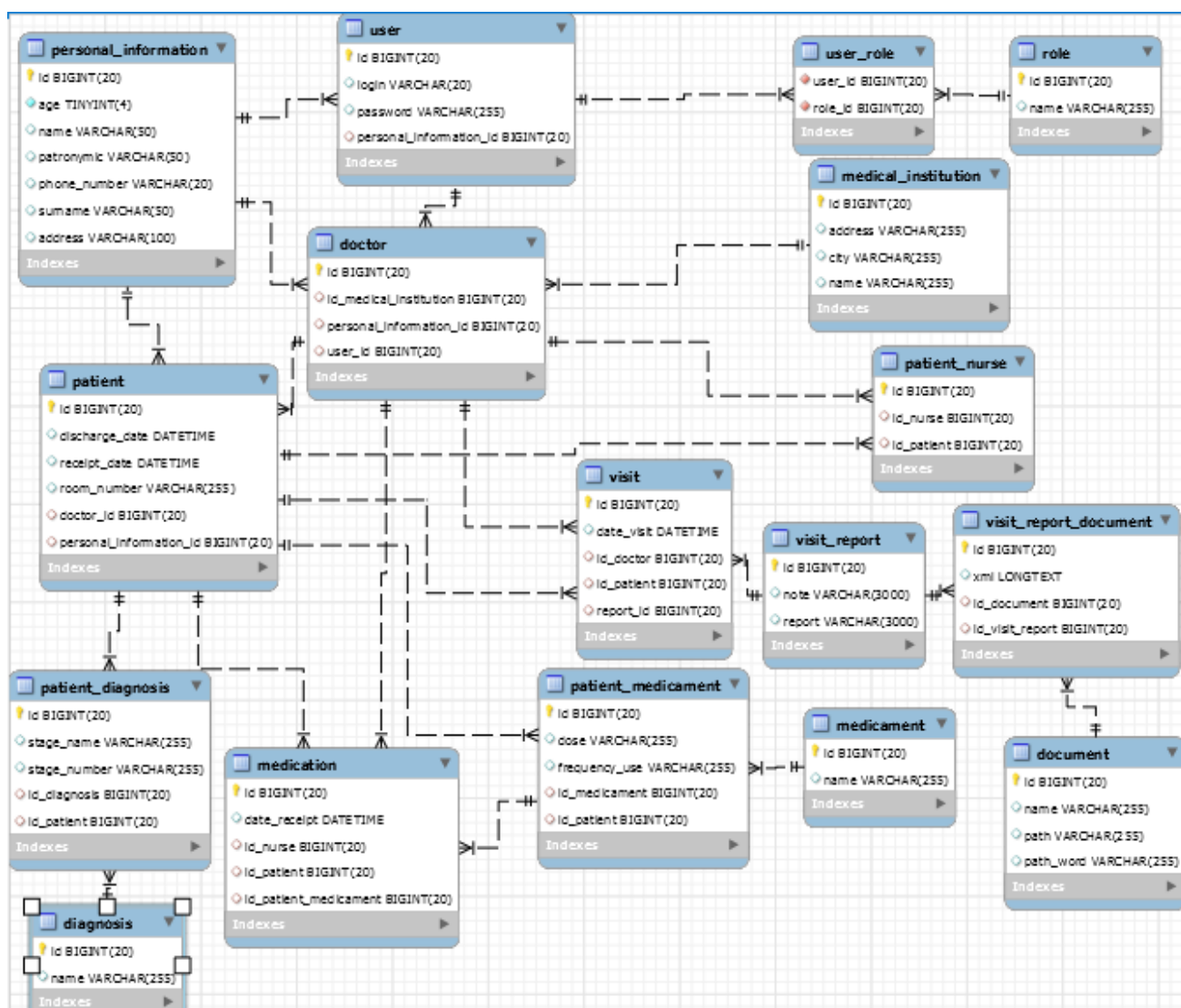


Рисунок 2.5 – Схема базы данных

На данной схеме изображена схема базы данных, используемой в приложении. Рассмотрим ее подробнее.

Таблица *Users* представлена в таблице 2.1.

Таблица 2.1 – Структура таблицы Users

Название поля	Тип переменной	Пояснение
id	INT(11)	Уникальный идентификатор
login	VARCHAR(20)	Логин пользователя
password	VARCHAR(255)	Зашифрованный пароль пользователя
personal_information_id	INT(11)	Внешний ключ на таблицу “personal_information”

*Users* – таблица, в которой хранятся пользователи в системе. Данные пользователи имеют личную информацию, поэтому данная таблица связана с таблицей ‘personal\_information’ связью “один ко многим”.

Таблица *Roles* представлена в таблице 2.2.

Таблица 2.2 – Структура таблицы *Roles*

Название поля	Тип переменной	Пояснение
id	INT(11)	Уникальный идентификатор
name	VARCHAR(30)	Название роли

*Roles* – таблица, в которой хранятся роли в системе. В данной таблице хранится 3 роли: ROLE\_ADMIN, ROLE\_DOCTOR, ROLE\_NURSE.

Таблица *Users\_Roles* представлена в таблице 2.3.

Таблица 2.3 – Структура таблицы *Users\_Roles*

Название поля	Тип переменной	Пояснение
<i>user_id</i>	INT(11)	Внешний ключ на таблицу “Users”
<i>role_id</i>	INT(11)	Внешний ключ на таблицу “Role”

*Users\_Roles* – таблица, которая связывает две таблицы “Users” и “Roles” связью “многие-ко-многим”. В системе имеется возможность иметь несколько ролей у пользователя.

Таблица *Medical\_Institutions* представлена в таблице 2.4.

Таблица 2.4 – Структура таблицы *Medical\_Institutions*

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
city	VARCHAR(255)	Название города, где расположено медицинское учреждение
address	VARCHAR(255)	Адрес медицинского учреждения
name	VARCHAR(255)	Название медицинского учреждения

*Medical\_Institutions* – таблица, которая хранит необходимую информацию об медицинском учреждении.

Таблица *Diagnoses* представлена в таблице 2.5.

Таблица 2.5 – Структура таблицы *Diagnoses*

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
name	VARCHAR(255)	Название болезни или диагноза

*Diagnoses* – таблица, которая хранит названия болезней или диагнозов.

Таблица *Medicaments* представлена в таблице 2.6.

Таблица 2.6 – Структура таблицы Medicaments

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
name	VARCHAR(255)	Название препарата

Medicaments – таблица, которая хранит названия препаратов.

Таблица Documents представлена в таблице 2.7.

Таблица 2.7 – Структура таблицы Documents

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
name	VARCHAR(255)	Название шаблона документа
path	VARCHAR(1000)	Путь на сервере до шаблона документа
path_word	VARCHAR(255)	Путь на сервере до шаблона html

Documents – таблица, которая хранит информацию о шаблонах документа, которые могут быть сгенерированы нашим приложением. Шаблоны хранятся на сервере и на основе их будут формироваться готовые word документы, основываясь на введенных данных пользователей.

Таблица Personal\_Informations представлена в таблице 2.8.

Таблица 2.8 – Структура таблицы Personal\_Informations

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
name	VARCHAR(50)	Имя человека
surname	VARCHAR(50)	Фамилия человека
patronymic	VARCHAR(50)	Отчество человека
age	INT(11)	Возраст человека
phone_number	VARCHAR(50)	Мобильный телефон человека
address	VARCHAR(100)	Адрес

Personal\_Informations – таблица, которая хранит персональную информацию человека (как медицинскому сотруднику, так и пациентам).

Таблица Doctors представлена в таблице 2.9.

Таблица 2.9 – Структура таблицы Doctors

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
medical_institution_id	INT(11)	Внешний ключ на таблицу “Medical_Institutions”
personal_information_id	INT(11)	Внешний ключ на таблицу “Personal_Informations”
user_id	INT(11)	Внешний ключ на таблицу “User”

Doctors – таблица, хранит информацию и медицинских работников в системе. Данная таблица связана с таблицей “Medical\_Institutions” связью “один-ко-многим”, а также с таблицей “Personal\_Informations” связью “один-ко-многим”.

Таблица Patients представлена в таблице 2.10.

Таблица 2.10 – Структура таблицы Patients

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
receipt_date	DATETIME	Дата поступления пациента
room_number	VARCHAR(50)	Номер палаты, где лежит пациент
personal_information_id	INT(11)	Внешний ключ на таблицу “Personal_Informations”
doctor_id	INT(11)	Внешний ключ на таблицу “Doctor”

Patients – таблица, которая хранит информацию о пациентах, которые находятся под наблюдением в системе. Данная таблица связана с таблицей “Personal\_Informations” связью “один-ко-многим”.

Таблица Patients\_Diagnoses представлена в таблице 2.11.

Таблица 2.11 – Структура таблицы Patients\_Diagnoses

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
stage_name	VARCHAR(50)	Название стадии болезни
stage_number	VARCHAR(50)	Номер стадии болезни
id_diagnosis	INT(11)	Внешний ключ на таблицу “Diagnoses”
id_patient	INT(11)	Внешний ключ на таблицу “Patients”

Patients\_Diagnoses – таблица, которая связывает две таблицы “Diagnoses” и “Patients” связью “многие-ко-многим”. У пациента может быть несколько диагнозов.

Таблица Patients\_Medicaments представлена в таблице 2.12.

Таблица 2.12 – Структура таблицы Patients\_Medicaments

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
dose	VARCHAR(50)	Доза препарата необходимая данному пациенту
frequency_use	VARCHAR(50)	Частота употребления
id_medicament	INT(11)	Внешний ключ на таблицу “Medicaments”
id_patient	INT(11)	Внешний ключ на таблицу “Patients”

Patients\_Medicaments – таблица, которая связывает две таблицы “Medicaments” и “Patients” связью “многие-ко-многим”. Данная таблица показывает какие препараты были назначены данному пациенту. У пациента может быть назначено несколько препаратов.

Таблица Medication представлена в таблице 2.13.

Таблица 2.13 – Структура таблицы Medication

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
date_receipt	DATETIME	Дата и время приема лекарства
id_nurse	INT(11)	Внешний ключ на таблицу “Doctor”
id_patient	INT(11)	Внешний ключ на таблицу “Patients”
id_patient_medicament	INT(11)	Внешний ключ на таблицу “Patients_Medicaments”

Medication – таблица, хранит информацию о приеме лекарств данным пациентом. Данная таблица связана с таблицей “Patients” связью “один-ко-многим”, а также с таблицей “Patients\_Medicaments” связью “один ко многим”.

Таблица Visit\_Reports представлена в таблице 2.14.

Таблица 2.14 – Структура таблицы Visit\_Reports

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
note	VARCHAR(1000)	Примечания о состоянии пациента.
report	VARCHAR(3000)	Небольшой отчет о посещении пациента.

Visit\_Reports – таблица, хранит информацию о посещении доктора пациента, а именно небольшой отчет о его состоянии.

Таблица Visit\_Reports\_Documents представлена в таблице 2.15.

Таблица 2.15 – Структура таблицы Visit\_Reports\_Documents

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
xml	LONGTEXT	Данные необходимые для создания word документа из шаблона
id_document	INT(11)	Внешний ключ на таблицу “Documents”
id_visit_report	INT(11)	Внешний ключ на таблицу “Visit_Reports”

Visit\_Reports\_Documents – таблица, которая связывает две таблицы “Documents” и “Visit\_Reports” связью “многие-ко-многим”. Данная таблица показывает какие документы были созданы для данного отчета.

Таблица Visits представлена в таблице 2.16.

Таблица 2.16 – Структура таблицы Visits

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
date_visit	DATETIME	Данные необходимые для создания word документа из шаблона
id_doctor	INT(11)	Внешний ключ на таблицу “Doctors”
id_patient	INT(11)	Внешний ключ на таблицу “Patients”
report_id	INT(11)	Внешний ключ на таблицу “Visit_Reports”

Visits – таблица, которая хранит информацию о посещении доктора пациента, а именно дату посещения и ссылку на отчет о данном посещении. Данная таблица связана с таблицей “Patients” связью “один-ко-многим”, а также с таблицей “Doctors” связью “один-ко-многим”.

Таблица Patient\_Nurse представлена в таблице 2.17.

Таблица 2.17 – Структура таблицы Patient\_Nurse

Название поля	Тип переменной	Пояснение
<i>id</i>	INT(11)	Уникальный идентификатор
id_patient	INT(11)	Внешний ключ на таблицу “Patients”
id_nurse	INT(11)	Внешний ключ на таблицу “Doctor”

Patient\_Nurse – таблица, которая связывает две таблицы “Patients” и “Doctor” связью “многие-ко-многим”. Данная таблица указывает какие медсестры закреплены за пациентом. У пациента может быть закреплено несколько медсестёр.



## 3 РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### 3.1 Серверная часть приложения

Как было сказано ранее, серверная часть приложения написана при помощи Spring фреймворк.

На рисунке 3.1 представлена структура серверной части

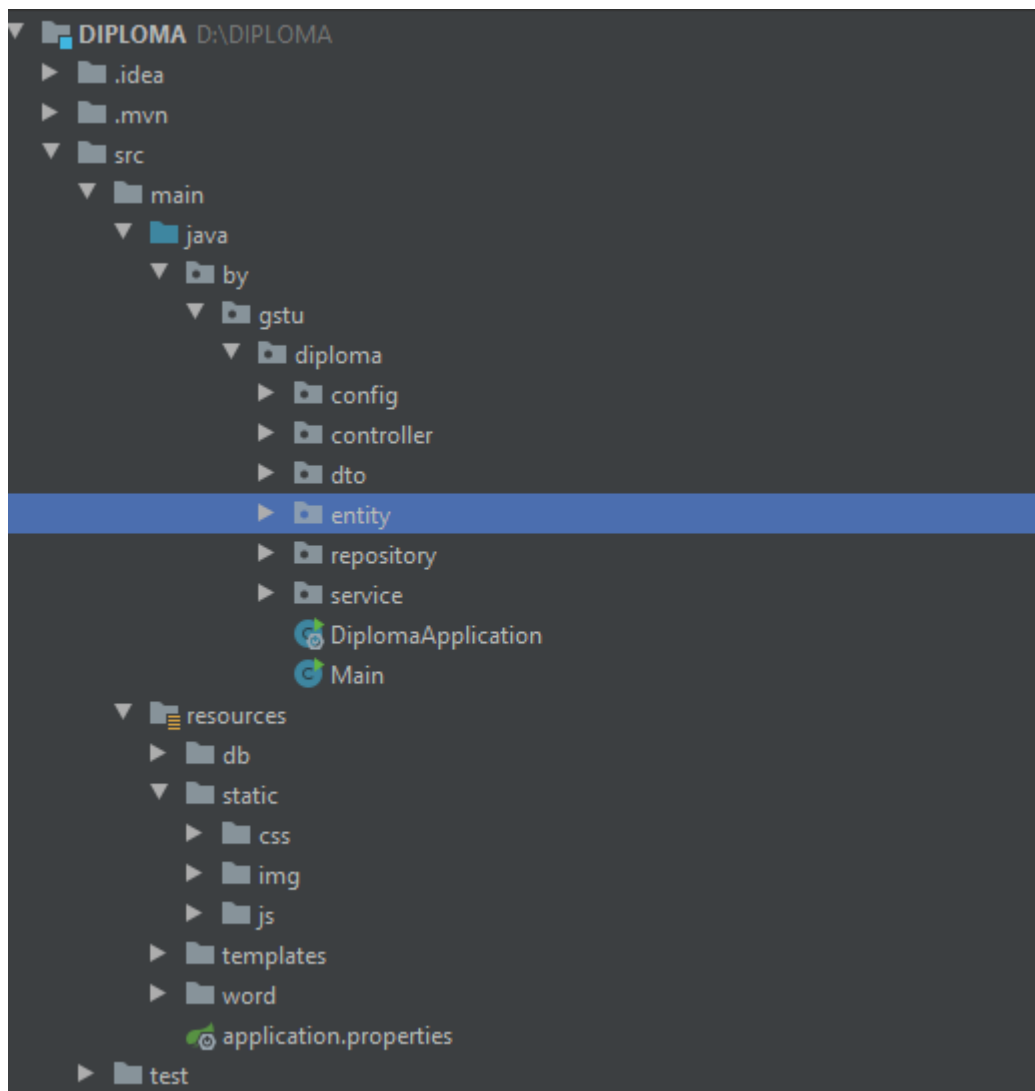
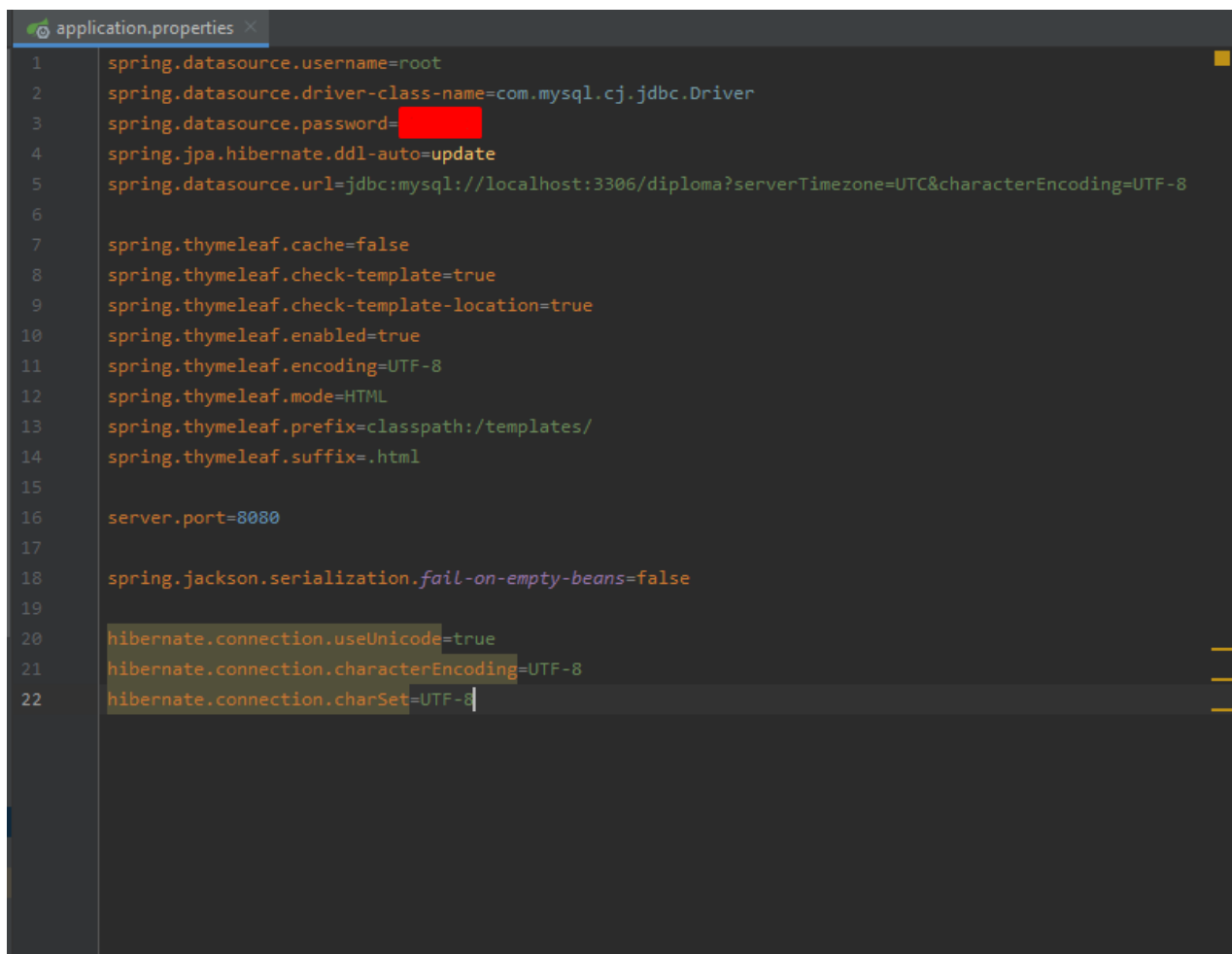


Рисунок 3.1 – Структура серверного приложения

Файл `application.properties` хранит конфигурацию сервера. На рисунке 3.2 представлена конфигурация сервера.



```
1  spring.datasource.username=root
2  spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
3  spring.datasource.password=
4  spring.jpa.hibernate.ddl-auto=update
5  spring.datasource.url=jdbc:mysql://localhost:3306/diploma?serverTimezone=UTC&characterEncoding=UTF-8
6
7  spring.thymeleaf.cache=false
8  spring.thymeleaf.check-template=true
9  spring.thymeleaf.check-template-location=true
10 spring.thymeleaf.enabled=true
11 spring.thymeleaf.encoding=UTF-8
12 spring.thymeleaf.mode=HTML
13 spring.thymeleaf.prefix=classpath:/templates/
14 spring.thymeleaf.suffix=.html
15
16 server.port=8080
17
18 spring.jackson.serialization.fail-on-empty-beans=false
19
20 hibernate.connection.useUnicode=true
21 hibernate.connection.characterEncoding=UTF-8
22 hibernate.connection.charSet=UTF-8
```

Рисунок 3.2 – Файл application.properties

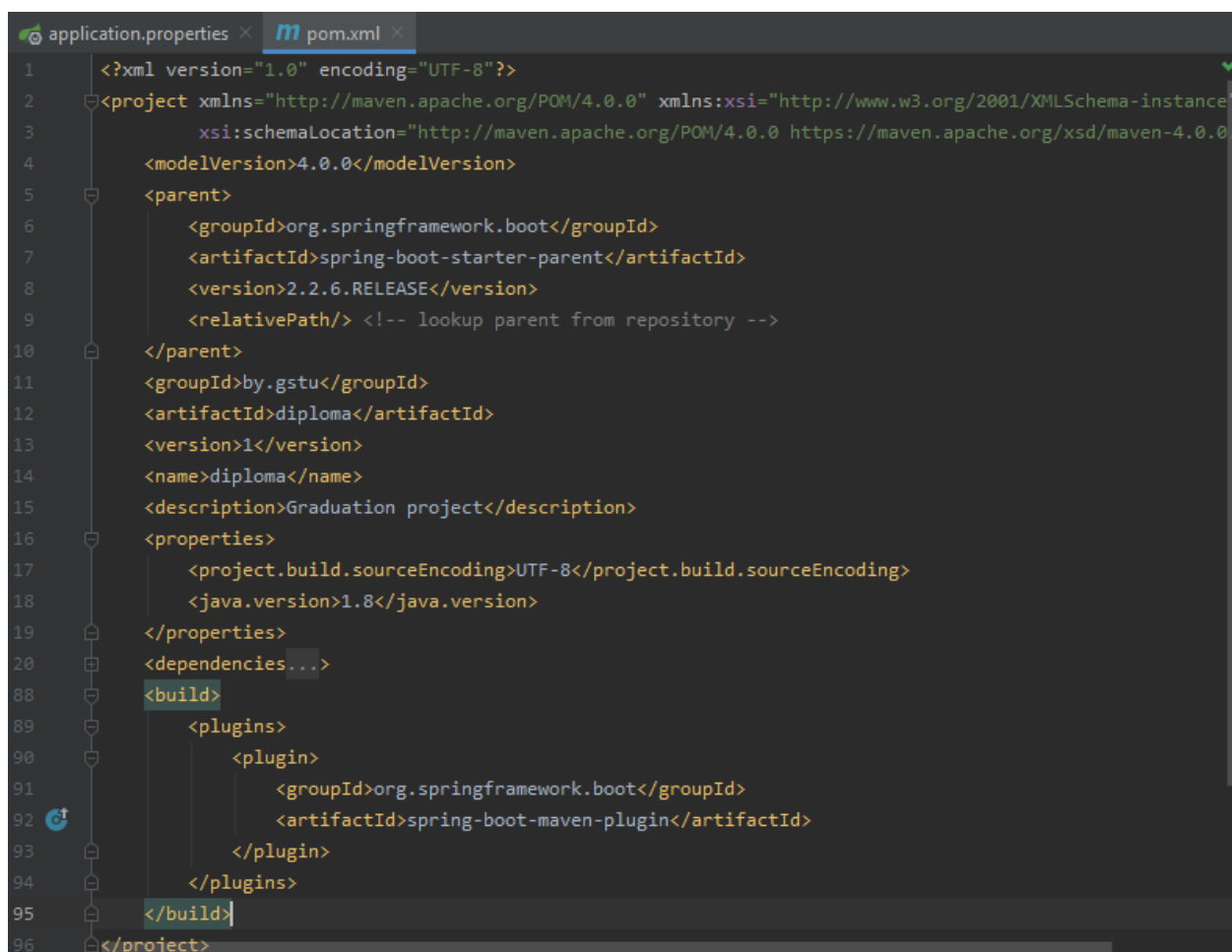
Для сборки проекта использовался фреймворк Apache Maven. Это фреймворк для автоматизации сборки проектов на основе описания их структуры в файлах на языке POM (англ. Project Object Model), являющемся подмножеством XML. Проект Maven издаётся сообществом Apache Software Foundation, где формально является частью Jakarta Project.

Maven обеспечивает декларативную, а не императивную (в отличие от средства автоматизации сборки Apache Ant) сборку проекта. В файлах описания проекта содержится его спецификация, а не отдельные команды выполнения. Все задачи по обработке файлов, описанные в спецификации, Maven выполняет посредством их обработки последовательностью встроенных и внешних плагинов.

Проект, поддерживаемый с помощью Maven, должен удовлетворять некоторым условиям для возможности его чтения утилитой, последующего анализа и возможности сборки. Это накладывает некоторые ограничения на структуру каталогов и требует дополнительных действий, если проект изначально имеет отличную структуру.

Для того, чтобы Maven распознал проект, как подлежащий обработке, он должен содержать установленную структуру каталогов. Все файлы с исходным кодом должны находиться по относительному пути «\src\main\java». Файл

конфигурации Maven-проекта `pom.xml` должен находиться в корневом каталоге проекта. На рисунке 3.3 представлен файл `pom.xml`.



```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3      xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
4      <modelVersion>4.0.0</modelVersion>
5      <parent>
6          <groupId>org.springframework.boot</groupId>
7          <artifactId>spring-boot-starter-parent</artifactId>
8          <version>2.2.6.RELEASE</version>
9          <relativePath/> <!-- lookup parent from repository -->
10     </parent>
11     <groupId>by.gstu</groupId>
12     <artifactId>diploma</artifactId>
13     <version>1</version>
14     <name>diploma</name>
15     <description>Graduation project</description>
16     <properties>
17         <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
18         <java.version>1.8</java.version>
19     </properties>
20     <dependencies>...</dependencies>
88    <build>
89        <plugins>
90            <plugin>
91                <groupId>org.springframework.boot</groupId>
92                <artifactId>spring-boot-maven-plugin</artifactId>
93            </plugin>
94        </plugins>
95    </build>
96 </project>
```

Рисунок 3.3 – Файл `pom.xml`

Для настройки Spring Security был создан специальный класс `SpringSecurityConfig`. На рисунке 3.4 представлен класс `SpringSecurityConfig`. В этом классе указано, что для поиска пользователей в системе нужно использовать `UserServiceDetails`, который имеет в себе всего один метод – это создание специального класса `User`, который предоставляет Spring фреймворк. Также в `SpringSecurityConfig` есть метод `configure` в котором расписаны доступы по ролям. Пример: пользователи смогут получить доступ ко всем URL, которые будут начинаться с `/doctor` в том случае, если они имеют роль `ROLE_DOCTOR`.

```

14
15 @EnableWebSecurity
16 public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {
17
18     @Autowired
19     @Qualifier(value = "myUserDetailsService")
20     private UserDetailsService userDetailsService;
21
22     @Override
23     protected void configure(AuthenticationManagerBuilder auth) throws Exception {
24         auth.userDetailsService(userDetailsService);
25     }
26
27     @Override
28     protected void configure(HttpSecurity http) throws Exception {
29         http
30             .authorizeRequests() ExpressionUrlAuthorizationConfigurer<H>.ExpressionInterceptUrlRegistry
31             .antMatchers(
32                 ...antPatterns: "/",
33                 "/registration",
34                 "/js/**",
35                 "/css/**",
36                 "/img/**",
37                 "/webjars/**",
38                 "/user").permitAll()
39             .antMatchers(...antPatterns: "/doctor/**").hasRole("DOCTOR")
40             .antMatchers(...antPatterns: "/admin/**", "/js/admin/**").hasRole("ADMIN")
41             .antMatchers(...antPatterns: "/nurse/**").hasRole("NURSE")
42             .anyRequest().authenticated()

```

Рисунок 3.4 – Файл SpringSecurityConfig

Для каждой сущности были созданы свои репозитории (рис. 3.5.), которые общаются с БД при помощи Spring Data фреймворк. Преимущество данного фреймворка в том, что он в себе имеет стандартную реализацию CRUD-операция, что облегчает и ускоряет разработку. Для создания репозитория, который будет использовать Spring Data фреймворк, нужно создать интерфейс, который будет наследоваться от интерфейса «JpaRepository<E, I>», где E – это класс сущности, а I – класс идентификатора (id).

Для генерации методов, с запросом, который выходит за рамки CRUD-операция используется аннотация «@Query», в которой при помощи SQL подобного языка HQL можно написать свой запрос. Преимущество HQL над SQL в том, что при написании запроса мы руководствуемся Entity, а не таблицами БД. Пример такого запроса «@Query("select d from Doctor d join fetch d.patients as p join fetch p.personalInformation")» (рис. 3.6.). Данный запрос выбирает всех медработников и присоединяет сущности пациенты и личную информацию пациентов к выборке.

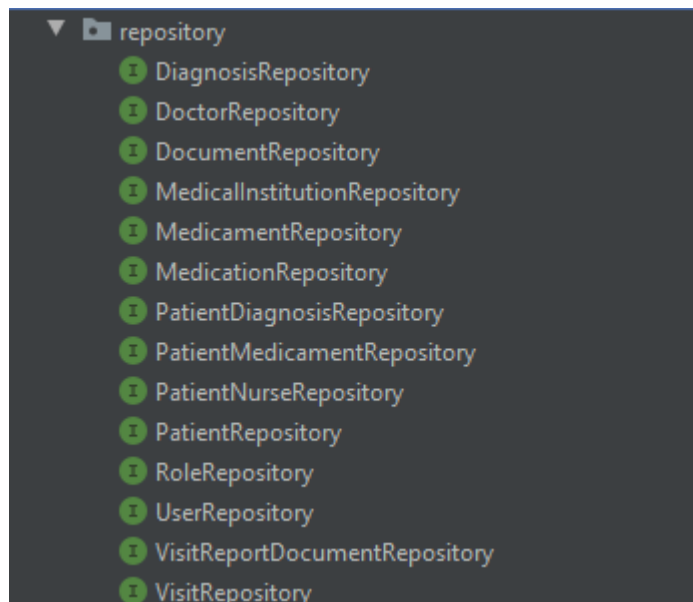


Рисунок 3.5 – Репозитории

```
@Repository
public interface DoctorRepository extends JpaRepository<Doctor, Long> {

    @Query("select d from Doctor d join fetch d.patients as p join fetch p.personalInformation")
    Doctor findWithAllPatientByUserId(Long id);

}
```

Рисунок 3.6 – Пример HQL запроса

Сервисы разделены на несколько пакетов. В пакете «interfaces» располагаются интерфейсы, которые реализуют сервисы, разработанные на данном сервере. Сервисы предназначены для выполнения бизнес логики приложения. Структура сервисов представлена на рисунке 3.7.

Для примера одного из сервисов на рисунке 3.8. представлен пример сервиса MyUserDetailsService. В данном сервисе происходит поиск пользователя по его логину, если такого пользователя не существует, то выбрасывается ошибка UsernameNotFoundException название которого говорит само за себя. Далее если пользователь все-таки существует, то дальше создаются специальная коллекция, состоящая из SimpleGrantedAuthority классов. SimpleGrantedAuthority используется для указания роли, которой владеет данный пользователь, этот класс принадлежит Spring фреймворк. И в конце создается специальный класс Spring Security User, который содержит логин, пароли и роли пользователя, который авторизовался в системе.

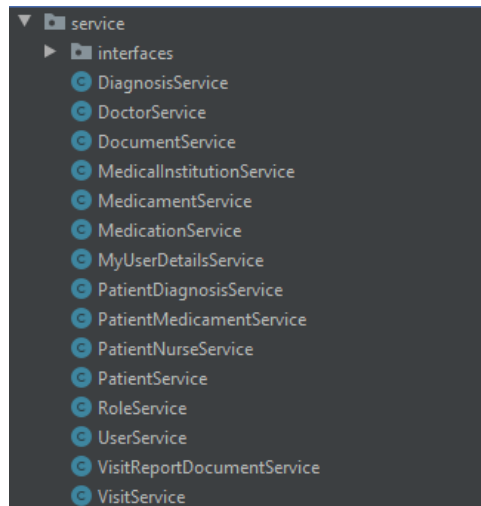


Рисунок 3.7 – Структура пакета сервисов

```
@Service
public class MyUserDetailsService implements UserDetailsService {

    @Autowired
    private UserRepository userRepository;

    @Override
    public UserDetails loadUserByUsername(String login) throws UsernameNotFoundException {

        Optional<User> optionalUser = userRepository.findWithRolesByLogin(login);

        optionalUser.orElseThrow(() -> new UsernameNotFoundException("Not found: " + login));

        User user = optionalUser.get();

        Set<SimpleGrantedAuthority> grantedAuthorityList = new HashSet<>();

        for (Role role : user.getRoles()) {
            grantedAuthorityList.add(new SimpleGrantedAuthority(role.getName().name()));
        }

        return new org.springframework.security.core.userdetails.User(user.getLogin(), user.getPassword(
    }
}
```

Рисунок 3.8 – сервис MyUserDetailsService

Для обработки запросов использовался Spring MVC фреймворк, который реализует шаблон MVC. Все запросы обрабатываются в контроллерах. Структура контроллеров представлена на рисунке 3.9. Контроллеры разделены на пакеты admin, doctor, nurse, сделано это с той целью, чтобы разграничить их по ролям, которые могут быть у пользователей. К примеру пакет admin отвечает за все запросы, которые могут сделать только тех пользователи, у которых есть роль ROLE\_ADMIN.

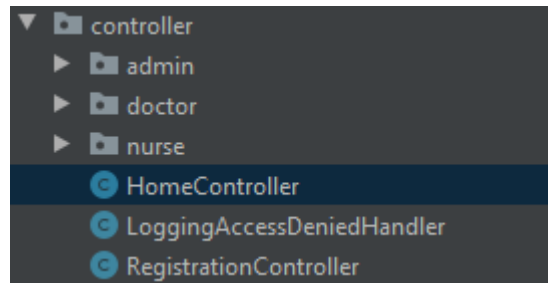


Рисунок 3.9 – Структура контроллеров

Для построения путей используются специальные аннотации над методами, которые их будут обрабатывать. Пример GET запроса представлен на рисунке 3.10. Данный метод возвращает представление, которое будет отображать диагнозы для конкретного пациента. Spring MVC позволяет обрабатывать динамические запросы, которые удовлетворяют шаблону. В данном примере используется такой запрос. Данный URL содержит в себе {id} конструкцию, которая подразумевает, что между «/doctor/patient/» и «/disease» будет находиться какая-то строка, в данном случае я назвал ее id, так как это строка является id пациента. Для получения доступа к ней, нужно передать в параметры данного метода переменную, которая будет соответствовать типу данных и указать данному параметру аннотацию @PathVariable.

```
@GetMapping("/doctor/patient/{id}/disease")
public String getMedicamentHtml(@PathVariable Long id, Model model) {
    model.addAttribute("id", id);
    return "/doctor/disease/disease";
}
```

Рисунок 3.10 – Пример GET запроса

Пример POST запроса представлен на рисунке 3.11. Данный метод меняет поставленный диагноз на другой диагноз. Схема построения URL точно такая же, как и для GET, который был описан выше. Для передачи информации, которая была взята из формы HTML страницы используется специальный механизм Spring MVC в связке с одним из Spring Template таким, как Thymeleaf. Spring и Thymeleaf позволяют сопоставить DTO с input-ами формы HTML страницы. После отправки данных с HTML страницы Spring сопоставляет данные взятые с формы с полями DTO. В данном примере вы можете увидеть такой объект как UpdatePatientDiagnosisRequestDto, которые является одним из DTO. Для сопоставления данных с DTO, необходимо передать его, как параметр метода и указать аннотацию @ModelAttribute.

```

@RequestMapping(value = "/doctor/patient/{id}/disease/update", method = RequestMethod.POST)
public String updateDiagnosis(@PathVariable Long id, @ModelAttribute UpdatePatientDiagnosisRequestDto requestDto) {
    patientDiagnosisService.update(dto, id);
    return "redirect:/doctor/patient/" + id + "/disease";
}

```

Рисунок 3.11 – Пример POST запроса

DTO классы разделены на пакеты и представлены на рисунке 3.12.

Схема разделения похожа на схему контроллеров, за одним исключением, что все ещё разделено на DTO, которые используются в запросах и в ответах.

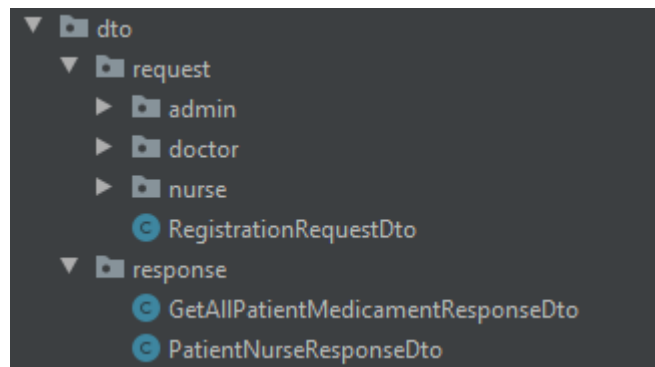


Рисунок 3.12 – Структура DTO классов

Для генерации документов использовалась библиотека Doc4j, а также стандартные функции разработчика в MS Word. Для этого были созданы шаблоны MS Word, а также аналоги HTML-страниц. Сопоставление полей происходит через XML, который генерирует клиент и сохраняет в БД.

### 3.2 Android приложение

Android приложение разделено на несколько частей: layout (это разметка для отображения элементов на экране телефона) и классы, которые обрабатывают эти layout. Структура проекта представлена на рисунке 3.13.



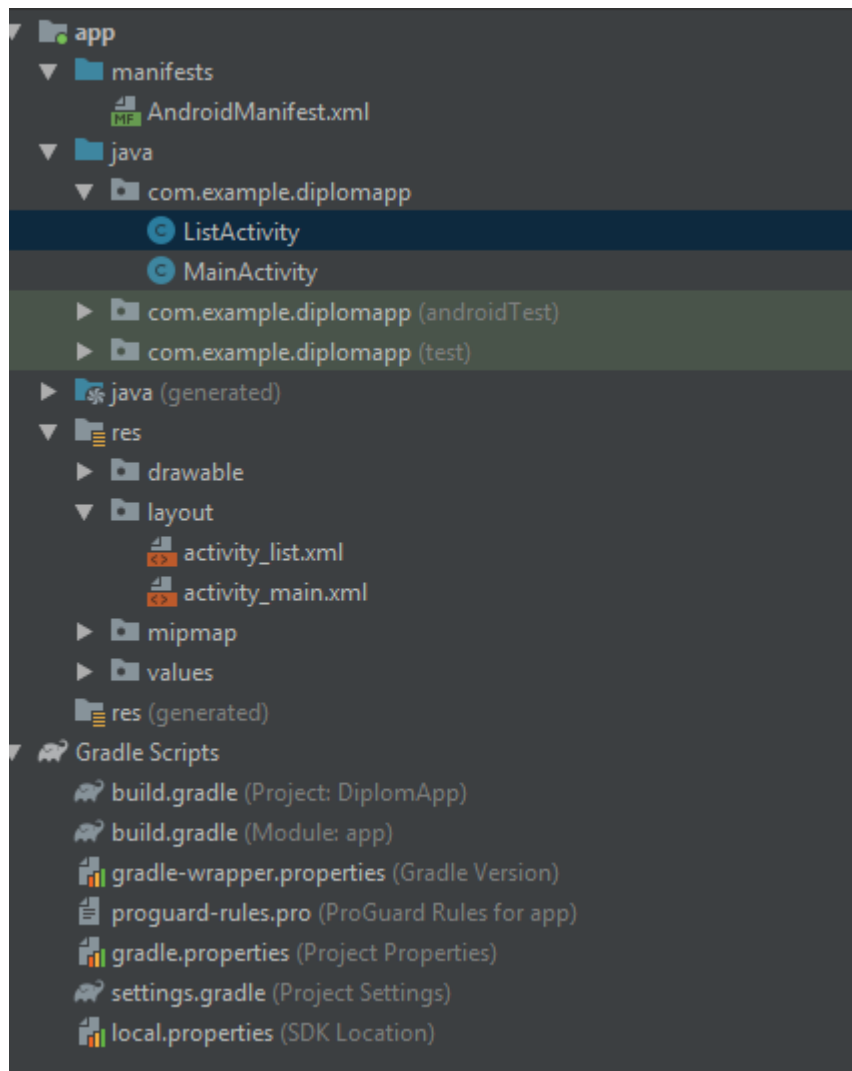
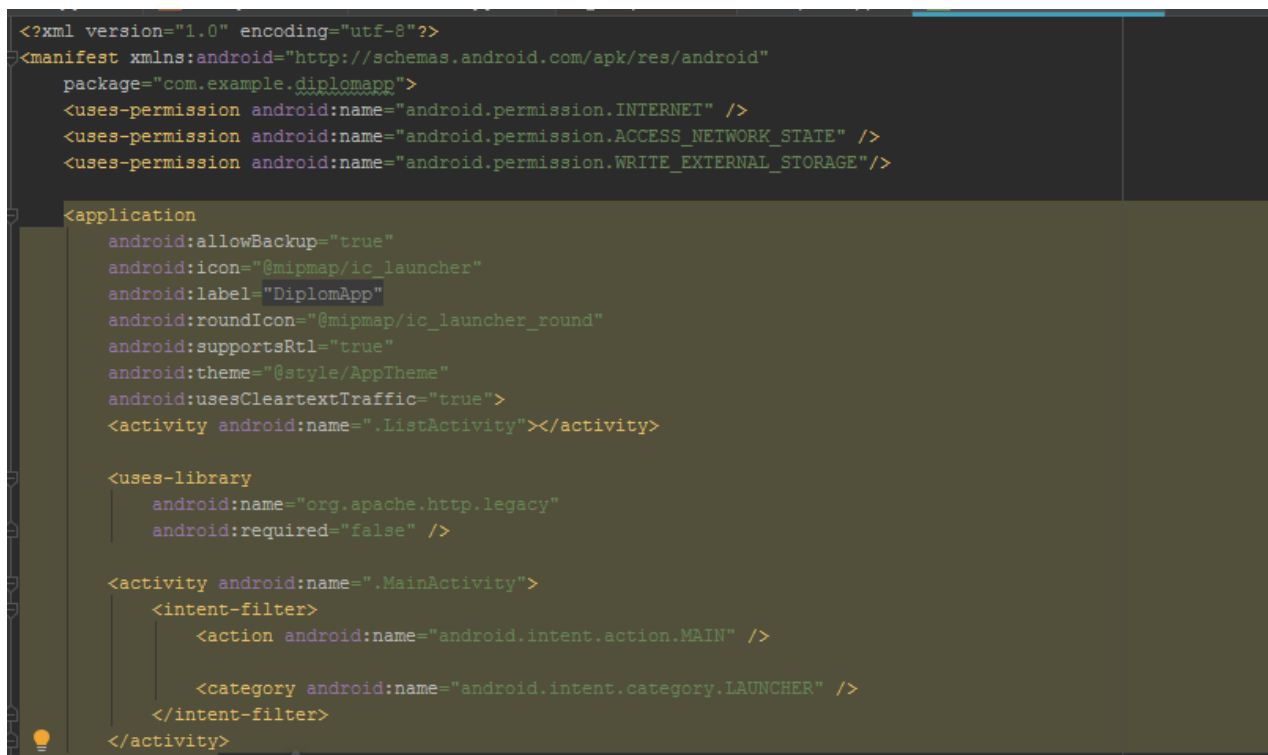


Рисунок 3.13 – Структура android приложения

Основной файл в данном проекте – это файл `AndroidManifest.xml`. Этот файл описывает конфигурацию приложения, его разрешение, а также описание Activity. Данный файл представлен на рисунке 3.14. Android приложение необходимо иметь доступ к интернет соединению, поэтому в манифест файле прописаны следующий разрешения `<<uses-permission android:name="android.permission.INTERNET" />>` и `<<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />>`. Приложение поддерживает скачивание файлов и для их сохранения в памяти телефона тоже необходимы следующее разрешение `<<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>>`.

Приложение состоит из двух Activity. Первое activity отвечает за авторизацию на сервер. Если у вас нет роли `ROLE_DOCTOR`, то вы не сможете авторизоваться в системе. Второе activity предоставляет выбрать некоторые параметры, чтобы узнать какой из документов вам необходим к скачиванию. Сначала вы выбираете пациента, после чего вам погружаются посещения данного пациента, после выбора посещения, вам подгружаются документы на

это посещение, после чего вы можете нажать клавишу «Скачать» и данный документ скачается в папку «Downloads».



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.diplomapp">
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="DiplomApp"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme"
        android:usesCleartextTraffic="true">
        <activity android:name=".ListActivity"></activity>

        <uses-library
            android:name="org.apache.http.legacy"
            android:required="false" />

        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Рисунок 3.14 – файл AndroidManifest.xml

Во время выбора документа может произойти ситуация, когда данные могут поменяться. Для этого предусмотрена специальная кнопка «Обновить», которая обновит список пациентов, после чего вы сможете сделать снова выбор, но уже с обновленными данными.

## **4 ТЕСТИРОВАНИЕ И ВЕРИФИКАЦИЯ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

Тестирование программного обеспечения — проверка соответствия между реальным и ожидаемым поведением программы, осуществляемая на конечном наборе тестов, выбранном определенным образом. В более широком смысле, тестирование — это одна из техник контроля качества, включающая в себя активности по планированию работ (Test Management), проектированию тестов (Test Design), выполнению тестирования (Test Execution) и анализу полученных результатов (Test Analysis). [9]

Для тестирования могут быть использованы статический и динамический подходы. Динамический подход включает в себя запуск программного обеспечения. Статистическое тестирование включает в себя проверку синтаксиса и другие особенности кода программы.

Тестирование может быть функциональным и не функциональным. Функциональное тестирование – это проверка рабочей области программного обеспечения. Не функциональное тестирование – проверка производительности, совместимости и безопасности тестируемой системы.

Также тестирование может быть ручным и автоматизированным.

Ручное тестирование – это процессы через которые разработчики, или Manual QA тестировщик тестируют продукт: вебсайт, платформу, SaaS, что угодно чтобы найти дефекты и ошибки. Ручное тестирование проходит от лица тестировщика, который выступает как конечный пользователь системы. Проверяет все функции, ссылки, пункты меню и другое, с целью избежать поломанных ссылок, или не рабочего функционала. [10]

Автоматизированное тестирование – это процессы, которые запускают программы и скрипты для тестирования отдельных модулей, используя повторяющиеся действия. Фактически, это значит, что программа запускает определенные скрипты, чтобы проверить все составляющие проекта и оценить его. Для того, чтобы создать программу тестирования требуются определенные ресурсы. В автоматизированном тестировании должен присутствовать тестировщик, который создаст программу и затем будет ее запускать. [10]

Одним из видов автоматизированного тестирования является модульное тестирование – вид тестирования, который проверяет функциональность и ищет дефекты в частях приложения, которые доступны и могут быть протестированы поотдельности (модули программ, объекты, классы, функции и т.д.). [9]

### **4.1 Ручное тестирование программного комплекса в режиме админ**

Режим админа позволяет вести справочники и изменять роли пользователей. Для получения доступ к странице админа необходимо авторизоваться (рисунок 4.1):

# Страница авторизации

Логин:

Пароль:

[Регистрация](#)  
[Вернуться на главную](#)

Рисунок 4.1 – Страница авторизации пользователя

Если пользователь введет неправильный логин или пароль, то получит следующую ошибку (рис. 4.2).

---

# Страница авторизации

Неправильное имя пользователя или пароль.

Логин:

Пароль:

[Регистрация](#)  
[Вернуться на главную](#)

Рисунок 4.2 – Ошибка авторизации

После авторизации на верхней панели вам будет доступна ссылка для перехода на страницу админа (Рисунок 4.3).

Рисунок 4.3 – Ссылка на страницу админа

После перехода на страницу админа верхняя панель поменяется в соответствии с меню админа (рис. 4.4).

Рисунок 4.4 – Панель управления админа

Для просмотра и редактирования диагнозов, нужно нажать на ссылку «Диагнозы». Данная ссылка ведет на таблицу, где отображаются все диагнозы, которые были созданы на сервере (рис. 4.5). На данной странице админ имеет возможность удалить диагноз, изменить диагноз или добавить новый диагноз.

Главная Диагнозы Медицинские Учреждения Медикаменты Пользователи					Выйти
№	Название Диагноза	Изменение	Удаление	Добавить новый диагноз	
1	Термический ожог	Update	Delete		
2	Токсическое поражение нервной системы	Update	Delete		

Рисунок 4.5 – Страница диагнозов

Для добавления нового диагноза необходимо нажать кнопку зеленого цвета с надписью: «Добавить новый диагноз». После чего админ перейдет на страницу добавления нового диагноза (рис. 4.6.). Если ввести название, которое не соответствует шаблону, то запрос на добавление отправлен не будет, а будет выведено сообщение с ошибкой (рис. 4.7.). Добавим новый диагноз «Кома», введет данное название в поле и нажмем зеленую кнопку «Создать диагноз». После добавления админ автоматически перейдет на страницу всех диагнозов (рис. 4.8.).

Диагноз

Название Диагноза

Название диагноза не может быть короче 2 символов. Может содержать латиницу и кириллицу.

Создать Диагноз

Рисунок 4.6 – Страница добавления диагноза

Диагноз

1

Название диагноза не может быть короче 2 символов. Может содержать латиницу и кириллицу.

Название диагноза введено неправильно.

Создать Диагноз

Рисунок 4.7 – Ошибка добавления нового диагноза

№	Название Диагноза	Изменение	Удаление	Добавить новый диагноз
1	Термический ожог	Изменить	Удалить	
2	Токсическое поражение нервной системы	Изменить	Удалить	
3	Кома	Изменить	Удалить	

Рисунок 4.8 – Страница диагнозов с новым диагнозом

Если админ ввел неправильное название для диагноза, то это можно с легкостью исправить, нажав кнопку «Изменить». После нажатия админа перебросить на страницу редактирования данного диагноза, где будет уже заполнено название диагноза (рис. 4.9.). После исправление названия, нажав клавишу «Изменить Диагноз», данный диагноз изменит имя и админа перебросить снова на страницу всех диагнозов.

Название Диагноза

Кома

Изменить Диагноз

Рисунок 4.9 – Страница диагнозов с новым диагнозом

Для удаления диагноза необходимо нажать клавишу «Удалить». После чего данный диагноз удалится из таблицы и из база данных на сервере.

Данная логика применима и для других справочников, таких как «Медикаменты» и «Медицинские учреждения». Поэтому их рассмотрение будет пропущено.

Админ имеет возможность управлять ролями зарегистрированных пользователей в системе, может удалять и добавлять роли, но за исключением роли админа. После нажатия ссылки «Пользователи» админ переходит на страницу всех пользователи в системе, где отображаются роли пользователей, которые можно изменить (рис. 4.10).

Логин	Имя	Фамилия	Отчество	Телефон	Возраст	Роль Доктор	Роль Медсестра
Mizak	Alex	Kazimov	Ruslanovich	+375(29)-891-32-84	21	Удалить роль доктора	Добавить роль медсестры
Asmanser	Зина	Огурцова	Степановна	+375(45)894-84-11	25	Добавить роль доктора	Удалить роль медсестры

Рисунок 4.10 – Страница всех пользователей

Для добавления роли необходимо нажать на кнопку «Добавить роль доктора» или «Добавить роль медсестры». Для удаления ролей необходимо нажать на кнопку «Удалить роль доктора» или «Удалить роль медсестры» (рис. 4.11).

Логин	Имя	Фамилия	Отчество	Телефон	Возраст	Роль Доктор	Роль Медсестра
Mizak	Alex	Kazimov	Ruslanovich	+375(29)-891-32-84	21	Добавить роль доктора	Удалить роль медсестры
Asmanser	Зина	Огурцова	Степановна	+375(45)894-84-11	25	Удалить роль доктора	Добавить роль медсестры

Рисунок 4.11 – Измененные роли пользователей

## 4.2 Ручное тестирование программного комплекса в режиме доктор

Режим пользователя позволяет вести карточку пациента. Для получения доступ к странице доктора необходимо авторизоваться. Процесс авторизации описан выше. После авторизации в верхней панели появляется ссылка «Доктор», которая ведет на страницу, доступная только пользователям, у которых есть роль доктор. На данной странице отображаются все пациенты данного пользователя, у которых вы можете подробно просмотреть информацию, либо создать нового пользователя (рис. 4.12).

№	Имя	Фамилия	Отчество	Возраст	Номер палаты	Дата поступления	Добавить пациента
1	Иван	Иванович	Иванов	25	15а	26-05-2020 18:18	Info

Рисунок 4.12 – Страница всех пациентов

Чтобы добавить нового пациента в систему необходимо нажать кнопку «Добавить пациента», после нажатия доктора перенаправит на страницу создания нового пользователя, где будет предложено ввести необходимые данные (рис. 4.13).

Имя	<input type="text" value="Имя Пациента"/> <small>Имя пациента не может быть короче 1 символа. Может содержать латиницу и кириллицу.</small>
Фамилия	<input type="text" value="Фамилия Пациента"/> <small>Фамилия пациента не может быть короче 1 символа. Может содержать латиницу и кириллицу.</small>
Отчество	<input type="text" value="Отчество Пациента"/> <small>Отчество пациента не может быть короче 1 символа. Может содержать латиницу и кириллицу.</small>
Возраст	<input type="text" value="0"/>
Адрес	<input type="text" value="Адрес Пациента"/> <small>Адрес пациента не может быть короче 2 символов. Может содержать латиницу и кириллицу.</small>
Телефон	<input type="text" value="Телефон Пациента"/>
Дата поступления	<input type="text" value="дд.мм.гггг --:--"/>

Рисунок 4.13 – Страница добавления пациентов

Если какие-то поля будут не удовлетворять требованиям, то данные поля будут подсвечены красным с сопровождающим текстом (рис. 4.14).

Имя	<input type="text" value="Имя Пациента"/> <small>Имя пациента не может быть короче 1 символа. Может содержать латиницу и кириллицу. Имя не удовлетворяет описанным выше требованиям.</small>
Фамилия	<input type="text" value="Фамилия Пациента"/> <small>Фамилия пациента не может быть короче 1 символа. Может содержать латиницу и кириллицу. Фамилия не удовлетворяет описанным выше требованиям.</small>
Отчество	<input type="text" value="Отчество Пациента"/> <small>Отчество пациента не может быть короче 1 символа. Может содержать латиницу и кириллицу. Отчество не удовлетворяет описанным выше требованиям.</small>
Возраст	<input type="text" value="0"/>
Адрес	<input type="text" value="Адрес Пациента"/> <small>Адрес пациента не может быть короче 2 символов. Может содержать латиницу и кириллицу.</small>
Телефон	<input type="text" value="Телефон Пациента"/>

Рисунок 4.14 – Ошибка заполнения полей пациента

Добавим нового пользователя «Пупкин Вася Петрович». После ввода данных необходимо нажать кнопку «Создать пациента» после чего доктора перебросит на страницу отображения всех пациентов (рис. 4.15).



№	Имя	Фамилия	Отчество	Возраст	Номер палаты	Дата поступления	Добавить пациента
1	Иван	Иванович	Иванов	25	15а	26-05-2020 18:18	Информация
2	Вася	Пупкин	Петрович	21	27г	8-06-2020 2:41	Информация

Рисунок 4.15 – Страница всех пациентов с новым пациентом

Для просмотра информации необходимо нажать кнопку «Информация» после чего доктора перенаправит на страницу данного пациента (рис. 4.16). На странице пациента доктор может наблюдать личную информацию пациента, список посещения данного пациента. Доктор имеет возможность изменить личные данные пациента, назначить медикаменты, поставить диагноз и назначить медсестер.

### Посещения

Добавить посещение

### Информация

Имя:Вася
Возраст:21

Фамилия:Пупкин
Адрес:Г. Гомель, п-кт Речицкий 135

Отчество:Петрович
Телефон:+375(45)894-84-11

Описание:

Примечание:

Изменить посещение

Просмотр документов

Изменить пациента

Медикаменты

Диагнозы

Медсестры

Прием лекарств

Рисунок 4.16 – Страница пациента

Для добавления посещения необходимо нажать кнопку «Добавить посещение» после чего доктора перебросит на страницу добавления посещения (рис. 4.17.). Введя все необходимые данные и пройдя валидацию, посещение будет добавлено в базу данных и будет отображаться в списке на странице пациента (рис. 4.18). При нажатии на данные посещения, будет подгружаться информация о посещении и заполняться в соответствующие поля, а также будут доступны функции, которые относятся только к данному посещению, а именно «Изменить посещение», где можно изменить посещение и «Просмотр документов», где можно редактировать документы, созданные на данное посещение (рис. 4.18).

Дата посещения

ДД.ММ.ГГГГ --:--

Описание

Рисунок 4.17 – Страница добавления посещения

Посещения

08-06-2020 02:50

Добавить посещение

Информация

Имя:Вася

Возраст:21

Фамилия:Пупкин

Адрес:Г. Гомель, п-кт Речицкий 135

Отчество:Петрович

Телефон:+375(45)894-84-11

Описание:

Плановый осмотр

Примечание:

Жалоб у пациента нет

Изменить посещение

Просмотр документов

Изменить пациента

Медикаменты

Диагнозы

Медсестры

Прием лекарств

Рисунок 4.18 – Страница пациента с добавленным посещением

Для постановки диагноза необходимо нажать кнопку «Диагнозы», чтобы перейти на страницу диагнозов для данного пациента (рис. 4.19). На данный момент таблица пустая, так как пациент был только что создан. Для добавления диагноза необходимо нажать кнопку «Добавить диагноз», после чего доктора перебросит на страницу добавления диагнозов (4.20). Для добавления необходимо выбрать диагноз из предложенного списка, а также заполнить стадию и название стадии диагноза, если таковые имеются. Если диагноз уже был добавлен, то в общем списке диагнозов он отображаться не будет. Добавим несколько диагнозов для данного пациента. После чего доктора перебросит на страницу всех диагнозов (рис. 4.21).

№	Название	Название стадии	Номер стадии	Изменить	Удалить	Добавить диагноз
						Назад

Рисунок 4.19 – Диагнозы пациента

Диагноз	<input type="text" value="Термический ожог"/>
Номер стадии болезни	<input type="text" value="Введите номер стадии диагноза"/>
Название стадии	<input type="text" value="Введите название стадии диагноза"/>
<input type="button" value="Добавить Диагноз"/>	

Рисунок 4.20 – Страница добавления диагноза

№	Название	Название стадии	Номер стадии	Изменить	Удалить	Добавить диагноз
1	Термический ожог	степени	1	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>	
2	Токсическое поражение нервной системы	стадии	2	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>	

Рисунок 4.21 – Страница диагнозов пациента

После добавления диагнозов, данные диагнозы отображаются в таблице, при необходимости их можно изменить или вовсе удалить.

Добавление медикаментов работают по схожему принципу, что и диагнозы. Медикаментов можно добавить несколько штук (рис. 4.22).

№	Название	Доза	Частота употребления	Изменить	Удалить	Добавить медикамент
1	Левомеколь	500	2 раза в день	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>	
2	Пантенол	1000	1 раз в день	<input type="button" value="Изменить"/>	<input type="button" value="Удалить"/>	

Рисунок 4.22 – Страница медикаментов пациента

Добавление медсестёр работает по схожему принципу, что диагнозы и медикаменты. Медсестёр можно добавить несколько штук (рис. 4.23). После добавление медсестры для данного пациента, данный пациент будет отображаться у добавленной медсестры, и она сможет вести журнал приема лекарств.

№	Фамилия	Имя	Отчество	Удалить	Добавить медсестру
1	Огурцова	Зина	Степановна	Delete	

Назад

Рисунок 4.23 – Страница медсестёр пациента

Для просмотра отчета о приеме медикаментов необходимо нажать кнопку «Прием лекарств» (рис 4.24). Доктор будет перенаправлен на страницу, где будет отображаться отчет о приеме лекарств данным пациентом.

№	Название	Доза	Частота употребления	Дата употребления
1	Левомеколь	1000	2 раза в день	8-6-2020 1 : 26
2	Левомеколь	1000	2 раза в день	27-5-2020 13 : 0
3	Левомеколь	1000	2 раза в день	26-5-2020 22 : 0

Назад

Рисунок 4.24 – Прием лекарств пациентом

Для добавления документа к выбранному посещению, необходимо нажать кнопку «Просмотр документов». Доктор будет перенаправлен на страницу отображения всех документов для данного посещения. На данный момент таблица пустая. Для добавления документа необходимо нажать кнопку «Добавить документ и выбрать из предложенного списка», если документ был выбран, то он не будет отображаться в списке (рис. 4.25).

№	Название документа	Просмотр	Скачать	Удалить	Добавить документ
1	Совместный осмотр врачей	Просмотр	Скачать	Delete	
2	Протокол интенсивной терапии и мониторинга	Просмотр	Скачать	Delete	

Рисунок 4.25 – Список документов для выбранного посещения

После добавления появляется возможность редактирования документов. Для каждого документа была создана своя HTML страница, которая после заполнения полей генерирует XML, который в последствии будет сохранен в базе данных для дальнейшего использования при генерации MS Word документа. Пример одного из документов можно увидеть на рисунке 4.26. При скачивании введенные данные переносятся в документ Word и скачиваются на компьютер (рис. 4.27).

Дата: " " 20 г.

Ф.И.О.:

Возраст:

Дата поступления: " " 20 г.

№ истории болезни:

Пол: муж

Масса тела:

кг.

Сутки в ОИТР:

Диагноз:

Рост:

см.

Наблюдение	Часы	9	10	11	12	13	14	15	16	17	18	19	20	21
Мониторинг	Артериальное давление, мм. Hg	Систолическое												
		Диастолическое												
	ЧСС, в мин.													
	ЦВД, см. H <sub>2</sub> O													
	S <sub>p</sub> O <sub>2</sub> , %													
	ЧД, в мин.													

Рисунок 4.26 – HTML версия документа «Протокол интенсивной терапии и мониторинга»

Дата: «08» июня 2020г.

Дата поступления: «07» июня 2020г.

ПРОТОКОЛ ИНТЕНСИВНОЙ ТЕРАПИИ И МОНИТОРИНГА

Ф.И.О.: Казимов Александр Русланович

№ истории болезни: 12

УТВЕРЖДЕНО:

Приказ Министерства здравоохранения Республики Беларусь от 17.1

Возраст: 21 лет

Аллергия: пенициллин

Пол: муж

Масса тела: 75кг.

Группа крови: 1 резус положительный

Сутки в ОИТР: 1

Диагноз: Химический ожог

Рост: 165 см.

Оценка по шкале: ACD баллов 5

Наблюдение	Часы	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	1	2	3	4	5	6	7	8	манипуляция	дата / время	установлен	заменен
Мониторинг	Артериальное давление, мм. Hg	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030	2031	2032	2033	2034	2035	2036	2037	2038	2039	2040	катетер в периферической вене: 3:14	3:15	3:16	
	Систолическое	2041	2042	2043	2044	2045	2046	2047	2048	2049	2050	2051	2052	2053	2054	2055	2056	2057	2058	2059	2060	2061	2062	2063	2064	катетер в центральной вене: 3:17	3:18	3:19	
	Диастолическое	2065	2066	2067	2068	2069	2070	2071	2072	2073	2074	2075	2076	2077	2078	2079	2080	2081	2082	2083	2084	2085	2086	2087	2088	моноканальный катетер: 3:20	3:21	3:22	
	ЧСС, в мин.	2089	2090	2091	2092	2093	2094	2095	2096	2097	2098	2099	2100	2101	2102	2103	2104	2105	2106	2107	2108	2109	2110	2111	2112	зонд через горло: 3:23	3:24	3:25	
	ЦВД, см. H <sub>2</sub> O	2113	2114	2115	2116	2117	2118	2119	2120	2121	2122	2123	2124	2125	2126	2127	2128	2129	2130	2131	2132	2133	2134	2135	2136	Манипуляция: 3:26	3:27	3:28	
	S <sub>p</sub> O <sub>2</sub> , %	2137	2138	2139	2140	2141	2142	2143	2144	2145	2146	2147	2148	2149	2150	2151	2152	2153	2154	2155	2156	2157	2158	2159	2160				
	ЧД, в мин.	2161	2162	2163	2164	2165	2166	2167	2168	2169	2170	2171	2172	2173	2174	2175	2176	2177	2178	2179	2180	2181	2182	2183	2184				
	Температура тела, °C	2185	2186	2187	2188	2189	2190	2191	2192	2193	2194	2195	2196	2197	2198	2199	2200	2201	2202	2203	2204	2205	2206	2207	2208				
	Давление, мм. Hg	2209	2210	2211	2212	2213	2214	2215	2216	2217	2218	2219	2220	2221	2222	2223	2224	2225	2226	2227	2228	2229	2230	2231	2232				
	Свч (печень/желтушность)	2233	2234	2235	2236	2237	2238	2239	2240	2241	2242	2243	2244	2245	2246	2247	2248	2249	2250	2251	2252	2253	2254	2255	2256				
Желудочный зонд (питание), мл.	2257	2258	2259	2260	2261	2262	2263	2264	2265	2266	2267	2268	2269	2270	2271	2272	2273	2274	2275	2276	2277	2278	2279	2280					
Желудочный зонд (потери), мл.	2281	2282	2283	2284	2285	2286	2287	2288	2289	2290	2291	2292	2293	2294	2295	2296	2297	2298	2299	2300	2301	2302	2303	2304					
Дренаж № 1, мл.	2305	2306	2307	2308	2309	2310	2311	2312	2313	2314	2315	2316	2317	2318	2319	2320	2321	2322	2323	2324	2325	2326	2327	2328					
Дренаж № 2, мл.	2329	2330	2331	2332	2333	2334	2335	2336	2337	2338	2339	2340	2341	2342	2343	2344	2345	2346	2347	2348	2349	2350	2351	2352					
Мониторинг	2353	2354	2355	2356	2357	2358	2359	2360	2361	2362	2363	2364	2365	2366	2367	2368	2369	2370	2371	2372	2373	2374	2375	2376					
Результат ИВЛ	2377	2378	2379	2380	2381	2382	2383	2384	2385	2386	2387	2388	2389	2390	2391	2392	2393	2394	2395	2396	2397	2398	2399	2400					
У, мл / R, mbar / ASB, mbar	2401	2402	2403	2404	2405	2406	2407	2408	2409	2410	2411	2412	2413	2414	2415	2416	2417	2418	2419	2420	2421	2422	2423	2424					
PEEP, mbar	2425	2426	2427	2428	2429	2430	2431	2432	2433	2434	2435	2436	2437	2438	2439	2440	2441	2442	2443	2444	2445	2446	2447	2448					
FiO <sub>2</sub>	2449	2450	2451	2452	2453	2454	2455	2456	2457	2458	2459	2460	2461	2462	2463	2464	2465	2466	2467	2468	2469	2470	2471	2472					
С, ml/min / R, mbar/ml/s	2473	2474	2475	2476	2477	2478	2479	2480	2481	2482	2483	2484	2485	2486	2487	2488	2489	2490	2491	2492	2493	2494	2495	2496					
Анализаторы	2497	2498	2499	2500	2501	2502	2503	2504	2505	2506	2507	2508	2509	2510	2511	2512	2513	2514	2515	2516	2517	2518	2519	2520					
Анализаторы 1	2521	2522	2523	2524	2525	2526	2527	2528	2529	2530	2531	2532	2533	2534	2535	2536	2537	2538	2539	2540	2541	2542	2543	2544					
Анализаторы 2	2545	2546	2547	2548	2549	2550	2551	2552	2553	2554	2555	2556	2557	2558	2559	2560	2561	2562	2563	2564	2565	2566	2567	2568					
Анализаторы 3	2569	2570	2571	2572	2573	2574	2575	2576	2577	2578	2579	2580	2581	2582	2583	2584	2585	2586	2587	2588	2589	2590	2591	2592					
Анализаторы 4	2593	2594	2595	2596	2597	2598	2599	2600	2601	2602	2603	2604	2605	2606	2607	2608	2609	2610	2611	2612	2613	2614	2615	2616					
Анализаторы 5	2617	2618	2619	2620	2621	2622	2623	2624	2625	2626	2627	2628	2629	2630	2631	2632	2633	2634	2635	2636	2637	2638	2639	2640					
Анализаторы 6	2641	2642	2643	2644	2645	2646	2647	2648	2649	2650	2651	2652	2653	2654	2655	2656	2657	2658	2659	2660	2661	2662	2663	2664					
Анализаторы 7	2665	2666	2667	2668	2669	2670	2671	2672	2673	2674	2675	2676	2677	2678	2679	2680	2681	2682	2683	2684	2685	2686	2687	2688					
Анализаторы 8	2689	2690	2691	2692	2693	2694	2695	2696	2697	2698	2699	2700	2701	2702	2703	2704	2705	2706	2707	2708	2709	2710	2711	2712					
Анализаторы 9	2713	2714	2715	2716	2717	2718	2719	2720	2721	2722	2723	2724	2725	2726	2727	2728	2729	2730	2731	2732	2733	2734	2735	2736					
Анализаторы 10	2737	2738	2739	2740	2741	2742	2743	2744	2745	2746	2747	2748	2749	2750	2751	2752	2753	2754	2755	2756	2757	2758	2759	2760					
Анализаторы 11	2761	2762	2763	2764	2765	2766	2767	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783	2784					
Анализаторы 12	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799	2800	2801	2802	2803	2804	2805	2806	2807	2808					
Анализаторы 13	2809	2810	2811	2812	2813	2814	2815	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831	2832					
Анализаторы 14	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847	2848	2849	2850	2851	2852	2853	2854	2855	2856					
Анализаторы 15	2857	2858	2859	2860	2861	2862	2863	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879	2880					
Анализаторы 16	2881	2882	2883	2884	2885	2886	2887	2888	2889	2890	2891	2892	2893	2894	2895	2896	2897	2898	2899	2900	2901	2902	2903	2904					
Анализаторы 17	2905	2906	2907	2908	2909	2910	2911	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927	2928					
Анализаторы 18	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943	2944	2945	2946	2947	2948	2949	2950	2951	2952					
Анализаторы 19	2953	2954	2955	2956	2957	2958	2959	2960	2961	2962	2963	2964	2965	2966	2967	2968	2969	2970	2971	2972	2973	2974	2975	2976					
Анализаторы 20	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991	2992	2993	2994	2995	2996	2997	2998	2999	3000					
Анализаторы 21	3001	3002	3003	3004	3005	3006	3007	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023	3024					
Анализаторы 22	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039	3040	3041												

№	Name	Surname	Patronymic	
1	Иван	Иванович	Иванов	Медикаменты
2	Вася	Пупкин	Петрович	Медикаменты

Рисунок 4.28 – Пациенты медсестры

Нажав на клавишу «Медикаменты», мы перейдем на страницу, где будут отображаться все медикаменты данного пользователя (рис. 4.29). Там медсестра может увидеть необходимую дозу и частоту употребления данного медикамента, а также последнее применение, если таковое имеется.

№	Название	Доза	Частота употребления	Последнее применение	
1	Левомеколь	500	2 раза в день		Подробнее
2	Пантенол	1000	1 раз в день	8-6-2020 3 : 34	Подробнее

Назад

Рисунок 4.29 – Медикаменты пациента

Для просмотра отчета о приеме выбранного медикамента необходимо нажать кнопку «Подробнее», после чего медсестру перенаправит на страницу отчета (рис. 4.30).

№	Дата	Добавить применение
1	8-6-2020 3 : 34	Удалить
2	8-6-2020 3 : 36	Удалить
3	2-6-2020 3 : 36	Удалить
4	20-6-2020 3 : 36	Удалить

Назад

Рисунок 4.30 – Применение медикамента

На данной странице можно добавить применение, либо удалить существующие. После заполнения данный отчет будет показываться доктору.

#### 4.4 Ручное тестирование android приложения

Данное приложение доступно только тем, пользователя у которых есть роль доктор, поэтому изначально пользователю необходимо авторизоваться в системе (рис 4.31).

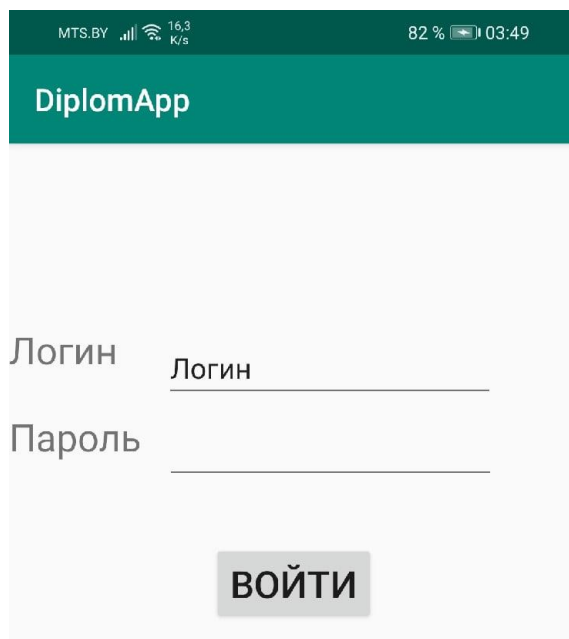


Рисунок 4.30 – Активити авторизации пользователя

Если пользователь допусти ошибку в логине или пароле, либо данный пользователь не существует в системе, то будет выведено соответствующее сообщение (рис. 4.31.а). Если пользователь не будет иметь роль доктора, то ему будет выведено соответствующее сообщение (рис. 4.31.б).

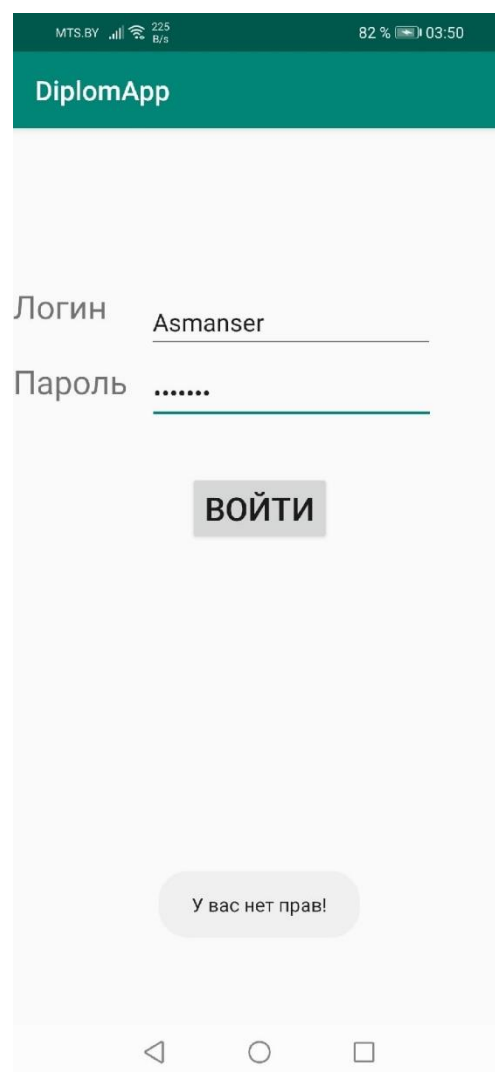
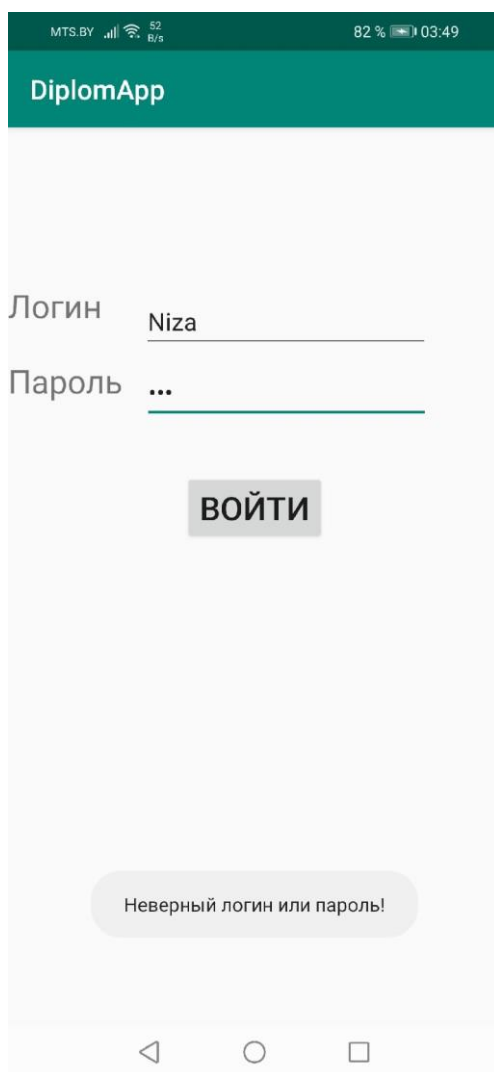


Рисунок 4.31 – а) Введен неверный пользователь б) У пользователя нет роли доктор

Если же авторизация прошла успешно, то пользователь переходит на следующую активити, где можно будет выбрать документ и скачать (рис. 4.32). Для отображения данных необходимо нажать на кнопку «Обновить», сделано это с той целью, чтобы доктор после изменений на сервере смог с легкостью обновить данные и в активити.



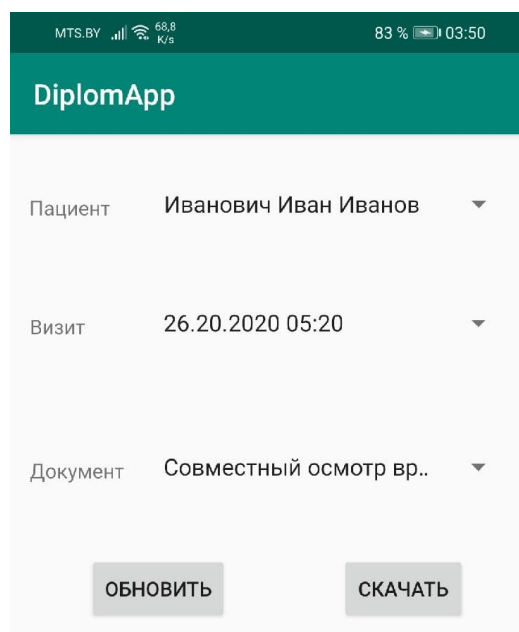


Рисунок 4.32 – Обновленные данные с сервера

Далее доктор имеет возможность выбрать необходимый документ. Выбрав пользователя автоматический подгружаются посещения данного пациента, после выбора посещения автоматически подгружаются документы для этого посещения, после чего можно нажать кнопку скачать и будет скачен файл на телефон в папку «Downloads» (рис. 4.33). После чего можно будет открыть этот файл любым приложением, который может отображать MS Word.

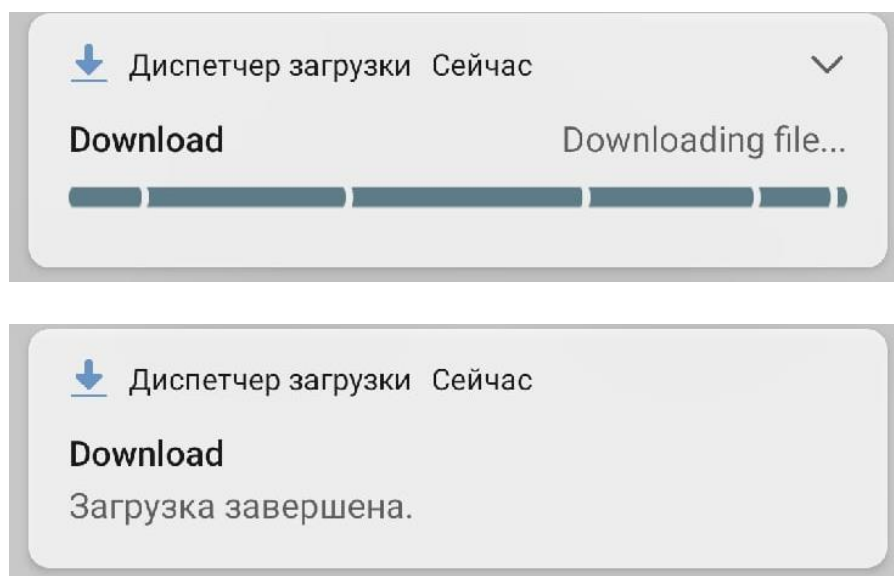


Рисунок 4.33 – Скачивание документа с сервера

#### 4.5 Модульное тестирование работы сложных алгоритмов программного комплекса

В данном разделе будут протестированы следующие алгоритмы:

- Алгоритм выборки не выбранных медикаментов для пациента;
- Алгоритм выборки не выбранных диагнозов для пациента;
- Алгоритм формирования документов;

В качестве фреймворка для тестирования был выбран JUnit – один из наиболее популярных фреймворков для тестирования на Java. Для того, чтобы симитировать работу SQL-базы данных, был использован фреймворк Mockito.

Пример имитации работы SQL-базы данных приведен на рисунке 4.:

```
doReturn(medicamentList).when(medicamentRepository).findAll();
```

Рисунок 4. – Имитация таблицы с медикаментами на существующие в базе данных медикаменты

Для создания мока на объект, в данном случае на репозитории, нужно прописать им аннотацию @MockBean. Пример приведен на рисунке 4..

```
@MockBean
private PatientMedicamentRepository patientMedicamentRepository;

@MockBean
private MedicamentRepository medicamentRepository;

@MockBean
private PatientDiagnosisRepository patientDiagnosisRepository;

@MockBean
private DiagnosisRepository diagnosisRepository;

@MockBean
private VisitReportDocumentRepository visitReportDocumentRepository;
```

Рисунок 4. – Пример создания мок объектов на репозитории

Для подмены данных через мокито, необходимо создать данные, которые будут возвращаться. Для этого был создан метод, помеченный аннотацией @Before, который создает данные для тестирования.

Далее приведены примеры тестов, которые выполнялись над тестируемыми алгоритмами (рисунки 4. – 4.):

```

@Test
void getUnusedMedicamentForPatient() {
    doReturn(patientMedicamentList).when(patientMedicamentRepository).findWithMedicamentAll(any());

    doReturn(medicamentList).when(medicamentRepository).findAll();

    List<Medicament> medicaments = patientMedicamentService.getAllMedicamentWhichUnused( id: 1L);

    assertEquals( expected: 1, medicaments.size());

    assertEquals( expected: 2L, medicaments.get(0).getId());
}

```

Рисунок 4. – Тест для верификации алгоритма выборки не выбранных медикаментов для пациента

```

@Test
void getUnusedDiagnosisForPatient() {
    doReturn(patientDiagnosisList).when(patientDiagnosisRepository).findWithDiagnosisAll(any());

    doReturn(diagnosisList).when(diagnosisRepository).findAll();

    List<Diagnosis> diagnoses = patientDiagnosisService.getAllDiagnosisWhichUnused( id: 1L);

    assertEquals( expected: 1, diagnoses.size());

    assertEquals( expected: 2L, diagnoses.get(0).getId());
}

```

Рисунок 4. – Тест для верификации алгоритма формирования документов

```

@Test
void generateDocument() throws Docx4JException {
    doReturn(Optional.of(visitReportDocument)).when(visitReportDocumentRepository).findById(any());

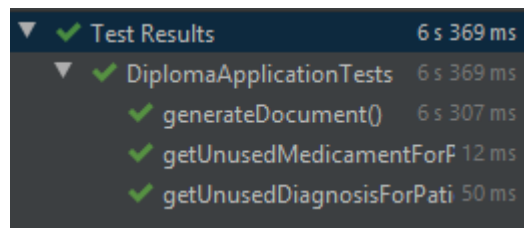
    ByteArrayOutputStream byteArrayOutputStream = documentService.generateDocument( idDocument: 1L);

    assertNotEquals( unexpected: null, byteArrayOutputStream);
    assertNotEquals( unexpected: 0, byteArrayOutputStream.size());
}

```

Рисунок 4. – Тест для верификации алгоритма выборки не выбранных диагнозов для пациента

Все вышеперечисленные модульные тесты были запущены в среде разработки IntelliJ IDEA и успешно пройдены (рисунок 4.):



The image shows a screenshot of a test results window with a dark background. It displays a hierarchical list of test results, all marked with green checkmarks, indicating successful execution. The root node is 'Test Results' with a duration of 6 s 369 ms. It contains a sub-node 'DiplomaApplicationTests' (6 s 369 ms), which in turn contains three test methods: 'generateDocument()' (6 s 307 ms), 'getUnusedMedicamentForP' (12 ms), and 'getUnusedDiagnosisForPati' (50 ms).

▼ ✓ Test Results	6 s 369 ms
▼ ✓ DiplomaApplicationTests	6 s 369 ms
✓ generateDocument()	6 s 307 ms
✓ getUnusedMedicamentForP	12 ms
✓ getUnusedDiagnosisForPati	50 ms

Рисунок 4. – Окно тестирования с успешным результатом выполнения модульных тестов

## 5 Экономическое обоснование дипломной работы

Организационно-экономический раздел дипломного проекта направлен на формирование экономической оценки эффективности разрабатываемого студентом-дипломником программного продукта, позволяющего усовершенствовать действующие или создать новые варианты информационных систем или технологий в сфере проектирования и производства. Эта работа основывается на знаниях, умениях и навыках, полученных студентами при изучении экономического блока дисциплин [8].

### 5.1 Расчет трудоемкости работ по разработке программного обеспечения

Общий объем ПО ( $V_o$ ) определяется исходя из количества и объема функций, реализуемых программой, по каталогу функций ПО в соответствии с таблицей 1.1 в приложении 1 в источнике [8] по формуле (5.1):

$$V_o = \sum_{i=1}^n V_i, \quad (5.1)$$

где  $V_i$  – объем отдельной функции ПО;  
 $n$  - общее число функций.

$$V_o = 130 + 490 + 1040 + 280 + 1970 + 3500 + 1980 + 2370 + 7860 + 4720 + 1540 + 420 + 570 = 26870$$

Анализируя разработанную программу, уточненный объем ПО ( $V_y$ ) определяем по формуле 5.2:

$$V_y = \sum_{i=1}^n V_{yi}, \quad (5.2)$$

где  $V_{yi}$  – уточненный объем отдельной функции ПО в строках исходного кода (LOC).

Уточненный объем функции получается снижением на 20% - 50% или увеличением на 10 % от значения по каталогу.

В ходе уточнения объёма функций проведены следующие операции:

а) уменьшен объём функций:

- контроль, предварительная обработка и ввод информации;
- обработка входного языка и формирование таблиц;
- организация ввода-вывода информации в интерактивном режиме;

- управление вводом-выводом;
- генерация структуры базы данных;
- обработка наборов и записей базы данных;
- организация поиска и поиск в базе данных;
- расчёт показателей;
- изменение состояния ресурсов в интерактивном режиме;
- обработка ошибочных сбойных ситуаций;
- манипулирование данными;
- b) увеличен объём функций:
  - организация ввода информации;
  - формирование базы данных;
  - организация ввода-вывода информации в интерактивном режиме.

Объемы (исходный и уточнённый) функций исходного кода разработанного программного комплекса представлены в табл. 5.1.

**Таблица 5.1 – Каталог функций программного обеспечения**

Код функций	Наименование (содержание) функций	Объем функции строк исходного кода (LOC)	
		С использованием среды разработки приложений Java	
		по каталогу (V <sub>0</sub> )	уточненный (V <sub>y</sub> )
1	2	3	4
<b>Ввод, анализ входной информации, генерация кодов и процессор входного языка</b>			
101	Организация ввода информации	130	125
102	Контроль, предварительная обработка и ввод информации	490	200
104	Обработка входного языка и формирование таблиц	1040	450
107	Организация ввода-вывода информации в интерактивном режиме	280	163
109	Управление вводом-выводом	1970	1400
<b>Формирование, ведение и обслуживание базы данных</b>			
201	Генерация структуры базы данных	3500	2350
202	Формирование базы данных	1980	1450

203	Обработка наборов и записей базы данных	2370	2000
206	Манипулирование данными	7860	4500
207	Организация поиска и поиск в базе данных	4720	3000
Управление ПО, компонентами ПО и внешними устройствами			
506	Обработка ошибочных сбойных ситуаций	1540	500
Расчетные задачи, формирование и вывод на внешние носители документов сложной формы и файлов			
703	Расчет показателей	420	350
709	Изменение состояния ресурсов в интерактивном режиме	570	435
<b>Итого:</b>		<b>26870</b>	<b>16923</b>

Таким образом для разработанного программного продукта общий объем будет определен следующим образом:

$$V_y = 125 + 200 + 450 + 163 + 1400 + 2350 + 1450 + 2000 + 4500 + 3000 + 500 + 350 + 435 = 16923$$

Разработанное программное обеспечение по своим характеристикам относится к категории 2 сложности программного обеспечения (относится к моделированию объектов и процессов, обеспечивает переносимость ПО).

На основании принятого к расчету (уточненного) объема ( $V_y$ ) и категории сложности ПО определяется нормативная трудоемкость ПО ( $T_n$ ) выполняемых работ с помощью таблицы 1.3 из источника [8].

Полученные данные представлены в таблице 5.2.

**Таблица 5.2 – Нормативная трудоемкость на разработку ПО ( $T_n$ )**

Уточнённый объем, $V_y$	2-я категория сложности ПО	Номер нормы
16923	789	72

Дополнительные затраты труда, связанные с повышением сложности разрабатываемого ПО, учитываются посредством коэффициента повышения сложности ПО ( $K_c$ ).

$K_c$  рассчитывается по формуле 5.3:

$$K_c = 1 + \sum_{i=1}^n K_i, \quad (5.3)$$

где  $K_i$  – коэффициент, соответствующий степени повышения сложности;  
 $n$  – количество учитываемых характеристик.

Таким образом:

$$K_c = 1 + 0,06 + 0,07 + 0,12 = 1,25.$$

Новизна разработанного ПО определяется путем экспертной оценки данных, полученных при сравнении характеристик разрабатываемого ПО с имеющимися аналогами. Влияние фактора новизны на трудоемкость учитывается путем умножения нормативной трудоемкости на соответствующий коэффициент, учитывающий новизну ПО ( $K_n$ ). В соответствии с таблицей 2.2 [8] разработанная программа обладает категорией новизны А, а значение  $K_n = 1,10$ .

Современные технологии разработки компьютерных программ предусматривают широкое использование коробочных продуктов (пакетов, модулей, объектов). Степень использования в разрабатываемом решении стандартных модулей определяется их удельным весом в общем объеме ПО. В данном программном комплексе используется от 20 до 40% стандартных модулей, что соответствует значению коэффициента  $K_T = 0,77$ , согласно [8].

Программный комплекс разработан *на языке Java* в операционной системе Windows и будет использоваться в глобальной сети, что согласно таблице 2.4 [8] соответствует коэффициенту, учитывающему средства разработки ПО,  $K_{yp} = 1,3$ .

Значение коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости ПО, определяются с учетом установленной категории новизны ПО согласно таблице 2.5 [8].

При этом сумма значений коэффициентов удельных весов всех стадий в общей трудоемкости равна единице. Значения коэффициентов приведены в таблице 5.3.

**Таблица 5.3 – Значения коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости ПО**

Категория новизны ПО	С применением CASE-технологии		
	Стадии разработки ПО		
	<i>ТЗ+ЭП+ТП</i>	<i>РП</i>	<i>ВН</i>
	Значения коэффициентов		
	$K_{ТЗ}+K_{ЭП}+K_{ТП}$	$K_{РП}$	$K_{ВН}$
А	<b>0,60</b>	<b>0,30</b>	<b>0,10</b>

Нормативная трудоемкость ПО ( $T_n$ ) выполняемых работ по стадиям разработки корректируется с учетом коэффициентов: повышения сложности ПО ( $K_c$ ), учитывающих новизну ПО ( $K_n$ ), учитывающих степень использования



стандартных модулей ( $K_T$ ), средства разработки ПО ( $K_{ур}$ ) и определяются по формулам:

1) для стадии ТЗ по формуле (5.4):

$$T_{умз} = T_n \times K_{мз} \times K_c \times K_n \times K_{ур}, \quad (5.4)$$

2) для стадии ЭП формуле (5.5):

$$T_{уэп} = T_n \times K_{эп} \times K_c \times K_n \times K_{ур}, \quad (5.5)$$

3) для стадии ТП формуле (5.6):

$$T_{утп} = T_n \times K_{мп} \times K_c \times K_n \times K_{ур}, \quad (5.6)$$

4) для стадии РП формуле (5.7):

$$T_{урп} = T_n \times K_{рп} \times K_c \times K_n \times K_m \times K_{ур}, \quad (5.7)$$

5) для стадии ВН формуле (5.8)

$$T_{увн} = T_n \times K_{вн} \times K_c \times K_n \times K_{ур}, \quad (5.8)$$

где  $K_{тз}$ ,  $K_{эп}$ ,  $K_{тп}$ ,  $K_{рп}$  и  $K_{вн}$  – значения коэффициентов удельных весов трудоемкости стадий разработки ПО в общей трудоемкости ПО.

Коэффициенты  $K_c$ ,  $K_n$ ,  $K_{ур}$ , вводятся на всех стадиях разработки, а коэффициент  $K_T$  вводится только на стадии РП.

Для уменьшения общей трудоёмкости разработки введем коэффициент используемости разработанного ПО, который для учебных проектов будет равен 0,1.

$$\begin{aligned} T_{утз} + T_{уэп} + T_{утп} &= T_n \times (K_{тз} + K_{эп} + K_{тп}) \times K_c \times K_n \times K_{ур} \times 0,1 \\ &= 789 \times 0,60 \times 1,25 \times 1,10 \times 1,30 \times 0,1 \approx 85 \end{aligned}$$

$$T_{урп} = 789 \times 0,30 \times 1,25 \times 1,10 \times 0,77 \times 1,30 \times 0,1 \approx 33$$

$$T_{увн} = 789 \times 0,10 \times 1,25 \times 1,10 \times 1,30 \times 0,1 \approx 14$$

Общая трудоемкость разработки ПО ( $T_o$ ) определяется суммированием нормативной (скорректированной) трудоемкости ПО по стадиям разработки согласно формуле (5.9):

$$T_o = \sum_{i=1}^n T_{yi} , \quad (5.9)$$

где  $T_{yi}$  – нормативная (скорректированная) трудоемкость разработки ПО на  $i$ -й стадии (чел.-дн.);

$n$  – количество стадий разработки.

Таким образом:

$$T_o = 85 + 33 + 14 \approx 132 \text{ чел./дней.}$$

Результаты расчетов по определению нормативной и скорректированной трудоемкости ПО по стадиям разработки и общей трудоемкости разработки ПО ( $T_o$ ) представлены в таблице 5.4 [8].

**Таблица 5.4 – Расчет общей трудоемкости разработки ПО**

Показатели	Стадии разработки					Итого
	ТЗ	ЭП	ТП	РП	ВН	
1	2	3	4	5	6	7
Общий объем ПО ( $V_o$ ), кол-во строк LOC	-	-	-	-	-	26870
Общий уточненный объем ПО ( $V_y$ ), кол-во строк LOC	-	-	-	-	-	16923
Категория сложности разрабатываемого ПО	-	-	-	-	-	2
Нормативная трудоемкость разработки ПО ( $T_n$ ), чел.-дн.	-	-	-	-	-	789
Коэффициент повышения сложности ПО ( $K_c$ )	1,25	1,25	1,25	1,25	1,25	-
Коэффициент, учитывающий новизну ПО ( $K_n$ )	1,10	1,10	1,10	1,10	1,10	-

Коэффициент, учитывающий степень использования стандартных модулей ( $K_T$ )	-	-	-	0,77	-	0,77
Коэффициент, учитывающий средства разработки ПО ( $K_{ур}$ )	1,3	1,3	1,3	1,3	1,3	-
Коэффициенты удельных весов трудоемкости стадий разработки ПО ( $K_{ТЗ}$ , $K_{ЭП}$ , $K_{ТП}$ , $K_{РП}$ , $K_{ВН}$ )	0,60			0,30	0,10	1,0
Распределение скорректированной (с учетом $K_c$ , $K_n$ , $K_T$ , $K_{ур}$ ) трудоемкости ПО по стадиям, чел.-дн.	85			33	14	-
Общая трудоемкость разработки ПО ( $T_o$ ), чел.-дн.	-	-	-	-	-	132

## 5.2 Расчет затрат на разработку программного продукта

В состав затрат на разработку программного продукта входят следующие статьи расходов:

- затраты труда на создание программного продукта (затраты по основной, дополнительной заработной плате и соответствующие отчисления) ( $Z_{тр}$ );
- затраты на изготовление эталонного экземпляра ( $Z_{эт}$ );
- затраты на технологию ( $Z_{тех}$ );
- затраты на машинное время (расходы на содержание и эксплуатацию технических средств разработки, эксплуатации и сопровождения) ( $Z_{мв}$ );
- затраты на материалы (информационные носители) ( $Z_{мат}$ );
- затраты на энергию, на использование каналов связи (для отдельных видов);
- общепроизводственные расходы (затраты на управленческий персонал, на содержание помещений) ( $Z_{общ\_пр}$ );
- непроизводственные (коммерческие) расходы (затраты, связанные с рекламой, поиском заказчиков, поставками конкретных экземпляров) ( $Z_{непр}$ ) [8].

В таблице 5.5 приведены значения основных параметров, необходимых для расчета затрат на разработку программного продукта.

**Таблица 5.5 – Параметры для расчета производственных затрат на разработку ПО**

Параметр	Единица измерения	Значение
Среднечасовая тарифная ставка ( $C_{\text{ср\_час}}$ )	руб.	4,1
Коэффициент $K_{\text{ув}}$	-	1,6
Норматив отчислений на доп. зарплату разработчиков ( $H_{\text{доп}}$ )	%	10
Численность обслуживающего персонала	чел.	1
Среднемесячная тарифная ставка обслуживающего работника ( $T_{\text{ci}}$ )	руб.	345
Норматив отчислений на доп. зарплату обслуживающего персонала ( $H_{\text{доп}}$ )	%	5
Средняя годовая ставка арендных платежей ( $C_{\text{ар}}$ )	руб./м <sup>2</sup>	65
Площадь помещения ( $S$ )	м <sup>2</sup>	9
Количество ПЭВМ ( $Q_{\text{ЭВМ}}$ )	шт.	1
Затраты на приобретение единицы ПЭВМ	руб.	923
Стоимость одного кВт-часа электроэнергии ( $C_{\text{эл}}$ )	руб.	0,21815
Коэффициент потерь рабочего времени ( $K_{\text{пот}}$ )	-	0,2
Затраты на технологию ( $З_{\text{тех}}$ )	руб.	-
Норматив общепроизводственных затрат ( $H_{\text{доп}}$ )	%	5
Норматив непроизводственных затрат ( $H_{\text{непр}}$ )	%	5

В качестве среднечасовой тарифной ставки ( $C_{\text{ср\_час}}$ ) для таблицы 5.5 была взята **средняя ставка Junior Java Developer - программиста** на момент написания данного раздела.

Суммарные затраты на разработку ПО ( $З_p$ ) определяются по формуле (5.10):

$$З_p = З_{tr} + З_{эт} + З_{тех} + З_{мв} + З_{мт} + З_{общ\_пр} + З_{непр}, \quad (5.10)$$

Расходы на оплату труда разработчиков с отчислениями равны:

$$З_{tr} = ЗП_{осн} + ЗП_{доп} + ОТЧ_{зп}, \quad (5.11)$$

где  $ЗП_{осн}$  – основная заработная плата разработчиков, руб.;

$ЗП_{доп}$  – дополнительная заработная плата разработчиков, руб.;

$ОТЧ_{зп}$  – сумма отчислений от заработной платы (социальные нужды, страхование от несчастных случаев), руб.

Основная заработная плата разработчиков рассчитывается по формуле (5.12):

$$ЗП_{осн} = C_{ср\_час} \times T_o \times K_{ув} , \quad (5.12)$$

где  $C_{ср\_час}$  – средняя часовая тарифная ставка;  
 $T_o$  – общая трудоемкость разработки, чел-час;  
 $K_{ув}$  – коэффициент, учитывающий доплаты стимулирующего характера.

Средняя часовая тарифная ставка определяется по формуле (5.13):

$$C_{ср\_час} = \frac{\sum_i C_{чи} \times n_i}{\sum_i n_i} , \quad (5.13)$$

где  $C_{чи}$  – часовая тарифная ставка разработчика  $i$  – й категории;  
 $n_i$  – количество разработчиков  $i$ -й категории.

Рассчитаем  $ЗП_{осн}$ :

$$ЗП_{осн} = 4,1 * 132 * 1,6 = 865,92 \text{ руб.}$$

Дополнительная заработная плата равна:

$$ЗП_{доп} = ЗП_{осн} \times H_{доп} / 100\% , \quad (5.14)$$

где  $H_{доп}$  – норматив отчислений на дополнительную заработную плату разработчиков.

Рассчитаем  $ЗП_{доп}$ :

$$ЗП_{доп} = 865,92 * \frac{10}{100} = 86,59 \text{ руб.}$$

Отчисления от основной и дополнительной заработной платы (отчисления на социальные нужды и обязательное страхование) рассчитываются по формуле (5.15):

$$ОТЧ_{сн} = (ЗП_{осн} + ЗП_{доп}) \times H_{зп} / 100\% , \quad (5.15)$$

где  $H_{зп}$  – процент отчислений на социальные нужды и обязательное страхование от суммы основной и дополнительной заработной платы ( $H_{зп} = 34\%$ ).

Рассчитаем **ОТЧ<sub>сн</sub>**:

$$\text{ОТЧ}_{\text{сн}} = (865,92 + 86,59) * \frac{34}{100} = 323,85 \text{ руб.}$$

Таким образом, **расходы на оплату труда разработчика**:

$$З_{\text{тр}} = 865,92 + 86,59 + 323,85 = 1276,36 \text{ руб.}$$

Затраты **машинного времени** определяются по формуле:

$$З_{\text{мв}} = C_{\text{ч}} \times K_{\text{т}} \times t_{\text{эвм}}, \quad (5.16)$$

где  $C_{\text{ч}}$  – стоимость 1 часа машинного времени (руб./ч.);

$K_{\text{т}}$  – коэффициент мультипрограммности, показывающий распределение времени работы ЭВМ в зависимости от кол-ва пользователей ЭВМ,  $K_{\text{т}}=1$ ;

$t_{\text{эвм}}$  – машинное время ЭВМ, необходимое для разработки и отладки проекта (ч.).

**Стоимость машино-часа** определяется по формуле (5.17):

$$C_{\text{ч}} = \frac{З_{\text{П}_{\text{обсл}}} + З_{\text{АР}} + З_{\text{АМ}} + З_{\text{ЭП}} + З_{\text{ВМ}} + З_{\text{ТР}} + З_{\text{ПР}}}{F_{\text{ЭВМ}}}, \quad (5.17)$$

где  $З_{\text{П}_{\text{обсл}}}$  – затраты на заработную плату обслуживающего персонала с учетом всех отчислений, (руб. в год);

$З_{\text{АР}}$  – стоимость аренды помещения, (руб. в год);

$З_{\text{АМ}}$  – амортизационные отчисления за год, (руб. в год);

$З_{\text{ЭП}}$  – затраты на электроэнергию, (руб. в год);

$З_{\text{ВМ}}$  – затраты на материалы, необходимые для обеспечения нормальной работы ПЭВМ (вспомогательные), (руб. в год);

$З_{\text{ТР}}$  – затраты на текущий и профилактический ремонт ЭВМ (руб. в год);

$З_{\text{ПР}}$  – прочие затраты, связанные с эксплуатацией ПЭВМ. (руб. в год);

$F_{\text{ЭВМ}}$  – действительный фонд времени работы ЭВМ, (час/год) [8].

Все статьи затрат формируются в расчете на единицу ПЭВМ.

– затраты на заработную плату обслуживающего персонала ( $З_{\text{П}_{\text{обсл}}}$ ) определяются по формуле (5.18):

$$З_{\text{П}_{\text{обсл}}} = \frac{З_{\text{П}_{\text{осн}}} + З_{\text{П}_{\text{доп}}} + \text{ОТЧ}_{\text{зн}}}{Q_{\text{ЭВМ}}}, \quad (5.18)$$

$$ЗП_{осн} = 12 \sum_{i=1}^n T_{ci} , \quad (5.19)$$

$$ЗП_{доп} = ЗП_{осн} \times H_{доп} / 100\% , \quad (5.20)$$

$$ОТЧ_{зп} = (ЗП_{осн} + ЗП_{доп}) \times H_{зп} / 100\% , \quad (5.21)$$

где  $ЗП_{осн}$  – основная заработная плата обслуживающего персонала, руб.;  
 $ЗП_{доп}$  – дополнительная заработная плата обслуживающего персонала, руб.;

$ОТЧ_{зп}$  – сумма отчислений от заработной платы (социальные нужды, страхование от несчастных случаев), руб.;

$Q_{ЭВМ}$  – количество обслуживаемых ПЭВМ, шт.;

$T_{ci}$  – месячная тарифная ставка i-го работника, руб.;

$n$  – численность обслуживающего персонала, чел.;

$H_{доп}$  – процент дополнительной заработной платы обслуживающего персонала от основной;

$H_{зп}$  – процент отчислений на социальные нужды и обязательное страхование от суммы основной и дополнительной заработной платы.

Таким образом, *затраты на оплату труда обслуживающего персонала:*

$$ЗП_{осн} = 12 * 345 = 4140 \text{ руб.}$$

$$ЗП_{доп} = 4140 * \frac{5}{100} = 207 \text{ руб.}$$

$$ОТЧ_{осн} = (4140 + 207) * \frac{34}{100} = 1477,98 \text{ руб.};$$

$$ЗП_{обсл} = 4140 + 207 + 1477,98 = 5824,98 \text{ руб.};$$

– годовые затраты на аренду помещения ( $З_{AP}$ ) определяются по формуле (5.22) [8]:

$$З_{AP} = \frac{C_{AP} \times S}{Q_{ЭВМ}} , \quad (5.22)$$

где  $C_{AP}$  – средняя годовая ставка арендных платежей, руб./м<sup>2</sup>;  
 $S$  – площадь помещения, м<sup>2</sup>.

Таким образом, *затраты на аренду*:

$$З_{AP} = 65 * 9 = 585 \text{ руб.}$$

– сумма годовых амортизационных отчислений ( $З_{AM}$ ) определяется по формуле (5.23):

$$З_{AM} = З_{приобр} \times (1 + K_{доп}) \times H_{AM}, \quad (5.23)$$

где  $З_{приобр}$  – затраты на приобретение (стоимость) единицы ПЭВМ, руб;  
 $K_{доп}$  – коэффициент, характеризующий дополнительные затраты, связанные с доставкой, монтажом и наладкой оборудования,  $K_{доп} = 12-13\%$  от  $З_{приобр}$ ;

$З_{приобр} \times (1 + K_{доп})$  – балансовая стоимость ЭВМ, руб;

$H_{AM}$  – норма амортизации, %.

Таким образом, *затраты на амортизацию*:

$$З_{AM} = 923,00 * (1 + 0,13) * 0,125 = 130,37 \text{ руб.}$$

– стоимость электроэнергии, потребляемой за год, ( $З_{ЭП}$ ) определяется по формуле (5.24):

$$З_{ЭП} = M \times F_{ЭВМ} \times C_{эл} \times A, \quad (5.24)$$

где  $M$  – паспортная мощность ПЭВМ, (кВт),  $M = 0,41$  кВт;

$C_{эл}$  – стоимость одного кВт-часа электроэнергии, руб;

$A$  – коэффициент интенсивного использования мощности,  $A = 0,98 \dots 0,9$ .

– действительный годовой фонд времени работы ПЭВМ ( $F_{ЭВМ}$ ) рассчитывается:

$$F_{ЭВМ} = (D_{г} - D_{вых} - D_{пр}) \times F_{см} \times K_{см} \times (1 - K_{пот}), \quad (5.25)$$

где  $D_{г}$  – общее количество дней в году,  $D_{г} = 365$  дней;

$D_{вых}$ ,  $D_{пр}$  – число выходных и праздничных дней в году,  $D_{вых} + D_{пр} = 112$  дн.;

$F_{см}$  – продолжительность 1 смены,  $F_{см} = 8$  часов;

$K_{см}$  – коэффициент сменности, т.е. количество рабочих смен ЭВМ,  $K_{см} = 1$ ;

$K_{пот}$  – коэффициент, учитывающий потери рабочего времени, связанные с профилактикой и ремонтом ЭВМ,  $K_{пот} = 0,15-0,30$ .



Таким образом, *годовой фонд времени работы ПЭВМ:*

$$F_{ЭВМ} = (365 - 112) * 8 * 1 * (1 - 0,2) = 1619,2 \text{ дн.};$$

*Стоимость электроэнергии:*

$$З_{ЭП} = 0,41 * 1619,2 * 0,21815 * 0,9 = 130,34 \text{ руб.}$$

– затраты на материалы ( $З_{ВМ}$ ), необходимые для обеспечения нормальной работы ПЭВМ составляют около 1% от балансовой стоимости ЭВМ и определяются [8]:

$$З_{ВМ} = З_{приобр} \times (1 + K_{доп}) \times K_{мз}, \quad (5.26)$$

где  $З_{приобр}$  – затраты на приобретение (стоимость) ЭВМ (руб);

$K_{доп}$  – коэффициент, характеризующий дополнительные затраты, связанные с доставкой, монтажом и наладкой оборудования,  $K_{доп} = 12-13 \%$  от  $З_{приобр}$ ;

$K_{мз}$  – коэффициент, характеризующий затраты на вспомогательные материалы ( $K_{мз} = 0,01$ ).

Таким образом, *затраты на материалы:*

$$З_{ВМ} = 923,00 * (1 + 0,13) * 0,01 = 10,43 \text{ руб.}$$

– затраты на текущий и профилактический ремонт ( $З_{ТР}$ ) принимаются равными 5% от балансовой стоимости ЭВМ:

$$З_{ТР} = З_{приобр} \times (1 + K_{доп}) \times K_{тр}, \quad (5.27)$$

где  $K_{тр}$  – коэффициент, характеризующий затраты на текущий и профилактический ремонт ( $K_{мз} = 0,08$ ).

Таким образом, *затраты на текущий и профилактический ремонт:*

$$З_{ТР} = 923,00 * (1 + 0,13) * 0,08 = 83,44 \text{ руб.}$$

– прочие затраты, связанные с эксплуатацией ЭВМ ( $З_{ПР}$ ) состоят из амортизационных отчислений на здания, стоимости услуг сторонних организаций и составляют 5 % от балансовой стоимости:

$$З_{ПР} = З_{приобр} \times (1 + K_{доп}) \times K_{пр}, \quad (5.28)$$

где  $K_{пр}$  – коэффициент, характеризующий размет прочих затрат, связанных с эксплуатацией ЭВМ ( $K_{пр} = 0,05$ ).

Таким образом, *прочие затраты*:

$$З_{пр} = 923,00 * (1 + 0,13) * 0,05 = 52,15 \text{ руб};$$

Для расчета машинного времени ЭВМ ( $t_{эвм}$  в часах), необходимого для разработки и отладки проекта, следует использовать формулу (5.29):

$$t_{эвм} = (t_{РП} + t_{ВН}) \times F_{см} \times K_{см}, \quad (5.29)$$

где  $t_{РП}$  – срок реализации стадии «Рабочий проект» (РП), 22 дней;  
 $t_{ВН}$  – срок реализации стадии «Ввод в действие» (ВП), 7 дней;  
 $F_{см}$  – продолжительность рабочей смены, (ч.),  $F_{см} = 8$  ч.;  
 $K_{см}$  – количество рабочих смен,  $K_{см} = 1$ .

$$t_{эвм} = (22 + 7) * 8 * 1 = 232 \text{ ч};$$

$$\begin{aligned} C_{ч} &= \frac{5824,98 + 585 + 130,37 + 130,34 + 10,43 + 83,44 + 52,15}{1619,2} \\ &= 4,21 \text{ руб/ч}; \end{aligned}$$

$$З_{мв} = 4,21 * 1 * 232 = 976,72 \text{ руб}.$$

Расчет затрат на изготовление эталонного экземпляра ( $З_{эт}$ ) осуществляется по формуле (5.30):

$$З_{эт} = (З_{пр} + З_{тех} + З_{мв}) \times K_{эт}, \quad (5.30)$$

где  $K_{эт}$  – коэффициент, учитывающий размер затрат на изготовление эталонного экземпляра, ( $K_{эт} = 0,05$ ).

При написании дипломной работы были использованы:

- IDE IntelliJ IDEA;
- MySQL Server.

Поэтому затраты на технологию ( $З_{тех}$ ) будут нулевыми.

$$З_{эт} = (83,44 + 0 + 976,72) * 0,05 = 53,08 \text{ руб}$$

Затраты на материалы (носители информации и пр.), необходимые для обеспечения нормальной работы ПЭВМ рассчитываются следующим образом:

$$З_{\text{мат}} = З_{\text{приобр}} \times (1 + K_{\text{доп}}) \times K_{\text{мз}} , \quad (5.31)$$

где  $З_{\text{приобр}}$  – затраты на приобретение ЭВМ, руб;

$K_{\text{доп}}$  – коэффициент, характеризующий дополнительные затраты, связанные с доставкой, монтажом и наладкой оборудования,  $K_{\text{доп}} = 12-13\%$  от  $З_{\text{приобр}}$ ;

$K_{\text{мз}}$  – коэффициент, характеризующий затраты материалы ( $K_{\text{мз}} = 0,01$ ).

$$З_{\text{мат}} = 923,00 \times (1 + 0,13) \times 0,01 = 10,43 \text{ руб.}$$

Общепроизводственные затраты рассчитываются по формуле (5.32):

$$З_{\text{общ\_пр}} = ЗП_{\text{осн}} \times H_{\text{доп}} / 100\% , \quad (5.32)$$

где  $H_{\text{доп}}$  – норматив общепроизводственных затрат.

Таким образом, **общепроизводственные затраты:**

$$З_{\text{общ\_пр}} = 865,92 \times 0,05 = 43,3 \text{ руб.}$$

Непроизводственные затраты рассчитываются по формуле (5.33):

$$З_{\text{непр}} = ЗП_{\text{осн}} \times H_{\text{доп}} / 100\% , \quad (5.33)$$

где  $H_{\text{непр}}$  – норматив непроизводственных затрат.

Таким образом, **непроизводственные затраты:**

$$З_{\text{непр}} = 865,92 \times 0,05 = 43,3 \text{ руб.}$$

Итого по формуле (5.10) получаем **суммарные затраты на разработку** [8]:

$$З_p = 1276,36 + 53,08 + 0 + 976,72 + 10,43 + 43,3 + 43,3 = 2403,19 \text{ руб}$$

Результаты расчетов приведены в таблице 5.6.

**Таблица 5.6 – Расчет суммарных затрат на разработку ПО, руб.**

Статья затрат	Итого
Затраты на оплату труда разработчиков ( $Z_{тр}$ )	1276,36
Основная заработная плата разработчиков, руб.	865,92
Дополнительная заработная плата разработчиков, руб.	86,59
Отчисления от основной и дополнительной заработной платы, руб.	323,85
Затраты машинного времени ( $Z_{мв}$ )	976,72
Стоимость машино-часа, руб/ч	4,21
Затраты на заработную плату обслуживающего персонала	5824,98
Затраты на аренду, руб.	585
Сумма годовых амортизационных отчислений, руб.	130,37
Стоимость электроэнергии, руб.	130,34
Действительный годовой фонд времени работы ПЭВМ, дн.	1619,2
Затраты на текущий и профилактический ремонт, руб.	83,44
Прочие затраты, связанные с эксплуатацией ЭВМ, руб.	52,15
Машинное время ЭВМ, ч.	232
Затраты на изготовление эталонного экземпляра ( $Z_{эт}$ ), руб.	53,08
Затраты на технологию ( $Z_{тех}$ ), руб.	0
Затраты на материалы, руб.	10,43
Общепроизводственные затраты, руб.	43,3
Непроизводственные затраты, руб.	43,3
Суммарные затраты на разработку ПО, руб.	2403,19

### 5.3 Формирование цены создания программного обеспечения

Оптовая цена ПП ( $C_{опт}$ ) определяется следующим образом:

$$C_{опт} = Z_p + P_p, \quad (5.35)$$

$$P_p = \frac{Z_p \times U_p}{100}, \quad (5.36)$$

где  $Z_p$  – себестоимость ПО, руб.;  
 $P_p$  – прибыль от реализации ПП, руб.;  
 $U_p$  – уровень рентабельности ПП, % ( $U_p = 42\%$ ).

Рассчитаем *прибыль*:

$$P_p = \frac{2403,19 \times 42}{100} = 1009,34 \text{ руб.}$$

Рассчитаем *оптовую цену*:

$$\text{Ц}_{\text{опт}} = 2403,19 + 1009,34 = 3412,53 \text{ руб.}$$

Прогнозируемая отпускная цена ПП без НДС рассчитывается:

$$\text{Ц}_{\text{отп}} = \text{З}_p + \text{П}_p + \text{Р}_{\text{ндс}}, \quad (5.37)$$

Налог на добавленную стоимость ( $\text{Р}_{\text{ндс}}$ ) рассчитывается по формуле

$$\text{Р}_{\text{ндс}} = (\text{З}_p + \text{П}_p) \times \frac{\text{Н}_{\text{ндс}}}{100}, \quad (5.38)$$

где  $\text{Н}_{\text{ндс}}$  – ставка налога на добавленную стоимость, %,  $\text{Н}_{\text{ндс}} = 20 \%$ .

Рассчитаем *НДС*:

$$\text{Р}_{\text{ндс}} = 3412,53 \times \frac{20}{100} = 682,51 \text{ руб.}$$

Рассчитаем *отпускную цену*:

$$\text{Ц}_{\text{отп}} = 3412,53 + 682,51 = 4095,04 \text{ руб.}$$

Розничную цену на ПП ( $\text{Ц}_{\text{розн}}$ ) можно определить следующим образом:

$$\text{Ц}_{\text{розн}} = \text{Ц}_{\text{отп}} + \text{Т}_n \quad (5.39)$$

где  $\text{Т}_n$  – торговая наценка при реализации программного обеспечения (через специализированные магазины (торговых посредников), ее значение принимается равной 10%).

Рассчитаем *розничную цену*:

$$\text{Ц}_{\text{розн}} = 4095,04 \times 1,1 = 4504,54 \text{ руб.}$$

**Таблица 5.7 - Плановая калькуляция разработки программного продукта, руб.**

Наименование статьи расходов	Условные обозначения	Значение
Затраты на оплату труда разработчиков	$\text{З}_{\text{тр}}$	1276,36
Основная заработная плата разработчиков		865,92

Дополнительная заработная плата разработчиков		86,59
Отчисления от основной и дополнительной заработной платы		323,85
Затраты машинного времени	$З_{\text{мв}}$	976,72
Затраты на изготовление эталонного экземпляра	$З_{\text{эт}}$	53,08
Затраты на технологию	$З_{\text{тех}}$	0
Затраты на материалы	$З_{\text{мат}}$	10,43
Общепроизводственные затраты	$З_{\text{общ.пр}}$	43,3
<b>Производственная себестоимость</b>		2359,89
Непроизводственные (коммерческие) затраты	$З_{\text{непр}}$	43,3
<b>Полная себестоимость (суммарные затраты на разработку ПО)</b>	$З_{\text{р}}$	2403,19
Прибыль от реализации ПО	$П_{\text{р}}$	1009,34
Оптовая цена ПП	$Ц_{\text{опт}}$	3412,53
Налог на добавленную стоимость	$Р_{\text{ндс}}$	682,51
Прогнозируемая отпускная цена ПП с НДС	$Ц_{\text{отп}}$	4095,04
<b>Розничная цена на ПП</b>	$Ц_{\text{розн}}$	4504,54

#### 5.4 Расчет эффекта от внедрения программного обеспечения

Методика расчета эффекта от внедрения ПО зависит от:

- коммерческого или некоммерческого характера разработки;
- области применения разработки, прежде всего в производственной или непроизводственной сфере;
  - стадии внедрения и степени ее реализации (опытная отладка или эксплуатация, полный цикл внедрения и т.д.);
  - наличия прямых эффектов (для коммерческих проектов) или (и) косвенных (для некоммерческих проектов);
  - наличие эффекта в виде роста количества или (и) качества выпускаемой продукции, выполняемых работ или услуг; снижение затрат на производство или маркетинг; снижение затрат на компьютерно-информационные работы или услуги и т.д.

Эффект (прибыль) может просчитываться по формуле (5.37) [8]:

$$\mathcal{E} = Z_{\text{баз}} - Z, \quad (5.37)$$

где  $Z_{\text{баз}}$  – текущие и инвестиционные затраты по базовому варианту, включающие затраты на приобретение продукта (цену), его эксплуатацию;

$Z$  – текущие и инвестиционные затраты по варианту, предложенному студентом-дипломником.

По результатам изучения рыночных цен программных продуктов схожего функционального назначения было установлено, что средняя стоимость аналога составляет 10545 руб.

Таким образом, *эффект*:

$$\mathcal{E} = 10545 - 4504,54 = 6040,46 \text{ руб.}$$

На основе рассчитанного эффекта от разработки ПО следует рассчитать следующие итоговые показатели, характеризующие экономическую эффективность проекта [8]:

– рентабельность затрат (З) или инвестиций (И) на новую информационную технологию, программных продукт:

$$P = \frac{\mathcal{E}(И)}{З(И)} \cdot 100\% , \quad (5.38)$$

Таким образом, *рентабельность*:

$$P = \frac{6040,46}{4504,54} \times 100\% = 1,34097 \times 100\% = 134,1\%.$$

– срок окупаемости затрат (инвестиций):

$$T = \frac{З(И)}{\mathcal{E}(И)}, \quad (5.39)$$

Таким образом, *срок окупаемости затрат*:

$$T = \frac{4504,54}{6040,46} = 0,75 \text{ года.}$$

Т.к. срок окупаемости составляет меньше одного календарного года, то проведение динамической оценки (расчёт динамических показателей эффективности) не целесообразно [8].

Годовой экономический эффект определяется:

$$\Gamma \mathcal{E} \mathcal{E} = \mathcal{E}(И) - P_{\text{баз}} \times З(И) , \quad (5.40)$$

где  $P_{\text{баз}}$  – рентабельность затрат (инвестиций) базового вар., руб 25%,

Таким образом, *годовой экономический эффект*:

$$\text{ГЭЭ} = 6040,46 - 0,25 \cdot 4504,54 = 4914,33 \text{ руб.}$$

Все данные приведены в итоговой таблице 5.8.

**Таблица 5.8 – Техничко-экономические показатели проекта**

Наименование показателя	Единица измерения	Проектный вариант
Показатели затрат на разработку		
Общая трудоемкость разработки ПО	чел.- дн	132
Затраты на разработку программы	руб.	2403,19
Затраты на оплату труда разработчиков	руб.	1276,36
Затраты машинного времени	руб.	976,72
Затраты на изготовление эталонного экземпляра	руб.	53,08
Затраты на технологию	руб.	0
Затраты на материалы	руб.	10,43
Общепроизводственные затраты	руб.	43,3
Непроизводственные (коммерческие) затраты	руб.	43,3
Оптовая цена ПП	руб.	3412,53
Прогнозируемая отпускная цена ПП с НДС	руб.	4095,04
Розничная цена на ПП	руб.	4504,54
Рентабельность затрат	%	134,1
Простой срок окупаемости проекта	лет	0,75
Годовой экономический эффект	руб.	4914,33

Таким образом, по результатам проведенной оценки установлено, что реализация проекта обоснована и является экономически целесообразной. Об этом свидетельствуют следующие показатели: срок окупаемости меньше года при размере годового экономического эффекта ГЭЭ = 4914,33 руб. с уравнением рентабельности  $P = 134,1\%$ .



## **6 ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ**

### **6.1 Понятие и содержание гигиены труда и производственной санитарии**

**Гигиена труда** — наука, изучающая воздействие окружающей производственной среды, характера трудовой деятельности на организм работающего.

В разделе гигиены труда изучаются организация труда на производстве, изменения функций и работоспособности у работающих в процессе работы, режим труда и отдыха.

Особое внимание уделяется санитарным условиям труда, состоянию здоровья людей на производстве.

**Производственная санитария** — система организационных гигиенических и санитарно-технических мероприятий и технических средств, предотвращающих воздействие на работающих вредных производственных факторов (ГОСТ 12.0.002—2003 ССБТ «Термины и определения»).

**К нормативным правовым актам по гигиене труда относятся** санитарные нормы, правила и гигиенические нормативы. Перечень действующих санитарных норм, правил и гигиенических нормативов приведен в Государственном реестре правил, норм, стандартов и других нормативных актов по охране труда РБ.

**Производственная санитария** — совокупность факторов производственной среды, оказывающих воздействие на здоровье и работоспособность человека в процессе труда (ГОСТ 19605—74 «Организация труда. Основные понятия. Термины и определения»).

Основной характеристикой условий труда является аттестация рабочих мест по условиям труда. Постановлением Кабинета Министров РБ от 02.08.1995 г. №409 (с изм. и доп.) определен Порядок проведения аттестации рабочих мест по условиям труда.

**Особые условия труда** определены Списками №1 и 2 производств, работ, профессий, дающих право на пенсию за работу с особыми условиями труда; разработки мероприятий по улучшению условий труда и оздоровлению работников.

### **6.2 Классификация вредных и опасных производственных факторов**

**Опасные и вредные производственные факторы** подразделяются по природе действия на следующие группы: физические; химические; биологические; психофизиологические.

**Физические опасные и вредные производственные факторы:**

- движущиеся машины и механизмы;
- подвижные части производственного оборудования;

- передвигающиеся изделия, заготовки, материалы;
- разрушающиеся конструкции;
- обрушивающиеся горные породы;
- повышенная запыленность и загазованность воздуха рабочей зоны;
- повышенная или пониженная температура поверхностей оборудования, материалов, воздуха рабочей зоны;
- повышенный уровень шума на рабочем месте, вибрации, инфразвуковых колебаний, ультразвука;
- повышенная или пониженная влажность воздуха, подвижность воздуха, ионизация воздуха;
- повышенный уровень ионизирующих излучений в рабочей зоне;
- повышенное значение напряжения в электрической цепи, замыкание которой может произойти через тело человека;
- повышенный уровень статического электричества, электромагнитных излучений;
- повышенная напряженность электрического, магнитного полей;
- отсутствие или недостаток естественного света;
- недостаточная освещенность рабочей зоны;
- повышенная яркость света;
- пониженная контрастность;
- прямая и отраженная блескость;
- повышенная пульсация светового потока;
- повышенный уровень ультрафиолетовой, инфракрасной радиации;
- острые кромки, заусенцы и шероховатость на поверхностях заготовок, инструментов и оборудования;
- расположение рабочего места на значительной высоте относительно поверхности земли (пола);
- невесомость.

**Химические опасные и вредные производственные факторы подразделяются:**

- токсические;
- раздражающие;
- канцерогенные;
- мутагенные;
- влияющие на репродуктивную функцию;
- по пути проникновения в организм человека:
- органы дыхания;
- желудочно-кишечный тракт;
- кожные покровы и слизистые оболочки.

**Биологические опасные и вредные производственные факторы:** включают патогенные микроорганизмы (бактерии, вирусы, риккетсии, спирохеты, грибы, простейшие) и продукты их жизнедеятельности.

**Психофизиологические опасные и вредные производственные факторы по характеру действия подразделяются:**

- статические;
- динамические.
- на нервно-психические перегрузки, в том числе:
- умственное перенапряжение;
- перенапряжение анализаторов;
- монотонность труда;
- эмоциональные перегрузки.

Один и тот же опасный и вредный производственный фактор по природе своего действия может относиться одновременно к различным группам.

### **6.3 Методы профилактики профессиональных заболеваний**

Если работник подвергается воздействию вредного фактора, обусловленного трудовым процессом, это еще не значит, что он обязательно заболеет. Мы говорим только о вероятности получения заболевания.

Во многом исход такой работы зависит от индивидуальных особенностей организма в виде предрасположенности к разного рода заболеваниям, но определяющую роль играют методы профилактики.

**К ним относятся:**

- Устранение негативного фактора из рабочего процесса. Замена производственного оборудования или материалов;
- Уменьшение времени воздействия фактора на работника. Технологические перерывы;
- Обеспеченность средствами индивидуальной и коллективной защиты;
- Обучение работников правилам пользования СИЗ;
- Ознакомление работников с вредными факторами на рабочих местах;
- Обучение работников правилам оказания первой помощи;
- Физкультура и спорт;
- Диспансеризация работников.

### **6.4 Медицинские предварительные и периодические осмотры работников**

Предварительные медицинские осмотры (обследования) при поступлении на работу проводятся с целью определения соответствия состояния здоровья работника (освидетельствуемого) поручаемой ему работе.

**Периодические медицинские осмотры (обследования) проводятся с целью:**

- динамического наблюдения за состоянием здоровья работников, своевременного выявления начальных форм профессиональных заболеваний, ранних признаков воздействия вредных и (или) опасных производственных факторов на состояние здоровья работников, формирования групп риска;

- выявления общих заболеваний, являющихся медицинскими противопоказаниями для продолжения работы, связанной с воздействием вредных и (или) опасных производственных факторов;
- своевременного проведения профилактических и реабилитационных мероприятий, направленных на сохранение здоровья и восстановление трудоспособности работников.

Частота проведения периодических медицинских осмотров (обследований) определяется территориальными органами

Федеральной службы по надзору в сфере защиты прав потребителей и благополучия человека совместно с работодателем исходя из конкретной санитарно-гигиенической и эпидемиологической ситуации, и приложения 1.2 приказа Минздрава РФ №90, но периодические медицинские осмотры (обследования) должны проводиться не реже чем один **раз в два года**.

Лица, не достигшие **возраста 21 года**, проходят периодические медицинские осмотры ежегодно.

Периодические медицинские осмотры (обследования) работников могут проводиться досрочно в соответствии с медицинским заключением или по заключению территориальных **органов Федеральной службы по надзору в сфере защиты прав потребителей** и благополучия человека с обязательным обоснованием в направлении причины досрочного (внеочередного) осмотра (обследования).

**Предварительные и периодические медицинские осмотры** (обследования) работников проводятся медицинскими организациями, имеющими лицензию на указанный вид деятельности.

Работникам, занятым **на вредных работах** и на работах с вредными и (или) опасными производственными факторами в течение пяти и более лет, периодические медицинские осмотры (обследования) проводятся в центрах профпатологии и других медицинских организациях, имеющих лицензии на экспертизу профпригодности и экспертизу связи заболевания с профессией, **один раз в пять лет**.

Работодатель определяет контингенты и составляет поименный список лиц, подлежащих периодическим медицинским осмотрам (обследованиям), с указанием участков, цехов, производств, вредных работ и вредных и (или) опасных производственных факторов, оказывающих воздействие на работников, и после согласования с территориальными органами Федеральной службы по надзору в сфере защиты прав потребителей и благополучия человека направляет его **за 2 месяца** до начала осмотра в медицинскую организацию, с которой заключен договор на проведение периодических медицинских осмотров (обследований).

Медицинская организация на основании полученного от работодателя поименного списка работников, подлежащих периодическим медицинским осмотрам (обследованиям), утверждает совместно с работодателем календарный план проведения медицинских осмотров (обследований).

Руководитель медицинской организации, осуществляющей предварительные и периодические медицинские осмотры (обследования), утверждает состав медицинской комиссии, председателем которой должен быть врач-профпатолог или врач иной специальности, имеющий профессиональную подготовку по профпатологии, членами комиссии - специалисты, прошедшие в рамках своей специальности подготовку по профессиональной патологии.

Комиссия определяет виды и объемы необходимых исследований с учетом специфики действующих производственных факторов и медицинских противопоказаний к осуществлению или продолжению работы на основании действующих нормативных правовых актов.

Работник для прохождения предварительного медицинского осмотра (обследования) представляет направление, выданное работодателем, в котором указываются вредные и (или) опасные производственные факторы и вредные работы, а также паспорт или другой документ, его заменяющий, амбулаторную карту или выписку из нее с результатами периодических осмотров по месту предыдущих работ и в случаях, предусмотренных законодательством - решение врачебной психиатрической комиссии.

## **6.5 Санитарно-бытовое обеспечение работников. Оборудование санитарно-бытовых помещений, их размещение**

Перевозка в лечебные учреждения или к месту жительства работников, пострадавших от несчастных случаев на производстве и профессиональных заболеваний, а также по иным медицинским показаниям производится транспортными средствами организации либо за ее счет. Обеспечение работающих санитарно-бытовыми помещениями, размещение и оборудование этих помещений производится согласно требованиям **СНиП 2.09.04-87 «Административные и бытовые здания»**.

**К вспомогательным помещениям относятся:**

- гардеробные;
- умывальные;
- душевые и полудушевые;
- ручные и ножные ванны;
- уборные, помещения личной гигиены для женщин;
- помещения для отдыха и приема пищи;
- помещения для обогрева работающих и др.

Необходимость вспомогательных помещений устанавливается в соответствии с санитарной характеристикой производственных процессов (группой и подгруппой) и числом работающих.

По санитарной характеристике выбирается состав специальных бытовых помещений и их оборудование, а по числу работающих в наиболее многочисленной смене, которым положены эти помещения, определяют площадь этих помещений и их оборудование.

**Помещения для отдыха** (в рабочее время) предусматривают в соответствии с технологической частью проекта. Их располагают на удалении не более 75 м от рабочих мест и оборудуют умывальником с подводом горячей и холодной воды и устройством питьевого водоснабжения.

**Помещения для личной гигиены женщин** оборудуют при работе 15 женщин и более в самой многочисленной смене и, как правило, совмещают с женскими уборными.

**Прачечные**, помещения для химической чистки, сушки, обеспыливания, обезжиривания, ремонта одежды и обуви устраивают в коммунальных и общих прачечных предприятия.

**Комнаты для приема пищи** оборудуются кипятильниками, холодильниками, умывальниками и электроплитами. Душевые следует размещать смежно с гардеробными.

**Нормы площади помещений на 1 человека**, единицу оборудования, расчетное число работников, обслуживаемых на единицу оборудования в санитарно-бытовых помещениях:

**Расстояние от рабочих мест** в производственных зданиях до уборных, курительных, помещений для обогрева или охлаждения, полудушей, устройств питьевого водоснабжения должно приниматься не более **75 м**, а от рабочих мест на площадке предприятия – **не более 150 м**.

Устройства для очистки обуви оборудуются при входах в гардеробные [11].

## **7 РЕСУРСО- И ЭНЕРГОСБЕРЕЖЕНИЕ ПРИ ВНЕДРЕНИИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

### **7.1 Основные понятия в области ресурсо- и энергосбережения**

Ресурсосбережение – организационная, экономическая, техническая, научная, практическая и информационная деятельность, в том числе методы, процессы, комплекс организационно-технических мер и мероприятий, сопровождающих все стадии жизненного цикла объектов и направленных на рациональное использование и экономное расходование ресурсов.

Ресурсы – ценности, запасы, возможности, источники дохода в государственном бюджете. В общем виде ресурсы делятся на природные и экономические (материальные, трудовые, финансовые).

Энергосбережение – реализация правовых, организационных, научных, производственных, технических и экономических мер, направленных на рациональное использование (и экономное расходование) энергетических ресурсов и на вовлечение в хозяйственный оборот возобновляемых источников энергии.

Энергосбережение – комплекс мер для обеспечения эффективного использования энергоресурсов.

Экономия энергии – результаты мер по снижению непроизводительных потерь. Меры могут быть пассивные (например, теплоизоляция), активные (утилизация теплоты) или организационные (замена технологий).

Рациональное использование энергии – расходование энергии наиболее целесообразным путем.

### **7.2 Регулирование вопросов, связанных с внедрением программного обеспечения и последующего ресурсосбережения**

Республиканским органом государственного управления, уполномоченным Правительством Республики Беларусь для проведения государственной политики в сфере энергосбережения, является Департамент по энергоэффективности Государственного комитета по стандартизации Республики Беларусь. Основными задачами Департамента являются проведение государственной политики в сфере энергосбережения и осуществление государственного надзора за рациональным использованием топлива, электрической и тепловой энергии.

Также, Департамент осуществляет техническое регулирование, выполнение программ, проведение контроля и прочие оперативные функции в области эффективного использования топливно-энергетических ресурсов и энергосбережения.

Ресурсосбережение и энергосбережение способствует росту эффективности экономики, повышению ее конкурентоспособности.

Экономия ресурсов, связанная с внедрением разработанного программного обеспечения, заключается в сокращении трудоемкости, а также канцелярских принадлежностей.

Применительно к данной работе можно сказать, что программное обеспечение призвано заменить ручной труд автоматизированным или автоматическим, а также максимально упростить работу. Таким образом, становится возможным значительно уменьшить количество времени, требуемое для реализации поставленной задачи. Но также необходимо рассчитать экономию остальных расходов.

Расчет экономии остальных расходов (канцелярские и пр.) происходит по формуле (7.1).

$$\mathcal{E} = K * C_{\text{зп}} * n * 12 * O_p \quad (7.1)$$

где:  $K$  – коэффициент сокращения остальных расходов;

$C_{\text{зп}}$  – среднемесячная заработная плата служащих, руб.;

$n$  – количество рабочих;

$O_p$  – остальные расходы, %.

Среднемесячная заработная плата служащего составляет 320,0 рублей. Остальные расходы принимаем равными 3 %. Коэффициент сокращения равен 0,5. Количество рабочих  $n = 1$ .

$$\mathcal{E} = K * C_{\text{зп}} * n * 12 * O_p$$

$$\mathcal{E} = 0,5 * 320,0 * 1 * 12 * 0,03 = 57,6 \text{ руб.}$$

По результатам проведенной оценки установлено, что внедрение нового программного продукта является целесообразным, так как можно сэкономить на других расходах.

Так же при разработке приложений (информационных систем) возможны следующие варианты экономии ресурсов:

- экономия средств на приобретение и сопровождение программного обеспечения в результате перехода на клиент-серверные технологии. Можно использовать бесплатные или условно бесплатные пакеты;

- нет необходимости в установке более современных клиентских компьютеров, т.к. программа не требует значительных вычислительных мощностей и работоспособна на имеющемся оборудовании.

## **7.2 Экономия энергоресурсов в результате внедрения программного обеспечения**

Мероприятия по энергосбережению могут быть разными. Один из самых действенных способов увеличения эффективности использования энергии – применение современных технологий энергосбережения.



Технологии энергосбережения не только дают значительное уменьшение расходов на энергетические затраты, но и имеют очевидные экологические плюсы.

Рассчитаем экономию электрической энергии при работе автоматизированной системы по формуле (7.2):

$$\mathcal{E} = (T_p - T_a) * P * C_{эл} * K_u \quad (7.2)$$

где:  $T_p$  – трудоемкость работы вручную, часов;

$T_a$  – трудоемкость работы с помощью программного продукта, ч;

$P$  – паспортная мощность персонального компьютера, кВт;

$C_{эл}$  – стоимость одного кВт/ч электроэнергии, руб;

$K_u$  – коэффициент использования персонального компьютера.

Для проведения всего цикла работы от реализации всех входных данных до расчетов тратится около одного человеко-часов рабочего времени с использованием одного персонального компьютера, для той же операции с применением разработанного программного продукта требуется порядка одного человека-часа на том же компьютере. По состоянию на 1 января 2020 года, стоимость одного кВт/ч электроэнергии в Республике Беларусь равна 0,21815 рублей. Приняв паспортную мощность персонального компьютера в 0,41 кВт, рассчитаем стоимость сэкономленной электроэнергии за один месяц при пятидневной рабочей недели и восьмичасовом рабочем дне:

$$\mathcal{E} = (2 - 1) * 0,41 * 0,21815 * (5 * 8 * 4) * 0,5 = 7,16 \text{ руб.}$$

Данный расчет показывает, что при внедрении разработанного программного продукта, можно сэкономить на электроэнергии 7,16 рублей в месяц, а за год данная сумма составит 85,92 рублей.

## ЗАКЛЮЧЕНИЕ

Исходя из постановки задачи дипломной работы, было разработано Web и android приложения автоматизации учета работы врача-анестезиолога, которое имеет ряд преимуществ по сравнению с аналогами. Приложение простое в использовании, имеет интуитивно понятный интерфейс. Интерфейс имеет современный и красивый дизайн, приложение можно использовать на всех видах устройств. При всех этих плюсах оно дешевое по сравнению с аналогами на рынке.

Изначально был сделан аналитический обзор существующих аналогов, который позволил ознакомиться с существующими решениями на рынке.

После постановки задачи был разработан дизайн. Затем было реализовано веб-приложение, которое представляет такие возможности, как ведения карточки пациента. У админа есть возможность добавлять роли пользователям, а также вести справочники. Доктор создает пациентов, ведет карточку пациента, имеет возможность создавать документы. Медсестра ведет прием лекарств пациентами.

Программное обеспечение имеет тесты, покрывающие места, которые могли приводить к проблемам при работе с приложением. Клиентская часть имеет обработку данных, вводимых пользователем, что также позволяет предотвратить ошибки при работе с данными.

В разделе 5 было произведено экономическое обоснование проекта и была рассчитана, учитывая отличие функционала, разработанного приложения от его аналогов, экономический эффект от производства и реализации его на рынке.

Разработанное программное обеспечение может использоваться в медицинских учреждениях, для помощи врачам-анестезиологам в их нелегкой работе.

В дальнейшем планируется выполнять поддержку и улучшение функционала разработанного программного обеспечения. А также возможное интегрирование в медицинскую систему, как одной из ее частей.

Поставленные цели данной дипломной работы были полностью достигнуты путём выполнения всех поставленных руководителем задач.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. История развития медицинских информационных систем - Sci.House [Электронный ресурс] - Режим доступа <<https://sci.house/tehnologii-meditsine-informatsionnyie-scibook/istoriya-razvitiya-meditsinskih-83600.html>> Дата обращения: 19.04.2020.
2. Стационар - МАПСОФТ [Электронный ресурс] - Режим доступа <<https://www.mapsoft.by/programs/medicine-software/hospital/>> Дата обращения: 19.04.2020.
3. Дейт К., Дж. Введение в системы баз данных. – 6-е изд. – К., М., СПб.: Вильямс, 2000. – 848 с.
4. Герберт Шилдт. Java 8. Полное руководство, 9-е издание = Java 8. The Complete Reference, 9th Edition. — М.: «Вильямс», 2015. — 1376 с.
5. Крис Шеффер, Кларенс Хо, Роб Харроп. Spring 4 для профессионалов = Pro Spring 4. — М.: «Вильямс», 2017. — 752 с.
6. Роберт Шелдон, Джоффри Мойе. MySQL 5: базовый курс = Beginning MySQL. — М.: «Диалектика», 2007. — 880 с.
7. Стив Макконеелл. Совершенный код — М.: Издательско-торговый дом «Русская Редакция»; СПб.: Питер, 2005. — С. 544.
8. ViewModel Overview – Android Developers [Электронный ресурс]. – Электрон. текстовые дан. – Режим доступа: <https://developer.android.com/topic/libraries/architecture/viewmodel>. – Дата доступа: 10.05.2020.
9. Тестирование. Фундаментальная теория [Электронный ресурс]. – Режим доступа: <https://habr.com/post/279535/> – Дата доступа: 15.05.2020.
10. Автоматизированное или ручное тестирование – что выбрать? [Электронный ресурс]. – Режим доступа: <http://merehead.com/blog-ru/manual-testing-vs-automated-testing-difference-definition-tools/> – Дата доступа: 15.05.2020.
11. ОСНОВЫ ГИГИЕНЫ ТРУДА И ПРОИЗВОДСТВЕННАЯ САНИТАРИЯ [Электронный ресурс]. – Режим доступа: <http://smorgonlizey.by/Metodkabinet/biblioteka/teorija/701.htm> – Дата доступа: 01.06.2020.