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

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

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Прикладная информатика в социальной сфере
(направленность (профиль)/специализация

БАКАЛАВРСКАЯ РАБОТА

на тему: «Разработка медицинской информационной системы для автоматизации работы регистратуры»

Студент	М.С. Ходырев	
	(И.О. Фамилия)	(личная подпись)
Руководитель	Е.А. Ерофеева	
	(И.О. Фамилия)	(личная подпись)
Консультанты	М.А. Четаева	
	(И.О. Фамилия)	(личная подпись)

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2019

АННОТАЦИЯ

Тема: «Разработка медицинской информационной системы для автоматизации работы регистратуры».

Целью данной выпускной квалификационной работы является разработка медицинской информационной системы для автоматизации работы регистратуры.

Объектом исследования является процесс обслуживания пациентов в регистратурах медицинских учреждений.

Предметом исследования является автоматизация процессов по обслуживанию пациентов.

В первом разделе производится анализ и функциональное моделирование предметной области. Выбираются технологии концептуального моделирования диаграмм «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ». Производится сравнение и анализ существующих на рынке решений и обосновывается необходимость автоматизированного варианта решения.

Второй раздел представляет собой описание логического проектирования информационной системы. Производится выбор технологии логического моделирования программной системы и проектирования базы данных. Производится концептуальное и логическое моделирование.

В третьем разделе производится описание физического проектирования информационной системы. В нее входит выбор архитектуры информационной системы, выбор технологий разработки, выбор системы управления базами данных, описывается физическая модель данных и производится разработка программной системы. В конце раздела описывается функциональность разработанной системы и производится ее тестирование.

Результатом выполнения бакалаврской работы является разработанная медицинская информационная система для автоматизации работы

регистратуры. Работа включает 74 страницы с приложениями, 30 рисунков, 4 таблицы и 35 источников.

ABSTRACT

The title of the bachelor’s thesis is "Development of a medical information system for automating the work of the registry".

The relevance of the bachelor’s thesis is due to the need to automate the work of the registry in a medical center.

The aim of the work is to develop a medical information system for automating the work of the registry.

The object of the bachelor’s thesis is the process of patient service in the registry of a medical center.

The subject of the bachelor’s thesis is the automation of patient service processes.

The first chapter is devoted to the analysis and functional modeling of domain processes. Particular attention is paid to selecting conceptual modeling technologies for building "AS-IS" and "TO-BE" charts. We also examine existing solutions on the development market, and next, we explain the need for an automated system.

The second chapter is a description of the logical design of the information system. We choose the technology of logical modeling and database design and produce conceptual and logical modeling.

The third chapter describes the physical design of the information system. It includes the choice of information system architecture, the choice of development technologies, the choice of a database management system, and a description of the physical data model. Finally, we describe the development process and test the information system.

The result of the bachelor’s thesis is the developed medical information system for automating the work of the registry. The bachelor’s thesis consists of an explanatory note on 74 pages, including 30 figures, 4 tables, the list of 35 references, including 5 foreign sources, and 5 appendices.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В медицинских учреждениях существует потребность в системах, обеспечивающих оперативный поиск пациентов, хранение выходной информации и вывод на физический носитель. Так же необходимо средство для учета обращений пациентов, ведения историй приемов, оказанных услуг и электронных медицинских карт. Специфика медицинских учреждений обязывает хранить как персональные, так и некоторые врачебные сведения о каждом пациенте. Это приводит к необходимости ведения базы данных пациентов и персонала с сложной структурой. Создание системы, объединяющей в себе систему биометрического и ручного поиска в результате обеспечило бы высокий прирост к скорости деятельности регистратуры медицинского учреждения. Внедрение такой «электронной регистратуры» должно повысить эффективность работы клиники. Пользователями этой системы являются регистраторы и администраторы.

Исходя из вышесказанного, делается вывод, что актуальность данной предметной области повышается с каждым годом, а, следовательно, с повышением актуальности, растет спрос на программное обеспечение для регистраторов медицинских учреждений.

Данная работа выполнялась по инициативе автора. Тенденция развития отрасли разработки программного обеспечения сложилась таким образом, что в большинстве случаев разработка ведется на базе существующего программного обеспечения в конкретной рассматриваемой предметной области. Вследствие чего, опираясь на уже существующее программное обеспечение, решается вопрос конкурентоспособности разрабатываемой программы, путем увеличения скорости и функциональности текущего продукта.

Целью данной выпускной квалификационной работы является разработка медицинской информационной системы для автоматизации работы регистратуры.

Объектом исследования является процесс обслуживания пациентов в регистратурах медицинских учреждений.

Предметом исследования является автоматизация процессов по обслуживанию пациентов.

Для глубокого исследования объекта необходимо построить множество моделей, которые смогут непротиворечиво и в полной мере раскрыть планируемую программную систему. IDEF0 и BPMN способны отразить функционирование аспекты системы на начальном этапе анализа. Для работы с данными строится модель сущность-связь методологии ERD, которая используется для проектирования баз данных. Для проектирования самой системы необходимо использовать методологию объектного проектирования на языке UML. UML включает в себя множество диаграмм, способных описывать статическое или динамическое взаимодействие объектов системы.

Для достижения поставленной цели в работе решаются следующие задачи:

- изучение предметной области;
- анализ существующих решений на рынке;
- внедрение биометрического поиска в систему;
- выбор технологий проектирования информационной системы;
- обоснование выбора архитектуры информационной системы;
- обоснование выбора технологий разработки;
- обоснование выбора системы управления базами данных;
- тестирование разработанной информационной системы.

Выпускная квалификационная работа состоит из трех разделов:

В первом разделе производится анализ и функциональное моделирование предметной области. Выбираются технологии концептуального моделирования для построения диаграмм «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ». Производится сравнение и анализ существующих на рынке решений и обосновывается необходимость автоматизированного варианта решения. Также

описываются требования к аппаратно-программному обеспечению. В конце раздела рассматривается математическая часть алгоритма биометрического распознавания пациентов.

Второй раздел представляет собой описание логического проектирования информационной системы. Производится выбор технологии логического моделирования программной системы и на основе выбора разрабатываются диаграммы классов, прецедентов и последовательности. Так же выбирается технология проектирования базы данных и производится ее непосредственное концептуальное и логическое моделирование.

В третьем заключительном разделе производится описание физического проектирования информационной системы. В нее входит выбор архитектуры информационной системы, выбор технологий разработки и выбирается система управления базами данных. Так же описывается физическая модель данных и производится непосредственная разработка программной системы. Для описания системы строятся диаграммы и описываются модули, включаемые системой. В конце раздела описывается функциональность разработанной системы и производится ее тестирования.

ФУНКЦИОНАЛЬНОЕ МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

1.1 Технико-экономическая характеристика предметной области

Подавляющее большинство государственных поликлиник на территории Российской Федерации работают по единой схеме, контролируемой министерством здравоохранения, в следствие чего представленные данные применимы ко множеству поликлиник.

Современная поликлиника является крупным многопрофильным, специализированным лечебно-профилактическим учреждением, предназначенным оказывать медицинскую помощь и осуществлять комплекс профилактических мероприятий по оздоровлению населения и предупреждению заболеваний.

В ее функции входят:

- оказание первой медицинской помощи при острых и внезапных заболеваниях, травмах;
- лечение больных при обращении в поликлинику и на дому;
- организация и проведение диспансеризации;
- экспертиза временной нетрудоспособности;
- освобождение больных от работы;
- направление на врачебно-трудовую экспертную комиссию лиц с признаками стойкой утраты трудоспособности;
- направление больных на санаторно-курортное лечение;
- своевременная госпитализация нуждающихся в стационарном лечении.

Поликлиника проводит большую профилактическую работу, противоэпидемические мероприятия, санитарно-просветительную работу среди обратившихся пациентов, изучает здоровье населения, выявляет раннюю заболеваемость, организует статистический учет и анализ показателей состояния здоровья населения.

Организационная структура городской поликлиники приведена на рисунке 1.1.

Рисунок 1.1 – Примерная организационная структура городской поликлиники для взрослых

Первое знакомство посетителей с поликлиникой начинается в регистратуре. Она является основным ее структурным подразделением по организации приема больных в поликлинике и на дому.

При обращении пациента, регистратор направляет его к нужному врачу или на консультацию к терапевту.

В задачи регистратуры входит:

- организация предварительной и неотложной записи больных на прием к врачу как при непосредственном обращении в поликлинику, так и по телефону;

- обеспечение четкого регулирования интенсивности потока населения с целью создания равномерной нагрузки врачей и распределение его по видам оказываемой помощи;
- проведение своевременного подбора и доставки документации в кабинеты врачей, правильное ведение и хранение картотеки поликлиники.

Рациональная организация приема призвана сократить время ожидания больных на прием к врачам.

Среднее время, затраченное на посещение поликлиники, в зависимости от цели посещения и ряда других объективных причин, связанных с особенностями обслуживания тех или иных больных, варьируется от 31,9 мин до 125,2 мин.

Управление сложным потоком больных в поликлинике обеспечивается внедрением прогрессивных форм организации труда врачебного и среднего медицинского персонала, а также путем совершенствования существующих форм работы регистратуры с учетом установленных норм нагрузок.

Работа регистратуры должна строиться по централизованной системе и исходить из участково-территориального принципа обслуживания населения и бригадного метода работы врачей поликлиники.

Работой регистратуры руководит заведующий, назначаемый на эту должность приказом главного врача поликлиники.

Медицинскому регистратору отводится большая роль в деле правильной организации приема больных. Он первым встречает больного, беседует с ним, в необходимых случаях помогает больному разобраться в организации приема.

Медицинский регистратор должен разбираться в вопросах сортировки больных по медицинским показаниям, что дает возможность правильно решать вопросы регулирования потока больных [27].

Таким образом, проектируемая система должна обеспечивать выполнение основных функциональных требований и удовлетворять потребностям предметной области.

1.2 Концептуальное моделирование предметной области

1.2.1 Выбор технологии концептуального моделирования предметной области

Концептуальное моделирование предметной области является одним из наиболее важных и приоритетных этапов при проектировании информационной системы. Отличительная особенность концептуальных моделей выражается в том, что информация отображается при помощи таблиц, графов, математических выражений, естественного языка, т. е. при помощи средств, понятных не только разработчику, но каждому человеку.

Исторически сложилось, что в качестве основополагающих методологий выступают четыре средства концептуального моделирование: IDEF0 (Integrated DEFinition), ARIS (Architecture of Integrated Information Systems), UML (Unified Modeling Language), а также BPMN (Business Process Model and Notation) [22].

UML представляет собой язык графического описания, предназначенный для объектно-ориентированного моделирования бизнес-процессов в основном для программных систем. Используется для создания абстрактной модели системы, которая называется UML моделью.

ARIS методология для моделирования бизнес-процессов организаций и программный продукт немецкой компании Software AG.

IDEF0 методология функционального моделирования бизнес-процессов. IDEF0 отображает функции и структуру системы, а также потоки информации, которые связывают эти функции.

BPMN – методология моделирования, анализа и реорганизации бизнес-процессов. Основной целью является обеспечение доступной нотацией описания бизнес-процессов каждого пользователя: от аналитиков, которые рисуют схемы процессов, и разработчиков, которые ответственные за введение технологий производства бизнес-процессов, до руководителей и простых пользователей, которые руководят данными бизнес-процессами и следят за их выполнением.

Чтобы выбрать наиболее подходящее средство, была построена таблица 1.1, в которой были указаны плюсы и минусы той или иной нотации.

Таблица 1.1 – Сравнительный анализ средств концептуального моделирования

Критерий	**IDEF0**	**ARIS**	**UML**	**BPMN**
Порог вхождения	Низкий	Высокий	Средний	Средний
Удобство построения моделей	Высокое	Высокое	Среднее	Высокое
Декомпозиция	Неограниченная			
Подход	Функцио-нальный	Процессный	Объектно-ориентированный	Процессный

Исходя из построенной выше таблицы 1.1 делается вывод, что нотация IDEF0 является наиболее подходящей для построение концептуальной модели предметной области.

Помимо выбранной в текущем разделе методологии концептуального моделирования IDEF0, используемой при построении диаграмм «КАК ЕСТЬ» и «КАК ДОЛЖНО БЫТЬ», необходимо произвести общее моделирование бизнес-процессов предметной области, для этой задачи была выбрана методология BPMN.

1.2.2 Моделирование бизнес-процессов регистратур городских поликлиник для постановки задачи автоматизированного варианта решения

Моделирование ставит перед собой цель описать все осуществляемые работы и этапы обработки информации. Регистратурой выполняются действия по первичной работе с пациентом. Необходимо грамотно и быстро предоставить всю интересующую клиента поликлиники информацию.

Моделирование бизнес-процессов регистратуры отображено на рисунке 1.2.

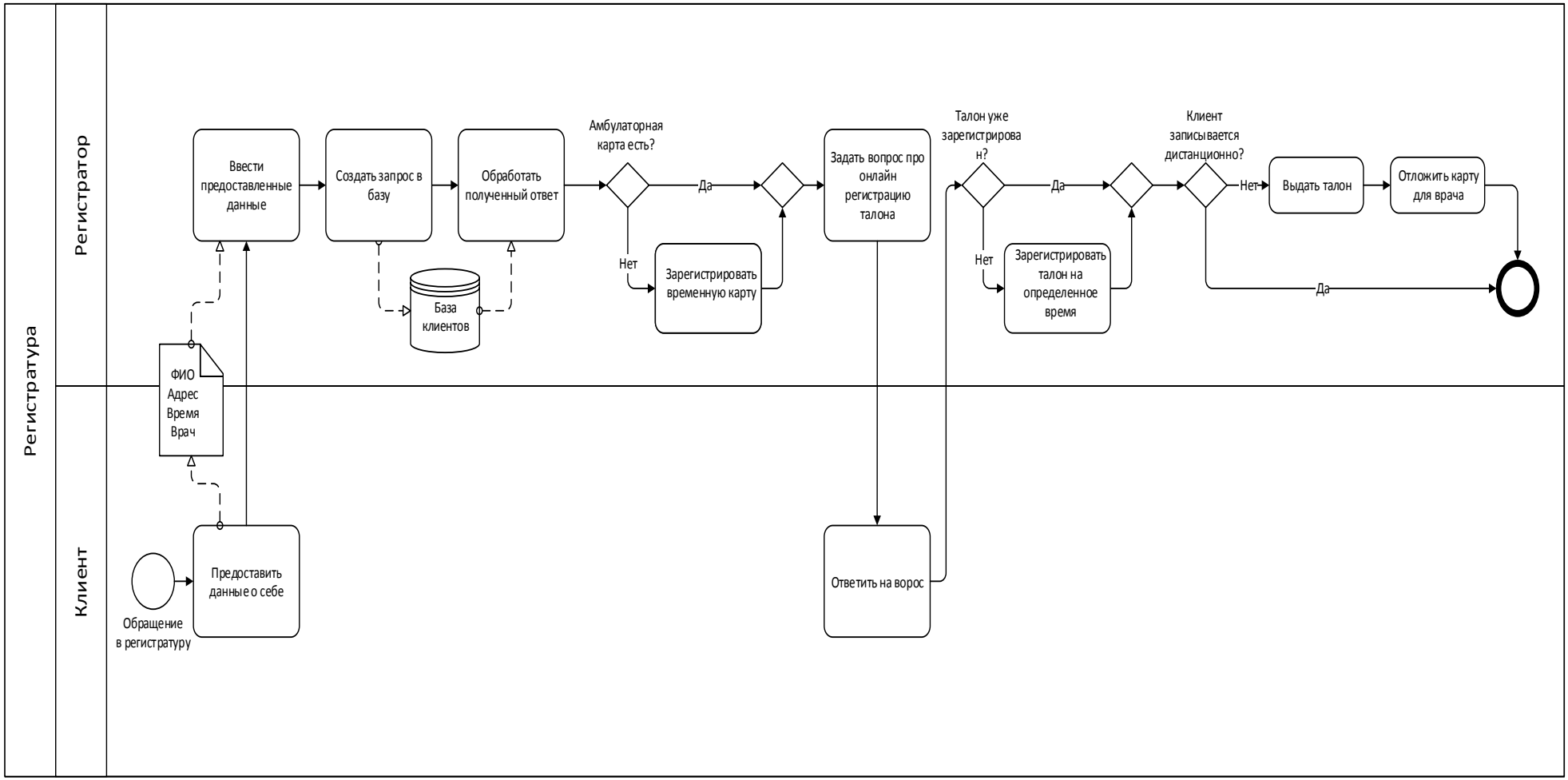

Рисунок 1.2 – BPMN диаграмма бизнес-процесса регистратуры городских поликлиник

На рисунке 1.2 приведена BPMN диаграмма, отображающая работу регистратуры городских поликлиник. В рамках выпускной квалификационной работы необходимо оптимизировать работу регистратуры, путем внедрения более совершенного оборудования и программного обеспечения. Из этого следует, что необходимо воспользоваться нотацией IDEF0 для построения диаграммы «КАК ЕСТЬ» и ее дальнейшей декомпозиции.

Данные шаги несут целью не только отобразить и раскрыть процессы, протекающие внутри, но также с их помощью можно наиболее полно и грамотно поставить задачу автоматизации.

1.2.3 Разработка и анализ модели бизнес-процесса «КАК ЕСТЬ»

Для медицинских учреждений скорость работы является неотъемлемым фактором, который показывает производительность труда. Квалифицированный медицинский персонал – это лишь часть успеха, во многом отзыв пациента о поликлинике формируется от эффективности работы регистратуры.

Анализ деятельности подразделения позволяет определить и построить существующую на данный момент бизнес модель регистратуры. Модель позволяет найти потенциально неэффективные бизнес-процессы, найти уязвимости системы, связанные с человеческим фактором. По построенным моделям система подвергается реинжинирингу, что способствует повышению эффективности и сокращению временных затрат на оказываемые услуги [12].

Для моделирования деятельности регистратуры будет использоваться Erwin Process Modeler 7.3, который поддерживает множество методологий, включая IDEF0.

Нулевой уровень методологии IDEF0 рассматривает все отделение в целом, что на диаграмме будет отображаться в виде одного единственного функционального блока «Деятельность регистратуры».

В целях подробного изучения бизнес-процессов регистратуры городских поликлиник, была построена IDEF0 диаграмма бизнес-процессов (рисунок 1.3)

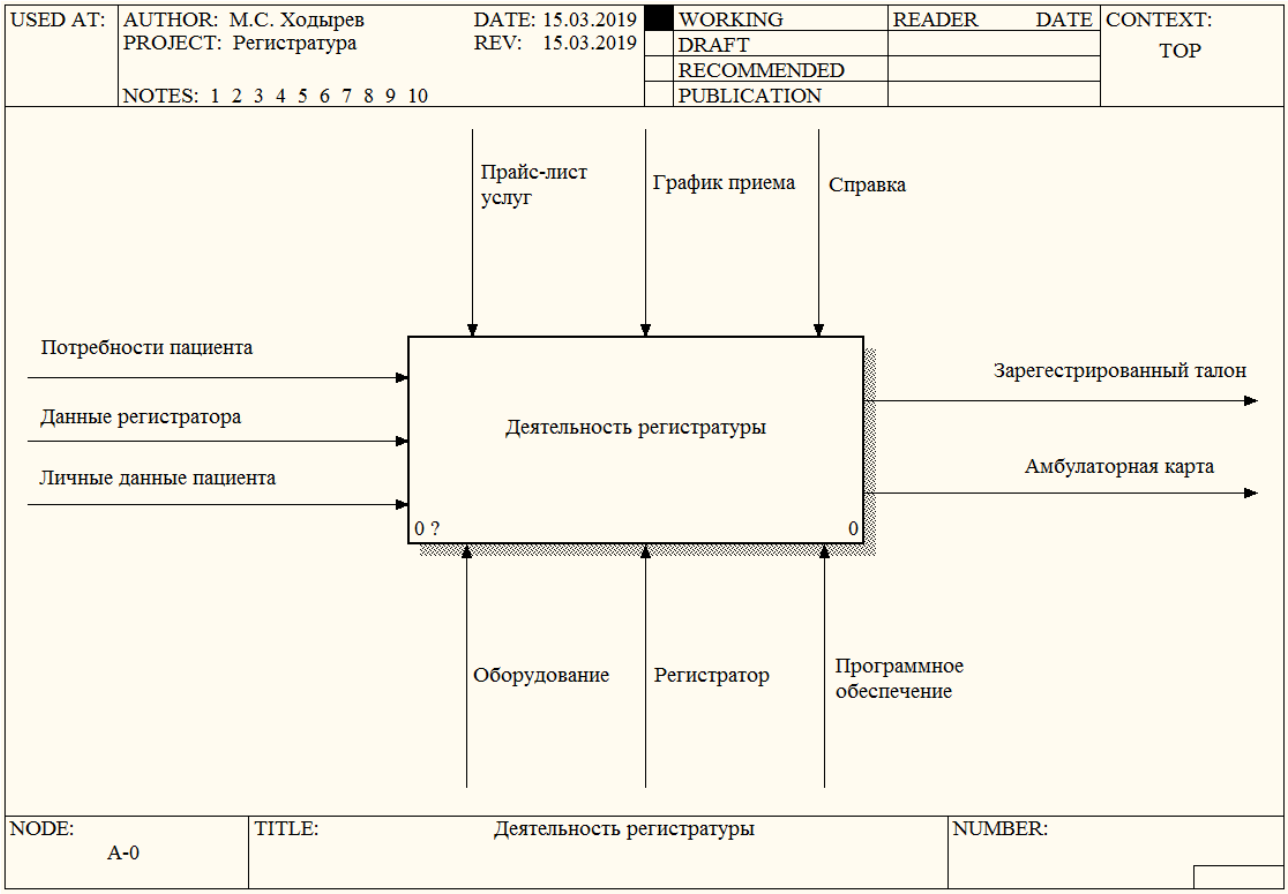

Рисунок 1.3 – IDEF0 диаграмма бизнес-процесса регистратуры

Из рисунка 1.3 видно, что бизнес-процесс верхнего уровня получает на вход личные данные персонала, а именно текущего регистратора, а также личные данные и потребности пациента. На выходе бизнес-процесса пациенту предоставляются амбулаторная карта и талон, зарегистрированный на согласованное сторонами время.

Следующим шагом после построения абстрактной модели нулевого уровня, является его декомпозиция по бизнес-процессам внутри него.

Детальное описание диаграммы раскроет модель более полно, отобразит внутренние особенности бизнес-процесса «Деятельность регистратуры», что послужит следующим шагом в обнаружении малоэффективного процесса, требующего переосмысления в рамках работ по реинжинирингу.

Была произведена декомпозиция нулевого уровня (рисунок 1.4) изображенного на рисунке 1.3.

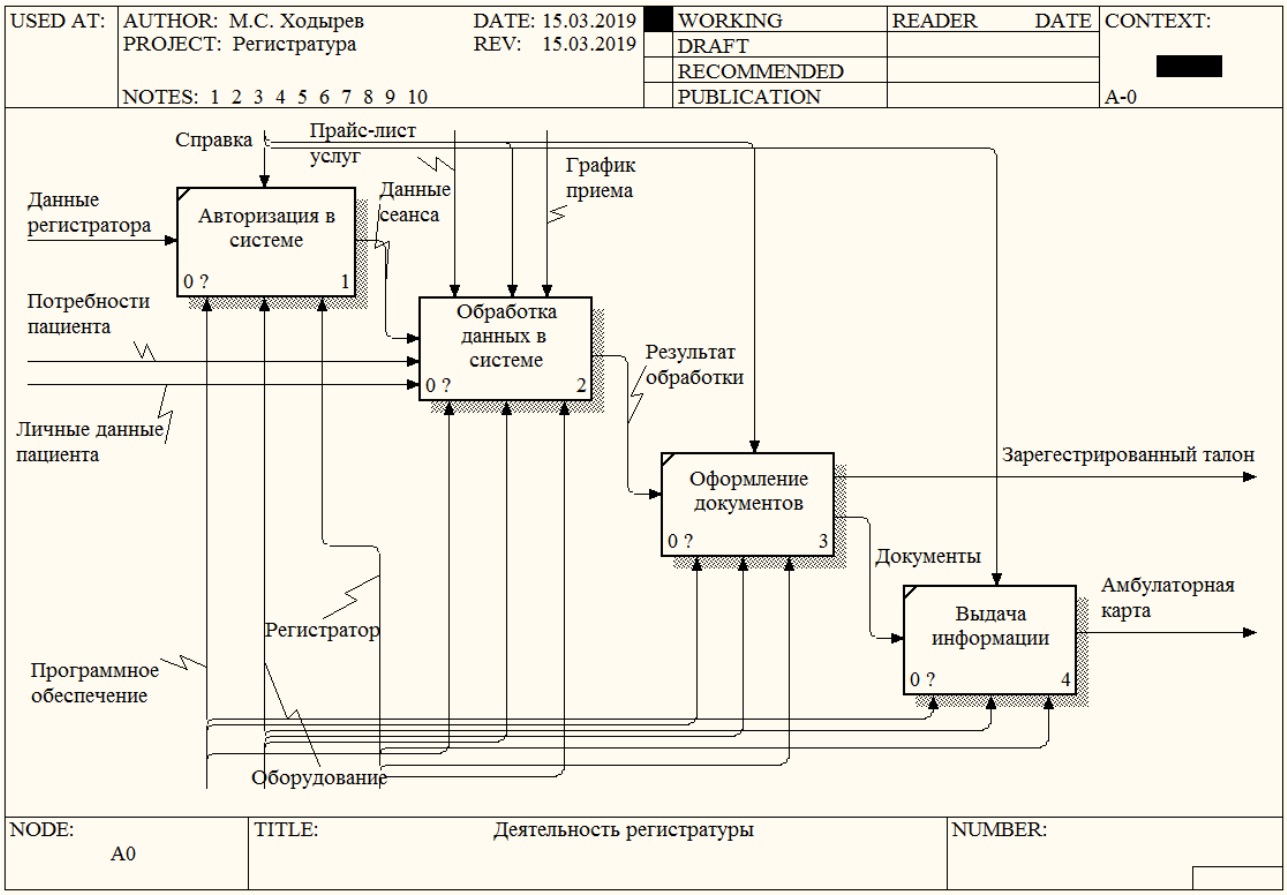

Рисунок 1.4 – IDEF0 диаграмма декомпозиции бизнес-процесса «Деятельность регистратуры»

Из рисунка 1.4 видно, что нулевой бизнес-процесс «Деятельность регистратуры» состоит из четырех подпроцессов: «Авторизация в системе», «Обработка данных в системе», «Оформление документов», «Выдача информации».

В первом процессе регистратор предоставляет свою информацию для авторизации в текущей системе. На данном этапе используется типичная общепринятая система авторизации путем введения логина и пароля, которая дают возможность идентифицировать текущего регистратора для внутренней работы системы.

Далее внутри процесса «Обработка данных в системе» производятся работы по обработке полученных от пациента данных. Регистратор при помощи полученной информации производит запись пациента ко врачу.

Данный бизнес-процесс необходимо разобрать подробнее, для дальнейшего построения.

Диаграмма декомпозиции бизнес-процесса «Обработка данных в системе» приведена на рисунке 1.5.

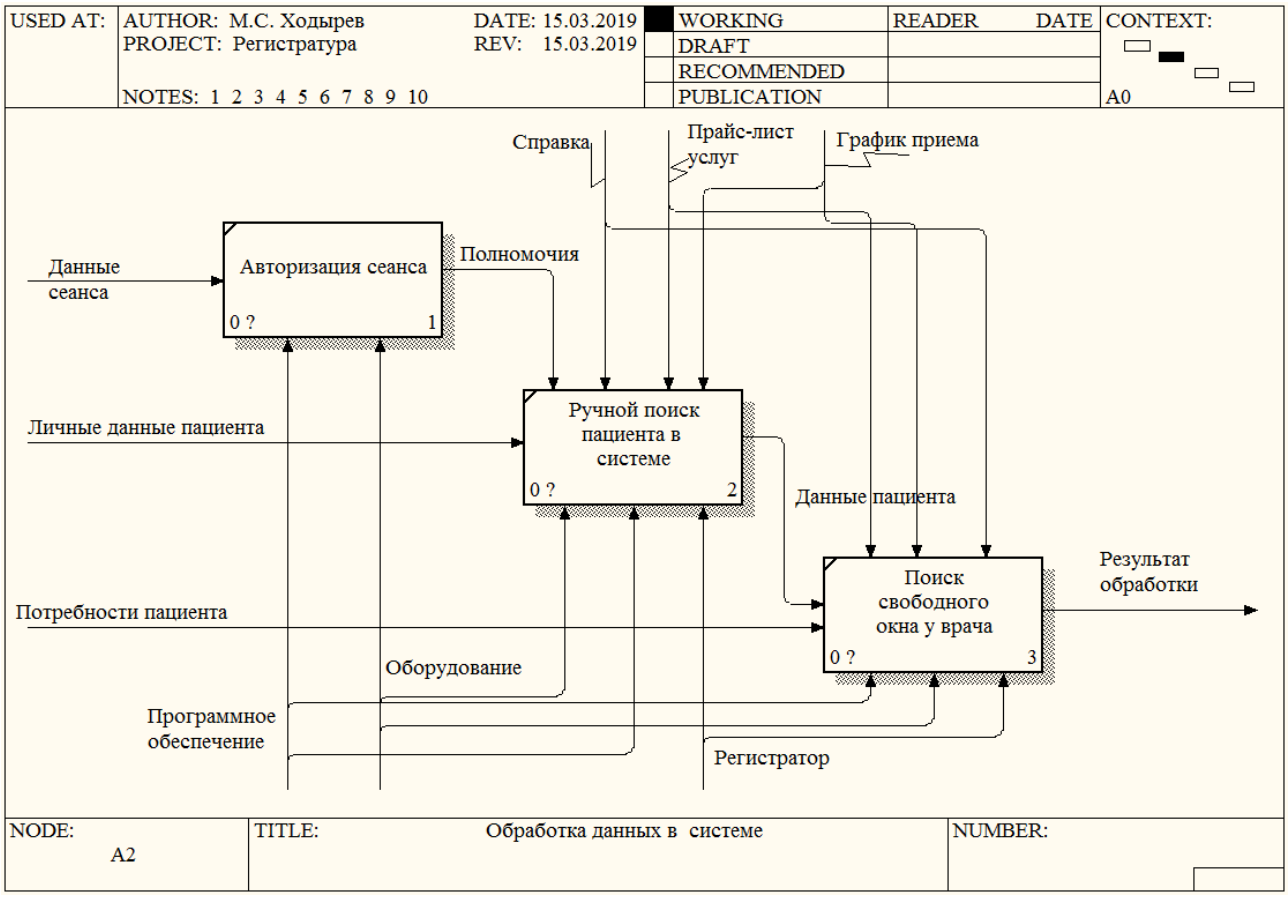

Рисунок 1.5 – IDEF0 диаграмма декомпозиции бизнес-процесса «Обработка данных в системе»

Далее на стадии оформления документов регистратор производит работы по формированию платежного документа и талона.

В заключительном процессе сформированные документы непосредственно выдаются заинтересованным лицам.

Исходя из приведенных диаграмм и анализа полученной в следствии построения информации, видно, что для оперативной работы регистратуры немаловажную роль играет качественное и продуманное программное обеспечение, заточенное под область ее применения. Так как регистратуры

поликлиник имеют ряд проблем, связанных со скоростью и оперативностью обслуживания, необходимо рассмотреть задачу внедрения более современного и технологически более продвинутого программного обеспечения, которое позволит увеличить скорость обслуживания пациентов. Без должного внимания к приведенным слагаемым успеха программное обеспечение будет не конкурентоспособно, что приведет отказу от его эксплуатирования.

1.2.4 Обоснование необходимости автоматизированного варианта решения и формирование требований к новой технологии

Регистратура является одним из самых важных и ключевых отделов в поликлинике. С точки зрения пациента, регистратура – этот тот элемент в цепочке операций, который способствует их скорому приему у определенного специалиста. Нынешние реалии таковы, что даже при достаточном количестве окон регистраторов, пациенту необходимо достаточно долгое время стоять в очереди, что отражается на его самочувствии в худшую сторону.

Медицинская информационная система предлагает ряд особенностей, которые способны изменить ситуацию в лучшую сторону как для регистраторов, так и для самих пациентов. В список изменений входят:

- создание единой базы данных;
- автоматизированная обработка и выдача отчетной информации;
- выработка системы хранения, использования и предоставления информации по запросу;
- использование современных технологий разработки, повышающих производительность и скорость системы;
- объединение необходимых для работы регистратуры модулей в одной системе;
- более понятный интерфейс с продуманной системой горячих клавиш;

- измененная модель регистрации пользователь с возможностью хранения биометрических данных;
- возможность поиска клиента по биометрическим данным (по чертам лица);
- автоматическое заполнение форм талонов при использовании биометрического поиска;

Исходя из приведенных выше изменений системы, автоматизированный вариант решения является целесообразным, так как способен повысить скорость обслуживания пациентов внедрением более современных технологий с разработанным биометрическим поиском по чертам лица.

После того как были описаны все недостатки текущего процесса, необходимо описать требования к новой проектируемой медицинской информационной системе. Перейдем к формированию требований, которым необходимо соответствовать будущей медицинской информационной системе.

При формировании требований упор делался на модель FURPS+ [28], которая четко классифицирует требования к программным системам. Аббревиатура FURSPS подразумевает следующее:

- Functionality (функциональность);
- Usability (удобство использования);
- Reliability (надежность);
- Performance (производительность);
- Supportability (пригодность к поддержке и эксплуатации).

Конечный символ «+» расширяет возможности модели FURPS, добавляя ей некотоыре ограничения разведенные на следующие группы:

- Design (ограничения проектирования);
- Interface (ограничения на интерфейсы);
- Physical (физические ограничения);
- Implementation (ограничения разработки).

Функции медицинской информационной системы должны обеспечивать:

- разделенную систему «Клиент-сервер» с трехуровневой архитектурой;
- защиту хранимой информации от несанкционированного доступа;
- выполнение ключевых задач, поставленных перед регистратурой;
- возможность автоматического формирования и распечатки отчетных данных;
- возможность бесперебойного доступа сотрудников к медицинской информационной системе.

Таким образом, в результате глубокого анализа были сформированы и описаны требования к проектируемой медицинской информационной системе, которые требуются для дальнейшей ее разработки.

1.3 Анализ существующих разработок на предмет соответствия сформулированным требованиям

1.3.1 Определение критериев анализа

При проведении работ по определению критериев анализа медицинской информационной системы необходимо учитывать требования, соблюдение которых играет важную роль в любом программном продукте, а не только в специализированных системах для медицинских учреждений. С учетом этого были выделены следующие требования:

Требования к функциональности системы:

- разграничение доступа;
- вывод дополнительной графической информации;
- вывод на печать;
- отслеживание произведенных операций и транзакций;
- работа с текстовыми данными;
- работа с биометрическими данными.

Требования к удобству использования:

- эстетичный и логичный интерфейс;

- вывод эксплуатационная документация.

Требования к надежности:

- возможность восстановления системы после сбоев;
- возможность восстановления связи с сервером после сбоев.

Требование к производительности:

- скорость запуска;
- время отклика системы;
- пропускная способность системы.

Требования к поддержке:

- возможность расширения системы;
- бесплатное распространение.

Программные системы будут тестироваться в соответствии с сформированными требованиями.

1.3.2 Сравнительная характеристика существующих разработок

После формирования требований к системам необходимо привести список сравниваемых программных продуктов в виде таблицы с плюсами и минусами. Полученный в итоге балл будет характеризовать соответствие системы сформулированным требованиям. В таблице 1.2 приведена сравнительная характеристика существующих ИТ-решение для медицинских учреждений.

Конфигурация «MedWork-Регистратура» - представляет собой медицинскую информационную систему, предназначенную для автоматизации Регистратуры. Призвано, оптимизировать работу учреждений, создать единое информационное пространство, повысить эффективность работы персонала, позволяет свести к минимуму бумажный документооборот, увеличить скорость доступа к информации, в целом увеличить прозрачность и эффективность управления учреждения [16].

Решение «МКТ-Современная регистратура» - решение «МКТ-Современная регистратура» предназначено для повышения доступности медицинской помощи, проявления открытости медицинской организации населению, упорядочения приема участковыми терапевтами (педиатрами) и врачами-специалистами, снижения социальной напряженности, сокращения очереди в регистратуру поликлиники за счет децентрализации записи, сокращения времени обслуживания пациента в регистратуре и повышения эффективности ее работы [17].

Программный комплекс для автоматизации работы регистратуры ООО «Ситиай-софт» - программный комплекс представляет собой информационную систему со встроенным модулем управления входящими и исходящими телефонными вызовами [18].

Таблица 1.2 – Сравнительный анализ программных продуктов

Требование	MedWork	МКТ	Ситиай-софт
Требования к функциональности системы			
разграничение доступа	+	+	-
вывод дополнительной графической информации	-	+	-
вывод на печать	+	+	+
отслеживание произведенных операций и транзакций	+	+	+
отслеживание произведенных операций и транзакций	+	+	+
работа с текстовыми данными;	+	+	+
работа с биометрическими данными	-	-	-
Требования к удобству использования			
эстетичный и логичный интерфейс	+	+	+
эксплуатационная документация	+	+	+
Требования к надежности			
возможность восстановления системы после сбоев	+	+	+
возможность восстановления связи с сервером после сбоев	-	+	-
Требование к производительности			
скорость запуска	+	+	+
время отклика системы	+	+	+
пропускная способность системы	+	+	+
Требования к поддержке			
возможность расширения системы	-	+	-

бесплатное распространение	-	-	-
Итог	11	14	10

Таким образом, в таблице 1.2 была произведена оценка систем и подсчет итоговых результатов, показывающих степень пригодности систем к данной предметной области. Исходя из результатов, полученных в таблице, видно, что каждое из сравниваемых программных продуктов удовлетворяет большинству поставленных требований. К явным минусам данных систем относится невозможность их бесплатного распространения, что является одним из важных критериев. Но основная проблема приведенных в таблице 1.2 систем в том, что они работаю лишь с текстовой информацией, в чем, в отличие от них, главное преимущество новой разрабатываемой медицинской информационной системы.

1.4 Постановка задачи на разработку проекта создания медицинской информационной системы

После произведения анализа в пункте 1.3 делается вывод, что каждая из систем имеет свои недостатки, которые выражаются в отсутствии тех или иных требований поставленных медицинской информационной системе и еще раз подтверждается необходимость разработки нового программного продукта, способного перенять и улучшить существующий функционал.

Были сформулированы цели и необходимые требования к будущей медицинской информационной системе.

Из основных идей разработки информационной системы следует выделить:

- создание единой базы данных;
- автоматизированная обработка и выдача отчетной информации;
- объединение необходимых для работы регистратуры модулей в одной системе;
- выработка системы хранения, использования и предоставления информации по запросу;

- использование современных технологий разработки для внедрения биометрического поиска;
- измененную модель регистрации пациентов с возможностью хранения биометрических данных;
- более понятный интерфейс с продуманной системой горячих клавиш.

Для достижения поставленных целей необходимо выполнить следующие задачи:

- произвести полный анализ предметной области;
- спроектировать архитектуру и выбрать технологию разработки программного обеспечения;
- спроектировать архитектуру и выбрать технологию разработки программного обеспечения;
- спроектировать структуру базы данных и выбрать систему управления базами данных для их хранения;
- выбрать оптимальный механизм распознавания и внедрить его в систему;
- свести затраты на разработку программного продукта к минимуму.

Со стороны непосредственно разработки самого программного продукта, а именно клиентское настольное программное приложение должно наглядно демонстрировать совершенные в нем действия, что обеспечивает безошибочную работу регистратора медицинских учреждений. По завершении операций необходимо сохранять в базу данных те действия, которые были произведены в системе. Для получения доступа к программному обеспечению необходимо идентифицировать себя как пользователя с определенной ролью через форму авторизации.

Со стороны серверной реализации при потере подключения клиента к серверу нужно производить попытки повторного подключения.

Так же системой должно обеспечиваться получения справочной информации как о самой системе, так и реализуемых ею возможностей.

1.5 Разработка модели бизнес-процесса «КАК ДОЛЖНО БЫТЬ»

На основе уже построенной модели «КАК ЕСТЬ» и проведенного анализа выносится как результат будущее предполагаемая структура предметной области. Взяв за основу существующие процессы модели «КАК ЕСТЬ», были устранены недостатки и добавлены нововведения, что предполагаемо приведет к оптимизации и усовершенствованию текущей работы регистратуры медицинских учреждений. Из основ реинжиниринга бизнес процессов известно, что разработку системы под конкретную область рекомендуется делать на основе бизнес-процесса «КАК ДОЛЖНО БЫТЬ», чтобы разрабатываемая медицинская информационная система строго соответствовала всем поставленным требованиям. Разработанная контекстная модель предметной области приведена на рисунке 1.6.

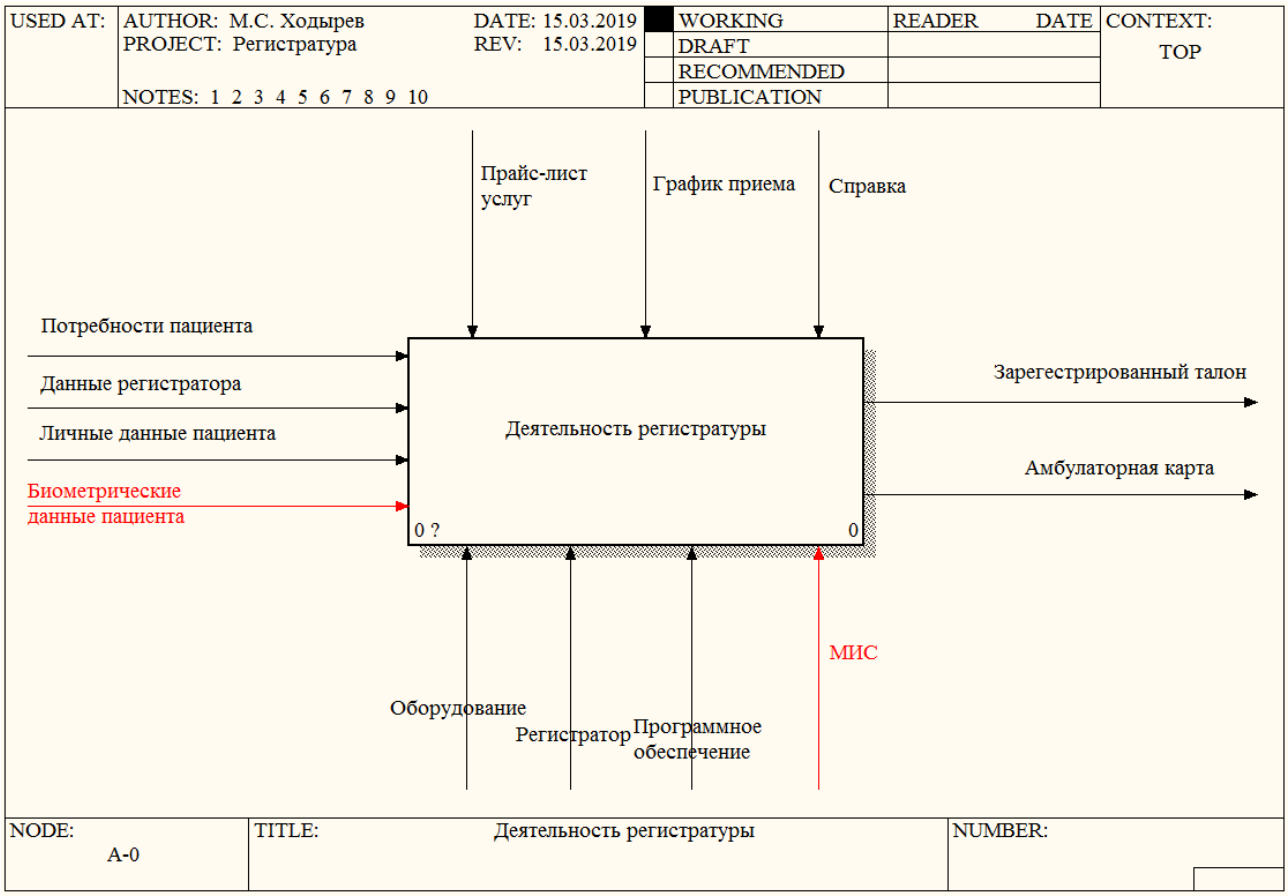

Рисунок 1.6 – IDEF0 диаграмма «КАК ДОЛЖНО БЫТЬ» бизнес-процесса «Деятельность регистратуры»

Из рисунка 1.6 видно, что в нулевой уровень добавились биометрические данные как вход бизнес-процесса, и так же к механизмам за место старого программного обеспечения добавилась новая разрабатываемая медицинская информационная система.

Новая бизнес-модель делает возможным ускорение работы регистратуры за счет возможностей современных технологий и биометрического распознавания, что делает не обязательным ручной ввод данных пациента. Так же стоит отметить, что возможность ручного поиска также остается в системе, так что биометрический поиск призван не полностью заменить предыдущий тип взаимодействия с системой, а дополнить его.

Для более детального рассмотрения бизнес-процесса «Деятельность регистратуры» проведем его декомпозицию. Декомпозиция приведена на рисунке 1.7.

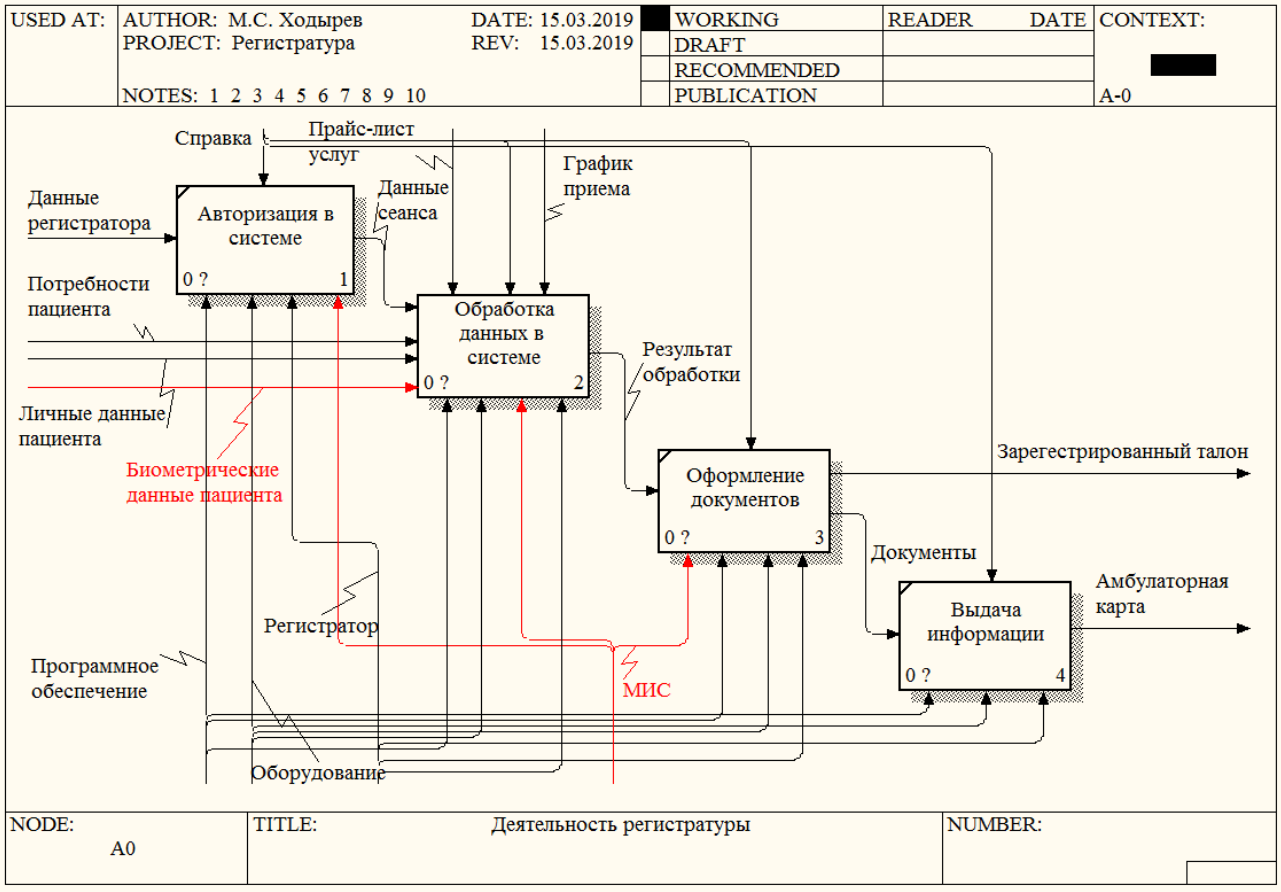

Рисунок 1.7 – IDEF0 диаграмма «КАК ДОЛЖНО БЫТЬ» декомпозиции бизнес-процесса «Деятельность регистратуры»

Из рисунка 1.7 видно, что биометрические данные идут как вход в бизнес-процесс «Обработка данных пациента в системе», следовательно, декомпозиция данного процесса позволит развернуто посмотреть на то, как должна работать система с предоставляемыми ей данными. Декомпозиция бизнес-процесса приведена на рисунке 1.8.

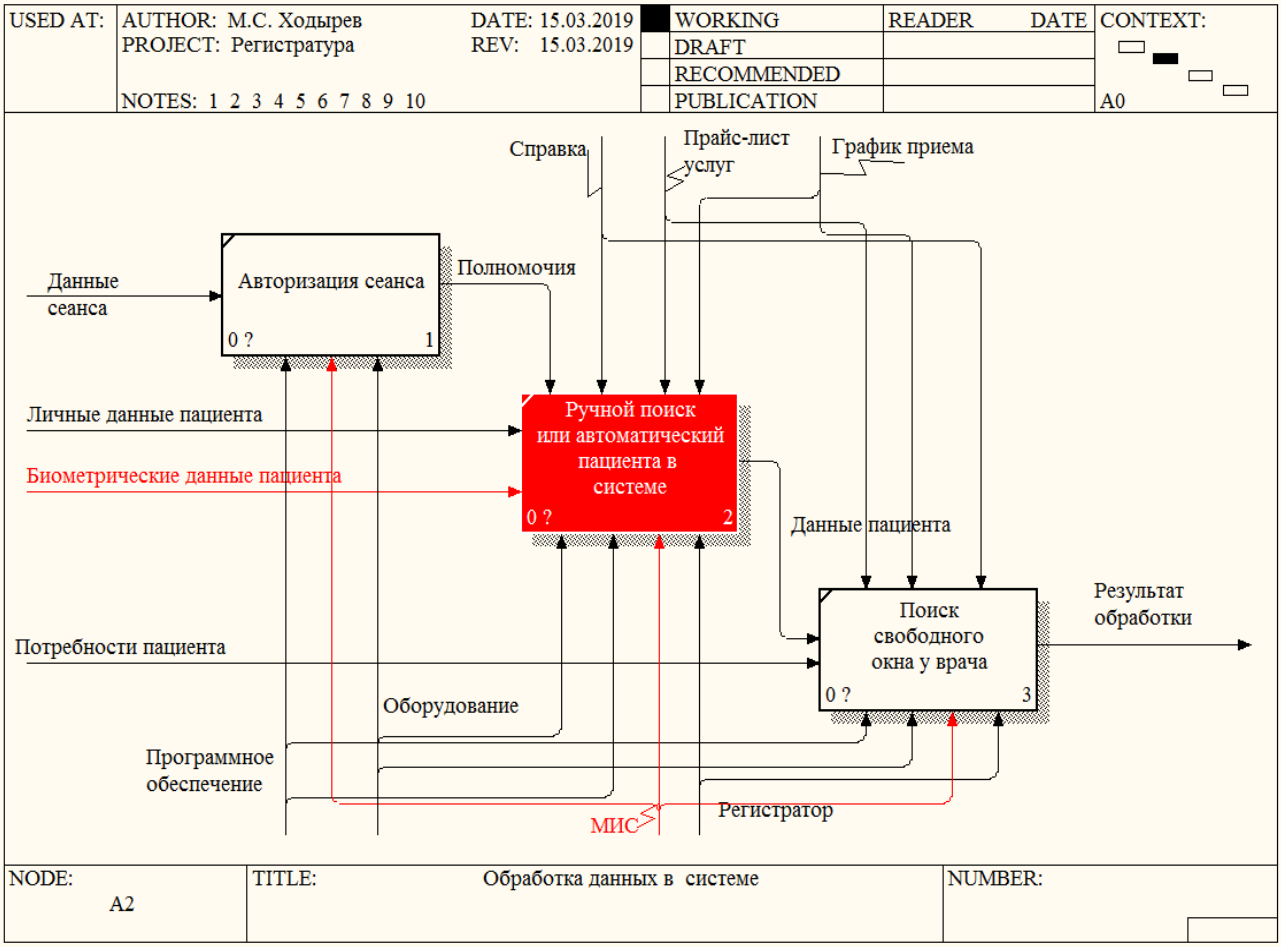

Рисунок 1.8 – IDEF0 диаграмма «КАК ДОЛЖНО БЫТЬ» декомпозиции бизнес-процесса «Обработка данных пациента в системе»

Следовательно, в бизнес-модели стал доступен вариант, когда поиск осуществляется не по текстовым данным, которые предоставляет пользователь, а по биометрическим, что расширяет возможности системы, приводя к ускорению протекания всей бизнес-модели в целом.

Выводы по разделу 1

В первом разделе выпускной квалификационной работы было произведено описание и функциональное моделирование предметной области. Была выбрана технология для концептуального моделирования и на основе выбора была построена диаграмма «КАК ЕСТЬ». Опираясь на построенную диаграмму «КАК ЕСТЬ» был произведен анализ бизнес-процессов и выявлены все узкие места, которые не соответствуют сформированным требованиям, поставленным данной предметной области. Так же для обоснования необходимости разработки нового программного продукта был произведен сравнительный анализ существующих систем. Данный анализ так же имеет целью помочь перенять и расширить функционал рассмотренных, уже созданных, систем, что должно сделать новую разрабатываемую медицинскую информационную систему более быстрой и технологически более продвинутой.

В заключении была разработана модель бизнес-процесса «КАК ДОЛЖНО БЫТЬ», в которую включаются нововведения, предложенные в рамках проведенного процесса реинжиниринга.

2 ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Выбор технологии логического моделирования медицинской информационной системы

Логическое проектирование информационной системы подразумевает под собой процесс разработки объектной модели системы и логической модели базы данных на основе уже созданной концептуальной модели, описанной в первом разделе работы.

В соответствии с концепцией бизнес-моделирования формализация модели проектируемой системы необходима для уточнения основных выводов из ее концептуальной модели и для постановки задачи на разработку программного обеспечения.

В методологии бизнес-моделирования на при построении логической модели системы предпочтение желательно отдавать методологиям объектно-ориентированного проектирования и анализа, использующим нотацию языка UML (Unified Modeling Language).

Для логического моделирования будет применяться объектно-ориентированная нотация языка UML пакета MS Visio.

Для демонстрации требований, описанных в первом разделе и предъявляемых к системе, используется диаграмма вариантов использования. В диаграмме вариантов использования продемонстрирована совокупность прецедентов и акторов, а также существующие отношения между ними. С помощью прецедентов моделируется поведение подсистемы или системы в целом.

Моделируя поведение элемента при помощи диаграммы вариантов использования, обеспечивается представление поведения информационной системы с высокой степенью детализации. Прецеденты делают возможным общение конечных пользователей и разработчиков на одном языке. Элементы, представляемые в диаграмме прецедентов, могут быть сложными

образованиями с большим количеством операций и составных частей. Описание прецедентов определенного элемента, даст понимание конечным пользователям каким образом с ним обращаться.

2.2 Логическая модель медицинской информационной системы и ее описание

В качестве инструмента проектирования логической модели медицинской информационной системы медицинских учреждений был выбран язык UML.

UML (Unified Modeling Language) - унифицированный язык моделирования)- язык графического описания для объектного моделирования в области разработки программного обеспечения, моделирования бизнес-процессов, системного проектирования и отображения организационных структур [22].

Язык UML является графическим языком. Данный стандарт применяется для создания абстрактной модели разрабатываемой системы, которую называют UML-моделью. Этот язык моделирования не относится к языкам программирования, но на основании разработанных моделей при помощи UML существует возможность генерации кода. Рассмотрим диаграммы, которыми в последствии воспользуемся для создания логической модели.

Создание логической системы несет целью переход от концептуальной точки зрения «КАК ДОЛЖНО БЫТЬ» к диаграмме прецедентов. Диаграмма отражает функциональные возможности системы взглядом со стороны, выявляет их взаимосвязь, а также способствует выявлению структуры системы. Важно понимать, что четкой корреляционной связи между диаграммой вариантов использования и диаграммой классов нет, так как система со стороны и система внутри – это разные понятия.

Разработанная диаграмма прецедентов приведена на рисунке 2.1.

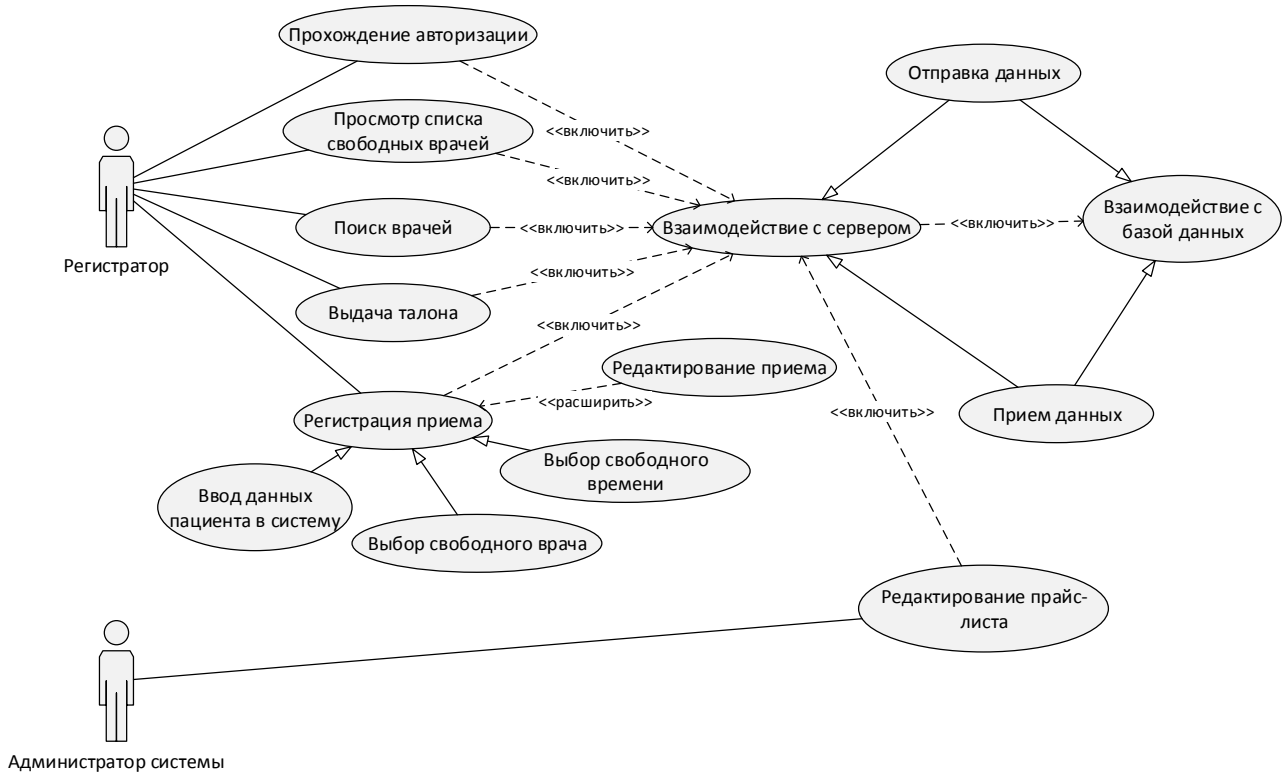

Рисунок 2.1 – Диаграмма вариантов использования программной системы регистратуры медицинских учреждений

Диаграмма вариантов использования программной системы на рисунке 2.1 демонстрирует базовый функционал программного продукта, который имеет возможность расширения при необходимости. Она отображает статическое поведение объектов в процессе. Основной акцент сделан на обобщенной схеме организации прецедентов, выполняющих определенные операции в системе [21].

Для динамического моделирования системы в нотации UML существует диаграмма последовательности, описывающая последовательность приема и отправки сообщений объектами системы, но стоит отметить, что она не акцентирует внимание на ролях, которые объекты играют во взаимодействии. При помощи диаграммы последовательности (рисунок 2.2) строится более приближенная к реальной работе системы диаграмма [20]. В качестве примера взаимодействия внутри системы будет рассмотрен прецедент «Регистрация

приема», который является основополагающим при работе в регистратурах медицинских учреждений.

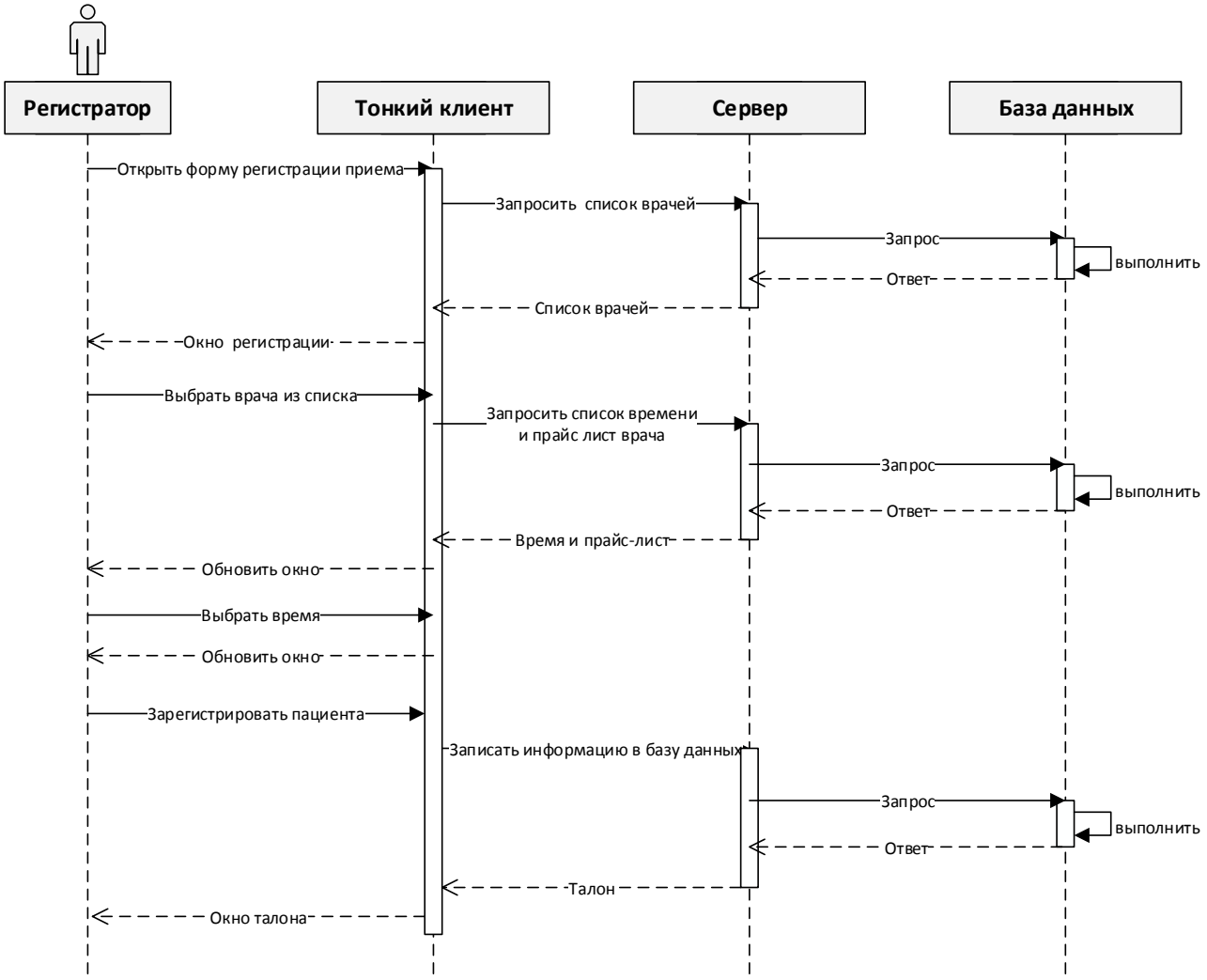

Рисунок 2.2 – Диаграмма последовательности регистрации приема пациента

Из рисунка 2.2 видно, что все операции с клиента были перенесены на сервер, который обрабатывает передаваемые ему запросы и перенаправляет их на исполнение в базу данных, таким образом, клиентское приложение, построенное по принципам тонкого клиента, не взаимодействует с базой данных на прямую, а предоставляет команды серверу через так называемые конечные точки (endpoints).

На основе построенных выше диаграмм были разработаны диаграммы классов, приведенные на рисунках 2.3 и 2.4, которые отображают статическое внутреннее устройство приложения [19].

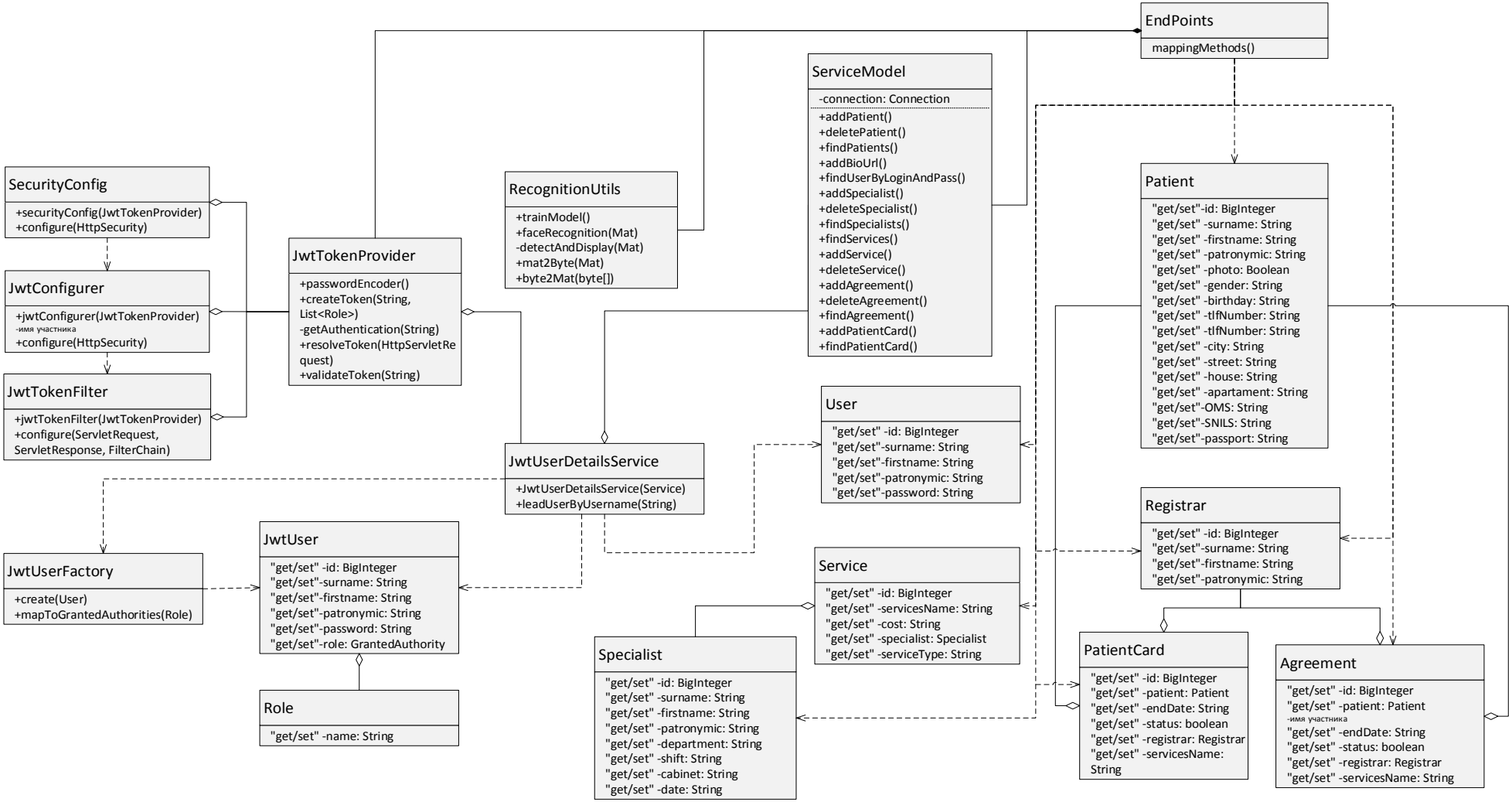

Рисунок 2.3 – Диаграмма классов серверной части МИС

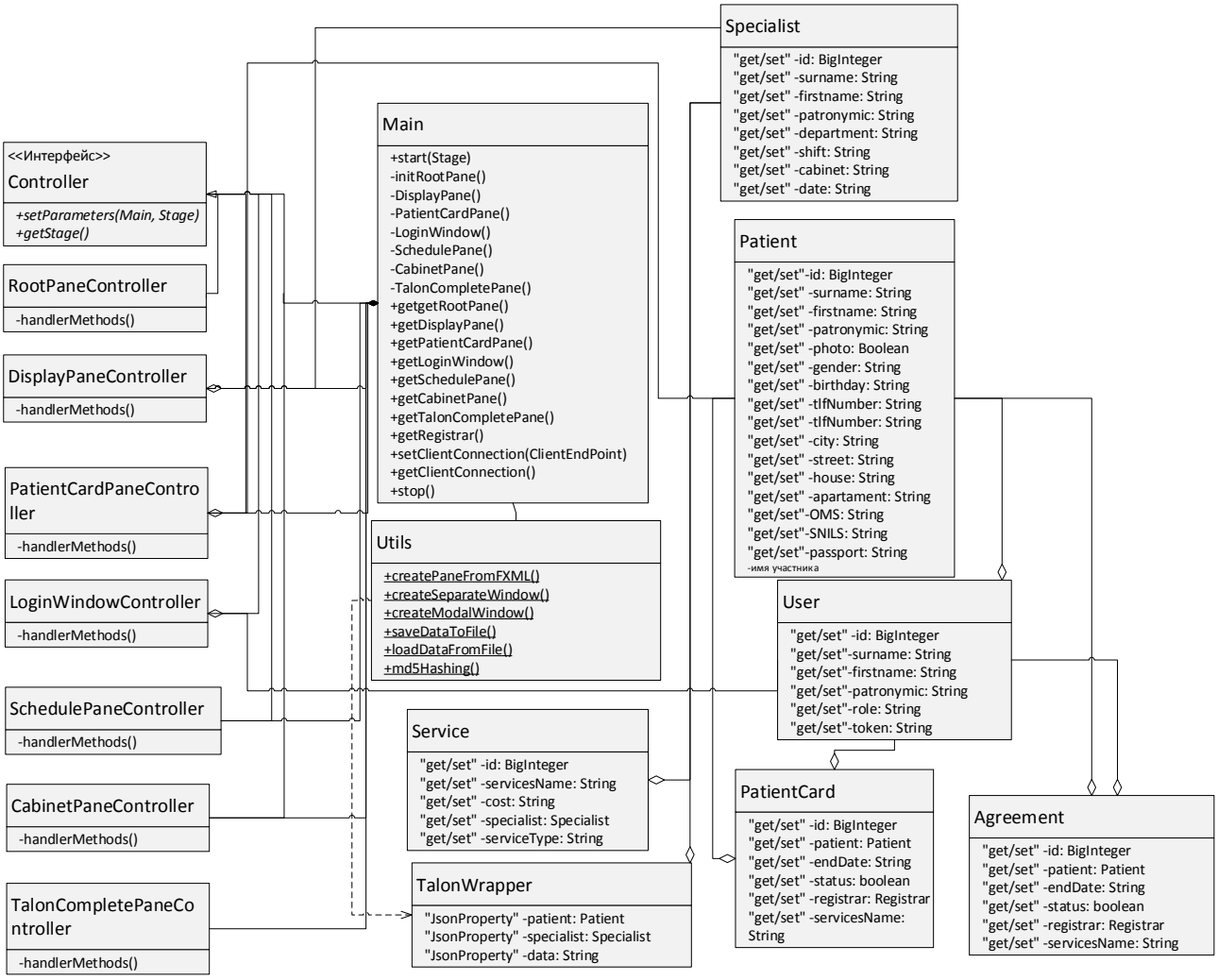

Рисунок 2.4 – Диаграмма классов клиентской части МИС

Интерфейс должен быть спроектирован с учетом экранных разрешений режимов отображения планшета: ни для одного сценария работы с приложением кнопки доступа к основным функциям не должны оказаться вне видимой области окна; для размещенных вне видимой области элементов управления должна быть предусмотрена возможность прокрутки; для всех допускающих прокрутку областей сам факт возможности прокрутки и метод прокрутки должен быть интуитивно понятен не менее чем 95% пользователей. Кнопки доступа к основным функциям приложения должны быть спроектированы с учетом антропометрической совместимости: пространственная компоновка форм и размерные характеристики кнопок

должны обеспечивать быстрый доступ к функциям приложения большими пальцами рук без стилуса и клавиатуры.

2.3 Проектирование базы данных медицинской информационной системы

2.3.1 Выбор технологии проектирования базы данных медицинской информационной системы

База данных – совокупность связанных данных, организованных по определенным правилам, предусматривающим общие для них принципы описания, манипулирования и хранения, независимая от прикладных программ.

База данных является динамической информационной моделью предметной области, а именно отображением внешнего мира. У каждого объекта существует ряд характерных для него свойств, параметров, признаков. Работа с БД осуществляется по атрибутам объектов. Каждая строка в базе данных содержит одну запись. Каждый столбец содержит все экземпляры одного конкретного типа данных. Согласно рекомендациям, каждая таблица в базе данных должна сдержать хотя бы один столбец с уникальным идентификатором.

Для создания физической модели данных разработчику необходимо выбрать конкретную СУБД и переключиться на физически уровень отображения диаграммы. Проектирование информационной системы, основанной на реляционной базе данных, будет выполнено средствами модели данных «сущность-связь». Модель «сущность-связь» (Entity Relationship) является визуальным средством представления объектов рассматриваемой предметной области, их характеристик (реквизитов) и отношений между объектами [13].

Так как на логическом уровне не рассматривается использование конкретной СУБД, и не определяются типы данных, то для построения структуры базы данных будет применяться CASE-средство ERwinDataModeler, при помощи которого разрабатывается модель «сущность-связь» в нотации IDEFX. ERwin – CASE-средство для проектирования баз данных, которое позволяет создавать, документировать и сопровождать базы данных,

хранилища и витрины данных [25]. Во время процесса проектирования структуры базы данных была принята корректной установка, при которой база данных должна пребывать в третьей нормальной форме.

2.3.2 Разработка концептуальной модели данных моделирования медицинской информационной системы

Целью концептуального моделирования является обеспечение наиболее понятных и естественных для человека способов представления и сбора информации, которая хранится в разрабатываемой базе данных. В следствии чего, концептуальная модель строится с использованием естественного языка. В качестве элементов при построении концептуальной модели базы данных являются сущности и связи между ними.

Главным этапом при разработке концептуальной модели, в первую очередь, является описание существующих объектов и их атрибутов. Объект – это сущность, обладающая определённым состоянием и поведением, имеющая заданные значения свойств и операций над ними. Атрибутом называют именованную характеристику объекта, при помощи которой моделируются его свойства.

Разработанная концептуальная модель приведена на рисунке 2.5.

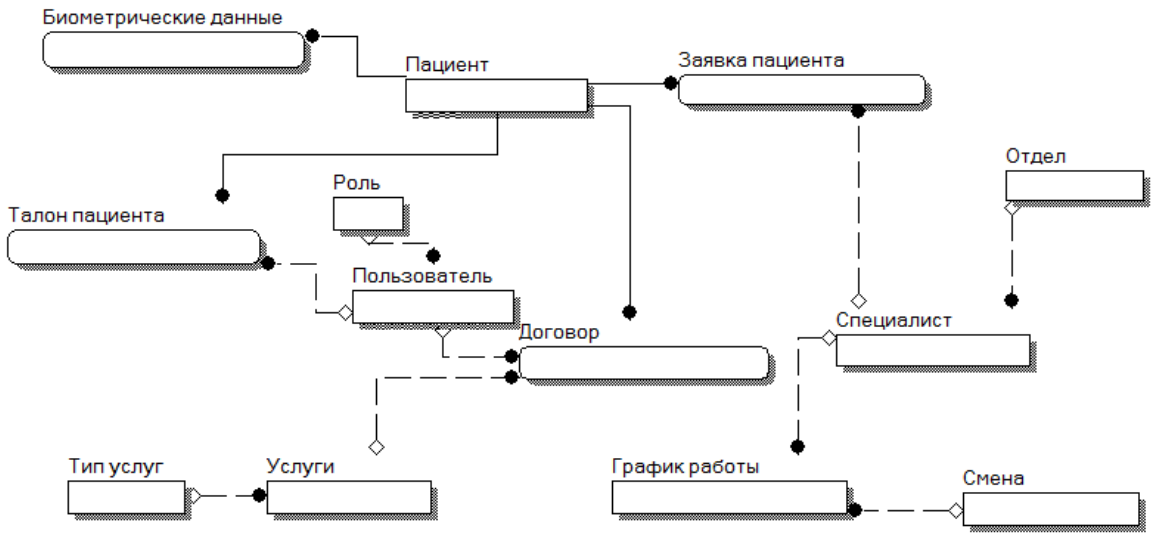

Рисунок 2.5 – Концептуальная модель базы данных регистратуры

Из рисунка 2.5 видно, что концептуальная модель полностью соответствует теоретическим требованиям, предъявляемым к подобного рода моделям и отображает связи между сущностями в базе данных.

2.3.3 Разработка логической модели данных моделирования медицинской информационной системы

Логическая модель данных представляет из себя некую структуру базы данных без привязки к конкретной системе управления базами данных. Эта модель производит расширения концептуальной модели путем определения атрибутов без их типов, а также уточнения состава сущностей и их взаимосвязей.

На основании концептуальной модели, построенной в прошлом подразделе, была разработана логическая модель базы данных.

Отличительной особенностью логической модели является то, что при ее построении уточняются атрибуты и типы сущностей, определяются их зависимости, а также проставляются первичные и внешние ключи.

Разработанная логическая модель базы данных приведена на рисунке 2.6.

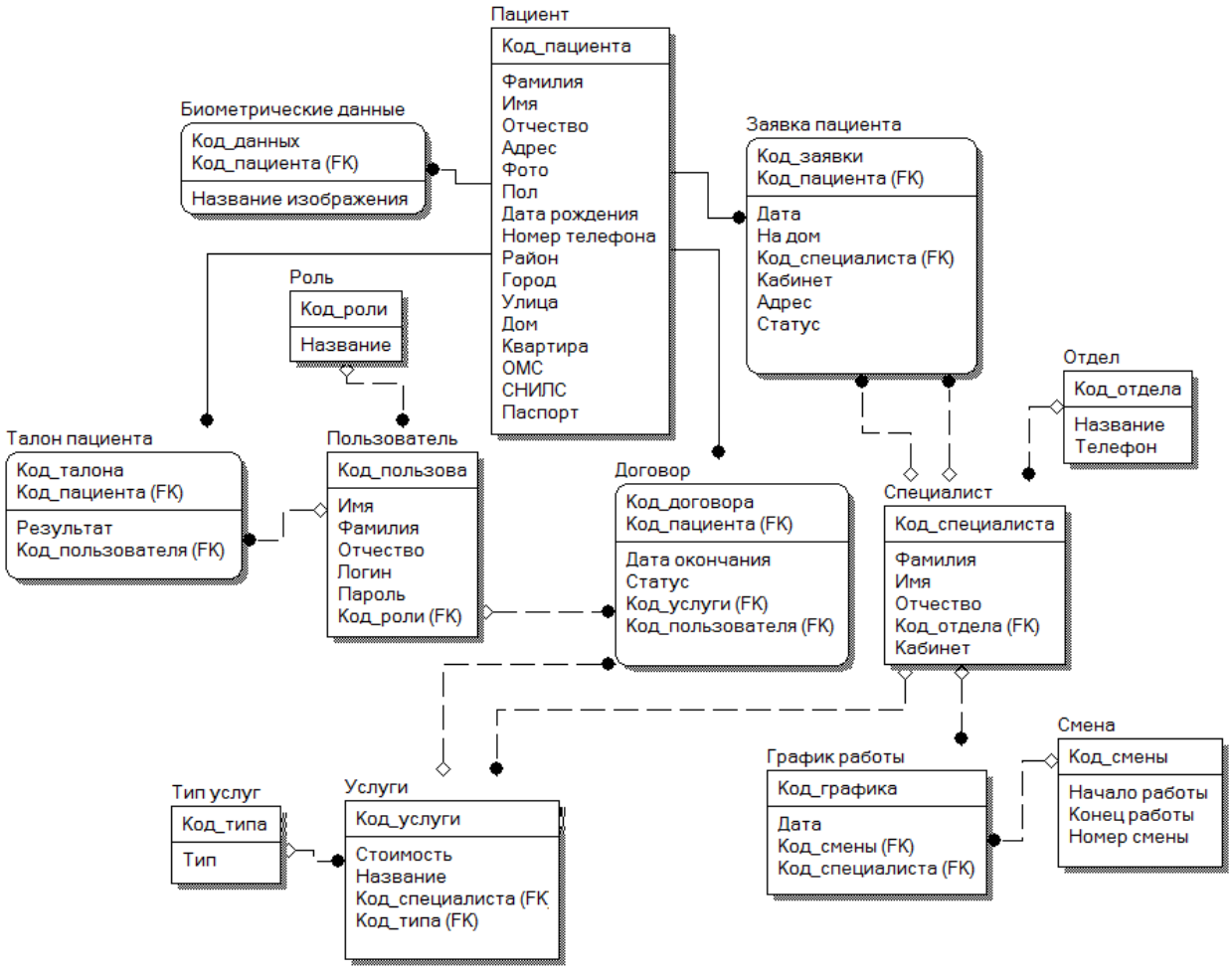

Рисунок 2.6 – Логическая модель базы данных регистратуры

Из диаграммы, изображенной на рисунке 2.6, видно, что база данных состоит из следующих таблиц: «Пациент», «Заявка пациента», «Отдел», «Специалист», «График работы», «Смена», «Услуги», «Тип услуги», «Талон пациента», «Биометрические данные», «Пользователь» и «Договор».

- сущность «Пациент» содержит в себе атрибуты, при помощи которых можно точно идентифицировать пациента поликлиники;
- сущность «Биометрические данные» представляет из себя ни что иное, как хранилище тестовых наборов для лицевого поиска пациента при обращении;
- сущность «Талон пациента» является зависимой от пациента и хранит в себе сведения о посещении пациентом специалиста;

- сущность «Заявка пациента» является зависимой от пациента и хранит в себе сведения о заявке пациента, которая может быть, как удаленной, так и инициированной при непосредственном присутствии самого пациента.
- сущность «Специалист» содержит в себе атрибуты, при помощи которых можно точно идентифицировать врача поликлиники;
- сущность «Отдел» содержит информацию об существующих в больнице отделах;
- сущность «График работы» хранит в себе внешнюю ссылку на сущность «Смена», они несут в себе целью хранения расписания в указанный в базе день;
- сущности «Услуги» и тип «Тип услуги» хранят информацию об услугах, предоставляемых медицинским учреждением;
- сущность «Договор» — это отражение реального договора на оказание услуг, заключаемого между пациентом и медицинским учреждением;
- сущность «Пользователь» хранит данные пользователя, работающего в системе;
- сущность «Роль» хранит список возможных ролей в системе.

Все хранимые в базе сущности связаны связью типа «один-ко-многим».

После построение концептуальной и физических моделей данных медицинской информационной системы следует рассмотреть требования к аппаратно-программному обеспечению.

Выводы по разделу 2

В данном разделе был произведен выбор технологии проектирования информационной системы и разработан ряд диаграмм, помогающих отразить логическое представление системы по раннее сделанным концептуальным диаграммам. В число разработанных диаграмм входят диаграмма классов и диаграмма последовательности, которые несут целью показать внутреннее устройство разрабатываемой системы в целом, а также раскрыть концепцию и

взаимодействие систем при проектировании трёхуровневой архитектуры приложения.

Так же была выбрана технология для проектирования базы данных в нотации IDEF1X. При помощи инструментального набора CASE-средства ERwinDataModeler была построена концептуальная модель системы, которая отображает сущности и их связи без конкретного описания внутреннего устройства сущностей. Для осуществления уточнения и дополнения концептуальной модели была продумана система внешних и первичных ключей, а также наполняющие сущности атрибуты, которые являются отражением реальных объектов предметной области.

3 ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ МЕДИЦИНСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

3.1 Выбор архитектуры моделирования медицинской информационной системы

При проектировании информационной системы предусматривается работа в многопоточном режиме, что делает необходимым разработку программного обеспечения с использованием трехуровневой архитектуры приложений, которая способна увеличить производительность.

Клиент – это интерфейсный компонент системы, предоставляемый конечному пользователю. Этот уровень не имеет прямых связей с базой данных и не нагружен основной бизнес-логикой, так же хранит состояние приложения. На этот уровень выносится только простейшая бизнес-логика: алгоритмы шифрования, интерфейс авторизации, несложные операции с данными (сортировка, группировка, подсчёт значений), уже загруженными на терминал проверка вводимых значений на допустимость и соответствие формату.

Сервер приложений находится на втором уровне, на нём сосредоточена наибольшая часть бизнес-логики. Вне его остаются фрагменты, экспортируемые на клиента, а также элементы логики, погруженные в базу данных. Реализация данного компонента обеспечивается специализированным программным обеспечением. Серверы приложений проектируются таким образом, чтобы добавление к ним дополнительных экземпляров обеспечивало горизонтальное масштабирование производительности программного комплекса и не требовало внесения изменений в программный код приложения.

Сервер баз данных обеспечивает хранение данных и выносится на отдельный уровень, реализуется, как правило, средствами систем управления базами данных, подключение к этому компоненту обеспечивается только с уровня сервера приложений.

По сравнению с двухзвенной клиент-серверной архитектурой или файл-серверной архитектурой трёхуровневая архитектура обеспечивает, как правило, большую масштабируемость, большую конфигурируемость. Реализация приложений, доступных из тонкого клиента, как правило, подразумевает развёртывание программного комплекса в трёхуровневой архитектуре [9].

Трехуровневая клиент-серверная архитектура сервера базы данных приведена на рисунке 3.1.

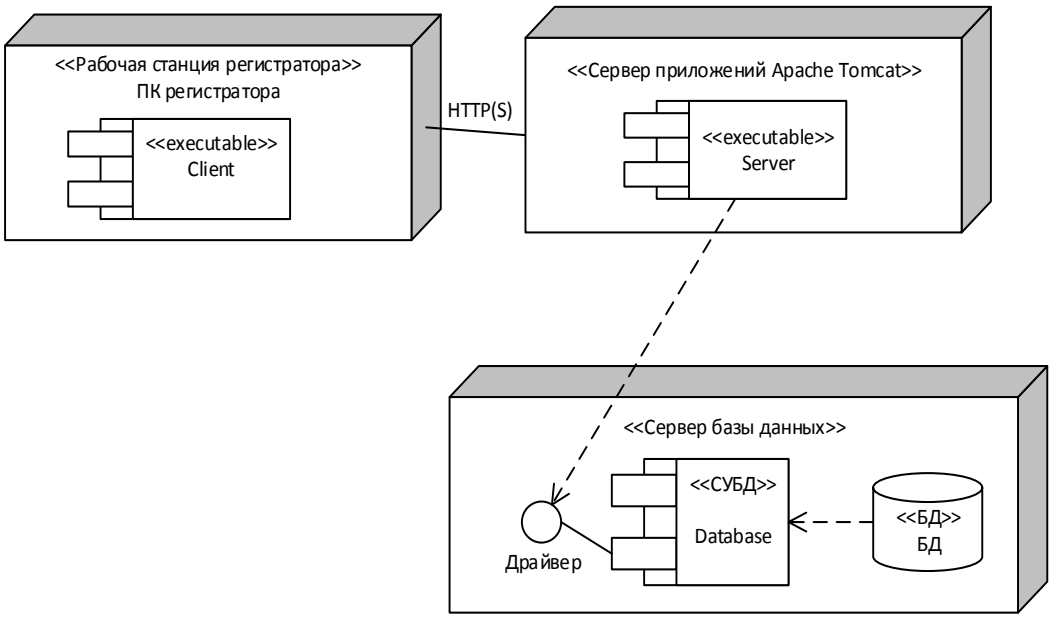

Рисунок 3.1 – Трехуровневая клиент-серверная архитектура приложения

Выбранная архитектура, изображенная на рисунке 3.1, обладает гибкостью, масштабируемостью, конфигурируемостью и безопасностью, что делает ее наиболее подходящей для поставленных в рамках разработки системы задач.

3.2 Выбор технологии разработки программного обеспечения моделирования медицинской информационной системы

При создании программного обеспечения на первый план выходит вопрос о том, какие технологии необходимо использовать, чтобы сократить время разработки и при этом не ухудшить программный продукт.

Рынок программного обеспечения сегодня предлагает множество полезных средств и инструментов для разработчика, способных ускорить и улучшить процесс разработки во много раз. Данные продукты поддерживают все технологические этапы проектирования, будь то кодирование, отладка или

тестирование. Пользователями таких продуктов являются прикладные и системные программисты.

В качестве операционной системы, на которой будет производится разработка программного обеспечения, была выбрана Windows 10.

Программирование будет осуществляться на языке Java, при помощи платформы Java SE (Standard Edition) 8-й версии, способной в полной мере удовлетворить потребностям разрабатываемой системы. Встроенные возможности Java SE включают такие полезные пакеты, как java.io – содержит классы для обеспечения файлового ввода-вывода информации, несколько классов абстракции ввода-вывода, а также набор классов для обработки вводимой информации: выделения токенов и т. д., java.sql – для работы с базами данных, а так же ряд других необходимых для разработки программного обеспечения пакетов.

Для инициации запросов и приема ответов со стороны клиентского приложения используется Java API JAX-RS 2.0.

Серверное REST API реализовано при помощи продукта Spring, включающего в себя такие полезные фреймворки как Spring MVC и Spring Security.

REST-приложение будет развернуто на сервере приложений Apache Tomcat.

Необходимые локальные и сетевые данные будут распространятся при помощи библиотеки Jackson в формате json.

В качестве фреймворка для автоматизации сборки проекта был выбран Maven, конфигурация которого осуществляется на языке POM, который в свою очередь является подмножеством XML.

Для возможности работы с видеопотоком используется библиотека алгоритмов машинного зрения OpenCV, написанная на C++.

Для соединения с базой данных будет использоваться инструмент JDBC, являющейся платформенно независимым промышленным стандартом

взаимодействия Java-приложений с различными СУБД. Находится данный инструмент в пакете java.sql, входящим в состав Java SE.

В качестве средств разработки GUI (Graphical User Interface) будет использоваться платформа, созданная на основе Java для создания приложений с насыщенным графическим интерфейсов. Так как разрабатываемая медицинская информационная система является настольным приложением, следовательно, JavaFX является наиболее удобной платформой, декларативно описывающей пользовательский интерфейс при помощи xml подобного языка разметы – fxml. Так же, в качестве дополнительного к JavaFX инструментария, будет использоваться конструктор форм JavaFX Scene Builder, который способен легко интегрироваться со средой разработки и давать возможность полного визуального контроля над расположением элементов в форме [15].

В качестве среды разработки была выбрана к использованию среда IntelliJ IDEA, которая является самой популярной IDE (Integrated Development Environment) на текущий момент.

3.3 Выбор СУБД моделирования медицинской информационной системы

Выбор системы управления баз данных представляет из себя сложную многопараметрическую задачу и является одним из самых важных этапов при разработке информационных систем. Выбранный программный продукт обязан удовлетворять текущим и будущим потребностям регистратур медицинских учреждений, при этом следует учитывать финансовые затраты на приобретение оборудования, которое необходимо для функционирования самой системы, а также обучение персонала.

На рассмотрение выносятся три популярные СУБД: Oracle DB, MySQL, PostgreSQL [30].

MySQL – свободная реляционная система управления базами данных. Разработку и поддержку MySQL осуществляет корпорация Oracle, получившая права на торговую марку вместе с поглощённой Sun Microsystems, которая ранее приобрела шведскую компанию MySQL AB. Данная СУБД

ориентирована на полное соответствие стандартам SQL. Отлично подходит для малых и средних предприятий.

Microsoft SQL Server – система управления реляционными базами данных, разработанная корпорацией Microsoft. Основной используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями. Используется для работы с базами данных размером от персональных до крупных баз данных масштаба предприятия

PostgreSQL – свободная объектно-реляционная система управления базами данных. Подходит для малых и средних предприятий.

В ходе анализа сформированы основные требования к СУБД:

- быстродействие;
- кроссплатформенность;
- бесплатное распространение;
- хорошая документированность;
- простота и удобство использования в малых и средних проектах;
- опыт работы;
- распространённость.

Сравнительный анализ по отобранным критериям отображен в таблице 3.1.

Таблица 3.1 – Сравнение СУБД

Критерий	**MySQL**	**Oracle DB**	**PostgreSQL**
Быстродействие	+	+	+
Кроссплатформенность	+	+	+

Продолжение таблицы 3.1

Бесплатное распространение	+	+	+
Хорошая документированность	+	-	+

Простота и удобство использования в малых и средних проектах	+	-	+
Опыт работы	+	+	+
Распространённость	+	+	+
Итог	7	5	7

Из анализа в таблице 3.1 видно, что Oracle DB не подходит под разрабатываемую систему, так как он обладает большим функционалом и больше подходит для крупных проектов, нежели для малых и средних. Для таких целей больше подходят PostgreSQL и MySQL, которые при сравнении набрали одинаковое количество баллов. Исходя из собственного опыта работы с данными СУБД, выбор был сделан в пользу MySQL из-за субъективно более понятного синтаксиса и установки. Так же язык MySQL поддерживается в таком мощном инструменте разработчика, как HeidiSQL, что дополнительно подтверждает правильность сделанного в пользу MySQL выбора.

3.4 Разработка физической модели данных моделирования медицинской информационной системы

В отличии от логической модели, физическая несет целью отобразить структуру базы данных непосредственно на примере конкретной СУБД. Из пункта 3.3 видно, что выбор был сделан в пользу СУБД MySQL.

Для создания физической модели базы данных был использован специальный свободно распространяемый инструмент для визуального проектирования БД MySQL Workbench.

Разработанная физическая модель базы данных представлена на рисунке 3.2.

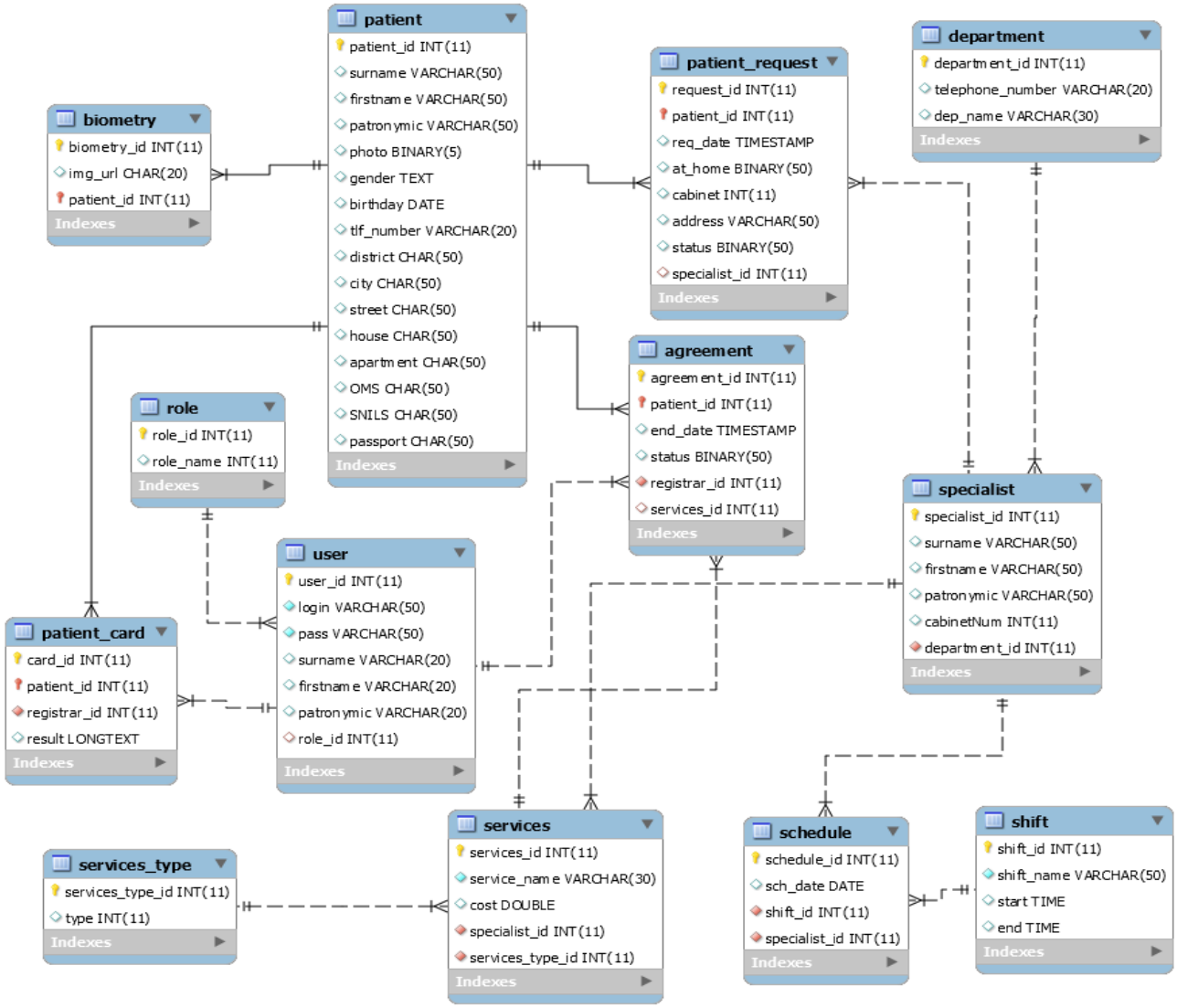

Рисунок 3.2 – Физическая модель базы данных

Физическая модель, представленная на рисунке 3.2 включает в себя 13 сущностей: «patient», «patient_request», «agreement», «biometry», «department», «patient_card», «user», «role», «schedule», «services», «services_type», «shift», «specialist».

Более подробная информация об структуре сущностей приведена в сущностях на рисунке 3.2.

3.5 Разработка программного обеспечения моделирования медицинской информационной системы

3.5.1 Схема взаимосвязи модулей приложения моделирования медицинской информационной системы

С учетом выбранной архитектуры программного обеспечения и разработанных физических и логических диаграмм, было принято решения разбить функционал на три модуля: клиентский модуль, серверный модуль и модуль для взаимодействия со сторонними приложениями медицинского учреждения.

Схема взаимосвязи модулей приведена на рисунке 3.3.

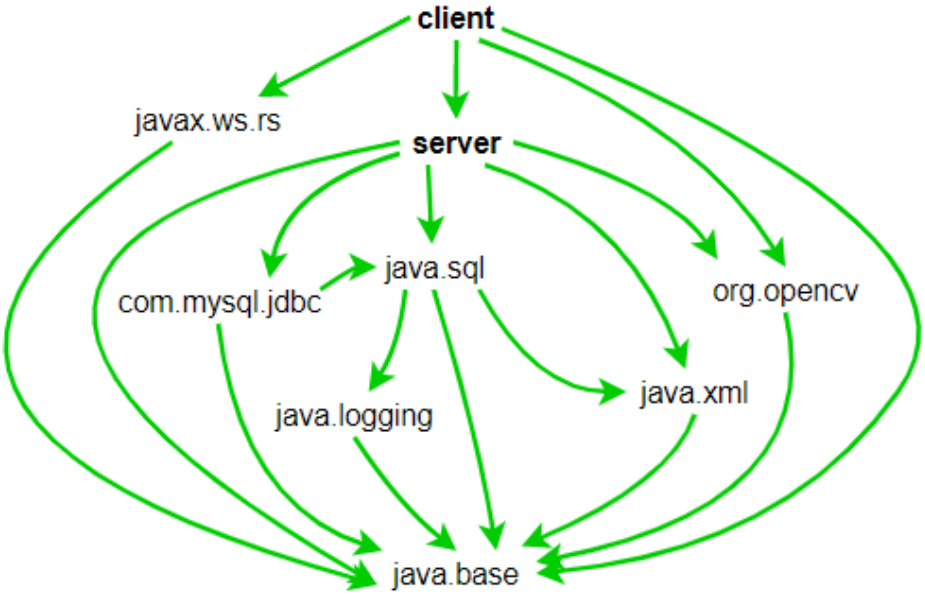

Рисунок 3.3 – Схема модулей программного обеспечения

Как видно из рисунка 3.3, кроме двух пользовательских модулей, выделенных жирным шрифтом, на диаграмме отображены дополнительные модули, предоставляемые платформой Java SE, которые несут целью показать более полно и развернуто модульную зависимость приложения. Серверный модуль, как и свойственно серверной части, является зависимостью «client». В свою очередь сервер зависит от модулей платформы Java SE для обеспечения работы по сети, связи с базой данных и т.д. Не только сервер, но и остальные

компоненты так или иначе связаны с модулями платформы, так как это необходимое условие запуска программ в JRE (Java Runtime Environment) [1].

3.5.2 Описание модулей медицинской информационной системы с примерами программного кода

В разделе 3.5.1 была разработана и изображена диаграмма модулей, которые были реализованы и использованы в системе.

Разработке клиентского модуля был использован классический MVC паттерн, который несет целью разделить пользовательский интерфейс и управляющую логику на три части, таким образом, что модификация каждой из частей осуществляется раздельно.

При запуске клиентского приложения пользователь системы должен предоставить свои авторотационные данные в специальную форму и, если логин и хэш пароля совпадут, пользователь получит доступ к основному окну приложения.

После получения персонального токена от сервера, перед пользователем открывается основное окно работы с системой.

Инициализация начинается с создания специального объекта встроенного класса «FXMLLoader», который при помощи FXML документа разметки главной панели, загружает иерархию объектов переданного в метод «createPaneFromFXML()» пользовательского класса «Utils» FXML документа, а затем возвращает объект типа «Pane», который в свою очередь является главным элементом в окне всего приложения, от которого осуществляется дальнейшее позиционирование. Далее в объект обобщающего интерфейса «Controller» записывается класс-контроллер, путь на который указан в FXML документе. В конструктор класса-контроллера передается ссылка на экземпляр класса «Main». И в завершении метода инициализации главной панели происходит установка сцены в JavaFX Stage класс, являющийся верхним уровнем JavaFX контейнера, экземпляром которого является созданный ранее

объект «primaryStage». Так же локальный экземпляр класса «Pane» записывается в созданный ранее «rootPane».

Инициализация окна авторизации, начинается до инициализации основной области. В отличие от методов инициализации основной панели авторизация имеет возвращаемое значение логического типа, в зависимости от которого программой будет произведена или не произведена дальнейшая работа по инициализации основного окна. Поскольку за работу данного окна отвечает другая сцена, которая работает независимо от основного окна, то ее создание происходит в специальном методе «createModalWindow()» класса «Utils», параметрами которой являются название и созданная панель. Функция «createModalWindow()» возвращает созданную сцену с уже помещенной в нее панелью авторизации, за которую отвечает класс-контроллер указанный в соответствующем FXML. Далее сцена при помощи функции «showAndWait()» помещается в режим ожидания закрытия встроенной функцией «close()», которая срабатывает при успешном чтении данных пользователя из базы данных по логину и паролю в «LoginWindowController» (см. Приложение Г).

Пример кода создания окна авторизации приведен на рисунке 3.4.

```
private boolean createLoginWindow() {
    FXMLLoader loader = new FXMLLoader();
    Pane pane = Utils.createPaneFromFXML(loader, VIEW_LOGIN_WINDOW_FXML);
    Stage dialogStage = Utils.createModalWindow(Constants.LOGIN, pane);
    LoginWindowController controller = loader.getController();
    controller.setParameters( main: this, dialogStage);
    loginWindowController = controller;
    dialogStage.showAndWait();

    return controller.isOkClicked();
}
```

Рисунок 3.4 – Пример кода создания окна авторизации

На основном окне располагается ряд функциональных вкладок, хранящих информацию о дополнительной функциональности в системе. Каждая вкладка имеет свой функционал и способна переключатся при помощи горячих клавиш. JavaFX вкладки имеют тип «Tab» и располагаются в едином контейнере «TabPane». На стороне клиентского приложения информация из списков не

хранится, списки только отображают информацию, получаемую с сервера. Приложение является многопоточным, так что данные актуализируются при изменении из других потоков. Сами списки обновляются самостоятельно путем привязывания специального типа «ObservableList», от разработчика требуется указать ссылку на текстовый тип «SimpleStringProperty» лежащую в соответствующей сущности.

Пример кода привязки списка к платформе приведен на рисунке 3.5.

```
@FXML
private void initialize() {
    patientTable.setItems(patientObservableList);
    patientIdColumn.setCellValueFactory(cellData -> cellData.getValue().getPatientIdProperty());
    surnameColumn.setCellValueFactory(cellData -> cellData.getValue().getSurnameProperty());
    firstnameColumn.setCellValueFactory(cellData -> cellData.getValue().getFirstnameProperty());
    patronymicColumn.setCellValueFactory(cellData -> cellData.getValue().getPatronymicProperty());
    photoColumn.setCellValueFactory(cellData -> cellData.getValue().getPhotoProperty());
    genderColumn.setCellValueFactory(cellData -> cellData.getValue().getGenderProperty());
    birthdayColumn.setCellValueFactory(cellData -> cellData.getValue().getBirthdayProperty());
    tlfColumn.setCellValueFactory(cellData -> cellData.getValue().getTlfProperty());
    snilsColumn.setCellValueFactory(cellData -> cellData.getValue().getSNILSProperty());
    omsColumn.setCellValueFactory(cellData -> cellData.getValue().getOMSProperty());
}
```

Рисунок 3.5 – Код привязки списка к платформе JavaFX

На рисунке 3.5 демонстрируется пример списка из вкладки «Регистр обслуживаемого населения». После данных манипуляций достаточно передавать данные типа «List<Patient>» в «patientObservableList», дальнейшие действия по обновлению отображения платформа выполнит сама.

Стоит заметить, что если не распечатывать талон, то данные сохранятся локально на компьютер клиента в формате XML при помощи выбранной в разделе 3.2 технологи JAXB, реализация которой приведена в Приложении Д.

Клиентское приложение разработано таким образом, что оно манипулирует только видеопотоком OpenCV, передаваемым на сервер. На сервере же реализуется основная логика по распознавания получаемого изображения.

В самом языке программирования Java очень мало возможностей для написания программ работающих в сети. Однако был использован JAX-RS,

позволяющий отправлять запросы и получать ответы. Для разработки потребовалось импортировать пакеты «javax.ws.rs». Клиентский API JAX-RS предоставляет высокоуровневый API для доступа к любым REST ресурсам. Для доступа к REST ресурсам необходимо использовать экземпляр «javax.ws.rs.client.Client» и на его основе создать и вызвать запрос. Клиентский API спроектирован так, чтобы быть свободным, с вызовами методов, связанными вместе для настройки и отправки запроса к ресурсу REST всего в нескольких строках кода. Пример такого запроса приведен на рисунке 3.6.

```
Response response = client.target( s: "http://localhost:8080/patients").
        request(MediaType.APPLICATION_JSON).
        get();

String json = response.readEntity(String.class);
Patient[] patients = new Patient[0];
try {
    patients = new ObjectMapper()
            .readerFor(Patient[].class)
            .readValue(json);
} catch (IOException e) {
    e.printStackTrace();
}
patientObservableList.addAll(patients);
```

Рисунок 3.6 – Пример клиентского запроса

В этом примере экземпляр клиента сначала создается путем вызова «javax.ws.rs.client.ClientBuilder.newClient» метода. Затем запрос конфигурируется и вызывается объединением вызовов методов в одну строку кода. «Client.target» метод устанавливает цель, основанную на URI. «javax.ws.rs.client.WebTarget.request» метод устанавливает тип носителя для возвращаемого объекта. «javax.ws.rs.client.Invocation.Builder.get» метод вызывает службу, используя HTTP – GET запрос, установив тип возвращаемого объекта как String. Так как приложения не хранит состояние, со стороны пользовательского приложения необходимо отправлять в шапке запроса сгенерированный токен, по которому сервер будет решать, предоставлять ли пользователю доступ к тем или иным ресурсам.

В настоящей программе было принято решение не выносить конечные запросы в отдельный класс конечных точек. Описанные выше и другие клиентские REST-запросы располагаются каждый в своем соответствующем JavaFX котроллере.

Принцип работы серверной части отличается от клиентской тем, что сервер реализует основную часть логики, по обработке данных, предоставляемым клиентом.

Серверный модуль выполняет различные манипуляции, которые несут целью удаление, изменение, добавление и обработку данных полученных из БД. Подключение к клиентам и обработка запросов и ответов осуществляется аналогичным образом в классе «EndPoint».

Главная идея серверного модуля – это отделить клиентскую реализацию от прямого соединения с базой данных. При помощи фреймворка SpringMVC были реализованы методы по обработке получаемых REST-запросов. Spring позволяет быстро и удобно на основе специальных аннотаций сопоставить приходящий запрос к определенному методу его обработки, примером такой аннотации является @MappingRequest, предоставляющий возможность инициализировать различные параметры, такие как URI запроса, его тип или тип данных.

Отдельно стоит описать работу с Spring Security, который обеспечивает гибкое разграничение доступа к тем или иным ресурсам. С клиентского JavaFX приложения отсылается запрос с содержащимся логином и паролем, на основе которого сервер генерируется специальный JWT токен, в который шифруется логин и роль юзера в системе. Реализация Spring Security в приложении базируется на следующих классах: «JwtTokenProvider» – компонент который производит манипуляции с токеном, «JwtConfigurer» – описывает необходимость прохождения каждого запроса через «JwtTokenFilter», «JwtTokenFilter» в свою очередь занимается предварительной обработкой запроса. Общей конфигурацией по предоставлению прав у тому или иному ресурсу занимается класс «SecurityConfig».

Сервер осуществляет взаимодействие с БД при помощи Java Database Connectivity (JDBC). JDBC – платформенно-независимый промышленный стандарт взаимодействия Java-приложений с различными СУБД, реализованный в виде пакета «java.sql», входящего в состав Java Standard Edition.

JDBC основан на концепции так называемых драйверов, позволяющих получать соединение с базой данных по специально описанному URL. Драйверы могут загружаться динамически. Загрузившись, драйвер сам регистрирует себя и вызывается автоматически, когда программа требует URL, содержащий протокол, за который драйвер отвечает.

При помощи класса «DriverManager» осуществляется неявная загрузка драйвера для подключения к СУБД MySQL и далее при помощи метода «getConnection()» в который передаются URL, логин и пароль базы данных, JDBC выполняет подключение и заносит его в переменную «con».

Пример взаимодействия с базой данных изображен на рисунках 3.7 и 3.8.

```
public User findUserByLoginAndPass(String login, String password) {
    User user =  new User();
    try (final PreparedStatement statement = this.con.prepareStatement (
            sql: "select * from registrar r where r.login = (?) and r.pass = (?)")) {
        statement.setString( parameterIndex: 1, login);
        statement.setString( parameterIndex: 2, password);

        try (final ResultSet rs = statement.executeQuery()) {
            if (rs.next()) {
                user.setId(rs.getString( columnLabel: "registrar_id"));
                user.setUserFirstName(rs.getString( columnLabel: "firstname"));
                user.setUserMiddleName(rs.getString( columnLabel: "patronymic"));
                user.setUserLastName(rs.getString( columnLabel: "surname"));
            } else {
                user = null;
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return user;
}
```

Рисунок 3.7 – Метод findUserByLoginAndPass()

А также пример более сложного запроса приведен на рисунке 3.8.

```
public List<Specialist> findSpecialists(String date, String department) {
    List<Specialist> patientList = new ArrayList<>();
    try (final PreparedStatement statement = this.con.prepareStatement(
            sql: "select specialist_id, surname, firstname, patronymic, shift_name, d.dep_name, sch_date " +
                    "from (select specialist_id, surname, firstname, patronymic, shift_name, department_id, sch_date " +
                        "from (select sc.specialist_id, surname, firstname, patronymic, shift_id, sp.department_id, sch_date  " +
                            "from schedule sc " +
                            "left join specialist sp on sp.specialist_id = sc.specialist_id and sch_date like (?)) cust " +
                        "left join shift sh on sh.shift_id = cust.shift_id) cust2 " +
                    "left join department d on d.department_id = cust2.department_id " +
                    "where d.dep_name like (?)")) {
        statement.setString( parameterIndex: 1, date.equals("") ? "%" : date);
        statement.setString( parameterIndex: 2, department.equals("") ? "%" : department);

        try (final ResultSet rs = statement.executeQuery()) {
            while (rs.next()) {
                patientList.add(new Specialist(
                                rs.getInt( columnLabel: "specialist_id"),
                                rs.getString( columnLabel: "surname"),
                                rs.getString( columnLabel: "firstname"),
                                rs.getString( columnLabel: "patronymic"),
                                rs.getString( columnLabel: "dep_name"),
                                rs.getString( columnLabel: "shift_name"),
                                rs.getString( columnLabel: "sch_date")
                        )
                );
            }
        }
    } catch (SQLException e) {
        e.printStackTrace();
    }
    return patientList;
}
```

Рисунок 3.8 – Метод findSpecialist()

На приведенных выше рисунках 3.7 и 3.8 коннектор используется встроенный метод «PreparedStatement()», который отправляет sql-запрос с переданными ему аргументами. Далее в метод поступает ответ от базы данных, который содержит данные местоположения и личную информацию одного конкретного пользователя. Метод «next()» класса «ResultSet» вызывает построчно ответ от базы данных и вносит их в указанные поля. В данном конкретном примере все элементы имеют строковый тип, в следствие чего используется метод «getString()», который преобразует полученные строки в тип «String». После успешной записи данных в сущность типа «Registrar» метод findUserByLoginAndPass() возвращает объект в функцию, где метод был вызван. Аналогично работает и метод с рисунка 3.12. Стоит отметить, что все подобные методы были добавлены в один класс-модель «JDBC», что позволяет контролировать запросы в БД с одного места в приложении.

Система биометрического распознавания была вынесена на серверную часть. Данная операция была проделана для того, чтобы не хранить ка каждой клиентской машине копию баз фотографий для инициализации алгоритма. На данный момент биометрический поиск реализован таким образом, что имя фотографии хранит в себе уникальный идентификатор и номер фотографии. При нахождении лица он определяет его по данным полученным в результате инициализации и записывает с специальную переменную идентификатор текущего пациента в видеопотоке. С клиента приходит запрос на получение данных о текущем пациенте и при помощи запроса к базе данных находит изображенного пациента и возвращает результат в список на клиенте, в случае неудачи на клиента придет пустой список – это будет значить, что биометрических данных пациента в системе нет. Пример с алгоритмом распознавания приведен в приложении А.

Работа с библиотекой написанной на языке C++ стала возможной путем подключения к проекту специально разработанного модуля для Java и скомпилированного файла с расширением dll, что позволило в полной мере воспользоваться предоставляемым OpenCV API [24].

3.6 Описание функциональности моделирования медицинской информационной системы

При старте медицинской информационной системы перед пользователем появляется окно авторизации, приведённое на рисунке

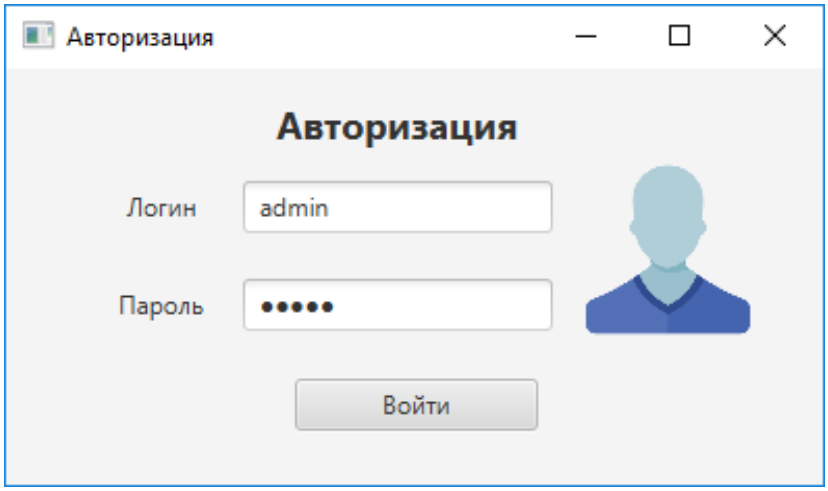

Рисунок 3.9 – Окно авторизации

Пользователю необходимо ввести своим данные и после проверки ему представятся права на взаимодействие с системой и откроется главное окно приложения.

Главное окно приложения представляет из себя многофункциональный интерфейс сделанный в серо-синих тонах. Приложение работает по принципу использования вкладок, каждая из которых хранит свою уникальную информацию. На вкладке «Регистр обслуживаемого населения» регистратору доступны различные манипуляции с клиентами медицинского учреждения. Например, кнопка «Посмотреть» открывает возможность использования дополнительного функционала, вынесенного в отдельное окно (рисунок 3.10).

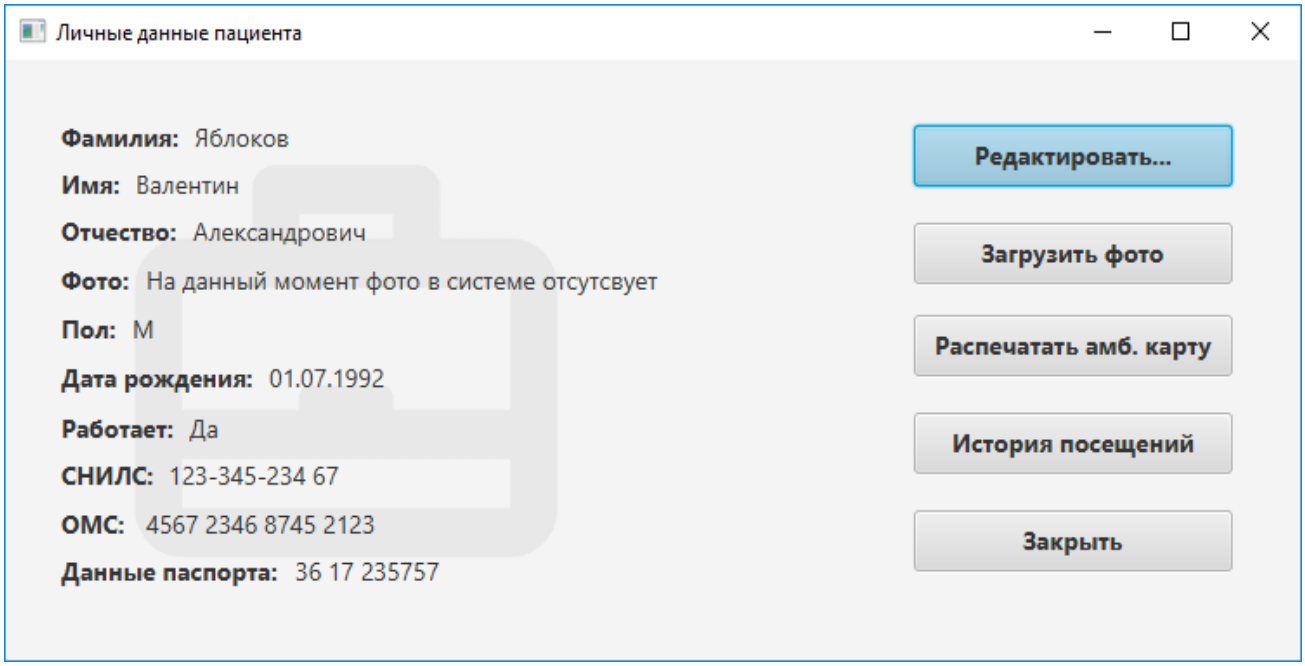

Рисунок 3.10 – Окно личных данных пациента

Как видно из рисунка 3.10 в окне личных данных пациента возможно совершать такие действия как редактирование данные, просмотр фотографий, загруженных в базу, распечатка амбулаторной карты при первом посещении, а также есть возможность посмотреть историю всех осуществленных посещений в медицинском учреждении.

Для более детальной демонстрации функционала главного окна был приведен рисунок 3.11.

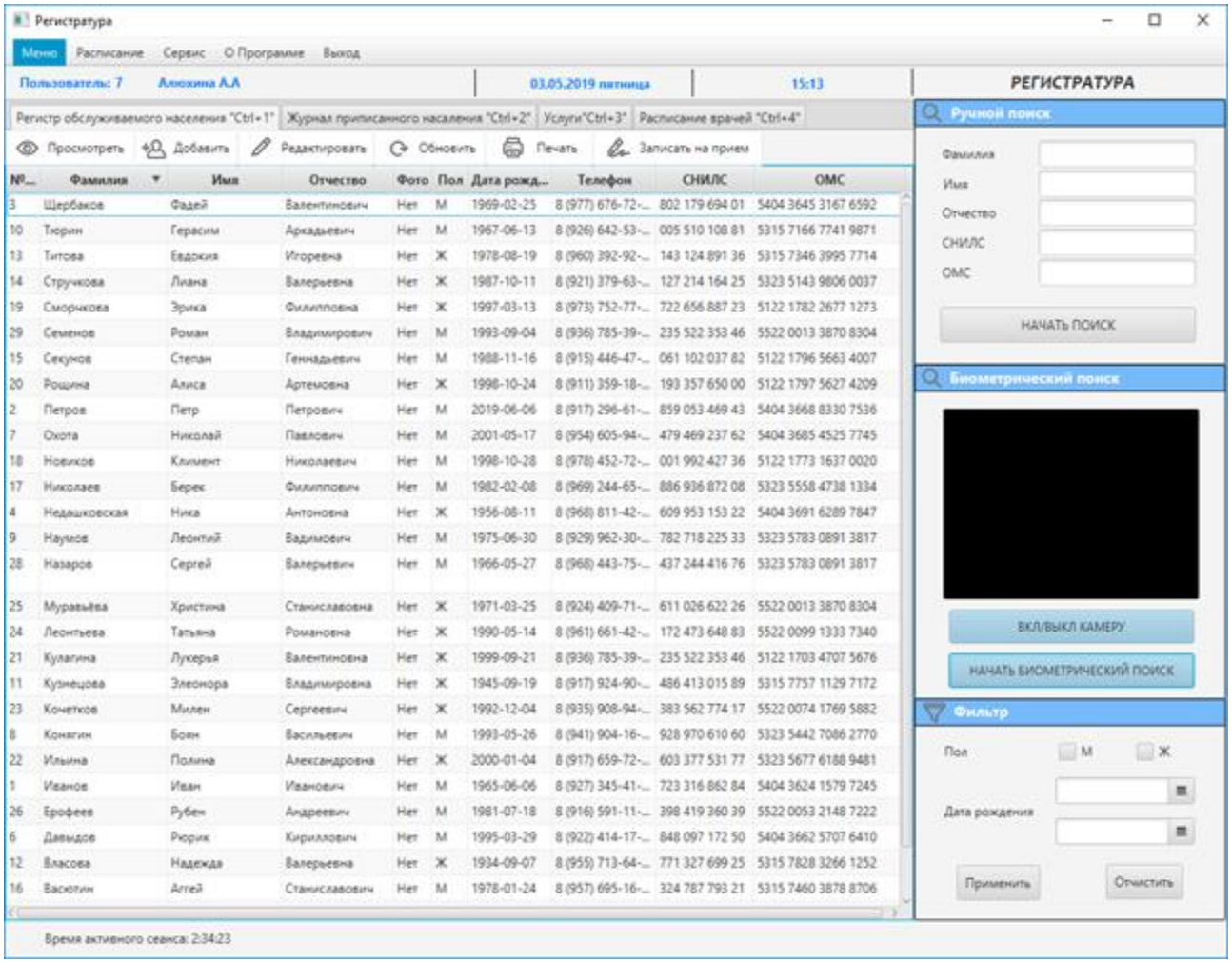

Рисунок 3.11 – Главное окно приложения

Из рисунка 3.11 видно, что справа расположено три части меню: «Ручной поиск», который дает возможность искать пациентов при неимении биометрических данных в системе, «Биометрический поиск», который содержит экран для отображения видеопотока с распознанным уникальным идентификатором пациента, найденном на сервере, регистратор имеет возможность управлять потоком, а именно включать и выключать передачу данных, также доступны дополнительные параметры фильтрации, которые можно применить по желанию.

Системой предусмотрено, что при переключении неиспользуемые поля становятся недоступными (рисунок 3.12).

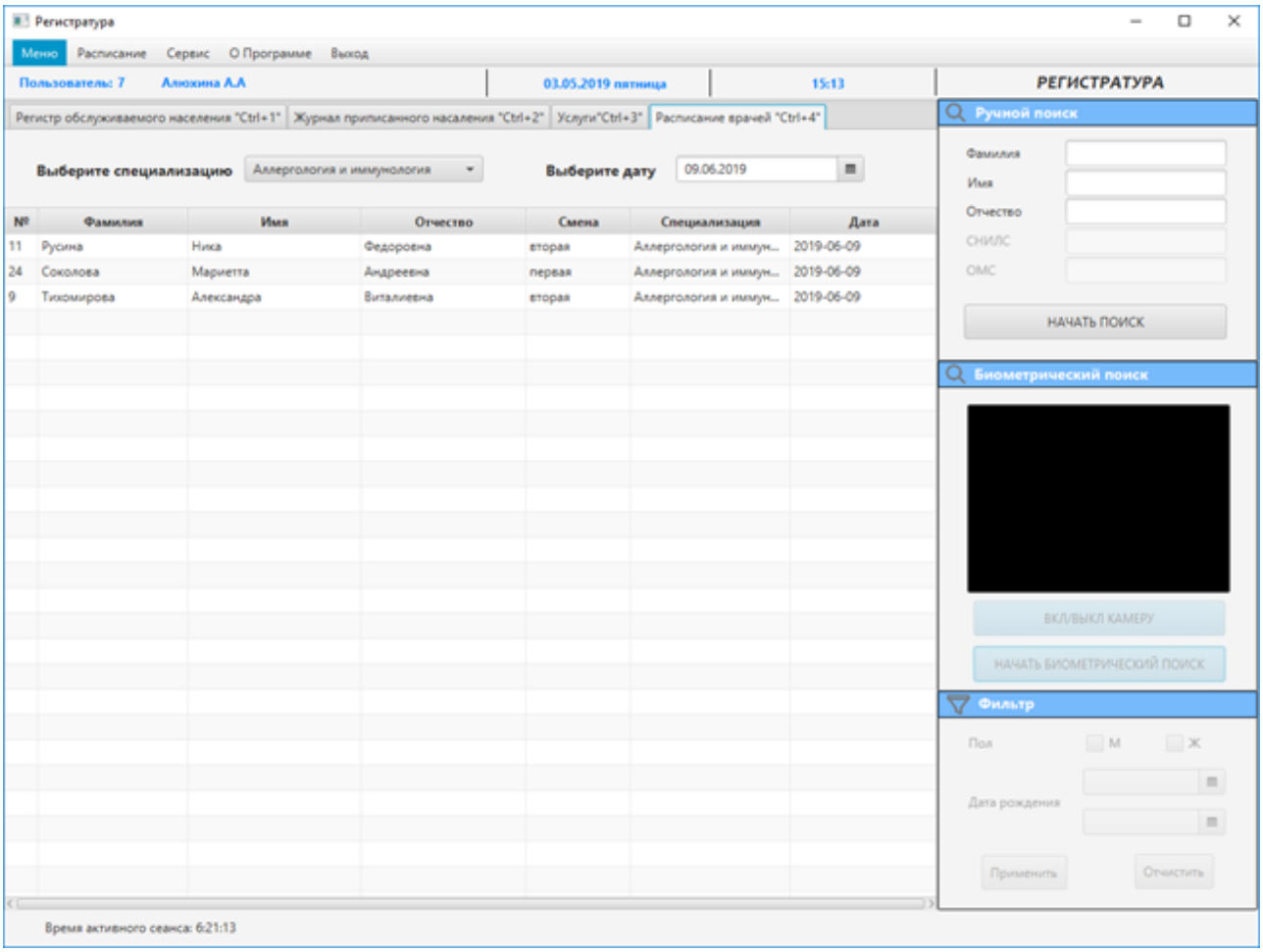

Рисунок 3.12 – Вкладка «Раписание врачей»

Так же из рисунка 3.12 видно, что в информационной системе предусмотрена возможность удобного просмотра раписания врачей по датам и специализациям.

Для дополнительного удобства система позволяет просматривать журнал приписанного населения по районам и улицам и просматривать список предоставляемых медицинским учреждением услуг. Предоставляется возможность составить и сохранить договор на оказание платных медицинских услуг. Предусмотрена возможность заполнения заявок пациентов, как на месте, так и по телефону. Для второго варианта была сделана возможность отложного

распечатывания талона на прием, т.е. данные хранятся локально в файле до востребования.

Когда пациент впервые обращается в медицинское учреждение, ему необходимо выдать специальную амбулаторную карту, в которой будут хранится физические данные о результатах приема (см. Приложение Б), также данные хранятся в электронном виде в БД. Доступ к данному функционалу отражен на рисунке 3.10.

Как было сказано ранее, талоны на прием к специалисту в системе не хранятся, их временным хранением занимается локальная машина, следовательно, чтобы воспользоваться данными об истории посещений пациента необходимо зайти в личные данные пользователя (рисунок 3.10) и открыть соотвествующее окно. Сделано так для того, чтобы хранить не только записи о состоявшихся приемах, но так же и о приемах, от которых пациет отказался.

Окно вывода талона на печать приведено на рисунке 3.13

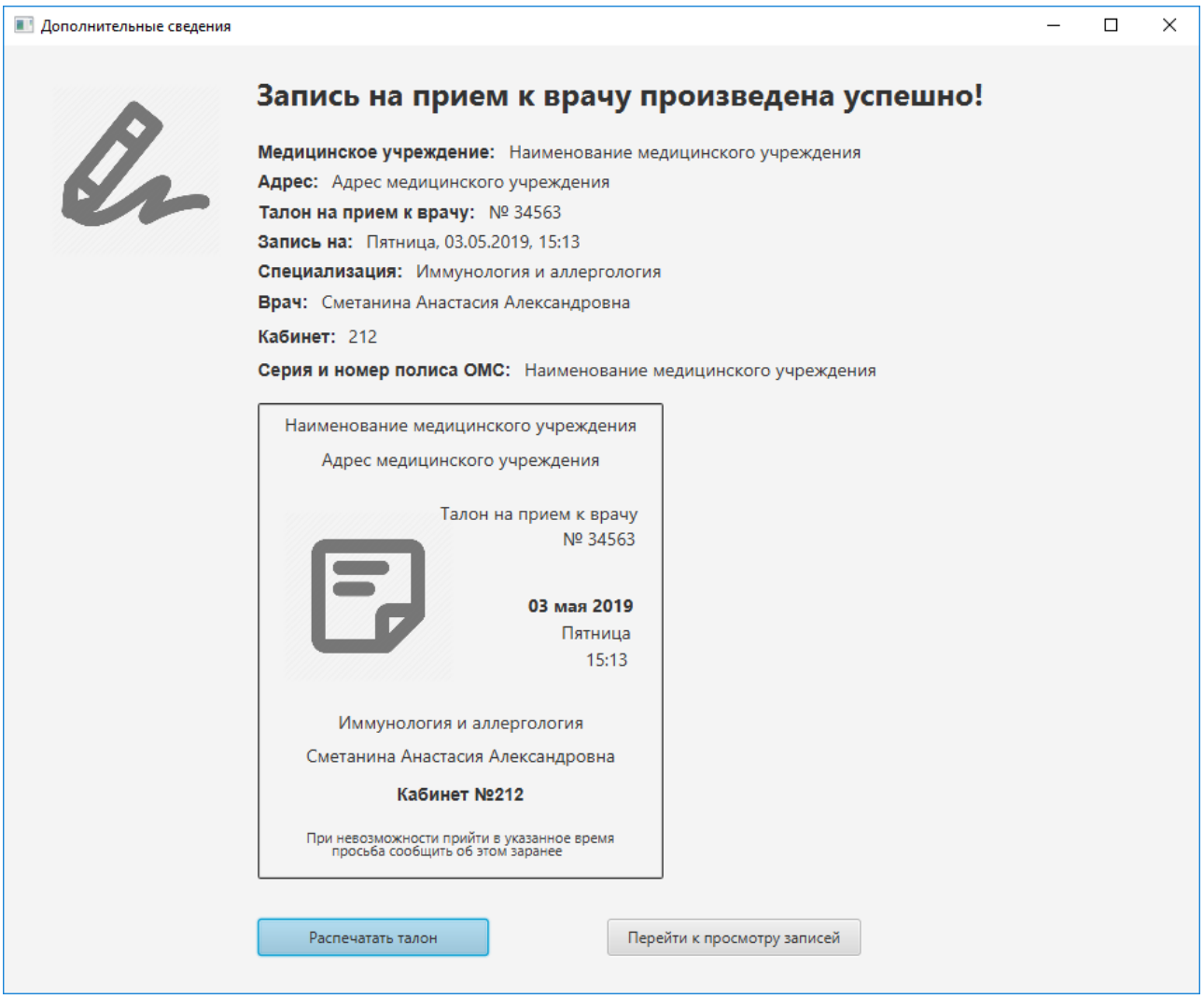

Рисунок 3.13 – Вывод талона на печать

В качестве средств дополнительного информирования пациентов, на главный экран регистратуры выводится форма с раписанием врачей по специализациям на текущую дату. Пример данной формы приведен в Приложении В.

Данный представленный функционал является достаточным для ускорения функционирования регистратур, а также для долговременного хранения и оперативного доступа к информации.

3.7 Тестирование программного проекта

Для тестирования программной системы был использован фреймворк Hamcrest.

Hamcrest – это фреймворк для кодирования тестирования, позволяющий декларировать правила сравнения. Существует ряд ситуаций, в которых сравнение объектов не является возможным, например, проверка пользовательского интерфейса или фильтрация данных, но именно в области написания юнит-тестов сопоставления наиболее часто используются. При написании тестов иногда бывает трудно установить правильный баланс между чрезмерной спецификацией теста (и сделать его хрупким к изменениям) и недостаточной спецификацией (делая тест менее ценным, поскольку он продолжает проходить, даже если тестируемая вещь сломана). Наличие инструмента, который позволяет точно выбрать тестируемый аспект и описать значения, которые он должен иметь, с достаточным уровнем точности, очень помогает в написании «passed» тестов. Такие тесты терпят неудачу, когда поведение тестируемого аспекта отклоняется от ожидаемого поведения, но продолжают проходить, когда в поведение вносятся незначительные, не связанные изменения [23].

Данный проект основан на работе программного обеспечения с базой данных. Все добавления, изменения, удаления какой-либо информации из программного обеспечения влекут за собой и удаление ее из БД. Поэтому нужно быть уверенными в правильно написанных запросах в БД для корректной работы пользовательской информацией. Следовательно, необходимо полностью покрыть тестами класс, использующийся для работы с данными.

Пример теста приведен на рисунке 3.14.

```
@Test
public void check_patient_equals() throws SQLException {
    jdbc.connect();
    Patient patient = new Patient( patientId: 3,  surname: "Щербаков", firstname: "Фадей", patronymic: "Валентинович",
            photo: false, gender: "М", birthday: "1969-02-25",  tlf_number: "8 (977) 676-72-11",
            district: "Комсомольский",  city: "Тольятти", street: "Дачная", house: "22", apartment: "2",
            oms: "5404 3645 3167 6592", snils: "802 179 694 01", passport: "1"
    );
    Patient patient1 = jdbc.findPatients( patient_id: "3", surname: "", firstname: "", patronymic: "",
            snils: "", oms: "", gender: "", start: "", end: "").get(0);
    assertThat(patient, equalTo(patient1));
}
```

Рисунок 3.14 – Пример теста check_patient_equals()

Из рисунка 3.14 видно, что тест несет целью проверить корректность работы возвращаемой из БД сущности, путем сравнение заведомо известного пациента с возвращаемым значением метода «findPatients()» класса «JDBC» при помощи метода «assertThat()».

Таблица 3.2 – Результаты юнит-тестирования

Класс	Метод	Результат
JDBC	addPatient()	Тест пройден
	deletePatient()	Тест пройден
	findPatients()	Тест пройден
	addBioUrl()	Тест пройден
	findUserByLoginAndPass()	Тест пройден
	addSpecialist()	Тест пройден
	deleteSpecialist()	Тест пройден
	findSpecialists()	Тест пройден
	findServices()	Тест пройден
	addService()	Тест пройден
	addService()	Тест пройден
	addAgreement()	Тест пройден
	deleteAgreement()	Тест пройден
	findAgreement()	Тест пройден
	addPatientCard()	Тест пройден
	findPatientCard()	Тест пройден
		100%
Utils	md5Hashing()	Тест пройден
	loadDataFromFile()	Тест пройден
	saveDataToFile()	Тест пройден
		100%

Так же дополнительно было произведено дымовое тестирование (smoke testing), где в ручном режиме было проверено соединение с базой данных, авторизация и элементы графического интерфейса пользователя.

Выводы по разделу 3

В данном разделе был произведен выбор архитектуры МИС. Выбрана технология разработки и вспомогательные сторонние средства. По спроектированной в разделе 2.3.3 логической диаграмме и результатам анализа была выбрана СУБД MySQL и разработана структура базы данных, а также ее физическое отображение.

Далее была произведена работа по созданию самой системы, успешно внедрено биометрическое распознавание лица. Были графически отображены модули, а именно построена диаграмма зависимостей модулей друг от друга. Согласно диаграмме, были описаны существующие модули с примерами программного кода, и также приведены скриншоты форм разработанного программного продукта. В заключении были описаны методы тестирования и приведены их результаты.

ЗАКЛЮЧЕНИЕ

В рамках выпускной квалификационной работы было произведено проектирование и разработка медицинской информационной системы для регистратур медицинских учреждений при помощи средств Java SE и сторонних продуктов.

Разработанная система предоставляет возможность дополнительного биометрического поиска пациентов в базе, а также надежного хранения и оперативного получения информации из базы данных.

Для достижения цели были решены следующие задачи:

- изучена предметная область;

- осуществлен анализ существующих на рынке решений;
- успешно внедрен биометрический поиск при помощи библиотеки OpenCV;
- выбрана технология для проектирования информационной системы;
- обоснован выбор архитектуры информационной системы;
- выбран ряд технологий для быстрой и качественной разработки;
- был произведен сравнительный анализ и на его основе выбрана наиболее удобная для использования СУБД;
- выполнено покрытие тестами.

Получившаяся медицинская информационная система способна увеличить скорость работы регистратур за счет внедренной технологии биометрического поиска пациентов и продуманной системы обмена данными. В дальнейшем планируется усовершенствование разработанной системы, горизонтальное расширение ее функционала и добавление новых локализаций.

Автором работы был приобретен ценный опыт проектирования и разработки информационных систем, который будет использован в дальнейшей профессиональной деятельности.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

Научная и методическая литература

1. Сеттер, Р.В. Изучаем Java на примерах и задачах – Наука и техника, 2016. – 240 с.

2. Вязовик, Н. А. Программирование на Java, Интуит 2016. – 600 с.

3. Александров, Д.В. Инструментальные средства информационного менеджмента. CASE-технологии и распределенные информационные системы / Д.В. Александров. - М.: Финансы и статистика, 2011. – 224 с.

4. Джошуа Блох. Java. Эффективное программирование: Лори, 2014. – 440с.

5. Васильков, А.В. Информационные системы и их безопасность: Учебное пособие / А.В. Васильков, А.А. Васильков, И.А. Васильков. - М.: Форум, 2013. – 528 с. 17.

6. Коваленко, В.В. Проектирование информационных систем. - М.: Форум, 2012. - 320с.

7. Филиппов, В. А. Многомерные СУБД при создании корпоративных информационных систем / В.А. Филиппов. - М.: Едиториал УРСС, 2014. – 465 с.

8. Харрингтон Дж. Совершенство управления процессами. – М. Стандарты и качество, 2007. – 192 с.

9. Кузнецов С.Д. Введение в модель данных SQL (2-е изд.): М.: НОУ "Интуит" , 2015. – 350 с.

10. Исаев, Г.Н. Проектирование информационных систем. Учебное пособие. - М.: Омега-Л, 2015. - 432с.

11. Карвин, Б. Программирование баз данных SQL. Типичные ошибки и их устранение / Б. Карвин. – М.: Рид Групп, 2012. – 336с. – (Профессиональные компьютерные книги).

12. Репин В. В., Бизнес-процессы. Моделирование, внедрение, управление. – М.: Манн, Иванов и Фербер, 2013. – 512 с.

13. Тарасов, С.В. СУБД для программиста. Базы данных изнутри: Солон-Пресс , 2015. – 320 с.

14. Шелухин, О. И. Моделирование информационных систем / О.И. Шелухин, А.М. Тенякшев, А.В. Осин. - М.: Радиотехника, 2011. – 368 с.

15. Машнин, Т.С. JavaFX 2.0: разработка RIA-приложений. - СПб.: БХВПетербург, 2012. - 320 с.: ил. - (Профессиональное программирование).

Электронные ресурсы

16. Конфигурация «MedWork-Регистратура». Режим доступа - http://www.medwork.ru/content/konfiguratsiya-medwork-registratura

17. Решение «МКТ-Современная регистратура». Режим доступа - http://medcomtech.ru/Products/SovReg/sovreg.html

18. Программный комплекс для автоматизации работы регистратуры ООО «Ситиай-софт» Режим доступа - http://www.ctisoft.ru/node/113

19. Диаграмма классов. Режим доступа - https://ru.wikipedia.org/wiki/Диаграмма_классов

20. Диаграмма последовательности. Режим доступа - https://ru.wikipedia.org/wiki/Диаграмма_последовательности

21. Диаграмма прецедентов. Режим доступа - https://ru.wikipedia.org/wiki/Диаграмма_прецедентов.

22. UML. Режим доступа - https://ru.wikipedia.org/wiki/UML

23. Hamcrest. Режим доступа - http://hamcrest.org/JavaHamcrest/

24. OpenCV Java documentation. Режим доступа - https://docs.opencv.org/master/javadoc/index.html

25. ER-Win Data Modeler. Режим доступа - https://ru.wikipedia.org/wiki/ERwin_Data_Modeler.

26. Java Database Connectivity. Режим доступа - https://ru.wikipedia.org/wiki/Java_Database_Connectivity

27. Работа регистратуры. Режим доступа - https://www.medkurs.ru/terap/amb/section3159/26194.html

28. FURPS+. Режим доступа - https://sysana.wordpress.com/2010/09/16/furps/

29. Распознавание изображений. Алгоритм Eigenface. Режим доступа - https://habr.com/ru/post/68870/

30. Обзор современных реляционных СУБД. Режим доступа - https://spravochnick.ru/bazy_dannyh/yazyk_sql_osnovy_raboty_s_relyacionnymi_subd_osnovy_yazyka_sql/obzor_sovremennyh_relyacionnyh_subd/

Литература на иностранном языке

31. Hevner, Alan, Chatterjee, Samir. Design Research in Information Systems - Theory and Practice, 2010.

32. Dhillon, G. (2007). Principles of Information Systems Security: Text and Cases. Hoboken, NJ: John Wiley & Sons. 73

33. Mandic, D, Lalic, N., Bandjur, V..: Computer Aided Research in Managing Educational Process, in the book 7th WSEAS International Conference on ENGINEERING EDUCATION (EDUCATION '10 Corfu, Greece, 2010.

34. Efrem, G. Mallach. Information Systems: What Every Business Student Needs to Know, p.458, 2015.

35. Ebbers, H. Mastering JavaFX 8 Controls, Oracle Publishing Group, 2014.

ПРИЛОЖЕНИЕ А

Листинг методов биометрического распознавания

```
private Mat detectAndDisplay(Mat frame) {
        MatOfRect faces = new MatOfRect();
        Mat grayFrame = new Mat();

        Imgproc.cvtColor(frame, grayFrame, Imgproc.COLOR_BGR2GRAY);
        Imgproc.equalizeHist(grayFrame, grayFrame);

        this.faceCascade.detectMultiScale(grayFrame, faces, 1.1, 2, 0 |
Objdetect.CASCADE_SCALE_IMAGE,
                new Size(this.absoluteFaceSize, this.absoluteFaceSize), new Size());
        Rect[] facesArray = faces.toArray();
        for (int i = 0; i < facesArray.length; i++) {
          Imgproc.rectangle(frame, facesArray[i].tl(), facesArray[i].br(), new
Scalar(0, 255, 0), 3);
            Rect rectCrop = new Rect(facesArray[i].tl(), facesArray[i].br());
            Mat croppedImage = new Mat(frame, rectCrop);
            Imgproc.cvtColor(croppedImage, croppedImage,
Imgproc.COLOR_BGR2GRAY);
            Imgproc.equalizeHist(croppedImage, croppedImage);
            Mat resizeImage = new Mat();
            Size size = new Size(250, 250);
            Imgproc.resize(croppedImage, resizeImage, size);
            double[] returnedResults = faceRecognition(resizeImage);
            double prediction = returnedResults[0];
            int label = (int) prediction;
            String name = "";
            if (names.containsKey(label)) {
              name = names.get(label);
            } else {
              name = "Unknown";
            }
            current = name;
            String box_text = name;
            double pos_x = Math.max(facesArray[i].tl().x - 10, 0);
            double pos_y = Math.max(facesArray[i].tl().y - 10, 0);
            Imgproc.putText(frame, box_text, new Point(pos_x, pos_y),
                Core.FONT_HERSHEY_TRIPLEX, 1.3, new Scalar(0, 255, 0,
2.0));      }
```

```
            return frame;
        }
}
```

ПРИЛОЖЕНИЕ Б

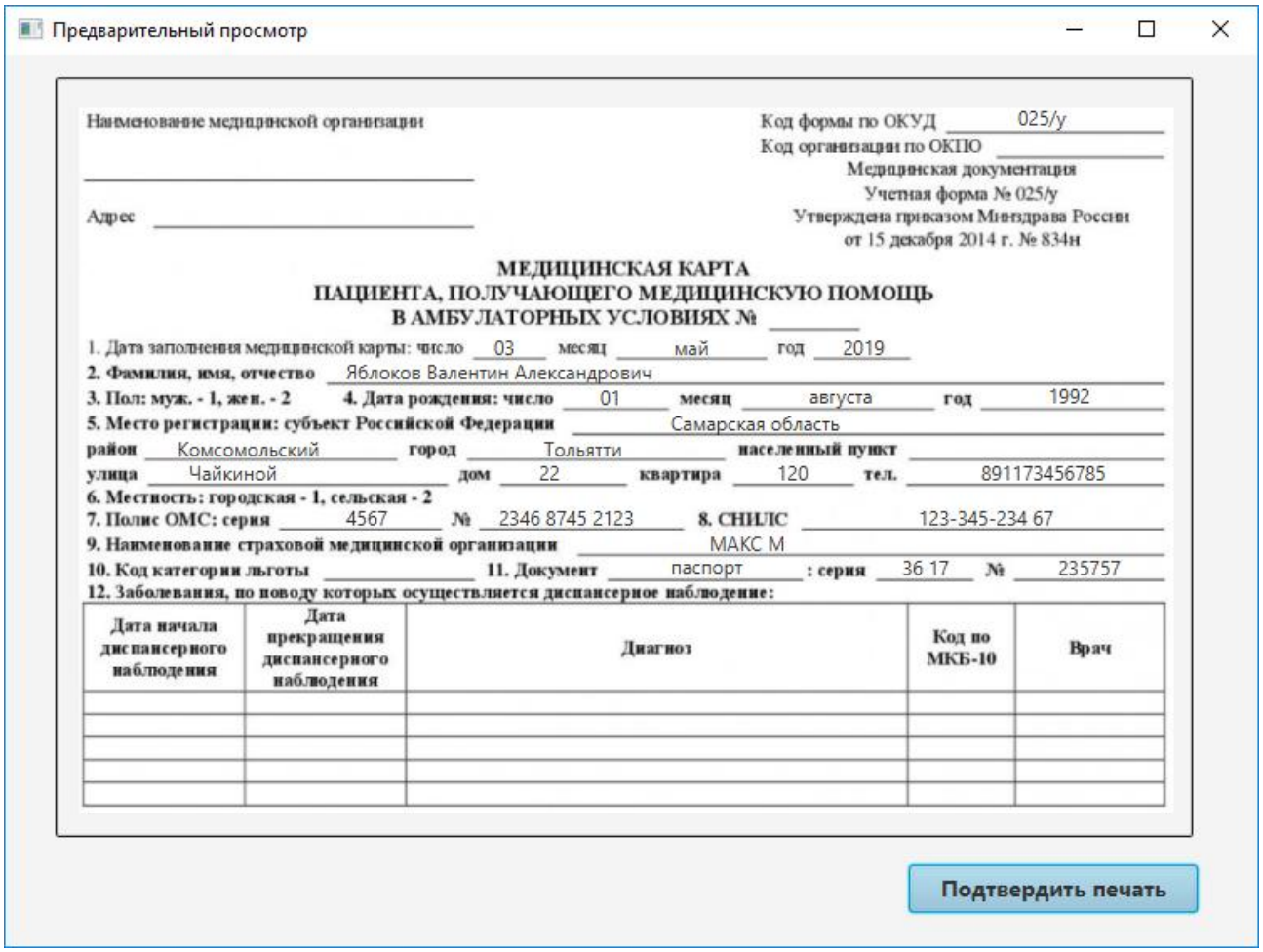

Рисунок Б.1 – Форма 025/у

ПРИЛОЖЕНИЕ В

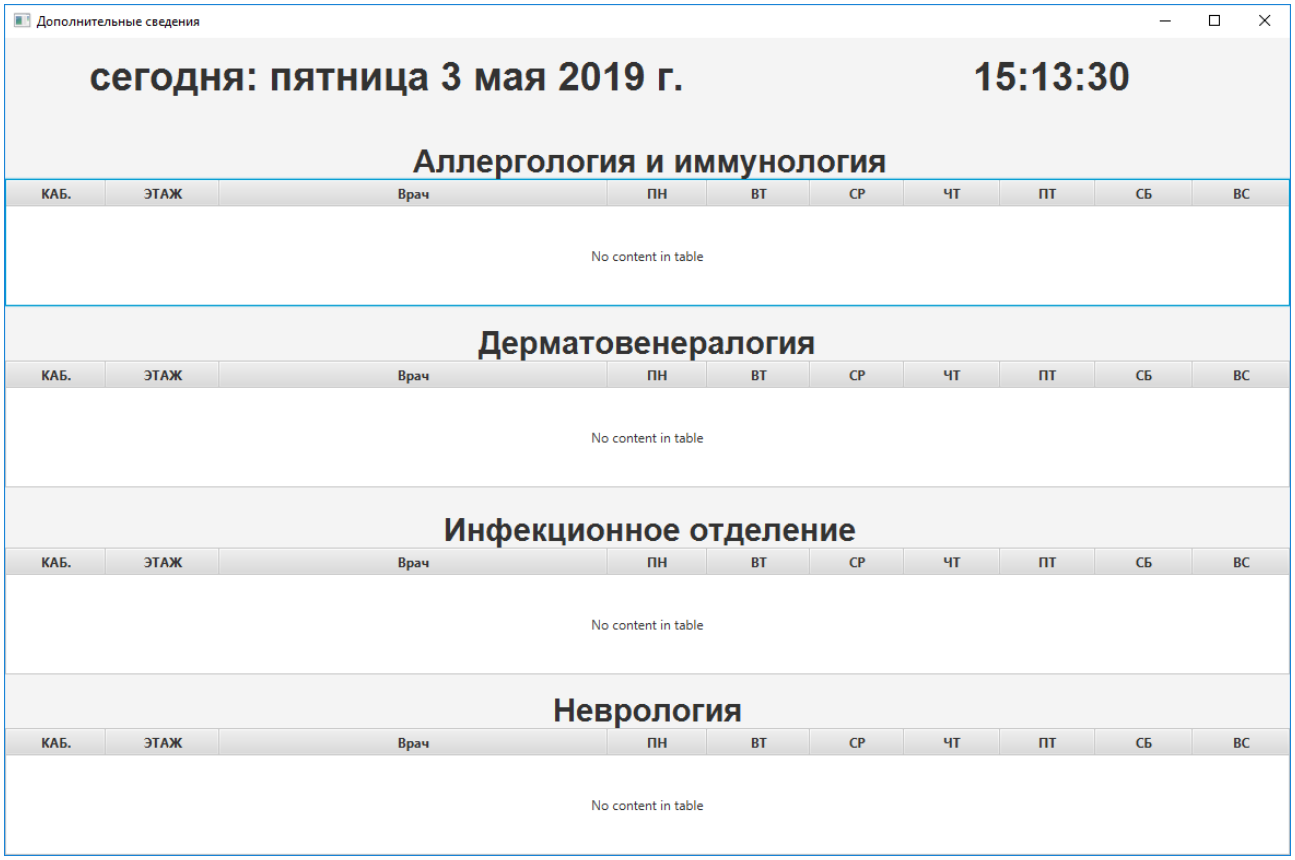

Рисунок В.1 – Форма, выводимая на общий экран регистратуры

ПРИЛОЖЕНИЕ Г

Листинг класса LoginWindowController

```
public class LoginWindowController implements Controller
{
   @FXML
   private TextField loginField;
   @FXML
   private PasswordField passField;
   @FXML
   private Label errorLabel;
   @FXML
   private Button okButton;

   private Stage dialogStage;
   private Main main;
   private boolean okClicked = false;

   @Override
   public void setParameters(Main main, Stage stage) {
      this.main = main;
      this.dialogStage = stage;
   }

   @Override
   public Stage getStage() {
      return null;
   }

   @FXML
   private void okButton() {
      okButton.setDisable(true);
      String[] auth = new String[2];
      auth[0] = loginField.getText();
      auth[1] = passField.getText();
      Entity<String[]> userEntity = Entity.entity(auth, "text/plain");
      Response response = client.target("http://localhost:8080/auth").
            request("text/plain").
            post(userEntity);

      User user = response.readEntity(User.class);
      authenticator(user);
   }
```

```
    private void authenticator(User user) {
        Platform.runLater(() -> {
            if (user != null) {
                main.getUser().setId(user.getId());
                main.getUser().setUserFirstName(user.getUserFirstName());
                main.getUser().setUserLastName(user.getUserLastName());
                main.getUser().setUserMiddleName(user.getUserMiddleName());
                okClicked = true;
                dialogStage.close();
            } else {
                errorLabel.setText("неверный логин/пароль");
            }
        });
    }



    public boolean isOkClicked() {
        return okClicked;
    }
}
```

ПРИЛОЖЕНИЕ Д

Листинг методов загрузки и сохранения локальных данных при помощи JAXB

```
private static void saveDataToFile(File file, ObservableList<TalonWrapper>
talonData)
{
   try
   {
      JAXBContext context = JAXBContext.newInstance(TaskListWrapper.class);
      Marshaller m = context.createMarshaller();
      m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, true);

      TalonWrapper wrapper = new TalonWrapper ();
      wrapper.setTalons(talonData);

      m.marshal(wrapper, file);
   }
   catch (Exception e)
   {
      TaskManagerException.createSavingException(file.getPath(), e);
   }
}

private static void loadDataFromFile(File file, ObservableList< TalonWrapper >
talonData)
{
   try
   {
      JAXBContext context = JAXBContext.newInstance(TalonWrapper.class);
      Unmarshaller um = context.createUnmarshaller();

      TalonWrapper wrapper = (TalonWrapper)um.unmarshal(file);

      talonData.clear();
      talonData.addAll(wrapper.getTalons());
   }
   catch (Exception e)
   {
      TaskManagerException.createLoadingException(file.getPath(), e);
   }
}
```