МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
вычислительной техники	«МОиПЭВМ»

Направление подготовки	09.04.04 – «Программная инженерия»
Магистерская программа	Программное обеспечение вычислительной техники и автоматизированных систем

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему

Модели и алгоритмы управления обслуживанием пациентов медицинского учреждения

Студент _____ Оплюшкина Олеся Олеговна
(подпись, дата) (ФИО полностью)

Руководитель _____ Балашова И.Ю.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Рецензент _____ Шеланков О.Е.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____№_____)

Заведующий кафедрой _____ Макарычев П.П.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____№____)

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2020

Содержание

Введение ... 3
1 Анализ моделей и алгоритмов обслуживания в медицинском учреждении . 8
1.1 Порядок приема пациента в медицинском учреждении ... 8
1.2 Анализ вспомогательных средств работы медицинского учреждения ... 12
1.3 Обзор готовых решений ... 14
1.4 Анализ поликлиники как системы массового обслуживания ... 16
1.5 Анализ алгоритмов формирования очередей в многоканальной системе массового обслуживания ... 20
Выводы ... 24
2 Модель управления потоками пациентов в очереди на обслуживание ... 26
2.1 Описание модели потоков пациентов в поликлинике ... 26
2.2 Алгоритм формирования потоков пациентов для приема у специалиста в медицинском учреждении ... 33
Выводы ... 39
3 Проектирование модуля расписания приема пациентов ... 41
3.1 Разработка средств управления расписанием приема пациентов ... 41
3.2 Проектирование базы данных ... 46
3.3 Построение логической модели базы данных ... 51
3.4 Построение физической модели базы данных ... 52
3.5 Разработка хранимых процедур, функций и триггеров базы данных ... 53
3.6 Проектирование программных средств ... 54
3.7 Тестирование программы ... 61
Выводы ... 63
Заключение ... 64
Список использованных источников ... 65
Приложение А. Глоссарий ... 68
Приложение Б. Скрипт создания базы данных ... 70
Приложение В. Скрипты хранимых процедур ... 74
Приложение Г. Тестирование программы ... 77

Введение

В настоящее время медицина является одной из самых важных и самых сложных отраслей человеческой деятельности, удовлетворение потребностей населения в качественной медицине является наиболее важной задачей российской политики в отрасли здравоохранения. Одной из целей здравоохранения является повышение эффективности работы государственных медицинских учреждений, не только за счет оптимизации лечебных и диагностических процессов, но и за счет совершенствования информационных технологий в области медицины и модификации процессов взаимодействия пациентов, сотрудников и специалистов медицинского учреждения.

На данный момент, наиболее важными направлениями реорганизации в области здравоохранения являются:

- повышение эффективности работы врачей;
- организация управления обслуживанием пациентов в лечебном учреждении;
- ускорение доступа к справочной информации медицинского учреждения;
- повышение уровня достоверности информации при формировании медицинской документации;
- оптимизация информационного взаимодействия между сотрудниками и специалистами медицинского учреждения.

Решение данных проблем обеспечит снижение количества очередей на прием к специалисту медицинского учреждения, снижение времени ожидания пациентов в очереди, увеличит показатели качества обслуживания пациентов, а также предоставит возможность быстрой и простой организации взаимоотношений пациента с медицинской организацией.

Современная медицина уже достигла значительных успехов в диагностике и лечении многих заболеваний, за счет использования более сложных инструментальных средств и методов. В то же время происходит

постепенная стандартизация многих процессов в области оказания медицинской помощи. Сложность медицины как области человеческой деятельности отражается в функционировании медицинских учреждений, а, следовательно, показывает проблематику разработки, внедрения и использования медицинских информационных систем.

В последнее время информационные технологии все чаще используются в медицине в виде системного анализа и поддержки принятия решений. К преимуществам данного подхода стоит отнести возможность генерации различных сценариев функционирования системы, а также их исследования с целью оптимизации процесса управления обслуживанием пациентов.

Существует множество исследований процессов обслуживания пациентов в поликлинике или в стационаре. Во всех исследованиях медицинские учреждения представлены, как многоканальная система массового обслуживания (СМО) с ожиданием и неограниченной очередью, но в разных моделях выделяются разнообразные наиболее важные характеристики СМО, описывающие состояние системы. Примерами таких характеристик являются приоритеты заявок на обслуживание (В.М. Хачумов, С.В. Погодин) или количество пациентов в очереди (Е.С. Прокофьева). Однако все проведенные исследования не учитывают возможность контроля нагрузок на каналы обслуживания и распределения потоков заявок в очереди, поскольку нагрузка специалиста медицинского учреждения так же существенно влияет на ожидание пациентов в очереди. Помимо узконаправленных разработок существуют более обобщенные исследования сложных моделей СМО (Ю.В. Жерновый), которые являются комбинированными с изменениями режимов обслуживания. Для процесса распределения потоков заявок необходимо провести обзор существующих алгоритмов распределения заявок в очередях. Существующие исследования алгоритмов приема заявок на обслуживания в очередях также не привязаны к конкретной предметной области.

Целью магистерской диссертации является разработка модели управления обслуживанием пациентов в медицинском учреждении и разработка алгоритма распределения потоков пациентов, позволяющих повысить эффективность и качество обслуживания пациентов в медицинских учреждениях.

В соответствии с поставленной целью выделены следующие задачи работы:

- провести анализ предметной области, в частности, процессов обслуживания пациентов в медицинском учреждении;
- провести анализ известных моделей массового обслуживания медицинского учреждения для определения методов контроля потоков пациентов на приеме у врача;
- провести анализ известных алгоритмов формирования очередей заявок на обслуживание;
- разработать модель распределения потоков пациентов в очереди в медицинском учреждении;
- разработать алгоритм формирования потоков пациентов в очереди на прием у специалиста медицинского учреждения;
- разработать модуль медицинской информационной системы в части формирования расписания приемов пациентов.

Объектом исследования является процесс обслуживания пациентов в медицинской организации.

Предметом исследования являются модели и алгоритмы управления обслуживанием пациентов медицинского учреждения.

В качестве методов исследования используются методы теории массового обслуживания, теория графов и сетей, теории расписаний, системного анализа, методы проектирования программных систем.

Научная новизна работы заключается в следующем:

- предложена модель распределения пациентов в очереди на прием у специалиста медицинской организации, которая, в отличие от известных,

учитывает рабочую нагрузку на специалистов и, в зависимости от показателя нагрузки, формирует очереди на прием;

− предложен алгоритм формирования потоков пациентов в городской поликлинике, который, в отличие от известных, позволяет учесть возможность контроля нагрузки на специалистов и равномерно распределить число пациентов в очередях;

− предложен прототип модуля медицинской информационной системы для управления потоками пациентов в медицинском учреждении, позволяющий контролировать рабочую нагрузку на специалистов медицинской организации и тем самым распределять очереди пациентов.

Разработанные модели и алгоритмы могут быть использованы для применения в процессах обслуживания пациентов медицинского учреждения.

Основные результаты, выносимые на защиту:

− модель управления потоками пациентов в очереди на обслуживание в медицинском учреждении, которая, в отличие от известных, учитывает изменение рабочей нагрузки на специалистов и контролирует длину очереди на обслуживание в зависимости от нагрузки;

− алгоритм формирования потоков пациентов на приеме у специалиста в медицинской организации, который, в отличие от известных, позволяет учесть возможность контроля нагрузки на специалистов и равномерно распределить число пациентов в очередях;

− модуль медицинской информационной системы, реализующий формирование расписаний приема пациентов.

Апробация работы. Основные положения и результаты диссертационной работы докладывались и обсуждались на следующих конференциях:

− VI ежегодная межвузовская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы» (Пенза, март 2019 г.);

− XVI международная научно-техническая конференция «Новые информационные технологии и системы» (Пенза, ноябрь 2019 г.);

− VII ежегодная всероссийская межвузовская научно-практическая конференция «Информационные технологии в науке и образовании. Проблемы и перспективы» (Пенза, март 2020 г.).

Публикации. По теме данной диссертационной работы опубликовано 3 печатных работы.

Работа включает введение, 3 главы, заключение, список использованных источников. Общий объем основного текста – 67 страниц, включая 19 рисунков и 16 таблиц. Список литературы изложен на 3 страницах и содержит 25 наименований.

1 Анализ моделей и алгоритмов обслуживания в медицинском учреждении

1.1 Порядок приема пациента в медицинском учреждении

Одной из главных задач здравоохранения в настоящее время является повышение качества обслуживания пациентов в государственных медицинских учреждениях, в частности, в поликлиниках. Длительное ожидание в очереди снижает доступность амбулаторно-поликлинической помощи и вызывает недовольство населения. Наиболее остро эта проблема выражена в оказании узкоспециализированной помощи.

Поликлиникой принято называть многопрофильное лечебно-профилактическое учреждение, в котором оказывается медицинская помощь пациентам, закрепленным за ним по месту проживания на догоспитальном этапе.

Количество отделений и кабинетов врачей определяются мощностью поликлиники и числом штатных должностей, эти факторы, в свою очередь, зависят от численности населения, закрепленного на данной территории. Структура поликлиники также зависит от потребностей населения в медицинской помощи и способности самой поликлиники оказать эту помощь.

Таким образом, к основным функциям поликлиники относят:

- оказание первой медицинской помощи пациентам на дому;
- оказание медицинской помощи людям с острыми заболеваниями, травмами, отравлениями и других неотложных состояниях независимо от места проживания больного;
- госпитализация больного, нуждающегося в стационарном лечении;
- определение временной нетрудоспособности, освобождение пациента от работы, выдача направлений на медицинские экспертизы людям с признаками утраты трудоспособности;

– организация набора профилактических мероприятий, направленных на снижение заболеваемости, инвалидности и смертности населения, проживающего в определенном районе;

– организация медицинского осмотра и диспансеризации населения;

– направление пациентов на санитарно-курортное лечение;

– организация и проведение мероприятий, направленных на санитарно-гигиеническое воспитание населения и пропаганда здорового образа жизни [1].

Работа поликлиники построена по территориальному принципу – к ней, как правило, прикреплены только те пациенты, которые прописаны в том же районе, где расположена поликлиника. Главной структурной единицей является территориальный терапевтический участок. Население по участкам распределяется руководителями медицинского учреждения.

Для контроля состояния здоровья населения в поликлиниках проводится диспансеризация. Диспансеризация направлена на сохранение на сохранение и быстрое восстановление здоровья населения, и снижение заболеваемости.

Профильные специалисты проводят работу по выявлению заболеваний на разных стадиях, осуществляют систематическое наблюдение, а также своевременное лечение пациента. Кроме того, врач должен предупредить возникновение и распространение вспышек болезней. Для каждого специалиста устанавливается свой перечень пациентов, подлежащих диспансеризации и медицинскому осмотру.

Участковый врач в поликлинике (как правило, он является терапевтом) ведет прием больных как на дому, так и в кабинете, в зависимости от состояния здоровья больного, поэтому он имеет скользящий график.

При анализе процессов обслуживания пациента врачом существуют различные алгоритмы. Порядок приема пациента в медицинском учреждении во многом зависит от его типа. В поликлиниках существует первичный и повторный прием. Кроме того, порядок обследования пациента зависит от

специализации врача (врач общей практики или профильный специалист) и причин обращения.

Если пациент обращается в медицинское учреждение впервые, то такой прием является первичным. Пациент может обратиться к специалисту или врачу общей практики по поводу острого заболевания или обострения хронического заболевания. При обращении к врачу общей практики ведется первичный осмотр пациента. Врач отмечает озвученную проблему, ставит предварительный диагноз и составляет план лечения, назначает дополнительную диагностику или дает направление к профильному специалисту. Кроме того, в зависимости от проблемы, врач может установить прогноз лечения и его объем, информировать пациента о возможных осложнениях. Порядок и результаты осмотра фиксируются в электронной карте пациента [2].

После того, как пациент получил план лечения, прошел данное лечение, он может снова обратиться к врачу через некоторое время. В таком случае, прием классифицируется как повторный. При повторном приеме врач может снова осмотреть пациента, оценить его текущее состояние, назначить новый план лечения и дополнительную диагностику. Все данные, полученные при осмотре и диагностике, попадают в медицинскую информационную систему (МИС) и передаются с ее помощью в разные структуры поликлиники для ускорения процесса оказания помощи.

На данный момент, пациентов в поликлинике можно классифицировать по разным критериям. Одним из важнейших критериев является причина обращения к врачу.

Как правило, в поликлинику пациенты обращаются по одной из следующих причин: консультация, медицинский осмотр, получение лечения или оформление больничного листа. Время, которое требуется врачу на работу с каждым из пациентов, соответственно разное и зависит от причины обращения.

Помимо этого, пациент может прийти по предварительной записи или без записи. Запись чаще всего происходит через телефонный звонок в регистратуру. По предварительной записи обычно приходят пациенты с конкретной проблемой, которые, как правило, проводят в кабинете больше времени, чем остальные. Пациенты без записи чаще всего приходят за больничным листом или за консультацией, в таких случаях, врачу не требуется много времени на прием [3].

На данный момент, основной проблемой поликлиник являются большие потоки пациентов. На рост очередей к специалистам влияет ряд факторов. Все эти факторы являются взаимосвязанными, одними из важнейших являются высокая нагрузка на врачей и их количество. Из-за частичной нехватки специалистов нагрузка перекладывается на врачей, находящихся в штате поликлиники, что затрудняет их работу и влияет на рост очередей. До недавнего времени на время обслуживания одного пациента влияла также и необходимость заполнения специалистом лишнее документации, которую можно оцифровать и хранить в единой базе. Данная проблема уже решена за счет создания медицинских информационных систем (МИС). Данные системы действительно сократили время работы врача с одним пациентом, однако потоки очередей сократились ненамного.

Другим фактором, влияющим на длительность приема, является сам процесс осмотра пациента. Поскольку пациенты обращаются с разными целями (необходимость лечения, консультация, получение справочной информации, получение больничного листа), и на обслуживание каждого пациента уходит разное время. Кроме того, часто пациент обращается с проблемами, не входящими в компетенцию конкретного врача. Таким образом, врач может дать пациенту направление к профильному специалисту, который и будет проводить соответствующую диагностику. Это значит, что пациенту придется обратиться к назначенному врачу, на прием которого тоже появляется очередь. То есть, необходима дополнительная организация приема у профильных специалистов [4].

1.2 Анализ вспомогательных средств работы медицинского учреждения

Хранением и учетом всей медицинской информации о пациенте занимается регистратура. До недавнего времени вся информация велась в бумажном виде, однако теперь все данные хранит информационная система. Медицинские информационные системы производят учет, хранение и оформление не только карт больного, но и других медицинских документов, регулируют нагрузку на врачей посредством талонной и системы и самозаписи пациентов.

Таким образом, медицинские информационные системы (МИС) в поликлиниках обеспечивают:

- информационную поддержку оказания медицинской помощи (ведение медицинских карт пациента и контроль медико-технологических процессов);
- информационную поддержку процесса управления медицинской организации (управление административной и хозяйственной деятельностями формирование передачи данных о техническом лекарственном обеспечении и затратах на них);
- информационную поддержку процессов записи пациентов на прием;
- информационного взаимодействия между несколькими медицинскими учреждениями в рамках оказания медицинской помощи (направление пациентов в другие медицинские учреждения для проведения подробной диагностики и получения медицинской помощи);
- информационного взаимодействия с другими региональными и федеральными учреждениями в части обмена информацией, связанной с лечебно-диагностическим процессом.

Медицинские информационные системы созданы и внедрены со следующими целями:

- повешение качества и доступности медицинской помощи;

– снижение издержек на оказание медицинской помощи при сохранении или повышении уровня результата;

– повышение эффективности работы медицинского учреждения;

– поддержка принятия врачебных решений;

– создание базы данных научно-исследовательской работы.

Медицинские информационные системы хранят большие объемы разных групп информации, которые можно разделить на три больших класса:

– структурированная информация простых типов (числа, строки, даты);

– слабо структурированная информация;

– неструктурированная информация (в основном это графические файлы или документы из различных офисных программ, с их содержимым МИС непосредственно не работает).

Источниками этой информации, как правило, являются:

– различные медицинские записи, которые создаются в процессе оказания помощи;

– данные, считанные медицинской техникой;

– данные, переданные из внешних информационных систем.

Медицинские информационные системы должны взаимодействовать с другими внешними ресурсами или смежными информационными системами. К ним относятся системы управления ресурсами организации, бухгалтерского учета, финансово-экономического планирования, управления проектами, федеральными и региональными медицинскими сервисами. Другими словами, структура любой МИС представляет собой взаимосвязанные модули, каждый из которых отвечает за определенные аспекты работы медицинской организации.

Таким образом, основной функцией медицинских информационных систем является сбор медицинской и административной информации. Данная функция направлена на ускорение работы медицинской организации, в том

числе, сокращение потока пациентов за счет сокращения времени на заполнение медицинской документации [5].

1.3 Обзор готовых решений

Для подробного анализа возможностей медицинских информационных систем в части процесса обслуживания пациентов будет рассмотрено два готовых решения. Следует отметить, что каждая из МИС создана российскими разработчиками, а, следовательно, должна удовлетворять требованиям к разработке МИС, установленными государственными стандартами, часть которых описана в пункте 1.2. Также все представленные системы являются продуктами российских разработчиков.

МИС «Медиалог» – это комплексная МИС для поликлиники, стационара и сети медицинских учреждений, включающая различные модули для работников медицинского учреждения, модули составления отчетов по диагнозам, отчеты по проведенным операциям.

МИС «Медиалог» имеет следующие модули:

- автоматизированное рабочее место заведующего отделения;
- автоматизированное рабочее место врача;
- автоматизированное место медсестры.

Рассматриваемая МИС позволяет пользователям осуществлять следующие наборы функций:

- ведение истории болезни пациентов;
- оптимизация планирования лечебных процессов;
- управление финансовыми ресурсами организации;
- считывание результатов диагностики с лечебных приборов;
- планирование размещения пациентов в стационаре и санатории;
- установка прав доступа пользователей к ресурсам системы.

Архитектура системы «Медиалог» клиент-серверная, в серверной части установлена база данных на основе СУБД Microsoft SQL Server, клиентские приложения работают на базе операционных систем Windows 2000, Windows

XP, Windows Vista, Windows 7 и т.д. в зависимости от количества необходимых автоматизированных рабочих мест (доступно более 500 мест).

Медицинская информационная система «Medods» является системой нового поколения, применяется и в частных клиниках, и в государственных.

Главным преимуществом «Medods» является простота ее интерфейса, что соответственно не требует временных затрат на обучение пользователей. Еще одно преимущество данной системы по сравнению с другими рассмотренными – низкая стоимость программного обеспечения.

МИС содержит довольно распространенный и полезный функционал:

- запись пациентов на прием онлайн;
- оповещение пациентов по SMS или электронной почте;
- ведение медицинской электронной карты;
- создание отчетов;
- ведение складского учета;
- интеграция с медицинской техникой;
- интеграция с телефонией;
- интеграция с онлайн кассами.

Проведенный анализ существующих медицинских информационных систем показал, что описанные программы обладают определенными достоинствами и недостатками. К достоинствам систем стоит отнести их большой функционал, позволяющий минимизировать временные затраты на оформление медицинской документации и тем самым сократить время осмотра одного пациента из-за отсутствия документов, которые нужно заполнять вручную [6].

Недостатком представленных систем является отсутствие функционала, направленного на ведение расписаний приема пациентов для каждого из врачей, которое помогло бы контролировать потоки пациентов за счет их распределения по временным интервалам в зависимости от поставленных проблем каждого из пациентов и нагрузки врача.

1.4 Анализ поликлиники как системы массового обслуживания

Поликлиника является организацией, в которой процессы обслуживания, моменты прихода новых пациентов, интенсивность потока пациентов и время их обслуживания происходят в случайные моменты времени, что можно описать вероятностными функциями. Такие системы можно представить, как математические модели, которые называются системами массового обслуживания. Любая подобная система обладает числом обслуживающих приборов, входным и выходным потоком заявок на обслуживание у прибора. Также существует дисциплина обслуживания – набор определенных условий, которые должна выполнить заявка, в случае их выполнения заявка считается выполненной и покидает систему во входном потоке. Любая модель массового обслуживания обладает следующими характеристиками:

– входной поток заявок на обслуживание (пациенты, желающие записаться на прием);

– в случае модели массового обслуживания с очередью существует буфер очереди, который может быть ограничен либо числом мест, либо времени ожидания в очереди, либо не ограничен ничем;

– обслуживающее устройство (врач);

– выходной поток обслуженных заявок (пациенты, завершившие прием у врача).

Считаем, что среднее время между поступлениями заявок известно и равно T. Значит, интенсивность потока заявок λ обратно пропорциональна T, то есть

$$\lambda = \frac{1}{T}.$$

Пусть среднее время обслуживания заявки равно $t_{обс}$ и является случайной величиной, а длительность обслуживания одной заявки μ обратно пропорциональна $t_{обс}$, то есть

$$\mu = \frac{1}{t_{\text{обс}}}.$$

Отношение интенсивности потока заявок к длительности обслуживания заявки обозначено через ρ. Таким образом,

$$\rho = \frac{\lambda}{\mu}.$$

Величина ρ является нагрузкой на канал обслуживания, и соответственно зависит от интенсивности потока заявок на обслуживание в канале и времени обслуживания заявок. Данный параметр существенно влияет на среднее время ожидания заявки в очереди [7].

Существует множество исследований систем массового обслуживания в рамках текущей предметной области.

В [8] описана работа лечебного учреждения как параллельно-последовательную систему массового обслуживания, в которой существуют первичный поток пациентов и вторичный поток больных. Врач-терапевт занимается распределением пациентов, оформляя направления к профильным специалистам. Направления к тому или иному специалисту осуществляется с определенными вероятностями. То есть, терапевт подвергает первичный поток пациентов операции разряжения. Данная операция заключается в следующем: каждый пациент с вероятностью p_i направляется в соответствующий профильный кабинет, в другой профильный кабинет с вероятностью $1 - p_i$. Одновременно с описанным процессом происходит подсчет среднего временного интервала между пациентами первичного потока, которые направляются в профильные кабинеты. Помимо первичного потока пациентов движется вторичный поток больных, которые продолжают лечение. Поток больных отправляется на обслуживание к профильным специалистам, где объединяется с первичным потоком. Данная модель учитывает кластеризацию первичного потока пациентов, а также образование вторичного потока больных, характерных для процесса обслуживания пациентов в медицинских учреждениях. Также модель учитывает время длительности заявки и время обработки заявки.

В [9] медицинское учреждение представлено как совокупность одноканальных и многоканальных СМО с очередью. Поликлиника рассмотрена на примере системы с несколькими структурами: регистратура, кабинеты хирурга, терапевта и окулиста. В системе также был описан стационар, в частности, отделения хирургии, офтальмологии и терапии. Модель показывает, что пациенты, покинувшие регистратуру и вставшие в очередь на прием к конкретному специалисту, обладают определенными показателями приоритетов. Значение приоритета определяется в зависимости от сложности заболевания. После обслуживания в кабинете врача пациент либо отправляется в выходной поток обслуженных пациентов, либо в стационар. В стационаре пациент обслуживается с длительностью, зависящей от сложности заболевания, а после окончания обслуживания отправляется в выходной поток.

В [10] проведено исследование медицинского учреждения как многоканальной системы массового обслуживания с очередью с использованием нотации Кендалла, учитывающей следующие факторы: время между поступлениями заявок в очередь, распределение времени обслуживания и количество обслуживающих устройств. Состояние системы определяется количеством пациентов в очереди или на приеме и вероятностью того, что поступивший пациент встанет в очередь.

В [11] проведено исследование модели работы медицинского учреждения, взяв за основу работу одного врача поликлиники смоделировав ее, как одноканальную СМО с ожиданием. В данной работе решается задача идентификации входного потока пациентов.

В работе [12] рассматривается система массового обслуживания поликлиники с циклической дисциплиной прохождения заявок, то есть заявка покинет систему только тогда, когда будет обслужена во всех каналах обслуживания. Все возможные структуры графа такой СМО описаны через метод вероятностных графов. Исследуемая в работе система является многоканальной с неограниченной очередью.

Помимо узконаправленных анализов моделей СМО медицинской организации существуют анализы более обобщенных моделей СМО, но при этом допускающие изменения режимов обслуживания.

В [13] были изучены две модели, отличительные их особенности заключаются в изменении числа заявок в системе и интенсивности обслуживания. Ниже представлена модель с изменяющейся интенсивностью потока заявок. В рассматриваемой системе существует l режимов поступлений заявок. Длительность каждого режима распределяется по показательному закону и является независимой случайной величиной. На вход в систему при i-м режиме поступает простейший поток заявок и установлено число задействованных каналов обслуживания. Если число задействованных каналов в момент переключения режимов не изменяется, то все обслуживаемые заявки продолжают обслуживаться в своих каналах и длина очереди не уменьшается. Если число работающих каналов увеличивается, то заявки, стоящие в очередях, поступают на обслуживание в новые пустые каналы. Если число каналов уменьшается, то заявки, стоявшие в очереди на обслуживание у потерянных каналов, вынуждены дожидаться окончания обслуживания в очереди.

Все описанные модели являются многоканальными СМО с неограниченной очередью. В проведенных исследованиях учитываются время поступления заявок, количество заявок и приоритет работ, связанных с обслуживанием заявок. Однако рассмотренные системы не учитывают нагрузку на каналы обслуживания, поскольку в случае перегрузки канал будет неспособен обслужить последующие пришедшие в систему заявки. В этом случае заявки могут находиться в очереди достаточно длительное время, поскольку нет ограничения на время ожидания в очереди, к тому же будут приходить новые заявки, поскольку очередь не ограничена. В таком случае, необходима разработка модели системы массового обслуживания, в которой учитывается степень нагрузки на каналы обслуживания для более равномерного распределения заявок в очередях.

1.5 Анализ алгоритмов формирования очередей в многоканальной системе массового обслуживания

Вопросы оптимального распределения потоков заявок в очередях решает теория расписаний.

В теории расписаний обслуживающее устройство (канал обслуживания), выполняющее действия над заявкой принято называть машиной. Заявку в интерпретации теории очередей принято называть работой.

Для рассмотрения алгоритмов управления потоками заявок в медицинской организации представлено два множества:

– множество машин $M = \{M_1, M_2, \ldots, M_m\}$ (каналы обслуживания);

– множество работ $J = \{J_1, J_2, \ldots, J_n\}$ (потоки заявок на обслуживание).

В системе функционируют несколько машин, работающих параллельно. Машинам не допускается выполнять одновременно две работы, также не допускается прерывание операций. Каждая *i*-я работа обладает длительностью p. Должно быть учтено, что любая работа может быть выполнена на любой из представленных машин (в поликлинике пациент может обследоваться у любого врача конкретного профиля). Необходимо сформировать порядок приема работ на обслуживание машинами с минимальным временем завершения всех работ.

В рамках текущей предметной области не производилось исследований алгоритмов формирования очередей. Однако существуют обобщенные алгоритмы формирования потоков заявок на обслуживание. Все они классифицируются между собой по различным значениям собственных характеристик. При исследовании многоканальных СМО стоит рассмотреть алгоритмы формирования очередей (или расписания) для количества машин $M \geq 2$. Далее представлены наиболее известные алгоритмы.

Для решения задачи оптимального формирования потоков заявок необходимо найти нижнюю оценку длительности работ *LB*

$$LB = max\left\{max\ p_i;\ \frac{1}{m}\sum_{i=1}^{n} p_i\right\},$$

где m – число машин;

n – число работ.

Для формирования порядка обслуживания работ необходимо взять список работ и загрузить машины последовательно. Первая машина возьмет i работ и выполнит их в интервале времени [0, LB], если последняя работа не выполнена до конца на этом интервале, то ее остатки переносятся на следующую машину, которая выполнит ее на интервале [$0, LB$].

Так как $LB \geq max(p_i)$, i = 1 ... n, то работа выполняется не более чем на одной машине. Поэтому данное распределение является допустимым.

Для подробного рассмотрения взят случай, когда длительности работ p_i разные, так как такой вариант системы является наиболее подходящим в рамках работы поликлиники. В таком случае работы с большими длительностями будут ставиться на более быстрые машины до тех пор, пока длительность последних добавленных в поток работ не сократится до такой степени, что совпадет с длительностью других работ, принятых на обслуживание.

Пусть $p_i(t)$ является частью работы i, которая еще не выполнена к моменту времени t. Тогда алгоритм распределения задач будет двигаться, но в определенный момент времени s он остановится, чтобы перераспределить работы по машинам. На рисунке 1 изображена блок-схема алгоритма распределения потоков заявок в случае, когда время обслуживания разное для каждой из задач.

На рисунке показано, что при существовании работ с ненулевой длительностью происходит назначение работ на определенные машины. Процесс распределения работ отмечен на схеме блоком «Назначение работ по машинам».

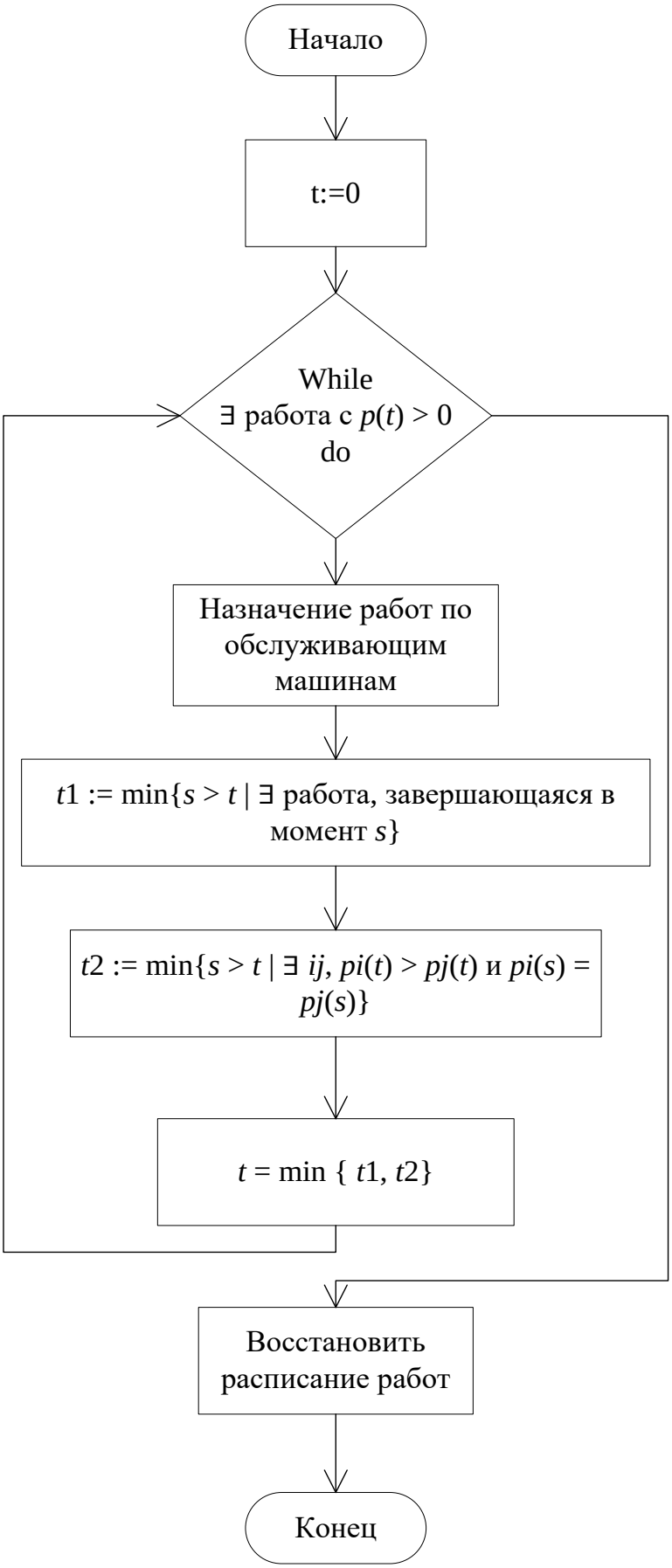

Рисунок 1 – Алгоритм формирования потоков заявок

Алгоритм назначения работ по машинам (на рисунке 1 – «Произвести назначение работ по машинам») изображен на рисунке 2.

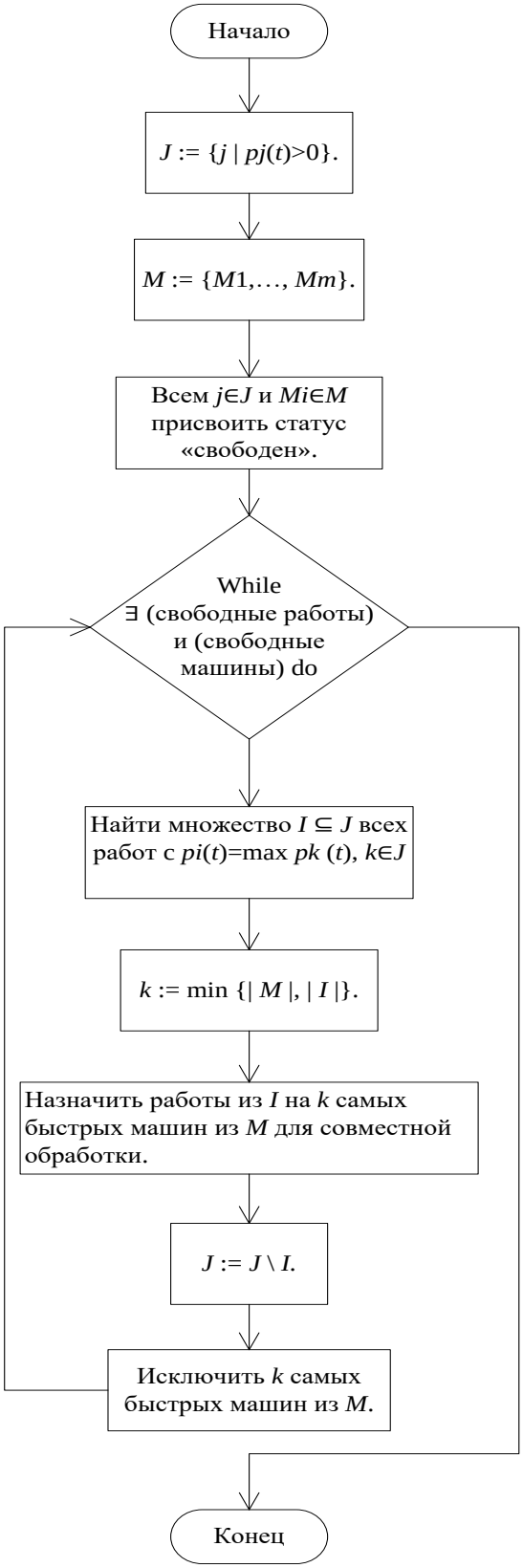

Рисунок 2 – Назначение работ по машинам

Недостатком рассмотренных алгоритмов является необходимость разделять одну работу на подзадачи и перераспределить их для обслуживания по машинам, что является невозможным к реализации в поликлинике. Кроме того, алгоритмы позволяют в конкретный момент времени перераспределить все работы по машинам, что также нереализуемо, поскольку невозможно совершить данное перераспределение в реальном времени. Также проблемой реализации данных алгоритмов на практике в поликлинике является невозможность совместного приема одного пациента двумя врачами. Поэтому необходимо разработать алгоритм распределения входного потока заявок на обслуживание с учетом выполнения работ целиком на одной машине [14].

Выводы

Анализ предметной области и соответствующих программных средств показал, что одной из проблем процесса обслуживания пациентов в медицинском учреждении (в частности, поликлинике) до сих пор остается неконтролируемый поток пациентов, образующих очередь на прием. Ожидание в очереди чаще всего зависит от времени обследования одного пациента и загруженности врача. Эти проблемы решает внедрение информационных систем, которые, за счет небольшого уменьшения временных затрат на оформление врачом документации. Для контроля потоков пациентов в поликлинике предлагается оптимизировать модель приема новых пациентов.

Модели, изученные в пункте 1.4, не учитывают возможность контроля нагрузки.

Математические алгоритмы управления потоками пациентов, исследованные в пункте 1.5 предполагают возможность разбиения работы, перенесения части работы на другую машину и множественного перераспределения по функционирующим машинам.

В условиях поликлиники такой подход некорректен, поскольку пациент в лучшем случае должен быть обслужен один раз одним врачом, в худшем,

записан на прием еще раз. Однако даже при таком возможном сценарии считается, что один прием равноценен одной работе и его нельзя разбивать на подзадачи. Однако, рассмотренные алгоритмы, при полном обслуживании работы на одной машины позволяют оптимально распределить поток пациентов на прием, что поможет контролировать поток очередей пациентов на прием.

Для удобства мониторинга очередей пациентов требуется разработка вспомогательного модуля медицинской информационной системы, в котором будет храниться информация о записях к тому или иному врачу, а также информация о предстоящих приемах.

2 Модель управления потоками пациентов в очереди на обслуживание

2.1 Описание модели потоков пациентов в поликлинике

Каналы обслуживания выполняют работу неравномерно, то есть с разной интенсивностью обслуживания, являющейся случайной величиной. Тогда нецелесообразно отправлять заявки в данный канал, стоит распределить их по более свободным каналам. Кроме того, модель СМО поликлиники должна быть с ограничением на длину очередей, но ограничение не будет фиксированным, а зависеть от нагрузки канала обслуживания и изменяться в момент времени t.

Изменение длины очереди во времени стоит рассматривать как режим обслуживания. С учетом высокой нагрузки на канал обслуживания возможное количество мест в очереди m стоит ограничить до тех пор, пока значение нагрузки на канал не упадет. Так как система массового обслуживания многоканальная, и нагрузка на каналы распределена неравномерно, велика вероятность того, что существует канал с более низкой нагрузкой, в этом случае необходимо перенаправить новые заявки в более пустой канал. В момент времени t последнее состояние системы фиксируется и новые пришедшие в момент t заявки распределяются в зависимости от состояния системы.

Для определения максимальной и минимальной нагрузки между специалистами введем максимальное значение нагрузки канала ρ_{max} и минимальное значение ρ_{min}. Важно отметить, что нагрузка, например, врача-терапевта должна быть сравнима соответственно только с нагрузками других врачей терапевтом медицинского учреждения. Поиск максимального и минимального значения нагрузок каналов обслуживания, функционирующих в системе, осуществляется путем сортировки значений от минимального к максимальному. Следовательно, степень нагрузки ρ на i-й канал находится в интервале

$$\rho_{min} \leq \rho_i \leq \rho_{max}.$$

Если нагрузка ρ_i близится к значению ρ_{min}, то допустимое количество мест в очереди может быть увеличено на величину l, в противном случае – уменьшено из-за уменьшения числа мест в очереди m. Таким образом, величина m является динамической.

В случае предложенной модели массового обслуживания алгоритм движения заявки по системе выглядит следующим образом. В момент времени t на вход поступает заявка, перед которой несколько очередей и несколько каналов обслуживания. В случае высокой занятости i-го канала обслуживания очередь ограничивается, если мест в очереди нет, заявку переводят в очередь на другой канал обслуживания, если его нагрузка ниже нагрузки i-го канала. Если нагрузки всех каналов высокие и мест в ограниченных очередях нет, то заявка покидает систему необслуженной до следующего момента t_1. Если нагрузка каналов снижается, ограничение на очередь отменяется.

На рисунке 3 изображен граф модели СМО с временным ограничением на длину очереди медицинского учреждения.

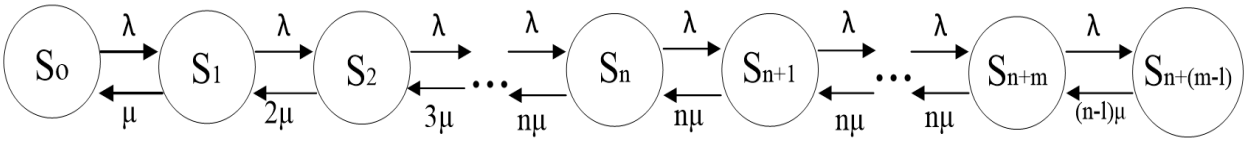

Рисунок 3 – СМО с временным ограничением на длину очереди

На рисунке видно, что модель обладает следующими состояниями:

− в системе 0 заявок;

− 1 заявка, очередь пустая;

− 2 заявки, очередь пустая;

− n заявок, очередь пустая;

− n заявок на обслуживании, 1 заявок в очереди;

− n заявок на обслуживании, m заявок в очереди, очередь ограничена и заполнена (при максимальной загрузке канала);

− *n* заявок на обслуживании, *m*-*l* заявок в очереди, ограничение на очередь снято (при снижении нагрузки канала).

Таким образом, модель медицинского учреждения может представлять собой комбинированную систему массового обслуживания с изменением режима очередей.

Для дальнейшей проверки оптимальности предложенной модели приведено подробное описание алгоритма обслуживания пациентов в системе массового обслуживания с изменением режимов очередей. На вход в систему поступает единый поток пациентов, обращающихся в регистратуру. Сотрудник регистратуры обслуживает каждого пациента посредством опроса и направляет к тому или иному врачу, тем самым распределяя очереди к специалистам. Если *i*-й врач обладает высокой нагрузкой, то очередь к нему ограничена, и пациент направляется к *j*-му врачу. Во время приема в кабинете врача пациент может быть перенаправлен к профильному специалисту (если прием велся у врача общей практики), либо на дополнительную диагностику. Алгоритм в этом случае повторяется, пациент будет перенаправлен к специалисту с более низкой нагрузкой. Пациенты, завершившие обслуживание, образуют выходной поток заявок, покидающий систему.

Одним из средств оценки эффективности моделей обслуживания является аппарат сетей Петри. Для того чтобы автоматизировать процесс построения сетей Петри, используется программный пакет «Petri net emulator».

Сеть Петри, описывающая рассмотренную модель, представлена на рисунке 4. Сетью Петри называется множество вершин (P) и переходов (T), между ними устанавливается связь. В данном варианте сети Петри под множеством вершин подразумеваются возможные состояния каналов обслуживания, переходы характеризуют движение очередей на обслуживание.

В данной сети рассматривается взаимодействие трех каналов обслуживания исследуемой СМО. В данном случае каналы обслуживания интерпретируются как врачи одного профиля поликлиники.

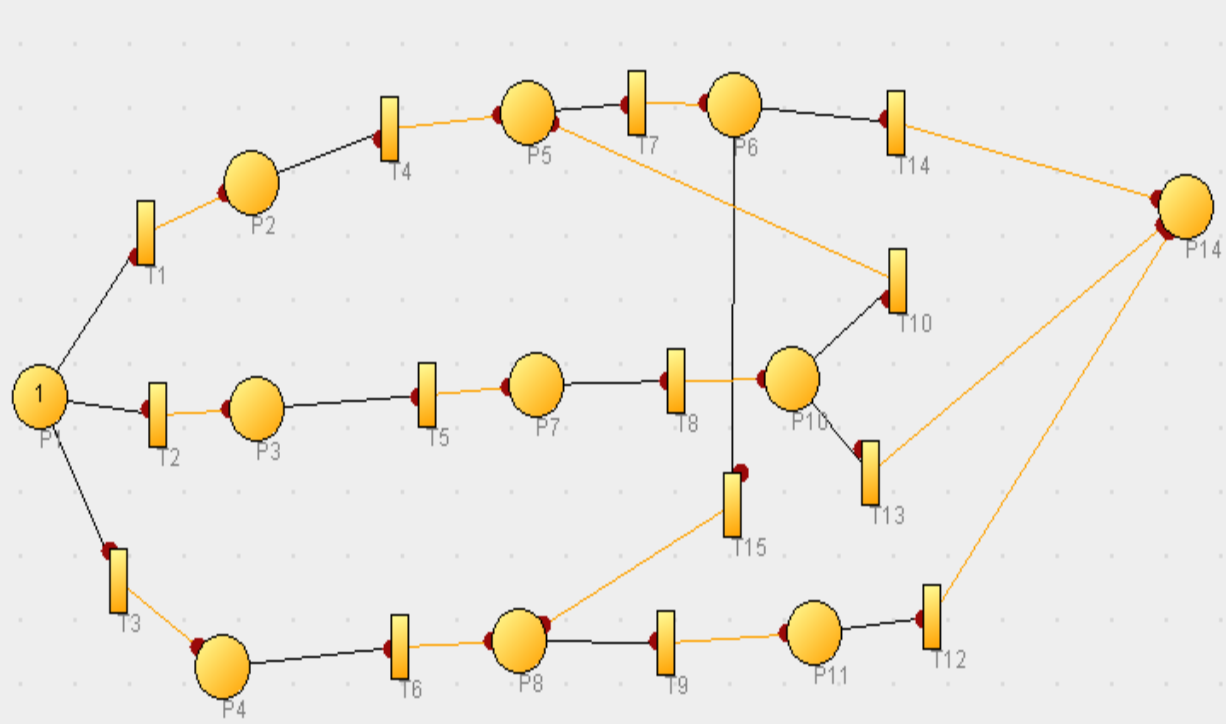

Рисунок 4 – Сеть Петри, имитирующая модель массового обслуживания поликлиники

Описания состояний системы, представленные в сети в виде вершин, приведены в таблице 1.

Таблица 1 – Условия срабатывания переходов

Вершина	Условие срабатывания
P_1	Все три канала пусты, в очереди нет заявок, начало обслуживания
P_2	В первом канале одна заявка, очереди нет
P_3	В втором канале одна заявка, очереди нет
P_4	В третьем канале одна заявка, очереди нет
P_5	В первом канале одна заявка, в очереди m заявок, нагрузка на канал низкая
P_6	В первом канале одна заявка, в очереди n заявок, нагрузка на канал высокая
P_7	Во втором канале одна заявка, в очереди m заявок, нагрузка на канал низкая
P_8	В третьем канале одна заявка, в очереди m заявок, нагрузка на канал низкая
P_{10}	Во втором канале одна заявка, в очереди n заявок, нагрузка на канал высокая
P_{11}	В третьем канале одна заявка, в очереди n заявок, нагрузка на канал высокая
P_{14}	Завершение обслуживания

Как правило, сети Петри описываются множествами вершин (P) и переходов (T), а также входными (I) и выходными функциями (O).

Далее представлено описание структуры сети Петри:

$C = (P, T, I, O)$,

$P = \{p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_{10}, p_{11}, p_{14}\}$,

$T = \{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{12}, t_{13}, t_{14}\}$.

В таблице 2 описаны входные и выходные функции для вершин данной сети Петри:

Таблица 2 – Описание входных и выходных функций вершин сети Петри

$I : P \rightarrow T^{\infty}$	$O : P \rightarrow T^{\infty}$
$I(p_1) = \{\}$	$O(p_1) = \{t_1, t_2, t_3\}$
$I(p_2) = \{t_1\}$	$O(p_2) = \{t_4\}$
$I(p_3) = \{t_2\}$	$O(p_3) = \{t_5\}$
$I(p_4) = \{t_3\}$	$O(p_4) = \{t_6\}$
$I(p_5) = \{t_4, t_{10}\}$	$O(p_5) = \{t_7\}$
$I(p_6) = \{t_7\}$	$O(p_6) = \{t_{15}, t_{14}\}$
$I(p_7) = \{t_5\}$	$O(p_7) = \{t_8\}$
$I(p_8) = \{t_6, t_{15}\}$	$O(p_8) = \{t_9\}$
$I(p_{10}) = \{t_8\}$	$O(p_{10}) = \{t_{10}, t_{13}\}$
$I(p_{11}) = \{t_9\}$	$O(p_{11}) = \{t_{12}\}$
$I(p_{14}) = \{ t_{14}, t_{13}, t_{12}\}$	$O(p_{14}) = \{\}$

В таблице 3 описаны входные и выходные функции для вершин данной сети Петри:

Таблица 3 – Описание входных и выходных функций переходов сети Петри

$I : T \rightarrow P^{\infty}$	$O : T \rightarrow P^{\infty}$
$I(t_1) = \{p_1\}$	$O(t_1) = \{p_2\}$
$I(t_2) = \{p_1\}$	$O(t_2) = \{p_3\}$
$I(t_3) = \{p_1\}$	$O(t_3) = \{p_4\}$
$I(t_4) = \{p_2\}$	$O(t_4) = \{p_5\}$
$I(t_5) = \{p_3\}$	$O(t_5) = \{p_7\}$
$I(t_6) = \{p_4\}$	$O(t_6) = \{p_8\}$

Продолжение таблицы 3

$I: T \rightarrow P^{\infty}$	$O: T \rightarrow P^{\infty}$
$I(t_7) = \{p_5\}$	$O(t_7) = \{p_5\}$
$I(t_8) = \{p_7\}$	$O(t_8) = \{p_{10}\}$
$I(t_9) = \{p_8\}$	$O(t_9) = \{p_{11}\}$
$I(t_{10}) = \{p_5\}$	$O(t_{10}) = \{p_5\}$
$I(t_{12}) = \{p_{11}\}$	$O(t_{12}) = \{p_{14}\}$
$I(t_{13}) = \{p_{10}\}$	$O(t_{13}) = \{p_{14}\}$
$I(t_{14}) = \{p_6\}$	$O(t_{14}) = \{p_{14}\}$
$I(t_{15}) = \{p_6\}$	$O(t_{15}) = \{p_8\}$

Для того чтобы рассмотреть свойства сети Петри, необходимо построить дерево достижимости, показывающее все возможные последовательности работы переходов. Для начала работы необходимо поставить маркировку на вершине p_1. На рисунке 5 представлено дерево достижимости для построенной сети Петри.

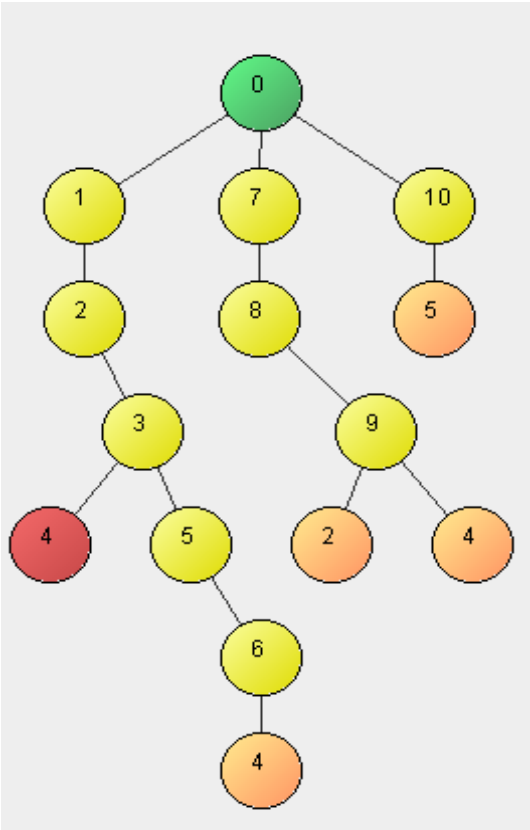

Рисунок 5 – Дерево достижимости сети Петри

На рисунке видно, что последовательность переходов имеет три тупиковые ситуации, которые можно охарактеризовать как завершение обслуживания заявки в одном из трех каналов.

На рисунке 6 представлено табличное описание дерева достижимости.

Для подтверждения корректности сети Петри необходимо рассмотреть три ее свойства: безопасность, ограниченность и сохранение.

#	Branch	Prev.marking	Worked Transiotions	Cur. marking	MarkType	Level
1	0	M0(1,0,0,0,0,0,0,0,0,0,0)		M0(1,0,0,0,0,0,0,0,0,0,0)	ROOT	0
2	1	M0(1,0,0,0,0,0,0,0,0,0,0)	T1	M1(0,1,0,0,0,0,0,0,0,0,0)	INTERNAL	1
3	1	M1(0,1,0,0,0,0,0,0,0,0,0)	T4	M2(0,0,0,0,1,0,0,0,0,0,0)	INTERNAL	2
4	1	M2(0,0,0,0,1,0,0,0,0,0,0)	T7	M3(0,0,0,0,0,1,0,0,0,0,0)	INTERNAL	3
5	1	M3(0,0,0,0,0,1,0,0,0,0,0)	T13	M4(0,0,0,0,0,0,0,0,0,0,1)	DEADLOCK	4
6	2	M3(0,0,0,0,0,1,0,0,0,0,0)	T14	M5(0,0,0,0,0,0,0,1,0,0,0)	INTERNAL	4
7	2	M5(0,0,0,0,0,0,0,1,0,0,0)	T9	M6(0,0,0,0,0,0,0,0,0,1,0)	INTERNAL	5
8	2	M6(0,0,0,0,0,0,0,0,0,1,0)		M4(0,0,0,0,0,0,0,0,0,0,1)	REPEATED	6
9	3	M0(1,0,0,0,0,0,0,0,0,0,0)	T2	M7(0,0,1,0,0,0,0,0,0,0,0)	INTERNAL	1
10	3	M7(0,0,1,0,0,0,0,0,0,0,0)	T5	M8(0,0,0,0,0,0,1,0,0,0,0)	INTERNAL	2
11	3	M8(0,0,0,0,0,0,1,0,0,0,0)	T8	M9(0,0,0,0,0,0,0,0,1,0,0)	INTERNAL	3
12	3	M9(0,0,0,0,0,0,0,0,1,0,0)	T10	M2(0,0,0,0,1,0,0,0,0,0,0)	REPEATED	4
13	4	M9(0,0,0,0,0,0,0,0,1,0,0)	T12	M4(0,0,0,0,0,0,0,0,0,0,1)	REPEATED	4
14	5	M0(1,0,0,0,0,0,0,0,0,0,0)	T3	M10(0,0,0,1,0,0,0,0,0,0,0)	INTERNAL	1
15	5	M10(0,0,0,1,0,0,0,0,0,0,0)	T6	M5(0,0,0,0,0,0,0,1,0,0,0)	REPEATED	2

Рисунок 6 – Табличное описание дерева достижимости

Сеть Петри является ограниченной, если число фишек в любой ее позиции не превышает некоторого значения *K*. В данном случае, сеть является ограниченной, поскольку по таблице видно, что в любой позиции число фишек никогда не превышает единицу. Таким образом свойство ограниченности подтверждено.

Сеть Петри является безопасной (частный случай ограниченности), если число фишек в любой позиции не превышает значения $K=1$. Как уже рассмотрено ранее, данная сеть имеет не более одной фишки в каждом переходе, следовательно, она является безопасной.

Сохранение сети Петри имеет место, если общее число маркеров является постоянной величиной, то есть новые маркеры не добавляются, а существующие не уничтожаются. В случае данной сети свойство сохранения имеет место.

Рассмотренные свойства данной сети Петри для исследуемой модели обслуживания подтверждают ее корректность. Использование подхода сетей Петри позволило рассмотреть все возможные состояния работы модели массового обслуживания медицинского учреждения на примере нескольких однопрофильных специалистов.

Анализ системы массового обслуживания был проведен с целью анализа качества функционирования конкретной области и выявления возможности оптимизации самой СМО. Качество функционирования в отдельных случаях имеет свое значение, которое может выражаться, в том числе и количественными показателями. В поликлинике, как системе массового обслуживания особенно важны такие критерии, как длина очереди на обслуживание, средняя длительность обслуживания, среднее время ожидания заявки в очереди, время простоя обслуживающих машин [15].

2.2 Алгоритм формирования потоков пациентов для приема у специалиста в медицинском учреждении

Пусть к моменту времени t в систему поступило множество работ, обозначенное, как $J = \{J_1, J_2, \ldots, J_n\}$, каждая из которых обладает длительностью ρ_i. В системе имеется множество машин $M = \{M_1, M_2, \ldots, M_n\}$, обладающих нагрузкой ρ_i. Под нагрузкой обслуживающей машины подразумевается отношение среднего числа пришедших заявок к средней скорости обслуживания. Необходимо составить расписание выполнения новых поступивших работ к моменту времени t_1 так, чтобы работа была выполнена целиком на одной машине без повторного перераспределения следующих работ. Для выполнения алгоритма необходимо сначала определить все значения нагрузок обслуживающих машин, затем значения нагрузок должны быть отсортированы от минимального значения к максимальному. Для выполнения набора работ, появившихся к моменту времени t необходим набор машин с минимальной нагрузкой. Считается, что нагрузки машин уже известны к моменту t. После сортировки нагрузок

очереди заполняются заявками, причем сначала добавляются заявки с большей длительностью обслуживания. Когда очередь заполнена данными заявками, повторно оценивается нагрузка каналов, производится сортировка каналов по нагрузке и в очереди к более свободным каналам добавляются заявки с меньшей длительностью обслуживания. Данный алгоритм учитывает тот факт, что мероприятия по обслуживанию пациента не являются разделимыми и не могут быть выполнены двумя специалистами одновременно. Также учитывается, что заявки, уже стоящие в очереди на обслуживание, не могут быть перенесены в другую очередь или быть исключены из очереди.

Процесс начала обслуживания пациентов у специалиста медицинского учреждения должен начинаться со сбора информации о поступивших заявках и оценки нагрузки врача для дальнейшего распределения пациентов в расписании. Сбор информации о пациентах, записавшихся на прием, происходит с начало приема звонков от самих пациентов, либо через web-сервис медицинского учреждения. Количество записей не может быть ограничено, но поскольку решено, что оптимальным вариантом контроля за очередями будет прием по записи, в зависимости от рабочей нагрузки конкретного врача, к нему на прием возможно записать определенное количество пациентов в день. От загруженности конкретного врача зависит, сколько еще пациентов можно записать к нему на прием.

Важно учитывать, что бывают пациенты, которые приходят на прием без записи. Для исследуемой модели обслуживания целесообразно формирование очереди только по записи, чтобы минимизировать ожидание пациентов в очереди. Но если появится пациент без записи, необходимо записать его на прием к врачу по рассматриваемому алгоритму с учетом установленных нагрузок на врача. Если в момент прихода пациента есть специалист с низкой нагрузкой и низким числом пациентов в очереди, то сотрудник регистратуры может направить пациента к нему.

На рисунке 7 изображена блок-схема словесного алгоритма принятия заявок на прием к врачу и процесса приема пациентов.

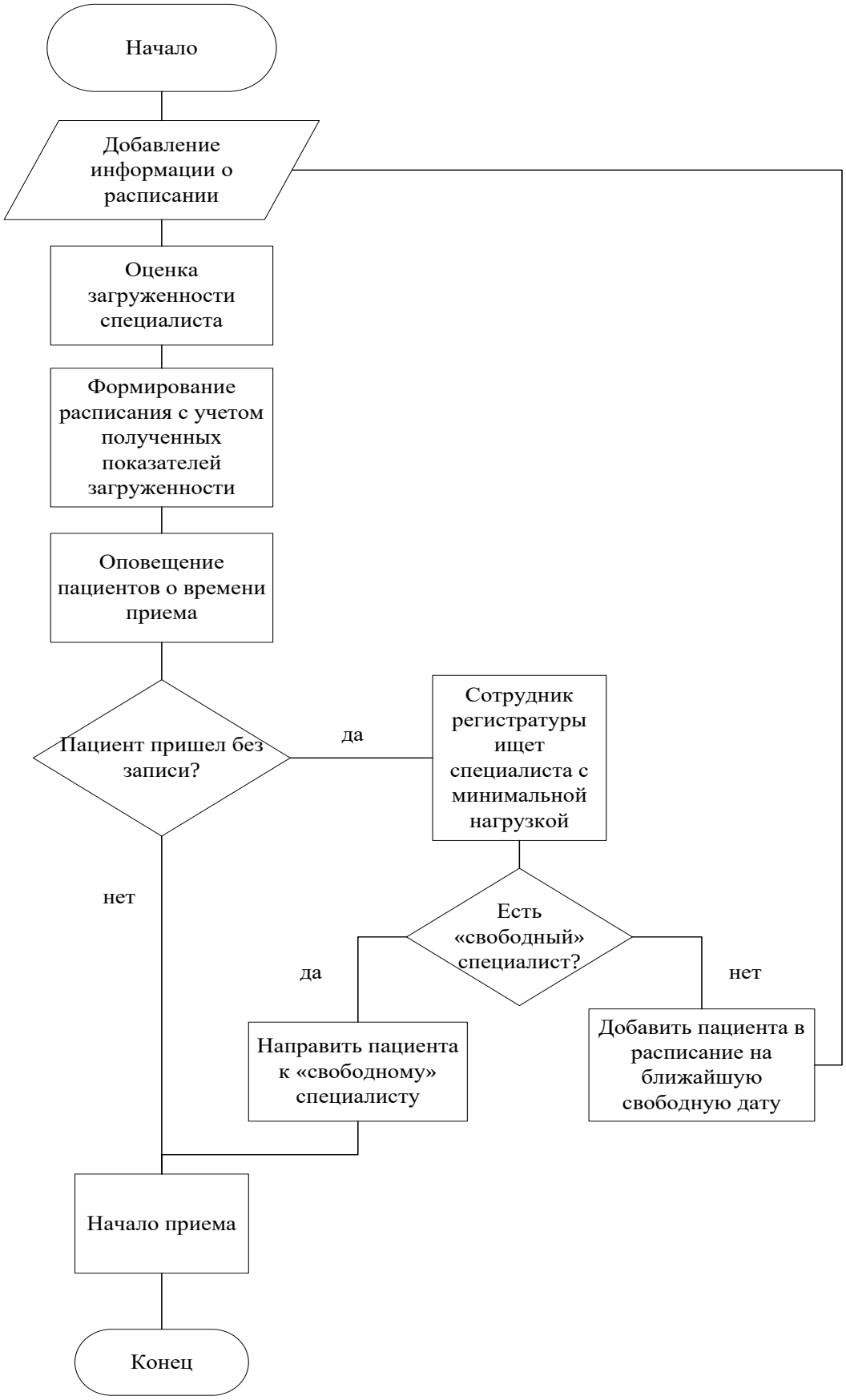

Рисунок 7 – Блок-схема процесса создания расписания и приема пациентов

Перед добавлением очередной записи в график сотрудником анализируется рабочая нагрузка специалистов необходимого профиля. Этот анализ осуществляется с помощью базы данных. Для этого необходимо осуществить выборку всех врачей необходимого профиля и отсортировать данные по критерию рабочей нагрузки специалиста. После проведения сортировки для дальнейшего добавления новой записи рассматриваются специалисты с установленной минимальной нагрузкой. Далее необходимо реализовать процедуру записи на прием к врачу и установку времени и даты приема.

Важным вопросом для рассмотрения является определение рабочей нагрузки специалистов. Из пункта 2.1 определено, что специалист медицинской организации в системе массового обслуживания, по сути, является каналом обслуживания, обладающим характеристикой ρ (коэффициентом использования обслуживающей машины). Данная величина может интерпретироваться как величина рабочей нагрузки специалиста медицинского учреждения и рассчитываться как отношение среднего количества пациентов, записывающихся на прием за последнее время к средней длительности обслуживания пациентов [16].

Для проверки корректности предложенного алгоритма повторно смоделирована сеть Петри в среде «Petri met emulator», представленная на рисунке 8.

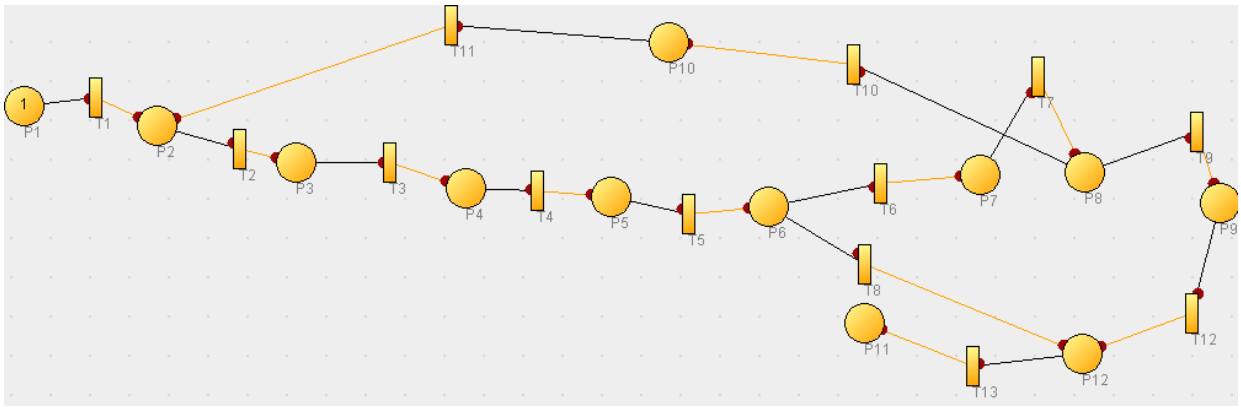

Рисунок 8 – Сеть Петри для алгоритма приема заявок и формирования расписания

Описание переходов сети Петри представлено в таблице 4.

Таблица 4 – Описание переходов сети Петри

Вершина	Условие срабатывания
P_1	Начало
P_2	Добавление информации о расписании
P_3	Оценка загруженности специалистов
P_4	Формирование расписания с учетом полученных показателей загруженности
P_5	Оповещение пациентов о времени приема
P_6	Проверка прихода пациента по записи
P_7	Пациент пришел без записи, сотрудник ищет специалиста с минимальной нагрузкой
P_8	Проверка существования специалиста с минимальной нагрузкой
P_{10}	Врача с минимальной нагрузкой нет, записать пациента на ближайшую дату, на шаг 2
P_9	Врач с минимальной нагрузкой есть, направить пациента к специалисту с минимальной нагрузкой
P_{12}	Начало приема
P_{11}	Конец

На рисунке 9 изображено дерево достижимости созданной сети Петри.

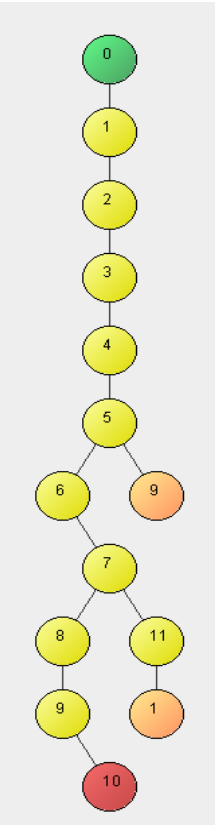

Рисунок 9 – Дерево достижимости сети Петри

Для описания свойств сети Петри будет рассмотрена таблица с описанием дерева достижимости, изображенную на рисунке 10.

#	Branch	Prev.marking	Worked Transiotions	Cur. marking	MarkType	Level
1	0	M0(1,0,0,0,0,0,0,0,0,0,0)		M0(1,0,0,0,0,0,0,0,0,0,0)	ROOT	0
2	1	M0(1,0,0,0,0,0,0,0,0,0,0)	T1	M1(0,1,0,0,0,0,0,0,0,0,0)	INTERNAL	1
3	1	M1(0,1,0,0,0,0,0,0,0,0,0)	T2	M2(0,0,1,0,0,0,0,0,0,0,0)	INTERNAL	2
4	1	M2(0,0,1,0,0,0,0,0,0,0,0)	T3	M3(0,0,0,1,0,0,0,0,0,0,0)	INTERNAL	3
5	1	M3(0,0,0,1,0,0,0,0,0,0,0)	T4	M4(0,0,0,0,1,0,0,0,0,0,0)	INTERNAL	4
6	1	M4(0,0,0,0,1,0,0,0,0,0,0)	T5	M5(0,0,0,0,0,1,0,0,0,0,0)	INTERNAL	5
7	1	M5(0,0,0,0,0,1,0,0,0,0,0)	T6	M6(0,0,0,0,0,0,1,0,0,0,0)	INTERNAL	6
8	1	M6(0,0,0,0,0,0,1,0,0,0,0)	T7	M7(0,0,0,0,0,0,0,1,0,0,0)	INTERNAL	7
9	1	M7(0,0,0,0,0,0,0,1,0,0,0)	T9	M8(0,0,0,0,0,0,0,0,1,0,0)	INTERNAL	8
10	1	M8(0,0,0,0,0,0,0,0,1,0,0)	T12	M9(0,0,0,0,0,0,0,0,0,0,1)	INTERNAL	9
11	1	M9(0,0,0,0,0,0,0,0,0,0,1)	T13	M10(0,0,0,0,0,0,0,0,0,0,1,...	DEADLOCK	10
12	2	M7(0,0,0,0,0,0,0,1,0,0,0)	T10	M11(0,0,0,0,0,0,0,0,0,1,0,...	INTERNAL	8
13	2	M11(0,0,0,0,0,0,0,0,0,1,0,...	T11	M1(0,1,0,0,0,0,0,0,0,0,0)	REPEATED	9
14	3	M5(0,0,0,0,0,1,0,0,0,0,0)	T8	M9(0,0,0,0,0,0,0,0,0,0,1)	REPEATED	6

Рисунок 10 – Табличное описание дерева достижимости

Для проверки корректности созданной сети Петри необходимо повторно рассмотреть ее свойства.

Сеть является ограниченной, поскольку количество фишек в сети не превышает некоторого числа *K* (в данной сети количество фишек равно 1).

Сеть является безопасной, так как число фишек не превышает числа *K*, равного 1.

Сеть Петри обладает свойством сохранения, так как число фишек в сети является постоянным.

Таким образом, корректность работы данной сети Петри по данному алгоритму подтверждает оптимальность его работы.

На рисунке 11 представлена информационная модель передачи данных между основными структурами поликлиники при режиме приема по записи.

На рисунке в виде блоков отмечены все возможные процессы обслуживания пациента в поликлинике, а стрелками обозначены пути и способы обмена той или иной информацией (заявка или данные о пациенте).

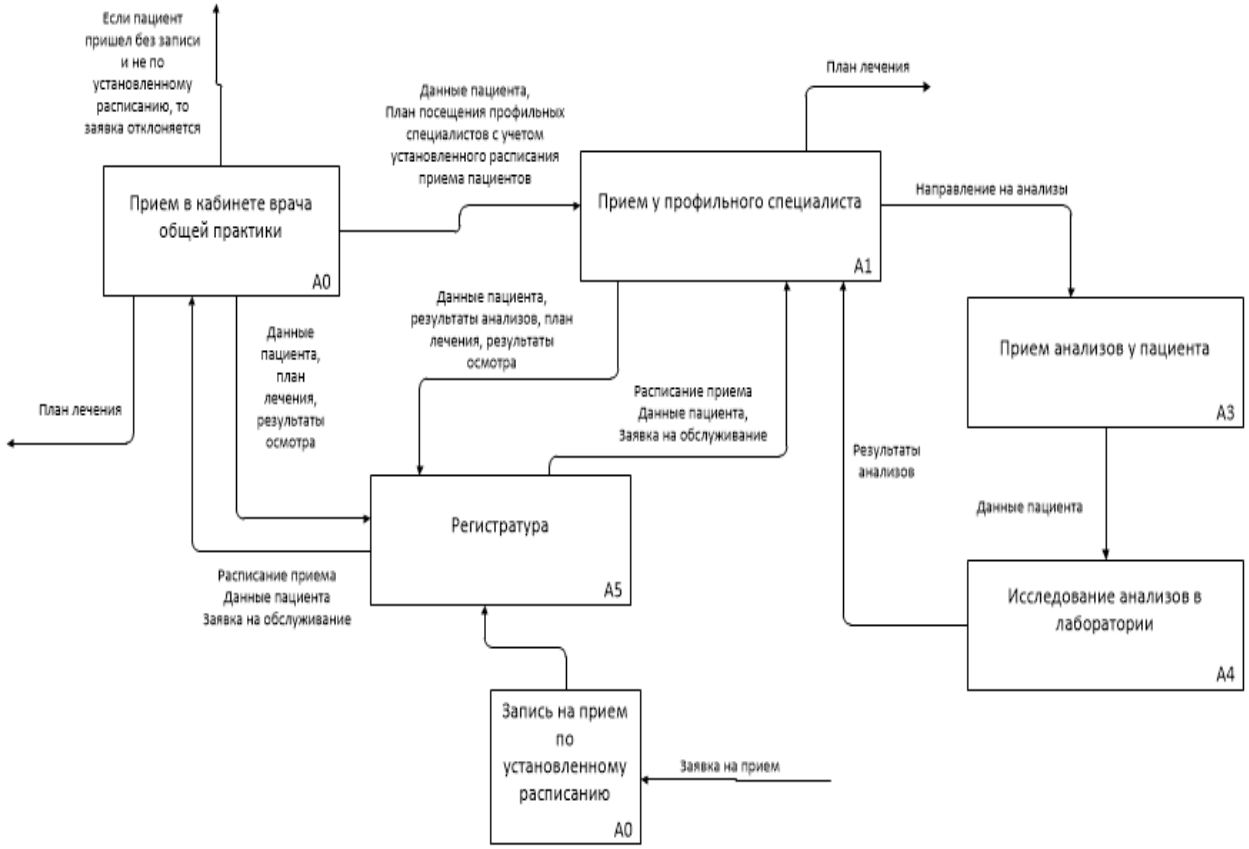

Рисунок 11 – Модель работы медицинского учреждения

Данная модель имеет место, если процесс обслуживания будет реорганизован по алгоритму, приведенному на рисунке 7. В данном случае, пациент, пришедший без предварительной записи, будет в любом случае записан на прием в данный момент времени, если есть свободный специалист, либо в другое ближайшее время.

Выводы

Разработанная модель системы массового обслуживания учитывает изменение рабочей нагрузки специалиста в тот или иной момент времени и, в зависимости от степени нагрузки, позволяет ограничить очередь за счет перераспределения заявок на обслуживание в других, более свободных каналах. Чтобы упорядочить процессы сбора информации о пациентах и нагрузках специалистов, а также процессы формирования потоков пациентов на обслуживание, сформулирован алгоритм распределения потоков на

обслуживание в зависимости от установленной нагрузки. По модели и алгоритму созданы сети Петри, свойства сетей подтверждают корректность их работы.

Для упрощения работы с алгоритмом необходима разработка вспомогательного модуля медицинской информационной системы, позволяющий формировать расписания приема специалистов. Данный модуль должен учитывать значения нагрузок врачей для записи новых пациентов на прием.

3 Проектирование модуля расписания приема пациентов

3.1 Разработка средств управления расписанием приема пациентов

Проведенный ранее анализ показал, что модуль расписаний приема пациентов должен выводить статистику потока пациентов, позволяющую базе составлять новые расписания. Разрабатываемая информационная система должна обеспечивать функционирование трех групп пользователей – администратора системы, доктора и сотрудника регистратуры. Краткое описание действующих лиц системы представлено в таблице 5.

Таблица 5 – Краткое описание действующих лиц

Название	Описание деятельности и возможностей
Администратор	Управляет пользователями системы (добавляет и удаляет новых пользователей), совершает авторизацию в системе
Доктор	Совершает авторизацию в системе, просматривает информацию о расписании, статистике приемов, пациентах
Сотрудник регистратуры	Совершает авторизацию в системе, составляет расписания для врачей, просматривают статистику приемов

Таким образом, к системе имеют доступ три группы пользователей. Администратор системы отвечает за управление пользователями. Основная задача доктора – просмотр медицинской информации, связанной с приемом пациентов, расписаниями, а также просмотр статистики. Сотрудник регистратуры занимается непосредственно составлением расписаний, однако ему также может быть доступна функция просмотра статистики. К возможностям администратора модуля относится управление аккаунтами пользователей: добавление и удаление записей.

На рисунке 12 представлена диаграмма вариантов использования, на которой изображены все возможные варианты действий в системе для каждого из актеров.

Рисунок 12 – Диаграмма вариантов использования

Далее рассмотрен один из наиболее важных вариантов использования – «Добавление записи». Актер, выполняющий данный вариант использования является сотрудником регистратуры. Актер открывает окно программы, предназначенное для добавления новой записи на прием, которая включает в себя данные о пациенте, данные о враче и его профиле. После заполнения данных о пациенте, актер нажимает копку «Добавить». Введенная информация должна иметь правильные типы данных. Если данные введены не верно, то должна появиться соответствующая ошибка. Данные добавляются в таблицу учета пациентов и в таблицу с расписанием одновременно. Если данные о пациенте уже существуют в базе (то есть пациент однажды уже был на приеме), то данные добавляются только в таблицу с расписанием. Если

пациент обратился впервые, то данные о нем добавляются в таблицу «Пациенты». Время и дата приема являются изменяемыми величинами или могут быть пустыми, но для того, чтобы составить итоговое расписание, их необходимо ввести.

Другой важной функцией данного модуля является «Просмотр статистики». Функция заключается в том, что сотрудник медицинской организации просматривает последнюю актуальную информацию о приемах у конкретного врача. Особенно важно в статистике учесть такие атрибуты, как причины обращений и количество приемов пациента, а также профиль специалиста, к которому пациенты записывались на прием. Эти характеристики, помимо общего количества обращений, можно применить для определения рабочей нагрузки на конкретного врача.

Актеры, имеющие доступ к данной функции – сотрудник регистратуры и врач. Актер открывает окно просмотра статистики и выбирает в выпадающем списке профиль врача, пациентов которого он хочет увидеть. После выбора профиля должен появиться список пациентов, обращавшихся к специалистам данного профиля на прием за последнее время. Также актер может выбрать конкретный временной промежуток, в который приходило определенное количество пациентов. Помимо профиля возможен выбор пациентов, пришедших на прием с конкретной причиной, а также вывод пациентов, пришедших на первичный или повторный прием.

Для того чтобы смоделировать поведение разрабатываемой системы, требуется не только представить процессы изменений ее состояний, но и детально показать особенности алгоритмической и логической реализации действий, которые система выполняет. Для моделирования подобных действий требуется построение диаграммы деятельности, которая показывает пошаговые действия актера системы и реакцию системы на эти действия. Для каждого конкретного действия обычно требуется построение отдельной диаграммы. В рамках разработки данной системы представлено две

диаграммы для наиболее важных вариантов использования – «Добавление записи» и «Просмотр статистики».

На рисунке 13 изображена диаграмма деятельности для варианта использования «Добавить запись», доступная сотруднику регистратуры.

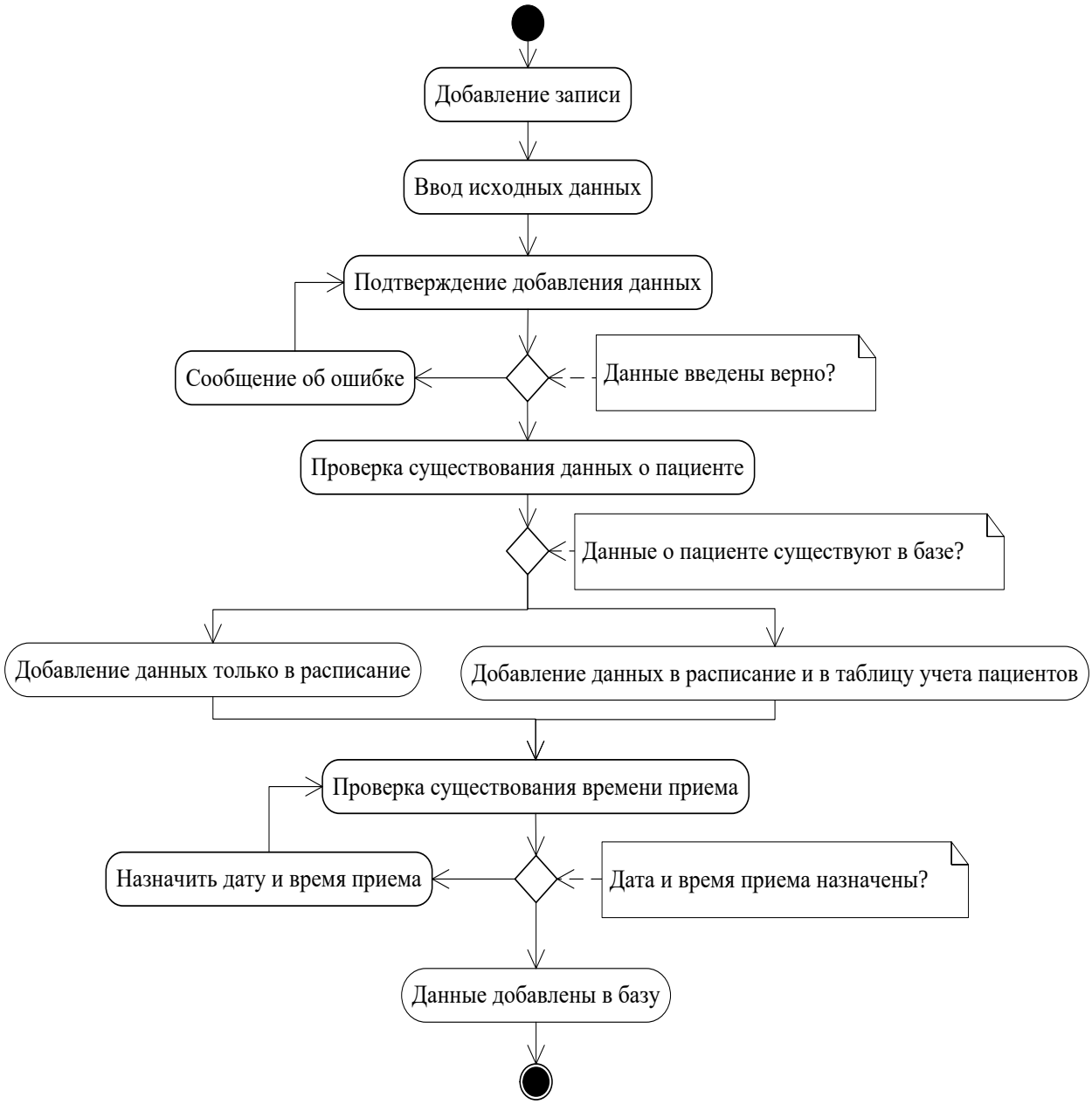

Рисунок 13 – Диаграмма деятельности для варианта использования «Добавить запись»

На рисунке 14 изображена диаграмма деятельности для варианта использования «Просмотр статистики», доступная как сотруднику

регистратуры, так и доктору. На данном рисунке вывод статистики по приемам в медицинском учреждении осуществляется по профилю врача.

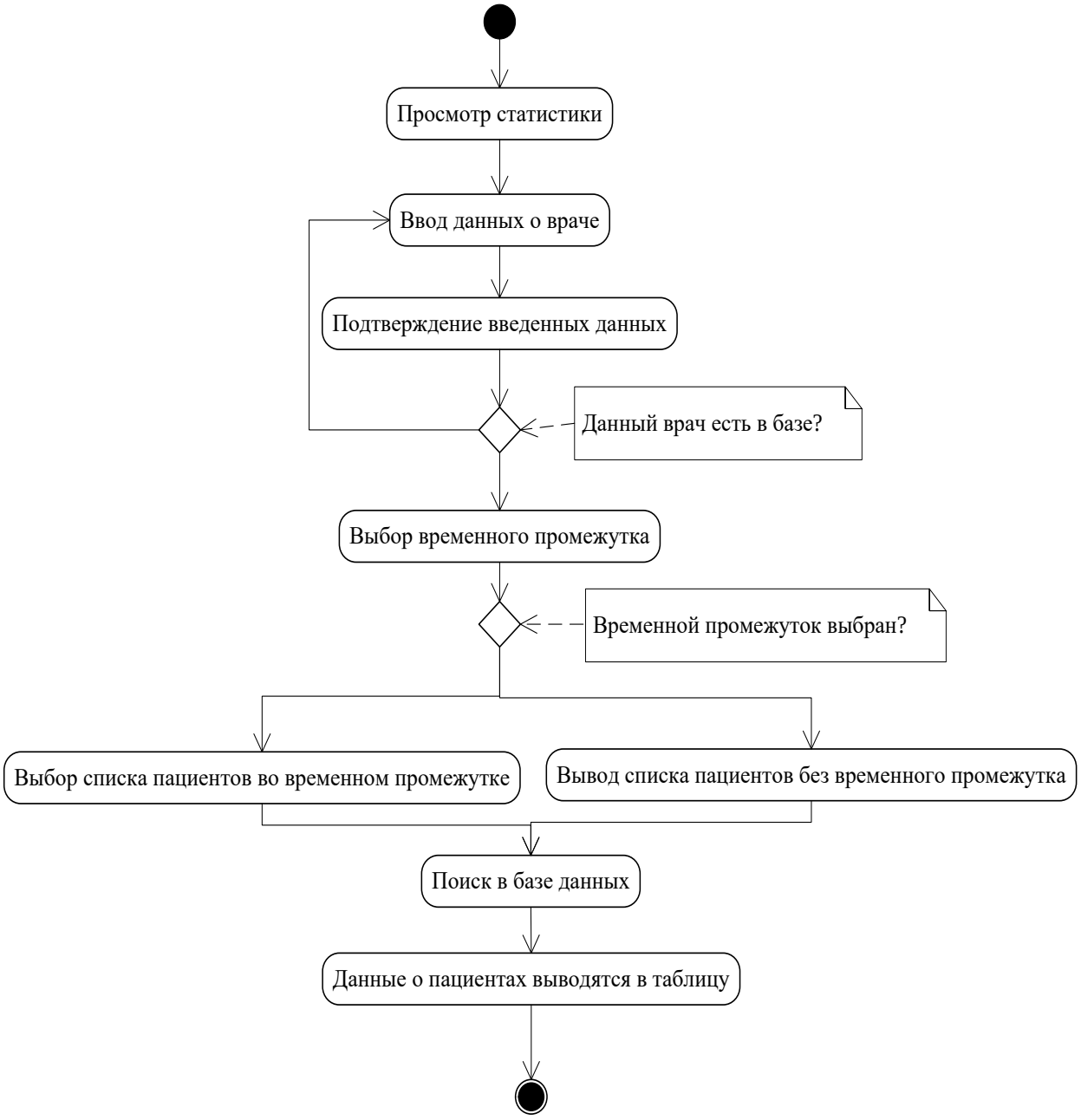

Рисунок 14 – Диаграмма деятельности для варианта использования «Просмотр статистики»

После разработки требований к проектируемой системы, определения вариантов использования и пользователей системы необходимо разработать базу данных.

3.2 Проектирование базы данных

Особенностью всех МИС, приведенных в главе 1, является их проектирование на основе реляционных баз данных. Как было отмечено ранее, медицинские информационные системы осуществляют хранение как структурированной, так и частично структурированной или вовсе неструктурированной информации. Реляционные базы вполне подходят для хранения такого рода данных, однако имеют свои недостатки. Главная сложность заключается в составлении сложных запросов к таблицам.

Так как типовые МИС разрабатываются с использованием реляционных баз данных, то целесообразнее делать разработку дополнительного модуля расписаний в реляционной СУБД. Однако не исключен вариант переноса всей системы на объектно-ориентированную базу данных. Объектно-ориентированные СУБД обладают определенными преимуществами перед реляционными. Главным их преимуществом является возможность хранения данных, не имеющих жесткую структуру [17].

Разрабатываемая база данных строится с использованием системы управления базы данных Microsoft SQL Server. Построение базы данных проходит в несколько этапов: создание концептуальной модели, логической и физической модели.

Построение концептуальной модели заключается в определении объектов, их характеристик и связей между ними. В процессе создания концептуальной модели формулирован набор сущностей и их характеристик. Список сущностей и их краткие описания представлены в таблице 6.

Таблица 6 – Описание сущностей базы данных

Имя сущности	Описание
Доктор	Сущность содержит сведения в докторе
Пациент	Сущность содержит первичные сведения о пациенте
Кабинет	Сущность хранит сведения о кабинетах, в которых ведется прием

Продолжение таблицы 6

Имя сущности	Описание
Проблема	Сущность отображает данные об основных проблемах, с которыми обращаются пациенты
Прием	Сущность хранит данные о числе приемов (первичный или повторный)
Аккаунт	Сущность содержит логины и пароли всех пользователей системы
Расписание	Сущность отображает данные о расписаниях приемов пациентов

Атрибуты всех сущностей и их описания приведены в таблицах.

В таблице 7 описаны атрибуты сущности «Доктор». Сущность «Доктор» содержит необходимые сведения о конкретном специалисте медицинского учреждения: имя, профиль, сведения о средней нагрузке. Сущность «Доктор» привязана к сущностям «Аккаунт» и «Кабинет» посредством соответствующих внешних ключей.

Таблица 7 – Сущность «Доктор»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
ID_Doctor	Идентификатор доктора	Целое		Нет	PK	3
Name_Last Name_Patr o	Фамилия, Имя, Отчество	Текст	128	Нет		Иванов Иван Иванович
Profile	Профиль	Текст	128	Нет		Терапевт
ID_Cabinet	Номер кабинета	Целое		Нет	FK	303
ID_Account	Идентификатор аккаунта	Целое		Нет	FK	34
Workload	Степень нагрузки врача (в процентах)	Действ ительн ое		Да		24

В таблице 8 приведены атрибуты сущности «Пациент». Данная сущность содержит основную информацию о пациенте: имя, фамилия, отчество, дата рождения, номер страхового полиса, номер телефона пациента.

Таблица 8 – Сущность «Пациент»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
ID_Patient	Идентификатор доктора	Целое		Нет	РК	3
Last_name	Фамилия	Текст	128	Нет		Иванов
Name	Имя	Текст	128	Нет		Иван
Patronymic	Отчество	Текст	128	Нет		Иванович
Birthday	Дата рождения	Дата		Нет		30.11.1996
Policy	Номер страхового полиса	Целое		Нет		5848300086 9000181
Telephone	Номер телефона пациента	Целое		Нет		8960329396 3

В таблице 9 описаны атрибуты сущности «Аккаунт». Сущность «Аккаунт» предназначена для хранения информации о паролях и логинах сотрудников медицинского учреждения. Также сущность содержит такие атрибуты, как дата инициализации аккаунта и тип аккаунта (доктор, сотрудник регистратуры или администратор).

Таблица 9 – Сущность «Аккаунт»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
Login	Логин	Текст	10	Нет		Admin
ID_Account	Идентификатор аккаунта	Целое		Нет	РК	7
Password	Пароль	Текст	8	Нет		Sc215iL5
Initialization_Date	Дата инициализации	Дата		Нет		20.09.2020
Account_Type	Тип аккаунта	Текст		Нет		Doctor

В таблице 10 приведены атрибуты сущности «Прием». Данная сущность хранит информацию о двух типах приема: первичный или повторный. Сущность связана с сущностью «Расписание» посредством соответствующего первичного ключа.

Таблица 10 – Сущность «Прием»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
ID_Appointment	Идентификатор приема	Целое		Нет	PK	1
Name_Appointment	Наименование	Текст	128	Нет		Первичный

В таблице 11 описаны атрибуты сущности «Кабинет». Сущность «Кабинет» предназначена для сбора информации о функционирующих кабинетах конкретного медицинского учреждения и привязана к записи о конкретном враче с помощью соответствующего первичного ключа.

Таблица 11 – Сущность «Кабинет»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
ID_Cabinet	Номер кабинета	Целое		Нет	PK	303
Floor	Этаж	Целое		Нет		3
Phone	Номер телефона в кабинете	Текст	4	Нет		22-33

В таблице 12 описаны атрибуты сущности «Проблема». Сущность содержит основную информацию о проблемах, с которыми пациент может обращаться к врачу. При этом, в информация о проблеме может быть записана как в общем виде (например, «Хроническое заболевание»), так и с пояснением конкретной проблемы. Данная информация может быть полезна позднее для сбора статистики по заболеваемости.

Таблица 12 – Сущность «Проблема»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
ID_Problem	Номер проблемы	Целое		Нет	PK	3
Name_Problem	Наименование проблемы	Текст		Нет		Консультация

Продолжение таблицы 12

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
Description	Описание проблемы	Текст	128	Нет		79
Complexity	Сложность проблемы	Действи тельное	10	Нет		3

В таблице 13 описаны атрибуты сущности «Расписание». Сущность является основной в базе, так как содержит информацию о числе приемов для конкретного врача в конкретный отрезок времени. Данная сущность в основном состоит из внешних ключей других сущностей базы. Также сущность «Расписание» может использоваться в создании хранимых процедур, направленных на сбор статистики по приемам пациентов.

Таблица 13 – Сущность «Расписание»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
ID_Doctor	Идентификатор доктора	Целое		Нет	FK	3
ID_Patient	Идентификатор пациента	Целое		Нет	FK	7
Time	Время приема	Время		Нет		12.50
Date	Дата приема	Дата		Нет		20.09.2020
ID_Cabinet	Идентификатор кабинета	Целое		Нет	FK	303
ID_Recepti on	Идентификатор приема	Целое		Нет	FK	2
ID_Proble m	Идентификатор проблемы	Целое		Нет	FK	203

В таблице 14 приведены атрибуты сущности «Сотрудник регистратуры». Сущность содержит информацию о сотрудниках регистратуры и, как и сущность «Доктор», привязана к сущности «Аккаунт». Сущность содержит наиболее важные для работы модуля атрибуты: уникальный номер

сотрудника, идентификатор аккаунта, привязанный к таблице «Аккаунт» и инициалы сотрудника.

Таблица 14 – Сущность «Сотрудник регистратуры»

Имя столбца	Содержательное описание	Тип данных	Размер-ность	Возможность значения Null	Роль	Пример
ID_Employee	Идентификатор доктора	Целое		Нет	PK	4
ID_Account	Идентификатор аккаунта	Целое		Нет	FK	10
Initials	Фамилия, Имя, Отчество	Текст	128	Нет		Маркина Елена Петровна

На основании имеющихся данных можно построить логическую и физическую модель базы данных, которые представлены далее в виде наглядных схем [18].

3.3 Построение логической модели базы данных

Логической моделью следует называть модель разрабатываемой базы данных, которая не привязана к конкретной СУБД. При построении логической модели выделяются конкретные сущности, атрибуты и связи между ними. На данном этапе разработки базы данных можно назначить типы данных для атрибутов. Также на этапе разработки логической модели можно выделить первичные и внешние ключи, которые будут связываться с другими сущностями.

Помимо обозначения сущностей и атрибутов необходимо обозначить связи между сущностями. Основные типы связей в данной модели – «один ко многим» и «один к одному». Например, связь «один к одному» присутствует у сущностей «Доктор» и «Аккаунт», или «Сотрудник регистратуры» и «Аккаунт», так как один сотрудник медицинского учреждения имеет только один аккаунт.

Логическая модель базы данных модуля составления и просмотра расписаний приема врачей, являющейся частью медицинской информационной системы, спроектирована в среде Erwin Data Modeler [19].

На рисунке 15 изображена схема логической модели разрабатываемой базы данных.

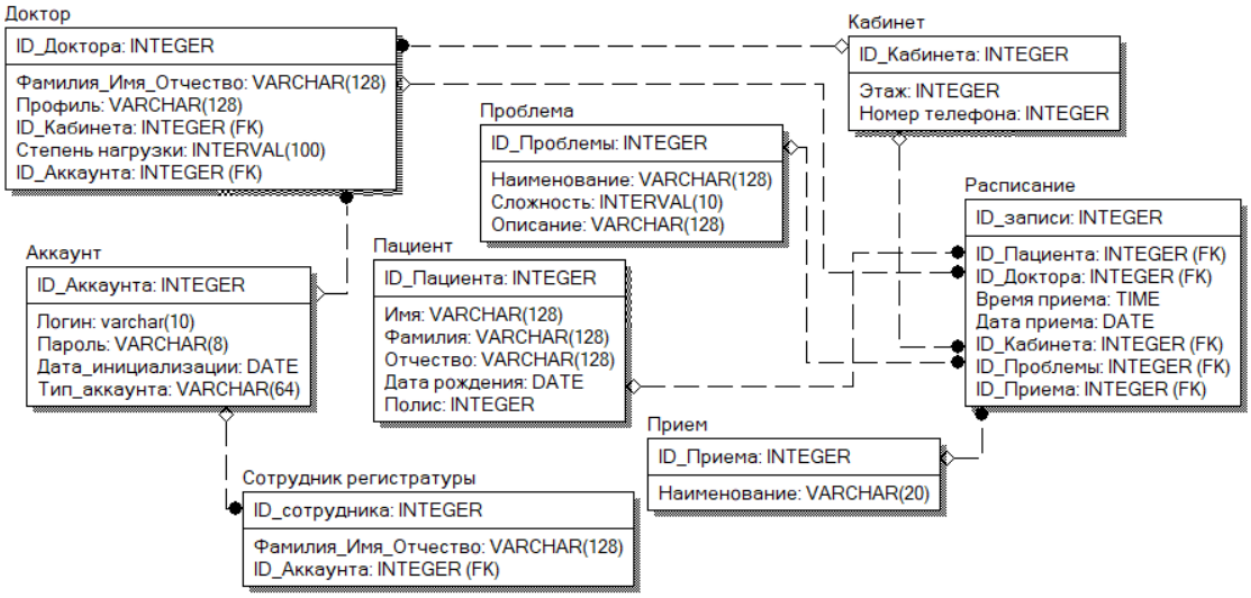

Рисунок 15 – Логическая модель базы данных

Следующим этапом проектирования является построение физической модели базы данных.

3.4 Построение физической модели базы данных

Физической моделью базы данных принято называть созданную логическую модель, но интерпретированную в рамках конкретной системы управления базами данных (СУБД).

На рисунке 16 изображена физическая модель базы данных проектируемой системы.

Следующим шагом после создания модели базы данных и непосредственно самой базы в выбранной СУБД необходимо создать все нужные хранимые процедуры, триггеры и представления, которые будут работать в данной базе [20].

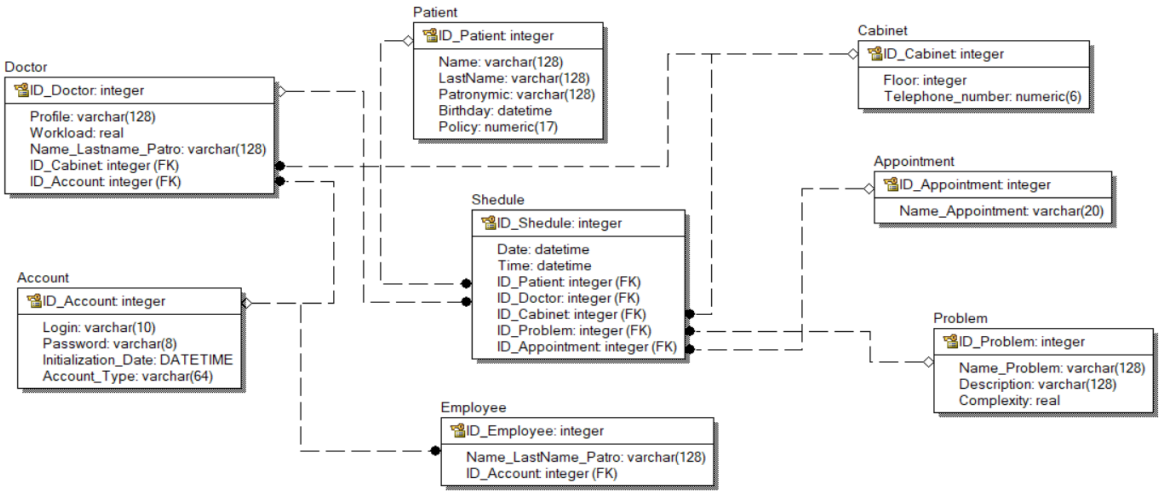

Рисунок 16 – Физическая модель базы данных

Основные хранимые процедуры направлены на работу с добавлением, удалением и изменением данных в созданных таблицах, однако также необходимо создать хранимую процедуру для сбора статистики по поступившим за последнее время пациентам и средней нагрузке врачей. Скрипт создания базы данных приведен в приложении Б.

3.5 Разработка хранимых процедур, функций и триггеров базы данных

Помимо стандартных функций добавления, изменения и удаления информации необходима разработка специализированных функций. В рамках разработки прототипа модуля расписаний выделено три необходимых функции.

Хранимая процедура «Вывод статистики» осуществляет выборку из таблиц «Пациенты», «Доктор» и «Расписание» всех пациентов, пришедших на прием к врачу конкретного профиля в установленные промежутки времени. При создании процедуры особенно важно учесть такие атрибуты пациентов, как число приемов и причина обращения, так как выборку также можно осуществлять по этим атрибутам. Процедура создается с целью получения статистических данных о заболеваемости и оценки загруженности специалистов конкретного профиля.

Хранимая процедура «Добавление записи» необходима для добавления в базу данных о пациенте и записи пациента на прием. Если пациент уже однажды посещал врача в данном учреждении, то добавляются только данные о записи на новый прием. Проверку существования пациенте можно проверять по существованию данных о пациенте в базе, а также о значении поля «Прием» в таблице «Расписание». Если прием повторный, то предполагается, что пациент уже есть в базе. Время и дата приема могут ставиться как сразу при записи, так и позже.

Хранимая процедура «Просмотр расписания» заключается в выборке расписания для конкретного врача в конкретную дату. Атрибуты, по которым осуществляется выборка – идентификатор врача и дата.

Кроме описанных процедур в базе должны быть сохранены хранимые процедуры для администратора базы, например, процедура добавления нового пользователя. Администратор добавляет все данные о сотруднике или враче, назначает им логин и пароль. Если администратор добавляет в базу врача, то следует учесть атрибут «Степень нагрузки». По умолчанию администратор присваивает этому атрибуту значение 0. Также администратор может соответственно изменять и удалять данные о сотрудниках медицинского учреждения, имеющих к данному модулю МИС доступ. Кроме того, администратор может добавлять данные о кабинетах.

Для изменения загруженности врача необходим триггер изменения значения атрибута «Степень загрузки», кроме того, возможно создание хранимой процедуры вычисления степени рабочей нагрузки доктора в зависимости от дня недели [21]. Скрипты хранимых процедур приведены в приложении В.

3.6 Проектирование программных средств

После этапа проектирования базы данных следует этап разработки программного обеспечения. Для разработки программных средств модуля медицинской информационной системы используются методики объектно-

ориентированного программирования. Данный подход к созданию программ предполагает представление программы в виде набора взаимосвязанных классов и методов. В данном случае классы содержат информацию о частях интерфейса программы (кнопках, таблицах, окнах для ввода информации). Также классы содержат методы подключения программы к серверной части приложения – базе данных, откуда в окна программы выводится информация. Кнопки привязаны к методам обработки полученной информации и ее обновления в базе данных. В результате разработки должна появиться диаграмма классов создаваемого приложения.

Диаграмма классов – это структурная диаграмма, создаваемая в рамках языка UML, которая наглядно показывает иерархию классов в разрабатываемой системе, а также отображает их атрибуты, методы, интерфейсы и взаимосвязи между ними.

На рисунке 17 изображена диаграмма классов для разрабатываемого модуля.

Поскольку архитектура приложения клиент-серверная, классы в клиентском приложении предназначены именно для отображения информации, считываемой с базы данных, включая хранимые процедуры добавления, изменения и удаления информации, а также выборки определенной информации [22].

На диаграмме видно, что точкой входа в программу является класс Program. Данный класс открывает окно авторизации, который обеспечивает пользователю доступ к функционалу в зависимости от того, каким конкретно актером системы является пользователь. То есть, если пользователь войдет в систему, используя логин и пароль сотрудника регистратуры, то ему будут доступны функции добавления и изменения записей в расписании. Однако существует функционал, доступный пользователям нескольких типов. Так, окна просмотра статистики могут быть доступны одновременно и сотруднику регистратуры, и доктору.

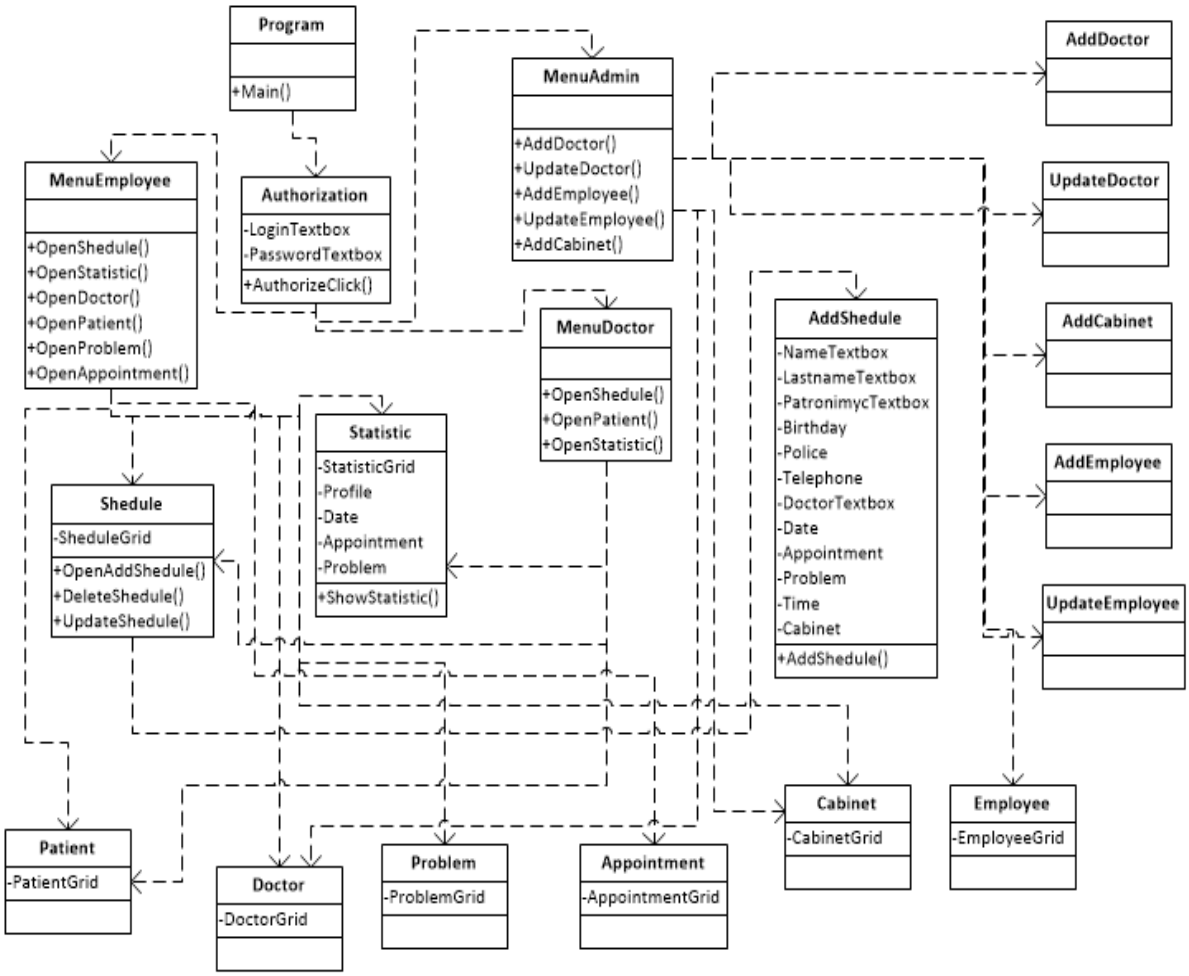

Рисунок 17 – Диаграмма классов

Далее рассмотрены классы, выделенные на диаграмме подробнее. В таблице 15 приведены все созданные классы программы и их функциональное назначение.

Данная диаграмма отображает функционал сотрудника регистратуры и врача медицинского учреждения.

Таблица 15 – Описание классов программы

Имя класса	Функциональное назначение
Authorization	Реализация окна авторизации пользователя в системе.
MenuDoctor	Реализация главного окна доктора.
MenuEmployee	Реализация главного окна сотрудника регистратуры.
MenuAdmin	Реализация главного окна администратора.

Продолжение таблицы 15

Имя класса	Функциональное назначение
Patient	Реализация окна с таблицей, отображающей список пациентов.
Doctor	Реализация окна с таблицей, отображающей список врачей.
Shedule	Класс реализует окно отображения расписания.
AddShedule	Класс реализует окно и функцию добавления новой записи и пациента.
Statistic	Класс реализует хранимую процедуру базы данных «Просмотр статистики».
AddDoctor	Класс реализует функцию добавления доктора.
AddEmployee	Класс реализует окно добавления сотрудника.
AddCabinet	Класс реализует окно добавления кабинета.
UpdateEmployee	Класс реализует окно изменения данных о сотруднике.
UpdateDoctor	Класс реализует окно изменения данных о докторе.
Appointment	Класс реализует окно отображения списка приемов.
Problem	Класс реализует окно отображения списка жалоб.
Cabinet	Класс реализует окно отображения списка кабинетов.
Employee	Класс реализует окно отображения списка сотрудников.

Проектирование классов осуществлено с использованием среды Microsoft Visual Studio на языке C#, позволяющей реализовать разработку программы с использованием объектно-ориентированного подхода [23].

Разработка интерфейса модуля медицинской информационной системы произведена также в среде Microsoft Visual Studio с использованием интерфейса Windows.Forms.

Для того чтобы понять взаимосвязь представленных классов, необходимо построить диаграмму компонентов. В качестве физических компонентов данном случае представлены классы программы. Диаграмма

компонентов проектируемого модуля, представленного на языке моделирования UML представлена на рисунке 18.

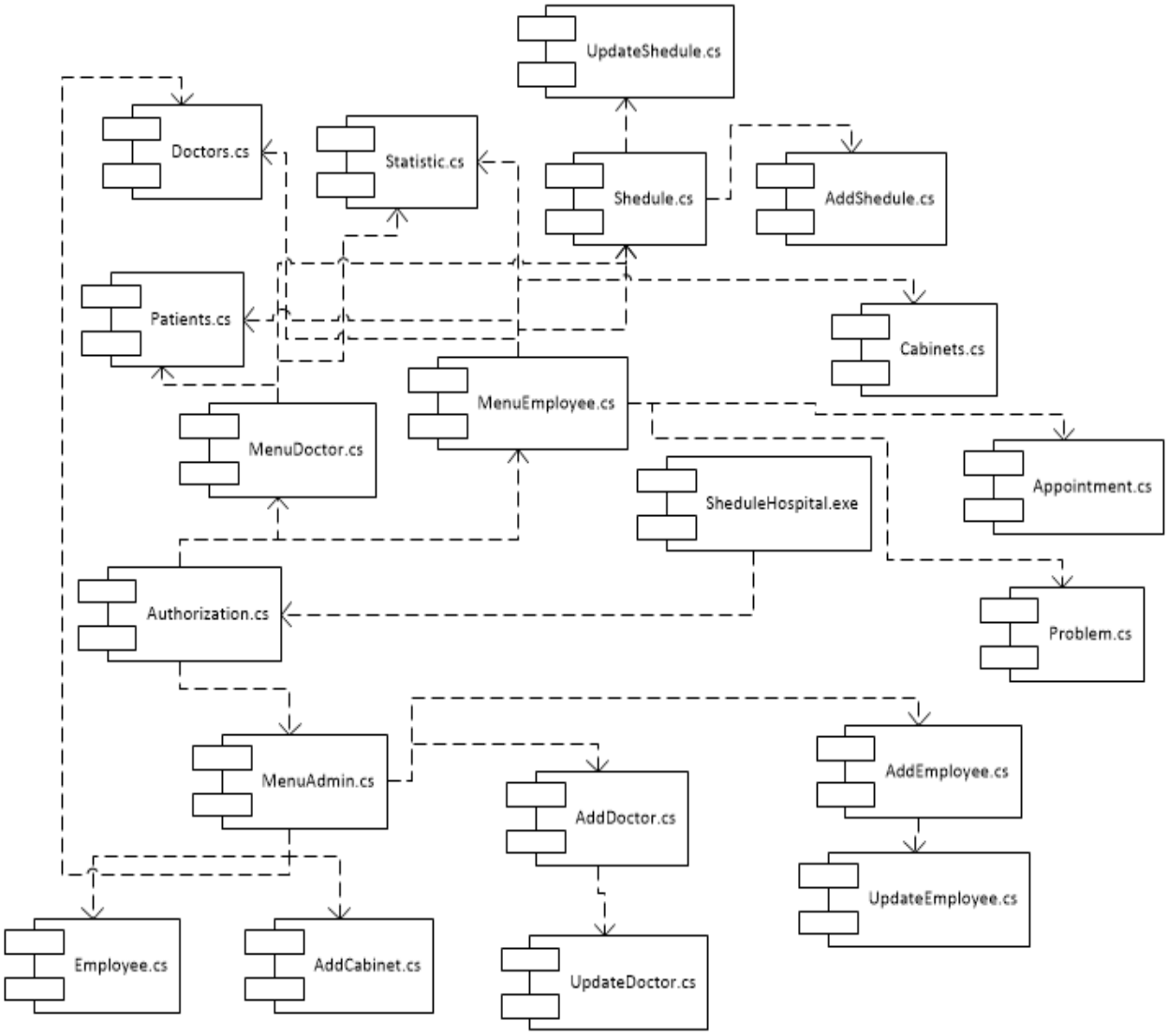

Рисунок 18 – Диаграмма компонентов

Компонент SheduleHospital.cs, изображенный на диаграмме компонентов, является точкой входа в программу. Все компоненты диаграммы представляют собой классы приложения, реализующие интерфейс пользователя. Далее приведено краткое описание компонентов:

– Authorization.cs – реализация окна авторизации пользователя в системе;

– MenuDoctor.cs – реализация главного окна программы, в котором отображается меню (для доктора);

– MenuEmployee.cs – реализация главного окна программы, в котором отображается меню (для сотрудника);

– MenuAdmin.cs – реализация окна главного окна программы, в котором отображается меню (для администратора);

– Shedule.cs – реализация окна просмотра расписания, окно также содержит кнопки открытия меню добавления новой записи, окно доступно для сотрудника регистратуры и для врача;

– Doctors.cs – реализация окна, отображающего данные о специалистах медицинского учреждения, окно доступно для сотрудника регистратуры;

– Patients.cs – реализация окна, которое отображает таблицу с информацией о пациентах поликлиники, окно доступно для сотрудника регистратуры и врача;

– Cabinets.cs – реализация окна, отображающего данные о количестве кабинетов медицинского учреждения, окно доступно для сотрудника регистратуры;

– Statistic.cs – реализация окна, отображающего статистические данные о приемах, окно доступно для сотрудника регистратуры и врача;

– AddShedule.cs – реализация окна добавления данных о новом пациенте и записи на прием, окно доступно для сотрудника регистратуры;

– UpdateShedule.cs – реализация окна изменения данных о расписании приема пациентов, доступ к окну имеет сотрудник регистратуры;

– Appointment.cs – реализация окна, в котором отображается список приемов (первичный или повторный);

– Problem.cs – реализация окна, в котором отображается список распространенных причин обращения к врачу;

– Employee.cs – реализация окна, которое отображает таблицу с информацией о сотрудниках поликлиники;

– AddDoctor.cs – реализация окна, в котором осуществляется добавление данных о докторе в базу, окно доступно для администратора базы;

– AddEmployee.cs – реализация окна, в котором осуществляется добавление данных о сотруднике регистратуры в базу, окно доступно только для администратора;

– AddCabinet.cs – реализация окна, в котором осуществляется добавление данных о кабинетах, окно доступно только для администратора;

– UpdateDoctor.cs – реализация окна, в котором осуществляется изменение данных о докторе, доступ к окну имеет только администратор;

– UpdateEmployee.cs – реализация окна, в котором осуществляется изменение данных о сотруднике, доступ к окну есть только у администратора.

Клиентское приложение спроектировано на языке программирования C# с использованием платформы .NET Framework в среде программирования Microsoft Visual Studio. Серверная часть создана с использованием СУБД Microsoft SQL Server [24]. Диаграмма развертывания системы приведена на рисунке 19.

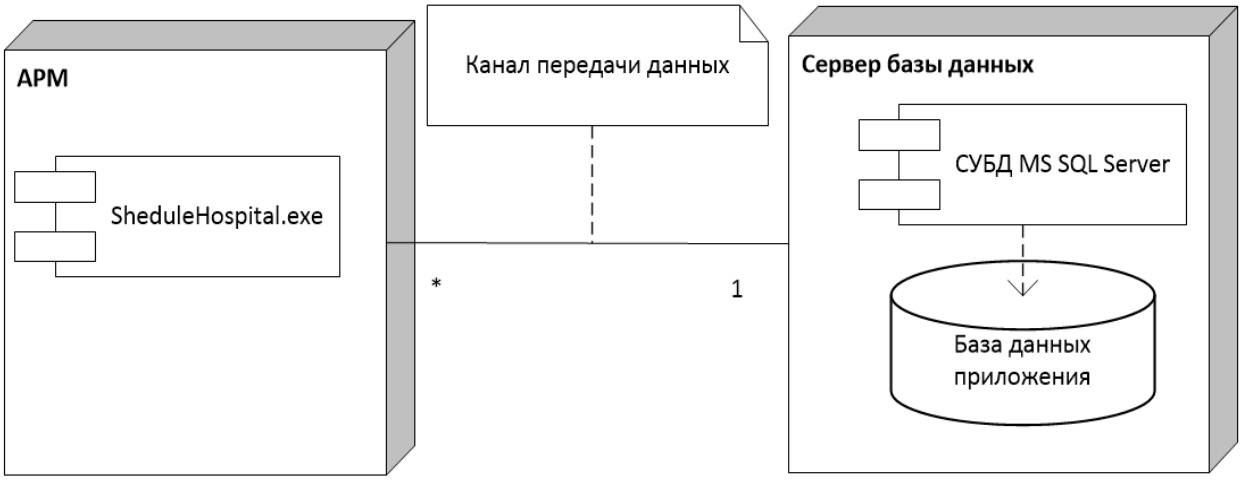

Рисунок 19 – Диаграмма развертывания системы

На диаграмме показано, что к автоматизированному рабочему месту (АРМ), привязано клиентское приложение SheduleHospital.exe. Каждое рабочее место соединено с серверной частью, где находится база данных, называемая Shedule_Hospital. Автоматизированные рабочие места соединены с базой данных связью «один ко многим», так как группа компьютеров с

программой подключена к одной единой базе. После этапа разработки необходимо провести тестирование функционала приложения.

3.7 Тестирование программы

Для проверки возможностей функционала и его точности и правильности работы проводят функциональное тестирование приложения.

Функциональные возможности программы выделены из вариантов использования.

Для данного модуля информационной системы проведено один из двух известных типов функционального тестирования – тестирование «черного ящика». Данный тип тестирования предназначен для проверки корректности работы функционала программного обеспечения без доступа к коду программы.

При разработке модуля формирования расписаний необходимо провести проверку следующих функций:

- ввод данных о пациенте и записи (необходимо учесть возможность первичной и повторной записи на прием);
- просмотр статистики по приемам пациентов на основе данных о первичном или повторном приеме (осуществление выборки из базы данных записей по значению атрибута «Наименование приема» в таблице «Прием»;
- просмотр статистики по приемам пациентов на основе данных о профиле врача (осуществление выборки из базы данных по значению атрибута «Профиль», расположенному в таблице «Доктор»;
- просмотр статистики по приемам пациентов на основе данных о причине обращения (осуществление выборки из базы данных по значению атрибута «Наименование проблемы» в таблице «Проблема»;
- просмотр расписания на текущий день (осуществление выборки из базы данных по инициалам доктора и конкретной дате приема) [25].

При тестировании также важно учитывать возможность защиты от неправильного ввода данных, например, даты. Этот факт должен быть учтен на этапе разработки интерфейса (создание специальных окон для ввода дат).

В таблице 16 приведены результаты проведенных тестов. Скриншоты результатов приведены на рисунках в приложении Г.

Таблица 16 – Тестирование функционала

№ п/п	Назначение теста	Описание теста	Результат
1	Добавление новой записи о повторном приеме	Добавить данные о записи и данные о пациенте, который записывается на повторный прием. Ожидается добавление повторной записи о приеме в таблицу «Расписание».	Тест выполнен успешно. Результат на рисунках Г.1, Г.2.
2	Добавить данные о первичном приеме	Добавить данные о записи пациента на первичный прием. Ожидается добавление новых записей в таблицы «Пациент» и «Расписание».	Тест выполнен успешно. Результат на рисунке Г.3, Г.4.
3	Просмотр статистики по приему	Для введенных данных о типе приема, и установленном временном промежутке, осуществляется выборка из таблицы «Расписание»	Тест выполнен успешно. Результат на рисунке Г.5.
4	Просмотр статистики по приему у доктора	Для введенных данных о докторе конкретного профиля и установленном временном промежутке осуществляется выборка из таблицы «Расписание»	Тест выполнен успешно. Результат на рисунке Г.6.
5	Просмотр статистики по причине обращения	Для введенных данных об установленном временного промежутка и причинах обращения осуществляется выборка из таблицы «Расписание»	Тест выполнен успешно. Результат на рисунке Г.7.

Данные тесты направлены на проверку работы хранимых процедур базы данных модуля составления расписаний приема пациентов в медицинском учреждении. Проведенные тесты показали точность предложенного функционала и его простоту для пользователей системы.

Выводы

В данной главе описаны основные этапы разработки модуля медицинской информационной системы, от требований пользователей, до проектирования диаграмм классов и компонентов приложения. Архитектура данного программного средства построена с учетом особенностей исследованной предметной области. Данный модуль является вспомогательным средством в процессе управления обслуживанием пациентов медицинской организации.

По итогам разработки проведено тестирование основного функционала программы, вследствие чего можно сделать вывод об эффективности разработанного средства в процессе формирования расписаний приемов.

Заключение

В работе проведен анализ процессов обслуживания пациентов в медицинском учреждении. В ходе анализа установлено, что проблемой текущих процессов приема являются неконтролируемые потоки очередей к врачу. В качестве вспомогательного средства управления процессами обслуживания пациентов в поликлинике выступают медицинские информационные системы. Проведен обзор существующих систем, выделены их достоинства и недостатки. Установлено, что для решения проблемы необходимо реорганизовать процесс распределения пациентов в очередях.

Для решения проблемы проведен анализ известных моделей управления обслуживанием пациентов и алгоритмов управления потоками пациентов на обслуживание в медицинском учреждении. В ходе работы были разработаны следующие модели и алгоритмы:

- модель распределения пациентов в очереди на прием у специалиста медицинской организации, которая, в отличие от известных, учитывает изменение рабочей нагрузки на специалистов и контролирует длину очереди на обслуживание в зависимости от нагрузки;
- алгоритм формирования потоков пациентов на приеме у специалиста в медицинской организации, который, в отличие от известных, позволяет учесть возможность контроля нагрузки на специалистов и равномерно распределить число пациентов в очередях;
- модуль медицинской информационной системы, реализующего формирование расписаний приема пациентов.

Спроектированный модуль реализован с использованием среды разработки Microsoft Visual Studio, база данных создана в системе управления базами данных Microsoft SQL Server. Проведенное тестирование показало правильность работы функционала приложения и его полноту.

Список использованных источников

1. Гулиева, И. Ф. Вопросы эффективности информационных технологий в медицине / И. Ф. Гулиева, Е. В. Рюмина, Я. И. Гулиев // Врач и информационные технологии. – 2011. – № 5. – С. 6 – 18.

2. Азанов, В. Г. Структурно-функциональная модель управления потоками пациентов региональной консультативной поликлиники / В. Г. Азанов // Системы и средства информации. – 2016. – №1. – С. 13 – 29.

3. Абухай, Т. М. Моделирование, анализ и прогнозирование процессов оказания кардиологической помощи в стационаре / Т. М. Абухай, С. В. Ковальчук, М. А. Балахонцева, А. В. Бухановский // Известия высших учебных заведений. Приборостроение. – №8. – 2018. – С. 730 – 733.

4. Гулиева, И. Ф. Медицинские информационные системы: затраты и выгоды / И. Ф. Гулиева, Е. В. Рюмина, Я. И. Гулиев // Врач и информационные технологии. – 2009. – № 3. – С. 4 – 18.

5. Гусев, А. В. Решение проблемы очередей пациентов в Карельской медицинской информационной системе / А. В. Гусев, Ф. А. Романов // Врач и информационные технологии. – 2008. – №4. – С. 46 – 53.

6. Гусев, А. В. Обзор рынка комплексных медицинских информационных систем / А. В. Гусев // Врач и информационные технологии. – 2009. – № 6. – С. 4 – 17.

7. Таташев, А. Г. Система массового обслуживания с переменной интенсивностью входного потока / А. Г. Таташев // Автоматика и телемеханика. – 1995. – №12. – С. 78 – 84.

8. Храмкова, М. А. Модели и алгоритмы управления процессами обслуживания пациентов медицинского учреждения : дис. … канд. техн. наук : 05.13.10 – Астрахань 2014. – 150 с.

9. Хачумов, В. М. Моделирование работы лечебного учреждения как системы массового обслуживания / В. М. Хачумов, С. В. Погодин // Искусственный интеллект и принятие решений. – 2010. – №1. – С. 49 – 56.

10. Прокофьева, Е.С. Совершенствование процесса обслуживания в медицинских учреждениях на основе математических моделей потоков пациентов // Высшая школа экономики [Электронный ресурс]. URL: https://aspirantura.hse.ru/data/2017/11/08/1158354102/2017-10-27-prokofyeva.pdf (дата обращения: 1.02.2020).

11. Ковалевский, С. С. Модели СМО в системах медицинского обслуживания / С. С. Ковалевский, В. Б. Шувалов // Точная наука. – 2017. – №10. – С. 49 – 51.

12. Холодов, А. Ю. Математическое моделирование систем массового обслуживания с циклической дисциплиной прохождения заявок : дис. … канд. техн. наук : 05.13.18 – Астрахань 2009. – 160 с.

13. Жерновый, Ю. В. Модели системы M/M/n/r с переключением режимов обслуживания в моменты изменения числа заявок / Ю. В. Жерновый // Информационные процессы. – 2010. – № 1. – С. 68 – 77.

14. Коновалов, А. А. Метод оптимизации потоков пациентов при организации мероприятий по оказанию санаторно-курортной медицинской помощи / А. А. Коновалов // Организация здравоохранения и общественное здоровье. – 2011. – №5. – С. 27 – 37.

15. Чернявский, А. Л. Алгоритм для решения комбинаторных задач, основанный на методе неявного перебора / А. Л. Чернявский // Автоматика и телемеханика. – 1972. – №2. – С. 98 – 108.

16. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. – М.: МЦНМО. 2000. – 960 с.

17. Малых, В. Л. Объектно-реляционный подход к построению хранилища данных / В. Л. Малых, А. Н. Калинин, Т. Ш. Юсуфов // Программные системы: Теоретические основы и приложения. – 2017. – №3. – С. 169 – 187.

18. Гладков, Л. А. Генетические алгоритмы: учебное пособие / Л. А. Гладков, В. В. Курейчик, В. М. Курейчик. – М.: Физматлит, 2004. – 407 с.

19. Емелин, И. В. О стандартизации структуры электронных медицинских данных / И. В. Емелин, Б. В. Зингерман, Г. С. Лебедев // Информационно-измерительные и управляющие системы. – 2010. – №12. – С. 8 – 24.

20. Хомоненко, А. Д. Базы данных. Учебник для вузов / А. Д. Хомоненко, В. М. Цыганков, М. Г. Мальцев. – ДМК. : Пресс, 2013. – 232 с.

21. Маклаков, С. В. BPwin и ERwin: CASE-средства для разработки информационных систем. / С. В. Маклаков. – Издательство: Диалог-МИФИ, 1999. – 132 с.

22. Система управления базами данных SQL Server // Программирование на языке C#, платформа .NET Framework + [Электронный ресурс]. URL: https://professorweb.ru/my/sqlserver/2012/level1/1_1.php/ (дата обращения 25.05.2020).

23. Иванова, Г. С. Объектно-ориентированное программирование. / Г. С. Иванова, Е. К. Пугачев. – М.: Издательство МГТУ имени Н.Э. Баумана, 2001. – 319 с.

24. Бахтизин, В. В. Технология разработки программного обеспечения / В. В. Бахтизин, Л. А. Глухова, – Минск БГУИР, 2010. – 408с.

25. Котляров, В. П. Основы тестирования программного обеспечения. / В. П. Котляров, Т. В. Коликова – М.: Бином. Лаборатория знаний, 2006. – 285 с.

Приложение А
ГЛОССАРИЙ
(справочное)

Медицинское учреждение – организация, осуществляющая деятельность в области здравоохранения или оказания медицинских услуг.

Медицинская организация – то же, что и медицинское учреждение.

Диспансеризация – набор мероприятий, включающий профилактический медицинский осмотр и дополнительные обследования, проводится с целью оценки состояния здоровья населения.

Медицинская информационная система – система автоматизации документооборота для медицинских учреждений.

Система массового обслуживания (СМО) – система, производящая обслуживание поступающих в нее требований с помощью обслуживающих приборов.

Заявка (СМО) – запрос на обслуживание.

Канал обслуживания (СМО) – обслуживающий прибор системы массового обслуживания.

Алгоритм – набор последовательных операций, описанный в формальных или математических терминах, направленный на решение определенных задач.

Приложение Б
СКРИПТ СОЗДАНИЯ БАЗЫ ДАННЫХ
(рекомендуемое)

```
CREATE TABLE Account
(
      ID_Account integer NOT NULL ,
      Login varchar(10) NOT NULL ,
      Password varchar(8) NOT NULL ,
      Initialization_Date DATETIME NOT NULL ,
      Account_Type varchar(64) NOT NULL
)
go
ALTER TABLE Account
      ADD CONSTRAINT XPK Account PRIMARY KEY NONCLUSTERED (ID_Account ASC)
go
CREATE TABLE Appointment
(
      ID_Appointment integer NOT NULL ,
      Name_Appointment varchar(20) NOT NULL
)
go
ALTER TABLE Appointment
      ADD CONSTRAINT XPK Appointment PRIMARY KEY NONCLUSTERED (ID_Appointment
ASC)
go
CREATE TABLE Cabinet
(
      ID_Cabinet integer NOT NULL ,
      Floor integer NOT NULL
)
go
ALTER TABLE Cabinet
      ADD CONSTRAINT XPK Cabinet PRIMARY KEY NONCLUSTERED (ID_Cabinet ASC)
go
CREATE TABLE Doctor
(
      ID_Doctor integer NOT NULL ,
      Profile varchar(128) NOT NULL ,
      Workload real NOT NULL ,
      Name_Lastname_Patro varchar(128) NOT NULL ,
      ID_Cabinet integer NULL ,
      ID_Account integer NULL
)
go
ALTER TABLE Doctor
      ADD CONSTRAINT XPKDoctor PRIMARY KEY NONCLUSTERED (ID_Doctor ASC)
go
CREATE TABLE Employee
(
      ID_Employee integer NOT NULL ,
      Name_LastName_Patro varchar(128) NOT NULL ,
      ID_Account integer NULL
)
go
ALTER TABLE Employee
      ADD CONSTRAINT XPK Employee PRIMARY KEY NONCLUSTERED (ID_Employee ASC)
go

CREATE TABLE Patient
(
      ID_Patient integer NOT NULL ,
      Name varchar(128) NOT NULL ,
      LastName varchar(128) NOT NULL ,
      Patronymic varchar(128) NOT NULL ,
      Birthday datetime NOT NULL ,
      Policy numeric(17) NOT NULL
)
```

```
go
ALTER TABLE Patient
      ADD CONSTRAINT XPKPatient PRIMARY KEY NONCLUSTERED (ID_Patient ASC)
go
CREATE TABLE Problem
(
      ID_Problem integer NOT NULL ,
      Name_Problem varchar(128) NOT NULL ,
      Description varchar(128) NOT NULL ,
      Complexity real NOT NULL
)
go
ALTER TABLE Problem
      ADD CONSTRAINT XPKProblem PRIMARY KEY NONCLUSTERED (ID_Problem ASC)
go
CREATE TABLE Shedule
(
      ID_Shedule integer NOT NULL ,
      Date datetime NOT NULL ,
      Time datetime NOT NULL ,
      ID_Patient integer NULL ,
      ID_Doctor integer NULL ,
      ID_Cabinet integer NULL ,
      ID_Problem integer NULL ,
      ID_Appointment integer NULL
)
go
ALTER TABLE Shedule
      ADD CONSTRAINT XPKShedule PRIMARY KEY NONCLUSTERED (ID_Shedule ASC)
go
ALTER TABLE Doctor
      ADD CONSTRAINT R_1 FOREIGN KEY (ID_Cabinet) REFERENCES
Cabinet(ID_Cabinet)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
ALTER TABLE Doctor
      ADD CONSTRAINT R_4 FOREIGN KEY (ID_Account) REFERENCES
Account(ID_Account)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
ALTER TABLE Employee
      ADD CONSTRAINT R_2 FOREIGN KEY (ID_Account) REFERENCES
Account(ID_Account)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
ALTER TABLE Shedule
      ADD CONSTRAINT R_5 FOREIGN KEY (ID_Patient) REFERENCES
Patient(ID_Patient)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
ALTER TABLE Shedule
      ADD CONSTRAINT R_6 FOREIGN KEY (ID_Doctor) REFERENCES Doctor(ID_Doctor)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
ALTER TABLE Shedule
      ADD CONSTRAINT R_7 FOREIGN KEY (ID_Cabinet) REFERENCES
Cabinet(ID_Cabinet)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
```

```
go
ALTER TABLE Shedule
      ADD CONSTRAINT R_8 FOREIGN KEY (ID_Problem) REFERENCES
Problem(ID_Problem)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
ALTER TABLE Shedule
      ADD CONSTRAINT R_9 FOREIGN KEY (ID_Appointment) REFERENCES
Appointment(ID_Appointment)
            ON DELETE NO ACTION
            ON UPDATE NO ACTION
go
```

Приложение В
СКРИПТЫ ХРАНИМЫХ ПРОЦЕДУР
(рекомендуемое)

Листинг В.1 – Код хранимой процедуры MakeShedule на языке программирования баз данных Transact SQL

```
CREATE PROCEDURE [dbo].[MakeShedule] (@name int , @lastname int, @patro int,
@birth date, @policy numeric(17), @datesh date, @timesh time, @iddoctor int,
@idcabinet int, @idproblem, @idappointment int, @telephone)
      AS
      DECLARE @i int
      BEGIN TRY
            SET @i = (SELECT idpatient FROM Patient WHERE( @policy LIKE
      Policy));
            IF (EXISTS(SELECT Policy FROM Patient WHERE (@policy LIKE
      Policy)))
                  BEGIN
                  BEGIN TRANSACTION
                        INSERT INTO Shedule (Date, Time, ID_Patient,
                        ID_Doctor, ID_Cabinet,ID_Problem,ID_Appointment)
                        VALUES (@datesh, @timesh, @i, @iddoctor,
                        @idcabinet,@idproblem,@idappointment);
                  COMMIT
                  END
            ELSE
                  BEGIN
                  BEGIN TRANSACTION
                  INSERT INTO Patient (Name, Lastname, Patronimyc, Birthday,
                  Policy, Telephone) VALUES (@name, @lastname, @patronimyc,
                  @birthday, @policy, @telephone);
                  SET @i= SCOPE_IDENTITY();
                  INSERT INTO Shedule (Date, Time, ID_Patient, ID_Doctor,
                  ID_Cabinet,ID_Problem,ID_Appointment) VALUES (@datesh,
                  @timesh, @i, @iddoctor,
                  @idcabinet,@idproblem,@idappointment);
                  COMMIT
                  END
      END TRY
      BEGIN CATCH
            IF (@@error <>0)
            ROLLBACK
      END CATCH
```

Листинг В.2 – Код хранимой процедуры StatisticProblem на языке программирования баз данных Transact SQL

```
CREATE PROCEDURE [dbo].[StatisticProblem] (@idproblem int, @idappointment
int, @profilevarchar(128), @begin date, @end date)
      AS
      BEGIN TRY
            SELECT * FROM Shedule WHERE ((@idproblem LIKE ID_Problem) AND
      (Date >= @begin) AND(Date <=@end)) ;
            RETURN 0
      END TRY
      BEGIN CATCH
            IF(@@error <> 0)
            ROLLBACK
      END CATCH
```

Листинг В.3 – Код хранимой процедуры Statistic на языке программирования баз данных Transact SQL

```
CREATE PROCEDURE [dbo].[Statistic] (@idproblem int, @idappointment int,
@profile varchar(128), @begin date, @end date)
      AS
      BEGIN TRY
            SELECT * FROM Shedule WHERE ((@idappointment LIKE ID_Appointment)
      AND (Date >= @begin) AND(Date <=@end)) ;
            RETURN 0
      END TRY
      BEGIN CATCH
            IF(@@error <> 0)
            ROLLBACK
      END CATCH
```

Листинг В.4 – Код хранимой процедуры StatisticDoctor на языке программирования баз данных Transact SQL

```
CREATE PROCEDURE [dbo].[StatisticDoctor] (@idproblem int, @idappointment int,
@profile varchar(128), @begin date, @end date)
      AS
      BEGIN TRY
            SELECT * FROM Shedule WHERE (SELECT ID_Doctor FROM Doctor
      WHERE(@profile LIKE Profile)) AND (Date >= @begin) AND(Date <=@end)) ;
            RETURN 0
      END TRY
      BEGIN CATCH
            IF(@@error <> 0)
            ROLLBACK
      END CATCH
```

Приложение Г
ТЕСТИРОВАНИЕ ПРОГРАММЫ
(рекомендуемое)

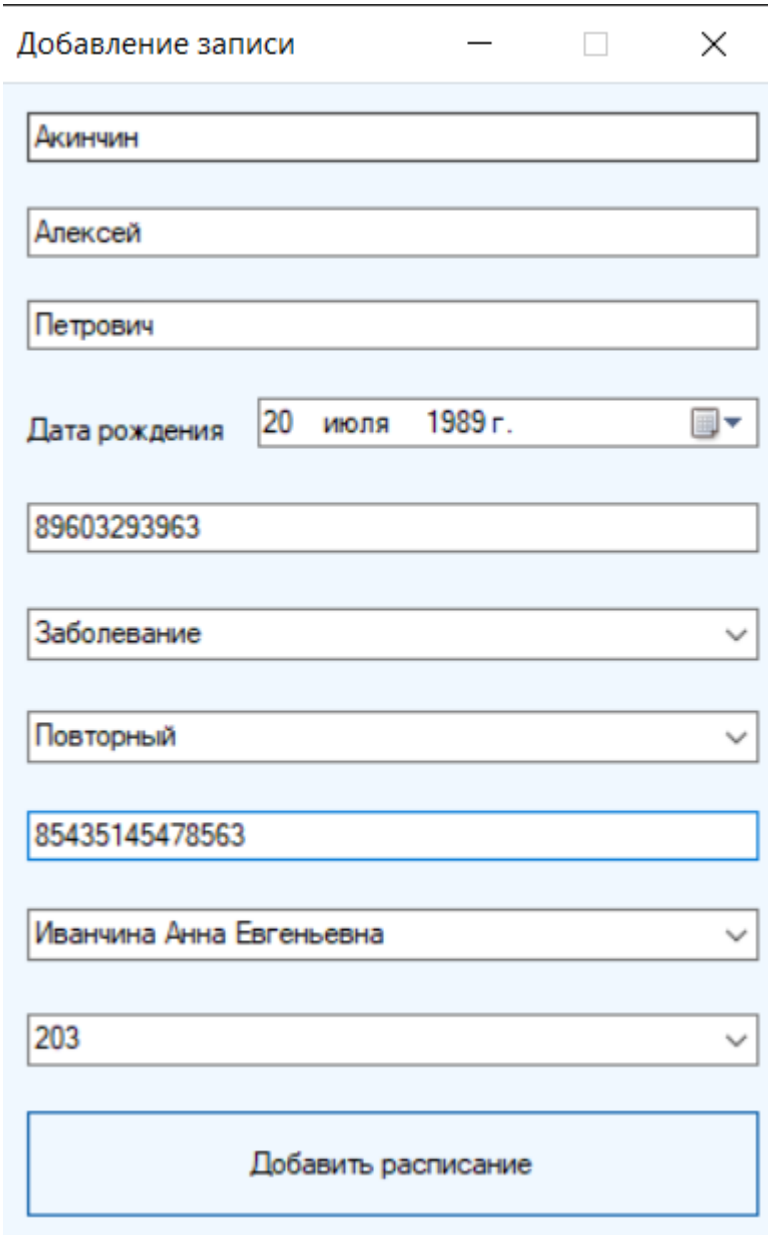

Рисунок Г.1 – Окно добавление записи о повторном приеме

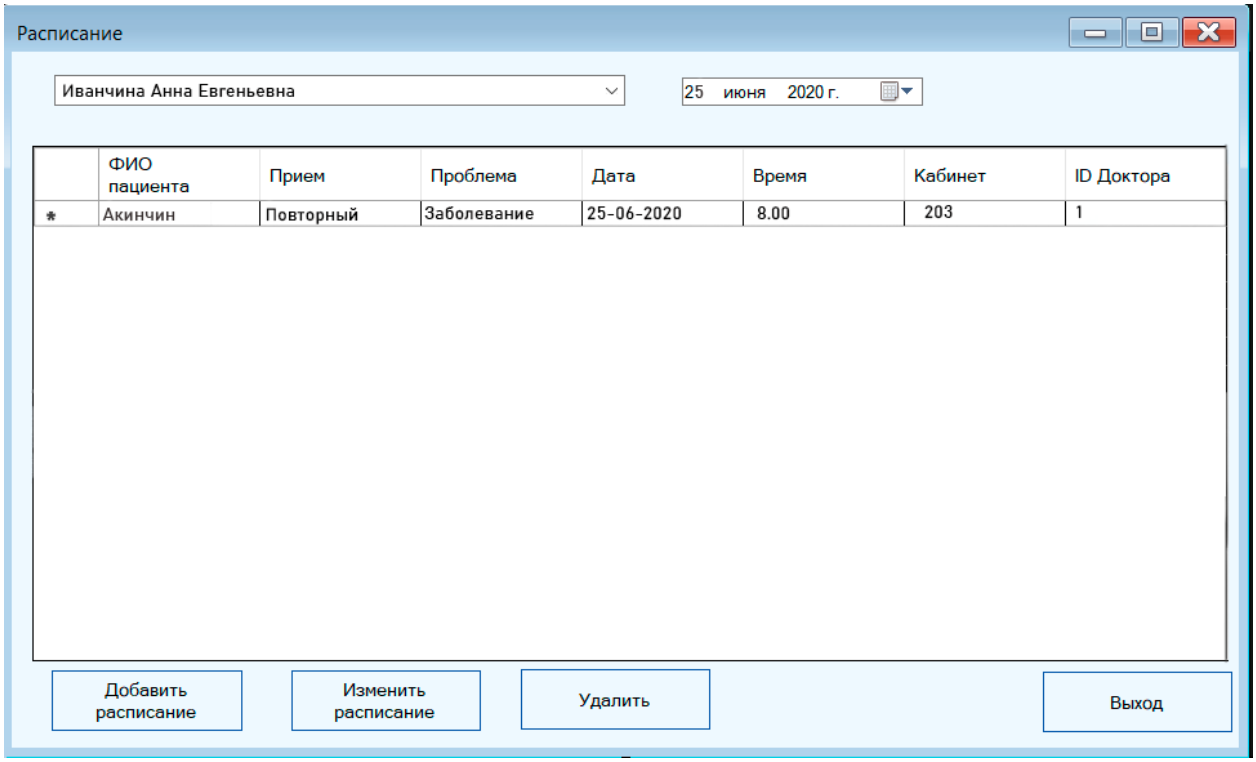

Рисунок Г.2 – Информация о повторном приеме

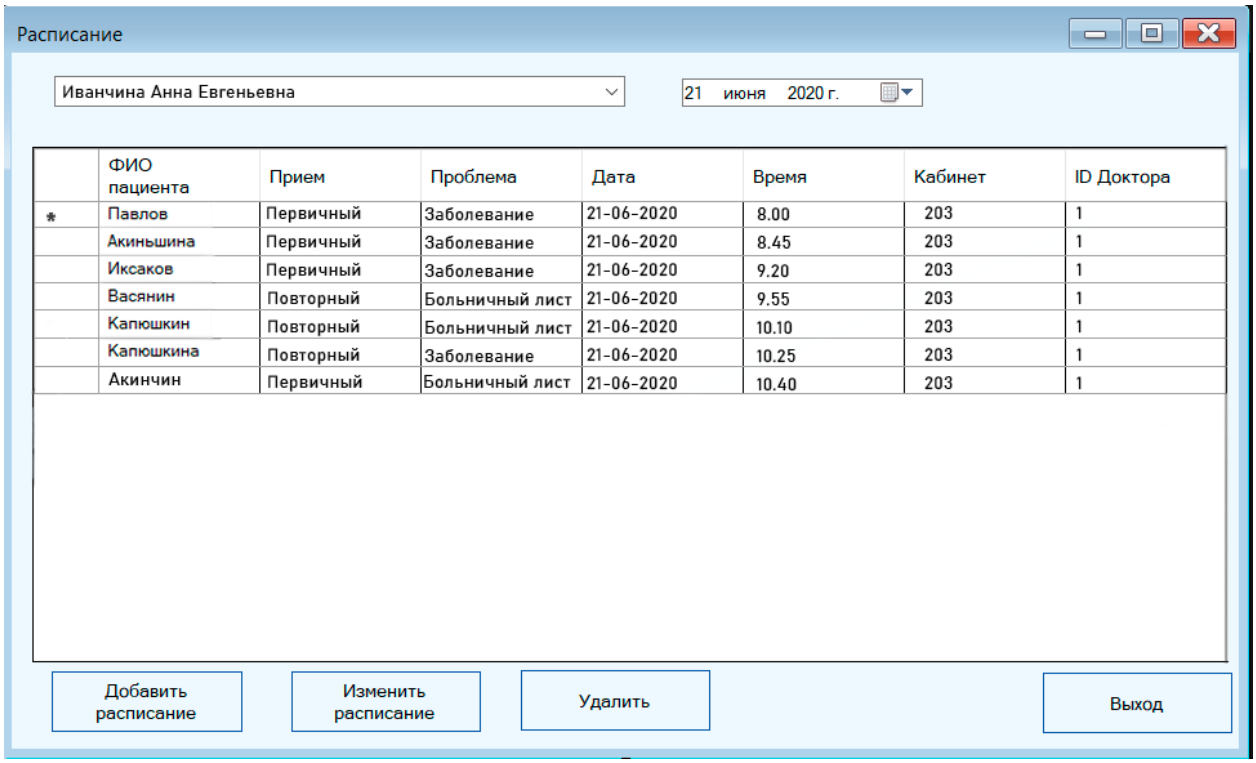

Рисунок Г.3 – Информация о первичном приеме

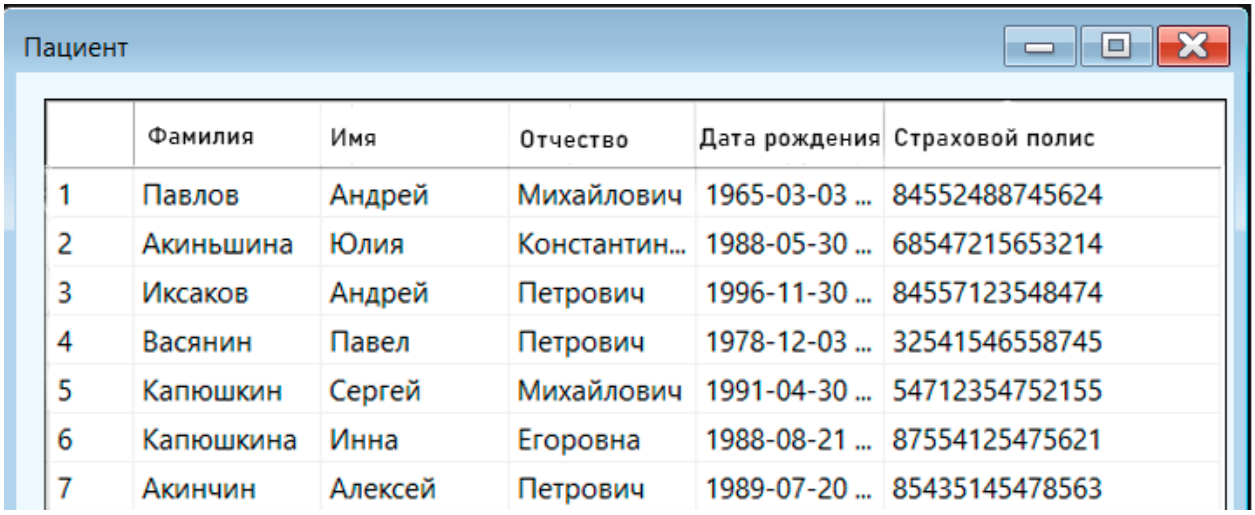

Рисунок Г.4 – Окно со списком пациентов

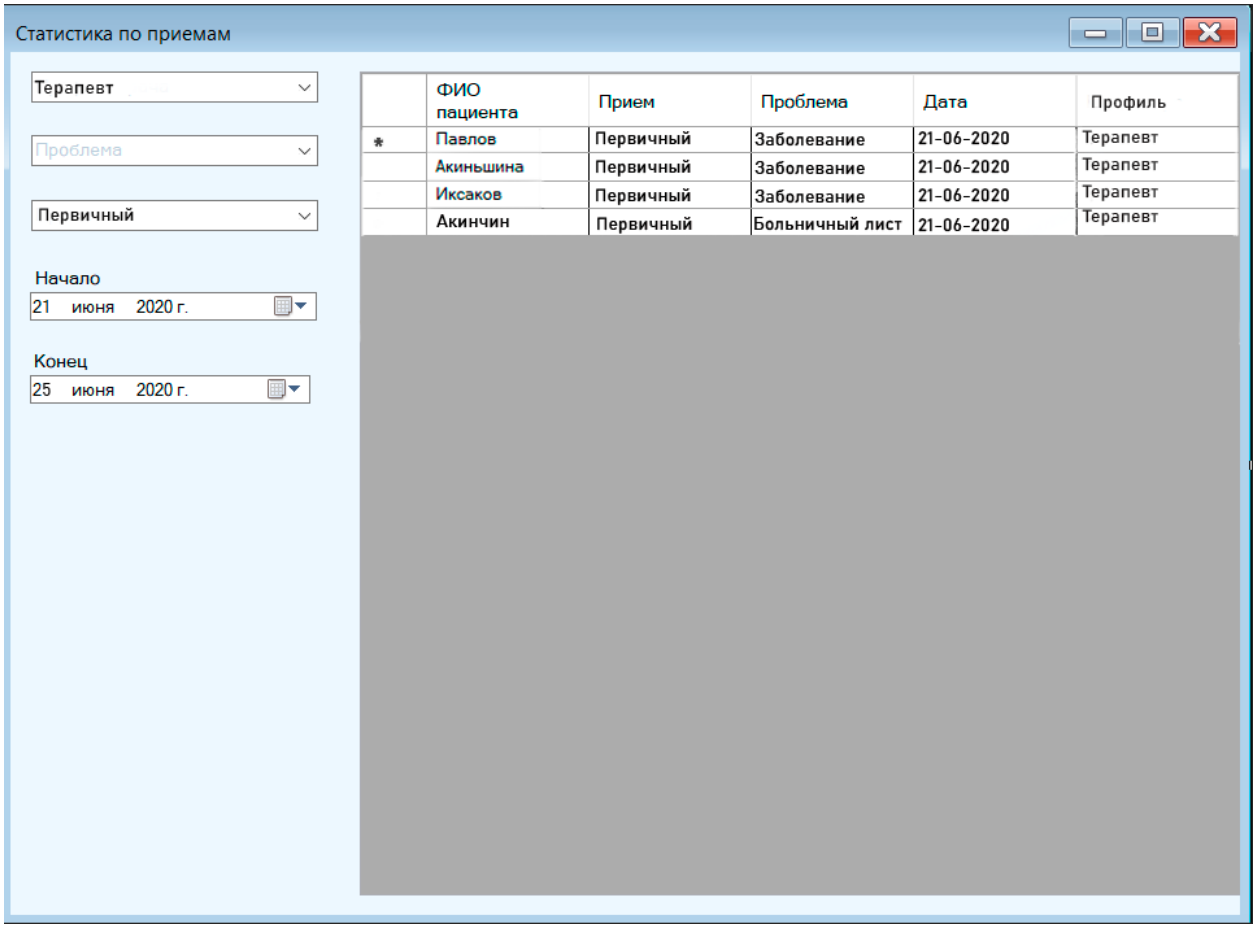

Рисунок Г.5 – Просмотр статистики по типу приема

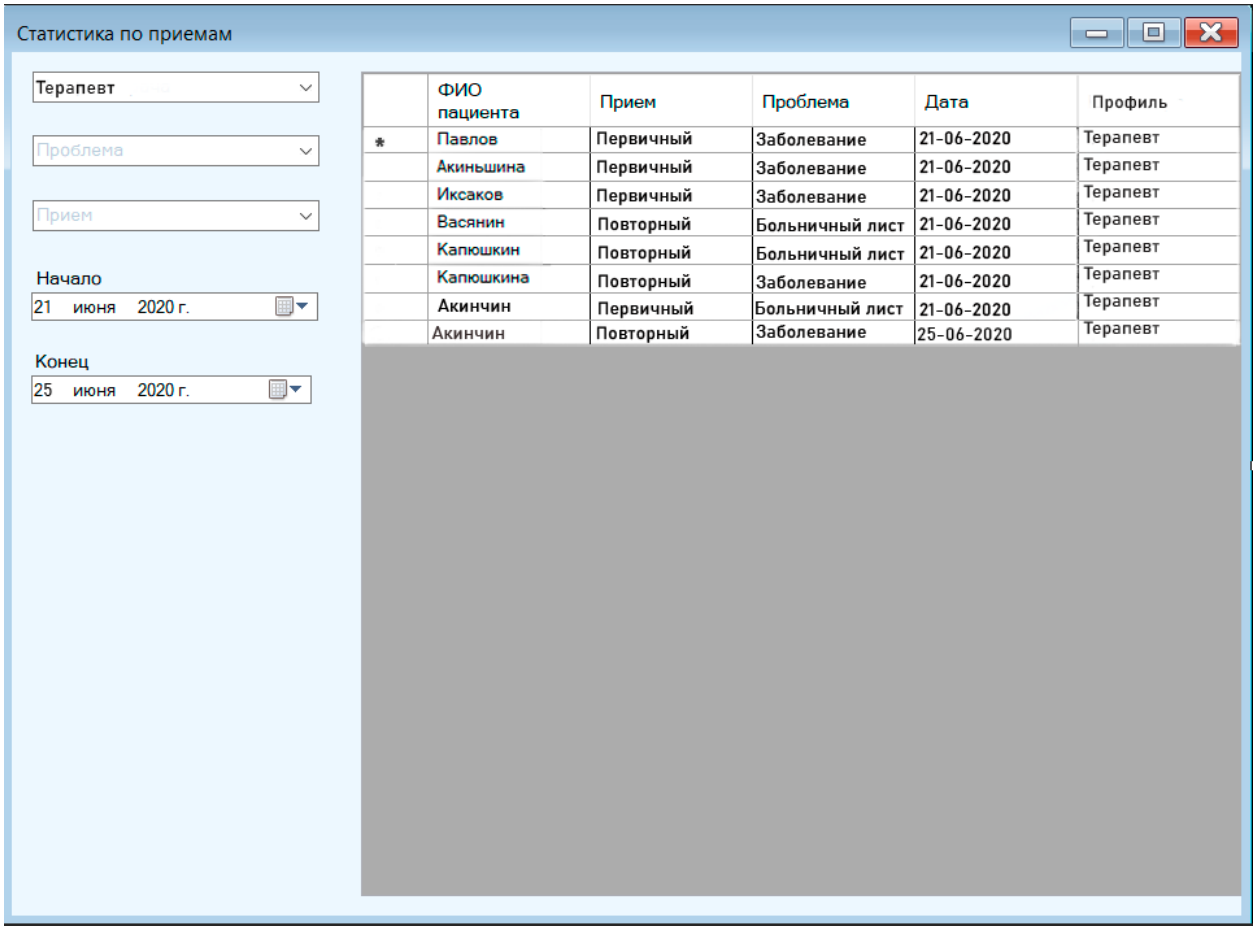

Рисунок Г.6 – Окно просмотра статистики по приемам у врача

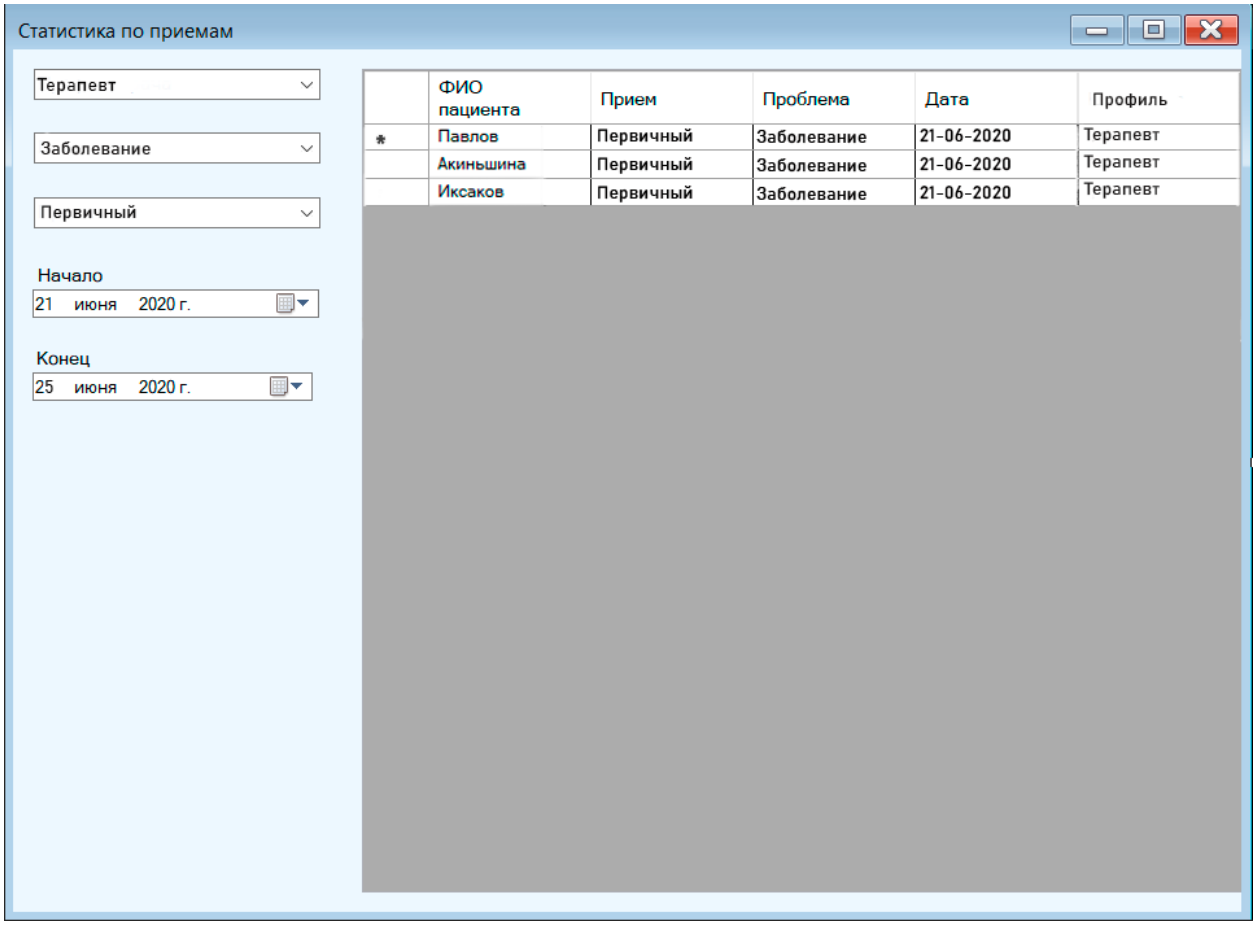

Рисунок Г.7 – Просмотр статистики по типу проблемы