МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Российский технологический университет»
МИРЭА

Физико-технологический институт
Кафедра оптических и биотехнических систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА НА ТЕМУ:

«АВТОМАТИЗАЦИЯ КЛЮЧЕВОГО БИЗНЕС-ПРОЦЕССА ВРАЧА ТЕРАПЕВТА В ГОРОДСКОЙ ПОЛИКЛИНИКЕ НА ОСНОВЕ СПИРАЛЕВИДНОЙ МОДЕЛИ ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ »

Студент:
Дудкин Д.И.

Научный руководитель:
к.т.н., доц. МИРЭА Степанов Д.Ю.

Москва – 2019

Оглавление

Введение

За последние годы компьютерные технологии сильно развиваются и внедряются во все возможные сферы жизни человека. Они имеют большой перечень функциональных возможностей, в то числе и возможность повысить эффективность работы врача лечебного учреждения.

В недалеком прошлом лечебные учреждения - поликлиники и работающие в них врачи имели определенные затруднения при приеме пациентов. Например: очень долго и трудоемко проходил процесс документального оформления приема пациентов; актуальной была проблема потери карты пациента в регистратуре или поиска её у других врачей и другие.

Все это приводило к образованию очереди пациентов на прием к врачу, затруднялась работа врачей, медицинского персонала и в целом работа лечебного учреждения. В связи с чем, остро назрела задача рационализировать механизм документального оформления осмотра больных и механизм приема пациентов в лечебном учреждении, что послужило стимулом к широкому внедрению компьютерных технологий в область здравоохранения. Врачам необходимо, чтобы технические средства и разработанное к ним программное обеспечение были простыми и удобными при использовании, не нарушали привычного стиля работы.

Главная проблема, с которой сталкиваются врачи при приёме пациента – это недостаток времени. Значительное время врача при приеме расходуется на ознакомление с медицинской картой обследуемого. Разработка медицинской базы данных и ее использование в работе врача позволит ускорить данный процесс, более качественно проводить обследование и снизить нагрузку на врача, даст возможность быстрого поиска в базе данных нужной информации о пациенте. Например, посмотреть всю историю болезни, даты приема, проведенные обследования и их результаты и др.

Разработку такого приложения можно произвести в программе MS Access. Данная среда совместима с операционной системой Microsoft Windows, которая используется на большинстве компьютеров медицинских учреждений, что важно для удобства использования. Таким образом, можно сделать вывод о целесообразности и актуальности проведения данной работы.

Цель и задачи

Целью данной работы является разработка программного комплекса для использования на рабочем месте врача-терапевта в виде автоматизированной системы. Программное обеспечение позволит сохранять всю информацию о пациенте, его посещениях, историю болезни, облегчить и ускорить работу врача. Благодаря данной системе снимается проблема с потерей амбулаторных карт пациентов, т.к. вся информация будет храниться в одной базе данных. Чтобы достичь вышеуказанной цели нам нужно реализовать следующие задачи:

- произвести детальный анализ спиралевидной методологии внедрения систем;
- идентифицировать требования и сформулировать список требований;
- произвести проектирование процессов и оргструктуры в моделях AS-IS и TO-BE нотации ARIS VACD и UML AD до 3-4 уровней детализации.

В спиралевидной методологии внедрения систем для каждого витка спирали произвести:

- моделирование разрабатываемых пользовательских интерфейсов;
- проектирование структуры данных и нормализация таблиц данных;
- реализация операции ключевого процесса в среде MS Access;
- тестирование и количественная оценка результатов тестирования;

- подготовка блок-схемы алгоритма работы программы;
- качественный и количественный анализ рисков.

Разработанная в ходе отчётной работы база данных позволит упорядочить и систематизировать работу врача-терапевта. А это в свою очередь может значительно повысить качество работы, ускорить приём пациентов и соответственно решить проблему с очередями на прием к врачу.

Раздел 1. Детальный анализ спиралевидной методологии внедрения систем

1.1. Жизненный цикл программного обеспечения

Жизненный цикл программного обеспечения (ПО) начинается от замысла или потребности, которая может быть удовлетворена полностью или частично программным средством, и завершается прекращением применения этого программного средства. Такая архитектура создается совокупностью процессов и взаимосвязями между этими процессами. Определение процессов жизненного цикла основывается на двух базовых принципах: связности и ответственности [1].

Принцип связности: процессы жизненного цикла являются связными и соединяются оптимальным образом, считающимся практичным и выполнимым. Принцип ответственности: процесс передается под ответственность какой-либо организации или стороне в пределах жизненного цикла программного средства.

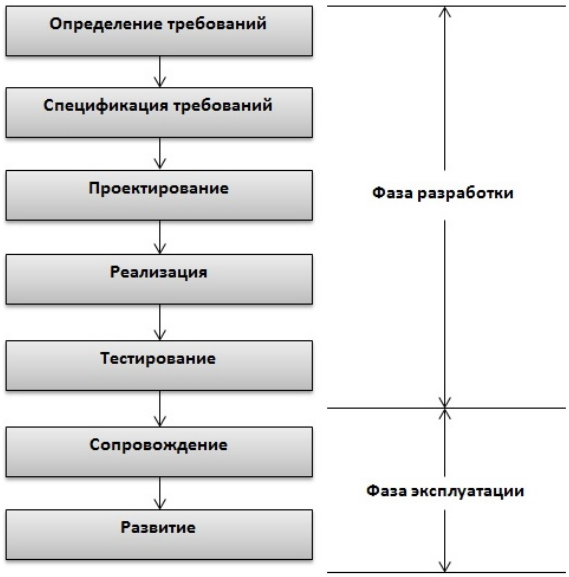


Рисунок 1.1 – *Общепринятая модель жизненного цикла ПО*

Рассмотрим подробнее элементы модели жизненного цикла ПО. Определение требований – определяются: задачи, ожидаемые функции,

ограничения. Проекта еще нет. Спецификации системы в соответствии с требованиями – определяется, что разрабатываемая система должна делать. Фактическое начало работ. Проектирование (конструирование, дизайн) – определяется декомпозиция системы /архитектура.

Реализация (кодирование) – разработка и реализация программы. Тестирование – проверка соответствия спецификациям. Сопровождение – поддержка использования продукта. Развитие – поддержка эволюции системы. Жизненный цикл можно представить в виде моделей. В настоящее время их существует большое количество. Но наиболее известными являются: каскадная, итерационная и спиральная модель жизненного цикла.

1.2. Спиралевидная методология внедрения

В данной методологии делается упор на начальные этапы жизненного цикла системы: анализ и проектирование. Спиральная модель (рисунок 1.2) – классический пример применения эволюционной стратегии разработки [2]. Следует отметить, спиралевидную модель нередко рассматривают как частный случай итерационной модели.

Рисунок 1.2 – *Спиралевидная модель*

Из рисунка 1.2 имеем:

- 1 – начальный сбор требований проекта;
- 2 – та же работа, но на основе рекомендаций заказчика;
- 3 – планирование проекта и анализ риска с использованием начальных требований;
- 4 – планирование и анализ риска реакции заказчика;
- 5 – переход к комплексной системе;
- 6 – начальный макет системы;
- 7 – версия системы следующего уровня;
- 8 – разработанная система;
- 9 – оценивание заказчиком.

Как показано на рисунке 1.2, модель определяет четыре действия, каждое из которых соответствует своему квадранту спирали [3]:

- Планирование – определение целей, вариантов и ограничений.
- Анализ риска – анализ вариантов и распознавание/выбор риска.
- Конструирование – разработка продукта следующего уровня.
- Оценивание – оценка заказчиком результатов конструирования.

Данная методология ориентирована на активную работу с пользователями и представляющая разрабатываемую информационную систему как постоянно корректируемую во время разработки. В спиральной модели основной упор делается на 2 этапа: анализ и проектирование. На них проверяется реализуемость технических решений путем создания прототипов.

Спиральная модель позволяет начинать работу над следующим этапом, не дожидаясь завершения предыдущего. Целью данной модели является возможность как можно раньше ознакомить пользователей с работоспособным продуктом, корректируя при необходимости требования к разрабатываемому

продукту и каждый "виток" спирали означает создание фрагмента или версии. Основная проблема спирального цикла – определение момента перехода на следующий этап, и возможным ее решением является принудительное ограничение по времени для каждого из этапа жизненного цикла [4].

Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла. Выделим десять самых распространённый рисков, которые выделил автор спиралевидной модели Барри Боэм [5]:

- дефицит специалистов;
- нереалистичные сроки и бюджет;
- реализация несоответствующей функциональности;
- разработка неправильного пользовательского интерфейса;
- «золотая сервировка», перфекционизм, ненужная оптимизация и оттачивание деталей;
- непрекращающийся поток изменений;
- нехватка информации о внешних компонентах, определяющих окружение системы или вовлечённых в интеграцию;
- недостатки в работах, выполняемых внешними (по отношению к проекту) ресурсами;
- недостаточная производительность получаемой системы;
- разрыв между квалификацией специалистов и требованиями проекта.

Основными плюсами данной методологии является:

- позволяет быстрее показать пользователям системы работоспособный продукт
- обеспечивает большую гибкость в управлении проектом

- возможность совершенствовать процесс разработки благодаря анализу, который проводится на каждом витке спирали. Это позволяет проводить оценку того, что должно быть изменено в организации разработки, и улучшить ее на следующем ветке спирали;
- позволяет получить более надежную и устойчивую систему.

Но также у данной системы есть свои минусы:

- увеличивается неопределенность у разработчика в перспективах развития проекта.
- затруднены операции временного и ресурсного планирования всего проекта в целом.

Для анализа риска используют количественные и качественные методы. Главной задачей качественного анализа является определение факторов риска, этапов работы, при выполнении которых риск возникает, т. е. установить потенциальные области риска, после чего идентифицировать все возможные риски.

Качественный анализ подразумевает выявление рисков, присущих проекту, их описание и группировку. Обычно выявляются специфические риски, непосредственно связанные с реализацией проекта (проектные), а также форс-мажорные, управленческие, юридические. Качественный анализ риска предполагает:

- выявление источников и причин риска, этапов и работ, при выполнении которых возникает риск, т.е. установление потенциальных зон риска;
- идентификацию (установление) всех возможных рисков;

- выявление практических выгод и возможных негативных последствий, которые могут наступить при реализации содержащего риск решения.

Количественный анализ рисков – это количественный анализ потенциального воздействия идентифицированных рисков на общие цели проекта. Количественный анализ рисков обычно выполняется для рисков, которые были квалифицированы в результате качественного анализа. При количественном анализе также оцениваются вероятности возникновения рисков и размеры ущерба/выгоды. Здесь анализируются риски, имеющие высокие и умеренные ранги. Выбор методов анализа определяется для каждого проекта и зависит от наличия времени и от бюджета.

Задача количественного анализа состоит в численном измерении степени влияния рискованных факторов проекта на поведение критериев эффективности всего инвестиционного проекта. Все количественные методы, применяемые в теории рисков, целесообразно классифицировать по целям оценки на прямые и обратные методы исследования. Оценка риска, связанная с определением его уровня, в прямых задачах происходит на основании априори известной информации. В обратных задачах определяются ограничения на один или несколько варьируемых исходных параметров с целью удовлетворения заданных ограничений на уровень приемлемого риска.

Количественная оценка риска – это этап анализа риска, имеющий целью определить его количественные характеристики: вероятность наступления неблагоприятных событий и возможный размер ущерба. Процесс количественного анализа риска включает следующие стадии:

- создание прогнозной модели;
- определение переменных риска;

- определение вероятностного распределения отобранных переменных и определение диапазона возможных значений для каждой из них;
- установление наличия или отсутствия корреляционных связей среди рисковых переменных;
- прогоны моделей (определение характеристик результативных величин как случайных величин);
- анализ результатов (построение уровней риска).

Раздел 2. Идентификация требований

Данный подраздел должен включать в себя требования к системе в целом, к функциям, выполняемым системой, видам обеспечения. Состав требований зависит от вида, назначения, специфических особенностей разрабатываемой системы.

В зависимости от вида системы приводятся требования к информационному, программному, техническому, лингвистическому, метрологическому, методическому, организационному и другим видам обеспечения проектируемой и внедряемой автоматизированной системы.

Проблемы, которые могут возникнуть при создании программного обеспечения для автоматизации какой либо системы, не всегда можно понять. Очень трудно чётко описать те действия, которые должна выполнять система. Описание функциональных возможностей и ограничений, накладываемых на систему, называется требованиями к этой системе. Требования к продукту должны быть установлены таким образом, что могло бы гарантировать их адекватность и верный «перевод» с языка пользователя. Для начала необходимо выявить все возможные требования, предъявляемые к разрабатываемому программному продукту.

Выявить их можно с помощью анализа требований – часть процесса разработки программного обеспечения, включающая в себя сбор требований к программному обеспечению (ПО), их систематизацию, выявление взаимосвязей, а также документирование. В процессе сбора требований важно принимать во внимание возможные противоречия требований различных заинтересованных лиц, таких как заказчики, разработчики или пользователи. Процесс формирования и анализа требований проходит через ряд этапов:

- Анализ предметной области. Аналитики должны изучить предметную область, где будет эксплуатироваться система.
- Сбор требований. Это процесс взаимодействия с лицами, формирующими требования. Во время этого процесса продолжается анализ предметной области.
- Назначение приоритетов. В любом наборе требований одни из них будут более важны, чем другие. На этом этапе совместно с лицами, формирующими требования, определяются наиболее важные требования. Идентификация предварительных требований к новому продукту позволяет, в соответствии с требованиями клиента, определить характеристики нового продукта.

2.1. Анализ предметной области

Терапевт – специалист в области терапии; врач, специализирующийся на выявлении, лечении, профилактике внутренних болезней. Он осуществляет первичную диагностику, координирует взаимодействие пациента с остальными специалистами, выписывает направления на большинство обследований и процедур, оформляет медицинскую документацию.

Основные задачи этого специалиста:

- первичный прием пациентов, сбор анамнеза, проведение осмотра и других объективных методов обследования;
- ранняя диагностика на основании результата обследования пациента и анализа его жалоб;
- первичная консультация, разъяснение пациенту причин его недуга. Хороший терапевт выступает еще и в роли психолога, успокаивая больного, предоставляя ему информацию о заболевании в нужном объеме;

- назначение лекарств, физиотерапевтических процедур и других лечебных мероприятий в пределах своей компетенции;
- назначение лабораторных анализов и инструментального обследования;
- в случае осложненного течения или неясного генеза заболевания – направление к профильному специалисту для более детальной диагностики и прохождения лечения в соответствии с его рекомендациями;
- разработка единой схемы лечения с учетом рекомендаций разных узких специалистов;
- принятие решения о госпитализации;
- оценка риска развития хронического заболевания и принятие мер к его снижению;
- консультации относительно укрепления иммунитета, профилактики осложнений, рецидивов и перехода заболевания в хроническую форму;
- регулярное наблюдение пациентов с хроническими заболеваниями;
- разработка рекомендаций относительно изменения образа жизни, условий труда, санаторно-курортного лечения и прочих;
- назначение схемы комплексного медицинского обследования при прохождении профосмотра, медкомиссии;
- осмотр перед вакцинацией и принятие решение относительно ее проведения.

В дальнейшем будем работать с такими ключевыми бизнес-процессами, как: «Приём пациента», «Лечение пациента», «Реабилитация пациента».

2.2. Пользовательские и функциональные требования, их приоритизация

Далее был составлен список выявленных пользовательских требований:

- хранение данных о пациенте (Фамилия, Имя, Отчество, дата рождения и т.д.);
- хранение данных о записи на приём (когда и сколько раз был на приёме, по какой причине);
- хранение данных об истории лечения пациента (какие лекарства были назначены, направление на анализы);
- хранение данных о реабилитации пациента (какими лекарствами происходило лечение, помогли ли они; если не помогли, назначение новых препаратов);
- сведения, полученные путём расспроса пациента (обследуемого);
- управление данными о записи на прием (изменение, удаление, добавление, сортировка);
- управление данными, полученными путём расспроса пациента (изменение, удаление, добавление, сортировка);
- управление данными о лечении пациента (изменение, удаление, добавление, сортировка);
- разграничение доступа (Для того, чтобы защитить данные пациента);
- разработка интерфейса (Для ускорения приема пациентов).

Следующим шагом выявим функциональные требования для того, чтобы выполнить пользовательские требования. Функциональные требования (functional requirements) определяют функциональные возможности разрабатываемого ПО для реализации пользовательских требований. Выделим данные требования для нашего ключевого бизнес процесса:

- база данных, содержащая таблицы «Личные данные пациентов», «Даты посещений», «Причины посещения», «Лечение пациента»,

«Направление на анализ», «Анализ мочи», «Анализ кала», «Анализ крови», «Реабилитация пациента»;

- вывод на экран данных из вышеперечисленных таблиц;
- добавление данных о пациенте;
- редактирование данных о пациенте во всех таблицах;
- поиск конкретного пациента по ФИО во всех таблицах;
- программа должна корректно работать на всех компьютерах медучреждения;
- данные хранятся непосредственно в создаваемом приложении;
- установление пароля на саму базу данных;
- разработка интерфейса для простоты использования продукта.

Приоритизация требований позволяет понять, какие требования пользователю необходимо реализовать в первую очередь и на что следует обратить особое внимание. Это позволит избежать излишних материальных и временных затрат на проектирование и разработку модулей или функционала. Расставим приоритеты для ранее представленных пользовательских и функциональных требований:

- Приоритет №1 – данные требования является основой разрабатываемой системы. Без выполнения данного пункта реализовать систему не получится.
- Приоритет №2 – данные требования критичны для функциональных возможностей системы, но не сильно отразятся на работе программы при их отсутствии.
- Приоритет №3 – данные требования не являются ключевыми для полноценного функционирования самой программы, но их было бы неплохо реализовать.

2.3. Список требований

После сбора и анализа пользовательских и функциональных требований создается список требований. Он позволяет связать и сопоставить все требования, их приоритеты и программные компоненты, отвечающие за реализацию требований. Список требований облегчает процесс отслеживания требований разработчиком и позволяет удостовериться в их полном выполнении перед окончанием работ.

Требования для разрабатываемого продукта представлены в Таблице 2.3.

Таблица 2.3 – *Список требований для разрабатываемого продукта*

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет требования
1	Хранение данных о пациенте	Таблица данных «Личные данные пациентов»	Программа по ведению данных	Приоритет №1
2	Хранение данных о записи на прием	Таблица данных «Даты посещений»		
3	Сведения, полученные путём расспроса пациента	Таблица данных «Причины посещения»		
4	Хранение данных об истории лечения пациента	Таблица данных «Лечение пациента»		
5	Хранение данных о том, на какой анализ был направлен пациент	Таблица данных «Направление на анализ»		
6	Хранение данных о результатах анализа мочи	Таблица данных «Анализ мочи»		
7	Хранение данных о результатах анализа кала	Таблица данных «Анализ кала»		
8	Хранение данных о результатах анализа крови	Таблица данных «Анализ крови»		
9	Хранение данных о том, как происходит реабилитация пациента	Таблица данных «Реабилитация пациента»		
10	Управление данными о записи на прием	Возможность редактирования, удаления,	Программа для добавления,	Приоритет №2

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет требования
	(изменение, удаление, добавление, сортировка)	добавления, поиска и сортировки записей в таблице данных «Даты посещений»	редактирования, удаления, поиска, сортировка и просмотра данных	
11	Управление данными, полученными путём расспроса пациента (изменение, удаление, добавление, сортировка)	Возможность редактирования, удаления, добавления, поиска и сортировки записей в таблице данных «Причины посещения»		
12	Управление данными о лечении пациента	В таблице данных «Лечение пациента»		
13	Управление данными о направлении на анализ	В таблице данных «Направление на анализ»		
14	Управление данными об анализе мочи	В таблице данных «Анализ мочи»		
15	Управление данными об анализе кала	В таблице данных «Анализ кала»		
16	Управление данными об анализе крови	В таблице данных «Анализ крови»		
17	Управление данными о реабилитации пациента	В таблице данных «Реабилитация пациента»		
18	Разграничение доступа	Установление пароля на саму базу данных	Программа авторизации пользователя	Приоритет №3
19	Разработка интерфейса	Разработка интерфейса	Программа упрощения работы пользователя	

2.4. Задание циклов разработки по спиралевидной модели

Разработка программного продукта согласно спиралевидной модели будет производиться в следующем порядке:

- начало:
  - анализ требований (19 требований – см. табл. 2.3);
  - моделирование ключевых бизнес-процессов с помощью нотаций ARIS VACD и UML AD;
  - моделирование данных и интерфейсов программ;

- анализ рисков на основе полученных требований;
- составление плана разработки по спиральной модели.

- 1-й виток спирали:
  - планирование текущего цикла разработки (реализовать требования 1-9 в среде MS Access);
  - моделирование данных и интерфейсов программ;
  - анализ рисков на основе полученных требований;
  - реализация требований 1-9 в среде MS Access;
  - тестирование реализованных возможностей программы;
  - демонстрация прототипа программы заказчику.
- 2-й виток спирали:
  - планирование текущего цикла разработки (реализовать требования 10-17 в среде MS Access);
  - моделирование данных и интерфейсов программ;
  - анализ рисков на основе полученных требований;
  - реализация требований 10-17 в среде MS Access;
  - тестирование реализованных возможностей программы;
  - демонстрация прототипа программы заказчику.
- 3-й виток спирали:
  - планирование текущего цикла разработки (реализовать требование 18-19 в среде MS Access);
  - анализ рисков на основе полученных требований и всего процесса реализации витков;
  - реализация требований 18-19 в среде MS Access;
  - тестирование реализованных возможностей программы;
  - подготовка к релизу (исправление мелких недостатков, ошибок и т.д.);

- тестирование конечного продукта;
- релиз конечного продукта.

Раздел 3. Проектирование ключевых бизнес-процессов

Бизнес-процесс – это устойчивая целенаправленная последовательность исполнения функций, направленная на создание результата, имеющего ценность для потребителя [6]. При проектировании ключевых бизнес-процессов используют две модели: AS-IS (как есть) и TO-BE (как будет):

- модель AS-IS, которая описывает состояние моделируемой предметной области на момент создания модели;
- модель TO-BE, описывающая возможное будущее состояние предметной области, в которое она перейдёт в результате оптимизации существующей системы и внедрения новых технологий [7].

Прежде чем пытаться выбрать существующую или создать собственную информационную систему, требуется проанализировать, как работает система на данный момент времени. После этого строится функциональная модель AS-IS. Анализ этой модели позволит выявить недостатки и понять, в чем будут состоять преимущества новых бизнес-процессов. В модели TO-BE как раз есть возможность исправление таких недостатков. Данная модель нужна для оценки последствий внедрения информационной системы и анализа альтернативных путей выполнения работы и документирования того, как система будет функционировать в будущем.

3.1. Проектирование на основе нотации ARIS VACD

Сила методологии ARIS заключается в ее комплексности, которая проявляется во взаимосвязи моделей, построенных в различных нотациях. Методология ARIS позволяет описывать деятельность организации с разных точек зрения, при этом полученные модели в определенной степени связаны между собой [8]. Основными минусами данного подхода является:

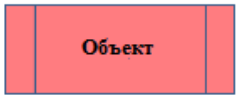
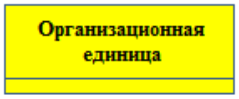
- Наличие инструментальной среды ARIS. Она является дорогостоящей и достаточно сложной в использовании. Но существует упрощенная и бесплатная версия под названием ARIS Express.
- Трудно реализуемые задачи на практике, так как влекут большой расход ресурсов (человеческих, материальных и финансовых) в течение длительного времени.

Одной из важнейших нотаций ARIS является нотация Value-added Chain Diagram. Она используется для описания бизнес-процессов организации на верхнем уровне.

Главным отличием данной модели от других процессных моделей является то, что информационные и материальные потоки на схеме VACD изображаются не стрелками, а объектами. При этом для каждого типа потока используется свой объект. На модели VACD методологии ARIS в отличие от классического подхода также используются логические связи между работами, которые позволяют отобразить логическую последовательность выполнения работ.

Данная нотация подходит для изучения организации в целом. Но для выявления возможных проблем этой нотации может быть недостаточно, она помогает определить те процессы, которые и нуждаются в изменениях и доработке. При описании бизнес-процессов воспользуемся приложением ARIS Express. Элементы, используемые в данной нотации, приведены в таблице 3.1.

Таблица 3.1 – *Условные обозначения нотации ARIS VACD*

Графическое представление	Наименование	Описание
	Процесс	Описание работ, выполняемых сотрудниками организации
	Входящих или исходящий объект	Элементы, содержащие носители информации
	Организационная единица	Отражает звенья, ответственные за исполнение процесса

3.2. Проектирование на основе нотации UML Activity Diagram

При моделировании поведения проектируемой системы появляется необходимость представить процесс изменения ее состояний и детализировать особенности алгоритмической и логической реализации выполняемых системой операций. Обычно для этого использовались блок-схемы или структурные схемы алгоритмов. Блок-схема — распространенный тип схем, описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединенных между собой линиями, указывающими направление последовательности. В данной работе один из выбранных языков был UML Activity Diagram.

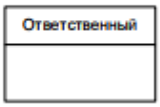
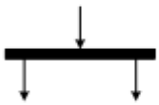
Язык UML (Unified Modeling Language), или унифицированный язык моделирования, предназначен для описания, визуализации, проектирования и документирования объектно-ориентированных систем и бизнес-процессов с ориентацией на их последующую реализацию в виде программного обеспечения [9].

Для моделирования процесса выполнения операций в языке UML используются диаграммы деятельности (Activity diagram). Они служат для моделирования последовательности действий, которые выполняются различными элементами, входящими в состав системы. Другими словами, этот

вид диаграмм раскрывает детали алгоритмической реализации операций, выполняемых системой, поэтому диаграмма деятельности похожа на обычную блок-схему [10]. Но в отличие от блок-схемы, диаграмма деятельности также показывает одновременно параллельную и последовательную деятельность.

UML AD нотация проста для изучения неподготовленным пользователем, она интуитивно понятна. Данная нотация использует широко известные графические элементы (табл.3.2). В UML AD нотации изображение процессов очень похоже на блок-схемы, то есть многим пользователям изображения в UML AD нотации сразу будут подсознательно ясны.

Таблица 3.2 – *Условные обозначения нотации ARIS VACD*

Графический элемент	Описание	Графический элемент	Описание
	Начало		Ответственный организационный уровень
	Конец		Условие
	Процесс		Разветвитель
	Входящие/исходящие документы		Соединитель

3.3. Проектирование процессов в моделях AS-IS и TO-BE с помощью UML AD и ARIS VACD

На рисунках 3.2 и 3.3 рассматривается первый уровень проектирования процесса работы врача-терапевта в аннотации ARIS VACD модели «AS-IS», которая подразумевает под собой проектирование процессов в настоящий момент времени и модели «TO-BE» после внедрения электронной базы данных.

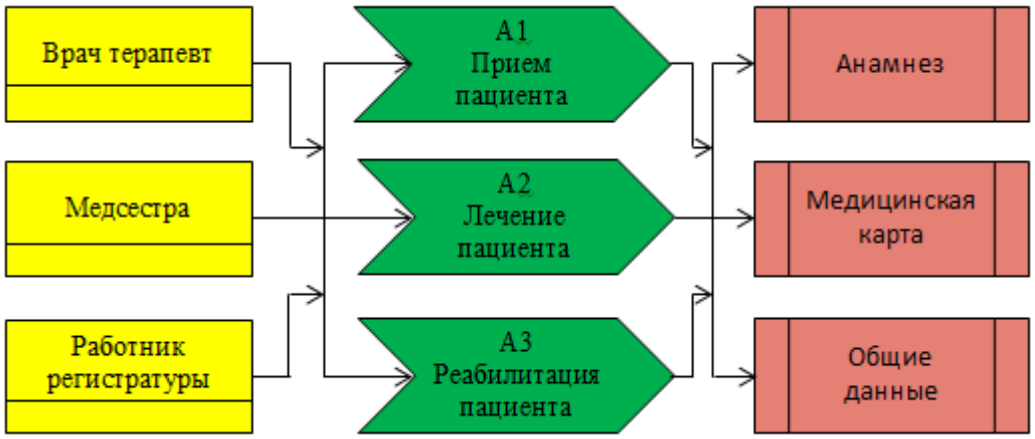


Рисунок 3.2 – *Представление процесса в ARIS VACD на первом уровне в модели «AS-IS»*

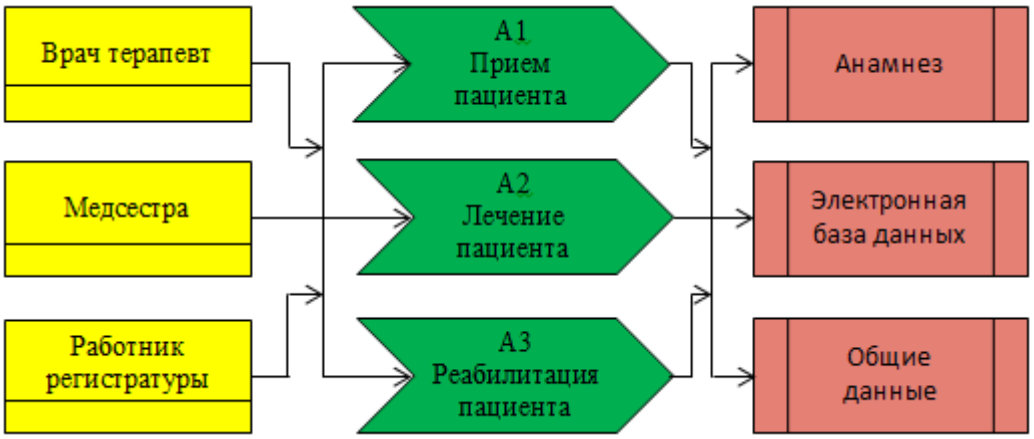


Рисунок 3.3 – *Представление процесса в ARIS VACD на первом уровне в модели «TO-BE»*

Для того чтобы создать программное обеспечение, требуется больше информации, поэтому необходимо провести более подробное проектирование. Для этого следует произвести второй и третий уровень детализации.

На рисунках 3.4 и 3.5 представлен второй уровень детализации для уточнения процесса «Прием пациента». Второй уровень детализации процессов «Лечение пациента» и «Реабилитация пациента» представлен в приложении (Рис.8.1, Рис.8.2, Рис.8.5, Рис.8.6).

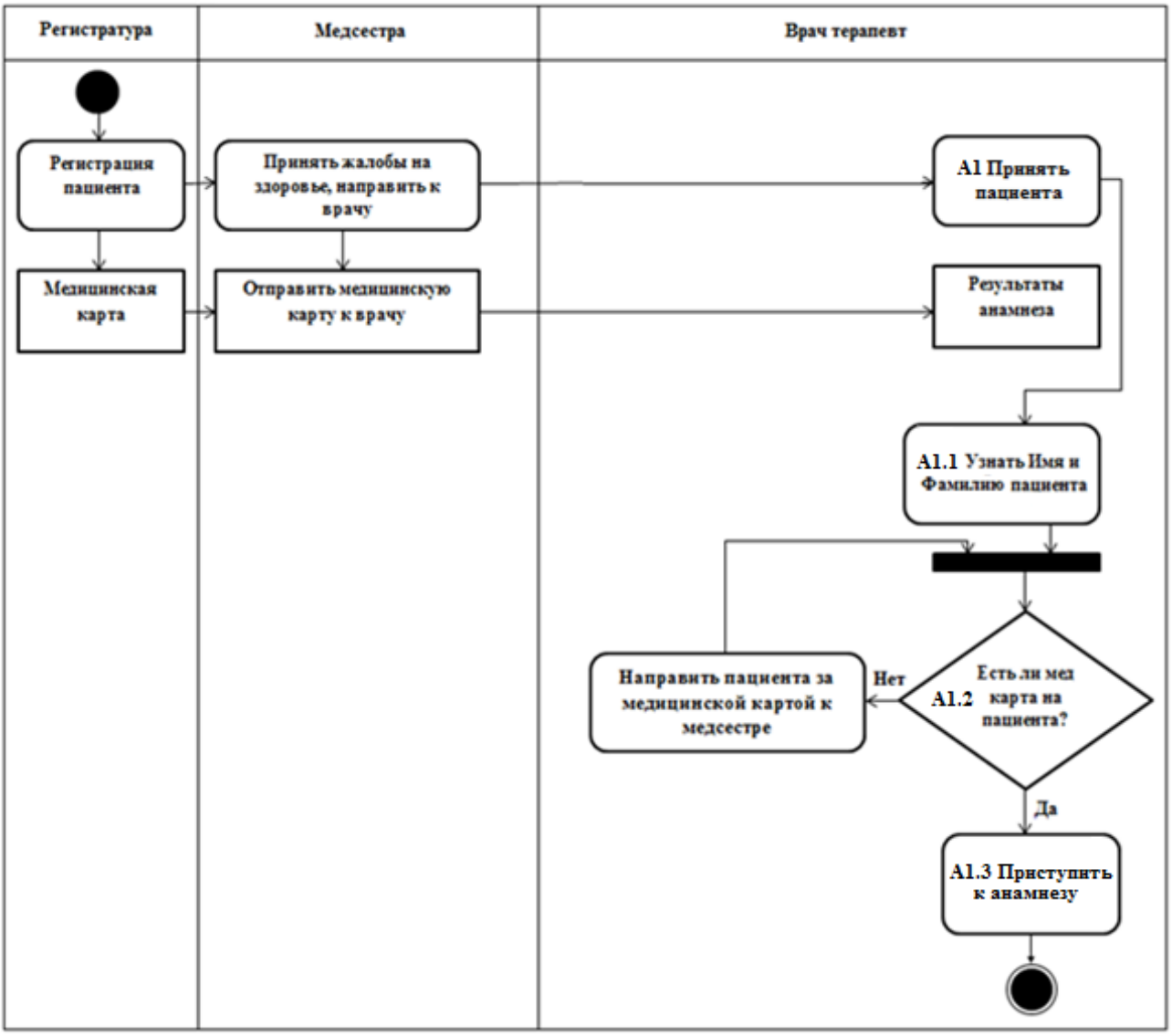


Рисунок 3.4 – *Представление подпроцесса в UML AD на втором уровне в модели «AS-IS» для уточнения процесса «Прием пациента»*

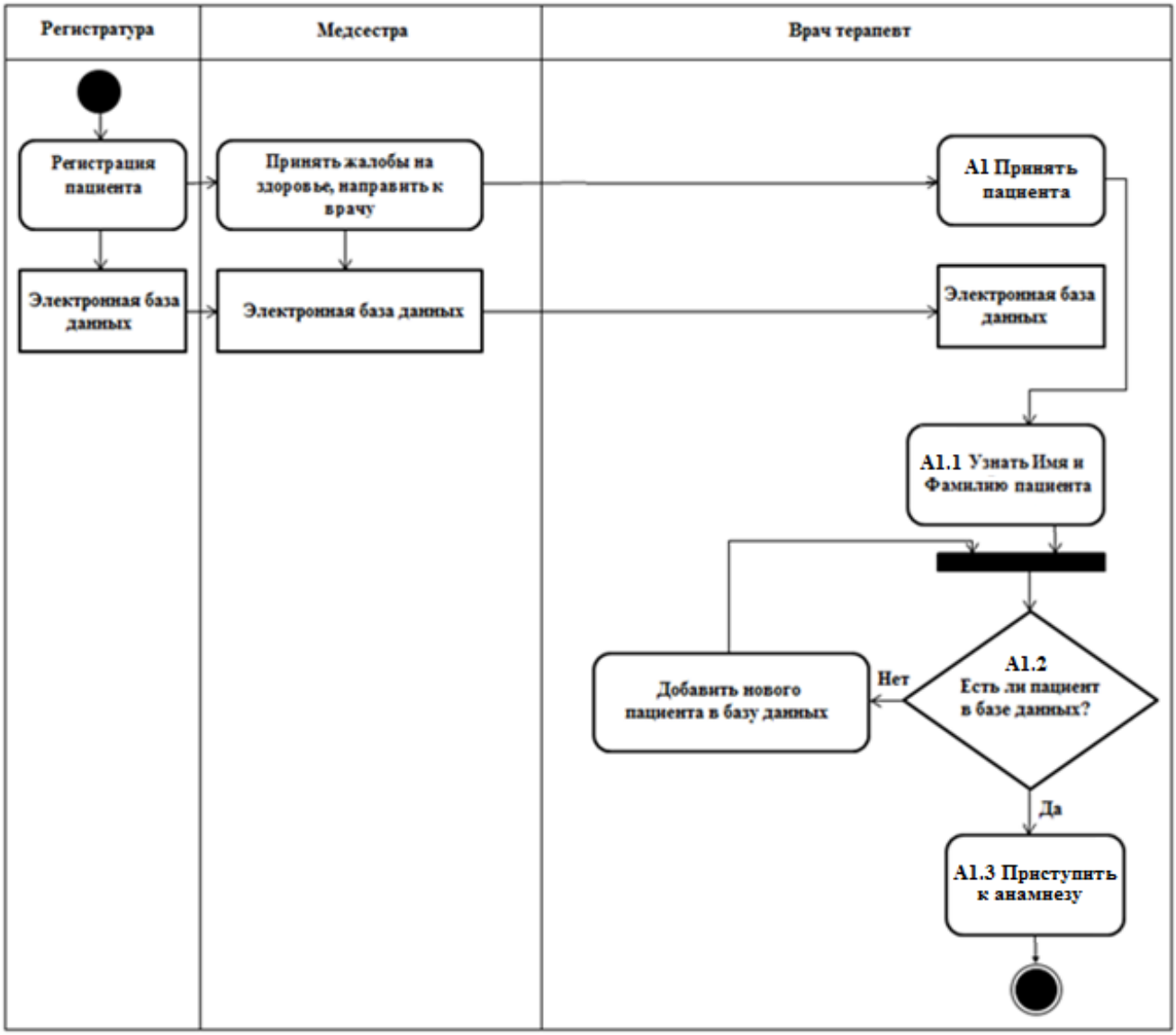


Рисунок 3.5 – *Представление подпроцесса в UML AD на втором уровне в модели «TO-BE» для уточнения процесса «Прием пациента»*

На рисунках 3.6 и 3.7 представлен третий уровень детализации для уточнения процесса «Приступить к анамнезу». Третий уровень детализации процессов «Лечение пациента» и «Реабилитация пациента» представлен в приложении (Рис.8.3, Рис.8.4, Рис.8.7, Рис.8.8.).

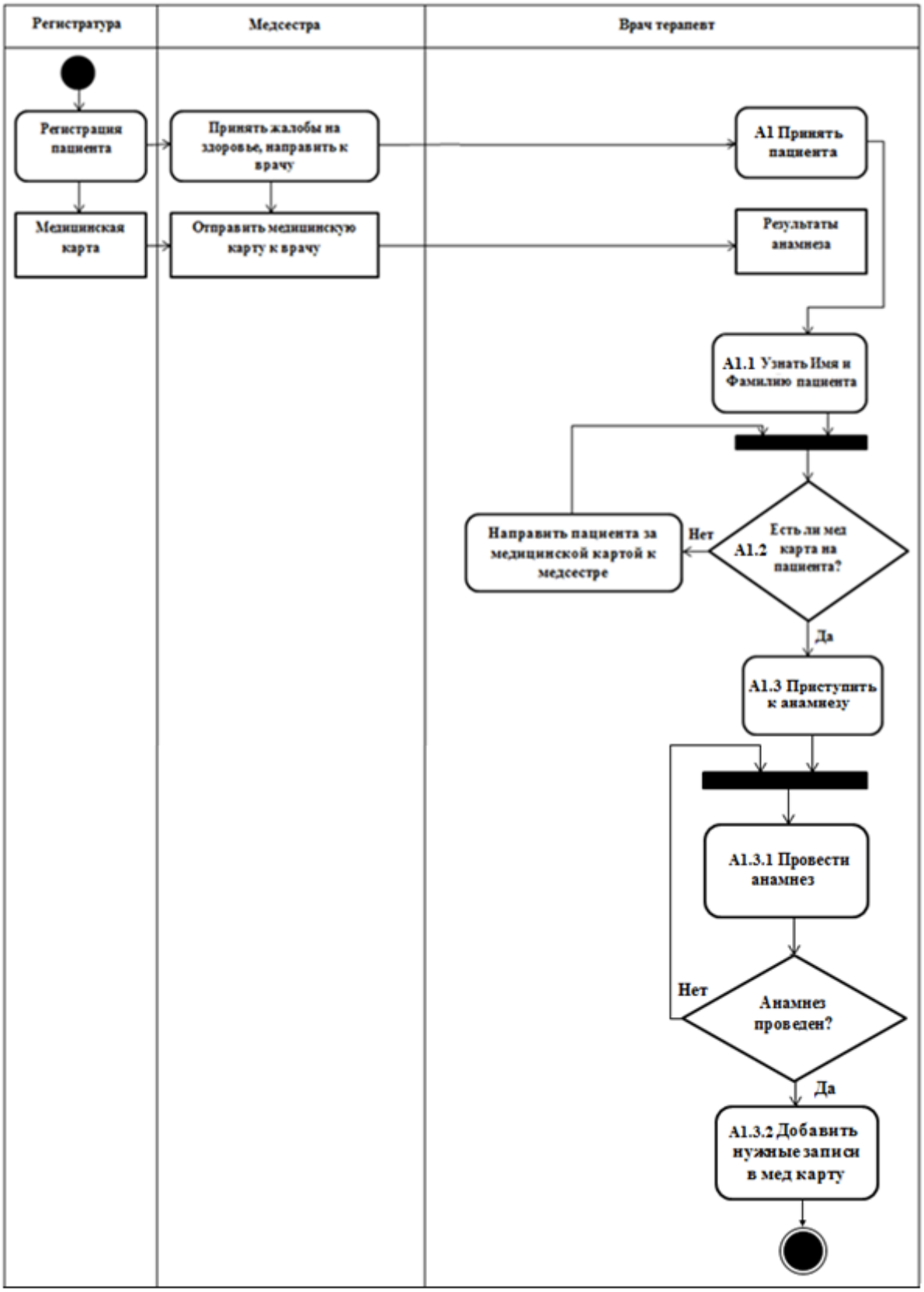


Рисунок 3.6 – *Представление процесса в UML AD на третьем уровне в модели «AS-IS» для уточнения процесса «Приступить к анамнезу»*

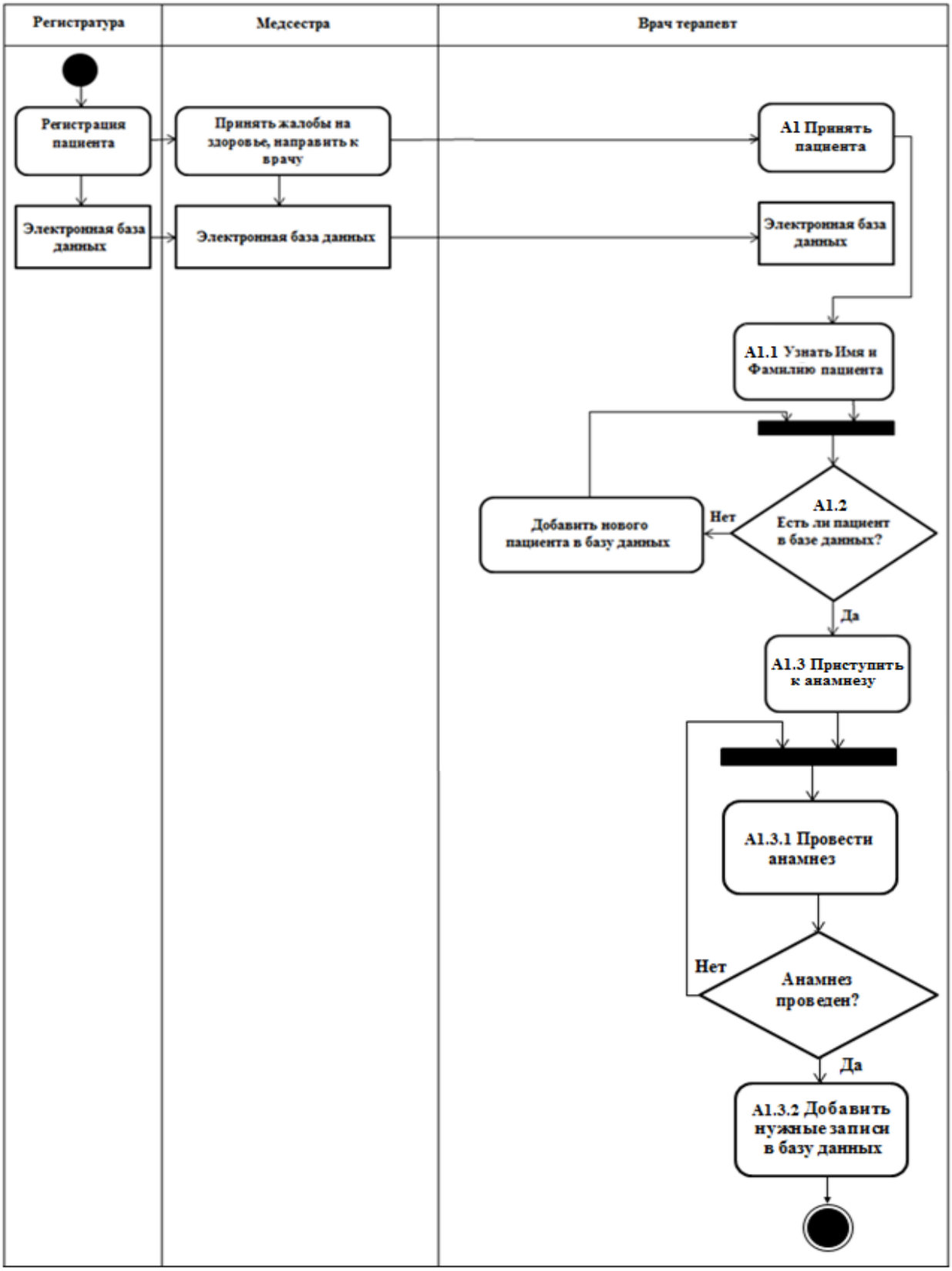


Рисунок 3.7 – *Представление процесса в UML AD на третьем уровне в модели «TO-BE» для уточнения процесса «Приступить к анамнезу»*

Для более наглядного представления вышеуказанных бизнес-процессов на 1-3 уровнях, построим карту бизнес-процессов в модели «TO-BE» (рисунок 3.8).

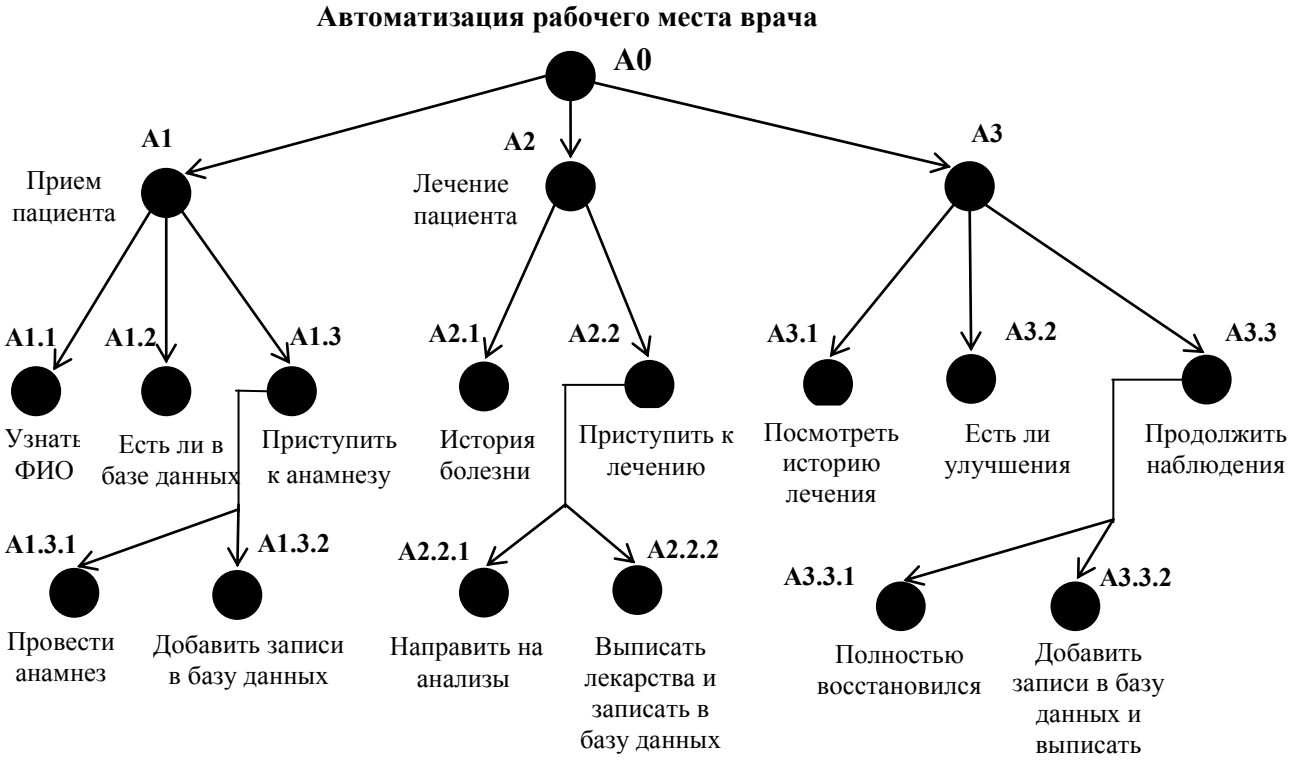


Рисунок 3.8 – *Карта бизнес-процессов в модели «TO-BE»*

3.4. Архитектура данных разрабатываемой системы

На основе порядка использования данных, рассмотренного выше, всю информацию можно разбить на классы, приведённые в таблице 3.4.

Таблица 3.4 – *Классы данных*

Название класса	Данные	Тип данных	Размерность
Личные данные пациентов	ФИО	Текстовые	65
	Дата рождения	Дата/время	Краткий формат даты
	Домашний адрес	Текстовые	70
	Номер паспорта	Текстовые	30
	Телефон	Текстовые	20

Название класса	Данные	Тип данных	Размерность
	Пол	Текстовые	1
	ОМС	Текстовые	20
Даты посещений	ФИО врача	Текстовые	85
	Пациент	Числовой	Длинное целое
	Дата посещения	Дата/время	Краткий формат даты
Причина посещения	Пациент	Числовой	Длинное целое
	Причина посещения	Текстовые	200
	Дата посещения	Числовой	Длинное целое
Лечение пациента	Пациент	Числовой	Длинное целое
	Причина посещения	Числовой	Длинное целое
	Дата посещения	Дата/время	Длинное целое
	Лечение	Текстовые	255
Направления на анализ	Пациент	Числовой	Длинное целое
	ФИО врача	Числовой	Длинное целое
	Дата посещения	Числовой	Длинное целое
	Вид анализа	Текстовые	50
	Печать	Текстовые	1
Анализ крови	Пациент	Числовой	Длинное целое
	Показатели	Текстовые	50
	Норма	Текстовые	50
	Результаты анализа	Текстовые	50
	Вердикт	Текстовые	50
Анализ мочи	Пациент	Числовой	Длинное целое
	Показатели	Текстовые	50
	Норма	Текстовые	50
	Результаты анализа	Текстовые	50
	Вердикт	Текстовые	50
Анализ кала	Пациент	Числовой	Длинное целое
	Показатели	Текстовые	50
	Норма	Текстовые	50
	Результаты анализа	Текстовые	50
	Вердикт	Текстовые	50

Название класса	Данные	Тип данных	Размерность
Реабилитация пациента	Пациент	Числовой	Длинное целое
	Врач	Числовой	Длинное целое
	Дата посещения	Числовой	Длинное целое
	Дата проведения осмотра	Дата/время	Краткий формат даты
	Результат проведения осмотра	Текстовые	150
	Примечание	Текстовые	150

Далее необходимо провести процедуру нормализации данных [11]. Стандартно все данные приводят к третьей нормальной форме (НФ), потому как нормализации такого уровня бывает достаточно. Согласно первой НФ, в одном поле таблицы должен храниться только один атрибут. Таким образом, в классе «Личные данные пациентов» и «Даты посещений» данные «ФИО» и «ФИО врача» необходимо разбить на три отдельные строки: «Фамилия», «Имя», «Отчество».

Согласно второй НФ, должны соблюдаться условия первой НФ и каждый не ключевой атрибут должен зависеть от ключа. Следовательно, в классы «Личные данные пациентов» и «Даты посещений» добавляется графа «Код пациента», а в таблицы «Даты посещений» и «Причина посещений» добавляется графа «Код причины посещения».

В третьей НФ должны соблюдаться условия второй НФ и все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы, должно выноситься в отдельные таблицы. Таким образом, после разделения всех данных на классы и проведения процедуры нормализации, получается нормализованная таблица класса данных (табл.3.5) и архитектура разрабатываемого приложения, приведённая на рисунке 3.9.

Таблица 3.5 – *Классы данных (нормализованная)*

Название класса	Данные	Тип данных	Размерность
Личные данные пациентов	Код пациента	Счетчик	Длинное целое
	Фамилия	Текстовые	20
	Имя	Текстовые	15
	Отчество	Текстовые	30
	Дата рождения	Дата/время	Краткий формат даты
	Домашний адрес	Текстовые	70
	Номер паспорта	Текстовые	30
	Телефон	Текстовые	20
	Пол	Текстовые	1
	ОМС	Текстовые	20
Даты посещений	Код посещения	Счетчик	Длинное целое
	Фамилия врача	Текстовые	50
	Имя врача	Текстовые	15
	Отчество врача	Текстовые	20
	Код пациента	Числовой	Длинное целое
	Дата посещения	Дата/время	Краткий формат даты
Причина посещения	Код причины посещения	Счетчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Причина посещения	Текстовые	200
	Дата посещения	Числовой	Длинное целое
Лечение пациента	Код лечения	Счетчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Причина посещения	Числовой	Длинное целое
	Дата посещения	Числовой	Длинное целое
	Лечение	Текстовые	255
Направления на анализ	Код анализа	Счетчик	Длинное целое
	Код пациента	Числовой	Длинное целое

Название класса	Данные	Тип данных	Размерность
	Фамилия врача	Числовой	Длинное целое
	Имя врача	Числовой	Длинное целое
	Отчество врача	Числовой	Длинное целое
	Дата посещения	Числовой	Длинное целое
	Вид анализа	Текстовые	50
	Печать	Текстовые	1
Анализ крови	🔑 Код анализа	Счетчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Показатели	Текстовые	50
	Норма	Текстовые	50
	Результаты анализа	Текстовые	50
	Вердикт	Текстовые	50
Анализ мочи	🔑 Код анализа	Счетчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Показатели	Текстовые	50
	Норма	Текстовые	50
	Результаты анализа	Текстовые	50
	Вердикт	Текстовые	50
Анализ кала	🔑 Код анализа	Счетчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Показатели	Текстовые	50
	Норма	Текстовые	50
	Результаты анализа	Текстовые	50
	Вердикт	Текстовые	50
Реабилитация пациента	🔑 Код реабилитации	Счетчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Врач	Числовой	Длинное целое
	Дата посещения	Числовой	Длинное целое
	Дата проведения осмотра	Дата/время	Краткий формат даты

Название класса	Данные	Тип данных	Размерность
	Результат проведения осмотра	Текстовые	150
	Примечание	Текстовые	150

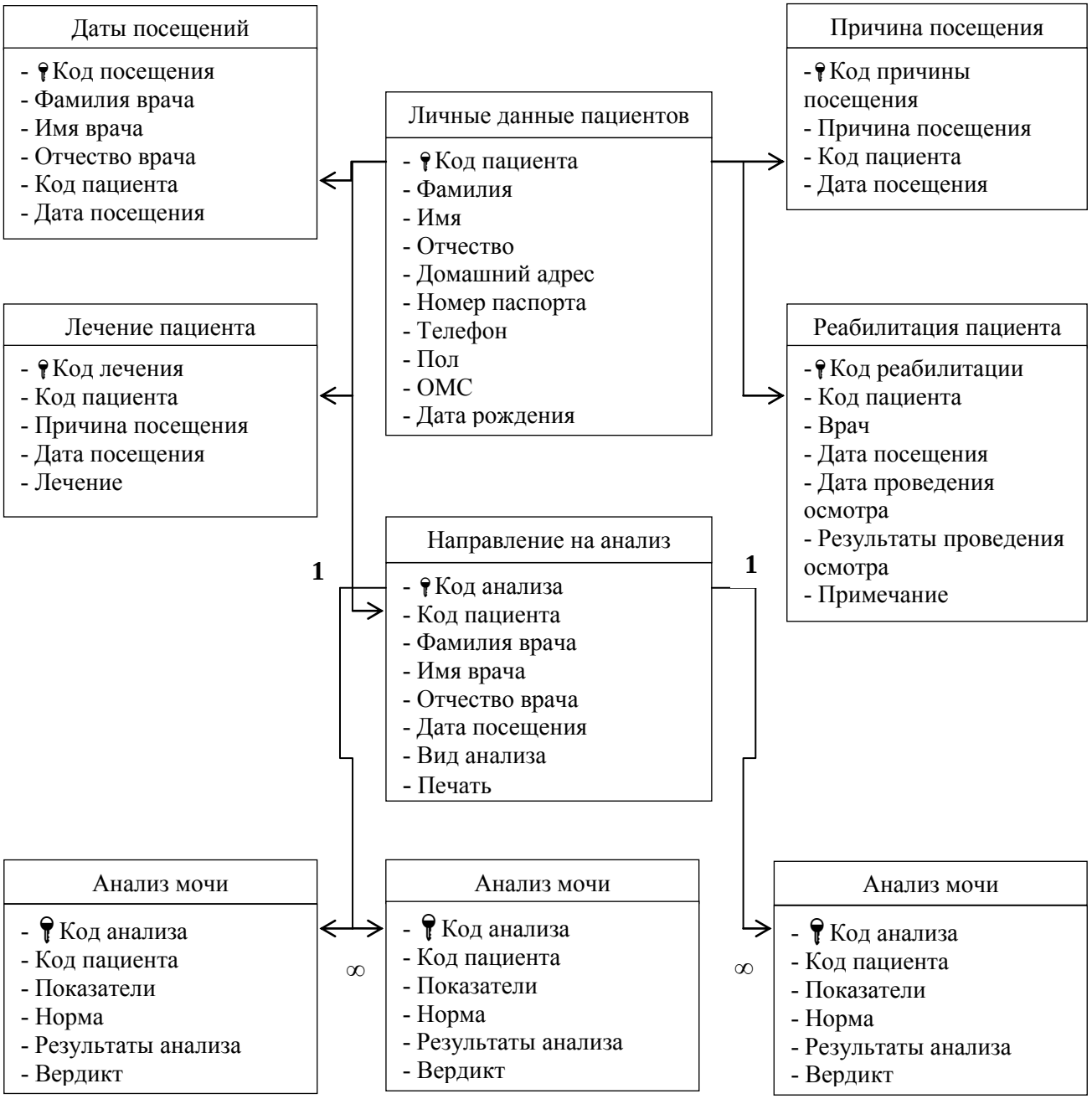


Рисунок 3.9 – *Архитектура данных разрабатываемого приложения*

3.5. Описание разрабатываемого приложения

Для наиболее удобной работы конечного пользователя с программой необходимо создать максимально понятные пользовательские интерфейсы программы. При входе в базу данных пользователю предстоит выбрать, нужно ли ему добавить нового или посмотреть личные данные пациента, либо посмотреть историю его болезни. Первый интерфейс представлен на рисунке 3.10.

Рисунок 3.10 – *Интерфейс базы данных*

Допустим, пользователь решил занести данные о новом пациенте, либо посмотреть информацию о пациенте. Для этого следует нажать на кнопку «Добавление и поиск пациента». На экране откроется окно с информацией о пациенте, которое представлено на рисунке 3.11.

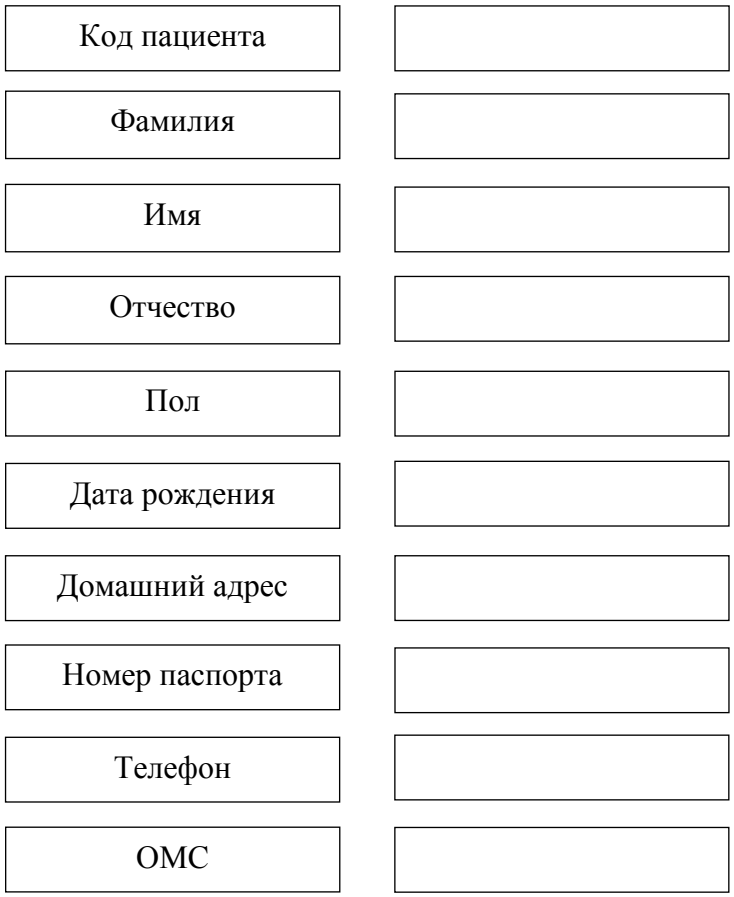


Рисунок 3.11 – *Интерфейс «Добавление и поиск пациента»*

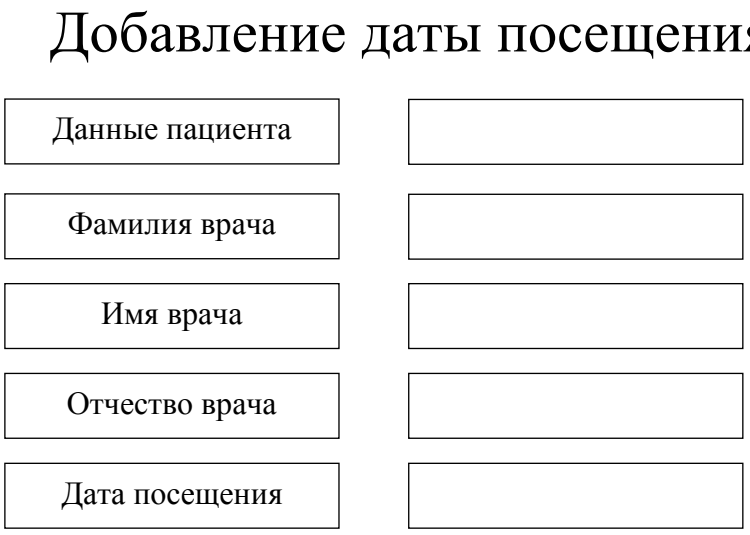


Рисунок 3.12 – *Интерфейс «Добавление даты посещения»*

Если пользователю необходимо добавить историю болезни пациента, либо посмотреть его историю, следует нажать на кнопку «Добавление причины посещения». На экране появится окно с историей болезни соответствующего пациента, представленное на рисунке 3.13. В него вносятся такие данные, как фамилия, имя, отчество пациента, причина посещения, дата посещения. После внесения нужной информации она отобразится в таблице «Причина посещения».

Рисунок 3.13 – *Интерфейс «Добавление причины посещения»*

Далее следует внести информации о лекарствах, которые назначил врач пациенту. Для этого следует нажать на кнопку «Добавить лечение пациента». На экране отобразится окно, представленное на рисунке 3.14. После внесения нужной информации она появится в таблице «Лечение пациента».

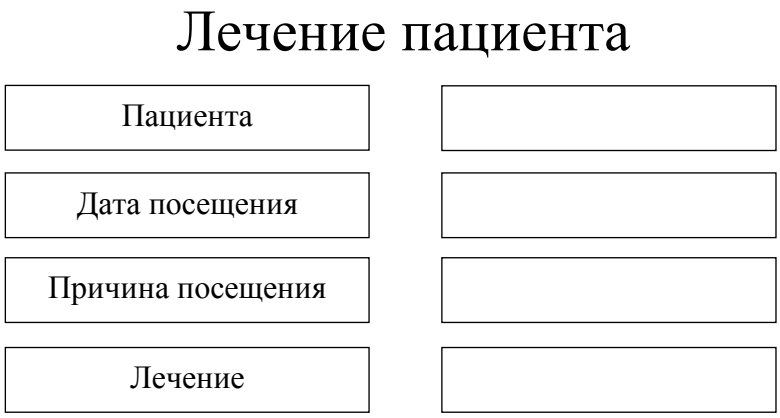


Рисунок 3.14 – *Интерфейс «Добавить лечение пациента»*

Если врач не может точно определить, в чём заключается причина болезни, то пациент направляется на сдачу анализов. Для этого врач-терапевт должен распечатать пациенту направление на анализ. Форма для печати представлена на рисунке 3.15.

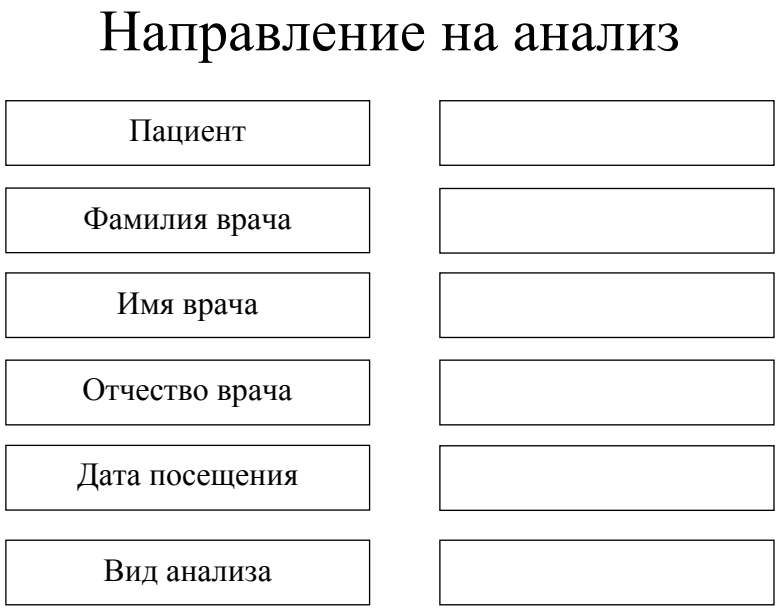


Рисунок 3.15 – *Интерфейс «Направление на анализ»*

После того, как будут готовы результаты анализы пациента, врач может назначить ему лечение. Результаты анализа будут заноситься в форму, представленную на рисунке 3.16. Данная форма представлена для анализа крови. Формы для анализа мочи и кала будет иметь такой же вид.

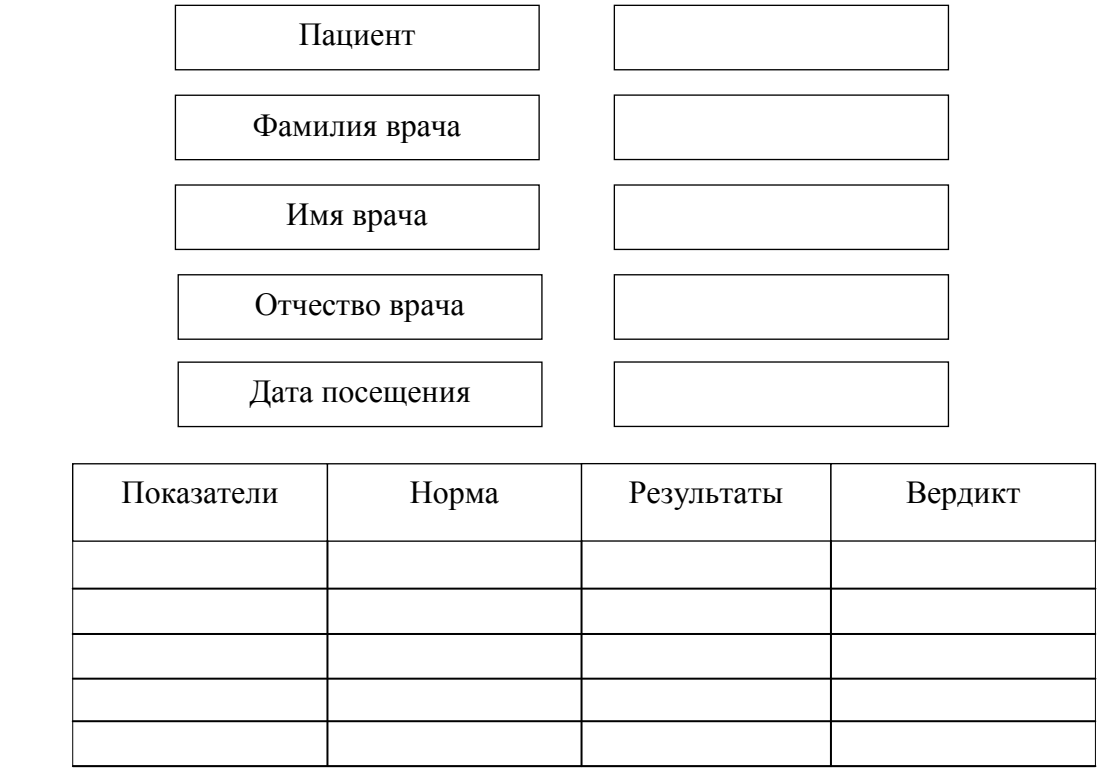


Рисунок 3.16 – *Интерфейс «Анализ крови»*

В случае плохого самочувствия пациента, появления осложнений или плохих анализов, пациента следует положить в палату под наблюдение. Результаты осмотров будут заноситься в форму, представленную на рисунке 3.17. В примечание будет заноситься информация о состоянии пациента.

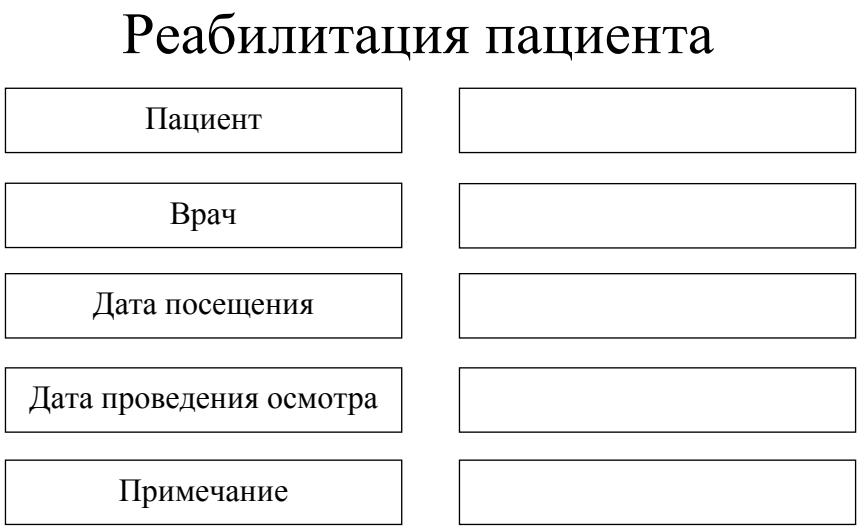


Рисунок 3.17 – *Интерфейс «Реабилитация пациента»*

Для более наглядного представления работы программы, создадим её блок-схему. Компоненты блок-схемы представлены на рисунке 3.18, а сама схема на рисунке 3.19.

Название	Обозначение	Назначение
Действие	Действие	Начало, завершение программы
Процесс	Действие	Обработка данных
Данные	Данные	Операции ввода-вывода
Решение	Условие	Логический блок (блок условия)

Рисунок 3.18 – *Компоненты блок-схемы*

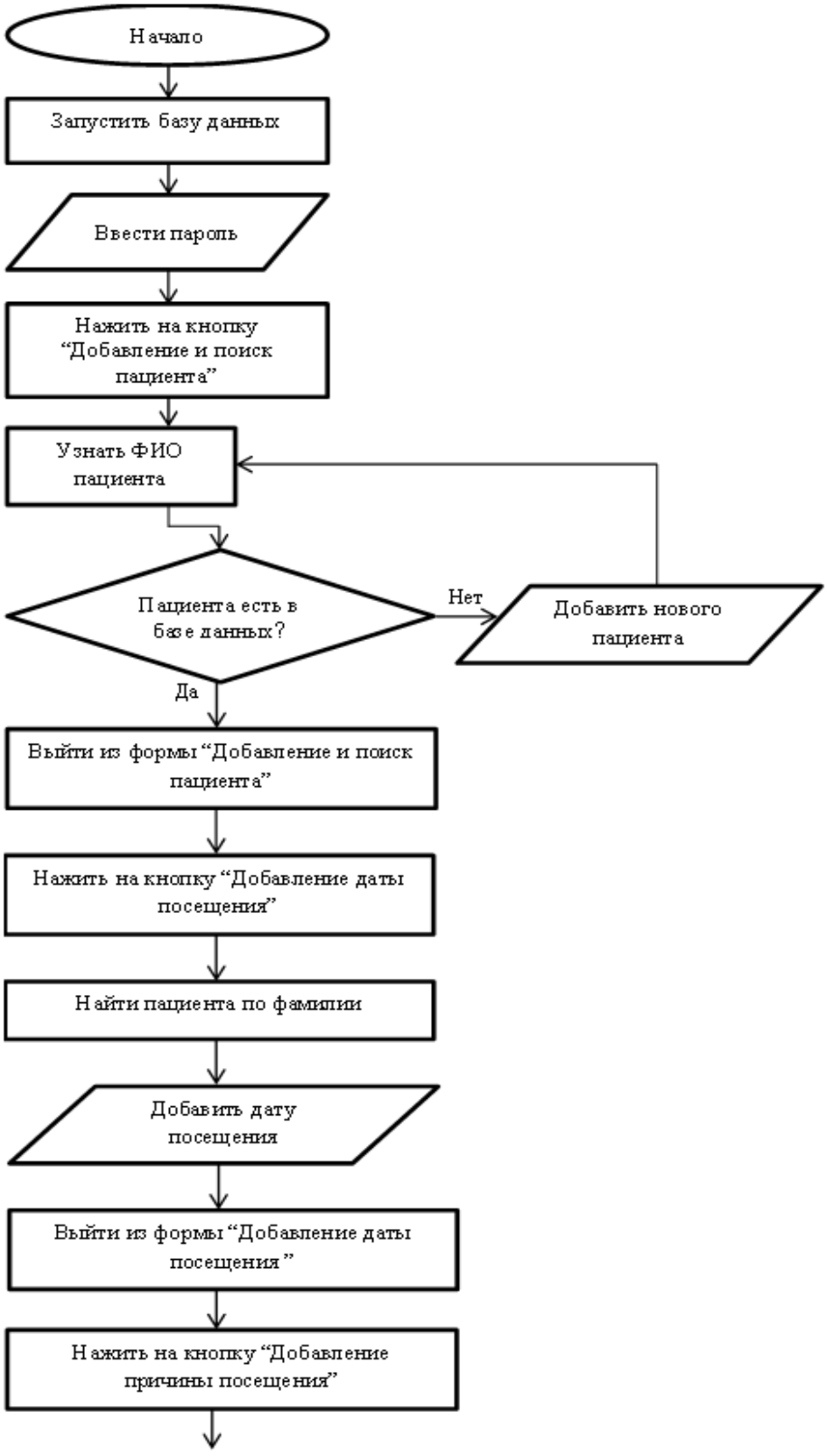


Рисунок 3.19.1 – *Блок-схема базы данных (Часть 1)*

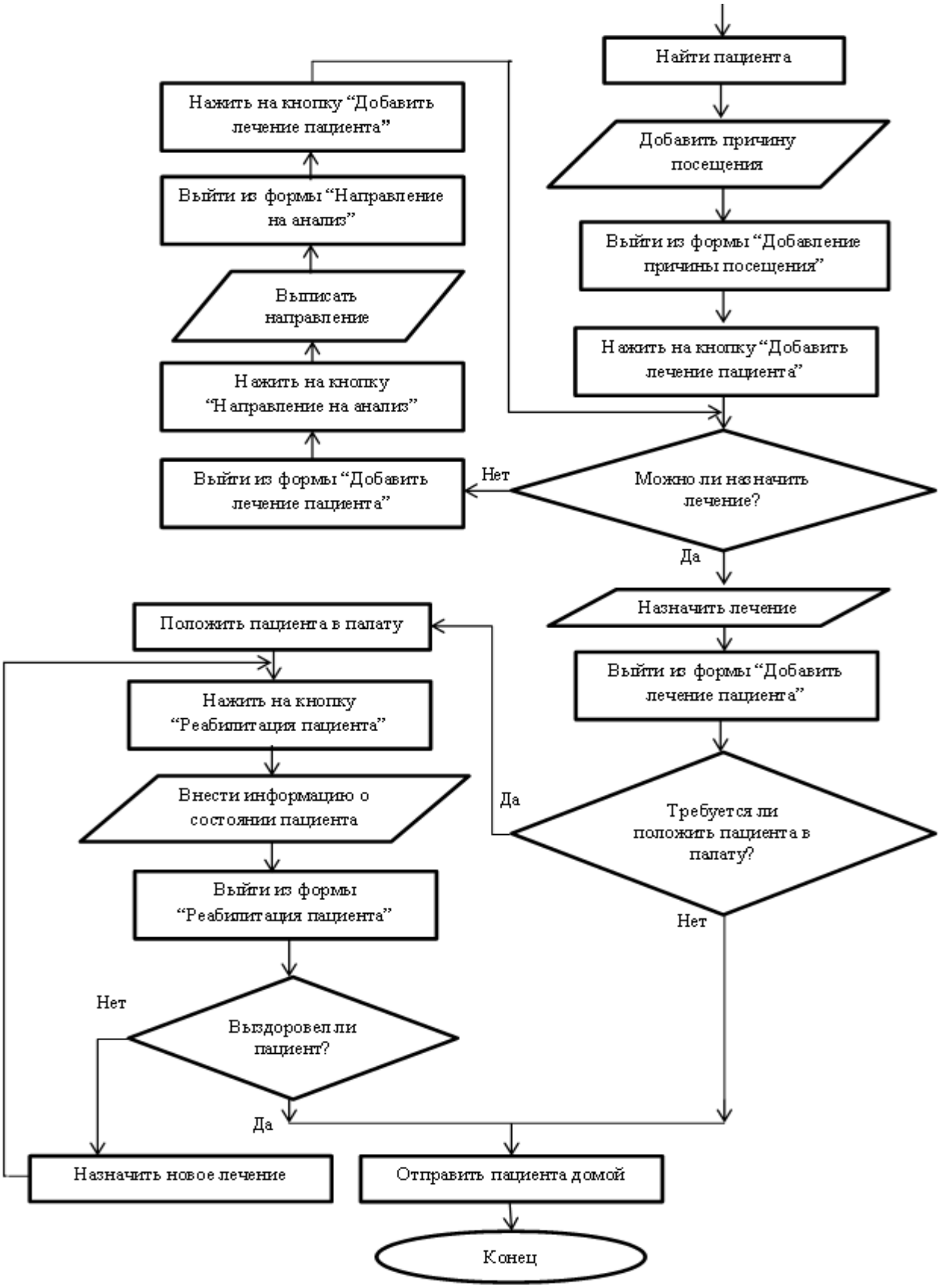


Рисунок 3.19.2 – *Блок-схема базы данных (Часть 2)*

Работа программы будет вестись согласно представленной схеме на рисунке 3.20.

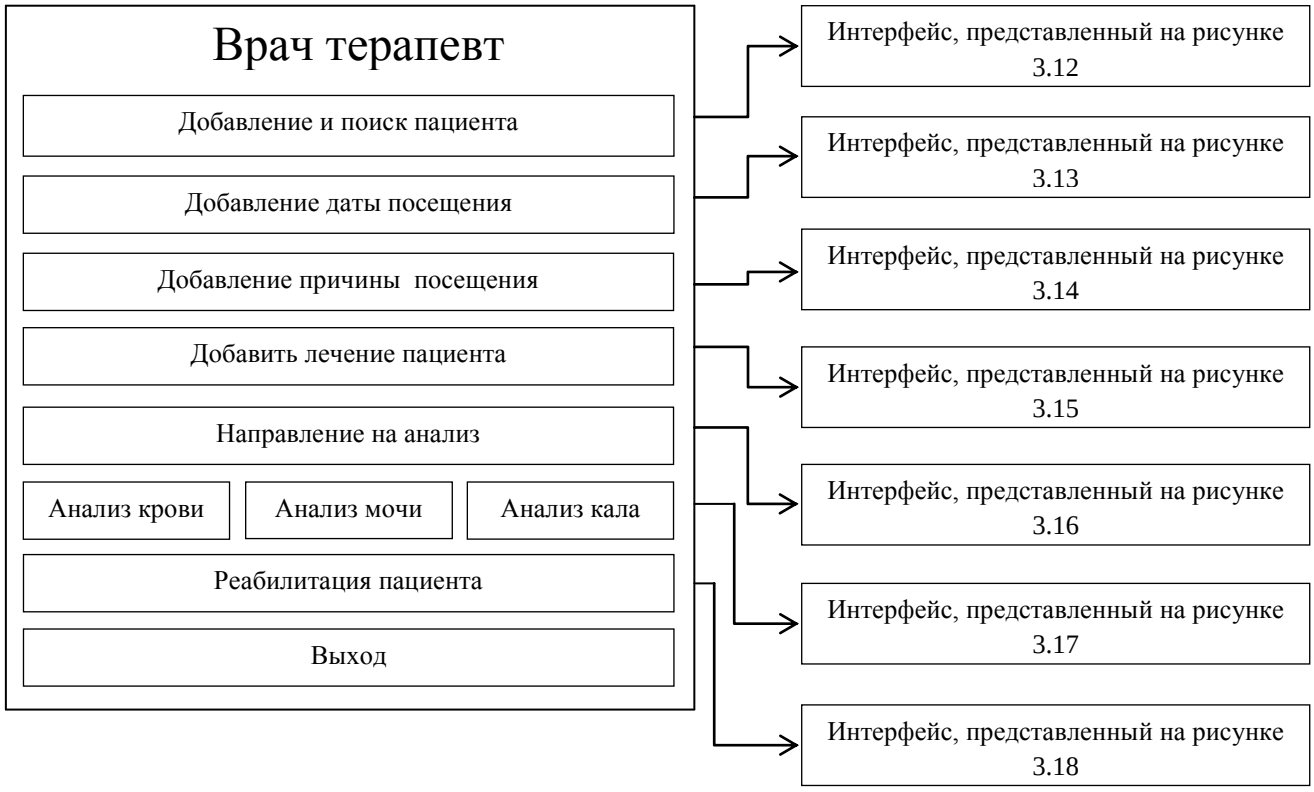


Рисунок 3.20 – *Схема программы*

Раздел 4. Разработка приложения

4.1. Первый прототип программы (I виток спирали)

Реализация программы на первом витке спирали представляет собой обычные таблицу с элементами управления самой среды разработки MS Access. Здесь нет никакого интерфейса, который мог бы облегчить работу пользователя с программой.

Основным риском на данном этапе разработки программы являются возникновение проблем с электропитанием. При возникновении данной проблемы возможен сбой работы программы. Это может привести к отказу работы базы данных, а так же не исключается возможность потери каких-либо данных. Для решения данной проблемы требуется иметь дополнительный источник электропитания/электричества.

На рисунках 4.1 – 4.9 представлены фрагменты таблиц «Личные данные пациента», «Даты посещений», «Причина посещения», «Лечение пациента», «Направление на анализ», «Анализ крови», «Анализ мочи», «Анализ кала», «Реабилитация пациента» созданные в программе MS Access.

Личные данные пациентов

Код пациента	Фамилия	Имя	Отчество	Дата рождения	Домашний адрес	Номер паспорта	Телефон	Пол	ОМС
1	Сидоров	Владимир	Петрович	13.03.1986	ул.Победы,д.8,кв.24	1324 465234	8-(916)-723-64-37	М	3244 3645 6457 6457
Код пациента	Фамилия	Имя	Отчество	Дата рождения	Домашний адрес	Номер паспорта	Телефон	Пол	ОМС
2	Иванов	Сергей	Владимирович	23.08.1991	ул.Севастопольская,д.12,кв.57	1234 748924	8-(903)-235-73-33	М	2323 8597 2983 5758
Код пациента	Фамилия	Имя	Отчество	Дата рождения	Домашний адрес	Номер паспорта	Телефон	Пол	ОМС
3	Кастров	Дмитрий	Игоревич	11.02.1989	ул.Проспект мира,д.20,кв.28	1426 274365	8-(911)-136-23-56	М	3453 4534 9059 3458
Код пациента	Фамилия	Имя	Отчество	Дата рождения	Домашний адрес	Номер паспорта	Телефон	Пол	ОМС
4	Васильев	Олег	Васильевич	24.08.1994	ул.Ленина,д.3,кв.5	1846 192385	8-(952)-375-92-74	М	3450 9345 9834 8505
Код пациента	Фамилия	Имя	Отчество	Дата рождения	Домашний адрес	Номер паспорта	Телефон	Пол	ОМС
5	Ларин	Иван	Сергеевич	05.01.1988	ул.Шлыково,д.31,кв.67	1385 283785	8-(903)-284-75-83	М	6579 8324 9234 8659

Рисунок 4.1 – *Фрагмент таблицы «Личные данные пациента» базы данных*

Даты посещений

Код посещения	Фамилия врача	Имя врача	Отчество врача	код пациента	Дата посещения
4	Паршина	Елена	Викторовна	Сидоров	08.02.2018
Код посещения	Фамилия врача	Имя врача	Отчество врача	код пациента	Дата посещения
5	Паршина	Елена	Викторовна	Ларин	08.03.2018
Код посещения	Фамилия врача	Имя врача	Отчество врача	код пациента	Дата посещения
6	Паршина	Елена	Викторовна	Кастров	16.03.2018
Код посещения	Фамилия врача	Имя врача	Отчество врача	код пациента	Дата посещения
7	Паршина	Елена	Викторовна	Васильев	30.03.2018

Рисунок 4.2 – *Фрагмент таблицы «Даты посещений» базы данных*

Причина посещения

Код причины посещения	Код пациента	Дата посещения	Причина посещения
4	Сидоров	08.02.2018	Болевые ощущения в пальцах, головные боли
Код причины посещения	Код пациента	Дата посещения	Причина посещения
6	Ларин	08.03.2018	Растяжение связок стопы
Код причины посещения	Код пациента	Дата посещения	Причина посещения
7	Кастров	16.03.2018	Повышенная температура, озноб, слабость, болезненное и учащенное мочеиспускание
Код причины посещения	Код пациента	Дата посещения	Причина посещения
8	Васильев	30.03.2018	Боли в животе(в районе пупка), запоры и поносы, вздутие живота, слабость, повышенная утомляемость,
Код причины посещения	Код пациента	Дата посещения	Причина посещения
9	Иванов	04.04.2018	Боли в животе
Код причины посещения	Код пациента	Дата посещения	Причина посещения
10	Кастров	12.04.2018	Ушиб спины

Рисунок 4.3 – *Фрагмент таблицы «Причина посещения» базы данных*

Лечение пациента

Код лечения	Код пациента	Причина посещения	Дата посещения	Лечение
1	Сидоров	Болевые ощущения в пальцах, головные боли	08.02.2018	Вывод из результатов анализа крови: Эритремия. Лечение: миелобромол, гидроксимочевина, курантил
2	Ларин	Растяжение связок стопы	08.03.2018	Для улучшения самочувствия: Ибупрофен, Кетопрофен. Для локального нанесения: гели и мази Фастум, Вольтарен
3	Кастров	Повышенная температура, озноб, слабость, болезненное и учащенное мочеиспускание	16.03.2018	Вывод из результатов анализа мочи: Простатит. Лечение: ципрофлоксацин, эритромицин, доксициклин, ректальные свечи
4	Васильев	Боли в животе(в районе пупка), запоры и поносы, вздутие живота, слабость, повышенная утомляемость, снижение массы тела	30.03.2018	Вывод из результатов анализа кала: Энтероколит. Лечение: Фуразолидон, нифуроксазид, амилаза, лоперамид.
5	Иванов	Боли в животе	04.04.2018	Таблетки: НО-ШПА, Спазмалгон, Метоклопрамид, Ранитидин

Рисунок 4.4 – *Фрагмент таблицы «Лечение пациента» базы данных*

Направление на анализ

Код анализа	Код пациента	Фамилия врача	Имя врача	Отчество врача	Дата посещения	Вид анализа	Печать
1	Сидоров	Паршина	Елена	Викторовна	08.02.2018	Анализ крови	
3	Кастров	Паршина	Елена	Викторовна	16.03.2018	Анализ мочи	
4	Васильев	Паршина	Елена	Викторовна	30.03.2018	Анализ кала	

Рисунок 4.5 – *Фрагмент таблицы «Направление на анализ» базы данных*

Анализ крови

Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
49	Сидоров	Эритроциты (RBC), 10*12 клеток/л	4.1 – 5.5	5	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
50	Сидоров	Гемоглобин (HGB), г/л	130.0 – 171.0	145	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
51	Сидоров	Гематокрит (HCT), %	37.0 – 50.0	55	Отклонение
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
52	Сидоров	Средний объем эритроцита (MCV) , фл	77.0 – 97.0	81	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
53	Сидоров	Содержание гемоглобина в эритроците (MCH) , пг	25.0 – 33.0	31	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
54	Сидоров	Концентрация гемоглобина в эритроците, г/дл	31.0 – 36.0	34	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
55	Сидоров	Цветовой показатель	0.9 – 1.1	1	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
56	Сидоров	Ретикулоциты (RTC), %	0.3 – 1.5	1	Норма

Рисунок 4.6 – *Фрагмент таблицы «Анализ крови» базы данных*

Анализ мочи

Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
48	Кастров	Цвет мочи (COL)	Светло-желтый/Соломенно-желтый	Светло-желтый	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
49	Кастров	Прозрачность (CLA)	Абсолютно прозрачная	Прозрачная	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
50	Кастров	Запах	Неспецифический	Неспецифический	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
51	Кастров	Плотность мочи (SG)	В пределах 1.002 - 1.009	1.004	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
52	Кастров	Реакция мочи (pH)	В пределах 5.0-7.0 (слабокислая)	6.2	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
53	Кастров	Белок (PRO)	Не обнаружен / менее 0	Не обнаружен	Норма

Рисунок 4.7 – *Фрагмент таблицы «Анализ мочи» базы данных*

Анализ кала

Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
23	Васильев	Форма	Оформленный	оформленный	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
24	Васильев	Консистенция	Плотный/Мягкий	Мягкий	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
25	Васильев	Цвет	Коричневый	Коричневый	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
26	Васильев	Запах	Нерезкий	Нерезкий	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
27	Васильев	Кислотность	Нейтральная	Нейтральная	Норма
Код анализа	Код пациента	Показатели	Норма	Результаты анализа	Вердикт
28	Васильев	Соединительная ткань	Не обнаружена	Не обнаружена	Норма

Рисунок 4.8 – *Фрагмент таблицы «Анализ кала» базы данных*

Реабилитация пациента

Код реабилитации	Код пациента	Врач	Дата посещения	Дата проведения осмотра	Результат проведения осмотра	Примечание
1	Сидоров	Паршина	08.02.2018	18.02.2018	Ухудшение состояния	Назначить новое лечение: гепарин, мальтофер, аллопуринол, антуран. Продолжить наблюдение.
Код реабилитации	Код пациента	Врач	Дата посещения	Дата проведения осмотра	Результат проведения осмотра	Примечание
2	Кастров	Паршина	16.03.2018	24.03.2018	Улучшение состояния	Продолжить лечение на дому. Выписать из поликлинники.
Код реабилитации	Код пациента	Врач	Дата посещения	Дата проведения осмотра	Результат проведения осмотра	Примечание
3	Васильев	Паршина	30.03.2018	10.04.2018	Состояние не меняется	Назначить дополнительные препараты: лизобакт, имудон, септолете плюс, анти -ангин. Продолжить наблюдение.
Код реабилитации	Код пациента	Врач	Дата посещения	Дата проведения осмотра	Результат проведения осмотра	Примечание
5	Сидоров	Паршина	08.02.2018	25.02.2018	Улучшение состояния	Продолжить лечение на дому. Выписать из поликлинники.
Код реабилитации	Код пациента	Врач	Дата посещения	Дата проведения осмотра	Результат проведения осмотра	Примечание
6	Васильев	Паршина	30.03.2018	20.04.2018	Улучшение состояния	Продолжить лечение на дому. Выписать из поликлинники.

Рисунок 4.9 – *Фрагмент таблицы «Реабилитация пациента» базы данных*

После того, как все нужные таблицы были созданы, мы создаем схему и связи всех используемых таблиц данных, необходимых для приложения. Схема данных, представленная на рисунке 4.10, строилась путем добавления таблиц данных, выделением из них ключевых полей, необходимых для создания связей между ними. Все это выполнялось в режиме «конструктора».

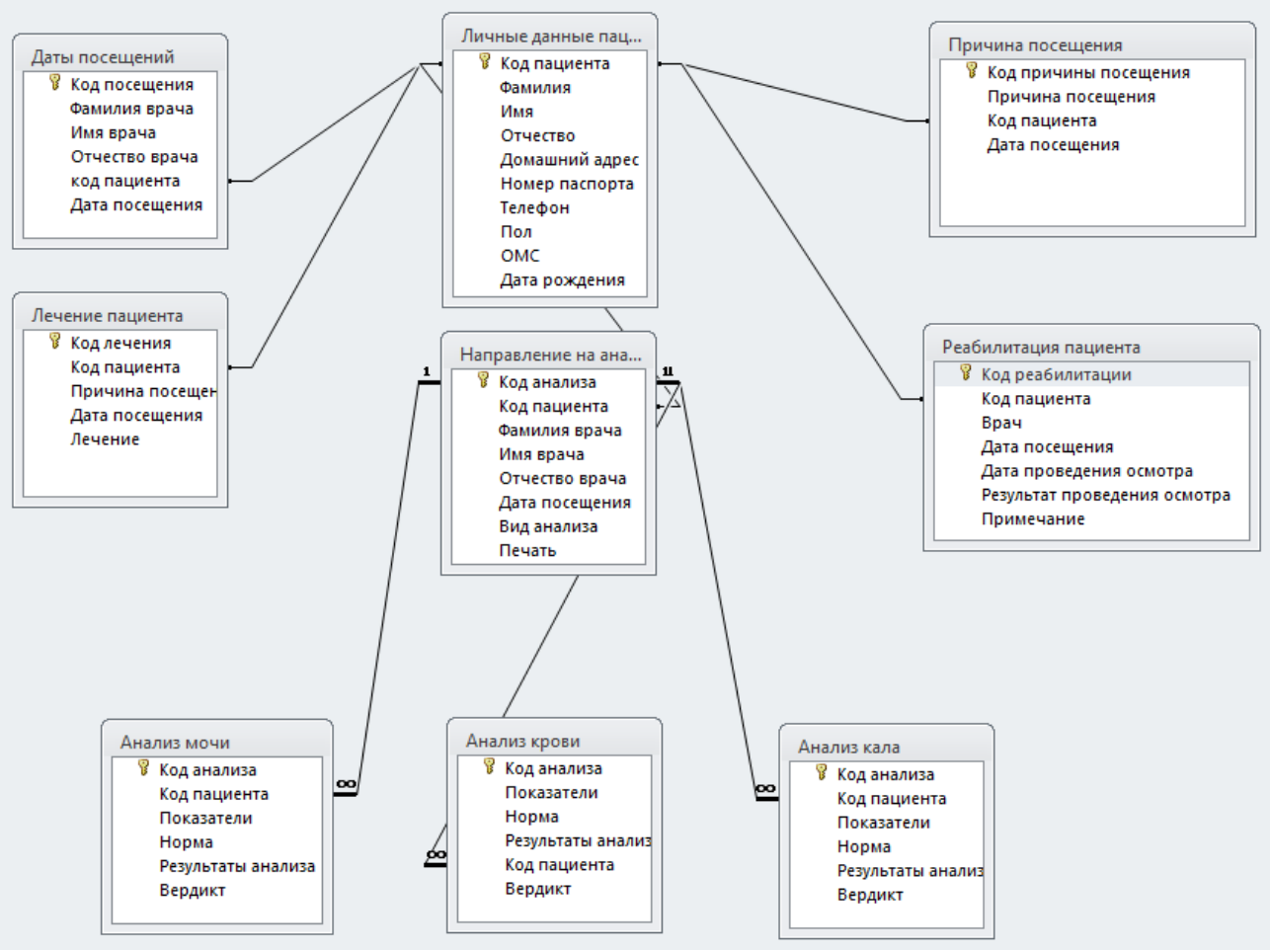


Рисунок 4.10 – *Схема данных*

4.2. Второй прототип программы (II виток спирали)

На данном этапе реализовывается возможность просмотра данных, а также возможность изменения, удаления, внесения и поиска информации о пациенте, а также для печати (рисунки 4.11 – 4.19). Для чего создаётся «Форма»

– объект, с помощью которого пользователи могут добавлять, редактировать и отображать данные.

Основным риском на данном этапе разработки программы является человеческий фактор-невнимательность врача при внесении данных пациента. Со стороны пользователя ПО – врача, необходимо быть более внимательным при работе с персональными данными пациентов.

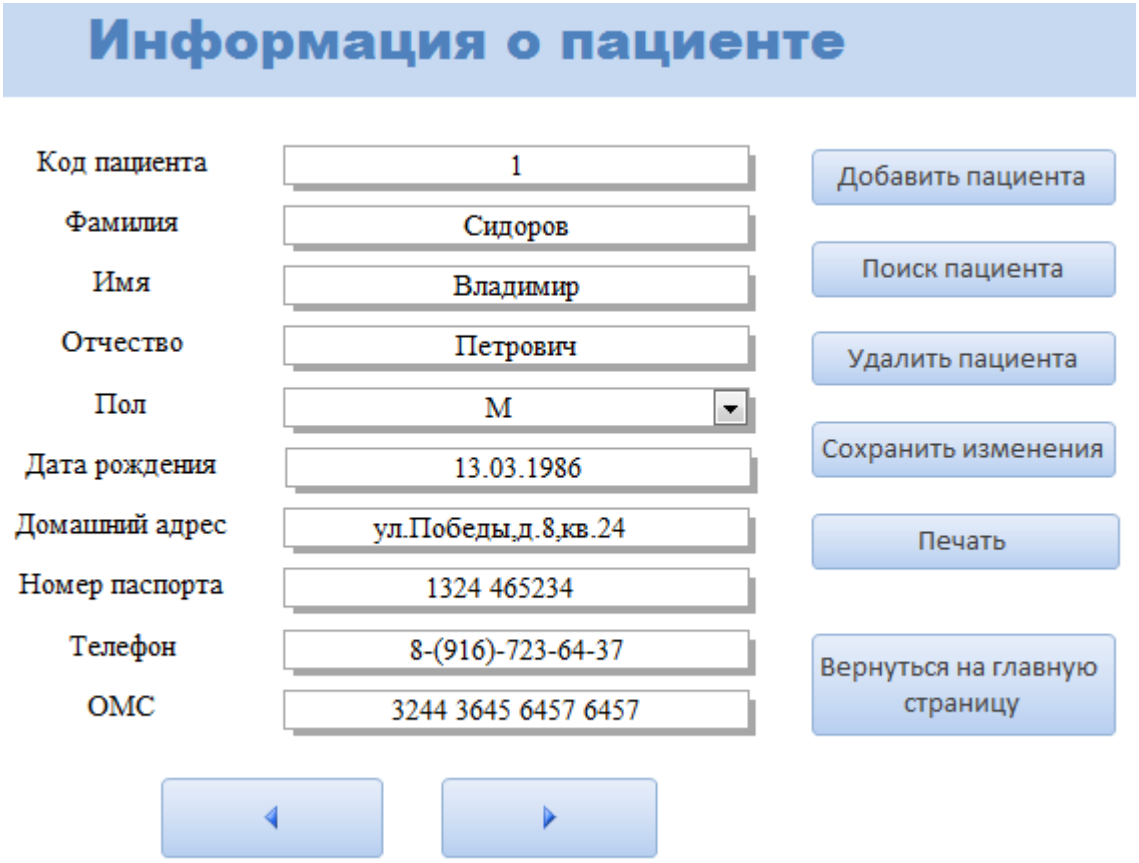


Рисунок 4.11 – *Форма «Добавление пациента» с возможностью поиска, удаления, добавления пациентов и печати информации*

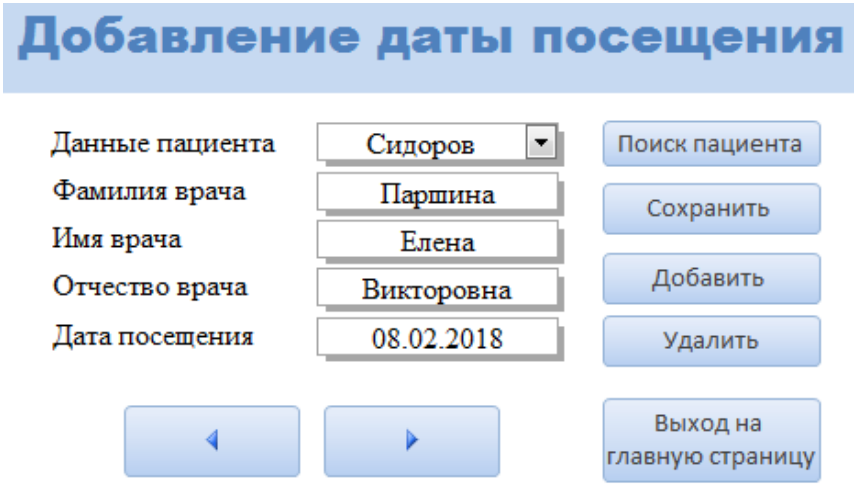


Рисунок 4.12 – *Форма «Добавление даты посещения» с возможностью поиска, добавления и удаления даты посещения пациента*

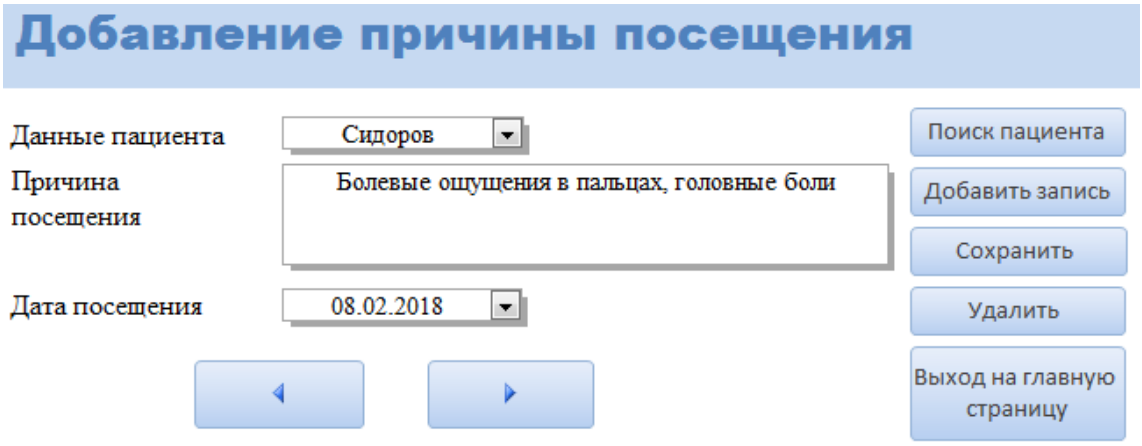


Рисунок 4.13 – *Форма «Добавление причины посещения» с возможностью поиска, добавления и удаления причины посещения пациента*

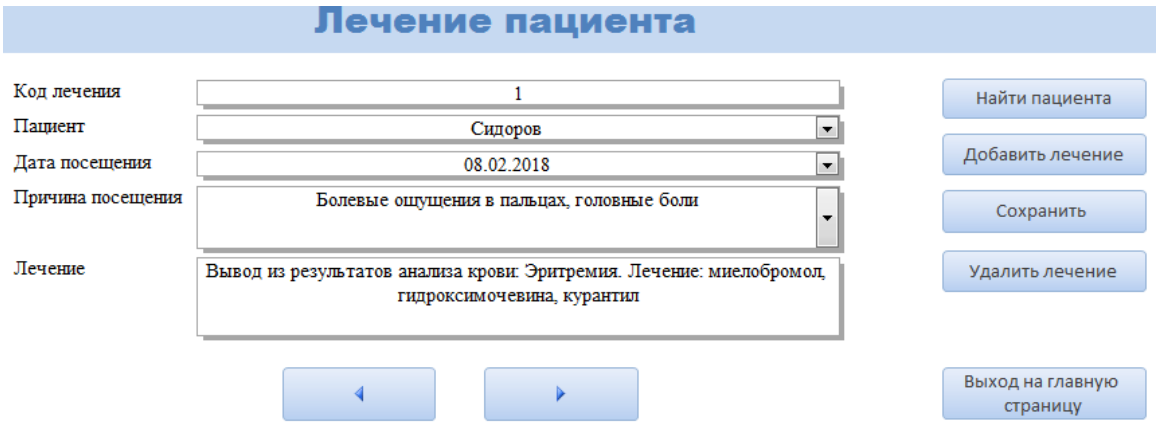


Рисунок 4.14 – *Форма «Лечение пациента» с возможностью поиска, добавления и удаления информации о лечении пациента*

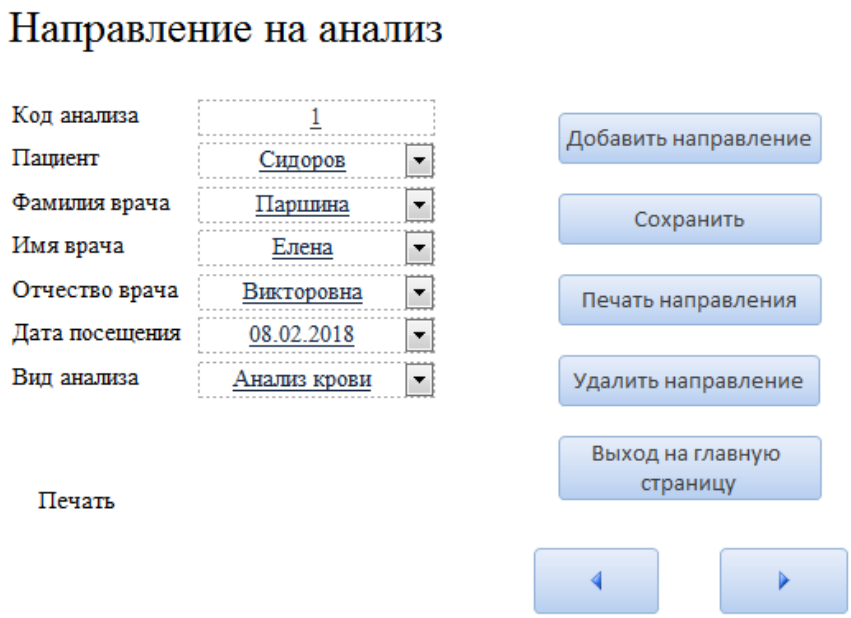


Рисунок 4.15 – *Форма «Направление на анализ» с возможностью поиска, добавления, удаления и печати направления на анализ пациента*

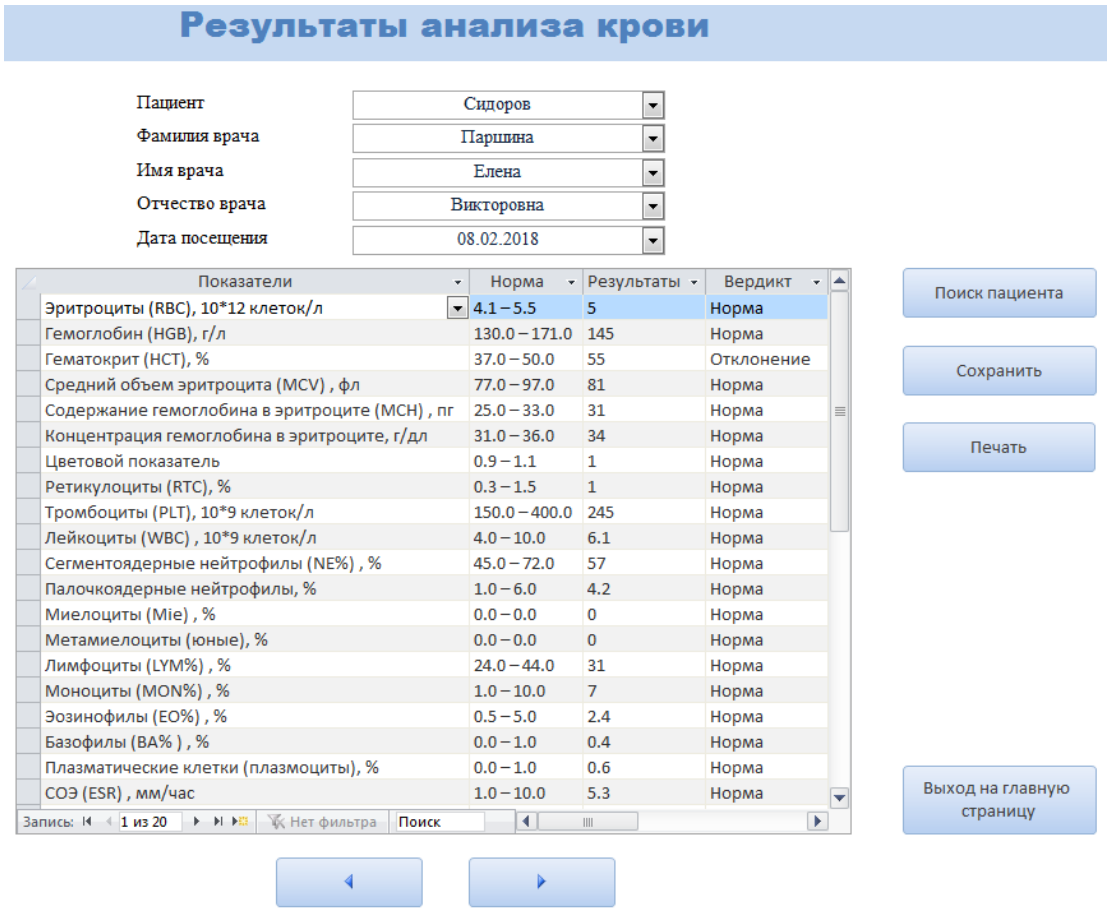


Рисунок 4.16 – *Форма «Результаты анализа крови» с возможностью поиска и печати результатов анализа пациента*

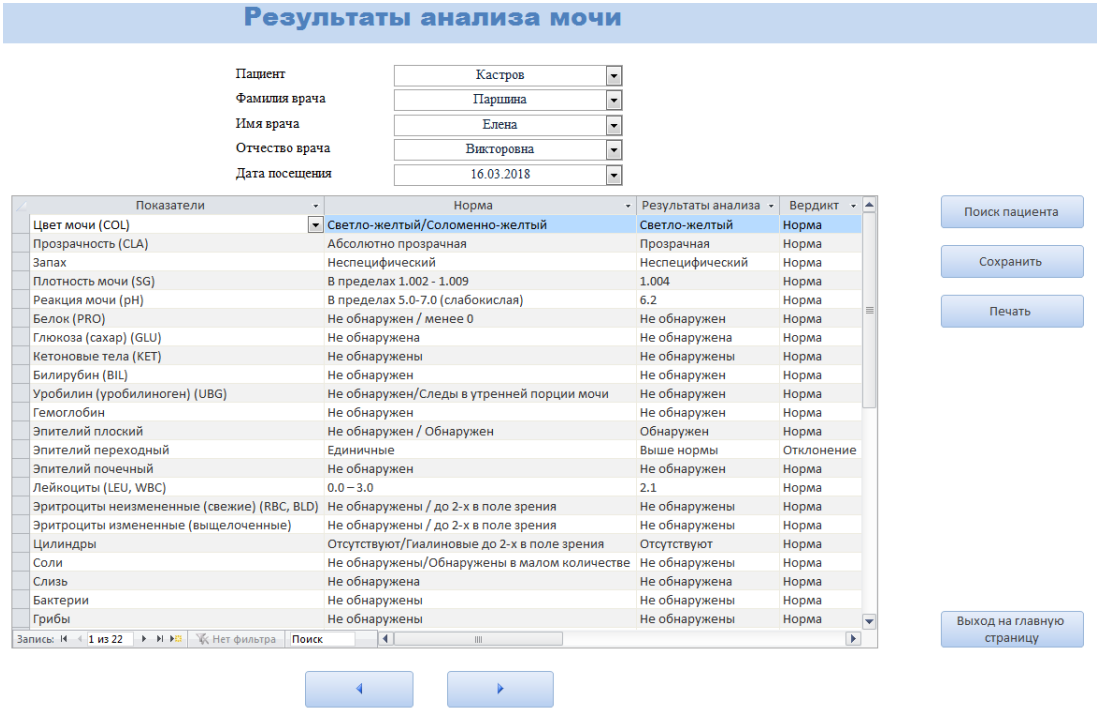


Рисунок 4.17 – *Форма «Результаты анализа мочи» с возможностью поиска и печати результатов анализа пациента*

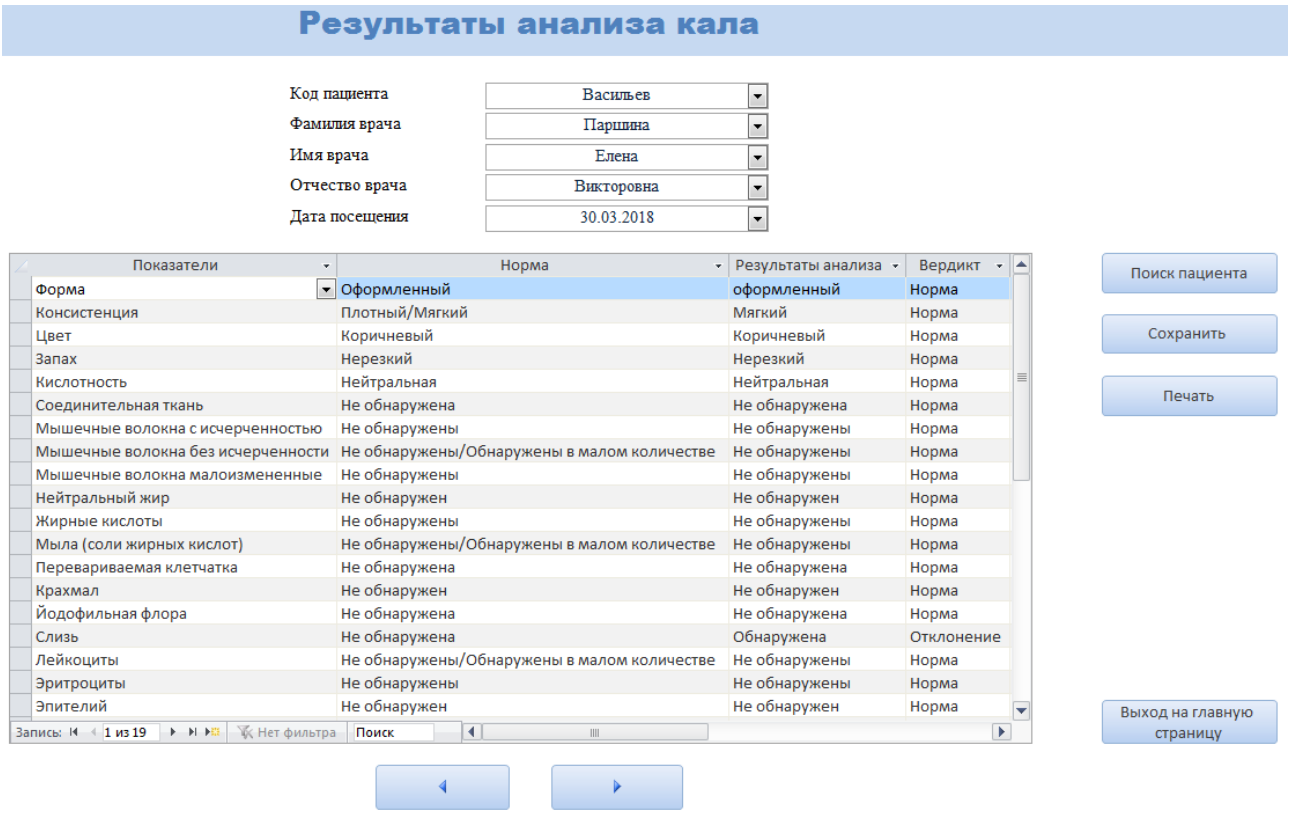


Рисунок 4.18 – *Форма «Результаты анализа кала» с возможностью поиска и печати результатов анализа пациента*

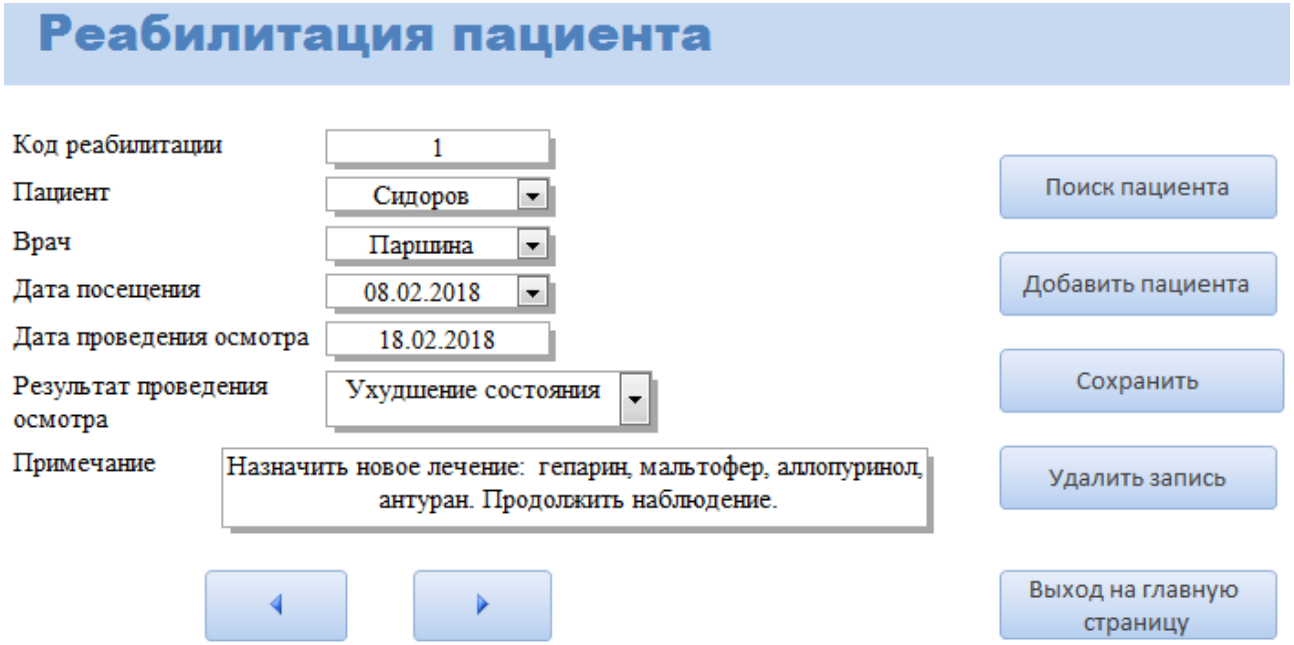


Рисунок 4.19 – *Форма «Реабилитация пациента» с возможностью поиска, добавления и удаления истории реабилитации пациента*

4.3. Конечный продукт (III виток спирали)

Следующим шагом требовалось защитить данные пациентов. Для этого следовало поставить пароль на базу данных (рисунок 4.20). Основным риском на данном этапе разработки программы является введение слишком легкого /простого пароля, а также небрежного отношения со стороны пользователей за соблюдением условий работы в ПО. Это может привести к входу в базу данных посторонних лиц. Для решения данной проблемы требуется тщательно подходить к выбору пароля, а врачу-пользователю внимательно подходить к хранению и использованию служебной информации (пароли, коды) и персональных данных пациентов.

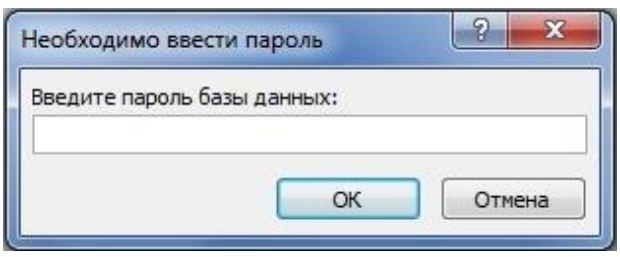


Рисунок 4.20 – *Окно для ввода пароля*

В случае если пароль будет введён неправильно, то программа не даст пользователю возможность войти в базу данных и выведет на экран окно оповещения, представленное на рисунке 4.21.

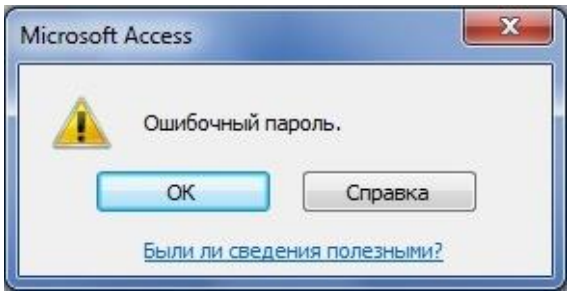


Рисунок 4.21 – *Окно, оповещающее о неправильном вводе пароля*

Так же следует создать интерфейс для ускорения приёма пациентов и упрощения работы пользователя (рисунок 4.22).

Рисунок 4.22 – *Главное меню приложения*

Первая кнопка – «Добавление и поиск пациента», при нажатии на эту кнопку происходит переход в форму с информацией о пациентах. Форма представлена на рисунке 4.11. В ней можно произвести поиск информации о нужном нам пациенте, добавить нового при нажатии на кнопку «Добавить пациента», удалить пациента, либо отредактировать данные, которые уже существуют в этой форме.

Вторая кнопка – «Добавление даты посещения». При нажатии на неё происходит переход в форму с датами посещений пациентов. В ней можно посмотреть даты посещений пациентов и добавить новые. Данная форма представлена на рисунке 4.12.

Третья кнопка – «Добавление причины посещения». При нажатии на неё происходит переход в форму с причинами посещений всех пациентов. В ней можно посмотреть историю болезней пациентов, а также добавить новые. Данная форма представлена на рисунке 4.13.

Четвертая кнопка – «Добавить лечение пациента». При нажатии на неё происходит переход в форму со всей информацией об истории лечения пациентов. В ней можно посмотреть историю лечения пациента, а также добавить новое, в случае необходимости. Данная форма представлена на рисунке 4.14.

Пятая кнопка – «Направление на анализ». В этой форме можно сделать направление пациенту на какой-либо анализ, распечатать его и поставить печать. Данная форма представлена на рисунке 4.15.

Шестая - восьмая кнопки – «Анализ крови/мочи/кала». При нажатии на эти кнопки будет происходить переход в формы результатами анализов пациентов. В ней можно посмотреть информацию о результатах анализов для принятия решения о том, чем производить лечение пациента. Данные формы представлены на рисунках 4.16 – 4.18.

Девятая кнопка – «Реабилитация пациента». При нажатии на эту кнопку будет происходить переход в форму, содержащую информация о том, как происходить реабилитация пациента. В ней можно посмотреть как происходит реабилитация пациента. В случае осложнения/ухудшения состояния назначить новые лекарства. Данная форма представлена на рисунке 4.19. Десятая кнопка – «Выход». При нажатии на неё происходит выхода из приложения.

Раздел 5. Тестирование разработанного приложения

Тестирование программного обеспечения – процесс анализа программного продукта и сопутствующей документации с целью выявления недостатков в работе и повышения его качества [13]. Обычно для проверки работоспособности разработанной программы проводят три вида тестирования: функциональное, интеграционное и нагрузочное.

Функциональное тестирование (Functional testing) – вид тестирования, направленный на проверку корректности работы функциональности приложения (корректность реализации функциональных требований). Описание разработанных модулей программы, соответствующих требованиям (табл. 2.3) приведено в п. 4.1 – 4.3. Функциональное тестирование данных модулей показало их полную работоспособность и отсутствие ошибок, т.е. все функциональные требования были успешно реализованы.

Интеграционное тестирование (integration testing) направлено на проверку взаимодействия между несколькими частями приложения. Данное тестирование проводится в случае, когда происходит взаимодействие модулей программы (II виток спирали). Данная проверка оказалась успешной. В частности, в процессе записи пациента на прием происходит добавление в таблицу «Личные данные пациентов» данных, введенных в форму для добавления пациента (рис.4.5). Данный процесс выполняется без сбоев в работе программы. Аналогично можно рассмотреть процесс добавления причины посещения пациента. В процессе добавления причины посещения для пациента с помощью нужной формы (рис.4.6), соответствующая запись появляется в таблице «Причины посещения». Данный процесс также работает без сбоев.

Нагрузочное тестирование (load testing, capacity testing) – исследование способности приложения сохранять заданные показатели качества при нагрузке

в допустимых пределах и некотором превышении этих пределов (определение «запаса прочности») [14]. Нагрузочное тестирование проводилось для исследования быстродействия таких процессов, как, например, запись пациента на прием (поскольку этот процесс призван уменьшить время ожидания пациентов в очередях, то предполагается, что его выполнение не должно занимать много времени) и поиск записи. Для тестирования было выбрано следующее количество записей: 5 (в среднем посетителей за час работы), 50 (в среднем посетителей за день) и 500 (в среднем посетителей на 2 недели). Результаты нагрузочного тестирования приведены в табл.6.

Вначале 5 раз проводится проверка отклика системы на различные действия, ее результаты вносятся в соответствующие строки столбцов t1-t5. Далее рассчитывается среднее арифметическое всех измерений по формуле (1):

$$t_{\text{ср.ариф.}} = \frac{\sum_{i=1}^{n} t_i}{n} \quad (1)$$

Результат заносится в столбец «Среднее время отклика» таблицы. В следующем столбце находятся соответствующие средние квадратические отклонения, рассчитанные по формуле (2):

$$\sigma = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} \left(t_i - t_{\text{ср.ариф.}}\right)^2} \quad (2)$$

Погрешность измерений рассчитывалась по формуле (3) и заносилась в столбец «Погрешность измерений», где n – число измерений, $t_{\alpha(N-1)}$ – доверительный коэффициент Стьюдента, равный 0.95, Δt_p – абсолютная погрешность электронного секундомера равная 0,005.

$$\Delta t = \sqrt{\left(\frac{\sigma}{n} \cdot t_{\alpha(N-1)}\right)^2 + \Delta t_p{}^2}. \quad (3)$$

В столбце «Время отклика» – итоговое время отклика. Оно рассчитывалось по формуле (4).

$$t_{\text{откл}} = t_{\text{ср.ариф.}} \pm \Delta t. \quad (4)$$

Таблица 6 – *Результаты нагрузочного тестирования*

Количество записей	Действие	t1, с.	t2, с.	t3, с.	t4, с.	t5, с.	Среднее время отклика, с.	Средн. квадр. отклон., с.	Погрешность измерений, с.	Время отклика, с.
5	Добавление	0,21	0,23	0,21	0,23	0,2	0,216	0,0235	0,0254	0,216±0,025
	Поиск	0,2	0,22	0,19	0,21	0,2	0,206	0,0225	0,0243	0,206±0,024
50	Добавление	0,3	0,29	0,31	0,31	0,3	0,302	0,0341	0,0369	0,302±0,037
	Поиск	0,27	0,26	0,28	0,29	0,3	0,28	0,0339	0,0358	0,28±0,036
500	Добавление	0,39	0,38	0,39	0,37	0,4	0,385	0,0376	0,0394	0,385±0,04
	Поиск	0,36	0,35	0,36	0,34	0,4	0,362	0,0369	0,0389	0,362±0,039

Как видно из табл. 6, время отклика не превышает 0,5 с и почти не замечается пользователем. Таким образом, можно сделать вывод, что нагрузочное тестирование проведено успешно.

Заключение

Целью данной работы являлось создание автоматизированного рабочего места для врача-терапевта. Необходимо было оптимизировать рабочий процесс с целью экономии времени сотрудника и уменьшение времени приема пациентов. Таким образом, в рамках данной работы были выполнение следующие действия:

- В процессе работы были проанализированы пользовательские требования (19 требований), полученные путем опроса и изучения документации.
- Были определены приоритеты полученных требований.
- На основе пользовательских требований были составлены функциональные требования, которые должны быть реализованы в разрабатываемом приложении.
- Составлен список требований, связывающий вместе пользовательские, функциональные требования, их приоритет, а также программные компоненты, отвечающие за реализацию данных требований;
- После формирования списка требований произошло деление всех требований на 1-3 уровни спирали.
- После того, как требования были проанализированы, проводилось составление моделей AS-IS и TO-BE ключевых бизнес-процессов организации в нотациях ARIS VACD (0-1 уровни) и UML Activity Diagram (2-3 уровень), и проведено их сопоставление на каждом уровне. Это позволило определить структуру разрабатываемого приложения.
- Разработка приложения производилась в среде MS Access, где поэтапно были реализованы все указанные требования, исправлены

недочеты и ошибки в работе программы. Эта программа является простой и понятной в обращении; является полностью совместимой с системой Windows и обладает огромными возможностями по импорту и экспорту данных.

- После каждого этапа (витка спирали) разработки проводилось функциональное тестирование разработанных модулей программы, показавшее их работоспособность и соответствие заявленным требованиям.
- При завершении заключительного этапа разработки проведено интеграционное тестирование для проверки взаимодействия программных модулей, также показавшее их полную работоспособность.
- Затем было проведено нагрузочное тестирование при различном количестве записей в БД, свидетельствующее о высокой производительности программы.

Список использованных литературных источников

1. Е.А. Кумагина, Е.А. Неймарк. «Модели жизненного цикла и технологии проектирования программного обеспечения». Учебно-методическое пособие, Нижний Новгород, 2016 - 250 с.

2. Орлов С. А. Технологии разработки программного обеспечения. Учебник для вузов. 4-е издание. Стандарт третьего поколения / С. А. Орлов, Б. Я. Цилькер. – М.: Издательский дом «Питер», 27 февр. 2012 г. – 608 с.

3. Елочкин М.Е., Брановский Ю.С., Николаенко И.Д. «Информационные технологии». Издательство «Оникс», Москва, 2009. 389 с.

4. Родигин Л. А. Оценка совокупной стоимости владения туристским интернет-проектом. / Л. А. Родигин, К. В. Наймарк. // Litres. – 5 сент. 2017 г.

5. Boehm B.W. A spiral model of software development and enhancement / Boehm B., Egyed A. // IEEE Computer, May 1988, 250 с.

6. А.В. Варзунов, Е.К. Торосян, Л.П. Сажнева. «Анализ и управление бизнес-процессами» .Учебное пособие, 2010 – 212 с.

7. И.В.Абрамов. Методические указания по дисциплине «Модели и методы информационно-управляющих систем». Ижевск, 2004 – 314 с.

8. Владимир Репин, Виталий Елиферов. «Процессный подход к управлению». Моделирование бизнес-процессов. Издательство «Манн, Иванов и Фербер», Москва, 2013 – 215 с.

9. Ю.А.Ларина. «Основы объектно ориентированного моделирования с использованием языка UML». Учебное пособие, Ярославль, 2010 – 152 с.

10. Ю.А.Ларина. «Основы объектно ориентированного моделирования с использованием языка UML». Учебное пособие, Ярославль, 2010 – 152 с.

11. Дейт К. Дж. Введение в системы баз данных / пер. с англ. и ред. К. А. Птицына – 8-е изд. – М.: Вильямс – 2016. – 327 с.

12. Епанешников А.М., Епанешников В.А. Практика создания приложений в Access. – 2009. – 440 с.

13. Штенников Д.Г. Разработка информационных систем в образовании. Учебное пособие. – СПб: СПбГУ ИТМО, 2012. – 242 с.

14. Куликов С. С. Тестирование программного обеспечения //Базовый курс: практ. пособие. Минск: Четыре четверти. – 2015. – 136 с.

ПРИЛОЖЕНИЕ А

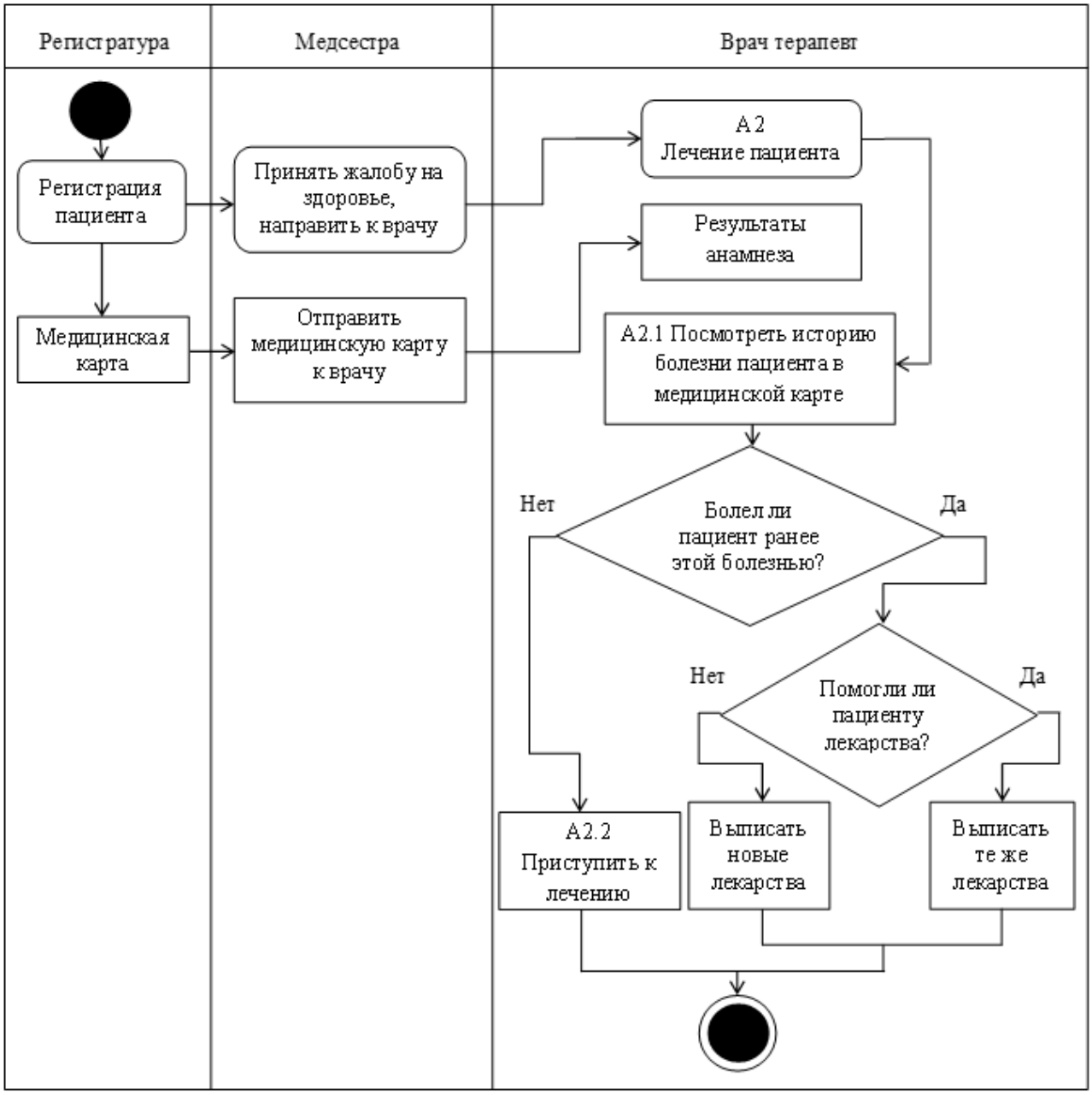


Рисунок 8.1 – *Представление процесса в UML AD на втором уровне в модели «AS-IS» для уточнения процесса «Лечение пациента»*

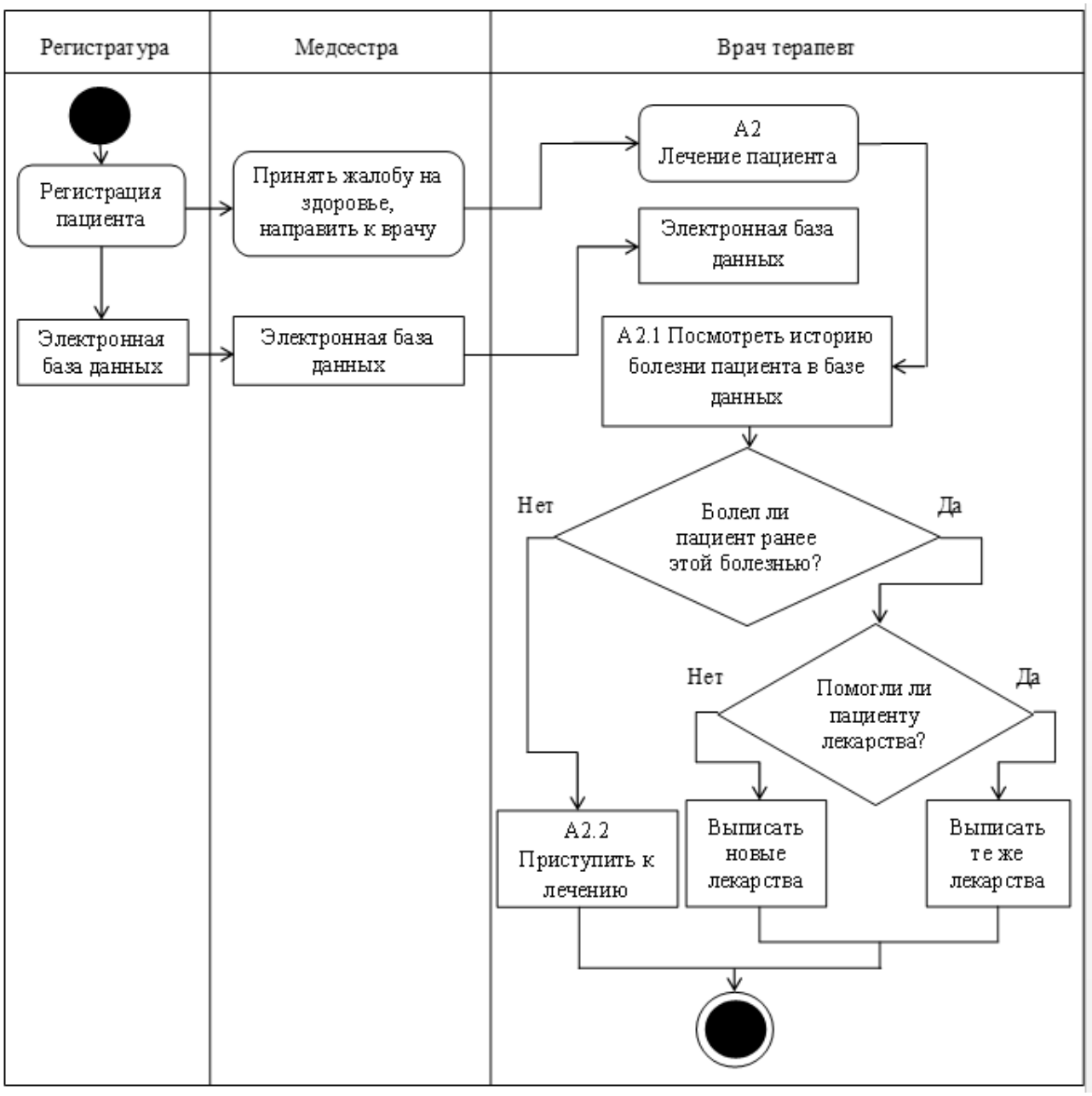


Рисунок 8.2 – *Представление процесса в UML AD на втором уровне в модели «TO-BE» для уточнения процесса «Лечение пациента»*

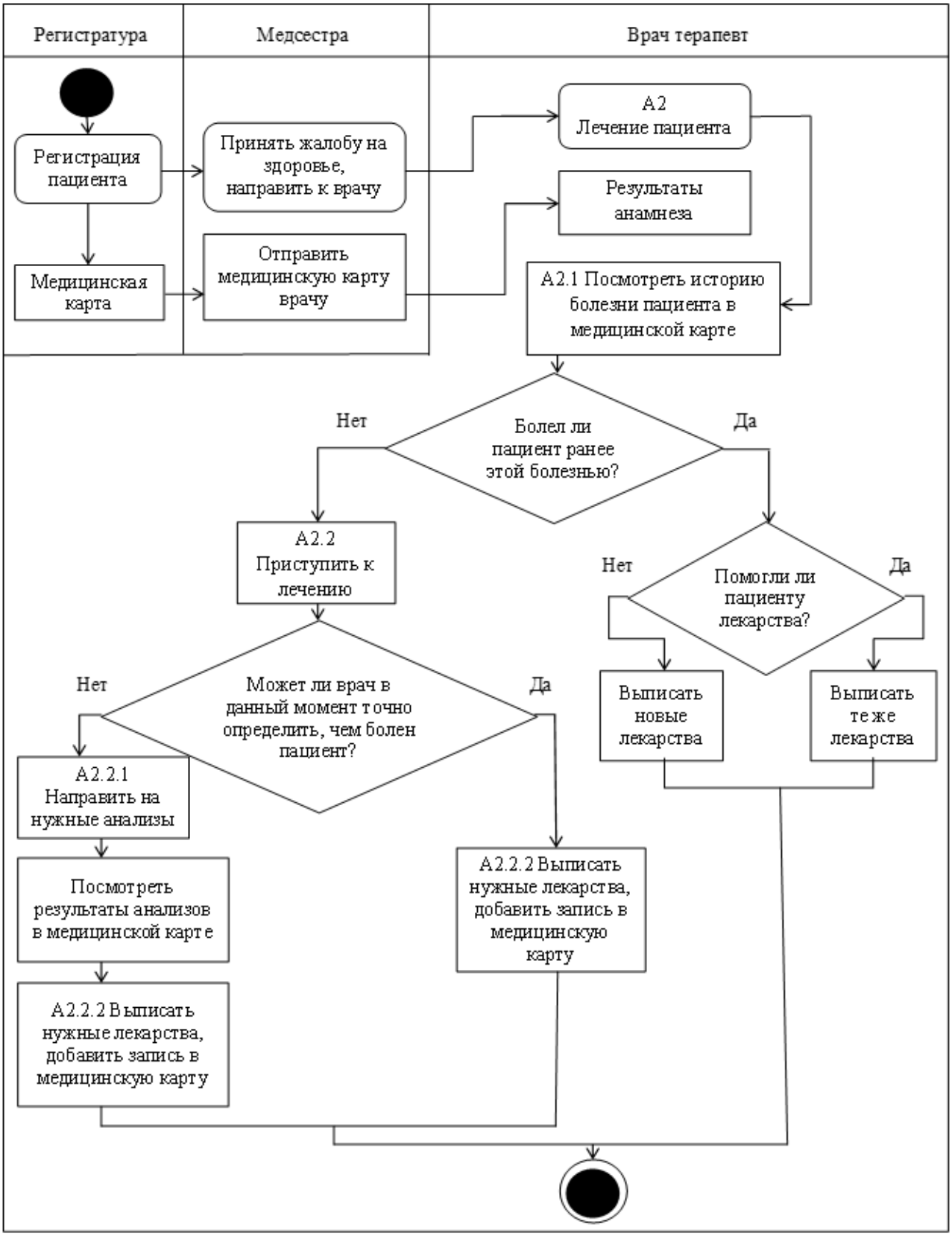


Рисунок 8.3 – *Представление процесса в UML AD на третьем уровне в модели «AS-IS» для уточнения процесса «Приступить к лечению»*

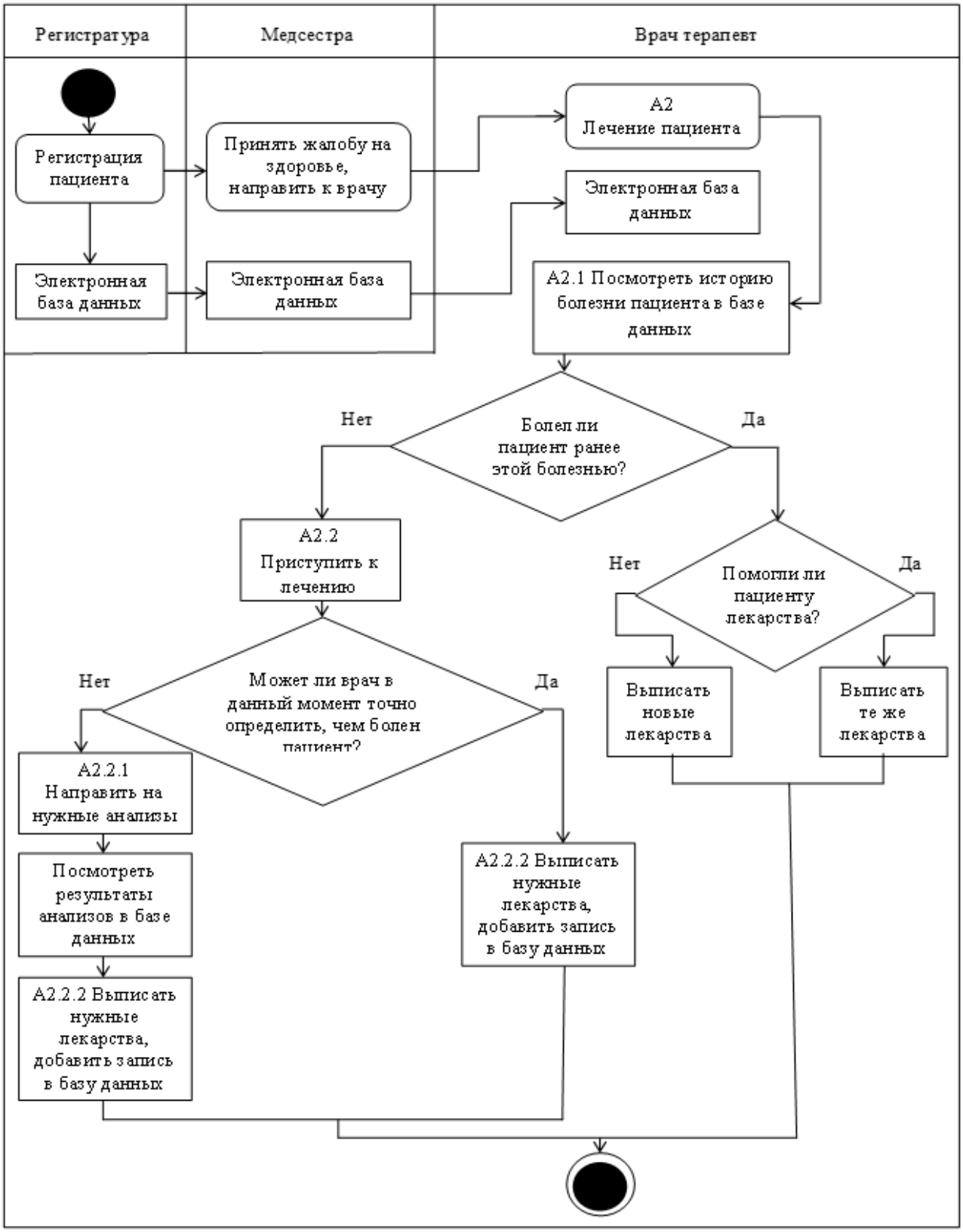


Рисунок 8.4 – *Представление процесса в UML AD на третьем уровне в модели «TO-BE» для уточнения процесса «Приступить к лечению»*

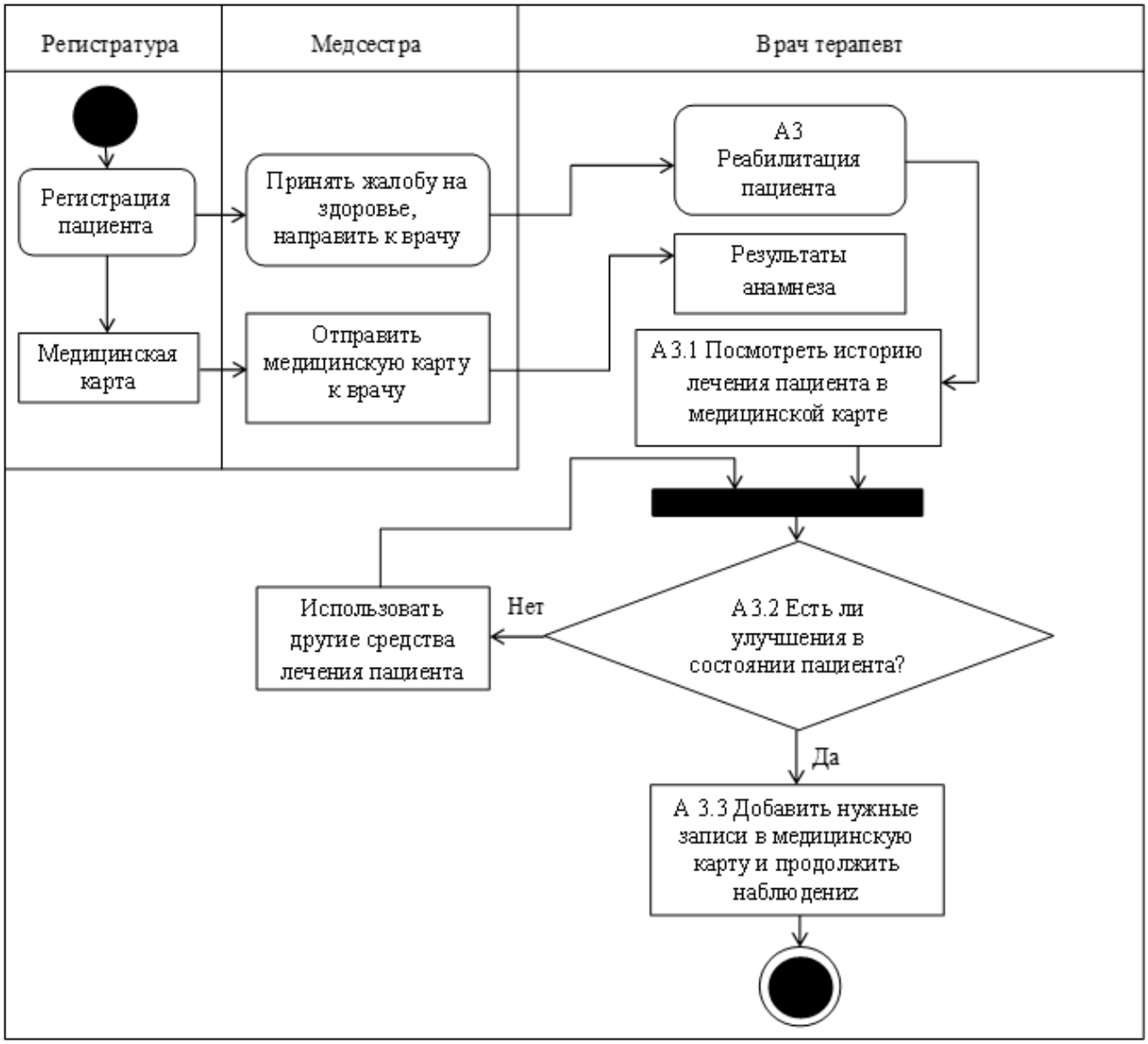


Рисунок 8.5 – *Представление процесса в UML AD на втором уровне в модели «AS-IS» для уточнения процесса «Реабилитация пациента»*

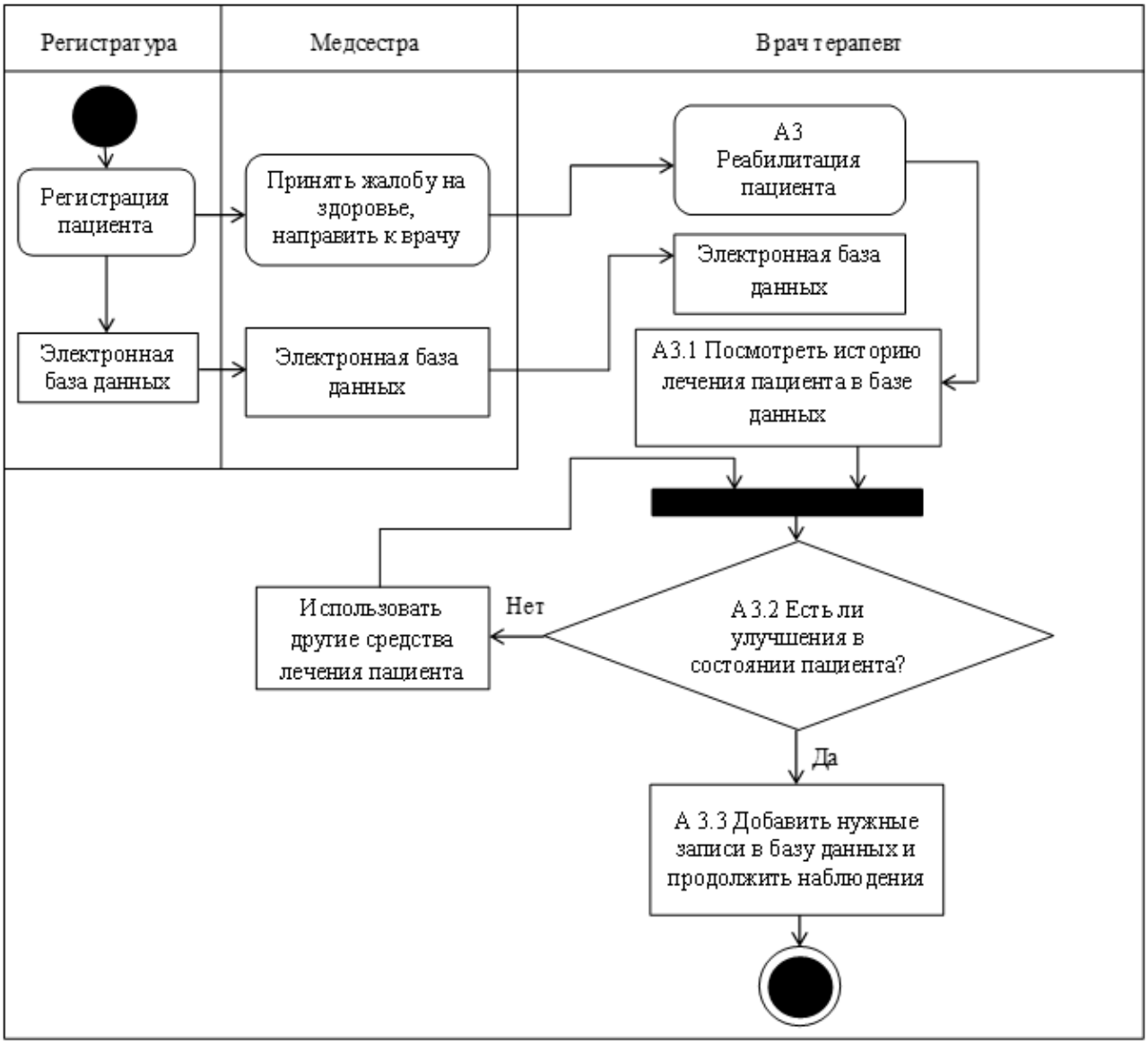


Рисунок 8.6 – *Представление процесса в UML AD на втором уровне в модели «TO-BE» для уточнения процесса «Реабилитация пациента»*

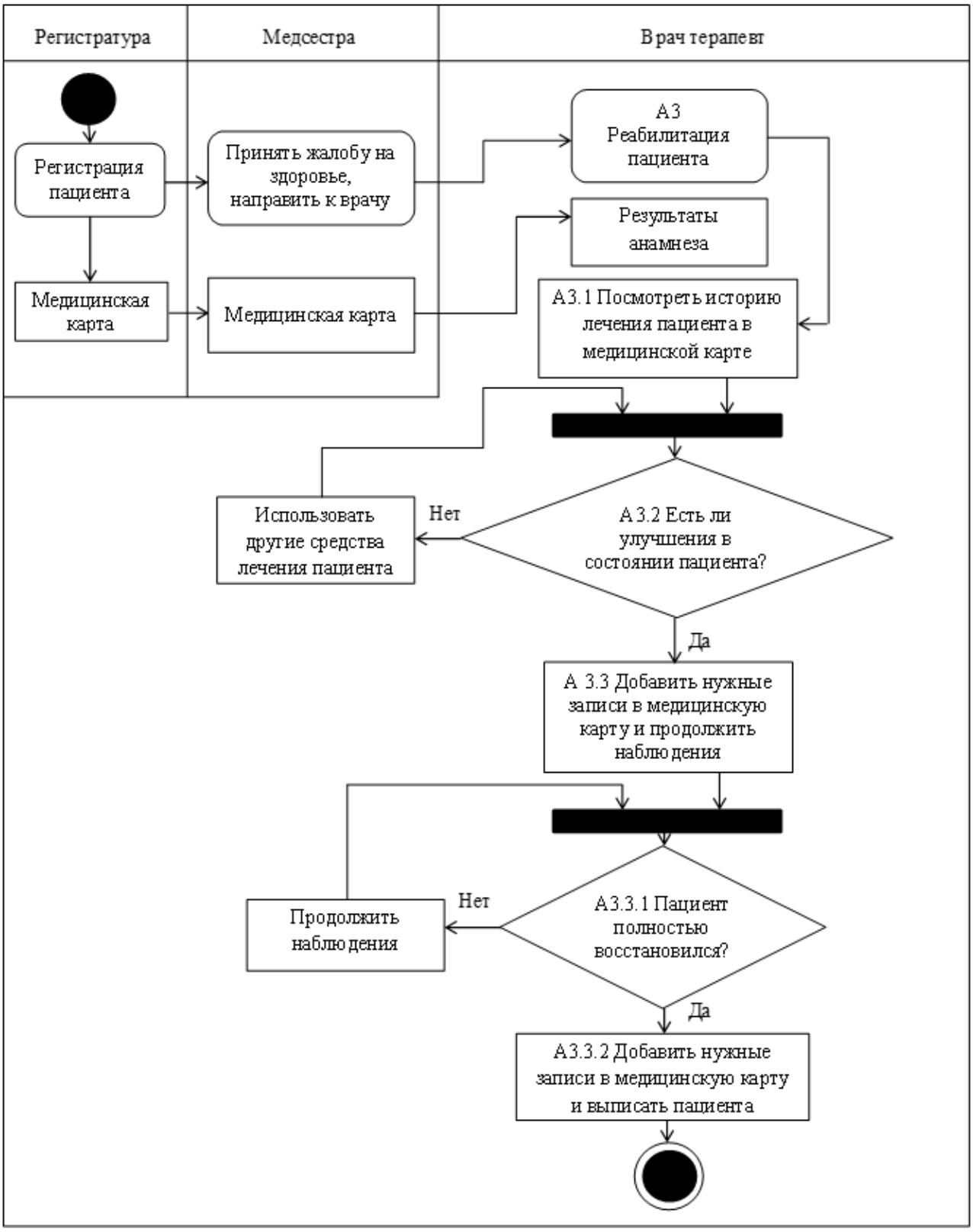


Рисунок 8.7 – *Представление процесса в UML AD на третьем уровне в модели «AS-IS» для уточнения процесса «Продолжить наблюдения»*

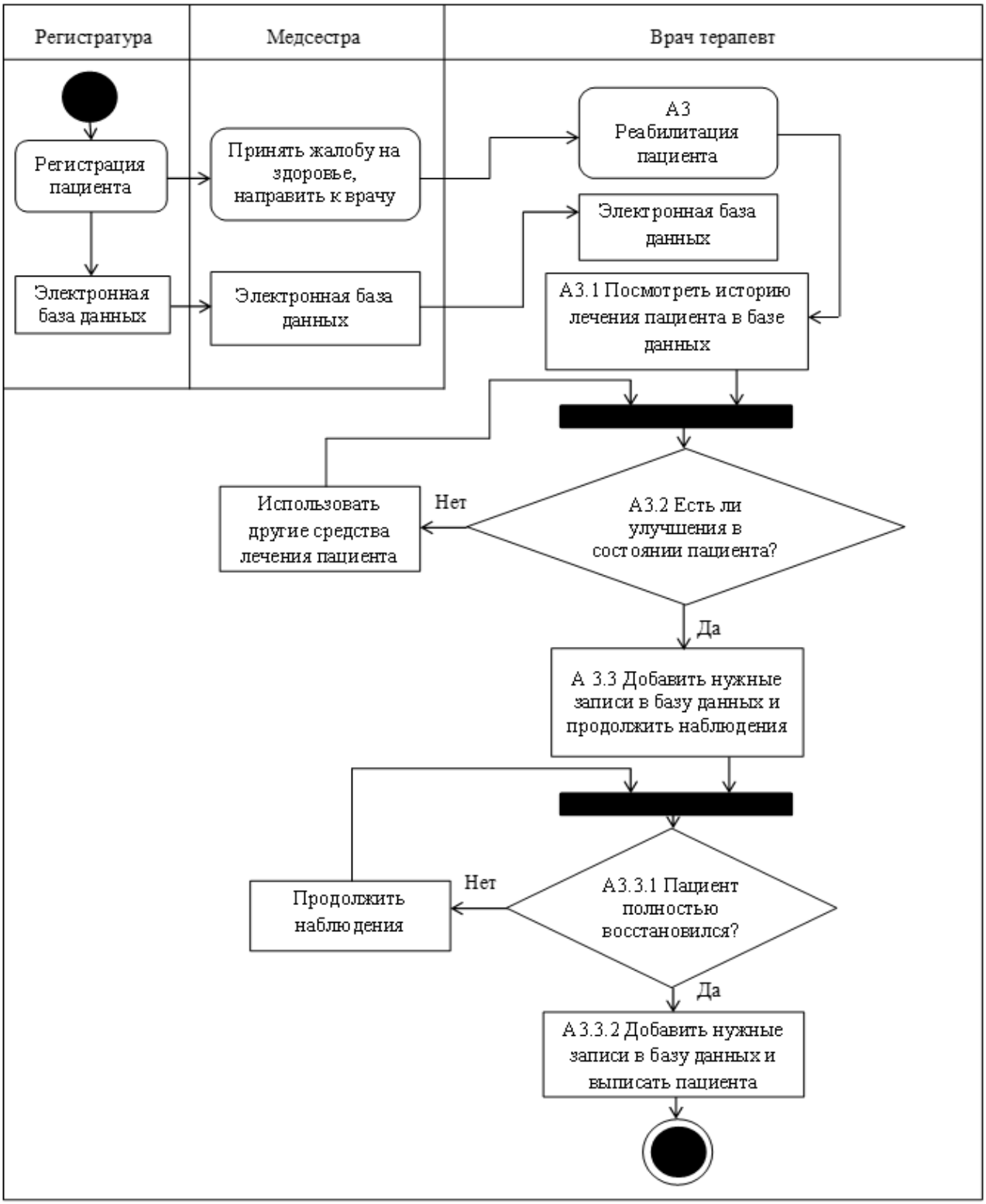


Рисунок 8.8 – *Представление процесса в UML AD на третьем уровне в модели «TO-BE» для уточнения процесса «Продолжить наблюдения»*