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:

«ПРИМЕНЕНИЕ МЕТОДОЛОГИИ AGILE KANBAN ДЛЯ АВТОМАТИЗАЦИИ КЛЮЧЕВЫХ БИЗНЕС-ПРОЦЕССОВ ГОРОДСКОЙ БОЛЬНИЦЫ»

Студент:
Мартынов М.В.

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

Москва – 2019

АННОТАЦИЯ

В данной бакалаврской работе реализуется программное обеспечение в среде MS Access для автоматизации ключевых бизнес процессов городской больницы. В работе содержится 8 глав, введение, заключение, а так же список используемых литературных источников.

Во «Введении» описываются существующие в настоящее время затруднения в деятельности городской больницы, доказывается актуальность представленной темы, ставятся цель и задачи.

В главе «Детальный анализ методологии внедрения Agile Kanban» рассматриваются основные принципы данной методологии гибкой разработки ПО.

В «Идентификация, формирование и приоритизация требований» идентифицируются пользовательские и функциональные требования, формируется бэклог продукта.

В главе «Проектирование ключевых бизнес процессов в моделях As-Is и To-Be» происходит графическое представление ключевых бизнес процессов на трех уровнях детализации в нотации UML AD.

В «Моделирование пользовательского интерфейса» описывается создание интерфейса ПО и схемы приложения.

В главе «Проектирование структуры данных» происходит построение архитектуры данных согласно 3НФ и схемы данных.

В «Разработка программного обеспечения» описываются этапы по разработке ПО согласно методологии Agile Kanban.

В главе «Тестирование программного обеспечения» выполняется функциональное, интеграционное и нагрузочное тестирование ПО. В главе «Экономическое обоснование проекта» составляется смета затрат на разработку ПО. В «Заключении» подводятся итоги проделанной работы. В

«Списке используемых литературных источников» указываются использованные в работе источники.

Объем выпускной квалификационной работы составляет 85 страниц пояснительной записки. В работе содержится 55 рисунков, 8 таблиц, 12 формул, 18 литературных источников.

Оглавление

Введение

Условия развития современного мира, сопровождающиеся ростом потока информации и количества предоставляемых услуг, оказывают существенное влияние на внедрение информационных технологий во все сферы жизнедеятельности человека.

Важнейшей областью в современном обществе является сфера здравоохранения. Увеличение динамики заболеваний среди населения, сопровождающееся ростом количества пациентов, способно затруднить деятельность медицинских учреждений их сотрудников.

В связи с чем, возникает необходимость создания медицинской информационной системы, способной повысить качество медицинского обслуживания и минимизировать риск возникновения врачебной ошибки, путем систематизации данных и автоматизации процессов, снижая избыточную нагрузку с медицинского персонала.

Наличие вышеприведенных проблем в медицинских учреждениях является доказательством актуальности представленной в данной работе темы. Целью данной работы является автоматизация на основе методологии Agile Kanban следующих ключевых бизнес процессов: назначить диагностику. расшифровать результаты диагностики, составить курс терапии.

Основная идея работы заключается в разработке программного обеспечения, которое будет способно автоматизировать данные ключевые бизнес процессы, в следующем виде:

- Автоматически выбрать необходимую диагностику, сформированную на основе жалоб пациента.
- Автоматически расшифровать результаты диагностики и вывести на экран сообщения «Норма» или «Отклонение».
- Сформировать курс терапии с учетом диагноза и возраста пациента.

Для достижения вышеуказанной цели, необходимо реализовать ряд следующих задач:

- Детально проанализировать методологию внедрения Agile Kanban.
- Идентифицировать, проанализировать, приоритизовать требования, составить список задач.
- Спроектировать процессы и оргструктуры в моделях AS-IS и TO-BE нотации UML AD до 3-4 уровней детализации.
- Смоделировать разрабатываемые пользовательские интерфейсы.
- Спроектировать структуру данных и нормализовать таблиц данных.
- Реализовать операции ключевых бизнес процессов в среде MS Access.
- Провести тестирование и количественную оценку результатов тестирования.
- Подготовить блок-схему алгоритма работы программы.

Раздел 1. Детальный анализ методологии внедрения Agile Kanban

1.1. Гибкая методология разработки Agile

Методология разработки Agile – это комплекс «гибких» подходов к разработке программного обеспечения. Главные принципы методологии были отображены в Манифесте Agile (Agile Manifesto), принятом в феврале 2001 года. Данный документ провозглашает двенадцать наиболее значимых принципов создания программного обеспечения (ПО) [1]. Четыре из них являются основополагающими и звучат следующим образом:

- Люди и коммуникация превыше процессов и инструментов.
- Рабочий продукт превыше полной документации.
- Взаимодействие с клиентом превыше согласования условий договора.
- Готовность адаптироваться к переменам превыше следования первичному плану.

На рисунке 1.1 представлена схема, отображающая главный принцип гибких методов разработки программного обеспечения.

Рисунок 1.1 – *Схема работы по Agile*

Специфика методологии Agile состоит в итеративности применяемых в процессе создания ПО подходов. Исходя из сказанного, планирование на

высшем уровне осуществляется в отношении всего цикла реализации проекта, после чего весь цикл разработки делится на итерации – последовательность этапов, в процессе которых планомерно разрабатывается и тестируется программное обеспечение. Итогом завершения итерации является инкремент, посредством которого расширяются возможности создаваемого программного обеспечения.

Основное преимущество данной методологии состоит в максимальном снижении рисков, поскольку в финальной стадии каждой итерации процесс создания программного обеспечения может контролироваться заказчиком, который в связи с этим может одобрить полученные результаты, внести новые запросы или изменить существующие.

1.2. Подход Kanban

Одним из фреймворков методологии Agile является подход Kanban. Данный подход основан на главных наиболее значимых началах Agile. Kanban применяет итеративный подход к созданию программного обеспечения и оперативную адаптацию к изменениям в ходе создания продукта. Специфика подхода заключается в том, что ее можно описать выражением «начните с того, что вы делаете сейчас» [2].

Kanban – это метод для установки, регулирования и оптимизации служб, которые предоставляют продукты интеллектуальной работы, такие как экспертные и творческие услуги, а также создание физической продукции и программного обеспечения [2].

Таким образом, Kanban является методологией, дающей возможность осуществлять преобразования, как в действующем рабочем процессе создания программного обеспечения, так и начинать новый цикл разработки программного обеспечения при помощи данного подхода [3]. Исходя из сказанного, были выведены основные принципы подхода Kanban [4]:

- Kanban основывается на действующих подходах к созданию продукта и по ходу осуществления оптимизирует, дополняет и совершенствует их.
- Осуществление значимых преобразований оговаривается заблаговременно: непрерывные преобразования являются путем к оптимизации действующего процесса создания программного обеспечения, при этом кардинальные трансформации кроют в себе немалые риски.
- Почтительное отношение к установленным правилам, ролям и выполняемым обязанностям.
- Стимулирование инициативы.

1.3. Внедрение подхода

Для того, что бы внедрить подход Kanban в процесс разработки программного обеспечения, необходимо соблюдение особых правил, называемых практиками Kanban. Практики помогают оптимизировать и усовершенствовать процесс разработки программного обеспечения [3]. Основными являются следующие практики:

- Визуализация процесса разработки.
- Ограничение количества одновременно выполняемых задач.
- Измерение и регулирование потока.

1.3.1. Визуализация рабочего процесса

В Kanban применяется процедура визуального мониторинга осуществления стадий процесса разработки [5]. Главным средством визуализации служит Канбан-доска, необходимая для того, чтобы продемонстрировать команде, на каком этапе находится реализация каждой промежуточной цели, из которых затем складывается весь проект в целом. Доска расчерчена и поделена на столбцы, которые отображают этапы процесса

разработки программного обеспечения. Карточки, содержащие задачи, передвигаются по этим столбцам по ходу реализации проекта.

На рисунке 1.2 приведен пример Kanban доски, применяемой в процессе разработки программного обеспечения.

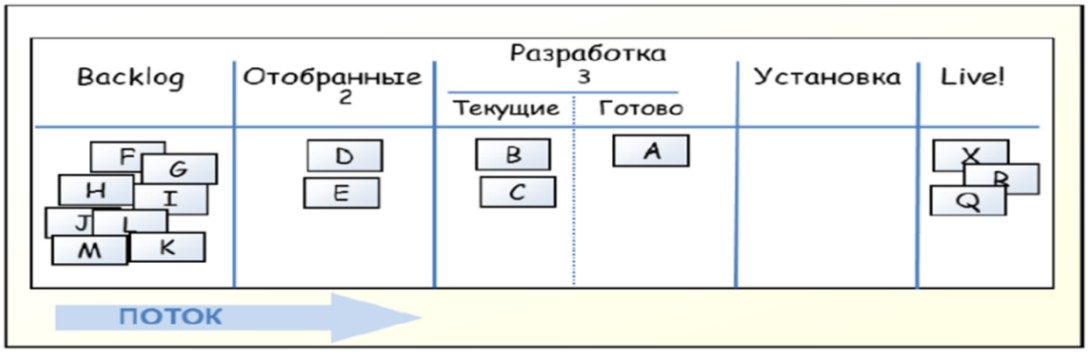


Рисунок 1.2 – *Пример Kanban доски*

1.3.2. Ограничение количества одновременно выполняемых задач

Для того чтобы использование подхода приносило ожидаемый результат, важно производить постоянный мониторинг, сокращение и оптимизацию незавершенной работы [2].

Соблюдение ограничения, позволяет сделать процесс разработки вытягивающим, в котором работа над новыми элементами начинается только по завершении предыдущих задач, что соответствует принципу «Точно в срок».

Следование данному принципу способствует снижению временных затрат на создание и увеличение качества продукта, поскольку большой объем выполненных не в полной мере задач снижает продуктивность и повышает необходимое время на разработку программного обеспечения, препятствуя своевременной реакции на изменения запросов клиента.

Лимит WIP (Work in progress) заключается в четком определении разрешенного количества заданий на каждой из стадий разработки. Когда команда разработчиков вводит ограничение на выполнение незавершенных работ, она добавляет цифру в колонку на доске, которая указывает на

максимальное число рабочих элементов, разрешенных на данном этапе разработки программного обеспечения [3]. Это способствует освобождению места сбора большого количества рабочих компонентов и усовершенствованию процесса разработки.

Зависимость между ограничением числа задач в работе и временем доставки продукта описана математически в теории очередей и называется закон Литтла. Согласно данному закону, повышение количества выполняемых операций способствует увеличению времени, необходимого для реализации каждой из них. Представим формулу, отражающую данный закон:

$$\text{Cycle Time} = \frac{\text{Work in Progress}}{\text{Throughput}}, \tag{1.1}$$

в данной формуле используются следующие обозначения (Kanban – метрики):

- Cycle Time – время, в течение которого определенное задание было на стадии выполнения, от начала работы над ним до окончания.
- Work in Progress – Количество незавершённой работы.
- Troughput – число, отображающее количество компонентов, которые коллектив разработчиков может производить за установленный временной промежуток.

1.3.3. Измерение и управление потоком

Под измерением и управлением потоком предполагается измерение количества текущих задач и регулирование хода разработки для его максимального увеличения [3]. Главным средством, применяемым для измерения потока, является кумулятивная диаграмма потока – CFD. Она отображает ограничение на реализацию неоконченных задач (WIP-лимит), общее количество рабочих компонентов в процессе разработки, количество компонентов, которые присоединяются к списку каждый день, среднее время, в

течение которого рабочие компоненты остаются на стадии реализации [3]. Приведем пример CFD диаграммы на рисунке 1.3.

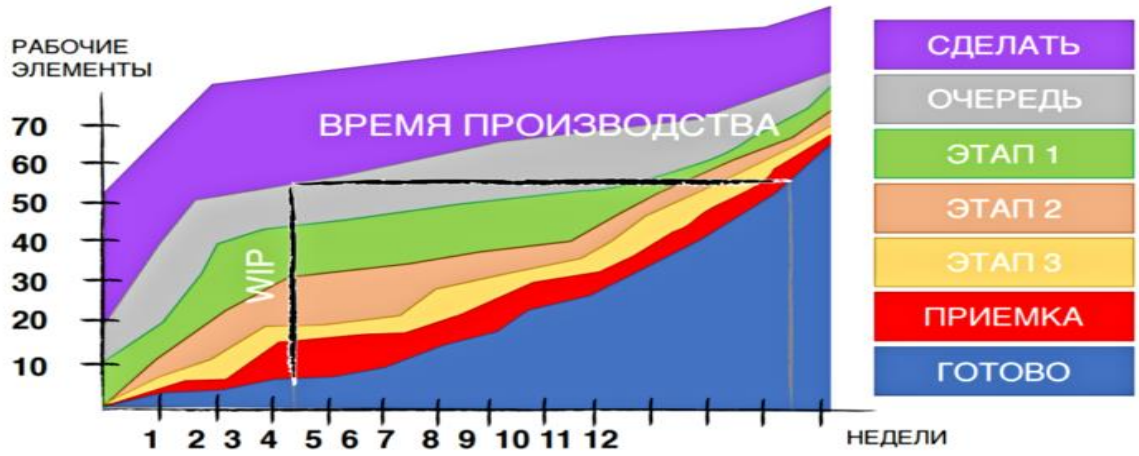


Рисунок 1.3 – *Кумулятивная диаграмма потока CFD*

Данная диаграмма передает мониторинг количества рабочих компонентов на каждой стадии, проводимый изо дня в день. Ось X отображает время, а ось

Y – количество задач. Таким образом, можно наглядно заметить наличие узких мест в процессе разработки, если на графике возрастает какая либо область.

1.4. Достоинства и недостатки подхода

К основным достоинствам подхода Agile Kanban можно отнести следующее:

- Гибкость и быстрая реакция на изменения.
- Адаптивность – не существует жестких правил и ограничений.
- Kanban не предполагает радикальных преобразований процесса разработки, все трансформации, которые имеют место, носят постепенный характер.
- Наглядность рабочего процесса и отсутствие узких мест в ходе разработки программного обеспечения.

- К недостаткам данного подхода можно отнести:
- Нет жестко выстроенной структуры рабочего процесса и механизмов реализации отдельных итераций — разделение общего проекта на разные блоки и нередкие преобразования могут повлечь утрату приоритетов и направления, в котором необходимо двигаться.
- Подход не предназначен для долгосрочного планирования.
- Kanban затруднительно использовать в рабочих коллективах с большим количеством участников (более пяти).

1.5. Реализация программного обеспечения на основе подхода Agile Kanban

Для реализации программного обеспечения, с помощью которого будут автоматизированы ключевые бизнес процессы больницы, необходимо определить последовательность шагов, согласно принципам методологии Agile Kanban. Таким образом, были определены следующие шаги:

- Идентификация и формирование требований.
- Формирование бэклога.
- Разделение процесса разработки на итерации, формирование бэклога итерации.
- Создание Kanban доски, определение этапов разработки программного обеспечения.
- Установление WIP лимита.
- Подготовка программного обеспечения к релизу.
- Релиз.

Раздел 2. Идентификация, формирование и приоритизация требований

Требования к программному обеспечению – комплекс выдвигаемых запросов по поводу признаков, свойств и качеств программного обеспечения, которые необходимо реализовать [6]. Требования к разрабатываемому программному обеспечению должны обладать следующими свойствами [7]:

- Полнота – каждое требование должно полностью описывать функциональность, которую следует реализовать в продукте.
- Корректность – каждое требование должно точно описывать желаемую функциональность.
- Осуществимость – возможность реализовать каждое требование при известных условиях.
- Необходимость – каждое предъявляемое требование должно демонстрировать функциональность, необходимую будущим пользователям или требуемую для соблюдения определенных системных норм и стандартов;
- Однозначность – все читатели требований должны интерпретировать их одинаково.

2.1. Анализ требований

Анализ требований – процесс классификации информации, касающейся требований, по различным категориям, оценка требований для определения желаемого качества, представление требований в различных формах, выделение детальных требований из требований более высокого уровня, обсуждение приоритетов требований [7]. Анализ требований включает в себя следующие этапы:

- Установка основной идеи продукта. На этой стадии осуществляется формирование общего понимания, как должен выглядеть

разрабатываемый продукт. В финале данной стадии принимается решение о необходимости разработки данного продукта;

- Сбор требований. На данной стадии большая часть работы посвящается коммуникации с клиентом и потенциальными пользователями разрабатываемого продукта. Цель на данной стадии состоит в четкой установке возможностей продукта и механизмов его внедрения в действующие процессы.
- Анализ требований. На данном этапе проходит структуризация собранных ранее требований. Цель этапа – предоставить четкий список не дублируемых требований к системе, которые должны быть выделены из избыточных и частично дублирующихся сценариев и пользовательских историй [8].
- Проектирований системы. Эта стадия предполагает принятие необходимых решений относительно того, какие возможности будет иметь будущий продукт, с целью максимального соответствия запросам пользователей.

2.2. Формирование требований

2.2.1. Пользовательские требования

Пользовательские требования – это требования, выдвигаемые потенциальными пользователями к создаваемому программному обеспечению, озвученные в простой естественной форме. Для ключевых бизнес процессов: «Назначить диагностику», «Расшифровать результаты диагностики», «Составить курс терапии» были определены следующие пользовательские требования:

- Хранение данных:
  - Данные о пациентах.
  - Данные о специалистах.

- Виды диагностики.
- Список диагнозов гастроэнтерологической направленности.
- Список характерных симптомов гастроэнтерологической направленности.
- Диапазон нормальных значений для каждого диагностического показателя.
- Результаты диагностики.
- Список лекарственных препаратов для диагнозов гастроэнтерологической направленности.
- Данные о дозировках лекарственных препаратов.
- Данные о первичном приеме пациента.
- Данные о назначенном курсе терапии.

- Ввод и редактирование данных:
  - Регистрация нового пациента.
  - Редактирование данных о пациенте.
  - Редактирование данных о специалисте.
  - Поиск зарегистрированного пациента.
  - Ввод данных о первичном приеме пациента.
  - Запись результатов диагностики.
  - Ввод данных о назначенном курсе терапии.
  - Редактирование данных о лекарственных препаратах.
- Вывод информации на экран.
- Автоматическое назначение диагностики на основе жалоб пациента.
- Автоматическая расшифровка результатов диагностики:
  - Вывод на экран сообщений «Норма», «Отклонение» для каждого диагностического показателя.
- Автоматическое назначение курса терапии.

- Назначение лекарственных препаратов для определенного диагноза;
- Подбор дозировки с учетом возраста пациента.

- Обеспечение конфиденциальности.
- Печать данных.

2.2.2. Функциональные требования

Функциональные требования – положение о фрагменте требуемой функциональности или поведения, которые система проявляет при определенных условиях [7]. Иными словами, функциональные требования устанавливают возможности создаваемого программного обеспечения – необходимые функции и операции, которые можно будет с помощью него производить. Таким образом, для разрабатываемого программного обеспечения были определены следующие функциональные требования:

- Таблицы.
  - Пациенты.
  - Специалисты.
  - Диагнозы.
  - Первичный прием.
  - Жалобы.
  - Виды диагностики.
  - Проведенная диагностика.
  - Показатели нормы.
  - Лекарства.
  - Диагнозы и лекарства.
  - Пациенты Жалобы.
  - Направления по жалобам.
  - Назначения.
- Редактирование содержимого таблиц, формы для ввода данных.

- Отчеты для отображения данных.
  - Параметрический запрос для автоматического назначения диагностики.
  - Параметрический запрос для автоматического сравнения полученных результатов диагностики с нормальными значениями.
- Запись диапазона нормальных значений для каждого диагностического показателя.
- Вывод на экран сообщений «Норма», «Отклонение» в результате процесса сравнения.
- Параметрический запрос для определения курса терапии:
  - Автоматическое вычисление возраста пациента.
- Пароль при входе в главное меню.
- Передача данных на устройство печати.

2.3. Формирование бэклога продукта

В Agile терминологии под бэклогом подразумевается перечень требований к создаваемому программному обеспечению. При формировании бэклога все требования необходимо выстроить в порядке их приоритетности. Для этого необходимо выполнить процесс приоритизации требований. Осуществление данного процесса позволяет понять разработчику, какие пользовательские требования необходимо реализовать первоочередно, а так же на что следует обратить особое внимание, тем самым позволяя избежать излишних материальных и временных затрат на проектирование и разработку.

Выполним приоритизацию требований с помощью модели Кано (Kano Model) [9]. Данный метод приоритизации позволяет характеризовать каждую задачу, выстроить очередность реализации требований и сформировать план поставок основываясь на ожиданиях пользователей, классифицируя их по определенным категориям. Каждая категория отображает степень значимости

для клиента каждого из требований. Приведем описание модели Кано в таблице 2.1.

Таблица 2.1 – *Описание модели Кано*

Приоритет	Описание приоритета
Must-be Quality	Обязательные для пользователя свойства ПО Данные требования наиболее важны для клиента, представляют собой неотъемлемую часть разрабатываемого продукта, без их выполнения его релиз невозможен
One-dimensional Quality	Одномерные свойства От воплощения в реальность данной возможности продукта зависит, будет ли удовлетворен клиент работой разработчиков
Attractive Quality	Привлекательные для пользователя свойства Клиент одобряет реализацию данных требований, однако, их несоблюдение не приводят к его неудовлетворенности
Indifferent Quality	Безразличные для пользователя характеристики Пользователей в целом не волнует степень реализации данных требований, от них никак не зависит степень удовлетворенности продуктом
Reverse Quality	В данную группу включаются требования, выполнение которых улучшает продукт, но при этом ведет к его чрезмерному усложнению. Как правило, сюда относятся различные дополнительные функции, благодаря которым продукт получает более широкие возможности

Таким образом, сформируем бэклог продукта для разрабатываемого программного обеспечения с учетом приоритизации по модели Кано. Представим результат в таблице 2.2.

Таблица 2.2 – *Бэклог продукта*

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет
1	Хранение данных: ▪ Данные о пациентах ▪ Данные о специалистах ▪ Виды диагностики ▪ Список диагнозов гастроэнтерологической направленности ▪ Список характерных симптомов гастроэнтерологической направленности ▪ Диапазон нормальных значений для каждого диагностического показателя ▪ Результаты диагностики ▪ Список лекарственных препаратов для диагнозов гастроэнтерологической направленности ▪ Данные о дозировках лекарственных препаратов ▪ Данные о первичном приеме пациента ▪ Данные о назначенном курсе терапии	Таблицы: ▪ Пациенты ▪ Специалисты ▪ Диагнозы ▪ Первичный прием ▪ Жалобы ▪ Виды диагностики ▪ Проведенная диагностика ▪ Показатели нормы ▪ Лекарства ▪ Диагнозы и лекарства ▪ Пациенты Жалобы ▪ Направление по жалобам	Программа ввода данных	Must-be Quality
2	Ввод и редактирование данных: ▪ Регистрация нового пациента ▪ Редактирование данных о пациенте ▪ Редактирование данных о специалисте ▪ Поиск зарегистрированного пациента ▪ Ввод данных о первичном приеме пациента ▪ Запись результатов диагностики ▪ Ввод данных о назначенном курсе терапии ▪ Ввод данных о лекарственных средствах	Формы для ввода данных	Программа для ввода и редактировани я данных	Must-be Quality
3	Вывод информации на экран	Отчеты для отображения данных.	Программа для работы с интерфейсом	Must-be Quality

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет
4	Автоматическое назначение диагностики на основе жалоб пациента	Параметрический запрос	Программа для создания запросов	Must-be Quality
5	Автоматическая расшифровка результатов диагностики Вывод на экран сообщений «Норма» или «Отклонение» для каждого диагностического показателя	Параметрический запрос для автоматического сравнения полученных результатов диагностики с нормальными значениями ▪ Запись диапазона нормальных значений для каждого диагностического показателя ▪ Вывод на экран сообщений «Норма», «Отклонение» в результате процесса сравнения	Программа для создания запросов	Must-be Quality
6	Автоматическое назначение курса терапии ▪ Назначение лекарственных препаратов для определенного диагноза ▪ Подбор дозировки с учетом возраста пациента	Параметрический запрос для определения курса терапии Автоматическое вычисление возраста пациента	Программа для создания запросов	Must-be Quality
7	Обеспечение конфиденциальности	Пароль при входе в главное меню	Программа для работы с БД	One-dimensional Quality
8	Печать данных	Передача данных на устройство печати	Программа для работы с БД	One-dimensional Quality

2.4. Разделение процесса разработки на итерации

Согласно методологии Agile Kanban, жизненный цикл разработки программного обеспечения необходимо разбить на итерации. Отобразим бэклог итераций и их количество в таблице 2.3.

Таблица 2.3 – *Бэклог итераций*

Итерация	Бэклог итерации
Итерация 1	▪ Моделирование ключевых бизнес процессов в моделях As-Is и To-Be в нотации UML AD ▪ Моделирование пользовательского интерфейса

Итерация	Бэклог итерации
	▪ Моделирование структуры данных разрабатываемого приложения
Итерация 2	▪ Реализация требований 1-3 ▪ Тестирование и отладка на промежуточном этапе ▪ Демонстрация инкремента
Итерация 3	▪ Реализация требований 4-6 ▪ Тестирование и отладка на промежуточном этапе ▪ Демонстрация инкремента
Итерация 4	▪ Реализация требований 7-8 ▪ Тестирование и отладка на промежуточном этапе ▪ Демонстрация инкремента
Тестирование	▪ Функциональное тестирование ▪ Интеграционное тестирование ▪ Нагрузочное тестирование

Раздел 3. Проектирование ключевых бизнес-процессов в моделях AS-IS и TO-BE (итерация 1)

Проектирование – процесс разработки планов и чертежей, технических спецификаций и операционных характеристик, необходимых для создания концепций разработки производства и маркетинга новых изделий и процессов [10]. В ходе проектирования бизнес-процессов происходит их декомпозиция. Декомпозиция – это метод, который дает возможность разложить сложную многоуровневую структуру на ряд простых взаимозависимых элементов [11]. Глубина осуществляемой декомпозиции зависит от целей, которые преследуются в процессе проектирования и, следовательно, определяет уровень подробности характеристики процесса.

3.1. Описание моделей проектирования

Основной целью проектирования бизнес процессов является описание реального хода бизнес – процессов предприятия. В процессе проектирования важно установить, что представляет собой итог реализации процесса, какие операции и кем производятся, их последовательность, какой осуществляется документооборот в процессе реализации, а также степень надежности рассматриваемого процесса и его потенциал для оптимизации. Таким образом, существуют следующие стадии проектирования бизнес процессов:

- Идентификация процессов и построение модели As-Is. Модель As-Is (как есть) – Представляет собой модель фактического состояния. Она дает возможность провести систематизацию происходящих в данное время процессов и применяемых информационных объектов.
- Анализ и уточнение исходной модели. На данном этапе выявляются противоречия и дублирование
- действий в процессе, устанавливаются взаимосвязи между процессами, определяется необходимость изменения процесса.

- Разработка модели To-Be. Данная модель, формируемая на базе итогов исследования модели As-Is, характеризует будущие свойства процессов, принимая во внимание требования клиента, а также результаты исследований и усовершенствования действующих процессов [12].
- Испытание и применение модели To-Be. На этом этапе проектирования происходит введение сформированной модели в эксплуатацию в процессе деятельности предприятия. По ходу использования модель тестируется в реальных условиях и, если того требует ситуация, ее подвергают правкам и дополнениям.

3.2. Нотация UML Activity Diagram

Проектирование бизнес-процессов состоит в наглядном отображении этих процессов посредством определенной нотации. Нотация – упорядоченная совокупность условных символов и знаков, используемая в определенном языке для наглядной демонстрации [13].

Унифицированный язык моделирования (Unified Modeling Language) UML – это графический язык для визуализации, специфицирования, конструирования и документирования систем, в которых главная роль принадлежит программному обеспечению [14].

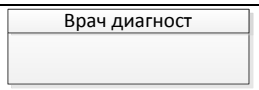
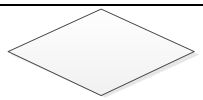
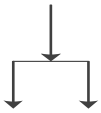
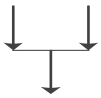
Одним из видов UML диаграмм является Activity Diagram – Диаграмма деятельности. При помощи Activity Diagram можно смоделировать последовательности действий, которые выполняются различными элементами, входящими в состав системы. Основу Activity Diagram составляют:

- Действие (action) – элементарная составляющая деятельности.
- Деятельность (activity) – состоит из последовательности выполнения действий.

Таким образом, графически диаграмма деятельности представляется в форме графа деятельности, вершинами которого являются состояния действия,

а дугами – переходы от одного состояния действия к другому. Приведем описание основных элементов UML AD в таблице 3.1.

Таблица 3.1 – *Условные обозначения нотации UML AD*

Графический элемент	Описание элемента
	Действие
	Входящий/Исходящий документ
	Ответственный организационный уровень
	Условие
	Разветвитель
	Соединитель
	Начальный узел
	Финальный узел

3.3. Проектирование ключевых бизнес процессов в модели As-Is

3.3.1. Проектирование на первом уровне детализации

Проектирование ключевых бизнес процессов на первом уровне детализации заключается в верхнеуровневом описании данных процессов. Верхнеуровневое описание процессов – описание, в котором система процессов представлена деревом, процессы как минимум идентифицированы и определено их первоначальное взаимодействие [10]. Представим результат проектирования бизнес процессов на первом уровне детализации на рисунке 3.1.

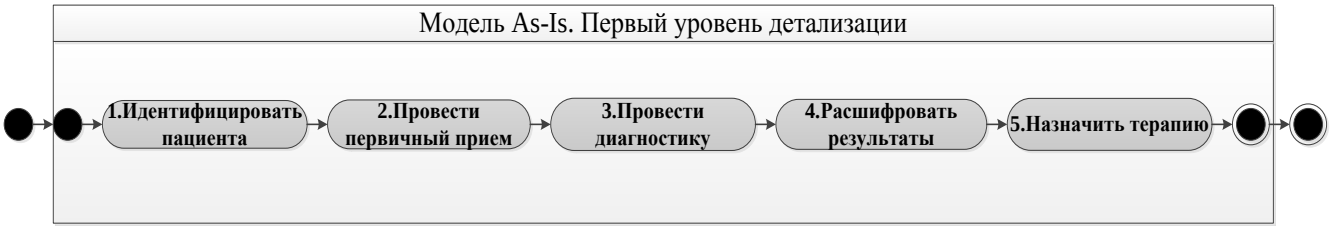


Рисунок 3.1 – *Описание бизнес процессов на первом уровне детализации*

3.3.2. Проектирование на втором уровне детализации

Для того, что бы детально проанализировать процессы, представленные на первом уровне и выявить наличие узких мест в рамках деятельности больницы, необходимо провести декомпозицию данных процессов. Представим на рисунке 3.2 описание процессов «Идентифицировать пациента» и «Провести первичный прием» на втором уровне детализации.

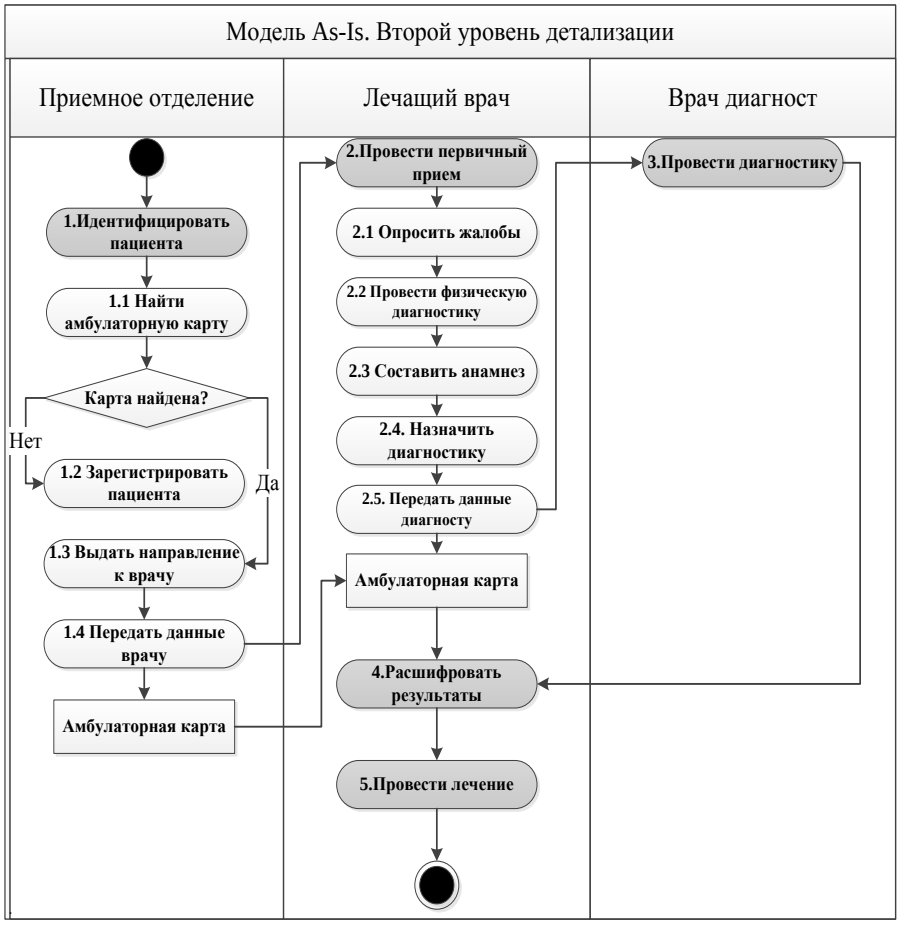


Рисунок 3.2 – *Проектирование процессов «Идентифицировать пациента», «Провести первичный прием» на втором уровне детализации*

Аналогичным способом проведем проектирование бизнес процессов: «Провести диагностику», «Расшифровать результаты», «Провести лечение» на втором уровне детализации. Результат представим в приложении А.

3.3.3. Проектирование на третьем уровне детализации

В результате декомпозиции процесса «Провести первичный прием» на втором уровне детализации была установлена необходимость автоматизации процесса 2.4 – «Назначить диагностику». Таким образом, представим описание процесса «Назначить диагностику» на третьем уровне детализации. Отобразим результат на рисунке 3.3.

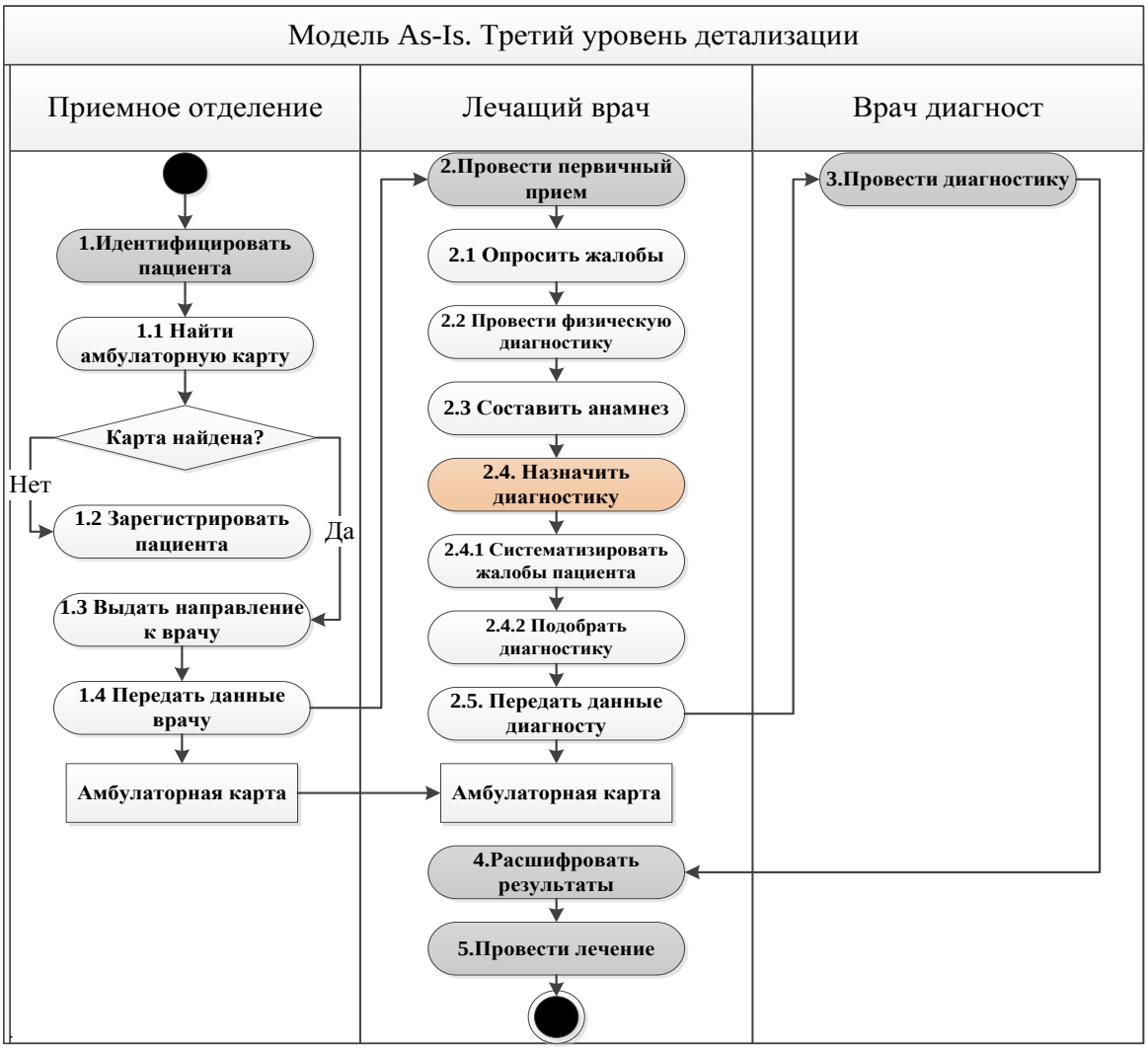


Рисунок 3.3 – *Детализация процесса «Назначить диагностику»*

Аналогичным способом приведем описание процессов: «Провести диагностику», «Расшифровать результаты», «Провести лечение» на третьем уровне детализации. Результат представим в приложении А. Таким образом, в ходе проектирования бизнес процессов на втором уровне детализации была проанализирована текущая деятельность больницы и установлены в ней следующие недостатки:

- Наличие амбулаторной карты пациента в бумажном виде.
- Поиск амбулаторной карты в регистратуре.
- Назначение необходимой диагностики на основе многочисленных жалоб пациента вручную.
- Составление бланка диагностики в бумажном виде.
- Расшифровка результатов диагностики вручную на основе медицинского справочника.
- Самостоятельное назначение курса терапии для установленного диагноза и подбор дозировки пациенту.

3.4. Проектирование ключевых бизнес процессов в модели To-Be

3.4.1. Проектирование на втором уровне детализации

Для отображения изменений, привносимых разрабатываемым программным обеспечением в бизнес процессы, описанные в модели As-Is, необходимо спроектировать данные процессы в модели To-Be. На рисунке 3.4 представим результат проектирования процессов «Идентифицировать пациента» и «Провести первичный прием» на втором уровне детализации.

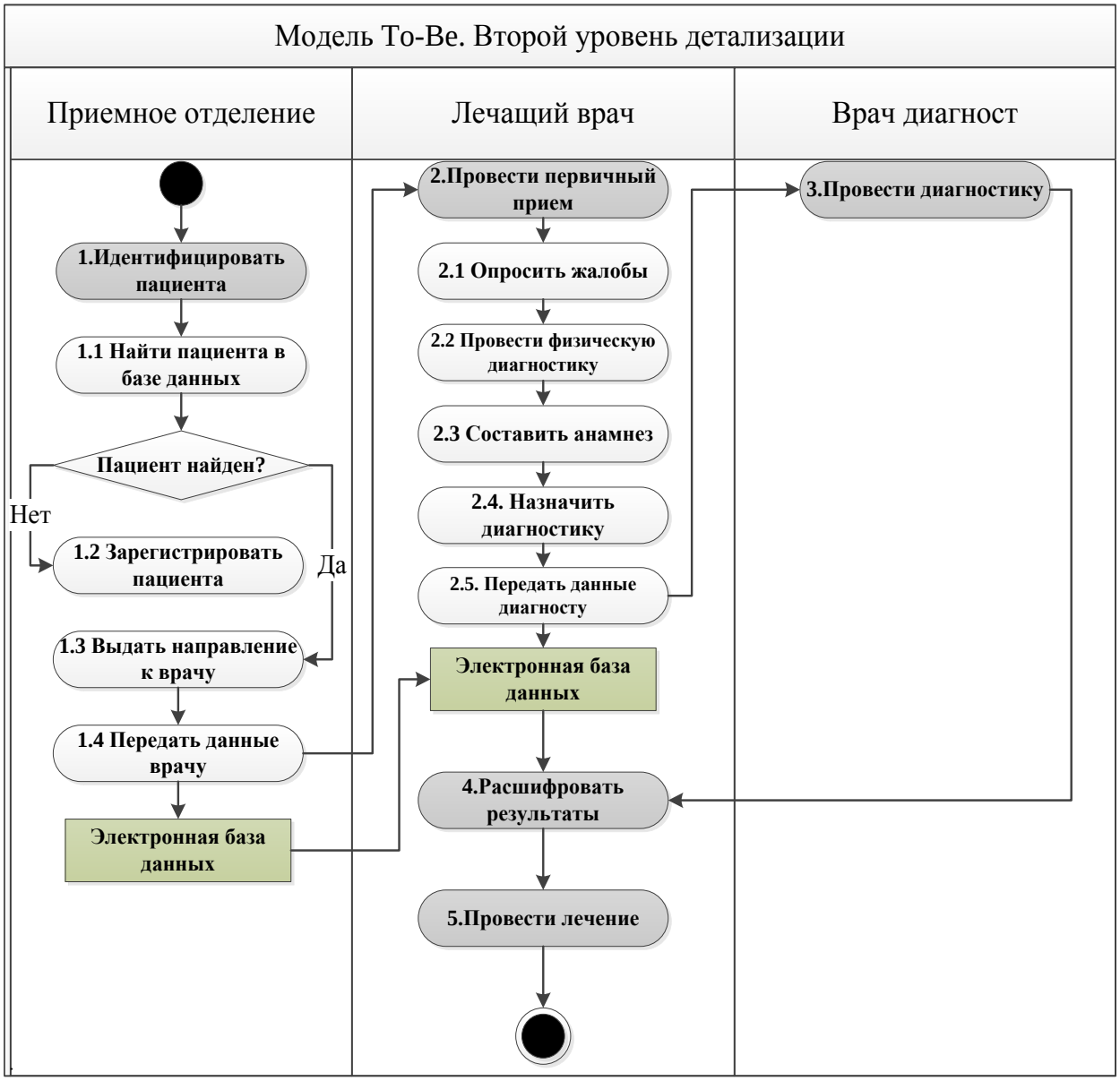


Рисунок 3.4 – *Описание процессов «Идентифицировать пациента», «Провести первичный прием»*

Аналогичным способом приведем описание процессов: «Провести диагностику», «Расшифровать результаты», «Провести лечение» на втором уровне детализации в модели To-Be. Результат представим в приложении Б.

3.4.2. Проектирование на третьем уровне детализации

Для уточнения всех изменений, привносимых в бизнес процесс «Провести первичный осмотр», необходимо произвести проектирование на

третьем уровне детализации, с целью уточнения процесса «Назначить диагностику». Представим результат проектирования на рисунке 3.5.

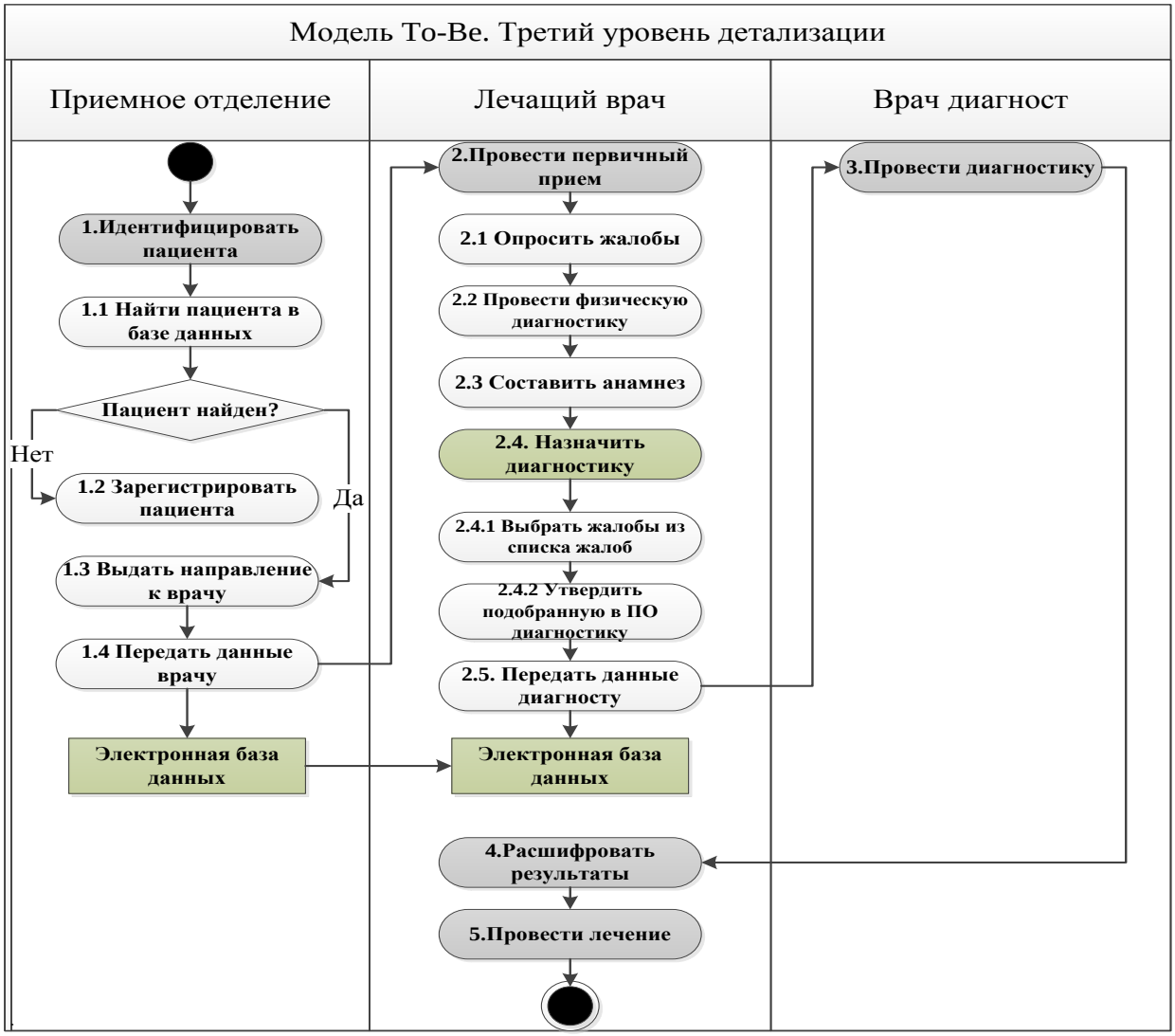


Рисунок 3.5 – *Описание процессов «Идентифицировать пациента», «Провести первичный прием», детализация процесса «Назначить диагностику»*

Аналогичным способом приведем описание процессов: «Расшифровать результаты», «Провести лечение» на третьем уровне детализации в модели To-Be. Результат представим в приложении Б.

3.5. Карта процессов

Картой бизнес процессов называют графическое представление бизнес-процессов в виде блок-схемы. На рисунке 3.6 отобразим карту процессов в модели To-Be.

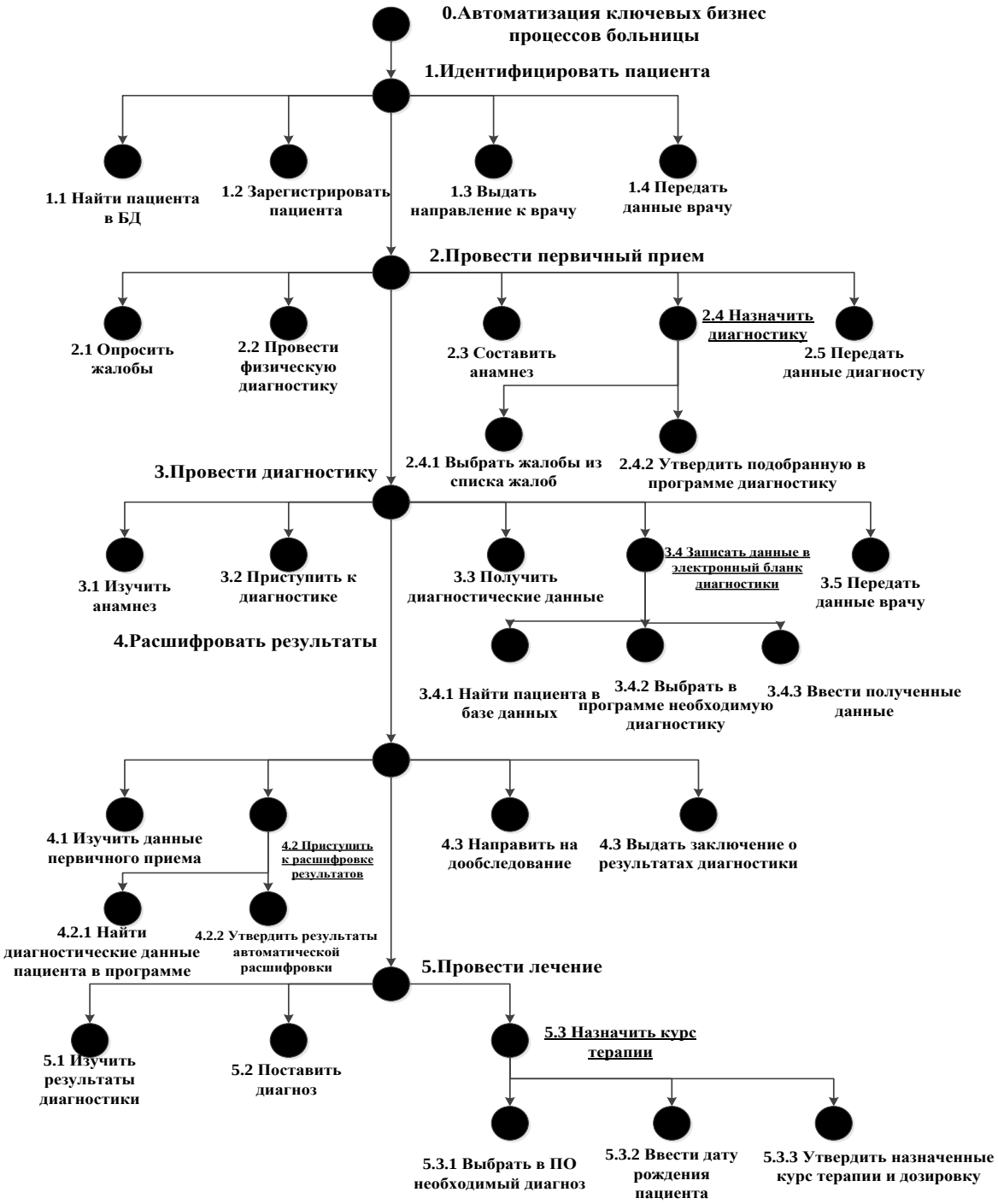


Рисунок 3.6 – *Карта процессов в модели To-Be*

Раздел 4. Моделирование пользовательского интерфейса (итерация 1)

Главная задача этапа моделирования пользовательского интерфейса состоит в формировании наиболее простого для понимания и использования интерфейса, используя который, пользователи не будут сталкиваться с большим количеством препятствий и проблем. В процессе проектирования необходимо принимать во внимание следующие условия: для кого и для чего предназначено разрабатываемое программное обеспечение; как распределяются функции системы по конкретным страницам, и какова их последовательность.

4.1. Интерфейс главного меню

Модель интерфейса главного меню разрабатываемого приложения, представленного на рисунке 4.1, состоит из шести кнопок, функционал которых соответствует требованиям, описанных в разделе 2.

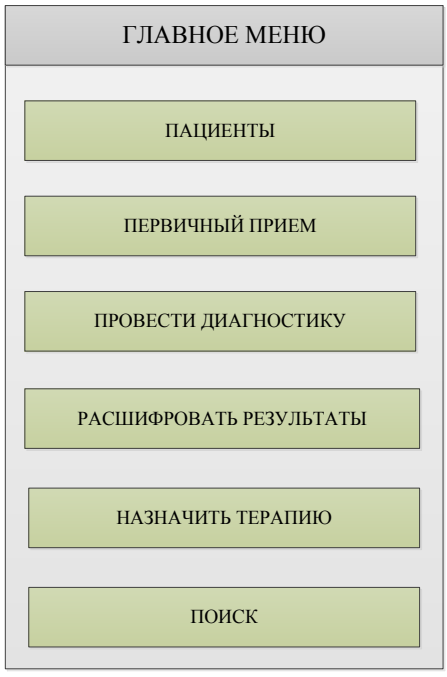


Рисунок 4.1 – *Модель интерфейса главного меню*

4.2. Интерфейсы кнопок главного меню

4.2.1. Кнопка «Пациенты»

При переходе по кнопке «Пациенты» пользователю программного обеспечения открывается страница, позволяющая просматривать информацию о зарегистрированных в больнице пациентах, редактировать и добавлять новые данные, совершать поиск пациента по ФИО. Представим модель кнопки «Пациенты» на рисунке 4.2.

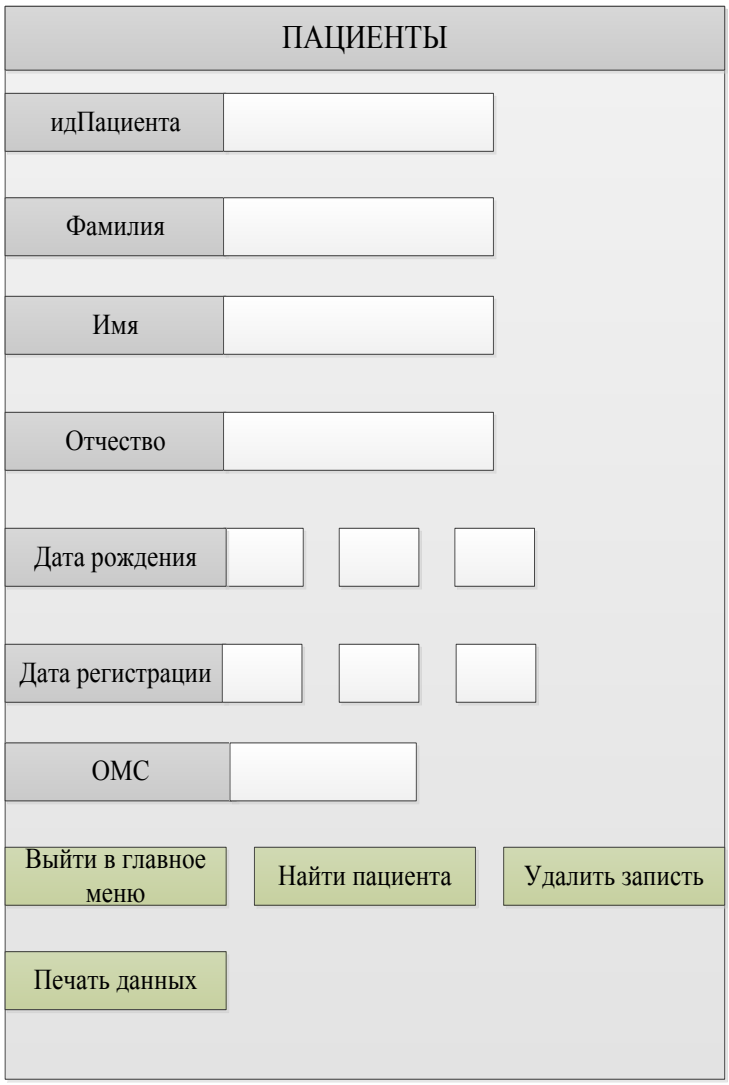


Рисунок 4.2 – *Модель кнопки «Пациенты»*

4.2.2. Кнопка «Первичный прием»

При переходе по кнопке «Первичный прием», представленной на рисунке 4.3, пользователю программного обеспечения открывается страница, содержащая следующую информацию:

- Данные о пациенте.
- Данные о специалисте.
- Поле для ввода результатов анамнеза.
- Возможность выбора жалоб из предложенного списка жалоб.
- Кнопка «Предложить диагностику» для автоматического назначения диагностики на основании указанных жалоб пациента.
- Функции печати, удаления и поиска данных.

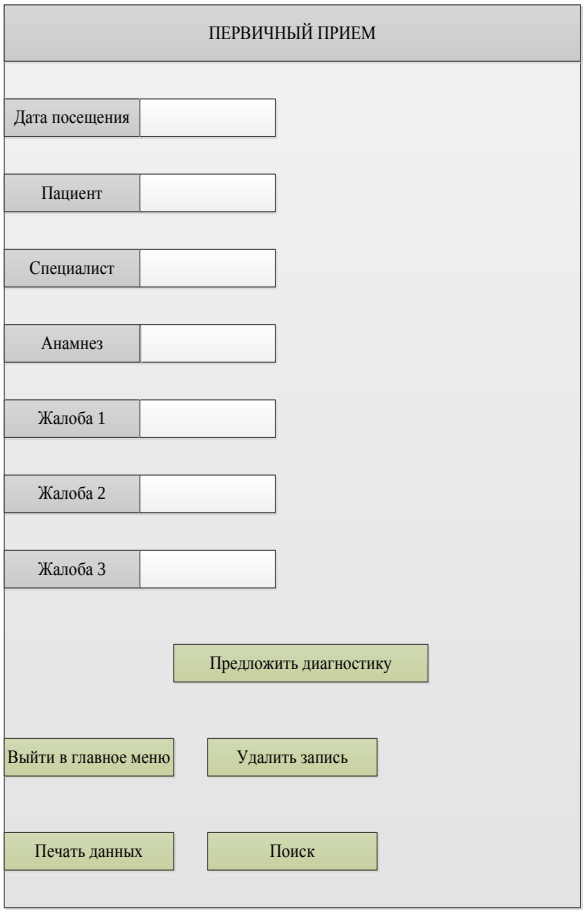


Рисунок 4.3 – *Модель кнопки «Первичный прием»*

4.2.3. Кнопка «Провести диагностику»

При переходе по данной кнопке, пользователю программного обеспечения открывается электронный бланк диагностики, позволяющей выбрать вид диагностики и зафиксировать диагностические показатели, полученные во время проведения обследования. Результат моделирования интерфейса кнопки представим на рисунке 4.4.

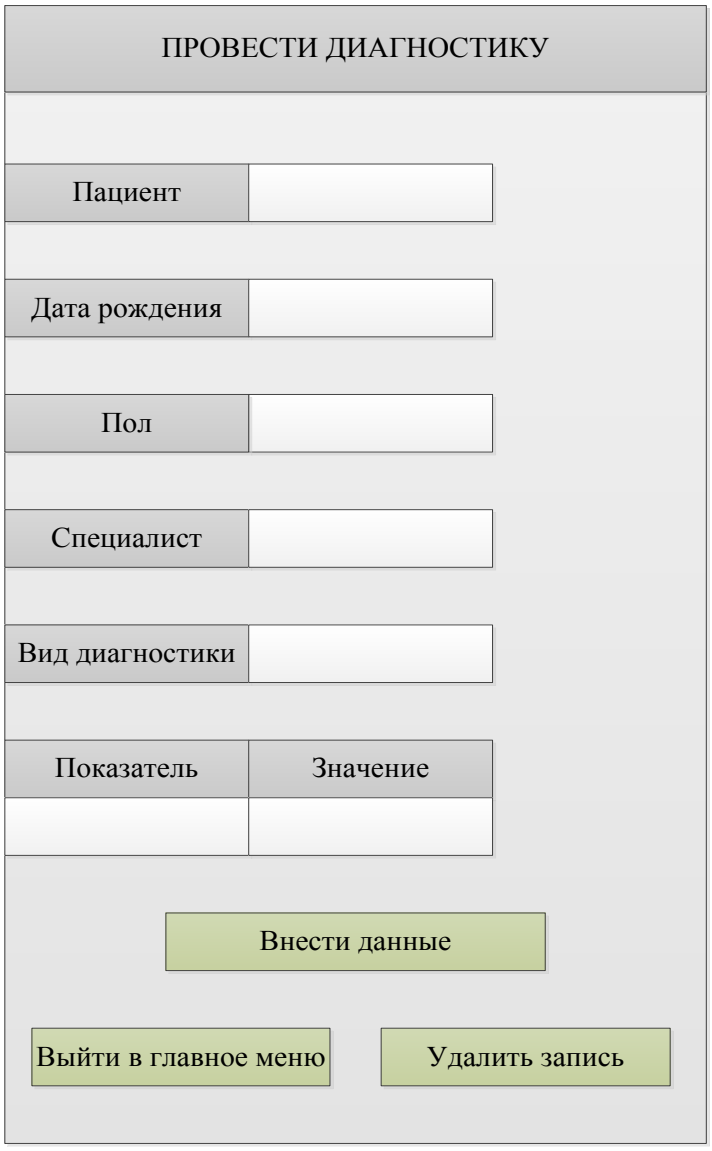


Рисунок 4.4 – *Модель кнопки «Провести диагностику»*

4.2.4. Кнопка «Расшифровать результаты»

При переходе по данной кнопке, пользователю программного обеспечения открывается окно, содержащее следующую информацию:

- Поле ввода ФИО пациента.
- Поле ввода даты диагностики.
- Кнопка «Получить результаты».

При переходе по кнопке «Получить результаты», отображенной на рисунке 4.5, пользователю программного обеспечения открывается страница, содержащая информацию о проведенных пациентом одного или несколько видов диагностики и результат расшифровки в виде сообщений «Норма» или «Отклонение».

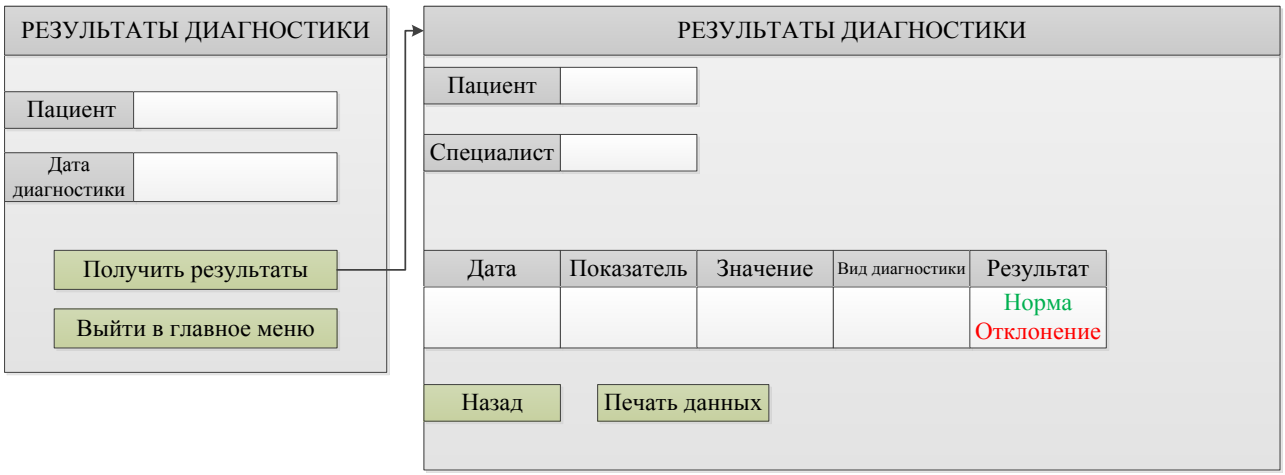


Рисунок 4.5 – *Модель кнопки «Расшифровать результаты»*

4.2.5. Кнопка «Назначить терапию»

При переходе по кнопке «Назначить курс терапии», представленной на рисунке 4.6 открывается окно, позволяющее пользователю назначить курс терапии согласно поставленному на основании проведенных диагностических процедур диагнозу. Кнопка «Подобрать терапию», располагающаяся в данном окне отображает информацию о назначенном курсе терапии, дозировках

лекарственных средств с учетом возраста пациента, частоту и продолжительность приема лекарственных средств.

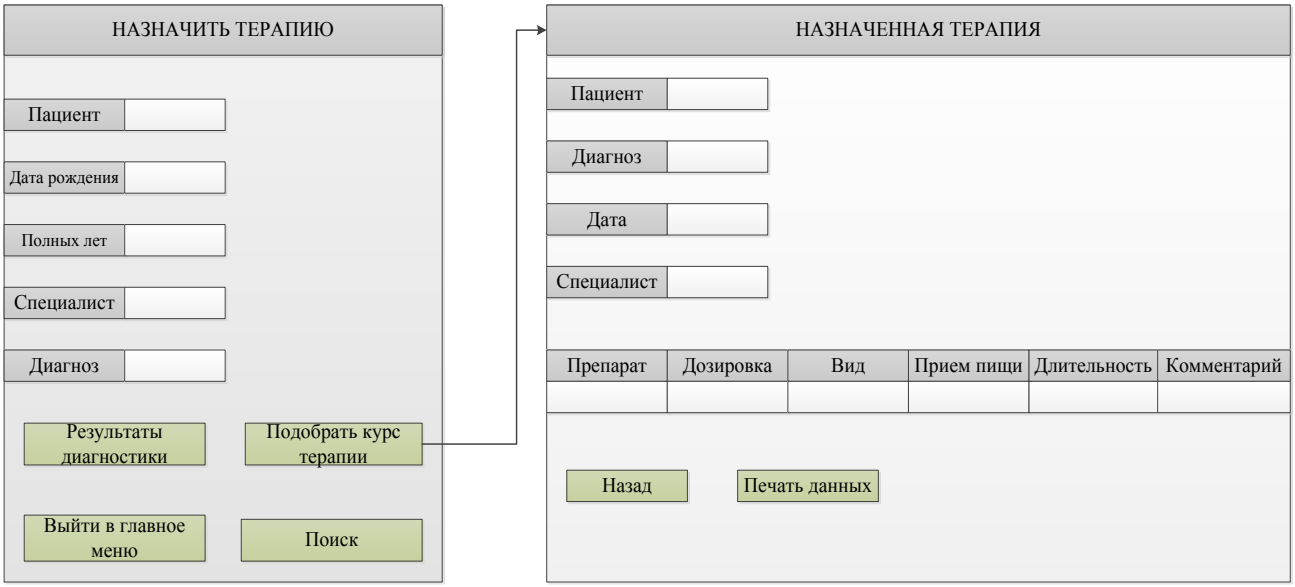


Рисунок 4.6 – *Модель кнопки «Назначить терапию»*

4.2.6. Кнопка «Поиск»

При переходе по кнопке «Поиск», представленной на рисунке 4.7 открывается окно, предоставляющее пользователю произвести поиск информации о пациенте по следующим направлениям:

- Общая информация о пациенте.
- Поиск истории обращений пациента.
- Поиск истории жалоб пациента.
- Поиск ранее проведенной диагностики и результаты расшифровки.
- Поиск назначенного курса терапии и дозировка.

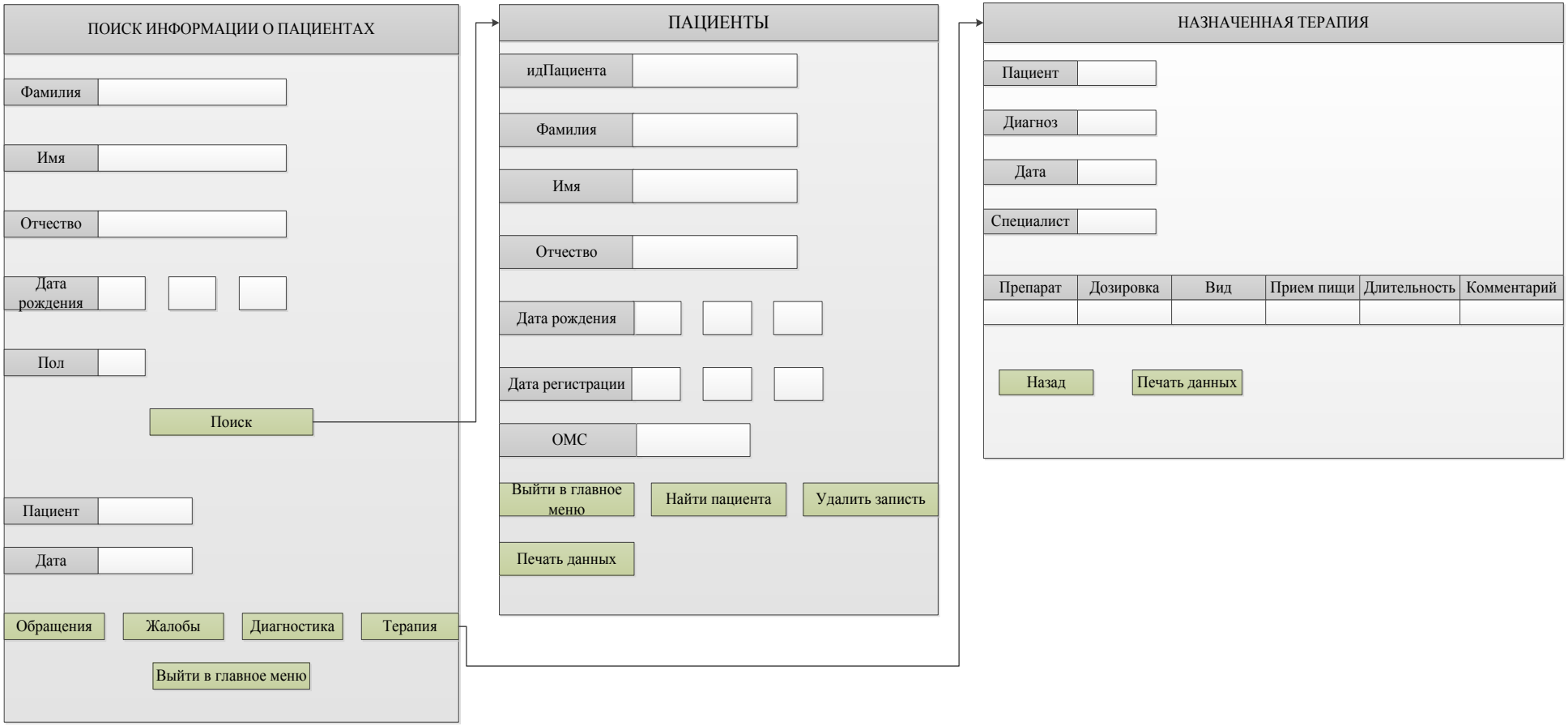


Рисунок 4.7 – *Модель кнопки «Поиск» с примерами поиска информации*

4.3. Схема приложения

На рисунке 4.8 построим схему приложения (часть 1), отображающую взаимодействие смоделированных интерфейсов между собой.

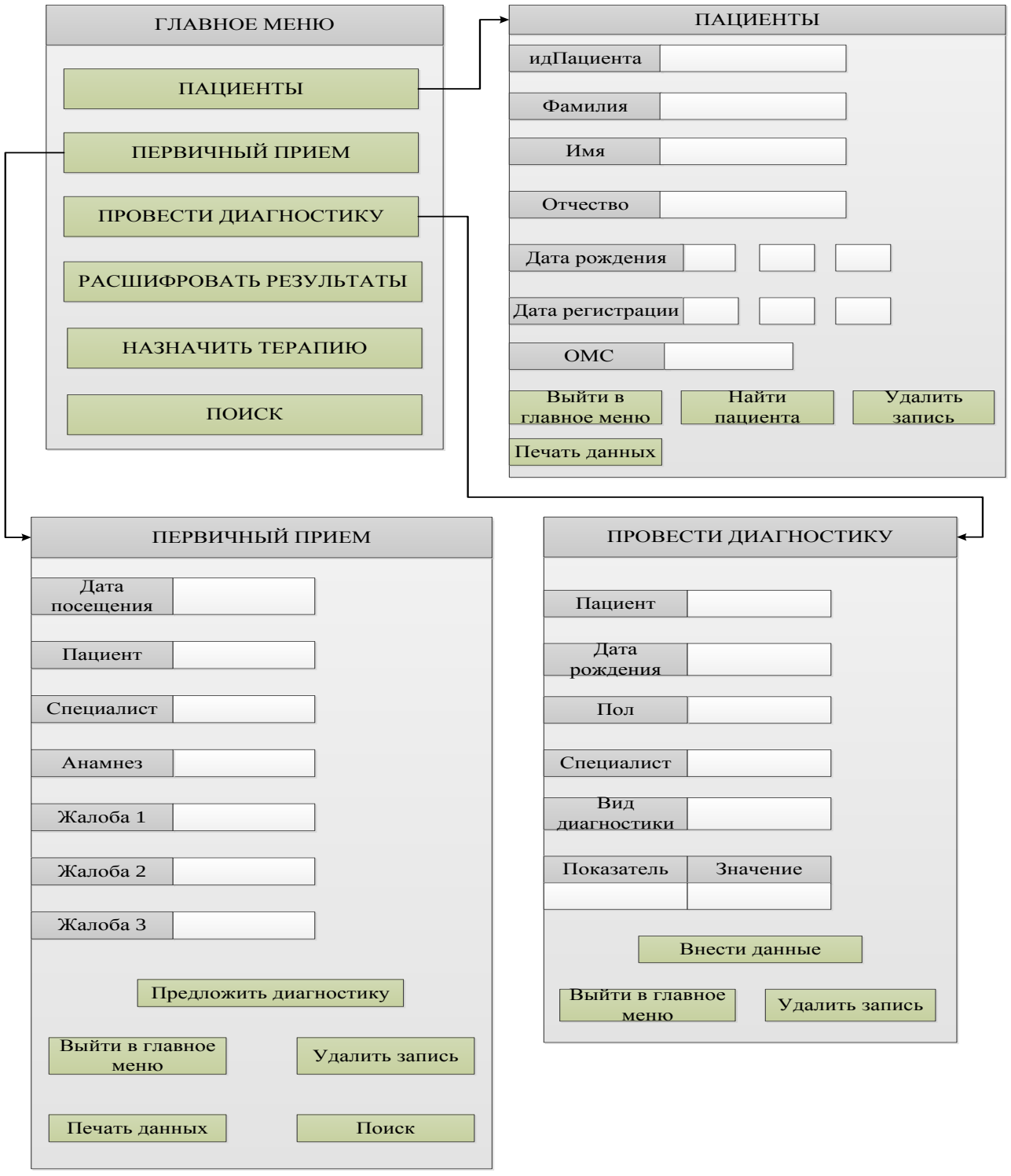


Рисунок 4.8 – *Схема приложения (часть 1)*

Построим схему приложения (часть 2). Отобразим результат на рисунке 4.9.

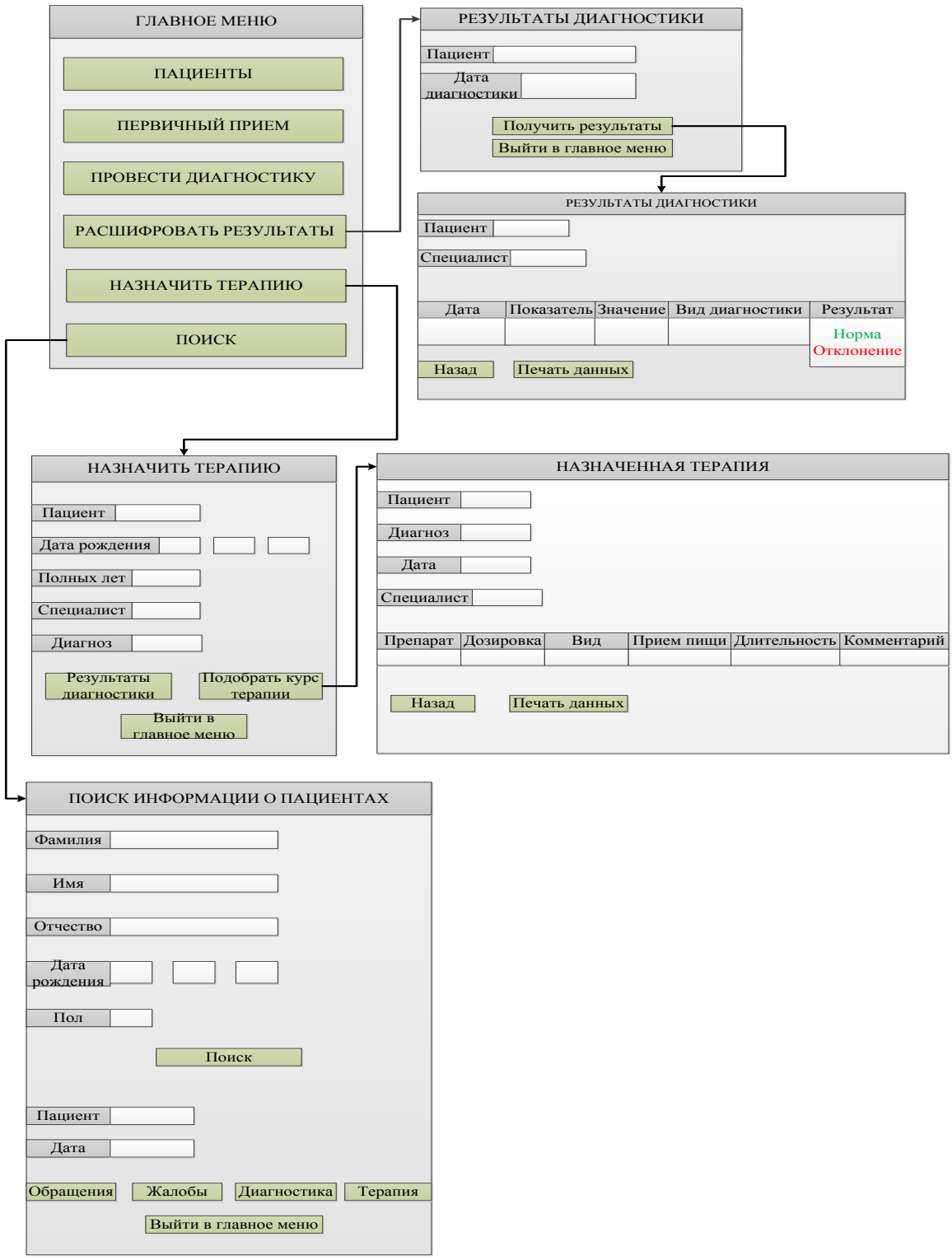


Рисунок 4.9 – *Схема приложения (часть 2)*

Раздел 5. Проектирование структуры данных (итерация 1)

5.1. Нормализация данных до 3НФ

Проектирование базы данных – процесс создания схемы базы данных, а так же определение необходимых ограничений целостности. Ключевая стадия проектирования структуры данных состоит в нормализации данных, что подразумевает процесс распределения данных по различным взаимосвязанным таблицам. Главная задача этого этапа состоит в устранении избыточности, дублирующихся данных и предупреждении повреждения их целостности в процессе преобразования.

Таким образом, описанные в разделе 2 данные, необходимо нормализовать. Существует несколько правил нормализации. Каждое правило называется нормальной формой [15]. Основными нормальными формами являются:

- Первая нормальная форма (1NF). Первая нормальная форма – подразумевает соблюдение следующих правил:
  - Определение ключевых полей.
  - Устранение повторяющихся групп – на пересечении каждого столбца и каждой строки содержится только одно (атомарное значение), а не множество значений.
  - Все атрибуты должны зависеть от первичного ключа. Таким образом, необходимо задать ключевые поля для таблиц, описанных в функциональных требованиях в разделе 2 и привести поля в таблицах к соответствию условия атомарности: одно поле – одно значение.
- Вторая нормальная форма (2NF). Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме, а

так же все не ключевые атрибуты зависят только от первичного ключа. Для приведения разрабатываемого программного обеспечения ко второй нормальной форме необходимо создание таблиц – справочников для не ключевых атрибутов. В проектируемой БД были созданы следующие таблицы-справочники:

- Жалобы.
- Диагнозы.
- Виды диагностики.
- Лекарства.

- Третья нормальная форма (3NF). Отношение находится в третьей нормальной форме (3NF), если оно находится во второй нормальной форме, и каждый не ключевой атрибут зависит только от первичного ключа и не зависят друг от друга.

Представим классы данных проектируемой базы данных. Отобразим результат в таблице 5.1.

Таблица 5.1 – *Классы данных согласно третьей нормальной форме*

Класс данных	Данные	Тип данных	Размерность
Пациенты	идПациента	Счетчик	Длинное целое
	Фамилия	Текстовый	20
	Имя	Текстовый	20
	Отчество	Текстовый	20
	Дата рождения	Дата/Время	Краткий формат даты
	Дата регистрации	Дата/Время	Краткий формат даты
	Пол	Текстовый	10
Специалисты	идСпециалиста	Счетчик	Длинное целое
	Фамилия	Текстовый	20
	Имя	Текстовый	20
	Отчество	Текстовый	20
Диагнозы	идДиагноза	Счетчик	Длинное целое
	Наименование диагноза	Текстовый	50
Жалобы	идЖалобы	Счетчик	Длинное целое
	Жалоба	Текстовый	100
Виды диагностики	идДиагностики	Счетчик	Длинное целое

Класс данных	Данные	Тип данных	Размерность
	Наименование диагностики	Текстовый	50
Показатели нормы	🔑 идПоказателя	Счетчик	Длинное целое
	Наименование показателя	Текстовый	50
	минЗначение	Числовой	Одинарное с плавающей точкой
	максЗначение	Числовой	Одинарное с плавающей точкой
	идДиагностики	Подстановка и отношение	Длинное целое
Проведенная диагностика	🔑 Код	Счетчик	Длинное целое
	Дата	Дата/Время	Краткий формат даты
	идПациента	Подстановка и отношение	Длинное целое
	идПоказателя	Подстановка и отношение	Длинное целое
	значениеПоказателя	Числовой	Одинарное с плавающей точкой
	идСпециалиста	Подстановка и отношение	Длинное целое
Лекарства	🔑 Код	Счетчик	Длинное целое
	Наименование	Текстовый	50
Диагнозы и лекарства	🔑 Код	Счетчик	Длинное целое
	идДиагноза	Подстановка и отношение	Длинное целое
	идЛекарства	Подстановка и отношение	Длинное целое
	Вид	Текстовый	50
	частотаПриема	Текстовый	50
	Дозировка возраст 1	Числовой	Длинное целое
	Дозировка возраст 2	Числовой	Длинное целое
	частотаПриема	Текстовый	50
	приемПищи	Текстовый	50
	продолжительностьПриема	Числовой	Длинное целое
	Комментарий	Поле MEMO	255
Первичный прием	🔑 идПриема	Счетчик	Длинное целое
	Дата	Дата/Время	Краткий формат даты
	идПациента	Подстановка и отношение	Длинное целое
	идСпециалиста	Подстановка и отношение	Длинное целое
	данныеАнамнеза	Поле MEMO	255
ПациентыЖалобы	🔑 Код	Счетчик	Длинное целое
	идПациента	Подстановка и отношение	Длинное целое
	идЖалобы	Подстановка и отношение	Длинное целое

Класс данных	Данные	Тип данных	Размерность
	Дата	Дата/Время	Краткий формат даты
Направление по жалобам	идЖалобы	Подстановка и отношение	Длинное целое
	идДиагностики	Подстановка и отношение	Длинное целое
Назначения	идНазначения	Счетчик	Длинное целое
	Дата	Дата/Время	Краткий формат даты
	идСпециалиста	Подстановка и отношение	Длинное целое
	идПрепДоз	Подстановка и отношение	Длинное целое

5.2. Схема данных

Схема БД (Рисунок 5.1) – совокупность всех схем таблиц, которые в нее входят, характеристика всех колонок, их видов, уместных значений, связей между таблицами, не принимая во внимание определенные данные. Существуют следующие связи между таблицами [16]:

- «Один ко многим» – При данном типе связи одной строке первой таблицы может отвечать большое количество строк другой таблицы. При этом каждой строке второй таблицы отвечает только одна строка первой таблицы.
- «Многие ко многим» – Связь, при которой каждой строке таблицы 1 может соответствовать множество строк таблицы 2 и наоборот. Такая связь создается при помощи третьей таблицы, называемой соединительной, первичный ключ которой состоит из внешних ключей, связанных с таблицами 1 и 2.
- «Один к одному» – Связь, при которой каждой строке таблицы 1 может соответствовать только одна строка таблицы 2 и наоборот.

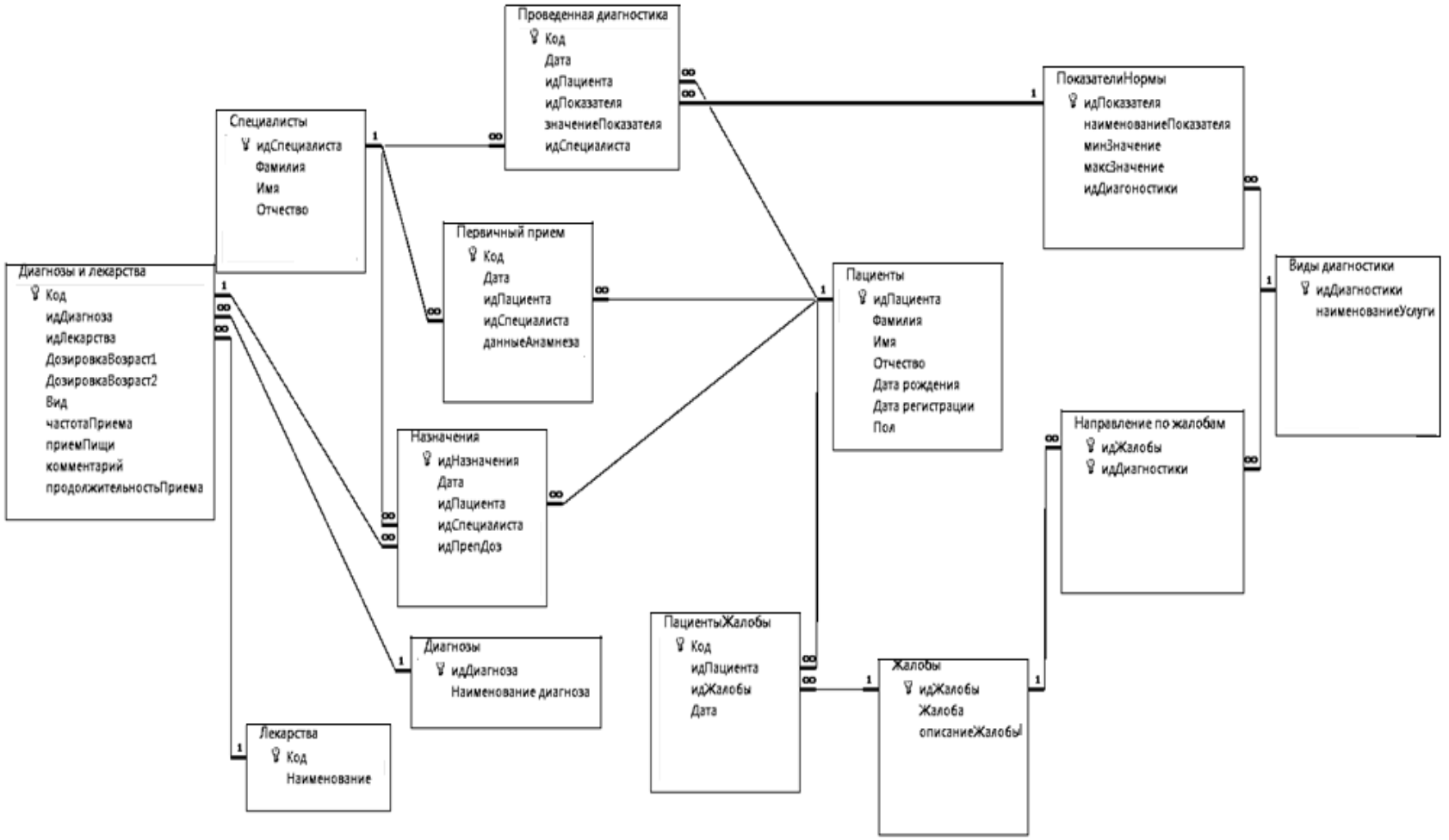


Рисунок 5.1– *Схема данных*

Раздел 6. Разработка программного обеспечения

6.1. Итерация 2: Реализация требований 1-3

Согласно методологии разработки программного обеспечения Agile Kanban, отобразим Kanban доску на начальном этапе разработки программного обеспечения, что соответствует итерации 2. Результат представим на рисунке 6.1.

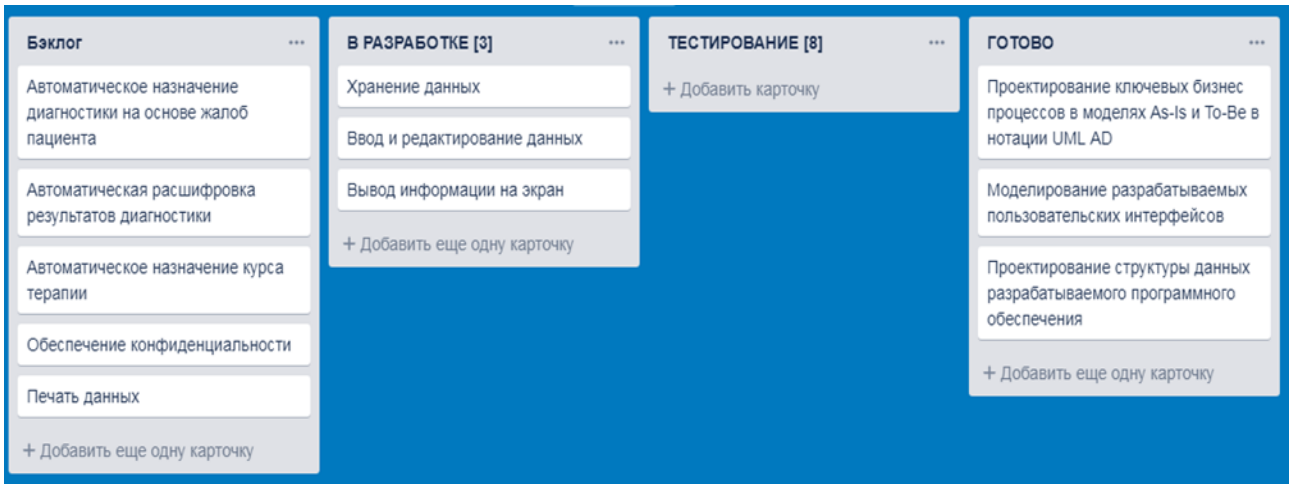


Рисунок 6.1– *Kanban доска на этапе реализации итерации 2*

Таким образом, в ходе реализации итерации 2, необходимо выполнить следующие пользовательские требования: хранение данных, ввод и редактирование данных, вывод информации на экран.

6.1.1. Пользовательское требование «Хранение данных»

Реализация данного требования заключается в создании в среде разработки программного обеспечения – Microsoft Access таблиц, описанных в функциональных требованиях в разделе 2. Отобразим полученный результат на рисунке 6.2.

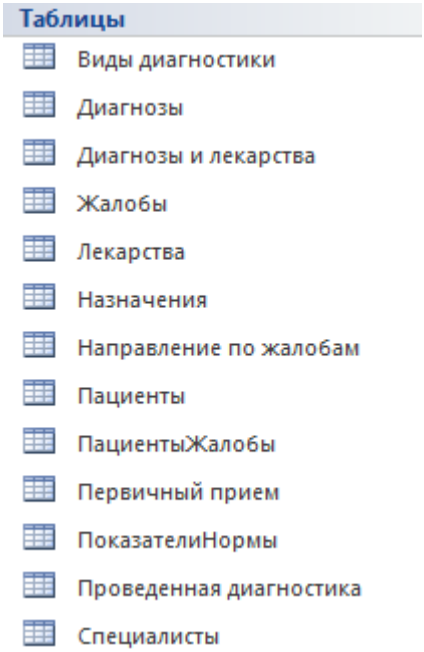


Рисунок 6.2 – *Созданные в среде MS Access таблицы*

В качестве проверки корректности введенных данных, на рисунке 6.3 приведем фрагмент созданной таблицы «Пациенты».

идПациента	Фамилия	Имя	Отчество	Дата рождения	Пол	Дата регистраци
1	Петренко	Вадим	Сергеевич	13.12.2010	М	10.10.2017
2	Максимов	Иван	Петрович	29.11.1992	М	30.10.2018
3	Денисова	Виктория	Андреевна	01.07.1980	Ж	06.09.2017
4	Петров	Василий	Дмитриевич	11.09.2001	М	19.08.2015
5	Фролова	Анастасия	Максимовна	05.03.1974	Ж	13.11.2015
6	Борисов	Илья	Андреевич	17.05.1982	М	26.12.2017
7	Семенова	Мария	Васильевна	10.10.2002	Ж	14.03.2018
8	Фролова	Ольга	Юрьевна	20.12.1962	Ж	06.09.2017
9	Макеев	Илья	Александрови	12.01.1985	М	03.09.2014
10	Круглова	Ирина	Витальевна	20.03.1995	Ж	12.10.2015

Рисунок 6.3 – *Фрагмент таблицы «Пациенты»*

6.1.2. Пользовательское требование «Ввод и редактирование данных»

Реализация данного требования заключается в выполнении функционального требования – создание форм. Формы – это объекты, предназначенные для ввода и отображения данных. На рисунке 6.4 представим форму, позволяющую автоматизировать один из процессов больницы, а именно «Составить бланк диагностики».

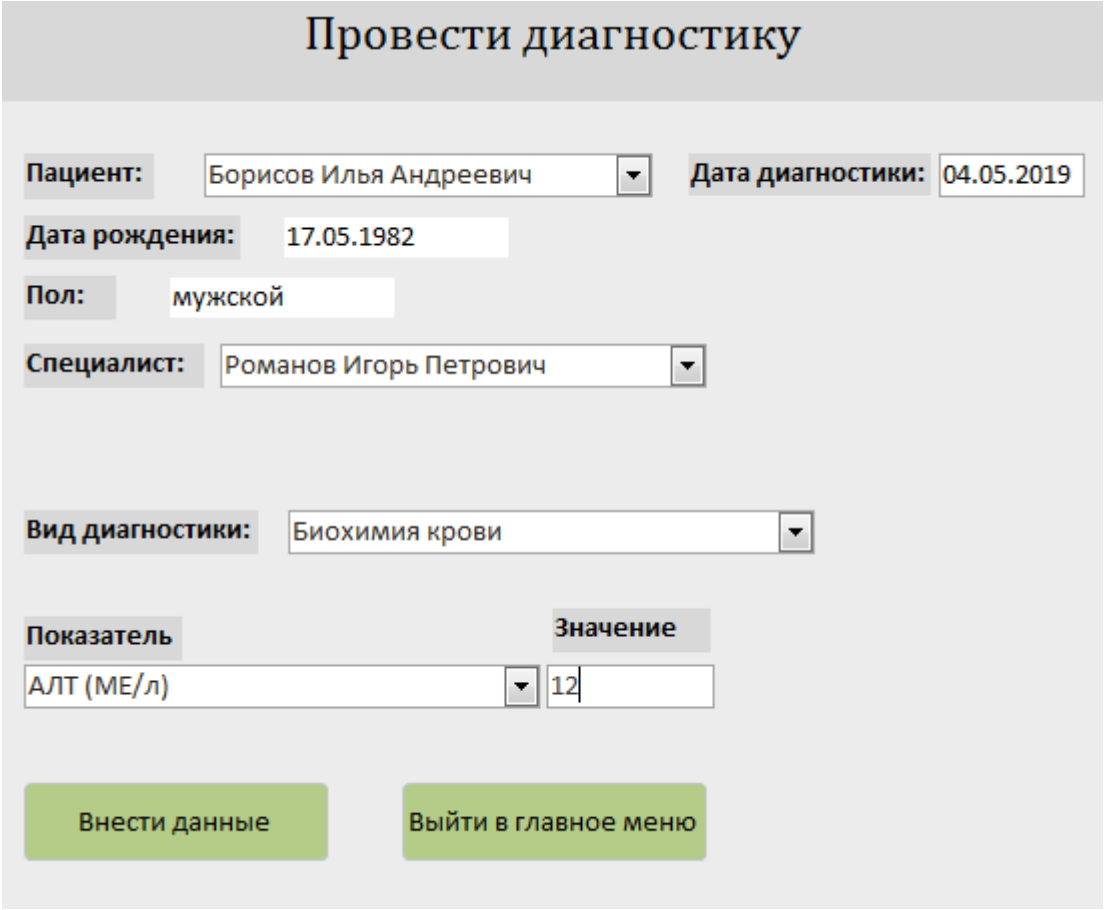


Рисунок 6.4 – *Электронный бланк диагностики*

В качестве проверки требования редактирования данных, на рисунке 6.5 представим запрос на удаление данных о пациенте.

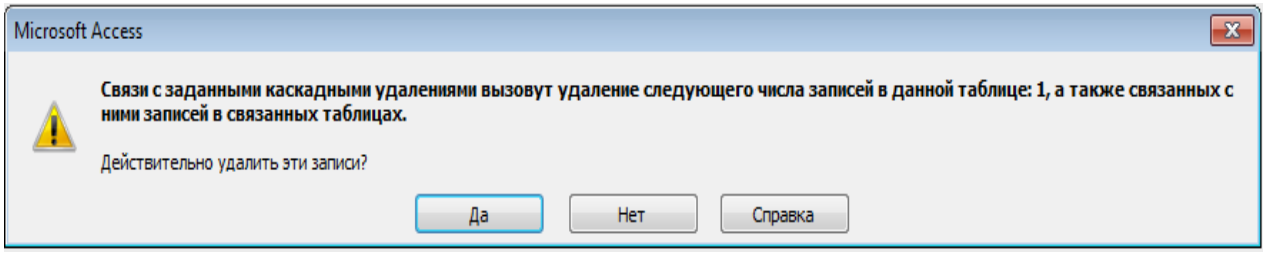


Рисунок 6.5 – *Запрос на удаление данных*

6.1.3. Пользовательское требование «Вывод информации на экран»

Для реализации данного пользовательского требования необходимо выполнение функциональных требований – создание форм и создание отчетов.

На рисунке 6.6 представим функцию поиска информации о пациенте.

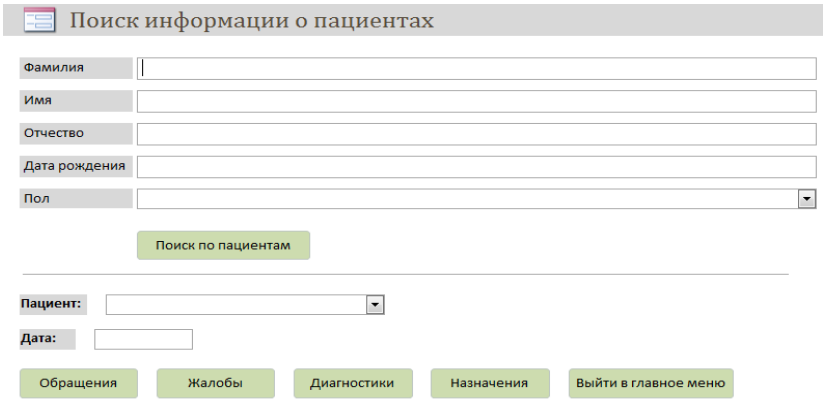


Рисунок 6.6 – *Функция поиска данных о пациенте*

В качестве проверки данной функции произведем поиск по обращениям по различным датам для определенного пациента. Результат поиска представим на рисунке 6.7.

Дата	идПациента	идСпециалиста	данныеАнамнеза
04.05.2019	Куликова Татьяна Ивановна	Щербаков Сергей Михайлович	Боли в правом подреберье,возможный диагноз гастрит
15.05.2019	Куликова Татьяна Ивановна	Юдина Ирина Альбертовна	По данным опроса жалоб и физической диагностики предполагаемый диагноз - поверхностный гастрит

Рисунок 6.7 – *Результат поиска по обращениям*

6.2. Итерация 3: Реализация требований 4–6

Представим на рисунке 6.8 вид Kanban доски на момент начала реализации итерации 3.

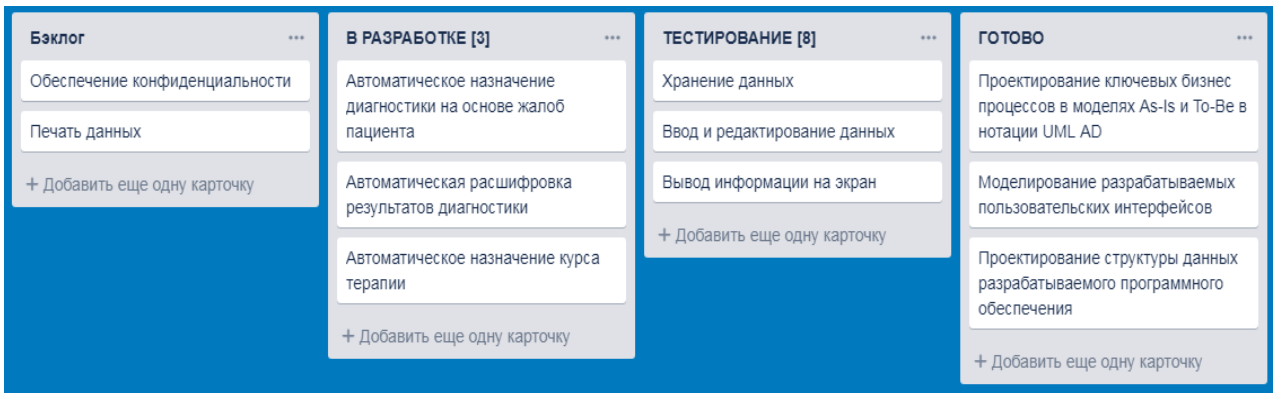


Рисунок 6.8 – *Kanban доска на этапе реализации итерации 3*

Таким образом, в ходе реализации итерации 3 необходимо разработать ключевые функции программного обеспечения, а именно: автоматическое назначение диагностики на основе жалоб пациента, автоматическая расшифровка результатов диагностики, автоматическое назначение курса терапии.

6.2.1. Пользовательское требование «Автоматическое назначение диагностики на основе жалоб пациента»

Для реализации данного пользовательского требования, позволяющего автоматизировать бизнес процесс – «Провести первичный прием» необходимо выполнение следующих действий:

- Установление комбинаций между жалобами из таблицы «Жалобы» и видами диагностики из таблицы «Виды диагностики».
- Создание запроса на предоставление данных.
- Создание отчета для отображения данных.

На рисунке 6.9 отобразим форму, служащую для записи данных о первичном приеме и автоматического назначения диагностики, на основе жалоб пациента.

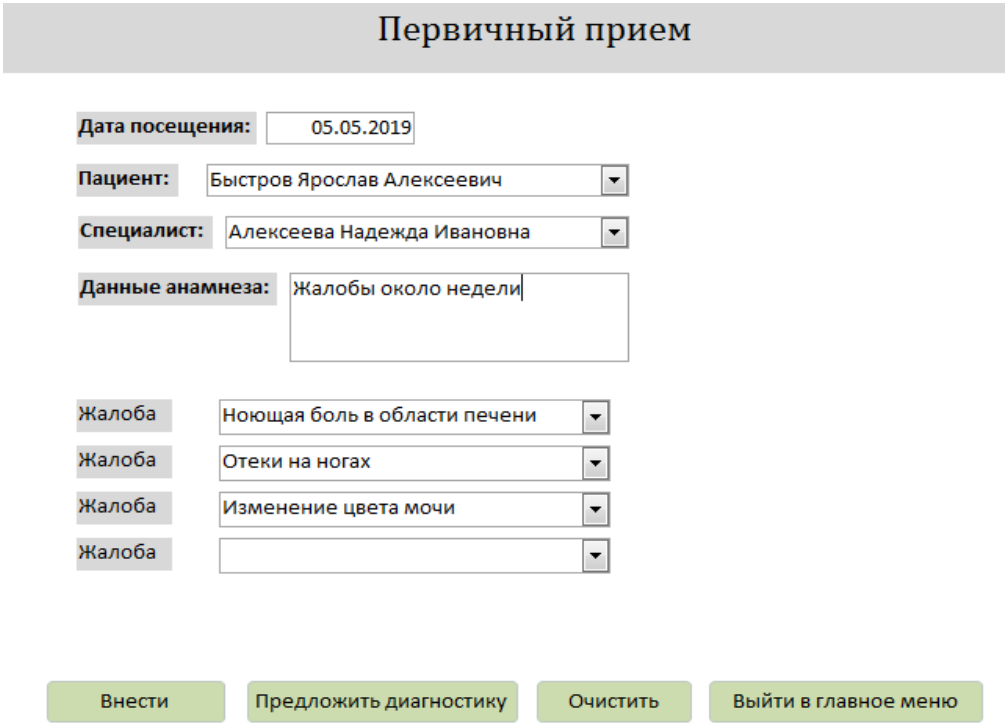


Рисунок 6.9 – *Окно ввода жалоб пациента*

В качестве проверки данной функции, на рисунке 6.10 отобразим результат нажатия на кнопку «Предложить диагностику».

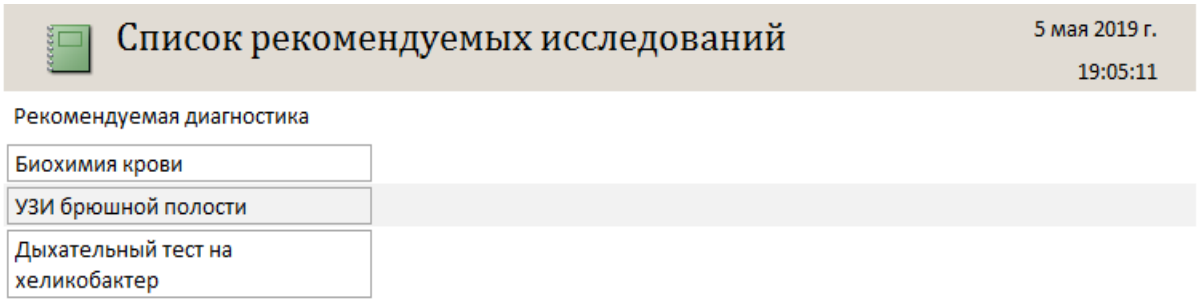


Рисунок 6.10 – *Результат нажатия кнопки «Предложить диагностику»*

6.2.2. Пользовательское требование «Автоматическая расшифровка результатов диагностики»

Для реализации данного пользовательского требования необходимо выполнение следующих задач: запись диапазона нормальных значений для каждого диагностического показателя, создание параметрического параметра.

Для выполнения функции сравнения диагностических показателей необходимо в создаваемый параметрический запрос добавить следующее условие: *If([Анализы]![значениеПоказателя]<[ПоказателиНормы] ! [минЗначение] Or [Анализы] ! [значениеПоказателя] >[ПоказателиНормы] ! [максЗначение]; "Отклонение"; «Норма").*

В данном алгоритме проходит проверка условия: если значение показателя пациента меньше минимального значения или больше максимального значения (из таблицы «Показатели нормы»), то в поле "Результат" выводится сообщение "Отклонение". В противном случае в поле "Результат" выводится сообщение "Норма". Отобразим результат выполнения данной функции на рисунке 6.11.

Дата	Показатель	Значение	Наименование диагностики	Результат
30.04.2019	Длина поджелудочной железы (мм)	90	УЗИ брюшной полости	Отклонение
30.04.2019	Длина печени (мм)	100	УЗИ брюшной полости	Отклонение
30.04.2019	Диаметр холедоха (мм)	3	УЗИ брюшной полости	Норма
30.04.2019	Диаметр селезеночной вены (мм)	8	УЗИ брюшной полости	Норма

Рисунок 6.11 – *Автоматическая расшифровка результатов диагностики*

Приведем блок схему разработанной функции. Результат представим на рисунке 6.12.

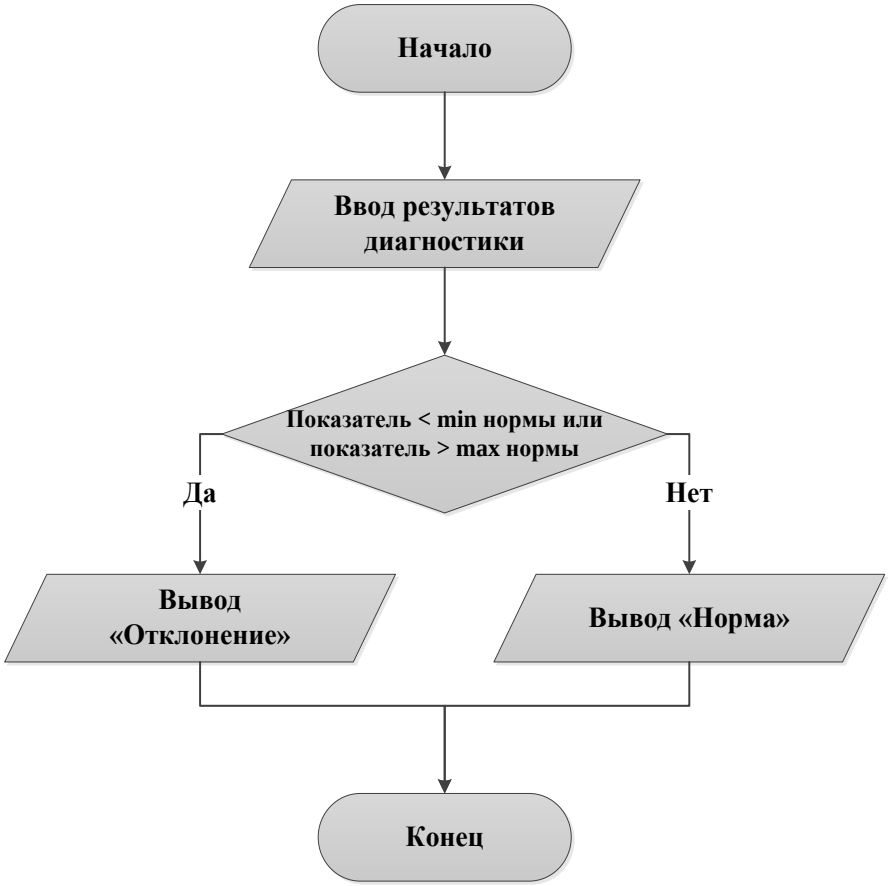


Рисунок 6.12 – *Блок схема разработанной функции*

6.2.3. Пользовательское требование «Автоматическое назначение курса терапии»

Для реализации данного требования необходимо выполнение следующих задач:

- Установление соотношение между диагнозами из таблицы «Диагнозы» и лекарственными препаратами из таблицы «Лекарства».
- Установление дозировок лекарственных препаратов для возрастной группы до 12 лет («Дозировка 1») и после 12 лет (Дозировка 2).
- Выполнение программного кода на языке VBA для автоматического определения возраста пациента.

- Выполнение SQL запроса на установку дозировки лекарственных препаратов на основе вычисленного возраста пациента и установленного диагноза.

На рисунке 6.13 отобразим блок схему разрабатываемого процесса.

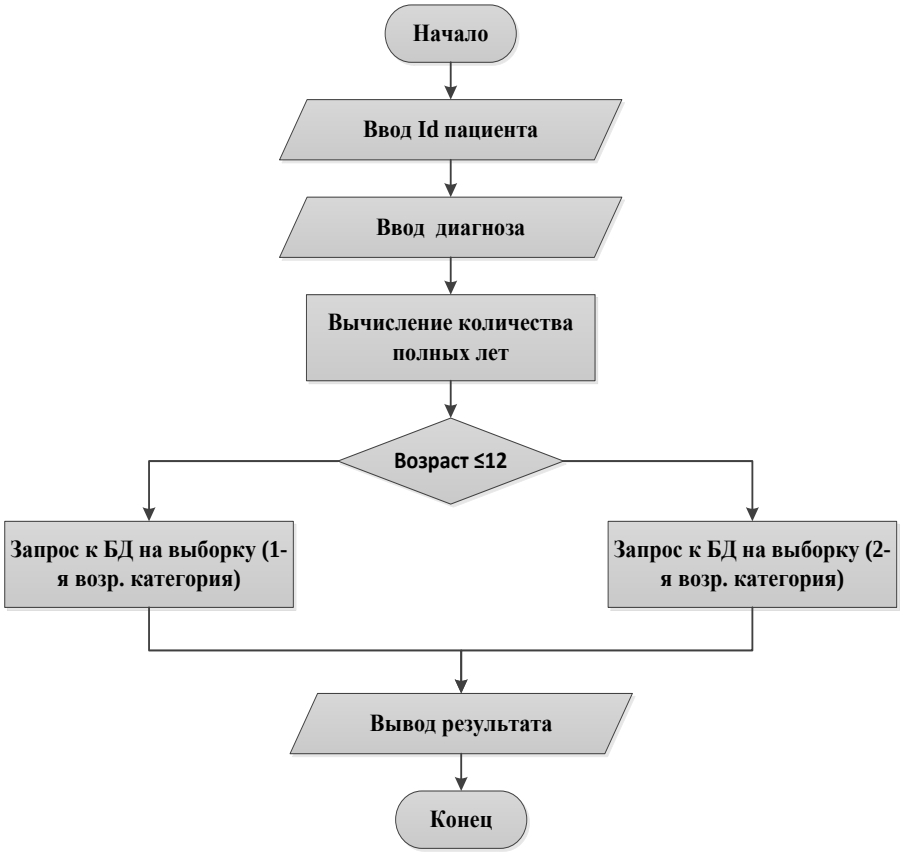


Рисунок 6.13 – *Блок схема разрабатываемого процесса*

Представим на рисунке 6. 14 программный код на языке VBA (Visual Basic for Applications), позволяющий в реальном времени установить возраст пациента для определения возрастной группы с целью дальнейшего подбора дозировки лекарственных препаратов.

```
Function ageToNow(varDOB As Variant, Optional varAsOf As Variant) As Variant

    Dim dtDOB As Date
    Dim dtAsOf As Date
    Dim dtBDay As Date

    ageToNow = Null


    If IsDate(varDOB) Then
        dtDOB = varDOB

        If Not IsDate(varAsOf) Then
            dtAsOf = date
        Else
            dtAsOf = varAsOf
        End If

        If dtAsOf >= dtDOB Then
            dtBDay = DateSerial(Year(dtAsOf), Month(dtDOB), Day(dtDOB))
            ageToNow = DateDiff("yyyy", dtDOB, dtAsOf) + (dtBDay > dtAsOf)
        End If
    End If
End Function
```

```
Private Sub client_drugs_AfterUpdate()

date_of_birth = DLookup("[Дата рождения]", "Пациенты", "идПациента = " & Me.client_drugs.Column(0))

Me.dateOfBirth.Value = date_of_birth

Me.age_years.Value = ageToNow(date_of_birth)

Me.todayDate.Value = date

End Sub
```

Рисунок 6.14 – *Листинг программы*

Таким образом, на рисунке 6.15 представим функцию автоматического назначения курса терапии.

Назначение лекарственных препаратов

5 мая 2019 г.
19:15:53

Пациент: Круглова Ирина Витальевна **Диагноз:** Хронический поверхнос

Специалист: Лукин Виктор Викторович

Препарат	Дозировка	Вид	Прием пищи	Длительность
Кларитромицин	2	Таблетка	15 минут до еды	2 Недели
Гастрацид	2	Таблетка	25 минут до еды	2 Недели
Гевискон	2	Суспензия	2 часа после еды	Месяц

Рисунок 6.15 – *Автоматическое назначение курса терапии*

6.3. Итерация 4: Реализация требований 7–8

На рисунке 6.16 отобразим вид Kanban доски на момент реализации итерации 4.

Рисунок 6.16 – *Kanban доска на этапе реализации итерации 4*

Таким образом, в ходе реализации данной итерации необходимо выполнить следующие пользовательские требования: обеспечение конфиденциальности. И печать данных.

Для реализации пользовательского требования «Обеспечение конфиденциальности», необходимо задать пароль для разработанной программы. На рисунках 6.17 и 6.18 отобразим запрос на ввод пароля при входе в программу и уведомление об ошибочном вводе пароля.

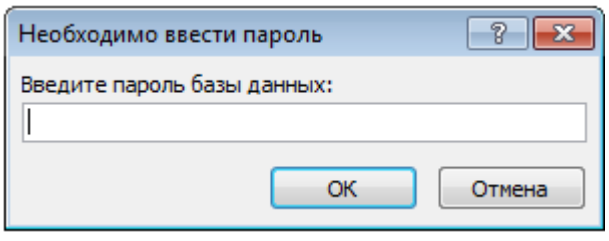


Рисунок 6.17 – *Запрос на ввод пароля при входе в программу*

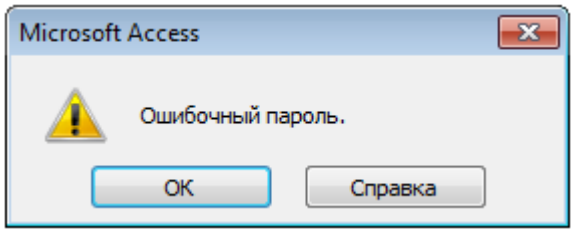


Рисунок 6.18 – *Уведомление при ошибочном вводе пароля*

Для реализации пользовательского требования «Печать данных», необходимо обеспечить передачу данных на устройство печати. Данная функция достигается путем добавления в ранее созданные отчеты кнопки «Печать данных». На рисунке 6.19 отобразим запрос на печать данных о пациенте.

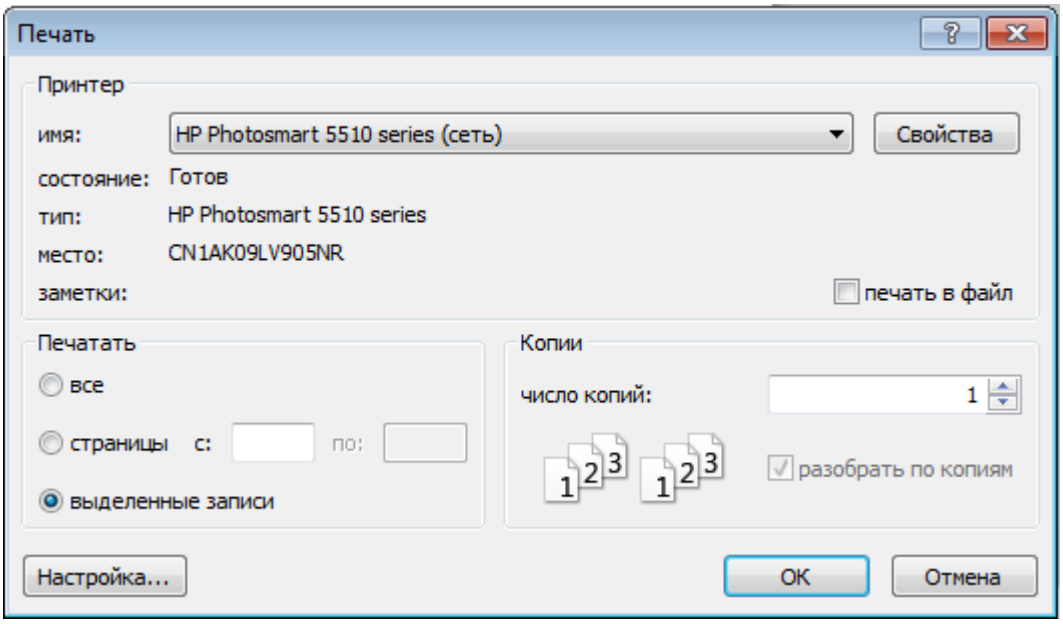


Рисунок 6.19 – *Запрос на печать данных*

6.4. Схема данных разработанной программы

На рисунке 6.20 представим схему данных разработанного программного обеспечения.

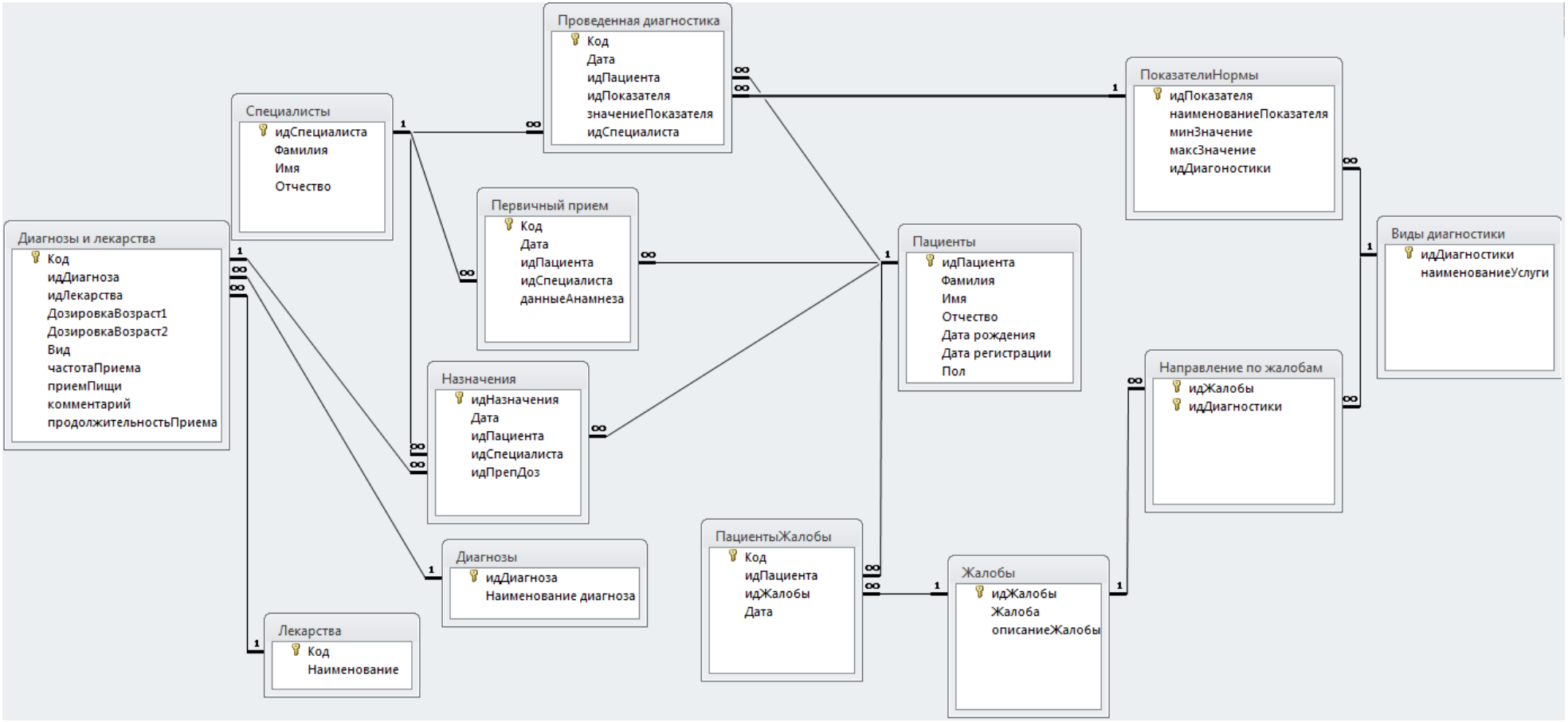


Рисунок 6.20 – *Схема данных разработанной программы*

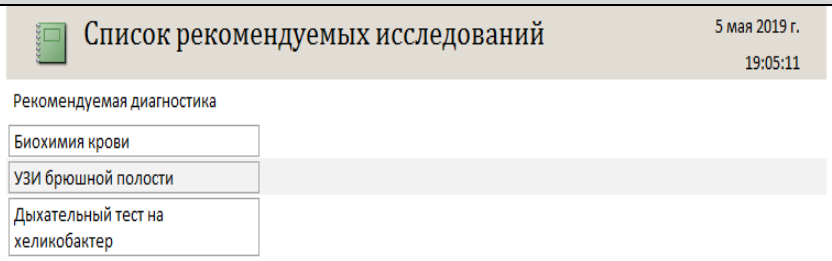
Раздел 7. Тестирование программного обеспечения

Тестирование программного обеспечения – процесс изучения и испытания программы, целью которого является проверка уровня соответствия реального результата запланированным требованиям, осуществляемый на конечном наборе тестов, выбранных определенным образом [17].

7.1. Функциональное и интеграционное тестирование

Функциональное тестирование – представляет собой вид тестирования, целью которого является проверка правильности осуществления работы функционала разработанного программного обеспечения [18]. Проведем функциональное тестирование основных функций разработанного программного обеспечения. Отобразим результат в таблице 7.1.

Таблица 7.1 – *Функциональное тестирование главных функций программы*

Функция	Результат тестирования
Автоматическое назначение диагностики на основе жалоб пациента.	
Автоматическая расшифровка результатов диагностики.	

Функция	Результат тестирования
Автоматическое назначение курса терапии.	

Таким образом, была произведена проверка разработанного функционала программы. В ходе тестирования ошибок обнаружено не было.

Проведем интеграционное тестирование для разработанного программного обеспечения. Интеграционное тестирование – вид тестирования, при котором осуществляется проверка коммуникации между несколькими частями приложения [18]. Для проведения интеграционного тестирования проведем проверку поиска пациента в базе данных. На рисунке 7.1 отобразим страницу поиска информации по пациентам.

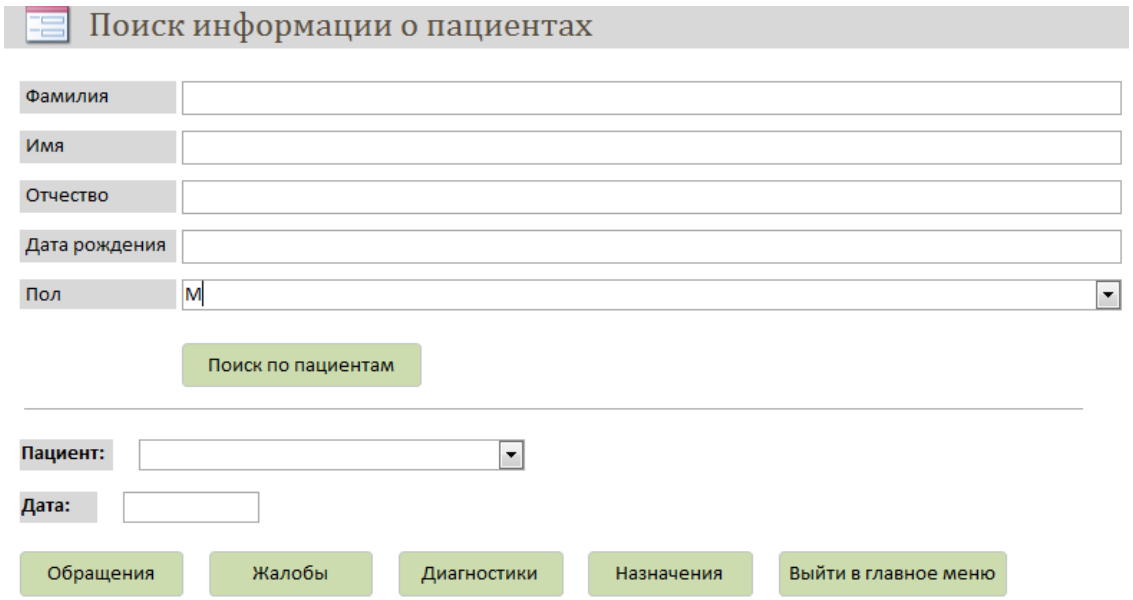


Рисунок 7.1 – *Поиск информации по пациентам*

В качестве критерия поиска установим, всех пациентов мужского пола, зарегистрированных в больнице. Для этого необходимо в графе «Пол» указать «М» и нажать на кнопку «Поиск по пациентам». Отобразим результат поиска на рисунке 7.2.

Фамилия	Имя	Отчество	Дата рождения	Пол
Максимов	Иван	Петрович	29.11.1992	М
Петров	Василий	Дмитриевич	11.09.2001	М
Борисов	Илья	Андреевич	17.05.1982	М
Макеев	Илья	Александрович	12.01.1985	М
Быстров	Ярослав	Алексеевич	11.04.2001	М
Ещенков	Федор	Олегович	10.01.1970	М
Холуев	Иван	Борисович	01.07.1999	М
Никифоров	Анатолий	Максимович	10.06.1972	М

Рисунок 7.2 – *Результат выполнения поиска*

Таким образом, было произведено интеграционное тестирование, в ходе которого ошибок в работе программного обеспечения обнаружено не было.

7.2. Нагрузочное тестирование

Нагрузочное тестирование – исследование способности приложения сохранять заданные показатели качества при нагрузке в допустимых пределах, а так же при некотором превышении этих пределов [18]. Проведем тестирование, позволяющее оценить работоспособность разработанного программного обеспечения при поиске и добавлении различного числа записей. В качестве проверки было выбрано следующее число записей: 1,50,100. Для того, что бы определить время отклика программы необходимо произвести расчет следующих параметров:

- Среднее арифметическое всех значений времени отклика для конкретного числа записей.
- Среднее квадратичное отклонение.
- Погрешность измерений.
- Время отклика.

Проведем проверку отклика программы и запишем результат проверки, как время отклика в секундах – t1,t2,t3 (Таблица 7.1). Рассчитаем среднее арифметическое всех значений времени отклика для конкретного числа записей. Найдем среднее арифметическое по формуле:

$$\overline{t_{\text{арифм.}}} = \frac{\sum_i t_i}{N}. \quad (7.1)$$

Рассчитав среднее арифметическое всех значений времени отклика, необходимо рассчитать среднеквадратичное отклонение. Воспользуемся формулой для нахождения среднеквадратичного отклонения:

$$\sigma = \sqrt{\frac{\Delta t_i{}^2}{N}}. \quad (7.2)$$

Далее необходимо рассчитать погрешность измерения по формуле:

$$\Delta t = \sqrt{(\frac{\sigma}{N} \cdot t_{\alpha(N-1)})^2 + \Delta \mathrm{t_p}^2}, \quad (7.3)$$

где $t_{\alpha(N-1)}$ – доверительный коэффициент Стьюдента, равный 0.95, $\mathrm{t_p}$ - абсолютная погрешность электронного секундомера, которым проводилось измерение равная 0.005.

Таким образом, найдем время отклика приложения при обработке следующего числа записей: 1,50,100 по формуле:

$$t_{\text{откл.}} = t_{\text{арифм.}} \pm \Delta t, \quad (7.4)$$

Таблица 7.2 – *Определение времени отклика программы*

Кол-во Записей	Действие	t1,с	t2,с	t3,с	Среднее время отклика	Средне квадратичное отклонение	Погрешн. Измерений Δt	Время отклика $t_{откл}$
1	Запись	0,1	0,12	0,09	0,3	0.00941	0,02	0,09±0,014
	Поиск	0,1	0,13	0,1	0.1	0.00821	0,02	0.12±0.017
50	Запись	0,2	0,27	0,2	0,223	0.02	0,04	0,197±0,023

Кол-во Записей	Действие	t1,с	t2,с	t3,с	Среднее время отклика	Средне квадратичное отклонение	Погрешн. Измерений Δt	Время отклика $t_{откл}$
	Поиск	0,23	0,2	0,21	0,213	0.0231	0,03	0,243±0,021
100	Запись	0,3	0,24	0,37	0,304	0.019	0,04	0,39±0,03
	Поиск	0,35	0,31	0,29	0,31	0.0246	0,043	0,32±0,034

Раздел 8. Экономическое обоснование проекта

Произведем расчет сметы затрат на разработку программного обеспечения. Смета затрат представляет собой расчет затрат за определенный период, составленный по экономическим элементам затрат. Смета затрат на разработку программного обеспечения включает следующие статьи:

- Материальные затраты.
- Затраты на оплату труда.
- Амортизационные отчисления.
- Прочие затраты.

8.1. Расчет материальных затрат

Так как выполнение проекта осуществлялось на основе ранее приобретенного персонального компьютера (ПК) со всем необходимым ПО, то к материальным затратам относятся только затраты на электроэнергию.

Произведем расчет по следующей формуле:

$$З_{эл} = P \times Ц_{эл} \times T_{и}, \quad (8.1)$$

где Р – потребляемая мощность оборудования, кВт/ч., $Ц_{эл}$ – стоимость одного кВт/ч, руб., $T_{и}$ – время использования оборудования при проведении работ, ч.

Работа выполнялась на ПК, мощностью 450Вт, а так же при использовании устройства печати мощностью 240Вт. Время использования ПК при реализации проекта – 30 дней по 8 часов в день, время использования принтера – 2 часа.

Стоимость одного кВт/ч по данным Мосэнергосбыта на 2019 год составляет 5,38 руб. Таким образом, произведем расчет затрат на электроэнергию:

$$З_{эл} = 0,45 * 5,38 * 30 * 8 + 0,24 * 5,38 * 2 = 583,6 \text{ руб.}$$

8.2. Расчет затрат на оплату труда

Произведем расчет затрат на заработную плату для всех участников проекта. Таким образом, длительность всего проекта занимает 30 дней, из которых 15 – руководитель, 30 – разработчик.

По данным HeadHunter.ru, средняя зарплата руководителя проекта по разработке ПО, составляет 160 тыс. руб. в месяц, средняя зарплата разработчика – 60 тыс. рублей в месяц. Количество рабочих дней в месяце составляет 22 дня.

Заработная плата руководителя проекта и разработчика: $ЗП_{рук} = \frac{160000}{22} * 15 =$ 109090,9 руб., $ЗП_{исп} = \frac{60000}{22} * 30$ = 81818,2 руб. Таким образом, рассчитаем основную заработную плату по формуле 8.2:

$$З_{осн} = ЗП_{рук} + ЗП_{исп}, \quad (8.2)$$

$З_{осн}$ = 109090,9 + 81818,2 = 190909,1 руб. Пусть дополнительная заработная плата составляет 10% от основной, тогда: $З_{доп}$ = 0,1 * 190909,1 = 19090,9 руб. Таким образом, вычислим фонд оплаты труда по формуле:

$$Ф_{зп} = З_{осн} + З_{доп}, \quad (8.3)$$

Фонд оплаты труда составит, $Ф_{зп}$ = 190909,1 + 19090,9 = 210000,0 руб.

8.3. Расчет амортизационных отчислений

Произведем расчет амортизационных отчислений по формуле:

$$A_{нир} = \frac{Ф_{перв} \times T_{и} \times H_{а}}{Ф_{эф}}, \quad (8.4)$$

где $Ф_{перв}$ – первоначальная стоимость оборудования, руб., $Т_{и}$ – время использования оборудования при проведении работ, дн., $Н_{а}$ – норма амортизации, $Ф_{эф}$ – годовой эффективный фонд времени работы оборудования, для односменной работы он составляет 256 дн.

Для расчета амортизационных отчислений примем, что стоимость ПК составляет 45000 руб., а срок его полезного использования – 5 лет. Произведем расчет нормы амортизации по формуле:

$$H_a = \frac{1}{T_{пи}}, \quad (8.5)$$

где $Т_{пи}$ – срок службы оборудования (5 лет), следовательно, $Н_а$ = 0,2. Таким образом, $А_{нир} = \frac{45000*30*0,2}{256} = 1054,7$ руб.

8.4. Прочие затраты

Для расчета прочих затрат ($З_{пр}$) необходимо определить:

- Затраты на страховые взносы ($З_{стр}$).
- Затраты на дополнительные прочие затраты ($З_{прдоп}$).

По данным Федеральной налоговой службы РФ, совокупный тариф страховых взносов в 2019 году составляет 30% от величины фонда оплаты труда. Следовательно, $З_{стр}$ = 210000,0 *0,3 = 63000,0 руб.

Дополнительные прочие затраты определяются от суммы прямых общих затрат в установленном размере. Для разработки программного обеспечения, представленного в данной работе, дополнительные прочие затраты составляет 10% от суммы прямых общих затрат. Рассчитаем общие прямые затраты по формуле (8.6).

$$З_{прям} = З_{эл} + Ф_{зп} + А_{нир}, \quad (8.6)$$

Следовательно, $З_{прям}$ = 583,6 + 210000,0 + 1054,7 = 211638,3 руб. Таким образом, $З_{прдоп}$ = 211638,3 * 0,1 = 21163,8 руб., $З_{пр}$ = 63000,0+ 21163,8 =84163,8 руб.

8.5. Расчет сметы затрат

Определим величину общих затрат на разработку программного продукта по формуле 8.7:

$$З = З_{прям} + З_{пр}, \quad (8.7)$$

Тогда 3 = 211638,3 + 84163,8 = 295802,1 руб. Представим смету затрат в таблице 8.1.

Таблица 8.1 – *Смета затрат*

Наименование статьи затрат	Сумма, руб.	Удельный вес, %
Затраты на электроэнергию	583,6	0,2
Затраты на заработную плату	210000,0	71,0
Затраты на амортизацию оборудования	1054,7	0,4
Прочие затраты	84163,8	28,4
Итого	295802,1	100,0

Таким образом, большая часть затрат – 71,0%, необходимых на разработку программного обеспечения, составляют затраты на заработную плату исполнителей. Прочие затраты составляют 28,4% от общей суммы затрат на разработку программного обеспечения. Наименьшая сумма затрат приходится на амортизацию оборудования и электроэнергию (0,4% и 0,2% соответственно).

Заключение

Целью данной работы являлась разработка автоматизации ключевых бизнес процессов городской больницы. В ходе работы детально изучена деятельность городской больницы, в результате чего были определены и спроектированы ключевые бизнес процессы в модели As –Is при помощи нотации UML AD. Далее, в результате анализа данной модели была спроектирована деятельность городской больницы в модели To-Be, где было отображено решение по оптимизации текущих процессов.

Исходя из этого, спроектировано, разработано и протестировано программное обеспечение, позволяющее оптимизировать ключевые бизнес процессы городской больницы. В процессе разработки были изучены и применены на практике основные этапы по внедрению и применению методологии разработки программного обеспечения – Agile Kanban.

Разработка велась на основе пользовательских требований, собранных на этапе анализа требований, вследствие чего был составлен бэклог продукта, отображающий пользовательские и соответствующие им функциональные требования, а так же их приоритет.

Основным достоинством разработанного программного обеспечения является возможность автоматизировать те операции, которые могут быть выполнены без участия медицинского персонала с целью более эффективного распределения времени при приеме пациента, а так же исключения вероятности возникновения ошибок, связанных с человеческим фактором.

Так, в частности, разработанная программа способна помочь лечащему врачу с выбором необходимой диагностики пациенту при возникновении каких-либо затруднений. Так же, важными являются функция автоматической расшифровки результатов диагностики, способная указать отклоняющиеся от нормы показатели и функция автоматического подбора курса терапии,

позволяющая назначить курс терапии с необходимыми дозировками лекарственных препаратов.

Резюмируя вышеизложенное, можно сделать вывод, что разработанное программное обеспечение соответствует требованиям современной медицины и должно облегчить и улучшить работу в целом лечебного учреждения, медицинского персонала и сотрудников больницы, с целью повышения качества предоставляемых ими медицинских услуг.

Список использованных литературных источников

1. Agile-манифест разработки программного обеспечения, статья «Agile манифест»// [Интернет–ресурс]. Режим доступа: http://agilemanifesto.org (Дата обращения 21.03.2019)
2. Андерсон Д., Кармайкл Э. Канбан: краткое руководство [Текст] / Д.Андерсон, Э. Кармайкл – LeanKanban University – 2015. – 79с.
3. Грин Д. Постигая Agile [Текст] / Грин Д. – Манн, Иванов и Фербер – 2016.–288 с.
4. Свободная энциклопедия Википедия, статья «Канбан (разработка)»// [Интернет–ресурс]. Режим доступа: https://ru.wikipedia.org (Дата обращения 21.03.2019)
5. Книберг Х., Скарин М. Kanban и Scrum: выжимаем максимум [Текст] / Х. Книберг, М. Скарин – InfoQ.com – 2010. – 76с.
6. Свободная энциклопедия Википедия, статья «Анализ требований» // [Интернет–ресурс]. Режим доступа: https://ru.wikipedia.org/wiki/Анализ_требований (Дата обращения 25.03.2019)
7. Вигерс К. Разработка требований к программному обеспечению [Текст] / К.Вигерс – М.: Русская редакция – 2014. — 736 с.
8. Химонин Ю. И. Сбор и анализ требований к программному продукту (Версия 1.03). – 2009, – 51 с.
9. Agile in IT, статья «Методы приоритизации задач»// [Интернет-ресурс]. Режим доступа: https://doitsmartly.ru (Дата обращения 27.03.2019)
10. Воронков, А.Н. Словарь по менеджменту [Текст]: учебное пособие/ А.Н. Воронков, Т.В. Колосова; Нижегород. гос. архит.-строит. ун-т. – Н. Новгород: ННГАСУ, 2013 – 125 с.

11. Статья «Описание бизнес-процессов» // [Интернет-ресурс] режим доступа: http://regcons.ru (Дата обращения 2.04.2019)
12. Онлайн библиотека Studbooks.net, статья «Модель AS-IS и Модель TO-BE» // [Интернет–ресурс]. Режим доступа: https://studbooks.net (Дата обращения 12.04.2019)
13. Ларина, Ю. А. Основы объектно ориентированного моделирования с использованием языка UML: учеб. пособие / Ю. А. Ларина; Яросл. гос. ун-т им. П. Г. Демидова. – Ярославль: ЯрГУ, 2010 – 151 с.
14. Буч Г., Якобсон А., Рамбо Дж. UML. Классика CS [Текст] / Г. Буч, А. Якобсон, Дж. Рамбо – СПб.: Питер – 2006. – 736с.
15. Официальный сайт компании Microsoft , статья «Описание основных приемов нормализации базы данных» // [Интернет–ресурс].Режим доступа: https://support.microsoft.com/ru-ru/help/283878/description-of-the-database-normalization-basics (Дата обращения 19.04.2019)
16. Официальный сайт компании Microsoft, статья «Создание связей между таблицами в базе данных» // [Интернет – ресурс]. Режим доступа: https://support.microsoft.com/ru-ru/help/304466/how-to-define-relationships-between-tables-in-an-access-database (Дата обращения 19.04.2019)
17. Блэк Р. Ключевые процессы тестирования [Текст] / Р. Блэк – Лори– 2008г. – 566с.
18. Куликов С.С. Тестирование программного обеспечения. Базовый курс. [Текст] / С.С. Куликов — Минск: Четыре четверти – 2017 — 312 с.

ПРИЛОЖЕНИЕ А

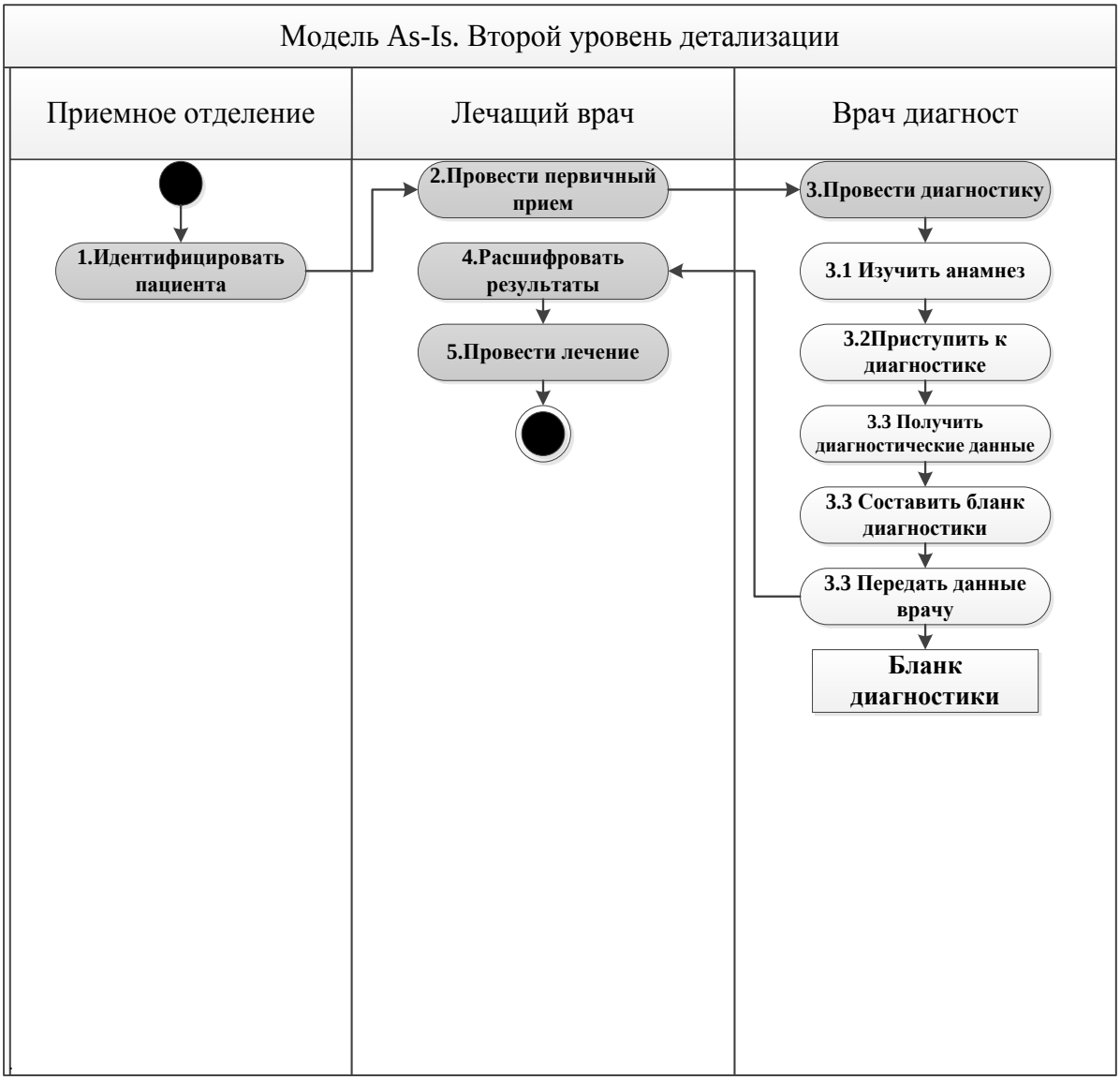


Рисунок 1.1 – *Описание процесса «Провести диагностику»*

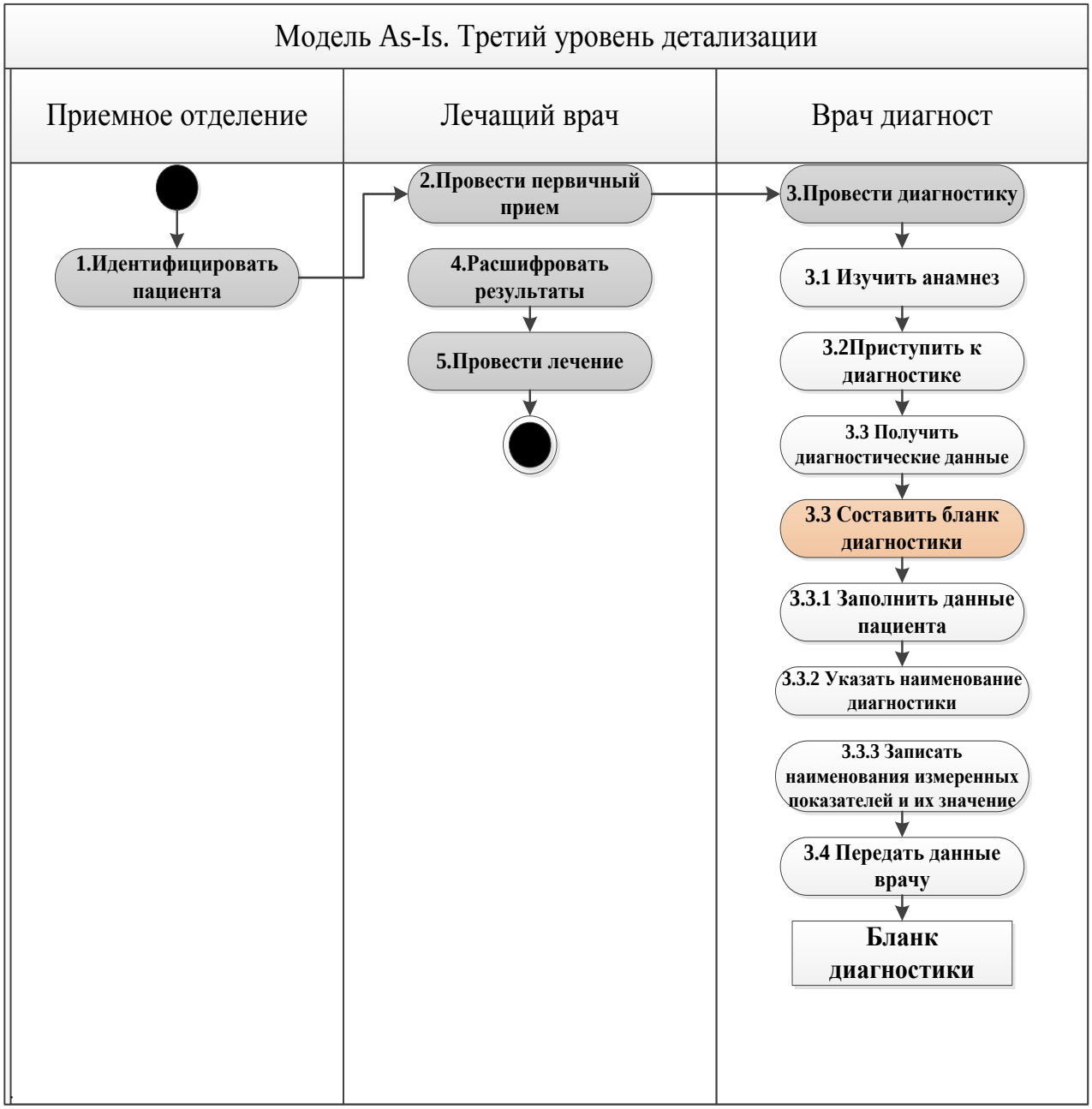


Рисунок 1.2 – *Описание процесса «Провести диагностику», детализация процесса «Составить бланк диагностики»*

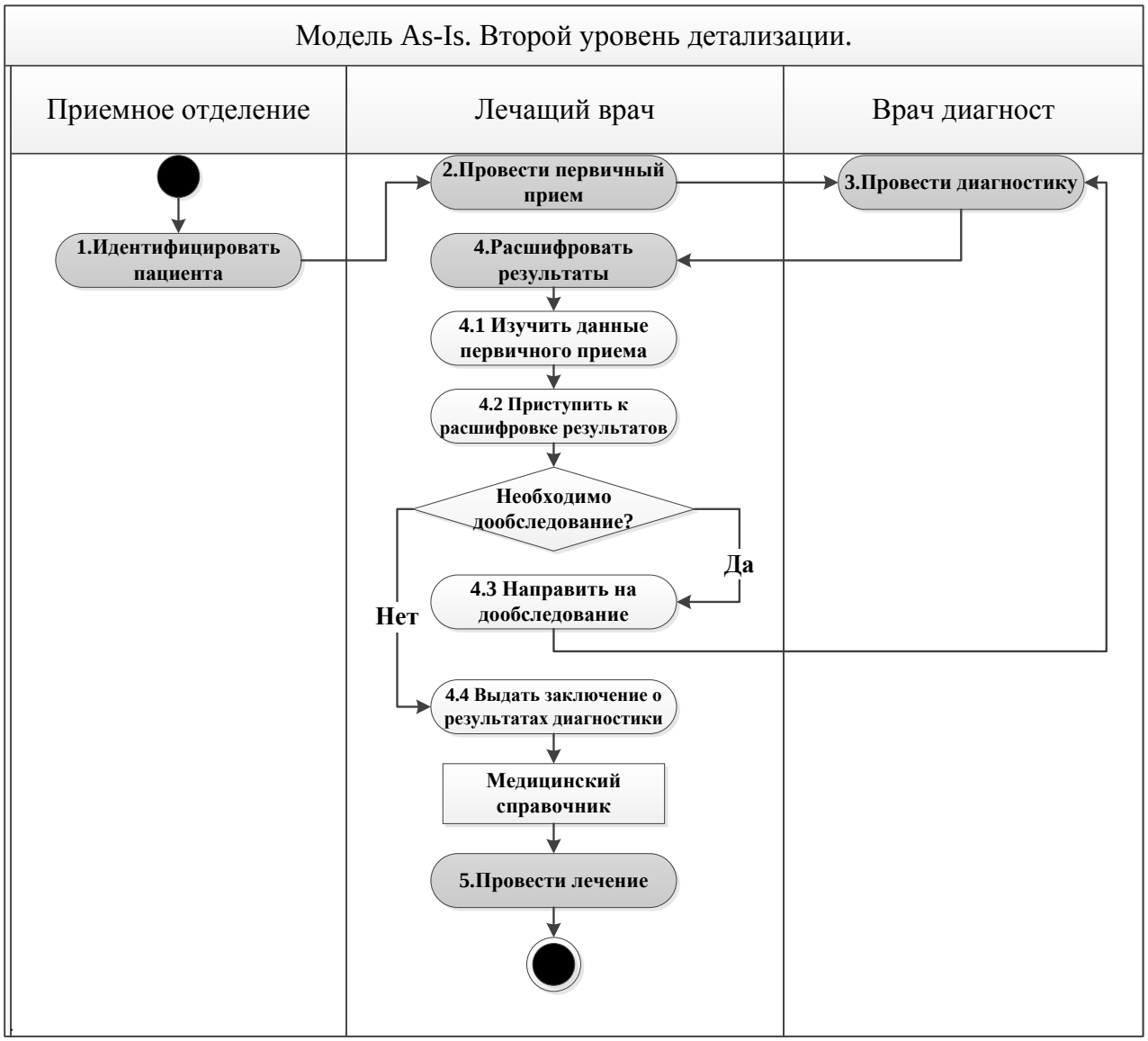


Рисунок 1.3 – *Описание процесса «Расшифровать результаты»*

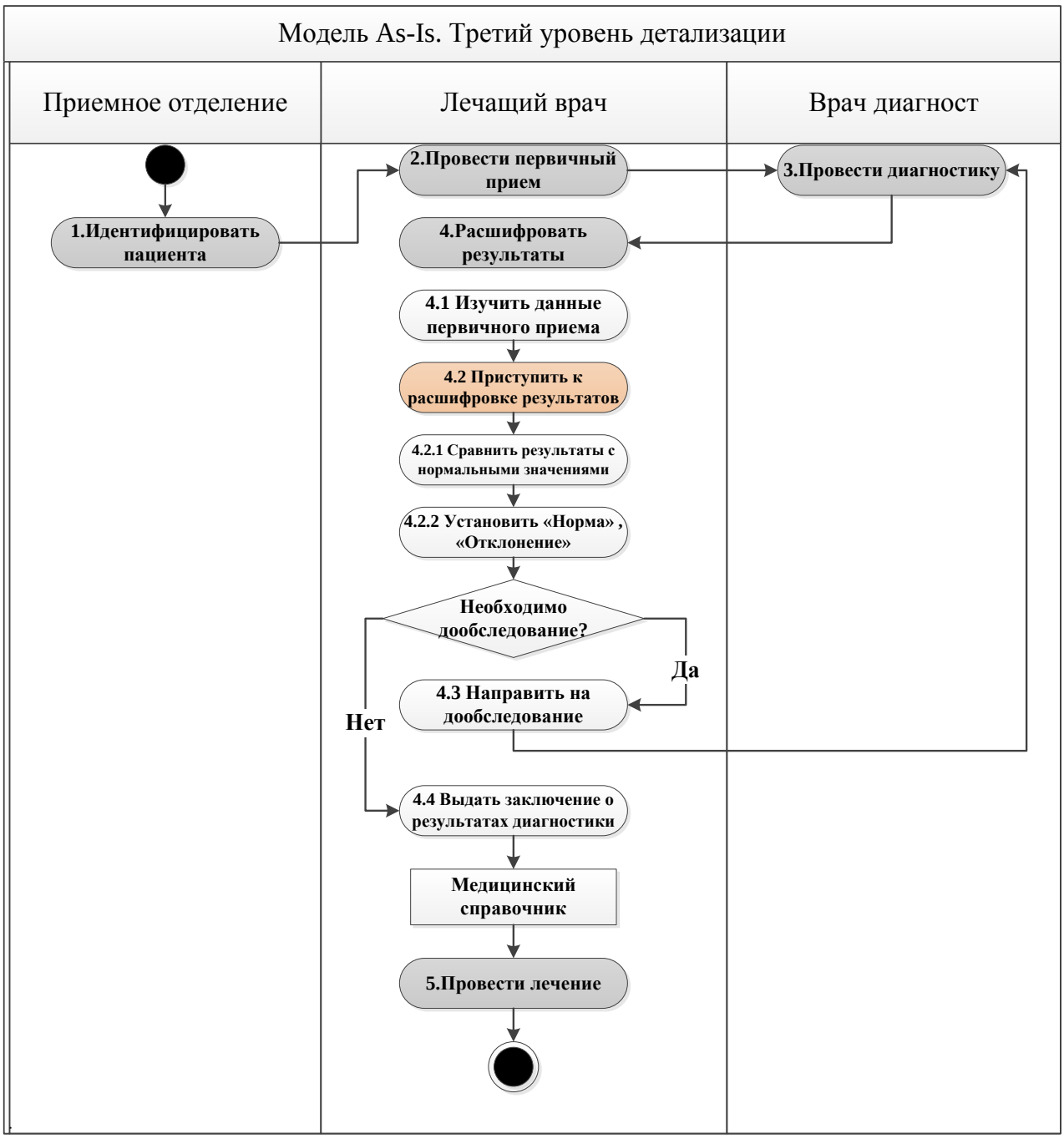


Рисунок 1.4 – *Описание процесса «Расшифровать результаты», детализация процесса «Приступить к расшифровке результатов»*

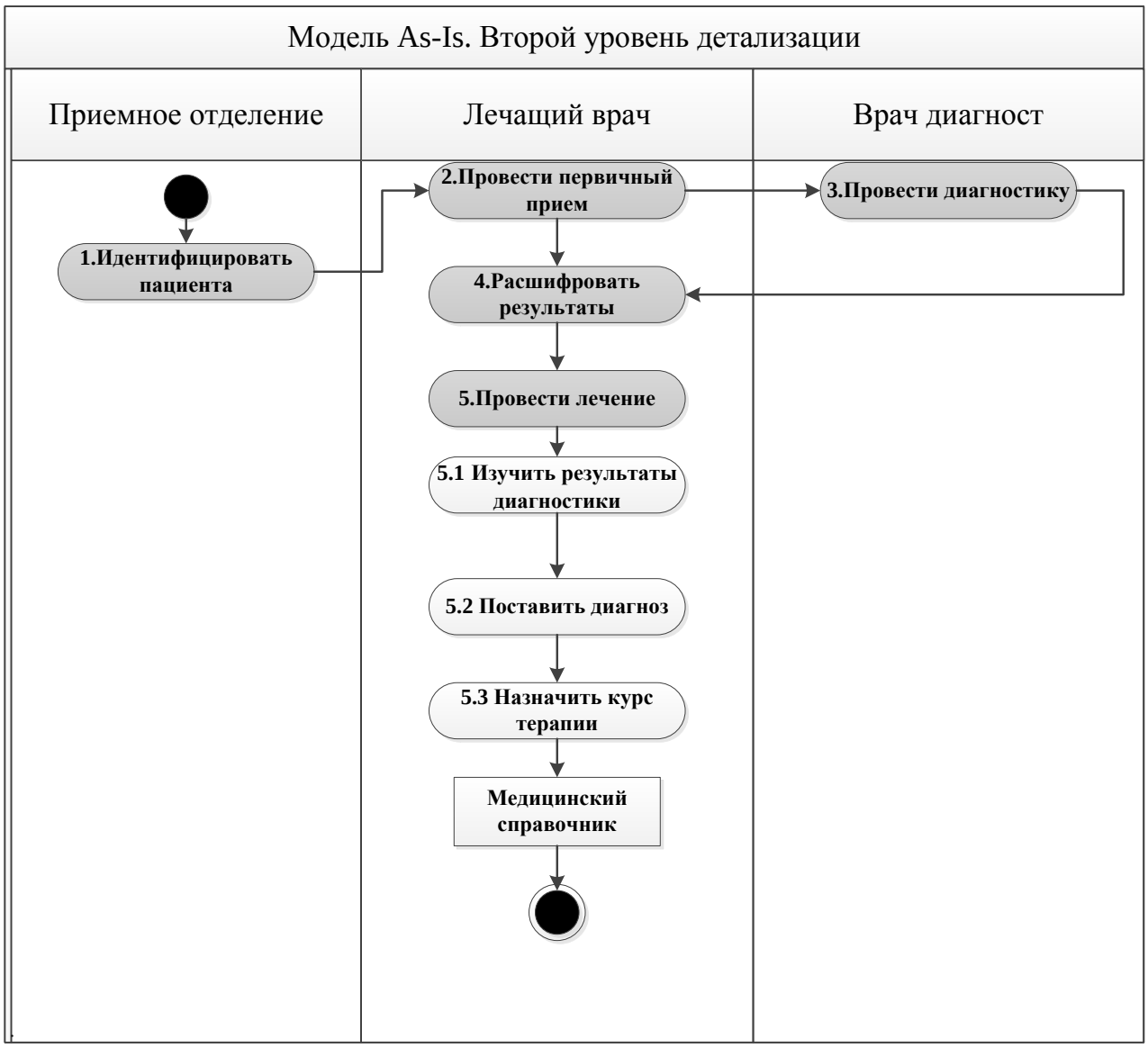


Рисунок 1.5– *Описание процесса «Провести лечение»*

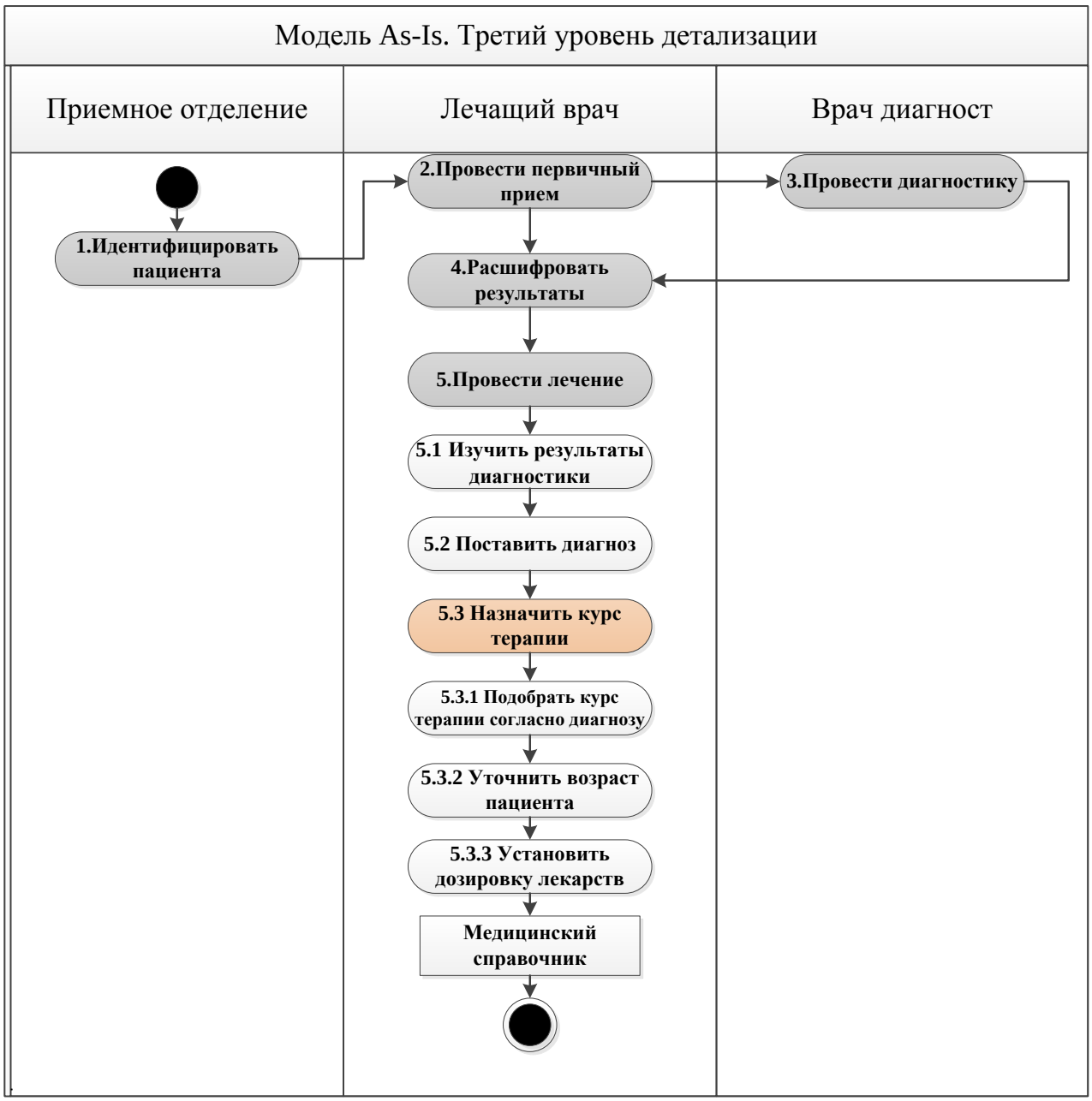


Рисунок 1.6 – *Описание процесса «Провести лечение», детализация процесса «Назначить курс терапии»*

ПРИЛОЖЕНИЕ Б

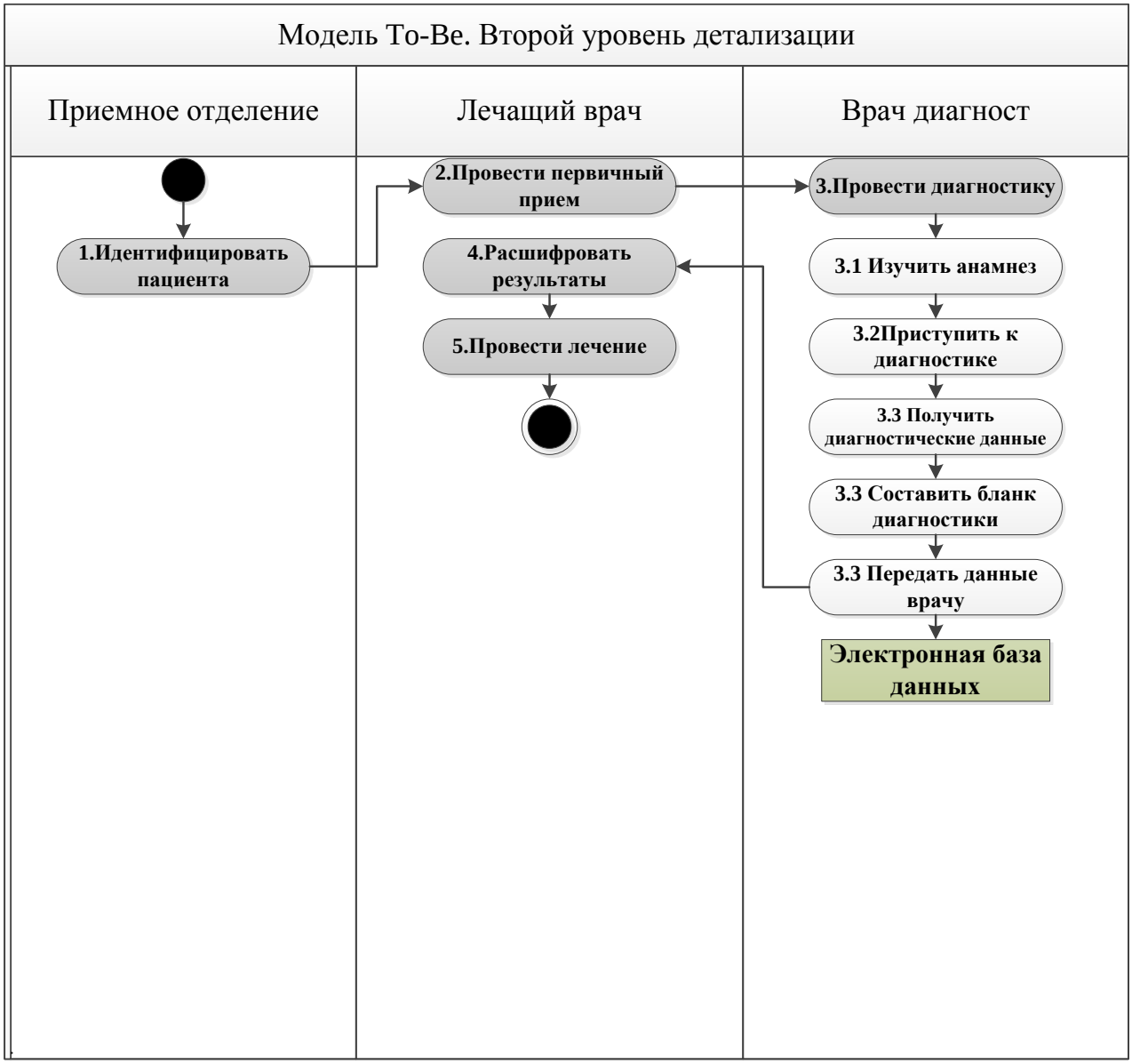


Рисунок 2.1 – *Описание процесса «Провести диагностику»*

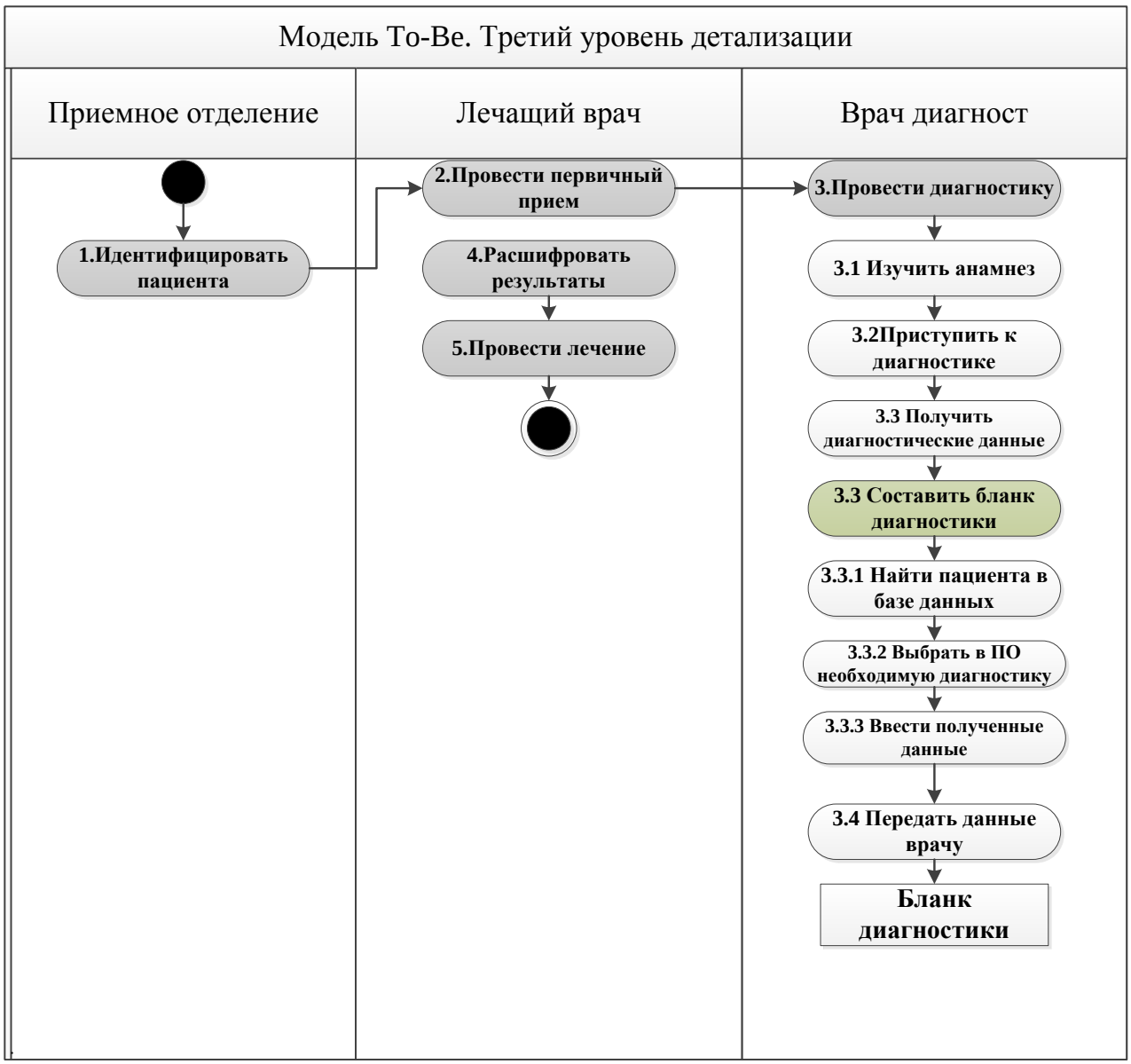


Рисунок 2.2 – *Описание процесса «Провести диагностику», детализация процесса «Составить бланк диагностики»*

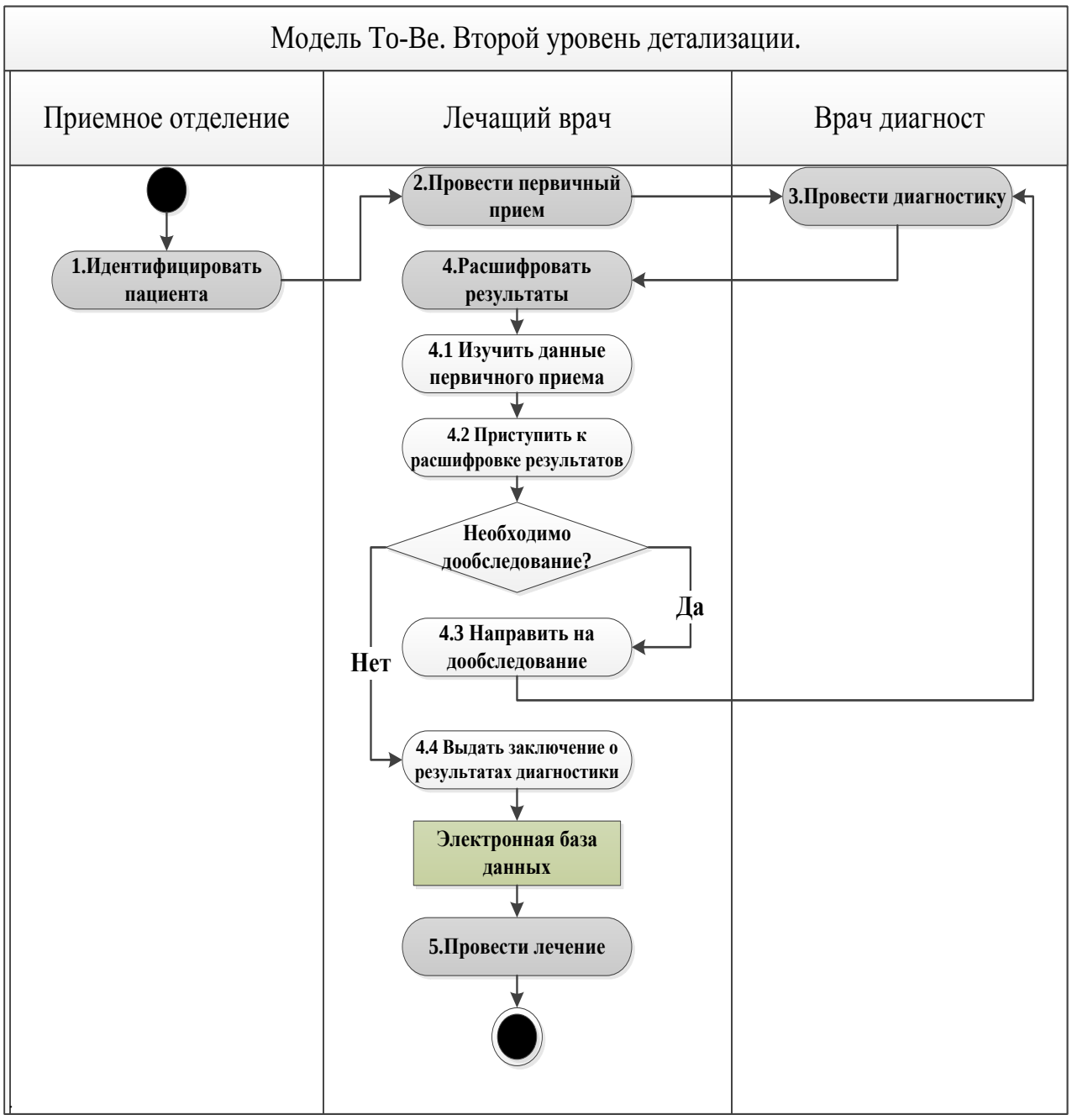


Рисунок 2.3 – *Описание процесса «Расшифровать результаты»*

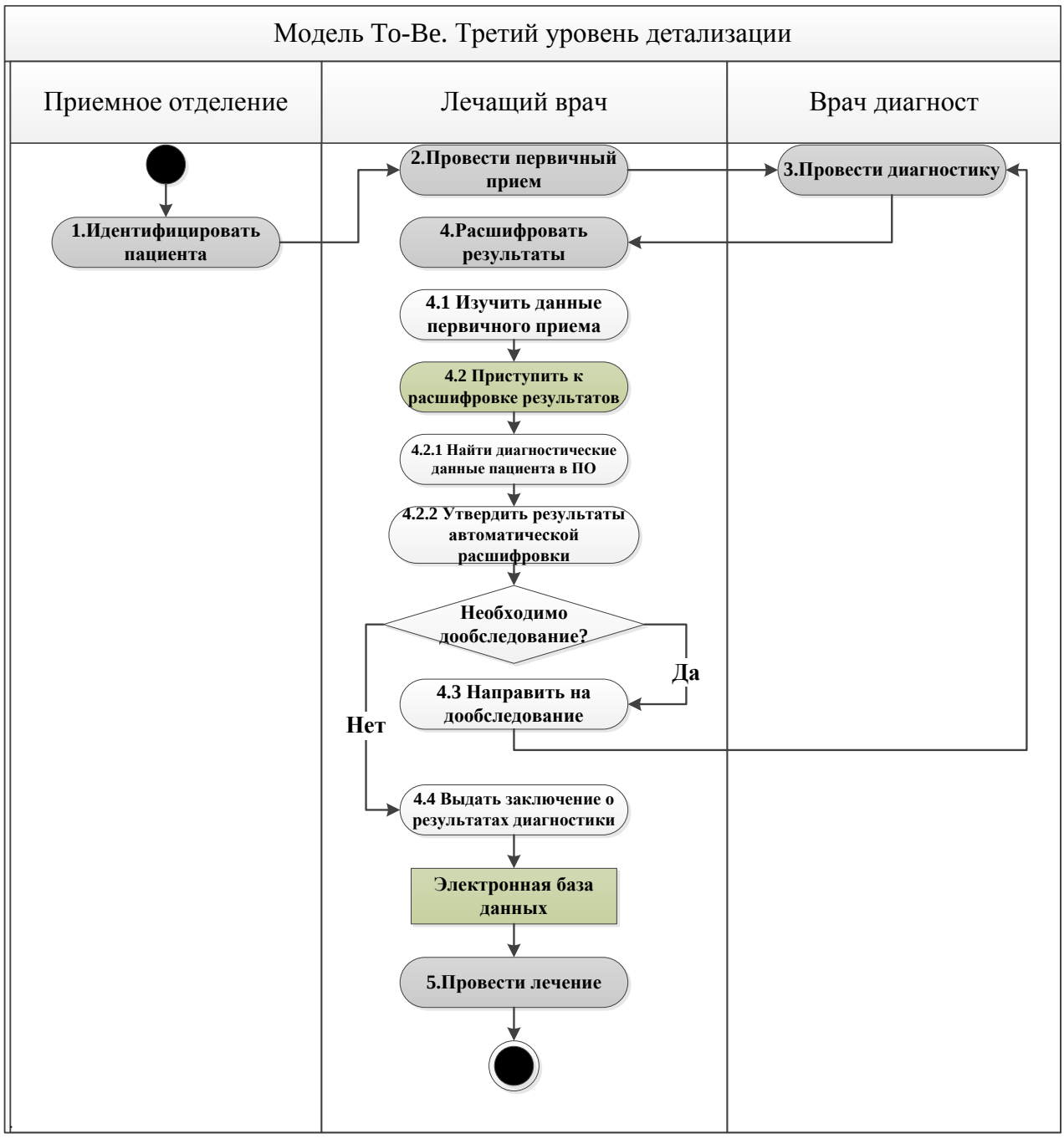


Рисунок 2.4 – *Описание процесса расшифровать результаты, детализация процесса «Приступить к расшифровке результатов»*

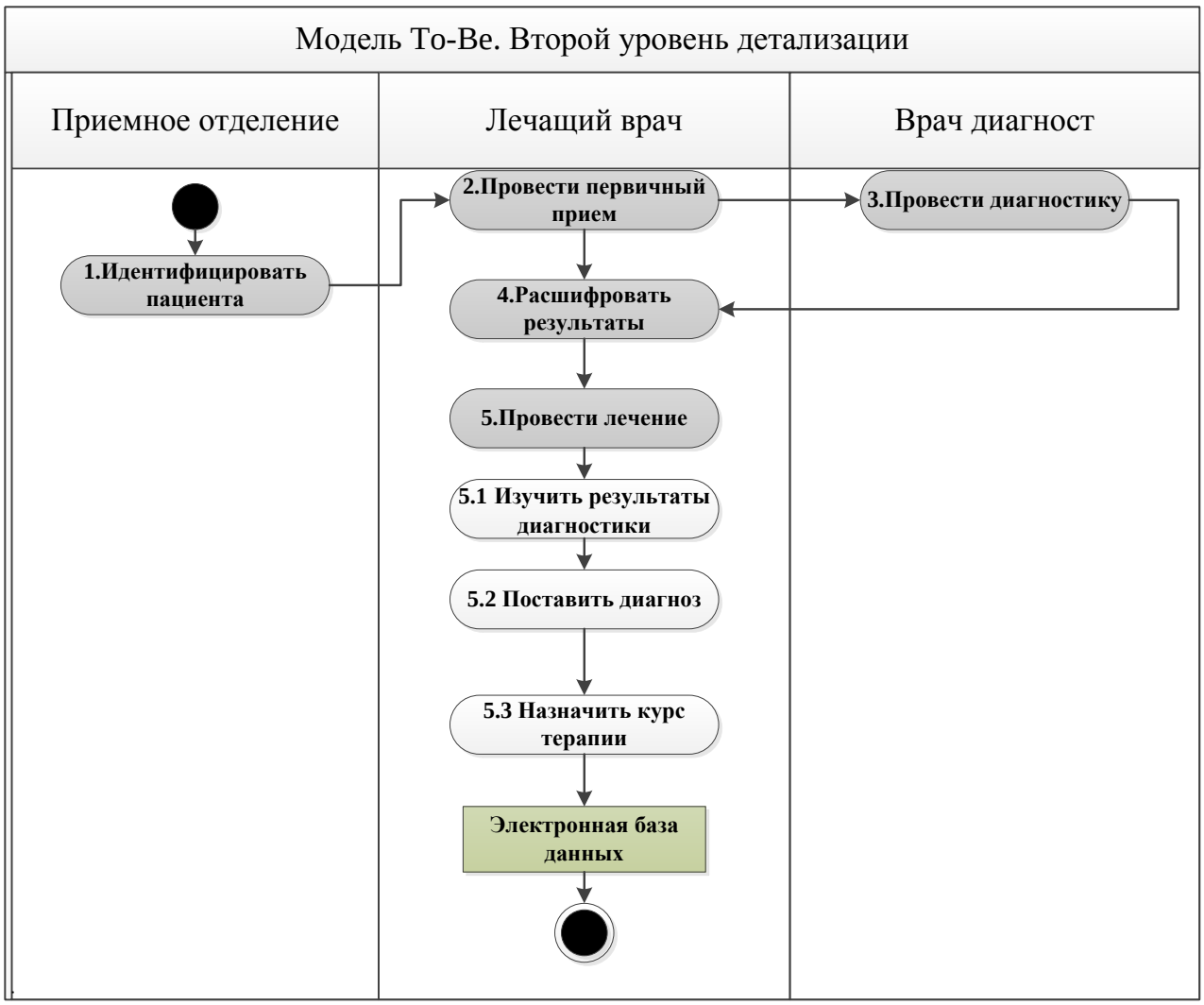


Рисунок 2.5 – *Описание процесса «Провести лечение»*

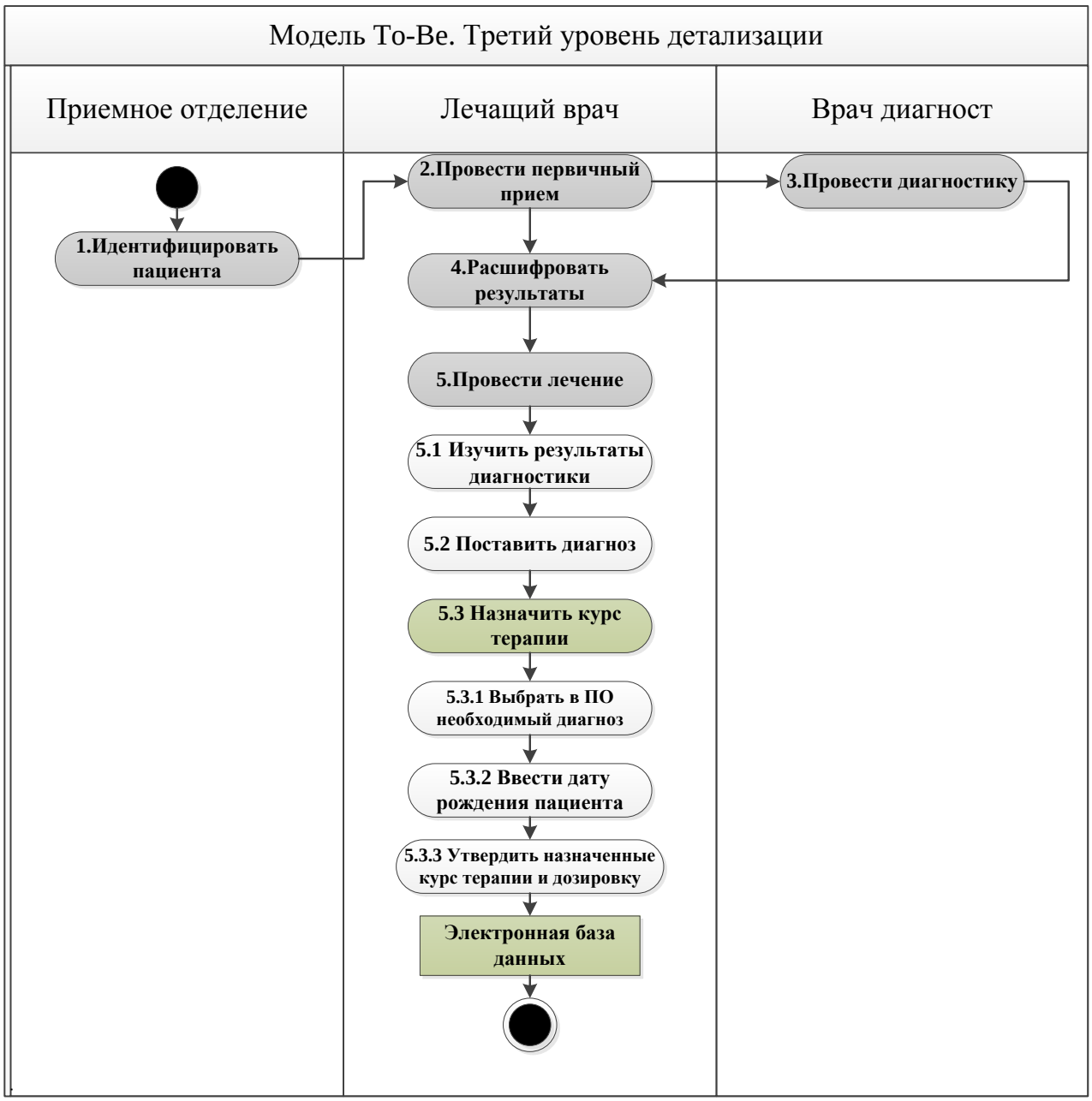


Рисунок 2.6 – *Описание процесса «Провести лечение», детализация процесса «Назначить курс терапии»*