



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 СПИРАЛЕВИДНОЙ МОДЕЛИ ВНЕДРЕНИЯ  
ИНФОРМАЦИОННЫХ СИСТЕМ В ГОРОДСКОЙ ПОЛИКЛИНИКЕ»**

Студент:

Гудков Е.А.

Научный руководитель:

к.т.н., доц. МИРЭА Степанов Д.Ю.

Москва – 2018

## Оглавление

<b>Введение .....</b>	<b>3</b>
<b>Раздел 1. Спиралевидная модель внедрения программных продуктов .....</b>	<b>5</b>
1.1. Понятие жизненного цикла и его модели .....	5
1.2. Спиралевидная модель и ее особенности .....	5
<b>Раздел 2. Идентификация требований .....</b>	<b>9</b>
2.1. Этапы сбора и анализа требований.....	9
2.2. Результаты сбора и анализа требований .....	10
2.3. Список требований .....	14
<b>Раздел 3. Проектирование ключевых бизнес-процессов.....</b>	<b>16</b>
3.1. Способы проектирования .....	18
3.2. Проектирование процессов в AS-IS с помощью IDEF0 и DFD .....	19
3.3. Проектирование процессов в TO-BE с помощью IDEF0 и DFD .....	22
<b>Раздел 4. Разработка и тестирование .....</b>	<b>26</b>
4.1. Задание циклов разработки по спиралевидной модели.....	26
4.2. 1-й прототип программы (I виток спирали).....	27
4.3. 2-й прототип программы (II виток спирали) .....	28
4.4. 3-й прототип программы (III виток спирали).....	32
4.5. Конечный продукт (IV виток спирали) .....	34
4.6. Тестирование программы .....	36
<b>Заключение.....</b>	<b>39</b>
<b>Список использованных литературных источников .....</b>	<b>41</b>

## Введение

В настоящее время многие поликлиники испытывают затруднения при обработке информации многочисленных пациентов. Причинами этого являются:

- длительный и трудоемкий процесс поиска необходимых сведений о пациентах в медицинских картах;
- отсутствие или несоблюдение единого формата внесения данных о пациенте в карту, записи на прием, написания протоколов осмотра, выписки направлений или рецептов;
- случаи потерь данных и/или карт в регистратуре или самими пациентами.

Все это приводит к длинным очередям в поликлиниках, трате большими деньгами и времени. Также это усугубляется процессами присоединения одних медицинских учреждений к другим, при которых администрация этих учреждений вынуждена решать вопрос объединения данных и создания единой базы пациентов.

Поэтому будет целесообразно некоторым образом автоматизировать деятельность поликлиник. Этого можно достигнуть за счет разработки приложения с помощью систем управления базами данных (СУБД), например, MS FoxPro. Разрабатываемая программа упростит процесс обработки информации, сделает его куда более быстрым и удобным; приложение совместимо с операционной системой Microsoft Windows, которая используется на большинстве компьютеров медицинских учреждений. Кроме того, с программой могут работать пользователи без специальных знаний, навыков и подготовки.

К настоящему времени разработано множество программных средств для автоматизации медучреждений. Но большинство из них имеют недостатки, среди которых:

- учтены не все аспекты деятельности поликлиник;
- сложность в освоении;
- неудобство в эксплуатации.

Например, в работах [1-2] описана разработка программного средства, реализующего управление информацией о пациенте (добавление записи, редактирование, удаление, поиск) и процессы записи пациента на прием. Однако, там не рассмотрены аспекты автоматизации деятельности врача, такие как, например, запись в медицинскую карту и возможность просмотра данных пациента. Автоматизация этих процессов позволила бы в значительной степени повысить производительность врача, и, как следствие – уменьшить время, проводимое пациентами в очередях.

Таким образом, можно сделать вывод о целесообразности проведения данной работы. Целью данной работы является автоматизация деятельности поликлиники с помощью приложения, разрабатываемого по спиралевидной модели. В процессе должны быть реализованы следующие задачи:

- анализ требований, предъявляемых пользователями;
- проектирование структуры приложения;
- разработка приложения в среде MS FoxPro;
- тестирование разработанной программы.

## Раздел 1. Спиралевидная модель внедрения программных продуктов

### *1.1. Понятие жизненного цикла и его модели*

Информационные системы (ИС) должны удовлетворять интересам бизнеса, а также быть легко модифицируемыми и недорогими. Плохо спроектированная система, в конечном счете, требует больших затрат и времени для ее содержания и обновления. Одним из базовых понятий методологии проектирования ИС является понятие жизненного цикла ее программного обеспечения (ЖЦ ПО). Жизненный цикл (Life Cycle) определяется как развитие системы, продукта, услуги, проекта или других изготовленных человеком объектов, начиная со стадии разработки концепции и заканчивая прекращением применения. Модель жизненного цикла (Life Cycle Model) – структура процессов и действий, связанных с жизненным циклом, организуемых в стадии, которые также служат в качестве общей ссылки для установления связей и взаимопонимания сторон [3].

### *1.2. Спиралевидная модель и ее особенности*

Существует множество различных моделей жизненного цикла ПО, одна из них – спиральная модель. В ней делается упор на начальные этапы ЖЦ системы: анализ и проектирование. Спиральная модель – классический пример применения эволюционной стратегии разработки [4]. Особенность данной модели заключается в том, что прикладное программное обеспечение создается не сразу, а по частям (модулям) с использованием метода прототипирования. В данной модели прототип – действующее программное обеспечение, реализующее отдельные функции и внешний интерфейс пользователя [5]. Как показано на рис. 1.1, модель определяет четыре действия, каждое из которых соответствует своему квадранту спирали:

- подготовка – сбор требований и ограничений;

- планирование – формирование плана проекта и анализ рисков;
- моделирование и конструирование (разработка) – подготовка моделей и реализация продукта следующего уровня;
- развертывание – оценка заказчиком текущей версии продукта.



**Рисунок 1.1 – Спиралевидная модель:**

*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*

Разработка здесь отображается движением по разворачивающейся спирали (по часовой стрелке), причем старт проекта происходит в первом квадранте. Интегрирующий аспект спиральной модели очевиден при учете радиального измерения спирали. С каждым витком по спирали (продвижением от центра к периферии) строятся все более полные версии ПО [4].

Спиральная модель позволяет начинать работу над следующим этапом, не дожидаясь завершения предыдущего. При необходимости на каждом цикле

корректируются требования к разрабатываемому продукту. Основная проблема спирального цикла – определение момента перехода на следующий этап. Возможным её решением является принудительное ограничение по времени для каждого этапа жизненного цикла [6].

Отличительной особенностью этой модели является специальное внимание рискам, влияющим на организацию жизненного цикла. Барри Боэм – автор модели – в своей работе [7] формулирует десять наиболее распространённых (по приоритетам) рисков:

- 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.

Достоинства спиральной модели:

- позволяет быстрее показать пользователям системы работоспособный продукт, тем самым, активизируя процесс уточнения и дополнения требований;

- допускает изменение требований при разработке информационной системы, что характерно для большинства разработок, в том числе и типовых;
- обеспечивает большую гибкость в управлении проектом;
- позволяет получить более надежную и устойчивую систему. По мере развития системы ошибки и слабые места обнаруживаются и исправляются на каждой итерации;
- позволяет совершенствовать процесс разработки – анализ, проводимый в каждой итерации, позволяет проводить оценку того, что должно быть изменено в организации разработки, и улучшить ее на следующей итерации;
- уменьшаются риски заказчика. Заказчик может с минимальными для себя финансовыми потерями завершить развитие неперспективного проекта.

Недостатки модели:

- увеличивается неопределенность у разработчика в перспективах развития проекта. Этот недостаток вытекает из предыдущего достоинства модели;
- затруднены операции временного и ресурсного планирования всего проекта в целом. Для решения этой проблемы необходимо ввести временные ограничения на каждую из стадий жизненного цикла. Переход осуществляется в соответствии с планом, даже если не вся запланированная работа выполнена. План составляется на основе статистических данных, полученных в предыдущих проектах и личного опыта разработчиков [8].



## Раздел 2. Идентификация требований

### *2.1. Этапы сбора и анализа требований*

Весь процесс работы с требованиями к разрабатываемому продукту можно разделить на четыре этапа:

- определение концепции продукта;
- сбор требований;
- анализ требований;
- проектирование системы.

На этапе определения концепции продукта проводится работа с его инвестором, целью которой является выработка единого видения будущего продукта. По окончании этого этапа производится вывод о том, будет этот продукт разрабатываться или нет.

На этапе сбора требований основная работа ведется с заказчиком системы и ее будущими пользователями. Цель этапа — точно определить функции продукта и способы его интеграции в существующие процессы. Качественное выполнение работ на этом этапе гарантирует то, что будущий продукт будет соответствовать ожиданиям заказчика. Четкая расстановка приоритетов обеспечивает реализацию наиболее востребованной функциональности и исключение второстепенной/невостребованной функциональности, что экономит бюджет и сроки [9].

Сбор таких требований может осуществляться посредством применения специальных запросов, анкетирования, интервью или при помощи налаживания постоянных процедур обратной связи на основе заранее созданных систем коммуникации. Сбор требований заинтересованных сторон предполагает также изучение их представления в средствах массовой информации, проведение анализа сложившихся социальных и экологических стандартов, рыночной конъюнктуры на товарном и финансовом рынках, на рынке труда [10].

На этапе анализа требований проходит структуризация уже собранных ранее требований. Цель этапа — предоставить четкий список недублируемых требований к системе, которые должны быть выделены из избыточных и частично дублируемых сценариев и пользовательских историй, которые были получены на предыдущем этапе. Правильно сгруппированные требования помогут обойтись минимальным количеством функционала для удовлетворения максимально большего количества целей, а это, в свою очередь, поможет сэкономить бюджет и не даст расползтись рамкам проекта.

Целью всех предыдущих этапов был сбор информации о том, кому и зачем необходим будущий продукт. Этап проектирования — это первый этап, на котором группа разработки принимает проектные решения о том, какую функциональность будет нести продукт, чтобы удовлетворить пользователей. Результатом этого этапа является законченное техническое задание к продукту. Оно должно содержать полное описание поведения будущего продукта и не содержать неоднозначностей и вопросов. На основе технического задания начинается моделирование работы продукта с конечными пользователями (используя макеты пользовательского интерфейса, к примеру) и производится тестирование технического задания. Это позволяет увеличить качество продукта и снизить его стоимость, так как стоимость внесения изменений в техническое задание всегда меньше, чем в конечный продукт [9].

## ***2.2. Результаты сбора и анализа требований***

В процессе сбора данных методами опроса и изучения литературы была получена и проанализирована информация о продукте, который предстоит разработать. Пользовательские требования – это требования, описывающие задачи, которые должны выполняться пользователями при помощи разрабатываемого ПО. На этапе сбора требований были определены следующие пункты:

- Хранение данных различных категорий:
  - пациенты;
  - врачи (персонал);
  - запись на прием;
  - анамнез пациентов.
- Возможности работы с вышеуказанными данными, включающие:
  - создание (добавление, регистрация, запись) – для всех категорий;
  - редактирование (изменение) – пациенты, персонал и протокол осмотра;
  - удаление (отмена записи) – для всех категорий;
  - отбор по ключевым параметрам (поиск) – для всех категорий;
  - сортировка по возрастанию/убыванию (в прямом/обратном алфавитном порядке) параметров – для всех категорий.
- Возможность авторизации пользователей в системе с помощью логина и пароля, хранящихся в БД.
- Разграничение доступа к данным в зависимости от исполняемой должности:
  - только работники регистратуры могут редактировать и удалять данные о пациентах, а также выписывать талон на прием;
  - только врачи могут работать с записями в электронной карте (анамнез). При этом полный доступ к записи (редактирование и удаление) получает только автор записи; остальные могут только просматривать записи без возможности изменения;
  - только главный врач (администратор) может работать с записями о сотрудниках в базе данных.
- Возможность отображения записей на прием в текущий рабочий день для каждого конкретного врача.

- Возможность просмотра врачом полной информации о пациенте, записанном на прием.

ПО будет разрабатываться с помощью среды MS FoxPro, т.к. возможности этой программы позволяют реализовать все необходимые требования достаточно быстро и полно, без потери функциональности. Приоритезация требований позволяет понять, какие требования пользователя следует реализовать в первую очередь и на что следует обратить особое внимание; как следствие, это позволяет избежать излишних материальных и временных затрат на проектирование и разработку модулей или функционала, которые не будут играть важной роли в созданном продукте. Существует несколько техник приоритезации требований, одна из них – MuSCoW, широко используемая в управлении, бизнес-аналитике, а также в разработке ПО. Суть данного метода заключается в разделении всех требований на четыре категории:

- **Must have** (обязательные) – требования с наибольшим приоритетом, без выполнения которых релиз продукта невозможен;
- **Should have** (желательные) – высокоприоритетные требования, которые критичны для функционала;
- **Could have** (возможные) – требования, которые можно включить в текущий релиз, но которые не влияют существенно на его успех;
- **Won't have** (отсутствующие) – требования, которые не являются необходимыми в текущем релизе, но которые можно включить в следующие [9].

Используя метод MuSCoW, определим приоритеты полученных пользовательских требований:

- Требование 1 – Must have. Это требование является основой всей разрабатываемой программы, и без его выполнения нельзя говорить о релизе продукта.
- Требование 2 – Should have. В среде MS FoxPro возможна работа с записями (создание, редактирование, удаление, поиск, сортировка) посредством панели управления самой программы, т.е. нет прямой необходимости реализации данного требования. Вместе с тем, его реализация значительно упрощает интерфейс, вследствие чего работать с программой может любой пользователь без специальной подготовки.
- Требования 3-6 – Could have. Требования не являются ключевыми для полноценного функционирования самой программы, но их было бы неплохо включить в текущий релиз.

Функциональные требования – это требования, объясняющие, что должно быть в разрабатываемом ПО, а также какие действия должны выполняться системой. Значит, пользовательские требования будут иметь следующий вид:

- Таблицы, содержащие данные различных категорий:
  - данные о пациентах – таблица «Пациенты»;
  - данные о врачах и работниках регистратуры – таблица «Персонал»;
  - данные о записи на прием – таблица «Запись на прием»;
  - анамнез пациентов – таблица «Анамнез»;
- Формы и программные компоненты, обеспечивающие:
  - создание записей;
  - редактирование записей;
  - просмотр записей без изменения (для таблиц «Пациенты» «Анамнез»);
  - удаление записей;

- отбор записей по ключевым параметрам;
  - сортировка записей по возрастанию/убыванию (в прямом/обратном алфавитном порядке) параметров.
- Форма и программные компоненты, обеспечивающие вход пользователя в систему.
  - Программа для вызова соответствующих форм, в зависимости от исполняемой должности пользователя, вошедшего в систему.
  - Форма и программные компоненты, отображающие запись на текущий день к данному врачу.
  - Форма и программные компоненты для вывода полной информации о пациенте на экран.

### 2.3. Список требований

После сбора и анализа всех 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 табл. 2.1.

**Таблица 2.1 – Список требований**

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет требования согласно MuSCoW
1	Хранение данных о пациенте	Таблица данных «Пациенты», содержащая информацию о пациенте	Программа по ведению данных	Must have (должно быть)
2	Хранение данных о пользователях системы	Таблица данных «Персонал», содержащая информацию о пользователях		
3	Хранение данных о записи на прием к врачу	Таблица данных «Запись на прием»		

№	Пользовательское требование	Функциональное требование	Программный компонент	Приоритет требования согласно MuSCoW
4	Хранение анамнеза пациентов	Таблица данных «Анамнез»		
5	Регистрация пациента	Возможность создания записи в таблице «Пациенты»	Программа для создания новой записи	Should have (желательно, чтобы это было)
6	Регистрация пользователя	Возможность создания записи в таблице «Персонал»		
7	Запись на прием к врачу	Возможность создания записи в таблице «Запись на прием»		
8	Добавление анамнеза	Возможность создания записи в таблице «Анамнез»		
9	Управление данными о пациентах (изменение, удаление, отбор, сортировка)	Возможности редактирования, удаления, поиска и сортировки записей в таблице «Пациенты»	Программы для редактирования, просмотра, удаления и индексирования записей, реализуемые в соответствующих формах	
10	Управление данными о пользователях (изменение, удаление, отбор, сортировка)	Возможности редактирования, удаления, поиска и сортировки записей в таблице «Персонал»		
11	Управление данными о записи на прием (удаление, отбор, сортировка)	Возможности удаления, поиска и сортировки записей в таблице «Запись на прием»		
12	Управление данными об анамнезе пациентов (изменение, просмотр, удаление, отбор, сортировка)	Возможности редактирования, просмотра, удаления, поиска и сортировки записей в таблице «Анамнез»		
13	Авторизация и разграничение доступа	Вход в систему с помощью логина и пароля, ограничение на доступ к данным в зависимости от должности	Программы для авторизации и вызова форм	Could have (может быть)

### Раздел 3. Проектирование ключевых бизнес-процессов

Фундаментальным принципом реинжиниринга (реорганизации) является рассмотрение деятельности компании не с точки зрения функционирования ее структурных подразделений, а с точки зрения организации и протекания в ней бизнес-процессов. Бизнес-процесс – это связанное множество внутренних видов деятельности компании, заканчивающихся созданием продукции или услуги, необходимой потребителю. Термин “потребитель” следует понимать в широком смысле – это может быть просто клиент, а может быть и другой процесс, протекающий во внешнем окружении, например, у партнеров или субподрядчиков [11].

Бизнес-модель – это описание бизнес-процессов, характеризующее их основные элементы и связи между этими элементами, позволяющее создать упрощенное целостное представление о бизнес-процессе и отразить его наиболее существенные характеристики [12]. Моделирование бизнес-процессов является значимой составной частью проектов по реинжинирингу бизнес-процессов и внедрению крупномасштабных систем ПО. Отсутствие таких моделей является одной из главных причин неудач многих проектов.

Ориентирование на процессы является основным фактором успешного реинжиниринга. Другим, не менее важным фактором, является переход предприятия на использование новых информационных технологий. Применение новых информационных технологий может привести не только к принципиальным изменениям в деятельности сотрудников, но и к полной замене существующих бизнес-процессов [11].

На начальных этапах создания СУБД необходимо понять, как работает организация, которую собираются автоматизировать. Руководитель хорошо знает работу в целом, но не в состоянии вникнуть в детали работы каждого



рядового сотрудника. Рядовой сотрудник хорошо знает, что творится на его рабочем месте, но может не знать, как работают коллеги. Поэтому для описания работы предприятия необходимо построить модель, которая будет адекватна предметной области и содержать в себе знания всех участников бизнес - процессов организации [13].

При проектировании ключевых бизнес-процессов используют две модели: AS-IS (как есть) и TO-BE (как будет). Модель AS-IS (как есть) основана на совокупности должностных инструкций, отчетов, приказов, нормативной документации и т.д., относящейся к предприятию. Она позволяет выяснить, «что мы делаем сегодня» перед тем, как перейти к тому, «что мы будем делать завтра». Анализ модели позволяет выявить слабые места в структуре организации, в чем будут состоять преимущества новых процессов и насколько глубоким изменениям подвергнется существующая организация деятельности предприятия (компании, отдела). Признаками неэффективной организации деятельности могут быть:

- бесполезные, неуправляемые и дублирующие работы;
- работы без результата;
- неэффективный документооборот (нужный документ не оказывается в нужное время в нужном месте) и т.д.

Найденные в модели недостатки исправляются в модели TO-BE (как будет): модели новой организации работы предприятия. Модель TO-BE нужна для анализа альтернативных путей решения задачи и выбора наилучшего из них.

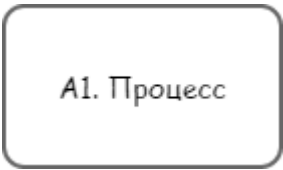
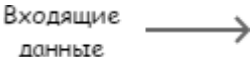
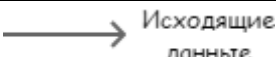


Следует указать на распространенную ошибку при создании модели TO-BE – это создание идеализированной модели. Примером может служить построение модели на основе знаний руководителя, а не конкретного исполнителя работ. Руководитель знает, как должна выполняться работа по

руководствам и должностным инструкциям и часто не имеет представления, как на самом деле его подчиненные выполняют работы. В результате получается приукрашенная, искаженная модель, которая несет ложную информацию и которую невозможно в дальнейшем использовать для анализа. Такая модель называется SHOULD-BE (как должно было быть) [8].

### 3.1. Способы проектирования

Существует несколько способов моделирования бизнес-процессов, один из которых – функциональное моделирование IDEF0. Эта нотация используется для создания функциональной модели, отображающей структуру и функции системы, а также потоки информации и материальных объектов, связывающие эти функции [14]. Элементы, используемые в процессе моделирования с помощью нотации IDEF0, приведены в табл. 3.1 [15].

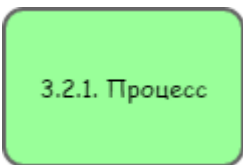
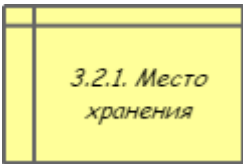
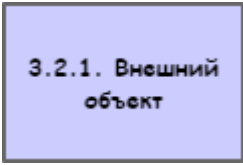

**Таблица 3.1** – Условные обозначения нотации IDEF0

Графический элемент	Описание
	Исходный процесс
	Входящие данные процесса
	Исходящие данные процесса
	Ограничение процесса
	Ресурс процесса

Основным достоинством этой модели является то, что с её помощью можно учесть все аспекты функционирования проекта: как административные, так и организационные.

Однако нотации IDEF0 недостаточно для полноценного проектирования бизнес-процесса; ведь нужно описать также и его нижние уровни, отобразив при этом документооборот и процессы преобразования информации в системе. Для этих целей используют нотацию DFD (Data Flow Diagram – диаграмма потока данных) [16]. Элементы, используемые в процессе моделирования с помощью нотации DFD, приведены в табл. 3.2 [15].

**Таблица 3.2 – Условные обозначения нотации DFD**

Графический элемент	Описание
	Исходный процесс
	Место хранения информации
	Внешний по отношению к системе объект
	Входящие/исходящие данные процесса

Недостатком данной нотации является отсутствие в схеме управляющих элементов. Впрочем, этот недостаток легко устранить, используя для моделирования верхнего уровня нотацию IDEF0.

### **3.2. Проектирование процессов в AS-IS с помощью IDEF0 и DFD**

Проектирование в модели AS-IS позволяет показать функциональную схему медицинского учреждения в том виде, как она выглядит на данный момент. Функционирование поликлиники в представлении IDEF0 на нулевом уровне заключается в выполнении трех бизнес-процессов: A1 –

The diagram illustrates the process of accepting a patient into a medical institution, involving three main activities: A1. Hire employee, A2. Bring medical card, and A3. Accept patient.

**Activities:**

- A1. нанять сотрудника (Hire employee):** Triggered by "Есть необходимость в сотрудниках" (Need for employees) and "Кандидат на должность" (Candidate for position). It produces "Список сотрудников" (List of employees).
- A2. Завести мед. карту (Bring medical card):** Triggered by "Правила регистрации" (Registration rules) and "Паспорт, полис ОМС" (Passport, OMS policy). It produces "Мед. карта" (Medical card).
- A3. Принять пациента (Accept patient):** Triggered by "График работы и занятость врача" (Doctor's work schedule and availability), "Мед. карта" (Medical card), and "Пациент" (Patient). It produces "Мед. карта" (Medical card).

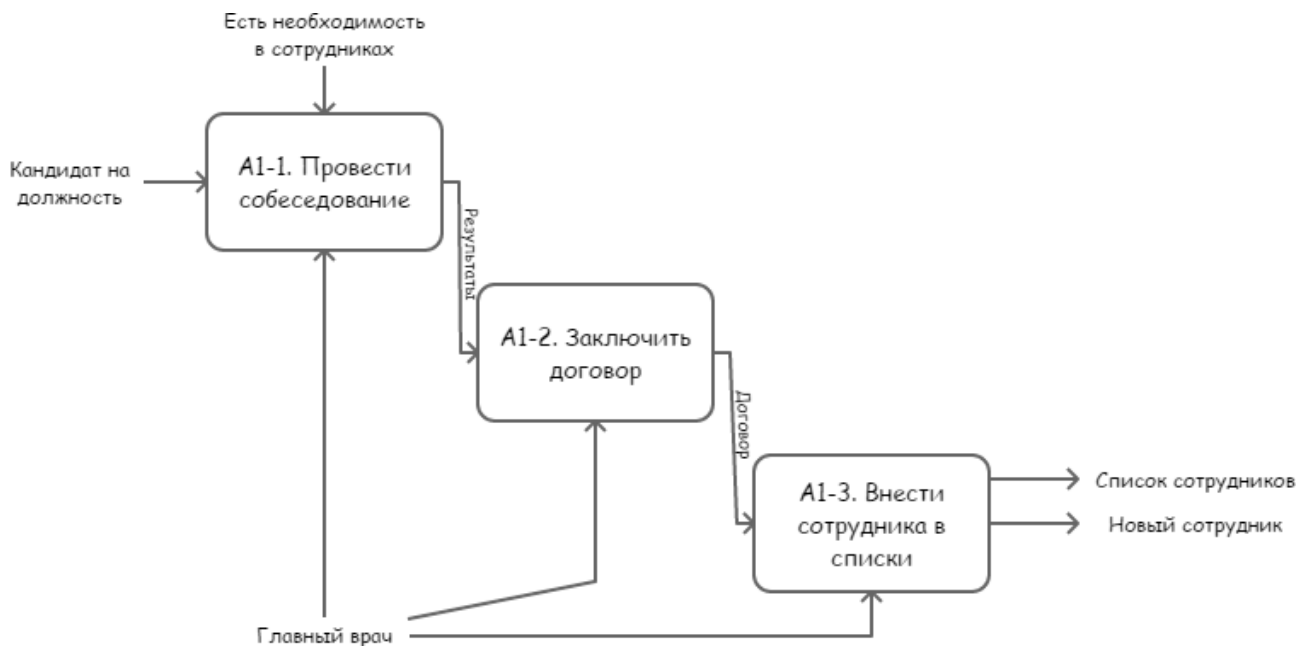
**Participants and their roles:**

- Главный врач (Chief doctor):** Initiates the hiring process and oversees the registration process.
- Регистратор (Registrar):** Assists in the registration process, specifically in bringing the medical card and accepting the patient.
- Врач (Doctor):** Oversees the acceptance of the patient.

**Flow of the process:**

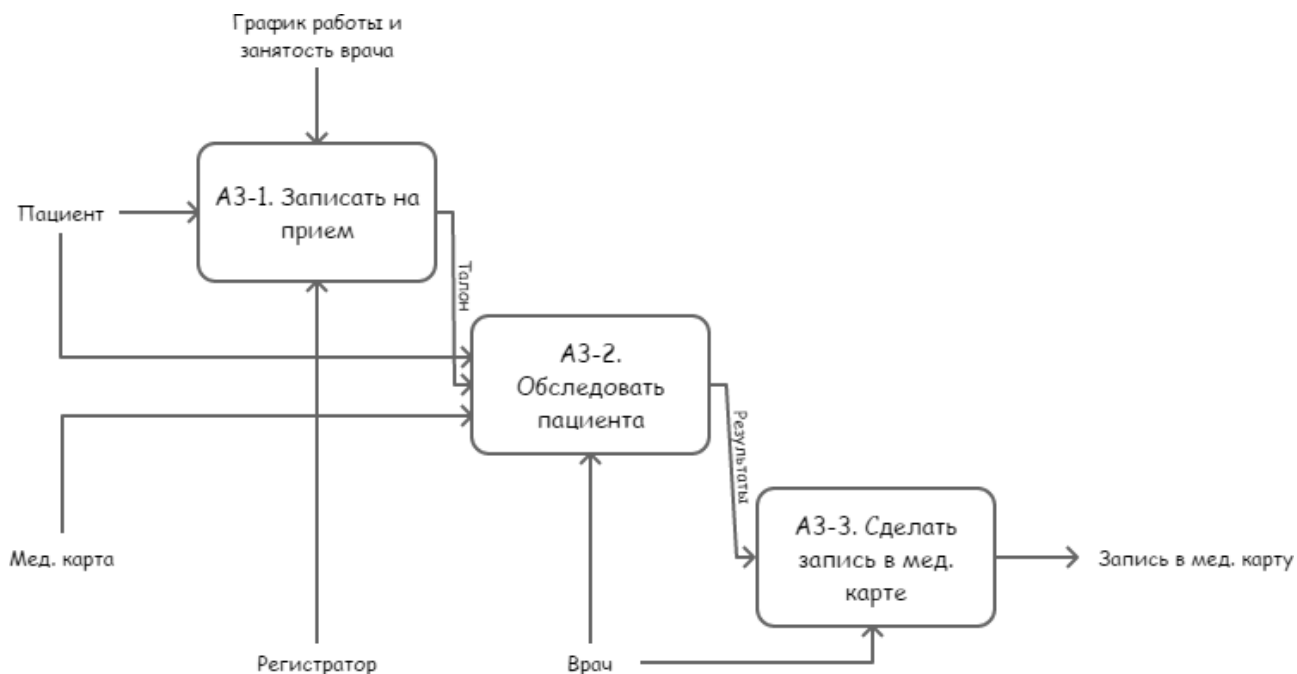
- The Chief doctor identifies the need for employees and initiates the hiring process (A1).
- The hiring process results in a list of employees.
- The Chief doctor and Registrar prepare the registration process (A2) by providing the passport and OMS policy.
- The Registrar brings the medical card (A2).
- The Chief doctor, Registrar, and Doctor prepare the acceptance of the patient (A3) by providing the doctor's work schedule and availability, the medical card, and the patient.
- The acceptance process results in the medical card.

**Рисунок 3.1 – Ключевые бизнес-процессы на начальном уровне описания**



**Рисунок 3.2** – Бизнес-процесс АІ «Нанять сотрудника» на 1 уровне описания

Подвергнем процессы A1 и A3 декомпозиции, получив описание IDEF0 на 1 уровне для каждого из них соответственно (рис. 3.2 – 3.3).

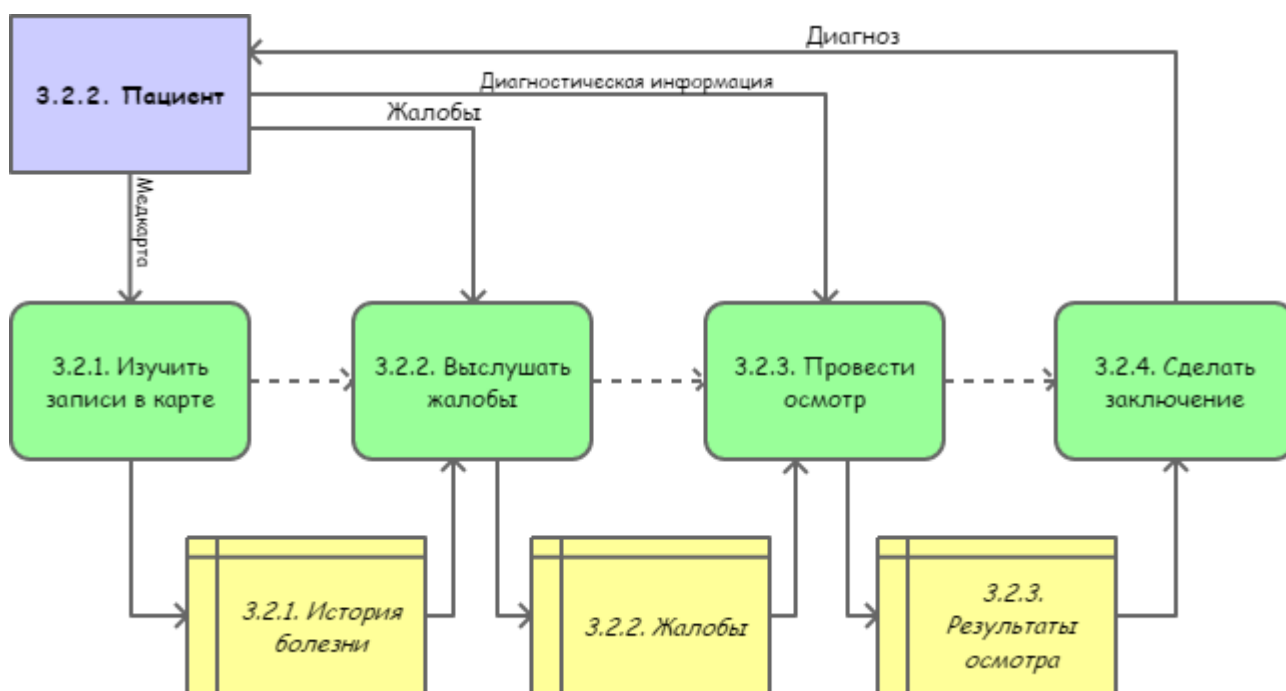


**Рисунок 3.3** – Бизнес-процесс A3 «Принять пациента» на 1 уровне описания

Следует указать, что процессы A1 – «Нанять сотрудника» и A2 – «Зарегистрировать мед. карту» в данном проекте являются второстепенными: не имеет особого смысла автоматизировать их по отдельности. Но в то же время без их автоматизации невозможно реорганизовать главный процесс A3 – «Принять пациента», так как для этого нужны списки (таблицы) врачей, которые могут принять пациентов, а также самих пациентов; это соответственно реализуется во вышеуказанных второстепенных процессах.

Подпроцесс A3.2 – «Обследовать пациента» нуждается в более подробном рассмотрении. Это можно сделать, подвергнув его декомпозиции. Однако выполнить моделирование 2 уровня детализации с помощью нотации IDEF0 невозможно, так как в процесс вовлекаются хранилища данных, связывающие между собой подпроцессы 2 уровня в единую цепь (поток

данных). В таком случае для моделирования хорошо подходит нотация DFD, о которой было сказано выше. Результат декомпозиции – описание 2 уровня с помощью нотации DFD представлено на рис. 3.4. Дальнейшая декомпозиция не имеет смысла в рамках данного проекта.

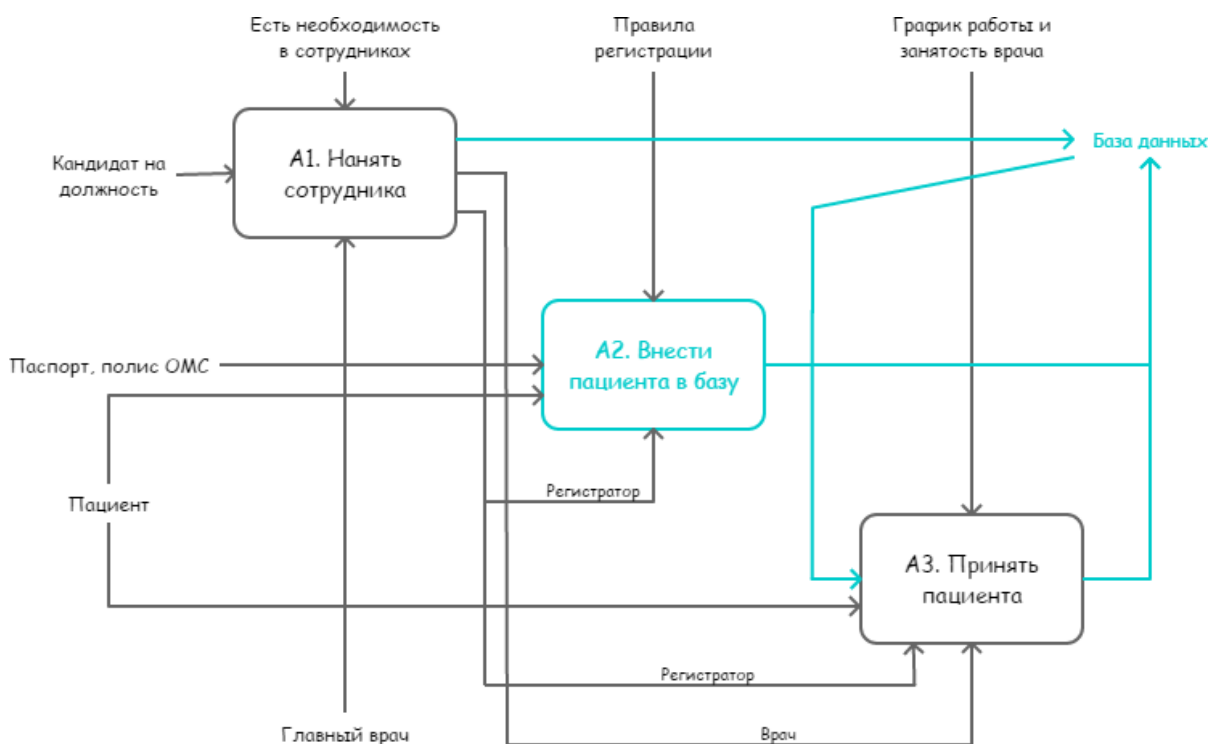


**Рисунок 3.4** – Результат декомпозиции процесса АЗ-2 – «Обследовать пациента» в нотации DFD (2 уровень описания)

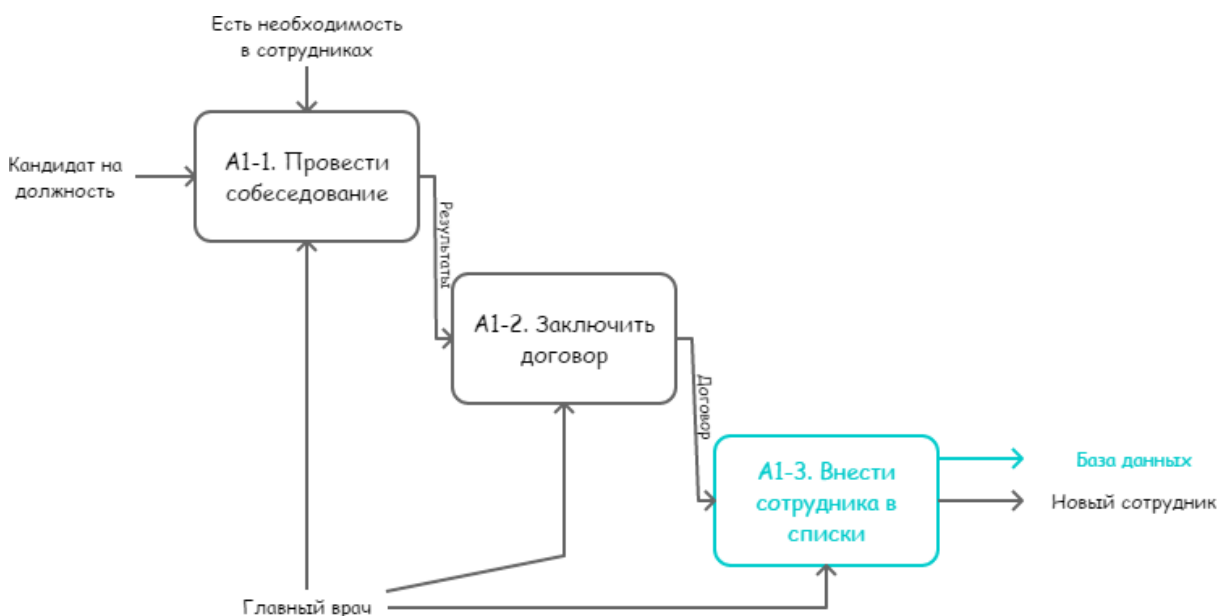
### 3.3. Проектирование процессов в ТО-ВЕ с помощью IDEF0 и DFD

Проектирование в модели ТО-ВЕ позволяет увидеть, как будут выглядеть ключевые бизнес-процессы функционирования медицинского учреждения после внедрения разрабатываемого приложения. Важной составляющей разрабатываемого приложения является база данных, призванная заменить все бумажные носители информации. Аналогично модели AS-IS, опишем ключевые бизнес-процессы на разных уровнях с помощью нотаций IDEF0 и DFD. Описание 0 уровня бизнес-процессов в модели ТО-ВЕ показано на рис. 3.5. По сравнению с моделью AS-IS, здесь процесс А2 –

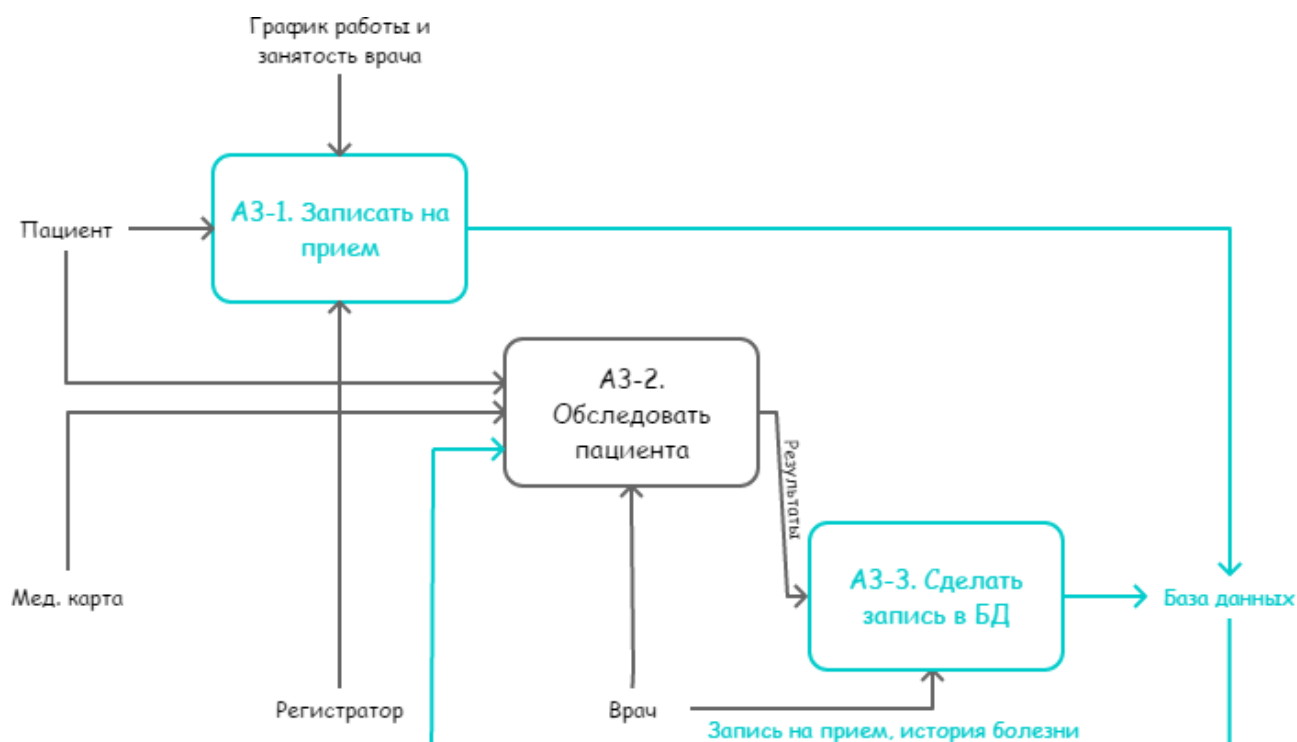
«Завести мед. карту» подвергся существенным изменениям, и его можно считать полностью автоматизированным.



**Рисунок 3.5** – ключевые бизнес-процессы на начальном уровне описания



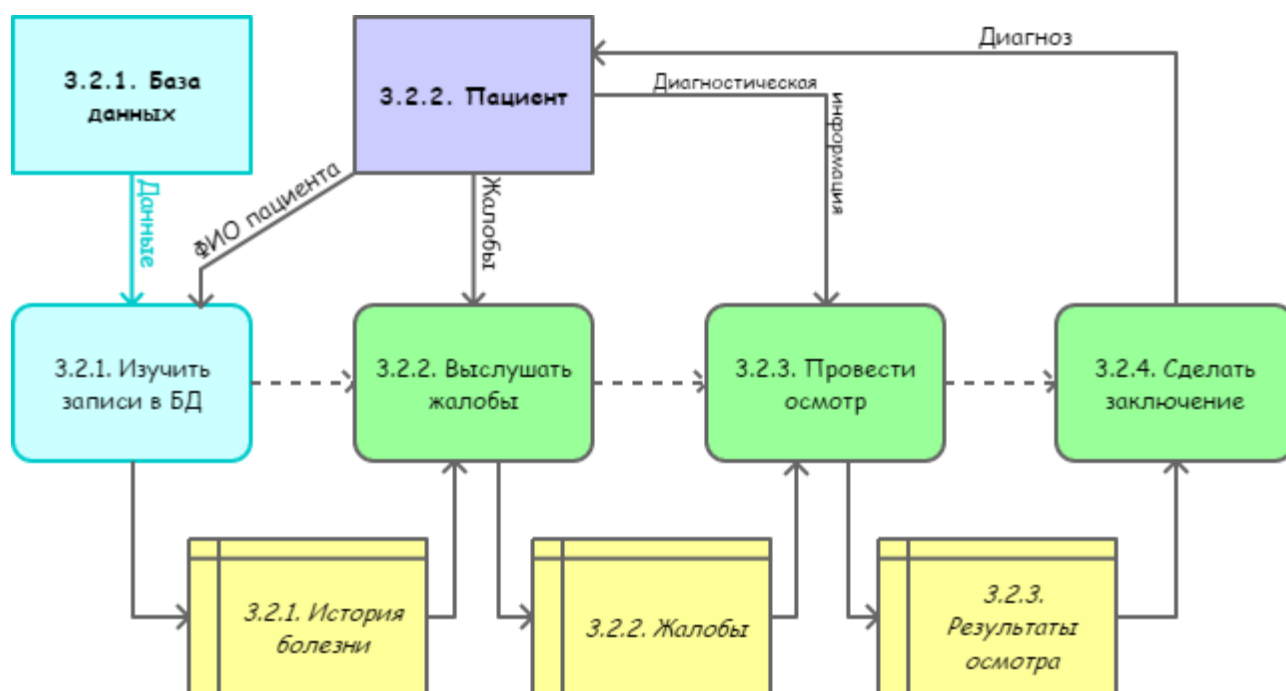
**Рисунок 3.6** – Бизнес-процесс «Нанять сотрудника» на 1 уровне описания



**Рисунок 3.7** – Бизнес-процесс «Принять пациента» на 1 уровне описания

Как и в модели AS-IS, для более подробного рассмотрения подвергнем процессы A1 и A3 декомпозиции, получив описание IDEF0 на 1 уровне для каждого из них соответственно (рис. 3.6 – 3.7). Представление подпроцесса A3-2 – «Принять пациента» на 2 уровне описания с помощью нотации DFD показано на рис. 3.8. Из рис. 3.5 – 3.8 можно заметить, что база данных в разрабатываемом приложении используется во многих аспектах деятельности поликлиники, связывая их воедино. Это позволяет существенно ускорить доступ к данным, работу с записями, а также повысить надежность хранения информации и упростить процесс реорганизации хранилищ данных. Также замена медицинских карт компьютерными таблицами повысит удобство пользования поликлиникой для пациентов и позволит освободить физическое пространство, занимаемое хранящимися картами.





**Рисунок 3.8** – Результат декомпозиции процесса A2.2 – «Обследовать пациента» в нотации DFD (2 уровень описания)

## Раздел 4. Разработка и тестирование

### *4.1. Задание циклов разработки по спиралевидной модели*

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

- Начало:
  - анализ требований (15 требований – см. табл. 2.1);
  - моделирование ключевых бизнес-процессов с помощью нотаций IDEF0 и DFD;
  - составление плана разработки по спиральной модели.
- I виток спирали:
  - планирование текущего цикла разработки (реализовать требования 1-4 в среде FoxPro);
  - анализ рисков на основе полученных требований;
  - реализация требований 1-4 в среде FoxPro;
  - тестирование реализованных возможностей программы;
  - демонстрация прототипа программы заказчику.
- II виток спирали:
  - планирование текущего цикла разработки (реализовать требования 5-6, 9-10 в среде FoxPro);
  - анализ рисков на основе полученных требований;
  - реализация требований 5-6, 9-10 в среде FoxPro;
  - тестирование реализованных возможностей программы;
  - демонстрация прототипа программы заказчику.
- III виток спирали:

- планирование текущего цикла разработки (реализовать требования 7-8, 11-12 в среде FoxPro);
  - анализ рисков на основе полученных требований;
  - реализация требований 7-8, 11-12 в среде FoxPro;
  - тестирование реализованных возможностей программы;
  - демонстрация прототипа программы заказчику.
- IV виток спирали:
- планирование текущего цикла разработки (реализовать требования 13-15 в среде FoxPro);
  - анализ рисков на основе полученных требований;
  - реализация требований 13-15 в среде FoxPro;
  - тестирование реализованных возможностей программы;
  - подготовка к релизу (исправление мелких недостатков, ошибок и т.д.);
  - тестирование конечного продукта;
  - релиз конечного продукта.

#### ***4.2. 1-й прототип программы (I виток спирали)***

Как упоминалось выше, реализация программы на первом витке спирали представляет собой совокупность таблиц с элементами управления самой среды разработки FoxPro, т.е. здесь нет никакого интерфейса, который мог бы облегчить работу пользователя с программой. На рис. 4.1 представлен фрагмент таблицы «Персонал», созданной в программе FoxPro. На рис. 4.2 показана структура разрабатываемого приложения. Согласно заявленным требованиям, в базу данных включены четыре таблицы: «Пациенты», «Персонал», «Запись на прием», «Анамнез», в которых хранятся соответствующие данные.

Фамилия	Имя	Отчество	Дата рождения	Пол	Домашний телефон	Рабочий телефон	Адрес	Должность	Специальность
Клыкков	Виктор	Александрович	12/12/79	М	9999999999	9999876543		Врач	Стоматолог
Абырвалг	Владимир	Петрович	04/19/88	М	9999999999	9999876543		Работник регистратуры	-
Иванов	Иван	Вальдемарович	05/05/75	М	9999999999	9999998765		Врач	Терапевт
Белов	Иван	Матвеевич	01/31/69	М	9999999999	9999999999		Врач	Рентгенолог
Краснова	Ольга	Петровна	10/17/90	Ж	9999999999	9999999999		Работник регистратуры	-
Тест	Тест	/ /	/ /						
Хитров	Николай	Исаакович	03/04/71	М	9999999877	9999999999		Врач	Ревматолог

Рисунок 4.1 – Фрагмент таблицы «Персонал»

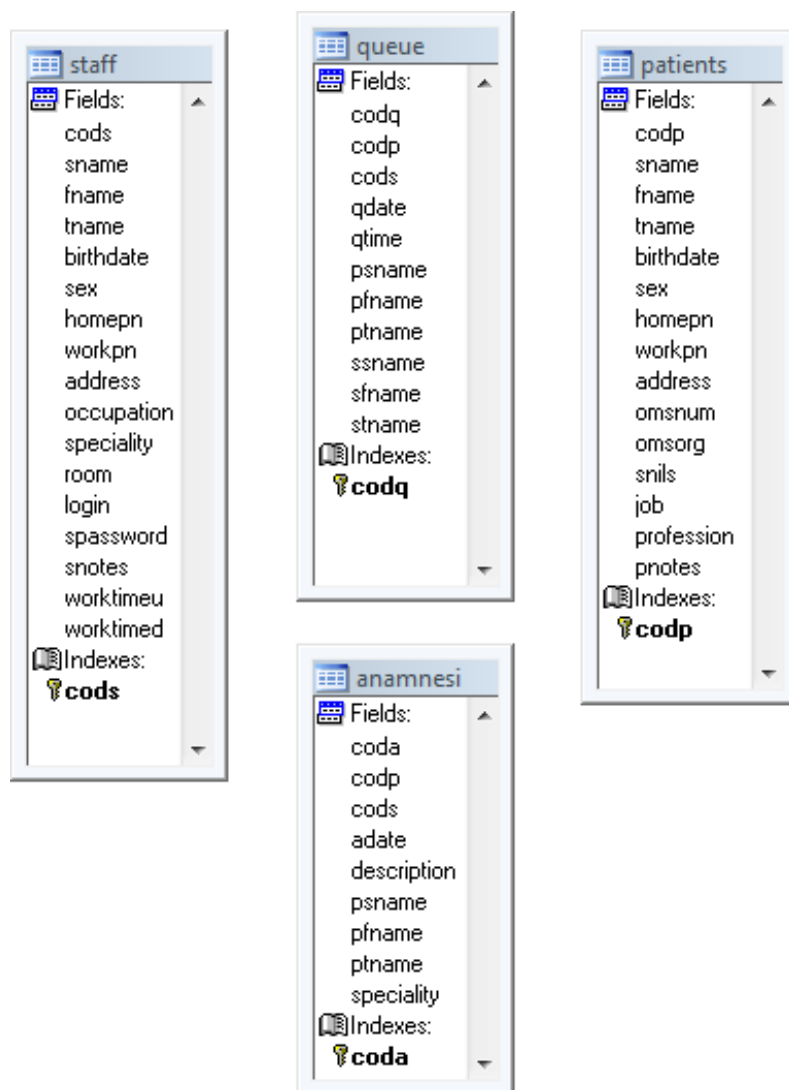
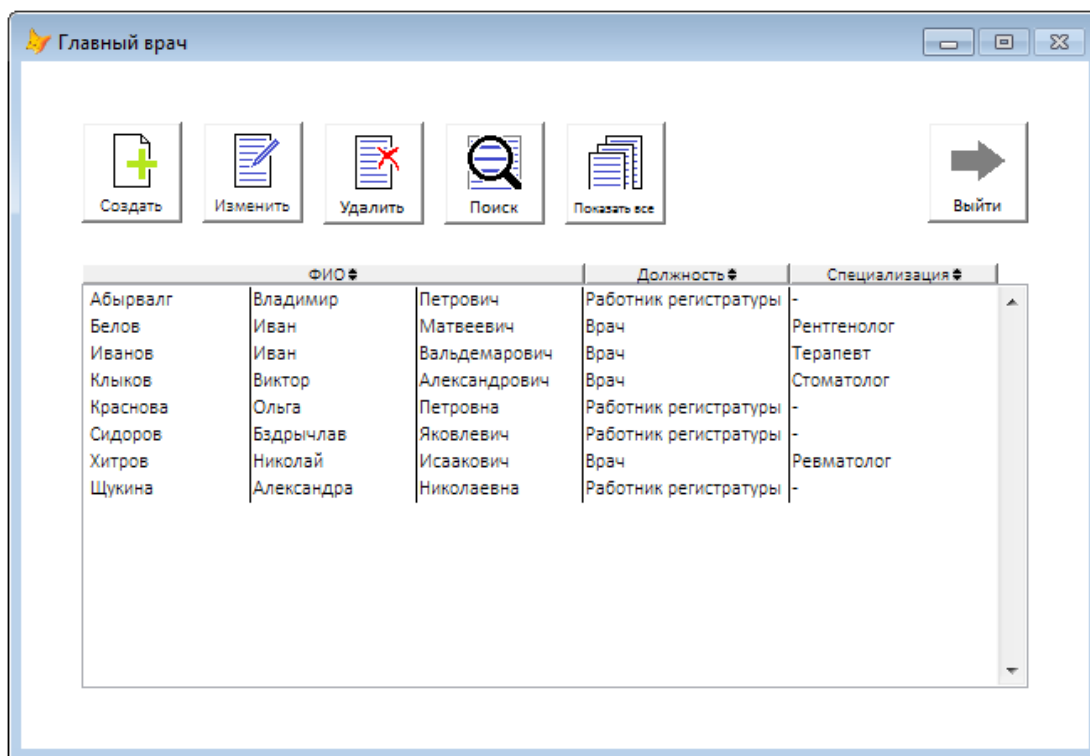


Рисунок 4.2 – Структура приложения в программе FoxPro

### 4.3. 2-й прототип программы (II виток спирали)

Во второй итерации в разрабатываемую программу добавлен пользовательский интерфейс для работы с таблицами «Пациенты» и

«Персонал», включающий основные возможности работы с записями (создание, редактирование, удаление, сортировка, поиск). Результаты разработки показаны на рис. 4.3 – 4.4.



**Рисунок 4.3** – Главное меню для работы с таблицей «Персонал»

Создать	Изменить	Удалить	Поиск	Показать все	Выйти
ФИО			Пол	Дата рождения	
Абдулов	Дмитрий	Павлович	М	06/21/86	
Гнолл	Ярослав	Иванович	М	08/08/94	
Гончарова	Юлия	Борисовна	Ж	08/04/56	
Петрова	Валерия	Алексеевна	Ж	11/04/89	
Шубин	Кирилл	Георгиевич	М	01/01/60	

**Рисунок 4.4** – Фрагмент главного меню для работы с таблицей «Пациенты»

При нажатии кнопки «Создать» открывается окно, представленное на рис. 4.5. В открывшемся окне пользователь может ввести данные. При нажатии

кнопки «Завершить регистрацию» запись добавляется в таблицу; кнопка «Назад» возвращает главное меню на экран.

Регистрация пациента

Фамилия  Дата рождения

Имя  Пол

Отчество

Домашний телефон ( ) -  Рабочий телефон ( ) -

Номер страхового полиса

Страховая медицинская организация

СНИЛС

Адрес

Место работы

Профессия

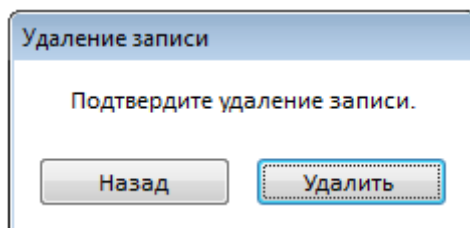
Примечания

Назад Завершить регистрацию

**Рисунок 4.5** – Окно создания новой записи в таблице «Пациенты»

В процессе работы с приложением может потребоваться изменить данные. Для этого можно использовать кнопку «Изменить», предварительно выбрав запись, которую нужно изменить, в списке. В результате открывается такое же окно, как и в процессе создания записи (рис. 4.5), где пользователь может редактировать данные. Может также возникнуть необходимость удаления записи. При выборе нужной записи и нажатии кнопки «Удалить»

открывается диалоговое окно, изображенное на рис. 4.6. Это сделано для того, чтобы предотвратить случайное удаление записи пользователем. При нажатии кнопки «Удалить» запись удаляется. Кнопка «Назад» возвращает пользователя к главному меню.



**Рисунок 4.6** – Окно, запрашивающее подтверждение пользователя на удаление записи

Для обеспечения высокой скорости работы пользователя добавлена функция поиска записей. При нажатии на кнопку «Поиск» открывается диалоговое окно, изображенное на рис. 4.7.

**Рисунок 4.7** – Окно поиска

После ввода в поля необходимых значений и нажатия кнопки «Поиск» в основном списке главного меню будут показаны только записи, удовлетворяющие указанным критериям. Если одно из полей в окне поиска не было заполнено, то поиск по данному критерию не производится. Кнопка

«Показать все» отменяет поиск и выводит на экран все записи. Сортировка данных в порядке возрастания происходит при нажатии на заголовок списка (например, «ФИО» или «Дата рождения» на рис. 4.3). При повторном нажатии происходит сортировка по тому же критерию, но в порядке убывания. При нажатии кнопки «Выход» программа закрывается.

#### 4.4. 3-й прототип программы (III виток спирали)

В третьей итерации в разрабатываемую программу добавлен пользовательский интерфейс для работы с таблицами «Запись на прием» и «Анамнез», включающий основные возможности работы с записями (создание, редактирование, удаление, сортировка, поиск). Результаты разработки показаны на рис. 4.8 – 4.9.

Записать на прием Отменить запись Поиск Показать все Выйти							
ФИО пациента		Дата записи		Время записи	ФИО врача		
Абдулов	Дмитрий	Павлович	04/27/18	14:00	Иванов	Иван	Вальдемарович
Абдулов	Дмитрий	Павлович	05/01/18	10:00	Белов	Иван	Матвеевич
Абдулов	Дмитрий	Павлович	05/03/18	14:00	Иванов	Иван	Вальдемарович
Гнолл	Ярослав	Иванович	05/03/18	11:00	Иванов	Иван	Вальдемарович
Гнолл	Ярослав	Иванович	05/04/18	12:00	Клыков	Виктор	Александрович
Гончарова	Юлия	Борисовна	05/15/18	09:00	Хитров	Николай	Исаакович
Петрова	Валерия	Алексеевна	05/15/18	10:00	Иванов	Иван	Вальдемарович
Шубин	Кирилл	Георгиевич	04/30/18	16:00	Клыков	Виктор	Александрович
Шубин	Кирилл	Георгиевич	05/02/18	15:00	Иванов	Иван	Вальдемарович

Рисунок 4.8 – Фрагмент главного меню для работы с таблицей «Запись на прием»

Добавить анамнез Просмотр Удалить Поиск Показать все Выйти				
ФИО			Врач	Дата
Абдулов	Дмитрий	Павлович		03/04/18
Абдулов	Дмитрий	Павлович	Стоматолог	04/28/18
Абдулов	Дмитрий	Павлович	Терапевт	04/28/18
Абдулов	Дмитрий	Павлович	Стоматолог	04/29/18
Гнолл	Ярослав	Иванович	Стоматолог	04/28/18
Гнолл	Ярослав	Иванович	Терапевт	04/29/18
Гончарова	Юлия	Борисовна	Терапевт	05/15/18
Шубин	Кирилл	Георгиевич	Стоматолог	04/30/18
Шубин	Кирилл	Георгиевич	Стоматолог	04/30/18

Рисунок 4.9 – Фрагмент главного меню для работы с таблицей «Анамнез»



Пациент				
Абдулов	Дмитрий	Павлович	М	06/21/86
Гнолл	Ярослав	Иванович	М	08/08/94
Шубин	Кирилл	Георгиевич	М	01/01/60
Петрова	Валерия	Алексеевна	Ж	11/04/89
Гончарова	Юлия	Борисовна	Ж	08/04/56

Врач				
Клыков	Виктор	Александрович	Стоматолог	09:00 17:00
Абырвалг	Владимир	Петрович	-	08:00 16:00
Иванов	Иван	Вальдемарович	Терапевт	08:00 16:00
Белов	Иван	Матвеевич	Рентгенолог	10:00 14:00
Краснова	Ольга	Петровна	-	08:00 16:00
Хитров	Николай	Исмакович	Ревматолог	09:00 17:00
Шукина	Александра	Николаевна	-	08:00 16:00
Сидоров	Бздрычлав	Яковлевич	-	08:00 16:00

Дата: / /

Время: :

Назад

Записать на прием

Рисунок 4.10 – Окно для записи на прием

В меню «Запись на прием» при нажатии кнопки «Записать на прием» откроется окно, изображенное на рис. 4.10. Выбрав в левом столбце пациента, а в правом – врача, а также назначив дату и время записи, следует нажать на кнопку «Записать на прием» для завершения создания записи. Кнопка «Назад» возвращает пользователя к главному меню. Кнопка «Отменить запись» отвечает за удаление записи. Назначение остальных кнопок идентично рассмотренному ранее. При нажатии на кнопку «Добавить анамнез» при работе с таблицей «Анамнез» откроется окно создания записи (рис 4.11).

Пациент				
Абдулов	Дмитрий	Павлович	М	06/21/86
Гнолл	Ярослав	Иванович	М	08/08/94
Шубин	Кирилл	Георгиевич	М	01/01/60
Петрова	Валерия	Алексеевна	Ж	11/04/89
Гончарова	Юлия	Борисовна	Ж	08/04/56

**Описание**

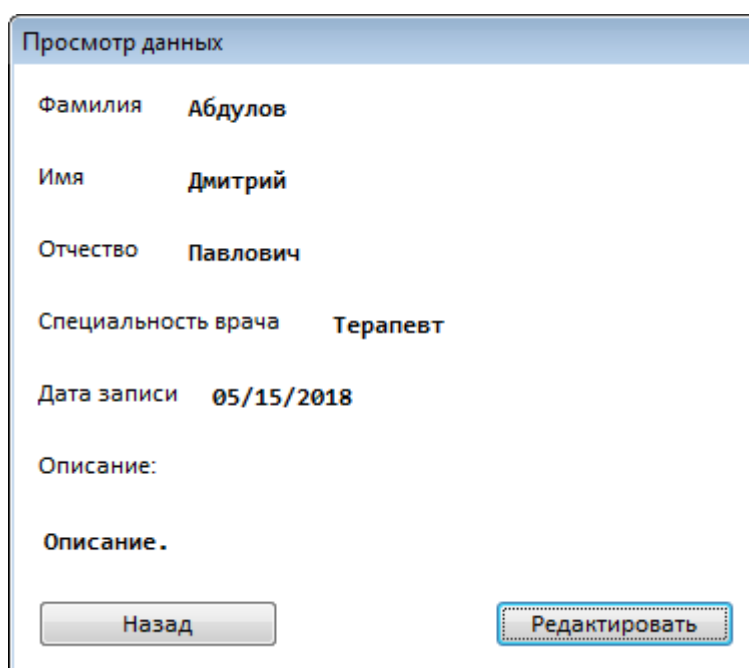
Описание. |

Назад

Добавить анамнез

Рисунок 4.11 – Окно для добавления протокола осмотра

После нажатия кнопки «Добавить анамнез» новая запись будет внесена в таблицу. При этом специальность врача и дата записи будут указаны автоматически. Нажатие на кнопку «Просмотр» при выборе записи в списке позволяет просмотреть запись без её изменения (рис. 4.12). Если пользователь является автором этой записи, то он сможет редактировать её, нажав на соответствующую кнопку. Также, если пользователь является автором записи, то он сможет её удалить, нажав на соответствующую кнопку в главном меню.



The screenshot shows a window titled "Просмотр данных" (View Data). It contains the following fields:

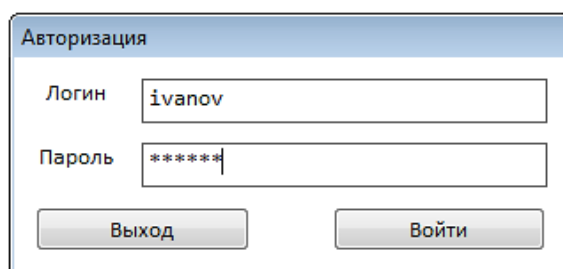
Фамилия	Абдулов
Имя	Дмитрий
Отчество	Павлович
Специальность врача	Терапевт
Дата записи	05/15/2018
Описание:	
Описание.	

At the bottom, there are two buttons: "Назад" (Back) and "Редактировать" (Edit).

**Рисунок 4.12** – Режим просмотра записи в таблице «Анамнез»

#### **4.5. Конечный продукт (IV виток спирали)**

В четвертой итерации в разрабатываемую программу добавлены формы для авторизации пользователей, отображения записи на текущий день и вывода полной информации о записанном пациенте. Форма авторизации открывается при запуске программы (рис. 4.13). При нажатии на кнопку «Войти», в зависимости от должности пользователя, открывается то или иное окно работы с данными.



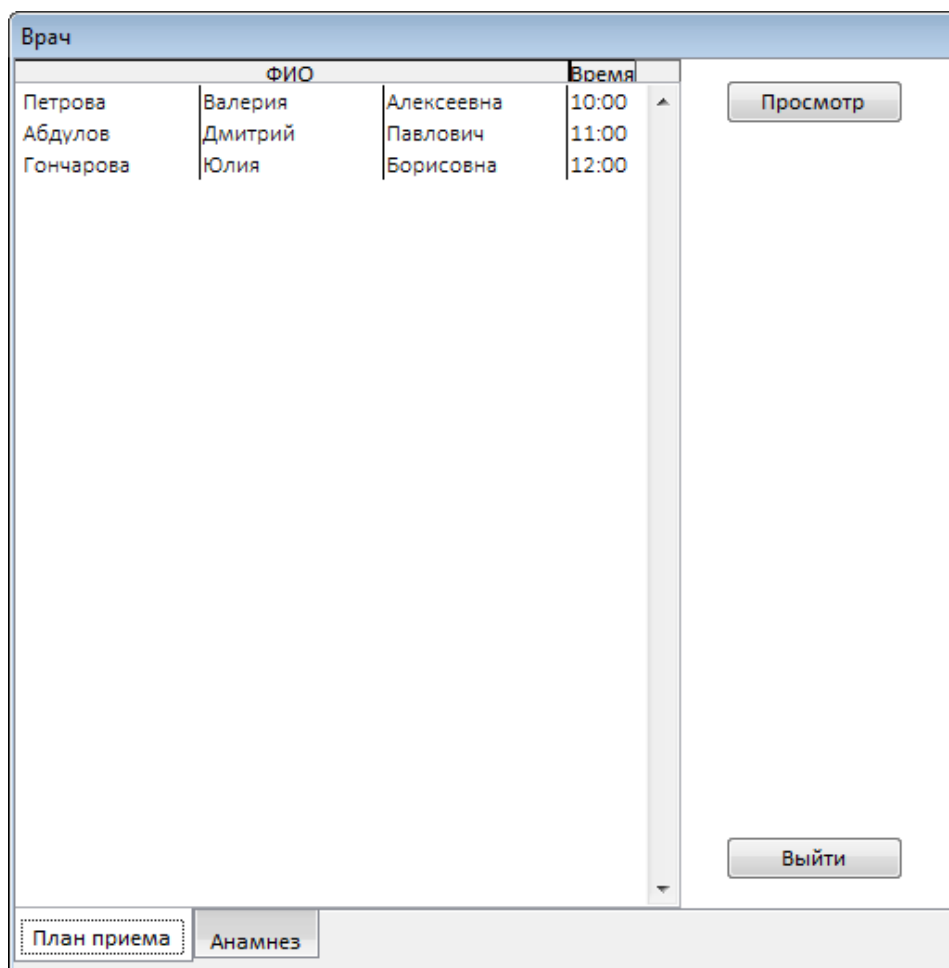
Авторизация

Логин

Пароль

**Рисунок 4.13** – Форма для авторизации

Теперь врач может отслеживать план приема пациентов на текущий день, выбирая соответствующую вкладку внизу главного меню (рис. 4.14). При выделении записи в списке и нажатии на кнопку «Просмотр» открывается окно, отображающее все данные о пациенте в режиме чтения (рис. 4.15).



Врач

ФИО			Время
Петрова	Валерия	Алексеевна	10:00
Абдулов	Дмитрий	Павлович	11:00
Гончарова	Юлия	Борисовна	12:00

**Рисунок 4.14** – Отображение плана приема пациентов

Фамилия	Абдулов	Дата рождения	06/21/1986
Имя	Дмитрий	Пол	М
Отчество	Павлович		
Домашний телефон	(999) 999-0000	Рабочий телефон	(999) 999-8888
Номер страхового полиса	123456789012345678901234		
Страховая медицинская организация	МАКС-М		
СНИЛС	98765432109876543210987		
Адрес	Москва, Михневская ул., 19		
Место работы	-		
Профессия	-		
Примечания:			

Назад

**Рисунок 4.15** – Вывод полной информации о пациенте

#### **4.6. Тестирование программы**

Тестирование программного обеспечения – процесс анализа программного продукта и сопутствующей документации с целью выявления недостатков в работе и повышения его качества [17]. Обычно для проверки работоспособности разработанного ПО 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).

Интеграционное тестирование (integration testing) направлено на проверку взаимодействия между несколькими частями приложения (каждая из которых, в свою очередь, проверена отдельно на стадии модульного

тестирования). 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») [18]. Описание разработанных модулей программы, соответствующих требованиям (табл. 3.1) приведено в п. 4.2 – 4.5. Функциональное тестирование данных модулей показало их полную работоспособность и отсутствие ошибок, т.е. все функциональные требования были успешно реализованы.

Интеграционное тестирование проводилось в случаях, когда происходит взаимодействие модулей программы (требования 7-8, 11-15, соответствующие III и IV виткам спирали), и также оказалось успешным. В частности, в процессе записи пациента на прием (рис. 4.10) происходит добавление в таблицу «Запись на прием» данных из таблиц «Пациенты» и «Персонал», выполняемое без сбоев в работе программы. Аналогично можно рассмотреть процессы отслеживания врачом записи на прием и просмотра данных о записанных пациентах (рис. 4.14, 4.15), где при основной выполняемой работе с таблицей «Анамнез», врач также может отображать данные из таблиц «Запись на прием» и «Пациенты».

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 (без индексирования таблицы). Результаты нагрузочного тестирования приведены в табл. 4.1.

**Таблица 4.1** – *Время отклика системы при различном числе записей*

Число записей	Время отклика	
	Добавление записи	Поиск
10	200 ± 50	200 ± 50
50	200 ± 50	200 ± 50

Число записей	Время отклика	
	Добавление записи	Поиск
100	$200 \pm 50$	$200 \pm 50$
500	$200 \pm 50$	$200 \pm 50$
10000	$2100 \pm 400$	$400 \pm 100$

Как видно из табл. 4.1, существенная задержка возникает лишь при количестве записей  $\sim 10000$  и более. В остальных случаях время отклика не превышает 0,5 с и почти не замечается пользователем. Десять тысяч записей в базе образуются приблизительно за две недели работы средней городской поликлиники (в случае общего приема  $\sim 1000$  чел./день за 10 рабочих дней), согласно данным, полученным в ходе опроса. В таком случае при очень большом числе записей возможно увеличение времени отклика. Однако следует учесть, что нагрузочное тестирование проводилось на обычном ПК, в то время как в поликлинике все действия обрабатываются сервером, вычислительная мощность которого во много раз выше. Таким образом, можно заключить, что нагрузочное тестирование проведено успешно.

## Заключение

В процессе работы были проанализированы пользовательские требования (15 требований), полученные путем опроса и изучения документации. После этого были определены приоритеты полученных требований с помощью метода MuSCoW. 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; список обеспечивал удобство отслеживания выполнения требований разработчиком на всех этапах проектирования.

После того, как требования были проанализированы, проводилось составление моделей AS-IS и TO-BE ключевых бизнес-процессов организации в нотациях IDEF0 (0-1 уровни) и DFD (2 уровень), и проведено их сопоставление на каждом уровне. Это позволило определить структуру разрабатываемого приложения. Затем был составлен план создания ПО согласно спиралевидной модели. Разработка приложения производилась в среде MS FoxPro, где поэтапно были реализованы все указанные требования, исправлены недочеты и ошибки в работе программы.

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. Таким образом, можно сказать, что все поставленные задачи были выполнены. Для повышения наглядности в процессе подготовки отчета было составлено 4 таблицы и приведено 24 рисунка, включающих элементы теории, модели бизнес-процессов на различных уровнях и снимки экрана, показывающие интерфейс разрабатываемой программы.



## Список использованных литературных источников

1. Чистякова Ю. А., Мартюгов А. С., Селяничев О. Л. Информационная система автоматизации работы медицинского учреждения // Новые информационные технологии в науке нового времени: сборник статей международной научно-практической конференции. – 2016. – С. 63-65.
2. Калиниченко В. И. Комплексная автоматизация услуги «Запись на прием к врачу» // II симпозиума «Информационные технологии для здравоохранения юга России. Сочи 2011». – С. 28.
3. ГОСТ Р ИСО/МЭК 12207-2010. Информационная технология. Системная и программная инженерия. Процессы жизненного цикла программных средств.
4. 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.
5. Невлюдов И. Ш., Евсеев В. В., Бортникова В. О. Модели жизненного цикла программного обеспечения при разработке корпоративных информационных систем технологической подготовки производства. – 2011.
6. Родигин Л. А. Оценка совокупной стоимости владения туристским интернет-проектом. / Л. А. Родигин, К. В. Наймарк. // Litres. – 5 сент. 2017 г.
7. Boehm B.W. A spiral model of software development and enhancement / Boehm B., Egyed A. // IEEE Computer, May 1988, pp. 61-72
8. Анисимов В. В. Проектирование информационных систем. Конспект лекций. М.: Дальневосточный государственный университет путей сообщения. URL: <https://sites.google.com/site/anisimovkhv/publication/umr/pris> (дата обращения 05.04.2018)

9. Химонин Ю. И. Сбор и анализ требований к программному продукту (Версия 1.03). – 2009, – 51 с.
10. Бариленко В. И. Бизнес-анализ как инструмент обоснования условий устойчивого развития // Вопросы региональной экономики. – 2015. – Т. 24. – №. 3. – С. 137.
11. Яблочников Е. И., Молочник В. И., Фомина Ю. Н. Реинжиниринг бизнес-процессов проектирования и производства // Учебное пособие. – СПб: СПбГУ ИТМО. – 2008.
12. Стрекалова Н. Д. Концепция бизнес-модели: методология системного анализа // Известия Российского государственного педагогического университета им. АИ Герцена. – 2009. – №. 92.
13. Половодов Д. А., Половодова Е. А., Сергин С. Е. Применение технологии IDEF для моделирования медицинских СУБД // Роль науки в развитии общества. – 2015. – С. 7.
14. РД IDEF0 – 2000. Методология функционального моделирования IDEF0.
15. Степанов Д. Ю. Анализ, проектирование и разработка корпоративных информационных систем: уровень бизнес-процессов / МИРЭА. - М., 2017. – URL: [http://stepanovd.com/training\\_erp\\_1-7\\_ru.pdf](http://stepanovd.com/training_erp_1-7_ru.pdf) (дата обращения 14.05.2018)
16. Петеляк В. Е. и др. Data Flow Diagramming: особенности построения моделей описания управления потоками данных в организационных системах // Фундаментальные исследования. – 2015. – Т. 2. – №. 8.
17. Штенников Д.Г. Разработка информационных систем в образовании. Учебное пособие. – СПб: СПбГУ ИТМО, 2012. – 242 с.
18. Куликов С. С. Тестирование программного обеспечения // Базовый курс: практ. пособие. Минск: Четыре четверти. – 2015. – С. 63-110.