

**Государственное образовательное учреждение высшего  
профессионального образования  
Кузбасский государственный технический университет  
имени Ф.А. Горбачева**

**УТВЕРЖДАЮ**

Зав. Кафедрой

\_\_\_\_\_ Чичерин И. В.

“ \_\_\_\_ ” \_\_\_\_\_

**Мониторинг детей с ВПС**

**ДИПЛОМНЫЙ ПРОЕКТ**

Инв. № подл.	Подп. и дата
Взам. инв. №	Инв. № дубл.
Подп. и дата	Подп. и дата

Дипломник

\_\_\_\_\_ Калесников Д.С.

“ \_\_\_\_ ” \_\_\_\_\_

Дипломник

\_\_\_\_\_ Кошкин Н.Г.

“ \_\_\_\_ ” \_\_\_\_\_

Кемерово 2013 г.

# Содержание

<b>Введение</b>	<b>8</b>
<b>1 Описание предприятия</b>	<b>10</b>
1.1 История предприятия .....	10
1.2 Организационная структура предприятия .....	12
1.3 Основные подразделения предприятия .....	12
1.3.1 СО РАМН (ФГБУ "НИИ КПССЗ"СО РАМН) .....	12
1.3.2 Кемеровский Кардиологический Диспансер МБУЗ ККД .....	13
1.3.3 Кафедра кардиологии .....	13
1.4 Подразделение связанное с предметной областью .....	13
<b>2 Существующие бизнес-процессы</b>	<b>15</b>
2.1 Система мониторинга как процесс .....	15
2.2 Система мониторинга как совокупность процессов .....	15
2.3 Амбулаторный педиатрический прием .....	15
2.4 Заключительная стадия мониторинга .....	16
<b>3 Проблемы</b>	<b>17</b>
3.1 Задержка с операционным вмешательством .....	17
3.2 Наблюдение в послеоперационный период .....	17
3.3 Расстояние .....	17
3.4 Взаимодействие .....	17
3.5 Анализ, прогнозирование, тенденции .....	18
3.6 Лечение в стационаре .....	18
<b>4 Цели</b>	<b>19</b>
4.1 Постоянный мониторинг состояния пациента .....	19
4.1.1 Амбулаторное наблюдение .....	19
4.1.2 Наблюдение в стационаре .....	19

Подп. и дата					
Инв. № дубл.					
Взам. инв. №					
Подп. и дата					
Инв. № подл.					

Дипломный проект					
Изм.	Лист	№ докум.	Подп.	Дата	
Разраб.		Калесников Д.С., Кошкин Н.Г.			
Пров.					
Н. контр.		Ванеев О.Н.			
Утв.					

Мониторинг детей с ВПС		Лит.	Лист	Листов
		У	2	108

4.1.3	Постоянный анализ получаемых данных .....	19
4.1.4	Постоянное взаимодействие пациента с врачом .....	20
4.1.5	Взаимодействие между врачами .....	20

## 5 Требования 21

5.1	Данные .....	21
5.2	Интерфейс .....	21
5.3	Архитектура .....	22
5.3.1	Надежность .....	22
5.3.2	Безопасность .....	22
5.3.3	Доступность .....	23
5.3.4	Масштабируемость .....	23
5.3.5	Гибкость .....	23
5.4	Технологии .....	24
5.5	Функциональность .....	25
5.5.1	Пациент .....	25
5.5.2	Доктор .....	26
5.5.3	Менеджер .....	27
5.5.4	Электронный (интернет) прием .....	27
5.5.5	Интернет-консультация .....	28

## 6 Готовые решения 29

6.1	Решения на базе системы 1С:Предприятие .....	29
6.1.1	1С Медицина Поликлиника .....	29
6.1.2	1С Рарус Амбулатория .....	29
6.2	Решения для автоматизации медицинского документооборота .....	29
6.3	Комплексная автоматизация медицинского предприятия .....	30
6.4	Выбор готового решения .....	30

## 7 Корректировка бизнес-процессов 32

7.1	Составляющие процесса мониторинга .....	32
7.2	Основные этапы процесса мониторинга .....	33
7.2.1	Регистрация в системе .....	33

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div style="text-align: center; font-size: 24px; font-weight: bold;">Дипломный проект</div>					Лист				
										3				
										Изм.	Лист	№ докум.	Подп.	Дата



11.1.1	Использование языка PHP .....	46
11.1.2	Zend Framework .....	46
11.1.3	Переход на платформу ASP.NET MVC .....	46
11.1.4	Большие затраты времени на конфигурирование .....	46
11.2	Выбор платформы Ruby on Rails .....	47
11.2.1	Регламентированный доступ к базе данных .....	47
11.2.2	Готовая система валидации вводимых данных .....	48
11.2.3	Создание связей между сущностями .....	48
11.2.4	Использование соглашений по конфигурации .....	49
11.2.5	Гибкость языка Ruby .....	49
11.2.6	Вывод .....	49
11.3	Frontend .....	50
11.3.1	Backbone.js .....	50
11.3.2	Coffeescript .....	50
11.3.3	RequireJs .....	51
11.3.4	Twitter Bootstrap .....	51
11.3.5	Ресурсы приложения .....	51
11.3.6	Средство построения графиков .....	52
11.4	Backend .....	52
11.4.1	Ruby .....	52
11.4.2	Ruby on Rails .....	53
11.4.3	Концепция MVC .....	53
11.5	Дополнительные возможности платформы Ruby on Rails .....	56
11.5.1	Встроенный генератор Rails Generator .....	56
11.5.2	Формы ввода данных .....	57
11.5.3	Рассылка электронной почты .....	57
11.5.4	Система контроля версий базы данных .....	58
11.6	Использование сторонних библиотек на языке Ruby .....	59
11.6.1	Аутентификация и авторизация .....	59
11.6.2	Доступ к базе данных .....	60
11.6.3	Служебная утилита rake .....	61
11.6.4	Тестирование отправки писем .....	61

Подп. и дата		Инв. № дубл.		Взам. инв. №		Подп. и дата		Инв. № подл.	
Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект				Лист
									5

11.7	Postgresql.....	62
11.8	Websocket .....	62

## 12 Организация процесса разработки 64

12.1	Определение условий разработки .....	64
12.2	Система управления версиями Git .....	64
12.3	Веб-сервис GitHub .....	64
12.4	Организация документации по проекту .....	65
12.4.1	Веб-приложение Google Docs .....	65
12.4.2	Веб-приложение diagram.ly .....	65
12.4.3	XMind.....	65
12.4.4	Plant UML .....	65

## 13 Разработка проекта 67

13.1	План разрабоки .....	67
13.1.1	Skeleton .....	67
13.1.2	General .....	67
13.1.3	Patient .....	67
13.1.4	Manager .....	68
13.1.5	Patient/Doctor .....	68
13.1.6	Doctor .....	68
13.2	Git Workflow.....	68
13.3	Rails Style Guide .....	69
13.4	Физическое проектирование базы данных .....	70
13.4.1	Миграции .....	70
13.5	Тестирование .....	71

## 14 Описание системы 72

14.1	Кабинеты .....	72
14.2	События .....	73
14.3	Диагностика .....	74
14.3.1	Прием данных .....	74
14.3.2	Доступ к диагностическим данным .....	75

Подп. и дата		Инв. № дубл.		Взам. инв. №		Подп. и дата		Инв. № подл.	
Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект				Лист
									6



## Введение

В настоящее время, люди страдающие серьезными системными заболеваниями (например, сердечно-сосудистыми) стали получать возможность проходить необходимое лечение и даже возвращаться (до определенной степени) к полноценной жизни. Основная трудность с которой они сталкиваются при этом - необходимость постоянного врачебного наблюдения с целью сохранения достигнутого состояния оздоровления. Наблюдение предполагает собой частые визиты к врачу; отсюда вытекает потеря личного времени пациента на преодоление расстояния, на ожидание в очереди и др. Помимо этого на медицинское учреждение накладывается функция сбора и анализа медицинской статистики.

Согласно исследованиям GBI Research<sup>1)</sup> в ближайшие годы здравоохранение столкнется с серьезными проблемами: повысится доля пожилых граждан в общей структуре населения и значительно увеличится численность пациентов с хроническими заболеваниями — сердечно-сосудистыми, легочными, а также диабетом. По оценкам Всемирного фонда диабета, к 2025 г. 80% пациентов с диабетом будут проживать в странах, где подавляющее число граждан обладают низкими или средними доходами.

На основе полученных результатов очевидно возрастание необходимости в удаленном медицинском обслуживании. Технические средства удаленного мониторинга, с одной стороны, избавляют пациентов от необходимости регулярно посещать лечащих врачей (что особенно важно для обитателей удаленных регионов), а с другой — на регулярной основе обеспечивают медицинских работников актуальной информацией о состоянии здоровья их подопечных.

После внимательного анализа приведенных выше фактов, стала проясняться общая проблема, присущая данному роду медицинского обслуживания. Пациенту для соблюдения непрерывного медицинского наблюдения необходимо личное присутствие в медицинском учреждении, даже в самых малозначимых ситуациях. В то же время, последние несколько лет возросли темпы компьютеризации населения, также повсеместно стало распространяться относительно недорогое подключение к сети Интернет. В связи с этим становится вполне логичной идея частично реализовать общение пациента и врача с использованием современных информа-

<sup>1)</sup> <http://ria-ami.ru/news/26944>

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										8				
										Изм.	Лист	№ докум.	Подп.	Дата



ционных технологий.

Таким образом, основной целью разработки является создание такой системы, которая бы позволила реализовать обмен медицинской информацией между доктором и пациентом дистанционно, через сеть Интернет. Система также должна хранить полученную информацию и выполнять типовые операции с ними с целью мониторинга. В целях исследования и разработки системы нами были использованы бизнес-процессы и организационная структура медицинского учреждения “Кузбасский кардиологический центр”.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										9
Изм.	Лист	№ докум.	Подп.	Дата						

Копировал

Формат А4

# 1 Описание предприятия

Кузбасский кардиологический центр представляет собой уникальный комплекс специализированных научных и лечебно-профилактических учреждений, осуществляющих высокотехнологичную медицинскую помощь пациентам с болезнями сердечно-сосудистой системы.

## 1.1 История предприятия

История создания Кузбасского кардиологического центра началась в марте 1957 года, когда в Кемеровской области была сделана первая операция на сердце - пальцевая митральная комиссуротомия при митральном стенозе. Операцию проводил заслуженный врач РФ, почетный гражданин города Кемерово, хирург М.А. Подгорбунский на базе отделения торакальной хирургии Областной клинической больницы №1.

Год спустя, осенью 1958 года был организован кабинет для ангиокардиографии. В 1974 году на основании приказа МЗ СССР «Об организации центра сердечно-сосудистой хирургии в г. Кемерово» на базе Областной клинической больницы № 1 открыто кардиологическое отделение на 40 коек, а с 1975 года - на 50 коек.

В 1989 году Администрация города Кемерово принимает решение о строительстве Кемеровского кардиологического диспансера (ККД) на правом берегу реки Томи в живописном сосновом бору. Организация такого специализированного учреждения была вызвана необходимостью расширения диагностических и лечебных возможностей кардиологической помощи больным, страдающим сердечно-сосудистыми заболеваниями. Возглавил кардиодиспансер доктор медицинских наук, профессор, в настоящее время академик РАМН Леонид Семенович Барбараш, один из пионеров кардиохирургии Кемеровской области. Созданию и развитию кардиодиспансера активно помогали руководители крупных промышленных предприятий, администрации города и области.

С 1994 года управление учреждением осуществляется двумя руководителями: генеральным директором Цыганковой Галиной Юсифовной и главным врачом Барбарашом Леонидом Семёновичем.

К 1994 году в ККД создана основная диагностическая и лечебная база. Это

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Дипломный проект					Лист
										10
Изм.	Лист	№ докум.	Подп.	Дата						

амбулаторная служба (многопрофильная районная и специализированная кардиологическая поликлиника), диагностические отделения (функциональной диагностики, ультразвуковых исследований, лучевой диагностики, клиническая лаборатория и др.) и стационарные отделения (острой коронарной патологии, общей кардиологии, реабилитационное отделение, отделения сердечно-сосудистой хирургии и реанимации). В составе кардиодиспансера активно развивались хозрасчетные структуры, мобильный кардиологический диспансер, гараж, гостиница и пр.

В этот же период началось развитие научно - производственной базы, открыты экспериментальная лаборатория, производство биопротезов клапанов сердца и сосудов. В 2001 году создается Государственное учреждение «Научно-производственная проблемная лаборатория реконструктивной хирургии сердца и сосудов Сибирского Отделения Российской академии медицинских наук» (ГУ НППЛ РХСС СО РАМН).

В августе 2005 года введен в эксплуатацию 12-ти этажный госпитальный корпус ККД, что увеличило количество стационарных коек с 142 до 172. Открылись отделение детской кардиологии, неврологическое, нейрохирургическое, значительно увеличились объемы работы отделений сердечно-сосудистой хирургии и рентгенхирургических методов диагностики и лечения.

С 2006 года ККД становится главным звеном медицинского комплекса «Кузбасский кардиологический центр» совместно с ГУ НППЛРХСС СО РАМН и производством биопротезов (ЗАО «Неокор»), обеспечивающий единый технологический цикл оказания помощи пациентам при сердечно-сосудистых заболеваниях. Центр стал базой кафедры кардиологии и сердечно-сосудистой хирургии КемГМА.

В декабре 2008 года ГУ НППЛРХСС СО РАМН реорганизуется в Научно-исследовательский институт комплексных проблем сердечно-сосудистых заболеваний Сибирского отделения РАМН, с большим научным потенциалом и хорошей лечебно-диагностической базой.

В 2010г. Кемеровская область вошла в федеральную программу "Совершенствование оказания медицинской помощи больным с острой сосудистой патологией". В рамках реализации этой программы создан 1 региональный сосудистый центр (РСЦ) и 3 первичных сосудистых центра (ПСО). Базой РСЦ стал МУЗ

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div style="text-align: center; font-size: 1.2em; font-weight: bold;">Дипломный проект</div>					Лист
										11
Изм.	Лист	№ докум.	Подп.	Дата						

"ККД". РСЦ - координирующий головной центр в регионе, оказывающий высокотехнологичную помощь больным с сосудистыми заболеваниями. Созданы отделения для лечения больных с острым нарушением мозгового кровообращения и острым коронарным синдромом.

## 1.2 Организационная структура предприятия

На верхнем уровне декомпозиции в составе предприятия можно выделить следующие группы работников:

- а) врачебный состав;
- б) обслуживающий персонал;
- в) административная служба.

Обслуживающий и административный персонал организован стандартным для большинства государственных предприятий здравоохранения, поэтому не представляют большого интереса для нашего исследования. Наоборот лечебная деятельность Кузбасского Кардиоцентра (далее ККЦ) и будет являться основной целью исследования организационной структуры предприятия. Итак, основные подразделения предприятия, занимающиеся лечебной деятельностью, можно отобразить на схеме.

## 1.3 Основные подразделения предприятия

### 1.3.1 СО РАМН (ФГБУ "НИИ КПССЗ"СО РАМН)

Учреждение (полное название "Научно - исследовательский институт комплексных проблем сердечно-сосудистых заболеваний") создано с целью получения на основе фундаментальных и прикладных исследований новых и углубления имеющихся знаний в области кардиологии, ангиологии и сердечно-сосудистой хирургии, направленных на сохранение и укрепление здоровья человека, развитие здравоохранения и медицинской науки, подготовку высококвалифицированных научных и медицинских кадров.

Основные функции подразделения:

- а) проведение фундаментальных и прикладных исследований;
- б) разработка и апробация заменителей элементов сердечно-сосудистой системы на основе биологических тканей, новых медицинских технологий лечения, диагностики и профилактики;

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div style="text-align: center; font-size: 24px; font-weight: bold;">Дипломный проект</div>					Лист
										12
Изм.	Лист	№ докум.	Подп.	Дата						

в) осуществление медицинской деятельности.

### 1.3.2 Кемеровский Кардиологический Диспансер МБУЗ ККД

Основные функции - предоставление населению медицинских услуг (лечения). В составе подразделения находится множество отделов, среди которых можно выделить поликлинику, научно-медицинские центры, а также стационар ККЦ, речь о котором пойдет чуть ниже.

### 1.3.3 Кафедра кардиологии

Основные функции: объединение терапевтических и хирургических аспектов преподавания для обучения специалистов с комплексным подходом к ведению пациентов с сердечно-сосудистой патологией.

### 1.4 Подразделение связанное с предметной областью

Поскольку цель нашей разработки является создание автоматизированной системы мониторинга пациентов с ВПС, рассмотрим подразделение, которое занимается этим вопросом.

Данным подразделением является Отделение детской кардиологии, которое входит в состав Стационара ККЦ.

Центр детской кардиологии функционально объединяет стационарное и поликлиническое звено. Основным направлением деятельности центра является диагностика и подготовка к хирургическому лечению врождённых пороков сердца у детей.

Для лечения детей с врождёнными пороками сердца используются современные методики: выполнение операций на открытом сердце в условиях искусственного кровообращения и эндоваскулярные малоинвазивные методики.

В ходе операций на открытом сердце устраняются врождённые пороки сердца с преполнением малого круга кровообращения (дефект межжелудочковой перегородки, дефект межпредсердной перегородки без чётких краёв, атриовентрикулярная коммуникация), «синие» пороки (тетрада Фалло). Среди эндоваскулярных вмешательств используются методики закрытия дефекта межпредсердной перегородки, открытого артериального протока системой «Amplatzer».

В ходе работы центра постоянно происходит ротация врачебного персонала,

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										13				
										Изм.	Лист	№ докум.	Подп.	Дата

что позволяет наблюдать пациента с момента обращения в клинику и до момента оказания хирургической коррекции, а так же осуществлять динамическое наблюдение в периоде реабилитации.

Отделение рассчитано на 25 пациентов. Практическая работа осуществляется 10 сотрудниками. В штатах 4 врача детских-кардиологов, из которых 1 имеет высшую категорию, 1 вторую квалификационную категорию, 6 медицинских сестёр, 3 с высшей квалификационной категорией, 2 с первой.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										14
Изм.	Лист	№ докум.	Подп.	Дата						

## 2 Существующие бизнес-процессы

Для анализа предметной области выявим все процессы, связанные с лечением пациентов с ВПС и отобразим их на IDEF0 диаграмме.

### 2.1 Система мониторинга как процесс

Входным объектом существующей в настоящее время системы мониторинга является сам пациент. На выходе системы врачи выдают медицинское заключение о состоянии здоровья пациента.

В процессе мониторинга в настоящее время используются всевозможные лабораторные анализы, а также дневник наблюдения (который ведут родители или опекуны пациента). В качестве оборудования также используются персональные компьютеры на которых ведется база данных пациентов (представляет собой файл электронной таблицы Excel). Следят за процессом мониторинга лица, назначенные руководством кардиоцентра и другие государственные служащие. Общая схема процесса мониторинга приведена в приложении А.1.

### 2.2 Система мониторинга как совокупность процессов

Как правило процесс мониторинга пациентов с ВПС (как и многие другие виды лечений) начинается с предварительного приема (приложение А.2). Прием проводится в учреждении здравоохранения по месту жительства - это позволяет к моменту приема непосредственно в кардиоцентре иметь некоторую медицинскую информацию (результаты анализов, самостоятельные наблюдения пациента) и соответственно разгрузить персонал и оборудование ККЦ от большой входной нагрузки, сконцентрировавшись на основной своей деятельности.

Во время врачебного приема родители пациента передают медицинскую информацию (как правило это результаты наблюдения за его состоянием) словесно, а также в виде дневника наблюдения.

### 2.3 Амбулаторный педиатрический прием

Одним из трудоемким для обеих сторон процессов является периодический амбулаторный прием по месту жительства (приложение А.3).

Данный процесс состоит из 3 стадий. Сперва больной записывается на прием к врачу. Во время записи родители пациента вносят записи о результатах

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	
Изм.	Лист
№ докум.	Подп.
Дата	

Дипломный проект

Лист  
15

своих наблюдений. Далее больной приходит на прием к врачу. Врач по итогам осмотра выдает заключение и направляет пациента на сдачу медицинских анализов. На основании анализов либо проводится либо повторное обследование, либо выдается расширенное направление на кардиобследование.

2.4 Заключительная стадия мониторинга

После кардиологического обследования пациента, врачи проводят анализ полученной медицинской информации. На основании сделанных выводов врачи составляют медицинское заключение и выдают рекомендации родителям и лечащим врачам. Данная информация сообщается пациенту на заключительном осмотре.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										16				
										Изм.	Лист	№ докум.	Подп.	Дата

Копировал

Формат А4



### 3 Проблемы

В существующем бизнес-процессе существует ряд недостатков которые снижают эффективность процесса лечения.

Процесс лечения и мониторинга детей с ВПС является достаточно длительным, сроки измеряются годами. Обусловлен такой длительный период многими факторами, рассмотрим основные из них.

#### 3.1 Задержка с операционным вмешательством

Лечение врожденного порока сердца возможно только с помощью операционного вмешательства, которое может задерживаться. Основной причиной задержки является денежный вопрос, потому что операции детей с ВПС достаточно дорогостоящие (средняя стоимость открытой операции на сердце — 236 000 рублей<sup>1)</sup>). Важно вести постоянный контроль за состоянием пациента в дооперационный период.

#### 3.2 Наблюдение в послеоперационный период

Наблюдение в послеоперационный период очень важно из-за рисков осложнений и возможности повторных операционных вмешательств.

#### 3.3 Расстояние

Не в каждом городе есть специализированная клиника для лечения детей с ВПС. Из-за задержки с операцией необходимо либо переезжать в другой город для того чтобы лечащий врач мог контролировать состояние ребенка, либо периодически приезжать на осмотр. Тот и другой способы достаточно затратны, и к тому же могут негативно сказаться на состоянии ребенка.

#### 3.4 Взаимодействие

В дооперационный и послеоперационный период наблюдение за состоянием ребенка ведет как правило кардиолог по месту жительства, а операцию проводит уже другой врач-хирург. Как правило хирург и кардиолог непосредственно не контактируют друг с другом. Предоставление возможностей общаться и делиться информацией о пациенте в между хирургом и кардиологом в процессе лечения

<sup>1)</sup> <http://www.pomogi.org/projects/heart>

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										17				
										Изм.	Лист	№ докум.	Подп.	Дата

позитивно скажется на процессе реабилитации и лечения.

3.5 Анализ, прогнозирование, тенденции

Выше было сказано что процесс лечения достаточно длителен. Важно хранить всю историю лечения в одном месте с возможностью простого доступа к ней.

3.6 Лечение в стационаре

Длительное пребывание пациента в стационаре снижает его социальные навыки - ребенок остается без общения со сверстниками, много времени проводит внутри помещения, затрудняется активное времяпрепровождение (если оно возможно). Также происходит отрыв ребенка от образовательного процесса, что очень влияет на его дальнейшие жизненные достижения. В связи с этим, важно свести реабилитационный период к минимуму.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										18
Изм.	Лист	№ докум.	Подп.	Дата						

## 4 Цели

#### 4.1 Постоянный мониторинг состояния пациента

Постоянный мониторинг позволит получать наиболее актуальную информацию о состоянии пациента в процессе лечения и во время реабилитационного периода. Так же важно организовать ненавязчивый мониторинг в течении повседневной жизни пациента. Рассмотрим основные направления мониторинга которые будут охвачены в системе.

#### 4.1.1 Амбулаторное наблюдение

Система должна позволять пациентам в добровольном порядке и в ненавязчивой форме предоставлять данные о состоянии своего здоровья. Так как данные будут приходить в систему из внешних незащищенных источников - необходимо обеспечивать максимальную защищенность каналов передачи данных.

### 4.1.2 Наблюдение в стационаре

Необходимо организовать круглосуточное наблюдение за больными, помещенными в специально оборудованное медицинское учреждение. В систему должны поступать данные:

- а) с медицинских устройств;
- б) данные по результатам обследования;
- в) данные по результатам приемов и обходов.

### 4.1.3 Постоянный анализ получаемых данных

Недостаточно просто хранить все данные в процессе лечения и возлагать ответственность за их обработку на врача. Необходимо организовать обработку данных в автоматическом режиме. Это позволит снизить нагрузку на врача и повысить его эффективность в процессе лечения. Реализация автоматической обработки диагностических данных - достаточно сложный процесс, поэтому ограничимся следующими направлениями в анализе данных:

- а) оценка эффективности лечения:
  - 1) оценка влияния лекарственных препаратов;
  - 2) оценка влияния процедур.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>щенными в специально оборудованное медицинское учреждение. В систему должны поступать данные:</p> <ul style="list-style-type: none"> <li>а) с медицинских устройств;</li> <li>б) данные по результатам обследования;</li> <li>в) данные по результатам приемов и обходов.</li> </ul> <p><b>4.1.3 Постоянный анализ получаемых данных</b></p> <p>Недостаточно просто хранить все данные в процессе лечения и возлагать ответственность за их обработку на врача. Необходимо организовать обработку данных в автоматическом режиме. Это позволит снизить нагрузку на врача и повысить его эффективность в процессе лечения. Реализация автоматической обработки диагностических данных - достаточно сложный процесс, поэтому ограничимся следующими направлениями в анализе данных:</p> <ul style="list-style-type: none"> <li>а) оценка эффективности лечения: <ul style="list-style-type: none"> <li>1) оценка влияния лекарственных препаратов;</li> <li>2) оценка влияния процедур.</li> </ul> </li> </ul>
Изм.	Лист	№ докум.	Подп.	Дата	<div style="text-align: center;"> <h2>Дипломный проект</h2> </div>
					Лист 19

б) полный жизненный цикл процесса лечения.

#### 4.1.4 Постоянное взаимодействие пациента с врачом

Для повышения эффективности лечения пациента необходимо снизить издержки со стороны пациента и врача на процесс общения и обмена информации между ними. Основным видом взаимодействия пациента и врача является личный прием у врача. Такая форма взаимодействия наиболее эффективна и привычна с социальной и профессиональных точек зрения, но она не всегда приемлема. В некоторых ситуациях, когда доктору или пациенту важно лишь уточнить некоторые детали, лучше организовать более простую форму взаимодействия между ними. Упрощенными формами личного приема у врача могут являться:

- а) интернет-прием - процесс представляющий из себя обычный прием у врача организованный по средствам сети Интернет;
- б) online-консультация - процесс получения интересующих пациента сведений у специалиста в определенной области или консультанта.

Введение данных видов взаимодействия позволит в значительной мере сократить нагрузку на врача и снизить временные и денежные издержки для пациента.

#### 4.1.5 Взаимодействие между врачами

В процессе лечения пациента принимает участие широкий круг специалистов. Каждый специалист должен иметь возможность получить в кратчайшие сроки информацию о:

- а) текущем состоянии пациента;
- б) заключениях других докторов;
- в) обследованиях и лекарственных препаратах назначенных пациенту.

Своевременное получение актуальной информации позволит более эффективно организовать процесс лечения, за счет снижения временных затрат как пациента, так и доктора.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										20				
										Изм.	Лист	№ докум.	Подп.	Дата

## 5 Требования

### 5.1 Данные

**Все в одном месте.** Система должна обеспечивать доступность всех необходимых данных для лечащего врача. Данная возможность позволит сократить время приема у врача и количество приемов, т.к. связующим звеном между врачами станет не пациент с карточкой, а система с набором всех данных необходимы для принятия дальнейших решений по процессу лечения.

Данные к которым система должна обеспечивать непосредственный доступ:

а) данные о пациенте:

- 1) данные обследований;
- 2) назначенное лечение;
- 3) самочувствие;
- 4) расписание приемов;
- 5) расписание операций.

б) справочные данные:

- 1) справочники;
- 2) законодательные акты;
- 3) словари.

в) данные о системе:

- 1) очереди на прием;
- 2) очереди на обследование;
- 3) наличие лекарственных препаратов;
- 4) наличие оборудования.

### 5.2 Интерфейс

**Эргономичность.** Интуитивный интерфейс должен обеспечивать простое взаимодействие с системой без организации специальной учебной программы по пользованию системой для врача и пациента.

Независимость от устройств с помощью которых врач и/или пациент получают доступ к системе.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										21				
										Изм.	Лист	№ докум.	Подп.	Дата



Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Формат А4

переработке существующего фугкционала.

**Минимум зависимостей.** Любая информационная система состоит из большого числа компонентов. Важно чтобы связи между компонентами были минимальны. Выполнение данного условия позволит сделать систему более независимой от конкретных технологий и технических решений.

5.4 Технологии

Надежность - способность системы сохранять работоспособность при нормальных условиях эксплуатации.

Доступность - возможность свободного (разумеется, при наличии необходимых прав доступа к системе) получения требуемой услуги

Актуальность - соответствие функциональности системы современным требованиям предполагаемой целевой аудитории

Поддержка - необходима дистанционная поддержка пользователей по вопросам возникшим в результате работы системы. Данное требование должно быть обязательно к исполнению в контексте предметной области (некоторые медицинские процессы не требуют отлагательства). Также желательна возможность относительно оперативного добавления или изменения текущего функционала системы.

Открытость - открытый доступ к системе, заключающийся в соблюдении международных и национальных стандартов в области используемых информационных технологий с целью свободного взаимодействия программных приложений, данных, персонала и пользователей системы.

Низкая стоимость - при исполнении данного требования желательно использование программного обеспечения с открытым исходным кодом. Аппаратное обеспечение должно без проблем поддерживать озвученные выше требования к системе, поэтому для снижения расходов предпочтительно привлечение спонсоров.

Функциональность (специфика бизнеса, стратегические приоритеты, географическая распределенность и т.д.)

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

					Дипломный проект	Лист
						24
Изм.	Лист	№ докум.	Подп.	Дата		





а) расписание:

- 1) время приема лекарств;
- 2) даты обследований;
- 3) даты приемов у врача;

б) регистрация в системе - процесс регистрации в системе состоит из следующих этапов:

- 1) заполнение и подача электронной заявки на регистрацию. Подать заявку (на данном этапе анализа моделирования) могут только пациенты, проходящие лечение в Кузбасском кардиоцентре. В заявке необходимо указать ФИО пациента и его матери (отца или опекуна), номер сотового телефона (для отправки на него аутентификационных данных), номер медицинской карточки, придуманный пользователем пароль;
- 2) получение отказа или подтверждение на регистрацию в системе. В случае успешной регистрации пользователь получит аутентификационные данные для входа в систему. В аутентификационные данные входить сгенерированный логин и оставленный пользователем пароль.

в) ввод показателей о состоянии здоровья согласно расписанию составленному лечащим врачом пациента. Список показателей для мониторинга также составляется врачом индивидуально для каждого пациента;

г) электронная запись - процесс записи пациента к врачу, на обследование, процедуры и другие услуги предоставляемые лечащим заведением с целью получать актуальные данные о состоянии здоровья пациента и оперативно вносить изменения в процесс лечения или мониторинга;

### 5.5.2 Доктор

Не менее важной ролью в системе является роль доктора. Пользователи с данной ролью могут иметь доступ к медицинским данным тех пациентов, которые к ним приписаны, просматривать их расписания, возможность связаться с пациентом для оказания консультации по тому или иному вопросу. Специфичными конкретно для роли доктора являются доступ к расписанию приема доктора,

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										26
Изм.	Лист	№ докум.	Подп.	Дата						

медицинским заключениям, выпискам лекарств.

Основные варианты использования:

- а) ведение персональной медицинской карты:
  - 1) заполнение данных о приеме;
  - 2) заполнение рекомендательных данных по улучшению состояния пациента или поддержания состояния в устойчивом положении;
  - 3) агрегирование и анализ результатов исследований, обследований.
- б) просмотр информации о пациентах - возможность врача получить доступ к любой необходимой, имеющейся в базе, информации о ведомом пациенте (или другом пациенте с его согласия) для анализа состояния пациента;
- в) назначение обследований;
- г) назначение лекарственных препаратов.

### 5.5.3 Менеджер

Данная роль является связующей (в плане управления) всех участников системы. У пользователей с данной ролью есть доступ к персональным немедицинским данным пациентов и докторов, заявкам на регистрацию, электронной очереди, некоторой общестатистической информации.

Основные варианты использования:

- а) оперативное управление системой - мониторинг состояния системы и консультация пользователей по работе с ней;
- б) рассмотрение заявок на регистрацию - обработка всех поступающих заявок и принятие решения о подтверждении или отклонении заявки;
- в) изменение электронной очереди на прием;
- г) оценка эффективности лечения;
- д) оценка качества лечения.

### 5.5.4 Электронный (интернет) прием

Является общим (кооперирующим) функционалом для пациентов и докторов. Необходим в случае невозможности пациента явиться на личный прием к врачу, а также если сам врач не может лично посетить пациента, например в случае отъезда того или иного участника приема. Должна быть возможность провести

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										27				
										Изм.	Лист	№ докум.	Подп.	Дата

удаленный прием с фиксацией всех данных полученных в результате приема. Различие в использовании данной функциональной возможности состоит в том, что пациент передает врачу и системе свои медицинские данные и получает медицинское заключение, а врач наоборот, на основании данных пациента выписывает лекарства и выдает заключение.

### 5.5.5 Интернет-консультация

Интернет-консультация должна сократить нагрузку на лечащего врача, за счет делегирования части обязанностей на консультантов. В случае если консультант не может помочь пациенту, пациента можно отправить на интернет-прием или на личный прием к врачу.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										28
Изм.	Лист	№ докум.	Подп.	Дата						

## 6 Готовые решения

Тема разработки программного обеспечения и информационных систем для медицинских учреждений в последнее время получила большое распространение. Многие разработчики решают начать делать свой так называемый “стартап”, также часто можно встретить предложения от ИТ-компаний.

Существующие решения можно разделить на несколько основных классов.

## 6.1 Решения на базе системы 1С:Предприятие

### 6.1.1 1С Медицина Поликлиника

Данное решение<sup>1)</sup> предназначено для автоматизации деятельности медицинских организаций различных организационно-правовых форм, оказывающих медицинскую помощь в амбулаторно-поликлинических условиях. Программный продукт служит для ведения взаиморасчетов с контрагентами, управления потоками пациентов, персонифицированного учета оказанной медицинской помощи.

### 6.1.2 1С Рарус Амбулатория

Данный продукт<sup>2)</sup> комплексно автоматизирует деятельность медицинского учреждения. Помимо глубоко реализованной системы автоматизации документооборота, хотелось бы отметить характерное для мира 1С систем наличие реестров и справочников служебной медицинской информации, например, Банк Стволовых Клеток «КриоЦентр».

## 6.2 Решения для автоматизации медицинского документооборота

Комплексная медицинская информационная система (КМИС). Уменьшает затраты доктора на ведение документации связанной с приемом пациентов, выдачей направлений и т.д. Медицинская информационная система AKSi-офис<sup>3)</sup> (на базе системы Microsoft Office). Программное обеспечение от фирмы ТрастМед - аналогичный функционал.

<sup>1)</sup> [http://www.v8.1c.ru/solutions/product.jsp?prod\\_id=149](http://www.v8.1c.ru/solutions/product.jsp?prod_id=149)

<sup>2)</sup> <http://rarus.ru/press/publications/126187/>

<sup>3)</sup> [http://www.aksimed.ru/products/aksi\\_line/AKSi-Office.php](http://www.aksimed.ru/products/aksi_line/AKSi-Office.php)

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>Данный продукт<sup>2)</sup> комплексно автоматизирует деятельность медицинского учреждения. Помимо глубоко реализованной системы автоматизации документооборота, хотелось бы отметить характерное для мира 1С систем наличие реестров и справочников служебной медицинской информации, например, Банк Стволовых Клеток «КриоЦентр».</p>
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p><b>6.2 Решения для автоматизации медицинского документооборота</b></p>
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>Комплексная медицинская информационная система (КМИС). Уменьшает затраты доктора на ведение документации связанной с приемом пациентов, выдачей направлений и т.д. Медицинская информационная система AKSi-офис<sup>3)</sup> (на базе системы Microsoft Office). Программное обеспечение от фирмы ТрастМед - аналогичный функционал.</p>
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p><sup>1)</sup> <a href="http://www.v8.1c.ru/solutions/product.jsp?prod_id=149">http://www.v8.1c.ru/solutions/product.jsp?prod_id=149</a></p> <p><sup>2)</sup> <a href="http://rarus.ru/press/publications/126187/">http://rarus.ru/press/publications/126187/</a></p> <p><sup>3)</sup> <a href="http://www.aksimed.ru/products/aksi_line/AKSi-Office.php">http://www.aksimed.ru/products/aksi_line/AKSi-Office.php</a></p>
Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>Дипломный проект</p>
Изм.	Лист	№ докум.	Подп.	Дата	<p>Лист</p>
					<p>29</p>

### 6.3 Комплексная автоматизация медицинского предприятия

В первую очередь хотелось бы отметить отечественную разработку - Медицинская информационная система AKSi-клиника от АКСИМЕД. Среди ее основных функций хотелось бы отметить следующие:

- а) комплексная автоматизация всех процессов наблюдения, диагностики и лечения амбулаторных и стационарных пациентов;
- б) эффективное управление персоналом, ресурсами и финансово-экономической деятельностью ЛПУ, автоматизация медико-статистического контроля и планирования;
- в) однократный ввод информации в электронную историю болезни (электронную медицинскую карту) пациента с последующим многократным использованием этих сведений и поддержкой принятия врачебных решений;
- г) сквозная компьютеризация работы регистратуры, поликлиники, стационара, отделения скорой медицинской помощи, стоматологических кабинетов и других подразделений ЛПУ;
- д) обеспечение безопасности персональных данных в соответствии с Федеральным законом от 27 июля 2006 г. № 152-ФЗ.

Также существуют множество зарубежных решений. Среди них можно упомянуть систему разработанную и используемую в США - Practicefusion, чей девиз “Больше пациентов - меньше работы”.

### 6.4 Выбор готового решения

Среди всех рассмотренных выше систем можно выявить общую тенденцию - ИТ-компании предлагают в первую очередь автоматизацию медицинского документооборота. Некоторые системы предлагают анализ и диагностику, но она заточена под широкое использование.

Разработанная нами система позволяет решить поставленные в начале исследования проблемы, а именно автоматизированный дистанционный (с определенной степенью) сбор медицинской информации, мониторинг (проведение какого-либо анализа над собранными), позволяет проводить коммуникацию между пациентами.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										30				
										Изм.	Лист	№ докум.	Подп.	Дата

Разумеется озвученные нами возможности реализованы в существующих ныне системах в том или ином виде. Здесь нужно отметить, что рассмотренные выше программные средства и системы стоят больших денег и медицинское учреждение может сэкономить внедряя нашу систему именно на том что наш система реализует конкретные возможности (необходимые прежде всего для лечения больных с ВПС), а не внедряя большой пакет возможностей, многие из которых могут никогда не пригодится.

Кроме того использование широко тиражированного программного обеспечения может поставить в зависимость от решений фирмы-разработчика, что может оказать нежелательным для предприятия со столь ответственной деятельностью. Наша же система готова к дальнейшим изменениям потенциального заказчика, так разрабатывается скорее для конкретных учреждений, а не для массовой реализации.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div> <div>Лист 31</div>				
Изм.	Лист	№ докум.	Подп.	Дата					

## 7 Корректировка бизнес-процессов

Рассмотрим процесс мониторинга в контексте будущей системы. Общая схема мониторинга представлена в приложении Г.1.

### 7.1 Составляющие процесса мониторинга

Единственным объектом исследования в системе является - обследуемый пациент. Система должна вести постоянный мониторинг и анализ состояния пациента.

Для получения данных о пациенте и их анализа могут быть использованы:

- а) Медицинские устройства. К ним относятся аппараты, расположенные в лечебном учреждении и находящиеся в общем доступе для всех пациентов. К ним можно отнести рентген-установку, МРТ-сканер, УЗИ, тонометры, термометры, пульсметры и другие.
- б) Персональные устройства мониторинга. К ним относятся приборы, доступные для использования в домашних условиях, а именно: электронные тонометры и термометры. Помимо этих приборов существуют так называемые комплексные датчики предназначены для пользователей, не являющихся специалистами в области сердечно-сосудистой диагностики. К таким приборам относится Ангиоскан-01М (Персональная версия) – предназначен для работы под управлением персонального компьютера. Данный прибор надевается на палец пациента и устанавливает подключение к персональному компьютеру или ноутбуку. Прибор позволяет измерять следующие показатели: частоты сердечных сокращений; жесткости сосудов; типа пульсовой волны; биологического возраста сосудов; индекса сатурации (насыщение гемоглобина кислородом); уровня стресса.
- в) Сервер с необходимым программным обеспечением. На нем будет развернута база данных, в которой будут храниться персональные данные всех участников системы, медицинская информация пациентов и прочие информационные объекты, которые будут определены ниже, в разделе Концептуальная модель предметной области. Также на сервере будет находиться веб-интерфейс системы, доступный пользователям.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										32				
										Изм.	Лист	№ докум.	Подп.	Дата



Процесс мониторинга должен соответствовать определенным стандартам и законодательным актам.

Контролировать процесс должен лечащий врач или врач, непосредственно, осуществляющий оказание той или иной услуги пациенту.

Результаты процесса мониторинга представляются в виде различного рода отчетов.

## 7.2 Основные этапы процесса мониторинга

В приложении Г.2 приведена декомпозиция процесса мониторинга. Рассмотрим каждый подпроцесс более подробно.

### 7.2.1 Регистрация в системе

Этап предназначен для создания учетной записи пациента при обращении в данное лечебное учреждение впервые (приложение Г.3). С данной учетной записью будут соотноситься данные, полученные в процессе лечения, обследований. Важно чтобы продолжительность данного этапа была минимальной, а процедура регистрации максимально простой, чтобы процесс обследования пациента начался максимально быстро.

Возможны два варианта регистрации пациента в системе:

- а) самостоятельная регистрация (для иногородних пациентов);
- б) регистрация при посещении лечебного учреждения.

После прохождения процедуры регистрации пациент может быть записан на первичный прием к врачу.

### 7.2.2 Первичное обследование

На первичном приеме (приложение Г.4) врач формирует электронную карту обследований, которые необходимо пройти пациенту для оценки состояния здоровья. Факторами влияющими на набор обследований который должен пройти пациент являются:

- а) Устные показания пациента
- б) Больничная карта пациента
- в) Опыт врача

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div style="text-align: center; font-size: 24px; font-weight: bold;">Дипломный проект</div>					Лист
										33
Изм.	Лист	№ докум.	Подп.	Дата						

Во время первичного приема у врача начинается непосредственный постоянный мониторинг за состоянием пациента.

После составления карты обследования пациент проходит первичное обследование. Первичное обследование необходимо для формализации состояния пациента на момент обращения в лечашее учреждение. Оценка состояния пациента, полученная в результате первичного обследования будет учитываться в показателях оценки эффективности лечения.

### 7.2.3 Лечение

После прохождения пациентом первичного обследования формируется план лечения пациента. План лечения пациента может быть многоэтапным. После завершения каждого этапа происходит оценка эффективности лечения.

План лечения на каждом этапе может включать в себя:

- а) режим дня пациента;
- б) режим приема лекарственных препаратов;
- в) расписание приемов;
- г) расписание обследований.

Основной задачей системы на данном этапе является отслеживание качественных и количественных показателей состояния пациента. Декомпозиция процесса лечения представлена в приложении Г.5.

### 7.2.4 Мониторинг

Процесс мониторинга (приложение Г.6) является “сквозным” и присутствует на многих этапах. Основной задачей процесса является сбор данных в процессе лечения и обследований пациента. Основными процессами в результате которых в систему попадают данные о пациенте являются:

- а) прием у врача;
- б) стационарное лечение;
- в) обследования;
- г) амбулаторное лечение.

Основными источниками данных о пациенте являются:

- а) результаты обследований;
- б) результаты анализов;

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект	Лист
											34

- в) результаты осмотров у врача;
- г) устные показания пациента;
- д) показания врача.

Основные способы внесения данных в систему:

- а) ручной ввод:
  - 1) ввод данных пациентом;
  - 2) ввод данных врачом.
- б) автоматический ввод данных медицинскими устройствами.

### 7.2.5 Накопление данных

Процесс заключается в сохранении поступающих в систему данных для их последующей обработки и анализа.

### 7.2.6 Анализ данных

Объем данных, поступающих в систему достаточно большой. Для ускорения обработки данных их необходимо анализировать. Согласно требованиям к системе, процесс анализа данных включает в себя:

- а) оценку эффективности лечения:
  - 1) оценки влияния лекарственных препаратов;
  - 2) оценки влияния процедур, операций.
- б) получение отчетов:
  - 1) агрегированные данные;
  - 2) рекомендации по лечению.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										35
Изм.	Лист	№ докум.	Подп.	Дата						

## 8 Анализ предметной области

### 8.1 Концептуальная модель предметной области

Определим основные объекты предметной области и выясним отношения между ними.

#### 8.1.1 Пациент

Сущность отражает личную информацию о реальном пациенте.

Атрибуты:

- а) паспортные данные;
- б) номер полиса;
- в) номер личной больничной карты<sup>1)</sup>;
- г) контактные данные.

#### 8.1.2 Врач

Сущность отражает данные о реальном докторе.

Атрибуты:

- а) паспортные данные;
- б) контактные данные;
- в) профессиональные данные;

#### 8.1.3 Менеджер

Сущность отражает человека контролирующего работу системы.

Атрибуты:

- а) паспортные данные;
- б) контактные данные;
- в) профессиональные данные.

#### 8.1.4 Диагноз

Сущность отражает реальный диагноз согласно “Международной статистической классификации болезней и проблем, связанных со здоровьем” (ICD 10).

<sup>1)</sup> Сложно будет перевести сразу все учреждение на электронные больничные карты поэтому некоторое время обычная больничная карта и электронная будут существовать параллельно.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										36
Изм.	Лист	№ докум.	Подп.	Дата						

Атрибуты:

- а) класс диагноза;
- б) название диагноза.

### 8.1.5 Лекарство

Сущность отражает реальный лекарственный препарат.

Атрибуты:

- а) название лекарства;
- б) побочные эффекты;
- в) время приема;
- г) дозы;
- д) порядок приема.

### 8.1.6 Обследование

Сущность отражает реальное обследование доступное пациентам лечящего учреждения в процессе лечения.

Атрибуты:

- а) суть обследования;
- б) дата обследования;
- в) результат обследования.

### 8.1.7 Прием

Сущность отражает реальный прием у врача. Так как система должна иметь возможность сопровождать два типа приема: обычный и интернет прием, необходимо чтобы набор атрибутов у них был максимально одинаковым. Выполнение данного условия облегчит перевод учреждения на электронный прием.

Атрибуты:

- а) дата приема;
- б) результат приема;
- в) данные сопровождающие прием.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										37				
										Изм.	Лист	№ докум.	Подп.	Дата

8.1.8 Документ

Сущность отражает документ, который врач составляет по результатам медицинского приема. Доступ к документу должен быть только у пациента, на которого составлен данный документ и у его лечащего врача, так как это медицинская информация. Атрибуты:

- а) время создания документа;
- б) содержащаяся информация.

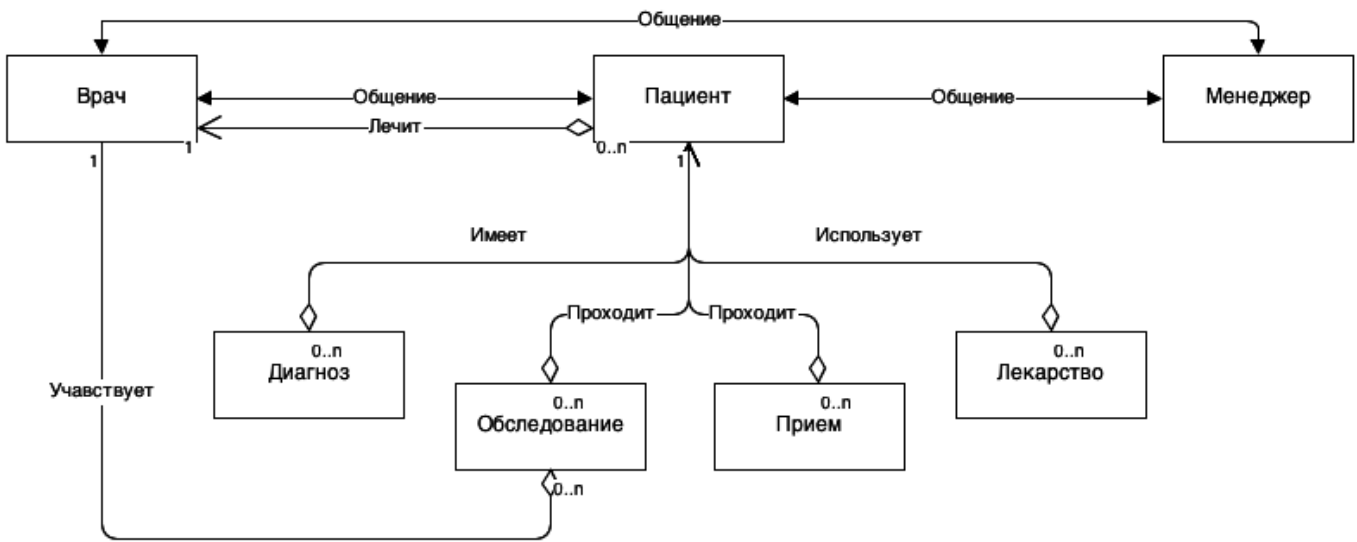


Рисунок 8.1 – Связи между объектами предметной области

Выявленные объекты предметной области (рисунок 8.1), не покрывают всех функциональных требований к системе и не учитывают технических реализаций и алгоритмов построения информационных систем.

8.2 Уточнение объектов предметной области

Объекты “Врач”, “Пациент” и “Менеджер” имеют общий набор атрибутов, который логично вынести в отдельный объект родитель “Пользователь”. Данный шаг упростит процедуры регистрации, авторизации и процесс контроля прав доступа к системе, так как необходимо будет контролировать только один объект вместо трех.

Объекты “Прием” и “Обследование” - это некоторые события. Логичным будет выделить отдельный объект родитель - “Событие”. “Событие” будет связано с пользователем через объект “Календарь” - отражающий определенный этап ле-

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

чения. Каждое событие имеет “Результат”. Событие может быть повторяющимся (прием лекарственного препарата 3 раза в день).

Лекарственные препараты и диагнозы являются справочниками, которые логичным будет наследовать от объекта родителя “Справочник”.

“Справочник” и “Электронная карта” - это документы. Для облегчения работы с ними логичным будет ввести объект родитель “Документ”.

На рисунке 8.2 изображена уточненная модель предметной области.

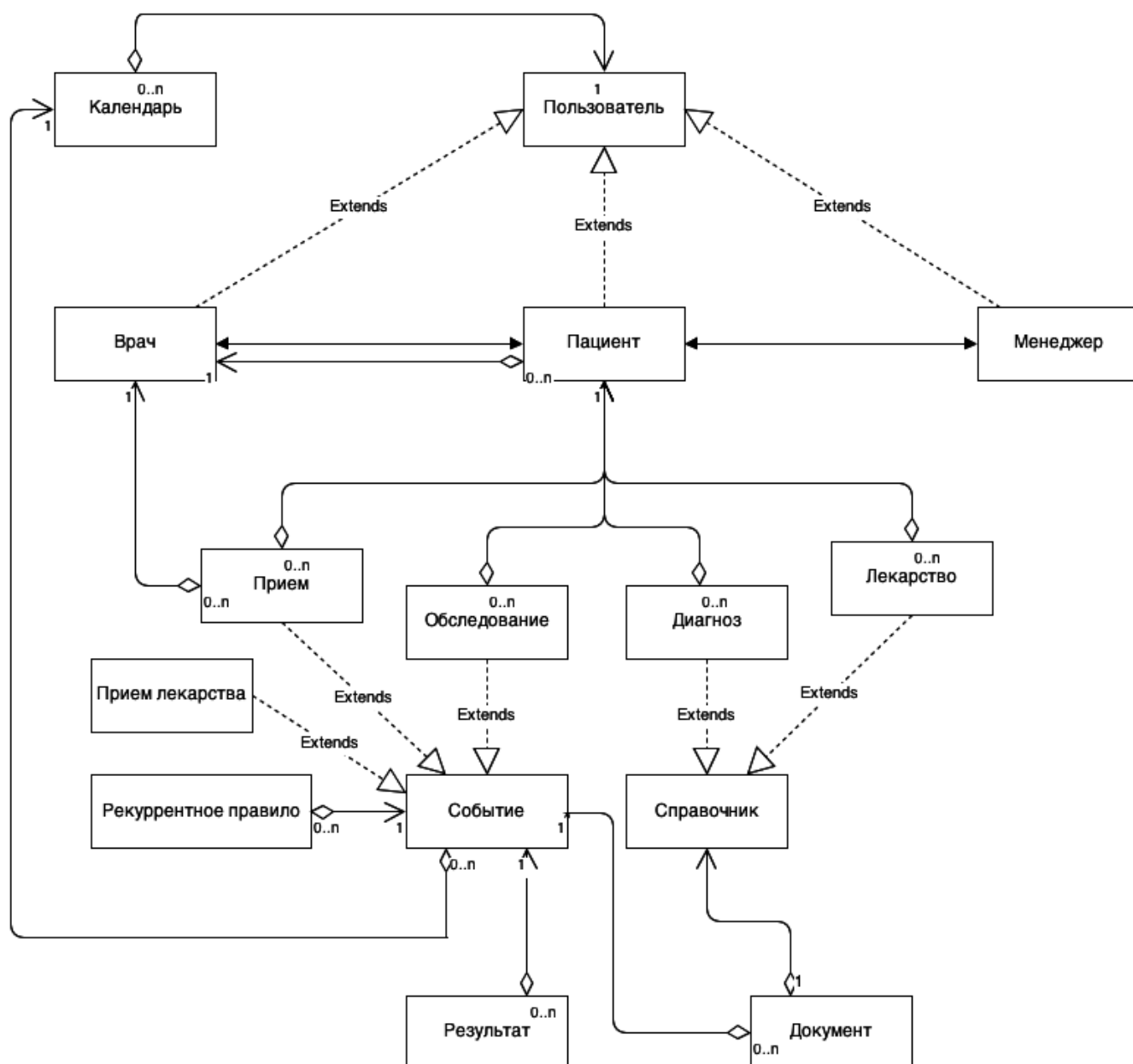


Рисунок 8.2 – Уточненная модель предметной области

## 9 Структура системы

Под структурой системы будем понимать совокупность элементов, составляющих систему, и связи между ними. Ниже рассмотрим основные компоненты системы. Деление в контексте структуры достаточно условное, но оно позволит разграничить области ответственности каждого компонента системы. Разграничение ответственностей позволит упростить структуру системы и снизить связанность элементов системы. Каждая из подсистем должна соответствовать требованиям к системе приведенным выше.

### 9.1 Подсистема ввода данных

Подсистема ввода данных должна соответствовать требованию “доступности на запись” и обеспечивать следующие возможности:

- а) Ручной ввод данных - непосредственно участниками системы, доктором, пациентом, менеджером.
- б) Автоматический ввод данных - получение данных с различных устройств диагностики состояния пациента.
- в) Проверка вводимых данных на корректность. Корректность данных определяется исходя из контекста использования данных и типа данных.
- г) Абстракция. Доступ к источнику данных должен быть через легкозаменяемую абстракцию. Это позволит не зависеть от конкретного поставщика и выполнить требование “гибкости”.

### 9.2 Подсистема доступа к данным

Подсистема доступа к данным должна соответствовать требованию “доступности на чтение” и обеспечивать уровень абстракции от источника данных чтобы соответствовать требованию “гибкости”. Так же должен обеспечиваться доступ как для внешних, так и для внутренних потребителей.

### 9.3 Подсистема хранения данных

Подсистема хранения данных должна обеспечивать:

- а) Целостность данных
- б) Возможность управления доступом к данным
- в) Возможности ускорения доступа к данным

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Дипломный проект					Лист
										40
Изм.	Лист	№ докум.	Подп.	Дата						



г) Возможность компенсировать увеличение нагрузки

Обязательными являются требования к “надежности”, “безопасности” и “масштабируемости”.

#### 9.4 Подсистема анализа данных

Подсистема анализа данных должна обеспечивать возможность анализа данных, поступающих из подсистемы ввода данных. Доступ к данным осуществляется через подсистему доступа к данным. Промежуточные результаты работы могут сохраняться в подсистеме хранения данных. Результаты работы подсистемы анализа данных должны быть представлены в виде двух видов отчетов:

- а) Отчет по запросу
- б) Автоматический отчет

#### 9.5 Подсистема управления доступом

Подсистема управления доступом должна:

- а) Обеспечивать возможность контроля доступа к данным в зависимости от роли пользователя в системе.
- б) Реагировать на попытки несанкционированного доступа к данным.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div style="text-align: center; font-size: 1.2em; font-weight: bold;">Дипломный проект</div>					Лист
										41
Изм.	Лист	№ докум.	Подп.	Дата						

## 10 Проектирование

Выше были выделены основные компоненты системы и разграничены функции и ответственность между ними. Теперь необходимо определиться с конкретной реализацией выбранных компонентов и со схемой взаимодействия между ними. Общая архитектура системы представлена на рисунке 10.1. Ниже будут рассмотрены основные компоненты и описаны их назначения.

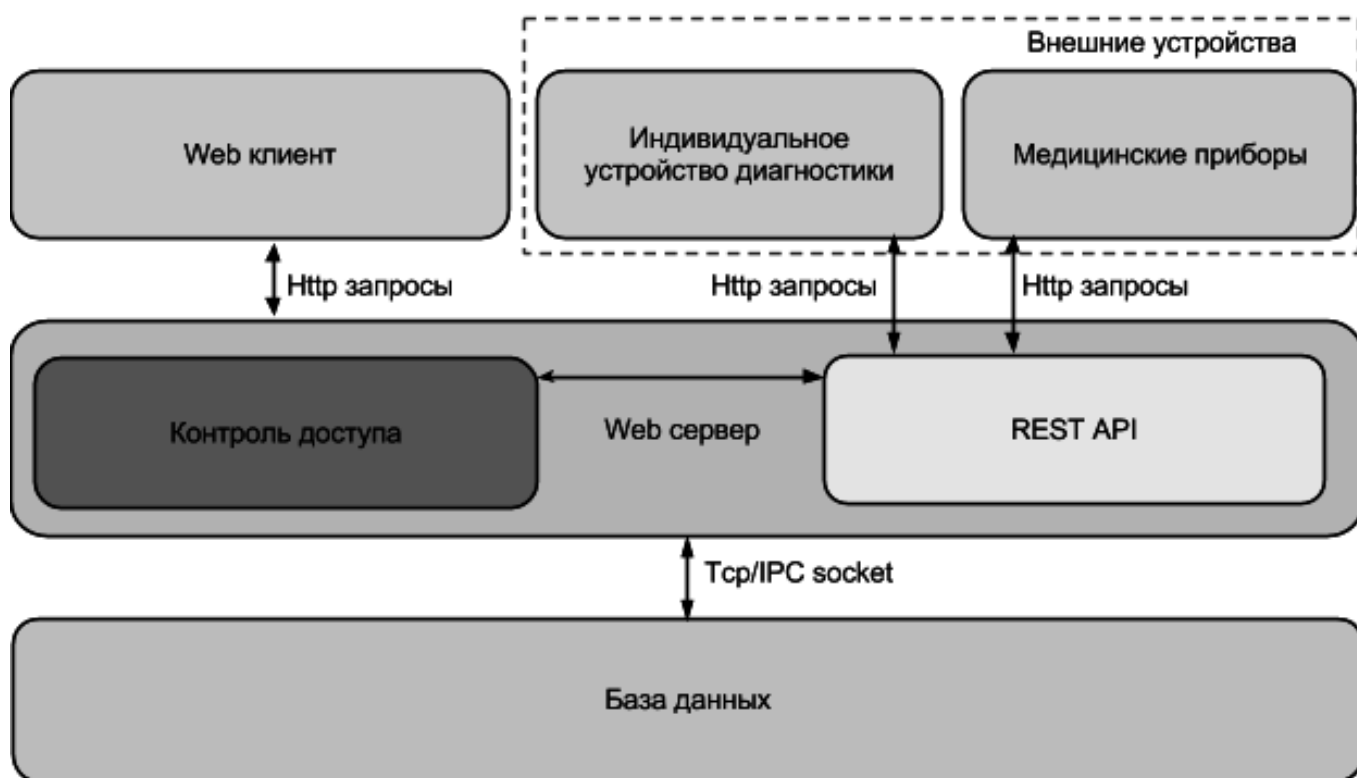


Рисунок 10.1 – Общая архитектура системы.

### 10.1 Web клиент

Клиент разделяет функции подсистемы ввода данных и подсистемы доступа к данным. Основной задачей клиента является предоставление доступа к системе пользователям с помощью веб-браузера или другого программного обеспечения способного работать с протоколом HTTP. С точки зрения требования доступности, реализация в виде web клиента наиболее оптимальна, так как веб-браузеры есть на всех современных платформах и устройствах.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div> <div>Лист 42</div>				
Изм.	Лист	№ докум.	Подп.	Дата					

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Достаточно долгое время основным типом системы хранения данных была SQL база данных. Однако в последние годы получает распространение NoSQL системы хранения. Данные подходы к организации источника данных преследуют одну цель - обеспечить долговременное, надежное хранение структурированных данных. Каждый подход имеет свои преимущества и недостатки. Для того чтобы определиться какого типа будет система хранения данных необходимо сформировать требования к системе, а затем на основании требований выбрать наиболее подходящий тип системы.

#### 10.4.1 Требования к системе хранения данных

Надежность - очень широкое понятие в терминах баз данных. Рассмотрим основные составляющие надежной системы хранения.

**Обеспечение целостности данных.** SQL системы изначально проектировались чтобы соответствовать основным принципам ACID и как следствие предоставляют возможность хранить данные в нормализованном виде, явно определяя связи между элементами базы данных. Однако такой подход несет дополнительную

- |      |      |          |       |      |                  |     |
|------|------|----------|-------|------|------------------|-----|
|      |      |          |       |      | Дипломный проект | Лис |
| Изм. | Лист | № докум. | Подп. | Дата |                  | 43  |

# Дипломный проект

43

## Копировал Формат А4

нагрузку на базу данных, т.к. необходимо контролировать целостность данных а базе. NoSQL решения изначально проектировались как полная альтернатива SQL решениям. Они позволяют хранить данные в максимально денормализованном виде. При таком подходе вся ответственность за целостность данных возлагается целиком на разработчиков.

**Масштабируемость.** При росте числа пользователей базы данных возникает проблема обработки большого числа запросов к базе данных. Данную проблему можно решить за счет горизонтальной или вертикальной масштабируемости ситемы.

При вертикальной масштабируемости предлагается обновлять конфигурацию сервера на более современную для повышения производительности. При таком подходе очевидно что общая производительность ситемы, если не брать в счет програмную составляющую, ограничивается только прогресом в области производства аппаратного обеспечения. Как правило местом преткновения становится скорость операций i/o на жестком диске. Также стоит учитывать что цены на новинки всегда завышены и нецелесообразно будет платить достаточно крупные суммы за повышение производительности на несколько процентов.

При горизонтальном масштабировании предлагается распределять нагрузку на несколько серверов баз данных. При таком подходе не нужно покупать новое дорогостоящее оборудование, производительность не упирается в скорость i/o операций на жестком диске, а производительность системы повышается прямопропорционально числу серверов. Достаточно обеспечить необходимое количество серверов чтобы балансировать нагрузку между ними. На самом деле на этом вопрос масштабируемости не ограничивается, т.к. необходимо учитывать еще один важный фактор - размер базы данных. Некоторые современные базы данных поддерживают механизм партицирования. Данный механизм позволяет разбивать таблицу на несколько частей. В результате чего возможно хранить данные на разных носителях. Данный механиз повышает скорость доступа к данным за счет того что выборка манипуляции с данными происходят не в контексте всей таблицы а в контексте конкретной части таблицы. Не стоит забывать и о выборе файловой системы под файлы базы данных и драйвера который будет управлять распреде-

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										44				
										Изм.	Лист	№ докум.	Подп.	Дата

лением данных в файловой системе.

**Быстродействие.** Скорость работы подсистемы хранения данных непосредственно влияет на продолжительность приема. Важно чтобы доступ к данным был максимально быстрым.

SQL решение накладывает некоторые ограничения. Прежде всего это индексы и транзакции, которые могут заметно снизить скорость вставки данных, но без них может значительно снижаться скорость выборки данных.

NoSQL решение потенциально не имеет проблем со вставкой данных. Теоретически вставка данных должна происходить со скоростью равной скорости записи в оперативную память. Стоит отметить что все современные SQL базы данных производят первичную запись данных так же в оперативную память.

**Выбор между SQL и NoSQL.** Выбор между двумя подходами достаточно сложная задача. В рамках выбранной предметной области система может быть спроектирована как NoSQL так и SQL подходом. Однако SQL подход обеспечивает большую согласованность данных и более простую реализацию. Так же немаловажным фактором в пользу SQL подхода является наличие более развитых средств разработки.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект	Лист
						45

## 11 Выбор технологий

### 11.1 В начале работы

### 11.1.1 Использование языка РНР

Первоначально в качестве языка программирования было решено использовать язык РНР. Это было вызвано низким порогом вхождения, его большой популярностью, а следовательно большим наличием учебных материалов и примеров.

### 11.1.2 Zend Framework

В процессе работы возникла необходимость грамотной организации исходного кода, использование тестов. Так в проект добавилась знаменитая среди разработчиков библиотека Zend Framework. Эта библиотека решила вопрос с организацией кода путем использования паттерна MVC - Model View Controller.

### 11.1.3 Переход на платформу ASP.NET MVC

В процессе разработки стали заметны недостатки Zend Framework: большая избыточность кода (много абстракций вследствие данной реализации концепции MVC), долгое время отклика, необходимость вручную составлять запросы, связывающие модель предметной области с базой данных.

Было решено перенести проект на стек технологий от компании Microsoft. В качестве базовой технологии была выбрана платформа ASP.NET, язык программирования C#. В качестве организации кода решено было продолжить использовать паттерн MVC.

#### 11.1.4 Большие затраты времени на конфигурирование

Вместе с использованием ASP.NET, для доступа моделей предметной области к данным был использован паттерн Repository. Однако это не решило вопрос с ручным созданием объектов в базе данных. Таким образом стало ясно, что без использования технологии ORM не обойтись. С этой целью был использована библиотека Entity Framework.

Несмотря на то что библиотека Entity Framework сильно облегчает и ускоряет работу программиста, регламентирование доступа к базе данных из модели с помощью паттерна Repository заставляет писать дополнительный программный

Изм.	Лист	№ докум.	Подп.	Дата	<p>Дипломный проект</p>	<p>Лист</p> <p>46</p>

код - помимо перечисления атрибутов сущностей в классе, для каждого поля необходимо прописывать атрибуты используемые в процессе работы Entity Framework.

Еще одним минусом стало ручное прописывание валидационных правил и метаданных сущностей. Для организации их работы приходилось вручную прописывать все атрибуты, что вызывает неудобство и повышает вероятность ошибки.

## 11.2 Выбор платформы Ruby on Rails

### 11.2.1 Регламентированный доступ к базе данных

Для доступа к данным в Rails используется ORM с реализацией паттерна ActiveRecord - шаблон проектирования приложений, описанный Мартином Фаулером. Основная суть заключается в том, что для каждой таблицы в БД создается соответствующий ей класс, каждой строке в данной таблице соответствует экземпляр соответствующего ей класса. Каждое действие с экземпляром данного класса (создание, изменение и удаление) сопровождается соответствующими SQL-запросом.

Запросы для выборки данных создаются через Query Interface. Query Interface представляет из себя набор классов, специфичных для каждой СУБД.

```
@appointmentEvents =
    DoctorUser.current.appointment_events.
    where('events.status <> ?', 'free').
    order('date start DESC')
```

## Листинг 1: Запрос через ActiveRecord

В листинге 1 представлен пример запроса, который возвращает все врачебные приемы (appointment\_events) для пользователя доктор (DoctorUser), который на данный момент авторизован (current) в системе, статус которых не свободно (where('events.status <> ?', 'free')) и сортирует их в порядке, в котором первым отображается самый поздний врачебный прием (order("date start DESC")).

Для изменения состава атрибутов сущности в Ruby on Rails используется инструмент мигрирования, который будет рассмотрен ниже.

### 11.2.2 Готовая система валидации вводимых данных

Валидации используются, чтобы быть уверенными, что только верно указанные данные сохраняются в базу данных.

В Ruby on Rails валидация реализуется с помощью predefined валидационных хелперов. Эти хелперы предоставляют общие правила валидации. Каждый раз, когда валидация проваливается, сообщение об ошибке добавляется в коллекцию `errors` объекта, и это сообщение связывается с атрибутом, который подлежал валидации.

Для того чтобы использовать валидационной хелпер, его необходимо вызывать в классе модели и через запятую указать те атрибуты класса, которые необходимо проверить на правильность вводимых данных.

### 11.2.3 Создание связей между сущностями

Связи между моделями нужны для облегчения выполнения обычных операций с объектами. Среда Ruby on Rails позволяет создавать связи типа один к одному, один ко многим и многие ко многим. В рассмотренном выше примере получения всех врачебных приемов доктора использовалась связь `appointment events`.

Для использования связей достаточно в классе сущности указать тип связи и класс сущности с которой создается связь. В качестве примера приведем связь между сущностями “Событие” и “Пользователь”, которая реализуется с целью определения пользователя-создателя события.

```
class Event < ActiveRecord::Base
  # Организатор события
  belongs_to :user
end
```

## Листинг 2: Связь на стороне события

```
class User < ActiveRecord::Base
  # События для которых пациент является организатором
  has_many :events

  # События в которых участвовал пациент
  has_many :attendees_events, :through => :attendees, :source => :event
end
```

### Листинг 3: Связь на стороне события

[illegible]



Как видно из листинга кода, сущность “Event” связывается связью “один ко многим” (belongs\_to на стороне “одного” и has\_many на стороне “многие”) с сущностью “User”.

#### 11.2.4 Использование соглашений по конфигурации

Convention over Configuration<sup>1)</sup> — это принцип построения фреймворков и библиотек, призванный сократить количество требуемой конфигурации без потери гибкости. Обычно переводится как «соглашения по конфигурации». В строгой форме этот принцип можно выразить так: аспект программной системы нуждается в конфигурации тогда и только тогда, когда этот аспект не удовлетворяет некоторой спецификации. В качестве примера можно привести соглашение по именованию таблиц и классов - при формировании названия таблицы имя класса пишется со строчной буквы с добавлением окончанием множественного числа (англ. языка) “s”.

#### 11.2.5 Гибкость языка Ruby

Основное назначение Ruby — создание простых и в то же время понятных программ, где важна не скорость работы программы, а малое время разработки, понятность и простота синтаксиса. Язык следует принципу «наименьшей неожиданности»<sup>2)</sup>: программа должна вести себя так, как ожидает программист.

#### 11.2.6 Вывод

Исходя из требований к системе, оптимальной формой интерфейса системы будет веб-сайт. На данный момент число технологий создания веб-сайтов достаточно велико, у каждой есть свои плюсы и минусы. Исходя из требований к технологиям оптимальным будет выбор фреймворка Ruby On Rails.

Основные преимущества перед другими технологиями того же уровня:

- а) наличие большого числа библиотек, решающих большинство типовых задач при веб-разработке;
- б) большое сообщество;
- в) быстрое развитие;

<sup>1)</sup> [http://en.wikipedia.org/wiki/Convention\\_over\\_configuration](http://en.wikipedia.org/wiki/Convention_over_configuration)

<sup>2)</sup> <http://ru.wikipedia.org/wiki/Ruby>

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>программ, где важна не скорость работы программы, а малое время разработки, понятность и простота синтаксиса. Язык следует принципу «наименьшей неожиданности»<sup>2)</sup>: программа должна вести себя так, как ожидает программист.</p>	
					<p><b>11.2.6 Вывод</b></p>	
					<p>Исходя из требований к системе, оптимальной формой интерфейса системы будет веб-сайт. На данный момент число технологий создания веб-сайтов достаточно велико, у каждой есть свои плюсы и минусы. Исходя из требований к технологиям оптимальным будет выбор фреймворка Ruby On Rails.</p>	
					<p>Основные преимущества перед другими технологиями того же уровня:</p> <ul style="list-style-type: none"><li>а) наличие большого числа библиотек, решающих большинство типовых задач при веб-разработке;</li><li>б) большое сообщество;</li><li>в) быстрое развитие;</li></ul>	
					<p><sup>1)</sup> <a href="http://en.wikipedia.org/wiki/Convention_over_configuration">http://en.wikipedia.org/wiki/Convention_over_configuration</a></p> <p><sup>2)</sup> <a href="http://ru.wikipedia.org/wiki/Ruby">http://ru.wikipedia.org/wiki/Ruby</a></p>	
Изм.	Лист	№ докум.	Подп.	Дата	<p>Дипломный проект</p>	Лист
						49

г) простота и удобство разработки.

### 11.3 Frontend

Front-end - часть программы, которая взаимодействует с пользователем. Здесь мы рассмотрим технологии используемые для построения графического интерфейса.

#### 11.3.1 Backbone.js

Backbone.js придает структуру веб-приложениям с помощью моделей с биндингами по ключу и пользовательскими событиями, коллекций с богатым набором методов с перечислимыми сущностями, представлений с декларативной обработкой событий; и соединяет это все с существующим REST-овым JSON API<sup>1)</sup>.

При использовании backbone.js данные предметной области представляются как Модели (Models), которые могут быть созданы, провалидированы, удалены, и сохранены на сервере. Всякий раз, когда в интерфейсе изменяется атрибуты модели, модель вызывает событие "change"; все Представления (Views), которые отображают состояние модели, могут быть уведомлены об изменении атрибутов модели, с тем чтобы они могли отреагировать соответствующим образом — например, перерисовать себя с учетом новых данных.

Основной полезный эффект возникающий от добавления backbone.js в проект заключается в том, что разработчику не надо писать код, ищущий элемент с определенным id в DOM и обновляющий HTML вручную. При изменении модели представление просто обновит себя самостоятельно.

#### 11.3.2 Coffeescript

Встроенная поддержка CoffeeScript была добавлена в Rails с версии 3.1. Программы написанные на данном языке перед выполнением компилируются в javascript. Язык CoffeeScript позволяет писать программы в функциональном стиле, в нем более полно реализовано использование классов.

CoffeeScript используется чтобы улучшить читаемость кода и уменьшить его размер. В среднем для выполнения одинаковых действий на CoffeeScript требуется

<sup>1)</sup> <http://backbonejs.ru/>

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	моделю, с тем чтобы они могли отреагировать соответствующим образом — например, перерисовать себя с учетом новых данных.
					Основной полезный эффект возникающий от добавления backbone.js в проект заключается в том, что разработчику не надо писать код, ищущий элемент с определенным id в DOM и обновляющий HTML вручную. При изменении модели представление просто обновит себя самостоятельно.
					<b>11.3.2 Coffeescript</b>
					Встроенная поддержка CoffeeScript была добавлена в Rails с версии 3.1. Программы написанные на данном языке перед выполнением компилируются в javascript. Язык CoffeeScript позволяет писать программы в функциональном стиле, в нем более полно реализовано использование классов.
					CoffeeScript используется чтобы улучшить читаемость кода и уменьшить его размер. В среднем для выполнения одинаковых действий на CoffeeScript требуется
					<div><div><div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div></div><div></div></div><div><div></div><div></div></div></div><div><div></div><div></div></div><div><div></div><div></div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div><div></div><div></div></div> <div>&lt;</div>

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Формат А4

чем режиме (production) исходные коды на этих языках компилируются в обычные CSS и javascript файлы и затем на все входящие запросы отдаются как статичные файлы, непосредственно веб-сервером. Благодаря такому подходу снижается нагрузку на серверную машину, а следовательно уменьшается время отклика. В режиме разработчика (development) перекомпиляция происходит при каждом запросе, для оперативного просмотра изменений в исходном коде в процессе разработки.

### 11.3.6 Средство построения графиков

Основной целью разрабатываемой информационной системы является мониторинг состояния здоровья пациентов. Основным средством визуального отображения результатов мониторинга являются информационные графики и диаграммы. Для вывода графиков используется javascript библиотека Highcharts<sup>1)</sup>.

## 11.4 Backend

В данном разделе рассмотрены технологии, с которыми пользователь непосредственно не взаимодействует. Поскольку разрабатываемая нами информационная система является клиент-серверным веб-приложением, здесь будет рассмотрена так называемая серверная компонента.

### 11.4.1 Ruby

Создатель Ruby — Юкихиро Мацумото (Matz) — интересовался языками программирования, ещё будучи студентом, но идея о разработке нового языка появилась позже. Ruby начал разрабатываться 23 февраля 1993 года и вышел в свет в 1995 году.

Название навеяно языком Perl, многие особенности синтаксиса и семантики из которого заимствованы в Ruby: англ. pearl — «жемчужина», ruby — «рубин».

Целью разработки было создание «настоящего объектно-ориентированного», лёгкого в разработке, интерпретируемого языка программирования.

Язык следует принципу «наименьшей неожиданности»: программа должна вести себя так, как ожидает программист.

<sup>1)</sup> <http://www.highcharts.com/>

Подп. и дата		ная система является клиент-серверным веб-приложением, здесь будет рассмотрена так называемая серверная компонента.				
Инв. № дубл.		11.4.1 Ruby				
Взам. инв. №		Создатель Ruby — Юкихиро Мацумото (Matz) — интересовался языками программирования, ещё будучи студентом, но идея о разработке нового языка появилась позже. Ruby начал разрабатываться 23 февраля 1993 года и вышел в свет в 1995 году.				
Подп. и дата		Название навеяно языком Perl, многие особенности синтаксиса и семантики из которого заимствованы в Ruby: англ. pearl — «жемчужина», ruby — «рубин».				
Инв. № подл.		Целью разработки было создание «настоящего объектно-ориентированного», лёгкого в разработке, интерпретируемого языка программирования.				
		Язык следует принципу «наименьшей неожиданности»: программа должна вести себя так, как ожидает программист.				
		<div>1) <a href="http://www.highcharts.com/">http://www.highcharts.com/</a></div>				
					Дипломный проект	Лист
						52
Изм.	Лист	№ докум.	Подп.	Дата		

### 11.4.2 Ruby on Rails

Фреймворк Ruby on Rails был создан Давидом Хейнемейером Ханссоном на основе его работы в компании 37signals над средством управления проектами Basecamp и выпущен в июле 2004 года.

Ruby on Rails определяет следующие принципы разработки приложений:

- а) предоставляет механизмы повторного использования, позволяющие минимизировать дублирование кода в приложениях (принцип Don't repeat yourself);
- б) по умолчанию используются соглашения по конфигурации, типичные для большинства приложений (принцип Convention over configuration).

### 11.4.3 Концепция MVC

Основными компонентами приложений Ruby on Rails являются модель (model), представление (view) и контроллер (controller). Ruby on Rails использует REST-стиль построения веб-приложений.

**Модель** Модель предоставляет остальным компонентам приложения объектно-ориентированное отображение данных (таких как каталог продуктов или список заказов). Объекты модели могут осуществлять загрузку и сохранение данных в реляционной базе данных, а также реализуют бизнес-логику.

Как уже говорилось выше, для хранения объектов модели в реляционной СУБД по умолчанию в Rails 3 используется библиотека ActiveRecord. Конкурирующий аналог — DataMapper. Существуют плагины для работы с нереляционными базами данных, например Mongoid для работы с MongoDB.

В качестве примера модели рассмотрим сущность Документ (листинг 4). Как видно, в начале кода описывающего модель вводятся связи с другими сущностями (в данном случае это пользователь создавший документ и событие к которому документ привязан). После этого описывает валидация, затем вводятся так называемые коллбэки - функции которые будут вызваны в случае инвольвинга определенных событий. В данном случае перед созданием экземпляра документа, в качестве создателя будет записан текущий пользователь.

```
class Document < ActiveRecord::Base
```

					<div style="text-align: center; font-size: 24px; font-weight: bold;">Дипломный проект</div>	Лист
						53
Изм.	Лист	№ докум.	Подп.	Дата		

```
# Создатель документа
belongs_to :user

# Событие к которому привязан документ
belongs_to :event

attr_accessible :event_id

validates :user_id, :presence => true

# Назначем создателем события текущего авторизованного пользователя
before_create do
  if User.current.present?
    self.user_id = User.current.id
  end
end
end
```

## Листинг 4: Модель документов

**Представление** Представление создает пользовательский интерфейс с использованием полученных от контроллера данных. Представление также передает запросы пользователя на манипуляцию данными в контроллер (как правило, представление не изменяет непосредственно модель).

В Ruby on Rails представление описывается при помощи шаблонов ERB. Они представляют собой файлы HTML с дополнительными включениями фрагментов кода Ruby (Embedded Ruby или ERb). Вывод, сгенерированный встроенным кодом Ruby, включается в текст шаблона, после чего получившаяся страница HTML возвращается пользователю. Кроме ERB возможно использовать ещё около 20 шаблонизаторов, в том числе Haml.

Пример представления приведен в листинге 5. В данном примере показано представление, отрисовывающее главную страницу. Проверяется условие - если пользователь зарегистрирован - то отрисовывается ссылка на личный кабинет пользователя, иначе отрисовывается общая информация.

```
<% if user_signed_in? %>
  <h1>Начать работу</h1>
  <p>В личном кабинете вы можете:</p>
```

	Подп. и дата							
	Инв. № дубл.							
	Взам. инв. №							
	Подп. и дата							
	Инв. № подл.							

Представление создает пользовательский интерфейс с использованием полученных от контроллера данных. Представление также передает запросы пользователя на манипуляцию данными в контроллер (как правило, представление не изменяет непосредственно модель).

В Ruby on Rails представление описывается при помощи шаблонов ERB. Они представляют собой файлы HTML с дополнительными включениями фрагментов кода Ruby (Embedded Ruby или ERb). Вывод, сгенерированный встроенным кодом Ruby, включается в текст шаблона, после чего получившаяся страница HTML возвращается пользователю. Кроме ERB возможно использовать ещё около 20 шаблонизаторов, в том числе Haml.

Пример представления приведен в листинге 5. В данном примере показано представление, отрисовывающее главную страницу. Проверяется условие - если пользователь зарегистрирован - то отрисуется ссылка на личный кабинет пользователя, иначе отрисуется общая информация.

```
<% if user_signed_in? %>  
    <h1>Начать работу</h1>  
  
    <p>В личном кабинете вы можете:</p>
```

					<div>Дипломный проект</div>	Лист
						54
Изм.	Лист	№ докум.	Подп.	Дата		

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Бл.	
-----	--

Инв. №	Ин
--------	----

Взаим

Подп. и дата	
--------------	--

list.html.erb.

Пример кода контроллера приведен в листинге 6. На данном рисунке хорошо демонстрируется суть контроллера - соединить поступивший HTTP-запрос (params) с моделью предметной области.

```
class Cabinet::Doctor::DiagnosticController < Cabinet::DoctorController
  def show
    @patient = PatientUser.find(params[:id])
  end

  def chart
    from = Time.at(params[:from].to_i)
    to = Time.at(params[:to].to_i)

    respond_to do |f|
      f.json {
        render json: ChartFactory.build(params[:patient_id], params[:
          parameter_id], from, to)
      }
    end
  end

  def raw
    from = Time.at(params[:from].to_i)
    to = Time.at(params[:to].to_i)
    @data = DiagnosticFactory.raw(params[:patient_id], params[:
      parameter_id], from, to)
  end
end
```

Листинг 6: Контроллер для приема диагностики

## 11.5 Дополнительные возможности платформы Ruby on Rails

### 11.5.1 Встроенный генератор Rails Generator

Использование генератора Rails сопровождает разработчика с момента инициализации нового проекта. Генератор - это скриптовая программа на языке Ruby, которая на основе полученных входных данных генерирует на основе шаблонов стандартные файлы исходного кода проекта. Это избавляет разработчика от рутин-

Подп. и дата		Инв. № дубл.		Взам. инв. №		Подп. и дата		Инв. № подл.	
Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект				Лист
									56



ной работы по созданию файлов и папок, которые стандартны для всех проектов Rails. Классический пример использования представлен в листинге 9 - инициализация нового проекта .

```
user@host$ rails generate some_application_name
```

Листинг 7: Создание нового приложения

11.5.2    **Формы ввода данных**

Формы в веб-приложениях – это основной интерфейс для пользовательского ввода. Однако, обработка форм может достаточно трудоемкой из-за необходимости описывать элементы форм, правила валидации данных на стороне клиента и сервера. Rails устраняет эти сложности, предоставляя хелперы для разметки форм. Помимо стандартных хелперов, существует библиотека `simple_form`. Данная библиотека сокращает время при написании кода веб-формы, а именно - разработчику не нужно указывать URL-адрес обработчика запроса (при нажатии кнопки submit); не нужно вручную прописывать HTML-разметку для элемента, отвечающего за отображение и хранение значения того или иного атрибута - алгоритм `simple_form` сам подберет необходимую разметку на основании типа данных. Кроме того `simple_form` сама преобразует существующую валидацию (реализованную средствами Rails) в валидацию на стороне клиента (работающую на javascript). Это дает очевидную выгоду - поскольку ошибки отсекаются на стороне клиента, снижается нагрузка на сетевое соединение и на обрабатывающий сервер.

11.5.3    **Рассылка электронной почты**

Action Mailer позволяет отправлять электронные письма из приложения, используя модель и представления рассыльщика. Таким образом, в Rails электронная почта используется посредством создание рассыльщиков, наследуемых от `ActionMailer::Base`, и находящихся в `app/mailers`. Эти рассыльщики имеют связанные представления, которые находятся среди представлений контроллеров в `app/views`.

Для рассылки почты не требуется приобретение и развертывание собственного почтового сервера. Достаточно подключить существующий аккаунт в популярных почтовых серверах (yandex, gmail) в конфигурационных файлах

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Дипломный проект				Лист
									57
Изм.	Лист	№ докум.	Подп.	Дата					

(config/environments/production.rb) приложения. Веб-сервер будет отсылать электронные письма подключившись к аккаунту через протокол SMTP.

В режиме разработчика (development) можно настроить имитацию отправки писем для проверки правильности работы мейлера и тестирования системы в целом. В этом случае веб-сервер будет сохранять отправляемые письма в виде файлов, в папку tmp.

#### 11.5.4 Система контроля версий базы данных

Поскольку очень часто (как и в нашем случае) разработчики работают в команде, возникает проблема контроля версий. Причем данный контроль должен выполняться не только в отношении исходного кода и задач (см. [git](#), [github](#)), но и за состоянием структуры базы данных.

Данная проблема успешно решается с помощью концепции мигрирования БД. Она заключается в том, что все изменения базы данных делятся на фрагменты - миграции.

В первых версиях фреймворка Rails разработчик должен был сам назначить имя миграции. Это часто приводило к коллизиям и приходилось вручную менять и миграцию и структуру БД.

В более поздних версиях к имени миграции стал добавляться хэш отражающий дату создания миграции.

С помощью выбранной системы контроля версий разработчики синхронизируют файлы миграций между собой и рабочим сервером (рис. 11.1).

Active Record отслеживает, какие миграции уже были выполнены, поэтому все, что нужно сделать, это обновить свой исходный код и запустить `rake db:migrate`. Active Record сам определит, какие миграции нужно запустить, проверив таблицу базы данных `schema_migrations`, автоматически создаваемую при изначальном вызове `rake db:migrate`. `schema_migrations` содержит единственный столбец с именем `versions`, содержащий временные метки, с которых начинаются созданные миграции Active Record (рис. 11.2). Каждая временная метка, содержащаяся в `schema_migrations`, показывает, что миграция, связанная с временной меткой, была вызвана ранее, и не должна быть вызвана при будущих вызовах `rake db:migrate`. Он также обновит файл `db/schema.rb` в соответствии с новой структурой.

Имя миграции. Это часто приводило к коллизиям и приходилось вручную менять и миграцию и структуру БД.	Подп. и дата
В более поздних версиях к имени миграции стал добавляться хэш отражающий дату создания миграции.	
С помощью выбранной системы контроля версий разработчики синхронизируют файлы миграций между собой и рабочим сервером (рис. 11.1).	Инв. № дубл.
Active Record отслеживает, какие миграции уже были выполнены, поэтому все, что нужно сделать, это обновить свой исходный код и запустить rake db:migrate. Active Record сам определит, какие миграции нужно запустить, проверив таблицу базы данных schema_migrations, автоматически создаваемую при изначальном вызове rake db:migrate. schema_migrations содержит единственный столбец с именем versions, содержащий временные метки, с которых начинаются созданные миграции Active Record (рис. 11.2). Каждая временная метка, содержащаяся в schema_migrations, показывает, что миграция, связанная с временной меткой, была вызвана ранее, и не должна быть вызвана при будущих вызовах rake db:migrate. Он также обновит файл db/schema.rb в соответствии с новой структу-	Взам. инв. №
	Подп. и дата
	Инв. № подл.

Изм.	Лист	№ докум.	Подп.	Дата

Дипломный проект

Лист 58

рой базы данных.

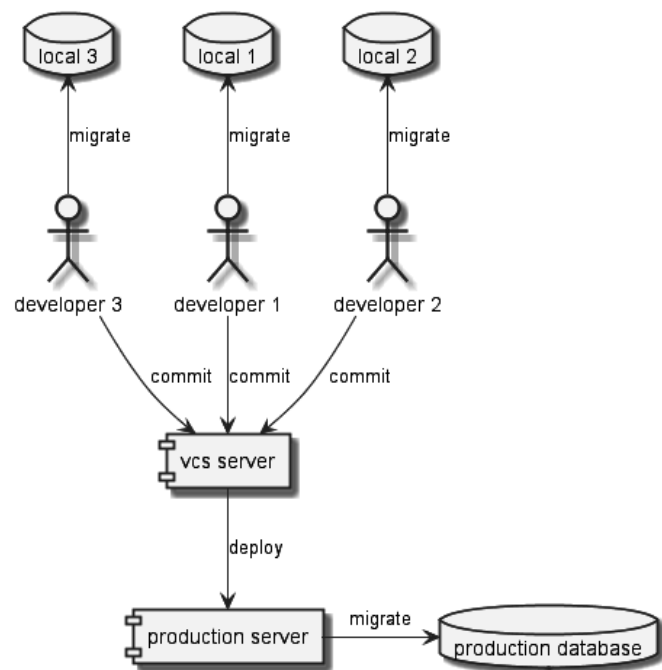


Рисунок 11.1 – Централизованный контроль версий базы данных.

11.6 Использование сторонних библиотек на языке Ruby

В процессе разработки проекта для решения многих типовых задач также были использованы библиотеки из хостинга RubyGems. Как правило для каждой задачи используется соответствующая библиотека (или группа библиотек). Перечень всех библиотек находится в файле Gemfile. Для установки библиотек на компьютер разработчика, а также на рабочую машину системы производится с помощью специальной утилиты Bundler, которая читает перечень гемов из Gemfile, скачивает необходимые библиотеки с хостинга и выполняет постинсталляционные скрипты. После установки всех необходимых библиотек Bundler фиксирует версию каждой библиотеки в файле Gemfile.lock. Фиксирование версий библиотек позволяет снижать риск несовместимости между библиотеками при разворачивании приложения на рабочем сервере.

11.6.1 Аутентификация и авторизация

Данная задача была самой первой и ее причины очевидны - обработка личной медицинской информации предполагает тщательной сохранение медицинской тайны. Разграничение прав доступа к данным и функционалу также крайне необ-

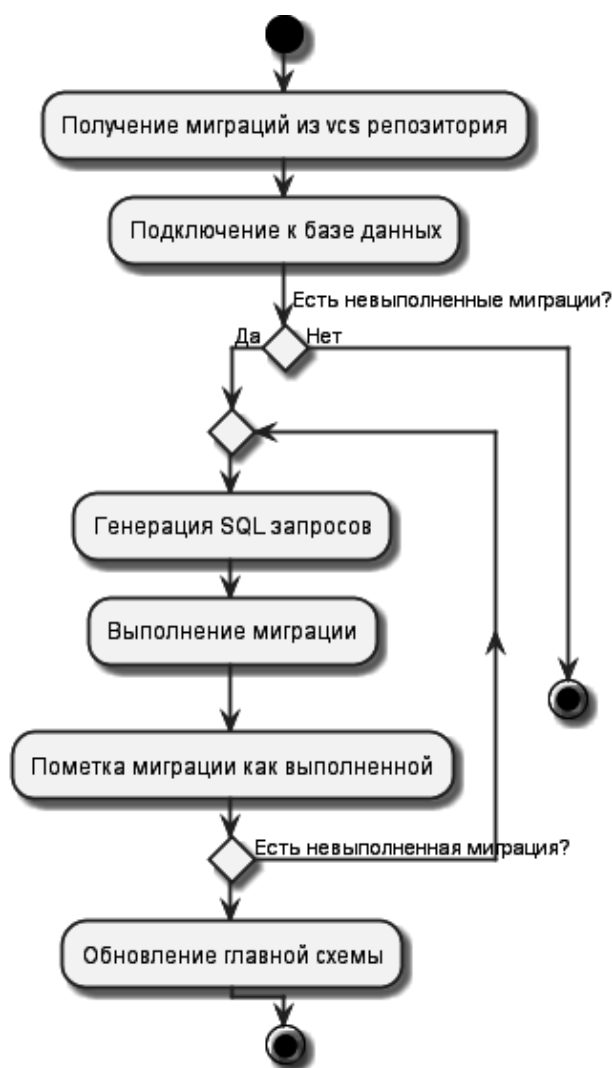


Рисунок 11.2 – Схема выполнения миграции.

ходимы - недопустимо чтобы доктор мог изменять значения, введенные пациентом. В то же время некоторые сведения о пациентах и о других пользователях могут быть изменены (например, фамилия). Проанализировав эти и другие требования (такие как простота и стоимость реализации) мы пришли к выводу что для организации функции аутентификации и аутентификации необходимо использовать стороннюю библиотеку devise, а для функции авторизации библиотеку CanCan.

### 11.6.2 Доступ к базе данных

Для взаимодействия с хранилищем данных проект на Ruby on Rails использует специальные библиотеки. Для каждой СУБД существует своя библиотека подключений. В нашем проекте использует Postgresql 9.1 (подробнее см. ниже). Для

Инв. № дубл.	Подп. и дата				
Взам. инв. №	Подп. и дата				
Инв. № подл.	Подп. и дата				
Изм.	Лист	№ докум.	Подп.	Дата	<div> <div>Дипломный проект</div> <div>Лист 60</div> </div>



Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Формат А4

- а) присоединение/отсоединение клиента от определенного канала;
- б) передача сообщений как в рамках определенного канала, так и широко-вещательных сообщений.

На стороне клиента используется стандартный объект WebSocket обернутый в класс на CoffeeScript для более удобной работы.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										63
Изм.	Лист	№ докум.	Подп.	Дата						

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

В связи с тем, что разработка проекта ведется в команде, необходимо решить проблему совместного доступа к файлам проекта. Помимо этого, файлы проекта во время работы постоянно претерпевают различные изменения. При этом часто бывает важно иметь не только последние версии, но и несколько предыдущих. В простейшем случае можно просто хранить несколько вариантов документа, нумеруя их соответствующим образом. Такой способ неэффективен (приходится хранить несколько практически идентичных копий), требует повышенного внимания и дисциплины и часто ведёт к ошибкам, поэтому были разработаны средства для автоматизации этой работы. В качестве решения данной проблемы неэффективности было решено использовать системы управления версиями (VCS - Version Control System).

Данная система спроектирована как набор программ, специально разработанных с учётом их использования в скриптах. Это позволяет удобно создавать специализированные системы контроля версий на базе Git или пользовательские интерфейсы. Достоинствами данной системы являются:

- а) высокая производительность;
- б) децентрализованность;
- в) развитые средства интеграции с IDE;
- г) продуманная система команд.

<sup>1)</sup> Это веб-сервис для хостинга проектов и их совместной разработки. GitHub позиционируется как веб-сервис хостинга проектов с использованием системы контроля версий git, а также как социальная сеть для разработчиков. Пользователи могут создавать неограниченное число публичных репозиторий, для каждого из которых предоставляется wiki, система issue tracking-a, есть возможность проводить code review и многое другое. GitHub на данный момент является самым

<sup>1)</sup> <https://github.com/>





диаграммы можно описать на специальном языке в текстовой форме, после чего получить диаграмму в виде png или svg файла.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата					
Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект	Лист			
						66			

## 13 Разработка проекта

### 13.1 План разработки

1) Процесс разработки разбит на несколько фаз. Фаза состоит из предварительного набора задач по завершению которых фаза будет считаться завершенной.

### 13.1.1 Skeleton

Цель - настройка среды для разработки, построение простейшего "скелета" системы:

- а) установка и настройка ruby, rvm, rails;
- б) установка и настройка postgres;
- в) инициализация проекта;
- г) настройка авторизации;
- д) установка админки.

### 13.1.2 General

Цель - разрабока основы предметной области, доработка каркаса приложения:

- а) отражение основных объектов предметной области в виде классов моделей;
- б) покрытие тестам;
- в) организация стурктуры для js клиента.

### 13.1.3 Patient

Цель - реализовать кабинет пациента:

- а) профиль;
- б) запись на прием;
- в) расписание приемов у врача;
- г) расписание приема лекарств;
- д) расписание ввода показателей здоровья.

<sup>1)</sup> <https://github.com/crashr42/shm/issues/milestones>

Инва. № подл.	Подп. и дата	Взам. инв. №	Инва. № дубл.	Подп. и дата	ния:  а) отражение основных объектов предметной области в виде классов моделей; б) покрытие тестам; в) организация стурктуры для js клиета.  <b>13.1.3 Patient</b>  Цель - реализовать кабинет пациента: а) профиль; б) запись на прием; в) расписание приемов у врача; г) расписание приема лекарств; д) расписание ввода показателей здоровья.  <hr/> <sup>1)</sup> <a href="https://github.com/crashr42/shm/issues/milestones">https://github.com/crashr42/shm/issues/milestones</a>

Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект	Лист
						67

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

- а) просмотр заявок;
- б) подтверждение заявок;
- в) отклонение заявок.

а) общение пациента с доктором;

- а) просмотр своих пользователей;
- б) электронный прием;
- в) электронная запись на прием;
- г) назначение лекарств;
- д) назначение диагнозов;
- е) визуализация данных за период времени;
- ж) отчеты.

В проекте использовались дополнительные центральные ветки для организации работы над отдельной фазой. Со временем стало понятно что вести раз-



## 13.4 Физическое проектирование базы данных

В процессе разработки возник вопрос как организовать физическое изменение структуры базы данных. Классический подход создания структуры базы данных для конкретной СУБД - код на языке SQL. Такой подход удобен если структура базы создается единожды и не меняется в процессе разработки. На практике структура базы данных меняется очень часто. Ситуация усугубляется если структуру базы данных могут менять несколько разработчиков одновременно. Система контроля версий может решить данную проблему, но лишь частично.

### 13.4.1 Миграции

Ruby On Rails предлагает встроенный механизм миграций. Миграции - это методология позволяющая решить проблему изменения структуры базы данных при разработке. Миграция представляет собой код, способный изменить структуру базы данных. Основные идеи:

- а) изменения в структуре базы данных должны быть атомарными;
- б) каждое атомарное изменение оформляется в виде миграции;
- в) должна быть возможность возвращать структуру базы данных к выбранному состоянию.

Каждая миграция снабжается временной меткой, характеризующей время создания миграции. По этим временным веткам происходит упорядочивание порядка выполнения миграций. Фиксация выполненных миграций производится на уровне базы данных в специальной таблице “`schema_migrations`”. В таблицу заносятся временные метки выполненных миграций.

```
class CreateBids < ActiveRecord::Migration
  def change
    create_table :bids do |t|
      t.string :first_name, :null => false
      t.string :last_name, :null => false
      t.string :third_name, :null => true
      t.string :address, :null => false
      t.string :policy, :null => false
      t.string :passport_scan, :null => false
      t.string :status, :null => false, :default => '
        created'
```

Изм.	Лист	№ докум.	Подп.	Дата	<div>Дипломный проект</div>	Лист 70

```

t.timestamps
end
end
end

```

### Листинг 10: Пример миграций

В листинге 10 представлен пример миграции, создающей в базе данных таблицу для хранения заявок на регистрацию. В приложении В представлена итоговая схема базы данных.

## 13.5 Тестирование

Для контроля верности выполнения бизнес-процессов в проекте используется unit тестирование с помощью библиотеки RSpec. Такой подход позволяет исключать логические ошибки без полноценного запуска системы, как следствие повышается скорость разработки.

Для удобства и автоматизации тестирования применяется концепция автоматического тестирования. При каждом изменении в коде, если для данного изменения есть тест, тест запускается и выводится уведомление о результате выполнения теста. Для организации данного подхода используется библиотека Autotest.

Еще один нюанс который нужно учитывать при тестировании заключается в том что Ruby On Rails окружение запускается достаточно долго из за этого в несколько раз увеличивается время выполнения тестов. Для решения данной проблемы используется библиотека Spork. Spork запускает окружение Ruby On Rails и исключает необходимость перезапускать окружение каждый раз. Так же Spork автоматически перезагружает классы при изменении их исходного кода.

--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

## 14 Описание системы

## 14.1 Кабинеты

Весь функционал системы распределен по личным кабинетам - самостоятельным (реализация каждого модуля независима друг от друга) javascript-приложениям. Данное распределение позволяет сделать систему понятной, безопасной, уменьшается избыточность исходного кода.

После авторизации на главной странице, пользователь переходит по ссылке в свой личный кабинет. Основная навигация происходит с помощью верхнего меню.

Доктор в своем кабинете может открыть список пациентов и кликнув на конкретного пациента в списке получает возможность работать с его учетной записью: просмотреть и указать диагноз, просмотреть список лекарств которые принимает пациент, назначить пациенту врачебный прием, а также записать пациента на обследование к другому доктору.

Одним из видов коммуникации между пациентом и доктором в системе является врачебный прием. Помимо возможности записи на прием, в кабинете доктора есть страничка приема. В момент когда пациент заходит в приемный кабинет, доктор (или его ассистент) должен нажать кнопку “Начать прием”, при этом будет отмечено фактическое начало приема. После этого в системе становится доступным отмена или назначение приема лекарств пациенту. По окончании приема необходимо нажать кнопку “Завершить прием”. При необходимости доктор составляет документ по результатам приема.

В кабинете пациента доступно расписание событий, которые назначены пациенту. Как правило это приемы у врача, лечебные процедуры, уведомления о приеме лекарств. Также у пациента есть возможность выбрать лечащего врача.

Основной функцией пользователя-пациента является дистанционная подача значений своих медицинских параметров в медицинское учреждение. Это производится путем ввода информации в специальные веб-формы. Доктор в своем личном кабинете имеет возможность просматривать значений параметров указанные его пациентами, а также может ознакомиться с результатами аналитики, которые представлены в виде таблиц, графиков и диаграмм.

В кабинете менеджера находится панель управления всеми пользователями системы. Также одной из функций менеджера является рассмотрение заявок на

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

тор (или его ассистент) должен нажать кнопку “Начать прием”, при этом будет отмечено фактическое начало приема. После этого в системе становится доступным отмена или назначение приема лекарств пациенту. По окончании приема необходимо нажать кнопку “Завершить прием”. При необходимости доктор составляет документ по результатам приема.

В кабинете пациента доступно расписание событий, которые назначены пациенту. Как правило это приемы у врача, лечебные процедуры, уведомления о приеме лекарств. Также у пациента есть возможность выбрать лечащего врача.

Основной функцией пользователя-пациента является дистанционная подача значений своих медицинских параметров в медицинское учреждение. Это производится путем ввода информации в специальные веб-формы. Доктор в своем личном кабинете имеет возможность просматривать значений параметров указанные его пациентами, а также может ознакомиться с результатами аналитики, которые представлены в виде таблиц, графиков и диаграмм.

В кабинете менеджера находится панель управления всеми пользователями системы. Также одной из функций менеджера является рассмотрение заявок на

Изм.	Лист	№ докум.	Подп.	Дата	<div>Дипломный проект</div>	Лист
						72



регистрацию нового пользователя в системе.

Если пациент хочет принять участие в проекте, то он должен заполнить заявку на регистрацию. В ней он указывает свои учетные данные медицинского учреждения (номер страхового полиса, больничной карты), а также прикладывает отсканированное изображение паспорта.

Менеджер системы просматривает поступившие заявки, проверяет верность указанных в них данных и либо создает нового пользователя, либо отклоняет заявку.

14.2 События

В системе реализована событийная модель, характеризующая реальные процессы: прием, обследование. Базовым классом для всех событий является класс Event. Событие может находиться в 4 состояниях:

- а) free - событие доступно для работы (например на прием у врача со статусом free можно записаться);
- б) busy - событие занято (например на прием к врачу со статусом busy записаться не получится);
- в) process - событие обрабатывается (врач ведет прием пациента);
- г) close - событие завершено (прием окончен).

Статус события может меняться только в определенном порядке:

- а) free -> busy (запись на прием);
- б) busy -> free (освободить запись);
- в) busy -> process (начать прием);
- г) process -> close (завершить прием).

Такой подход обусловлен тем что с каждым переходом может быть связано определенное действие. Если не фиксировать возможные переходы, то для перехода, например, со статуса free -> close нужно будет создавать дополнительный обработчик.

Для контроля смены статуса событий и создания обработчиков этих переходов был создан модуль Workflow. Реализация в виде модуля обусловлена тем что позволяет внедрять данный модуль в любой класс. Пример использования модуля для обработки переходов для событий приведен в листинге 11.

```
class Event < ActiveRecord::Base
```

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

include Workflow

# Возможные переходы для статуса события

```
workflow : status do
```

$$\text{flow} : \text{default} , \quad : \text{busy} \Rightarrow : \text{process}$$
$$\text{flow} : \text{default} , \quad : \text{process} \Rightarrow : \text{close}$$

```
flow :reset duration, :free => :busy do
```

```
if self.event.present?
```

```
self.event.duration -= self.date end - self.date start
```

end

```
self.duration = 0
```

end

```
flow :reset_duration , :busy => :free do
```

```
if self.event.present?
```

```
self.event.duration += self.date_end - self.date_start
```

end

```
self.duration = self.date_end - self.date_start
```

end

end

end

### Листинг 11: Использование модуля Workflow

### 14.3 Диагностика

### 14.3.1 Прием данных

Прием диагностических данных в системе осуществляется через отсылку запроса на REST API. Запрос представляется из себя стандартный POST запрос по адресу `http://localhost:3000/diagnostic/parameter` (листинг 12) с указанием дополнительных параметров:

- а) user\_id - идентификатор пользователя в системе;
- б) parameter\_id - идентификатор параметра;
- в) value - значение параметра.

```
user@localhost$ curl -d "user_id=1&parameter_id=2&value=44" \
http://localhost/diagnostic/parameter
```

### Листинг 12: Отправка диагностических данных

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>end</div> <div>end</div> <div>Листинг 11: Использование модуля Workflow</div> <div>14.3 Диагностика</div> <div>14.3.1 Прием данных</div> <div>Прием диагностических данных в системе осуществляется через отсылку запроса на REST API. Запрос представляется из себя стандартный POST запрос по адресу http://localhost:3000/diagnostic/parameter (листинг 12) с указанием дополнительных параметров:</div> <div><div>а) user_id - идентификатор пользователя в системе;</div><div>б) parameter_id - идентификатор параметра;</div><div>в) value - значение параметра.</div></div> <div>user@localhost\$ curl -d "user_id=1&amp;parameter_id=2&amp;value=44" \ http://localhost/diagnostic/parameter</div> <div>Листинг 12: Отправка диагностических данных</div>					
					<table><tr><td>Изм.</td><td>Лист</td><td>№ докум.</td><td>Подп.</td><td>Дата</td></tr><tr><td></td><td></td><td></td><td></td><td></td></tr></table>	Изм.	Лист	№ докум.	Подп.	Дата
Изм.	Лист	№ докум.	Подп.	Дата						

Дипломный проект					Лист
					74

### 14.3.2 Доступ к диагностическим данным

Доступ к диагностическим данным предоставляется доктору. Доктор может просматривать данные в виде графиков или таблиц. Графики формируются с помощью класса ChartFactory в зависимости от класса параметра. ChartFactory имеет метод build который принимает в качестве параметров:

- а) patient\_id - идентификатор пациента;
- б) parameter\_id - идентификатор параметра;
- в) from - начальная дата для выборки данных;
- г) to - конечная дата для выборки данных.

Метод возвращает ассоциативный массив со структурой понятной Highcharts. После чего массив сериализуется в JSON и отдается клиенту.

### 14.3.3 События

После поступления диагностических данных в систему, системы инициирует специальное событие. Событие указывает WebSocket серверу оповестить всех заинтересованных подписчиков о том что диагностические данные обновились.

Данный механизм позволяет доктору просматривать поступающие диагностические данные в реальном времени.

Для доступа к диагностическим данным в реальном времени используется Websocket клиент.

## 14.4 Тесты

Рассмотрим тестирование на примере подачи заявки на регистрацию (листинг 13). При подаче заявке важно чтобы при создании, одобрении или отклонении заявки - заявитель был уведомлен по email о соответствующем действии.

```
require 'spec_helper'
```

```
describe Bid do
  before { BidMailer.deliveries.clear }
end
```

```
it 'should be valid' do
  b = build(:bid)
  b.should be_valid
end
```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>Данный механизм позволяет доктору просматривать поступающие диагностические данные в реальном времени.</p> <p>Для доступа к диагностическим данным в реальном времени используется Websocket клиент.</p> <h3>14.4 Тесты</h3> <p>Рассмотрим тестирование на примере подачи заявки на регистрацию (листинг 13). При подаче заявке важно чтобы при создании, одобрении или отклонении заявки - заявитель был уведомлен по email о соответствующем действии.</p> <pre>require 'spec_helper'  describe Bid do   before { BidMailer.deliveries.clear }    it 'should be valid' do     b = build(:bid)     b.should be_valid   end end</pre>
Изм.	Лист	№ докум.	Подп.	Дата	<div>Дипломный проект</div> <div>75</div>



```

        :values => %w(true false),
        :default => 'false'
    })
    p.should be_valid
end

it 'should be invalid' do
  p = build(:bool_parameter, :metadata => {
    :values => '',
    :default => ''
  })
  p.should_not be_valid
  p.errors[:metadata].should include('parameter.bool.metadata.
    errors.default')
  p.errors[:metadata].should include('parameter.bool.metadata.
    errors.values')
end
end

context 'validate value' do
  before(:each) do
    @pr = build(:bool_parameter)
  end

  it 'should be valid' do
    @pr.validate_value(true).should eq(true)
    @pr.validate_value(false).should eq(true)
    @pr.validate_value('true').should eq(true)
    @pr.validate_value('false').should eq(true)
  end

  it 'should be not valid' do
    @pr.validate_value(Class).should eq(false)
    @pr.validate_value(123).should eq(false)
  end
end
end
end

```

Листинг 14: Тестирование возможных значений для булевого параметра

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	Дипломный проект					Лист
										77
					Изм.	Лист	№ докум.	Подп.	Дата	

Более сложными тесты проверяют правильность обработки событий в системе. Например событие нельзя сразу перевести и статуса free в close, так как нарушается очередность состояний события. Тесты для проверки событий приведены в приложении Б.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										78
Изм.	Лист	№ докум.	Подп.	Дата						

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Для достижения минимальных затрат по закупке оборудования - система может располагаться на одном физическом сервере. Однако данная схема не рекомендуется из за ее ненадежности.

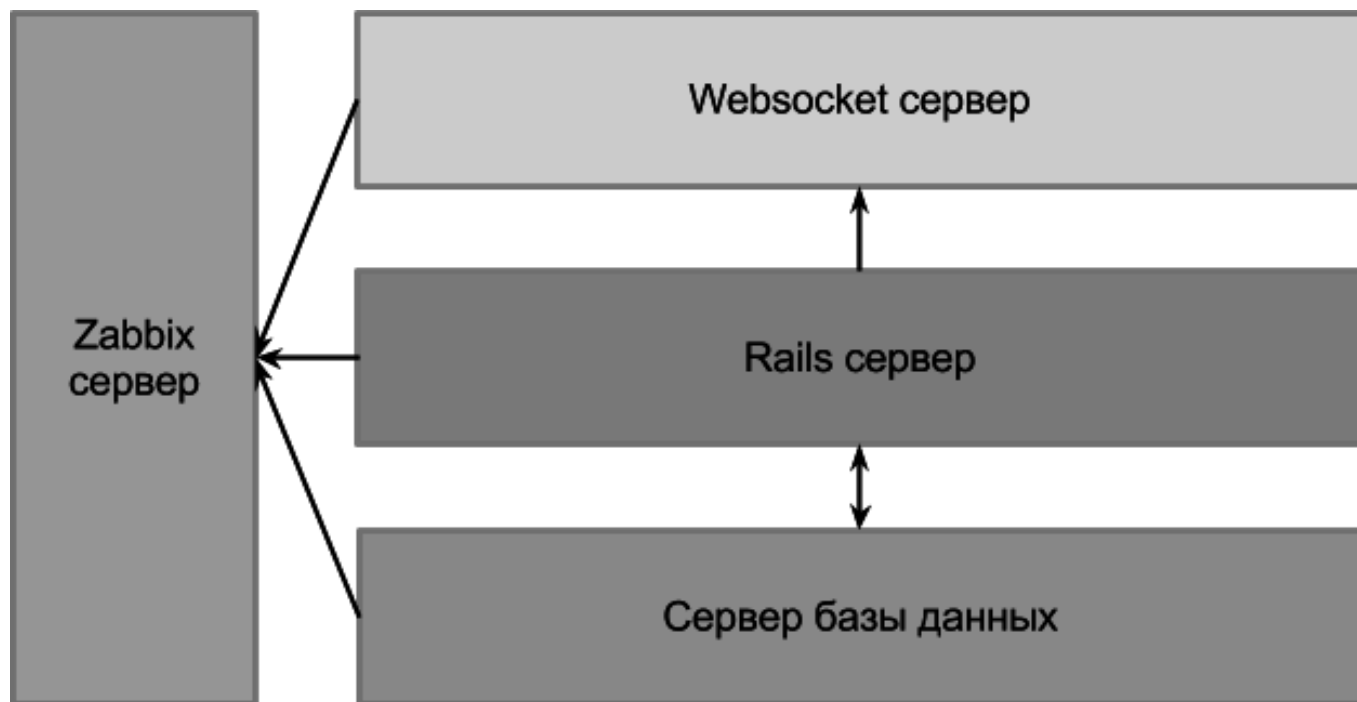


Рисунок 15.1 – Схема связи между серверами

На рисунке 15.1 изображена минимально рекомендуемая схема серверов. При такой схеме каждый сервер выполняет свою задачу независимо от других серверов:

- а) Websocket сервер обслуживает обмен сообщений между пользователями;
- б) Rails сервер обеспечивает работу непосредственно системы;
- в) Сервер базы данных обслуживает работу Postgresql базы;
- г) Zabbix сервер занимается мониторингом работы всех серверов.

Их схемы развертывания (рис. 15.2) видно что развертывание - это многоэтапный процесс в котором важна последовательность этапов. Так же важно учитывать что приложение считается обновленным только в случае если все этапы выполнены успешно. В случае неуспешного выполнения хотя бы одного из эта-

пов необходимо обратить все изменения. Исходя из анных фактов вытекает важно требования для системы развертывания - транзакционность, т.е. любое изменение должно быть обратимо.




Рисунок 15.2 – Схема развертывания Ruby On Rails сайта

Для развертывания и обновления сайта на рабочем сервере существует библиотека Capistrano. Данная библиотека позволяет настроить полностью контролируемый процесс развертывания сайта.

Основные преимущества от использования данной библиотеки:

- а) поддержка транзакций и возможность обратить все изменения;
- б) гибкая настройка процесса за счет возможности выполнять любой удаленный
- в) код на сервере;
- г) поддержка протокола ssh - необходимо для обеспечения безопасности;
- д) интеграция с Ruby On Rails;
- е) развертывание на несколько серверов одновременно;

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Перезапуск сервера приложений</div> 															
<p>Рисунок 15.2 – Схема развертывания Ruby On Rails сайта</p>																				
<p>Для развертывания и обновления сайта на рабочем сервере существует библиотека Capistrano. Данная библиотека позволяет настроить полностью контролируемый процесс развертывания сайта.</p>																				
<p>Основные преимущества от использования данной библиотеки:</p>																				
<p>а) поддержка транзакций и возможность обратить все изменения;</p>																				
<p>б) гибкая настройка процесса за счет возможности выполнять любой удаленный</p>																				
<p>в) код на сервере;</p>																				
<p>г) поддержка протокола ssh - необходимо для обеспечения безопасности;</p>																				
<p>д) интеграция с Ruby On Rails;</p>																				
<p>е) развертывание на несколько серверов одновременно;</p>																				
<table><tr><td>Изм.</td><td>Лист</td><td>№ докум.</td><td>Подп.</td><td>Дата</td></tr><tr><td> </td><td> </td><td> </td><td> </td><td> </td></tr></table>					Изм.	Лист	№ докум.	Подп.	Дата						<div>Дипломный проект</div>					Лист
Изм.	Лист	№ докум.	Подп.	Дата																
										80										



ж) установка окружения ruby.

На данном этапе разработки библиотека не используется, так как нет необходимости разворачивать приложение на рабочем сервере.

Рассмотрим инструкцию для запуска сайта в режиме разработчика.

В качестве операционной системы можно использовать любой UNIX-like дистрибутив.

Для начала нужно установить окружение для работы ruby, установку лучше всего производить через `rvm`. Для работы проекта требуется ruby версии 1.9.3.

Далее необходимо установить базу данных PostgreSQL и создать пользователя в базе данных с правами на создание баз данных. В конфигурационном файле `config/database.yml` в секции `development` нужно выставить соответствующие логин и пароль ждя подключения к базе данных.

Следующим шагом создадим непосредственно базу данных с помощью команды `rake db:create`. База данных создана, но в ней нет необходимых таблиц. Чтобы добавить таблицы в базу данных выполним `rake db:migrate`.

Для работы сайта создадим набор тестовых данных с помощью команды `rake db:seed`.

Запустим Websocket сервер с помощью команды `rake wserver`.

Теперь можно запускать непосредственно сайт - `rails s`. После чего сайт будет доступен по адресу `http://localhost:3000`.

### 15.3 Nginx

Архитектура работающего веб-сервера является двухуровневой. На первом уровне находится HTTP-сервер, который перехватывает все HTTP запросы поступающие от клиентов. В качестве такого сервера в нашем проекте используется бесплатный сервер от Игоря Сысоева - `nginx`. Данный сервер длительное время он обслуживает серверы многих высоконагруженных российских сайтов, таких как Яндекс, Mail.Ru, ВКонтакте и Рамблер. Согласно статистике Netcraft `nginx` обслуживал или проксировал 13.54% самых нагруженных сайтов в мае 2013 года<sup>1)</sup>.

Настройка сервера начинается с его установки. Это можно сделать обычными для \*nix систем способами - установить его через репозиторий пакетов, либо

<sup>1)</sup> <http://news.netcraft.com/archives/2013/05/03/may-2013-web-server-survey.html>

Инв. № подл.	Подп. и дата		Инв. № дубл.		Взам. инв. №		Подп. и дата		<div>Дипломный проект</div>					Лист	
Изм.		Лист		№ докум.		Подп.		Дата							81

скомпилировать из исходников с учетом особенностей конкретной рабочей машины.

Далее необходимо настроить конфигурацию для конкретного сайта (nginx позволяет хостить множество сайтов). Для это в папке конфигурации надо создать файл с именем сайта, как правило он расположен в папке /etc/nginx/enabled-sites/site\_name.conf. В данном файле необходимо указать полный URL сайта и номер порта. Также надо указать директорию в которой хранится сайт. Как правило для Ruby on Rails это /srv/site\_name/public.

С сервером второго уровня nginx связывается с помощью IPC-сокета, путь к которому указываются в конфигурационном файле сайта.

15.4 Unicorn

Сервер второго уровня получает поступающие запросы от сервера первого уровня, выполняет их в среде Ruby on Rails, результат вычислений отдает в виде стандартных веб-файлов (css, html, js) назад серверу первого уровня, который отдает их уже клиенту.

В качестве сервера второго уровня нами был выбран Unicorn. Данный сервер был выбран за его популярность среди Ruby on Rails - разработчиков, что означает наличие большого количества примеров файлов конфигурации, что ускоряет и облегчает процесс развертывания.

Двухуровневая архитектура предоставляет возможность горизонтального масштабирования системы. Например можно организовать кластер из серверов второго уровня и распределять нагрузку между ними с помощью сервера первого уровня.

15.5 Логи

Аппаратная конфигурация системы предусматривает наличие нескольких физических серверов. Для обеспечения дополнительного контроля за ними необходимо настроить систему централизованного сбора логов. Данный подход позволит ускорить процесс анализа логов за счет более удобного доступа или за счет использования утилит для автоматического анализа логов.

Подп. и дата		<div>Дипломный проект</div>					Лист
Инв. № дубл.							82
Взам. инв. №							
Подп. и дата							
Инв. № подл.		Изм.	Лист	№ докум.	Подп.	Дата	

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Формат А4

- а) zabbix-agent - сервис устанавливаемый непосредственно на контролируемом сервере, позволяющий получать различные метрики работы сервера;
- б) zabbix-server - сервер собирает информацию с zabbix-agent'ов и сохраняет ее в базу данных; сервер также занимается анализом поступающих данных и оповещает ответственных в случае если требуется вмешательство;
- в) web интерфейс - позволяет производить настройку zabbix-server'а и просматривать поступающие от серверов данные.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div style="text-align: center; font-size: 1.2em; font-weight: bold;">Дипломный проект</div>					Лист
										84
Изм.	Лист	№ докум.	Подп.	Дата						

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

## 16.1 Основные угрозы

Предже чем приступать к составлению рекомендаций нужно определиться с терминологией.

Угрозой будем считать процесс, результатом которого является предоставление несанкционированного доступа к какой-либо информации.

Человека, объект или ПО целью которого является получение доступа к защищенной информации будем считать злоумышленником.

Будем рассматривать следующие виды угроз информационной безопасности:

- а) Первая – это аппаратные средства. Сбой в работе или выход из строя процессора, материнской платы, линий связи, периферийных устройств может привести к частичной или полной потери информации, хранящейся в компьютере.
- б) Второй источник угрозы – программное обеспечение. Угрозу могут представлять исходные и приобретенные программы, утилиты и операционные системы. Для обеспечения сохранности информации штатным пользователям рекомендуется устанавливать лицензионные антивирусные программы.
- в) Третий вид угрозы информационной безопасности – данные, которые хранятся на отдельных носителях или в печатном виде. Необходимо принимать отдельные меры по хранению данных, которые не находятся в компьютерной системе.
- г) Четвертый вид угрозы – пользователи компьютеров и обслуживающий персонал. Люди могут нанести вред информации как случайно, так и специально. Поэтому всегда количество людей, имеющих доступ к ин-

Лист
85

формации организации, сведен до минимума. Большинство организаций открывают в штате должность специалиста по информационной безопасности, который отвечает за сохранность данных компьютерных систем.

## 16.2 Обеспечение безопасности

### 16.2.1 Аппаратный уровень

Прежде всего нужно предотвращать физический доступ к серверам на которых расположена важная информации. В контексте разработанной системы - это любой физический сервер непосредственно обеспечивающий работоспособность системы.

Так же важно резервировать основные компоненты системы и производить постоянное архивирование важных данных для предотвращения потерь в случае сбоя в оборудовании.

Давать более конкретные рекомендации для данного уровня не имеет смысла, так как по большей части все зависит от конкретной схемы установки системы.

### 16.2.2 Программный уровень

На данном уровне обеспечение безопасности необходимо как на уровне разрабатываемой системы, так и на уровне операционной системы и на уровне обслуживающего программного обеспечения.

Важно устанавливать программное обеспечение только из доверенных источников;

На каждом физическом узле системы обеспечивать связь с другими узлами на минимальном необходимом уровне. Прежде всего это значит закрытие все неиспользуемых при работе системы портов. Конфигурация используемы портов зависит от расположения компонентов системы и настроек системы. По-умолчанию в системе используются следующие порты:

- а) 80 - Nginx;
- б) 5432 - Postgresql;
- в) 22 - ssh;
- г) 25 - SMTP;
- д) 8081 - Websocket server.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	На данном уровне обеспечение безопасности необходимо как на уровне разрабатываемой системы, так и на уровне операционной системы и на уровне обслуживающего программного обеспечения.
					Важно устанавливать программное обеспечение только из доверенных источников;
					На каждом физическом узле системы обеспечивать связь с другими узлами на минимальном необходимом уровне. Прежде всего это значит закрытие все неиспользуемых при работе системы портов. Конфигурация используемы портов зависит от расположения компонентов системы и настроек системы. По-умолчанию в системе используются следующие порты:
					а) 80 - Nginx; б) 5432 - Postgresql; в) 22 - ssh; г) 25 - SMTP; д) 8081 - Websocket server.

При использовании дополнительного программного обеспечения (бэкапы, логи) могут использоваться:

- а) 9101, 9102, 9103 - Bacula;
- б) 514 - syslog.

Доступ к серверам по протоколу ssh должен быть разрешен только с компьютеров ответственных лиц.

На каждом сервере должно быть настроено логирование всех действий, для расследования сбоев системы и случаев несанкционированного доступа.

Любые конфигурационные файлы и настройки системы не должны располагаться в публичном доступе.

По возможности необходимо обеспечить использование SSL протокола для доступа к серверу приложений.

Для развертывания системы нужны права на модификацию и изменение схемы базы данных. Необходимо забирать данные привилегии после установки для предотвращения несанкционированного изменения схемы базы данных.

### 16.2.3 Человеческий фактор

Частично на уровне системы реализована защита от человеческого фактора. В частности при длительном бездействии авторизованного пользователя будет произведена блокировка аккаунта. Так же система предоставляет доступ авторизованному пользователю только к определенной информации.

### 16.2.4 Политика информационной безопасности

Принятие ПИБ важный этап при организации безопасности системы. Политика должна быть разработана соответствующим отделом или руководством предприятия с целью повышения уровня защищенности информации.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист				
										87				
										Изм.	Лист	№ докум.	Подп.	Дата

## Выводы

Основная цель исследований представленных в данной ипломной работе заключалась в создании системы мониторинга состояния детей в врожденным пороком сердца. Базой для исследований стала деятельность Кузбасского кардиологического центра. Были формализованны текущие бизнес-процессы и выявлены проблемы их функционирования. Основной проблемой оказалась невозможность постоянного наблюдения за состоянием пациента.

На основе проблем были сформированы цели. Основной целью стала организация постоянного мониторинга состояния пациента.

Для достижения цели были составлены требования к будущей системе. Основным требованием стало создание технической базы, которая позволяла бы получать диагностические данные от пациента в максимально удобной для пациента форме. Так же важной технической возможностью системы является получение диагностических данных с медицинских устройств.

На основе целей были сформированы требования к будущей системе. Требования включают в себя как набор необходимых функциональных возможностей системы, так и требования к технической реализации.

Ни одно готовое решение не подошло под составленные требования. Основными причинами были: спорная техническая реализация, отсутствие части функционала, высокая стоимость. Именно после этого шага было принято разрабатывать собственную систему.

На начальном этапе разработки были скорректированы существующие бизнес-процессы, для адаптации их к новым требованиям. Далее была спроектирована основная архитектура системы. Было решено реализовывать всю функциональность на основе web-технологий. Основной причиной стала высокая доступность и распространенность этих технологий.

Основную сложность при разработке новой системы создал вопрос с выбором конкретных технологий. Были рассмотрены и испробованы различные технические решения. В итоге выбор был сделан в пользу Ruby On Rails, как решения предоставляющего наилучшую инфраструктуру для разработки.

После продолжительного этапа разработки была реализована основная архитектура системы и функциональность. Однако на данном этапе система еще

	<i>Подп. и дата</i>							
	<i>Инв. № дубл.</i>							
	<i>Взам. инв. №</i>							
	<i>Подп. и дата</i>							
	<i>Инд. № подл.</i>							

системы, так и требования к технической реализации.  
Ни одно готовое решение не подошло под составленные требования. Основными причинами были: спорная техническая реализация, отсутствие части функционала, высокая стоимость. Именно после этого шага было принято разрабатывать собственную систему.  
  
На начальном этапе разработки были скорректированы существующие бизнес-процессы, для адаптации их к новым требованиям. Далее была спроектирована основная архитектура системы. Было решено реализовывать всю функциональность на основе web-технологий. Основной причиной стала высокая доступность и распространенность этих технологий.  
  
Основную сложность при разработке новой системы создал вопрос с выбором конкретных технологий. Были рассмотрены и испробованы различные технические решения. В итоге выбор был сделан в пользу Ruby On Rails, как решения предоставляющего наилучшую инфраструктуру для разработки.  
  
После продолжительного этапа разработки была реализована основная архитектура системы и функциональность. Однако на данном этапе система еще

Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект	Лист
						88



непригодна для использования на реальном предприятии.

Исходный код текущей реализации доступен в публичном репозитории (<https://github.com/crashr42/shm>).

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div>Дипломный проект</div>					Лист
										89
Изм.	Лист	№ докум.	Подп.	Дата						

## Словарь терминов и определений

Развертывание - процесс переноса приложения на рабочий сервер и последующий запуск приложения в рабочем режиме.

Issue tracking - программное обеспечение для создания задач с возможностями:

- а) отслеживать статус выполнения задач;
- б) комментировать задачи;
- в) соотносить изменения в коде с задачей.

MVC («Модель-представление-контроллер») - схема использования нескольких шаблонов проектирования, с помощью которых модель данных приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента так, что модификация одного из компонентов оказывает минимальное воздействие на остальные. Каждый из компонентов означает:

- а) Модель - предоставляет знания: данные и методы работы с этими данными, реагирует на запросы, изменяя своё состояние. Не содержит информации, как эти знания можно визуализировать.
- б) Представление, вид - отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма (окно) с графическими элементами.
- в) Контроллер - обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.

ORM (Object-relational mapping) - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».

REST (Representational State Transfer) - «передача представлений состояний». Был предложен в 2000 году Роем Филдингом. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол (как и HTTP) должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ».

HTTP (HyperText Transfer Protocol) - протокол прикладного уровня передачи

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<p>б) Представление, вид - отвечает за отображение информации (визуализацию). Часто в качестве представления выступает форма (окно) с графическими элементами.</p> <p>в) Контроллер - обеспечивает связь между пользователем и системой: контролирует ввод данных пользователем и использует модель и представление для реализации необходимой реакции.</p> <p><u>ORM</u> (Object-relational mapping) - технология программирования, которая связывает базы данных с концепциями объектно-ориентированных языков программирования, создавая «виртуальную объектную базу данных».</p> <p><u>REST</u> (Representational State Transfer) - «передача представлений состояний». Был предложен в 2000 году Роем Филдингом. Данные в REST должны передаваться в виде небольшого количества стандартных форматов (например HTML, XML, JSON). Сетевой протокол (как и HTTP) должен поддерживать кэширование, не должен зависеть от сетевого слоя, не должен сохранять информацию о состоянии между парами «запрос-ответ».</p> <p><u>HTTP</u> (HyperText Transfer Protocol) - протокол прикладного уровня передачи</p>
Изм.	Лист	№ докум.	Подп.	Дата	<div style="text-align: center;"> <h2>Дипломный проект</h2> </div>
					Лист 90

данных (изначально — в виде гипертекстовых документов). Основой HTTP является технология «клиент-сервер», то есть предполагается существование потребителей (клиентов), которые инициируют соединение и посылают запрос, и поставщиков (серверов), которые ожидают соединения для получения запроса, производят необходимые действия и возвращают обратно сообщение с результатом.

XML (eXtensible Markup Language) - рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки. Спецификация XML описывает XML-документы и частично описывает поведение XML-процессоров (программ, читающих XML-документы и обеспечивающих доступ к их содержимому). XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как собственно XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

JSON (JavaScript Object Notation) - текстовый формат обмена данными, основанный на JavaScript и обычно используемый именно с этим языком. Как и многие другие текстовые форматы, JSON легко читается людьми.

HTML (HyperText Markup Language) - стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа в удобной для человека форме. HTML является приложением («частным случаем») SGML (стандартного обобщённого языка разметки) и соответствует международному стандарту ISO 8879. XHTML же является приложением XML.

CRUD (Create Read Update Delete) - сокращённое именование 4 базовых функций при работе с персистентными хранилищами данных — создание, чтение,

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата	<div style="text-align: center; font-size: 1.2em; font-weight: bold;">Дипломный проект</div>					Лист
										91
Изм.	Лист	№ докум.	Подп.	Дата						

редактирование и удаление.

DDL (Data Definition Language) - это семейство компьютерных языков, используемых в компьютерных программах для описания структуры баз данных.

Websocket - протокол полнодуплексной связи поверх TCP-соединения, предназначенный для обмена сообщениями между браузером и веб-сервером в режиме реального времени.

Hot Standby - механизм поддержки состояния резервного компонента системы в актуальном состоянии, позволяющий производить замену основного компонента без задержки.

SMTP (Simple Mail Transfer Protocol — простой протокол передачи почты) — это сетевой протокол, предназначенный для передачи электронной почты в сетях TCP/IP.

SSH (Secure SHell) - сетевой протокол прикладного уровня, позволяющий производить удалённое управление операционной системой и туннелирование TCP-соединений (например, для передачи файлов). Схож по функциональности с протоколами Telnet и rlogin, но, в отличие от них, шифрует весь трафик, включая и передаваемые пароли. SSH допускает выбор различных алгоритмов шифрования. SSH-клиенты и SSH-серверы доступны для большинства сетевых операционных систем.

Unix domain socket (Доменный сокет Unix) или IPC-сокет (сокет межпроцессного взаимодействия) — конечная точка обмена данными, схожая с Интернет-сокетом, но не использующая сетевой протокол для взаимодействия (обмена данными). Он используется в операционных системах, поддерживающих стандарт POSIX, для межпроцессного взаимодействия. Корректным термином стандарта POSIX является POSIX Local IPC Sockets.

SSL (Secure Sockets Layer) - криптографический протокол, который обеспечивает безопасность связи через Интернет. Он использует асимметричную криптографию для аутентификации ключей обмена, симметричное шифрование для сохранения конфиденциальности, а коды аутентификации сообщений для целостности сообщений.

					<div style="text-align: center; font-size: 24px; font-weight: bold;">Дипломный проект</div>	Лист
						92
Изм.	Лист	№ докум.	Подп.	Дата		

## Список литературы

1. A.Black David. The Well-Grounded Rubyist. — Manning Publications Co., 2009.
2. Hartl Michael. Ruby on Rails Tutorial Learn Web Development with Rails. — <http://ruby.railstutorial.org/>. — 2012.
3. Osmani Addy. Developing Backbone.js Applications. — <http://addyosmani.github.io/backbone-fundamentals/>, 2012.
4. Блинков Иван. Интерактивные сайты. — <http://www.insight-it.ru/interactive/>. — 2012.
5. Васильев А. Ю. Работа с PostgreSQL: настройка и масштабирование. — <http://leopard.in.ua/>, 2012.
6. Д. Флэнаган Ю. Мацумото. The Ruby Programming Language. — Питер, 2011.

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата					
					<div style="display: flex; justify-content: space-between; align-items: center;"> <div> <div style="border: 1px solid black; padding: 2px;">Изм.</div> <div style="border: 1px solid black; padding: 2px;">Лист</div> <div style="border: 1px solid black; padding: 2px;">№ докум.</div> <div style="border: 1px solid black; padding: 2px;">Подп.</div> <div style="border: 1px solid black; padding: 2px;">Дата</div> </div> <div style="flex-grow: 1; text-align: center;"> <h1 style="margin: 0;">Дипломный проект</h1> </div> <div style="border: 1px solid black; padding: 2px;">Лист</div> </div>				
					93				

Приложение А  
(справочное)  
Существующие бизнес-процессы

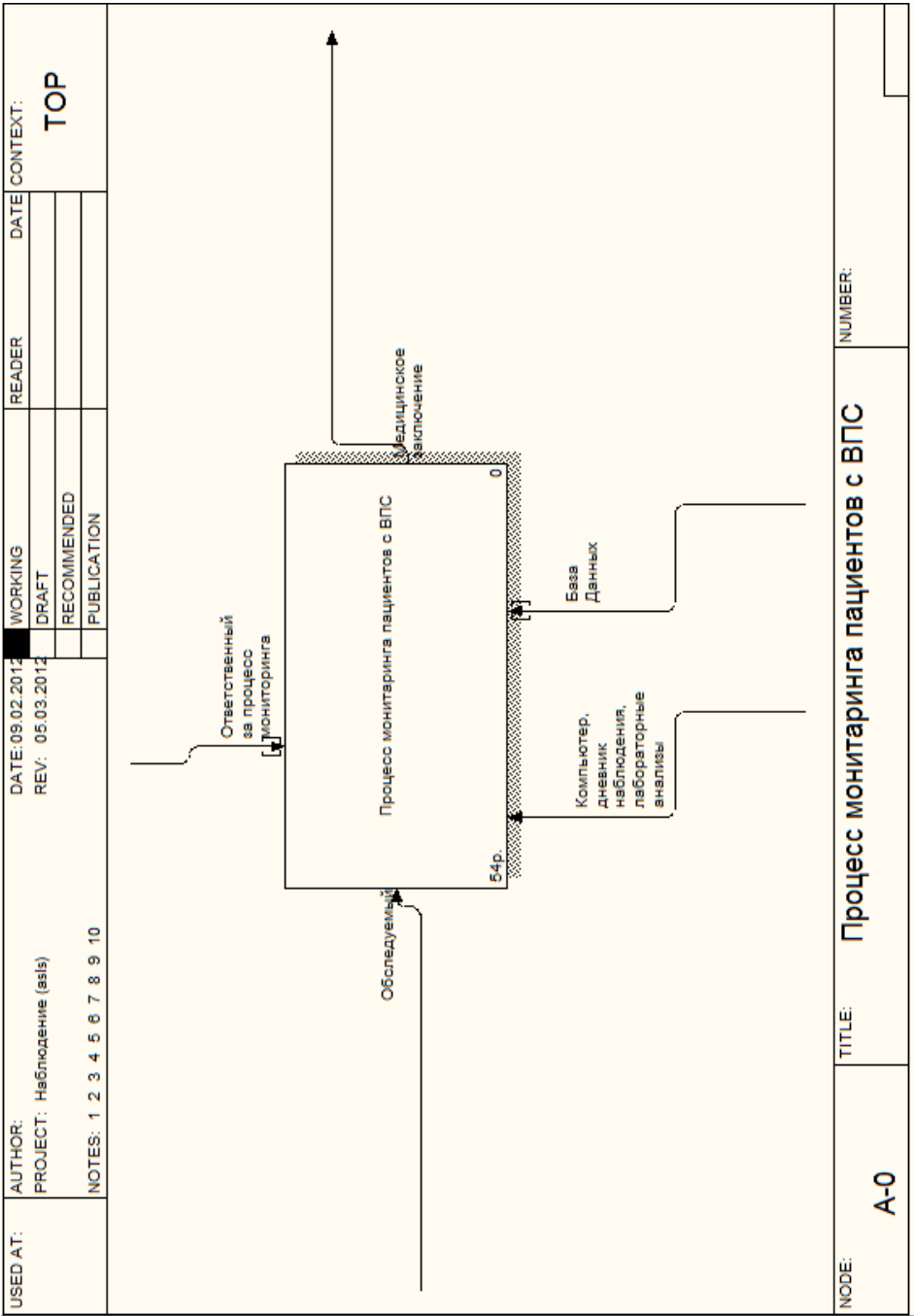


Рисунок А.1 – Общая схема процесса мониторинга

Инов. № подл.	Подп. и дата	Взам. инв. №	Инов. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата



Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

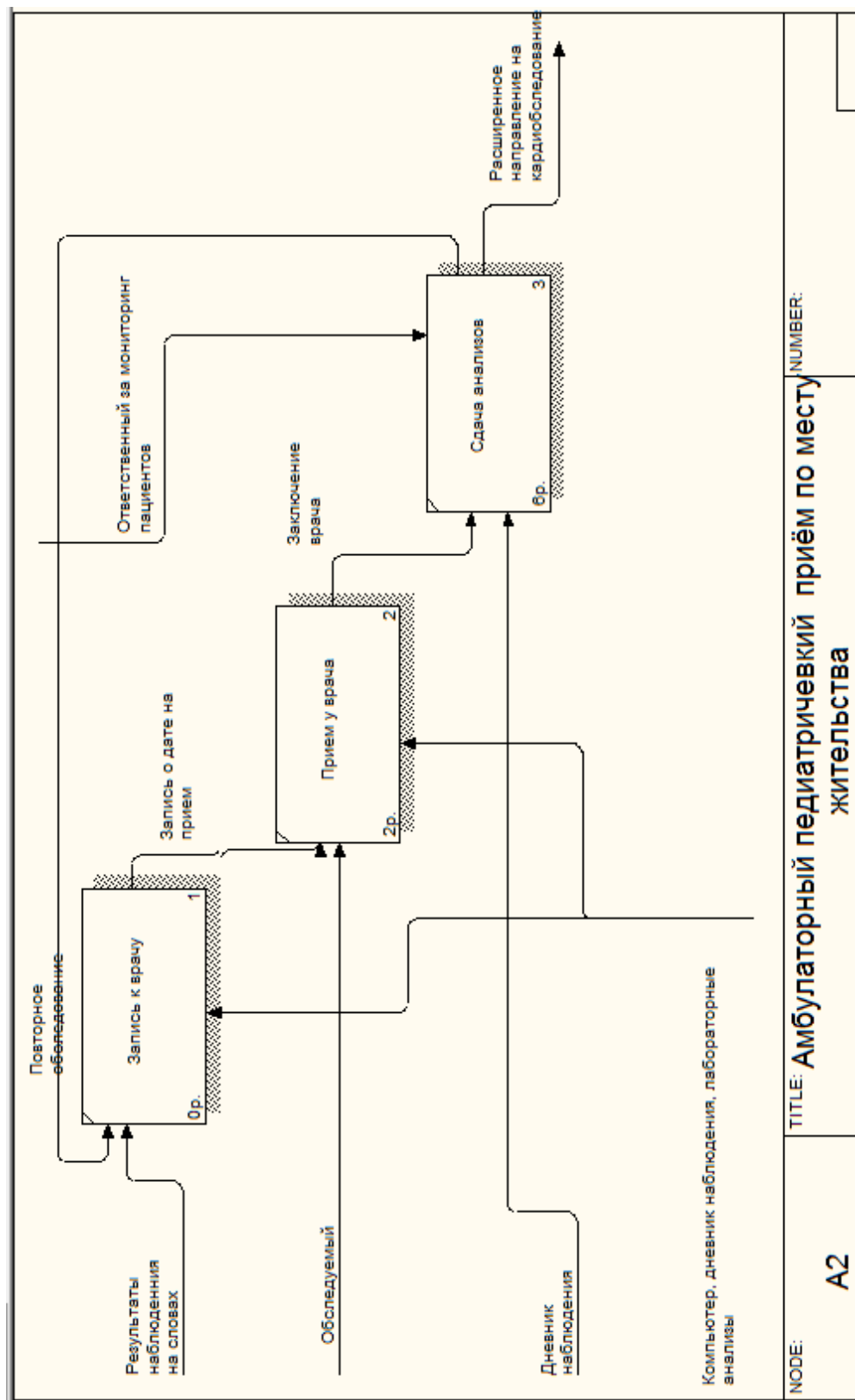


Рисунок А.3 – Прием у врача



## Тестирование событий

```

# Если событие меняет статус с busy на free ->
# продолжительность события вычисляется как разница в секундах
# между датой окончания и датой начала события
it 'change duration to equal difference between date_end
    and date_start if status change from busy to free' do
  e = create(:busy_event)
  e.status.should eq(:busy)
  e.duration.should eq(0)
  e.status = :free
  e.save!
  e.status.should eq(:free)
  e.duration.should eq((e.date_end - e.date_start).to_i)
end

# Если продолжительность события выставляется больше 0
# и статус события равен busy ->
# статус события меняется на free
it 'change status to free if duration set greater than 0' do
  e = create(:busy_event)
  e.duration = 2
  e.save!
  e.duration.should eq(2)
  e.status.should eq(:free)
end

# Изменение статуса приоритетнее чем изменение продолжительности события
it 'change in the status of priority than change duration' do
  e = create(:free_event)
  e.duration = 2
  e.status = :busy
  e.save!
  e.duration.should eq(0)
  e.status.should eq(:busy)
end

# Для новой сущности продолжительность вычисляется автоматически как
# разница между датой начала и датой окончания события
it 'calculate duration for new record' do
  e = build(:free_event)

```

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

					Дипломный проект	Лист
						98
Изм.	Лист	№ докум.	Подп.	Дата		

```

e.duration.should eq(nil)
e.save!
e.duration.should eq((e.date_end - e.date_start).to_i)
end

# Для уже сохраненной сущности продолжительность события не вычисляется
it 'don\'t calculate duration for saved record' do
  e = create(:free_event)
  new_duration = (e.date_end - e.date_start).to_i + 1
  e.duration = new_duration
  e.save!
  e.duration.should eq(new_duration)
end

# Проверяем создание дочерних событий
it 'should create child events' do
  d = DateTime.now.at_beginning_of_hour
  e = T.new({
    :date_start => d,
    :date_end => d + 2.hours,
    :description => 'some d',
    :summary => 'some s'
  })

  e.save!
  e.events.count.should eq(8)
end

# Проверяем что
# изменени статуса для дочернего события с free на busy ->
# уменьшает продолжительность родительского события
# изменени статуса для дочернего события с busy на free ->
# увеличивает продолжительность родительского события
it 'should subtract parent duration then status change from free to
  busy' do
  d = DateTime.now.at_beginning_of_hour
  e = T.new({
    :date_start => d,
    :date_end => d + 2.hours,
    :description => 'some d',

```

Подп. и дата	
Инв. № дубл.	
Взам. инв. №	
Подп. и дата	
Инв. № подл.	

Изм.	Лист	№ докум.	Подп.	Дата	<div>Дипломный проект</div>	Лист
						99

```

        :summary => 'some s'
    })

    e.save!
    ef = e.events.first

    equal_duration = e.duration - ef.duration
    ef.status = :busy
    ef.save!
    e.reload
    ef.status.should eq(:busy)
    e.duration.should eq(equal_duration)

    equal_duration = e.duration + (ef.date_end - ef.date_start)
    ef.status = :free
    ef.save!
    e.reload
    e.duration.should eq(equal_duration)
end

# Проверка изменения статуса события на другие статусы
context 'not available change status from busy' do
  it 'to close' do
    e = create(:busy_event)
    e.status = :close
    e.should_not be_valid
  end
end

context 'not available change status from process' do
  [:busy, :free].each do |s|
    it "to s" do
      e = create(:process_event)
      e.status = s
      e.should_not be_valid
    end
  end
end

context 'not available change status from close' do

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата					
Изм.	Лист	№ докум.	Подп.	Дата	<div style="text-align: center; font-size: 1.2em; font-weight: bold;">Дипломный проект</div>				
					Лист				
					100				

```

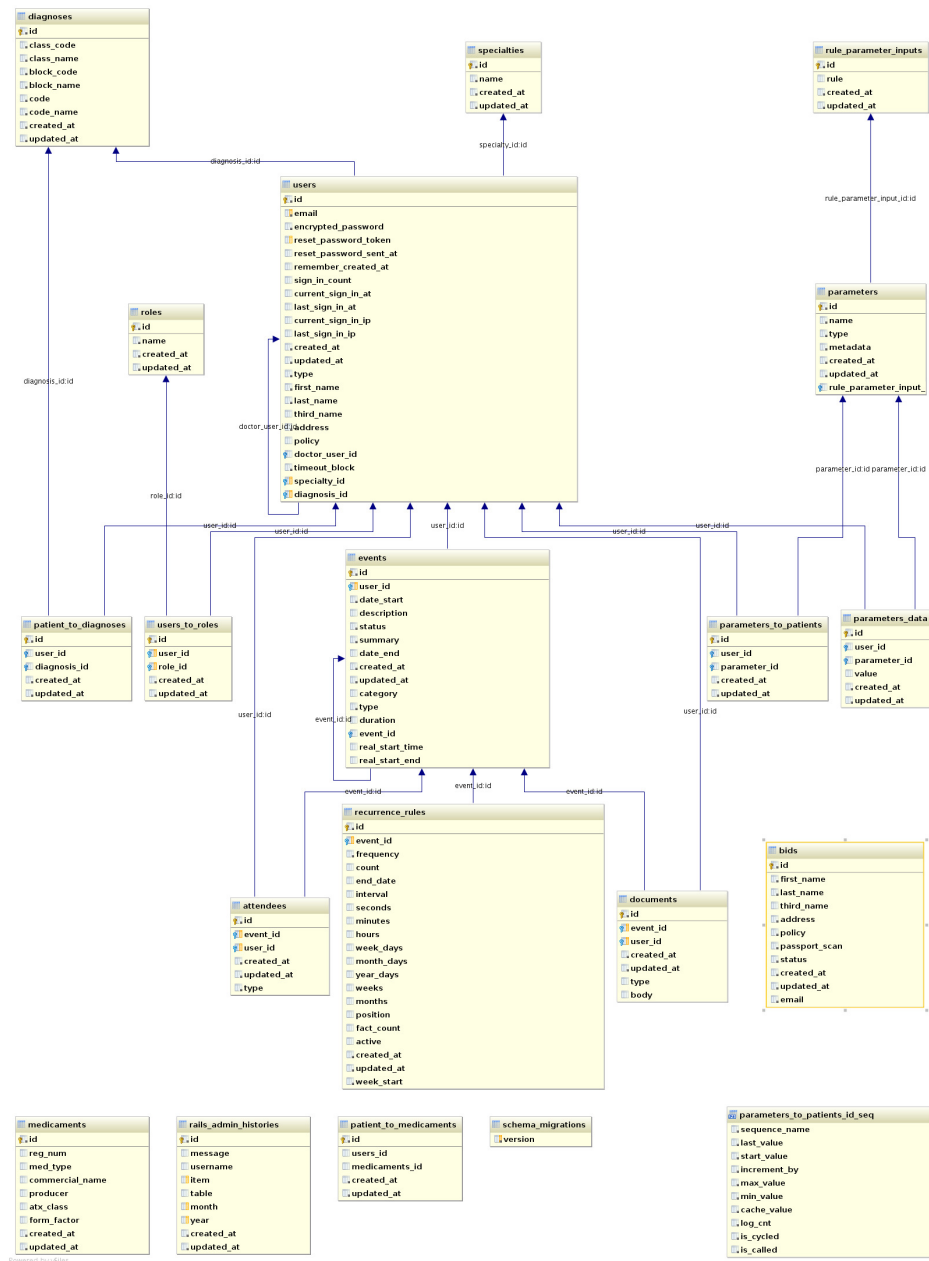
[:process , :busy , :free ].each do |s|
  it "to s" do
    e = create(:close_event)
    e.status = s
    e.should_not be_valid
  end
end

context 'not available change status from free' do
  [:process , :close ].each do |s|
    it "to s" do
      e = create(:free_event)
      e.status = s
      e.should_not be_valid
    end
  end
end
end

```

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата					
Изм.	Лист	№ докум.	Подп.	Дата	<div>Дипломный проект</div>				Лист
									101

Приложение В  
(справочное)  
Диаграмма базы данных



Инв. № подл.	Подп. и дата	Инв. № дубл.	Взам. инв. №	Подп. и дата	Инв. № подл.
Изм.	Лист	№ докум.	Подп.	Дата	

Приложение Г  
(справочное)  
Процесс мониторинга

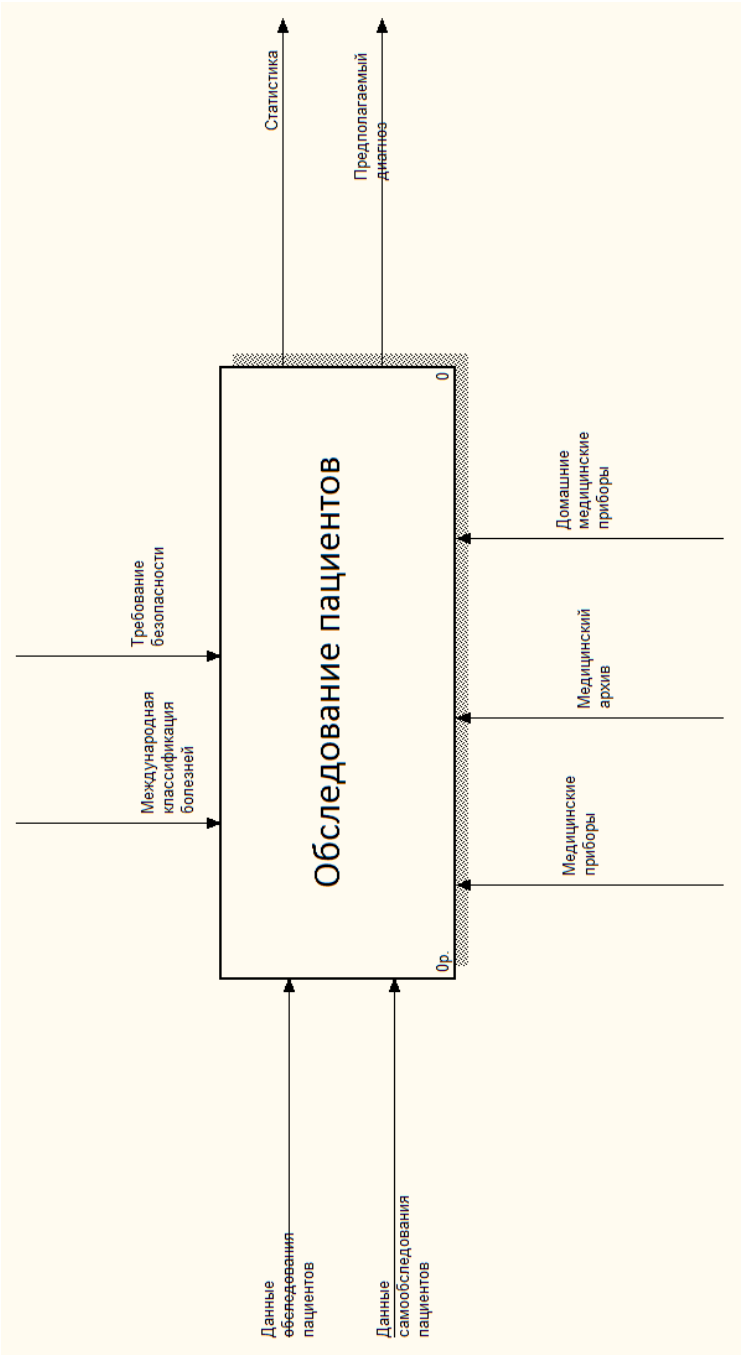


Рисунок Г.1 – Общая схема процесса мониторинга

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата	Дипломный проект	Лист
						103

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

# Дипломный проект

Лист  
104

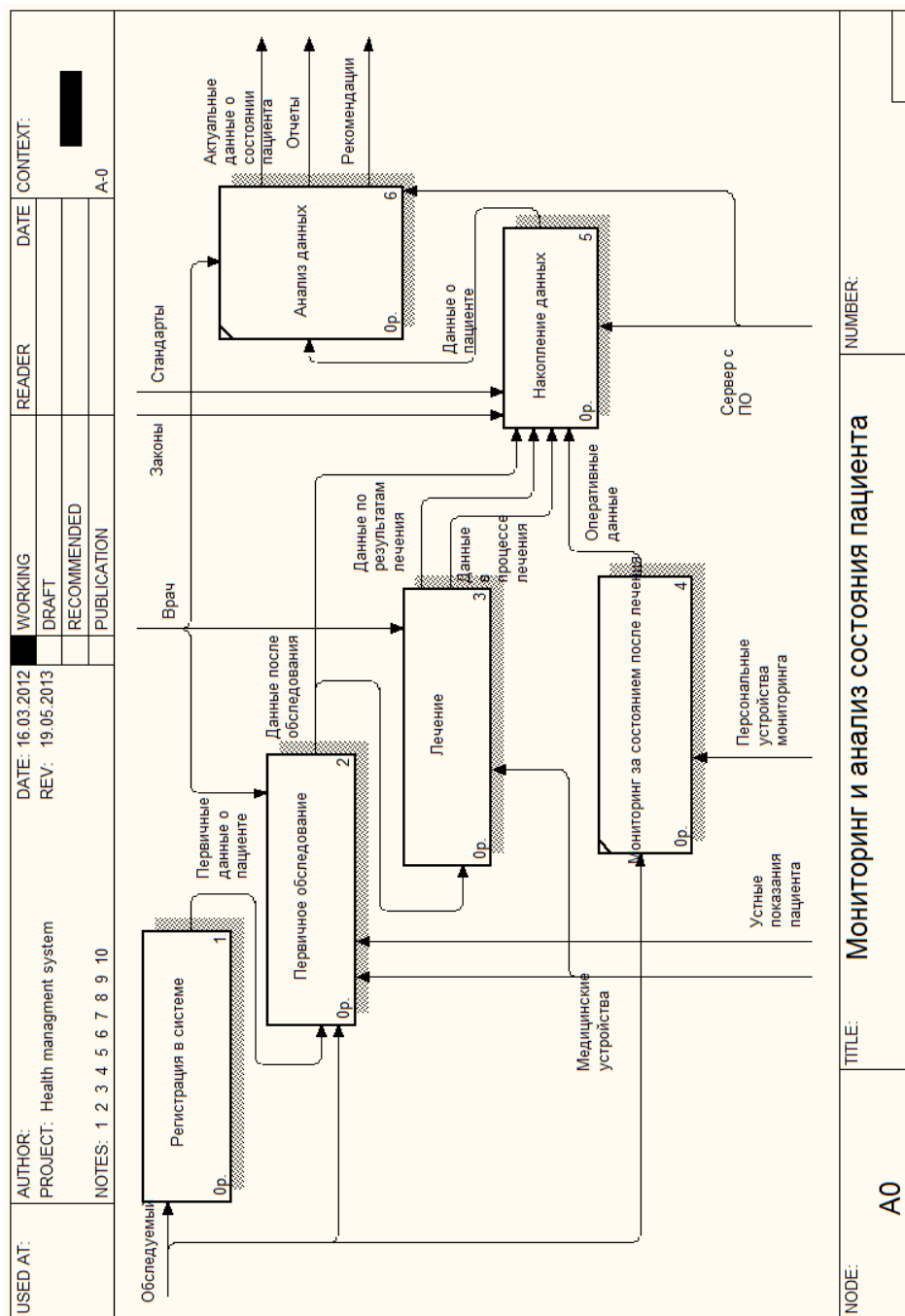


Рисунок Г.2 – Декомпозиция процесса мониторинга

Копировал

Формат А4



Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

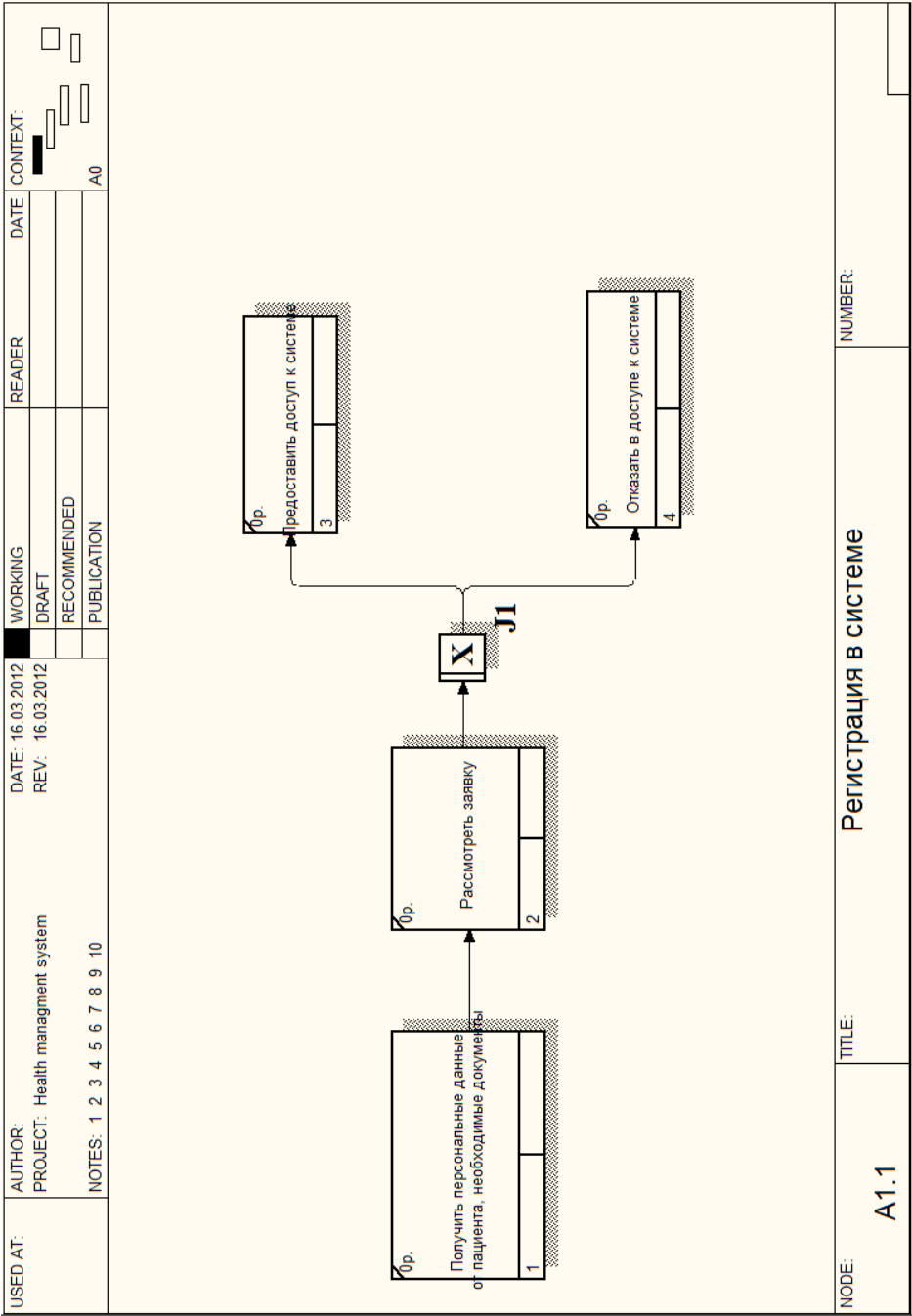


Рисунок Г.3 – Регистрация в системе

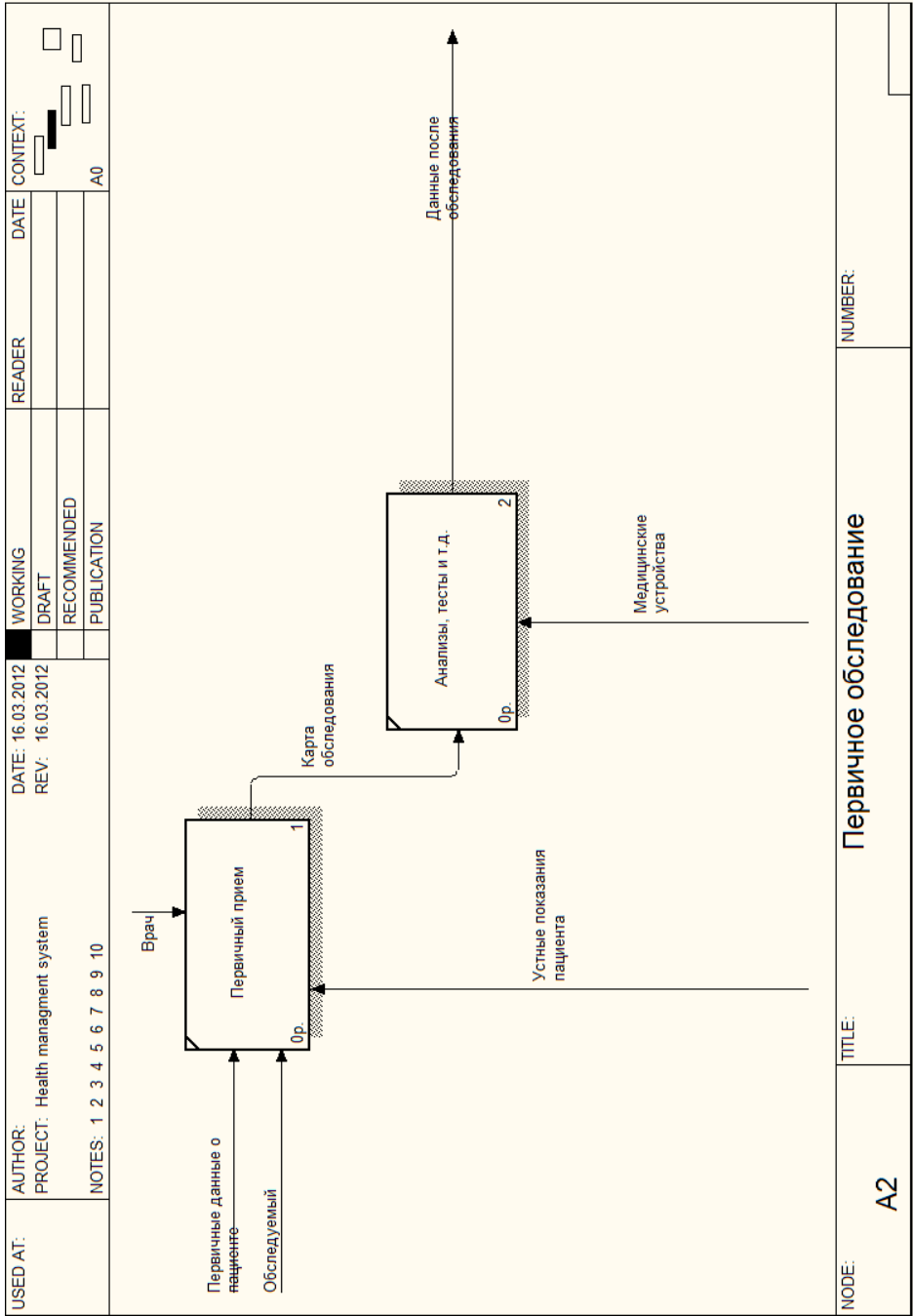


Рисунок Г.4 – Первичное обследование

Инв. № подл.	Подп. и дата	Взам. инв. №	Инв. № дубл.	Подп. и дата

Изм.	Лист	№ докум.	Подп.	Дата

# Дипломный проект

Лист

107

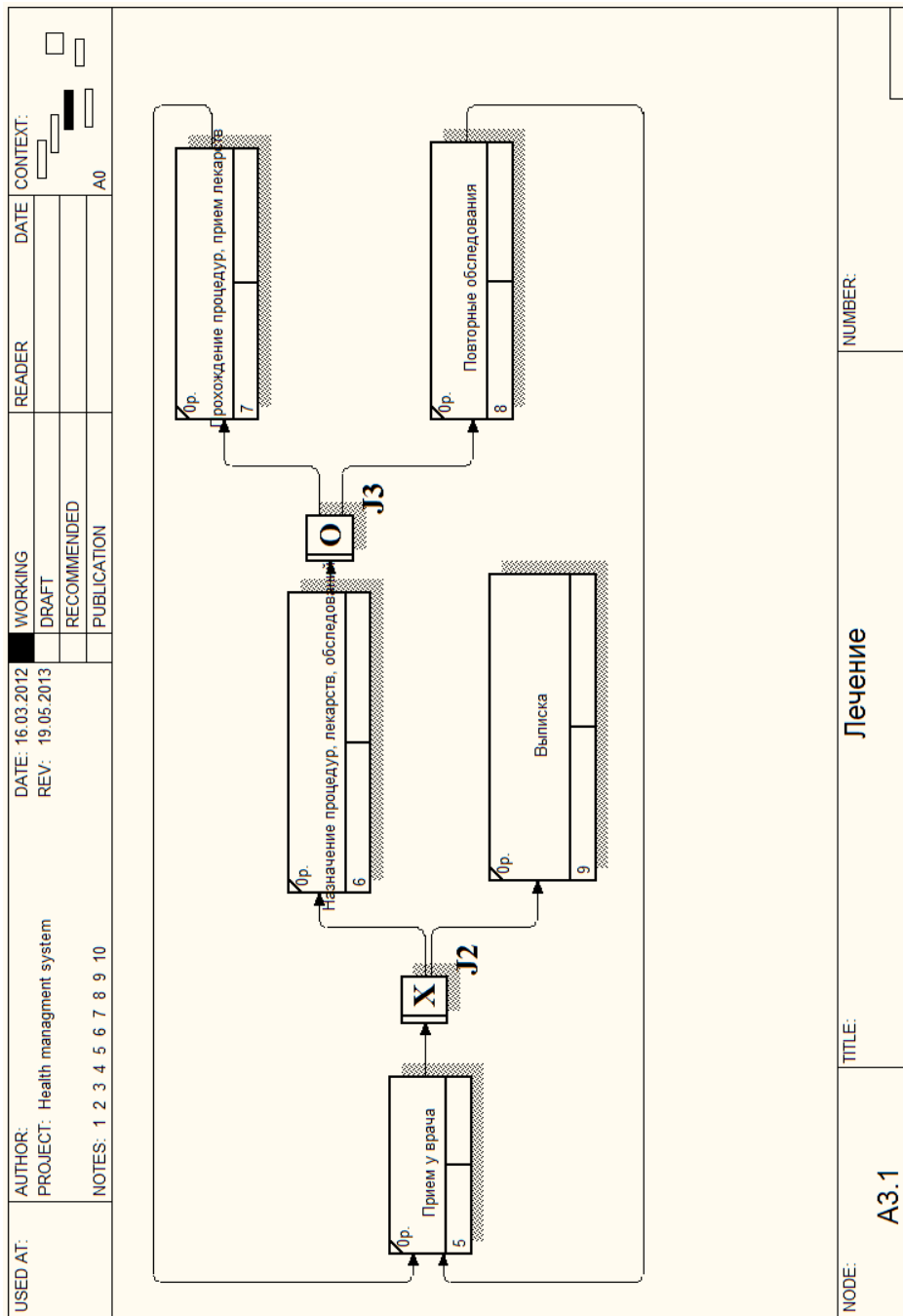


Рисунок Г.5 – Лечение

