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

Физико-технологический институт
Кафедра оптических и биотехнических систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА НА ТЕМУ:

«ПРИМЕНЕНИЕ КАСКАДНОЙ, ИТЕРАЦИОННОЙ И СПИРАЛЕВИДНОЙ МОДЕЛЕЙ ВНЕДРЕНИЯ ИНФОРМАЦИОННЫХ СИСТЕМ ПРИ РЕАЛИЗАЦИИ КЛЮЧЕВЫХ БИЗНЕС-ПРОЦЕССОВ ГОРОДСКОЙ БОЛЬНИЦЫ»

Студент:
Катасонова Н.С.

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

Москва – 2020

АННОТАЦИЯ

В рамках данной выпускной квалификационной работы рассмотрены и исследованы каскадная, итерационная и спиральная модели внедрения информационных систем. Для каждой модели выбрана соответствующая методология (ASAP – для каскадной, Scrum – для итерационной и RAD – для спиралевидной) и расписаны основные шаги при проведении разработки.

Опираясь на изученные данные, проведено описание ключевых бизнес-процессов городской больницы в модели «AS-IS» и составлена карта процессов. Также собран перечень пользовательских и законодательных требований к разрабатываемой системе.

В рамках реализации ключевых бизнес-процессов проведена пошаговая разработка согласно описанным ранее планам внедрения – составлен план разработки, затем оформлен перечень требований, представленных к реализации, осуществлено описание процессов до 3-го уровня описания в нотациях ARIS VACD и ARIS eEPC в моделях «AS-IS» и «TO-BE», и составлены соответствующие карты процессов. Кроме того, проведено проектирование архитектуры данных и схемы взаимодействия пользовательских интерфейсов в среде MS Visio. На основе полученных данных проведена разработка веб-приложения с использованием языков php, html и javascript, а полученные результаты прошли функциональное и нагрузочное тестирование.

С учетом полученных в работе данных проведен анализ полученных результатов и оценка целесообразности применения различных методологий в подобных проектах.

Отчет представлен на 154 страницы. В процессе подготовки отчета для наглядности в нем приведены 26 таблиц, 73 рисунка/снимков экрана и ссылки на 39 литературных источников.

ОГЛАВЛЕНИЕ

Введение

Темпы современной жизни, как и население планеты, возрастают с каждым годом. В связи с этим повышается объем работы с информацией для персонала различных государственных учреждений. При этом при работе с населением и информацией возникает необходимость в следующих аспектах:

- быстрое и качественное обслуживание населения;
- быстрая работа с данными;
- увеличение персонала и числа учреждений, соответствующее росту населения;
- доступность данных.

Следовательно, будет целесообразно создать систему, которая позволит персоналу, какого-либо заведения, облегчить работу не снижая ее качества. Наличие баз данных в городской больнице позволит решить многие вопросы, такие как:

- скорость обслуживания – отсутствие необходимости заполнения большого количества бумажной документации и наличие электронного устройства с доступом в Интернет позволит увеличить скорость работы с данными и решить вопрос отсутствия очередей;
- улучшение качества работы врачей – в связи с тем, что у врачей сокращается время в работе с заполнением бумажных документов и освобождается больше времени для работы с пациентами;
- доступность данных – при обращении в другое лечебное учреждение пациенту не нужно будет брать с собой медицинскую карту, достаточно просто показать ее на сайте; врачи и медсестры смогут работать и следить за состоянием пациентов дистанционно;

- удобство работы с данными – наличие связанных между собой баз данных позволит легко их дополнять или редактировать;
- сохранность данных – срок хранения существенно возрастает по сравнению с бумажными носителями.

Изучению вопросов автоматизации ИС посвящены труды Маглинец Ю.А. [1, 2]. В них рассматриваются различные понятия, относящиеся с ИС, требования к этим системам и всевозможные стандарты. Моделирование бизнес-процессов описывает руководитель компании IDS Scheer – производителя системы ARIS, А.В. Шеер [3, 4]. Подобные задачи решались в выпускных квалификационных работах (ВКР) Орешкиной А.М. [5], Катасоновой Н.С. [6]. Применение автоматизации бизнес-процессов в сфере телекоммуникационных систем рассматривается в книге Самуйлова К.Е. [7], в бизнесе – в книге Старовойтовой Т. Ф. [8], в строительстве – в книге Теличенко В. И. [9]. Учитывая значительный список авторов, занимающихся изучением вопросов автоматизации ИС, можно с уверенностью сказать, что разработка данной системы будет весьма актуальной и поможет объединить и улучшить предыдущие разработки и закрепить достижения в этой области. При этом необходимо заметить, что разработанная система не исключает возможности для легкого совершенствования ее функционала без радикальных изменений в структуре (например, система записи пациентов на прием к врачу и т.д.).

В отличие от предыдущей, в данной работе необходимо провести разработку данной системы не только при помощи каскадной модели внедрения ИС, но и двух новых моделей, рассмотреть соответствующие этим моделям методологии, их пошаговое выполнение и преимущества , а затем использовать в работе. После чего проанализировать преимущества и недостатки каждой из них и влияние выбора на конечный результат

основываясь на удобстве использования и показателях, полученных в процессе тестирования разрабатываемого ПО на различных этапах выполнения.

Цель и задачи

Целью данной работы является разработка программного обеспечения средствами PHP для автоматизации ключевых бизнес-процессов городской больницы на основе каскадной, итерационной и спиралевидной модели внедрения информационных систем (ИС). При разработке будет использована система управления базами данных (СУБД) – MySQL, и для программной реализации – html и php.

Для удобства работы с системой необходим простой интерфейс для создания и просмотра медицинской карты пациента, а также базы данных для хранения этих сведений. В процессе реализации необходимо решить следующие задачи:

- описать модели (каскадная, итерационная и спиралевидная) и методологии (ASAP, Scrum, RAD) внедрения ИС;
- провести проектирование бизнес-процессов до 3 уровня описания при помощи нотаций ARIS VACD и ARIS eEPC в моделях AS-IS и TO-BE;
- спроектировать архитектуру данных, структуру приложений с помощью MS Visio;
- реализовать ключевые бизнес-процессы с применением PHP по выбранным методологиям;
- провести нагрузочное и функциональное тестирование;
- провести сравнение методологий внедрения ИС.

Раздел 1. Описание методологий внедрения ИС по каскадной, итерационной и спиральной схеме

1.1. Описание методологии внедрения ИС по каскадной схеме

Каскадная модель (англ. waterfall model или «Водопад») – модель, описывающая разработку ПО (программного обеспечения), основной характеристикой которой является разбивка на этапы. При этом переход на последующий этап возможен исключительно при успешном окончании предыдущего (рисунок 1.1). Делает возможным быстрое создание системы без дополнительных расходов на организацию процесса разработки [10].

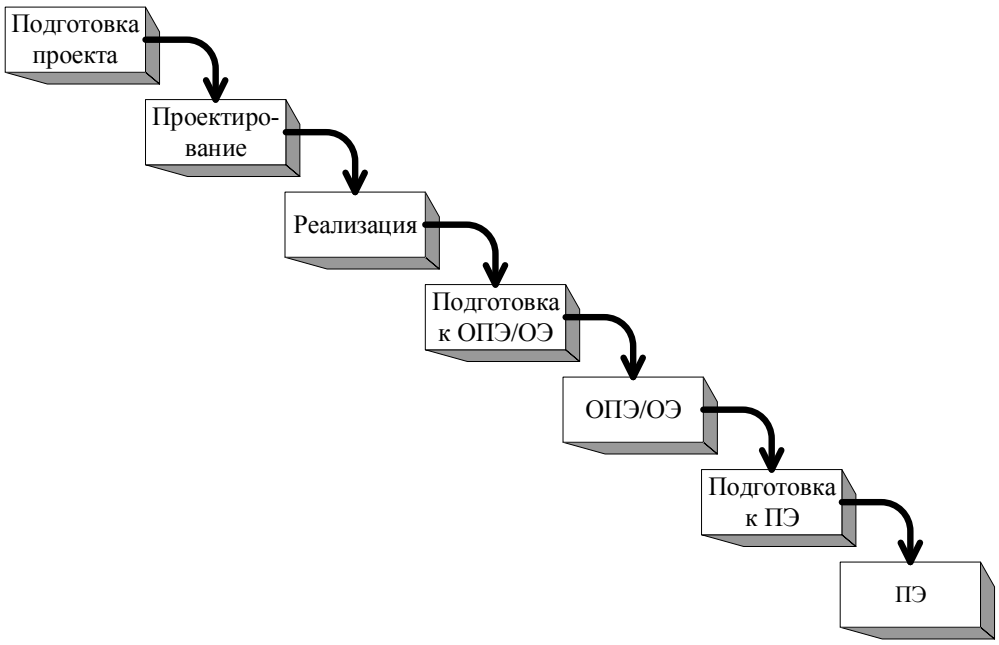


Рисунок 1.1 – *Каскадная схема внедрения ИС*

Каскадный метод реализации ИС имеет такие преимущества, как:

- неизменные требования во время всего ЖЦ (жизненного цикла);
- способность поэтапно устранять появляющиеся проблемы;
- простота и ясность шагов модели и легкость ее использования;
- облегчение менеджмента систем;

- понятность для восприятия заказчиков;
- высокая продуктивность для заданий с простыми и понятными, но трудными в осуществлении требованиями;
- высокая продуктивность для заданий, имеющих высокие запросы к качеству, не имеющих при этом жестких лимитов по графику работы и ее оплате.

Ключевым изъяном каскадного подхода стоит назвать значительную задержку с результатами, согласовываемыми с заказчиками только после окончания каждой ступени каскада. Требования к ИС остаются неизменными в виде ТЗ (технического задания) на протяжении выполнения работ. То есть, заказчик может прокомментировать только уже готовый продукт. Если результаты представлены недостаточно точно или во время длительного периода создания ПО они подверглись изменениям, то получится система, не отвечающая потребностям заказчиков. Недостатки каскадного метода также включают:

- трудности с конкретной формулировкой требований на начальной стадии и неосуществимость редактирования;
- линейность процесса создания, вследствие чего отсутствует возможность для возврата к предшествующим шагам для урегулирования появляющихся неполадок;
- промежуточный продукт не готов к работе;
- не имеющие аналогов системы невозможно гибко моделировать;
- проблемы сборки конечного продукта выявляются поздно – после интеграции всех результатов;
- заказчик участвует в разработке системы только при формировании требований и во время приемочных испытаний;

- пользователь не имеет возможности заранее оценить качество проведенных работ;
- сложности с распределением денежных ресурсов на проект из-за трудностей с разовой выплатой значительных сумм.

Применение модели максимально действенно при:

- создании проектов с четкими требованиями не изменяющимися во время ЖЦ, доступной реализацией и техническими методиками;
- создании проекта, направленного на разработку системы или продукта уже существующего типа;
- выполнении крупных проектов, в которых задействованы несколько крупных команд разработчиков.

На основе выбранной модели работа над системой будет также разделена на 7 этапов:

- Подготовка проекта – избрание схемы реализации, оценка предварительного масштаба. На первом этапе, исходя из темы дипломной работы, выбрана каскадная модель внедрения. Проанализированы ее преимущества и недостатки. Оценены возможности применения готового программного продукта в городской больнице. Определен объем выполняемых работ. Результат: применение каскадной модели рационально.
- Проектирование – анализ требований, планирование и одобрение методов решения, формирование списка доработок системы. Второй этап включает в себя общение с заказчиком, работу с заявленными и выявленными требованиями, поиск пути решения проблем по разработке системы. Результат: в ходе общения определены требования и способы их выполнения.

- Реализация – наладка и доводка системы, тестирование, документирование. На третьем этапе необходимо сосредоточиться на реализации программных компонентов для выполнения требований заказчика. Результат: готова промежуточная версия системы. Проведены необходимые тесты, система функционирует должным образом.
- Подготовка к ОПЭ (опытно-промышленная эксплуатация) /ПЭ (промышленная эксплуатация) – фиксация тестовых данных. На четвертом этапе необходимо собрать и проанализировать данные с предыдущих тестов. Результат: готова информация о работе промежуточной версии системы.
- ОПЭ/ОЭ (опытная эксплуатация) – фиксация и ликвидация изъянов. Пятый этап подразумевает тестовые запуски системы в выбранном учреждении, обнаружение и устранение дефектов. Появление готового программного продукта. Результат: в ходе тестовых запусков система работает нормально, дефекты устранены.
- Переход к ПЭ (промышленная эксплуатация) – техническая подготовка системы. На шестом этапе проводятся финальные приготовления перед окончательным внедрением системы. Последние тесты и устранение оставшихся дефектов. Результат: система готова к работе в городской больнице.
- ПЭ – обновление документации, передача в поддержку. На заключительном, седьмом, этапе происходит установка на рабочий сервер, написание инструкций и документации, обучение пользователей, техподдержка. Результат: система функционирует согласно требованиям заказчика.

1.2. Описание методологии внедрения ИС по итерационной схеме

Одной из альтернатив каскадной модели внедрения ИС является итерационная модель внедрения, представленная на рисунке 1.2. Проект имплементации КИС согласно предлагаемой модели состоит из шагов:

- идентификация, анализ и приоритизация изначальных требований;
- определение числа и продолжительности итераций разработки;
- доработка и кастомизация ИС, функциональное и интеграционное тестирование с последующей демонстрацией полученного продукта заказчику для уточнения требований, реализуемых на последующих итерациях, перенос решения в продуктивную среду (для каждой итерации);
- проведение приёмочного тестирования;
- миграция данных и обучение пользователей;
- переход к продуктивной эксплуатации и поддержка.

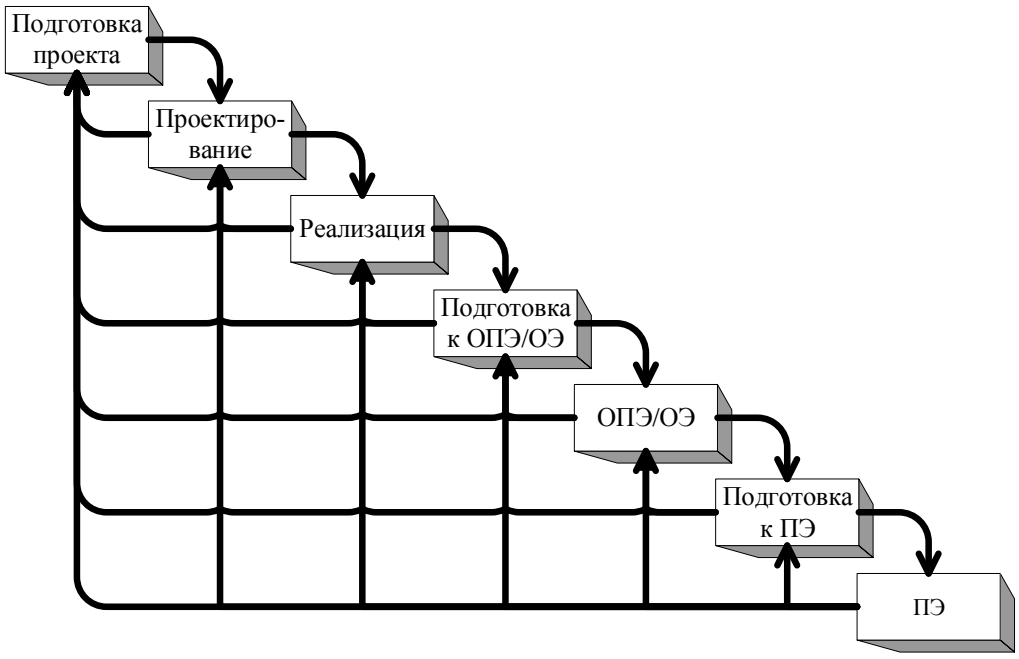


Рисунок 1.2 – *Итерационная схема внедрения ИС*

Создание сложных автоматизированных информационных систем (АИС) предполагает проведение согласований проектных решений, полученных при реализации отдельных задач. Подход к проектированию «снизу – вверх» обусловливает необходимость таких итераций возвратов, когда проектные решения по отдельным задачам объединяются в общие системные решения. При этом возникает потребность в пересмотре ранее сформировавшихся требований.

Итерационная модель предусматривает деление ЖЦ проекта на более мелкие задачи (итерации), которые являются мини-проектом, содержащим в себе все процессы разработки, но в уменьшенном, по сравнению с проектом, объеме. При этом в каждой итерации необходимо получить работающую версию системы, воплощающую в себе результаты проведенных ранее итераций. При этом результат последней из них включает в себя полный функционал продукта. В итоге, после каждой из итераций продукт получает приращение к его возможностям. К достоинствам итерационной модели стоит отнести:

- рабочее ПО появляется на ранних этапах;
- требования можно изменять в любой момент;
- простота тестирования для каждой итерации.

Из недостатков данного метода можно выделить:

- каждая фаза – самостоятельна, отдельные итерации не накладываются;
- могут возникнуть проблемы с реализацией общей архитектуры системы, поскольку не все требования известны к началу проектирования;
- выполнение каждого этапа затягивается на все время разработки;

- по причине большого числа итераций могут появиться рассогласования в выполнении решений и документации;
- сложности применения документации проекта на этапах внедрения и эксплуатации могут спровоцировать перепроектирование всей системы.

Итеративная модель состоит из четырех стадий, повторяющихся в каждой итерации (plan-do-check-act):

- На первом этапе проводится беседа с заказчиком, определяются и анализируются полученные требования. На первой итерации необходимо произвести сбор и анализ требований к системе и систематизировать их. На последующих итерациях – добавляются ранее не указанные требования, необходимые для выпуска готового продукта.
- На втором этапе происходит проектирование и дизайн создаваемой системы в соответствии с полученными требованиями или доработка уже существующего проекта для реализации новых. Для каждой итерации действия аналогичны – необходимо произвести проектирование данных и интерфейсов для выполнения требований из текущей итерации.
- На третьем этапе разрабатываются, внедряются и тестируются созданные программные решения. На данном этапе для каждой итерации необходимо программно реализовать задокументированные требования и произвести тестирование разработки для последующего устранения недочетов.
- На четвертом этапе производится анализ проведенных тестов, выявляются недостатки, которые затем составляют часть требований на новой итерации разработки или продукт переводится

в завершающий этап выпуска. На данном этапе работы будет проводиться анализ результатов тестирования и решение возможных недостатков текущей итерации путем переноса требований на последующую.

В качестве оптимального решения разработка будет проводиться в три итерации, включающих в себя приведенные выше этапы. По результатам каждой итерации принимается решение – будут ли использованы ее результаты для дополнения существующей функциональности в качестве входной точки для начала следующей итерации (т.н. инкрементальное прототипирование). В конечном итоге, достигается точка, в которой все требования были воплощены в продукте – происходит релиз [12].

1.3. Описание методологии внедрения ИС по спиральной схеме

Спиральная модель ЖЦ (рисунок 1.3), мини-водопад или спираль Боэма, предложена в середине 80-х годов. Она обеспечивает большую степень взаимосвязи с потребителем. Спиральная модель базируется на лучших свойствах водопадного процесса и метода прототипирования (макетирования), к которым добавляется новый элемент – анализ рисков. В этой модели цикл разработки разбивается на небольшие участки.

Каждый участок представляет собой водопадный процесс, в котором выполняются обзор требований, обновление документации и некоторая разработка кода. Результат текущей итерации является входным значением для следующей. При этом программная система создается по частям с использованием метода прототипирования. Под прототипом понимается действующий программный компонент, реализующий отдельные функции и внешние интерфейсы разрабатываемого ПО. С каждой итерацией по спирали

(продвижением от центра к периферии) строятся все более полные версии системы.

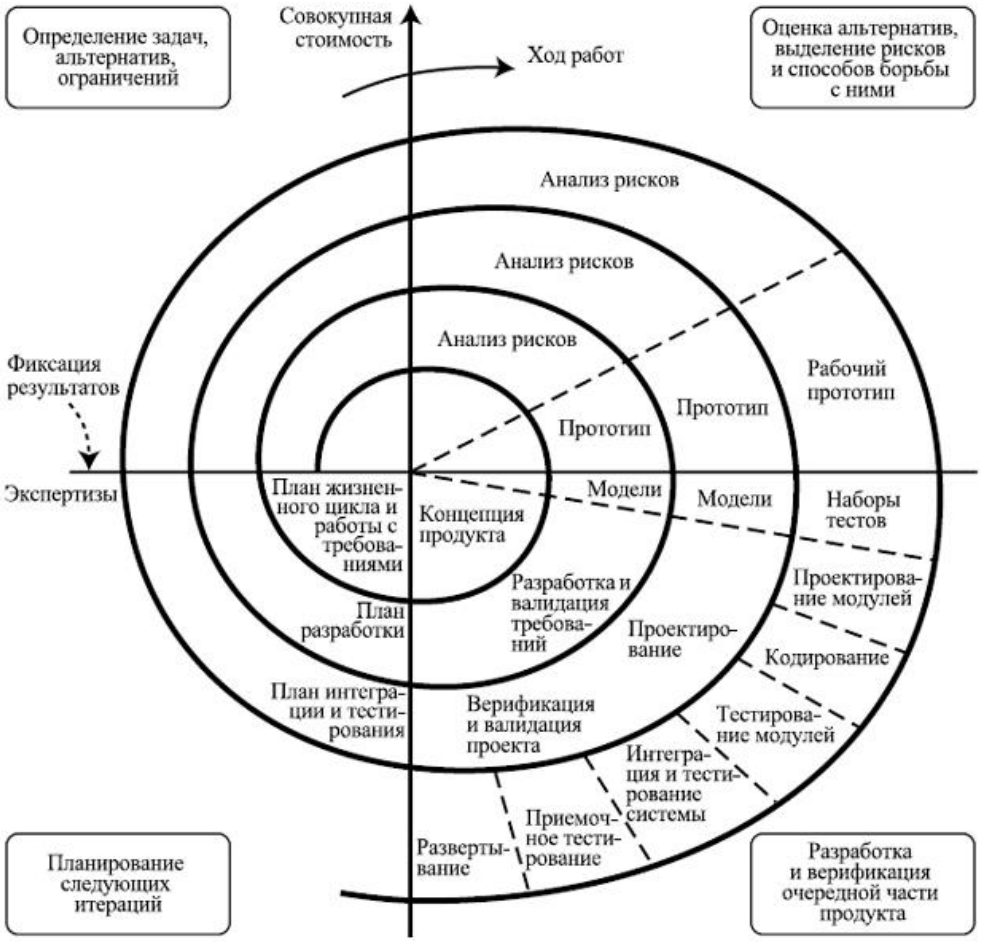


Рисунок 1.3 – *Спиральная модель внедрения ИС*

На каждой итерации происходит уточнение целей и характеристик проекта, оценивается качество полученных результатов и планируются работы следующей итерации. По завершению итерации производится тщательная оценка риска превышения сроков и стоимости проекта, чтобы определить необходимость выполнения еще одной итерации, степень полноты и точности понимания требований к системе, а также целесообразность прекращения проекта.

Спиральная модель избавляет пользователей и разработчиков ПО от необходимости полного и точного формулирования требований к системе на начальной стадии, поскольку они уточняются на каждой итерации. Таким образом, углубляются и последовательно конкретизируются детали проекта, и в результате выбирается обоснованный вариант, который доводится до реализации.

Модель отображает базовую концепцию, которая заключается в том, что каждый цикл представляет собой набор операций, которому соответствует такое же количество стадий, как и в модели каскадного процесса. Причем принимается во внимание каждая составляющая часть продукта, и каждый уровень сложности, начиная с общей формулировки потребностей и заканчивая кодированием каждой отдельной программы. К преимуществам модели можно отнести:

- спиральная модель разрешает пользователям «увидеть» систему на ранних этапах, что обеспечивается посредством использования ускоренного прототипирования в жизненном цикле разработки ПО;
- обеспечивается определение непреодолимых рисков без особых дополнительных затрат;
- эта модель разрешает пользователям активно принимать участие при планировании, анализе рисков, разработке, а также при выполнении оценочных действий;
- она обеспечивает разбиение большого потенциального объема работы по разработке продукта на небольшие части, в которых сначала реализуются решающие функции с высокой степенью риска, позволяющие устранить необходимость продолжения работы над проектом (таким образом, в случае необходимости становится

возможным прекратить работу над проектом, уменьшаются расходы);

- в модели предусмотрена возможность гибкого проектирования, поскольку в ней воплощены преимущества каскадной модели, и в тоже время, разрешены итерации по всем фазам этой же модели;
- обратная связь по направлению от пользователей к разработчикам выполняется с высокой частотой и на ранних этапах модели, что обеспечивает создание нужного продукта высокого качества;
- при использовании спиральной модели не нужно распределять заранее все необходимые для выполнения проекта финансовые ресурсы.

Недостатки спиральной модели внедрения ИС:

- если проект имеет низкую степень риска или небольшие размеры, модель может оказаться дорогостоящей. Оценка рисков после прохождения каждой спирали связана с большими затратами;
- модель имеет усложненную структуру, поэтому может быть затруднено ее применение разработчиками, менеджерами и заказчиками;
- серьезная нужда в высокопрофессиональных знаниях для оценки рисков;
- спираль может продолжаться до бесконечности, поскольку каждая ответная реакция заказчика на созданную версию может порождать новый цикл.

1.3.1. Оценка рисков

Оценка рисков – это совокупность аналитических мероприятий, позволяющих спрогнозировать возможность получения дополнительного предпринимательского дохода или определенной величины ущерба от

возникшей рисковой ситуации и несвоевременного принятия мер по предотвращению риска. Степень риска – это вероятность наступления случая потерь, а также размер возможного ущерба от него. Риск может быть:

- допустимым – имеется угроза полной потери прибыли от реализации планируемого проекта;
- критическим – возможны не поступление не только прибыли, но и выручки и покрытие убытков за счет средств предпринимателя;
- катастрофическим – возможны потеря капитала, имущества и банкротство предпринимателя.

Количественный анализ – это определение конкретного размера денежного ущерба отдельных подвидов финансового риска и финансового риска в совокупности. В абсолютном выражении риск может определяться величиной возможных потерь в материально-вещественном (физическом) или стоимостном (денежном) выражении.

Качественная оценка рисков – процесс представления качественного анализа идентификации рисков и определения рисков, требующих быстрого реагирования. Процесс проведения качественного анализа проектных рисков должен включать описательный, «инвентаризационный» аспект определения тех или иных конкретных видов риска данного проекта, выявления возможных причин их возникновения, анализа предполагаемых последствий их реализации и предложения по минимизации выявленных рисков. Специфической особенностью качественного анализа проектных рисков является определение стоимостного эквивалента гипотетических последствий возможной реализации отмеченных рисков. Поэтому качественный анализ является базой для проведения количественного анализа.

Задачи качественного анализа рисков состоят в том, чтобы субъективно оценить вероятность воздействия каждого риска, создать более короткий список рисков, определить критические риски и методы их устранения. Кроме того, на стадии качественного анализа принимается решение о судьбе проекта: продолжать проект или закрывать. Проведение качественного анализа основывается на сборе информации об условиях реализации проекта и факторах, на него влияющих.

Так называемые владельцы рисков (risk owners) – это сотрудники, которым руководитель проекта поручает наблюдать за триггерами некоторого определенного риска, становятся владельцами рисков в силу специфических экспертных знаний относительно той или иной проблемы или в связи с тем, что они обладают определенным контролем над специфическим риском. Обычно чем раньше в процесс управления рисками вводится владелец риска, тем лучше.

Далее необходимо определить степень воздействия на проект каждого риска. Для этого необходимо понять, какие шкалы степени воздействия рисков будут использованы и какие методы качественного анализа могут применяться. Шкалы представляют собой определенные наборы степеней воздействия рисков на проект в целом. На данном шаге шкала воздействия определяется субъективно.

Кроме степени влияния, необходимо определить вероятность возникновения риска. На этом шаге вероятность также определяется субъективно. Необходимо помнить тот факт, что риск не может быть вероятен на 100% или даже на 80%. Такая вероятность выводит проблему из разряда рисков и переводит в разряд фактов, а потому должна быть учтена в плане проекта.

Далее риски необходимо отсортировать. Суть метода сортировки состоит в том, чтобы распределить риски по специальной карте (другое ее название – PI-матрица). Карта должны выглядеть так, как показано в таблице 1.2. Обычно все идентифицированные риски распределяются между сотрудниками группы по работе с рисками. За риск, как правило, отвечает тот, кто идентифицировал данный риск. Риски, определенные теми, кто не присутствует при данной процедуре, делятся поровну между всеми остальными участниками. Затем участники распределяют имеющиеся у них риски по определенным квадратам, то есть ранжируют вероятности и степени влияния данных рисков.

Таблица 1.2 – Карта сортировки рисков

Вероятность	10									
	9									
	8									
	7									
	6									
	5									
	4									
	3									
	2									
	1									
		1	2	3	4	5	6	7	8	9	10
	Степень воздействия										

Кроме процедуры сортировки рисков необходимо проранжировать риски – определить RR (risk ranking) для каждого риска. Формула 1.1 предназначена для определения RR.

RR = Вероятность риска * Степень воздействия риска. (1.1)

Самое важное на этом шаге – принять решение по поводу пороговых величин рисков, которые будут участвовать в дальнейшем рассмотрении. Это сложный вопрос, по которому трудно дать конкретные рекомендации.

Огромную роль здесь играет опыт руководителя проекта, а также уровни рисков, которые приняты как пороговые в компании.

1.3.1.1. Стратегии реагирования на негативные риски

После сортировки и определения пороговых величин необходимо определиться со стратегией реагирования на выявленные риски.

1.3.1.1.1. Уклонение

Уклонение от риска предполагает изменение плана управления проектом таким образом, чтобы исключить угрозу, вызванную негативным риском, оградить цели проекта от последствий риска или ослабить цели, находящиеся под угрозой (например, расширить рамки расписания или уменьшить содержание проекта). Некоторых рисков, возникающих на ранних стадиях проекта, можно избежать при помощи уточнения требований, получения информации, улучшения коммуникации или проведения экспертизы. Риски избегаются путем простого невыполнения части проекта.

Примером стратегии уклонения является использование проверенной технологии вместо недавно разработанной, еще не отработанной технологии, что, вероятно, предотвратит технический риск. Выбор поставщика из политически более стабильного региона снизит вероятность того, что политические риски поставщика повлияют на поставки нашего проекта.

Проработка нескольких альтернативных направлений создания продукта на ранних стадиях технологических проектов, которые впоследствии определят ключевое направление, позволит избежать получения продукта, который не будет соответствовать целям проекта. При выборе стратегии уклонения команда проекта несет затраты до реализации рискового события, причем эти затраты меньше возможных последствий риска с учетом его вероятности.

1.3.1.1.2. Передача и разделение

Передача и разделение риска подразумевает переложение негативных последствий угрозы с ответственностью за реагирование на риск на третью сторону, частично или полностью. Передача риска просто переносит ответственность за его управление другой стороне, риск при этом не устраняется. Передача ответственности за риск наиболее эффективна в отношении финансовых рисков. Передача риска практически всегда предполагает выплату премии за риск стороне, принимающей на себя риск. Инструменты передачи рисков включают в себя, в частности:

- страхование;
- гарантии выполнения контракта;
- поручительства и гарантийные обязательства;
- прописывание условий в контракте;
- прочее.

Условия передачи ответственности за определенные риски третьей стороне могут определяться в контракте. Во многих случаях в контракте с оплатой фактических издержек затраты на риски могут перекладываться на покупателя, а в контракте с фиксированной ценой риск может перекладываться на продавца, если разработка проекта уже находится в стабильном состоянии.

Передача риска страхованием вероятна, если есть возможность оценки риска и страховая компания готова принять его на себя за определенную премию. Либо контрагент может и готов управлять риском, опять же, получив за это определенную премию за принятие рисков проекта. Проблема варианта страхования состоит в том, что в ряде случаев для рискового брокера трудно определить события и условия риска, особенно

если он не знаком со спецификой проекта, или же процесс оценки ресурсоемок.

Более мягким вариантом передачи является разделение рисков, которому уделяется все больше внимания в последние годы. При данной стратегии ответственность за риск несут обе стороны договора при реализации проекта. Разделение рисков между поставщиком и командой проекта инициирует взаимовыгодный процесс улучшения, побуждая поставщиков к инновациям. Однако подобный метод несет в себе дополнительный риск того, что предлагаемые инновации не заработают, при этом инновационный процесс отвлекает ресурсы, как у поставщика, так и у команды проекта.

При реализации стратегии разделения или передачи риска команда проекта несет затраты до реализации рискового события.

1.3.1.1.3. Снижение

Стратегия снижения (смягчения) рисков предполагает:

- понижение вероятности реализации риска;
- понижение последствий негативного рискованного события до приемлемых пределов – риск либо не сбудется, либо сбудется, но с меньшими последствиями.

Принятие предупредительных мер по снижению вероятности наступления риска или его последствий часто оказывается более эффективным, нежели усилия по устранению негативных последствий, предпринимаемые после наступления события риска. В качестве примеров мероприятий по снижению рисков можно привести:

- Внедрение менее сложных процессов, структурное упрощение, детализацию процессов до такого уровня, который позволит достаточно снизить вероятность реализации риска. Помимо

упрощения процессов, вероятность рисков может снизить более детальное описание процессов или применение дополнительных программ обучения персонала проектов.

- Проведение большего количества испытаний или реализацию прототипов, на которых производится отработка основных решений проекта. Например, при реализации промышленных проектов возможно выделение опытной группы или участка, на котором производится проверка разработанных технических решений.
- Выбор поставщика, поставки которого носят более стабильный характер. Выбор может производиться на основании данных архивов прошлых проектов.

Для снижения рисков может потребоваться разработка прототипа, на основе которого производится пропорциональное увеличение вероятности риска от стендовой модели до процесса или продукта. Если невозможно снизить вероятность, ослабление риска должно быть направлено на последствия риска, а именно – на те связи, которые определяют их серьезность. Например, разработка дублирующей подсистемы может сократить последствия отказа основной системы. В случае выбора стратегии снижения команда проекта несет затраты до реализации рискового события.

1.3.2. Применение спиральной модели в работе

Как показано на рисунке 1.3, в каждый квадрант модели входят целевые и вспомогательные действия. Ниже перечислены эти квадранты.

- Определение целей, альтернативных вариантов и ограничений. Происходит определение целей проекта, выполняемых функций, возможностей для корректировки. Определяются альтернативные варианты выполнения этого фрагмента продукта; выявляются ограничения, которые налагаются на возможность применения

найденных альтернатив; оформляется документация, связанная с первоначальными рисками.

- Оценка альтернативных вариантов, идентификация и разрешение рисков. Происходит оценка найденных ранее альтернатив, определяются и разрешаются риски.
- Разработка продукта следующего уровня. На данном этапе происходит создание и анализ проекта, разработка и тестирование, причем на каждом витке спирали версия ПО будет все более соответствовать требованиям заказчика, в конечном итоге приводя к реализации полного функционала. Изменения в коде неизбежно сопровождают процесс разработки программного обеспечения. Добавляется новая функциональность, вносятся изменения в существующую, устраняются дефекты. При этом любые изменения могут затронуть уже существующую функциональность, которая исправно работала. Проверить, чтобы изменения не «поломали» рабочее ПО – задача регрессионного тестирования. Цель регрессионного тестирования – удостовериться в том, что существующая функциональность не была затронута изменениями в коде.
- Планирование следующей фазы. На данном этапе происходит анализ проделанной работы, планирование задач на следующий этап и оценка рисков перехода на него. Начало разработки происходит из центра схемы в 1 квадранте и далее проходит полную спираль. При этом отсутствует ограничение на число циклов, которое варьируется в зависимости от проекта.

Стоит отметить, что разработка происходит несколько позже, чем в других моделях. Задержка в разработке связана с тем, что перед ее началом

необходимо проанализировать и минимизировать имеющиеся риски с помощью последовательных уточнений требований у заказчика на каждом витке спирали. Это конкретизирует пожелания и повысит качество выполняемой работы. По причине большей проработки модели особую часть в ней занимают анализ и оценка рисков и анализ альтернатив, которые позволяют, в случае необходимости, либо продолжить разработку, либо завершить ее после текущего этапа. Для оптимизации процесса разработки по спиралевидной модели внедрения, он будет проводиться в рамках 3 витков спирали, каждый из которых будет включать в себя действия по прохождению каждого квадранта спирали.

1.4. Анализ прикладных методов ASAP, Agile Scrum и RAD

Для улучшения рабочего процесса, конкретизации работ и, соответственно, улучшения результата, необходимо в соответствие каждой модели внедрения ИС выбрать методологию, базирующуюся на нужной модели, конкретизирующую ее и позволяющую взять лучшее для разработки.

1.4.1. Методология ASAP

На сегодняшний день SAP является одним из крупнейших в мире поставщиков ИС, которые применяются бизнесом. Многие решения SAP очень сложны, поскольку охватывают все предприятия (к примеру, SAP ERP, охватывающая все направления деятельности компании). В связи с этим компания разработала методологию внедрения своих информационных систем. Однако эта методология может использоваться и при внедрении других ИС. Методология получила название Accelerated SAP, или, сокращенно, ASAP.

ASAP – методология быстрого внедрения и постоянной оптимизации – состоит из методологии Сетевого графика (Roadmap), который связан с

такими инструментами, как IMG (Implementation Guide, «Руководство по внедрению»), причем ASAP задумывалась специально для средних и малых предприятий, которые не могут отводить на внедрение длительное время.

Методология ASAP состоит из множества списков контрольных вопросов, таблиц, опросных листов, ответов, шаблонов документов, рекомендаций и т. д. Кроме того, в ASAP предусмотрены руководства, средства обучения и акселераторы по огромному диапазону технических вопросов, связанных с инфраструктурой, установкой и операциями SAP. Различные обзоры и списки контрольных вопросов, имеющиеся в распоряжении ASAP, контролируют не только ход собственно проекта, но также стабильность и интеграцию системы на всех стадиях проекта. Методология Сетевого графика также включает в себя задачи по управлению изменениями и акселераторы, необходимые для управления изменениями на предприятии, вызванными внедрением SAP.

Сетевой график выступает как проводник проекта, который уточняет этапы, необходимые рубежи и задает общий темп всего проекта с целью получения работоспособной системы в максимально сжатые сроки, с максимальным качеством и в рамках бюджета. Сетевой график ASAP состоит из следующих этапов: подготовка проекта, концептуальное проектирование, реализация, окончательная подготовка, запуск и поддержка, оптимизация (рисунок 1.4).

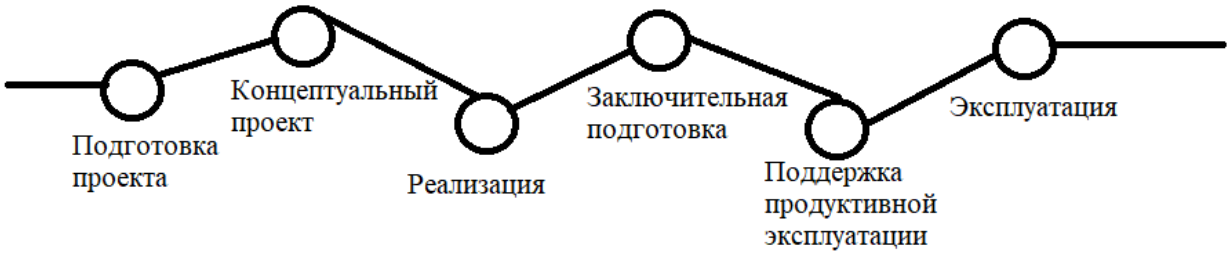


Рисунок 1.4 – *Маршрутная карта ASAP*

1.4.1.1. Подготовка проекта (Project Preparation)

Цель данной стадии – обеспечить планирование и подготовку проекта внедрения SAP. Среди ключевых задач, которые необходимо решить в процессе подготовки проекта:

- Определение целей и перспектив проекта.
- Максимально точная оценка объема внедрения.
- Определение стратегии внедрения.
- Определение общего графика работ по проекту и последовательности внедрения системы.
- Определение организационной структуры проекта и комитетов.
- Распределение ресурсов.

Точные ответы на эти вопросы в начале внедрения обеспечивают эффективное выполнение проектных работ и являются залогом успешного внедрения SAP.

1.4.1.2. Концептуальный проект (Business Blueprint)

Цель данной фазы – создание Концептуального проекта (Business Blueprint), который представляет собой подробно документированные технические и бизнес-требования заказчика к системе, за счет чего достигается полное понимание представления заказчика об организации бизнес-процессов в рамках системы SAP. На этапе концептуального проектирования определяется стратегия создания информационной системы, обеспечивающая быструю окупаемость инвестиций. Разработка концептуального проекта позволяет не только иметь программу действий при поэтапном внедрении, но и получить существенные экономические выгоды за счет усовершенствования существующей системы управления

предприятием. В рамках концептуального проектирования решаются следующие задачи:

- Определение и анализ бизнес-процессов.
- Формализация и документирование требований к будущей системе.
- Детализация и утверждение окончательного объема проекта.
- Выработка архитектуры системы и определение основных проектных решений.
- Анализ и формирование стратегии обучения.
- Предоставление заказчику рекомендаций по совершенствованию бизнес-процессов и итогового плана внедрения.

Для документирования и отслеживания требований к системе будет использоваться матрица отслеживания требований, которая представляет собой таблицу, которая связывает требования (функциональные) с их происхождением (пользовательские требования) и отслеживает их на протяжении жизненного цикла проекта. Применение матрицы отслеживания требований помогает удостовериться, что каждое требование увеличивает ценность бизнеса, связывая его с целями бизнеса и проекта. Это позволяет отслеживать требования на протяжении жизненного цикла проекта, что помогает удостовериться в том, что требования, одобренные в документах по требованиям, выполнены в конце проекта. Наконец, матрица отслеживания требований обеспечивает структуру для управления изменениями содержания продукта [22].

Параметры, связанные с каждым требованием, могут быть записаны в матрице отслеживания требований. Данные параметры помогают определить ключевую информацию относительно требований. Типичные параметры, используемые в матрице отслеживания требований, могут включать в себя:

- уникальный идентификатор;

- текстовое описание требования;
- обоснование включения в список требований;
- владельца;
- источник;
- приоритет;
- версию;
- текущий статус;
- дату выполнения.

Далее разрабатывается пакет работ по созданию прототипа решения. При этом усилия сосредотачиваются на процессах, которые могут быть сконфигурированы без дополнительного программирования или расширений системы SAP. Реализация требований, нуждающихся в дополнительном программировании или расширениях, происходит в отдельных пакетах работ фазы реализации.

1.4.1.3. Реализация (Realization)

На этапе реализации проекта проектная группа осуществляет конфигурацию системы с учетом требований Концептуального проекта посредством IMG (руководства по внедрению), необходимую доработку и тестирование системы. Тестирование системы на соответствие требованиям заказчика проводится в отдельной системе SAP для контроля качества и осуществляется в два этапа:

- Функциональное тестирование.
- Интеграционное тестирование.

На этапе функционального тестирования проверяются функции каждой из подсистем. На этапе интеграционного тестирования проверяется работоспособность сквозных бизнес-процессов. Результаты тестирования фиксируются в протоколе и на их основе выявляются несоответствия

системы требованиям, и производится устранение замечаний. Результатом этого этапа должна быть полностью сконфигурированная и проверенная система SAP, которая отвечает всем требованиям компании.

1.4.1.4. Заключительная подготовка (Final Preparation)

В рамках данного этапа система проходит подготовку к промышленной эксплуатации. Решаются все исключительные ситуации и устраняются нестыковки. Проводится нагрузочное тестирование, когда проверяется устойчивость системы к высоким нагрузкам на сервер. Также производится миграция данных из исторических систем компании-заказчика в систему SAP. Консультанты SAP проводят обучение ключевых и конечных пользователей работе с системой. В частности, разрабатываются инструкции пользователей, учебные материалы, планируется состав групп и сроки обучения, а в завершении проводится итоговая аттестация пользователей.

1.4.1.5. Запуск и поддержка (Go Live Support)

Это фаза решения вопросов, связанных с запуском системы. Проводится проверка готовности к запуску, а также устраняются возможные проблемы. Консультанты SAP осуществляют ежедневную поддержку пользователей, а также устраняют ранее скрытые ошибки и несоответствия в системе. При необходимости по согласованию с Заказчиком система может быть дополнена функциями, не предусмотренными согласованным ранее объемом проекта. В таком случае в проектную документацию вносятся изменения в соответствии с новыми требованиями к развитию готовой системы и выполнятся доработка системы.

1.4.1.6. Оптимизация (Run SAP)

Основная цель этого этапа является обеспечение надежности и оптимизации решения. Центр поддержки осуществляет клиентскую

поддержку и мониторинг системы для выявления процессов, которые необходимо оптимизировать, например:

- Оптимизация документации решения.
- Оптимизация реализации решения.
- Оптимизация шаблона.
- Оптимизация тест управления.
- Оптимизация обслуживания, апргейда и др.

Использование этой четкой и пошаговой схемы позволяет сокращать время внедрения, снижать затраты и минимизировать проектные риски, что так важно для Заказчика.

1.4.2. Методология Agile Scrum

В отличие классического Project Management (PM), когда проект жестко регламентирован заранее установленными требованиями (контрактами), Agile предполагает быстроту реагирования, а также гибкую адаптацию к внешним и внутренним изменениям. Это достигается с помощью итеративной разработки продукта и эффективного межличностного общения. В водопадной (каскадной) модели PM, которая считалась стандартом де-факто, проект состоит из функциональных задач, где каждая последующая работа четко регламентирована и начинается строго после окончания предыдущей, например, тестирование начнется только после того, как написан весь код. Жесткая определенность и обилие регламентирующей документации обусловливают длину производственного цикла. При этом продукт считается готовым лишь после выполнения всех этапов. Когда говорят о методологии Scrum, чаще всего имеют ввиду гибкую методологию разработки ПО, построенную на основе правил и практик Scrum (рисунок 1.5).

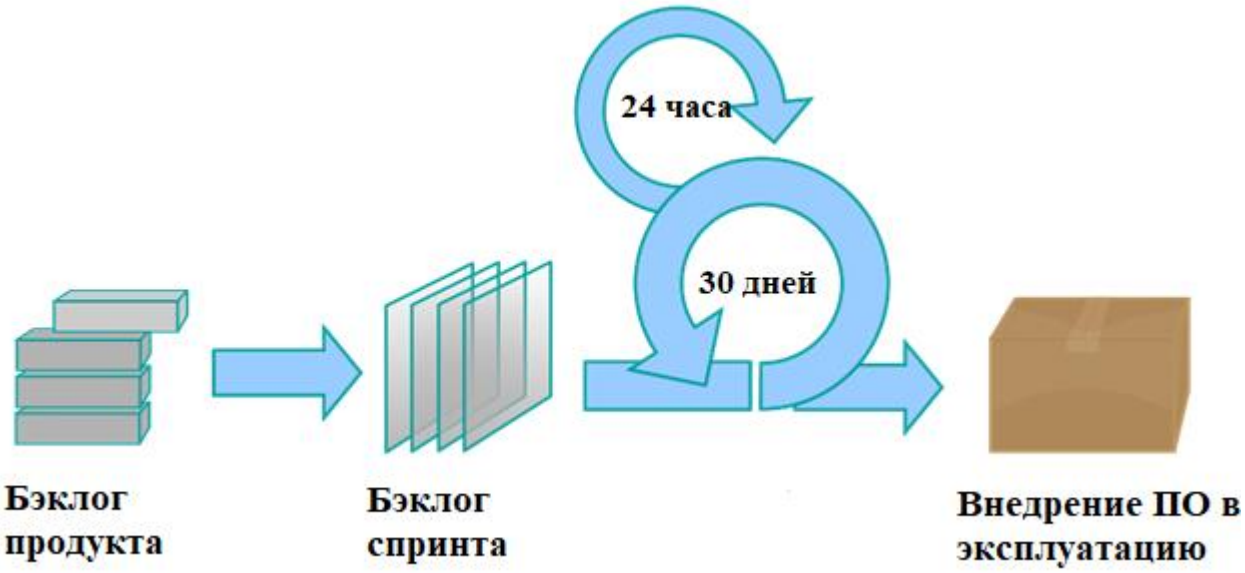


Рисунок 1.5 – *Схема модели Agile Scrum*

В настоящее время, Scrum является одной из самых популярных методологий разработки ПО. Согласно определению, Scrum – это каркас разработки, с использованием которого люди могут решать появляющиеся проблемы, при этом продуктивно и производя продукты высочайшей значимости (с точки зрения клиента) [16].

В Scrum над проектом работает команда профильных технических специалистов (например, аналитик, программист, тестировщик, администратор) вместе с владельцем продукта (product owner) и модератором (scrum-мастер). Product owner аккумулирует бизнес-требования, соединяет команду исполнителей с заказчиком и следит за развитием проекта. Scrum-мастер управляет процессом организации разработки по Agile-принципам: проводит общие собрания (meetings, митинги), мотивирует и поддерживает команду.

В Scrum рабочий процесс делится на равные периоды от 1 до 4-х недель (спринты), в зависимости от проекта и команды. Перед стартом

каждого спринта на митинге формулируются его задачи, а в конце обсуждаются результаты. Краткосрочность и измеряемость спринтов позволяет эффективно управлять проектной деятельностью, не перегружая участников проекта авралами.

Бэклог продукта (product backlog) – это упорядоченный набор элементов, очередь задач, перечень всех функций, которые заинтересованные люди хотят получить от продукта. Этот список содержит краткие описания всех желаемых возможностей продукта.

Product manager или product owner представляют бэклог команде и управляют им, описывает его главные элементы во время митинга по планированию спринта. Описание бэклога следует производить на простом и доступном языке, без технических спецификаций, чтобы оно было понятно каждому в команде. Любые изменения и требования по продукту должны быть своевременно отражены в этой очереди задач.

Бэклог спринта – это список определенных задач по воплощению в жизнь выбранных элементов бэклога продукта. Это список для оптимизации, которой команда займется в ближайший спринт, а также описание, каким образом они эту оптимизацию будут реализовывать.

Бэклог продукта составляет product owner, а за бэклог спринта отвечает команда разработчиков. Еще одним важным отличием является время создания бэклога: Product backlog создается на самом первом планировании спринта, а Sprint backlog должен создаваться командой на каждом планировании нового спринта. Таким образом, первый бэклог живет на протяжении всей разработки продукта, а Sprint backlog – на протяжении 1-4 недель, то есть, в течение одного спринта.

Основой Scrum является Sprint, в течении которого выполняется работа над продуктом. По окончанию Sprint должна быть получена новая рабочая

версия продукта. Sprint всегда ограничен по времени (1-4 недели) и имеет одинаковую продолжительность на протяжении всей жизни продукта [17].

Перед началом каждого Sprint производится Sprint Planning, на котором производится оценка содержимого Product Backlog и формирование Sprint Backlog, который содержит задачи (Story, Bugs, Tasks), которые должны быть выполнены в текущем спринте. Каждый спринт должен иметь цель, которая является мотивирующим фактором и достигается с помощью выполнения задач из Sprint Backlog.

По окончанию Sprint'a производятся Sprint Review и Sprint Retrospective, задача которых оценить эффективность (производительность) команды в прошедшем Sprint'e, спрогнозировать ожидаемую эффективность (производительность) в следующем спринте, выявлении имеющихся проблем, оценки вероятности завершения всех необходимых работ по продукту и другое.

1.4.3. Методология RAD

Идея RAD зародилась в 1980-х годах как альтернатива устаревающей методологии Waterfall. Каскадная модель программирования уже тогда воспринималась как перегруженная формальностями и недостаточно гибкая. Заказчик выдавал разработчику техническое задание и не видел результата до тех пор, пока программа не «сходила с конвейера» уже готовой, – и ожидания пользователя зачастую не оправдывались. Продукт мог оказаться слишком сложным, неудобным, а мог и устареть за время разработки.

В каскадной модели на ранних этапах работы проводится тщательное планирование, но это не помогает предусмотреть все риски и сложности. Поэтому проект дорожает, а время уходит впустую.

В 1988 году американский инженер-программист Барри Боэм (Barry Boehm) опубликовал статью «Спиральная модель разработки и

совершенствование программного обеспечения», в которой предложил создавать не цельную программу, а выпускать ряд прототипов, каждый из которых содержит дополнительную или расширенную функциональность по сравнению с предыдущим. Пользователь может изучить и попробовать в деле каждый прототип. Получая обратную связь, разработчик дорабатывает приложение, пока заказчик не получит готовый продукт, который полностью его устраивает.

Идея оказалась перспективной. Ее проработал специалист IBM Джеймс Мартин – в 1991 году вышла его книга «Быстрая разработка приложений» с изложением оригинальной методики применения RAD или Rapid Application Development. Спустя два года Джеймс Керр и Ричард Хантер написали книгу «Внутри RAD: как построить полностью функциональную систему за 90 дней или меньше», где проанализировали подводные камни и возможности, которые они выявили при планировании и реализации успешного проекта RAD. Эти книги заложили фундамент для практического применения RAD, и с тех пор эта методология остается в арсенале IT-разработчиков.

Методология разработки информационных систем, основанная на использовании средств быстрой разработки приложений, получила в последнее время широкое распространение и приобрела название методологии быстрой разработки приложений – RAD (Rapid Application Development). Данная методология охватывает все этапы жизненного цикла современных информационных систем. RAD – это комплекс специальных инструментальных средств быстрой разработки прикладных информационных систем, позволяющих оперировать с определенным набором графических объектов, функционально отображающих отдельные информационные компоненты приложений. Под методологией быстрой

разработки приложений обычно понимается процесс разработки информационных систем, основанный на трех основных элементах:

- небольшой команде программистов (обычно от 2 до 10 человек);
- тщательно проработанный производственный график работ, рассчитанный на сравнительно короткий срок разработки (от 2 до 6 мес.);
- спиралевидная модель разработки, основанная на тесном взаимодействии с заказчиком – по мере выполнения проекта разработчики уточняют и реализуют в продукте требования, выдвигаемые заказчиком.

При использовании методологии быстрой разработки приложений жизненный цикл информационной системы состоит из четырех фаз (рисунок 1.6).

1.4.3.1. Анализ и планирование

Здесь определяются цели и задачи проекта – что и для чего будет делать приложение. Совместными усилиями заказчик и разработчик выявляют риски, устанавливают сроки и бюджет, определяют ключевые моменты разработки.

1.4.3.2. Пользовательское проектирование

На этом этапе создается серия работающих прототипов программы. Каждый очередной прототип отличается от предыдущего дополненной функциональностью, изменениями дизайна и производительности. Процесс создания одного прототипа называется итерацией. RAD не накладывает жестких временных рамок на продолжительность одной итерации, но рекомендует, чтобы она была максимально быстрой.

В начале очередного цикла разработки, заказчик и программист вместе формулируют требования, которым должна соответствовать очередная версия. Преимущество RAD заключается в том, что не надо заранее продумывать каждую мелочь: сначала разрабатывается самая общая концепция, которая на следующих итерациях будет дополняться и уточняться.

1.4.3.3. Конструирование

Затем в работу включаются программисты. С помощью инструментов CASE они воплощают требования в виде модели, создавая очередной прототип. Его показывают пользователю и получают обратную связь. Уточняя пожелания и требования к программе, заказчик фактически руководит разработкой.

Прототип зачастую создается на скорую руку, только для проверки концепций. Это нормально: если пользователя устраивает новая функциональность и все работает, как должно работать, в следующей итерации разработчик «отполирует» интерфейс и код.

Как только заказчик дал обратную связь, цикл начинается заново. Вырабатывается план на следующую итерацию. Если пользователя что-то не устроило в прототипе, на новом витке цикла изменения откатывают назад и реализуют альтернативный вариант.

Если заказчик принял прототип – уточняются требования к функциональности, она прорабатывается более детально, планируется новая. Обговаривается визуальные элементы и интерфейсы.

От прототипа к прототипу программный продукт приобретает вид завершенного приложения. Итерации выполняются, пока не будут реализованы последние требования. Когда моделирование завершено,

начинается конструирование: автоматически сгенерированный код дорабатывается и совершенствуется.

1.4.3.4. Переключение

Готовый программный продукт тестируют, развертывают на пользовательских машинах, конвертируют информацию в новый формат или «заливают» в новые базы данных, подготавливают документацию и обучают операторов работе в системе.

Рисунок 1.6 – *Жизненный цикл по модели RAD*

1.5. Результаты и их обсуждения

В данной главе рассмотрены и проанализированы каскадная, итерационная и спиралевидная модели внедрения, представлены пошаговые этапы их реализации. В итоге следует заявить, что подходы значительно отличаются друг от друга и необходимо более детально проанализировать каждый из них в рамках внедрения информационной системы. Для этой цели выбраны соответствующие каждой модели методологии и определены основные понятия и процессы внутри них.

Сравнивая каскадную, итерационную и спиральную модели, можно сказать, что каскадная модель ограничена в своем применении из-за отсутствия обратной связи между этапами разработки, итерационная модель разработана для устранения этого и других недостатков каскадной модели, а спиральная модель является улучшенной итерационной моделью, позволяя совершенствовать программный продукт последовательно и неограниченно.

Основываясь на приведенных преимуществах и недостатках моделей внедрения была составлена таблица 1.3.

Таблица 1.3 – *Сравнение моделей внедрения ИС*

<table>
<tr><th rowspan="2">Характеристика проекта</th><th colspan="3">Модель (стратегия)</th></tr>
<tr><th>Каскадная</th><th>Итерационная</th><th>Спиральная</th></tr>
<tr><td rowspan="2">Новизна разработки и обеспеченность ресурсами</td><td colspan="2">Типовой. Хорошо проработаны технология и методы решения задачи</td><td rowspan="2">Нетиповой (новаторский). Нетрадиционный для разработчика</td></tr>
<tr><td>Ресурсов заказчика и разработчика хватает для реализации проекта в длительные сроки</td><td>Ресурсов заказчика или разработчика не хватает для реализации проекта в сжатые сроки</td></tr>
<tr><td>Масштаб проекта</td><td>Средние и крупные проекты</td><td>Малые и средние проекты</td><td>Любые проекты</td></tr>
<tr><td>Сроки выполнения проекта</td><td colspan="3">До нескольких лет. Разработка одной версии может занимать срок от нескольких недель до года</td></tr>
<tr><td>Заключение отдельных договоров на отдельные версии</td><td>Заключается один договор. Версия и есть итоговый результат проекта</td><td>Заключение отдельных договоров на отдельные версии</td><td>Заключается один договор. Версия и есть итоговый результат проекта</td></tr>
<tr><td>Определение основных требований в начале проекта</td><td>Да</td><td>Определение основных требований в начале проекта</td><td>Да</td></tr>
</table>

Характеристика проекта	Модель (стратегия)		
	Каскадная	Итерационная	Спиральная
Изменение требований по мере развития проекта	Нет	Изменение требований по мере развития проекта	Нет
Разработка итерациями (версиями)	Нет	Разработка итерациями (версиями)	Нет
Распространение промежуточного ПО	Нет	Распространение промежуточного ПО	Нет

Раздел 2. Идентификация и анализ требований, а также бизнес-процессов

Анализ требований – этап производства ПО, который включает в себя сбор требований, их анализ, обозначение взаимосвязей и документирование. Во время сбора требований необходимо учитывать возможные несостыковки в требованиях, разных вовлеченных в создание системы сторон.

Доскональность и добротность анализа занимают важнейшее место в достижении поставленных целей. Требования к ПО необходимо документально подтвердить, выполнить и проверить с уточнением, необходимым для проектирования всей системы. Анализ требований включает в себя:

- Первым и самым важным этапом в разработке продукта является сбор бизнес-требований. Цель этой работы – определить основные требования бизнеса (исходные данные, истинные цели, которым должен служить продукт и проблемы, которые нужно решить). Для продуктов под заказ и продуктов для открытого рынка процесс сбора бизнес-требований существенно различается. К способам сбора требований можно отнести:
  - Интервьюирование – заключается в беседе представителя разработчика и заинтересованного лица. Применяется в случае, когда большим объемом знаний обладает ограниченных круг людей. Обычно используется для беседы с одним человеком с глазу на глаз. Метод используется в том случае, когда нереально собрать насколько ключевых пользователей одновременно в одном месте. Опасность применения метода заключается в том,

что могут быть опрошены не все заинтересованные лица, и разработчики могут упустить ключевую информацию.

- Анкетирование – заключается в составлении списка вопросов и предоставлении их заинтересованным лицам. Метод удобен в случае, когда имеется большая база респондентов, а разработчики могут квалифицированно сформулировать список вопросов. Опасность метода заключается в том, что вопросы могут быть неоднозначно поняты пользователями и соответственно даны неточные ответы. Возможно, что пользователи вообще не смогут ответить на некоторые вопросы, которые потребуют дополнительного пояснения или просто забудут ответить на анкету.
- Мозговой штурм – заключается в обсуждении требований и записи всего, что придет в голову с последующей сортировкой и обработкой полученных требований, и установкой приоритетов. Метод удобен, когда обсуждаются нестандартные решения незнакомой проблемы. Большинство хороших идей часто бывают результатом комбинации нескольких, не связанных на первый взгляд. В самом процессе генерации идей не нужно критиковать и устанавливать ограничения. Главная задача мозгового штурма – выработка как можно большего количества требований, анализ требований нужно производить позднее.
- Сценарии и ролевые игры – метод заключается в создании легко модифицируемого схематичного сценария работы системы (не прототипа), который обсуждается с пользователем. Метод направлен на поиск актеров, участвующих в работе системы и получения от будущих пользователей обратной связи, например,

по вопросам интерфейса или по вопросам «что будет если...». Для ролевых игр могут использоваться доски, простые листы бумаги или даже интерактивные сценарии. Главное, чтобы пользователь понимал, как происходит взаимодействие с системой. Сценарии особенно полезны в случае, когда в систему включены новые функции, плохо поняты требования, не определены актеры и варианты использования. Они также позволяют обсудить пользовательский интерфейс, не прибегая к трудоемкому созданию прототипов.

- Создание прототипов – метод заключается в создании скелета системы – прототипа, который позволяет предметно обсуждать с пользователем требования к системе, демонстрировать принятые решения, внешний вид системы и порядок работы. Прототип – не является рабочей версией программы, поскольку в нем не реализуются многие возможности необходимые для нормальной работы. Такие возможности как проверка входных параметров, обработка ошибок, резервное копирование обычно в прототип не включаются. Также при создании прототипа не уделяется внимание вопросам производительности и масштабируемости. Прототипы могут служить доказательством того, что команда проекта работает над системой, однако при их использовании может возникнуть опасность, что прототипы будут приняты руководством за рабочий вариант со всеми вытекающими из этого последствиями. В частности, до требований, которые не реализованы в прототипах, могут вообще не дойти руки.
- Совместная разработка приложений (joint application design [JAD-совещание]) – метод заключается в том, чтобы собрать всех

заинтересованных лиц в одной комнате на день-два для интенсивной работы по идентификации требований, их документированием и назначением приоритетов. Метод позволяет значительно сократить время опроса пользователей по отдельности, но требует высокой квалификации того, кто ведет JAD-совещание. Ведущий должен быть политически нейтральным, достаточно хорошо знакомым с областью деятельности, на которую рассчитан проект. Успех во многом зависит от позиции заинтересованных сторон и тех, кто принимает решения и их готовности к сотрудничеству. Этот метод хорошо сочетается с моделированием и, с некоторыми ограничениями, – созданием прототипов.

- Моделирование – метод заключается в создании и обсуждении моделей автоматизируемых процессов, информационных потоков, отношений между объектами и т.д. Модели могут готовиться при помощи различных программных средств, таких как BPwin, Rational Rose, Visio и различных нотациях, которые понятны заинтересованным лицам. Нужно определять в каких случаях дешевле и проще создать модели, а в каких прототипы, поскольку прототип можно назвать моделью предварительного варианта использования. Обычно моделирование дешевле и гибче прототипирования, поскольку модель может иметь различные уровни абстракции от высокоуровневых до подробного и весьма детализированного описания системы. К недостаткам этого метода можно отнести высокие требования к квалификации как разработчиков, которые создают модели и обсуждают их с заинтересованными лицами, так и тем, кто эти

модели рассматривает. Нельзя забывать и то, что слишком детализированное моделирование системы может занимать столько же времени, сколько заняло бы создание прототипа, и в то же время прототип можно «пощупать» и использовать его части в дальнейшем создании системы, тогда как модель – только иллюстрация.

- Изучение существующей документации. Данная методика может быть использована при наличии в организации документации, которая может помочь в определении потребностей Заказчика. Примеры документации включают в себя: регламенты, описания процессов, структура организации, спецификации продукта, различные процедуры, стандарты и инструкции, шаблоны документов и т.д. Выявленные требования являются основой для дальнейшего анализа и должны быть детализированы. Данная методика применима, например, при автоматизации устоявшихся в организации регламентированных бизнес процессов.

- Анализ – определение того, являются ли собранные требования неясными, неполными, неоднозначными или противоречивыми; решение этих проблем; определение их соотношений.
- Документирование – требования могут быть документально подтверждены в различных формах, таких как простое описание, сценарии использования, истории пользователей или спецификации процесса [18].

2.1. Список требований

Пользовательские требования – требования, формулируемые пользователями к конечному продукту. На основе методов интервьюирования (проведение беседы с персоналом городской больницы) и изучения существующей документации (работа с бумажными формами документов и законодательством в области здравоохранения) были выявлены требования, представленные в таблице 2.1.

Таблица 2.1 – *Перечень пользовательских и законодательных требований*

№	Пользовательское требование
1	Возможность вывода данных о пациенте
2	Возможность просмотра информации о персонале: ▪ Фамилия, Имя, Отчество специалиста;- занимаемая должность; ▪ порядок записи к специалисту; ▪ график приема специалистом; ▪ данные сертификата специалиста – специальность, срок действия сертификата, соответствие занимающей должности; ▪ дополнительные данные сертификата – кем выдан, когда выдан, уровень образования, квалификация
3	Возможность для заведения медицинской карты
4	Возможность для удаления медицинской карты
5	Возможность просмотра и редактирования данных из БД «Пациенты»
6	Возможность просмотра и редактирования данных из БД «Эпикриз»
7	Возможность просмотра и редактирования данных из БД «Анамнез»
8	Возможность просмотра и редактирования данных из БД «Температура»
9	Возможность просмотра и редактирования данных из БД «Назначения»
10	Возможность просмотра и редактирования данных из БД «Дневник»
11	Возможность просмотра и редактирования данных из БД «Персонал»
12	Возможность печати карты при выписке

№	Пользовательское требование
13	Возможность поиска медицинской карты по ее номеру или фамилии пациента
14	Возможность работы на любом устройстве
15	Возможность хранения данных
16	Доступность
17	Легкость в использовании
18	Наличие базы данных для температурных листов пациентов
19	Наличие базы данных для учета лечебных назначений
20	Наличие базы данных для хранения врачебных заключений
21	Наличие базы данных для хранения данных о пациенте после его поступления
22	Наличие базы данных для хранения информации о медицинском персонале
23	Наличие базы данных для хранения медицинских карт пациентов
24	Наличие базы данных для эпикриза пациентов
25	Наличие данных об анамнезе пациента
26	Невозможность подмены пациентом данных о себе
27	Работа системы в любом месте
28	Удобный интерфейс на мобильных устройствах
29	Доступность данных
30	Конфиденциальность персональных данных
31	Осуществление медицинской деятельности и медицинских услуг
32	Дата гос. регистрации учреждения
33	Полные сведения об учредителях
34	Структура организации, как осуществляется управление
35	Контактные данные, а именно – почтовый адрес, телефон, электронный адрес, схема проезда
36	Полное наименование предприятия

№	Пользовательское требование
37	Информация о подразделениях, филиалах, корпусах
38	Режим работы учреждения
39	Правила внутреннего распорядка
40	Лицензии организации на ведение медицинской деятельности, представленные в электронном виде
41	Виды оказываемой медицинской помощи
42	Правила записи на прием, обследование, консультацию, диагностику
43	Правила подготовки к проведению диагностических манипуляций
44	Правила диспансеризации и госпитализации
45	Стоимость оказываемых медицинских услуг, с приложением утвержденного документа с ценами в электронном виде
46	Графики приема врачей, контактные данные специалистов или организации – электронная почта, телефон
47	Информация об органах охраны здоровья, надзору в сфере здравоохранения, надзору защиты прав потребителей (почтовый адрес, телефоны, электронный адрес)
48	Информация об страховых учреждениях, с которыми заключены договора на оказание и оплату мед. услуг по ОМС
49	Обязательно должна быть размещена информация о возможности проведения независимой оценки качества оказываемых услуг
50	Карта сайта для работы с ресурсом и удобной навигации
51	Версия сайта для слабовидящих людей
52	Иные инструменты, обеспечивающие удобную работу с ресурсом пользователя
53	Язык сайта (меню, карта сайта, информация на сайте) обязательно должен быть русским

2.2. Ключевой бизнес-процесс в модели «AS-IS»

Модель «AS-IS» («как есть») делает возможным отображение настоящего положения дел, классифицирование происходящих в учреждении процессов и потоков информации в их пределах. С помощью этой модели определяются проблемные места при реализации и взаимодействии процессов, выявляется потребность в записи различных преобразований. В

пределах моделирования бизнес-процессов производится анализ их нынешнего состояния и варианты улучшения. Для определения низкой эффективности в функционирующих процессах производится разложение до простейших операций. Благодаря этому возможно определение частей процессов, препятствующих плодотворному осуществлению:

- излишние или повторяющиеся функции;
- неупорядоченные или слабо упорядоченные действия, выполнение которых варьируется у сотрудников;
- несоответствие документооборота реализуемому бизнес-процессу;
- неимение обратных связей по менеджменту;
- неимение обратной связи по входным данным и т.д.

Конкретизация процессов по модели «AS-IS» дает возможность обнаружить недочеты в проверяемой сфере деятельности, учитываемые при разработке модели «TO-BE» («как должно быть») – модели улучшенной организации процессов [19].

Обнаруженные в модели «AS-IS» недочеты устраняются с помощью модели «TO-BE». Производство и ввод в эксплуатацию ИС позволяет изменить условия осуществления единичных операций, устройство процессов и компании в целом. Это служит источником потребности в преобразование системы норм на предприятии, изменения служебных правил работников.

Функциональная модель «TO-BE» дает возможность на ранних этапах выявить преобразования в системе. Ее применение делает возможным не только ускорение сроков внедрения ИС, но и снижение рисков, связанных с невосприимчивостью или отсутствием опыта сотрудников в работе с новыми технологиями. Она необходима для исследования иных способов реализации и создания документации будущей деятельности фирмы.

Обычная методика планирования ИС предполагает изначальное формирование модели «AS-IS», после анализа которой выявляются направления совершенствования процессов. Если же в качестве основы автоматизации фирмы первоначально берется данная модель, то вместо информатизации фирмы с помощью новых технологий случится обычная компьютеризация неидеальных процессов. В итоге, ввод и работа такой ИС обусловит излишние затраты на покупку оборудования, создание ПО и их поддержку.

2.3. Описание ключевого бизнес-процесса

На рисунке 2.1 представлен верхний (0) уровень описания ключевых бизнес-процессов городской больницы.

Рисунок 2.1 – *Верхний уровень описания ключевых бизнес-процессов*

При рассмотрении первого уровня описания бизнес-процессов (рисунок 2.2) возможно рассмотреть документацию и ответственных за 3 ключевых бизнес-процессах: «Принять пациента», «Лечить пациента» и «Выписать пациента» [20]. Для более точного понимания текущей ситуации в городской больнице необходимо углубиться в рассмотрение бизнес-процесса «Лечить пациента» на 2 (рисунки 2.3 и 2.4) и 3 уровень описания процессов (рисунки 2.5 и 2.6).

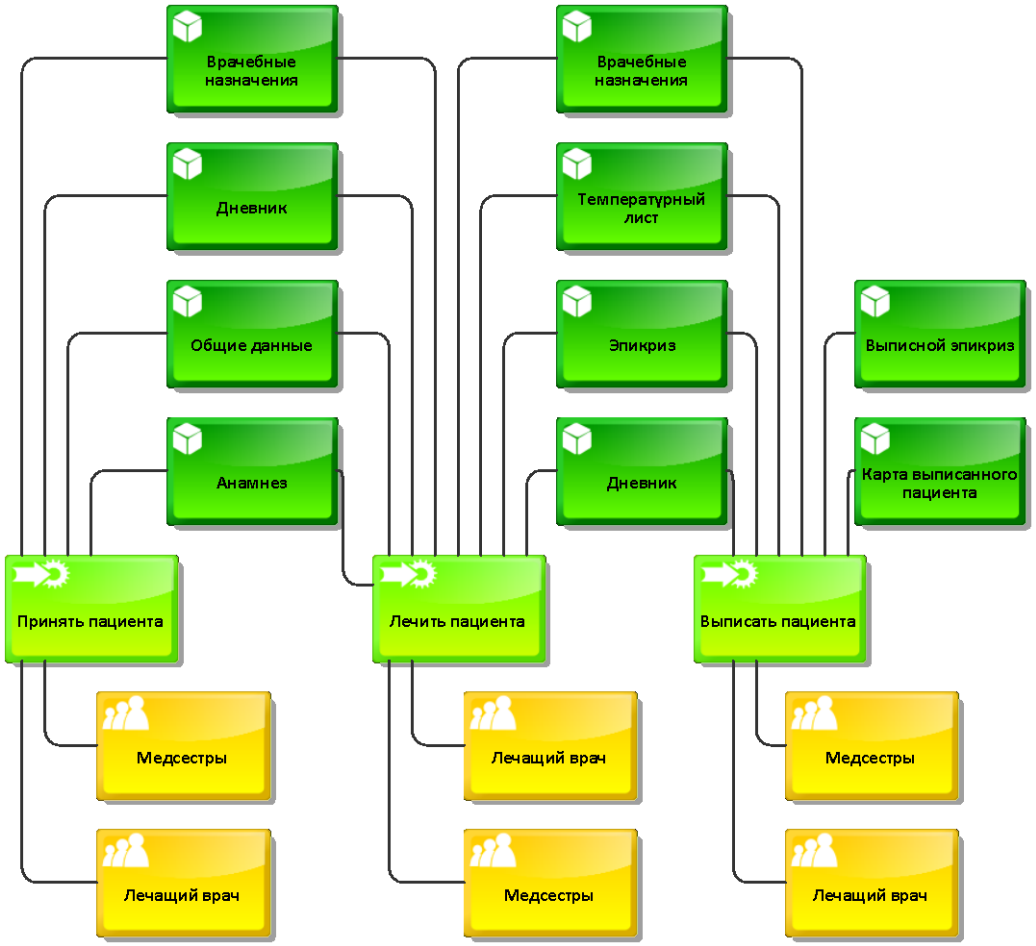


Рисунок 2.2 – *Первый уровень описания процессов ARIS VACD в модели «AS-IS»*

В отличие от IDFE0, которая, несмотря на возможность в полной мере описать всю деятельность организации, обычно используется для описания верхних уровней, нотация IDFE3 чаще применяется для процессов нижнего уровня [14]. Графические символы, используемые в нотации, представлены в таблице 2.3.

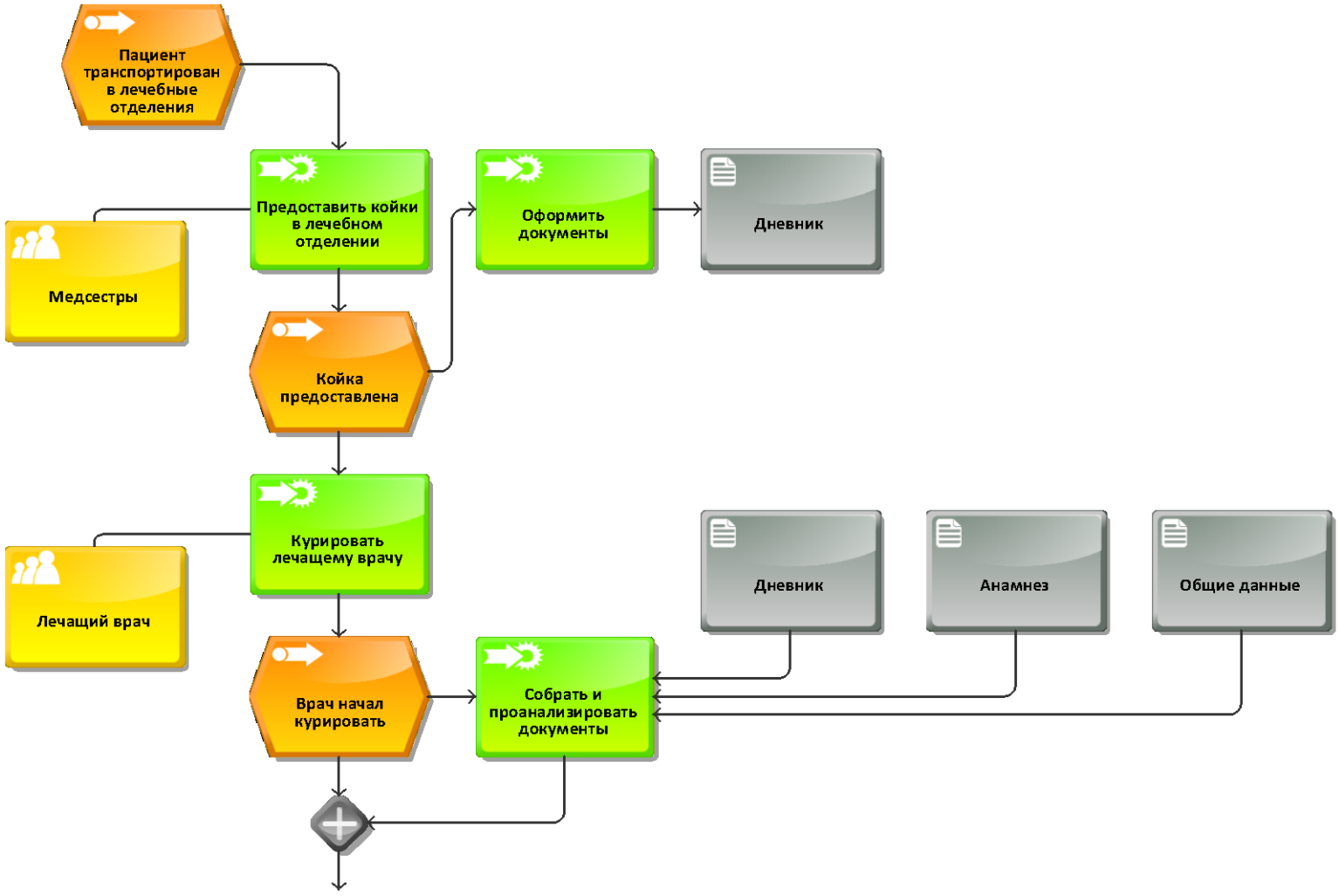


Рисунок 2.3 – *Процесс «Лечить пациента» в модели ARIS eEPC «AS-IS» ч. 1 (2 уровень)*

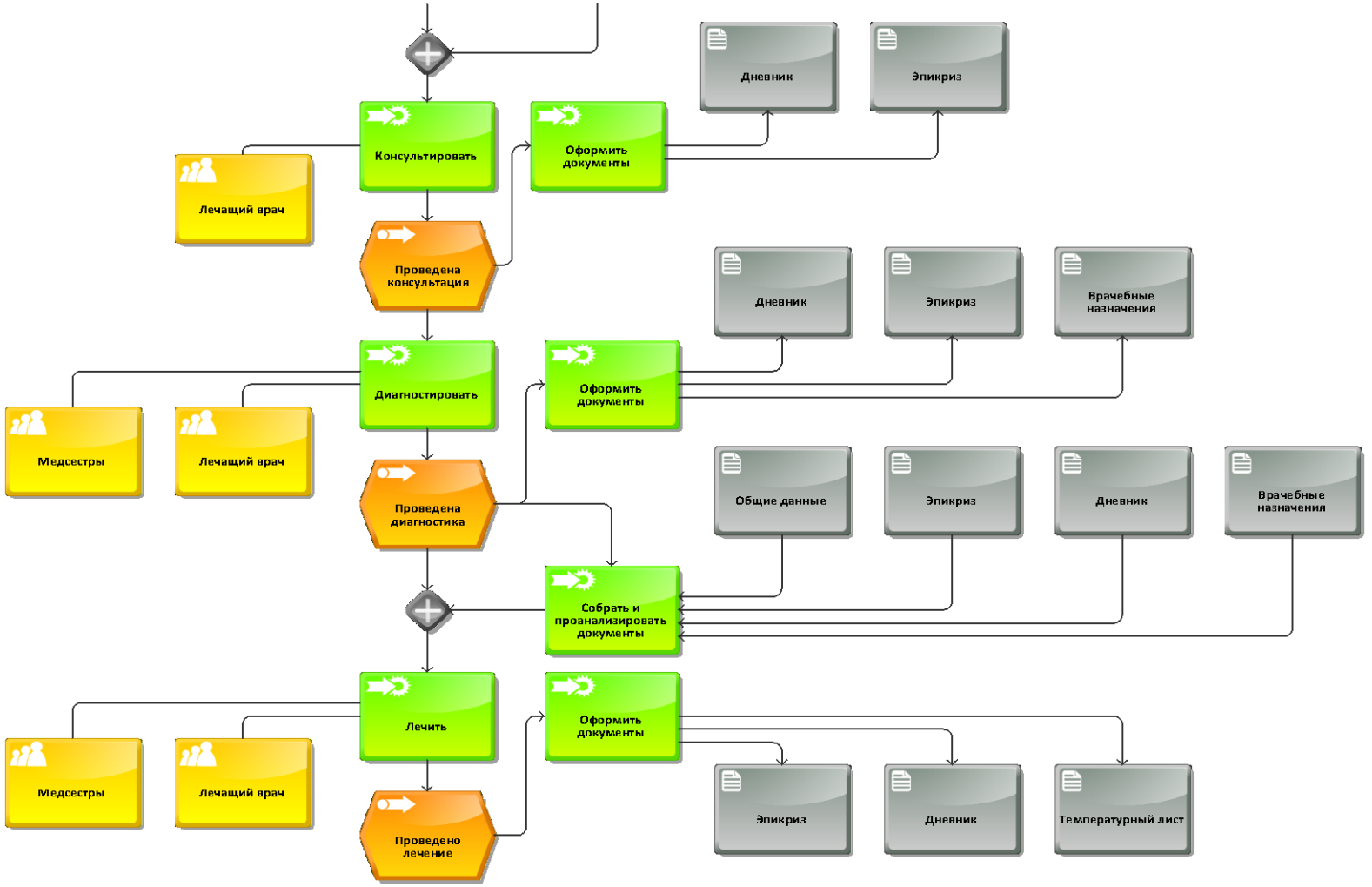


Рисунок 2.4 – *Процесс «Лечить пациента» в модели ARIS eEPC «AS-IS» ч. 2 (2 уровень)*

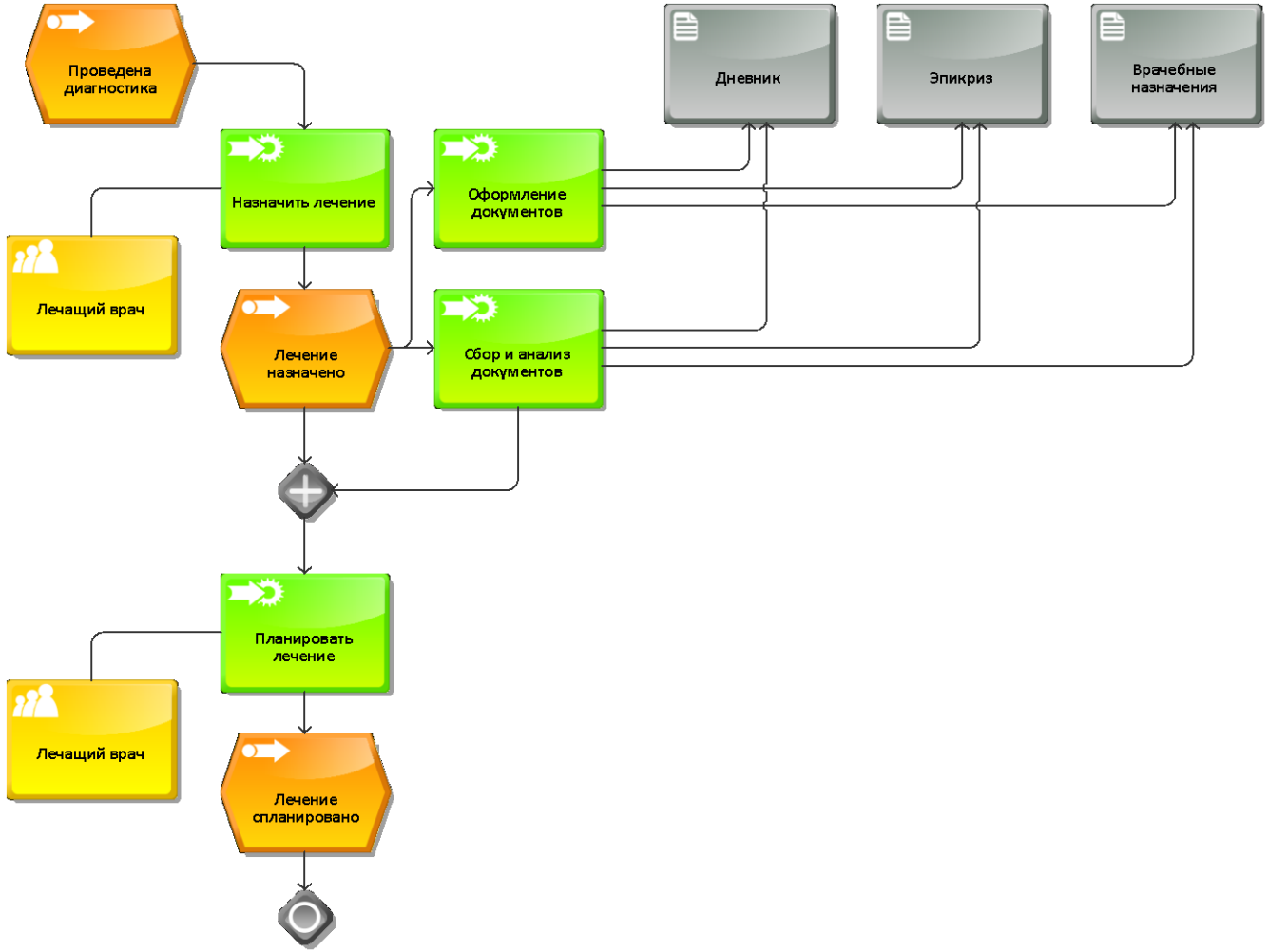


Рисунок 2.5 – *Процесс «Лечить пациента» в модели ARIS eEPC «AS-IS» ч. 1 (3 уровень)*

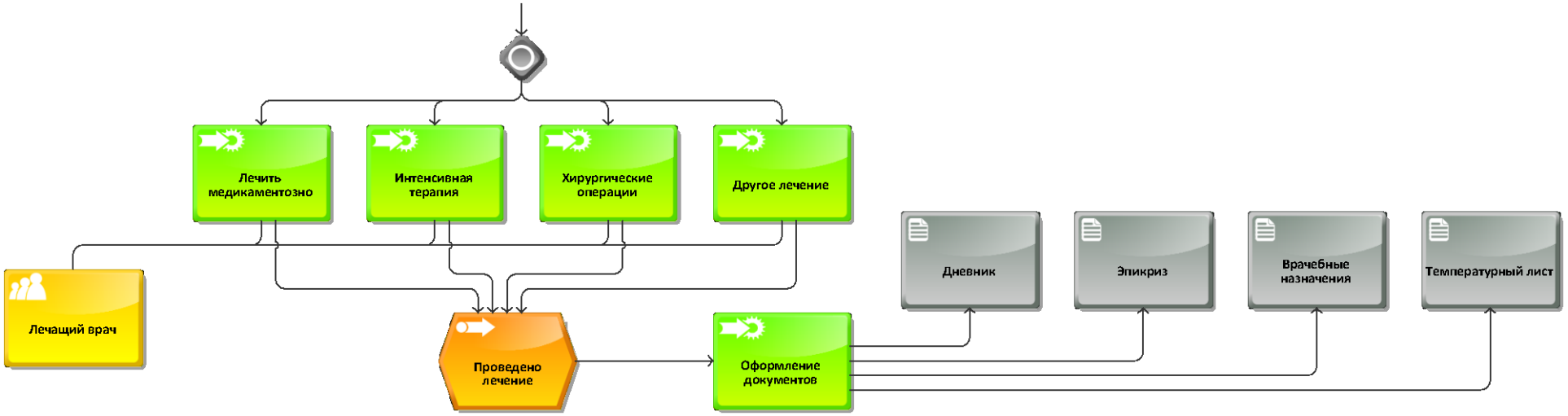


Рисунок 2.6 – *Процесс «Лечить пациента» в модели ARIS eEPC «AS-IS» ч. 2 (3 уровень)*

После рассмотрения бизнес-процесса до 3 уровня описания можно составить карту процесса, приведенную на рисунке 2.7.

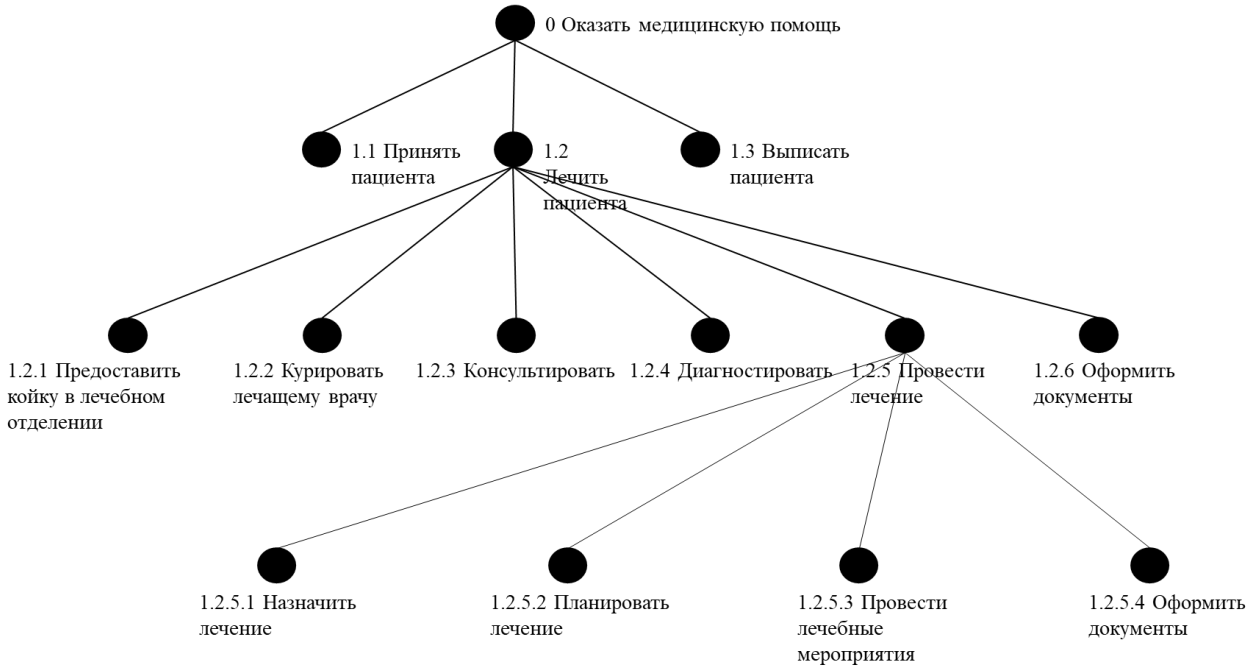


Рисунок 2.7 – *Карта бизнес-процессов в модели «AS-IS»*

2.4. Результаты и их обсуждения

Данная глава посвящена первичным работам при формировании концептуального проекта. На основе нескольких из ранее определенных методов сформирован перечень пользовательских и законодательных требований к реализуемой системе.

На основе полученных требований далее будет сформирована матрица отслеживания требований, после чего будет продолжено проектирование под реализацию при помощи методологии ASAP.

Затем рассмотрено описание ключевых бизнес-процессов в моделях «AS-IS» и «TO-BE» на верхних уровнях (нулевой и первый) при помощи методологии ARIS VACD, затем для процесса «Лечить пациента» произведено углубление на второй и третий уровни описания при помощи ARIS eEPC для более точного описания текущей ситуации в городской

больнице. Полученные данные помогут построить архитектуру данных, необходимую в реализации БД.

На основе разобранного до третьего уровня описания бизнес-процесса «Лечить пациента», составлена карта процессов в виде иерархического дерева, наглядно показывающая структуру данного процесса до 3 уровня.

Раздел 3. Реализация ключевого бизнес-процесса «Лечить пациента» на основе каскадной модели внедрения, используя метод ASAP

3.1. Матрица отслеживания требований

После выбора методологии внедрения и описания ключевого бизнес-процесса для реализации продолжается этап составления концептуального проекта. Реализуемые требования занесены в матрицу отслеживания требований разрабатываемого интерфейса для автоматизации ключевых бизнес-процессов городской больницы и представлены в таблице 3.1.

Таблица 3.1 – *Матрица отслеживания требований*

№	Пользовательское требование	Компонент покрытия
1	Доступность данных	Веб-сайт – Главная страница
2	Возможность для заведения медицинской карты	Форма регистрации пациента
3	Возможность хранения данных	База данных MySQL
4	Наличие таблицы для температурных листов пациентов	Таблица «Температурный лист» в БД
5	Наличие таблицы для учета лечебных назначений	Таблица «Лечебные назначения» в БД
6	Наличие базы данных для хранения врачебных заключений	Таблица «Врачебные заключения» в БД
7	Наличие базы данных для эпикриза пациентов	Таблица «Эпикриз» в БД
8	Осуществление медицинской деятельности и медицинских услуг	Веб-сайт – Страница «О нас»
9	Дата гос. регистрации учреждения	Веб-сайт – Страница «О нас»
10	Полные сведения об учредителях	Веб-сайт – Страница «О нас»
11	Структура организации, как осуществляется управление	Веб-сайт – Страница «О нас»
12	Контактные данные, а именно – почтовый адрес, телефон, электронный адрес, схема проезда	Веб-сайт – Страница «Контакты»
13	Полное наименование предприятия	Веб-сайт – Страница «Сведения»
14	Информация о подразделениях, филиалах, корпусах	Веб-сайт – Страница «Контакты»
15	Режим работы учреждения	Веб-сайт – Страница «Сведения»

№	Пользовательское требование	Компонент покрытия
16	Правила внутреннего распорядка	Веб-сайт – Страница «Сведения»
17	Лицензии организации на ведение медицинской деятельности, представленные в электронном виде	Веб-сайт – Страница «Сведения»
18	Виды оказываемой медицинской помощи	Веб-сайт – Страница «Сведения»
19	Правила записи на прием, обследование, консультацию, диагностику	Веб-сайт – Страница «Сведения»

3.2. Проектирование процессов, данных и пользовательских интерфейсов

На основе порядка использования данных, приведённого выше, всю информацию можно разбить на классы, приведённые в таблице 3.2.

Таблица 3.2 – *Классы данных*

Класс данных	Поле	Тип данных	Количество символов
Эпикриз	Номер записи	Счетчик	10
	Номер карты	Числовой	10
	Код сотрудника	Числовой	10
	Эпикриз	Текст	1000
Врачебные заключения	Номер записи	Счетчик	10
	Номер карты	Числовой	10
	Код сотрудника	Числовой	10
	Заключение	Текст	1000
Лечебные назначения	Номер записи	Счетчик	10
	Номер карты	Числовой	10
	Код сотрудника	Числовой	10
	Назначение	Текст	1000
Температурный лист	Номер записи	Счетчик	10
	Номер карты	Числовой	10
	Код сотрудника	Числовой	10
	Дата	Дата	8
	Время	Время	6
	Дыхание	Числовой	10
	Температура	Числовой	10

После разделения всех данных на классы получается архитектура разрабатываемой системы, приведённая на рисунке 3.1.

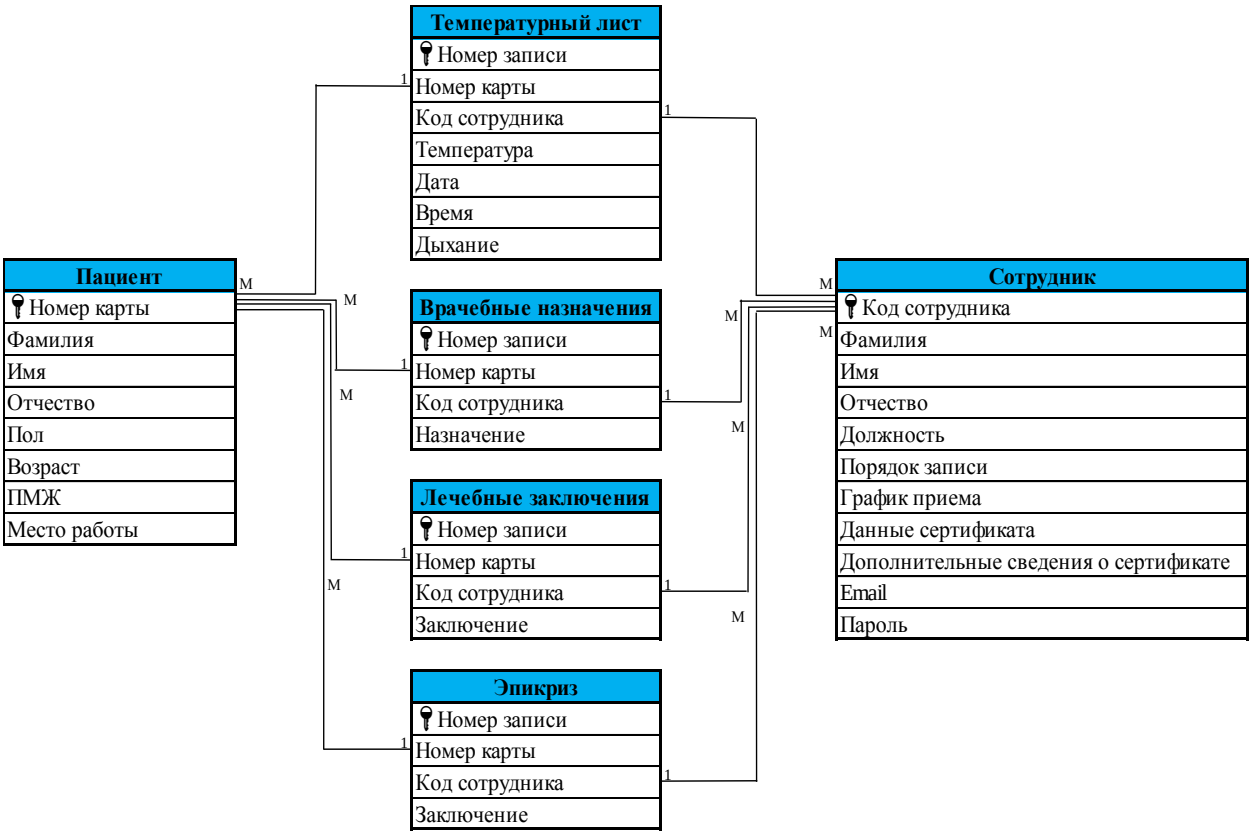


Рисунок 3.1 – *Архитектура данных разрабатываемой системы*

Рисунок 3.2 – *Схема главной страницы разрабатываемого сайта*

Для удобства использования системы, как пациентами, так и медицинским персоналом, необходимо разработать доступный для понимания новыми пользователями интерфейс. Для создания схем воспользуемся программой MS Visio. На рисунке 3.2 приведена схема главной страницы разрабатываемого сайта городской больницы.

Остальные страницы сайта должны иметь структуру, подобную приведенной выше, за исключением блока новостей, который будет заменен на блок вывода информации, принадлежащей к конкретной ссылке.

Также при помощи MS Visio составлен проект формы регистрации карты нового пациента (рисунок 3.3), а для формирования общего понимания о проектируемой системе составлена схема взаимодействия пользовательских интерфейсов (рисунок 3.4). При этом на ней обозначены элементы (1 и 2), которые будут добавлены на более поздних этапах разработки, а также не включены таблицы из БД по причине временного отсутствия связи их с сайтом.

Рисунок 3.3 – *Проект формы регистрации нового пациента*

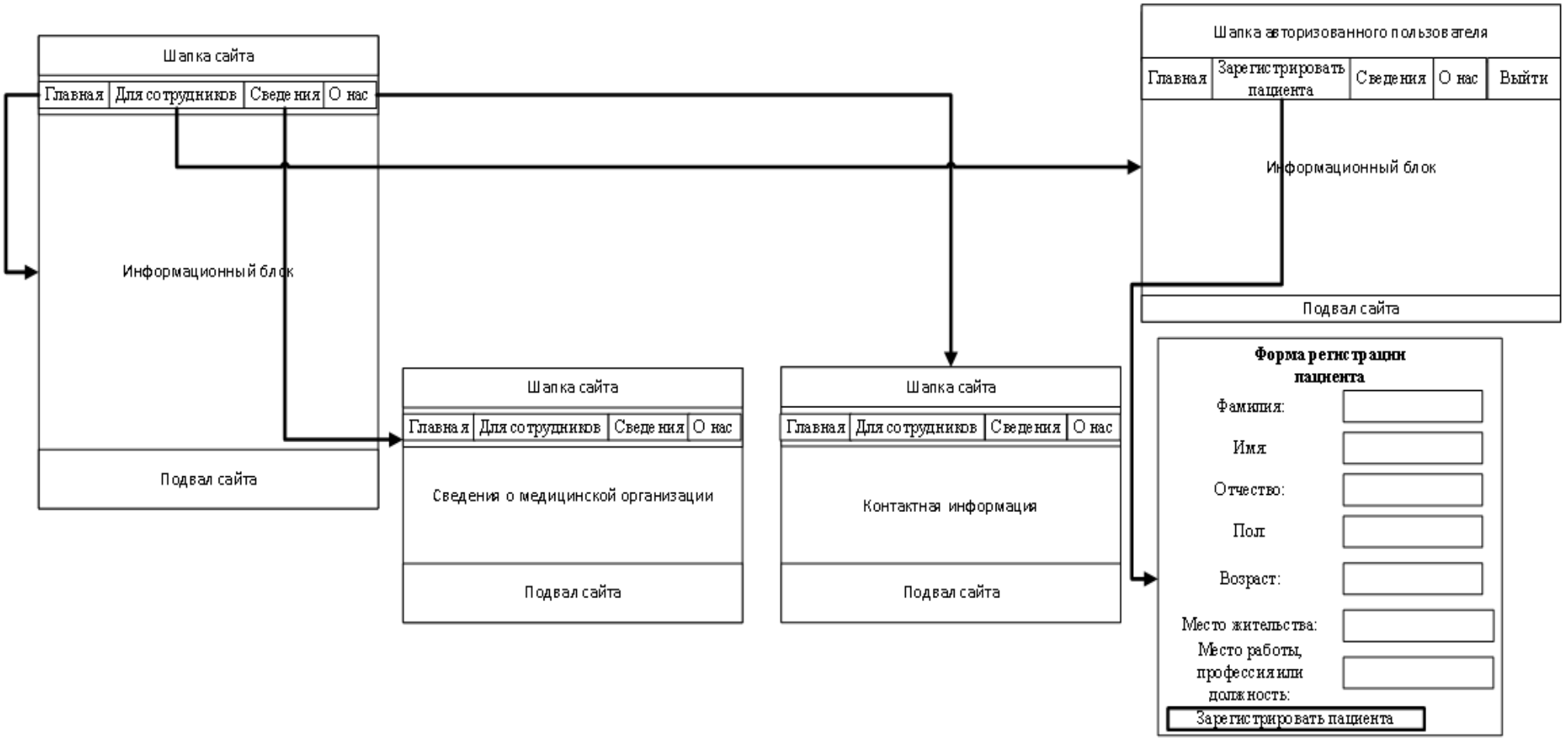


Рисунок 3.4 – *Схема взаимодействия пользовательских экранов*

Для описания того, каким должен быть ключевой бизнес-процесс «Лечить пациента» на 2 и 3 уровнях описания воспользуемся ранее описанной моделью «TO-BE» в ARIS eEPC (рисунки 3.5 – 3.7).

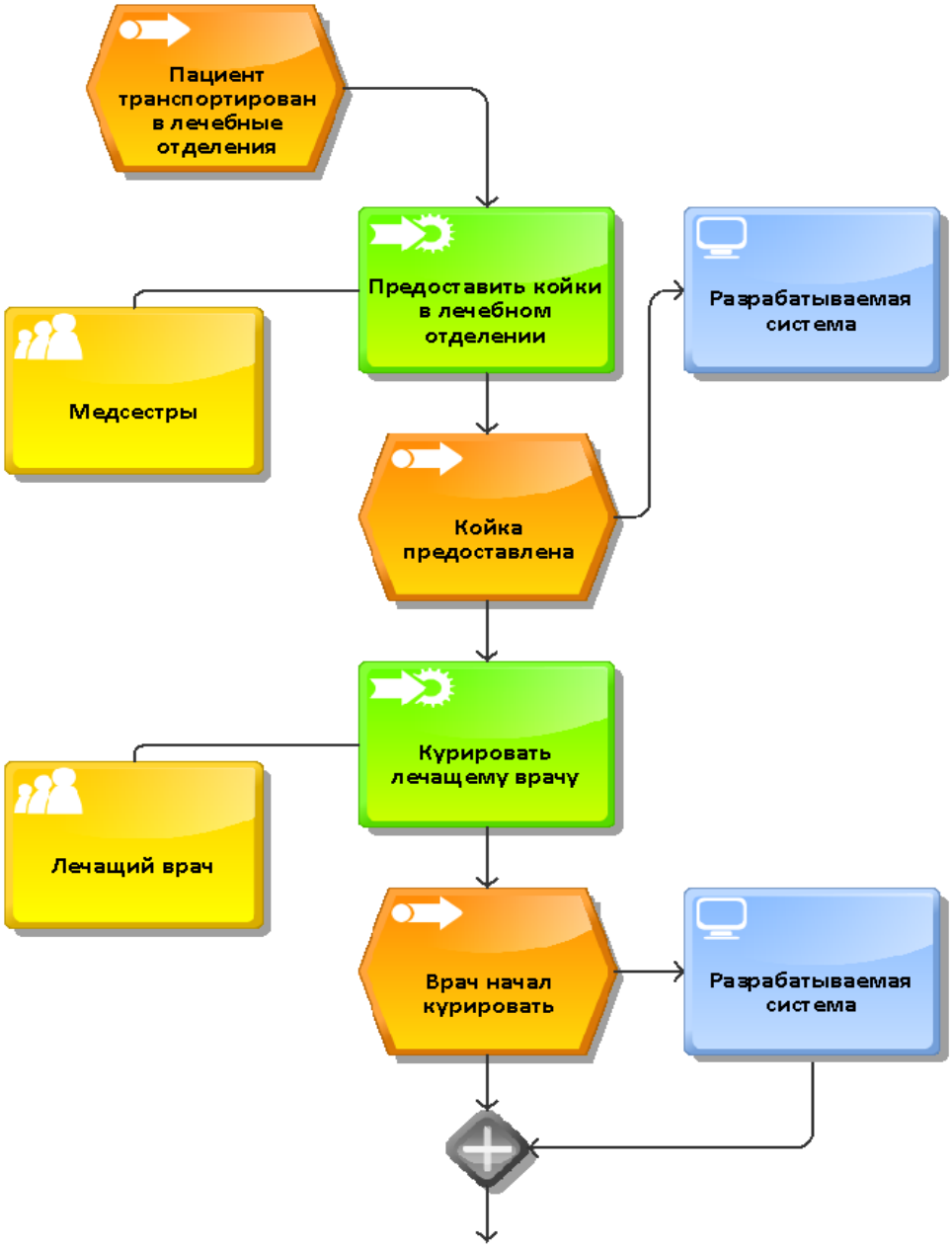


Рисунок 3.5 – *Процесс «Лечить пациента» в модели ARIS eEPC «TO-BE» ч.1 (2 уровень)*

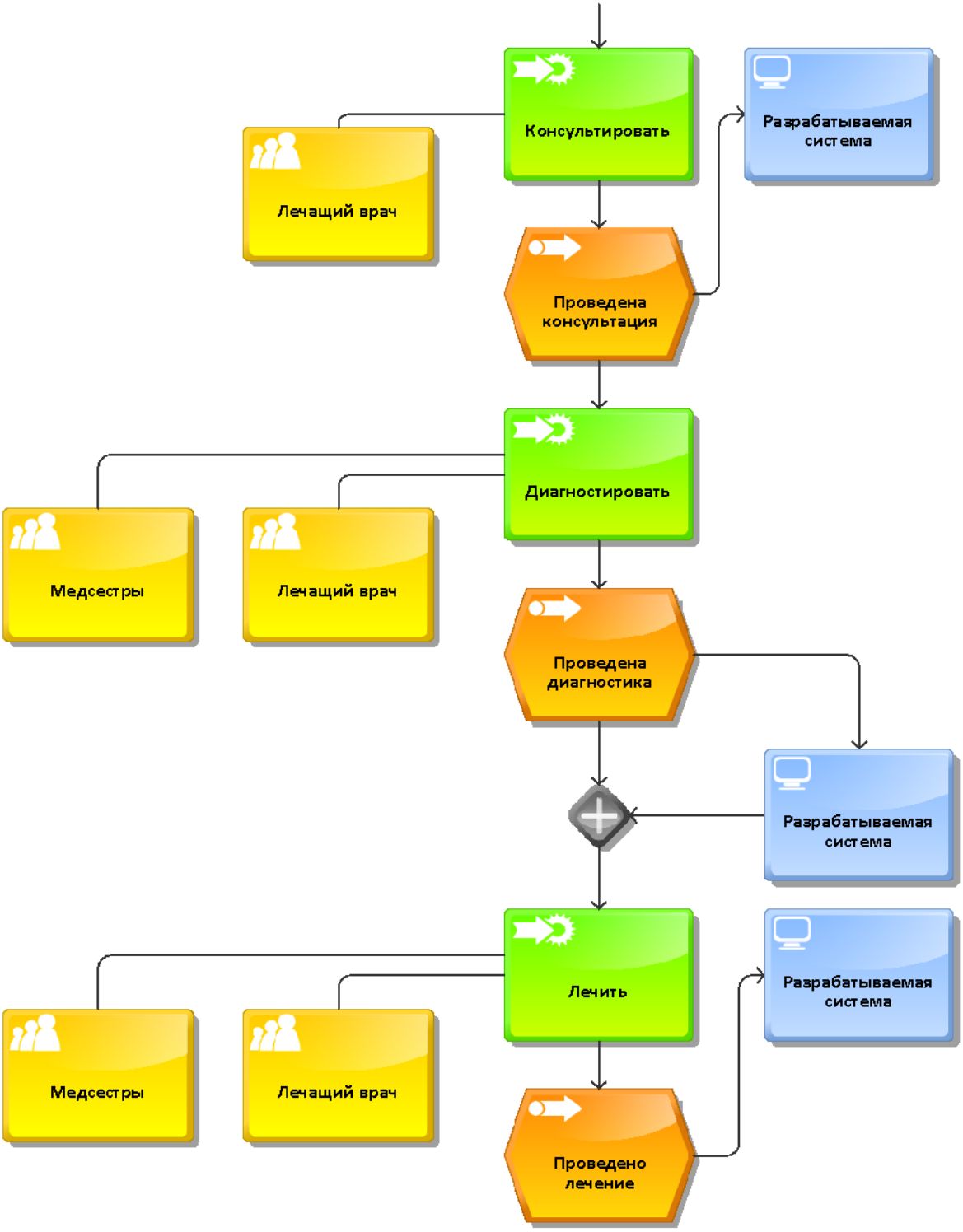


Рисунок 3.6 – *Процесс «Лечить пациента» в модели ARIS eEPC «TO-BE» ч.2 (2 уровень)*

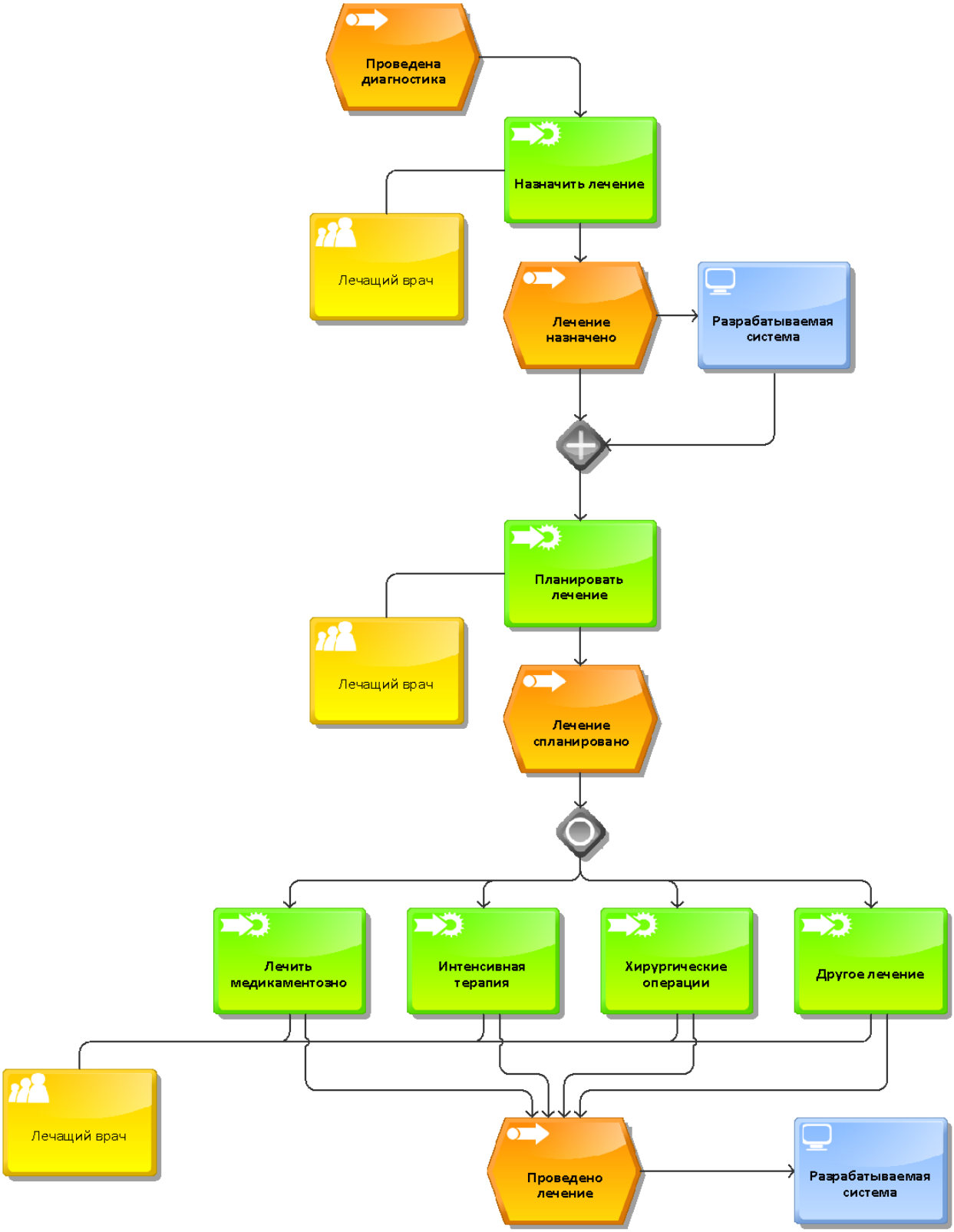


Рисунок 3.7 – *Процесс «Лечить пациента» в модели ARIS eEPC «TO-BE» (3 уровень)*

В результате, после описания ключевого бизнес-процесса «Лечить пациента» в модели «ТО-ВЕ» можно составить обновленную карту бизнес-процессов (рисунок 3.8).

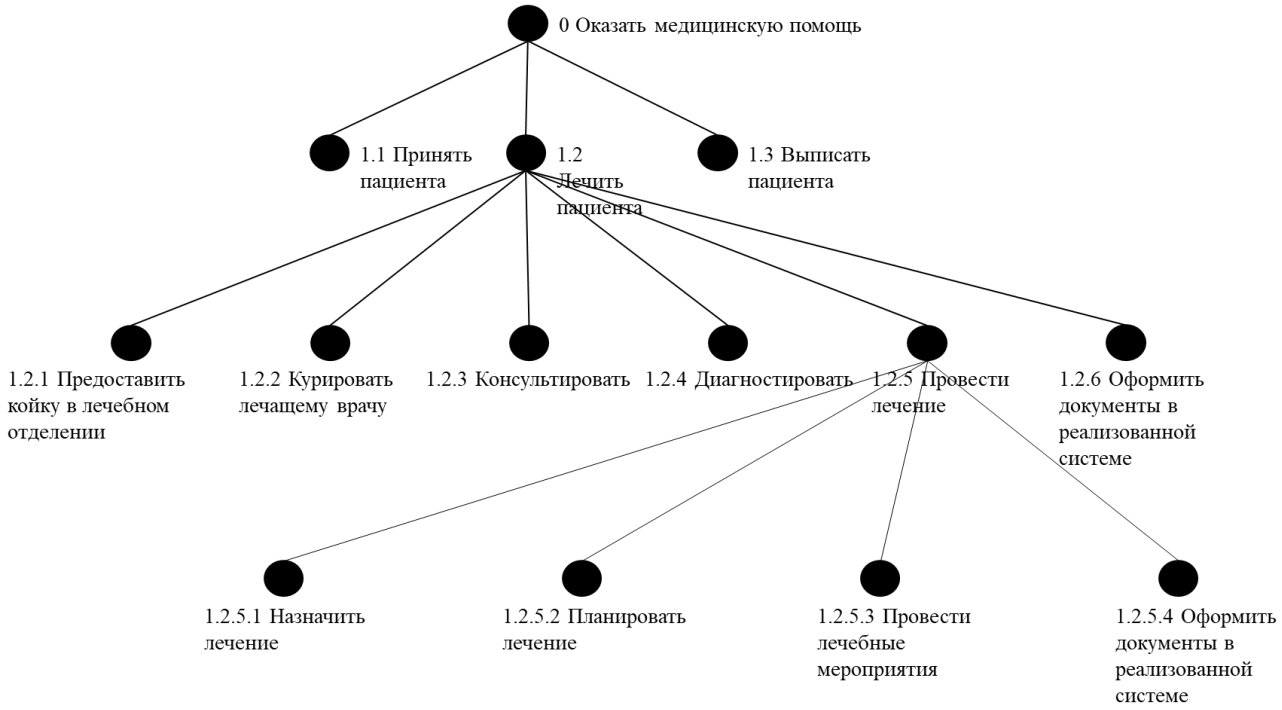


Рисунок 3.8 – *Карта процессов в модели «ТО-ВЕ»*

3.3. Реализация ключевого бизнес-процесса средствами PHP

При помощи Denwer (дистрибутив для локальной разработки web-сайтов) была создана БД в Phpmyadmin для хранения данных о пациентах и персонале городской больницы (рисунок 3.9).

Для удобства использования системы, как пациентами так и медицинским персоналом, необходимо разработать доступный для понимания новыми пользователями интерфейс. На рисунке 3.10 приведен фрагмент шапки разрабатываемого сайта городской больницы.

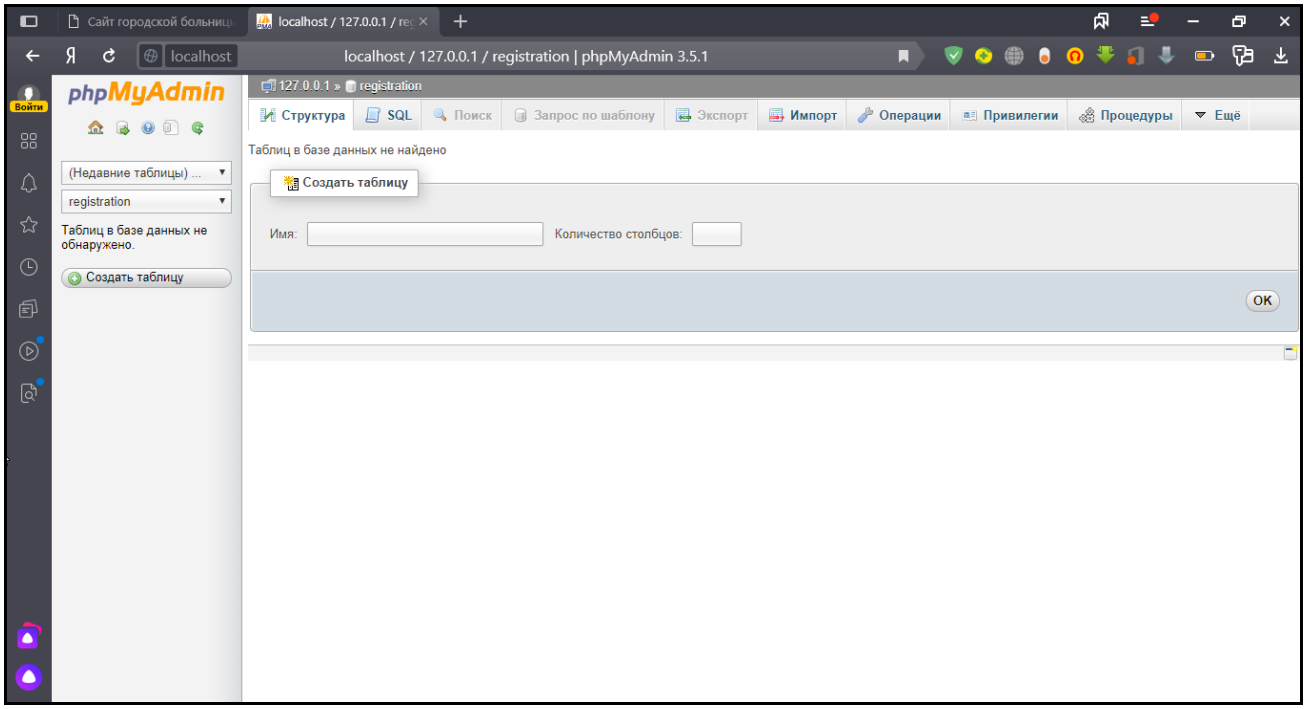


Рисунок 3.9 – *База данных для больницы*

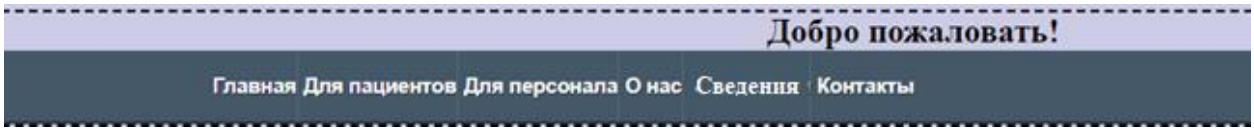


Рисунок 3.10 – *Фрагмент шапки разрабатываемого сайта*

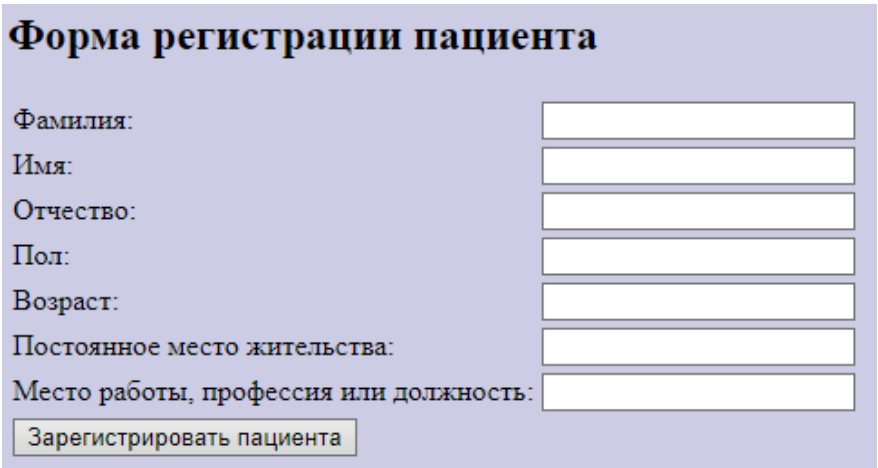


Рисунок 3.11 – *Форма регистрации пациента*

Данные о пациентах хранятся в соответствующих таблицах (рисунок 3.12).

id	stuff_id	date	time	dyx	temp
4	1	2018-04-11	14:00:05	60	37

Рисунок 3.12 – *Данные в таблице «Температурный лист»*

Листинг кода разработанных веб-страниц приведен в приложении А.

3.4. Тестирование разработанной программы

В рамках данного этапа будет проводиться 2 вида тестирования: функциональное, результаты которого будут занесены в чек-лист представленный в таблице 3.3, и нагрузочное, результаты которого будут представлены снимками экрана ПО для проведения нагрузочного тестирования. После чего, данные будут занесены в таблицу 3.4.

Таблица 3.3 – *Результаты функционального тестирования*

№ теста	Наименование теста	ME*	YB	IE	GC	O	MF	Комментарии
1	Валидация обязательных полей	+	+	+	+	+	+	
2	Знак звездочки у всех обязательных полей	-	-	-	-	-	-	Есть всплывающее уведомление
3	Нет ошибок при незаполненных необязательных окнах	+	+	+	+	+	+	
4	Исходящие ссылки	+	+	+	+	+	+	
5	Корректность внутренних ссылок	+	+	+	+	+	+	
6	Отсутствие ссылок, ведущих к одной странице	+	+	+	+	+	+	
7	Ссылки, которые используются для отправки электронной почты (фидбек)	-	-	-	-	-	-	Нет формы обратной связи

№ теста	Наименование теста	ME*	YB	IE	GC	O	MF	Комментарии
8	Страницы, на которые не указаны ссылки	+	+	+	+	+	+	
9	Отсутствие неработающих ссылок	+	+	+	+	+	+	
10	Действительность входных данных	+	+	+	+	+	+	
11	Допустимые значения для поля данных	+	+	+	+	+	+	
12	Недопустимые входные значения для поля данных	+	+	+	+	+	+	
13	Параметры форм, в которых возможно удаление или любая другая модификация данных	-	-	-	-	-	-	Формы для редактирования отсутствуют
14	Функциональность доступных кнопок	+	+	+	+	+	+	
15	Проверка сайта с отключенными cookies	+	+	+	+	+	+	
16	Проверка сайта с включенными cookies	+	+	+	+	+	+	
17	Если cookies имеют продолжительность действия, то активны ли они в указанный период времени	+	+	+	+	+	+	
18	Java Script верно работает	+	+	+	+	+	+	Часы работают
19	Что будет, если пользователь удалит cookies, находясь на сайте	+	+	+	+	+	+	Сайт работает, выбивает авторизацию
20	Что будет, если пользователь удалит cookies после посещения сайта	+	+	+	+	+	+	
21	Проверка всех данных в выпадающих списках	+	+	+	+	+	+	
22	Синтаксические ошибки HTML	+	+	+	+	+	+	
23	Сайт доступен для поисковых машин	-	-	-	-	-	-	Локальный сайт

№ теста	Наименование теста	ME*	YB	IE	GC	O	MF	Комментарии
24	Веб-страница имеет точную карту сайта в формате XML и HTML	-	-	-	-	-	-	Карта сайта отсутствует

В заголовках столбцов приведенной выше таблице используются следующие сокращения:

- ME – Microsoft Edge 44.18362.449.0;
- YB – Yandex Browser 20.3.2.242;
- IE – Internet Explorer 11.778.18362.0;
- GC – Google Chrome 80.0.3987.163;
- O – Opera 66.0.3515.27;
- MF – Mozilla Firefox 75.0.

Для проведения нагрузочного тестирования установлена программа Apache JMeter версии 5.2.1, в которую был загружен сценарий тестирования (рисунок 3.13). В данный сценарий загружен список URL сайта, реализованных на текущий момент.

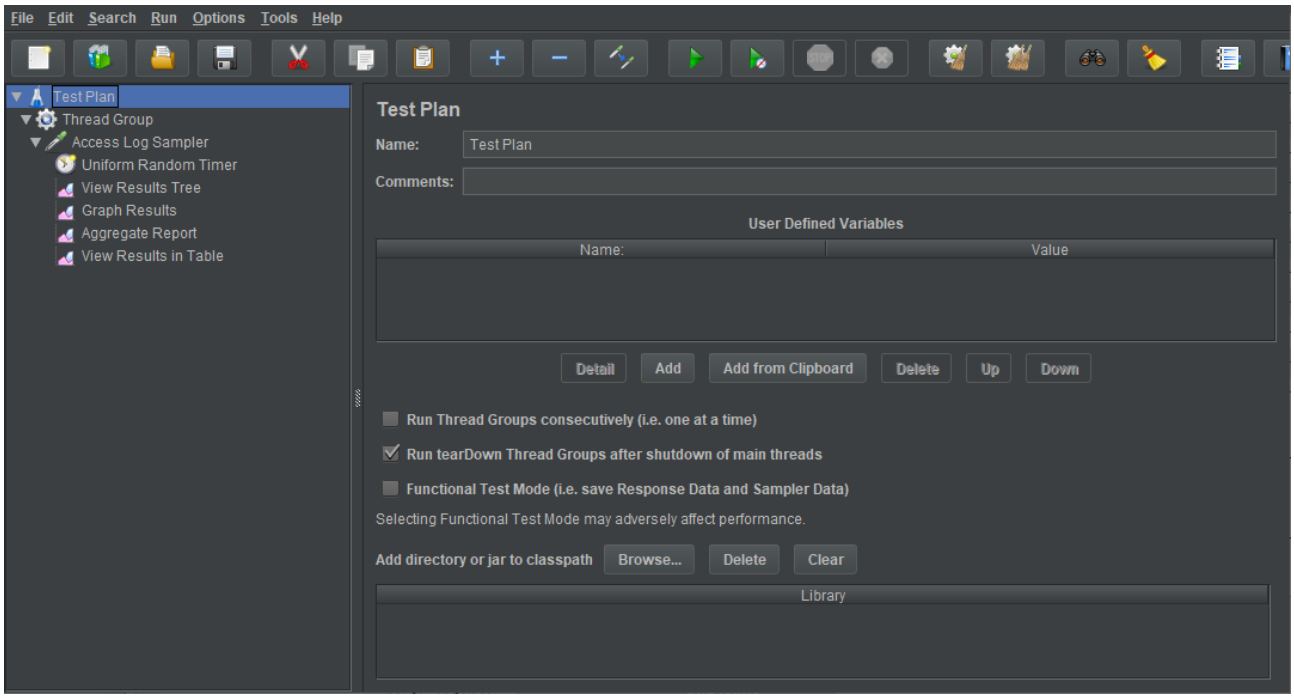


Рисунок 3.13 – Сценарий тестирования в Apache JMeter

После выполнения сценария тестирования на 1000 пользователей, находящихся на сайте, в разделе «Aggregate Report» можно увидеть таблицу 3.3.

Таблица 3.4 – *Результат нагрузочного тестирования на 1000 пользователей*

Label	# Sam-ples	Avg	Median	Min	Max	Err	Throughput	Received
*/index.php	1000	1056	883	8	26155	0	35,3	153,64
*/css/styles.css	1000	1355	1228	2	2421	0	34,5	107,74
*/about.php	1000	1035	808	4	23287	0	33,7	183,42
*/form_register 1.php	1000	1372	1375	6	2346	0	33,6	159,21
*/svedenia.php	1000	1425	1220	4	17111	0	34,8	204,66
*/contacts.php	1000	1249	1187	3	20934	0	37,3	216,59
TOTAL	6000	1248, 66667	1037	2	26155	0	34,9	170,88

В заголовках приведенной выше таблицы используются следующие обозначения:

- Label – веб-страница, на которую осуществляется вход пользователями;
- #Samples – число пользователей, которые пытались войти на соответствующую страницу;
- Avg – среднее время отклика при входе [мс];
- Median – медиана выборки на 1000 пользователей – наиболее часто встречающееся время отклика [мс];
- Min – минимальное значение отклика сайта на вход пользователя [мс];
- Max – максимальное значение отклика сайта на вход пользователя [мс];
- Err – процент неудачных попыток входа на соответствующую страницу [%];

- Throughput – пропускная способность веб-страницы [польз/с];
- Received – скорость передачи данных с веб-страницы [Кб/с].

Как можно увидеть из данных таблицы, среднее время отклика страницы при единовременном входе 1000 пользователей на страницы разработанного сайта составляет 1249 мс. или примерно 1.25 секунд. Это является хорошим результатом для времени отклика веб-страницы, но связаны такие цифры в первую очередь с тем, что отсутствуют страницы, обращающиеся к БД MySQL. По прогнозу, среднее время отклика в дальнейшем снизится. Еще одним важным представленным значением является пропускная способность сайта (throughput), в среднем равная примерно 46 пользователей в секунду.

Для того, чтобы получить полную картину по нагрузочному тестированию сайта оно было проведено в 5 этапов (от t_1 до t_5, в частности, t_1 для 1000 пользователей – поле TOTAL Avg из таблицы 3.4) для повышения точности полученных результатов для 1, 10, 25, 50, 100, 250, 500, 1000, 2500 и 5000 пользователей с одновременным входом. Данные о проведенных тестах занесены в таблицу 3.5.

В начале пять раз проводится проверка отклика системы на различные действия, ее результаты вносятся в соответствующие строки 2-6 столбцов. Далее рассчитывается среднее арифметическое всех измерений по формуле (3.1):

$$t_{ср.ариф.} = \frac{\sum_{i=1}^{n} t_i}{n}, \quad (3.1)$$

а результат заносится в 7 столбец таблицы. В 8 столбце находятся соответствующие средние квадратические отклонения, рассчитанные по формуле (3.2):

$$\sigma = \sqrt{\frac{1}{n} * \sum_{i=1}^{n} (t_i - t_{\text{ср.ариф.}})^2}. \quad (3.2)$$

Погрешность измерений (3.3):

$$\Delta t = \sqrt{(\frac{\sigma}{\sqrt{n}} * t_{\alpha(n-1)})^2 + (A)^2)}, \quad (3.3)$$

в 9 столбце, где n – число измерений, $t_{\alpha(n-1)}$ – доверительный коэффициент Стьюдента, равный 0,95, а A – абсолютная погрешность прибора (в данном случае – электронного секундомера ПО JMeter, который считает данные с точностью до десятитысячной доли секунды), округленная до тысячных долей и равная 0,0005; а в 10 столбце – итоговое время отклика (3.4):

$$t_{\text{отк}} = t_{\text{ср.ариф}} \pm \Delta t. \quad (3.4)$$

Таблица 3.5 – *Результаты нагрузочного тестирования*

К-во польз.	t_1, с	t_2, с	t_3, с	t_4, с	t_5, с	$t_{\text{ср.ариф}}$, с	$t_{\text{ср.квадр.}}$, с	Δt, с	$t_{\text{отк}}$, с
1	0,035	0,07	0,07	0,042	0,028	0,049	0,0198	0,00843	0,049±0,008
10	0,028	0,014	0,014	0,014	0,007	0,0154	0,00767	0,0033	0,015±0,003
25	0,014	0,014	0,021	0,014	0,014	0,0154	0,00313	0,00142	0,015±0,001
50	0,022	0,006	0,005	0,008	0,005	0,0092	0,00726	0,00312	0,009±0,003
100	0,007	0,009	0,038	0,167	0,029	0,05	0,06672	0,02835	0,067±0,028
250	0,035	0,075	0,519	0,103	0,05	0,1564	0,20434	0,08681	0,156±0,087
500	0,079	0,028	0,041	0,055	0,406	0,1218	0,15999	0,06798	0,122±0,068
1000	1,248	1,602	1,034	1,729	2,156	1,5538	0,43571	0,18511	1,554±0,185
2500	5,127	4,913	4,033	4,854	5,036	4,7926	0,4377	0,18596	4,793±0,438
5000	10,11	8,135	8,911	8,547	8,555	8,8516	0,75521	0,32085	8,852±0,755

На основе данных таблицы 3.5 (столбцы 1 и 7) была составлена диаграмма (рисунок 3.14), наглядно показывающая зависимость среднего времени отклика веб-сайта в целом от количества единовременно входящих на него пользователей.

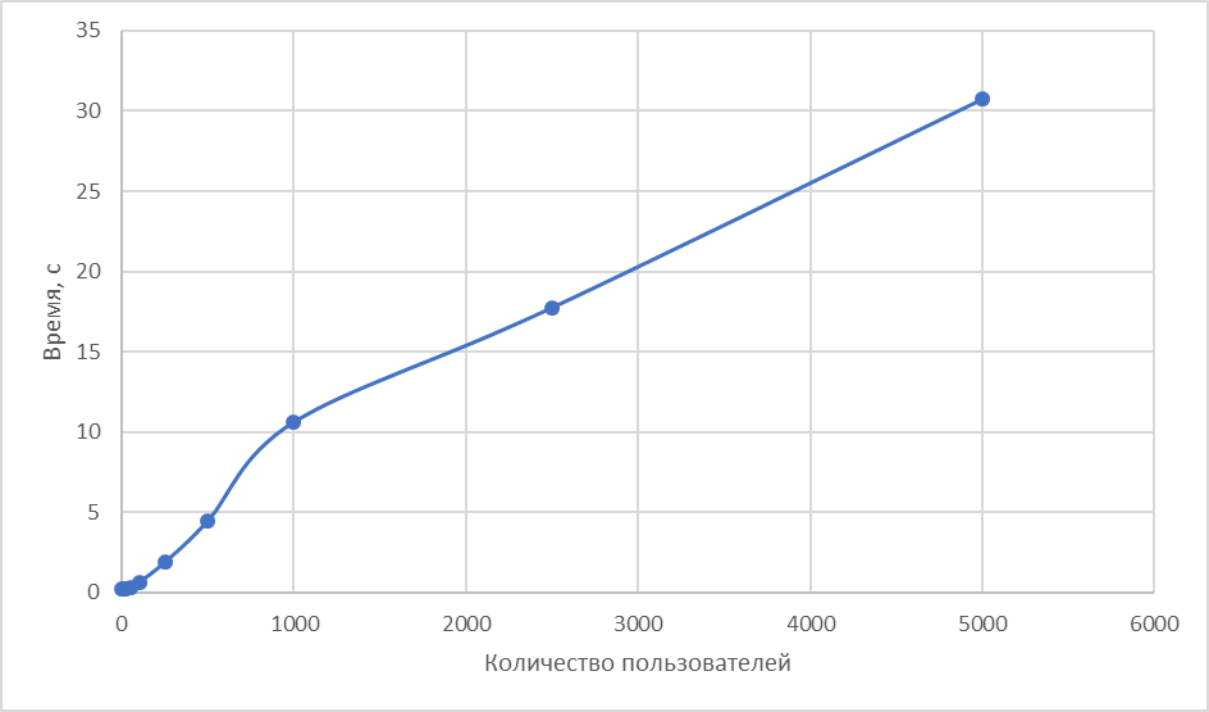


Рисунок 3.14 – *Зависимость среднеарифметического времени отклика от числа пользователей*

3.5. Результаты и их обсуждения

В данной главе на основе ранее выбранной каскадной модели внедрения и методологии ASAP осуществлена реализация ключевого бизнес-процесса «Лечить пациента». Первым этапом было формирование матрицы отслеживания требований, которая позволила наглядно отобразить и систематизировать полученные в ходе изучения требования. На следующем этапе спроектированы пользовательские интерфейсы и данные, на основе которых проведена разработка ПО. Проведено моделирование процессов в

модели «TO-BE» до 3 уровня описания и на основе этих данных составлена обновленная карта процессов.

Далее проведено функциональное тестирование, показавшее некоторые недоработки, которые позднее будут устранены дополнительным функционалом. Суммарное число ошибок = 30 (по 5 в каждом из 6 проверяемых браузеров).

На основе полученных результатов нагрузочного тестирования можно заключить, что сайт единовременно выдерживает на должном уровне небольшое количество пользователей, соответствующее районной больнице (до 1000 человек). После этого числа возрастает время ожидания загрузки страницы. Однако, при совершенствовании мощностей, либо распределении запросов пользователей во времени, время отклика можно снизить до необходимого уровня.

Раздел 4. Реализация бизнес-процесса «Принять пациента» на основе итерационной модели, используя Agile Scrum

Разработка программного обеспечения будет производиться в последовательности, приведенной в таблице 4.1.

Таблица 4.1 – *Этапы разработки ПО с использованием Agile Scrum*

№ действия \ Этап разработки	Начало	Первый спринт (реализация требований 8-11 бэклога)	Второй спринт (реализация требований 1-4 бэклога)	Третий спринт (реализация требований 5-7 бэклога)
1	Формирование бэклога продукта (таблица 4.2)	Описание ключевого бизнес-процесса «Принять пациента» в модели «AS-IS»	Описание ключевого бизнес-процесса «Принять пациента» в модели «AS-IS»	Описание ключевого бизнес-процесса «Принять пациента» в модели «TO-BE»
2		Проектирование данных	Проектирование данных	Проектирование данных
3		Проектирование пользовательских интерфейсов	Проектирование пользовательских интерфейсов	Проектирование пользовательских интерфейсов
4		Реализация процессов средствами PHP	Реализация процессов средствами PHP	Реализация процессов средствами PHP
5		Тестирование	Тестирование	Тестирование
6				Оценка результатов

4.1. Бэклог продукта

Для формирования бэклога продукта из таблицы 2.1 были взяты требования, относящиеся к бизнес-процессу «Принять пациента» и занесены в таблицу 4.2 с указанием приоритета (также отсортированные по нему) и номера спринта (продолжительность спринта – 1 неделя), на котором требование планируется к реализации.

Таблица 4.2 – *Бэклог продукта*

№	Пользовательская история	Компонент покрытия	Приоритет	№ спринта
1	Как врачу мне необходимо место для хранения информации обо мне, а также для авторизации	Таблица «Персонал» в БД	Высокий	1
2	Как врачу мне необходимо место для хранения карты пациента	Таблица «Пациенты» в БД	Высокий	1
3	Как врачу мне необходимо место для хранения информации об анамнезе пациента	Таблица «Анамнез» в БД	Средний	1
4	Как врачу мне необходимо не допустить возможность для изменения данных в карте пациента	Формы регистрации и авторизации персонала	Высокий	1
5	Как врачу мне необходима возможность для удаления карты пациента	Форма удаления медицинской карты	Средний	2
6	Как врачу мне необходимо редактировать данные из карты пациента	Форма просмотра и редактирования медицинской карты	Средний	2
7	Как врачу мне необходимо добавлять и редактировать эпикриз пациента	Форма просмотра и редактирования медицинской карты	Средний	2
8	Как врачу мне необходимо добавлять и редактировать анамнез пациента при поступлении	Форма просмотра и редактирования медицинской карты	Средний	2
9	Как врачу мне необходимо добавлять и редактировать данные о температуре пациента	Форма просмотра и редактирования медицинской карты	Средний	3
10	Как врачу мне необходимо добавлять и редактировать врачебные назначения	Форма просмотра и редактирования медицинской карты	Средний	3
11	Как врачу мне необходимо добавлять и изменять данные в «Дневнике» пациента	Форма просмотра и редактирования медицинской карты	Средний	3

4.2. Первый спринт: реализация требований 9-12 бэклога

В данном спринте необходимо реализовать три требования, связанных с добавлением новых таблиц в ранее созданную БД, а также требование для создания формы регистрации и авторизации персонала, которая будет обращаться к создаваемой в этом спринте таблице.

4.2.1. Описание ключевого бизнес-процесса «Принять пациента в модели «AS-IS»

Аналогично предыдущему этапу разработки, для понимания ситуации в учреждении необходимо более детально углубиться в процесс до 2 уровня (рисунки 4.1 и 4.2).

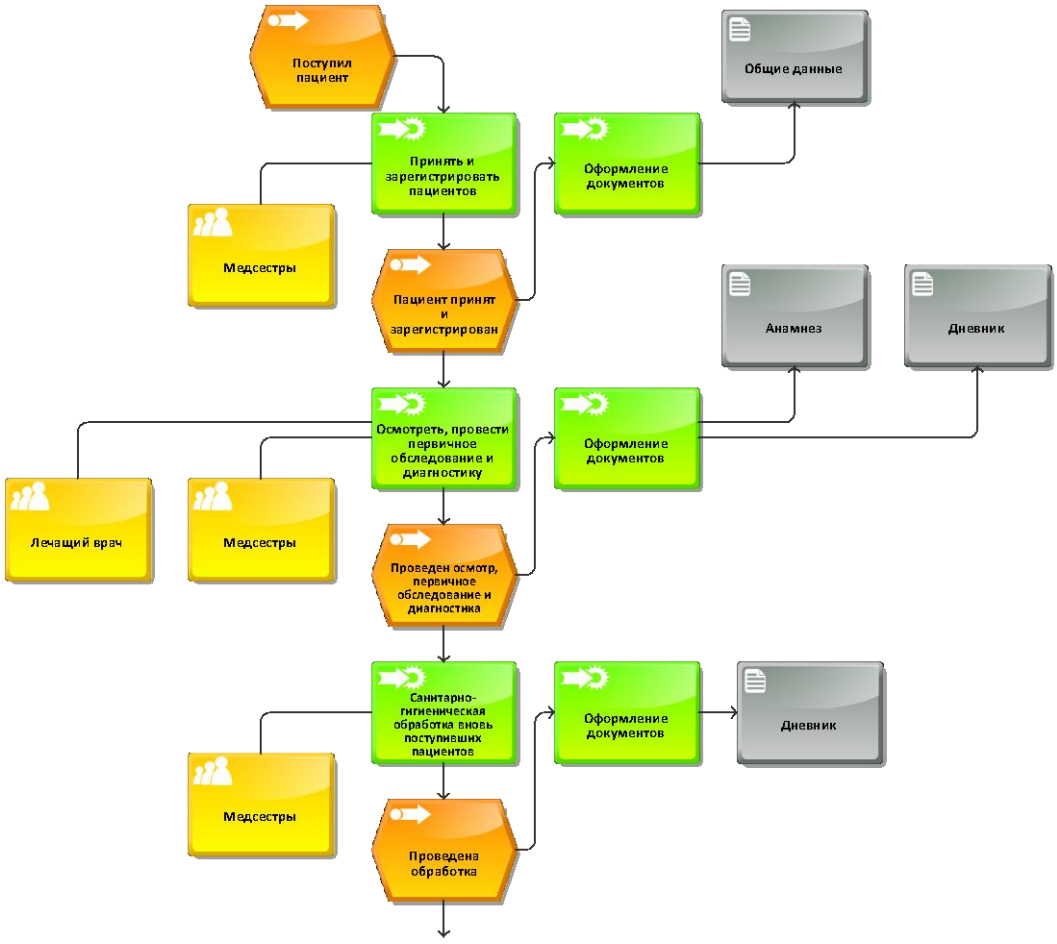


Рисунок 4.1 – *Процесс «Принять пациента» в модели ARIS eEPC «AS-IS» ч.1 (2 уровень)*

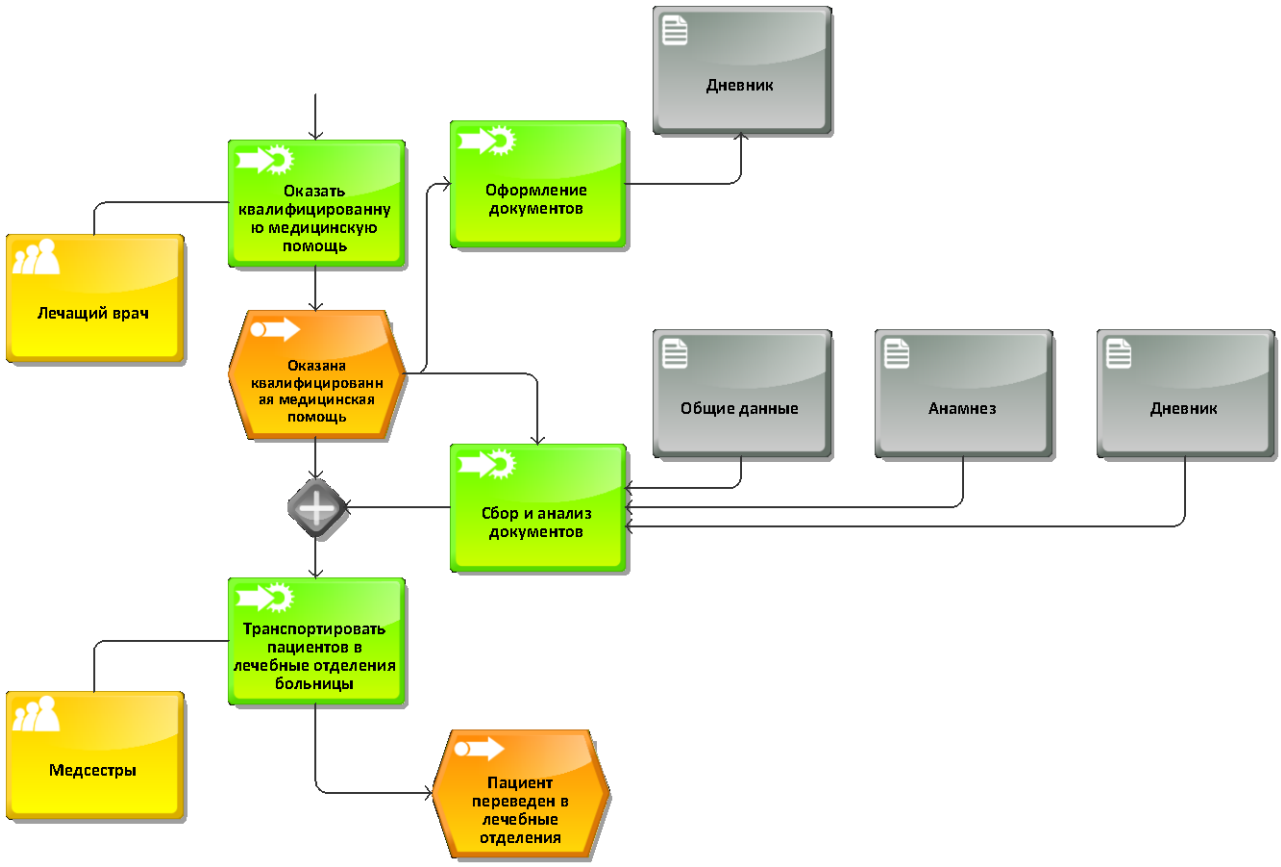


Рисунок 4.2 – *Процесс «Принять пациента» в модели ARIS eEPC «AS-IS» ч.2 (2 уровень)*

4.2.2. Проектирование данных

В таблице 4.3 представлены типы и классы добавляемых данных.

Таблица 4.3 – *Классы данных*

Класс данных	Поле	Тип данных	Количество символов
Пациент	Номер карты	Счетчик	10
	Фамилия	Небольшой текст	25
	Имя	Небольшой текст	25
	Отчество	Небольшой текст	25
	Пол	Небольшой текст	5
	Возраст	Числовой	5
	Постоянное место жительства (ПМЖ)	Средний текст	255
	Место работы, профессия или должность	Средний текст	255

Класс данных	Поле	Тип данных	Количество символов
Сотрудник	Код сотрудника	Счетчик	10
	Фамилия	Небольшой текст	25
	Имя	Небольшой текст	25
	Отчество	Небольшой текст	25
	Должность	Небольшой текст	25
	Порядок записи	Средний текст	100
	График приема	Дата и время	14
	Данные сертификата специалиста	Средний текст	255
	Дополнительные данные сертификата	Средний текст	255
	E-mail	Средний текст	255
	Пароль	Средний текст	255
Анамнез	Номер записи	Счетчик	10
	Номер карты	Числовой	10
	Код сотрудника	Числовой	10
	Анамнез пациента	Текст	1000

После их нормализации и приведения к 3 нормальной форме, они были добавлены в уже существующую архитектуру (рисунок 4.3).

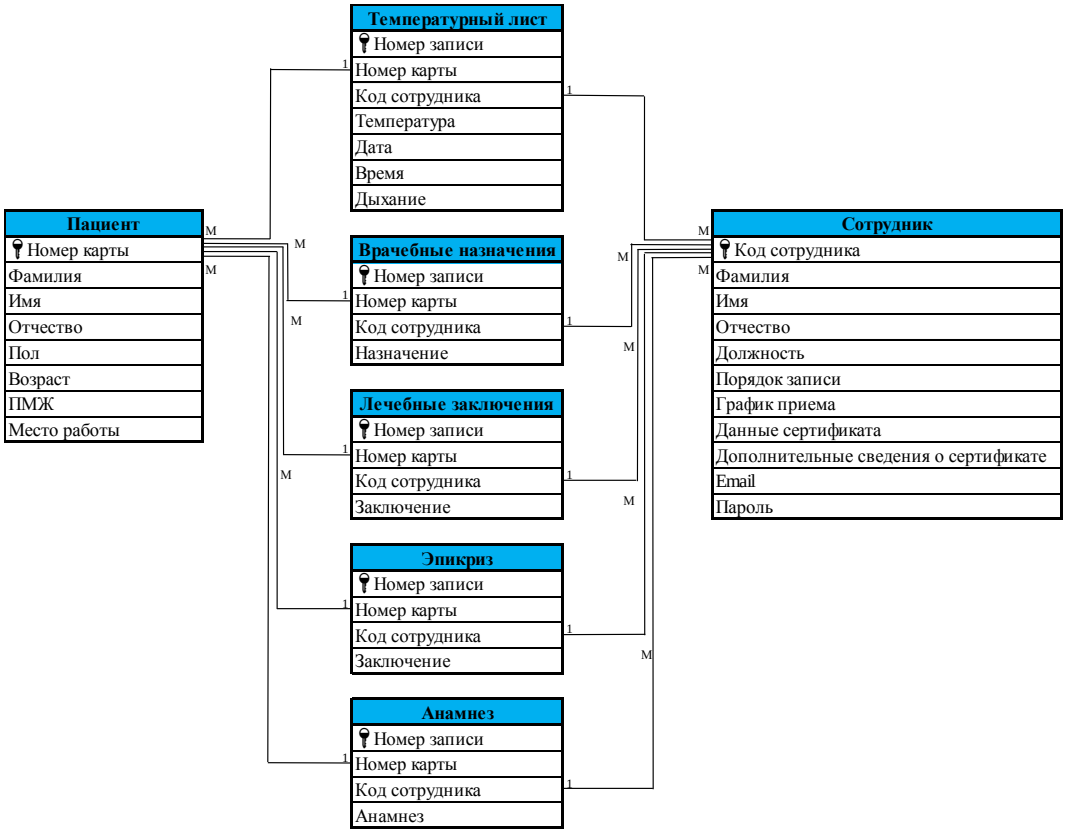


Рисунок 4.3 – *Архитектура классов данных*

4.2.3. Проектирование пользовательских интерфейсов

В MS Visio спроектированы формы регистрации (рисунок 4.4) и авторизации (рисунок 4.5) персонала для реализации функции разделения доступа между сотрудником и пациентом.

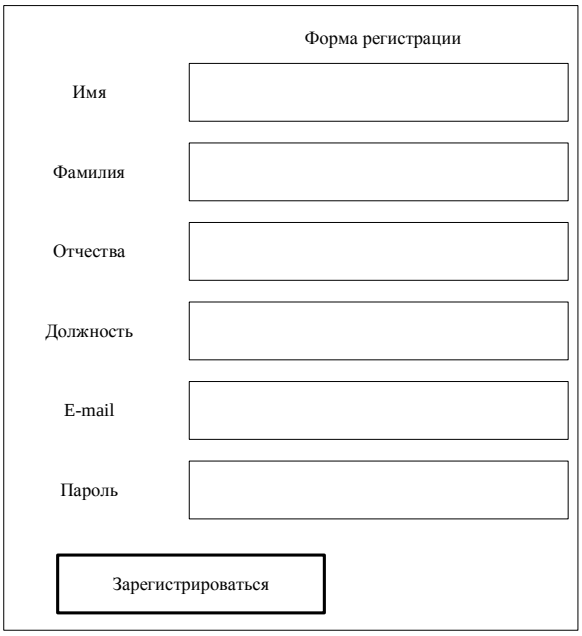


Рисунок 4.4 – *Схема формы регистрации сотрудника*

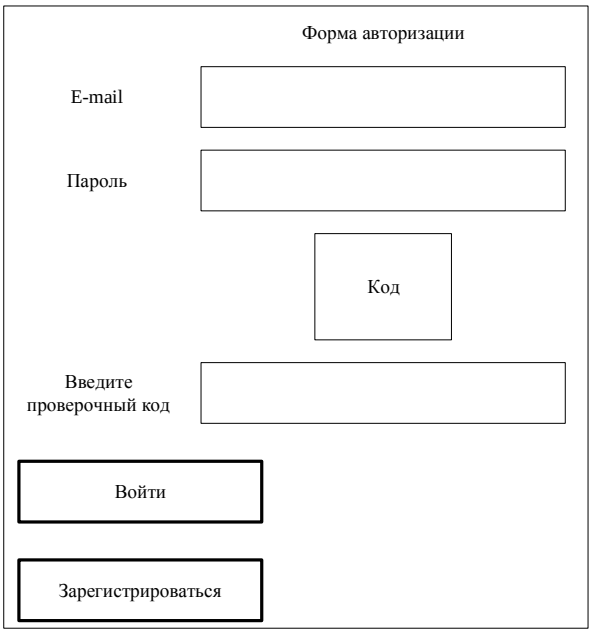


Рисунок 4.5 – *Схема формы авторизации сотрудника*

Поскольку в архитектуру данных и пользовательские интерфейсы вносятся изменения, необходимо отразить их на ранее приведенной (рисунок 3.4) схеме взаимодействия пользовательских экранов – элементе 1 (рисунок 4.6).

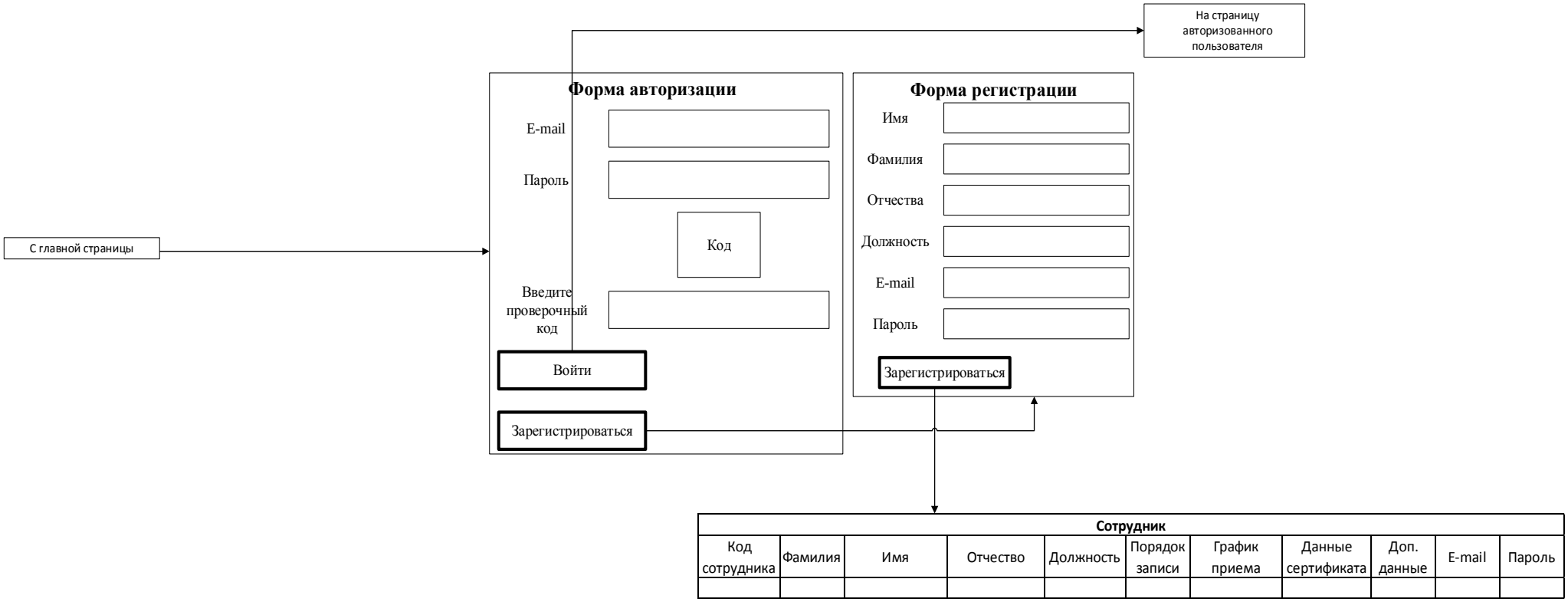


Рисунок 4.6 – *Дополнение к схеме взаимодействия пользовательских экранов*

Также интерфейс пополнится связью между формой регистрации нового пациента и добавляемой в данном спринте таблицей (рисунок 4.7).

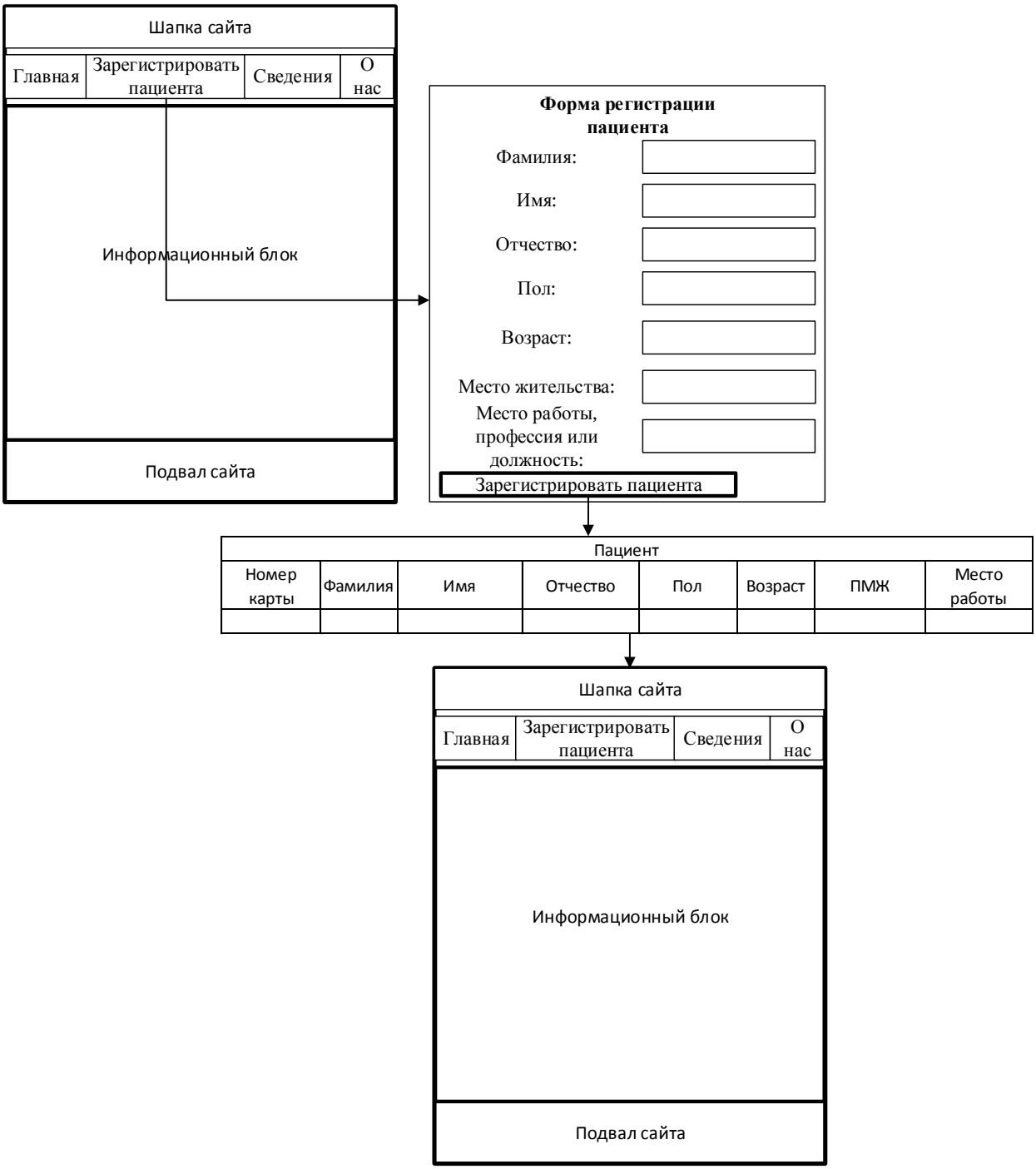


Рисунок 4.7 – *Взаимодействие экранов при добавлении нового пациента*

Добавленная таблица «Анамнез» и другие ранее созданные таблицы и их взаимодействие с веб-интерфейсом будут описаны позднее, на этапе добавления форм изменения этих таблиц (2 и 3 спринты).

4.2.4. Реализация при помощи средств PHP

Фрагменты реализованных БД представлены на рисунках 4.8 – 4.10. Данные в таблицы вносились через СУБД MySQL.

id	first_name	last_name	name	gender	age	pmg	job
1	Сергей	Иванов	Петрович	М	25	Москва	Учитель
4	Петр	Сидоров	Федорович	Мужской	63	Москва	Работа
5	Игорь	Иванов	Петрович	Мужской	13	Россия	школа

Рисунок 4.8 – *Данные в таблице «Пациенты»*

stuff_id	first_name	last_name	name	dolg	email	password
1	Иван	Иванов	Иванович	Терапевт	123@mail.ru	4828140403f6eaee3b5af62a0b09ae61
2	Наталья	Катасонова	Сергеевна	медсестра	1234@mail.ru	4828140403f6eaee3b5af62a0b09ae61

Рисунок 4.9 – *Данные в таблице «Персонал»*

id	stuff_id	zap
4	1	Поступил больным

Рисунок 4.10 – *Данные в таблице «Анамнез»*

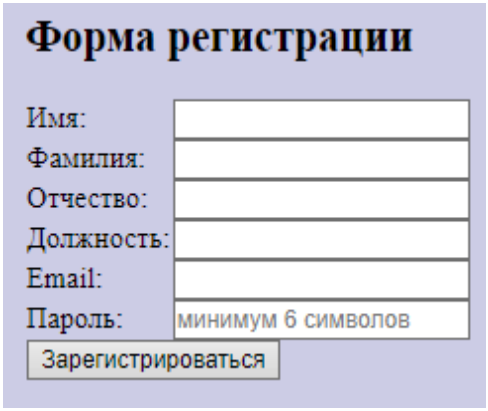


Рисунок 4.11 – *Форма регистрации персонала*

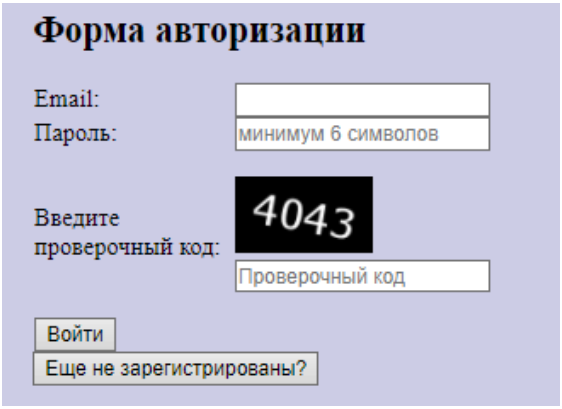


Рисунок 4.12 – *Форма авторизации персонала*

Далее, при помощи средств php и html реализованы соответствующие элементы пользовательского интерфейса, представленные на рисунках 4.11 и 4.12. Листинг продемонстрированных веб-страниц приведен в приложении А.

4.2.5. Тестирование разработанной программы

Добавленные элементы кода и веб-сайт функционально протестированы по чек-листу, приведенному в таблице 3.3, а не прошедшие тест пункты занесены в таблицу 4.4.

Таблица 4.4 – *Не пройденные функциональные тесты*

№ теста	Наименование теста	ME	YB	IE	GC	O	MF	Комментарии
7	Ссылки, которые используются для отправки электронной почты (фидбек)	-	-	-	-	-	-	Нет формы обратной связи
8	Страницы, на которые не указаны ссылки	-	-	-	-	-	-	Добавлены новые страницы

4.3. Второй спринт: реализация требований 1-4 бэклога

В данном спринте необходимо реализовать систему поиска данных в соответствующих таблицах, созданных ранее, и их выгрузке на веб-

интерфейс для возможности последующей обработки (редактирования или удаления).

Исходя из требований, необходима реализация веб-форм для возможности запроса информации из БД, с последующей их загрузкой. Для данной задачи проектирование данных не потребуется, т.к. в структуру БД изменения вноситься не будут (необходимые для выгрузки таблицы созданы ранее). Также в бэклог спринта добавлены действия по устранению дефектов из 8, 9, 22 из предыдущего функционального тестирования.

4.3.1. Описание ключевого бизнес-процесса «Принять пациента» в модели «AS-IS»

В рамках данного спринта произведено углубление в описание процесса до 3 уровня (рисунки 4.13 и 4.14). На основе рассмотренных за два спринта процессов можно дополнить ранее представленную на рисунке 2.6 карту процессов (рисунок 4.15).

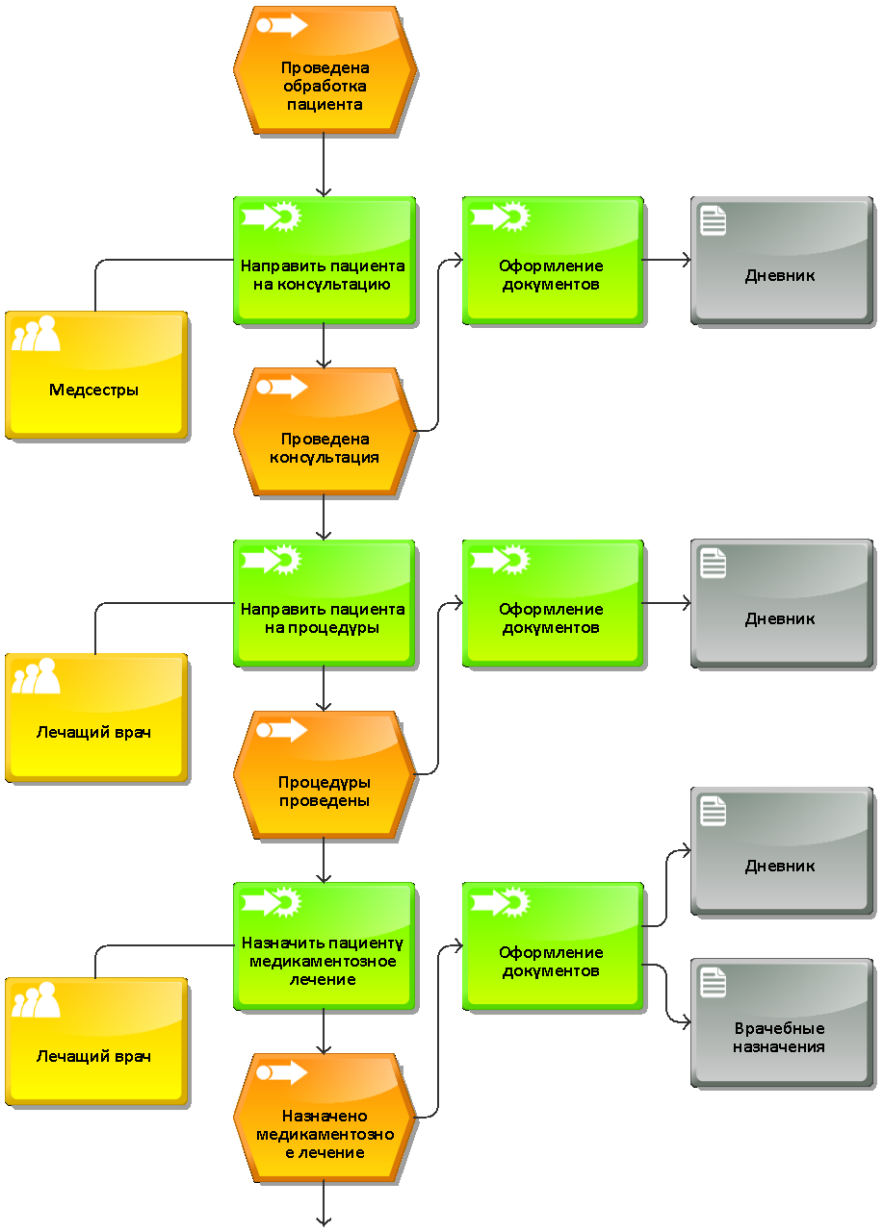


Рисунок 4.13 – *Процесс «Оказать квалифицированную медицинскую помощь» в модели ARIS eEPC «AS-IS» ч.1 (3 уровень)*

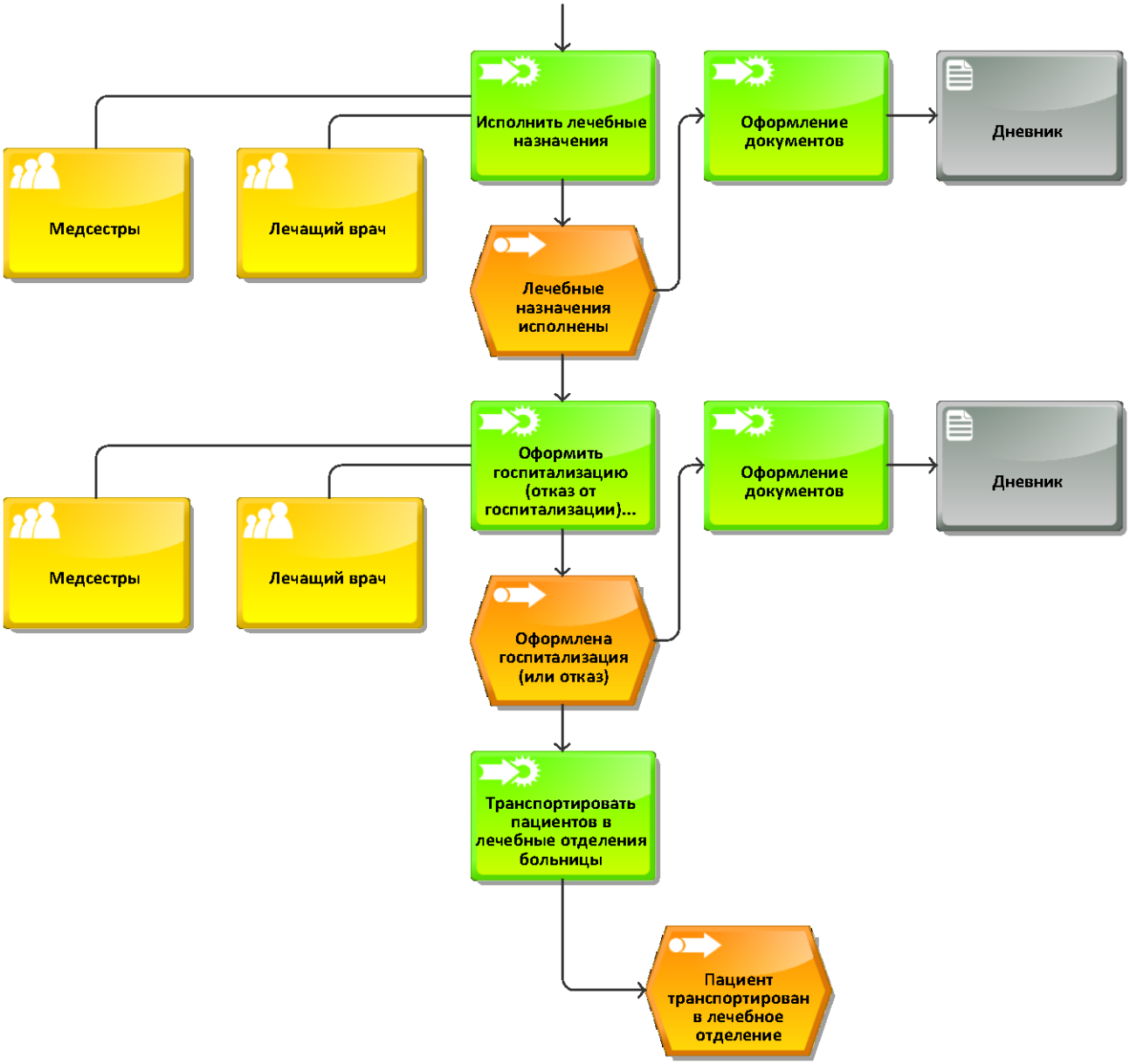


Рисунок 4.14 – *Процесс «Оказать квалифицированную медицинскую помощь» в модели ARIS eEPC «AS-IS» ч.2 (3 уровень)*

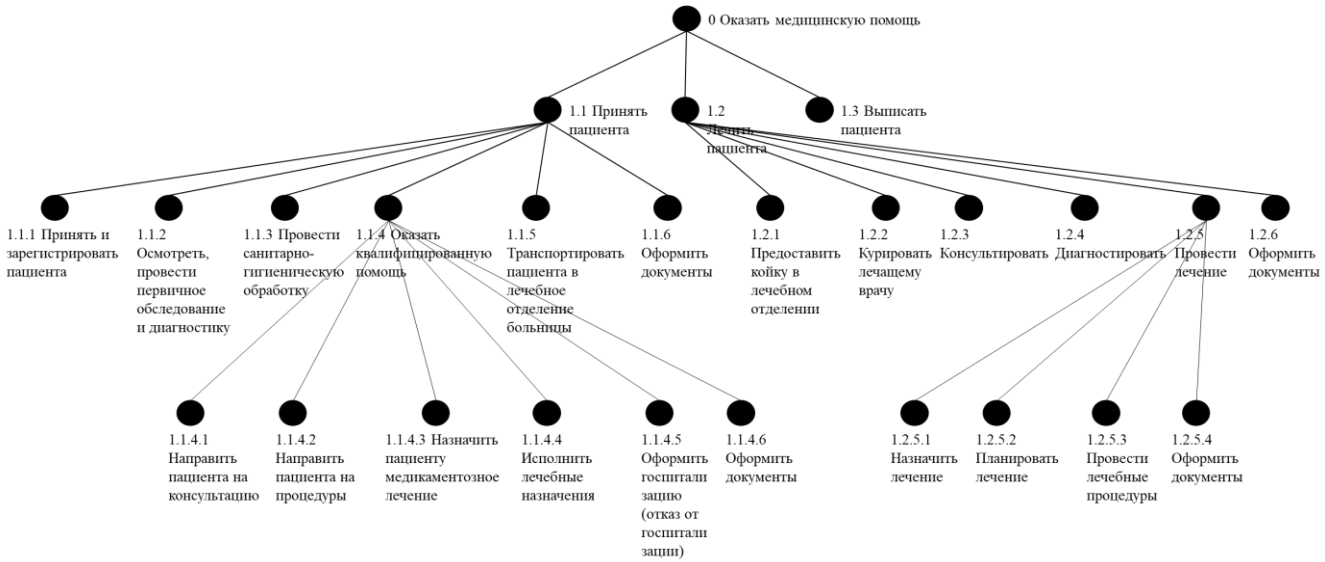


Рисунок 4.15 – *Карта процессов в модели «AS-IS»*

4.3.2. Проектирование пользовательских интерфейсов

Примерное содержание веб-формы с запросом данных по пациенту представлено на рисунке 4.16.

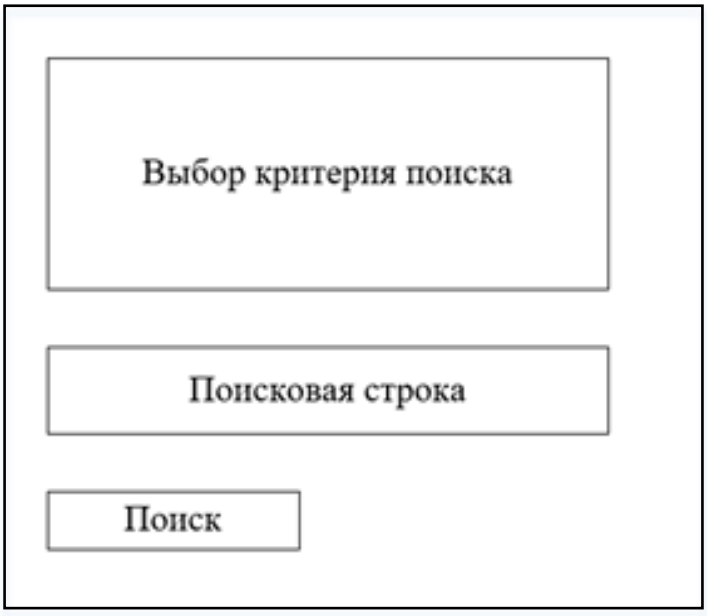


Рисунок 4.16 – *Проект веб-формы поиска карты*

На рисунке 4.17 приведена схема формы просмотра и редактирования данных из таблиц с данными из карты пациента, а на рисунке 4.18 –

взаимодействие реализуемых в данном спринте форм с уже созданными таблицами.

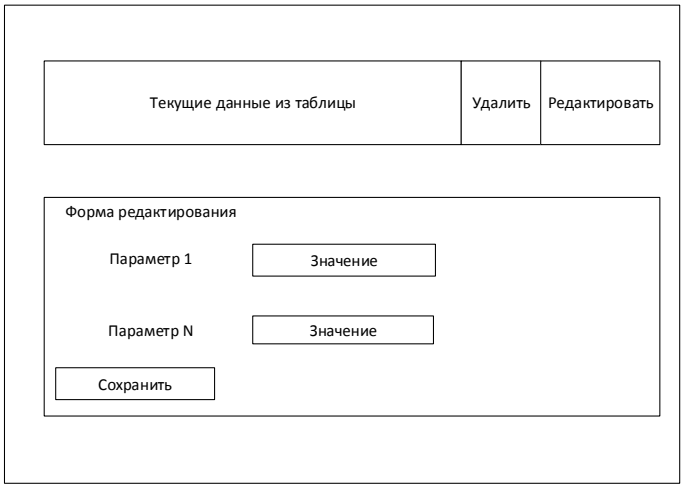


Рисунок 4.17 – *Форма просмотра и редактирования данных карты*

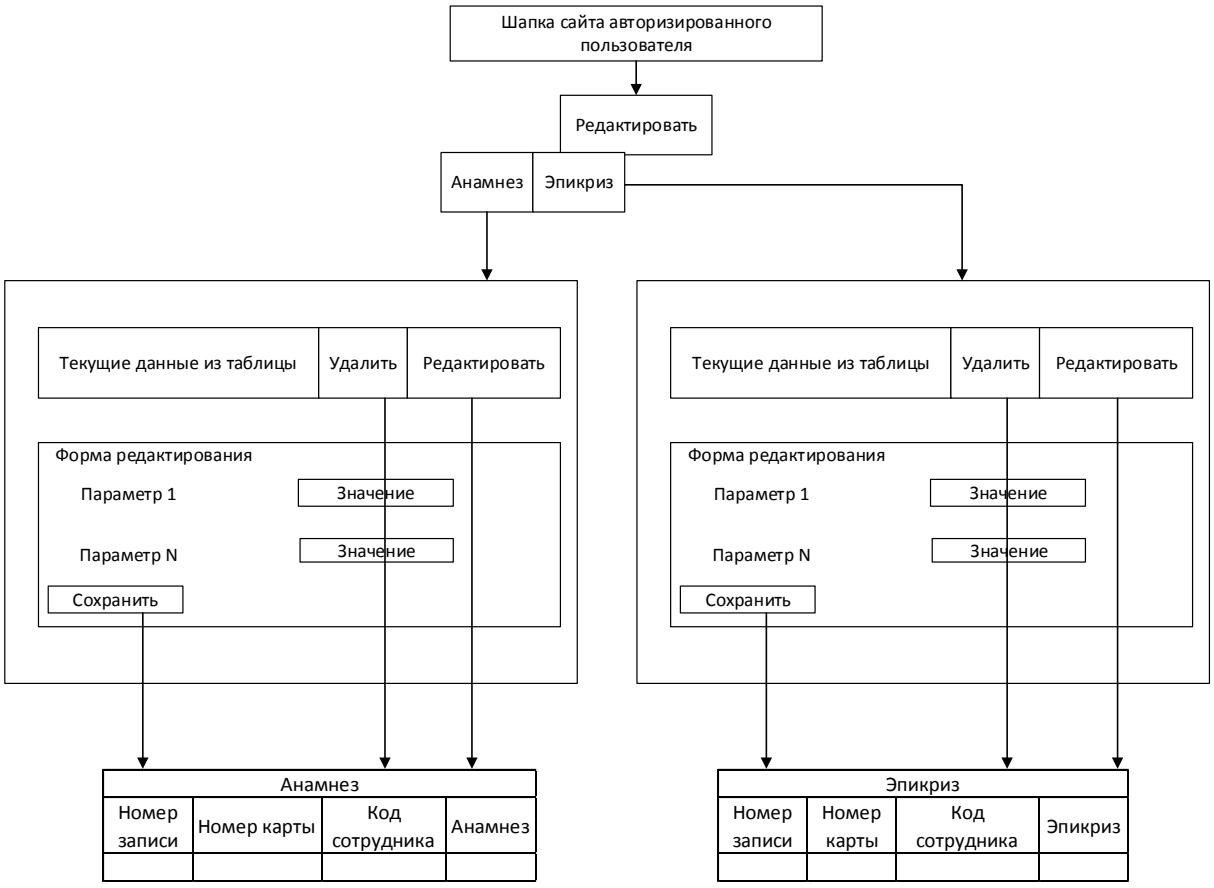


Рисунок 4.18 – *Схема взаимодействия форм редактирования с таблицами БД*

Также необходимо показать взаимодействие формы поиска и удаления карты с таблицами и веб-интерфейсом – элемент 2 (рисунок 4.19).

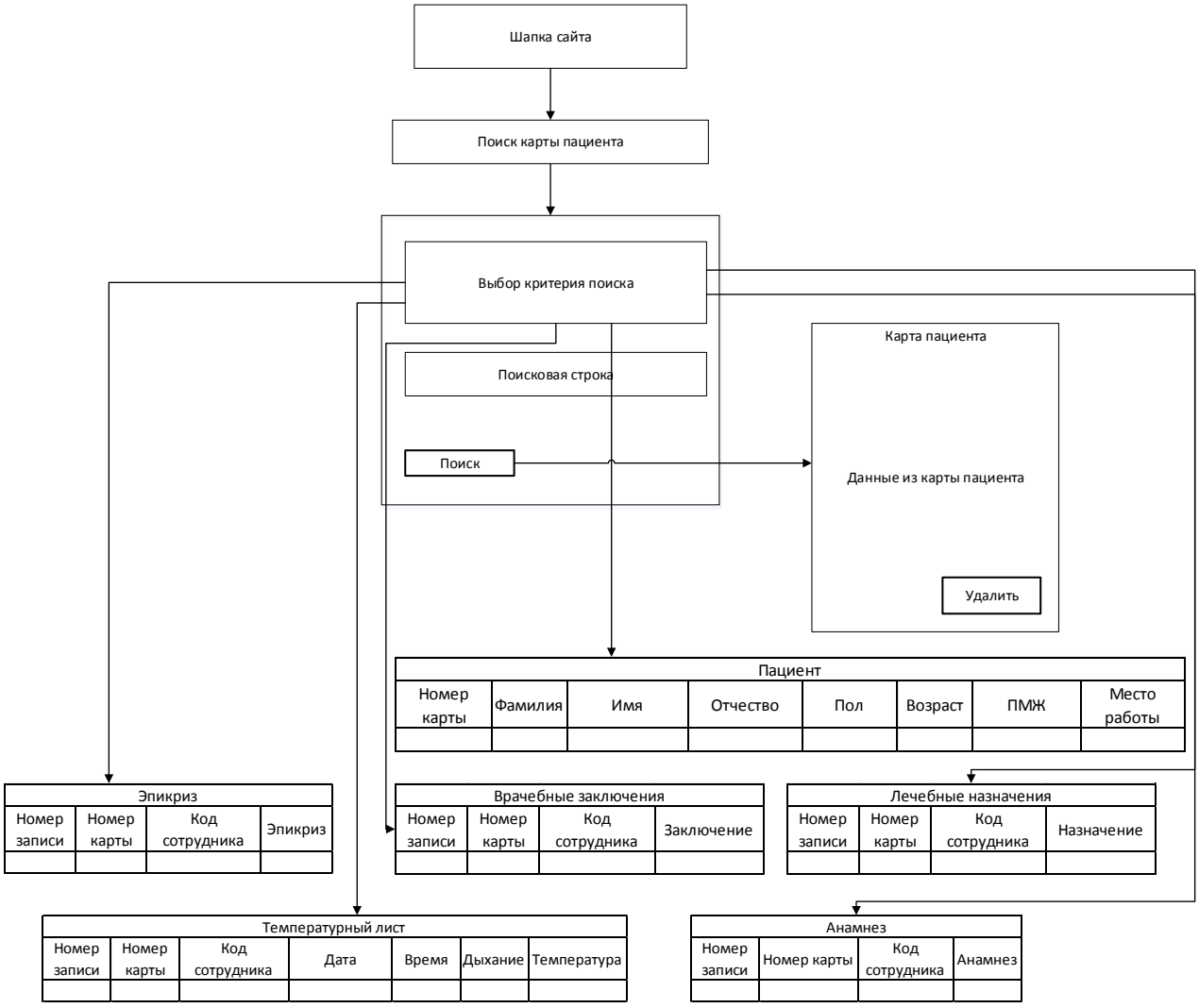


Рисунок 4.19 – *Схема взаимодействия пользовательских экранов при поиске и удалении карты пациента*

4.3.3. Реализация средствами PHP

В результате выполнения спринта с помощью средств PHP и HTML добавлена веб-форма запроса (рисунок 4.20), на рисунке 4.21 пример выполнения запроса информации по карте (для пациента).

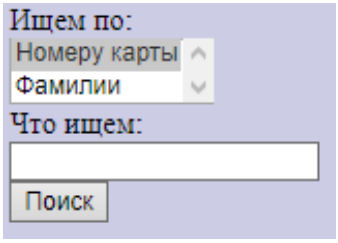


Рисунок 4.20 – *Веб-форма поиска карты*

Рисунок 4.21 – *Результат поиска карты пациента*

На рисунке 4.22 представлена форма редактирования информации о пациенте для сотрудников (механизм редактирования информации для каждой таблицы идентичен).

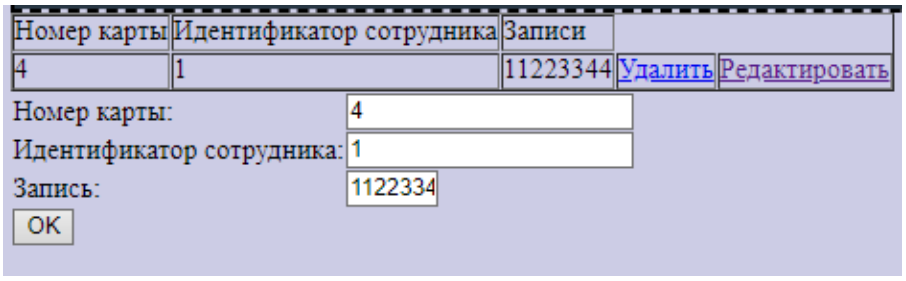


Рисунок 4.22 – *Форма редактирования записи из таблицы*

В приложении А приведен листинг соответствующих веб-страниц (edit.php и search.php).

4.3.4. Тестирование разработанной программы

Добавленные элементы кода, исправления в нем и веб-сайт снова функционально протестированы, а не пройденные тесты занесены в таблицу 4.5.

Таблица 4.5 – *Не пройденные функциональные тесты*

№ тест	Наименование теста	ME	YB	IE	GC	O	MF	Комментарии
9	Отсутствие неработающих ссылок	-	-	-	-	-	-	Не сработала одна ссылка
13	Параметры форм, в которых возможно удаление или любая другая модификация данных	-	-	-	-	-	-	Формы для редактирования отсутствуют

4.4. Третий спринт: реализация требований 5-7 бэклога

В данном спринте необходимо реализовать схожие с ранее созданными функции поиска и редактирования содержимого таблиц, а также создать объединяющую их выгрузку данных по конкретному пациенту. Для выполнения задачи по данному спринту изменение архитектуры пользовательских данных не требуется. В рамках данного спринта добавляется количество данных из медицинской карты пациента, которые выгружаются при запросе (при этом сам запрос и его форма остаются неизменны).

4.4.1. Описание ключевого бизнес-процесса «Принять пациента» в модели «TO-BE»

Для понимания изменений, которые вносит разрабатываемая система в структуру процессов организации, необходимо рассмотреть процессы на втором (рисунки 4.23 и 4.24) и третьем (рисунки 4.25 и 4.26) уровне описания

в модели «TO-BE». На основе полученных данных можно составить обновленную карту процессов в модели «TO-BE» (рисунок 4.27).

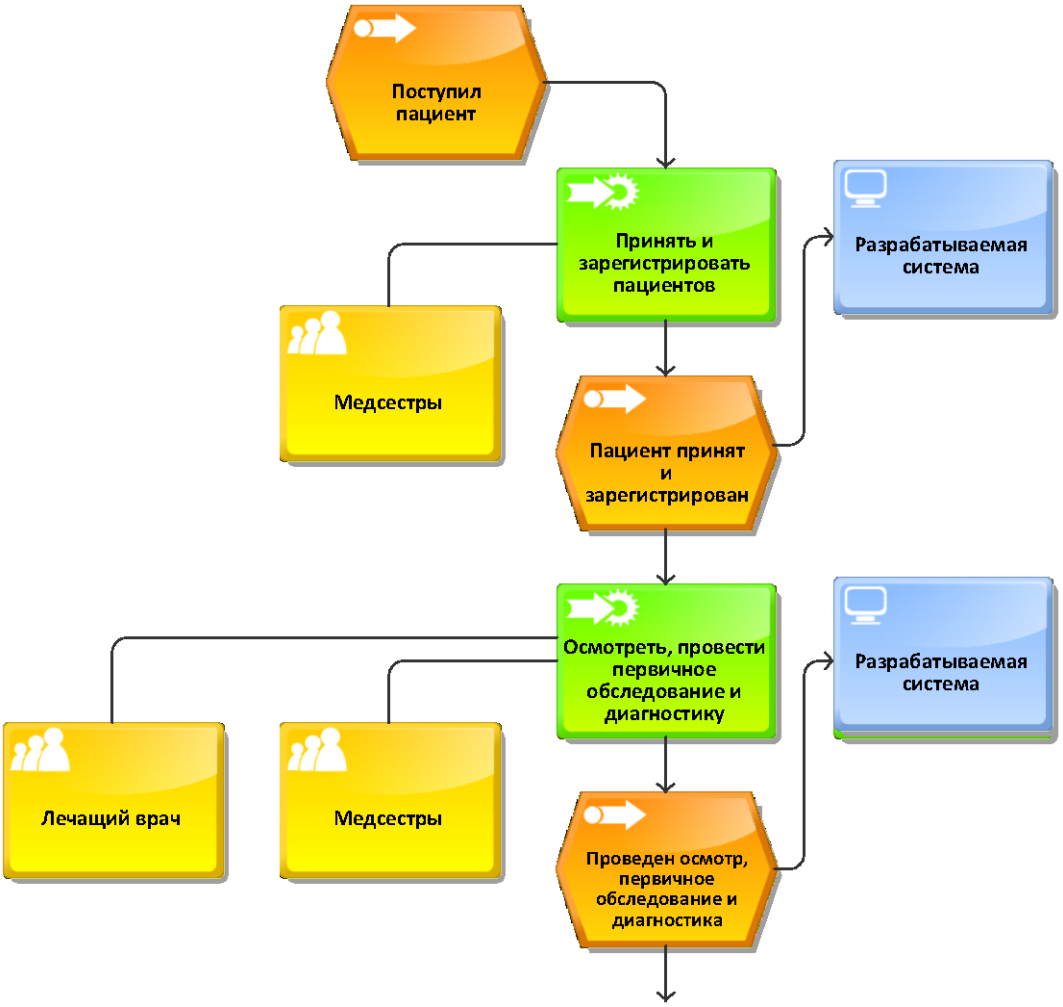


Рисунок 4.23 – *Процесс «Принять пациента» в модели ARIS eEPC «TO-BE» ч.1 (2 уровень)*

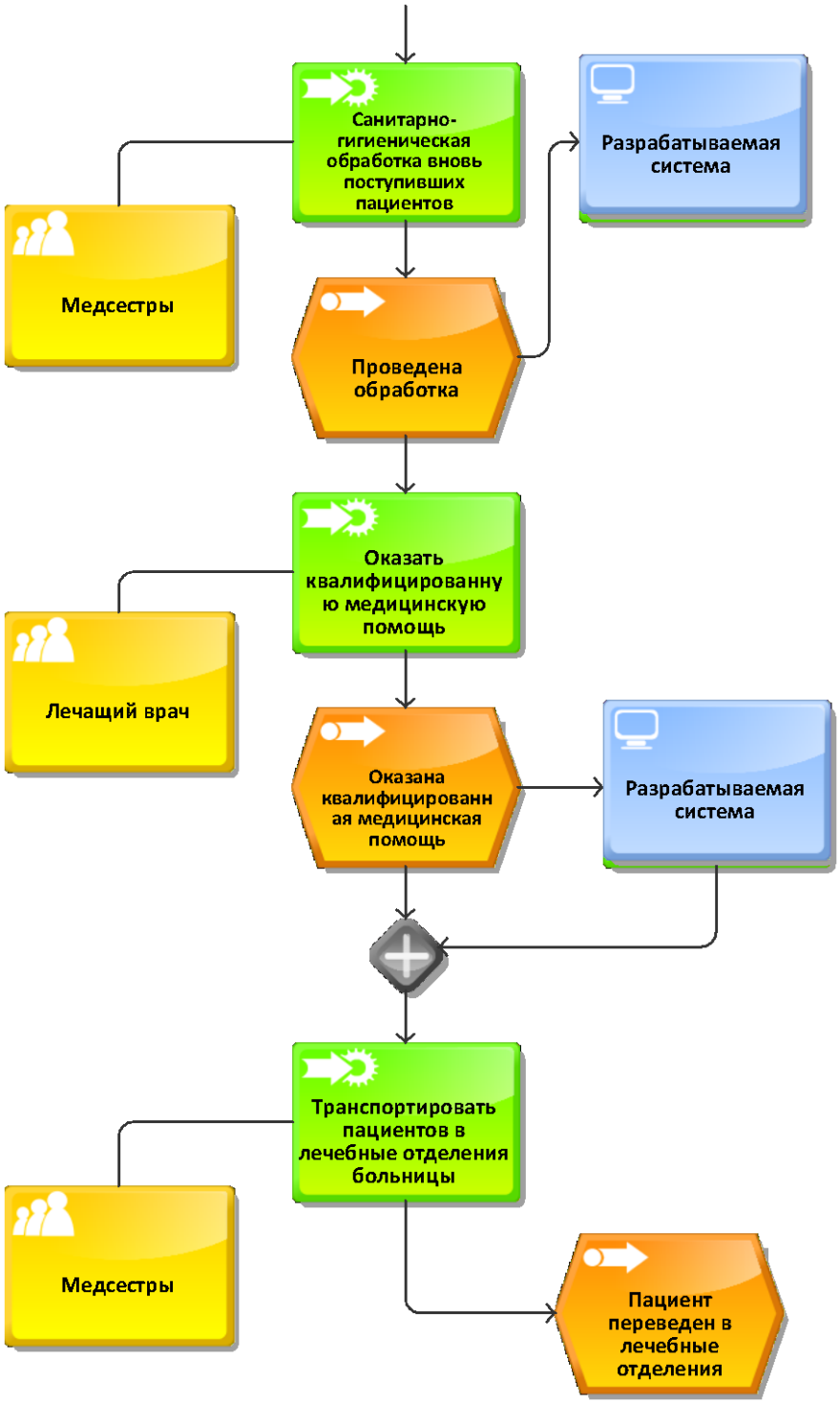


Рисунок 4.24 – *Процесс «Принять пациента» в модели ARIS eEPC «TO-BE» ч.2 (2 уровень)*

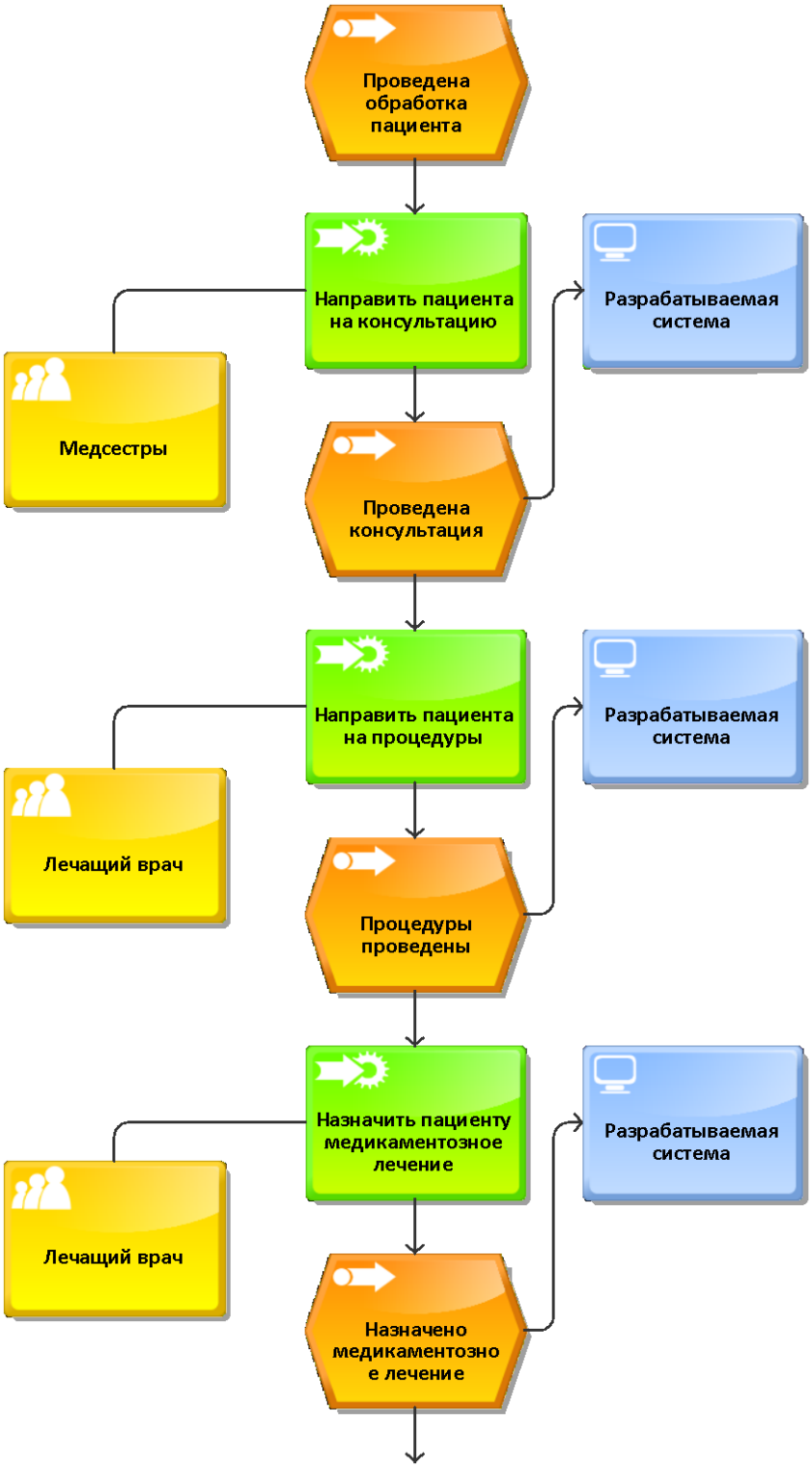


Рисунок 4.25 – *Процесс «Оказать квалифицированную медицинскую помощь» в модели ARIS eEPC «TO-BE» ч.1 (3 уровень)*

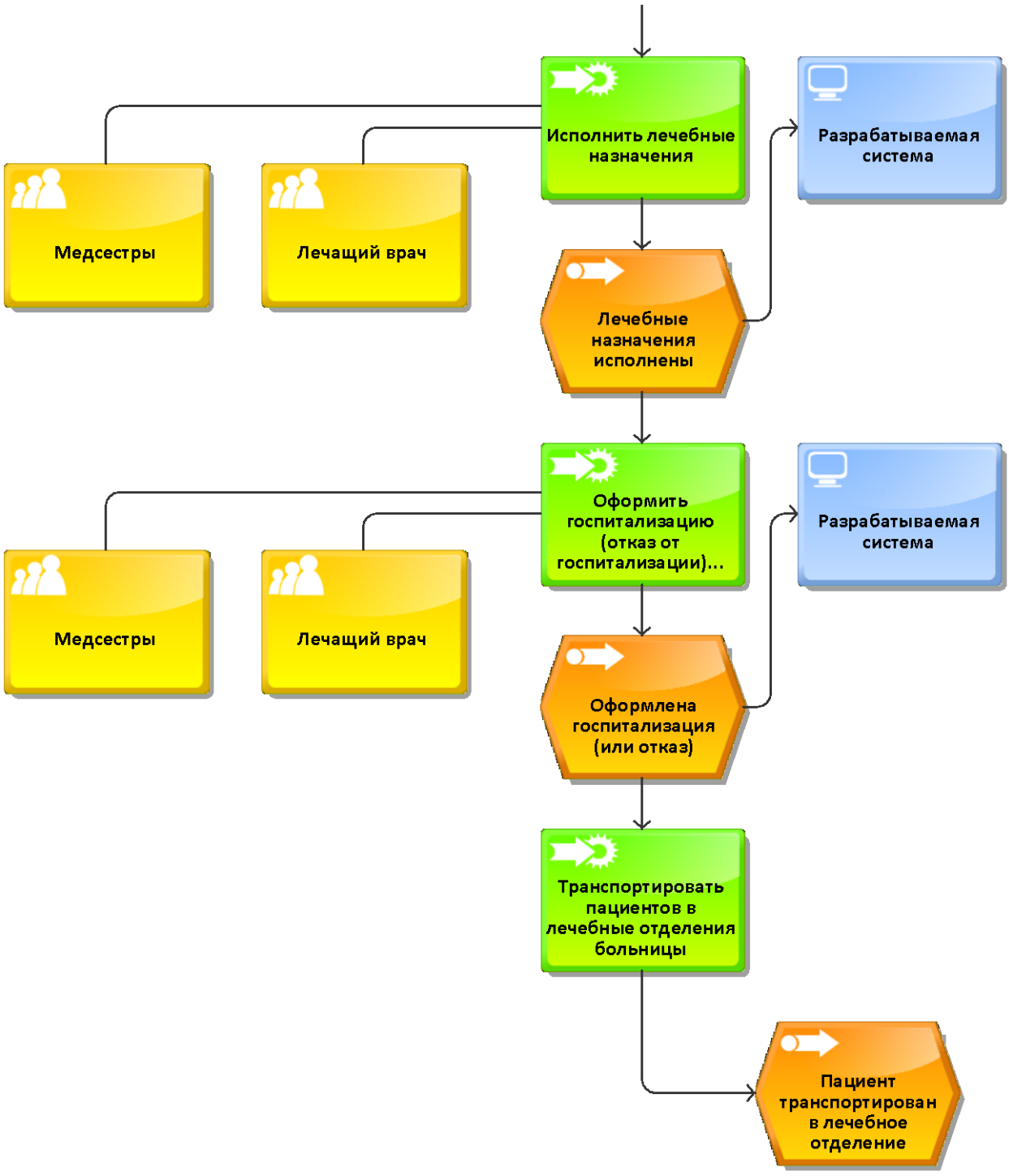


Рисунок 4.26 – *Процесс «Оказать квалифицированную медицинскую помощь» в модели ARIS eEPC «TO-BE» ч.2 (3 уровень)*

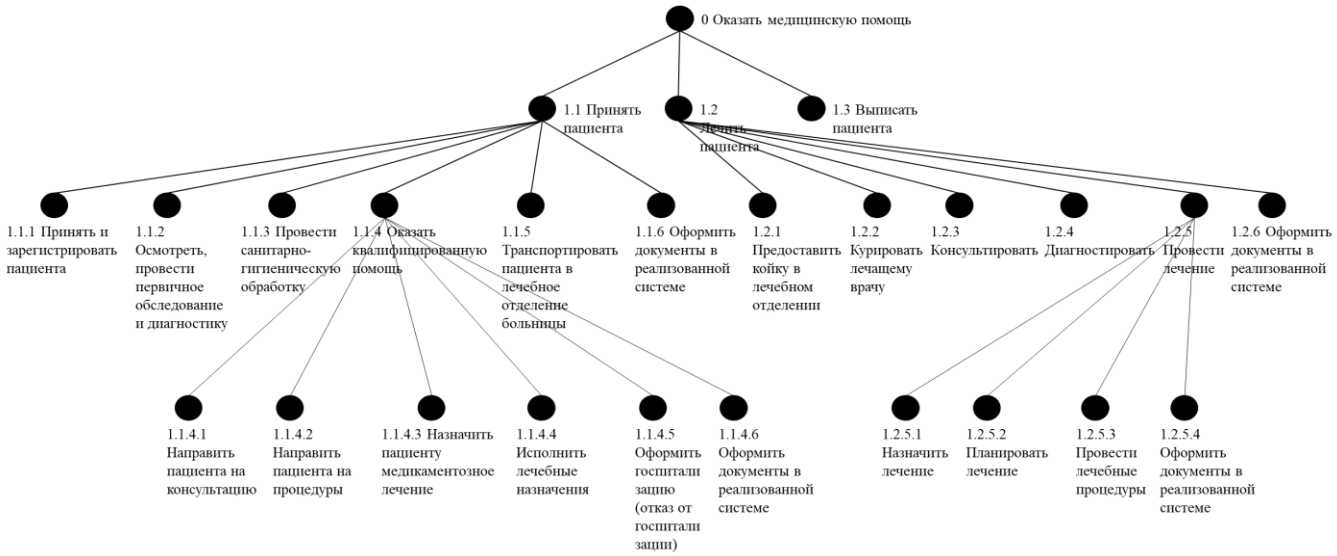


Рисунок 4.27 – *Карта процессов в модели «TO-BE»*

4.4.2. Проектирование пользовательских интерфейсов

В данном спринте добавляются подобные с описанными на прошлом этапе формы просмотра и редактирование данных из таблиц БД MySQL (общая схема приведена на рисунке 4.17). Однако для каждой новой формы появится новая схема взаимодействия с соответствующей таблицей. Для этого построено новое дополнение к схеме взаимодействия пользовательских экранов (рисунок 4.28).

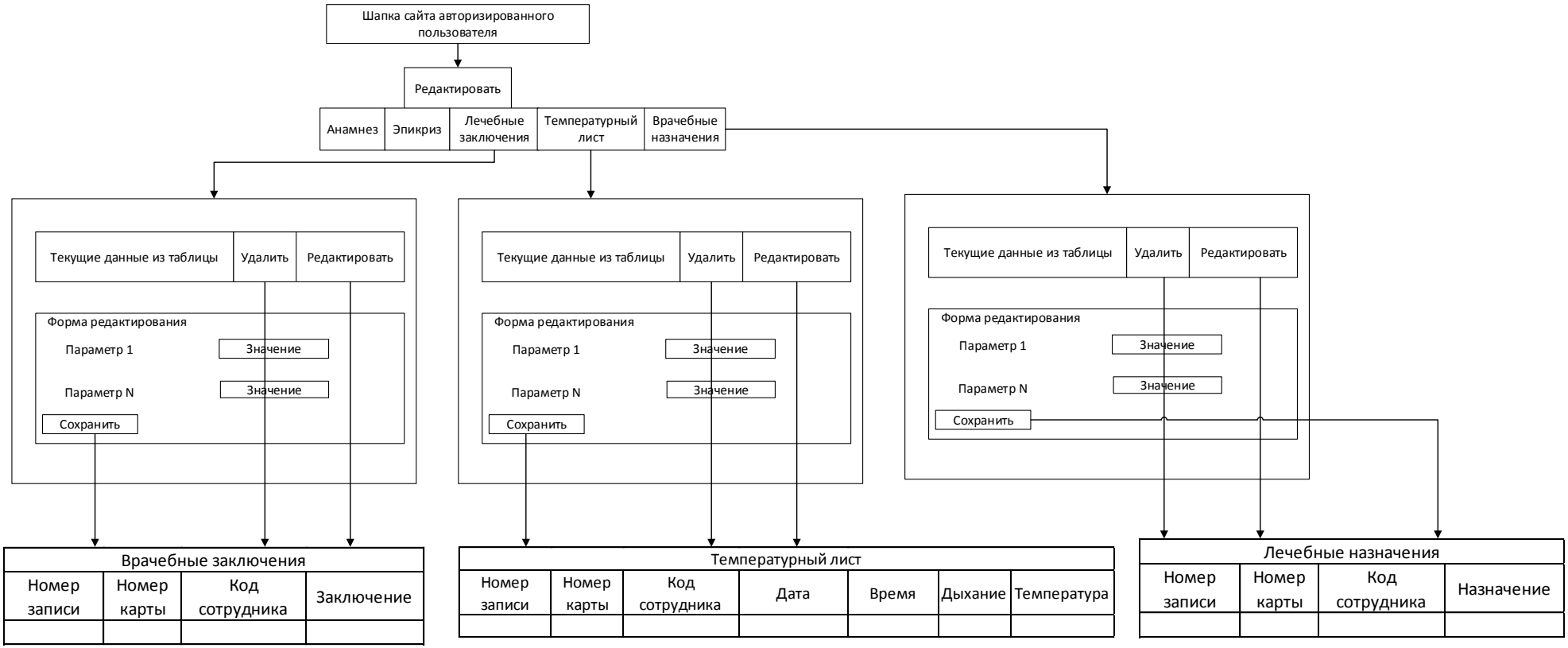
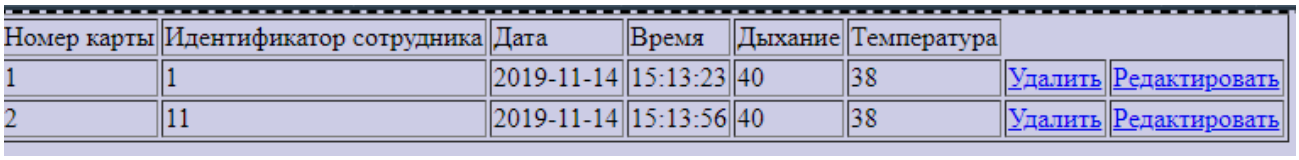


Рисунок 4.28 – *Обновленная схема взаимодействия экранов при редактировании таблиц*

4.4.3. Реализация средствами PHP

Результаты выполнения спринта представлены на рисунке 4.29. Формы редактирования и добавления в таблицы, связанные с пациентами, аналогичны представленной ранее (во втором спринте) форме.



Номер карты	Идентификатор сотрудника	Дата	Время	Дыхание	Температура		
1	1	2019-11-14	15:13:23	40	38	Удалить	Редактировать
2	11	2019-11-14	15:13:56	40	38	Удалить	Редактировать

Рисунок 4.29 – *Форма редактирования данных из таблицы «Температурный лист»*

4.4.4. Тестирование разработанной программы

Добавленные элементы кода, исправления в нем и веб-сайт снова функционально протестированы, а не пройденные тесты занесены в таблицу 4.6.

Таблица 4.6 – *Не пройденные функциональные тесты*

№ тест	Наименование теста	ME	YB	IE	GC	O	MF	Комментарии
23	Сайт доступен для поисковых машин	-	-	-	-	-	-	Локальный сайт
24	Веб-страница имеет точную карту сайта в формате XML и HTML	-	-	-	-	-	-	Карта сайта отсутствует

И последним проведено нагрузочное тестирование, аналогичное, проведенному ранее, но с добавлением в тест новых, разработанных в рамках Agile Scrum, веб-страниц. Результаты нагрузочного тестирования занесены в таблицу 4.7. Тестирование проводилось по аналогии с главой 3 по формулам 3.1 – 3.4.

Таблица 4.7 – *Результаты нагрузочного тестирования после автоматизации*

К-во польз.	t_1, с	t_2, с	t_3, с	t_4, с	t_5, с	$t_{ср.ариф}$, с	$t_{ср.квадр.}$, с	Δt, с	$t_{отк}$, с
1	0,22	0,21	0,21	0,21	0,21	0,212	0,004	0,005	0,212±0,005
10	0,23	0,23	0,23	0,23	0,23	0,23	0	0,005	0,230±0,005
25	0,23	0,23	0,23	0,27	0,23	0,238	0,016	0,008	0,238±0,008
50	0,29	0,31	0,29	0,34	0,28	0,302	0,021	0,01	0,302±0,010
100	0,61	0,64	0,63	0,59	0,61	0,616	0,017	0,009	0,616±0,009
250	1,8	1,83	2,09	1,83	1,75	1,86	0,119	0,051	1,860±0,051
500	4,82	4,38	4,29	4,25	4,5	4,448	0,205	0,087	4,448±0,087
1000	12,67	10,59	9,94	9,72	10,09	10,602	1,073	0,456	10,602±0,456
2500	18,82	17,36	17,45	17,11	17,93	17,734	0,604	0,257	17,734±0,257
5000	31,15	30,93	30,52	29,85	31,2	30,73	0,501	0,213	30,730±0,213

Для наглядного анализа полученных результатов тестирования на основе таблицы 4.7 была составлена диаграмма, представленная на рисунке 4.30.

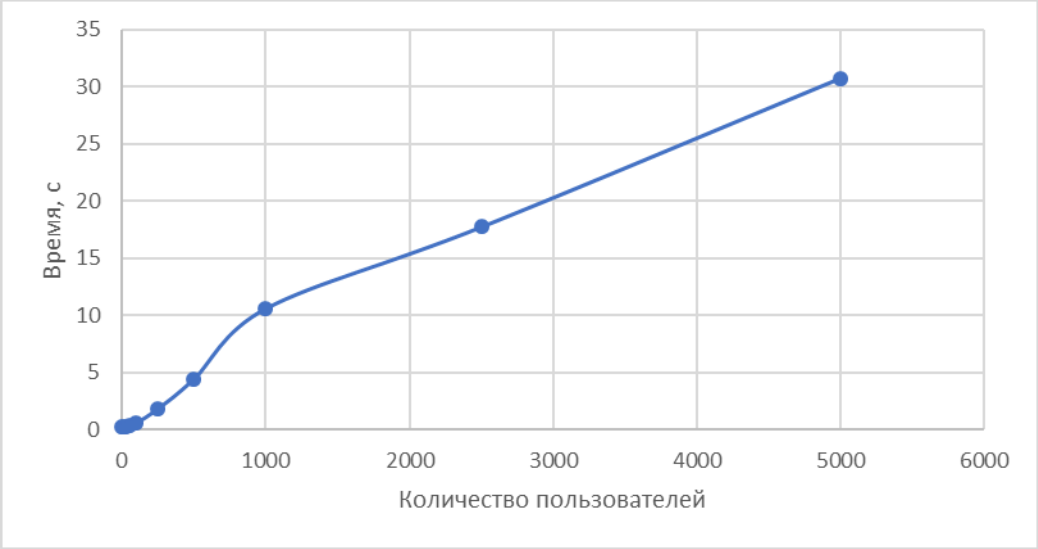


Рисунок 4.30 – *Зависимость среднеарифметического времени отклика от числа пользователей*

4.5. Результаты и их обсуждения

Подводя итоги необходимо отметить следующие моменты:

- Все поставленные задачи в рамках автоматизации ключевого бизнес-процесса «Принять пациента выполнены».
- Итерационная модель внедрения, в частности методология Agile Scrum, отлично подходят для разработки систем, в которых требования постоянно дополняются, либо инкременты аналогичны реализованным ранее функциям и необходимо внесение незначительных изменений.
- Использование Agile Scrum позволило снизить среднее значение ошибок за итерацию в функциональном тестировании по сравнению с ASAP.
- Результаты нагрузочного тестирования показали приемлемые (до 500-1000 единовременных пользователей), но все-таки ухудшенные результаты, по сравнению с предыдущим тестом, в связи с добавлением новых веб-страниц, которые осуществляют SQL-запросы к БД, что увеличивает время ожидания отклика сайта.
- Использование Agile Scrum позволило исправить имеющиеся недочеты, добавляя их в бэклог следующего спринта, не дожидаясь при этом завершения работ.

Раздел 5. Реализация бизнес-процесса «Выписать пациента» на основе спиралевидной модели с использованием методологии RAD

Разработка программного обеспечения будет производиться в последовательности, приведенной в таблице 5.1 в соответствие с представленным на рисунке 1.6 жизненным циклом [37].

Таблица 5.1 – *Этапы разработки ПО с использованием RAD*

№ действия \ Этап разработки	Начало	Первый таймбокс [38], [39] (реализация требований 1-7 бэклога)	Второй таймбокс (реализация требований 8-11 бэклога)	Третий таймбокс (реализация требований 12-15 бэклога)	Четвертый таймбокс (реализация требований 16-19 бэклога)
Планирование	Анализ требований (19 требований – таблица 5.1)	Планирование текущего цикла разработки	Планирование текущего цикла разработки	Планирование текущего цикла разработки	Планирование текущего цикла разработки
	Составление плана разработки по спиральной модели	Описание ключевого бизнес-процесса «Выписать пациента» в модели «AS-IS»	Описание ключевого бизнес-процесса «Выписать пациента» в модели «AS-IS»	Описание ключевого бизнес-процесса «Выписать пациента» в модели «TO-BE»	Описание ключевого бизнес-процесса «Выписать пациента» в модели «TO-BE»
		Анализ рисков на основе полученных требований	Анализ рисков на основе полученных требований	Анализ рисков на основе полученных требований	Анализ рисков на основе полученных требований
Пользовательское проектирование		Проектирование данных	Проектирование данных	Проектирование данных	Проектирование данных
		Проектирование пользовательских интерфейсов	Проектирование пользовательских интерфейсов	Проектирование пользовательских интерфейсов	Проектирование пользовательских интерфейсов
Конструирование		Реализация процессов средствами PHP	Реализация процессов средствами PHP	Реализация процессов средствами PHP	Реализация процессов средствами PHP

№ действия \ Этап разработки	Начало	Первый таймбокс [38], [39] (реализация требований 1-7 бэклога)	Второй таймбокс (реализация требований 8-11 бэклога)	Третий таймбокс (реализация требований 12-15 бэклога)	Четвертый таймбокс (реализация требований 16-19 бэклога)
		Тестирование	Тестирование	Тестирование	Тестирование
					Подготовка к релизу (исправление мелких недостатков, ошибок и т.д.)
					Тестирование конечного продукта
Переключение					Релиз конечного продукта

5.1. Бэклог продукта и витки спирали

Для того, чтобы понять полный объем проводимых работ, оставшиеся из таблицы 2.1 требования были вынесены в отдельный бэклог продукта (таблица 5.2), расставлены по убыванию приоритета (первоочередные – законодательные требования) и распределены по итерациям (виткам спирали).

Таблица 5.2 – *Перечень требований*

№	Пользовательское требование	Приоритет	Чекбокс
1	Правила подготовки к проведению диагностических манипуляций	Высокий	1
2	Правила диспансеризации и госпитализации	Высокий	1
3	Стоимость оказываемых медицинских услуг, с приложением утвержденного документа с ценами в электронном виде	Высокий	1
4	Конфиденциальность персональных данных	Высокий	1
5	Информация об органах охраны здоровья, надзору в сфере здравоохранения, надзору защиты прав потребителей (почтовый адрес, телефоны, электронный адрес)	Высокий	1

№	Пользовательское требование	Приоритет	Чекбокс
6	Информация об страховых учреждениях, с которыми заключены договора на оказание и оплату мед. услуг по ОМС	Высокий	1
7	Обязательно должна быть размещена информация о возможности проведения независимой оценки качества оказываемых услуг	Высокий	1
8	Возможность вывода данных о пациенте	Высокий	2
9	Доступность	Высокий	2
10	Легкость в использовании	Высокий	2
11	Работа системы в любом месте	Высокий	2
12	Карта сайта для работы с ресурсом и удобной навигации	Высокий	3
13	Версия сайта для слабовидящих людей	Высокий	3
14	Иные инструменты, обеспечивающие удобную работу с ресурсом пользователя	Высокий	3
15	Язык сайта (меню, карта сайта, информация на сайте) обязательно должен быть русским	Высокий	3
16	Возможность просмотра информации о персонале: ▪ Фамилия, Имя, Отчество специалиста; ▪ занимаемая должность; ▪ порядок записи к специалисту; ▪ график приема специалистом; ▪ данные сертификата специалиста – специальность, срок действия сертификата, соответствие занимающей должности; ▪ дополнительные данные сертификата – кем выдан, когда выдан, уровень образования, квалификация	Средний	4
17	Возможность печати карты при выписке	Средний	4
18	Удобный интерфейс на мобильных устройствах	Низкий	4
19	Графики приема врачей, контактные данные специалистов или организации – электронная почта, телефон	Средний	4

5.2. Первый таймбокс: реализация требований 1-7 бэклога

Реализация на данном витке посвящена выполнению требований законодательства, которые необходимы для предоставления справочной информации о медицинском учреждении.

5.2.1. Описание ключевого бизнес-процесса «Выписать пациента» в модели «AS-IS»

На рисунке 5.1 представлено описание данного бизнес-процесса. Как можно заметить, данный процесс является достаточно простым с точки зрения его организации, то есть он, по сути, состоит исключительно из работы с документами, связанными с выпиской. По причине несложности внутренних процессов и наглядности происходящих изменений уже на втором уровне не имеет смысла дальнейшее углубление в структуру бизнес-процесса «Выписать пациента».

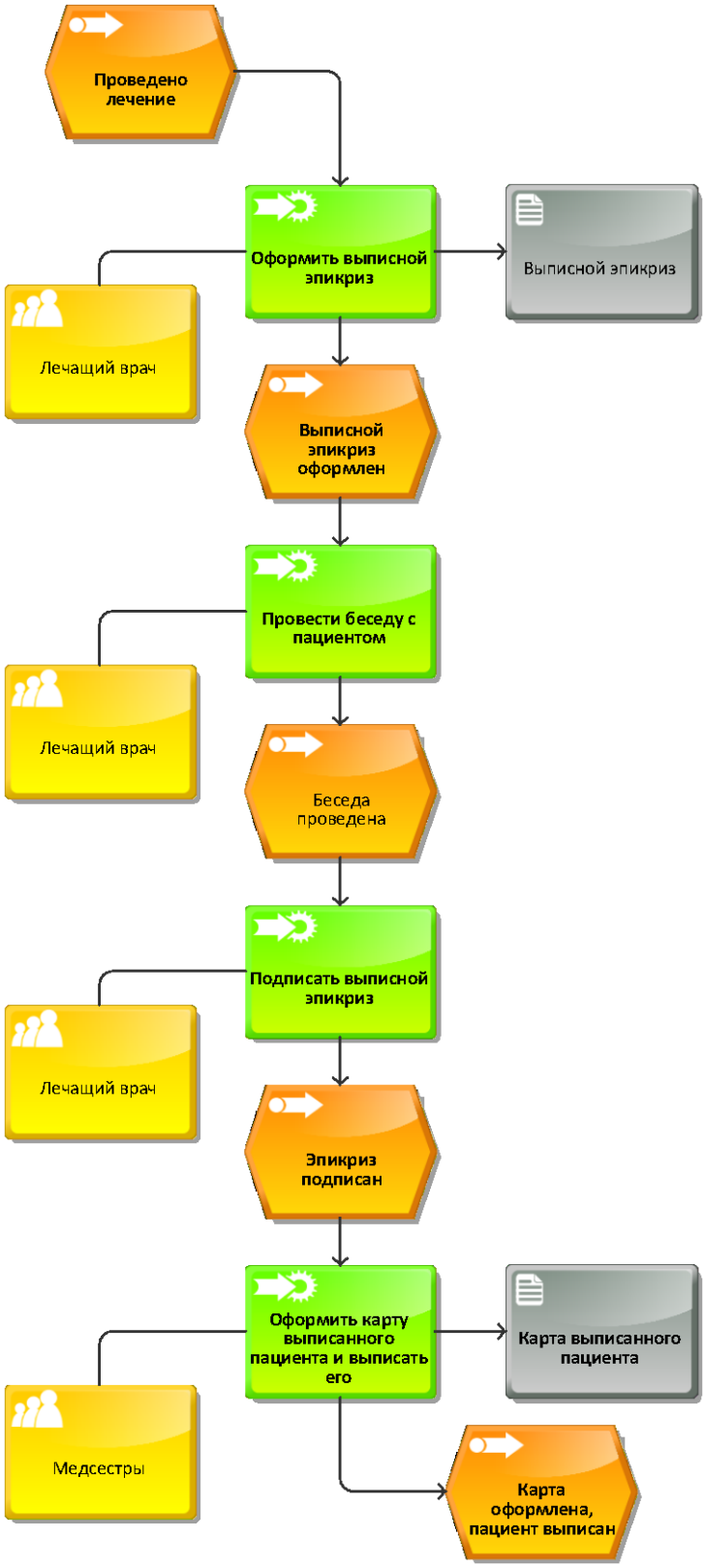


Рисунок 5.1 – *Процесс «Выписать пациента» в модели ARIS eEPC «AS-IS» (2 уровень)*

5.2.2. Оценка рисков

Оценка рисков производится согласно алгоритму, подробно описанному в главе 1. Поскольку все требования на данном витке посвящены оформлению веб-интерфейса с добавлением несложных описаний, то возможные риски можно объединить в один – невыполнение требований законодательства РФ. Вероятность возникновения данного риска минимальна – 1, однако невыполнение данных условий может повлечь проблемы с невыполнением требований законодательства. Следовательно, степень воздействия возможных рисков высока – 9. Для удобства отслеживания здесь и далее имеющиеся риски вносятся в таблицу отслеживания качественных рисков (таблица 5.3). Для оценки критичности риска в рамках работы будут рассмотрены только риски с рангом более 30, поскольку проект небольшой и рисков должно быть немного.

Таблица 5.3 – *Таблица отслеживания качественных рисков*

Требование	Описание риска	Вероятность (1…10)	Критичность (1…10)	Ранг (1…100)	Стратегия реагирования	Описание стратегии
Создание и оформление веб-страниц	Невыполнение требований законодательства РФ	1	9	9	Исключение	-
Наличие и доступность правил диспансеризации и госпитализации	Несоответствие требованиям законодательства и МинЗдрава	1	10	10	Исключение	-
Конфиденциальность персональных данных	Утечка персональных данных	4	9	36	Понижение вероятности	Шифрование данных в БД

Для первого риска проведем расчет RR по формуле 1.1 (формула 5.1) и внесем результат в 5 столбец таблицы 5.3. Здесь и далее расчет RR будет проводиться по приведенной формуле.

RR = Вероятность * Критичность. (5.1)

При этом изменений ни в архитектуру имеющихся данных, ни в пользовательские интерфейсы вносить не нужно – достаточно воспользоваться стандартными инструментами php и html для разметки необходимого текста на веб-странице и написание самой страницы. Также необходимо реализовать шифрование паролей для защищенного их хранения в таблице «Персонал».

5.2.3. Реализация требований

Для реализации требований добавлена страница svedenia.php, внешний вид которой представлен на рисунке 5.2, а ее листинг приведен в приложении А.

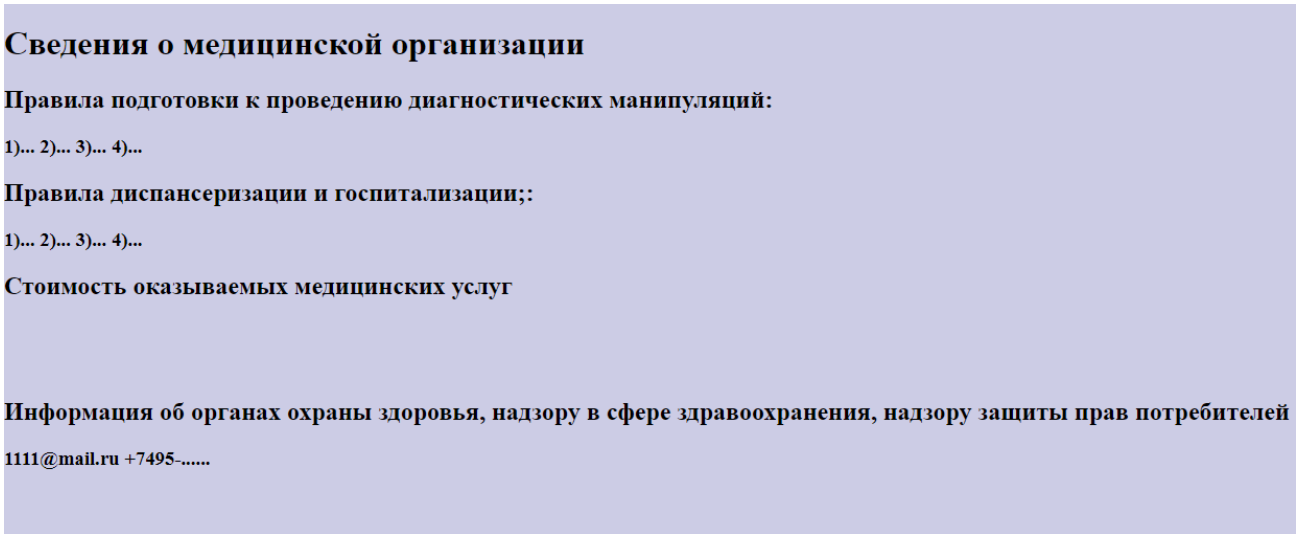


Рисунок 5.2 – *Информационный блок страницы сведений о медицинской организации*

На рисунке 5.3 приведен пример зашифрованного при помощи алгоритма md5 пароля, шифрование которым вставлено в страницу регистрации и авторизации персонала.

email	password
123@mail.ru	4828140403f6eaee3b5af62a0b09ae61

Рисунок 5.3 – *Фрагмент таблицы «Персонал» с зашифрованным паролем*

5.2.4. Тестирование разработанной программы

Как и для двух предыдущих глав после разработки проведено функциональное тестирование, результаты которого представлены в таблице 5.4:

Таблица 5.4 – *Не пройденные функциональные тесты*

№ теста	Наименование теста	ME	YB	IE	GC	O	MF	Комментарии
7	Ссылки, которые используются для отправки электронной почты (фидбек)	-	-	-	-	-	-	Нет формы обратной связи

5.3. Второй таймбокс: реализация требований 8-11 бэклога

Разработка на данном витке посвящена реализации функции поиска и выгрузки информации по карте пациента, а также общему функционалу сайта. И к бэклогу витка добавлены работы по устранению дефектов по тестам 9 и 22 таблицы 5.4.

5.3.1. Описание ключевого бизнес-процесса «Выписать пациента» в модели «AS-IS»

На основе данных до 2 уровня описания процесса выписать пациента и ранее реализованной карте процессов можно составить итоговую карту в модели «AS-IS» (рисунок 5.4).

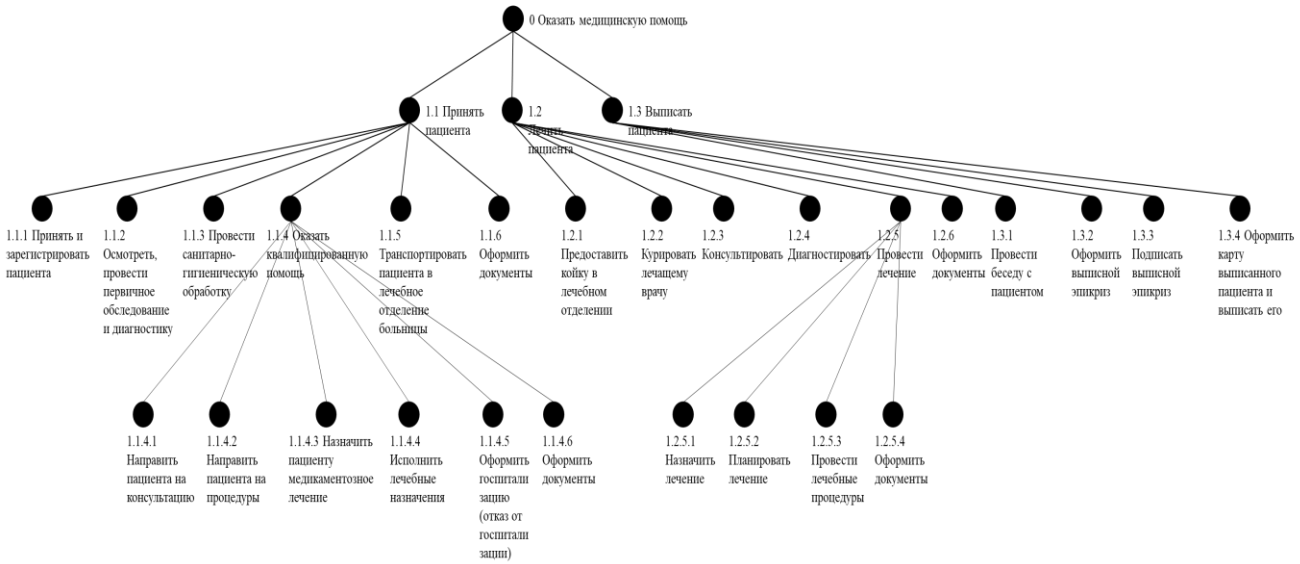


Рисунок 5.4 – *Карта процессов в модели «AS-IS»*

5.3.2. Оценка рисков

На данном этапе разработки имеется несколько рисков:

- Связан с формой поиска информации о пациенте – возможны некорректные запросы к БД MySQL и таблицам внутри, либо ошибка в коде самой страницы.
- С оставшимися на данном спринте требованиями – некорректное отображение веб-страниц. Они маловероятны по причине изначальной направленности при разработке сайта на гибкий функционал и удобные средства разработки и интерфейс.

В связи с высоким (но некритичным) уровнем приоритета требований, соответствующие им риски получат оценки 8 и 7 соответственно по шкале

воздействия. При этом у первого риска достаточно высокая вероятность (8) возникновения по причине ранее реализованных механизмов поиска и выгрузки информации по картам пациентов, а у второго риска – достаточно низкая (2). Риск, связанный с доступностью маловероятен (2), т.к. на тестовом сервере denwer он доступен для всей внутренней сети. При этом, при развертывании в продуктивной среде он станет доступен и извне, но при этом риск достаточно критичен в связи с необходимостью в предоставлении информации для пациентов (8). Имеющиеся риски внесены на таблицу отслеживания рисков, представленную в таблице 5.5. Расчет рангов рисков проведен по формуле 1.1 и результаты внесены в 5 столбец таблицы.

Таблица 5.5 – *Таблица отслеживания рисков*

Требование	Описание риска	Вероятность (1…10)	Критичность (1…10)	Ранг (1…100)	Стратегия реагирования	Описание стратегии
Форма поиска информации о пациенте	Некорректные запросы к БД MySQL и таблицам или ошибка в коде	8	8	64	Понижение вероятности	Необходимо организовать единую форму запроса к БД MySQL на основе уже созданных запросов
Удобство функционала и оформления	Некорректное отображение веб-страниц	2	7	14	Исключение	-
Доступность	Отсутствие возможности доступа к сайту извне сети	3	8	24	Исключение	-

Изменения в архитектуру данных в рамках данного витка спирали внесены не будут, а в пользовательский интерфейс будет добавлена форма поиска информации о пациенте, уже описанная ранее.

5.3.3. Проектирование пользовательских интерфейсов

Отличием данной формы поиска будет то, что поиск в данном случае будет осуществляться неавторизованным пользователем (пациентом), поэтому поиск будет возможен только при указании нескольких параметров, привязанных к одному пациенту в соответствующей таблице – в противном случае карта не будет выведена. Отсюда следует проект формы поиска (рисунок 5.5). Также для внесения понимания процессов обращения к таблицам БД опишем процедуру 2 с рисунка 3.14.

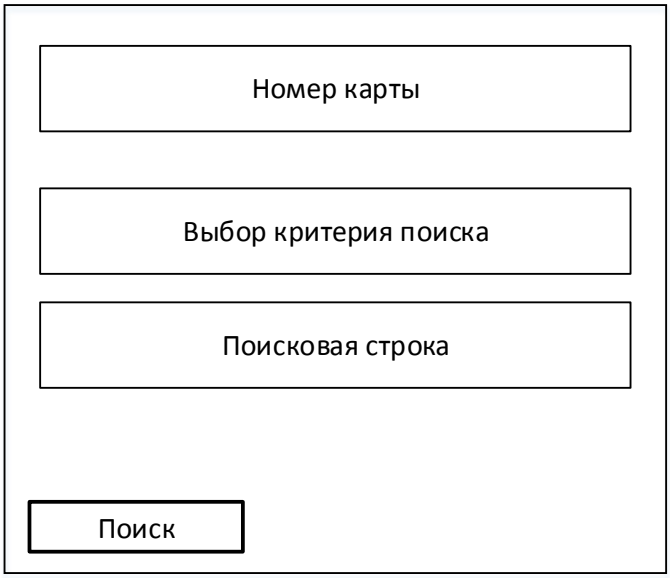


Рисунок 5.5 – Форма поиска карты для пациента

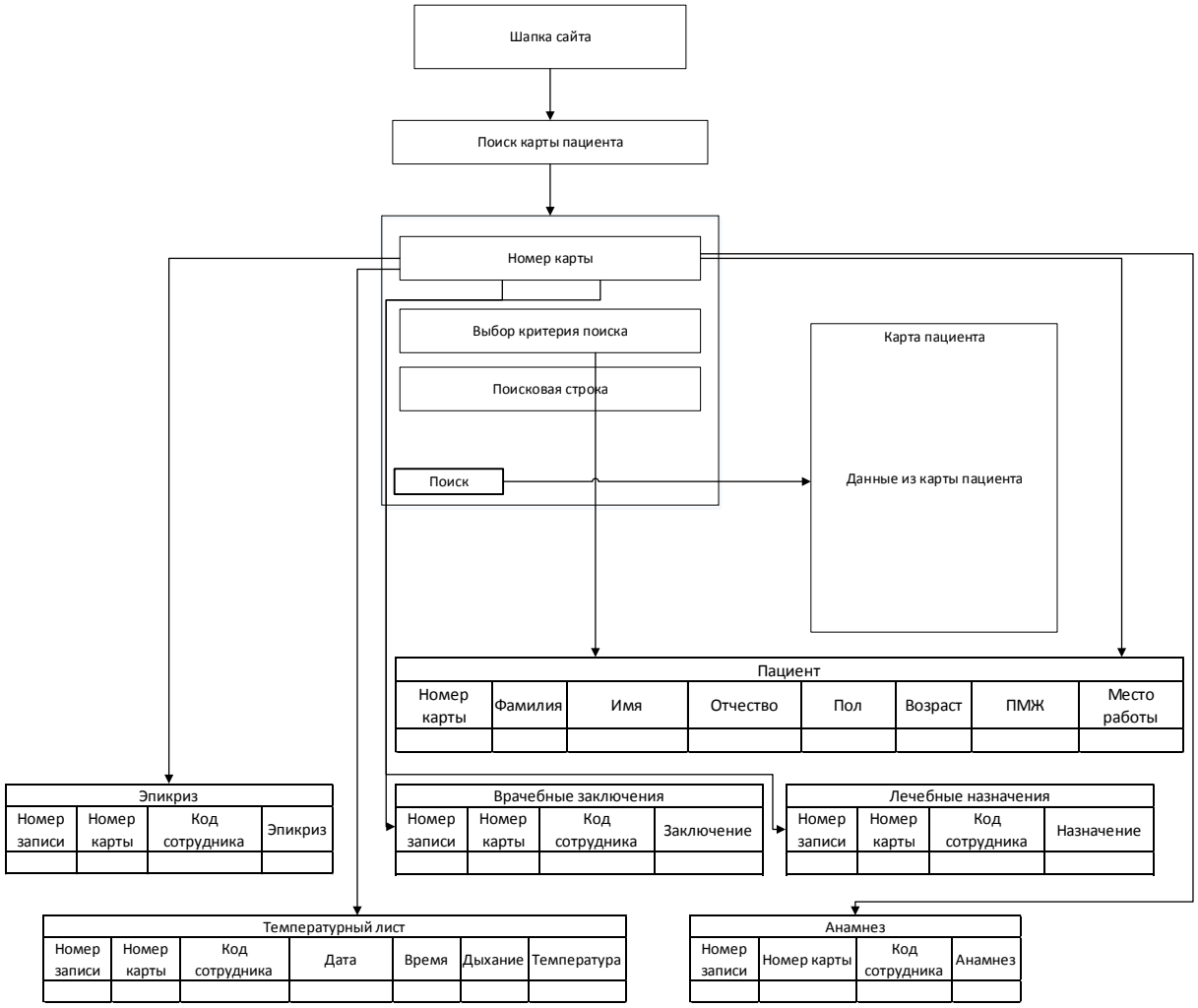


Рисунок 5.6 – *Схема взаимодействия пользовательских экранов при поиске карты пациентом*

5.3.4. Реализация требований

Для реализации требований необходимо создание веб-формы поиска, вывода на экран и на печать медицинской карты (рисунок 5.7) пациента. Для вывода медицинской карты будут использованы формы и методы, аналогичные представленным ранее в главе 4, но урезанные только до номера карты, для получения данных самим пациентом. Для исключения рисков реализована страница dbconnect.php для вызова и подключения к БД MySQL (листинг приведен в приложении А).

Рисунок 5.7 – *Результат поиска карты пациента*

5.3.5. Тестирование разработанной программы

После разработки вновь проведено функциональное тестирование, результаты которого представлены в таблице 5.6:

Таблица 5.6 – *Не пройдённые функциональные тесты*

№ теста	Наименование теста	ME	YB	IE	GC	O	MF	Комментарии
9	Отсутствие неработающих ссылок	-	-	-	-	-	-	Не сработала ссылка на формирование письма

5.4. Третий таймбокс: реализация требований 12-19 бэклога

Данный виток посвящен оформлению веб-сайта и реализации карты сайта на языке XML. Также на данном витке необходимо выполнить требования законодательства, которые необходимы для предоставления

информации о сотрудниках и о веб-сайте, а также общему функционалу сайта.

5.4.1. Описание ключевого бизнес-процесса «Выписать пациента» в модели «TO-BE»

На рисунке 5.8 представлено описание данного бизнес-процесса в модели «TO-BE».

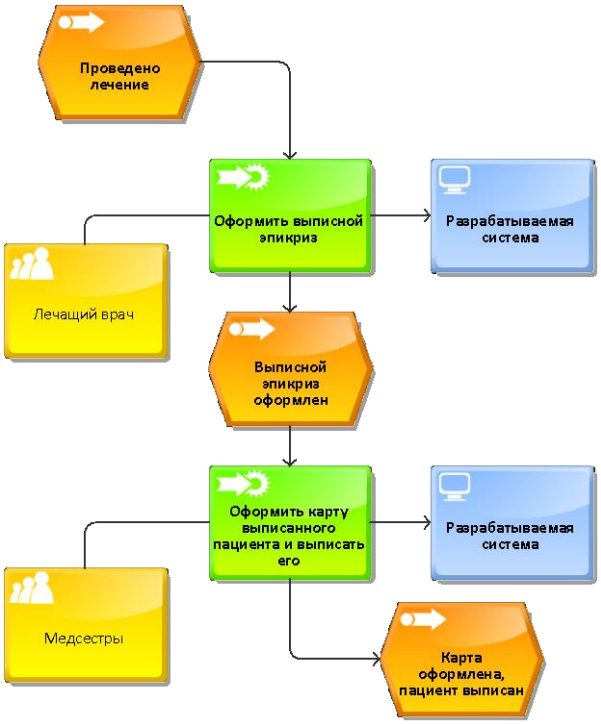


Рисунок 5.8 – *Процесс «Выписать пациента» в модели ARIS eEPC «TO-BE» (2 уровень)*

На рисунке 5.9 представлена карта процессов в модели «TO-BE» с учетом всех вносимых в организацию изменений.

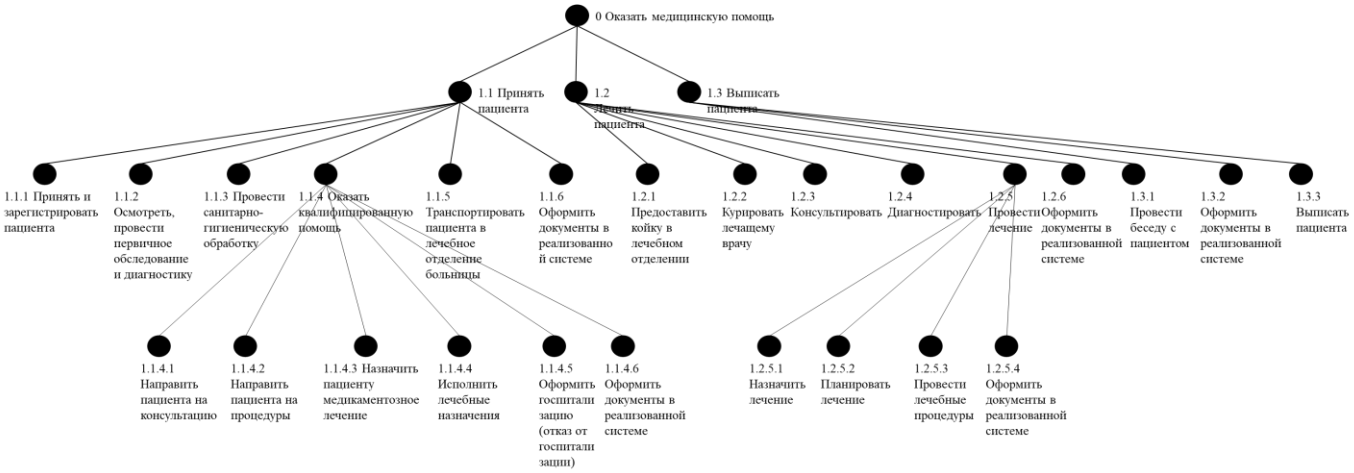


Рисунок 5.9 – *Карта процессов в модели «TO-BE»*

5.4.2. Оценка рисков

На данном этапе разработки имеется несколько рисков:

- Некорректное оформление сайта – связано с тремя требованиями.
- Некорректное оформление карты сайта – нерабочие ссылки, ошибки синтаксиса и др.
- Некорректное языковое оформление – отсутствие русифицированного интерфейса.
- Связан с выводом информации о сотрудниках учреждения – возможны некорректные запросы к БД MySQL и таблице внутри, либо ошибка в коде самой страницы, что повлечет некорректный вывод данных.
- Далее риск, связанный с формой печати – возможность некорректного вывода данных.
- С оставшимися на данном спринте требованиями – несоответствия требованиям удобства функционала. Они маловероятны по причине изначальной направленности при разработке сайта на гибкий функционал и удобные средства разработки и интерфейс.

Соответствующие требованиям риски 1-3 получат оценки уровня воздействия 9, 3 и 1 соответственно. При этом, у первого риска достаточно низкая вероятность (2) возникновения по причине ранее реализованных (на 1 витке) механизмов поиска и выгрузки информации, у второго риска – средняя (6), а у третьего – низкая – 2, по причине уже реализованных в целом имеющихся требований. Имеющиеся риски внесем на карту сортировки рисков, представленную в таблице 5.9.

Соответствующие требованиям риски 3-5 получат оценки уровня воздействия 9 и 9 соответственно по причине требований законодательства. При этом, у первого риска достаточно низкая вероятность (3) возникновения по причине ранее реализованного практически полностью веб-интерфейса, соответствующего законодательным требованиям, а второго риска – достаточно низкая (4), в связи с имеющейся схемой взаимодействия интерфейсов, описанной в четвертой главе. Имеющиеся данные по рискам внесены в таблицу 5.7, а расчет ранга проведен по формуле 1.1.

Таблица 5.7 – *Таблица отслеживания рисков*

Требование	Описание риска	Вероятность (1…10)	Критичность (1…10)	Ранг (1…100)	Стратегия реагирования	Описание стратегии
Оформление сайта	Непонятность сайта для людей с ограниченными возможностями	3	9	27	Исключение	-
Оформление карты сайта	Нерабочие ссылки, ошибки синтаксиса и пр.	4	9	36	Понижение вероятности	Для исключения ошибок применить ранее использованное ПО Xenu для формирования перечня страниц для карты

Требование	Описание риска	Вероятность (1…10)	Критичность (1…10)	Ранг (1…100)	Стратегия реагирования	Описание стратегии
Языковое оформление сайта	Непонятность сайта большинству пользователей	1	8	8	Исключение	-
Вывод информации о сотрудниках	Ошибки в запросе либо в коде самой страницы	2	9	18	Исключение	-
Форма печати карты пациента	Некорректный вывод данных	6	5	30	Понижение вероятности	Использовать для вывода на печать ранее реализованную форму вывода карты пациента
Функционал сайта	Несоответствие требованиям качества предоставления услуг	2	8	16	Исключение	-

При этом изменений в архитектуре данных не будет, но будет добавлен новый интерфейс – страница, наполненная картой сайта на языке XML.

5.4.3. Проектирование пользовательских интерфейсов

В рамках данного спринта планируется добавить карту сайта, добавляемую в подвал сайта на всех представленных ранее проектах интерфейсов. Ссылка на карту будет добавлена в подвал и взаимодействие будет реализовано по схеме на рисунке 5.10.

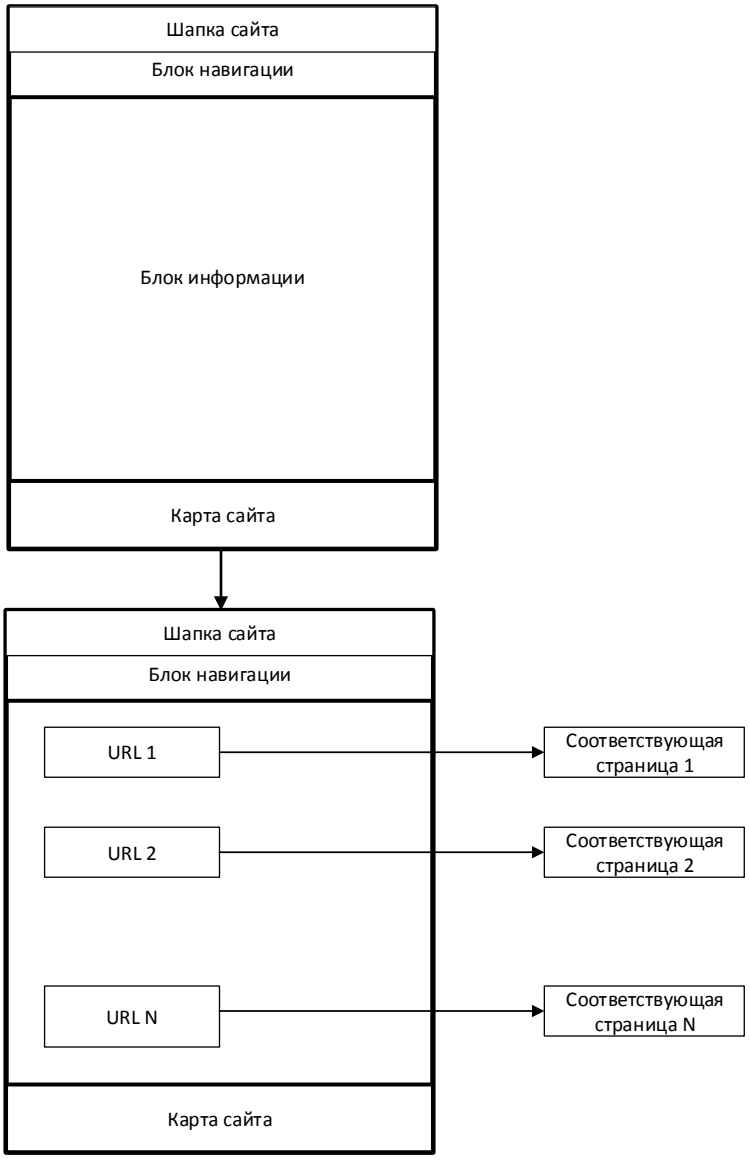


Рисунок 5.10 – *Схема взаимодействия пользовательских экранов при работе с картой сайта*

Также будет добавлен новый интерфейс – таблица, заполненная информацией о сотрудниках больницы. В формы с результатами поиска карты пациента будет добавлена кнопка печати. Взаимодействие при печати (для пациента) будет осуществляться по схеме, представленной на рисунке 5.11.

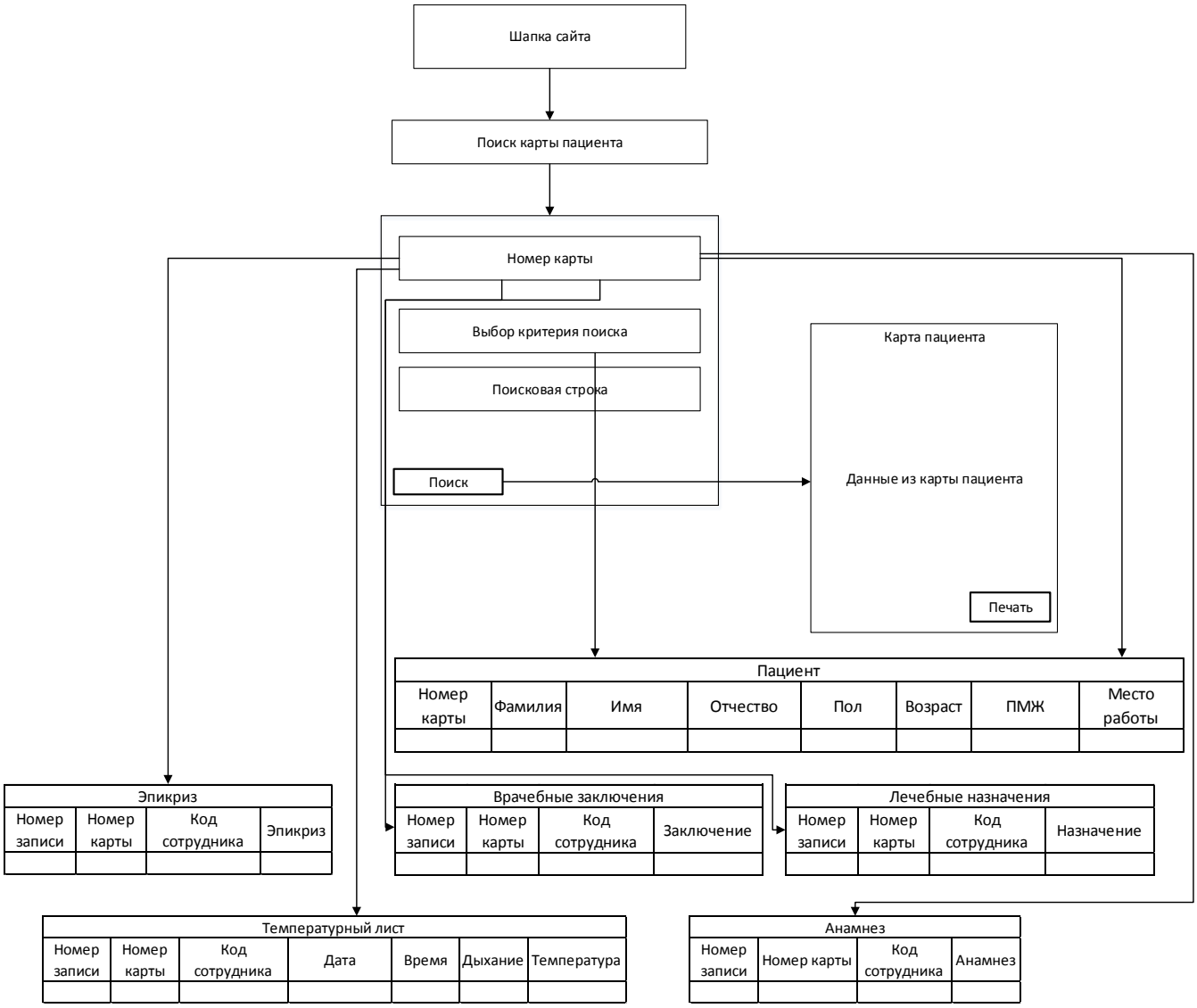


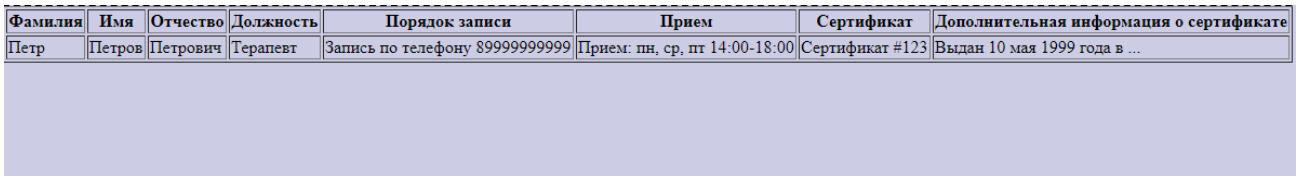
Рисунок 5.11 – *Схема взаимодействия (для пациента) пользовательских экранов при поиске или печати карты пациента*

5.4.4. Реализация требований

Список URL-адресов сайта был отсканирован при помощи Xenu и внесен в текстовый файл. Далее для реализации требований добавлена страница map.php с гипертекстовой разметкой выявленных адресов. Внешний вид страницы представлен на рисунке 5.12, а ее листинг приведен в приложении А.

Рисунок 5.12 – *Карта разработанного сайта*

Для реализации оставшихся требований добавлена страница staff.php, внешний вид которой представлен на рисунке 5.13, а ее листинг приведен в приложении А. На страницу контактной информации добавлен e-mail и телефон для обратной связи.



Фамилия	Имя	Отчество	Должность	Порядок записи	Прием	Сертификат	Дополнительная информация о сертификате
Петр	Петров	Петрович	Терапевт	Запись по телефону 89999999999	Прием: пн, ср, пт 14:00-18:00	Сертификат #123	Выдан 10 мая 1999 года в ...

Рисунок 5.13 – Вид страницы с информацией о персонале

5.4.5. Тестирование разработанной программы

Требование 23 удалено в связи с тем, что проведена ручная индексация сайта для реализации карты. В случае необходимости можно добавить ее в поисковые системы. Для оценки успешности реализации вновь проведено функциональное тестирование, в результате которого все тесты были пройдены: исправлены ошибки в html-коде, добавлены данные для обратной связи в контактную информацию. Также для проверки

реализованных разработок вновь проведено нагрузочное тестирование, расчеты которого проводились по формулам 3.1 – 3.4, а результаты внесены в таблицу 5.8.

Таблица 5.8 – Результаты нагрузочного тестирования

К-во польз.	t_1, с	t_2, с	t_3, с	t_4, с	t_5, с	$t_{ср.ариф}$, с	$t_{ср.квадр.}$, с	Δt, с	$t_{отк}$, с
1	0,11	0,15	0,07	0,13	0,17	0,126	0,03847	0,01635	0,126±0,016
10	0,1	0,11	0,13	0,07	0,11	0,104	0,02191	0,00932	0,104±0,009
25	0,12	0,15	0,1	0,08	0,11	0,112	0,02588	0,01101	0,112±0,011
50	0,22	0,2	0,15	0,18	0,15	0,18	0,03082	0,0131	0,18±0,013
100	0,17	0,19	0,18	0,17	0,19	0,18	0,01	0,00428	0,18±0,004
250	0,15	0,15	0,19	0,13	0,15	0,154	0,02191	0,00932	0,154±0,009
500	0,19	0,18	0,11	0,15	0,16	0,158	0,03114	0,01324	0,158±0,013
1000	0,58	0,6	0,53	0,72	0,56	0,598	0,07294	0,03099	0,598±0,031
2500	1,12	1,41	1,03	1,85	1,03	1,288	0,35074	0,14901	1,288±0,149
5000	5,11	5,13	5,91	5,55	5,17	5,374	0,34969	0,14857	5,374±0,149

Для наглядного анализа полученных результатов тестирования на основе таблицы 5.8 была составлена диаграмма, представленная на рисунке 5.14.

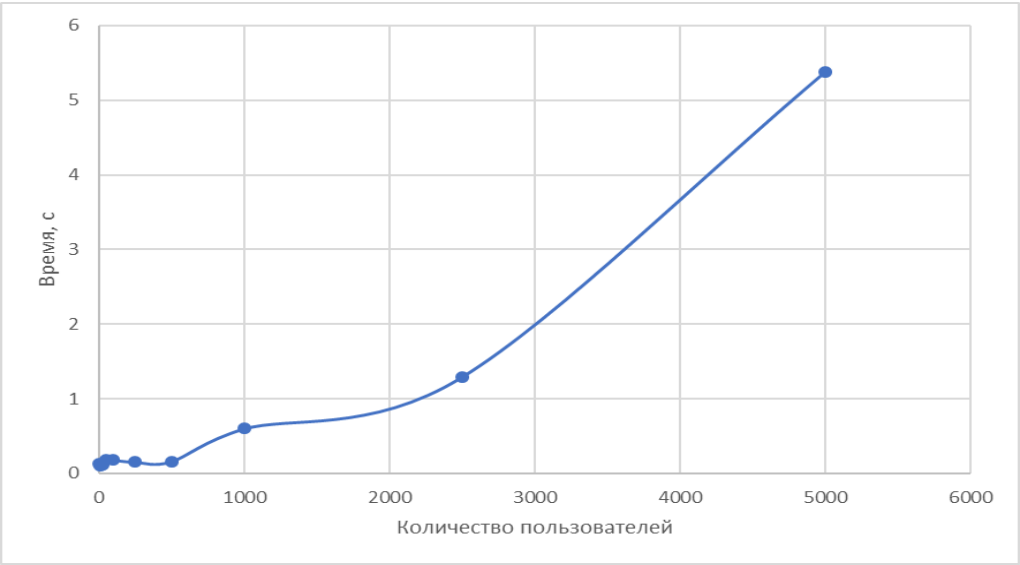


Рисунок 5.14 – *Зависимость среднеарифметического времени отклика от числа пользователей*

5.5. Результаты и их обсуждения

В рамках данной главы составлен план разработки приложения согласно методологии RAD. В соответствии с приведенным планом сформулирован бэклог продукта и витков спирали. В каждом витке было проведено проектирование автоматизируемого бизнес-процесса, проектирование данных и пользовательских интерфейсов, разработка и функциональное тестирование. В конце разработки проведено нагрузочное тестирование, вновь показавшее удовлетворительные результаты для небольшой городской больницы, и замечен рост производительности, наиболее вероятно связанный с добавлением общей страницы для запроса к БД MySQL.

Наличие нескольких витков позволило проводить усовершенствования разработки, не дожидаясь окончания работ. При этом время между каждым этапом существенно сократилось, и на каждом витке был реализован готовый к работе прототип.

Раздел 6. Сравнение методологий внедрения

После полной реализации системы и автоматизации бизнес-процессов, необходимо провести сравнение проведенных работ и полученных после них результатов.

6.1. Качественный и количественный анализ методологий внедрения

Для того, чтобы проанализировать результаты исследования, необходимо определить последовательность действий при анализе. Для этого воспользуемся качественным и количественным методами. На рисунке 6.1 показано, какие действия необходимо предпринять после сбора данных.

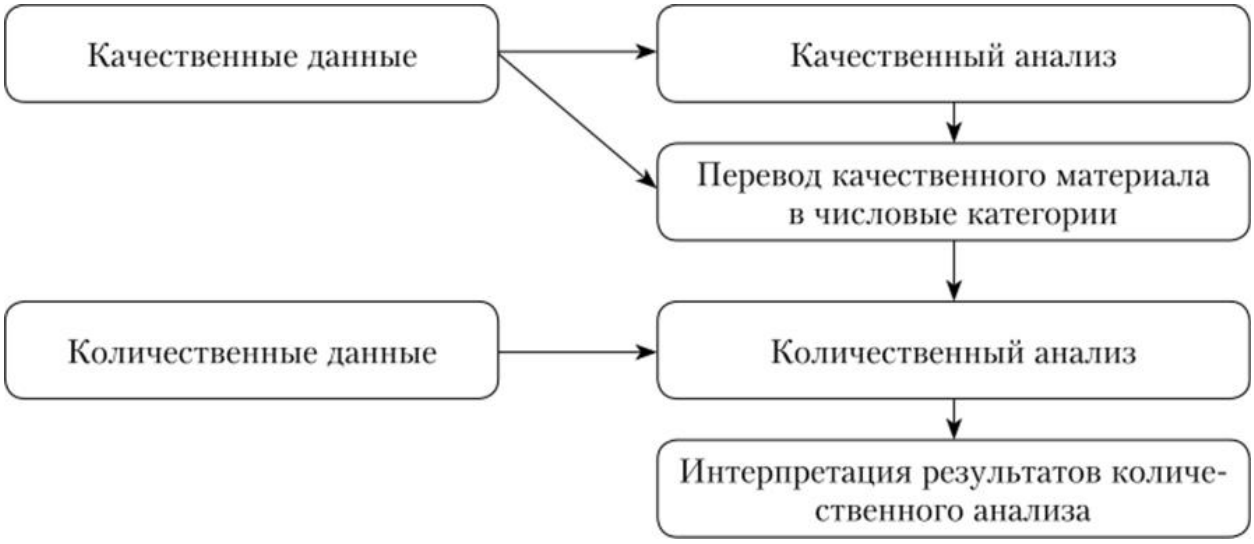


Рисунок 6.1 – *Обработка качественных и количественных данных*

Качественный и количественный анализ – два фундаментальных метода сбора и интерпретации данных в исследованиях. Эти методы могут использоваться независимо или одновременно, поскольку все они имеют одни и те же цели. У них есть некоторые ошибки, и поэтому одновременное использование их может компенсировать ошибки, которые каждый из них имеет, а затем дает качественные результаты.

6.1.1. Количественный анализ

Количественный анализ часто связан с численным анализом, когда данные собираются, классифицируются, а затем вычисляются для определенных результатов с использованием набора статистических методов. Данные выбираются случайным образом в больших образцах и затем анализируются. Преимущество количественного анализа результатов может быть применено в общей популяции с использованием моделей исследований, разработанных в образце. Это является недостатком качественного анализа данных из-за ограниченного обобщения результатов.

Количественный анализ носит более объективный характер. Он стремится понять возникновение событий, а затем описать их с использованием статистических методов. Однако большая ясность может быть получена путем одновременного использования качественных и количественных методов. Количественный анализ обычно оставляет случайные и скудные события в результатах исследований, тогда как качественный анализ их рассматривает.

Количественный анализ обычно касается измеряемых величин, таких как вес, длина, температура, скорость, ширина и многое другое. Данные могут быть выражены в табличной форме или в любом графическом представлении с использованием графиков или диаграмм. Количественные данные можно классифицировать как непрерывные или дискретные и их часто получают с помощью обследований, наблюдений, экспериментов или интервью.

Однако в количественном анализе имеются ограничения. Например, может быть сложно выявить относительно новые концепции, используя количественный анализ и именно там, где качественный анализ входит в уравнение, чтобы выяснить «почему» возникает определенное явление. Вот почему эти методы часто используются одновременно.

6.1.2. Качественный анализ

Качественный анализ связан с анализом данных, которые не могут быть количественно определены. Этот тип данных касается понимания и понимания свойств и атрибутов объектов. Качественный анализ может получить более глубокое понимание «почему» происходит определенное явление. Анализ может использоваться в сочетании с количественным анализом или предшествующим ему.

В отличие от количественного анализа, который ограничен определенными правилами или цифрами классификации, анализ качественных данных может быть широким и многогранным. И это субъективно, описательно, нестатистическое и исследовательское по своей природе.

Поскольку качественный анализ стремится получить более глубокое понимание, исследователь должен быть хорошо округлен с учетом каких-либо физических свойств или атрибутов, на которых основано исследование. В количественном анализе характеристики объектов часто не раскрываются.

В качественном анализе есть ограничения. Например, он не может быть использован для обобщения населения. Небольшие образцы используются в неструктурированном подходе и они не являются репрезентативными для общей популяции, поэтому этот метод не может быть использован для обобщения всей популяции.

6.2. Качественный анализ моделей внедрения ИС

Чтобы провести качественную оценку необходимо оценить проведенные в рамках работы действия и провести между ними сравнение. В рамках 3 главы данной работы проведена автоматизация ключевого бизнес-процесса «Лечить пациента» с использованием методологии ASAP. Однако, определение целей и глубины разработки – в рамках этапа «Подготовка

проекта» – осуществлено во введении, далее, часть этого этапа осуществлена во 2 главе – описание бизнес-процессов и составление их карты.

Для минимизации возможных потерь в виде времени и других ресурсов, необходима тщательная подготовка для единственной итерации разработки – прохода по «Маршрутной карте». В рамках данной части работы к реализации принято 19 требований, большая часть из которых связана с организацией работы БД и веб-сайта. Затем необходимо определить классы и типы планируемых к использованию данных и сформировать их проекты. Основываясь на них можно спроектировать необходимые взаимосвязи между интерфейсами и БД. Также описан ключевой бизнес-процесс в модели «TO-BE» до 3 уровня и составлена соответствующая описанию карта в аналогичной модели. После этого за 1 этап разработки реализовано несколько веб-страниц, БД и часть таблиц в ней, а затем проведены функциональное и нагрузочное тестирование. Полный цикл разработки занял 25 дней. На основе схемы методологии составлен план разработки на 3 спринта длительностью по 7 дней.

На следующем шаге необходимо было составить бэклог продукта и спринтов, для этого была взята часть требований из общего списка, описанного во второй главе, на основе этих требований были сформулированы пользовательские истории и затем отсортированы по приоритетам и далее по спринтам. В рамках каждого из спринтов проводились действия, аналогичные описанным и реализованным в разработке с помощью ASAP, однако этапы проектирования, разработки и тестирования проводятся на каждом спринте. К описаниям и картам процессов добавлены новые элементы в соответствии с описанием бизнес-процесса «Принять пациента» до 3 уровня. Итерационный подход позволяет снизить число ошибок на последующих этапах за счет добавления пункта об устранении дефекта в бэклог следующего спринта. Также стоит отметить,

что большая часть сложностей связана с распределением обязанностей – если при реализации с помощью ASAP сроки были достаточными для поэтапной разработки, то здесь возникали сложности с достаточно быстрым переходом от проектирования к разработке и тестированию, а затем обратно.

При этом создание доработок существенно не влияет на работоспособность системы и позволяет реализовывать рабочую версию, готовую к эксплуатации уже после первого спринта, а также после любого другого. Отсюда можно заключить, что спринты могут продолжаться до момента реализации или исправления всех необходимых в работе функций и объем проекта может быть значительно больше, точно так же, как и продолжительность этих спринтов.

После реализации бэклога спринтов в Scrum и части возникших ошибок при разработке, начаты разработки при помощи методологии RAD. Составлен план на 3 таймбокса длительностью 6 дней и сформирован бэклог из оставшихся от общего списка требований и также отсортированы по приоритетам, а затем по таймбоксам.

Структура разработки схожа со структурой при использовании Scrum, однако, здесь появилась дополнительная стадия анализа и обработки негативных рисков на этапе проектирования, в следствие чего стало возможным превентивно реагировать на возможные недочеты и ошибки. Основной проблемой при обработке рисков было определение пороговой величины в связи с тем, что технологии разработки были отработаны на предыдущих этапах, а поэтому большинство рисков стало маловероятными. Исходя из исследованных методических указаний по обработке рисков можно сказать, что в обработку на проектах принимаются риски с рангом выше или равным 81, а такой показатель наиболее вероятно был недостижим по причине, описанной выше и невысокому числу возможных рисков из-за небольшого объема работ над системой. Отсюда последовало логичное

решение – снизить порог реагирования на риски до ранга => 30. Также сократилось время работы над каждой итерацией, но появилась возможность параллельно проводить проектирование и разработку.

В качестве CASE-средства на данном этапе разработки выступило ПО Xenu для автоматического проведения индексации реализованных веб-страниц с целью создания подробной html и xml карты сайта. Полученный опыт разработки структурирован и занесен в таблицу 6.3.

6.3. Количественный анализ каскадной итерационной и спиралевидной модели внедрения ИС

На основе проведенных после каждого этапа разработки нагрузочных тестов, их результатов и составленных по ним графикам зависимости, составлен итоговый график всей разработки, приведенный на рисунке 6.2. Оценивая результаты проведенных тестов можно заключить, что время отклика допустимо низко для единовременного использования страниц 500 пользователями единовременно, что соответствует потребностям небольшой больницы. При этом тесты проводились на локальном ПК и время отклика при переходе в продуктивную среду, то есть на сервер, снизится для всех исследованных объемов пользователей, и вырастет пропускная способность сайта. На основе проведенных функциональных тестов составлена таблица 6.1 для удобства отслеживания возникшего числа ошибок на каждой стадии разработки, а на основе приведенной ниже таблицы составлен накопительный график, приведенный на рисунке 6.3.

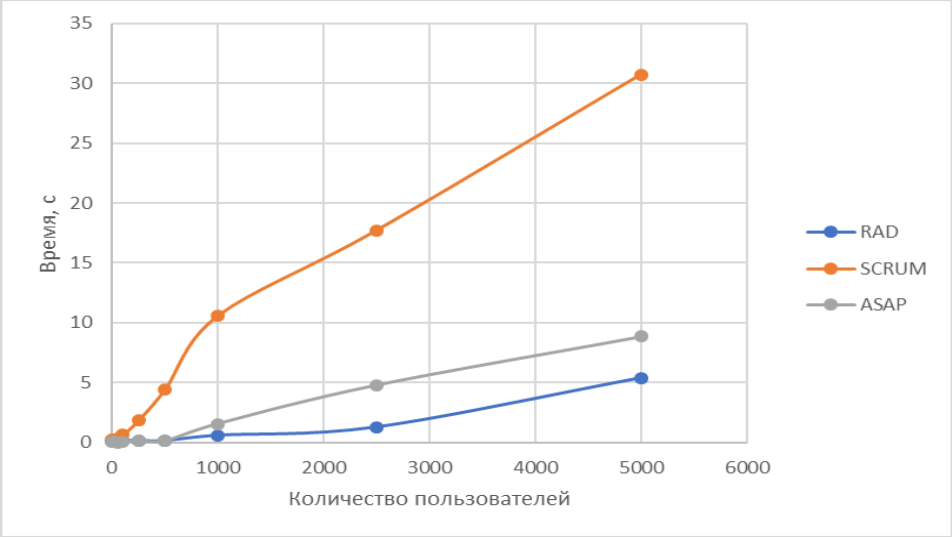


Рисунок 6.2 – *Сравнение результатов нагрузочных тестов для использованных методологий*

Таблица 6.1 – *Сравнение результатов функционального тестирования*

Методология	**RAD**			**Scrum**			**ASAP**
Число ошибок	2	1	0	2	2	2	8
Этап	1	2	3	1	2	3	1

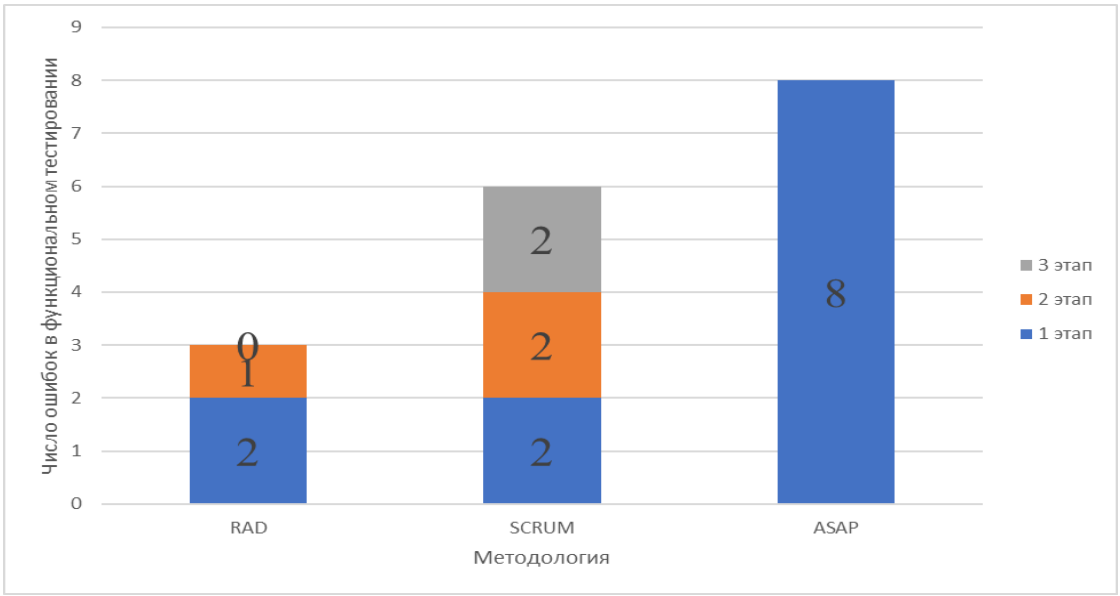


Рисунок 6.3 – *Накопительная диаграмма ошибок в функциональном тестировании*

6.4. Результаты и выводы

После реализованной автоматизации трех ключевых бизнес-процессов городской больницы (по 1 процессу на каждую методологию внедрения ИС) и проведенного качественного и количественного анализа, все данные будут занесены в таблицу 6.2, схожую с ранее приведенной таблицей 1.1, с добавлением критериев, суммирующих описанные выше результаты и описывающие прикладную, а не теоретическую сторону процесса разработки.

Таблица 6.2 – *Сравнение методологий внедрения ИС*

<table>
<tr><th rowspan="2">Характеристика проекта</th><th colspan="3">Прикладная методология</th></tr>
<tr><th>ASAP</th><th>Agile Scrum</th><th>RAD</th></tr>
<tr><td rowspan="2">Новизна разработки и обеспеченность ресурсами</td><td colspan="3">Типовой. Хорошо проработаны технология и методы решения задачи</td></tr>
<tr><td>Ресурсов достаточно по причине проработки процесса разработки со стороны SAP</td><td>Достаточное число ресурсов в виду адаптивности методологии под различные сроки реализации системы</td><td>Недостаток ресурсов в области анализа рисков</td></tr>
<tr><td>Масштаб проекта</td><td>Средние и крупные проекты</td><td>Любые проекты</td><td>Любые проекты</td></tr>
<tr><td>Сроки выполнения</td><td colspan="2">От нескольких недель до многих лет в зависимости от масштаба проекта</td><td>От 2 до 6 месяцев</td></tr>
<tr><td>Заключение отдельных договоров на отдельные версии</td><td>Заключается один договор. Версия и есть итоговый результат проекта</td><td colspan="2">На отдельную версию или несколько последовательных версий обычно заключается отдельный договор</td></tr>
</table>

Характеристика проекта	Прикладная методология		
	ASAP	Agile Scrum	RAD
Определение основных требований в начале проекта	Да	Не обязательно	Нет
Изменение требований по мере развития проекта	Нет	Возможно (доработки и пожелания от клиента)	Да
Разработка итерациями (версиями)	Нет	Да	
Распространение промежуточного ПО	Нет	Да	
Стоимость внесения изменений в проект	Дешево на стадии формирования требований, затем все дороже	Приблизительно равномерно по стоимости	
Особенность методологии	Строгое следование по «Маршрутной карте»	Строгое наличие 3-х участников разработки и следование плану разработки	Небольшая длительность таймбоксов, небольшая команда, наличие case-средств, фаза анализа рисков для решения о продолжении разработки. Отсутствие нужды в большом бюджете на старте
Реализация	Последовательная, согласно «Маршрутной карте»	Итерационная, со спринтами изменяемой длительности	Итерационная, с возможностью для параллельного ведения работ внутри таймбокса

Характеристика проекта	Прикладная методология		
	ASAP	Agile Scrum	RAD
Простота реализации	Легко	Средне	Средне
Число функциональных ошибок	~10, все в результате единственного этапа разработки	~10, но на каждом этапе меньшее число ошибок	~10, в среднем за виток примерно так же как за спринт в Scrum, но позволило снизить число ошибок на последних этапах за счет анализа рисков
Итоговое время разработки	25 дней	21 день	20 дней
Среднее время отклика	Незначительное на числе пользователей до 500, затем значительный рост		

ЗАКЛЮЧЕНИЕ

В рамках данной работы изучен каскадный метод внедрения ИС, его достоинства и недостатки, доказана целесообразность применения в разработке системы для автоматизации ключевых бизнес-процессов городской больницы. В ходе работы был произведен сбор требований с помощью опроса персонала, изучена существующая документация и проведен анализ полученных данных, на основе которого были сформулированы пользовательские и функциональные требования к разрабатываемой системе, определен их приоритет в разработке.

На основе рассмотренных материалов в среде ARIS Express смоделированы ключевые бизнес-процессы учреждения, каждый из которых рассмотрен с углублением до 3 уровня. Далее составлены бизнес-процессы в модели «TO-BE» для того, чтобы показать какие изменения привнесет разрабатываемая система на каждом из уровней организации. Ввиду того, что создание системы предполагало работу с БД, была проведена классификация классов данных, их нормализация и спроектирована архитектура данных. Затем на базе собранных и проанализированных данных созданы соответствующие таблицы с использованием БД MySQL, установлены связи между соответствующими ключевыми полям, после чего была непосредственно начата разработка сайта и его верстка. После создания сайта, продемонстрирован интерфейс и произведено его функциональное и системное тестирование, которое показало работоспособность и готовность каждого элемента в отдельности и всей системы в целом.

Таким образом, можно заключить, что все поставленные в ходе работы задачи выполнены. В отчете для наглядности отображены рисунки, снимки экрана. Всего отчет проиллюстрирован 6-ю таблицами и 41 рисунок/снимок экрана. Безусловно, разработанная система имеет свои

преимущества, но также не исключает возможности ее улучшения, дальнейшей работы по расширению функциональных возможностей ИС. Основные преимущества создаваемой системы:

- простота освоения и использования технологии (для этого достаточно иметь минимальные навыки пользования ПК (персональным компьютером) или мобильным и наличие сети Internet);
- доступность (просмотр или редактура доступны из любого места с любого устройства, при наличии интернета);
- удобность администрирования (сайт и базу данных можно хранить на локальном сервере, расположенном в больнице);
- наглядность;
- обновление в реальном времени (если в карту внесены изменения, они тут же отобразятся у пациента, нет необходимости в постоянном ее получении);
- повышения качества обслуживания пациентов (отсутствие очередей в регистратуру) и другие.

Список использованных литературных источников

1. Маглинец Ю.А. Анализ требований к автоматизированным информационным системам: учебное пособие – Москва: Интуит НОУ, 2016. – 192 с.
2. Маглинец Ю.А. Разработка информационных систем. Часть 1, Структурные методы. – Красноярск: Кларитеанум, 2004. – 120 с.
3. Шеер А.-В. Бизнес-процессы: основные понятия, теории, методы. – М.: Просветитель, 1999.
4. Шеер А.В. Моделирование бизнес-процессов – М.: Серебряные нити, 2014. – 219 с.
5. Орешкина А.М. Анализ, проектирование, разработка и тестирование приложения для автоматизации городской больницы: диплом бакалавра / МИРЭА. – М., 2017. – 49 с.
6. Катасонова Н.С. Автоматизация ключевых бизнес-процессов городской больницы на основе каскадной модели внедрения / МИРЭА. – М., 2018. – 55 с. – URL: https://stepanovd.com/training/20-vkr/64-vkrb-2018-2-katasonova.
7. Самуйлов К.Е. Бизнес-процессы и информационные технологии в управлении телекоммуникационными компаниями / К.Е. Самуйлов. – М.: Альпина Паблишер, 2014. – 220 с.
8. Старовойтова Т. Ф. Информационные системы в бизнесе / Т.Ф. Старовойтова, А.Н. Лавренов. – М.: Академия управления при Президенте Республики Беларусь, 2015. – 150 с.
9. Теличенко В. И. Информационное моделирование технологий и бизнес-процессов в строительстве / В.И. Теличенко, А.А. Лапидус,

А.А. Морозенко. – М.: Издательство Ассоциации строительных вузов, 2017. – 144 с.

10. Модели жизненного цикла: учеб. пособие / Д.Б. Берг, Е.А. Ульянова, П.В. Добряк. – Екатеринбург: Издательство Уральского университета, 2014. – 74 с.
11. Гудков Е.А., Деревнина А.М., Катасонова Н.С. Анализ каскадной, итерационной и спиралевидной моделей внедрения корпоративных информационных систем // Корпоративные информационные системы. – 2018. – №1. – С. 18-29. – URL: https://corpinfosys.ru/archive/issue-1/48-2018-1-models.
12. Степанов Д.Ю. Анализ, проектирование и разработка корпоративных информационных систем: теория и практика // Российский технологический журнал, 2015. – т.8, №3.
13. Илья Корнипаев. Требования для программного обеспечения: рекомендации по сбору и документированию – М.: Издательство «Книга по требованию», 2013. – 118 с.
14. Степанов Д.Ю. Анализ, проектирование и разработка корпоративных информационных систем: основные термины и определения / МИРЭА. – М., 2017.
15. Леффингуэлл Дин, Уидриг Дон. Принципы работы с требованиями к программному обеспечению. Унифицированный подход. М.: Вильямс, 2002. – 448 с.
16. Шматалюк А. и др. Моделирование бизнеса. Методология ARIS. Практическое руководство. – М.: Серебряные нити, 2001.
17. Шеер А.В. ARIS – моделирование бизнес-процессов / пер. с англ. и ред. Рыбянец А.А. – 3-е изд. – М.: Вильямс, 2009.
18. Репин В. В. Бизнес-процессы компании: построение, анализ, регламентация. – М.: Стандарты и качество, 2007.

19. Учебное пособие по работе со средствами ARIS Express – http://library.miit.ru/methodics/29.09.17/Уч-мет.ARIS.pdf.
20. Гулиев Я.И., Белышев Д.В., Михеев А.Е. Моделирование бизнес-процессов медицинской организации: классификация процессов: научная статья – (Институт программных систем им. А.К. Айламазяна РАН, Переславль-Залесский, Россия, ГБУ «Инфогород», Москва, Россия).
21. Дейт К.Дж. Введение в системы баз данных / пер. с англ. и ред. Птицына К.А. – 8-е изд. – М.: Вильямс, 2016.
22. Иан Соммервилл. Инженерия программного обеспечения. – Москва: Вильямс, 2002. – с. 642.
23. Голицына О.Л., Партыка Т.Л., Попов И.И. Языки программирования. Учебное пособие – ФОРУМ, ИНФРА-М, 2008.
24. Мишель Е. Дэвис Изучаем PHP и MySQL. – М.: Символ-плюс, 2008.
25. Линн Бейли, Майкл Моррисон Изучаем PHP и MySQL – Эксмо, 2010.
26. Робин Никсон - Создаем веб-сайты с помощью PHP, MySQL и JS – O'Reilly, 2011.
27. Ларри Ульман: Основы программирования на PHP. – М.: ДМК-Пресс, 2001.
28. Фримен Эрик, Фримен Элизабет Изучаем HTML, XHTML и CSS. – СПб.: Питер.
29. Гладкий А. Веб-Самоделкин. Как самому создать сайт быстро и профессионально – Литрес, 2012. – 4374 с.
30. Ташков П. А. Веб-мастеринг. HTML, CSS, JavaScript, PHP, CMS, AJAX, раскрутка – Питер, 2010. – 512 с.
31. Влад Мержевич Вёрстка веб-страниц – HTMLBOOKS, 2011. – 378 с.
32. Филиппов С.А. Основы современного веб-программирования – НИЯУ МИФИ, 2011. – 160 с.

33. Постановление Правительства РФ от 01.01.2002 N 1 (ред. от 28.04.2018) "О Классификации основных средств, включаемых в амортизационные группы".
34. Майк Кон. Пользовательские истории. Гибкая разработка программного обеспечения = User Stories Applied: For Agile Software Developme. – Вильямс, 2012. – 256 с.
35. Маркус Гэртнер. ATDD - разработка программного обеспечения через приемочные тесты. – ДМК-Пресс, 2013.
36. Джефф Сазерленд. Scrum. Революционный метод управления проектами = Scrum. The art of doing twice the work in half the time. – Манн, Иванов и Фербер, 2016. – 288 с.
37. Гудков Е.А. Применение спиралевидной модели внедрения информационных систем в городской поликлинике / МИРЭА. – М., 2018. – 45 с. – URL: https://stepanovd.com/training/20-vkr/63-vkrb-2018-1-gudkov.
38. Kerr J and Hunter R (1994) Inside RAD: how to build fully functional computer systems in 90 days or less. Mcgraw-Hill, NewYork.
39. P. Beynon-Davies, C. Carne, H. Mackay and D. Tudhope (1999) Rapid application development (RAD): an empirical review. European Journal of Information Systems (1999) 8, 211–223 Operational Research Society Ltd.

ПРИЛОЖЕНИЕ А

Index.php

```
<?php
        header("Content-type: text/html; charset=utf-8");
    require_once("header.php");
        $_SESSION["error_messages"] = '';
    $_SESSION["success_messages"] = '';
?>
...
<?php
    require_once("footer.php");
?>
```

About.php

```
<?php
    require_once("header.php");
?>
<DOCTYPE !html>
<html>
<head>
<h1>О нас</h1>
...
<?php
    require_once("footer.php");
?>
```

Form_auth.php

```
<?php
    require_once("header.php");
?>
<div class="block_for_messages">
<body bgcolor="ccccе5">
</body>
```

```
<?php

    if(isset($_SESSION["error_messages"]) &&
!empty($_SESSION["error_messages"])){
        echo $_SESSION["error_messages"];
        unset($_SESSION["error_messages"]);
    }

    if(isset($_SESSION["success_messages"]) &&
!empty($_SESSION["success_messages"])){
        echo $_SESSION["success_messages"];
        unset($_SESSION["success_messages"]);
    }
  ?>
</div>
<?php
  if(!isset($_SESSION["email"]) && !isset($_SESSION["password"])){
?>
<div id="form_auth">
  <h2>Форма авторизации</h2>
  <form action="auth.php" method="post" name="form_auth" >
    <table>
      <tr>
        <td> Email: </td>
        <td>
          <input type="email" name="email" required="required" /><br
/>
          <span id="valid_email_message"
class="message_error"></span>
        </td>
      </tr>

      <tr>
        <td> Пароль: </td>
        <td>
```

```
                    <input type="password" name="password"
placeholder="минимум 6 символов" required="required" /><br />
                    <span id="valid_password_message"
class="message_error"></span>
                    </td>
                </tr>
                <tr>
                    <td> Введите проверочный код: </td>
                    <td>
                        <p>
                            <img src="captcha.php" alt="Капча" /> <br />
                            <input type="text" name="captcha"
placeholder="Проверочный код" />
                        </p>
                    </td>
                </tr>
                <tr>
                    <td colspan="2">
                        <input type="submit" name="btn_submit_auth"
value="Войти" />
                    </td>
                </tr>
            </table>
        </form>
        </form>
            <form action="/form_register.php">
                    <button type="submit">Еще не зарегистрированы?</button>
                    </form>
        </div>
        <?php
        }else{
        ?>
        <div id="authorized">
            <h2>Вы уже авторизованы</h2>
        </div>
```

```
<?php
   }
?>
<?php
   require_once("footer.php");
?>
```

Form-register.php

```
<?php
   require_once("header.php");
?>
<div class="block_for_messages">
   <?php
      if(isset($_SESSION["error_messages"])                    &&
!empty($_SESSION["error_messages"])){
         echo $_SESSION["error_messages"];
         unset($_SESSION["error_messages"]);
      }

      if(isset($_SESSION["success_messages"])                  &&
!empty($_SESSION["success_messages"])){
         echo $_SESSION["success_messages"];
         unset($_SESSION["success_messages"]);
      }
   ?>
</div>
<?php
   if(!isset($_SESSION["email"]) && !isset($_SESSION["password"])){
?>
<div id="form_register">
   <h2>Форма регистрации</h2>
   <form action="register.php" method="post" name="form_register" >
      <table>
         <tr>
            <td> Имя: </td>
            <td>
```

```
                    <input type="text" name="first_name" required="required"/>
                </td>
            </tr>
            <tr>
                <td> Фамилия: </td>
                <td>
                    <input type="text" name="last_name" required="required" />
                </td>
            </tr>
                        <tr>
                <td> Отчество: </td>
                <td>
                    <input type="text" name="name" required="required"/>
                </td>
            </tr>
                        <tr>
                <td> Должность: </td>
                <td>
                    <input type="text" name="dolg" required="required"/>
                </td>
            </tr>
            <tr>
                <td> Email: </td>
                <td>
                    <input type="email" name="email" required="required" /><br
/>
                    <span                                id="valid_email_message"
class="mesage_error"></span>
                </td>
            </tr>
            <tr>
                <td> Пароль: </td>
                <td>
                    <input              type="password"              name="password"
placeholder="минимум 6 символов" required="required" /><br />
```

```
                    <span id="valid_password_message" class="message_error"></span>
                    </td>
                </tr>
                <tr>
                    <td colspan="2">
                    <input type="submit" name="btn_submit_register" value="Зарегистрироваться" />
                    </td>
                </tr>
            </table>
        </form>
    </div>
    <?php
        }else{
    ?>
        <div id="authorized">
            <h2>Вы уже зарегистрированы</h2>
        </div>
    <?php
        }
        require_once("footer.php");
    ?>
```

Edit.php

```
    <?php
            header("Content-type: text/html; charset=utf-8");
        require_once("header.php");
    ?>
    <body>
    <?php
        $host="localhost";
        $user="root";
        $pass="";
        $db_name="registration";
        $link=mysql_connect($host,$user,$pass);
```

```
            mysql_select_db($db_name,$link);

            if (isset($_GET['del_id'])) {
               $sql = mysql_query('DELETE FROM `patients` WHERE `ID` =
'.$_GET['del_id']);
            }

            if (isset($_GET['red_id'])) {
               if (isset($_POST['pmg'])) {
                  $sql = mysql_query('UPDATE `patients` SET '
                        .'`pmg` = "'.$_POST['pmg'].'",'
                        .'WHERE `ID` = '.$_GET['red_id']);
               }
            }
         ?>
         <table border='1'>
         <tr>
            <td>Записи</td>
         </tr>
         <?php
         $sql = mysql_query("SELECT `ID`, `pmg`  FROM `patients`", $link);
         while ($result = mysql_fetch_array($sql)) {
            echo     '<tr><td>'.$result['ID'].'</td>'.
                  '<td>'.$result['pmg'].'</td>'.
                  '<td><a href="?del_id='.$result['ID'].'">Удалить</a></td>'.
                  '<td><a
href="?red_id='.$result['ID'].'">Редактировать</a></td></tr>';
         }
         ?>
         </table>
         <?php
            if (isset($_GET['red_id'])) {
               $sql = mysql_query("SELECT `ID`, `pmg` FROM `patients` WHERE
`ID`=".$_GET['red_id'], $link);
               $result = mysql_fetch_array($sql);
               ?>
```

```
<table>
<form action="" method="post">
   <tr>
      <td>Записи:</td>
      <td><input type="text" name="pmg" value="<?php echo
($result['pmg']); ?>"></td>
   </tr>
   <tr>
      <td colspan="2"><input type="submit" value="OK"></td>
   </tr>
</form>
</table>
      <?php
   }
?>
</body>
</html>
<?php
   require_once("footer.php");
?>
```

Search.php

```
<?php
      header("Content-type: text/html; charset=utf-8");
   require_once("header.php");
$searchterm = trim ( $_POST['searchterm'] );
if (!$searchterm)
   die ("Не все данные введены.<br>
   Пожалуйста, вернитесь назад и закончите ввод");
$searchterm = addslashes ($searchterm);
   $server = "localhost";
   $username = "root";
   $password = "";
$link = mysql_pconnect ($server,$username,$password);
if ( !$link ) die ("Невозможно подключение к MySQL");
   $database = "registration";
```

```
mysql_select_db ($database) or die ("Невозможно открыть $database");
$query = "SELECT * FROM patients WHERE "
   .$_POST['searchtype']." like '%".$searchterm."%'";
$result = mysql_query ( $query );
$n = mysql_num_rows ( $result );
for ( $i=0; $i<$n; $i++ )
{
   $row = mysql_fetch_array($result);
   echo "<b><p>Номер карты ".($i). $row['id']. "</b><br>";
   echo "Фамилия: ".$row['last_name']."<br>";
   echo "Имя: ".$row['first_name']."<br>";
   echo "Отчество: ".$row['name']."<br>";
      echo "Пол: ".$row['gender']."</p>";
   echo "Возраст: ".$row['age']."</p>";
      echo "ПМЖ: ".$row['pmg']."</p>";
      echo "Место работы: ".$row['job']."</p>";
}
$query = "SELECT * FROM anamnez WHERE "
   .$_POST['searchtype']." like '%".$searchterm."%'";
$result = mysql_query ( $query );
$n = mysql_num_rows ( $result );
for ( $i=0; $i<$n; $i++ )
{
   $row = mysql_fetch_array($result);
   echo "<b><p>Анамнез" ."</b><br>";
   echo "Записи: ".$row['zap']."<br>";
}
$query = "SELECT * FROM dnevnik WHERE "
   .$_POST['searchtype']." like '%".$searchterm."%'";
$result = mysql_query ( $query );
$n = mysql_num_rows ( $result );
for ( $i=0; $i<$n; $i++ )
{
   $row = mysql_fetch_array($result);
   echo "<b><p>Дневник" ."</b><br>";
   echo "Записи: ".$row['zap']."<br>";
```

```
}
$query = "SELECT * FROM epikriz WHERE "
   .$_POST['searchtype']." like '%".$searchterm."%'";
$result = mysql_query ( $query );
$n = mysql_num_rows ( $result );
for ( $i=0; $i<$n; $i++ )
{
   $row = mysql_fetch_array($result);
   echo "<b><p>Эпикриз" ."</b><br>";
   echo "Записи: ".$row['info']."<br>";
}
$query = "SELECT * FROM naznach WHERE "
   .$_POST['searchtype']." like '%".$searchterm."%'";
$result = mysql_query ( $query );
$n = mysql_num_rows ( $result );
for ( $i=0; $i<$n; $i++ )
{
   $row = mysql_fetch_array($result);
   echo "<b><p>Назначения" ."</b><br>";
   echo "Дата: ".$row['date']."<br>";
   echo "Назначения: ".$row['naznach']."<br>";
}
$query = "SELECT * FROM temp WHERE "
   .$_POST['searchtype']." like '%".$searchterm."%'";
$result = mysql_query ( $query );
$n = mysql_num_rows ( $result );
for ( $i=0; $i<$n; $i++ )
{
   $row = mysql_fetch_array($result);
   echo "<b><p>Температурный лист" ."</b><br>";
   echo "Дата: ".$row['date']."<br>";
   echo "Время: ".$row['time']."<br>";
   echo "Дыхание: ".$row['dyx']."<br>";
   echo "Температура: ".$row['temp']."<br>";
}
?>
```

```
<?php
if ( $n == 0 ) echo "Ничего не можем предложить. Извините";
mysql_close ( $link );
?>
<?php
   require_once("footer.php");
?>
```

Dbconnect.php

```
<?php
   header("Content-type: text/html; charset=utf-8");
   $server = "localhost";
   $username = "root";
   $password = "";
   $database = "registration";
   $mysqli = new mysqli($server, $username, $password, $database);
   if (mysqli_connect_errno()) {
      echo "<p><strong>Ошибка подключения к БД</strong>. Описание
ошибки: ".mysqli_connect_error()."</p>";
      exit();
   }
   $mysqli->set_charset('utf8');
   $address_site = "http://mark2.loc";
?>
```