



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

Физико-технологический институт
Кафедра оптических и биотехнических систем и технологий

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА НА ТЕМУ:

**«ПРИМЕНЕНИЕ МЕТОДОЛОГИИ AGILE SCRUM ДЛЯ РЕАЛИЗАЦИИ
АВТОМАТИЗИРОВАННОГО РАБОЧЕГО МЕСТА ВРАЧА ТЕРАПЕВТА
В ГОРОДСКОЙ БОЛЬНИЦЕ»**

Студент:

Болохнов А.А.

Научный руководитель:

к.т.н., доц. МИРЭА Степанов Д.Ю.

Москва – 2019

Оглавление

Оглавление	2
Введение	4
Раздел 1. Описание модели внедрения программных продуктов Agile	
Scrum.....	6
1.1. Методология Agile.....	6
1.2. Метод Scrum.....	8
Раздел 2. Идентификация требований.....	12
2.1. Пользовательские и функциональные требования, их приоритизация ..	12
2.2. Бэклог продукта	17
2.3. Формирование спринтов.....	18
Раздел 3. Проектирование ключевых бизнес-процессов.....	21
3.1. Способы моделирования бизнес-процессов	22
3.2. Проектирование процессов в AS-IS с помощью VACD и DFD (первый спринт).....	24
3.3. Проектирование процессов в TO-BE с помощью VACD и DFD.....	26
3.4. Проектирование данных	28
3.5. Проектирование интерфейсов	30
Раздел 4. Разработка приложения.....	33
4.1. Прототип программы (первый спринт-продолжение).....	33
4.2. Прототип программы (второй спринт).....	34
4.3. Прототип программы (третий спринт).....	35
Раздел 5. Тестирование приложения.....	38
5.1. Функциональное тестирование	38
5.2. Нагрузочное тестирование	41
Заключение.....	43

Список использованных литературных источников	45
Приложение А	44

Введение

Всемирное использование компьютерных технологий в медицине началось с появлением компьютеров в начале 1950-х годов. В 1949 году в Германии Густав Вагнер учредил первую профессиональную организацию по информатике в области здравоохранения. Информатизация здравоохранения также называется Health Information Systems - это дисциплина на стыке информации, информатики и здравоохранения. Дисциплина касается ресурсов, устройств и методов, необходимых для оптимизации сбора, хранения, извлечения и использования информации в области здравоохранения и биомедицины. В состав инструментов информатики в области здравоохранения включаются компьютеры, клинические руководства, официальная медицинская терминология, информационные и коммуникационные системы. Все это применяется в областях сестринского дела, клинической помощи, стоматологии, фармацевтики, здравоохранения, профессиональной терапии, биомедицинских исследований и многих других.

Важным фактором развития информационных технологий в области медицины стало создание медицинских баз данных и автоматизированных рабочих мест (АРМ). Ежедневно врач принимает значительное количество пациентов за смену/прием. Медицинские базы данных помогают решать проблему учета всех пациентов, систематизацию их данных и ускорить процесс получения необходимой информации. Автоматизированное рабочее место врача представляет собой комплекс вычислительной техники и программного обеспечения, который располагается на рабочем месте врача-специалиста и служит для автоматизации его деятельности. Совокупность обозначенных выше решений позволяет не тратить рабочее время врача на поиск информации

на бумажных карточках и в картотеках, а использовать его непосредственно для обследования пациента и оперативно назначить необходимое лечение.

Использование систем управления базами данных, как например, Microsoft Access, в которой и будет разработано приложение для создания автоматизированного рабочего места врача – терапевта, позволит существенно упростить и ускорить процесс приема врачом пациента и работы с его персональными данными.

Цель данной работы – автоматизация ключевых бизнес-процессов врача-терапевта на основе метода Agile Scrum. Разработанная в конечном итоге программа позволит оптимизировать работу врача-терапевта и улучшить качество предоставляемых медицинских услуг населению. Исходя из цели работы, необходимо решить следующие задачи:

- провести детальный анализ методологии внедрения Agile Scrum;
- идентифицировать требования и сформировать бэклог;
- спроектировать процессы и оргструктуры в моделях AS-IS и TO-BE нотации ARIS VACD и DFD до 3-4 уровней детализации;
- для каждого спринта смоделировать разрабатываемые пользовательские интерфейсы;
- спроектировать структуры данных и нормализации таблиц данных;
- реализовать операции ключевого процесса в среде MS Access;
- протестировать и количественно оценить результаты тестирования;
- подготовить блок-схемы алгоритма работы программы;
- подвести итоги проделанной работы.

Раздел 1. Описание модели внедрения программных продуктов Agile Scrum

1.1. Методология Agile

Гибкая разработка программного обеспечения относится к группе методологий разработки программного обеспечения, основанных на итеративном развитии, где требования и решения развиваются благодаря сотрудничеству между самоорганизующимися кросс-функциональными командами. Гибкие методы или гибкие процессы в целом способствуют дисциплинированному процессу управления проектами.

Agile – это процесс, который помогает командам быстро и непредсказуемо реагировать на отзывы, которые они получают в своем проекте. Это создает возможности для оценки направления проекта в течение цикла разработки. Команды оценивают проект на регулярных встречах, называемых спринтами или итерациями. Процесс управления очень полезен для компаний-разработчиков программного обеспечения потому, что он помогает анализировать и совершенствовать свой продукт на протяжении его развития. Это позволяет компаниям выпускать на рынок высокоценные и конкурентоспособные продукты. В 2001 году небольшой группой людей, уставших от традиционного подхода к управлению проектами разработки программного обеспечения, был разработан Agile манифест (Agile Manifesto). Agile манифест имеет четыре основные ценности:

- люди и взаимодействие важнее процессов и инструментов;
- работающий продукт важнее исчерпывающей документации;
- сотрудничество с заказчиком важнее согласования условий контракта;
- готовность к изменениям важнее следования первоначальному плану.

Принципы Agile:

- наивысшим приоритетом является удовлетворение потребностей заказчика, благодаря регулярной и ранней поставке ценного программного обеспечения;
- изменение требований приветствуется, даже на поздних стадиях разработки;
- работающий продукт следует выпускать как можно чаще с периодичностью от пары недель до пары месяцев;
- на протяжении всего проекта разработчики и представители бизнеса должны ежедневно работать вместе;
- над проектом должны работать мотивированные профессионалы. Чтобы работа была сделана, создайте условия, обеспечьте поддержку и полностью доверьтесь им;
- непосредственное общение является наиболее практичным и эффективным способом обмена информацией как с самой командой, так и внутри команды;
- работающий продукт – основной показатель прогресса;
- инвесторы, разработчики и пользователи должны иметь возможность поддерживать постоянный ритм бесконечно;
- постоянное внимание к техническому совершенству и качеству проектирования повышает гибкость проекта;
- простота – искусство минимизации лишней работы – крайне необходима;
- самые лучшие требования, архитектурные и технические решения рождаются у самоорганизующихся команд;

- команда должна систематически анализировать возможные способы улучшения эффективности и соответственно корректировать стиль своей работы.

Применение Agile снижает общие риски, связанные с доставкой, объемом и бюджетом проекта. Он поощряет сотрудничество между клиентом и командой, предлагая взаимную выгоду в смягчении высоких рисков при разработке программного обеспечения.

1.2. Метод Scrum

Scrum – это подмножество Agile. Это легкая, наиболее широко используемая структура процесса для гибкого развития. «Структура процесса» представляет собой определенный набор практик, который должен соблюдаться, чтобы процесс соответствовал структуре (например, структура процесса Scrum требует использования циклов разработки под названием Sprints). «Легкий вес» означает, что накладные расходы процесса хранятся как можно меньше, чтобы максимально увеличить время, необходимое для получения полезной работы.

Команда в Scrum создаёт три артефакта. Это специальные объекты, которые помогают организовать работу Команды и выпустить работающий Продукт. Первичным артефактом в разработке Scrum является, конечно же, сам продукт. Модель Scrum ожидает, что команда выведет продукт или систему в потенциально рабочее состояние в конце каждого спринта Scrum.

«Бэклог продукта» – еще один артефакт Scrum. Это полный список функций, которые еще предстоит добавить в продукт. Владелец продукта приоритизирует бэклог. Поэтому команда всегда работает с наиболее ценными функциями. Самый популярный и успешный способ создания бэклога продукта с использованием методологии Scrum состоит в том, чтобы заполнить его

«рассказами» пользователей, которые представляют собой краткое описание функций, описанных с точки зрения пользователя или клиента [5].

Дополнительный артефакт, являющийся результатом гибкой методологии Scrum – это диаграмма сгорания задач (Burndownchart). Диаграмма Burndown показывают объем работы, оставшийся либо в спринте, либо в релизе. Он является эффективным инструментом разработки программного обеспечения Scrum, чтобы определить, будет ли спринт или релиз по расписанию, чтобы все запланированные работы были завершены к нужному времени.

В проекте Scrum есть три роли: владелец продукта (The Product Owner), Scrum мастер (The Scrum Master) и члены команды (The team). Владелец продукта (The Product Owner) контролирует все бизнес-условия проекта, чтобы гарантировать, что нужный продукт построен и находится в правильном порядке. Ответственный владелец продукта уравнивает конкурирующие приоритеты, всегда на связи с командой и принимает решения по проекту.

Scrum Мастер (The Scrum Master) – это, по сути, тренер команды. Он помогает команде эффективно работать вместе. Scrum мастер обслуживает команду, устраняя препятствия, которые ухудшают прогресс в работе, облегчает встречи и создает дискуссионные группы, отслеживая прогресс. Решает возникающие проблемы и выполняет другие обязанности по управлению проектами.

Члены команды (The team) работают вместе, чтобы определить наилучший подход к достижению целей продукта, которые обозначены владельцем продукта. Команда решает, какие члены команды будут управлять конкретными задачами, и намечает техническую практику, необходимую для достижения желаемых целей.

Модель Scrum предполагает, что проекты продвигаются через серию спринтов. В соответствии с гибкой методологией спринты имеют временные рамки не более месяца, чаще всего две недели. Методика Scrum выступает за совещание по планированию в начале каждого спринта. Затем создают «Бэклог продукта» (Product Backlog) - список задач, которые необходимо выполнить во время спринта.

Каждый день спринта все члены команды должны посещать ежедневную встречу (Scrum meeting), включая Scrum мастера и владельца продукта. Как правило, встреча не должна длиться более 15 минут. В течение этого времени члены команды делятся тем, что они сделали в предыдущий день, над чем будут работать в текущий день и выявлять возникающие препятствия для прогресса.

В конце спринта команда проводит обзор проделанной работы, в ходе которого демонстрирует новую функциональность для ПО или любого другого заинтересованного лица, желающего предоставить обратную связь, которая может повлиять на следующий спринт. Цикл обратной связи в разработке программного обеспечения Scrum может привести к изменениям в новой функциональности, что может привести к пересмотру или добавлению элементов в «Бэклог продукта».

Еще одна деятельность в управлении проектом Scrum – ретроспектива спринта в конце каждого спринта. В этой встрече участвует вся команда, включая Scrum мастера и владельца продукта. Встреча – это возможность задуматься о завершении спринта и определить возможности для улучшения. Гибкий процесс Scrum приносит пользу организации, помогая ей:

- повысить качество результатов;
- лучше справляться с изменениями (и ожидать изменений);

- предоставлять более качественные оценки, используя меньше времени на их создание;
- жестче контролировать график проекта и его состояние.

В данной работе будет использована следующая последовательность шагов для реализации программы:

- определен заказчик (владелец продукта);
- определены цели и задачи проекта;
- определена команда разработки проекта;
- сформирован бэклог продукта;
- сформированы спринты;
- в рамках спринта разработано и протестировано приложение.

Раздел 2. Идентификация требований

Идентификация предварительных требований к новому продукту позволяет, в соответствии с требованиями клиента, определить характеристики нового продукта.

Зачастую клиент не может описать конкретные возможности желаемой системы, а лишь формулирует свои нечёткие представления о конечном виде продукта. В связи с этим, грамотное определение требований становится первым важным этапом на пути создания нового продукта или модернизации уже имеющегося. Следует выделить четыре основных этапа работы с требованиями:

- сбор требований;
- анализ требований;
- документирование требований;
- проектирование системы.

2.1. Пользовательские и функциональные требования, их приоритизация

Заказчиком оптимизации процесса был выбран врач – терапевт. В качестве ключевого процесса для оптимизации выбран процесс учета анализов пациентов. Терапевт - врач общей практики. Его задача – поставить диагноз, проанализировав жалобы пациента и при необходимости направить его к более узкому специалисту. Врач-терапевт должен обладать широкими знаниями, в том числе в области гастроэнтерологии, хирургии, кардиологии и др. Участковые врачи-терапевты ведут прием в поликлинике и по вызову больного на дому. В обязанности данного врача входит также заполнение большого объема бумажных документов: амбулаторных карточек, больничных листов и отчетов и др. [1]

После обсуждения с заказчиком был составлен список пользовательских требований. Пользовательские требования – это короткие, простые описания функций, которые рассматриваются с точки зрения конечного пользователя. Пользовательские требования описывают, что хочет пользователь и почему. Они помогают создать упрощенное описание конечного продукта. Обычно пользовательские требования имеют следующий вид: Как < тип пользователя >, Я хочу < какая-то цель > исходя из этого < причина > [3]. Пример: как администратор, я хочу просмотреть фотографии до их публикации, чтобы я мог убедиться, что они соответствуют требованиям.

Заказчик, врач – терапевт, работает в государственной больнице, где применяется Единая медицинская информационно-аналитическая система (ЕМИАС). ЕМИАС – это государственная единая медицинская информационно-аналитическая система города Москвы, которая создается с целью повышения качества и доступности медицинских услуг государственных учреждений здравоохранения [2]. Исходя из ключевых бизнес-процессов для автоматизации выбраны процессы, а именно:

- учет анализов пациентов;
- выдача направлений на обследование;
- назначение лекарственных препаратов.

На основании пользовательских требований, и с учетом желаемых улучшений составлен следующий их список:

- возможность хранения и просмотра данных о пациенте;
- возможность создания, редактирования и удаления данных о пациентах;
- возможность поиска по заданным параметрам;
- возможность назначить анализ для пациента;
- возможность назначить обследование для пациента;

- возможность создания рецепта на выдачу лекарственных препаратов;
- возможность отображения выданных пациенту рецептов на лекарственные препараты;
- возможность редактировать и удалять назначенный анализ; возможность редактирования и удаления рецептов на выдачу лекарственных препаратов; возможность редактировать и удалять ранее назначенные обследования;
- возможность поиска выданных рецептов пациенту по заданным параметрам; возможность поиска назначенных пациенту обследований по заданным параметрам;
- возможность отображения даты и времени назначенного анализа;
- возможность отображения даты и времени поступления результатов анализа;
- возможность отображения результатов анализов пациента; защита данных от несанкционированного доступа;
- возможность создания отчетов о:
 - доступных и выданных препаратах;
 - выписанных направлениях на обследование;
- возможность отображения наличия на складе аптечного киоска больницы лекарственных препаратов.

Приоритизация требований является одной из основных концепций в Agile практики. Приоритизация – принятие решений о том, в каком порядке команда разработчиков будет работать над требованиями в проекте. Понимание приоритетности имеет особенно важное значение при Agile проектировании, поскольку проект имеет временную шкалу с фиксированным набором ресурсов, которые требует приоритизации, чтобы учесть временные и бюджетные

ограничения. Программное обеспечение будет реализовано в среде Microsoft Access 2010.

Приоритеты будут ранжированы в зависимости от значимости, на три основные степени: высокая, средняя, низкая. Высокая значимость – требования необходимые для полноценной работы всей программы. Средняя значимость – реализация требований значительно упрощает процесс работы с программой. Низкая значимость – требования не являются необходимыми для функционирования программы, но их реализация положительно отразится на релизе. На основании, приведенной выше приоритизации, составлен список ранжированных пользовательских требований:

- высокая степень значимости: требования 1-7;
- средняя степень значимости: требование 8, 9;
- низкая степень значимости: требования 10-14.

Следующим шагом определяем функциональные требования (то, какие действия будут выполняться системой для реализации пользовательских требований):

- отображение в таблице данных о пациентах;
- использование компонентов программы, обеспечивающие создание, редактирование и удаление данных о пациентах;
- использование компонентов программы, обеспечивающие поиск данных по ключевым параметрам;
- использование компонентов программы, обеспечивающих возможность назначать анализы пациентам;
- использование компонентов программы, обеспечивающих возможность назначать обследование пациентам;

- использование компонентов программы, обеспечивающих возможность создания рецептов на выдачу лекарственных препаратов пациентам;
- отображение выданных пациенту рецептов на лекарственные препараты;
- использование компонентов программы, обеспечивающих возможность редактировать и удалять: назначенный анализ, рецепт на выдачу лекарственных препаратов, назначенное обследование;
- использование компонентов программы, обеспечивающих поиск выданных пациенту рецептов на лекарственные препараты и назначенных обследований по заданным параметрам;
- отображение даты и времени назначенного анализа в отдельной графе интерфейса программы;
- отображение на экране данных о дате и времени поступления результатов анализа;
- отображение информации, содержащей результаты анализов пациента;
- использование программного кода, обеспечивающего защиту от несанкционированного доступа, реализация формы авторизации пользователя в системе;
- использование программного кода, позволяющего вводить и проверять введенный пользователем пароль;
- использование компонентов программы, обеспечивающих возможность создания отчетов;
- использование компонентов программы, позволяющих отображать наличие лекарственных препаратов на складе аптечного киоска больницы.

2.2. Бэклог продукта

После сбора и анализа пользовательских и функциональных требований будет составлен бэклог продукта, что это обеспечит упрощение процесса отслеживания требований и позволит контролировать их выполнение. Бэклог составлен в виде таблицы 2.1.

Таблица 2.1 – Бэклог продукта

№	Степень приоритизации	Пользовательское требование	Функциональное требование
1	Высокая	Возможность хранения и просмотра данных о пациенте	Отображение в таблице данных о пациентах
2	Высокая	Возможность создания, редактирования и удаления данных о пациентах	Использование компонентов программы, обеспечивающие создание, редактирование и удаление данных о пациентах
3	Высокая	Возможность поиска по заданным параметрам	Использование компонентов программы, обеспечивающие поиск данных по заданным параметрам
4	Высокая	Возможность назначить анализ для пациента	Использование компонентов программы, обеспечивающих возможность назначать анализы пациентам
5	Высокая	Возможность назначить обследование для пациента	Использование компонентов программы, обеспечивающих возможность назначать обследование пациентам
6	Высокая	Возможность создания рецепта на выдачу лекарственных препаратов	Использование компонентов программы, обеспечивающих возможность создания рецептов на выдачу лекарственных препаратов пациентам
7	Высокая	Возможность отображения выданных пациенту рецептов на лекарственные препараты	Отображение выданных пациенту рецептов на лекарственные препараты
8	Средняя	Возможность редактировать и удалять назначенный анализ; возможность редактирования и удаления рецептов на выдачу лекарственных препаратов; возможность редактировать и удалять ранее назначенные обследования	Использование компонентов программы, обеспечивающих возможность редактировать и удалять: назначенный анализ, рецепт на выдачу лекарственных препаратов, назначенное обследование
9	Средняя	Возможность поиска выданных рецептов пациенту по заданным параметрам; возможность поиска назначенных пациенту обследований по заданным параметрам	Использование компонентов программы, обеспечивающих поиск выданных пациенту рецептов на лекарственные препараты и назначенных обследований по заданным параметрам
10	Низкая	Возможность отображения даты и времени назначения анализа	Отображение даты и времени назначенного анализа в отдельной

№	Степень приоритизации	Пользовательское требование	Функциональное требование
			графе интерфейса программы
11	Низкая	Возможность отображения даты и времени поступления результатов анализа	Отображение на экране данных о дате и времени поступления результатов анализа
12	Низкая	Возможность отображения результатов анализов пациента; защита данных от несанкционированного доступа	Отображение информации, содержащей результаты анализов пациента; использование программного кода, обеспечивающего защиту от несанкционированного доступа, реализация формы авторизации пользователя в системе; использование программного кода, позволяющего вводить и проверять введенный пользователем пароль
13	Низкая	Возможность создания отчетов о: доступных и выданных препаратах, выписанных направлениях на обследование	Использование компонентов программы, обеспечивающих возможность создания отчетов
14	Низкая	Возможность отображения наличия на складе аптечного киоска больницы лекарственных препаратов	Использование компонентов программы, позволяющих отображать наличие лекарственных препаратов на складе аптечного киоска больницы

2.3. Формирование спринтов

Один из основных элементов при разработке проекта согласно Agile Scrum – спринт (Sprint). По сути, это короткий цикл, следующий один за другим без перерывов. На основании представленной выше приоритизации, будут сформированы спринты. К первому спринту будут относиться все требования с высокой степенью приоритизации, второй спринт – средняя степень, третий спринт – низкая степень приоритизации. Разработка ПО будет осуществляться в последовательности, данной ниже.

Первый спринт (высокая степень приоритизации):

- анализ пользовательских и функциональных требований для конкретного спринта;
- моделирование ключевого бизнес-процесса нотации VACD и DFD;
- планирование спринта;

- создание баз данных и проектирование пользовательских интерфейсов;
- разработка возможности отображения, изменения и удаления данных о пациентах, разработка возможности поиска данных пациентов по заданным параметрам, разработка возможности назначения анализов и обследований, разработка возможности выдачи рецептов на лекарственные препараты пациентам (требования 1-7 в бэклоге продукта, таблица 2.1);
- создание первой тестовой версии интерфейса программы;
- тестирование реализованных возможностей;
- обзор спринта;
- ретроспектива спринта.

Для каждого спринта будут проведены следующие действия: анализ пользовательских и функциональных требований, планирование спринта, тестирование реализованных возможностей, обзор спринта и ретроспектива. В связи с этим, далее будут описаны только отличительные особенности каждого спринта.

Второй спринт (средняя степень приоритизации):

- разработка возможности редактирования и удаления назначенных пациенту анализов, рецептов на выдачу лекарственных препаратов, а также возможность редактировать и удалять ранее назначенные обследования (требования 8, 9 в бэклоге продукта, таблица 2.1);
- доработка интерфейса программы.

Третий спринт (низкая степень приоритизации):

- разработка возможности отображения даты и времени назначения анализа, даты и времени поступления результатов

анализа, возможность отображения результатов анализов пациента, использование компонентов программы, позволяющих вводить и проверять введенный пользователем пароль, разработка возможности создания отчётов и возможность отображения наличия на складе аптечного киоска больницы лекарственных препаратов (требования 10-14 в бэклоге продукта, таблица 2.1);

- создание итоговой версии интерфейса программы.

Раздел 3. Проектирование ключевых бизнес-процессов

Бизнес-процесс – это деятельность или набор действий, которые выполняют определенную организационную цель. Бизнес-процессы должны иметь целенаправленность, быть максимально конкретными и иметь последовательные результаты. Одним из эффективных методов преобразования идеи в конкретные результаты является разработка и заполнение диаграмм As-Is (как есть) и To-Be (как будет). Диаграмма As-Is описывает текущее состояние процесса, возможностей организации. Диаграмма To-Be описывает будущее состояние, другими словами, какие процессы и возможности организации появятся в будущем [6].

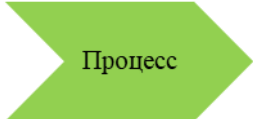

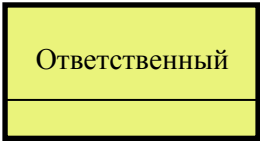
Первый шаг – определение необходимого процесса As-Is, как только он определен, его необходимо проанализировать. В момент анализа требуется задокументировать текущее состояние процесса, определить точку отсчета. Необходимо четко понимать функционирование, возможности процесса и системы. После чего можно приступать к графическому отображению процессов As-Is. Второй шаг - выявить любые недостатки в существующих процессах. Третий шаг – документирование и реализация процесса To-Be. На этом же этапе производится детальный анализ желаемых улучшений процесса и формируются конкретные предложения по его улучшению. Далее, как и в модели, As-Is создается графическое отображение процессов (блок-схема). Это наиболее сложный этап. При его реализации часто может оказаться, что желаемые улучшения не столь эффективны, как планировалось ранее. Последним шагом будет отслеживание реальной эффективности и актуальности введенных улучшений.

3.1. Способы моделирования бизнес-процессов

Одна из методологий для моделирования бизнес-процессов – методология ARIS (Architecture of Integrated Information Systems) или «архитектура интегрированных информационных систем». Методология состоит из различных нотаций (система условных обозначений, принятая в конкретной модели) и позволяет описать с различных точек зрения деятельность организации. ARIS рассматривает деятельность организации с четырех точек зрения: организационная структура, информационная структура, функциональная иерархия, управление бизнес-процессами. Преимуществом моделирования в ARIS является его наглядность в представлении сложных фактов, также это позволяет систематизировать подход к анализу. В зависимости от интересующей информации используются различные методы моделирования. К числу нотаций ARIS относятся: Value-added Chain Diagram (диаграмма цепочки процесса, добавляющего стоимость); нотации eEPC, Extended Event-driven Process Chain (расширенная нотация цепочки процесса, управляемого событиями) и PCD (диаграмма цепочки процесса); нотация Organizational Chart (организационная диаграмма); нотация FunctionTree (дерево функций); нотация Product Tree (дерево продуктов) [7]. Далее будет рассмотрена нотация Value-added Chain Diagram (VACD).


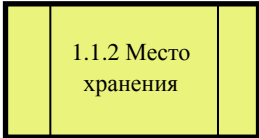
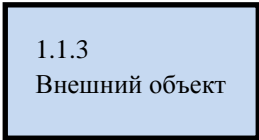
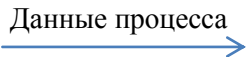
Нотация Value-added Chain Diagram (VACD) или «диаграмма цепочки процесса, добавляющего стоимость» – это тип модели, используемый для формулировки основных уровней процесса. VACD в основном используется для описания процессов верхнего уровня внутри организации. Основным объектом нотации – процесс или группа функций организации, служащие для получения добавленной стоимости. Элементы, используемые для моделирования нотации VACD, представлены в таблице 3.1 [8].

Таблица 3.1 – Условные обозначения нотации VACD

Графический элемент	Описание
	Процесс
	Входящий/исходящий объект
	Ответственный

Для описания нижних уровней организации будет использована нотация DFD (Data flow diagrams) или «диаграмма потоков данных». Диаграмма потоков данных (DFD) отображает поток информации для любого процесса или системы. Для этого используются определенные символы, такие как: прямоугольники, круги и стрелки, а также короткие текстовые метки для отображения входов данных, выходов, точек хранения и маршрутов между каждым пунктом. Блок-схемы данных могут варьироваться от простых, даже отрисованных вручную процессов, до глубоких многоуровневых, которые постепенно углубляются в процесс обработки данных. Они могут использоваться для анализа существующей системы или моделирования новой. Элементы, используемые для моделирования нотации DFD, представлены в таблице 3.2 [8].

Таблица 3.2 – Условные обозначения нотации DFD

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

3.2. Проектирование процессов в AS-IS с помощью VACD и DFD (первый спринт)

На рисунке 3.1 представлена диаграмма функционирования больницы As-Is (как есть) в нотации VACD. Основу функционирования составляют 3 процесса, а именно: «Пациент обратился в больницу», «Лечить пациента», «Выписать пациента». Также на диаграмме представлены ответственные за каждый процесс и входящие объекты.

Процесс «Лечить пациента» следует рассмотреть более подробно, так как именно он представляет наибольшую значимость в работе врача-терапевта. Для этого процесс будет подвергнут декомпозиции. Однако нотация VACD не подходит для этого. В данном случае хорошо подойдет нотация DFD. Результат декомпозиции – описание 2 уровня с использованием нотации DFD представлено в Приложении А на рисунке 1.1. Как видно из рисунка 1.1,

процесс «Лечить пациента» сложный и многоступенчатый, состоит из множества подпроцессов.

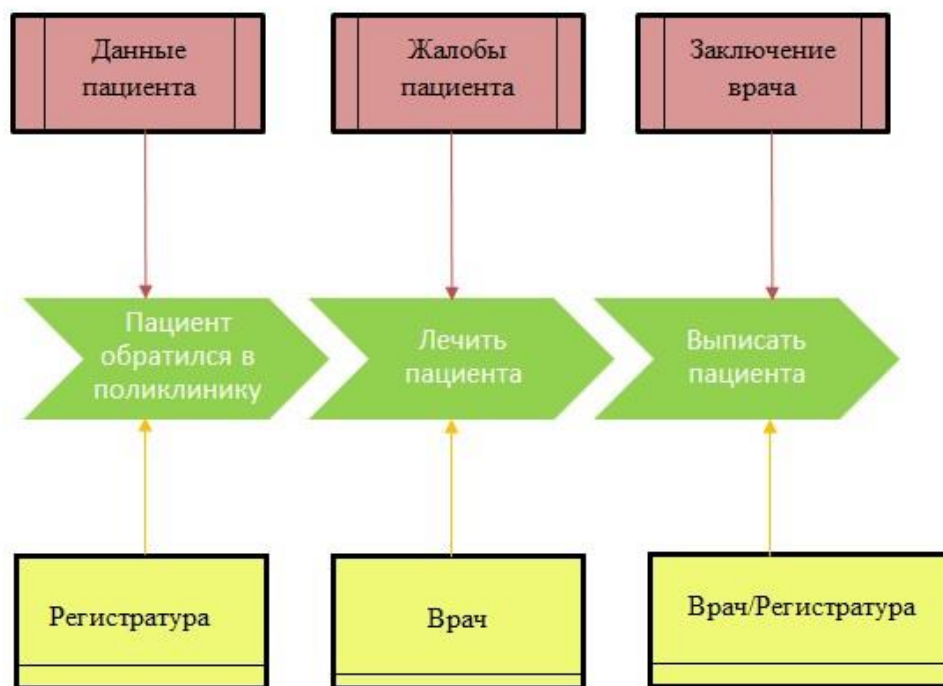


Рисунок 3.1 – Ключевые процессы первого (верхнего) уровня описания нотации VACD

В данном случае хорошо подойдет нотация DFD. Результат декомпозиции – описание 2 уровня с использованием нотации DFD представлено в Приложении А на рисунке 1.1. Как видно из рисунка 1.1, процесс «Лечить пациента» сложный и многоступенчатый, состоит из множества подпроцессов. Наибольший интерес в данной работе представляют подпроцессы «Назначить анализы», «Назначить обследование» и «Назначить лекарства» т.к. именно они были выбраны в качестве ключевых бизнес-процессов для оптимизации. Подвергаем их также декомпозиции в нотации DFD, до 3 уровня описания. Результат декомпозиции процесса «Назначить анализы» представлен в Приложении А на рисунке 1.2. На рисунке 1.3 в Приложении А представлен результат декомпозиции процесса «Назначить

обследование» и на рисунке 1.4 результат декомпозиции процесса «Назначить лекарства», соответственно. Дальнейшая декомпозиция не имеет смысла.

3.3. Проектирование процессов в TO-VE с помощью VACD и DFD

Модель To-Be позволяет рассмотреть ключевые бизнес-процессы медицинского учреждения после внедрения разрабатываемого ПО. Основное нововведение – использование базы данных, позволяющей заменить основные бумажные носители и уменьшить время поиска необходимой врачу информации о пациенте. Описание будет проводиться как и в модели As-Isс применением нотации VACD (для верхнего уровня описания) и DFD (для нижних уровней описания). Стоит отметить, что первый уровень описания в нотации VACD, приведенный на рисунке 3.2, ничем не отличается от описания в модели As-Is.

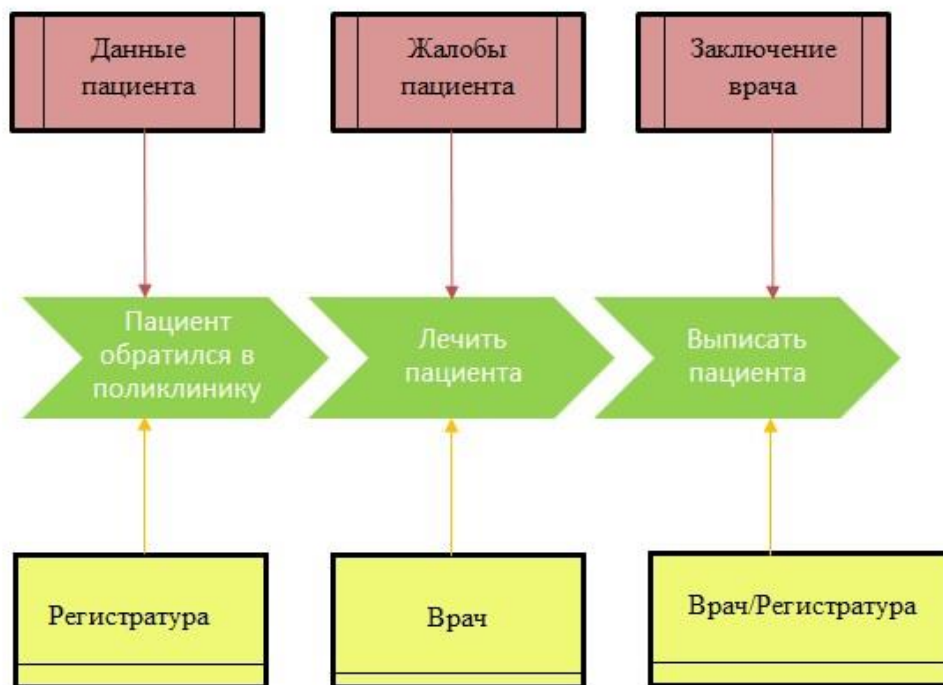


Рисунок 3.2 – Ключевые процессы первого (верхнего) уровня описания нотации VACD

На 2 уровне описания, как и в модели As-Is будет использована нотация DFD. «Медицинская карта пациента», содержащая информацию об анализах пациента, заменена базой данных. На рисунке 1.5 в Приложении А приведено описание 2 уровня детализации процесса «Лечить пациента».

Ключевые изменения в модели To-Be, по сравнению с As-Is наглядно отображены на 3 уровне описания нотации DFD. На рисунке 1.6 в Приложении А видно, что бумажные носители информации в процессе «Назначить анализы» заменены базой данных. Врачу не нужно больше искать данные в медицинской карте и тратить, соответственно, на это время приема. Похожие изменения произошли и в двух других ключевых бизнес-процессах. Так, на рисунке 1.7 в Приложении А представлена декомпозиция процесса «Назначить обследование» до 3 уровня описания нотации DFD. Декомпозиция процесса «Назначить лекарства» представлена на рисунке 1.8 в Приложении А.



Рисунок 3.3 – Блок-схема процесса «Назначить анализ»

Далее будет создана блок-схема для процесса «Назначить анализ» и представлена карта процессов. На рисунке 3.3 представлена блок-схема процесса «Назначить анализ». Также на рисунке 1.9 в Приложении А представлена общая карта процессов.

3.4. Проектирование данных

Вначале создается необходимая для дальнейшей работы база данных. Как правило, все данные приводят к третьей нормальной форме (НФ). Согласно 3НФ, все не ключевые поля, содержимое которых можно отнести к нескольким записям в таблице, должны быть вынесены в отдельную таблицу. В данном случае будет использоваться база данных, содержащая шесть таблиц: Пациенты, Анализы, Накладные на выдачу, Направления на обследования, Список обследований, Лекарства. На рисунке 3.4 представлена архитектура данных и связей между ними.



Рисунок 3.4 – Архитектура данных

Таблица «Пациенты» имеет ключевое поле «Код пациента» и связана с таблицей «Анализы», с ключевым полем «Код анализа», связью «Один ко многим». Также таблица «Пациенты» связана с таблицей «Накладные на выдачу» с ключевым полем «код заявки» связью «Один ко многим». Таблица «Накладные на выдачу» с ключевым полем «Код заявки», в свою очередь,

связана с таблицей «Лекарства», ключевое поле которой «Код препарата», связью «Один ко многим». Таблица «Пациенты» связана и с таблицей «Направления на обследование», ключевое поле которой «Код направления» связью «Один ко многим». В свою очередь таблица «Направления на обследование» связана с таблицей «Список обследований» с ключевым полем «Код обследования» связью «Один ко многим». Информацию из таблиц можно разбить на классы. В таблице 3.3 приведена информация о классах и типах данных, используемых в базе данных.

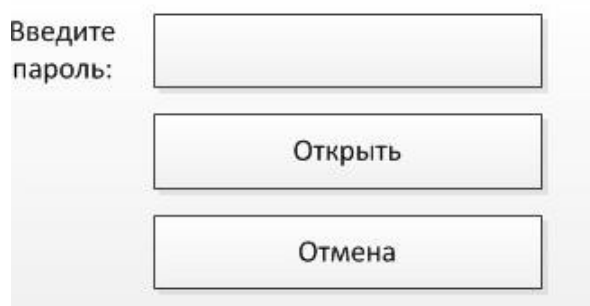
Таблица 3.3 – Классы и типы данных

Название класса	Поле данных	Тип данных	Размер поля
Пациенты	Код пациента (Ключевое поле)	Счётчик	Длинное целое
	Фамилия	Текстовый	255
	Имя	Текстовый	255
	Отчество	Текстовый	255
	Дата рождения	Дата/время	255
	Пол	Текстовый	255
	Номер мобильного	Текстовый	255
	Номер ОМС	Текстовый	255
Анализы	Постоянное место жительства	Текстовый	255
	Код анализа (Ключевое поле)	Счётчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Дата анализа	Текстовый	255
	Название анализа	Текстовый	255
	Дата поступления результатов	Дата/время	255
Накладные на выдачу	Результат анализа	Текстовый	255
	Код заявки (Ключевое поле)	Счётчик	Длинное целое

Название класса	Поле данных	Тип данных	Размер поля
	Код препарата	Числовой	Длинное целое
	Код пациента	Числовой	Длинное целое
	Нужное количество	Числовой	Длинное целое
	Дата оформления	Дата/время	255
Лекарства	Код препарата (Ключевое поле)	Числовой	Длинное целое
	Наименование препарата	Текстовый	255
	Количество на складе	Числовой	255
Направления на обследования	Код направления (Ключевое поле)	Счётчик	Длинное целое
	Код пациента	Числовой	Длинное целое
	Код обследования	Числовой	Длинное целое
	Дата оформления	Дата/время	255
Список обследований	Код обследования (Ключевое поле)	Числовой	Длинное целое
	Название обследования	Текстовый	255
	Номер кабинета	Числовой	Длинное целое

3.5. Проектирование интерфейсов

Для большей наглядности в Приложении А на рисунке 1.10 представлена схема приложения. Рисунок отображает расположение основных элементов управления в планируемой программе. Для защиты данных от несанкционированного доступа программа будет запрашивать пароль. Программа содержит в себе 12 основных страниц интерфейса. На рисунке 3.5 отдельно представлен интерфейс входа в систему.



Введите
пароль:

Открыть

Отмена

Рисунок 3.5 – Планируемый вид формы входа в систему

Также планируется создание главного меню для доступа к рабочим формам врача-терапевта. На рисунке 3.6 изображен планируемый вид главного меню. Для удобства использования, разрабатываемая программа должна быть информативной и понятной пользователю. При этом она должна соответствовать всем функциональным требованиям. На рисунке 3.7 приведен планируемый вид рабочей формы врача-терапевта для просмотра сведений о пациентах и их анализах. В приложении А на рисунке 1.11 представлена схема рабочей формы для выдачи, редактирования и удаления направлений на обследования. В приложении А на рисунке 1.12 изображена схема рабочей формы для выдачи, редактирования и удаления рецептов на лекарственные препараты.



Главное меню

Данные пациентов

Анализы пациентов

Направления на
обследования

Аптечный киоск

Выход

Рисунок 3.6 – Планируемый вид формы главного меню

Код пациента	<input type="text"/>	Номер ОМС	<input type="text"/>		
Фамилия	<input type="text"/>	Постоянное место жительства	<input type="text"/>		
Имя	<input type="text"/>				
Отчество	<input type="text"/>				
Дата рождения	<input type="text"/>				
Пол	<input type="text"/>				
Номер мобильного	<input type="text"/>				
<div>Анализы пациента</div>					
Код анализа	Код пациента	Дата анализа	Название анализа	Дата поступления результата	Результат анализа
В начало	Пред. запись	Сл. запись	В конец	Поиск	Сохранить изменения
	Добавить нового пациента	Удалить пациента	Печать формы	Заккрыть	

Рисунок 3.7 – Планируемый вид рабочей формы врача-терапевта

Раздел 4. Разработка приложения

4.1. Прототип программы (первый спринт-продолжение)

В первом спринте, была создана таблица с персональными данными пациентов: Фамилия, Имя, Отчество, Дата рождения, Пол, Номер мобильного телефона, Номер ОМС, Постоянное место жительства. В таблице отсутствует какой-либо интерфейс. Фрагмент таблицы представлен на рисунке 4.1.

к	фамилия	имя	отчество	дата рож	пол	номер мобил	номер ОМС	Постоянное место жительства
1	Петров	Александр	Григорьевич	12.12.1965	М	12243243243244	1232432453543423	г. Москва, пр-т Вернадского, 78
2	Сидоров	Олег	Петрович	13.08.1992	М	12312312312321	123123123123213123	г. Москва, ул. Стромынка, 20
4	Пронькин	Андрей	Юрьевич	12.11.1992	М	12321432432423	12312445546	г. Москва, Сокольнические пер., 34
5	Иванова	Екатерина	Фёдоровна	03.04.1978	Ж	77856345453457	655555547789435654	г. Москва, Яузская ул., 7
6	Полюкова	Ольга	Владимировн	07.09.1988	Ж	87968796789	789789789879	г. Москва, Сокольнический пер., 11

Рисунок 4.1 – Фрагмент таблицы «Пациенты»

На основании существующей таблицы с данными переходим к созданию интерфейса, обеспечивающего: просмотр, добавление и удаление информации о пациентах, а также поиск по ключевым параметрам. Фрагмент интерфейса представлен на рисунке 4.2.

Данные пациентов

код пациента

фамилия

имя

отчество

дата рождения

пол

номер мобильного

номер ОМС

Постоянное место жительства

Петров

Александр

Григорьевич

12.12.1965

М

12243243243244

1232432453543423

г. Москва, пр-т Вернадского, 78

В начало

Пред. запись

Сл. запись

В конец

Поиск по пациентам

Сохранить изменения

Добавить нового пациента

Удалить пациента

Просмотр всех пациентов

Печать формы

Закрыть

Рисунок 4.2 – Фрагмент интерфейса «Данные пациентов» на 1 спринте

В первом спринте также были созданы таблицы «Анализы», «Направления на обследования», «Накладные на выдачу», «Лекарства», «Список обследований». На основании данных в таблицах были созданы рабочие интерфейсы врача-терапевта. На рисунке 1.13 в Приложении А представлен фрагмент интерфейса «Анализы пациентов». Фрагмент интерфейса «Направления на обследования» представлен в Приложении А на рисунке 1.14. Также в Приложении А на рисунке 1.15 представлен фрагмент рабочего интерфейса «Форма выдачи препаратов».

4.2. Прототип программы (второй спринт)

Для второго спринта были определены следующие функции, необходимые для его реализации: разработка возможности редактирования и удаления назначенных пациенту анализов, рецептов на выдачу лекарственных препаратов, а также возможность редактировать и удалять ранее назначенные обследования. Для создания полноценной базы данных нужно создать схемы и связать необходимые таблицы с данными. На рисунке 4.3 представлена схема данных с применяемыми связями. Для чего, пользовательский интерфейс был доработан.

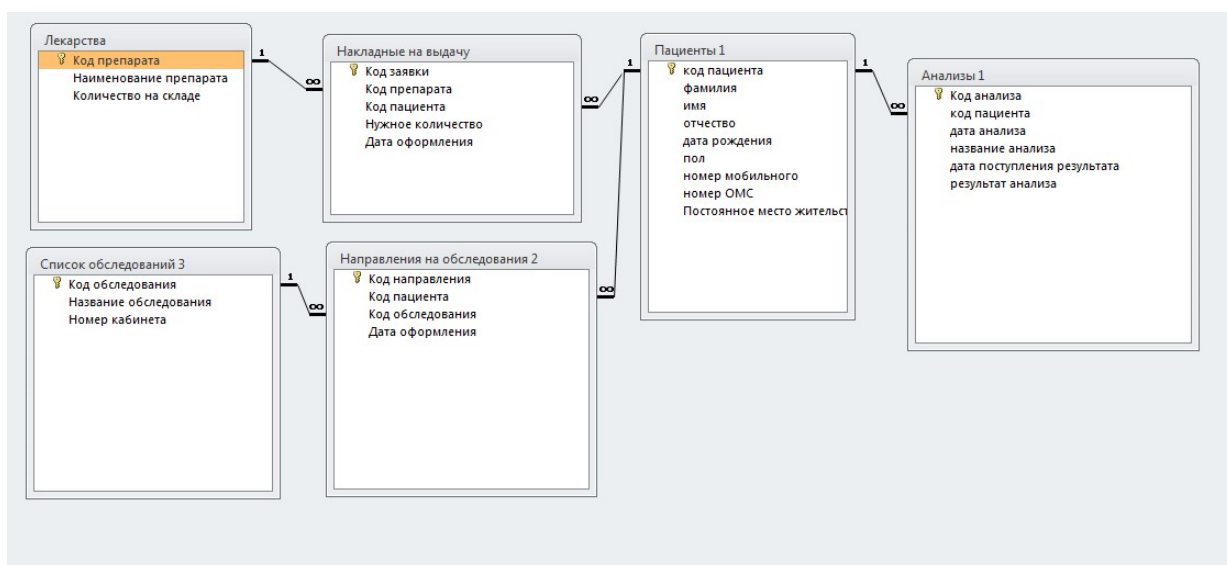


Рисунок 4.3 – Схема данных с применяемыми связями

4.3. Прототип программы (третий спринт)

В третьем спринте необходимо реализовать следующие возможности: отображение даты и времени назначения анализа, даты и времени поступления результатов анализа, возможность отображения результатов анализов пациента, использование компонентов программы, позволяющих вводить и проверять введенный пользователем пароль, разработка возможности создания отчётов и возможности отображения наличия на складе аптечного киоска больницы лекарственных препаратов. На рисунке 4.4 представлен фрагмент интерфейса «Анализы пациентов», с добавленными возможностями.

Анализы пациентов

Код пациента

1

Фамилия

Петров

Имя

Александр

Отчество

Григорьевич

Дата рождения

12.12.1965

Пол

М

Номер мобильного

12243243243244

Номер ОМС

1232432453543423

Постоянное место жительства

г. Москва, пр-т Вернадского, 78

Список анализов пациента

Код анализа	код пациента	дата анализа	название анализа	дата поступления результата	результат анализа
1	1	04.04.2019	анализ крови общий		показатели в норме
3	1	01.04.2019	биохимия		
4	1	08.04.2019	Общеклинический анализ мочи		
5	1	02.04.2019	Биохимический анализ крови		
6	1	07.04.2019	Кровь на гормоны		
*	(№)				

Записи: 1 из 5

Нет фильтра

Поиск

В начало

Пред. запись

Сл. запись

В конец

Поиск по пациентам

Сохранить изменения

Печать формы

Заккрыть

Рисунок 4.4 – Фрагмент интерфейса «Анализы пациентов» на 3 спринте

В завершении спринта разработан интерфейс программы с возможностью защиты данных от несанкционированного доступа путем задания пароля для входа в базу данных. Создана форма для ввода пароля. На рисунке 4.5 представлен фрагмент интерфейса.

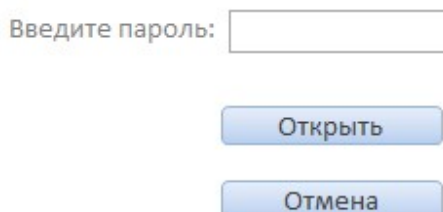


Рисунок 4.5 – *Фрагмент интерфейса для ввода пароля*

При вводе верного пароля пользователь попадает в главное меню, фрагмент интерфейса представлен на рисунке 4.6. При нажатии на кнопку «Данные пациентов» пользователь переходит в рабочий интерфейс для добавления, редактирования и удаления данных пациентов. По кнопке «Анализы пациентов» врач-терапевт попадает в рабочий интерфейс для создания, редактирования и удаления направлений на анализы. Кнопки «Направления на обследования» и «Аптечный киоск» открывают небольшие меню для дальнейшего перехода в рабочие интерфейсы. Кнопка «Выход» закрывает программу. В случае неверного ввода пароля система выдает предупреждение об этом. На рисунке 4.7 представлен фрагмент интерфейса.



Рисунок 4.6 – *Главное меню программы*

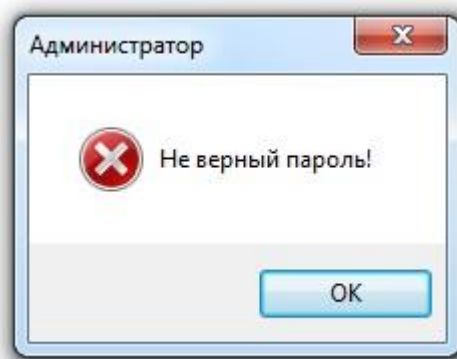


Рисунок 4.7 – Фрагмент интерфейса с ошибкой при вводе пароля

В рабочих интерфейсах «Анализы пациентов», «Направления на обследование» и «Форма выдачи препаратов» были реализованы 8 основных активных кнопок. Кнопки «В начало», «Предыдущая запись», «Следующая запись», «В конец» позволяют переключаться между пациентами. Кнопка «Поиск» позволяет искать информацию. Для добавления новых и редактирования уже существующих записей реализована специальная таблица в нижней части интерфейса. Также есть возможность вывести информацию на печать, нажатием соответствующей кнопки. Для выхода из программы предназначена кнопка «Выход».

Раздел 5. Тестирование приложения

Для выявления неточностей и ошибок в работе программного обеспечения используется процесс анализа продукта – его тестирование. Благодаря тестированию команда разработчиков может устранить существующие недостатки и улучшить продукт в дальнейшем. Выделяется три основных вида тестирования: функциональное, интеграционное и нагрузочное.

Функциональное тестирование в первую очередь используется для проверки того, что ПО обеспечивает именно тот результат, который требуется конечному его пользователю или бизнесу. Как правило, функциональное тестирование включает в себя оценку и сравнение каждой функции программного обеспечения с бизнес-требованиями. Кроме того, в ходе функционального тестирования проверяется программное обеспечение на удобство использования.

Интеграционное тестирование – это уровень тестирования программного обеспечения, в котором отдельные единицы объединяются, и тестируется как группа. Целью данного уровня тестирования является выявление ошибок во взаимодействии между интегрированными модулями.

Нагрузочное тестирование проводится для определения быстродействия системы. В связи с тем, что ключевой бизнес-процесс – учет анализа пациентов, где врач должен иметь возможность оперативно получать информацию о текущем состоянии результатов анализа пациента.

5.1. Функциональное тестирование

Для проведения функционального тестирования составляется список функциональных требований и в нем расписываются реализованные программные компоненты:

- отображение в таблице данных о пациентах – таблица с персональными данными пациентов;
- использование компонентов программы, обеспечивающие создание, редактирование и удаление данных о пациентах – форма с активными элементами и полями для добавления, редактирования и удаления данных о пациенте;
- использование компонентов программы, обеспечивающие поиск данных по ключевым параметрам – активные кнопки на форме, позволяющие выполнить поиск по заданным параметрам;
- использование компонентов программы, обеспечивающих возможность назначать анализы пациентам – форма с активными элементами и полями для добавления новых данных по анализам пациентов;
- использование компонентов программы, обеспечивающих возможность назначать обследование пациентам – форма с активными элементами и полями для добавления новых направлений на обследование;
- использование компонентов программы, обеспечивающих возможность создания рецептов на выдачу лекарственных препаратов пациентам – форма с активными элементами и полями для добавления новых рецептов на выдачу лекарственных препаратов;
- отображение выданных пациенту рецептов на лекарственные препараты – форма с активными элементами, позволяющая отобразить выданные рецепты по каждому пациенту;
- использование компонентов программы, обеспечивающих возможность редактировать и удалять: назначенный анализ, рецепт

на выдачу лекарственных препаратов, назначенное обследование – форма с активными элементами и полями для редактирования и удаления данных по каждому пациенту;

- использование компонентов программы, обеспечивающих поиск выданных пациенту рецептов на лекарственные препараты и назначенных обследований по заданным параметрам – активные кнопки на форме, позволяющие выполнять поиск по заданным параметрам;
- отображение даты и времени назначенного анализа в отдельной графе интерфейса программы – форма с активными элементами, отображающая дату и время назначенного анализа в отдельной графе интерфейса;
- отображение на экране данных о дате и времени поступления результатов анализа – отдельная графа в рабочей форме, отображающая дату и время поступления результатов анализа;
- отображение информации, содержащей результаты анализов пациента – отдельная графа в рабочей форме, отображающая результаты анализа пациента;
- использование программного кода, обеспечивающего защиту от несанкционированного доступа, реализация формы авторизации пользователя в системе – отдельная форма авторизации пользователя в системе при открытии базы данных;
- использование программного кода, позволяющего вводить и проверять введенный пользователем пароль – поле для ввода пароля на отдельной форме авторизации;

- использование компонентов программы, обеспечивающих возможность создания отчетов – активные кнопки на формах, позволяющие создать отчет;
- использование компонентов программы, позволяющих отображать наличие лекарственных препаратов на складе аптечного киоска больницы – форма с активными элементами, отображающая остаток на складе аптечного киоска больницы лекарственных препаратов.

На основании этого списка можно установить, что все заявленные функциональные требования были выполнены, программа работает, следовательно, функциональное тестирование успешно пройдено.

5.2. Нагрузочное тестирование

Нагрузочное тестирование служит для анализа производительности разработанного ПО. Рассмотрим время отклика ПО при работе с разным количеством записей в базе данных (1, 10, 50, 100). На рисунке 5.1 представлен снимок таблицы, сделанной в MS Excel. С помощью секундомера 5 раз была проведена проверка времени отклика системы на такие действия, как: запись и поиск. Затем с помощью встроенных функций Excel было рассчитано среднее время отклика и среднее квадратичное отклонение. Далее по формуле (5.1):

$$\Delta t = \sqrt{\left(\frac{\sigma}{\sqrt{n}} t_{a(n-1)}\right)^2 + A^2} \quad (5.1)$$

рассчитана погрешность измерений. Время отклика было рассчитано по формуле (5.2):

$$t_{\text{отк}} = t_{\text{ср.ар.}} \pm \Delta t \quad (5.2)$$

Количество записей	Действие	t1,с	t2,с	t3,с	t4,с	t5,с	Среднее время отклика, с	Средн. Квадр. Отклон.,с	Погрешность измерений, с	Время отклика, с
1	Запись	0,09	0,10	0,1	0,10	0,11	0,1040	0,0102	0,0066	0,104±0,006
	Поиск	0,11	0,1	0,1	0,09	0,10	0,1020	0,0117	0,0070	0,102±0,007
10	Запись	0,09	0,10	0,1	0,09	0,1	0,0980	0,0117	0,0070	0,098±0,007
	Поиск	0,10	0,10	0,1	0,12	0,1	0,1140	0,0120	0,0071	0,114±0,007
50	Запись	0,19	0,2	0,2	0,20	0,20	0,2000	0,0110	0,0068	0,200±0,006
	Поиск	0,20	0,20	0,2	0,19	0,20	0,1940	0,0080	0,0060	0,194±0,006
100	Запись	0,27	0,3	0,3	0,28	0,3	0,2860	0,0136	0,0076	0,286±0,007
	Поиск	0,29	0,30	0,3	0,31	0,30	0,3020	0,0075	0,0059	0,302±0,007

Рисунок 5.1 – Расчеты в рамках проведения нагрузочного тестирования

Как видно из результатов тестирования на рисунке 5.1, задержка незначительно возрастает при увеличении количества записей. Однако это не оказывает существенного влияния на стабильное функционирование программы и практически незаметно для пользователя. Можно сказать, что нагрузочное тестирование успешно пройдено.

Заключение

В данной работе была изучена гибкая методология Agile, в частности метод Scrum, его теоретические основы и ключевые особенности. Была обозначена структура работы, согласно требованиям методологии сформулированы пользовательские и функциональные требования, составлен бэклог продукта с расставленными приоритетами.

Для моделирования верхнего уровня ключевых бизнес-процессов применена методология ARIS (Architecture of Integrated Information Systems). Ключевые бизнес-процессы декомпозированы с помощью методологии DFD до 3-го уровня детализации. Описание процессов создано в моделях «As-Is» и «To-Be»; изучены теоретические основы моделей. Для большей наглядности и удобства внедрения улучшений была создана общая карта процессов. Для графического отображения процессов и создания карты использована программа MS Visio.

На основе подготовленных и проанализированных данных была создана программа в среде Microsoft Access. Согласно составленным ранее спринтам программа была доработана, улучшена и дополнена новыми возможностями. Затем разработанное приложение протестировано с помощью функционального и нагрузочного теста. Результаты теста показали стабильность и безошибочность работы программы.

Безусловно, созданное приложение в дальнейшем может быть доработано для улучшения графического интерфейса пользователя, для удобства работы врача, также функционал системы может быть расширен, исходя из меняющихся пользовательских требований.

Таким образом, цель данной работы была достигнута, все поставленные задачи выполнены полностью. Стоит отметить, что данная работа объединяет в себе большой объем информации по различным разделам экономики,

программирования и автоматизации. Анализ и применение этой информации на практике позволяет решать как существующие, так и перспективные проблемы.

Список использованных литературных источников

1. Учёба.ру. Определение врача-терапевта [Электронный ресурс]. – Режим доступа: URL: <https://www.ucheba.ru/prof/137> (дата обращения: 29.10.2018)
2. ЕМИАС.ИНФО. О проекте ЕМИАС [Электронный ресурс]. – Режим доступа: URL: <https://emias.info/about> (дата обращения: 29.10.2018)
3. User stories an Agile [Электронный ресурс]. – Режим доступа: URL: <https://www.mountaingoatsoftware.com/agile/user-stories> (дата обращения: 29.10.2018)
4. Habr. Техники определения приоритетов [Электронный ресурс]. – Режим доступа: URL: <https://habr.com/company/hygger/blog/351238/> (дата обращения: 29.10.2018)
5. Product development. What is Agile methodology [Электронный ресурс]. – Режим доступа: URL: <https://luis-goncalves.com/what-is-agile-methodology/> (дата обращения: 29.10.2018)
6. Маргарет Роуз. Бизнес процессы [Электронный ресурс]. – Режим доступа: URL: <https://searchcio.techtarget.com/definition/business-process> (дата обращения: 11.11.2018)
7. Uchebnik.Online. Методология ARIS [Электронный ресурс]. – Режим доступа: URL: <http://uchebnik.online/biznes-protsessov-modelirovanie/metodologiya-aris-54083.html> (дата обращения: 11.11.2018)
8. Степанов Д. Ю. Анализ, проектирование и разработка корпоративных информационных систем: уровень бизнес-процессов / МИРЭА. - М., 2017 [Электронный ресурс]. – Режим доступа: URL: https://stepanovd.com/documents/training/1_erp/7-processlevel/processlevel.pdf (дата обращения: 11.11.2018)

9. What is Agile. What is Scrum? [Электронный ресурс]. – Режим доступа: URL: <https://www.cprime.com/resources/what-is-agile-what-is-scrum> (дата обращения: 29.11.2018)
10. Charles Sturt University. ARIS Standards and Conventions Manual [Электронный ресурс]. – Режим доступа: URL: https://cdn.csu.edu.au/__data/assets/pdf_file/0020/1314173/CSU_ARIS_Modelling_Standards_and_Conventions_Manual_V1_0.pdf (дата обращения: 25.11.2018)
11. Lucidchart. What is a Data Flow Diagram [Электронный ресурс]. – Режим доступа: URL: <https://www.lucidchart.com/pages/data-flow-diagram> (дата обращения: 17.11.2018)
12. ARIS community. Which model to use? [Электронный ресурс]. – Режим доступа: URL: <https://www.ariscommunity.com/users/shameem/2011-01-19-which-model-use>(дата обращения: 17.11.2018)
13. Е.М. Карчевский, И.Е. Филиппов, И.А. Филиппова. Access 2010 в примерах. Учебное пособие. Казанский университет. Казань. 2012. [Электронный ресурс]. – Режим доступа: URL: https://kpfu.ru/docs/F1448756111/Access_2010.pdf (дата обращения: 21.04.2019)
14. Официальный сайт Российской Государственной Библиотеки [Электронный ресурс]. – Режим доступа: URL: <https://www.rsl.ru/> (дата обращения: 26.11.2018)

ПРИЛОЖЕНИЕ А

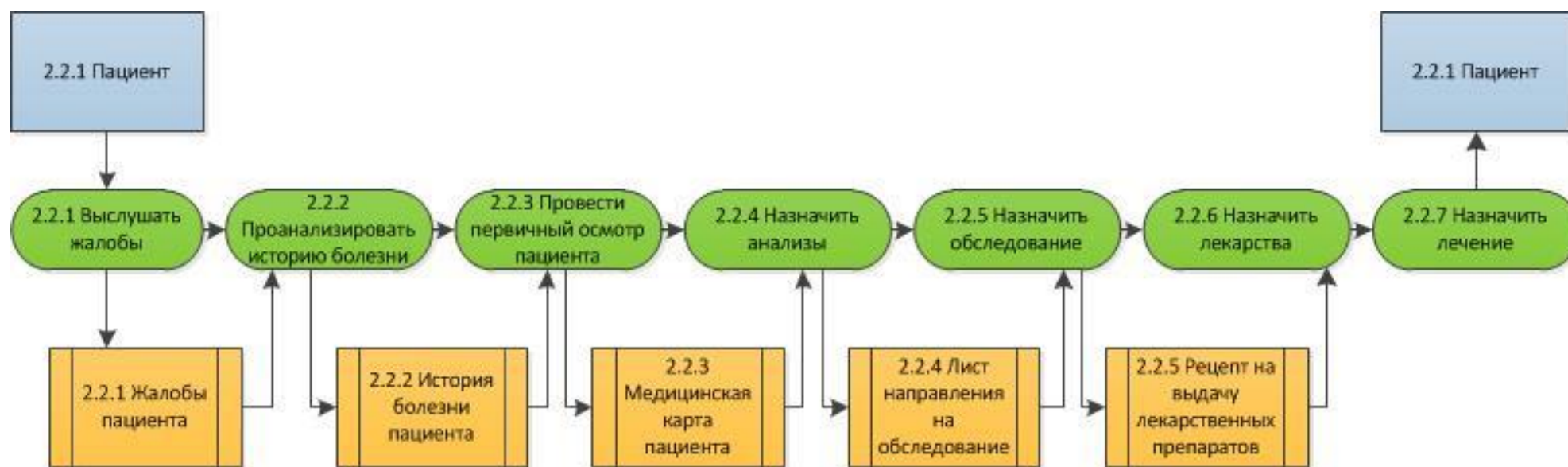


Рисунок 1.1 – Описание процесса «Лечить пациента» 2 уровня As-Is в нотации DFD

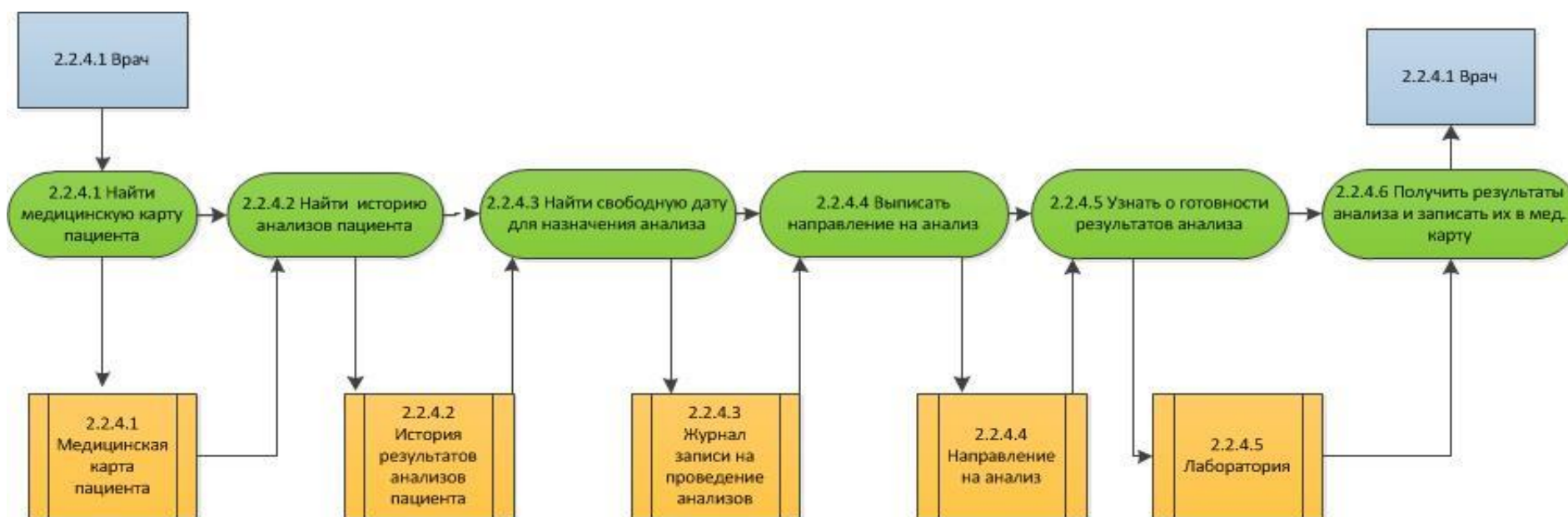


Рисунок 1.2 – Описание процесса «Назначить анализы» 3 уровня As-Is в нотации DFD



Рисунок 1.3 – Описание процесса «Назначить обследование» 3 уровня As-Is в нотации DFD



Рисунок 1.4 – Описание процесса «Назначить лекарства» 3 уровня As-Is в нотации DFD

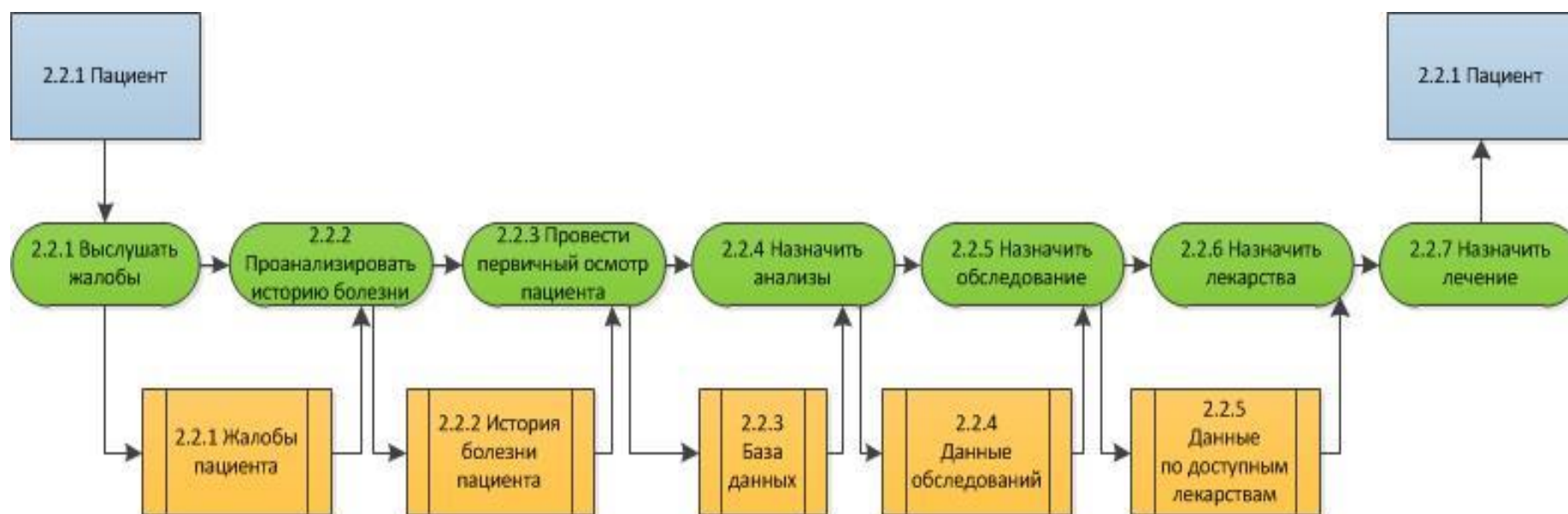


Рисунок 1.5 – Описание процесса «Лечить пациента» 2 уровня To-Be в нотации DFD



Рисунок 1.6 – Описание процесса «Назначить анализы» 3 уровня To-Be в нотации DFD

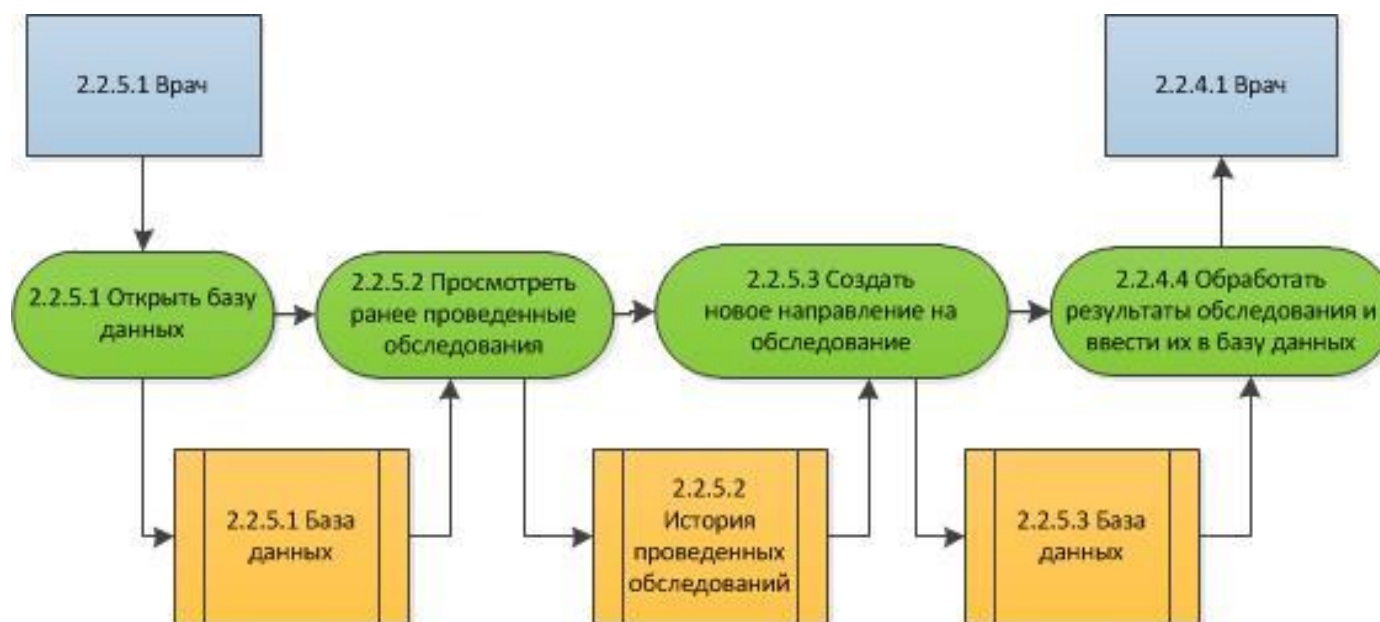


Рисунок 1.7 – Описание процесса «Назначить обследование» 3 уровня To-Be в нотации DFD

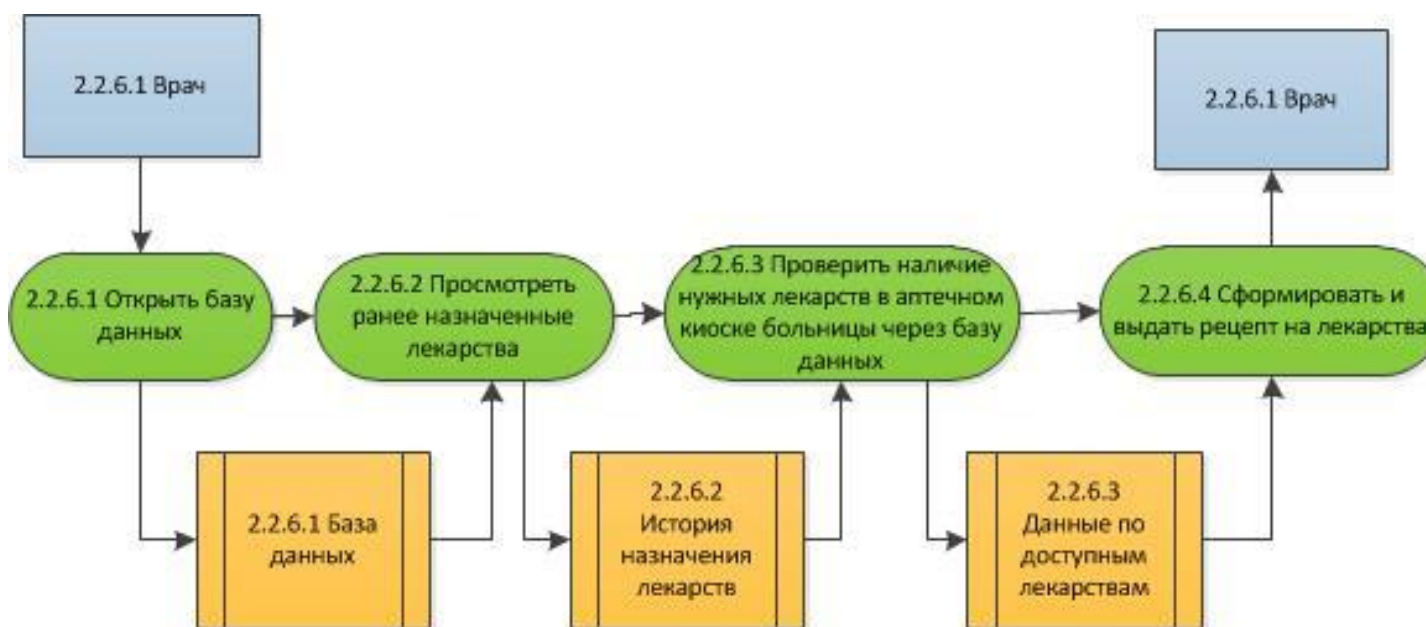


Рисунок 1.8 – Описание процесса «Назначить лекарства» 3 уровня To-Be в нотации DFD

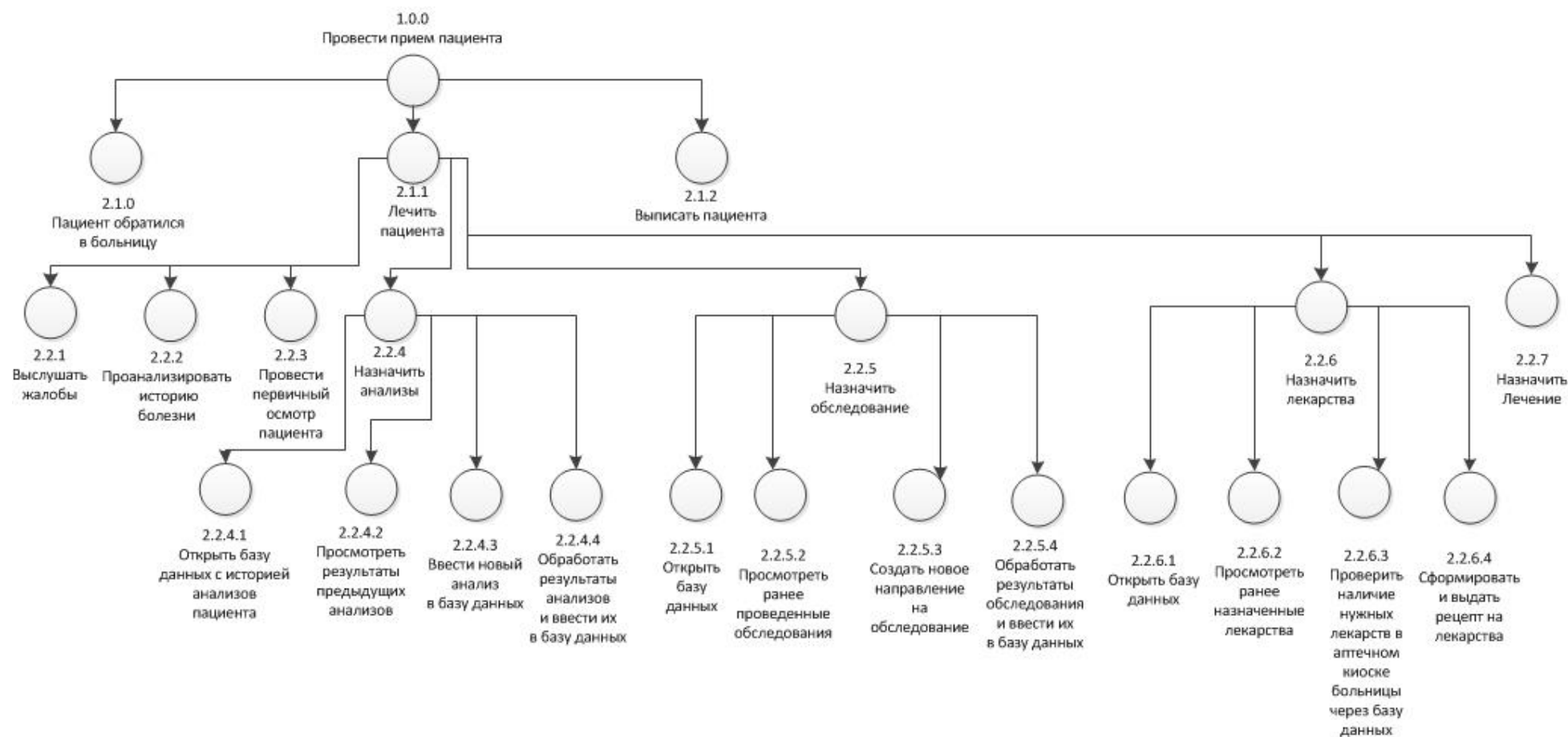


Рисунок 1.9 – Карта процессов, описывающая прием пациента в модели To-Be

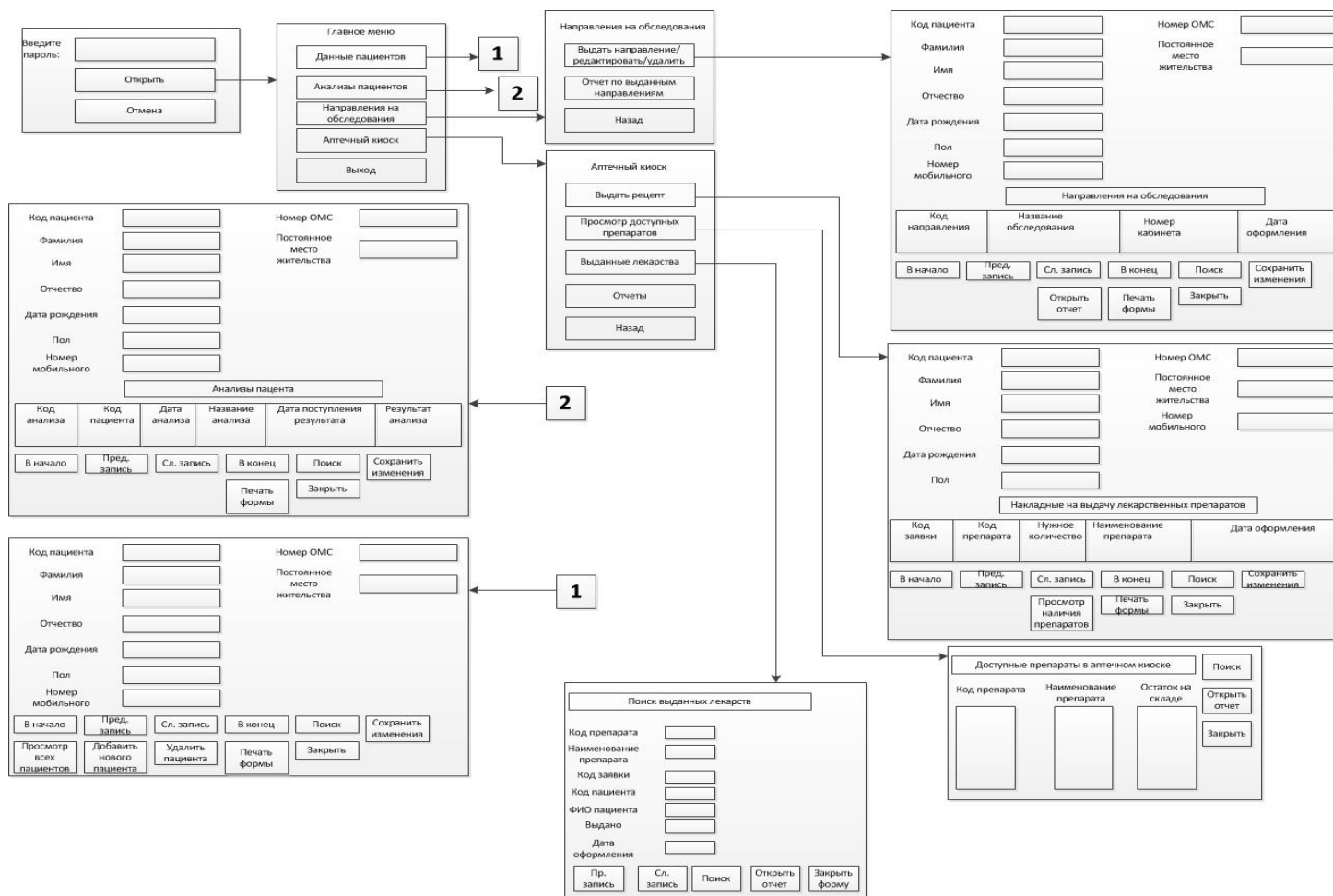


Рисунок 1.10 – Схема приложения

Код пациента	<input type="text"/>	Номер ОМС	<input type="text"/>
Фамилия	<input type="text"/>	Постоянное место жительства	<input type="text"/>
Имя	<input type="text"/>		
Отчество	<input type="text"/>		
Дата рождения	<input type="text"/>		
Пол	<input type="text"/>		
Номер мобильного	<input type="text"/>		
<input type="text" value="Направления на обследования"/>			
Код направления	Название обследования	Номер кабинета	Дата оформления
<input type="button" value="В начало"/>	<input type="button" value="Пред. запись"/>	<input type="button" value="Сл. запись"/>	<input type="button" value="В конец"/>
<input type="button" value="Открыть отчет"/>		<input type="button" value="Печать формы"/>	<input type="button" value="Закреть"/>
			<input type="button" value="Поиск"/>
			<input type="button" value="Сохранить изменения"/>

Рисунок 1.11 – Схема рабочей формы для выдачи, редактирования и удаления направлений на обследования

Код пациента	<input type="text"/>	Номер ОМС	<input type="text"/>
Фамилия	<input type="text"/>	Постоянное место жительства	<input type="text"/>
Имя	<input type="text"/>	Номер мобильного	<input type="text"/>
Отчество	<input type="text"/>		
Дата рождения	<input type="text"/>		
Пол	<input type="text"/>		

Накладные на выдачу лекарственных препаратов

Код заявки	Код препарата	Нужное количество	Наименование препарата	Дата оформления
------------	---------------	-------------------	------------------------	-----------------

В начало

Пред. запись

Сл. запись

В конец

Поиск

Сохранить изменения

Просмотр наличия препаратов

Печать формы

Заккрыть

Рисунок 1.12 – Схема рабочей формы для выдачи, редактирования и удаления рецептов на лекарственные препараты

Анализы пациентов

Код пациента: 1

Фамилия: Петров

Имя: Александр

Отчество: Григорьевич

Дата рождения: 12.12.1965

Пол: М

Номер мобильного: 12243243244

Номер ОМС: 1232432453543423

Постоянное место жительства: г. Москва, пр-т Вернадского, 78

Список анализов пациента

Код анализа	код пациента	дата анализа	название анализа
1	1	04.04.2019	анализ крови общий
3	1	01.04.2019	биохимия
4	1	08.04.2019	Общеклинический анализ мочи
5	1	02.04.2019	Биохимический анализ крови
6	1	07.04.2019	Кровь на гормоны
* (№)			

Записи: 14 1 из 5

В начало Пред. запись Сл. запись В конец

Поиск по пациентам Сохранить изменения Печать формы Заккрыть

Рисунок 1.13 – Фрагмент рабочего интерфейса «Анализы пациентов» на 1 спринте

Рисунок 1.14 – Фрагмент рабочего интерфейса «Направления на обследования» на 1 спринте

Код пациента	1	Постоянное место жительства	г. Москва, пр-т Вернадского, 78
Фамилия	Петров	Номер ОМС	1232432453543423
Имя	Александр	Номер мобильного	12243243243244
Отчество	Григорьевич		
Дата рождения	12.12.1965		
Пол	М		

[illegible]

[В начало](#)
[Пред. запись](#)
[Сл. запись](#)
[В конец](#)
[Поиск по пациентам](#)
[Сохранить изменения](#)
[Просмотр наличия препаратов](#)
[Печать формы](#)
[Закрыть](#)

Рисунок 1.15 – Фрагмент рабочего интерфейса «Форма выдачи препаратов» на 1 спринте