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:

«АВТОМАТИЗАЦИЯ КЛЮЧЕВЫХ БИЗНЕС-ПРОЦЕССОВ СТОМАТОЛОГИЧЕСКОЙ КЛИНИКИ С ИСПОЛЬЗОВАНИЕМ СПИРАЛЕВИДНОЙ МОДЕЛИ»

Студент:
Худяков С.Д.

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

Москва – 2019

ОГЛАВЛЕНИЕ

Введение

С изобретением компьютера человек старается заменить бытовые функции и действия автоматизированными системами, так как во многих ситуациях это является более выигрышным вариантом в силу ряда преимуществ. Например, таких как сокращение времени затрачиваемого на обработку данных, на циклические процессы. Применение компьютерных технологий позволяет снизить фактор «человеческой ошибки» в силу усталости человека, невнимательности, недостаточности наглядности предоставляемых данных и др.

Для любого автоматизированного процесса необходимо программное обеспечение (ПО), которое обеспечивает работу необходимых функций и обслуживания процесса, имеет возможность его дальнейшего усовершенствования, доработки, внесения изменений и др.

В общем комплексе автоматизации жизненных сфер и процессов отдельным классом можно выделить автоматизацию в сфере здравоохранения – автоматизацию деятельности медицинских лечебных учреждений. В целом, это заключается в переводе бумажных документов в цифровой формат и создании базы данных с возможностью фиксирования процесса лечения пациента в электронном виде. При этом сама деятельность различных медицинских учреждений отличается друг от друга, что делает невозможным создание универсального ПО.

В данной работе будет произведена автоматизация деятельности стоматологической клиники. Актуальность выбранного направления объясняется ежегодно увеличивающимся количеством стоматологических клиник и не снижающимся количеством обращений в них, что приводит к росту очередей и высокой нагрузке менеджеров и медицинского персонала

стоматологии. Современные рабочие места врачей-стоматологов оборудованы персональным компьютером, что даёт теоретическую возможность внедрить написанный продукт, тем самым упростить работу врача и сделать для него предоставляемую информацию наглядной и быстро доступной по сравнению с использованием в работе бумажных носителей.

В настоящее время на рынке программного обеспечения имеется ряд продуктов [1], решающих вопрос автоматизации деятельности стоматологии. Однако их отличает сложность внедрения, требовательность к вычислительной мощности установленных персональных компьютеров (ПК), зачастую высокая стоимость, а также отсутствие возможности доработки с учетом особенностей работы конкретной клиники.

Раздел 1. Цель и задачи практической работы

Целью данной работы является написание программного обеспечения, которое позволит осуществить автоматизацию деятельности стоматологической клиники по всем видам её деятельности, а также обеспечить врачу-стоматологу доступ к информации о ходе лечения конкретного пациента непосредственно на рабочем месте у стоматологического кресла. Для реализации данного проекта необходимо выполнить следующие задачи:

- Выполнить детальный анализ спиралевидной модели.
- Провести анализ предметной области.
- Провести опрос стейкхолдеров.
- Определить бизнес-процессы стоматологии и выявить из них ключевой бизнес-процесс.
- Составить список требований.
- Провести ранжирование требований по их значимости.
- Проектирование процессов и оргструктуры в моделях AS-IS и TO-BE нотации ARIS VACD и eEPC до 3-4 уровней детализации.
- Определение очерёдности выполнения требований по спиралям.
- Разработка ПО обеспечения (для каждого витка спирали):
    - моделирование разрабатываемых пользовательских интерфейсов;
    - проектирование структуры данных и нормализация таблиц данных;
    - реализация операции ключевого процесса в среде MS Access;
    - тестирование и количественная оценка результатов тестирования;
    - качественный и количественный анализ рисков.

Раздел 2. Детальный анализ спиралевидной методологии внедрения систем

2.1. Понятие жизненного цикла

Процесс создания ПО является сложным из-за количества необходимых действий. ПО претерпевает ряд процессов, которые в совокупности называются жизненным циклом [2]. На данный момент существует множество моделей ЖЦ, позволяющих наладить и структурировать процесс создания ПО от начала задумки до этапа ввода в эксплуатацию. Все эти модели схожи по наличию следующих действий [3]:

- разработку требований или технического задания;
- разработку системы или технического проекта;
- программирование или рабочее проектирование;
- пробную эксплуатацию;
- сопровождение и улучшение;
- снятие с эксплуатации.

При выборе метода ЖЦ проекта необходимо учитывать масштабность проекта и возможности заказчика. Так же важным фактором является принятие решения о необходимости учёта или не учёта возможных рисков в процессе реализации проекта. В данной работе принято решение использовать спиралевидную модель ЖЦ проекта, что аргументируется следующим:

- необходима возможность вносить изменения в продукт в ходе его реализации;
- данное ПО создаётся для частной стоматологии, в силу чего необходимо учесть возникновение возможных рисков и их значимость во время проекта;
- необходимо иметь промежуточные результаты продукта.

2.2. Общее описание принципа спиралевидной модели

Суть спиралевидной модели заключается в том, что процесс создания итогового продукта можно условно разбить на 4 квадранта, через которые будет проходить каждый виток спирали, количество которых определяется в ходе жизненного цикла самого проекта, в зависимости от удовлетворения требований заказчика. Данными квадрантами являются (рисунок 2.1):

- Формирование цели и требований.
- Оценка и расчёт рисков.
- Конструирование (разработка, кодирование, тестирование).
- Оценка заказчика (внедрение и сопровождение).

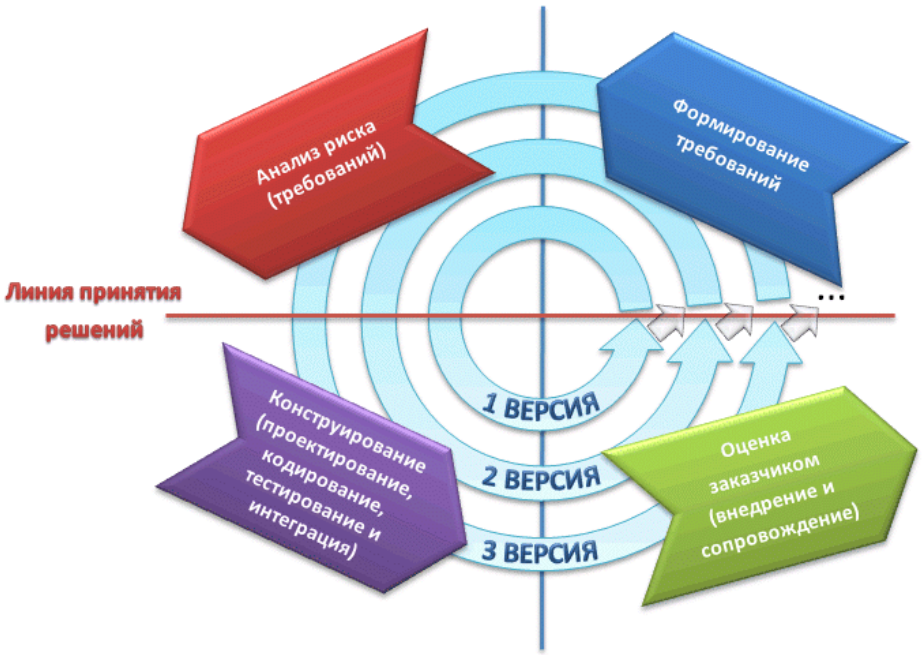


Рисунок 2.1 – *Спиралевидная модель*

Отличительной чертой от других моделей является возможность принятия решения о завершении или прекращении разработки проекта не только в момент готового продукта (в данном случае в конце витка спирали), но и также после оценки рисков, что способствует сохранению времени и ресурсов.

Каждый виток спирали в частности может быть рассмотрен как каскадная или V-образная модель, при чём по завершению витка имеется билд с готовым функционалом, который наращивается в следующем витке. Рассмотрим подробнее действия на каждом из квадрантов.

Квадрант 1: формирование цели и требований. На каждом витке фиксируются известные на данный момент времени требования от заказчика, которые должны быть осуществлены на данном витке. Например, такие как что должен выполнить продукт функционально и что реализуем в первую очередь из этого; какие бизнес-цели преследуются заказчиком (определяем ключевой бизнес-процесс на данный виток); какие имеются ограничения по техническим и бизнес-ресурсам; что предлагают конкуренты и какие существуют на данный момент альтернативы. Так же определяются все аспекты условий выполнения проекта. На последующих стадиях на данном витке заказчик может внести свои корректировки и новые предложения.

Квадрант 2: оценка и расчёт рисков. На данном этапе оценивается вся полученная информация из первого квадранта и используется метод оценки рисков Боэма или другой метод, более подходящей для конкретного проекта (модели оценки трудоемкости разработки программных систем, утвержденные Госкомтруда в 1986 году) [4].

Проводится два анализа рисков: качественный и количественный. Качественный анализ рисков позволяет выявить и распознать возможные виды рисков, а также причины и факторы, влияющие на уровень данного вида риска и при этом рассчитать численно-экономические вероятности и возможные потери ресурсов.

Главной задачей разработчиков является выявления всех возможных рисков, которым после присваивается определённый уровень приоритета на

основе их значимости с последующей разработкой стратегии по их преодолению. После оценки рисков проект-менеджер или руководитель проекта принимает одно из следующих решений:

- создание прототипа и переход к следующему этапу;
- остановка дальнейшего усовершенствования продукта и передача заказчику имеющейся работы (начиная со второго витка спирали);
- возврат к предыдущей стадии разработки проекта;
- прекращение разработки в силу неоправданности проекта.

При принятии решения и развитии продукта создаётся прототип, который минимизирует затраты средств и времени, а также позволяет нагляднее понять необходимые действия на этапе разработки.

Квадрант 3: конструирование (разработка, кодирование, интегрирование, тестирование). Происходит непосредственная разработка и написание кода продукта, в том числе и пользовательского интерфейса. При первой итерации создаётся концепция продукта (Proof Of Concept) [5], необходимая для первоначальной оценки заказчиком, после чего при последующих витках спирали создаются билды (builds) – готовые версии продукта, которые позволяют заказчику видеть ход работы над проектом и точнее формировать требования, дополнения и необходимые исправления билдов на пути к готовому продукту. По завершении создания производится тестирование билда и исправление ошибок.

Квадрант 4: Оценка заказчика (внедрение и сопровождение). Последний этап витка спирали заключает в себе разворачивание и внедрение билда или же уже конечного продукта с возможностью его сопровождения и усовершенствования. При необходимости дальнейшего усовершенствования

функционала, проходит анализ выполненной работы и переход на новый виток спирали или же откат назад для исправления недоработок и недочётов.

Следует отметить, что спиральная модель является гибкой, что позволяет производить как прямое «движение» по спирали, так и возврат к предыдущим этапам. Всё сводиться к минимизации риска траты трудовых и временных ресурсов на этапе «конструирования», для этого и проводится тщательный анализ требований заказчика с выполнением не всей работы сразу, а небольшим продуктами (билдами), что способствует быстрой обратной связи с заказчиком.

2.3. Преимущества и недостатки

Спиральная модель обладает следующими преимуществами:

- Возможность заказчику «видеть» развитие системы на ранних стадиях.
- Частая обратная связь с заказчиком в процессе всего жизненного цикла проекта.
- Изменения могут быть внесены на поздних стадиях разработки.
- Учёт и проработка рисков до того, как они могут настать.
- Большой потенциальный объём над проектом «разбивается» на малые задачи, которые оцениваются и воплощаются, при этом с учётом возможных последующих рисков.
- Наличие преимущества инкрементной модели: билды различных версий продукта.
- Высокий уровень административного управления, так как в конце каждой итерации проводиться анализ проведённых работ и расчёт необходимости дальнейшего продолжения проекта.
- Повышается предсказуемость поведения система из-за частой корректировки поставленных требований к продукту.

Спиральная модель обладает следующими недостатками:

- Отсутствие полного списка желаемого функционала от заказчика.
- Если проект не является заведомо с низким уровнем риска, то постоянный контроль этих рисков после каждой итерации делает его дорогостоящим.
- Усложненная структура модели может вызвать трудности в организации связи между разработчиками, менеджером и заказчиком.
- Необходимость высокой квалификации у персонала, просчитывающего риски.
- Отсутствие конкретных сроков окончания проекта в силу неизвестного числа витков спирали.
- Большое число внутренних операций ведёт к росту количества документации.

2.4. Последовательность шагов реализации приложения

Спиралевидная модель разработки приложения, как было описано выше, состоит из спиралей, которые с каждым разом усовершенствует разработанный билд. Количество спиралей будет уточнено после формирования списка требований, анализа их сложности и необходимости выполнения. На данный момент планируется использовать 4 спирали, которые будут содержать примерно следующие этапы разработки (таблица 2.1).

Таблица 2.1 – *Предварительное содержание витков спирали*

Этап спирали	Номер спирали	Предполагаемые действия
Поиск решения задач, поставленных в требованиях	I	Общая структура
	II	Взаимосвязи данных
	III	Интерфейс
	IV	Доработка дополнительных функций и дизайна приложения
Оценка возможных рисков и их влияния	I, II, III, IV	Рассмотрение рисков и вариантов взаимодействия с ними

Этап спирали	Номер спирали	Предполагаемые действия
Разработка и кодирование этой части проекта	I, II, III, IV	Реализация требований в программной среде Microsoft Access 2016
Тестирование приложения	IV	Проведение тестирования приложения

Раздел 3. Идентификация требований и формирование списка требований

3.1. Анализ требований

Требования – это описание тех пожеланий, которые необходимо выполнить при реализации продукта. Требования лишь описывают желаемый результат, но никак не учитывают технический аспект осуществления проекта [6]. Для упрощения работы, требования должны соответствовать следующим критериям:

- однозначность;
- непротиворечивость друг другу;
- понятность;
- корректность.

Требования заказчика к продукту фиксируется исходя из опроса-интервью. Для корректности и точности собранной информации необходимо выполнить следующие 4 этапа [6]:

- Выбор стейкхолдеров – необходимо определить пользователей системы, которые точно или теоретически будут использовать систему, чтобы учесть запрос как можно большего числа заинтересованных лиц в продукте.
- Сбор требований – непосредственное общение со стейкхолдерами, проводиться анализ предметной области. Определяется, что именно желают видеть в программе и какие функции она должна выполнять. Результатом данного этапа будет продукт, который будет отвечать ожиданиям заказчика. При этом учитывается специфика внедрения продукта и его дальнейшее предполагаемое обслуживание.

- Анализ требований – проверка требований на понятность, однозначность, полноту и прочие свойства требований, а также поиск взаимосвязей между ними. При этом требования классифицируются по значимости, что в дальнейшем влияет на очередность их выполнения.
- Документирование требований – фиксация всех требований в выбранном формате для дальнейшего использования, чтобы была возможность отслеживать ход реализации проекта, а также утверждение этого документа заказчиком.

Стейкхолдарами выступили директор стоматологии и старшая медсестра. Посредством методики интервьюирования от них были получены следующие пользовательские требования.

3.2. Пользовательские требования

Пользовательские требования – это те требования, которые описывают задачи и цели, которые пользователям даст разрабатываемое ПО, то есть что клиенты/заказчики смогут делать при помощи данной системы. В ходе опроса стейкхолдеров были выявлены следующие требования:

- Наличие личной карточки пациента с основными данными:
  - ФИО;
  - дата рождения;
  - возраст;
  - пол;
  - номер ОМС;
  - паспортные данные (серия, номер, кем выдан);
  - номер телефона.
- Описание всего лечения пациента:
  - перечень сопутствующих заболеваний;

- анамнез пациента;
- фиксация результата осмотра ротовой полости в «зубной карте»;
- описание проведённого лечения;
- возможность добавления рентген-снимков.

- Возможность описания состояния зуба:
  - состояние зуба;
  - проведённые на нём процедуры;
  - использованные препараты;
  - отметка о необходимости дальнейшего лечения.
- Информация о специалистах стоматологии:
  - ФИО специалиста;
  - специальность;
  - рабочий телефон.
- Информация об оказываемых в стоматологии услугах:
  - наименование услуги;
  - стоимость услуги.
- Возможность вывода следующей информации:
  - личной карточки пациента;
  - «зубной карты» определённого пациента;
  - внесённой информации по зубу пациента;
  - списка всех специалистов;
  - перечня услуг и их стоимости.
- Возможность вносить в программу информацию о новом пациенте и описывать детали его лечения.
- Отдельный доступ для:

- врача, с возможностью внесения и изменения информации о проведенном лечении;
- пациента только просмотр описания проведённого лечения.

- «Зубная карта» схематично должна показывать ротовую область.
- При запросе информация не должна быть показана вся сразу.
- Незагруженный простой интерфейс.

3.3. Функциональные требования

Функциональные требования описывают функции, которые должны выполняться в ПО для реализации персональных требованиях. Вследствие этого имеем перечень необходимых элементов, а именно:

- База данных должна содержать следующие таблицы:
  - «Пациент» (с содержанием полей «ФИО», «Дата рождения», «Возраст», «Пол», «Номер ОМС», «Паспортные данные», «Телефон»).
  - «Анамнез» (с содержанием полей «Дата посещения», «Описание»).
  - «Сопутствующие заболевания и противопоказания» (с содержанием полей «Соп.заболевания» и «Противопоказания»).
  - «Карта зубов» (с содержанием графы «номер зуба»).
  - «Описание зуба» (с содержанием полей «Состояние», «Процедуры», «Исп. препараты», «Завершённость лечения»).
  - «Персонал» (с содержанием графов «ФИО», «специализация»).
  - «Услуги» (с содержанием графов «Название», «Стоимость»).
- Обеспечение возможности вывода на экран:
  - информации из таблицы «Пациенты»;
  - информации из таблицы «Зубная карта»;
  - информации из таблицы «Описание зуба»;
  - информации из таблицы «Персонал»;

- информации из таблицы «Услуги».

- Возможность вносить в программу информацию о новом пациенте и описывать детали его лечения.
- Реализация отдельного доступа:
  - для врача, с возможностью внесения и изменения информации о проведенном лечении;
  - для пациента только просмотр описания проведённого лечения.
- Информация показывается в свёрнутых списках.
- Каждый зуб является отдельной кнопкой, нажатие на которую выводит информацию о выбранном зубе из таблицы «Описание зуба».
- Незагруженный простой интерфейс.

3.4. Приоритезация требований и матрица их отслеживания

В процессе опроса стейкхолдера был зафиксирован ряд требований. Для реализации проекта составляется матрица отслеживания этих требований, в которой происходит сопоставление персональных требований функциональным. На данном этапе также происходит приоритезация требований с целью выявление первостепенных задач, с последующим определением порядка их выполнения. Наиболее используемыми методами приоритезации требований на данный момент являются следующие методы [8]:

- Метод Кано – позволяет определить удовлетворённость клиентов функциями программного обеспечения.
- Метод MoSCoW.
- Метод QFD (Quality Function Deployment) – сопоставление желаний клиента и компании.
- User Story Mapping – построение карты пользовательских событий использования.

- Lean Prioritization с универсальной матрицей Value and Effort (график ценности и усилий требований).

Наиболее наглядным и подходящим для нашей цели является метод MoSCoW, суть которого заключается в присваивание к каждому требованию одной из четырёх категорий [8]:

- Must – наиболее важное и срочное.
- Should – важное.
- Could – может быть отложено на некоторое время.
- Would – не является приоритетным вообще (возможно осуществление в следующем релизе).

Совокупность всех требований и их приоритетов находится в табл. 3.1.

Таблица 3.1 – *Матрица отслеживания и приоритетов требований*

№	Пользовательские требования	Функциональные требования	Компонент программы	Уровень приоритета
1	Наличие личной карточки пациента с основными данными	Таблица «Пациент» с необходимыми полями для внесения данных из анкеты и документов	Программа по ведению данных о пациенте	Must
2	Наличие перечня сопутствующих заболеваний и противопоказаний	Таблица «Сопутствующие заболевания» и таблица «Противопоказания»	Программа по ведению данных сопутствующих заболеваниях и противопоказаниях	Must
3	Наличие анамнеза пациента	Таблица «Анамнез» с полями для данных из опроса пациента с датой обращения	Программа по ведению данных об анамнезе пациента	Must
4	Возможность фиксация результатов осмотра ротовой полости в «зубной карте»	Таблицы «Карта зубов» и «Описание зуба»	Программа по ведению записи результатов осмотра	Must
5	Наличие информация о специалистах стоматологии	Таблица «Персонал», с необходимым данными о специалистах	Программа по ведению данных о специалистах стоматологии	Must

№	Пользовательские требования	Функциональные требования	Компонент программы	Уровень приоритета
6	Наличие информации об оказываемых в стоматологии услугах	Таблица «Услуги и цены» содержащая оказываемые процедуры и их стоимость	Программа по ведению данных об услугах	Should
7	Возможность записи и изменения информации в таблицах	Возможность редактирование полей программ	Средство для ввода информации	Should
8	Показ информации по таблицам	Вывод на экран запрашиваемой информации из таблиц	Программа по выводу на экран запрашиваемых таблиц	Should
9	При запросе информация не должна быть показана вся сразу	Диалоговая форма и разворачивающиеся списки	Диалоговая форма и разворачивающиеся списки	Should
10	Возможность хранения рентген снимков	Таблица «Рентген снимки» с сохранёнными фотографиями	Средство работы с ссылками на изображения	Could
11	Возможность отдельного доступа для врача и пациента с различным уровнем доступа к просмотру и изменению информации	Возможность авторизации пользователей	Программное разделение информации по уровню доступа	Could

3.5. Последовательность шагов спиралей при разработке приложения

Для реализации проекта с использованием спиралевидного метода, как уже было описано в разделе детального описания этого метода, необходимо распределить требования по виткам спирали в соответствии с их важностью. Тогда последовательность шагов реализации приложения примет вид:

- Предварительные действия:
  - Определение предметной области.
  - Опрос стейкхолдеров с фиксацией их требований (см. таблицу 3.1).
  - Разработка бизнес-модели с использованием нотаций ARIS VACD и eEPC.

- I виток спирали:
  - Поиск решения для реализации хранения информации в базе данных (пункты № 1-6 требований из таблицы 3.1).
  - Оценка возможных рисков и их влияния.
  - Разработка и кодирование этой части проекта.
  - Предоставление заказчику билда продукции и фиксация его оценки.
- II виток спирали:
  - Принятие во внимание оценки предыдущего билда от заказчика. Поиск решения для реализации работы с данными для пользователя (пункт № 7 из таблицы 3.1).
  - Оценка возможных рисков и их влияния.
  - Разработка и кодирование этой части проекта.
  - Предоставление заказчику билда продукции и фиксация его оценки.
- III виток спирали:
  - Принятие во внимание оценки предыдущего билда от заказчика. Поиск решения для реализации рабочего интерфейса (пункт № 8 из таблицы 3.1).
  - Оценка возможных рисков и их влияния.
  - Разработка и кодирование этой части проекта.
  - Предоставление заказчику билда продукции и фиксация его оценки.
- IV виток спирали:
  - Принятие во внимание оценки предыдущего билда от заказчика. Поиск решения для реализации дополнительных функций приложения и усовершенствование пользовательского интерфейса (пункты № 9-11 из таблицы 3.1).
  - Оценка возможных рисков и их влияния.

- Разработка, кодирование этой части проекта.
- Тестирование и отладка готового приложения.
- Предоставление заказчику готового продукта.

Раздел 4. Проектирование приложения

4.1. Проектирование ключевых бизнес-процессов

Для создания ПО обеспечения, целью которого является автоматизация определённых действий, необходимо понять и описать, как происходит работа объекта сейчас и как мы хотим, чтобы как это было. Для начала рассматривают бизнес-модель объекта, которая в свою очередь состоит из отдельных бизнес-процессов. Бизнес-модель – это совокупность бизнес-процессов и их взаимосвязей между собой, показывающая внутреннюю деятельность объекта в целом, направленная на достижение цели работы компании (в нашем случае оказание медицинских услуг в стоматологии) [8].

Бизнес-процесс – непосредственный набор действий (подпроцессов), который необходим для достижения конечной цели объекта. Задачей практической работы является автоматизация не всех протекающих в стоматологии бизнес-процессов, а лишь определённого, называемого ключевым бизнес-процессом.

Ключевым бизнес-процессом данной работы является фиксирование в электронном виде процесса лечения в формате «зубной карты». Для того, чтобы осуществить ключевой бизнес-процесс, рассмотрим деятельность стоматологии и проведём проектирование при помощи моделей «AS-IS» и «TO-BE» до 3-4 уровня, используя нотации ARIS VACD и eEPC.

Функциональная модель «AS-IS» описывает деятельность организации «как есть», то есть именно те действия, которые осуществляются на данный момент. Данная модель составляется из анализа существующих положений организации, опроса сотрудников о роде их деятельности, при непосредственном личном наблюдении эксперта за ходом работы. После построения модели «AS-IS» выявляются слабые места, действия, которые

можно ускорить за счёт их автоматизации и проводится реинжиниринг бизнеса – построение модели «TO-BE» (модель «как будет»). В этой модели учитываются необходимые изменения и тем самым описывается усовершенствованный бизнес-процесс [9]. Для проектирования процессов были выбраны две нотации ARIS VACD и ARIS eEPC.

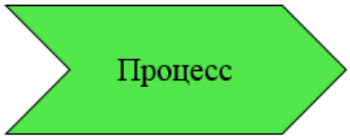
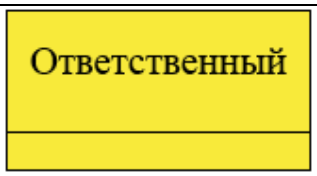
Методология Architecture of Integrated Information Systems (ARIS) – проектирование интегрированных информационных систем – на данный момент является одной из самых распространённых и широко используемых. Она содержит около 100 нотаций, позволяющих отразить процессы на разных уровнях детализации. В целом, методология ARIS делит процессы на четыре группы [10]:

- Группа «Оргструктура»: описание организационной структуры и внутренней инфраструктуры компании.
- Группа «Функции»: описание стратегических целей компании, функций и прочих элементов функциональной деятельности организации.
- Группа «Информация»: описание информации, которая используется в деятельности организации.
- Группа «Процессы»: описание взаимосвязей между структурой, функциями и информацией.

Для верхнего уровня проектирования будем использовать нотацию ARIS VACD (Value Added Chain), отличительной чертой которой является то, что информационные и материальные потоки на схеме отображаются объектами. При помощи данной нотации сможем отобразить логические связи между работами и их последовательность. Основное достоинство – это наличие обозначения для некоторой группы функций организации, которая служит для

получения добавленной ценности [11]. Используемые элементы в данной нотации [12] приведены в таблице 4.1.

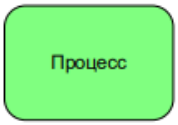
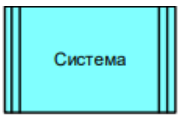
Таблица 4.1 – *Графические элементы нотации ARIS VACD*

Наименование	Описание	Графический элемент
Процесс	Обозначение процессов, происходящих в организации	
Объект	Обозначение переходов носителей с информацией	
Ответственный	Обозначение организационных взаимосвязей и полномочий, обеспечивающих выполнение процесса	

Для описания нижних уровней необходима нотация с большим количеством графических элементов, которые позволят детальнее отобразить происходящие бизнес-процессы в организации. Используем нотацию ARIS eEPC, отличительной чертой которой является наличие объектов «события», при помощи которого фиксируются факт, время или условие, инициирующих начало/окончание выполнения работ бизнес-процесса [11]. Используемые элементы в данной нотации [12] приведены в таблице 4.2.

Таблица 4.2 – *Графические элементы нотации ARIS eEPC*

Наименование	Описание	Графический элемент
Инициирующий/последующий процесс	Обозначение процесса, который влияет на данный этап	

Наименование	Описание	Графический элемент
Инициирующее/последующее событие	Обозначение события, которое оказывает влияние на бизнес-процесс(-ы)	
Процесс	Обозначение выполняемого процесса	
Ответственный	Обозначение организационных взаимосвязей и полномочий, обеспечивающих выполнение процесса	
Входящий/исходящий документ	Обозначение необходимых бумажных документов при выполнении функции	
Прикладная система	Обозначение используемой программы для осуществления процесса	
Логический элемент «И»	Ставится в том случае, если для осуществления функции необходима одновременная инициация нескольких событий	
Логический элемент «ИЛИ»	Ставится в том случае, если для осуществления функции необходимо выполнение хотя бы одного из событий	
Исключающий логический элемент «ИЛИ»	Ставится в том случае, если для осуществления функции необходимо одно и только одно из событий в зависимости от условий	

Используя выше описанные графические элементы, перейдём к созданию диаграмм, описывающих деятельность стоматологии. Диаграммы будут созданы в среде Business Studio 4.2.

4.2. Диаграммы бизнес-процессов в моделях «AS-IS» и «TO-BE»

Для создания приложения необходимо проанализировать работу стоматологии, а именно зафиксировать протекающие в ней процессы, а также их последовательность. Изучение деятельности будет происходить поэтапно с каждым разом увеличивая конкретику действий персонала.

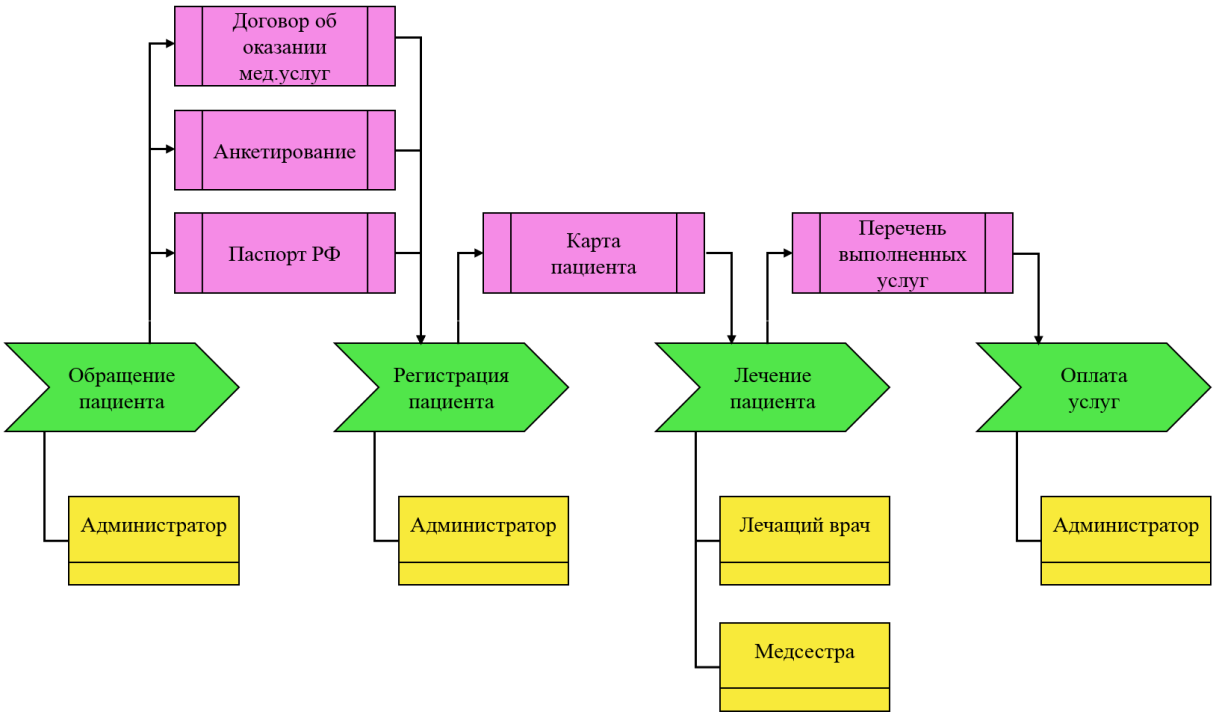


Рисунок 4.1 – *Первый уровень: процесс «Накопление информации»*

Первый уровень бизнес модели включает в себя 4 основных бизнес-процесса, а именно «Обращение пациента», «Регистрация пациента», «Лечение пациента», «Оплата услуг» (рисунок 4.1). Как можно заметить, что уже начиная с первого уровня идёт накопление информации, в частности о пациенте. Разрабатываемое приложение обеспечит удобный её поиск и предоставление в более удобном формате, по сравнению с бумажным носителем.

Для поставленной задачи необходимо провести декомпозицию процесса «Лечение пациента», так как на этом этапе происходит фиксирование процесса

самого лечения, а также его результатов. Таким образом, получим второй уровень бизнес-модели (рисунок 4.2).

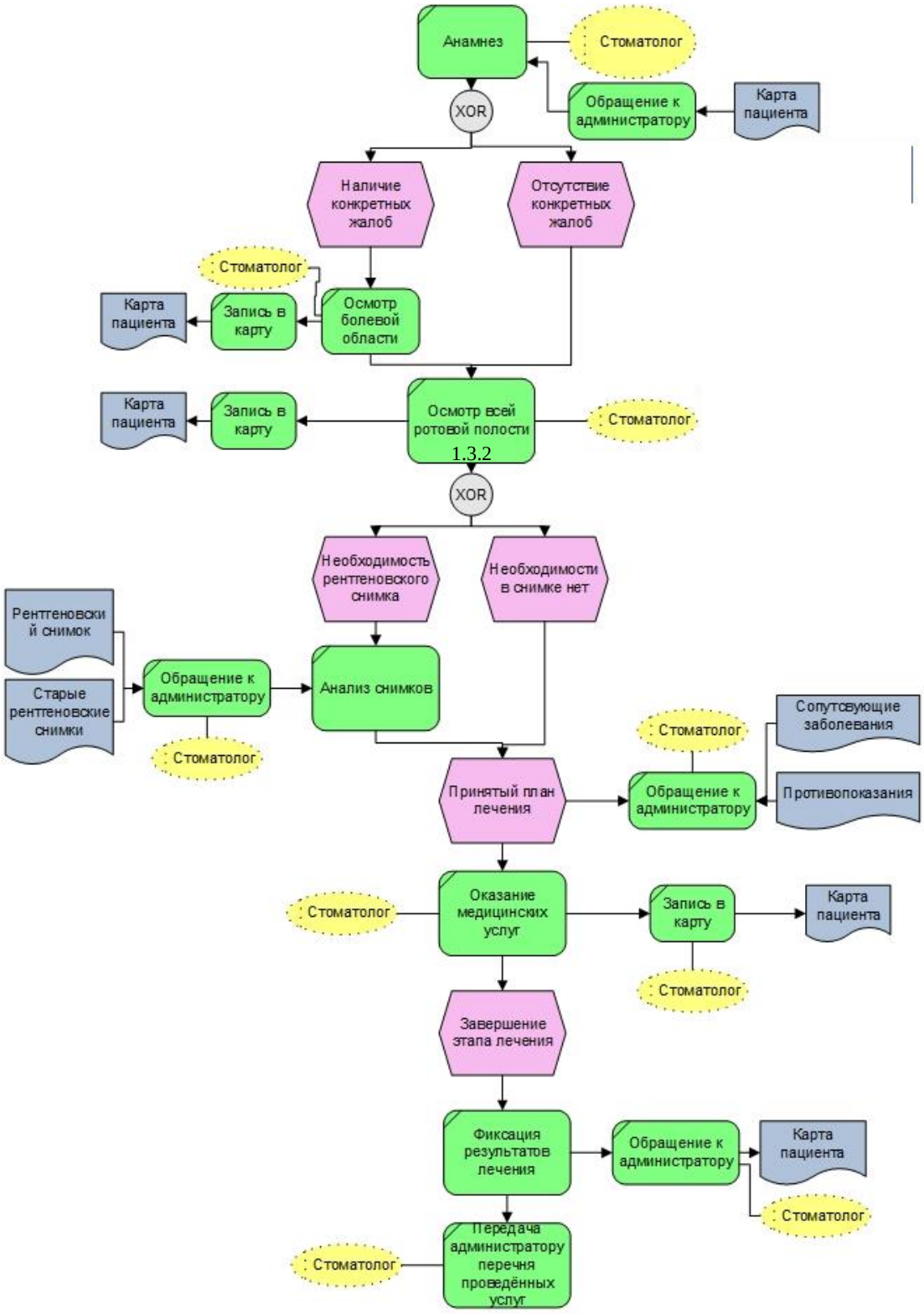


Рисунок 4.2 – *Второй уровень: процесс «Лечение пациента» в модели «AS-IS»*

Второй уровень отражает конкретику действий врача-стоматолога. Основным бизнес-процессами были отмечены процессы проведения «Анамнеза», «Осмотра ротовой полости», «Анализа рентген-снимков», «Оказания медицинских услуг», «Фиксации результатов лечения» и «Передача администратору перечня проведённых услуг». Достоинством нотации ARIS eEPC, как уже было отмечено выше, является отображения объекта, совершающего процесс, а также их событийность. Благодаря этому видно, что стоматолог в ряде случаев вынужден обращаться к администратору с целью получения необходимой информации по пациенту, что приводит к следующему:

- увеличивается время приёма за счёт вынужденных передвижений персонала;
- стоматолог не контролирует в это время процесс лечения;
- стоматолог не обладает актуальной информацией на момент начала лечения.

Разрабатываемое приложение устранит вышеперечисленные проблемы. Тогда после реинжиниринга деятельность стоматологии на этом уровне примет вид, отражённый в модели «TO-BE» на рисунке 4.3.

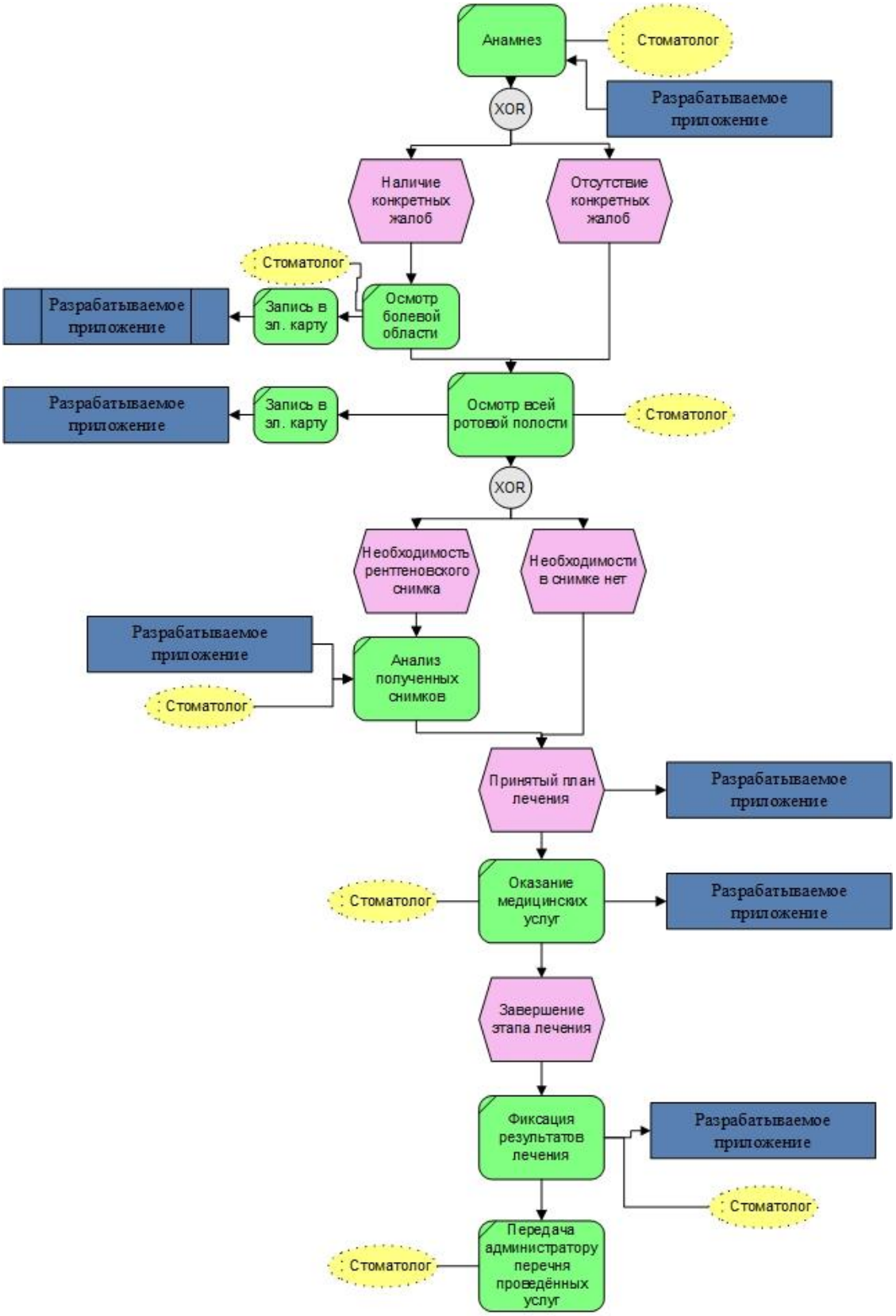


Рисунок 4.3 – *Второй уровень: процесс «Лечение пациента» в модели* «TO-BE»

Одним из основных требований к разрабатываемому приложению являлось наличие «зубной карты» – формы, которая позволяла бы вести подробный учёт состояния каждого зуба пациента. Для понимания необходимых действий проведём декомпозицию процесса «Фиксация результатов лечения» (рисунок 4.4).

На третьем уровне отражены ключевые бизнес-процессы, необходимые для создания «зубной карты». При создании приложения стоматолог сможет на рабочем месте получить всю необходимую информацию по интересующему зубу пациента. После внедрения приложения ожидается получение следующего формата ведения данных, который отражён в диаграмме на рисунке 4.5.

Бизнес-процесс «Регистрация» заключает основное событие, а именно «наличие пациента в БД», так как от этого действия зависят дальнейшие действия администратора. При наличии пациента в БД, администратор переходит к записи на приём, при отсутствии – заводит новую ЛК.

На третьем уровне детализации происходит заполнение личных данных пациента. Для более детального анализа деятельности стоматологической клиники рассмотрим дополнительно два бизнес-процесса. А именно – процесс регистрации пациента и процесс оплаты оказанных стоматологических услуг.

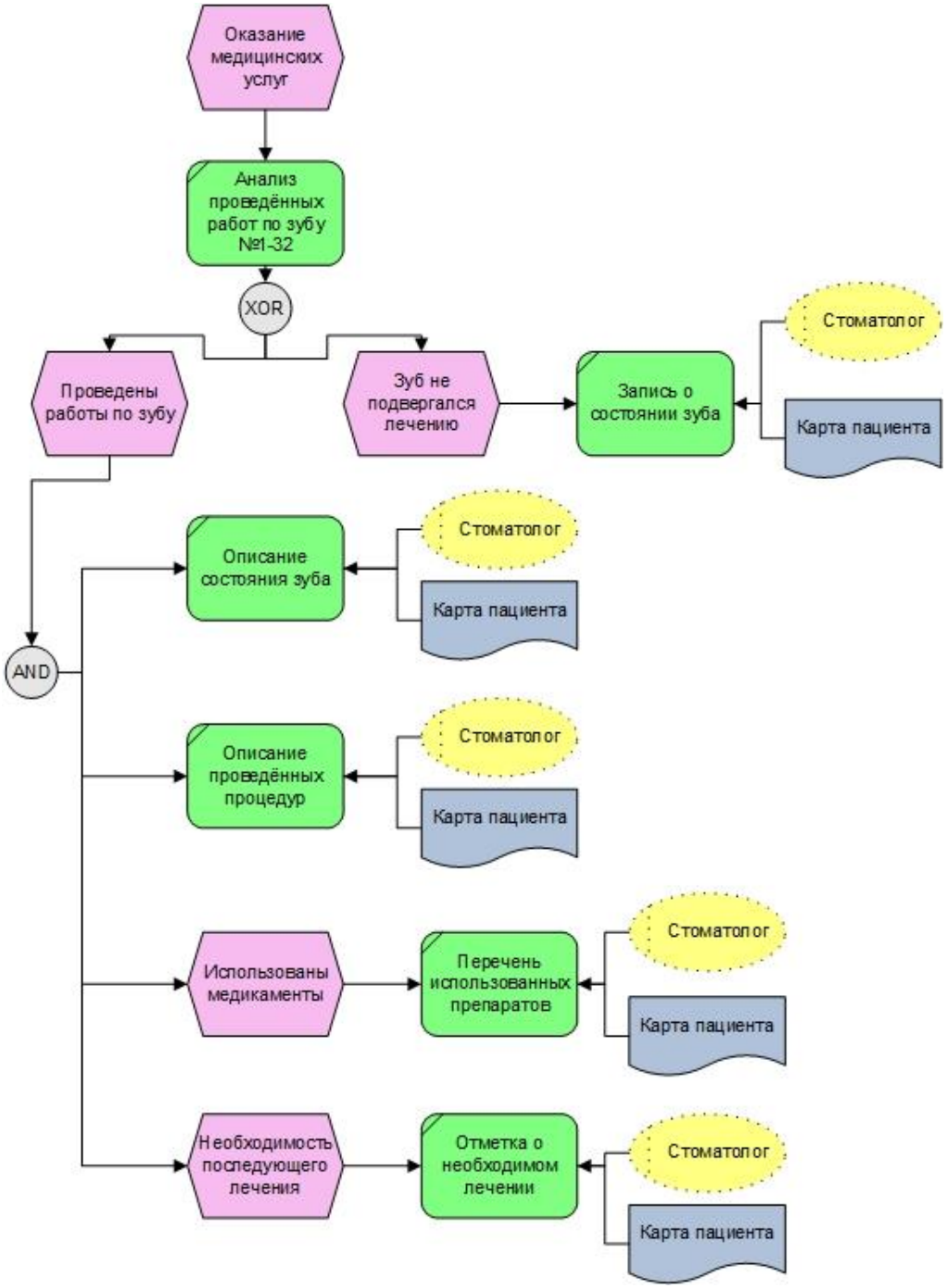


Рисунок 4.4 – *Третий уровень: процесс «Фиксация результатов лечения» в модели «AS-IS»*

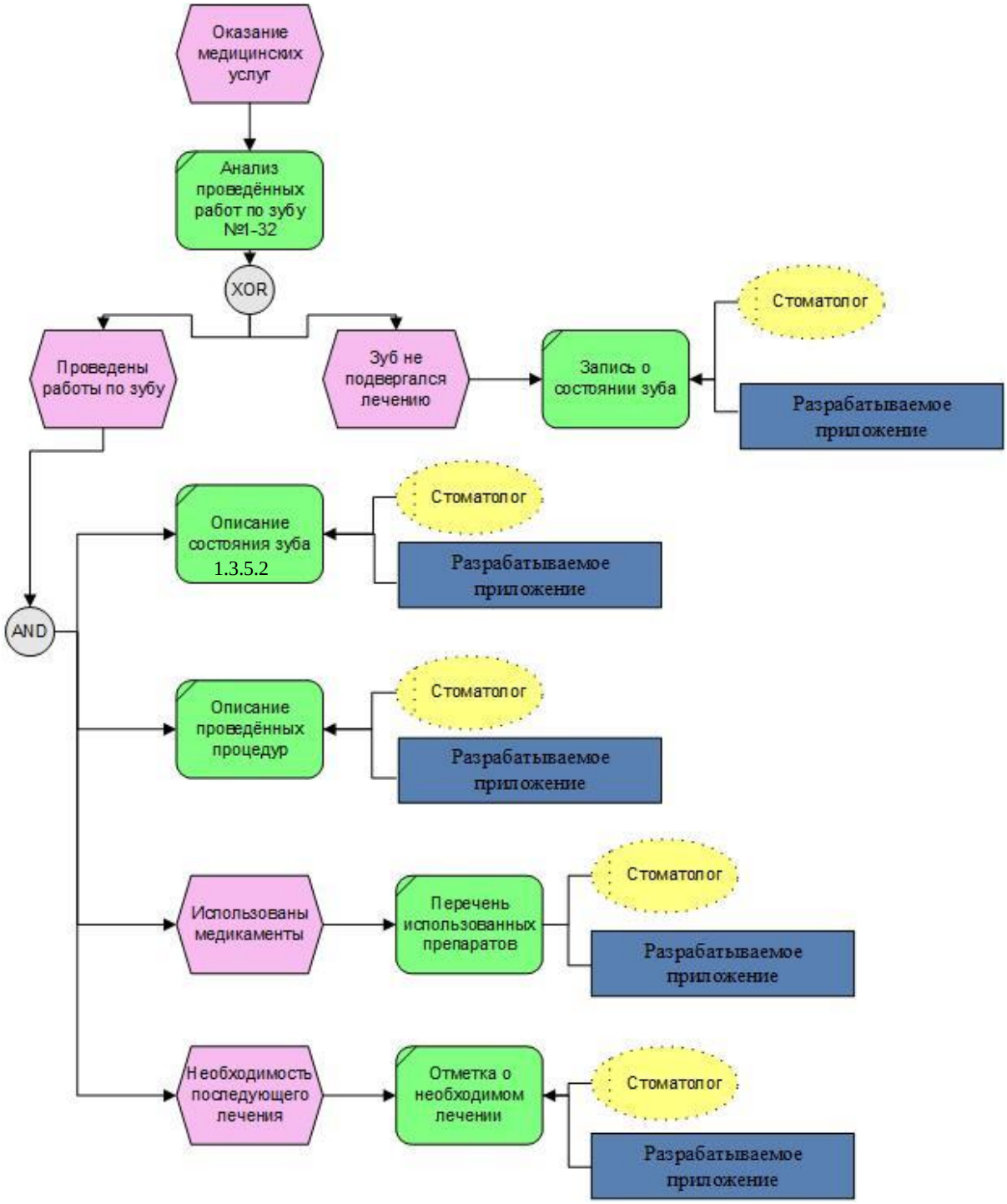


Рисунок 4.5 – *Третий уровень: процесс «Фиксация результатов лечения» в модели «TO-BE»*

Бизнес-процесс «Оплата» включает в себя определение оплачиваемого лица (юридическое лицо, физическое лицо, оплата по ОМС). В зависимости от этого происходит создание различных листов оплаты. В частности при выборе юридического лица, нет возможности оплатить наличными средствами (в соответствии с законодательством РФ). Таким образом, для реализации поставленной задачи деятельность стоматологии была расписана до 3 уровня детализации.

4.3. Архитектура данных разрабатываемого приложения

Проанализировав процессы бизнес-модели и необходимые требования к приложению, сделаем вывод о данных, которые будут храниться в базе. Эти данные распределим в таблице 3.1 по классам.

Таблица 4.3 – *Классы данных (до нормализации)*

Название класса	Поля данных
Пациент	ФИО
	Дата рождения
	Паспортные данные (серия и номер)
	Номер телефона
	Пол
Анамнез	Жалобы
	Дата обращения
Сопутствующие заболевания и противопоказания	Соп заболевания
	Противопоказания
Карта зубов	Зуб №1-№32
Состояние зуба	Состояние
	Проведенные процедуры
	Использованные препараты
	Необходимость дальнейшего лечения

Название класса	Поля данных
	Лечащий врач
Рентген снимки	Дата снимка
	Приложенный снимок
Услуги	Название услуги
	Цена
Специалисты	ФИО
	Специализация
	Номер телефона

Для реализации практического задания на стадии проектирования необходимо провести нормализацию данных будущей системы. Нормализация отношений позволяет устранить избыточность данных, что позволит снизить объём повторяющиеся информации, а также снизить риск проявления аномалий увеличения, удаления, модификации и обратимости [3].

В данном проекте будет использована нормализация до третьей нормальной формы (НФ). 1НФ, 2НФ, 3НФ позволяют ограничить зависимость непервичных атрибутов от ключей. Начнём последовательный переход к нормальным формам данных.

Данные находятся в 1 НФ, если все атрибуты имеют атомарные значения. Для этого разобьём поле "ФИО" на составляющие поля, а именно "Фамилия", "Имя", "Отчество". Перейдёт ко 2 НФ.

Данные находится во 2 НФ, если они соответствуют 1НФ и каждый атрибут полностью функционально зависит от первичного ключа. Для этого в каждый класс данных добавляем поле для записи первичного ключа, например, «Код пациента», «Код услуги», «Код специалиста» и т.д., а также добавляются соответствующие внешние ключи. Перейдём к 3 НФ.

Данные находится в 3 НФ, если оно соответствует 2 НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Для выполнения последнего условия необходимо изменить структуру "карты зубов", так как «описание зуба» зависит от «пациента» через «номер зуба». В данной работе используется стандартная нумерация зубов, которая представляет последовательное присвоение номера каждому зубу, начиная с верхней челюсти по часовой стрелке (рисунок 4.6).

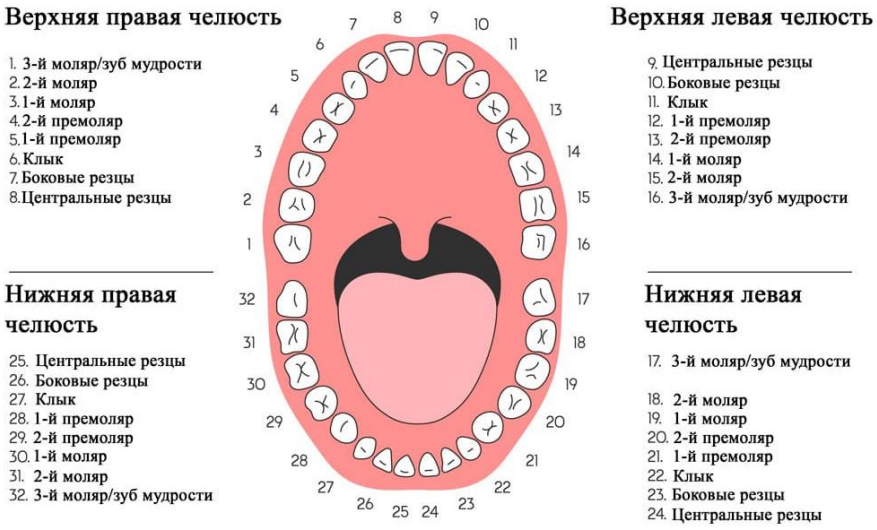


Рисунок 4.6 – *Нумерация зубов*

В условиях данной практической работы «зубная карта» будет вестись без учёта аномалий количества зубов. Нумерации зубов одного пациента не отличается от нумерации другого пациента, хотя при этом сами зубы будут иметь различные описания. Для того, чтобы система могла различать каждый зуб, введём регистрацию отдельных лечений, т.е. каждый совершенный приём врача будет являться отдельной операцией над конкретным зубом у конкретного пациента. Тогда у класса данных «состояние зуба» ключевым полем станет «Код проведённого лечения» и он будет зависеть от «Кода пациента» и «Номера зуба». Тем самым, денные примут вид 3 НФ. Отобразим

их в таблице 4.4 с указанием типов данных, размерности данных и ключевых полей (подчёркнутый текст в полях).

Таблица 4.4 – *Данные после нормализации*

Название класса	Поля данных	Тип данных	Размерность
Пациент	Код пациента	Числовой	До 5 символов
	Фамилия	Краткий текст	До 30 символов
	Имя	Краткий текст	До 30 символов
	Отчество	Краткий текст	До 30 символов
	Дата рождения	Дата и время	До 10 символов
	Паспортные данные (серия и номер)	Числовой	10 символов
	Номер телефона	Числовой	11 символов
	Пол	Подстановка	До 255 символов
Анамнез	Код анамнеза	Числовой	До 5 символов
	Код пациента	Числовой	До 5 символов
	Жалобы	Длинный текст	До 64000 символов
	Дата обращения	Дата и время	До 20 символов
Сопутствующие заболевания и противопоказания	Код записи	Числовой	До 5 символов
	Код пациента	Числовой	До 5 символов
	Соп заболевания	Текстовый	До 255 символов
	Противопоказания	Текстовый	До 255 символов
Состояние зуба	Код проведённого лечения	Числовой	До 5 символов
	Код пациента	Числовой	До 5 символов
	Дата лечения	Дата и время	До 20 символов
	Состояние	Текстовый	До 255 символов
	Проведенные процедуры	Длинный текст	До 64000 символов
	Использованные препараты	Текстовый	До 255 символов
	Необходимость дальнейшего лечения	Текстовый	До 255 символов
	Лечащий врач	Подстановка	До 255 символов
Рентген снимки	Код снимка	Числовой	До 5 символов
	Код пациента	Числовой	До 5 символов
	Дата снимка	Дата и время	До 20 символов

Название класса	Поля данных	Тип данных	Размерность
	Приложенный снимок	Вложение	До 255 символов
Услуги	Код услуги	Числовой	До 5 символов
	Название услуги	Текстовый	До 255 символов
	Цена	Числовой	До 255 символов
Специалисты	Код специалиста	Числовой	До 5 символов
	ФИО	Текстовый	До 255 символов
	Специализация	Текстовый	До 255 символов
	Номер телефона	Числовой	11 символов

Проведя ряд изменений в структуре данных, была получена архитектура данных разрабатываемого приложения, которая отображена на рисунке 4.7. Ключевыми полями классов данных являются записи в блоках, начинающиеся со слова «Код …», например, «Код пациента».

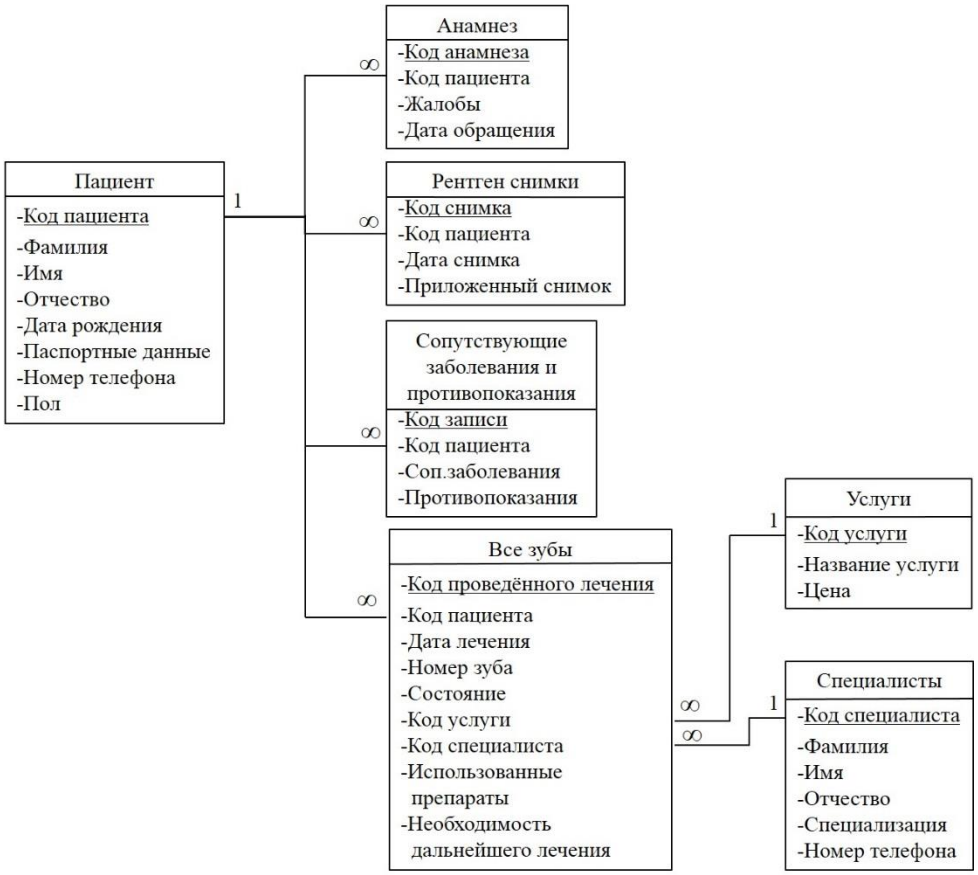


Рисунок 4.7 – *Архитектура данных*

4.4. Проектирование интерфейсов приложения

Для обеспечения удобной работы пользователя с базой данных необходимо разработать интерфейс, который позволит взаимодействовать со всеми функциональными возможностями разработанного приложения. Для удовлетворения потребностей пользователя, желательно, чтобы интерфейс соответствовал следующим критериям:

- простота и наглядность рабочих окон;
- отсутствие нагромождения кнопок и полей для ввода информации;
- разделение функционала приложения по категориям.

Для удобства перемещения между функциями приложения необходимо начальный экран, на которое будут реализованы ссылки с остальных интерфейсов. На нём также будут находиться кнопки задающие переход на последующие интерфейсы приложения (рисунок 4.8).

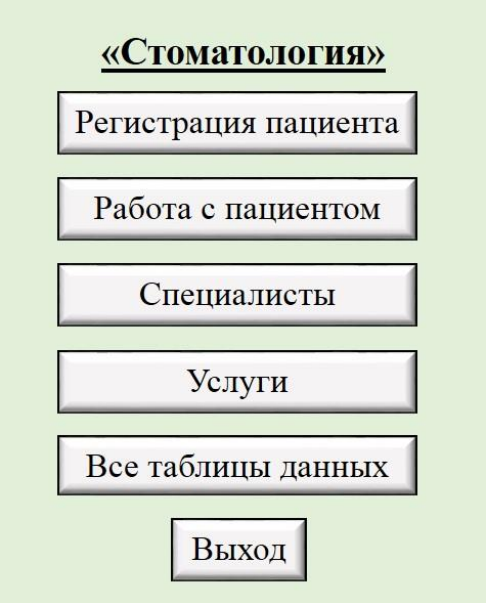


Рисунок 4.8 – *Начальный экран*

Рисунок 4.9 – *Окно «Регистрация пациента»*

Для внесения данных нового пациента необходима определённая форма с полями, которая упростит добавление информации в базу данных. Для этой цели пользователь увидит окно, изображенное на рисунке 4.9.

Для удобной работы с данными пациента и всей необходимой для лечения дополнительной информацией (анамнез, сопутствующие заболевания и противопоказания, рентген-снимки, зубная карта) организуется окно «Работа с пациентом» (рисунок 4.10).

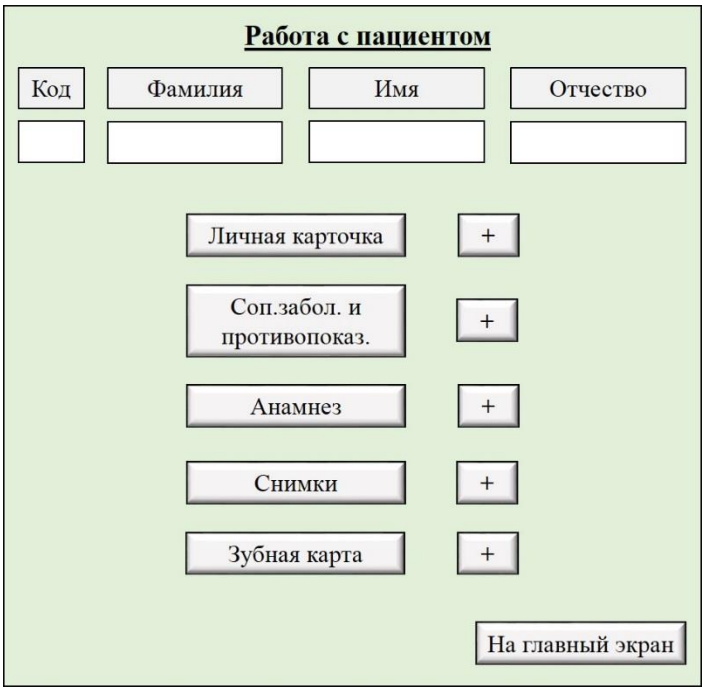


Рисунок 4.10 – *Окно «Работа с пациентом»*

Отдельным функционалом приложения является работа с «зубной картой». Для поиска необходимого зуба пациента реализуется оконная форма «Поиск информации по зубу» (рисунок 4.11).

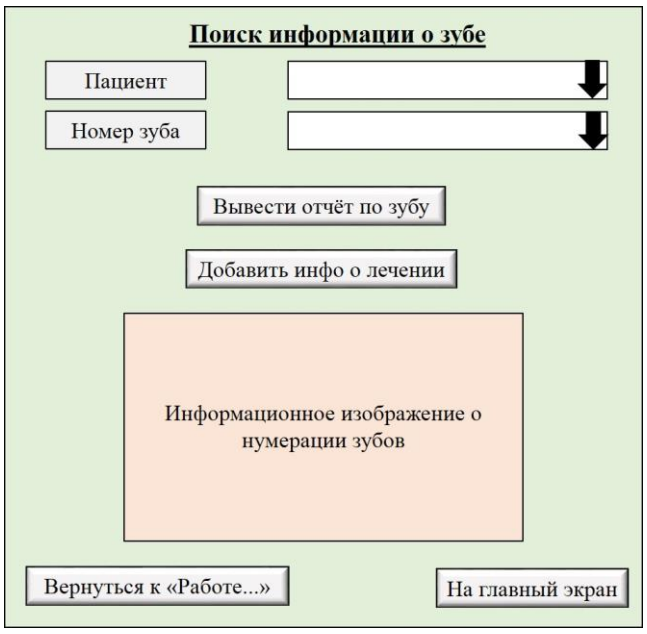


Рисунок 4.11 – *Окно «Поиск информации о зубе»*

Работа разрабатываемого приложения предполагает включение нескольких таблиц с данными. Для вывода их на экран будет реализовано окно «Все таблицы данных» (рисунок 4.12), которое позволит быстро найти и отобразить таблицы, имеющиеся в базе данных.

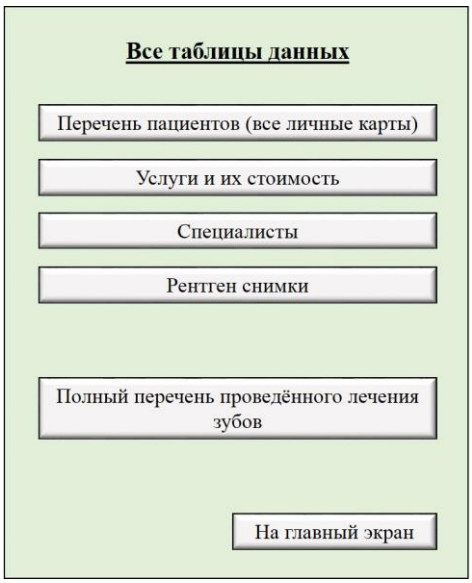


Рисунок 4.12 – *Окно «Все таблицы данных»*

Работа с данными, хранящимися в таблице, будет заключаться в поиске нужных записей, их просмотре и изменении. Например, будет требоваться такая работа с таблицами по специалистам и по услугам стоматологии. Для выполнения этих действий реализуются окна «Специалисты» (рисунок 4.13) и «Услуги» (рисунок 4.14).

Рисунок 4.13 – *Окно «Услуги»*

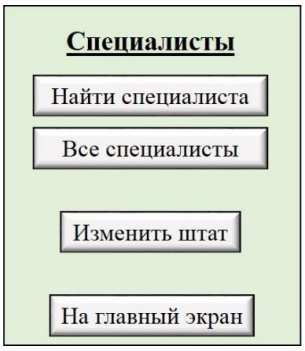


Рисунок 4.14 – *Окно «Специалисты»*

Благодаря разработанному интерфейсу, приложение сможет быть доступным для пользователя и обеспечит доступ к своему функционалу. Для наглядности и понимания, что будет отображаться на экране и какова последовательность открытия окон при различных нажатиях на кнопки, составлена схема приложения (рисунок 4.15.1 и 4.15.2).

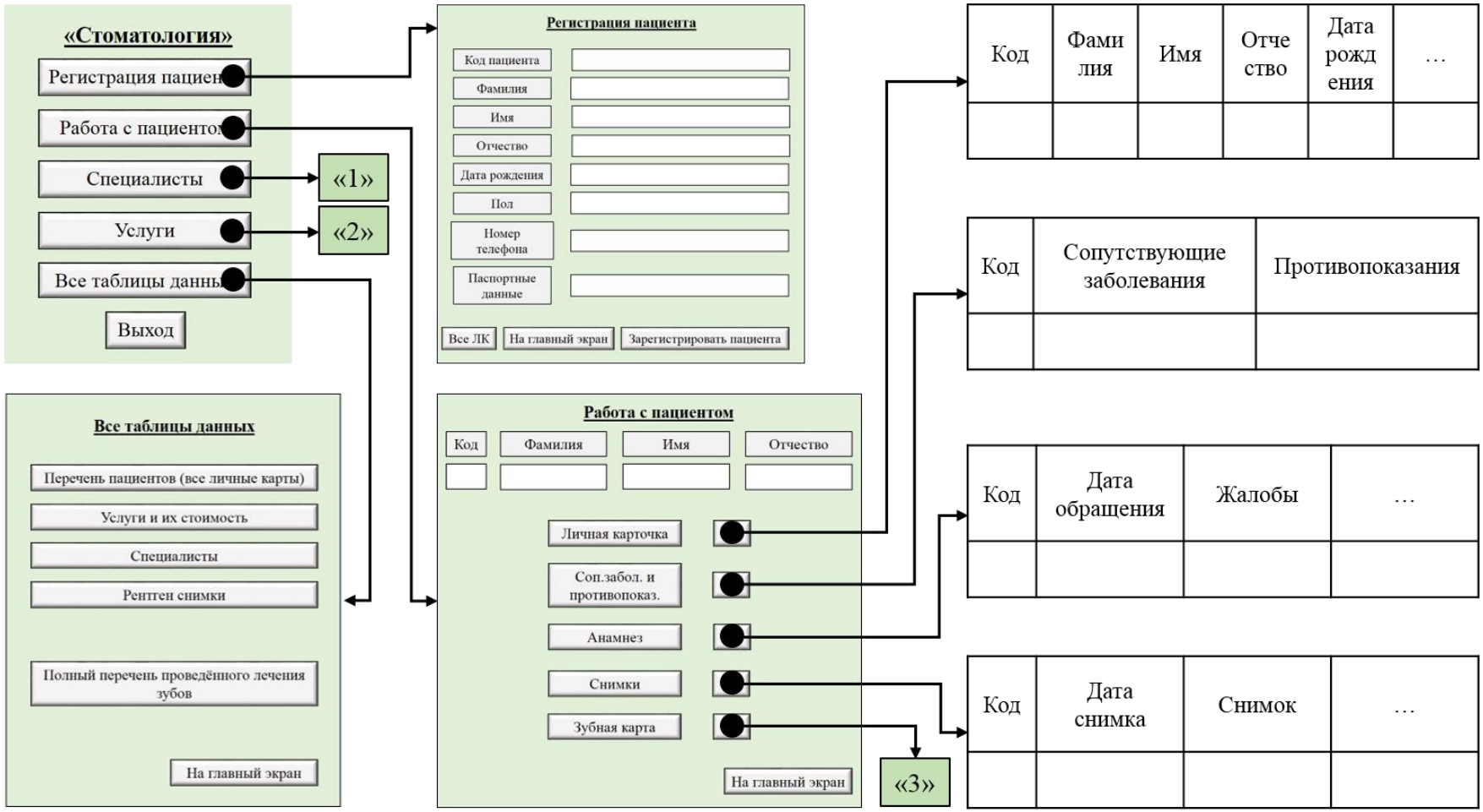


Рисунок 4.15.1 – *Схема приложения (часть 1)*

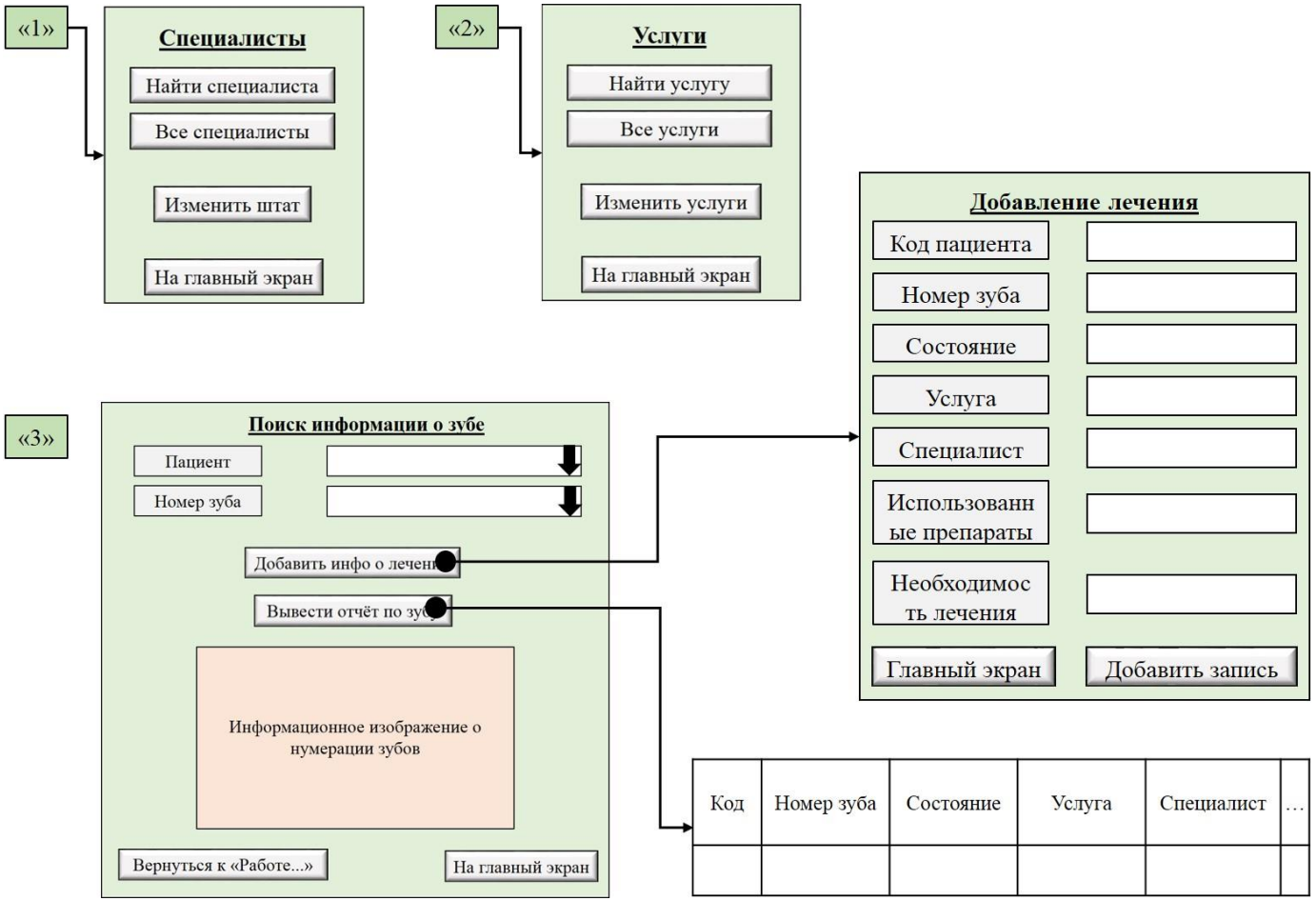


Рисунок 4.15.2 – *Схема приложения (часть 2)*

Раздел 5. Реализация приложения

5.1. Первый виток спирали (1 билд приложения)

В соответствии с обозначенными в разделе 3.5 шагами, приступим к реализации приложения в программной среде Microsoft Access 2016. В первую очередь создадим таблицы для хранения необходимой информации и организуем связи между ними, тем самым получим базу для хранения данных.

На рисунке 5.1 представлен скриншот, демонстрирующий организованные таблицы с имеющимися в них атрибутами, а также связи между этими таблицами. На рисунке 5.2, для примера, изображена таблица «Пациенты».

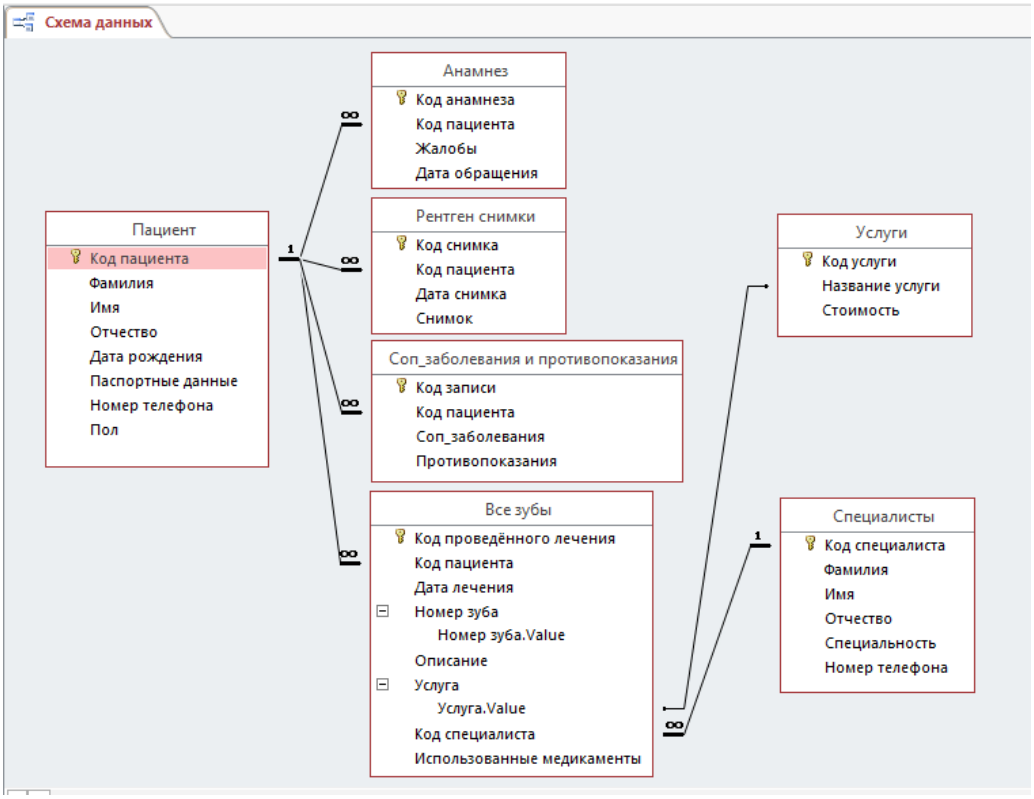


Рисунок 5.1 – *Схема данных*

Пациент

Код	Фамилия	Имя	Отчество	Дата рождения	Паспортные данные	Номер телефона	Пол
0	Петров	Петр	Петрович	06.12.1991	4617123321	321567	муж
1	Иванов	Иван	Иванович	01.01.1970	1212121212	948304	муж
23	Петренко	Иван	Игоревич	03.07.2018	1234567	987654	муж
0					0	0	

Рисунок 5.2 – *Таблица «Пациенты»*

5.2. Второй виток спирали (2 билд приложения)

Таблицы Microsoft Access позволяют добавлять, удалять и редактировать хранящуюся в них информацию. Однако, это является неудобным для пользователя когда есть необходимость внести лишь конкретную информацию о единичном пациенте или лечении. Для обеспечения удобного добавления информации создадим ряд форм, которые позволят добавлять данные в таблицу (рисунки 5.3 – 5.7).

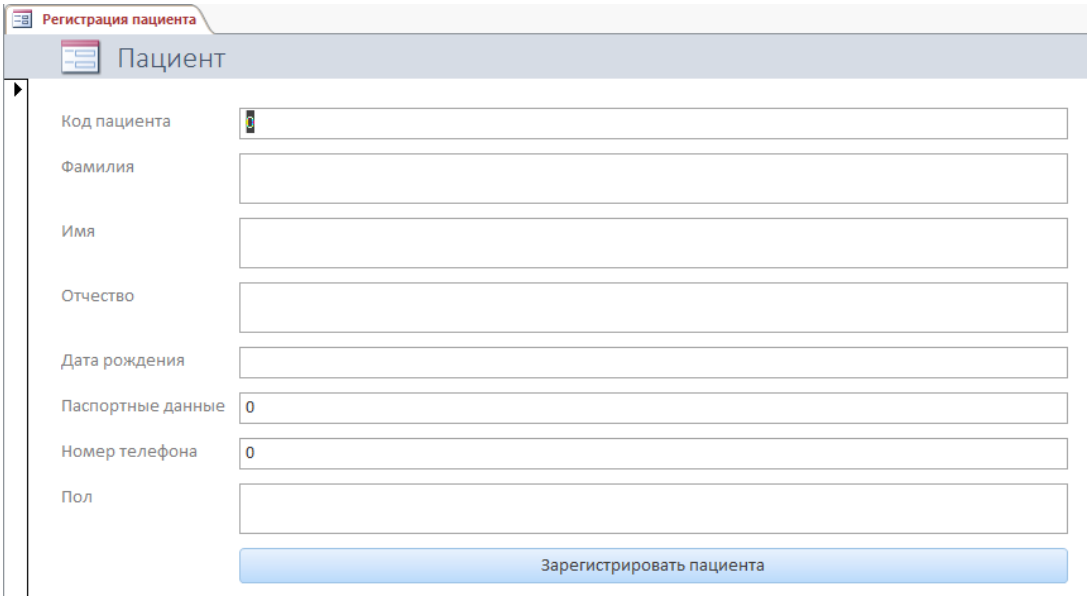


Рисунок 5.3 – *Форма «Регистрация пациента»*

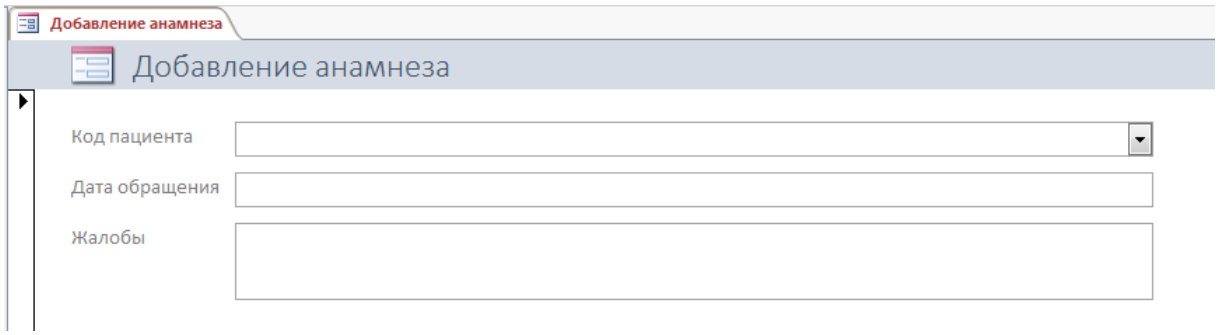


Рисунок 5.4 – *Форма «Добавление анамнеза»*

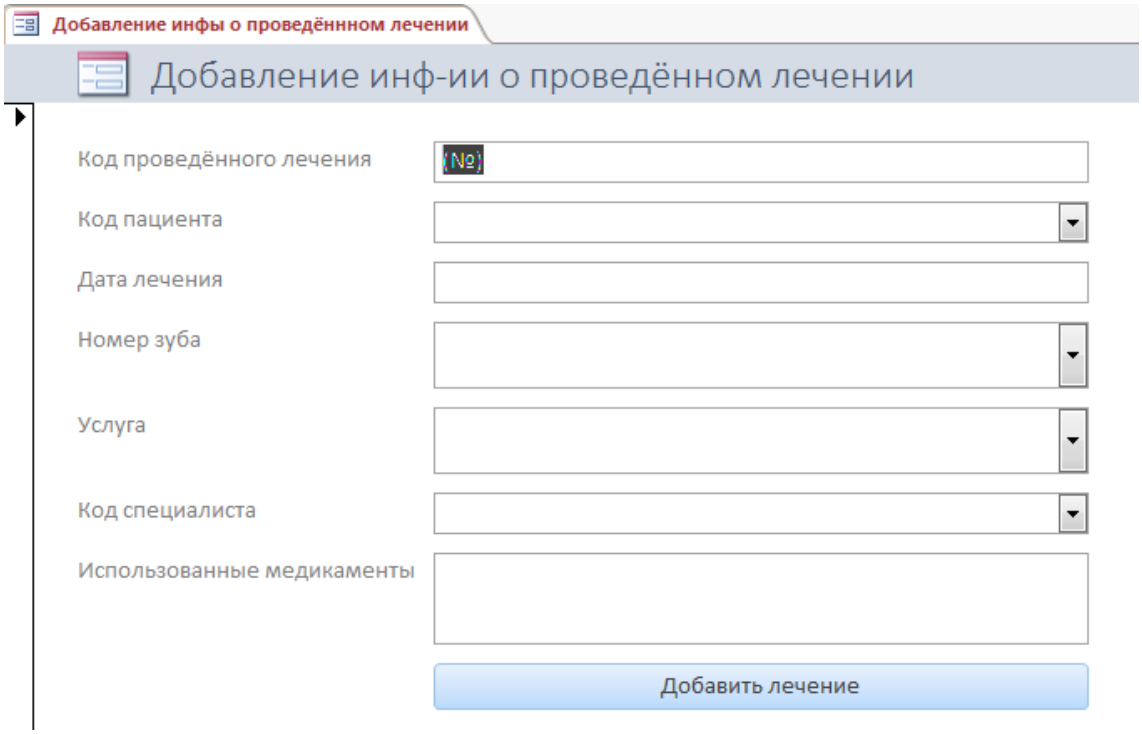


Рисунок 5.5 – *Форма «Добавление информации о проведённом лечении»*

5.3. Третий виток спирали (3 билд приложения)

После создания форм для работы с новыми записями перейдём к организации поисковых запросов и интерфейса к ним. База данных содержит несколько таблиц, поиск информации из которых будет осуществляться посредством выборки данных по критериям, которые задаст сам пользователь. На примере поиска личной карточки пациента продемонстрируем данный процесс (другие запросы имеют схожую конструкцию).

Для поиска личной карточки организуется запрос (рисунок 5.6), в условиях отбора которого прописаны команды, предоставляющие возможность внести критерии отбора по отдельности (например, «Like "*" & [Введите код пациента] & "*"»). Данные команды позволяют вывести на экран диалоговое окно, в поле которого пользователь вводит интересующие его данные (рисунок 5.7). При необходимости поля могу быть оставлены пустыми, тогда выборка пройдёт без учёта этого критерия.

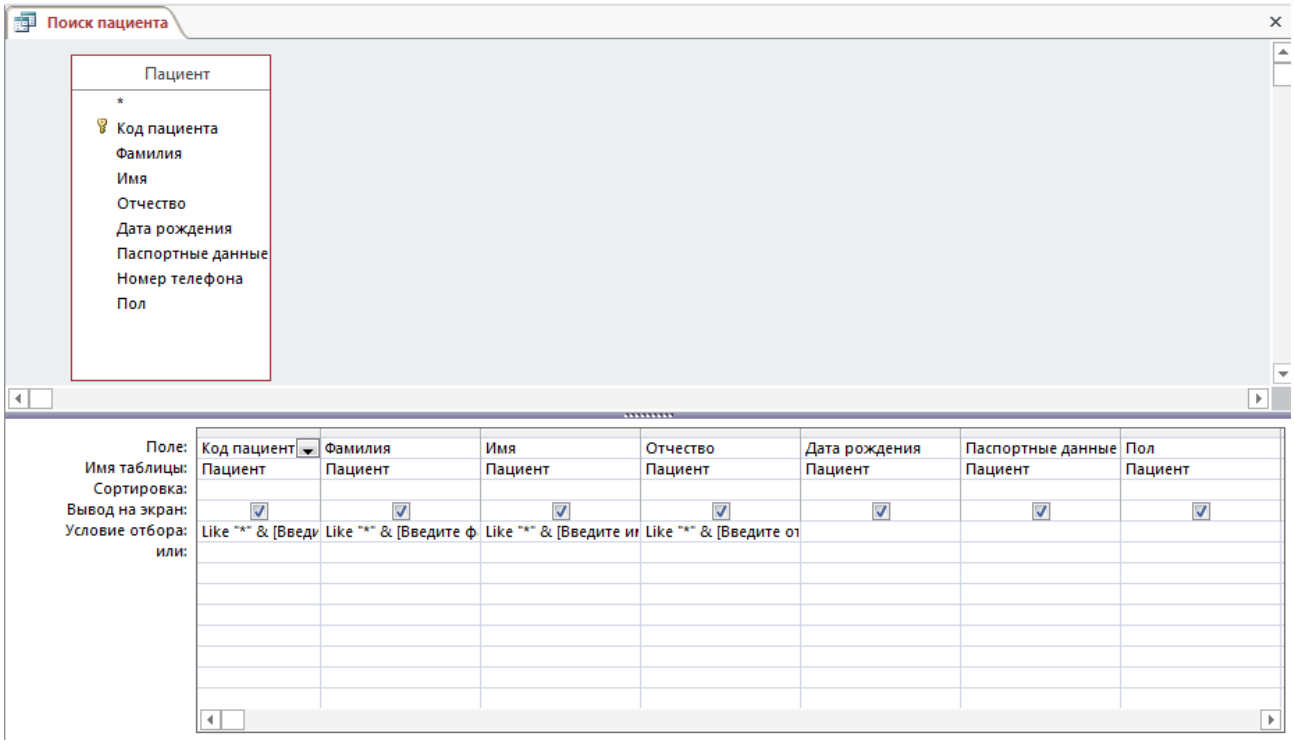


Рисунок 5.6 – *Запрос «Поиск пациента»*

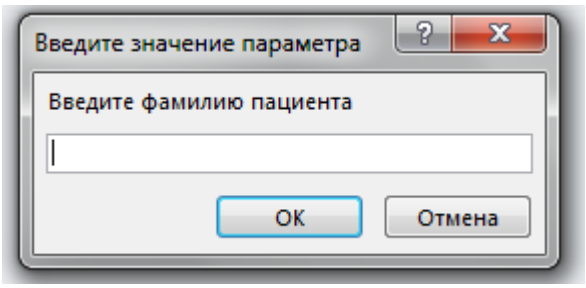


Рисунок 5.7 – *Диалоговое окно для ввода параметра выборки «Фамилия»*

Аналогичным образом реализованы и другие выборки. Скриншоты, демонстрирующие конструкционную организацию остальных запросов, отображены на рисунках 5.8 – 5.12.

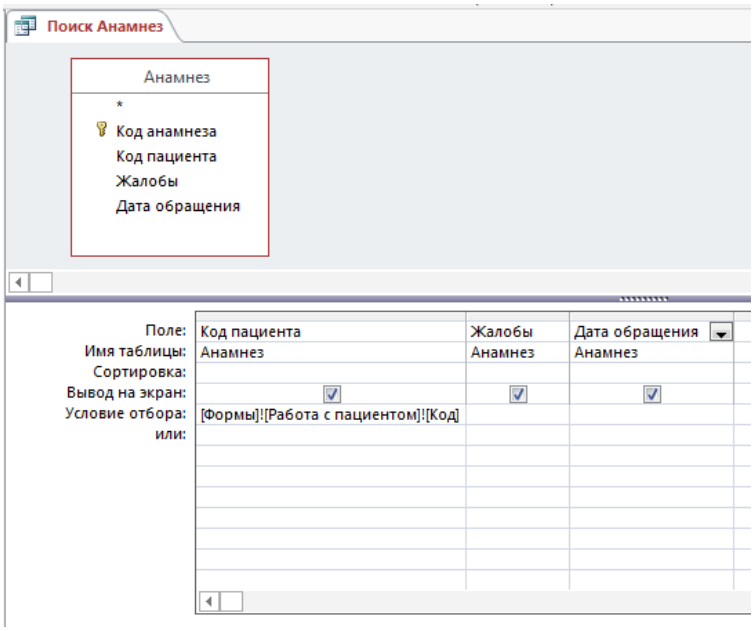


Рисунок 5.8 – *Запрос «Поиск анамнез»*

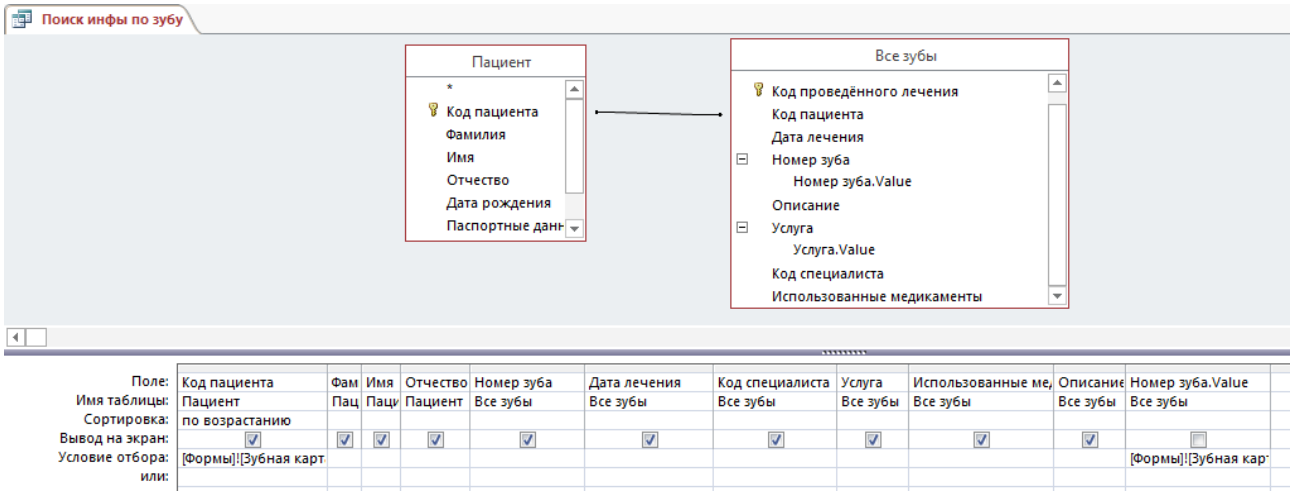


Рисунок 5.9 – *Запрос «Поиск информации по зубу»*

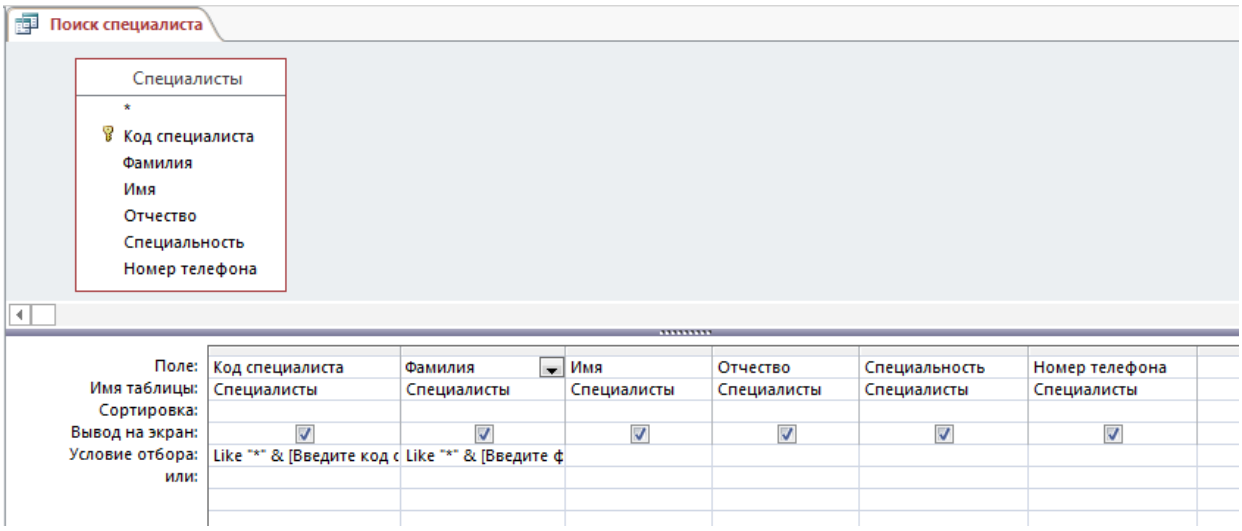


Рисунок 5.10 – *Запрос «Поиск специалиста»*

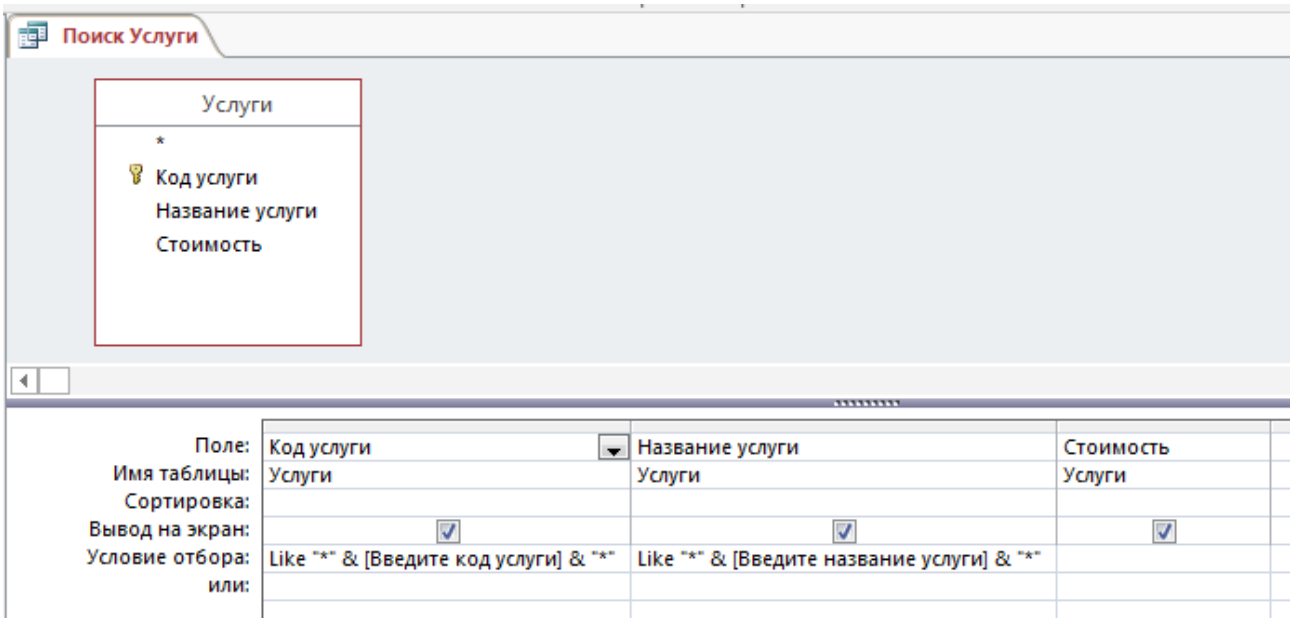


Рисунок 5.11 – *Запрос «Поиск услуги»*

Интерфейс приложения будет организован с помощью форм, позволяющих добавить на экран кнопки – вызовы необходимого раздела базы данных. Интерфейс главного экрана, позволяющий постепенно перейти ко всем функциям и данным приложения, продемонстрирован на рисунке 5.12.

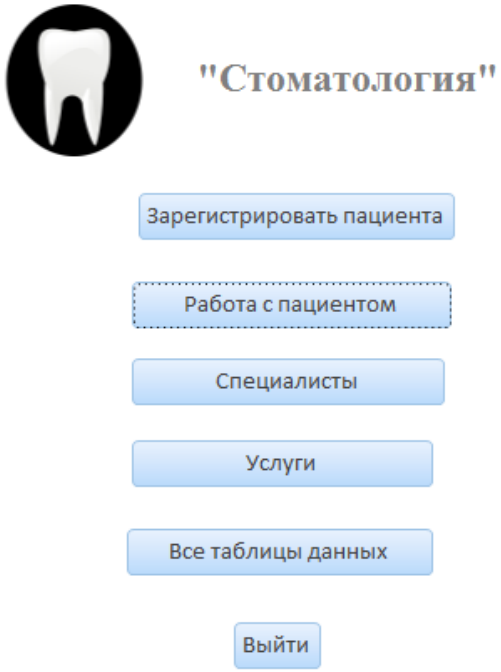


Рисунок 5.12 – *Интерфейс главного окна*

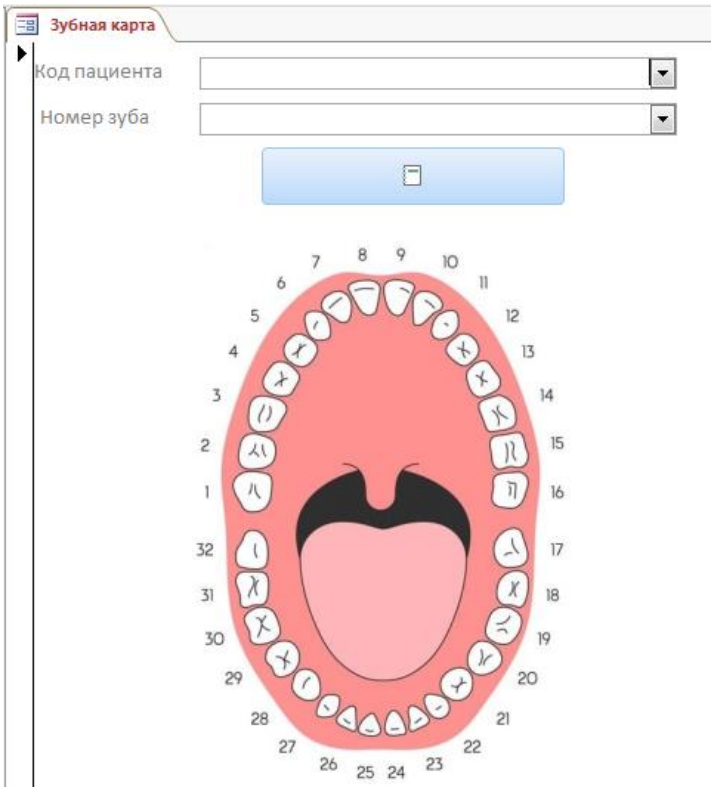


Рисунок 5.13 – *Форма «Зубная карта»*

«Зубная карта» имеет вид общей таблицы со всеми зубами, обслуживаемыми в стоматологии. Запрос на поиск необходимого зуба состоит из выбора пациента и номера необходимого зуба. Для удобства выборы этих критериев реализованы выпадающими списками на форме (рисунок 5.13).

Доступ управления данных о специалистах и услугах осуществляется через кнопочные формы, изображённые на рисунках 5.14 и 5.15 соответственно.

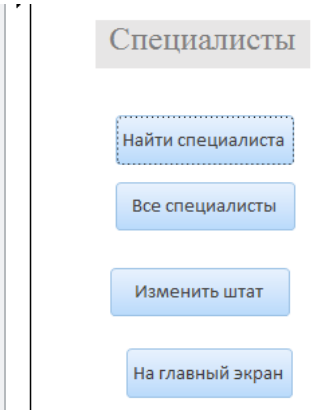


Рисунок 5.14 – *Форма «Услуги»* **Рисунок 5.15** – *Форма «Специалисты»*

5.4. Четвёртый виток спирали (4 билд приложения)

Рисунок 5.16 – *Отчёт «Услуги»*

Представление информации из таблиц выполним, используя инструмент «отчёты». В таком виде информация представляется нагляднее, не подлежит изменению и, при необходимости, может быть распечатана. На рисунках 5.16 – 5.19 изображены примеры скриншотов отчётов различных запросов.

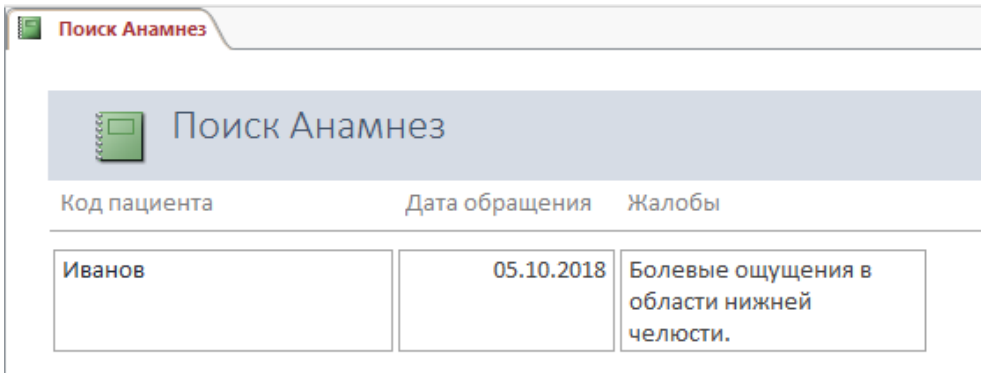


Рисунок 5.17 – *Отчёт «Анамнез»*

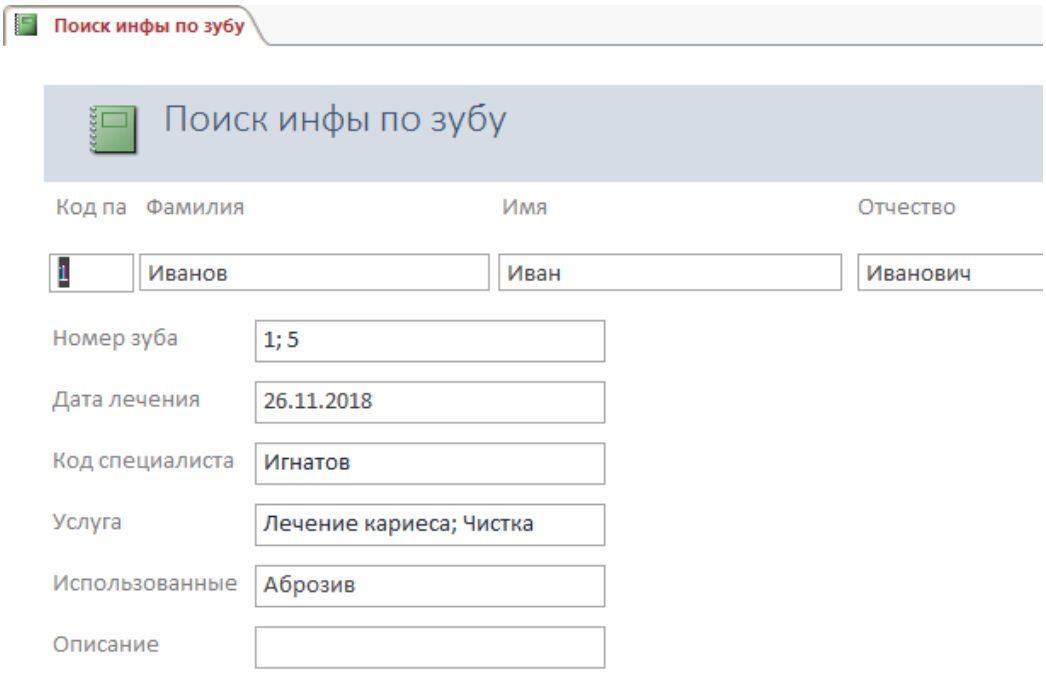


Рисунок 5.18 – *Отчёт «Поиск информации по зубу»*

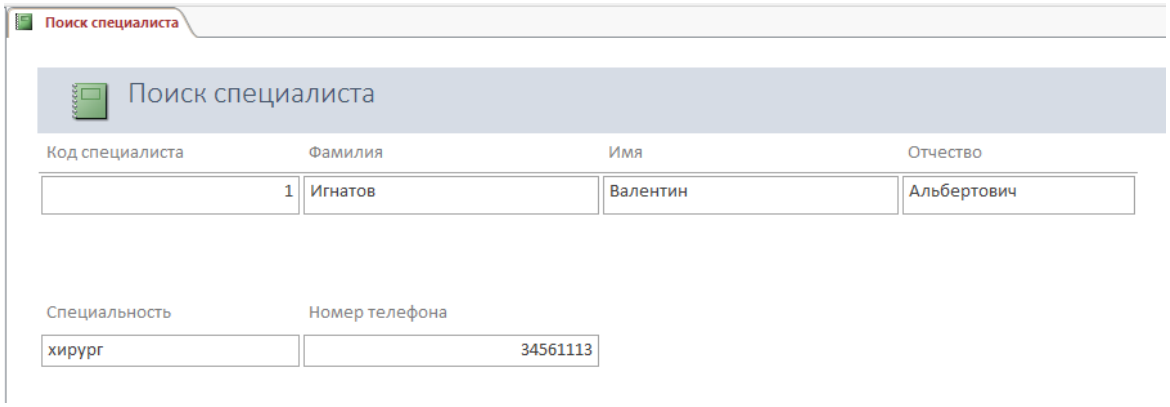


Рисунок 5.19 – *Отчёт «Поиск специалиста»*

Раздел 6. Тестирование приложения

После создания приложения проводится его тестирование, для выявления проблем в работе, исправление ошибок, а также проверку соответствия изначальных требований. Тестирование ПО проводится до передачи готового продукта заказчику. Условно виды тестирования можно подразделить на три группы [13]:

- Функциональные.
- Нефункциональные.
- Связанные с изменениями.

Функциональное тестирование направлено на проверку функций и взаимодействие с другими системами. Основной задачей этого типа является определение соответствия результата требуемого действия, изложенного в функциональных требованиях, с тем, что, в конечном счёте, происходит в приложении.

Нефункциональное тестирование направлено на определение характеристик приложения, которые могут быть оценены определёнными величинами. В целом, этот тип тестирования показывает, как система работает в различных условиях её использования.

Тесты, связанные с изменениями, направлены на повторное тестирование после исправления ошибок в программном обеспечении. Проводится в тех случаях, когда необходимо удостовериться в правильности доработки приложения, и что работоспособность приложения не нарушена.

6.1. Функциональное тестирование

В рамках функционального тестирования проведём два теста: на модульном уровне и на интеграционном уровне [7]. Модульное тестирование будет заключаться в постановке определённой задачи и проверке её надлежащего исполнения. Поставим задачи, охватывающие ключевой бизнес-процесс практической работы, после чего отразим пошаговые действия в приложении по их выполнению. Задача 1. Регистрация личной карточки пациента с известными данными:

- ФИО – Измайлов Пётр Викторович.
- Дата рождения – 17.05.1987.
- Паспортные данные – 4617321123.
- Номер телефона – 495345543.
- Пол – муж.

Реализация задачи требует выполнения следующих шагов:

- Открытие приложения (рисунок 6.1).

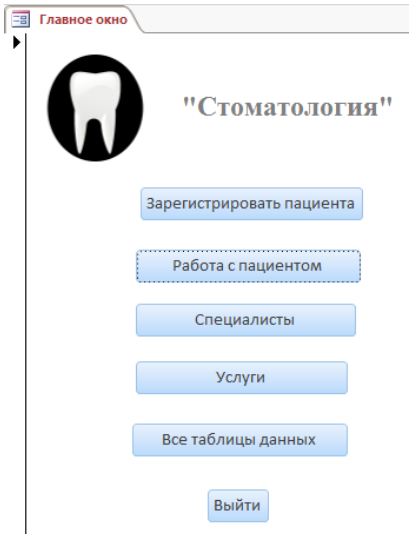


Рисунок 6.1 – *Главное окно*

- После нажатия клавиши «Зарегистрировать пациента» открывается форма для внесения личных данных нового пациента. Заполняем поля (рисунок 6.2).

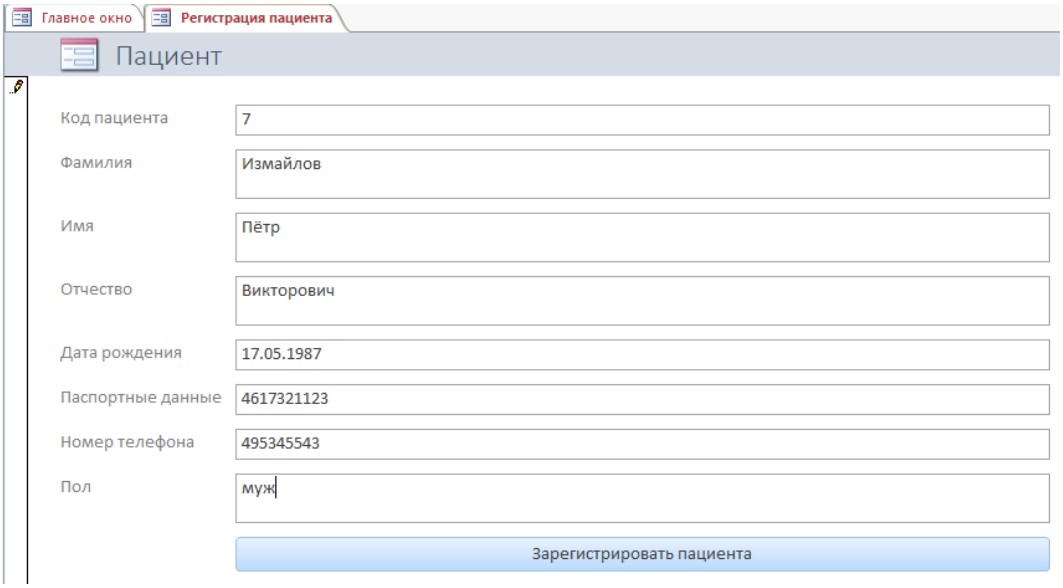


Рисунок 6.2 – *Занесение данных о пациенте*

- После через главное форму ищем пациента через «Работу с пациентами» и заполняем поля «Фамилия» и «Имя» (рисунок 6.3).

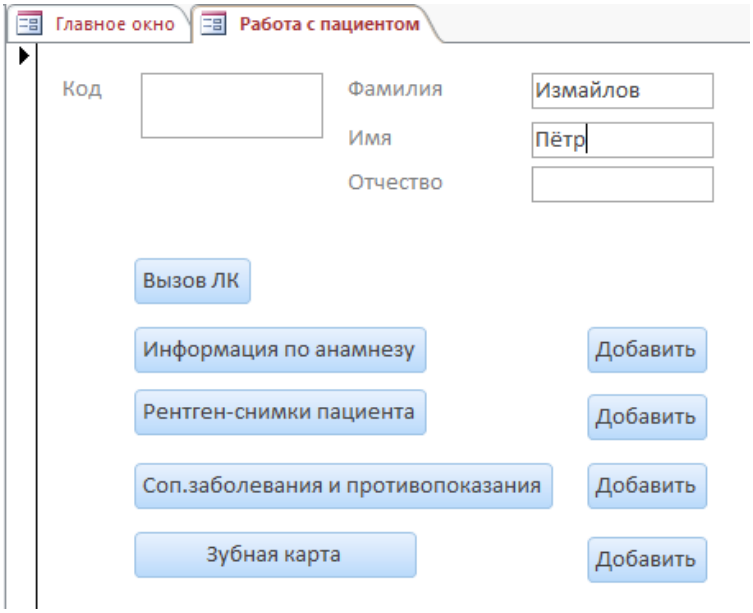


Рисунок 6.3 – *Поиск пациента Измайлова Петра*

- После нажатия кнопки «Вызов ЛК» была выведена информация по необходимому пациенту (рисунок 6.4).

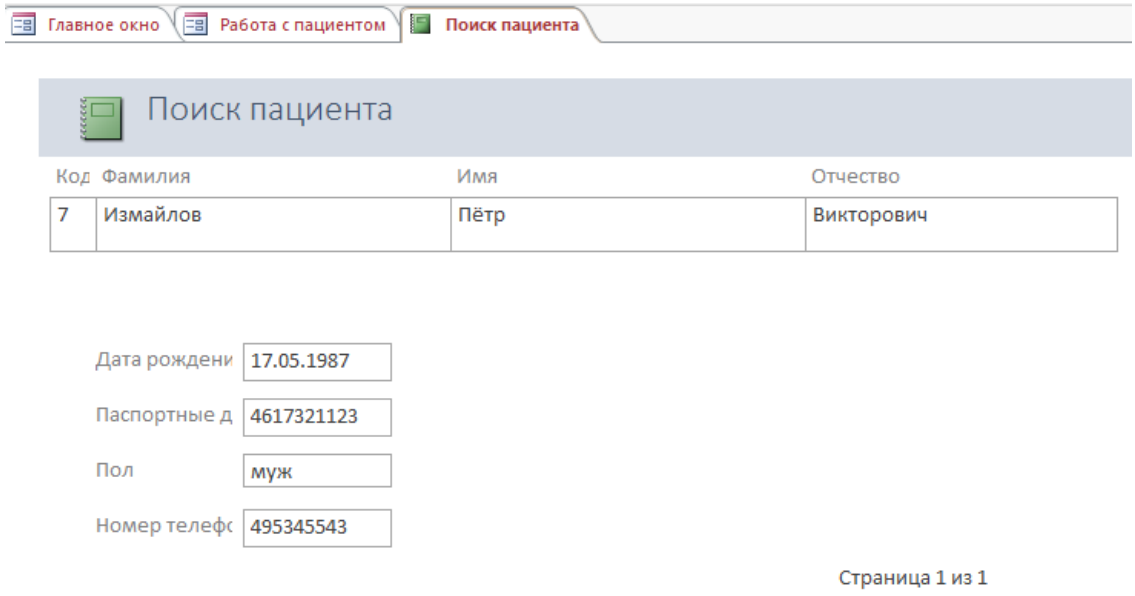


Рисунок 6.4 – *Личная карточка необходимого пациента*

Задача 2. Занесение информации о проведённом лечении:

- Пациент – Измайлов Пётр Викторович.
- Дата лечения – 11.11.2018.
- Номера зубов – 6,7.
- Услуга – чистка, лечение кариеса.
- Код специалиста – 2.
- Использованные материалы – фреза, пломба.

Реализация задачи требует выполнения следующих шагов:

- Открытие приложение с главным окном (рисунок 6.1) и последовательное нажатие кнопок: «Работа с пациентом» -> кнопка «Добавить», напротив кнопки «Зубная карта», (рисунок 6.5).

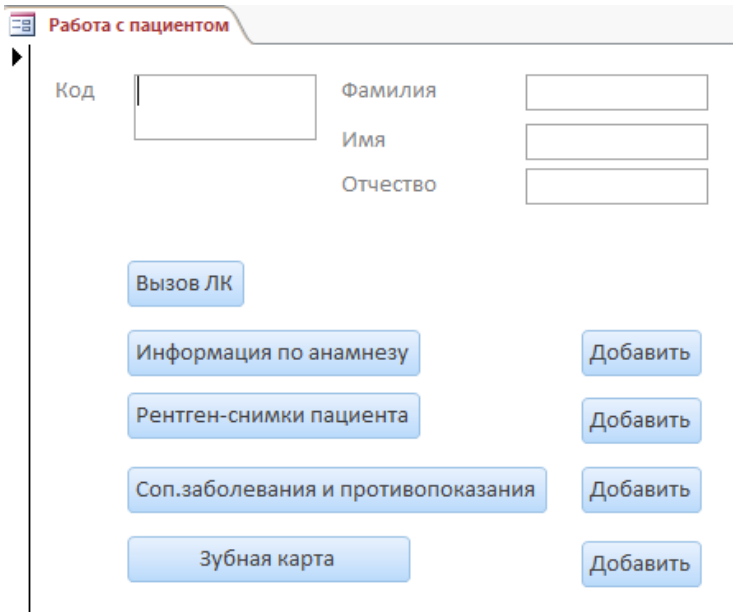


Рисунок 6.5 – *Окно «Работа с пациентом»*

- В открывшееся окно добавления информации о проведённом лечении вносим данные (рисунок 6.6).

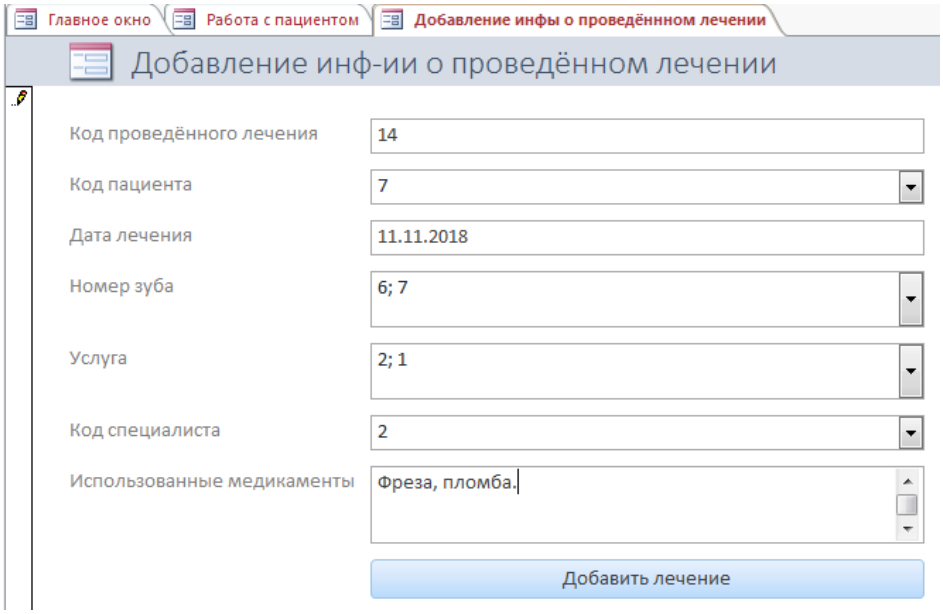


Рисунок 6.6 – *Внесение данных о проведённом лечении*

- После добавления лечения найдём информацию по проведённому лечению зуба №7 пациента Измайлова. Откроем приложение с главным

окном (рисунок 6.1) и последовательное нажатие кнопок: «Работа с пациентом», «Зубная карта». В открывшееся окно введем данные (рисунок 6.7).

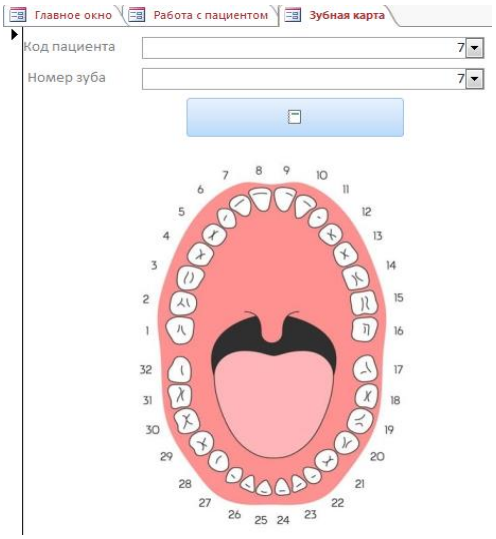


Рисунок 6.7 – *Поиск данных по искомому зубу*

- После нажатия кнопки поиска приложение вывело отчёт, содержащий информацию о необходимом зубе (рисунок 6.8).

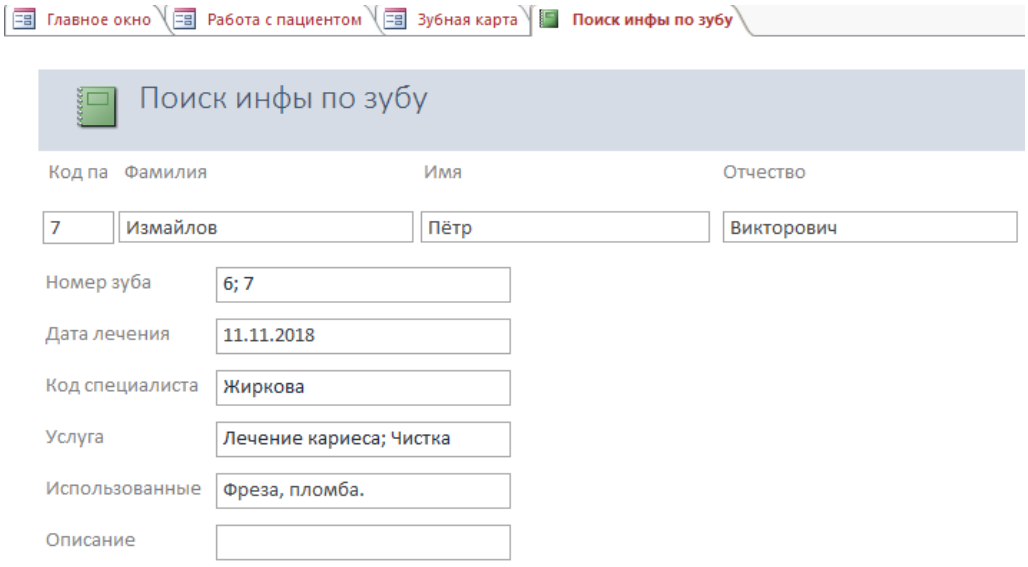


Рисунок 6.8 – *Отчёт по искомому зубу*

Функциональное тестирование прошло успешно, так как результат от проделанных действий в приложении соответствует требованиям и в процессе работы с информацией не возникло системных ошибок.

6.2. Нефункциональное тестирование

В нефункциональном тестировании выберем тест на производительность системы, а именно нагрузочный тест [7]. Данный тест позволяет проверить, как будет вести себя разработанное приложение при наличии различного числа хранящихся данных. Выбор обусловлен так же тем, что основной задачей программного обеспечения является поиск и выведение искомой информации из области хранения данных. Быстрота действия системы на прямую влияет на скорость обслуживание пациента. Основные действия программы – это запись и поиск информации (с выводом отчёта). Количество записей выбрано следующее: 1, 10, 100. Расчёт проводится по среднеквадратичному отклонению [14]. Для этого замеряются временные промежутки выполняющегося действия, а далее находится время отклика по среднему арифметическому по формуле (1), где $t_{ср}$ – среднее время, t_i – время одного замера, n – количество замеров. Полученные результаты занесём в таблицу 6.1.

$$t_{\text{ср}} = \frac{\sum_{i=1}^{N} t_i}{n} \quad (1)$$

Для расчёта среднеквадратичного отклонения необходимо рассчитать дисперсию, для чего воспользуемся формулой (2). (Рассчитанная дисперсия оказалась в 10^{-4} степени, что во много раз меньше секунды, поэтому примем $\sigma = 0$).

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

Конечная погрешность рассчитывается по формуле (3), в которой «P» – доверительный коэффициент Стьюдента (примем его равным 0,95), а «A» – абсолютная погрешность секундомера, равная 0.005.

$$\Delta t = \sqrt{(\frac{P*\sigma}{\sqrt{n}})^2 - A^2)} \quad (3)$$

Итоговое значение времени отклика примет вид, обозначенный в формуле (4).

$$t_{отв} = t_{ср} \pm \Delta t \quad (4)$$

Таблица 6.1 – *Результаты нагрузочного тестирования*

Кол-во записей	Действие	t_1, с	t_2, с	t_3, с	t_4, с	t_5, с	Дисперсия, с	Время отклика, с
1	Запись	0,11	0,13	0,10	0,10	0,11	0	0,088±0.005
	Поиск	0,20	0,21	0,19	0,19	0,20	0	0,198±0.005
10	Запись	0,13	0,12	0,12	0,12	0,13	0	0,124±0.005
	Поиск	0,10	0,12	0,11	0,14	0,11	0	0,116±0.005
100	Запись	0,13	0,11	0,14	0,11	0,12	0	0,122±0.005
	Поиск	0,16	0,12	0,13	0,11	0,14	0	0,132±0.005

Нагрузочное тестирование прошло успешно, так как время выполнения команды является комфортным для восприятия информации.

Раздел 7. Риски в проекте приложения

В спиралевидной модели, как уже было отмечено в разделе «Общее описание принципа спиралевидной модели», отдельное внимание уделяется рискам, возникающим во время проектирования, разработки, эксплуатации приложения. Словарь русского языка С.И. Ожегова и Н.Ю. Шведовой даёт следующее определения риска: риск – это возможность опасности, неудачи. В ходе разработки приложения возможно возникновение таких ситуаций и факторов, которые могут замедлить процесс разработки, усложнить его, повысить необходимость некоторых ресурсов или вовсе привести к отказу от выпуска продукта.

Для минимизации влияния со стороны возможного риска проводится управление риском. Управление риском содержит в себе шесть действий [16]:

- Идентификация риска – выявление элементов риска в проекте.
- Анализ риска – оценка вероятности величины потери по каждому элементу риска.
- Ранжирование риска – упорядочение элементов риска по степени их влияния.
- Планирование управления риском – подготовка к работе с каждым элементом.
- Разрешение риска – устранение или разрешения элементов риска.
- Наблюдение риска – отслеживание динамики элементов риска, выполнение корректирующих действий.

Действия 1-3 относятся к этапу оценивания риска, действия 4-6 – к этапу контроля риска. Рассмотрим некоторые действия более детально. Идентификация риска позволяет определить список элементов рисков, который возможен в условиях данного проекта. Определяют проектный (связанный с

ресурсами, формированием требований, форматом взаимодействия с заказчиком), технический (связанный с трудностями проектирования и реализацией программного обеспечения), коммерческий (связанный с рыночной оценкой и необходимостью программного обеспечения) источник рисков. Для более точного определения рисков используют проверочные списки рисков, которые содержат в себе наиболее часто встречающиеся известные риски.

Анализ риска заключает в себе оценку риска по формуле 5, где RE – показатель риска, *P(UO)* – вероятность неудовлетворительного результата, *L(UO)* – потеря при неудовлетворительном риске [16].

$$RE = P(UO) \times L(UO) \quad (5)$$

Вероятность риска определяют с помощью экспертного мнения или же учитывая имеющуюся статистику. После чего данные сводятся в таблицу (таблица 7.1).

Таблица 7.1 – *Пример таблицы данных оценки элементов риска*

Элемент риска	Вероятность, %	Потери	Влияние риска
Наименование риска	5-7	8	20-40

Ранжирование риска позволяет присвоить каждому риску свой уровень приоритета, который пропорционален влиянию элемента на весь проект в целом. Для этого создаётся шкала степени воздействия рисков на проект [17], в которой определяются теоретически возможные изменения в проекте (например, степени влияния «10» соответствует полному провалу проекта, «4» - превышение сроков на 40% от ранее установленных, «1» – отсутствие влияния на проект). Разрешение риска включает в себя три варианта действия с риском:

- принятие риска (риск принимается и никакие действия не предпринимаются);
- понижение риска (риск берётся во внимание, осуществляется поиск решения задачи по снижению уровня его влияние на проект, после чего предпринимаются действия для достижения цели снижения влияния);
- передача третьей стороне (риск принимается во внимание, но дальнейшие действия с ним возлагаются на другое ответственное лицо).

Заключение

В рамках практической работы было реализовано создание приложения для стоматологии, обеспечивающее электронное хранение информации по пациентам и данных, связанных с их лечением. Также проведена автоматизация поиска информации о состоянии и всех видах совершённых процедур над конкретным зубом конкретного пациента, что позволяет упростить работу врача-стоматолога, сократив время и количество действий врача, необходимых для данных процедур.

Приложение было разработано согласно спиралевидной методологии. Для чего был изучен её принцип и необходимые этапы разработки. А именно, анализ требований, создание модели, проектирование и тестирование. После выполнения этих действий был намечен план шагов по реализации приложения на каждом из витков.

Для выявления ожиданий от приложения, был проведён опрос стейкхолдеров с дальнейшей фиксацией персональных требований, которые были доработаны до функциональных и получили свой уровень приоритетности в соответствии с методом MoSCoW. Далее из них была составлена матрица отслеживания выполнения требований.

Деятельности стоматологической клиники была подвержена анализу и фиксации её бизнес-процессов посредством модели «AS-IS» с последующим её реинженирингом в модель «TO-BE». Модели описаны с использованием нотаций ARIS VACD для верхнего уровня и ARIS eEPC для нижних уровней детализации.

По бизнес-модели был определён используемый в деятельности стоматологии массив данных, который претерпел процесс нормализации до

третьей нормальной формы включительно. На основе этих данных определены сущности будущей базы данных с последующей реализацией её в программной среде Microsoft Access 2016. Разработанное приложение было подвергнуто функциональному и нагрузочному тестирование, которые подтвердили готовность продукта.

Список использованных литературных источников

1. Программы для стоматологий – URL: http://www.livemedical.ru/tools/dental/ (дата обращения 12.12.2018)
2. Карпович Е.Е. Жизненный цикл программного обеспечения - М.:Центр дистанционного обучения. НИТУ «МИСиС», 2016 г. – 131 с.
3. Кумагина, Е.А., Неймарк, Е.А. Модели жизненного цикла и технологии проектирования программного обеспечения: учебно-методическое пособие / Е.А. Кумагина, Е.А. Неймарк. – Нижний Новгород: Изд-во ННГУ, 2016. – 41 с.
4. Михайловский Н.Э. Конференция «Теория и практика управления предприятием». Доклад «Сравнение методов оценки стоимости проектов по разработке информационных систем». Дата публикации 20.02.2003
5. Гурендо Д. Жизненный цикл разработки ПО – URL: https://xbsoftware.ru/blog/zhiznennyj-tsykl-razrabotki-spiral/ (дата обращения 12.12.2018)
6. Требования. Анализ требований, виды требований – URL: https://intellect.ml/trebovaniya-analiz-trebovanij-vidy-trebovanij-5188 (дата обращения 12.12.2018)
7. Куликов, С. С. К90 Тестирование программного обеспечения. Базовый курс / С. С. Куликов. – Минск: Четыре четверти, 2017. – 312 с.
8. Методы приоритезации – URL: https://habr.com/company/hygger/blog/359208/ (дата обращения 12.12.2018)
9. Учитель Ю. Г. Разработка управленческих решений / Ю. Г. Учитель, А. И. Терновой, К. И. Терновой. – 2-е изд., перераб. и доп. – М., 2007. – 383 с.
10. Ковалёв С.М., Ковалёв В.М. Статья «Современные методологии описания бизнес-процессов – просто о сложном» – URL: http://www.betec.ru/index.php?id=6&sid=33 (дата обращения 12.12.2018)

11. «Введение в описание бизнес-процессов. Часть 2», статья от 21.07.2014 – URL: http://becmology.ru/blog/business/bp02.htm (дата обращения 12.12.2018)
12. Степанов Д. Ю. Анализ, проектирование и разработка корпоративных информационных систем: уровень бизнес-процессов / МИРЭА. - М., 2017. – URL: https://stepanovd.com/training_erp_1-7_ru.pdf (дата обращения 12.12.2018)
13. Виды тестирования программного обеспечения – URL: http://www.protesting.ru/testing/testtypes.html (дата обращения 12.12.2018)
14. Степанова, Е. А. Основы обработки результатов измерений : [учеб. пособие] / Е. А. Степанова, Н. А. Скулкина, А. С. Волегов ; [под общ. ред. Е. А. Степановой] ; Мин-во образования и науки Рос. Федерации, Урал. федер. ун-т. – Екатеринбург: Изд-во Урал. ун-та, 2014. – 95 с.
15. Варзунов А. В., Торосян Е. К., Сажнева Л. П., Анализ и управление бизнес-процессами // Учебное пособие. – СПб: Университет ИТМО, 2016. –112 с.
16. Орлов С.А., Цилькер Б.Я., Технологии разработки программного обеспечения. Учебник для вузов. 4-е издание. Стандарт третьего поколения // Учебник для вузов. - Издательский дом "Питер", 2012. – 608 с.
17. Галкин Г. Управление рисками. Часть 3. Качественный анализ рисков // Intelligent enterprise – 2005. – №15