Автоматизация работы стоматологической клиники с использованием спиралевидной модели внедрения информационных систем (часть 2)

Худяков Сергей Дмитриевич

Аннотация: в статье анализируется деятельность стоматологической клиники, зафиксированы бизнес-процессы посредством модели «AS-IS» с последующим её реинжинирингом в модель «TO-BE». Модели описаны с использованием нотаций ARIS VACD и eEPC. Используя бизнес-модель, были определены структура данных, а также схема приложения. На основе этой информации разработана программа в среде Microsoft Access.

4. Проектирование приложения

Классический подход к проектированию информационных систем подразумевает моделирование трех составляющих программы: процесс, данные и структуру пользовательских экранов [7]. Начнем с первой – бизнес-процессы. При создании ПО необходимо понять и описать, как происходит функционирование объекта автоматизации. Рассмотрим бизнес-модель объекта, представляющую собой совокупность бизнес-процессов и их взаимосвязей, показывающую внутреннюю сущность объекта в целом, направленную на достижение заранее определенной цели: в нашем случае оказание медицинских услуг стоматологии. Одним из ключевых бизнес-процессов стоматологии является лечение пациентов. Для того, чтобы реализовать ключевой бизнес-процесс стоматологии, проведём проектирование деятельности организации при помощи моделей «AS-IS» и «TO-BE».

Модель «AS-IS» описывает деятельность организации «как есть», то есть осуществляемую на данный момент. Модель строится за счет 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 устранить путем их автоматизации, далее проводится реинжиниринг бизнес-процессов: построение модели «TO-BE» (модель «как будет»). В этой модели учитываются необходимые изменения и тем самым задается усовершенствованный бизнес-процесс [8]. Для проектирования процессов были выбраны две наиболее используемые при

внедрении информационных систем нотации ARIS VACD и ARIS eEPC, задающие верхний и нижние уровни моделирования в CASE-среде Business Studio 4.2.

Верхний уровень бизнес-модели включает в себя четыре ключевых бизнес-процесса: обращение пациента, регистрация пациента, лечение пациента и оплата услуг (рисунок 2). Можно заметить, что уже, начиная с первого уровня детализации, идёт накопление информации, о пациенте в частности. Верхнеуровневая модель имеет схожее представление как в «AS-IS», так и «TO-BE»: особенность нотации ARIS VACD состоит в том, что в ней отсутствуют графические объекты, обозначающие прикладную информационную систему.

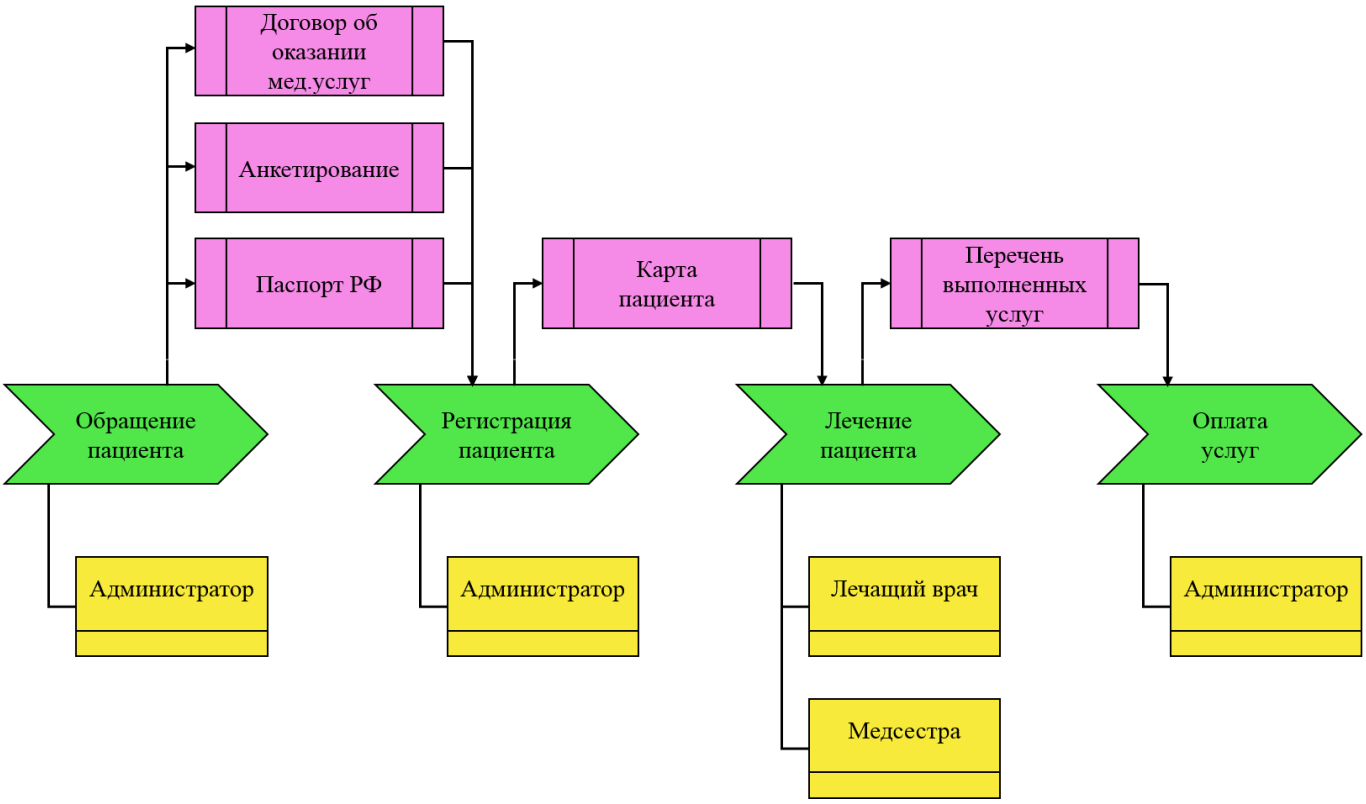

Рис. 2. *Первый уровень процесса «Оказать стоматологические услуги пациенту»*

Далее проводим декомпозицию ключевого процесса лечения пациента, так на этом этапе происходит уточнение самой операции лечения, а также ее результатов. Таким образом, получаем второй уровень детализации согласно нотации моделирования ARIS eEPC (рисунок 3).

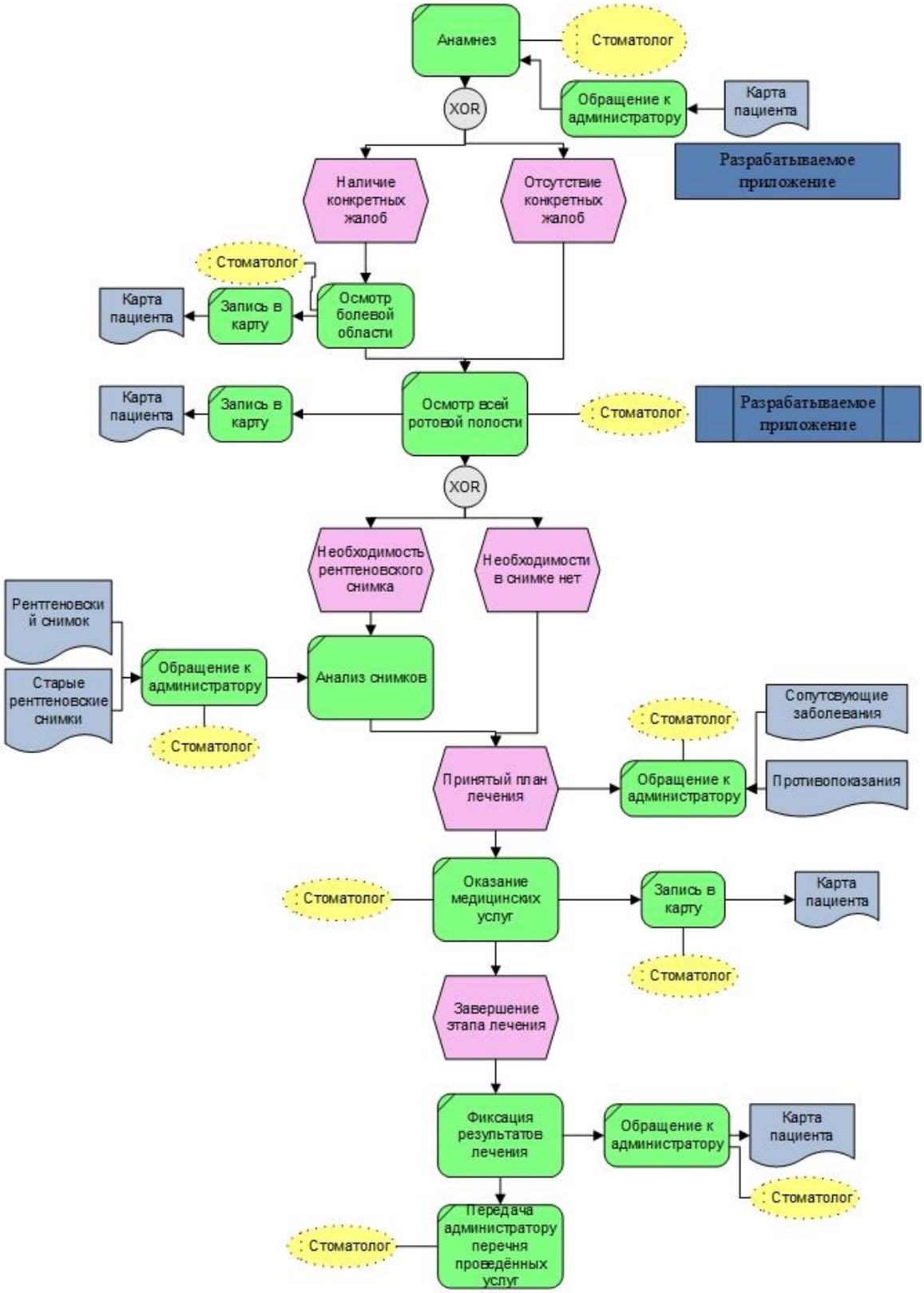

Рис. 3. *Второй уровень процесса «Лечение пациента» в модели «AS-IS»*

Второй уровень отражает конкретику действий врача-стоматолога. Основным были отмечены подпроцессы анамнеза, осмотра ротовой полости, анализа рентген-снимков, оказания медицинских услуг, фиксации результатов лечения и передачи администратору перечня проведённых услуг. Достоинством нотации ARIS eEPC является отображение последовательности процессов, 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 устраняет вышеперечисленные проблемы. После реинжиниринга деятельность стоматологии на этом уровне примет вид, отражённый в модели «TO-BE» рисунка 4.

Выявив требования к приложению и проанализировав бизнес-процессы, проведем анализ данных, которые будут храниться в разрабатываемой программе. Результаты анализа проиллюстрированы в таблице 4.1 в разрезе классов данных, соответствующих ключевым бизнес-процессам.

На стадии проектирования проводится нормализация баз данных будущей системы, позволяющая устранить избыточность, а также снизить риск проявления аномалий увеличения, удаления, модификации и обратимости данных [3]. В данном проекте применялась нормализация до третьей нормальной формы (далее – 3-НФ). Финальный результат продемонстрирован в таблице ниже (табл. 4.2).

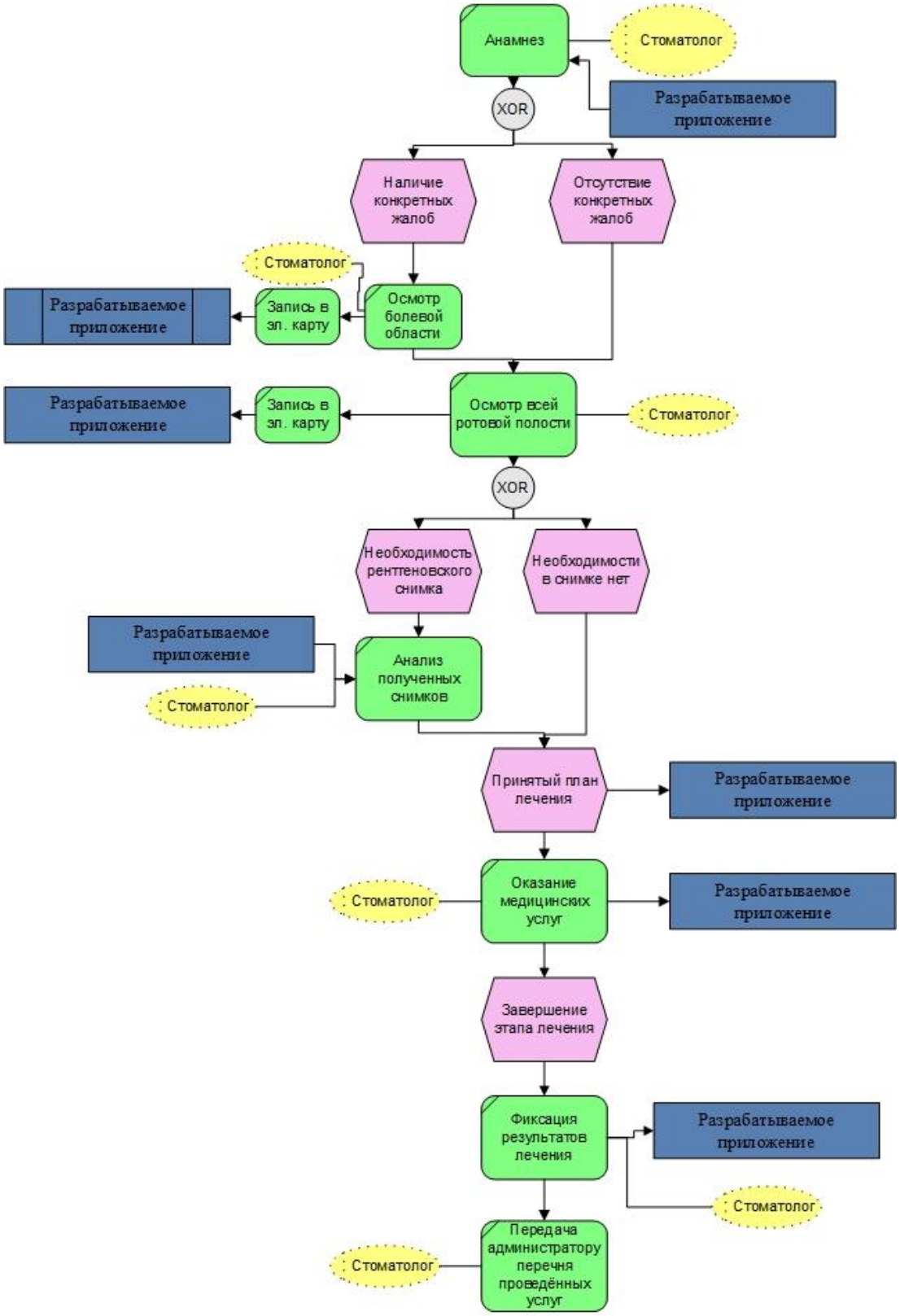

Рис. 4. *Второй уровень процесса «Лечение пациента» в модели «TO-BE»*

Таблица 4.1 Классы данных (до нормализации)

Класс данных	Поле
Пациент	ФИО
	Дата рождения
	Паспортные данные (серия и номер)
	Номер телефона
	Пол
Анамнез	Жалобы
	Дата обращения
Сопутствующие заболевания и противопоказания	Сопутствующие заболевания
	Противопоказания
Карта зубов	Зуб №1-№32
Состояние зуба	Состояние
	Проведенные процедуры
	Использованные препараты
	Необходимость дальнейшего лечения
	Лечащий врач
Рентген снимки	Дата снимка
	Приложенный снимок
Услуги	Название услуги
	Цена
Специалисты	ФИО
	Специализация
	Номер телефона

Таблица 4.2 Структура данных после нормализации до 3-НФ

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

Название класса	Поля данных	Тип данных	Размерность
			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
	Сопутствующие 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
	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

Название класса	Поля данных	Тип данных	Размерность
	ФИО	Текстовый	До 255 символов
	Специализация	Текстовый	До 255 символов
	Номер телефона	Числовой	11 символов

Проведя ряд изменений в начальной структуре данных, получаем финальную архитектуру данных разрабатываемого приложения, которая задана на рисунке 5 в виде ER-диаграмм.

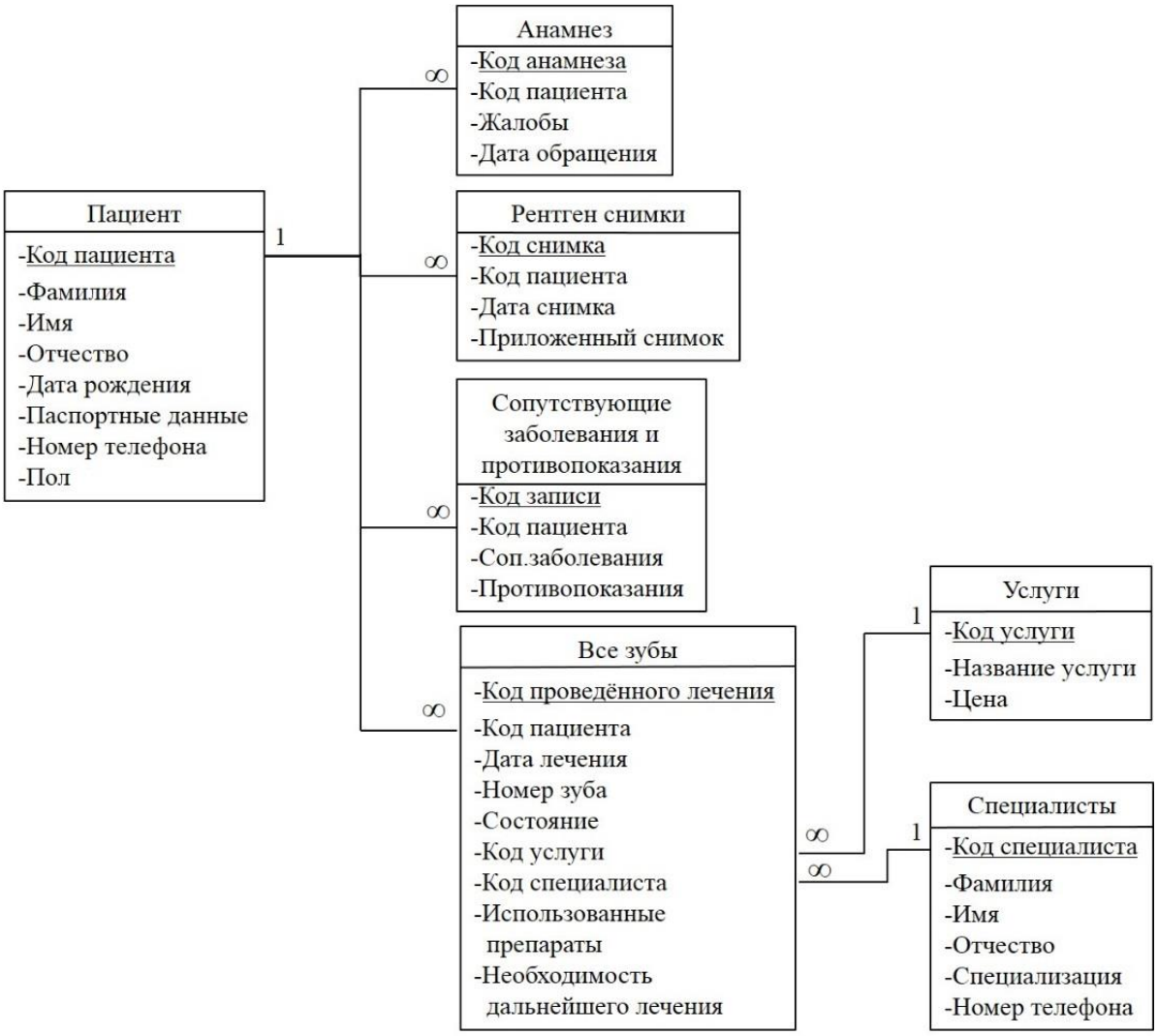

Рис. 5. *Архитектура данных*

Для обеспечения удобной работы пользователя с таблицами баз данных необходимо разработать экранные пользовательские формы, которые позволяют взаимодействовать со всеми функциональными возможностями приложения. Для удовлетворения потребностей пользователя интерфейс программы разрабатывался согласно следующим критериям:

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

Удобство навигации в приложении потребовало введение начального экрана, на котором реализованы ссылки к остальным формам программы. Предполагаемая схема приложения приведена на рисунках 6, 7 и является основной для понимания и ведения последующей разработки.

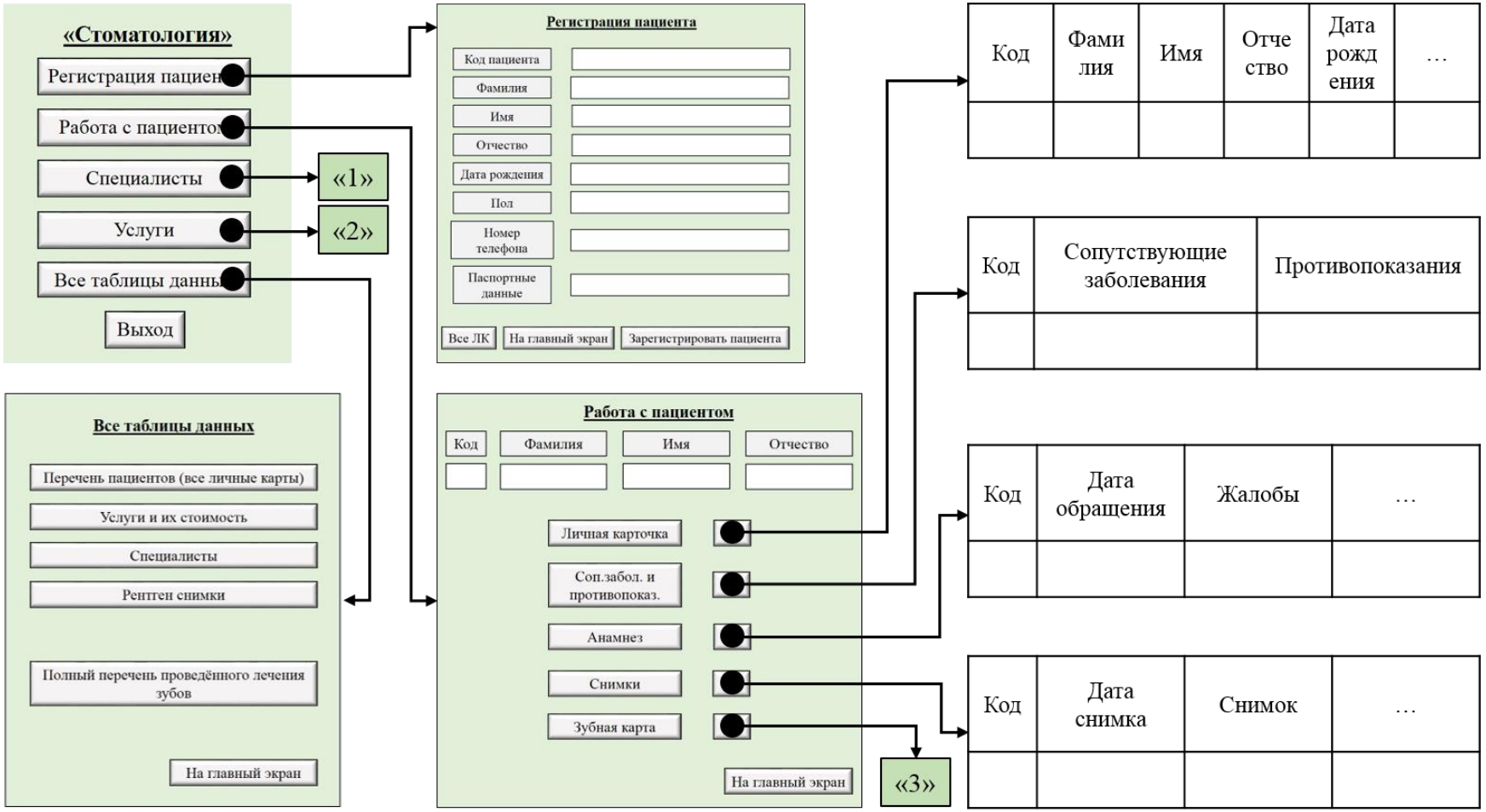

Рис. 6. *Схема приложения (часть 1)*

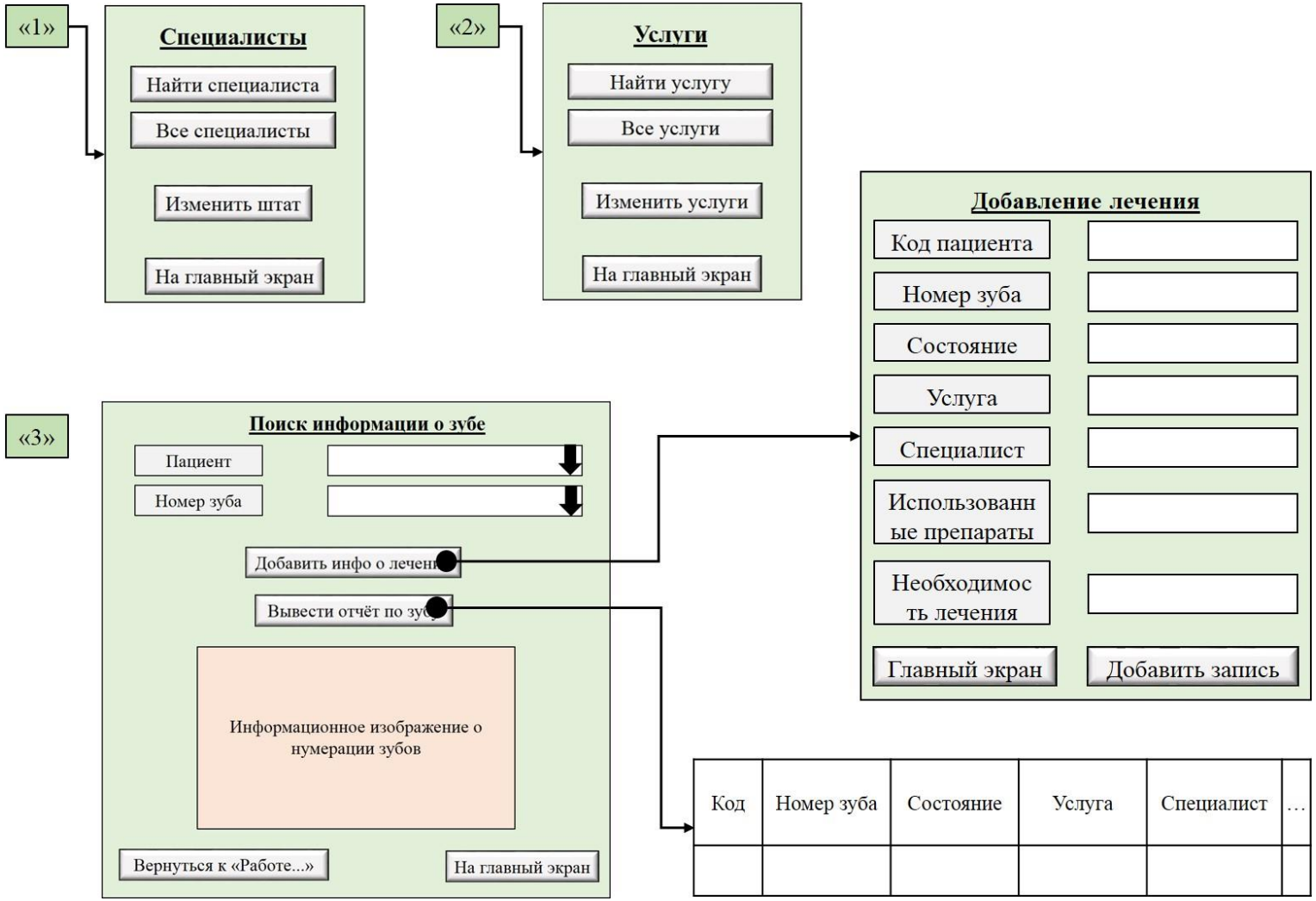

Рис. 7. *Схема приложения (часть 2)*

5. Разработка и тестирование приложения

В соответствии с шагами раздела 2, описывающими распределение требований по итерациям разработки, программировалось приложение в системе управления базами данных Microsoft Access 2016. Основной разработки приложения являлись спроектированные бизнес-процессы, данные и структура пользовательских экранов, приведенные в разделе 4.

На 1-м витке разработки были созданы таблицы баз данных для хранения информации стоматологической клиники. Рисунок 8 содержит схему таблиц данных с имеющимися атрибутами, а также связи между этими таблицами. Как пример рис. 9 показывает созданную таблицу пациентов.

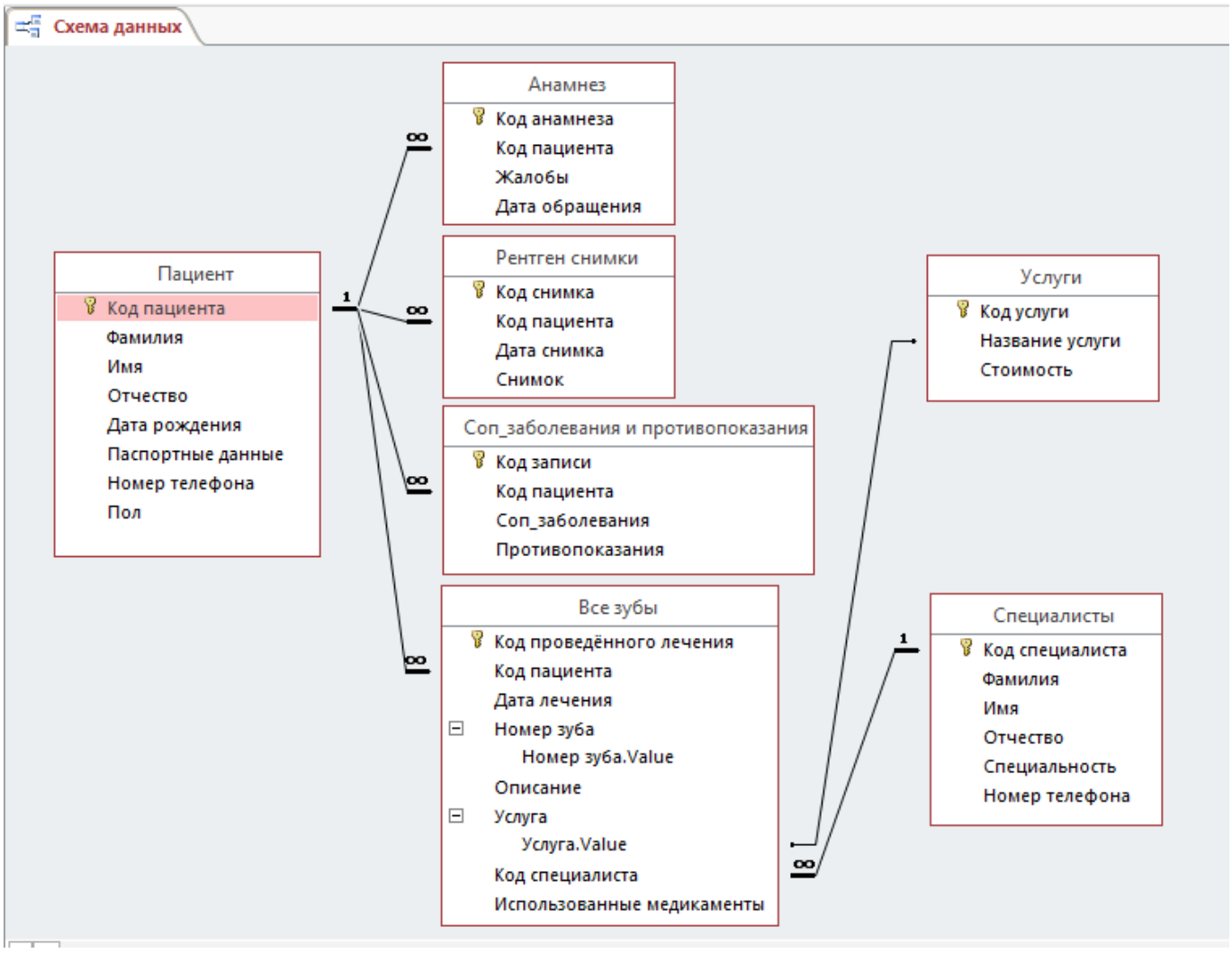

Рис. 8. *Реализованная схема данных*

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	

Рис. 9. *Реализованная таблица «Пациенты»*

Microsoft Access позволяет добавлять, удалять и редактировать хранящуюся в таблицах информацию. Что неудобно для пользователей, особенно когда требуется внести данные о единичном пациенте или ходе его лечения. Для обеспечения редактирования информации на 2-й итерации разработки создается ряд пользовательских форм, которые позволяют управлять данными в таблицах (рисунки 10 – 11).

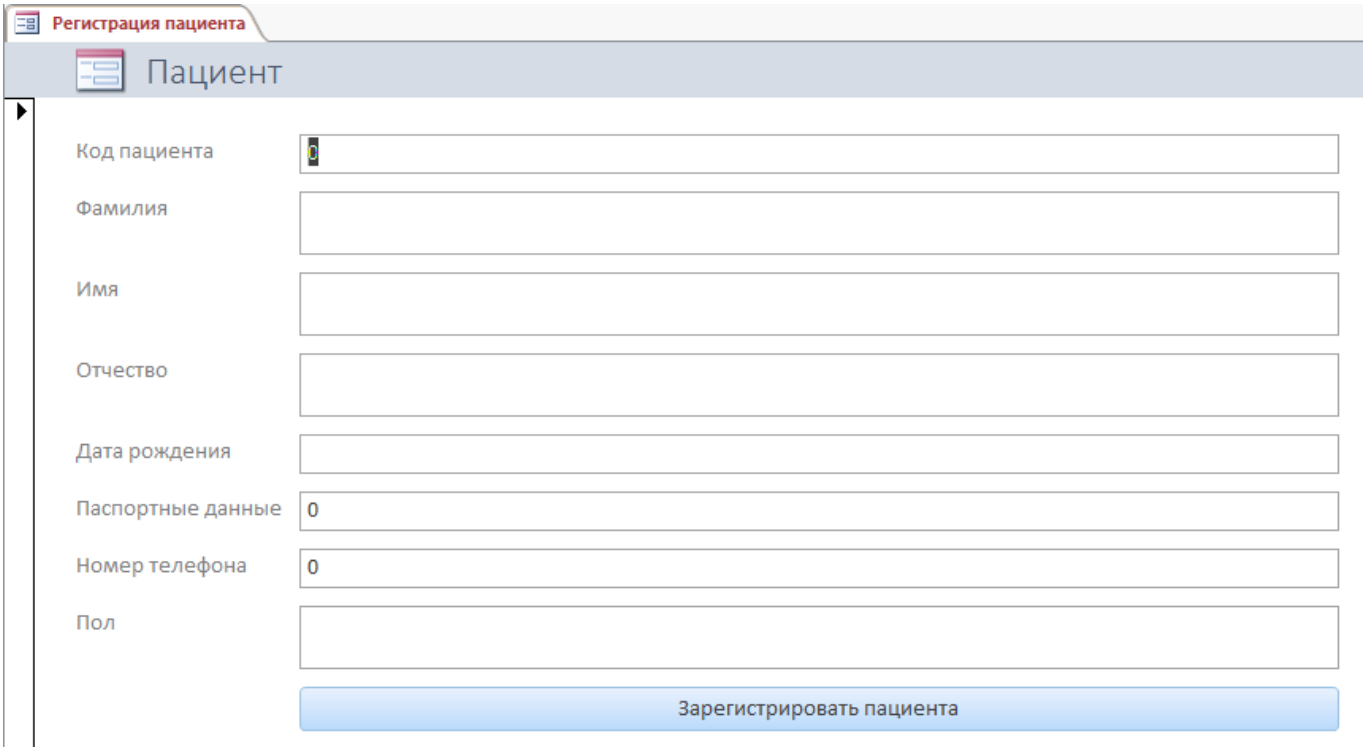

Рис. 10. *Форма редактирования данных «Регистрация пациента»*

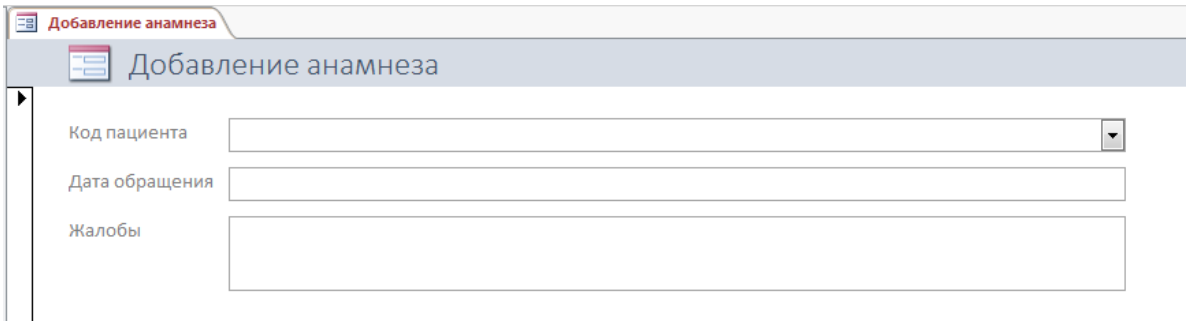

Рис. 11. *Форма редактирования данных «Добавление анамнеза»*

Следуя 3-му витку разработки, перейдём к организации поисковых запросов и интерфейса к ним. Приложение содержит несколько таблиц, поиск информации из которых осуществляется посредством выборки данных на основе критериев, задаваемых самим пользователем. Рассмотрим пример поиска личной карточки пациента. Для поиска карточки организуется запрос (рисунок 12), в условиях которого даны команды, призванные выполнить выборку данных согласно критериям, например, «Like "*" & [Введите код пациента] & "*"». Подобные команды так же запускают диалоговое окно, куда пользователь вводит данные для поиска (рисунок 13). При необходимости поля могут оставаться пустыми, тогда выборка пройдёт без учёта этого критерия.

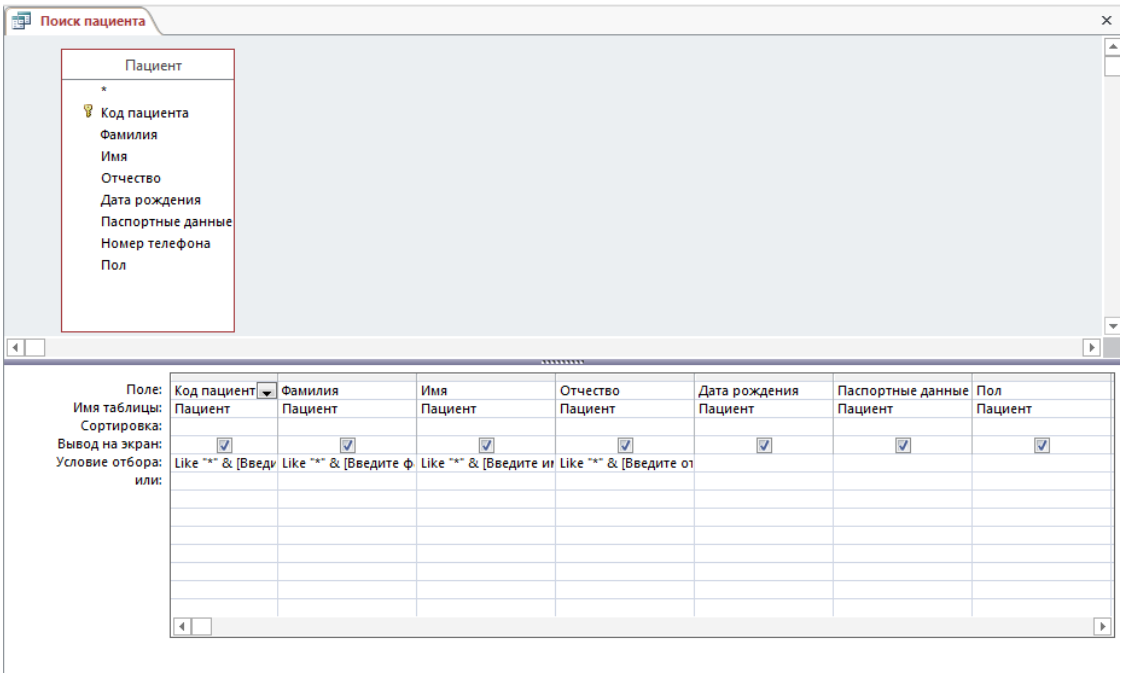

Рис. 12. *Запрос «Поиск пациента»*

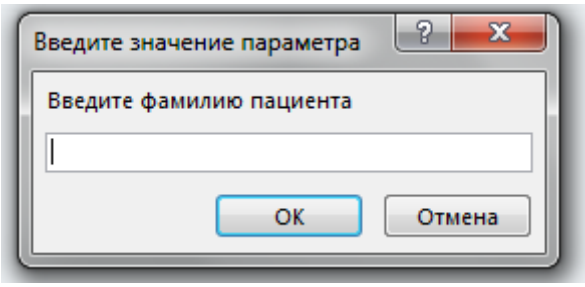

Рис. 13. *Диалоговое окно для ввода параметра выборки «Фамилия»*

Интерфейс приложения организуется с помощью последовательных форм. Форма главного экрана, позволяющая перейти ко всем функциям приложения, продемонстрирована на рисунке 14. Форма зубной карты имеет вид таблицы с перечислением всех зубов, обслуживаемыми в стоматологии. Запрос на поиск требует указания кода пациента и номера необходимого зуба. Для удобства выбора реализованы выпадающие списки (рисунок 15).

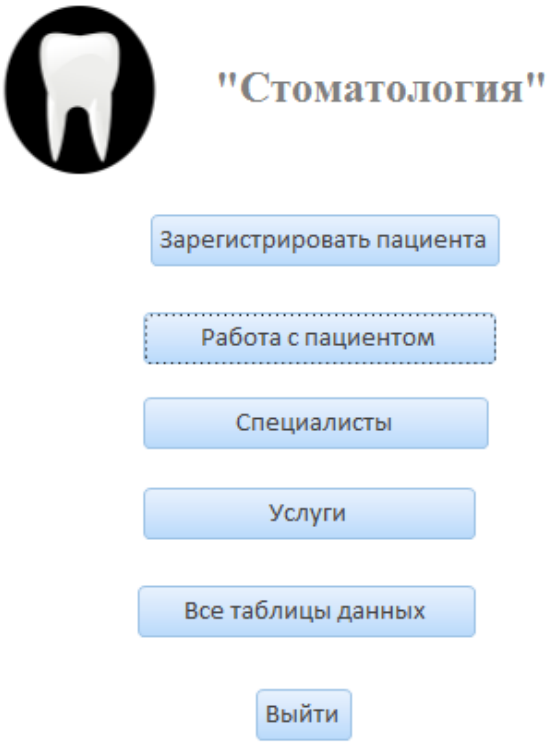

Рис. 14. *Интерфейс главного окна программы*

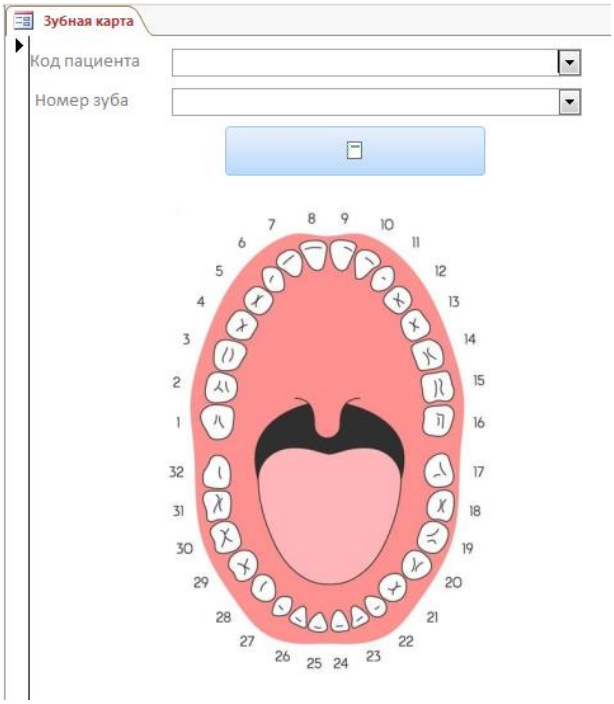

Рис. 15. *Форма «Зубная карта»*

4-я итерация разработки позволяет осуществить представление информации из таблиц, используя инструмент отчетов в Microsoft Access. При этом информация представляется нагляднее, не подлежит изменению и при необходимости может быть распечатана. На рисунке 16 дан пример работы отчёта на основе запроса.

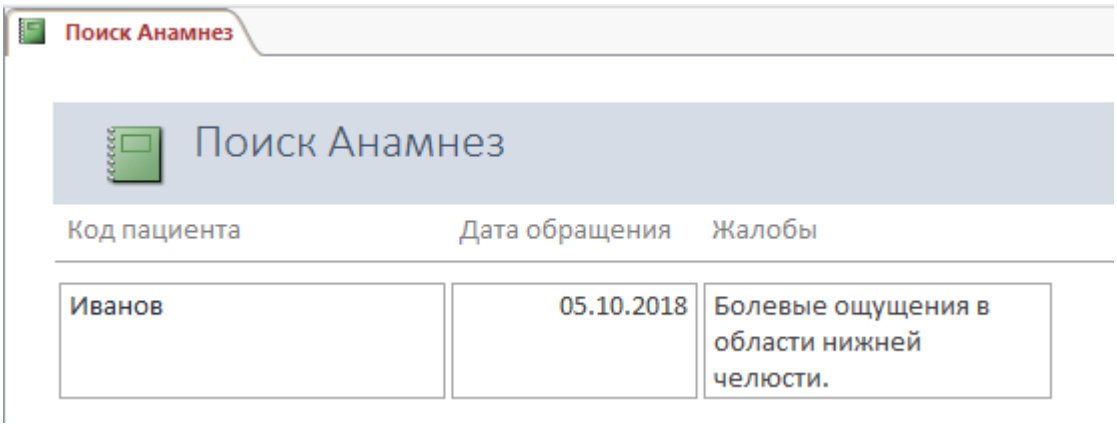

Рис. 16. *Отчёт «Анамнез»*

После создания приложения проводится его тестирование, для выявления проблем в работе, исправления ошибок, а также проверки соответствия начальным

требованиям. Тестирование ПО, полученного как результат реализации 1-4 витков спирали, проводится до передачи готового продукта заказчику. Виды тестирования можно подразделить на три группы [9]: функциональные, нефункциональные и связанные с изменениями. Функциональное тестирование направлено на проверку функций системы. Основной задачей этого типа испытаний является определение соответствия ожидаемого результата, изложенного в функциональных требованиях, с тем, что получено в приложении. Функционально-модульное тестирование выполнялось силами разработчиков как завершающий этап реализации каждого витка спирали.

Тесты, связанные с изменениями, проводятся для того, чтобы удостовериться в том, что работа приложения не нарушена и не влияет на функционирование смежных подсистем и функций. Нефункциональное тестирование направлено на определение характеристик приложения, которые могут быть оценены определёнными численными величинами. В целом, этот тип тестирования показывает, как система работает в различных экстремальных условиях её использования. В контексте проекта проводились нефункциональные испытания, относящийся к категории тестирования системы на производительность: нагрузочный тест [10].

Нагрузочный тест позволяет проверить, как будет функционировать разработанное приложение при разном объеме хранящихся данных. Выбор этого вида тестирования обусловлен тем, что основной задачей программного обеспечения является поиск и показ искомой информации из таблиц баз данных, так как быстрота работы программы напрямую влияет на скорость обслуживание пациентов. В ходе эксперимента количество тестовых записей задавалось величинами 1, 10 и 100. Испытывались операции сохранения записи и поиска информации с выводом в отчёт. Оценка времени выполнения операций рассчитывалась как среднее арифметическое $t_{ср}$ по формуле (1):

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

где t_i – время одного замера, n – количество замеров (n = 5). Среднеквадратичное отклонение замера вычислялось согласно (2):

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

Итоговая погрешность замера определялась по формуле (3):

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

где P – доверительный коэффициент Стьюдента (P = 0,95), A – абсолютная погрешность средства измерения (A = 0,0005). Итоговое значение времени отклика задается (4):

$$t_{\text{отк}} = t_{\text{ср}} \pm \Delta t. \quad (4)$$

Результаты расчета времени отклика с учетом погрешности замера для операций сохранения и поиска записей в таблице баз данных приведены в табл. 5.1. Наблюдается увеличение времени отклика программы при увеличении числа обрабатываемых записей. Однако, значение времени отклика достаточно мало и не нарушает работу приложения при обработке единичной записи.

Таблица 5.1 Результаты нагрузочного тестирования

Кол-во записей	Действие	t_1, сек	t_2, сек	t_3, сек	t_4, сек	t_5, сек	Время отклика, сек
1	Запись	0,11	0,13	0,1	0,1	0,11	0,110 ± 0,005
	Поиск	0,1	0,12	0,11	0,14	0,11	0,116 ± 0,006
10	Запись	0,13	0,11	0,14	0,11	0,12	0,122 ± 0,005
	Поиск	0,16	0,12	0,13	0,11	0,14	0,132 ± 0,007
100	Запись	0,13	0,12	0,12	0,12	0,13	0,124 ± 0,002
	Поиск	0,2	0,21	0,19	0,19	0,2	0,198 ± 0,003

6. Заключение

В рамках проделанной работы было реализовано приложение для стоматологии, обеспечивающее электронное хранение информации пациентов и данных, связанных с их лечением. Также проведена автоматизация процесса поиска информации о состоянии пациента и совершённых процедурах над конкретными зубами пациента, что позволяет упростить работу врача-стоматолога, сократив время и количество действий.

Приложение было разработано согласно спиралевидной методологии. Для чего был изучен её принцип и необходимые этапы реализации: анализ требований, создание модели процессов, данных и приложения, разработка и тестирование. Был подготовлен план действий по реализации приложения на каждом витке разработки. Для выявления ожиданий от приложения, был проведён опрос стейкхолдеров с дальнейшей фиксацией пользовательских требований, которые были приоритизированы в соответствии с методом MoSCoW. Далее из них был составлен бэклог требований.

Проанализирована деятельность стоматологической клиники, зафиксированы бизнес-процессы посредством модели «AS-IS» с последующим её реинжинирингом в модель «TO-BE». Модели описаны с использованием нотаций ARIS VACD для верхнего уровня и ARIS eEPC для нижних уровней детализации. Используя бизнес-модель, были определены структура данных, нормализованная до третьей нормальной формы включительно, а также схема приложения. На основе этой информации разработана программа в среде Microsoft Access. Реализованное приложение было испытано путем проведения функционального и нагрузочного тестирований, которые подтвердили готовность и качество продукта.

Литература

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. – 131 с.
3. Кумагина Е.А., Неймарк Е.А. Модели жизненного цикла и технологии проектирования программного обеспечения: учебно-методическое пособие – Нижний Новгород: Изд-во ННГУ, 2016. – 41 с.
4. Гурендо Д. Жизненный цикл разработки ПО – URL: https://xbsoftware.ru/blog/zhiznennyj-tsykl-razrabotki-spiral/ (дата обращения 12.12.2018).
5. Требования. Анализ требований, виды требований – URL: https://intellect.ml/trebovaniya-analiz-trebovanij-vidy-trebovanij-5188 (дата обращения 12.12.2018).
6. Методы приоритизации – URL: https://habr.com/company/hygger/blog/359208/ (дата обращения 12.12.2018).

7. Степанов Д.Ю. Анализ, проектирование и разработка корпоративных информационных систем: теория и практика // Российский технологический журнал. – 2015. – т.8, №3. – с.227-238.
8. Разработка управленческих решений / Ю.Г. Учитель, А.И. Терновой, К.И. Терновой. – 2-е изд., перераб. и доп. – М., 2007. – 383 с.
9. Виды тестирования программного обеспечения – URL: http://www.protesting.ru/testing/testtypes.html (дата обращения 12.12.2018).
10. Куликов С. С. Тестирование программного обеспечения. Базовый курс. – Минск: Четыре четверти, 2017. – 312 с.

Выходные данные статьи

Худяков С.Д. Автоматизация работы стоматологической клиники с использованием спиралевидной модели внедрения информационных систем (часть 2) // Корпоративные информационные системы. – 2021. – №2 (14) – С. 1-20. – URL: https://corpinfosys.ru/archive/issue-14/95-2021-14-dentalautomation.

Об авторе

Худяков Сергей Дмитриевич – студент 4-го курса кафедры оптических и биотехнических систем и технологий физико-технологического института РТУ МИРЭА. Тема выпускной квалификационной работы бакалавра «Автоматизация ключевых бизнес-процессов стоматологической клиники с использованием спиралевидной модели внедрения». Электронная почта: khudyakov.sd@yandex.ru.