



**МИНОБРНАУКИ РОССИИ**  
**федеральное государственное бюджетное образовательное учреждение**  
**высшего образования**  
**«Московский государственный технологический университет**  
**«СТАНКИН» (ФГБОУ ВО МГТУ «СТАНКИН»)**

---

**Институт**  
информационных технологий

**Кафедра**  
информационных систем

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Управление данными»  
на тему: Проектирование БД стоматологии

**Студент**  
группа ИДБ–20–07

**Руководитель**  
старший преподаватель

ПОДПИСЬ

**Казакова Е.Д.**

ПОДПИСЬ

**Быстрикова В. А.**

Москва 2022 г.

## ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.....	3
1. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ.....	4
1.1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ.....	4
1.2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ.....	13
1.3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ.....	17
1.4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ.....	23
2. ОПИСАНИЕ ФУНКЦИОНИРОВАНИЯ БАЗЫ ДАННЫХ.....	32
2.1. НАЗНАЧЕНИЕ И ПЕРЕЧЕНЬ ФУНКЦИЙ БАЗЫ ДАННЫХ.....	32
2.2. ОПИСАНИЕ РАБОТЫ С БАЗОЙ ДАННЫХ.....	33
ЗАКЛЮЧЕНИЕ.....	52
СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ.....	54
ПРИЛОЖЕНИЕ А. ЗАПОЛНЕНИЕ ТАБЛИЦ ДАННЫМИ.....	55
ПРИЛОЖЕНИЕ Б. SQL-КОМАНДЫ СОЗДАНИЯ ПРОГРАММНЫХ ОБЪЕКТОВ БД.....	63
ПРИЛОЖЕНИЕ В. КОД ПРОГРАММЫ.....	70

## **ВВЕДЕНИЕ**

Объектом исследования курсовой работы является стоматология. Стоматология, также известная как стоматология и медицина полости рта, является разделом медицины, специализирующимся на зубах, деснах и полости рта. Она состоит из изучения, диагностики, профилактики и лечения заболеваний, расстройств и состояний полости рта, чаще всего сосредоточенных на зубочелюстной системе (развитии и расположении зубов), а также на слизистой оболочке полости рта [1]. Стоматология может также охватывать другие аспекты черепно-лицевого комплекса, включая височно-нижнечелюстной сустав. Практикующий врач называется стоматологом. Объектом исследования в данном проекте стала проблема организации специализированного лечебно-профилактического учреждения – стоматологии [2].

Данная тема актуальна по ряду причин: на сегодняшний день во многих областях либо проводится цифровизация, либо уже проведена и активно используется, также для обеспечения оперативности ведения информации о деятельности стоматологии и обслуживания больных необходима автоматизированная система, основанная на современной базе данных.

Для выполнения курсовой работы необходимо провести анализ предметной области и выявить функции, которые выполняет аптека. На начальном этапе проектирования нам необходимо выявить основное содержание базы данных без ориентации на какую-либо СУБД. Следующим шагом к выполнению курсового проекта является логическое проектирование. На данном этапе необходимо построить ER-диаграммы, где будет показано, как разные сущности связаны между собой в системе. Следующим этапом является физическое проектирование. На этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения. После этого необходимо создать интерфейс для работы с базой данных.

# **1. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ**

## **1.1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметная область – это совокупность объектов реального или предполагаемого мира, рассматриваемых в пределах данного контекста, который понимается как отдельное рассуждение, фрагмент научной теории или теории в целом.

Деятельность, направленная на выявление реальных потребностей заказчика, а также на выяснения предметной области. Анализ предметной области – это первый шаг этапа системного анализа, с которого начинается разработка программной системы.

Анализ предметной области, позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта. Модель предметной области должна быть документирована, храниться и поддерживаться в актуальном состоянии до этапа реализации.

Анализ предметной области является основой для анализа осуществимости проекта и определения образа (концепции) продукта и границ проекта.

Предметной областью в данной самостоятельной работе является деятельность стоматологии.

Стоматологическая клиника является специализированным лечебно-профилактическим учреждением, предназначенным оказывать медицинскую помощь и осуществлять комплекс профилактических мероприятий.

Деятельность учреждений частной системы здравоохранения осуществляется в соответствии с настоящими Основами, другими актами законодательства Российской Федерации, республик в составе Российской Федерации, правовыми актами автономной области, автономных округов, краев, областей, нормативными актами Министерства здравоохранения Российской Федерации, министерств здравоохранения республик в составе Российской Федерации и органов местного самоуправления.

Для приема и регистрации медицинских услуг в стоматологии имеется отдел по вводу и обработке данных выполненных объемов медицинской помощи, возглавляемый начальником, который осуществляет контроль над работой отдела обработки данных и вместе с руководителем медицинского учреждения несет ответственность за правильную организацию работы, а также четкое медицинское обслуживание клиентов частной стоматологии.

Лечебное учреждение сталкивается с необходимостью работать с большим количеством информации:

- ввод пациентов в базу данных частной стоматологии;
- ведение данных о врачах;
- регистрация, услуг, оказываемых пациенту;
- составление отчета об оказанных услугах за период времени по пациентам;
- истории болезней, зубной формулы пациентов;
- учет используемого в процессе работы материала, складской учет;
- создание графика работы, записи пациентов.

Становится понятно, что вести учет вручную практически невозможно, поэтому необходимо использовать информационные технологии, позволяющие автоматизировать деятельность всех уровней предприятия.

Автоматизация предприятия будет включать в себя все ранее перечисленные требования к учету.

Отчеты, сформированные с помощью базы данным предназначены для руководства клиники, а также ежемесячного отчета организациям, с которыми заключены договоры.

Наиболее известным и удобным является программное обеспечение от компании Простой софт – конфигурация «Стоматология» [3].

Конфигурация "Стоматология" предназначена как для стоматологических клиник и медицинских центров, так и для частнопрактикующих врачей-стоматологов. Данная программа позволит оптимизировать работу клиники и упростить процесс обслуживания пациентов благодаря ведению всестороннего учета. В функционал программы

"Стоматология" входит:

- ведение данных пациентов (рис.1.1), врачей, страховых компаний и оказываемых услуг.

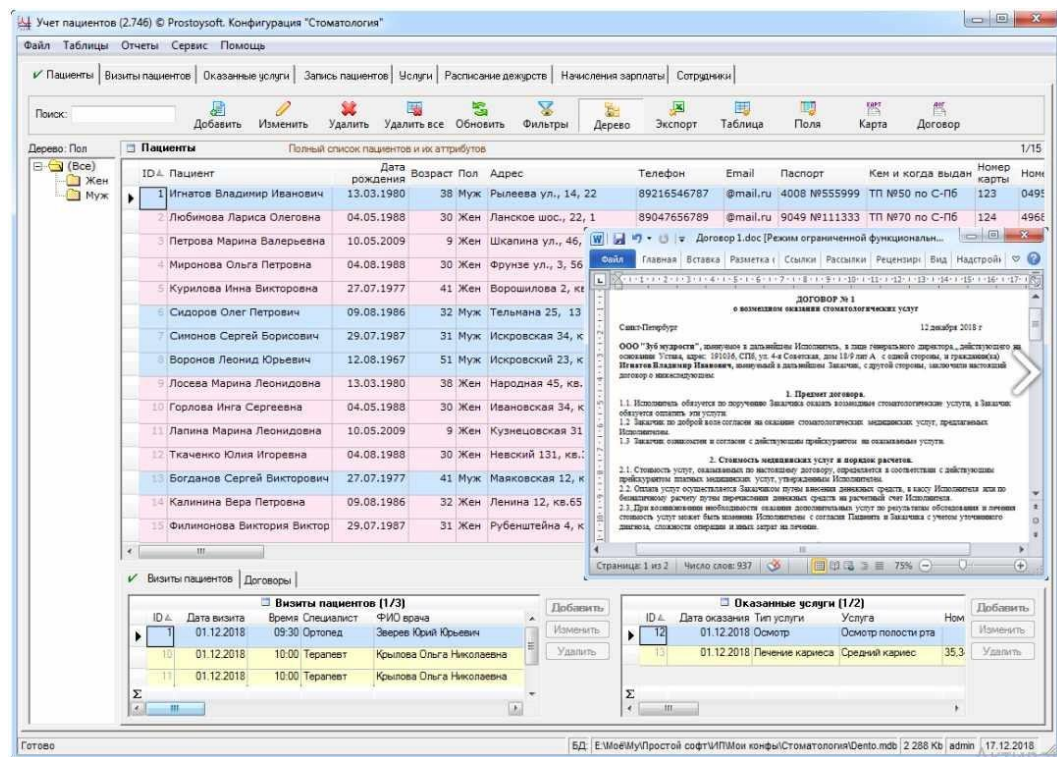


Рис.1.1 Ведение данных пациентов

- Формирование предварительной записи пациентов на прием (рис.1.2) с учетом графика дежурств врачей (рис.1.3).

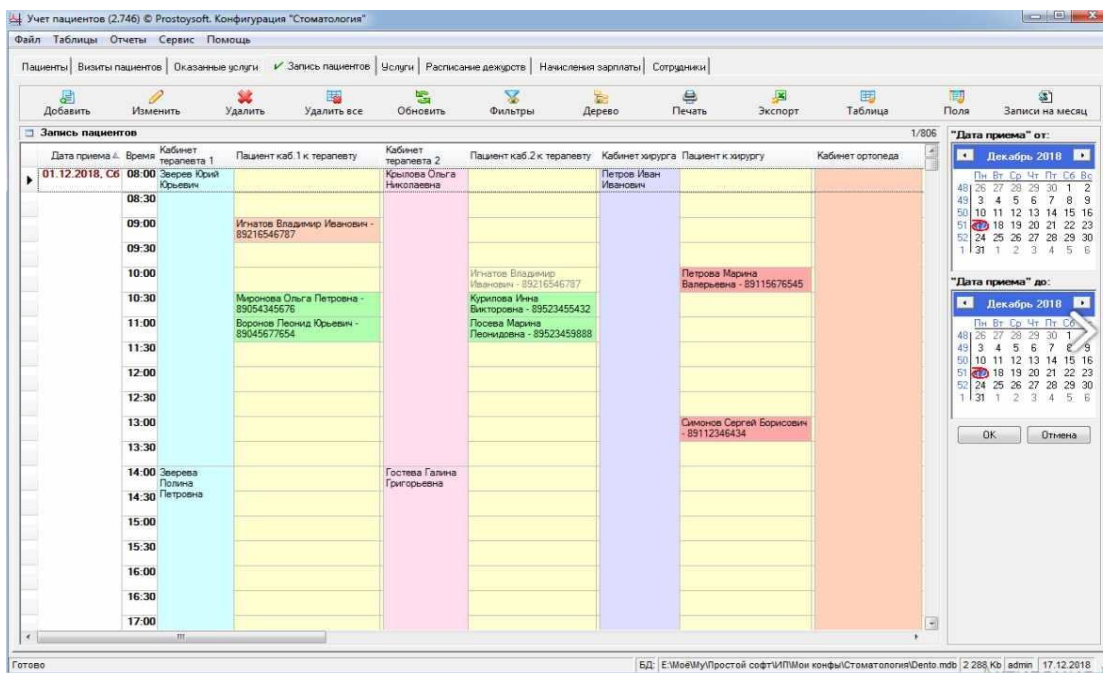


Рис.1.2 Ведение предварительной записи пациентов

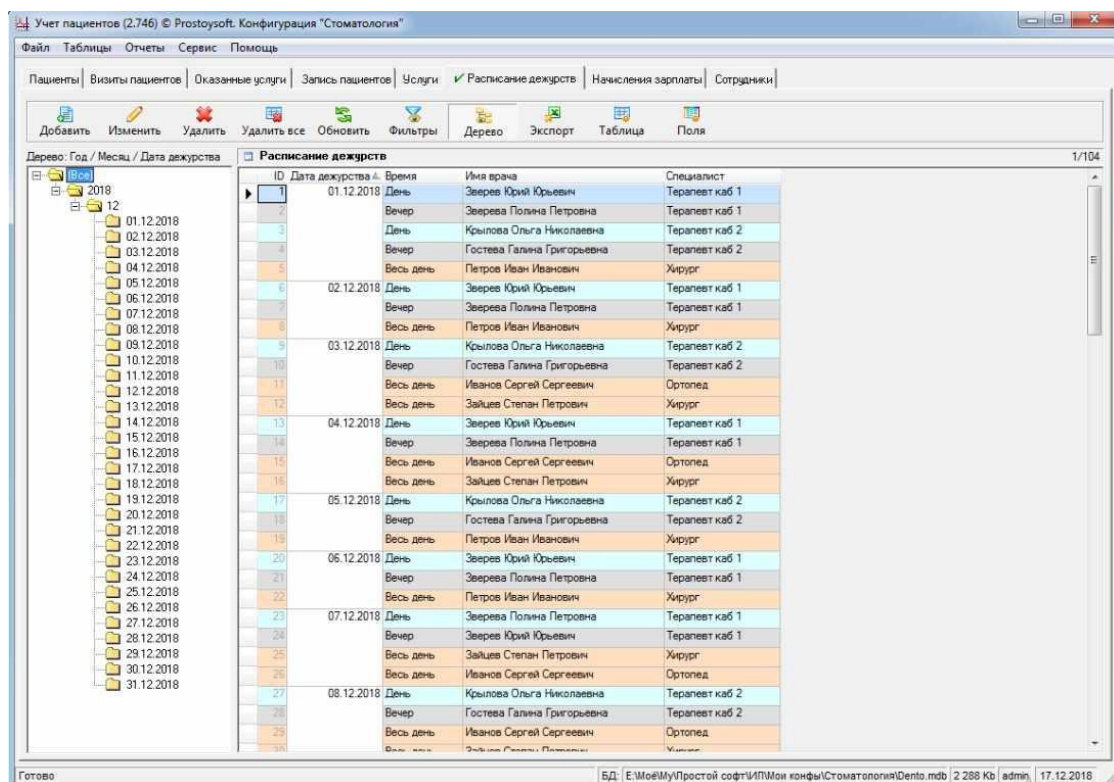


Рис.1.3 Учет графика дежурств врачей

- Контроль оказанных стоматологических услуг, хранение истории лечения каждого зуба пациента (рис.1.4).

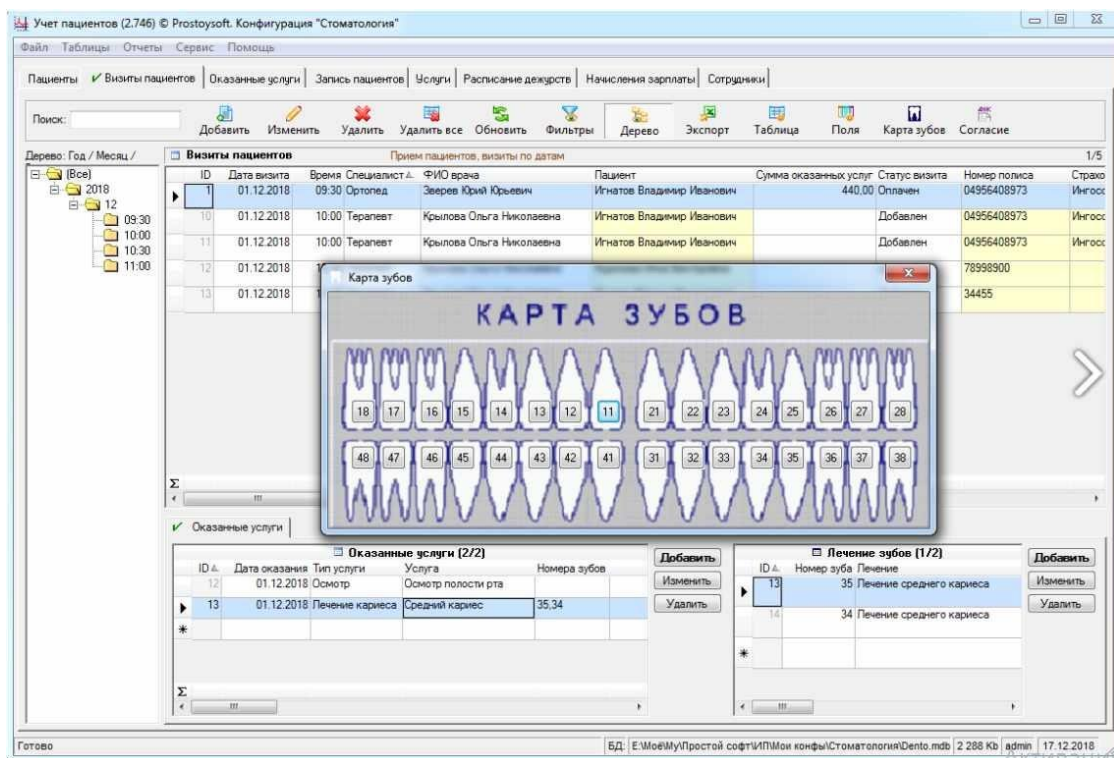


Рис.1.4 Зубная формула пациента

- Расчет заработной платы врачей с учетом процентов от оказанных услуг (рис.1.5).



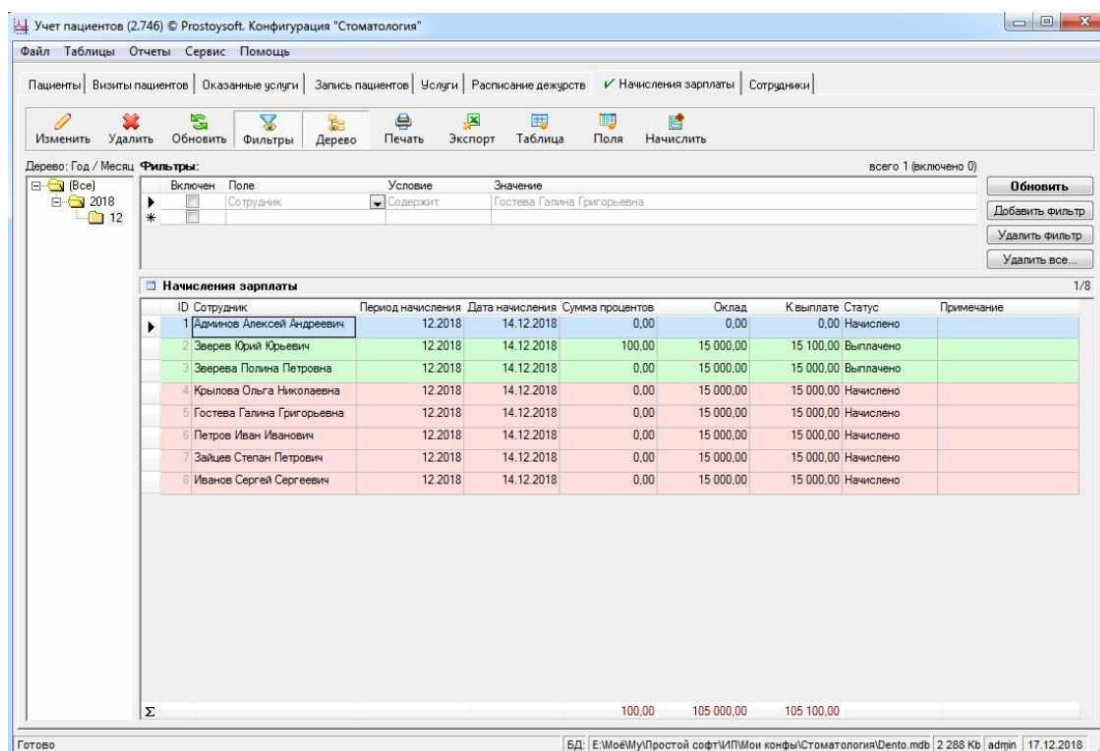


Рис.1.5 Расчет заработной платы врачей

- Генерация амбулаторных карт, договоров, согласий пациентов на осмотр и лечение (рис.1.6).

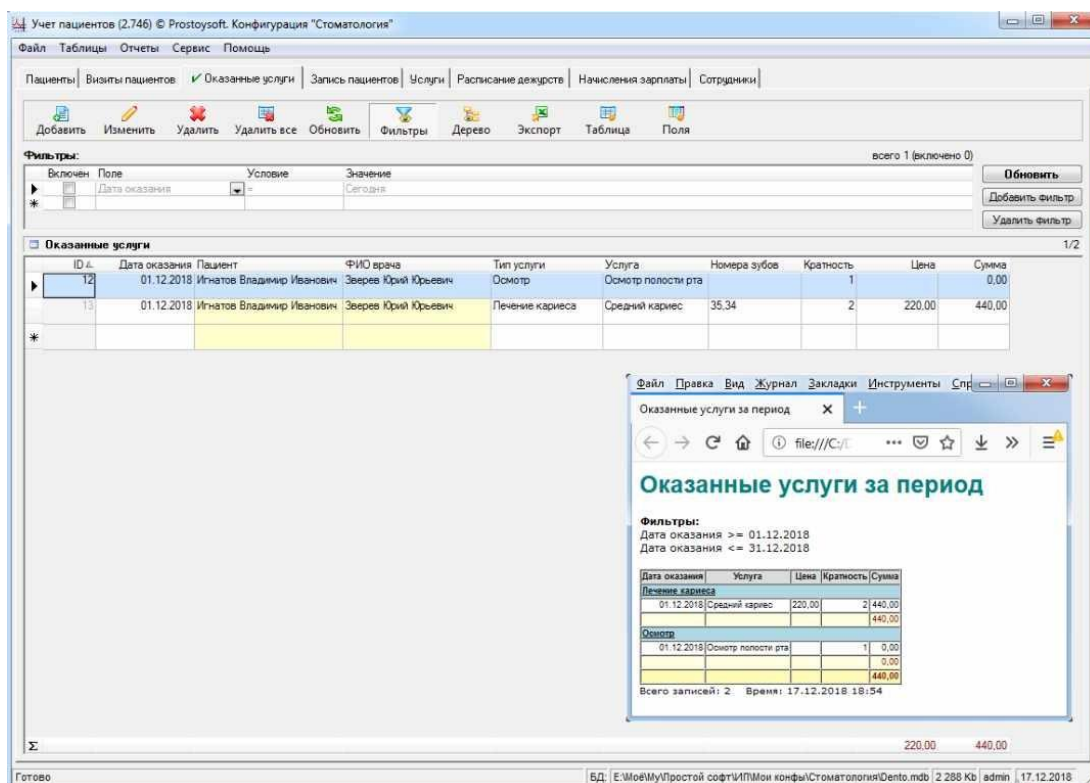


Рис.1.6 Генерация амбулаторных карт пациента

В качестве второго программного продукта рассмотрим 1С:Медицина «Стоматологическая клиника» [4].



Программный продукт предназначен для автоматизации большинства бизнес-процессов как частных практикующих врачей-стоматологов, так и стоматологических центров и клиник, в том числе крупных сетей, состоящих из множества клиник, работающих обособленно или в рамках единой сети с общей базой клиентов, лицевых счетов и бонусных начислений. Среди основных функций данной программы стоит выделить:

- Графическое представление зубной формулы (рис.1.7).

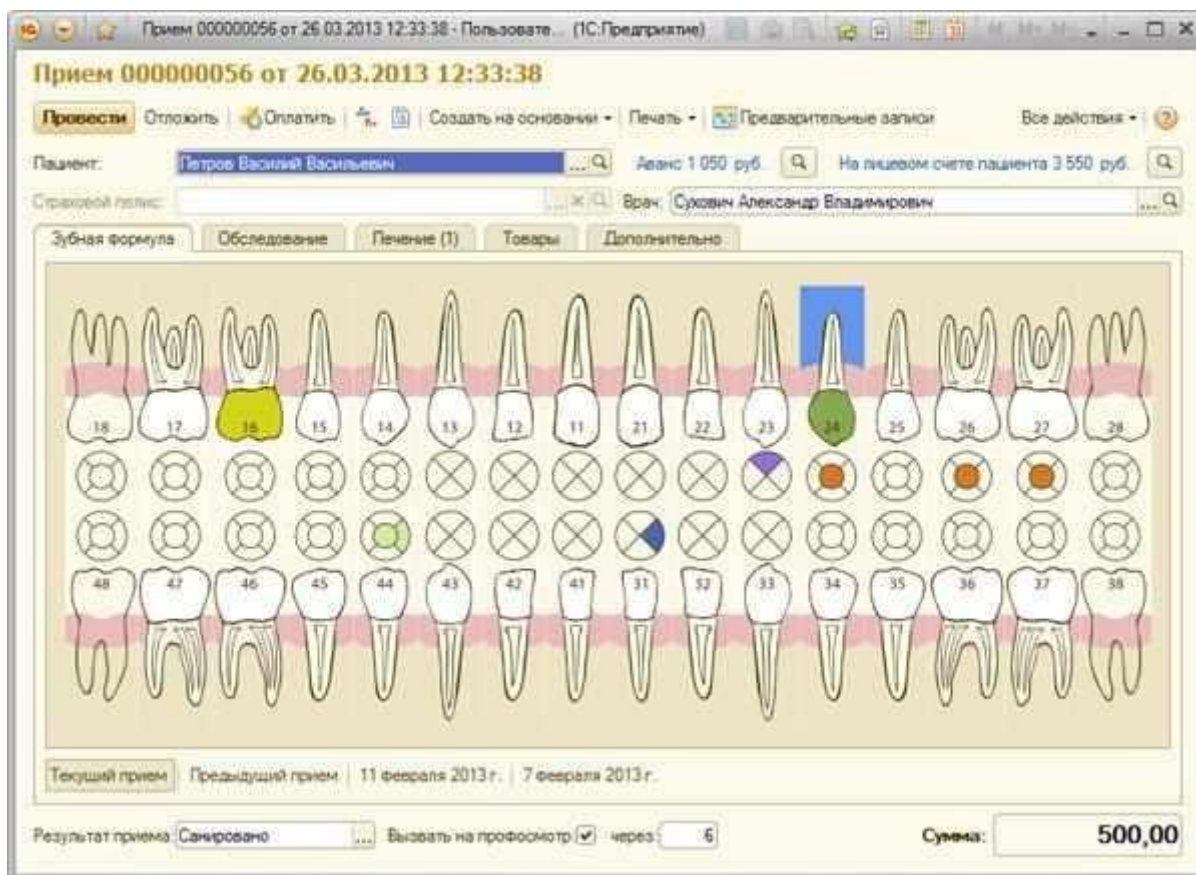


Рис.1.7 Графическое представление зубной формулы

- Журнал предварительной записи с возможностью указать дату, время, врача, услугу, кабинет и служебные метки (рис.1.8).



- Формирование заказов поставщикам (рис.1.10).

Перемещение запасов 00000000001 от 04.02.2013 12:00:00 - Пользователь: (ИС Предприятие)

Провести и закрыть | Провести | Печать

Номер: 00000000001 | Дата: 04.02.2013 12:00:00 | Вид операции: Перемещение между складами

Склад отправитель: Склад материалов | Склад получатель: Витрина

Структурная ед.: Клиника "32 зуба" | Стр. ед. получатель: Клиника "32 зуба"

Запасы (7)

N	Номенклатура	Характеристика, серия	Количество	Упак.
1	Зубная паста, Royal Denta Gold	«не используются»	2,000	
2	Зубная паста, Royal Denta Sensitive	«не используются»	2,000	
3	Зубная паста, Royal Denta Silver	«не используются»	2,000	
4	Зубная щетка, Royal Denta Gold M...	«не используются»	2,000	
5	Зубная щетка, Royal Denta Gold Soft	«не используются»	2,000	

Комментарий:

Рис.1.10 Заказы поставщикам

- Формирование отчетов (рис.1.11).

Прибыль - Пользователь: Русланов, Структурная единица: Клиника "32 зуба" (ИС Предприятие)

Вариант отчета: Прибыль

Сериализовать | Настройка

Начало периода: 01.01.2013 | Конец периода: 31.01.2013 | Начало следующего месяца: 01.02.2013

Структурная единица: Равно | Номенклатура: Равно

**Прибыль**

Структурная единица	Количество	Выручка	Себестоимость	Зарплата	Затраты	Прибыль	Рентабельность, %
Интрансформационная анестезия	7	1 400.00	342.00	300.00		758.00	54.14
Сетевые пломбы	5	1 500.00	160.00			840.00	84.00
Временная пломба	7	1 750.00	1 000.00	325.00		417.00	23.83
Распломбировка одного корневого канала	7	4 500.00		800.00		3 700.00	82.22
Пломба из светополнереабilitating композита	6	7 700.00	952.00	1 430.00		5 318.00	69.06
Парадентология	2	7 500.00	150.00	1 500.00		5 842.00	77.89
Препарация	1	500.00	150.00	100.00		242.00	48.40
Лескутная операция	1	7 000.00		1 400.00		5 600.00	80.00
Аренда					15 000.00	-15 000.00	0
<b>Итого</b>	<b>44</b>	<b>34 200.00</b>	<b>2 078.00</b>	<b>5 852.50</b>	<b>15 000.00</b>	<b>11 269.50</b>	<b>32.95</b>

Рис.1.11 Формирование отчетов

Для рассмотренных программных продуктов следует выделить общие функции:

- Ведение учетной записи пациента с учетом данных зубной формулы.
- Ведение учетной записи врача, составление графика работы и дежурств.

- Расчет заработной платы врача.
- Редактирование календаря записи пациентов.

Особенностью программного продукта 1С:Медицина «Стоматологическая клиника» является формирование отчетов. Следует отметить, что руководителю полезен функционал оценки рентабельности работы, механизмов подсчета прибыли и наличие информации по статье затрат.

После проведения анализа предметной области был выделен перечень функций, которые будут реализованы в данной работе:

- Учет пациентов и приемов.
- Ведение зубной формулы пациентов.
- Ведение историй лечения.
- Учет материалов.
- Составление расписания работ сотрудников.

## 1.2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Цель концептуального проектирования – определение содержания базы данных. Концептуальное проектирование заключается в формализованном описании предметной области, чтобы можно было проанализировать корректность схемы БД, учитывая отсутствие привязки к СУБД. На этом этапе необходимо проанализировать задачи и построить диаграмму вариантов использования (ДВИ).

ДВИ нужна для формализации функциональных требований к системе с помощью действующих личностей и вариантов использования. Суть данной диаграммы состоит в следующем: проектируемая система представляется в виде множества объектов, взаимодействующих с системой с помощью вариантов использования. Объектом называется любой объект, субъект или система, взаимодействующая с разрабатываемой системой извне. Вариант использования – это спецификация функций, которые система предоставляет объекту. Каждый вариант использования определяет некоторый набор действий, совершаемых системой при взаимодействии с объектом. При этом в модели никак не отражается то, каким образом будет реализован этот набор действий.

Составим словесное описание проектируемой БД.

По телефону у администратора (главного администратора) потенциальные клиенты могут получить справки об оказываемых услугах, их ценах и, при желании, записаться на прием к врачу, а также отменить запись.

Клиент, пришедший в стоматологию, также может получить информацию об услугах, их ценах и, при желании, записаться на прием или отменить запись.

Администратор производит записи и отмену записей на прием к врачу, выдает информацию об услугах и их ценах. Также администратор проводит взаиморасчеты с клиентами после факта оказания услуги.

Главный администратор имеет такие же функции, что и администратор, однако, дополнительно, имеет возможность формировать расписание врачей, медсестёр (медбратьев).

Врач имеет возможность вести истории лечения пациента (клиента), которые включают ведение зубной формулы клиента, получать свое расписание, сформированное главным администратором.

Медсёстры (медбратья) производят учет расходуемых материалов, могут вести зубную формулу клиента и имеют возможность получать свое расписание, сформированное главным администратором.

С проектируемой системой будут взаимодействовать следующие действующие лица:

1. Администратор.
2. Главный администратор.
3. Врач.
4. Медсестра (медбрат).

В реальной жизни клиент стоматологии может просматривать электронный каталог услуг, проводить процедуру предварительной записи на прием к врачу. Однако в проектируемой системе такой возможности у него нет, поэтому клиент не будет являться действующим лицом.

Рассмотрим варианты использования каждого из действующих лиц.

Администратору доступны следующие функции:

1. Запись пациента на прием к врачу.
2. Отмена записи пациента на прием к врачу.
3. Взаиморасчет с пациентом.
4. Выдача информации о ценах и услугах.

Главному администратору доступны следующие функции:

1. Запись пациента на прием к врачу.
2. Отмена записи пациента на прием к врачу.
3. Взаиморасчет с пациентом.
4. Выдача информации о ценах и услугах.



## 5. Формирование расписания врачей и медсестёр(медбратьев).

Врачу доступны следующие функции:

1. Ведение истории лечения пациента.
2. Получение расписания работы.

Медсестре (медбрату) доступны следующие функции:

1. Ведение зубной формулы пациента.
2. Учет материалов.
3. Получение расписания работы.

Исходя из возможных вариантов использования, диаграмма вариантов имеет следующий вид (рис.1.12).

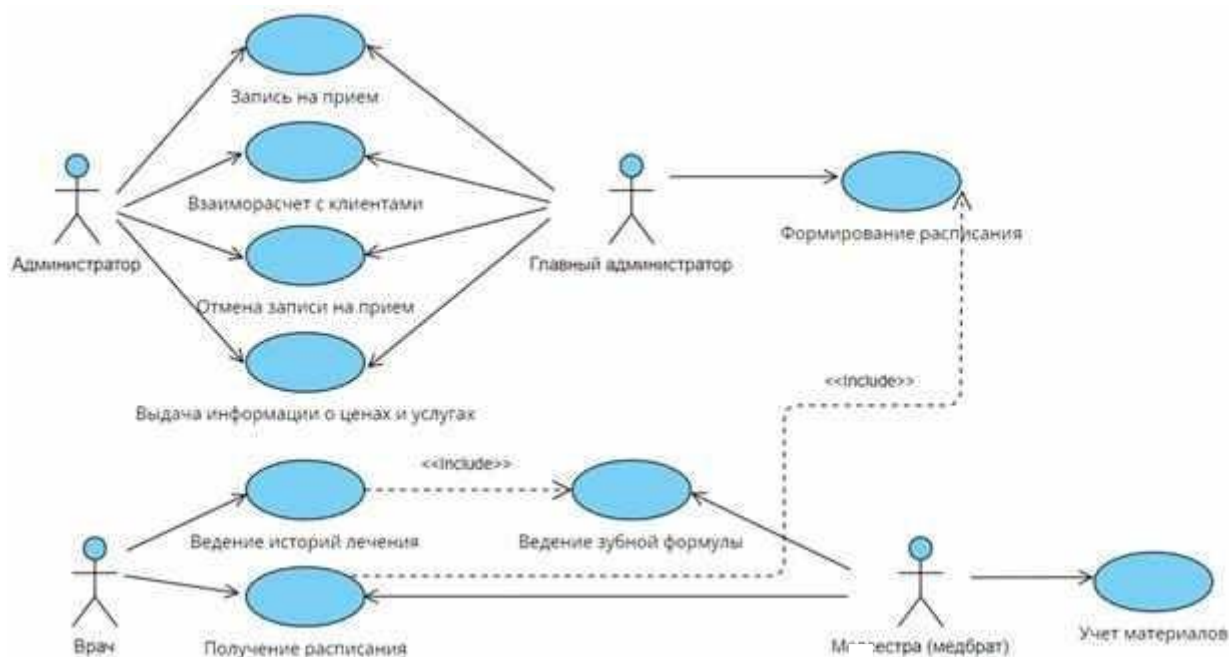


Рис.1.12 Диаграмма вариантов использования.

Для решения выделенных задач необходимо хранение данных о следующих объектах:

- Услуги.
- Клиенты.
- Сотрудники.
- Журнал историй болезней, включая зубные формулы.
- Расписание врачей.
- Расходные материалы.



- Записи на приемы.

Необходимые данные можно классифицировать по частоте их изменения:

- Условно-постоянные данные (список услуг, информация о сотрудниках).
- Данные, которые обновляются при каждом новом посещении (информация о клиентах, данные в истории лечений, история взаиморасчетов с клиентами).

### 1.3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Логическое проектирование базы данных это – преобразование требований к данным в структуры данных. Результат – СУБД-ориентированная структура базы данных и спецификации прикладных программ.

Цель логического проектирования - определение состава и структуры таблицы БД на основе результатов концептуального проектирования и проверка полученной модели с помощью методов нормализации.

В стадии логического проектирования входит:

- формирование отношений на основе логической модели данных;
- проверка отношений с использованием средств нормализации;
- определение ограничений целостности.

Для построения отношений используются диаграммы ER-типа сущность - связь. Основные ее определения:

- сущность – объект, информация о котором должна храниться в базе данных. Для ее изображения используется прямоугольный блок с именем сущности в виде существительного;
- связь – ассоциация между двумя сущностями, она предполагает наличие общих атрибутов. Для ее изображения используется ромб с именем связи в виде глагола.

Связи бывают:

- типа 1:1, 1:M, M:M;
- обязательные и необязательные.

На предыдущем этапе были выделены объекты, которые необходимо хранить в базе данных. Эти объекты становятся сущностями при ER-моделировании.

Построим ER-диаграммы всех сущностей и связей между ними.

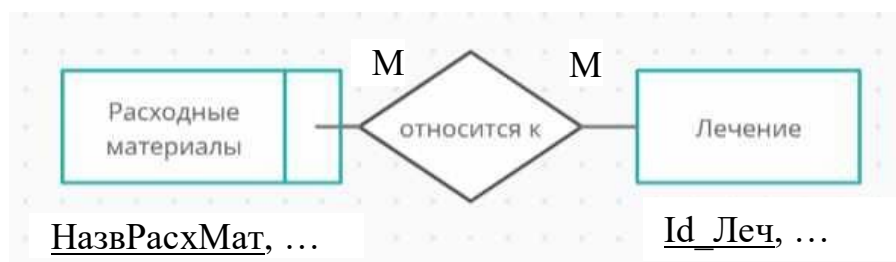


Рис.1.13 Связь «Относится к»

Связь «Относится к» (рис.1.13) имеет тип связи М:М, так как в процессе ЛЕЧЕНИЯ может быть использовано много расходных материалов или ни один, РАСХОДНЫЕ МАТЕРИАЛЫ могут быть использованы во многих процессах лечения или ни в одном. Сущность РАСХОДНЫЕ МАТЕРИАЛЫ имеет обязательный КП, так как расходные материалы обязательно должны относиться хотя бы к одному лечению. Сущность ЛЕЧЕНИЕ имеет необязательный КП, так как в процессе лечения может не участвовать расходный материал, например, осмотр зубов.



Рис.1.14 Связь «Получает»

Связь «Получает» (рис. 1.14) имеет тип 1:М, так как КЛИЕНТ может присутствовать в одном или нескольких процессах лечения, а в конкретном ЛЕЧЕНИИ присутствует только один пациент. Сущность КЛИЕНТ имеет необязательный класс принадлежности, так как, например, клиент мог отменить запись на лечение. Сущность ЛЕЧЕНИЕ имеет обязательный КП, так как в процессе лечения обязательно присутствует клиент.



Рис.1.15 Связь «Предоставляет»

Связь «Предоставляет» (рис. 1.15) имеет тип 1:M, так как СОТРУДНИК (врач) может принимать участие в одном и более процессах лечения, а в конкретном ЛЕЧЕНИИ обязательно присутствует только один сотрудник (врач). Сущность СОТРУДНИК (врач) имеет необязательный класс принадлежности, так как сотрудник может не присутствовать в лечении, например, если сотрудник является администратором. Сущность ЛЕЧЕНИЕ имеет обязательный КП, так как лечение обязательно должно предоставляться сотрудником (врачом).

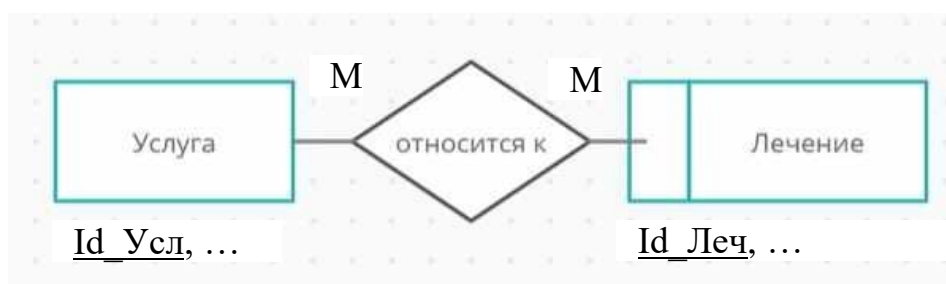


Рис.1.16 Связь «Относится к»

Связь «Относится к» (рис.1.16) имеет тип М:М, так как УСЛУГА относится ко многим процессам лечения, а ЛЕЧЕНИЕ состоит из оказания множества услуг, например, анестезия, удаление нерва, удаление пульпита, установка пломбы. Сущность УСЛУГА имеет необязательный класс принадлежности, так как, может добавиться ранее не используемая услуга. Сущность ЛЕЧЕНИЕ имеет обязательный КП, так как в лечении обязательно указывается услуга.

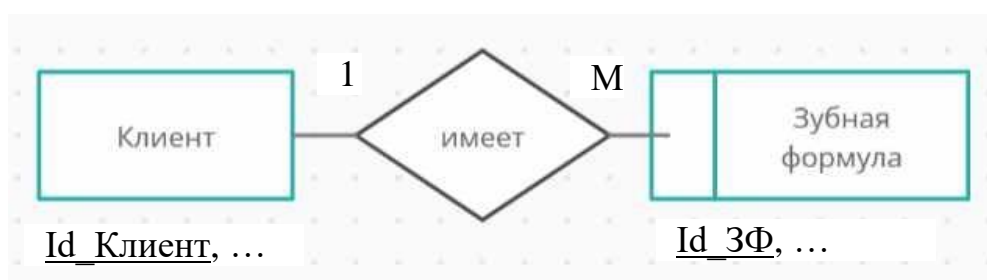


Рис.1.17 Связь «Имеет»

Связь «Имеет» (рис.1.17) имеет тип связи 1:1, так как КЛИЕНТ может иметь только несколько ЗУБНЫХ ФОРМУЛ для каждого больного зуба, а в каждой ЗУБНОЙ ФОРМУЛЕ указывается только один КЛИЕНТ. Сущность КЛИЕНТ имеет необязательный класс принадлежности, так как клиент может

не иметь зубную формулу. Сущность ЗУБНАЯ ФОРМУЛА имеет необязательный КП, так как зубная формула обязательно принадлежит клиенту.



Рис.1.18 Связь «Указывается»

Связь «Указывается» (рис.1.18) имеет тип связи 1:M, так как в СОТРУДНИК (врач, медсестра) получает множество расписаний, а в РАСПИСАНИИ указывается один сотрудник. Сущность РАСПИСАНИЕ имеет обязательные КП, так как в расписании обязательно указывается сотрудник. Сущность СОТРУДНИК имеет необязательный КП, так как сотрудник может не получать расписание, потому что находится в отпуске.



Рис.1.19 Связь «Состоит из»

Связь «Состоит из» (рис.1.19) имеет тип связи M:1, так как в РАСПИСАНИИ присутствует один день недели, а ДЕНЬ НЕДЕЛИ относится ко многим расписаниям. Сущность РАСПИСАНИЕ имеет обязательный КП, так как РАСПИСАНИЕ обязательно состоит из дней недели. Сущность ДЕНЬ НЕДЕЛИ имеет обязательны КП, так как день недели обязательно относится к расписанию.



Рис.1.20 Связь «Проводится»

Связь «Проводится» (рис.1.20) имеет тип связи 1:М, так как ЗАПИСЬ НА ПРИЕМ относится к одному клиенту, а КЛИЕНТ может проводить одну запись на прием. Сущность КЛИЕНТ имеет необязательный КП, так как клиент мог не проводить запись на прием. Сущность ЗАПИСЬ НА ПРИЕМ имеет обязательный КП, так как в записи на прием обязательно присутствует клиент.

А) Сформируем набор предварительных отношений с указанием предполагаемого первичного ключа для каждого отношения.

Связь ОТНОСИТСЯ К (рис.1.13) удовлетворяет условиям правила 6, в соответствии с которым получается три отношения:

1. Расходные материалы (НазвРасхМат, ...)
2. Лечение (Id\_Леч, ...)
3. Расход (НазвРасхМат, Id\_Леч)

Связь ПОЛУЧАЕТ (рис.1.14) удовлетворяет условиям правила 4, в соответствии с которым получается два отношения:

1. Клиент (Id\_Клиент, ...)
2. Лечение (Id\_Леч, Id\_Клиент, ...)

Связь ПРЕДОСТАВЛЯЕТ (рис.1.15) удовлетворяет условию 4, в соответствии с которым получается два отношения:

1. Сотрудник (Id\_Сотр, ...)
2. Лечение (Id\_Леч, Id\_Сотр, ...)

Связь ОТНОСИТСЯ К (рис.1.16) удовлетворяет условиям правила 6, в соответствии с которым получается два отношения:

1. Услуга (НазвУсл, ...)
2. Лечение (Id\_Леч, ...)
3. Услуги лечения (НазвУсл, Id\_Леч)

Связь ИМЕЕТ (рис.1.17) удовлетворяет условиям правила 4, в соответствии с которым получает два отношения:

1. Клиент (Id\_Клиент, ...)
2. Зубная формула (Id\_ЗФ, Id\_Клиент, ...)

Связь **УКАЗЫВАЕТСЯ** (рис.1.18) удовлетворяет условиям правила 4, в соответствии с которым получается два отношения:

1. Расписание (Id\_Расп, Id\_Сотр, ...)
2. Сотрудник (Id\_Сотр, ...)

Связь **СОСТОИТ ИЗ** (рис.1.19) удовлетворяет условиям правила 4, в соответствии с которым получается две сущности:

1. Расписание (Id\_Расп, НазваниеДня, ...)
2. День недели (НазваниеДня, ...)

Связь **ПРОВОДИТСЯ** (рис.1.20) удовлетворяет условиям правила 2, в соответствии с которым получается две сущности:

1. Запись на прием (Id\_Запись, Id\_Клиент, ...)
2. Клиент (Id\_Клиент, ...)

Б) Добавим неключевые атрибуты в каждое из предварительных отношений с условием, чтобы отношения отвечали требованиям третьей нормальной формы.

Сотрудник (Id\_Сотр, ФИО, НазвДолж, Образование, №Телефона, Зарплата, №Паспорта)

Клиент (Id\_Клиент, ФИО, ДатаРождения, Пол, №Телефона, №Паспорта)

Лечение (Id\_Леч, Id\_Клиент, Id\_Вр, ДатаЛеч, Жалобы, Рекомендации)

Услуга (НазвУсл, Цена)

Расходные материалы (НазвРасхМат, Количество)

Зубная формула (Id\_ЗФ, Id\_Клиент, №Зуба, Деформация)

Расписание (Id\_Расписание, Id\_Сотр, Название Дня, Время приема)

Запись на прием (Id\_Запись, Id\_Клиент, Дата, Время, Служ\_Пометка)

Расход (НазвРасхМат, Id\_Леч, Количество, ЕдИзм)

Услуги лечения (НазвУсл, Id\_Леч)



## 1.4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Во время предыдущего этапа проектирования была определена логическая структура базы данных (которая описывает отношения и ограничения в рассматриваемой прикладной области). Хотя эта структура не зависит от конкретной целевой СУБД, она создается с учетом выбранной модели хранения данных, например реляционной, сетевой или иерархической. Однако, приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

В качестве СУБД для проектирования базы данных выбран SQL Server Management Studio – это бесплатная графическая среда, включающая набор инструментов для разработки сценариев на T-SQL и управления инфраструктурой Microsoft SQL Server.

Среда SQL Server Management Studio – это основной, стандартный и полнофункциональный инструмент для работы с Microsoft SQL Server,

разработанный компанией Microsoft, который предназначен как для разработчиков, так и для администраторов SQL Server.

Так как в задачи входит полное сопровождение Microsoft SQL Server, начиная от создания баз данных, написания SQL запросов, создания хранимых процедур и функций, и заканчивая администрированием SQL Server, включая управление безопасностью, то в качестве основного инструмента отлично подходит среда SQL Server Management Studio.

На основе сведений, полученных в ходе выполнения предыдущего этапа, где были построены ER-диаграммы, а также сформированы отношения и добавлены не ключевые атрибуты, необходимо составить структуры таблиц, с помощью которых можно будет физически спроектировать базу данных. Для создания БД необходимо составить структуры вышеперечисленных таблиц. Необходимо установить типы данных, возможность оставить поля незаполненным, назначить первичные и внешние ключи, установить ограничения, значения по умолчанию, ссылки на другие таблицы.

В таблице 1.1 представлена структура таблицы «Клиент», в приложении А (рис. А.1) содержатся данные таблицы «Клиент».

Таблица 1.1

Требования к структуре таблицы «Клиент»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
Id_Клиент	int	нет	первичный			
ФИО	varchar(90)	нет				
пол	char(1)	нет		м	м о г ж	
ДатаРождения	date	нет			>=18	
№Телефона	int	нет			уникальный	
№Паспорта	int	нет			уникальный	

В таблице 1.2 представлена структура таблицы «Сотрудник», в приложении А (рис. А.2) содержатся данные таблицы «Сотрудник».

Таблица 1.2

Требования к структуре таблицы «Сотрудник»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
Id_Сотр	int	нет	первичный			
ФИО	varchar(90)	нет				
Образование	varchar(50)	нет				
НазвДолж	varchar(50)	нет				
№Телефона	int	нет			уникальный	
Зарплата	money	нет			> 0	
№Паспорта	int	нет			уникальный	

В таблице 1.3 представлена структура таблицы «Расходные материалы», в приложении А (рис. А.3) содержатся данные таблицы «Расходные материалы».

Таблица 1.3

Требования к структуре таблицы «Расходные материалы»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
НазвРасхМат	varchar(50)	нет	первичный			
Количество	int	нет		0	>=0	
ЕдИзм	varchar(2)	нет		шт	шт or г	

В таблице 1.4 представлена структура таблицы «Услуга», в приложении А (рис. А.4) содержатся данные таблицы «Услуга».

Таблица 1.4

Требования к структуре таблицы «Услуга»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
НазвУсл	varchar(50)	нет	первичный			
Цена	money	нет				

В таблице 1.5 представлена структура таблицы «Лечение», в приложении А (рис. А.5) содержатся данные таблицы «Лечение».

Таблица 1.5

Требования к структуре таблицы «Лечение»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
Id_Леч	int	нет	первичный			
Id_Клиент	int	нет	внешний			Клиент (Id_Клиент)
Id_Вр	int	нет	внешний			Сотрудник (Id_Сотр)
ДатаЛеч	date	нет		Тек. дата		
Жалобы	varchar(50)	да				
Рекомендации	varchar(100)	да				

В таблице 1.6 представлена структура таблицы «Зубная формула», в приложении А (рис. А.6) содержатся данные таблицы «Зубная формула».

Таблица 1.6

Требования к структуре таблицы «Зубная формула»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
Id_ЗФ	int	нет	первичный			
Id_Клиент	int	нет	внешний			Клиент (Id_Клиент)
№Зуба	int	нет			от 1 до 32	
Деформация	varchar(50)	нет				

В таблице 1.7 представлена структура таблицы «Расписание», в приложении А (рис. А.7) содержатся данные таблицы «Расписание».

Таблица 1.7

## Требования к структуре таблицы «Расписание»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
Id_Расписание	int	нет	первичный			
Id_Сотр	int	нет	внешний			Сотрудник (Id_Сотр)
Название дня	date	нет				
Время приема	time	нет				

В таблице 1.8 представлена структура таблицы «Запись на прием», в приложении А (рис. А.8) содержатся данные таблицы «Запись на прием».

Таблица 1.8

## Требования к структуре таблицы «Запись на прием»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
Id_Запись	int	нет	первичный			
Id_Клиент	int	нет	внешний			Клиент (Id)
ДатаВремя	datetime	нет		Тек. Дата		
Служ_Пометка	varchar(100)	да				
Врач	varchar(100)	нет	внешний			Сотрудник и (ФИО)

В таблице 1.9 представлена структура таблицы «Услуги лечения», в приложении А (рис. А.9) содержатся данные таблицы «Услуги лечения».

Таблица 1.9

## Требования к структуре таблицы «Услуги лечения»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
НазвУсл	varchar(50)	нет	первичный			Услуга (НазвЛеч)
Id_Леч	int	нет	первичный			Лечение (Id_Леч)

В таблице 1.10 представлена структура таблицы «Расход», в приложении А (рис. А.10) содержатся данные таблицы «Расход».

Требования к структуре таблицы «Расход»

Столбец	Тип данных	Ноль	Ключ	По умолчанию	Ограничение	Ссылка
Id_Леч	int	нет	первичный			Лечение (Id_Леч)
НазвРасхМат	int	нет	первичный			Расходные материалы (НазвРасхМат)
Количество	int	нет		0	$\geq 0$	
ЕдИзм	varchar(2)	нет		шт	шт or г	

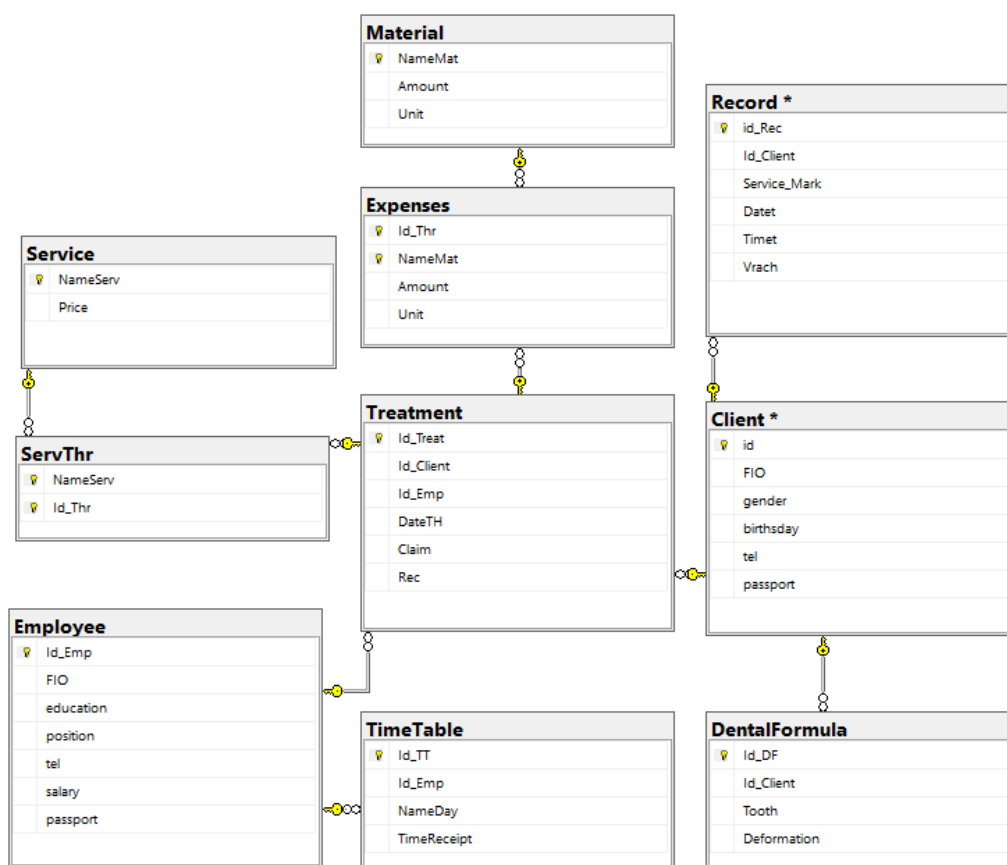


Рис. 1.21 Схема данных

После создания структуры таблиц необходимо установить связи между таблицами. Для построения схемы данных (рис. 1.21) была использована SQL Server Manegment Studio [8].

У таблиц также есть собственные функции, триггеры и процедуры.

У таблицы Клиент есть один триггер `dbo.Payment_INSERT`, а также хранимая процедура `dbo.crcl` (Приложение Б, п.1, п.6). Процедура `dbo.crcl`

необходима для добавления в БД нового клиента. Входными данными являются ФИО клиента, пол, день рождения, номер телефона и серия паспорта. Внутри процедуры данные проверяются на корректность, в случае существования такой записи процедура возвращает код ошибки 1, иначе добавляет запись в таблицу. Триггер `dbo.Payment_INSERT` предназначен для генерации `id` клиента, применяется при добавлении в таблицу нового клиента.

У таблицы *Зубная формула* есть одна хранимая процедура `dbo.addf` (Приложение Б, п.5). Входными параметрами функции являются ФИО клиента, номер зуба и описание деформации. Внутри процедуры данные проверяются на корректность, в случае существования такой записи процедура возвращает код ошибки 1, иначе добавляет запись в таблицу.

У таблицы *Сотрудники* есть две хранимые процедуры `dbo.pass` и `dbo.reg` (Приложение Б, п.12, п.13). Входными параметрами процедуры `dbo.reg` являются ФИО сотрудника и его номер телефона. Внутри процедуры данные проверяются на корректность, в случае отсутствия такой записи в таблице *Сотрудники*, процедура выводит код ошибки 2, иначе возвращает значение «Стоматолог», «Медбрат» или «Медсестра», «Администратор», «Главный администратор» в зависимости с соответствием входных параметров. Хранимая процедура `dbo.pass` предназначена для напоминания пользователю пароля. Входным параметром является ФИО сотрудника. На выходе клиент получает свой пароль.

У таблиц *Расход* и *Материалы* есть процедура `dbo.ExMat` (Приложение Б, п.9). Входными параметрами являются `id` лечения, Количество и единица измерения. В результате выполнения процедуры в таблицу *Расход* добавляется новая запись, а в таблице *Материалы* изменяется Столбец количество у указанного материала.

Таблицы *Запись на прием* и *Расписание* имеют процедуры `dbo.addrec` и `dbo.delrec` (Приложение Б, п.2, п.7), также таблица *Расписание* имеет процедуры `dbo.DeltimeLable` и `dbo.NimeTable` (Приложение Б, п.8, п.10). Входными параметрами хранимой процедуры `dbo.addrec` являются ФИО



сотрудника, служебная запись, дата, время и ФИО клиента. Внутри процедуры данные проверяются на корректность, в случае существования такой записи в таблице Запись, процедура выводит код ошибки 1, 2 или 3 в зависимости от существования записи в таблице Запись, иначе в таблицы Запись и Расписание добавляются новые записи. Входными параметрами хранимой процедуры `dbo.delrec` являются ФИО клиента, дата и время. В результате выполнения процедуры из таблиц Запись и Расписание удаляются соответствующие строки. Входными параметрами процедуры `dbo.NimeTable` являются ФИО сотрудника, дата и время. В результате выполнения процедуры в таблицу Расписание добавляется новая запись. Входными параметрами процедуры `dbo.Deltimelable` являются ФИО сотрудника, дата и время. В результате выполнения процедуры из таблицы Расписание удаляется запись.

У таблицы Услуги лечения есть одна хранимая процедура `dbo.addservthr` (Приложение Б, п.3), Также таблицы Услуги и Услуги лечения имеют табличную функцию `dbo.Nill` (Приложение Б, п.14). Входными параметрами процедуры являются название услуги и `id`. В результате выполнения процедуры в таблицу Услуги лечения добавляется новая запись. Входным параметром функции является `id` лечения. В результате выполнения функции во временную таблицу `res` добавляется новая запись.

У таблицы Лечение есть две хранимые процедуры `dbo.addthr` и `dbo.param` (Приложение Б, п.4, п.11). Входными параметрами процедуры `dbo.addthr` являются ФИО клиента, ФИО сотрудника, дата, рекомендации и назначение. Внутри процедуры данные проверяются на корректность, в случае существования такой записи в таблице Лечение, процедура выводит код ошибки 1, в случае указания неверной даты процедура выводит код ошибки 2. В результате успешного выполнения процедуры в таблицу Лечение добавляется новая запись. Входным параметром процедуры `dbo.param` является `id` лечения. В результате выполнения процедуры всем выходным параметрам присваивается нужное значение.

Также в БД есть скалярная функция `dbo.priise` (Приложение Б, п.15).

Функция подсчитывает суммарную цену на услуги, записанные во временную таблицу res.

## **2. ОПИСАНИЕ ФУНКЦИОНИРОВАНИЯ БАЗЫ ДАННЫХ**

### **2.1. Назначение и перечень функций базы данных**

Интерфейс, созданный специально для определенного пользователя – важнейшая задача данного проекта. Так как в работе представлены четыре пользователя: администратор, главный администратор, врач и медсестра (медбрат), каждому из них необходим свой интерфейс для взаимодействия с базой данных.

Администратор имеет такие возможности, как:

- Изменение записи на прием.
- Добавление нового клиента.
- Выдача информации об услугах.
- Взаиморасчет с клиентом.

Главный администратор, в свою очередь, имеет функционал намного обширнее. Помимо функций администратора в него входят:

- Формирование расписания.
- Просмотр списка сотрудников.

Врач имеет следующие возможности:

- Изменение истории лечения.
- Изменение зубной формулы.
- Просмотр расписания.
- Изменение услуг лечения.

В заключении, медбрат (медсестра) имеет такие возможности как:

- Изменение зубной формулы.
- Просмотр расписания.
- Учет материалов.

Для разработки пользовательского интерфейса использовалась среда разработки Visual Studio [5] и язык программирования C# [6].

## 2.2. Описание работы с базой данных

При входе в приложение открывается главная страница (рис. 2.1).

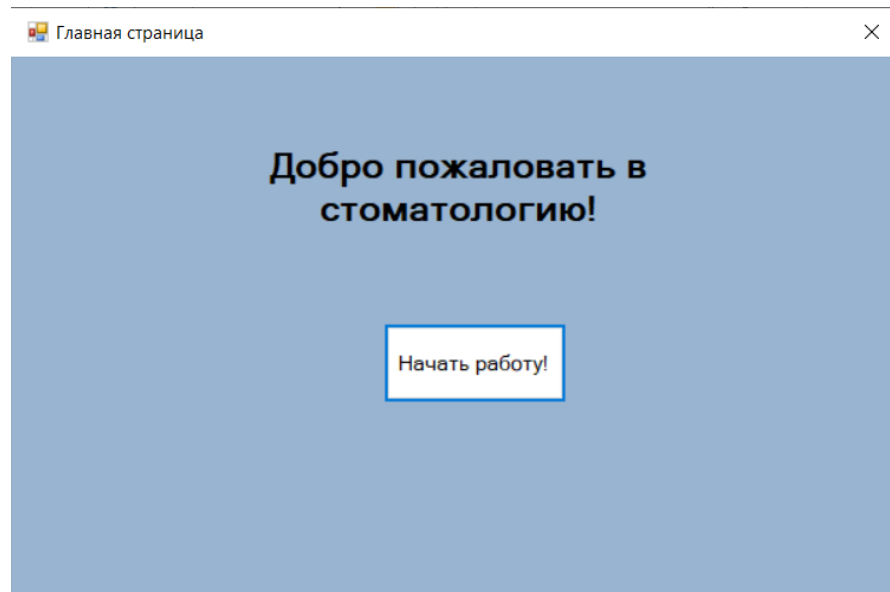


Рис. 2.1 Главная страница приложения

Пользователю доступна единственная функция начала работы. При нажатии на кнопку, открывается окно авторизации (рис. 2.2).

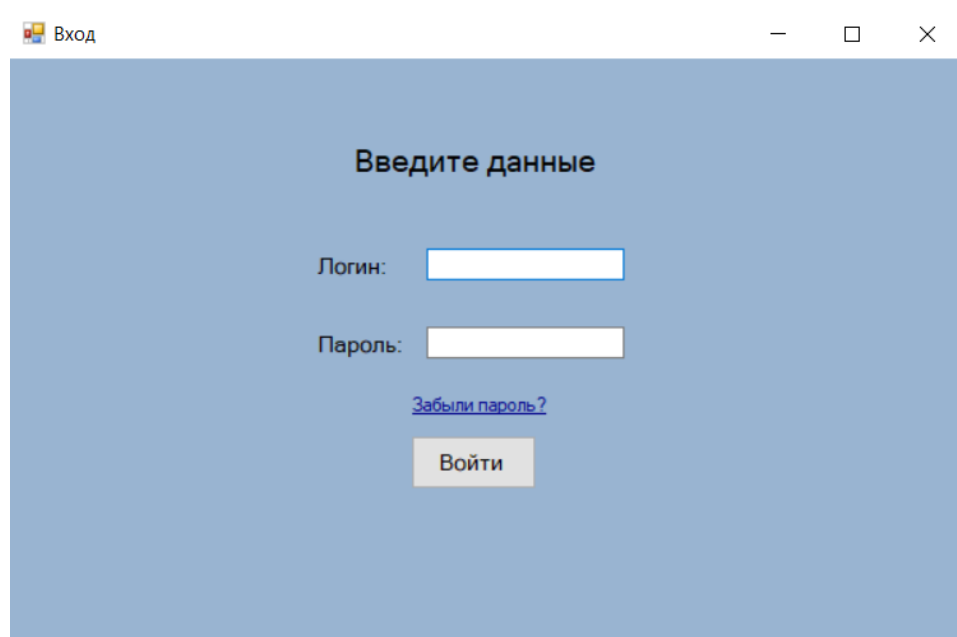


Рис. 2.2 Вход в приложение

На этом этапе пользователю доступно два действия:

1. Использовать свой никнейм и пароль для входа;
2. Перейти на форму напоминания пароля.

Разберем форму напоминания пароля.

При нажатии на надпись «Забыли пароль?» форма входа меняет свои поля (рис.2.3).

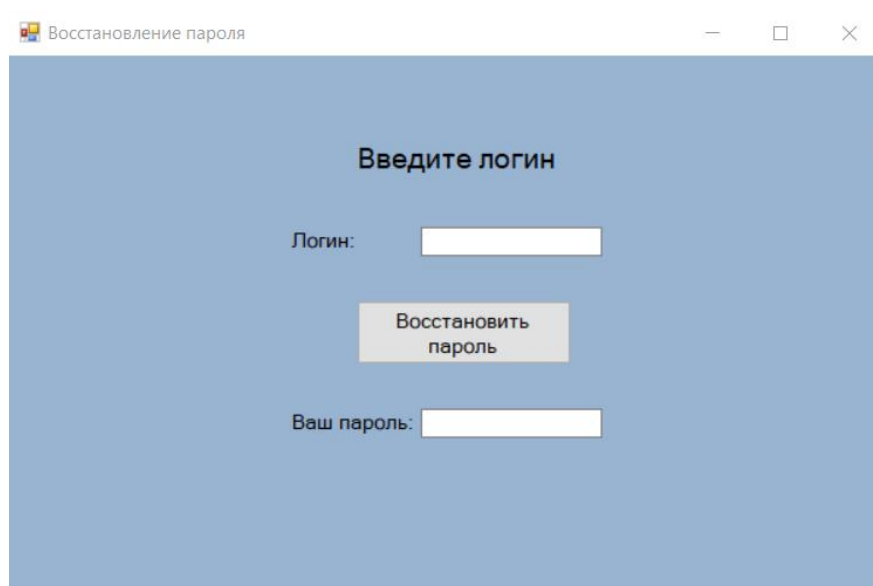


Рис. 2.3 Восстановление пароля

Пользователь вводит необходимые данные и нажимает на кнопку «Восстановить пароль», после этого срабатывает хранимая процедура `dbo.pass` (приложение Б). При этом в окне ниже кнопки высвечивается забытый пароль.

В случае успешного восстановления пароля, пользователь переходит на основную форму входа.

Теперь переходим непосредственно к форме входа.

При попытке входа с некорректными данными в полях высветится окно с текстом ошибки и счетчиком, отвечающим за количество доступных попыток входа в приложение (рис. 2.4).

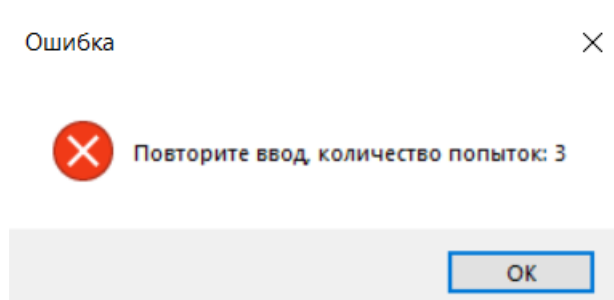


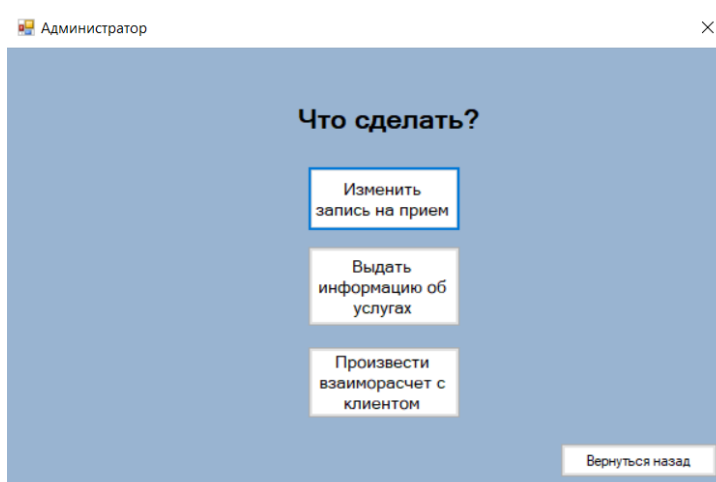
Рис. 2.4 Окно ошибки

Когда данный счетчик достигнет нуля, кнопка «Вход» станет недоступной.

При успешной аутентификации срабатывает хранимая процедура dbo.reg (приложение Б) и можно попасть на определенную форму, а именно:

1. Администратор;
2. Главный администратор;
3. Врач;
4. Медсестра (медбрат).

Разберем форму администратора (рис. 2.5).



Рисю 2.5 Форма админимтратора

На этом этапе пользователю доступно четыре действия:

1. Изменение записи на прием;
2. Выдача информации об услугах;
3. Взаиморасчет с клиентом;
4. Возвращение на форму входа в приложение.

Разберем форму изменения записи на прием.

При нажатии кнопки «Изменить запись на прием», администратор переходит на форму «Обновление запись на прием» (рис. 2.6).

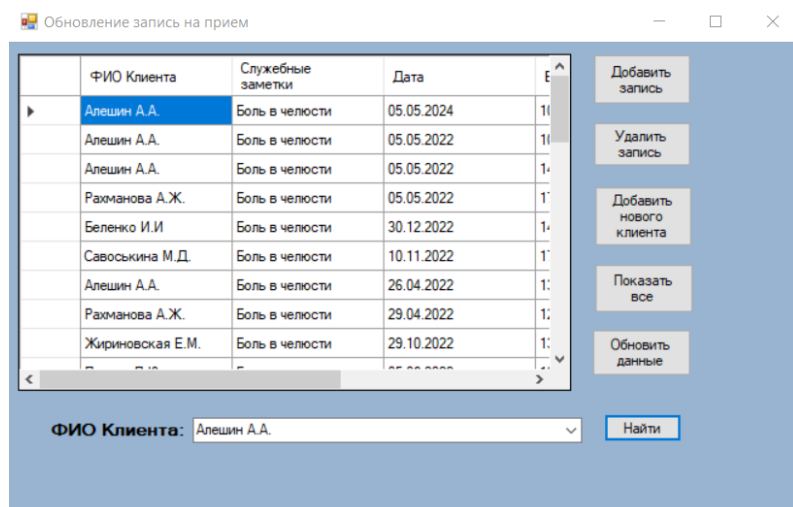


Рис. 2.6 Обновление запись на прием

На этой форме администратор может:

1. Просмотреть таблицу существующих записей на прием;
2. Произвести поиск по фамилиям клиентов;
3. Добавить новую запись;
4. Удалить существующую запись;
5. Добавить нового клиента;
6. Обновить данные в таблице «Запись на прием».

Начнем с поиска записи на прием. Для поиска определено отдельное поле в нижней части формы и кнопка «Найти». Поиск осуществляется по всем возможным совпадениям ФИО клиента (рис. 2.7). Вернуть изначальный вид таблицы можно с помощью кнопки «Показать все» в правой части формы (рис. 2.8).

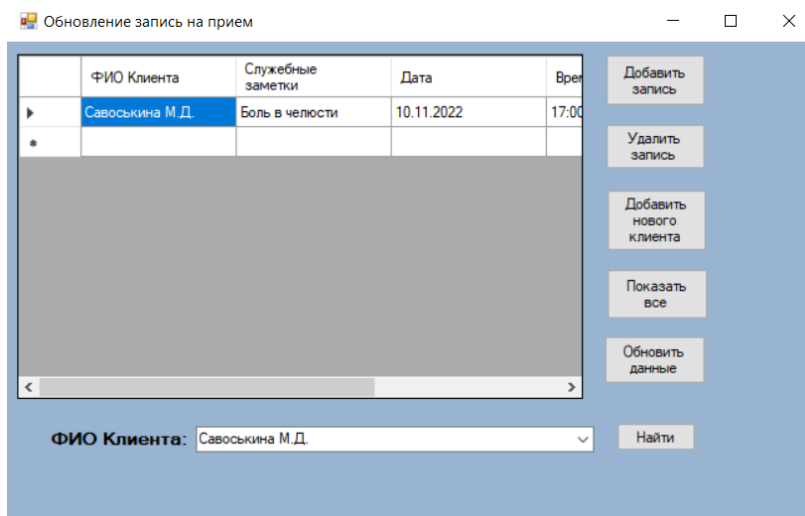


Рис. 2.7 Поиск по ФИО клиента



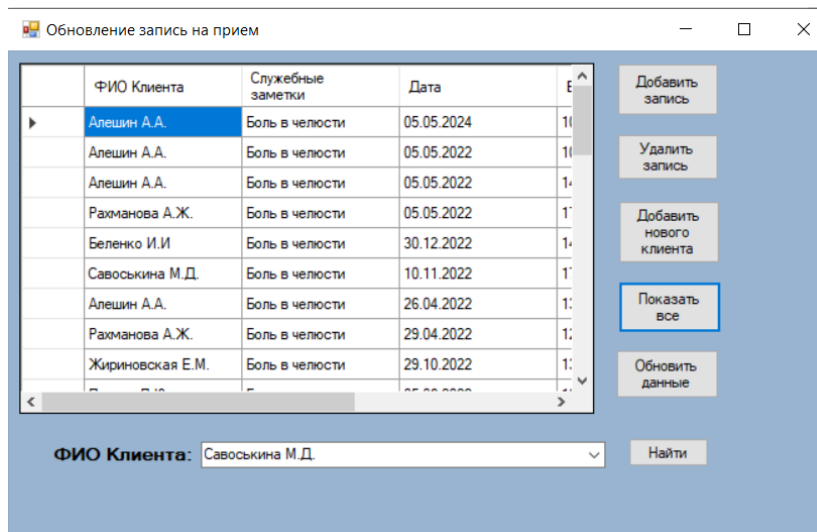


Рис.2.8 Работа кнопки «Показать все»

Перейдем к кнопкам «Добавить запись», «Удалить запись» и «Добавить нового клиента», каждая переводит администратора на соответствующие формы (рис. 2.9 – 2.11). Заполнив поля на открывшейся форме и нажав на кнопку, пользователь выполнит нужное действие. При нажатии на кнопку, срабатывают хранимые процедуры `dbo.addrec`, `dbo.delrec` и `dbo.crcl`, а также триггер на таблице `Client - Payment_INSERT` (приложение Б) соответственно. При неверном вводе данных, на каждой из форм, система выводит сообщение об ошибке (рис. 2.12).

Рис. 2.9 Форма «Добавить запись»

Удалить запись

**Введите данные**

ФИО Клиента: Алешин А.А.

Дата приема: 11 декабря 2022 г.

Время приема:

Удалить запись

Рис. 2.10 Форма «Удалить запись»

Добавление клиента

**Введите данные клиента**

ФИО: Алешин А.А.

Пол: м

Дата рождения: 1 января 2000 г.

Номер телефона:

Паспорт: 1111

Добавить клиента

Рис. 2.11 Форма «Добавление нового клиента»

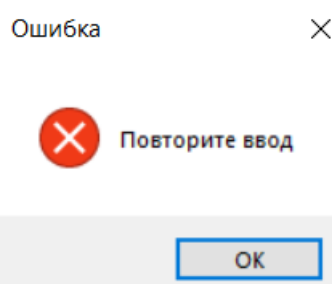


Рис. 2.12 Сообщение об ошибке

Вернемся к основной форме администратора. Следующей на очереди располагается кнопка, переводящая на форму выдачи информации об услугах (рис.2.13).

Информация об услугах

### Выберите услугу

Название услуги:

	Название услуги	Цена
▶	Осмотр зубов	1000.0000
	Отбеливание	5000.0000
	Пломба	3200.0000
	Полировка	4500.0000
	Слепок зубов	1500.0000
*		

Рис.2.13 Форма «Информация об услугах»

На форме пользователю доступна функция поиска по имеющимся в базе данных услугам. Для большего удобства поле ввода названия услуги представлено в виде элемента управления ComboBox (рис.2.14).

Информация об услугах

### Выберите услугу

Название услуги:

Осмотр зубов

Отбеливание

Пломба

Полировка

Слепок зубов

	Название услуги	Цена
▶	Осмотр зубов	1000.0000
	Отбеливание	5000.0000
	Пломба	3200.0000
	Полировка	4500.0000
	Слепок зубов	1500.0000
*		

Рис. 2.14 Элемент ввода данных ComboBox

Следующей на форме администратора располагается кнопка «Произвести взаиморасчет с клиентом». При нажатии, система переводит пользователя на форму «Взаиморасчет» (рис. 2.15).

The screenshot shows a window titled 'Взаиморасчет'. It contains a table with the following data:

	ФИО Клиента	Id Лечения	Дата лечения
▶	Алешин А.А.	1111	05.05.2022
	Сафонов К.М.	1112	05.05.2022
	Клоков Т.Е	1113	06.05.2022
	Олейник А.А.	1114	06.05.2022
	Мияйлова М.А.	1115	06.05.2022
	Ногаева Д.М.	1116	07.05.2022
	Павлов А.Д	1117	07.05.2022
	Мищенко Я.Д.	1118	07.05.2022
	Казаков С.Д.	1119	08.05.2022
	Савченко А.Д.	1120	08.05.2022

Below the table are search filters:

- ☐ **ФИО** Алешин А.А. [dropdown] [Найти]
- ☐ **Дата** 11 декабря 2022 г. [calendar icon]
- [Произвести взаиморасчет] [Показать все]

Рис. 2.15 Форма «Взаиморасчет»

В нижней части формы располагаются поля ввода данных для комбинированного поиска данных (рис. 2.16).

The screenshot shows the same window as Figure 2.15, but with search results for 'Алешин А.А.' displayed in the table:

	ФИО Клиента	Id Лечения	Дата лечения
▶	Алешин А.А.	1111	05.05.2022
	Алешин А.А.	1156	13.05.2022
	Алешин А.А.	1157	05.05.2029
	Алешин А.А.	1158	05.12.2022
	Алешин А.А.	1159	26.11.2022
	Алешин А.А.	1160	25.12.2022
	Алешин А.А.	1161	10.02.2022
	Алешин А.А.	1162	27.02.2022
	Алешин А.А.	1163	08.11.2022

The search filters remain the same as in Figure 2.15.

Рис. 2.16 Поиск по полю ФИО

Нажатие кнопки «Показать все» возвращает, представленную на форме,

таблицу в исходное состояние. Кнопка «Произвести взаиморасчет» переносит пользователя на форму «Взаиморасчет» (рис. 2.17), где администратору предоставлена функция проведения взаиморасчета с клиентом по выбранному Id лечения.

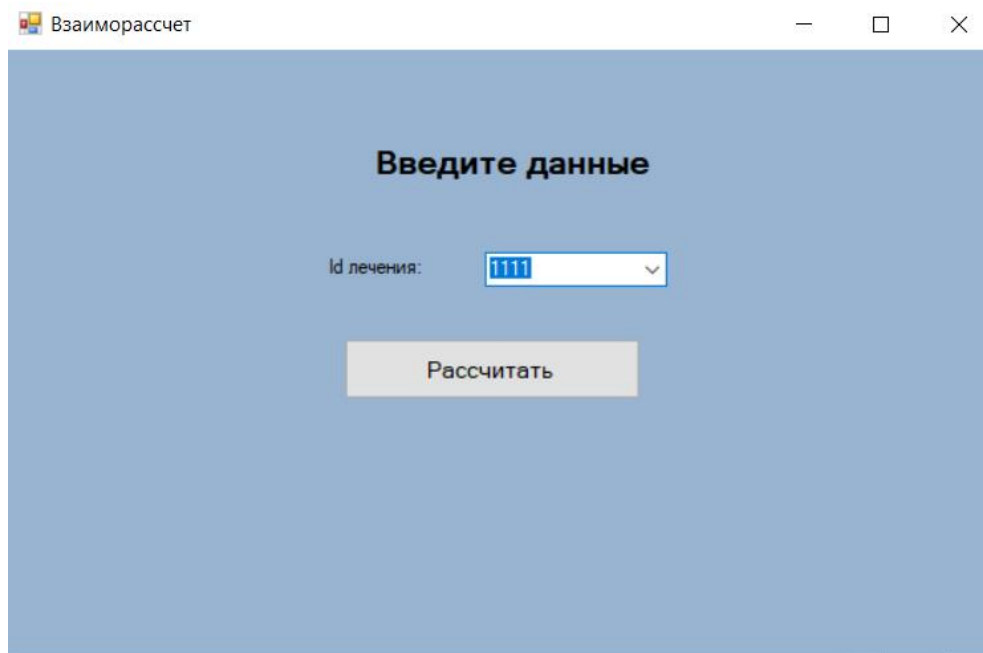
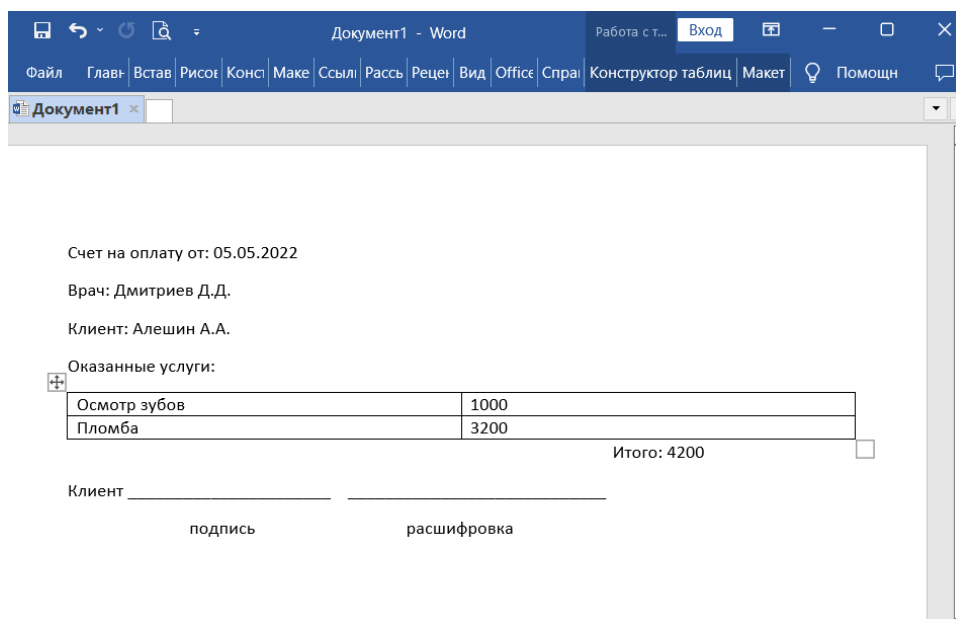


Рис. 2.17 Форма «Взаиморасчет»

Выбрав нужный id лечения и нажав на кнопку «Рассчитать», пользователь формирует расчетный лист в приложении Word (рис. 2.18) [7].



Осмотр зубов	1000
Пломба	3200

Итого: 4200

Клиент \_\_\_\_\_

подпись

расшифровка

Рис. 2.18 Расчетный лист

На этом функции администратора заканчиваются, поэтому переходим к функциям главного администратора (рис. 2.19).

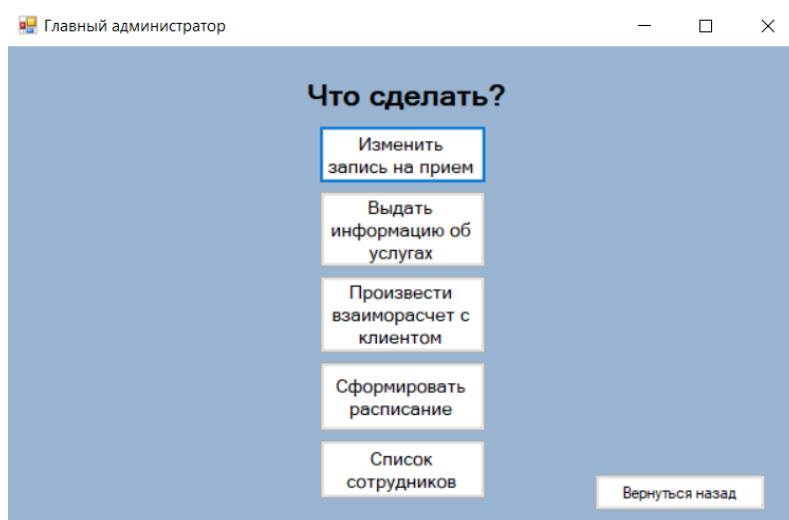


Рис. 2.19 Форма «Главный администратор»

На этом этапе главному администратору доступно шесть действий:

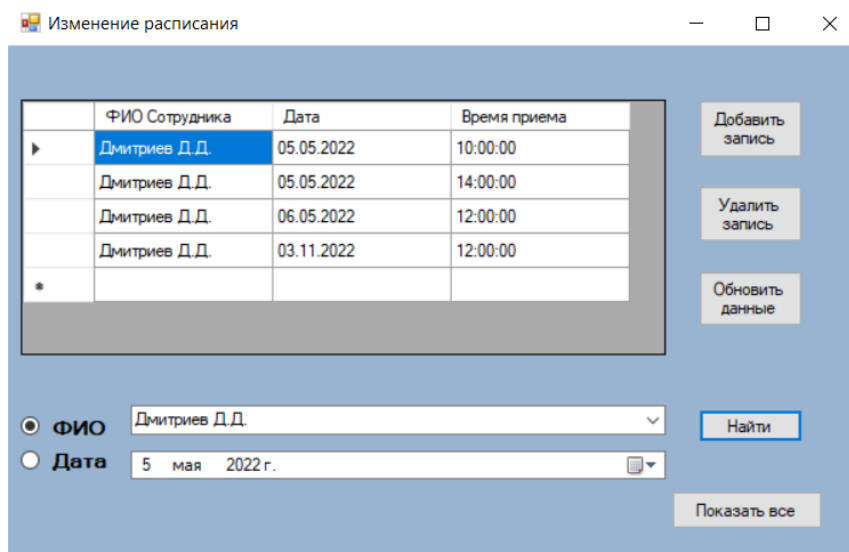
1. Изменение записи на прием;
2. Выдача информации об услугах;
3. Взаиморасчет с клиентом;
4. Возвращение на форму входа в приложение;
5. Формирование расписания сотрудников;
6. Просмотр списка сотрудников.

Первые четыре действия были описаны в функциях администратора и совпадают с ними. Поэтому перейдем к формированию расписания сотрудников. При нажатии на кнопку «Сформировать расписание», открывается форма «Изменение расписания» (рис. 2.20).

ФИО Сотрудника	Дата	Время приема
Дмитриев Д.Д.	05.05.2022	10:00:00
Дмитриев Д.Д.	05.05.2022	14:00:00
Мишин О.В.	06.05.2022	10:00:00
Дмитриев Д.Д.	06.05.2022	12:00:00
Дмитриев Д.Д.	03.11.2022	12:00:00

Рис. 2.20 Форма «Изменение расписания»

В нижней части формы располагаются поля ввода данных для комбинированного поиска данных (рис. 2.21).



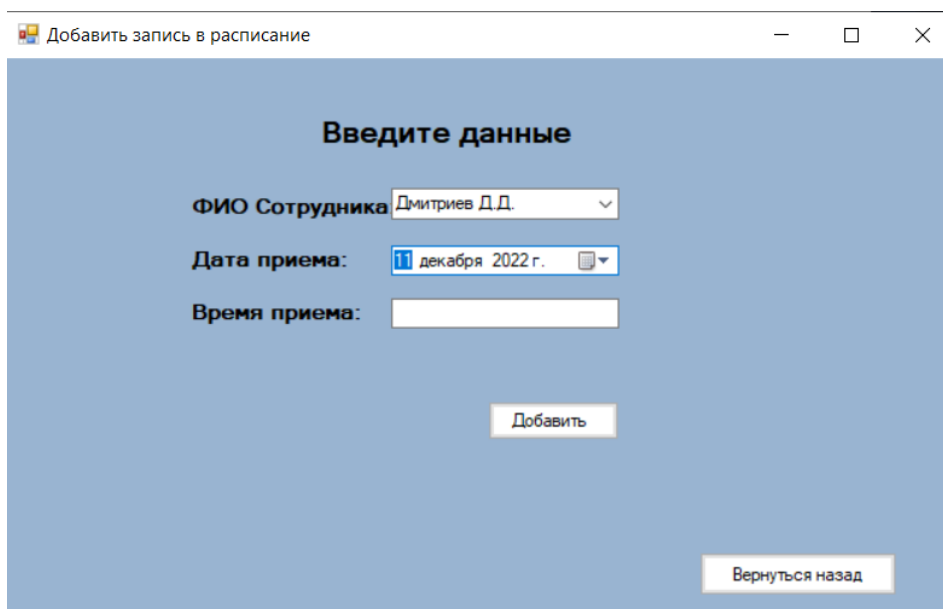
	ФИО Сотрудника	Дата	Время приема
▶	Дмитриев Д.Д.	05.05.2022	10:00:00
	Дмитриев Д.Д.	05.05.2022	14:00:00
	Дмитриев Д.Д.	06.05.2022	12:00:00
	Дмитриев Д.Д.	03.11.2022	12:00:00
*			

☒ **ФИО**

☐ **Дата**

Рис. 2.21 Поиск по ФИО

Нажатие кнопки «Показать все» возвращает таблицу в исходное состояние. Кнопка «Обновить данные» обновляет таблицу. Перейдем к добавлению и удалению записей. Соответствующие кнопки переносят пользователя на нужные формы (рис. 2.22 – 2.23). При неверном вводе данных на каждой из форм, система выводит сообщение об ошибке (рис.2.24).



**Введите данные**

ФИО Сотрудника  ▼

Дата приема:  ▼

Время приема:

Рис. 2.22 Форма «Добавить запись в расписание»

Рис. 2.23 Форма «Удалить запись»

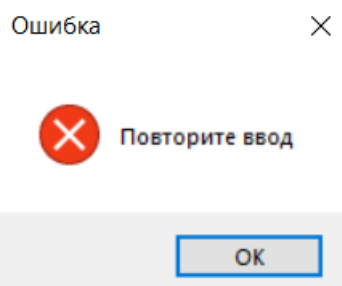


Рис. 2.24 Сообщение об ошибке

Нажатие кнопок «Добавить» и «Удалить» вызывают хранимые процедуры `dbo.NimeTable` и `dbo.DeltimеTable` (приложение Б) соответственно.

Вернемся к функциям главного администратора. Последней на очереди является функция просмотра списка сотрудников. При нажатии на кнопку «Список сотрудников», открывается форма «Список сотрудников» (рис. 2.25).

ФИО Сотрудника	Образование	Должность	Телефон	Зарплата
Дмитриев Д.Д.	Высшее	Стоматолог	98647238	5000
Аджинян Ф.И.	Среднее специальное	Медбрат	98643788	3500
Мишин О.В.	Высшее	Стоматолог	98787364	5000
Венина А.И.	Высшее	Администратор	98765673	3500
Петров П.П.	Высшее	Главный администрат...	98656473	4000

Рис. 2.25 Форма «Список сотрудников»



В нижней части формы располагается текстовый элемент управления – ComboBox, облегчающий поиск по сотрудникам (рис.2.26).

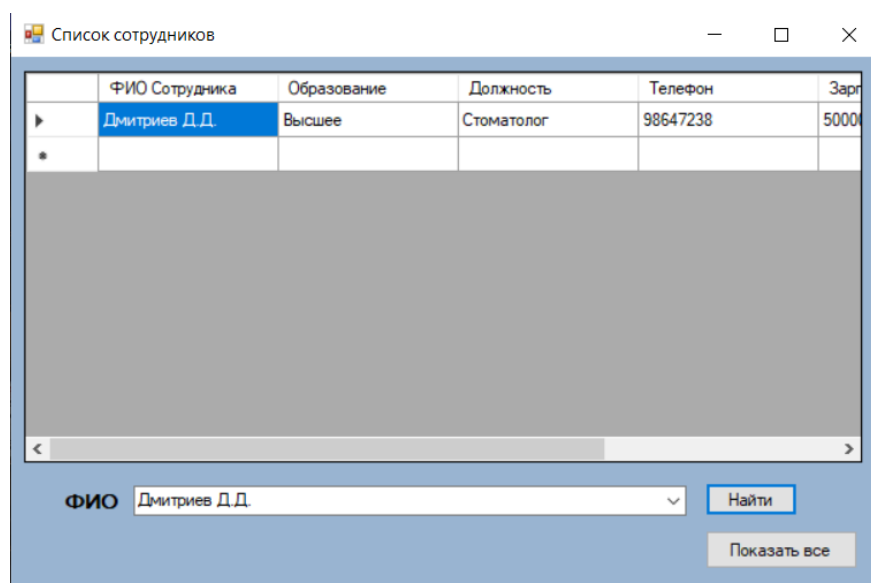


Рис. 2.26 Поиск по ФИО

Нажатие кнопки «Показать все» возвращает таблицу в исходное состояние.

На этом функции главного администратора заканчиваются. Перейдем к функциям врача (рис. 2.27).

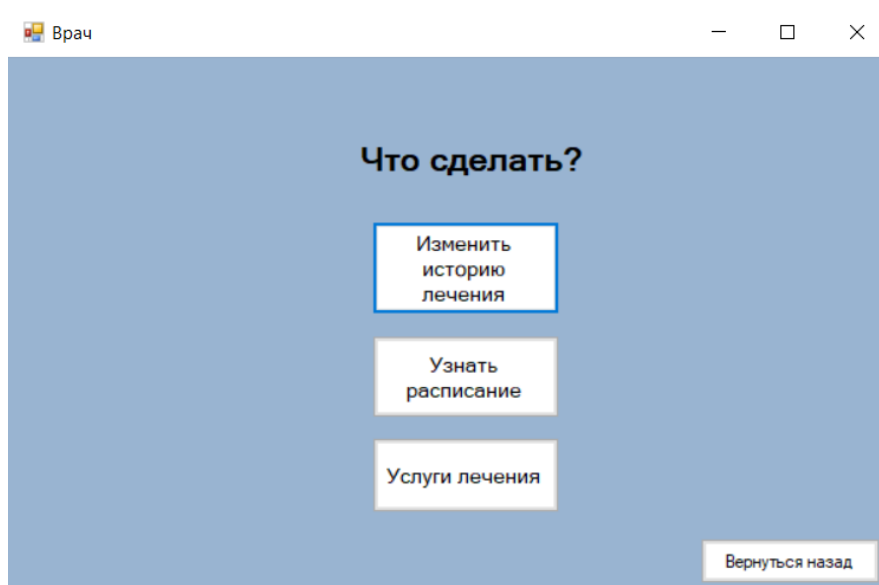


Рис. 2.27 Форма «Врач»

На этом этапе врачу доступно четыре действия:

1. Изменение истории лечения;
2. Просмотр расписания;

3. Изменение услуг лечения;
4. Возвращение на форму входа в приложение;

Начнем с изменения истории лечения. При нажатии на кнопку «Изменить историю лечения», врач переходит на форму «История лечения» (рис.2.28).

	ФИО Клиента	ФИО Сотрудника	Дата Лечение	Id Лечение	Ж
▶	Алешин А.А.	Дмитриев Д.Д.	05.05.2022	1111	Бо
	Сафонов К.М.	Мишин О.В.	05.05.2022	1112	Ка
	Клоков Т.Е.	Мишин О.В.	06.05.2022	1113	Бо
	Олейник А.А.	Дмитриев Д.Д.	06.05.2022	1114	Ка
	Мияйлова М.А.	Дмитриев Д.Д.	06.05.2022	1115	Ка
	Ногачева Д.М.	Афинин Ф.И.	07.05.2022	1116	На
	Павлов А.Д.	Мишин О.В.	07.05.2022	1117	Ка
	Михайленко Я.П.	Венина А.И.	07.05.2022	1118	Ка

ФИО Алешин А.А.

Рис. 2.28 Форма «История лечения»

На этой форме врач имеет следующие функции:

1. Просмотр историй лечений;
2. Поиск по историям лечения;
3. Добавление истории лечения;
4. Просмотр зубной формулы пациента.

Поиск по историям лечения происходит при заполнении текстового окна ComboBox и нажатии кнопки «Найти» (рис. 2.29).

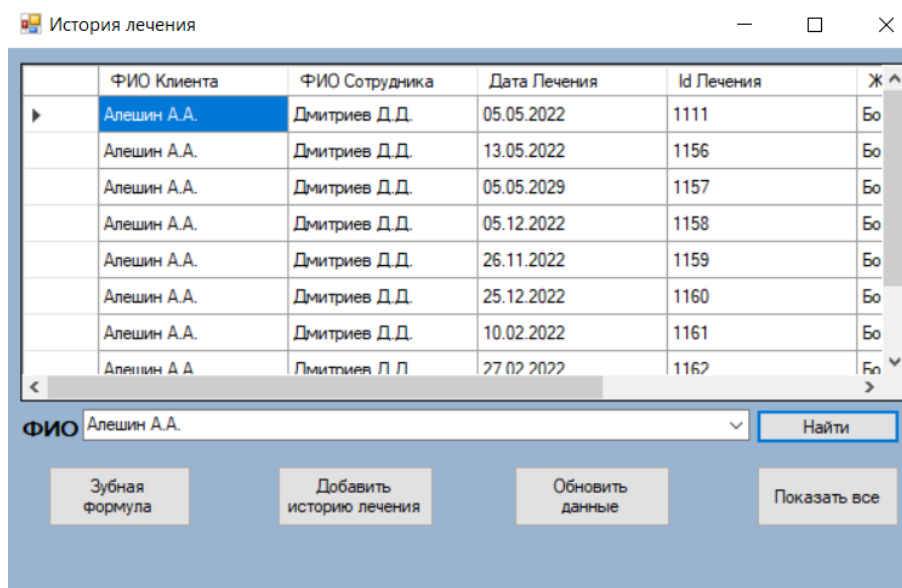


Рис. 2.29 Поиск по ФИО

Кнопка «Показать все» возвращает таблицу в исходное состояние. Кнопка «Обновить данные» обновляет данные в представленной на форме таблице.

При нажатии на кнопку «Добавить историю лечения», пользователь переходит на форму «Добавить историю лечения» (рис.2.30).

Рис. 2.30 Форма «Добавить историю лечения»

Кнопка «Добавить историю лечения» добавляет запись в базу данных с помощью хранимой процедуры dbo.addthr (приложение Б).

Перейдем к изменению зубной формулы пациента. При нажатии на кнопку «Зубная формула», открывается новая форма «Зубная формула» (рис. 2.31).

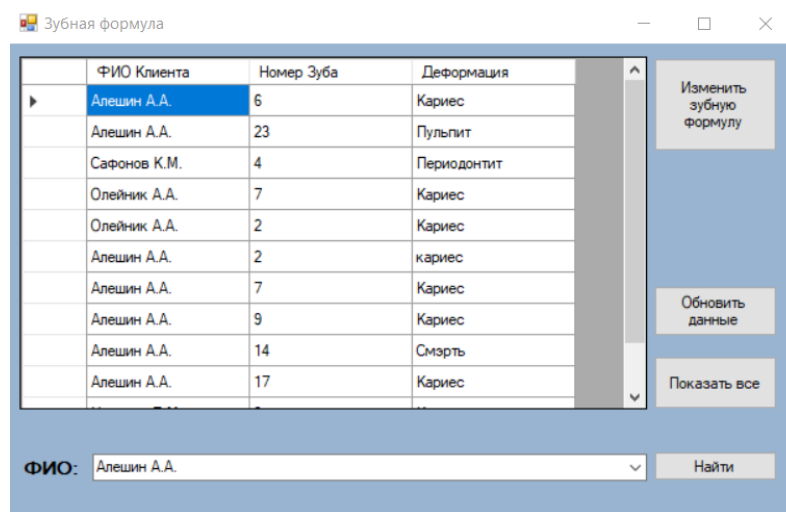


Рис. 2.31 Форма «Зубная формула»

На этой форме врач может производить поиск по уже существующим зубным формулам и добавлять новые (рис. 2.32 – 2.33).

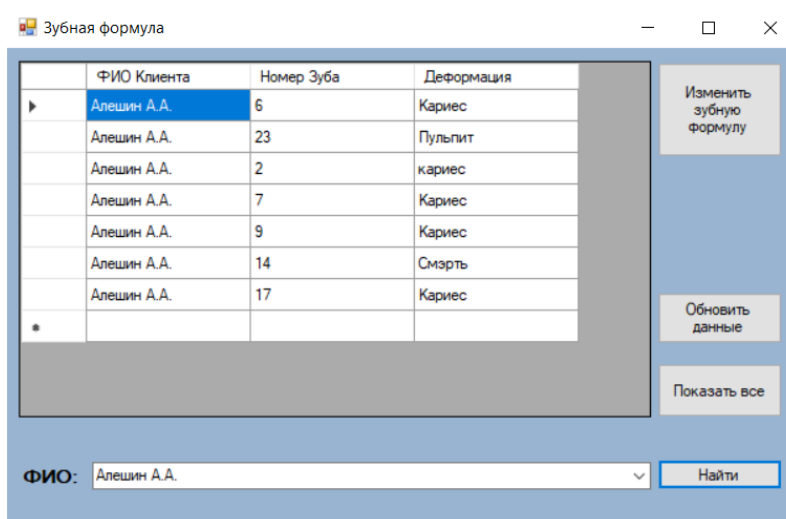


Рис. 2.32 Поиск по зубным формулам

**Введите данные**

ФИО клиента:

Номер зуба:

Деформация:

Рис. 2.33 Добавление новой зубной формулы

При изменении зубной формулы пациента, пользователь должен ввести необходимые данные и нажать на кнопку «Добавить данные», при этом сработает хранящая процедура dbo.adf (приложение Б).

Перейдем к возможностям медсестры (медбрата) (рис.2.34).

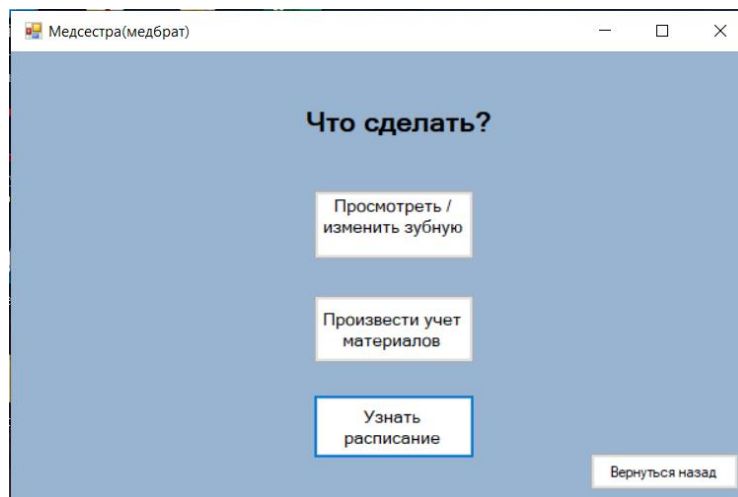


Рис. 2.34 Функции медсестры (медбрата)

Пользователю на данном этапе доступно три функции:

1. Изменение зубной формулы пациента;
2. Просмотр расписания;
3. Учет материалов.

Первые две функции аналогичны с соответствующими функциями у действующего лица врач, пропустим их и перейдем к учету материалов.

При нажатии на кнопку «Произвести учет материалов», пользователь переходит на форму «Материалы» (рис. 2.35).

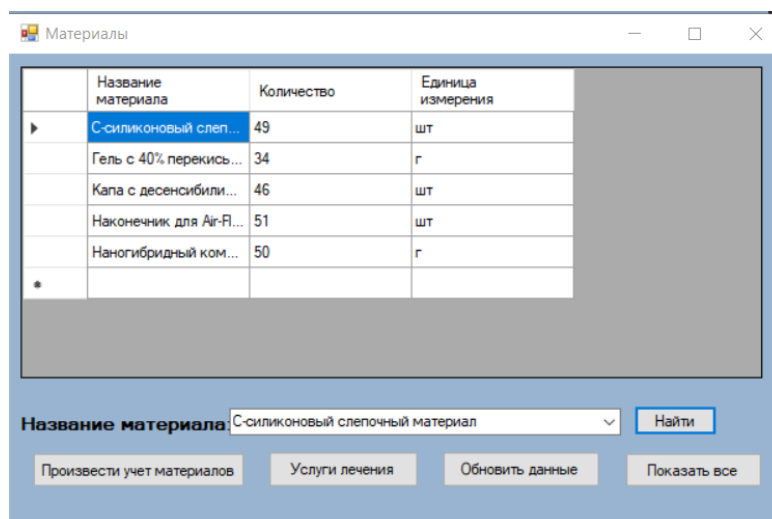


Рис. 2.35 Форма «Материалы»

На этом этапе пользователю доступны четыре функции:

1. Просмотр материалов;
2. Поиск по всем материалам;
3. Просмотр таблицы «Услуги лечения»;
4. Учет материалов.

Просмотр и поиск материалов осуществляется на форме «Материалы». Для более удобного поиска в нижней части формы представлен элемент текстового ввода ComboBox (рис. 2.36).

	Название материала	Количество	Единица измерения
▶	С-силиконовый слеп...	49	шт
*			

Название материала: С-силиконовый слепочный материал [Найти]

Произвести учет материалов   Услуги лечения   Обновить данные   Показать все

Рис. 2.36 Поиск материалов

Перейдем к просмотру услуг лечения. Нажав на кнопку «Услуги лечения», пользователь переходит на форму «Услуги лечения» (рис. 2.37).

	Название услуги	Id лечения
▶	Осмотр зубов	1111
	Осмотр зубов	1114
	Осмотр зубов	1119
	Пломба	1111
	Пломба	1112
	Пломба	1118
	Пломба	1164
	Полировка	1113
	Полировка	1164
	С-силиконовый слеп...	1112

Id лечения: 1111 [Найти]

Показать все

Рис. 2.37 Форма «Услуги лечения»

Для удобного поиска данных по таблице, внизу формы представлено окно поиска по Id лечения (рис. 2.38).

Название услуги	Id лечения
Осмотр зубов	1111
Пломба	1111

Id лечения: 1111

Найти Показать все

Рис. 2.37 Поиск по услугам лечения

Вернемся к учету материалов. Нажав на кнопку «Произвести учет материалов» на форме «Материалы», открывается форма «Учет материалов» (рис. 2.38).

Введите данные

Название материала:

Количество: 1

Единица измерения:

id услуги: 1111

Произвести учет

Рис. 2.37 Форма «Учет материалов»

Пользователь должен ввести все необходимые данные и нажать на кнопку «Произвести учет». При этом работает хранимая процедура `dbo.ExMat` (приложение Б).

## ЗАКЛЮЧЕНИЕ

Тема курсового проекта - проектирование базы данных для упрощения работы персонала стоматологической клиники и для автоматизации процессов записи на прием, формирования расписания, ведения историй лечения.

Этот проект очень актуален как для малых клиник, так и для крупных медицинский стоматологических центров, потому что он не только прост и удобен в использовании, но и значительно сокращает время обслуживания каждого клиента, заменяя рукописное ведение списков и медицинских карт на автоматизированное печатное.

В ходе выполнения работы были получены следующие результаты:

1. Изучена предметная область, проанализированы следующие примеры реализации данного проекта в приложениях «Стоматология» и «Стоматологическая клиника». Выделены основные функции, аспекты будущей системы.

2. На этапе концептуального проектирования выделены основные действующие лица БД: администратор, главный администратор, врач и медсестра (медбрат). Определены их функции, и в результате этого составлена диаграмма вариантов использования. По результатам концептуального проектирования выделены основные сущности, информация о которых будет храниться в БД.

3. По результатам предыдущего этапа построены ER-диаграммы для этих объектов, определены типы связей между сущностями и классы принадлежности. По данным типам связи и классам принадлежности сформированы отношения. Во все полученные отношения (сотрудник, клиент, услуга, расходные материалы, расход, услуги лечения, лечение, зубная формула, расписание, запись на прием) добавлены не ключевые атрибуты.

4. На этапе физического проектирования в среде SQL Server Management Studio были созданы таблицы и связи между ними. Для обеспечения полноценного функционирования программной части были разработаны



хранимые процедуры, функции и триггеры.

5. На заключительном этапе был разработан пользовательский интерфейс для взаимодействия с базой данных с помощью среды разработки Visual Studio 2019, языка программирования C#. На основе данных, которые были определены в процессе проектирования, была разработана БД, позволяющая выполнить различные функции в зависимости от роли пользователя. Для Администратора доступны следующие функции: изменение записи на прием, выдача информации об услугах, взаиморасчет с клиентом, а также добавление новых клиентов. Главный администратор имеет все возможности администратора, дополнительно ему доступно формирование расписания и просмотр списка сотрудников. Врач может изменять историю лечения и зубную формулу, просматривать расписание и изменять услуги лечения. Для медсестры (медбрата) доступны следующие функции: изменение зубной формулы пациента, учет материалов и просмотр расписания.

В итоге был получен программный продукт, полностью соответствующий поставленным задачам.

## Список используемых источников

1. Американская стоматологическая ассоциация Словарь стоматологических клинических и административных терминов [Электронный ресурс] / Американская стоматологическая ассоциация – 2016. – Режим доступа: [https://en.wikipedia.org/wiki/American\\_Dental\\_Association](https://en.wikipedia.org/wiki/American_Dental_Association), свободный. Дата обращения: 11.12.2022 г.
2. WikipediA [Электронный ресурс] – 2022. – Режим доступа: [https://en.wikipedia.org/wiki/Odonto-stomatology#cite\\_note-3](https://en.wikipedia.org/wiki/Odonto-stomatology#cite_note-3), свободный. Дата обращения: 11.12.2022 г.
3. Простой софт [Электронный ресурс] – 2022. – Режим доступа: <https://prostoysoft.ru/Dento.htm>, свободный. Дата обращения: 11.12.2022 г.
4. 1c:Медицина. Стоматологическая клиника [Электронный ресурс] – 2022. – Режим доступа: <https://solutions.1c.ru/catalog/stoma/features>, свободный. Дата обращения: 11.12.2022 г.
5. Документация по с# [Электронный ресурс] – Microsoft, 2022. – Режим доступа: <https://docs.microsoft.com/ru-ru/dotnet/csharp/>. Дата обращения: 11.12.2022 г.
6. Программирование на с# [Электронный ресурс] – 2019. – Режим доступа: <https://metanit.com/>, свободный. Дата обращения: 11.12.2022 г.
7. Документация по Работе с Com сервером Word [Электронный ресурс] – Microsoft, 2022. – Режим доступа: [http://www.wladm.narod.ru/C\\_Sharp/comword.html](http://www.wladm.narod.ru/C_Sharp/comword.html). Дата обращения: 11.12.2022 г.
8. Документация по SQL [Электронный ресурс] – Microsoft, 2022. – Режим доступа: <https://learn.microsoft.com/ru-ru/sql/?view=sql-server-ver16>. Дата обращения: 11.12.2022 г.

Заполнение таблиц данными

	id	FIO	gender	birthsday	tel	passport
►	111	Алешин А.А.	м	2000-01-01	899977766	1111
	112	Сафонов К....	м	1999-12-30	809865438	1044
	113	Клоков Т.Е	м	2002-11-12	863276237	8734
	114	Олейник А....	м	2003-02-23	865367465	6747
	115	Мияйлова ...	ж	1997-06-12	876545677	6787
	116	Ногаева Д....	ж	2003-04-28	890138527	1213
	117	Павлов А.Д	м	2002-01-09	892373712	2314
	118	Мищенко Я...	м	2002-01-12	948498322	9384
	119	Казаков С.Д.	м	1997-06-02	28482034	8743
	120	Савченко А....	м	1998-08-09	23423654	3456
	121	Молоканов...	м	2002-02-02	143526435	1345
	122	Зиневьев К...	м	2002-05-06	987654567	1346
	123	Нифонтов ...	м	2002-09-08	21345678	6899
	124	Кирияков С...	м	2002-09-09	132435465	4567
	125	Жидкая Е.Д.	ж	2001-07-08	98765456	6783
	126	Жириновск...	ж	2000-04-04	98656788	9292
	127	Клокова С....	ж	1999-02-02	24354657	2456
	128	Миронова ...	ж	2000-08-09	98765678	7654
	129	Рахманова ...	ж	2000-09-09	76543234	2345
	130	Пиркач Д.Ю.	м	1991-01-01	65456565	3546
	131	Саркисянц ...	м	2002-09-09	76567878	1232
	132	Беленко И.И	м	2003-09-01	13876588	1344
	133	Савоськин...	ж	1999-01-02	76545654	6545
	134	Шляпин И.Т	м	2002-07-09	64534536	977
	135	Искоркин ...	м	2003-02-02	356342562	2565
	136	Кривель И....	м	2002-09-08	987898898	6789
	137	Мирова М.М	ж	2002-09-08	563542343	1245
	138	Полмаргор...	ж	1999-08-02	968077908	3466
	139	Самохвало...	ж	2001-03-12	678346878	2578
	140	Рефн А.А	м	2000-01-30	897547895	8934

Рис. А.1. Данные таблицы «Клиент»

	Id_Emp	FIO	education	position	tel	salary	passport
	111	Дмитриев ...	Высшее	Стоматолог	98647238	50000,0000	7634
	112	Афинин Ф.И.	Среднее сп...	Медбрат	98643788	35000,0000	8765
	113	Мишин О.В.	Высшее	Стоматолог	98787364	50000,0000	6453
	114	Венина А.И.	Высшее	Администр...	98765673	35000,0000	3453
	115	Петров П.П.	Высшее	Главный ад...	98656473	40000,0000	6583
	116	Литвин А.Е.	Среднее	Медбрат	89363784	35000,0000	3456
	117	Литвина А.Е.	Среднее	Медсестра	89363783	35000,0000	3457
	118	Дмитриева...	Высшее	Стоматолог	98647233	50000,0000	7637

Рис. А.2. Данные таблицы «Сотрудник»

	NameMat	Amount	Unit
	С-силиконовый слепочный материал	35	шт
	Гель с 40% перекисью водорода	53	г
	Капа с десенсибилизирующим гелем	97	шт
►	Наконечник для Air-Flow для полирования зубов	60	шт
	Наногибридный композиционный материал	76	г

Рис. А.3. Данные таблицы «Расходные материалы»

	NameServ	Price
►	Осмотр зубов	1000,0000
	Отбеливание	5000,0000
	Пломба	3200,0000
	Полировка	4500,0000
	Слепок зубов	1500,0000

Рис. А.4. Данные таблицы «Услуга»

	Id_Treat	Id_Client	Id_Emp	DateTH	Claim	Rec
►	1111	111	111	2022-05-05	Боль в 6 зу...	Полоскание
	1112	112	113	2022-05-05	Кариес в 1...	Лечение вс...
	1113	113	113	2022-05-06	Боль в 8 зу...	Чистка зуб...
	1114	114	111	2022-05-06	Кариес в 1...	Полоскание
	1115	115	111	2022-05-06	Кариес в 1...	NULL
	1116	116	112	2022-05-07	Налёт на зу...	Чистка зуб...
	1117	117	113	2022-05-07	Кариес в 8 ...	Полоскание
	1118	118	114	2022-05-07	Кариес в 1 ...	Полоскание
	1119	119	115	2022-05-08	Боль в 8 зу...	Чистка зуб...
	1120	120	111	2022-05-08	Боль в 15 з...	Чистка зуб...
	1121	121	115	2022-05-09	Налёт на зу...	Чистка зуб...
	1122	122	111	2022-05-10	Боль в 13 з...	Чистка зуб...
	1123	123	111	2022-05-10	Кариес в 1 ...	Полоскание
	1124	124	111	2022-05-10	Налёт на зу...	Чистка зуб...
	1125	125	112	2022-05-11	Кариес в 6 ...	Полоскание
	1126	126	112	2022-05-11	Боль в 5 зу...	Лечение вс...
	1127	127	112	2022-05-11	Налёт на зу...	Полоскание
	1128	128	112	2022-05-11	Боль в 16 з...	Чистка зуб...
	1129	129	113	2022-05-12	Кариес в 4 ...	Лечение вс...
	1130	130	113	2022-05-12	Боль в 3 зу...	Полоскание
	1131	131	113	2022-05-12	Налёт на зу...	Полоскание
	1132	132	113	2022-05-12	Кариес в 8 ...	Лечение вс...
	1133	133	114	2022-05-13	Боль в 10 з...	Полоскание
	1134	134	114	2022-05-13	Кариес во ...	Полоскание
	1135	135	114	2022-05-13	Налёт на зу...	Полоскание
	1136	136	114	2022-05-13	Боль в 4 зу...	Чистка зуб...
	1137	137	115	2022-05-14	Налёт на зу...	Полоскание
	1138	138	115	2022-05-14	Боль в 1 зу...	Чистка зуб...
	1139	139	115	2022-05-14	Кариес в 3 ...	Полоскание
	1140	140	115	2022-05-14	Боль в 10 з...	Полоскание

Рис. А.5. Данные таблицы «Лечение»

	Id_DF	Id_Client	Tooth	Deformation
	1111	111	6	Кариес
	1112	111	23	Пульпит
	1113	112	4	Периодонт...
	1114	114	7	Кариес
	1115	114	2	Кариес
	1116	111	2	кариес
	1117	111	7	Кариес
	1118	111	9	Кариес
	1119	111	14	Кариес
	1120	111	17	Кариес
	1121	116	9	Кариес
	1122	111	23	Пульпит
	1123	116	14	Кариес
	1124	112	14	Кариес
	1125	113	14	Кариес
	116	112	4	Периодонт...
	1	111	2	кариес
	2	111	14	Кариес
	3	111	17	Кариес
	4	116	14	Кариес
	5	114	7	Кариес
	6	114	2	Кариес
	7	111	23	Пульпит
	8	117	4	Пульпит
	9	145	2	Пульпит
	10	176	23	Пульпит

Рис. А.6. Данные таблицы «Зубная формула»

	Id_TT	Id_Emp	NameDay	TimeReceipt
	11111	111	2022-05-05	10:00:00
	11112	111	2022-05-05	14:00:00
	11115	113	2022-05-06	10:00:00
	11117	111	2022-05-06	12:00:00
	11125	111	2022-11-03	12:00:00
	11126	111	2022-05-05	10:00:00
	11127	113	2022-05-05	10:00:00
	11113	113	2022-05-06	10:00:00
	11114	111	2022-05-06	12:00:00
	11116	111	2022-05-05	10:00:00
	11118	113	2022-05-05	10:00:00
	11119	113	2022-05-05	14:00:00
	11120	111	2022-05-06	10:00:00
	11121	113	2022-05-06	12:00:00
	11122	111	2022-11-03	12:00:00
	11123	111	2022-05-05	10:00:00
	11124	113	2022-05-05	10:00:00
	11128	111	2022-05-06	12:00:00
	11129	111	2022-11-03	12:00:00
	11130	111	2022-05-05	10:00:00
	11131	114	2022-05-05	10:00:00
	11132	111	2022-05-05	10:00:00
	11133	111	2022-05-05	14:00:00
	11134	113	2022-05-06	10:00:00
	11135	113	2022-05-06	12:00:00
	11136	113	2022-11-03	12:00:00
	11137	113	2022-05-05	10:00:00

Рис. А.7. Данные таблицы «Расписание»

	id_Rec	Id_Client	Service_Ma...	Datet	Timet	Vrach
▶	1121	111	Боль в чел...	2024-05-05	10:00:00	Дмитриев ...
	1122	111	Боль в чел...	2022-05-05	10:00:00	Дмитриев ...
	1123	111	Боль в чел...	2022-05-05	14:00:00	Дмитриев ...
	1124	129	Боль в чел...	2022-05-05	17:00:00	Мишин О.В.
	1125	132	Боль в чел...	2022-12-30	14:00:00	Дмитриев ...
	1126	133	Боль в чел...	2022-11-10	17:00:00	Мишин О.В.
	1127	111	Боль в чел...	2022-04-26	13:00:00	Дмитриев ...
	1128	129	Боль в чел...	2022-04-29	12:00:00	Мишин О.В.
	1129	126	Боль в чел...	2022-10-29	13:00:00	Мишин О.В.
	1130	130	Боль в чел...	2022-06-05	12:00:00	Мишин О.В.
	1131	123	Боль в чел...	2022-06-04	12:00:00	Мишин О.В.
	1132	124	Боль в чел...	2022-08-20	12:00:00	Мишин О.В.
	1133	131	Боль в чел...	2022-08-20	16:00:00	Мишин О.В.
	1134	131	Боль в чел...	2022-08-20	16:00:00	Дмитриев ...
	1135	131	Боль в чел...	2022-08-05	16:00:00	Дмитриев ...
	1136	131	Боль в чел...	2022-08-02	16:00:00	Дмитриев ...
	1137	114	Боль в чел...	2022-07-08	12:00:00	Мишин О.В.
	1138	121	Боль в чел...	2022-07-08	15:00:00	Мишин О.В.
	1139	140	Боль в чел...	2022-01-12	12:00:00	Мишин О.В.
	1140	111	Боль в чел...	2022-10-30	13:00:00	Мишин О.В.
	1141	111	Боль в чел...	2022-10-31	13:00:00	Мишин О.В.
	1143	138	Боль в чел...	2022-09-08	12:00:00	Дмитриев ...
	1144	125	Боль в чел...	2022-09-25	13:00:00	Дмитриев ...
	1145	118	Боль в чел...	2022-11-17	12:00:00	Мишин О.В.
	1146	155	Боль в чел...	2022-07-12	12:00:00	Дмитриев ...
	1147	120	Боль в чел...	2022-11-23	12:00:00	Дмитриев ...
	1148	111	Боль в чел...	2022-05-05	20:00:00	Дмитриев ...
	1149	111	Боль в чел...	2022-05-05	21:00:00	Дмитриев ...
	1150	111	Боль в чел...	2022-12-22	12:23:00	Дмитриев ...
	1151	119	Боль в чел...	2022-12-06	12:23:00	Дмитриев ...

Рис. А.8. Данные таблицы «Запись на прием»

	NameServ	Id_Thr
	Осмотр зуб...	1111
	Осмотр зуб...	1114
	Осмотр зуб...	1119
	Пломба	1111
	Пломба	1112
	Пломба	1118
	Пломба	1164
	Полировка	1113
	Полировка	1164
	Слепок зуб...	1112
	Пломба	1113
	Пломба	1115
	Полировка	1114
	Слепок зуб...	1116
	Осмотр зуб...	1115
	Пломба	1117
	Пломба	1116
	Полировка	1118
	Слепок зуб...	1117
	Пломба	1120
	Полировка	1119
	Слепок зуб...	1121
	Осмотр зуб...	1120
	Пломба	1122
	Пломба	1121
	Полировка	1123
	Слепок зуб...	1122
	Осмотр зуб...	1124
	Пломба	1123
	Пломба	1125

Рис. А.9. Данные таблицы «Услуги лечения»

	Id_Thr	NameMat	Amount	Unit
	1111	С-силикон...	1	шт
	1114	С-силикон...	1	шт
	1111	Гель с 40% ...	1	г
	1113	Гель с 40% ...	1	г
	1114	Гель с 40% ...	1	г
	1111	Капа с десе...	2	шт
	1119	Капа с десе...	1	шт
	1113	Наконечни...	1	шт
	1111	Наногибри...	2	г
	1112	Наногибри...	4	г
	1164	Наногибри...	1	г

Рис. А.10. Данные таблицы «Расход»



**SQL-команды создания программных объектов БД**

1. Триггер Payment\_INSERT

```
CREATE TRIGGER [dbo].[Payment_INSERT]
ON [dbo].[Client]
INSTEAD OF INSERT
AS
BEGIN
DECLARE @lastId INT
SELECT @lastId = Max(id) FROM Client
INSERT INTO Client SELECT @lastId+1, FIO, gender, birthday, tel,
passport FROM inserted
END
```

2. Хранимая процедура addrec

```
CREATE PROCEDURE [dbo].[addrec]
@fiosotr varchar(max), @sm varchar(max), @dt Date, @time time, @vr
varchar(max), @a int output
AS
BEGIN
    declare @idrec int
    declare @idcl int
    declare @idtt int
    declare @idvr int
    set @a = 0
    set @idtt = (select max(Id_TT) from TimeTable) + 1
    set @idvr = (select Id_Emp from Employee where @vr = FIO)
```

```

SELECT FIO, id_rec, id_Client, Service_Mark, Datet, Timet from
Record join Client on Client. id = Record.Id_Client

set @idrec = (select max(id_Rec) from Record) + 1

set @idcl = (select Client.Id from Client where FIO = @fiosotr)

if exists (select id_Client, Service_Mark, Datet, Vrach, Timet from
Record where @idcl = Id_Client and @sm = Service_Mark and @dt = Datet
and @time = Timet and @vr = Vrach)

set @a = 1

else if exists (select Datet, Vrach, Timet from Record where @dt = Datet
and @vr = Vrach and @time = Timet)

set @a = 2

else if exists (select Datet, Id_Client, Timet from Record where @dt =
Datet and @idcl = Id_Client and @time = Timet)

set @a = 3

else

insert into Record values (@idrec, @idcl, @sm, @dt, @time, @vr)

insert into TimeTable values (@idtt, @idvr, @dt, @time)

END

```

### 3. Хранимая процедура addservthr

```

CREATE PROCEDURE [dbo].[addservthr]
@name varchar(max), @id int
AS
BEGIN

```

```

insert into ServThr values (@name, @id)

```

```

END

```

### 4. Хранимая процедура addthr

```

CREATE PROCEDURE [dbo].[addthr]

```

```
@fiocl varchar(max), @fioemp varchar(max), @date date, @claim  
varchar(max), @rec varchar(max), @a int output
```

```
AS
```

```
BEGIN
```

```
set @a = 0
```

```
declare @idcl int, @idemp int, @idthr int
```

```
select Id_Treat, Id_Client, Employee.Id_Emp, DateTH, Claim, Rec from  
Treatment, Client, Employee where Treatment.Id_Client = Client.id and  
Employee.Id_Emp = Treatment.Id_Emp
```

```
set @idthr = (select max(Id_Treat) from Treatment) + 1
```

```
set @idemp = (select Id_Emp from Employee where @fioemp = FIO)
```

```
set @idcl = (select Id from Client where @fiocl = FIO)
```

```
if @date > GETDATE()
```

```
set @a = 2
```

```
else if exists (select Id_Client, Id_Emp, DateTH from Treatment where  
Id_Client = @idcl and Id_Emp = @idemp and DateTH = @date)
```

```
set @a = 1
```

```
else
```

```
insert into Treatment values (@idthr, @idcl, @idemp, @date, @claim, @rec)
```

```
END
```

## 5. Хранимая процедура adf

```
CREATE PROCEDURE [dbo].[adf]
```

```
@fiocl varchar(max), @tooth int, @def varchar(max), @a int output
```

```
AS
```

```
BEGIN
```

```
set @a = 0
```

```

Declare @idf int, @idcl int

select Id_DF, Id_Client, Tooth, Deformation, FIO from DentalFormula, Client
where DentalFormula.Id_Client = Client.id

set @idf = (select max(Id_DF) from DentalFormula) + 1

set @idcl = (select Id from Client where FIO = @fiocl)

if exists (select Id_Client, Tooth from DentalFormula where Id_Client =
@idcl and Tooth = @tooth)

set @a = 1

else

insert into DentalFormula values (@idf, @idcl, @tooth, @def)

END

```

#### 6. Хранимая процедура [dbo].[crcl]

```
CREATE PROCEDURE [dbo].[crcl]
```

```

@fiocl varchar(max), @gen char(1), @birth Date, @tel int, @pass int, @a int
output

```

```
AS
```

```
BEGIN
```

```
    set @a = 0
```

```
    SELECT Id, FIO, gender, birthday, tel, passport from Client
```

```

    if exists (select @fiocl, @gen, @birth, @tel, @pass from Client where
@fiocl = FIO and @gen = gender and @birth = birthday and @tel = tel and
@pass = passport)

```

```
    set @a = 1
```

```
    insert into Client values (-1, @fiocl, @gen, @birth, @tel, @pass)
```

```
END
```

#### 7. Хранимая процедура delrec

```
CREATE PROCEDURE [dbo].[delrec]
```

```

@fiosotr varchar(max), @dt Date, @time Time

AS

BEGIN

    declare @idcl int

    declare @idvr int

    declare @vr varchar(max)

    SELECT FIO, id_Client, Datet, Timet from Record join Client on
Client. id = Record.Id_Client

    set @idcl = (select Client.Id from Client where FIO = @fiosotr)

    set @vr = (select Vrach from Record where @idcl = Id_Client and @dt
= Datet and @time = Timet)

    set @idvr = (select Id_Emp from Employee where @vr = FIO)

    delete from Record where @idcl = id_Client and @dt = Datet and
@time = Timet

    delete from TimeTable where @idvr = Id_Emp and @dt = NameDay
and @time = TimeReceipt

END

```

#### 8. Хранимая процедура DelttimeTable

```

CREATE PROCEDURE [dbo].[DelttimeTable]

@fiosotr varchar(max), @NameD Date, @TimeR Time

AS

BEGIN

    declare @maxidtt int;

    declare @idemp int

    SELECT Max(Id_TT) as mit, Employee.Id_Emp, NameDay,
TimeReceipt, FIO from TimeTable join Employee on TimeTable.Id_Emp =
Employee.Id_Emp group by Employee.Id_Emp, NameDay, TimeReceipt, FIO

```

```
set @idemp = (select Employee.Id_Emp from Employee where FIO =  
@fiosotr)
```

```
delete from TimeTable where @NameD = NameDay and @TimeR =  
TimeReceipt and @idemp = Id_Emp
```

```
END
```

#### 9. Хранимая процедура ExMat

```
CREATE PROCEDURE [dbo].[ExMat]
```

```
@idserv int, @amount int, @unit varchar(2), @namemat varchar(max)
```

```
AS
```

```
BEGIN
```

```
insert into Expenses values (@idserv, @namemat, @amount, @unit)
```

```
set @amount = (select Amount from Material where NameMat =  
@namemat) - @amount
```

```
update Material set Amount = @amount where NameMat = @namemat
```

```
END
```

#### 10. Хранимая процедура NimeTable

```
CREATE PROCEDURE [dbo].[NimeTable]
```

```
@fiosotr varchar(max), @NameD Date, @TimeR Time
```

```
AS
```

```
BEGIN
```

```
declare @maxidtt int;
```

```
declare @idemp int
```

```
SELECT Max(Id_TT) as mit, Employee.Id_Emp, NameDay,  
TimeReceipt, FIO from TimeTable join Employee on TimeTable.Id_Emp =  
Employee.Id_Emp group by Employee.Id_Emp, NameDay, TimeReceipt, FIO
```

```
set @maxidtt = (select max(Id_TT) from TimeTable) + 1;
```

```
set @idemp = (select Employee.Id_Emp from Employee where FIO =
```

@fiosotr)

insert into TimeTable values (@maxidtt, @idemp, @NameD, @TimeR)

END

11.Хранимая процедура param

CREATE PROCEDURE [dbo].[param]

@id int, @date date output, @cl varchar(max) output, @vr varchar(max)  
output

AS

BEGIN

declare @idvr int , @idcl int

set @date = (select DateTH from Treatment where @id = Id\_Treat)

set @idcl = (select Id\_Client from Treatment where @id = Id\_Treat)

set @idvr = (select Id\_Emp from Treatment where @id = Id\_Treat)

set @cl = ( select FIO from Client where id = @idcl)

set @vr = ( select FIO from Employee where Id\_Emp = @idvr)

END

12.Хранимая процедура pass

CREATE PROCEDURE [dbo].[pass]

@log varchar(max), @pass int output

AS

BEGIN

set @pass = (select tel from Employee where FIO = @log)

END

13.Хранимая процедура reg

CREATE PROCEDURE [dbo].[reg]

@fiosotr varchar(max), @tel int, @a int output

AS

BEGIN

set @a = 0

declare @dolj varchar(max)

select FIO, tel, position from Employee

set @dolj = (select position from Employee where @fiosotr = FIO and @tel = tel)

if not exists (select tel, FIO from Employee where @tel = tel and FIO = @fiosotr)

set @a = 2

if @dolj = 'Стоматолог'

set @a = 3

if @dolj = 'Медбрат' or @dolj = 'Медсестра'

set @a = 4

if @dolj = 'Администратор'

set @a = 5

if @dolj = 'Главный администратор'

set @a = 6

END

#### 14. Функция Nill

CREATE FUNCTION [dbo].[Nill]

(

    @id int

)

RETURNS

@res TABLE (NameServ VARCHAR (30), Price int)



AS

BEGIN

INSERT INTO @res SELECT ServThr.NameServ, Price FROM ServThr,  
Service WHERE ServThr.NameServ = Service.NameServ and Id\_Thr = @id  
group by ServThr.NameServ, Price

RETURN

END

15. Функция price

CREATE FUNCTION [dbo].[price]

(

    @id int

)

RETURNS INT

AS

BEGIN

    DECLARE @answer INT

    SELECT @answer = Sum (Price) FROM dbo.Nill(@id)

    RETURN @answer

END

## Код программы

## 1. Форма «Вход»

```

public partial class Form5 : Form
{
    byte EnterTry = 3;

    public Form5()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        if (EnterTry == 0)
        {
            button1.Enabled = false;
            MessageBox.Show("У вас закончились попытки входа", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        try
        {
            sqlCommand1.Parameters["@fiosotr"].Value = textBox1.Text;
            sqlCommand1.Parameters["@tel"].Value = textBox2.Text;
            sqlConnection1.Open();
            sqlCommand1.ExecuteNonQuery();
            sqlConnection1.Close();
            if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 2)
            {
                MessageBox.Show($"Повторите ввод, количество попыток: {EnterTry}",
                    "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                EnterTry--;
                textBox1.Clear();
                textBox2.Clear();
            }
            else if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 3)
            {
                Form form10 = new Form10();
                form10.Show();
                this.Hide();
            }
            else if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 4)
            {
                Form form11 = new Form11();
                form11.Show();
                this.Hide();
            }
            else if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 5)
            {
                Form form2 = new Form2();
                form2.Show();
                this.Hide();
            }
            else if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 6)
            {
                Form form9 = new Form9();
                form9.Show();
                this.Hide();
            }
        }
    }
}

```

```

        catch
        {
            MessageBox.Show($"Повторите ввод, количество попыток: {EnterTry}", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            EnterTry--;
            textBox1.Clear();
            textBox2.Clear();
            sqlCommand1.Parameters["@fiosotr"].Value = 0;
            sqlCommand1.Parameters["@tel"].Value = 0;
            sqlCommand1.Parameters["@a"].Value = 0;

            sqlConnection1.Close();
        }
    }

    private void label4_Click(object sender, EventArgs e)
    {
        Form form31 = new Form31();
        form31.Show();
    }
}

```

## 2. Форма «Добавление клиента»

```

public partial class Form6 : Form
{
    public Form6()
    {
        InitializeComponent();
    }

    private void clientBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.clientBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
    }

    private void Form6_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Client". При необходимости она может быть перемещена или удалена.
        this.clientTableAdapter.Fill(this.dentistryDataSet.Client);
    }

    private void fIOLabel_Click(object sender, EventArgs e)
    {
    }

    private void genderLabel1_Click(object sender, EventArgs e)
    {
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            sqlCommand1.Parameters["@fiocl"].Value = fIOTextBox.Text;
            sqlCommand1.Parameters["@gen"].Value = genderComboBox.Text;
            sqlCommand1.Parameters["@birth"].Value = birthDayDateTimePicker.Text;
            sqlCommand1.Parameters["@tel"].Value = textBox1.Text;

```

```

        sqlCommand1.Parameters["@pass"].Value = passportTextBox.Text;
        sqlConnection1.Open();
        sqlCommand1.ExecuteNonQuery();
        sqlConnection1.Close();
        if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) > 0)
            MessageBox.Show("Клиент уже был добавлен!", "Добавление клиента  
невозможно ", MessageBoxButtons.OK, MessageBoxIcon.Error);
        else
            MessageBox.Show("Клиент добавлен!", "Добавление клиента ",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch
    {
        MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
}

```

### 3. Форма «Добавить запись на прием»

```

public partial class Form8 : Form
{
    public Form8()
    {
        InitializeComponent();
    }

    private void recordBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.recordBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
    }

    private void recordBindingNavigatorSaveItem_Click_1(object sender, EventArgs e)
    {
        this.Validate();
        this.recordBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
    }

    private void Form8_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Employee". При необходимости она может быть перемещена или удалена.
        this.employeeTableAdapter.Fill(this.dentistryDataSet.Employee);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet1.Client". При необходимости она может быть перемещена или удалена.
        this.clientTableAdapter.Fill(this.dentistryDataSet1.Client);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet1.Client". При необходимости она может быть перемещена или удалена.
        this.clientTableAdapter.Fill(this.dentistryDataSet1.Client);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Client". При необходимости она может быть перемещена или удалена.
        this.clientTableAdapter.Fill(this.dentistryDataSet.Client);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Record". При необходимости она может быть перемещена или удалена.
        this.recordTableAdapter.Fill(this.dentistryDataSet.Record);
    }

    private void button1_Click(object sender, EventArgs e)
    {

```

```

try
{
    sqlCommand1.Parameters["@fiosotr"].Value = fIOComboBox.Text;
    sqlCommand1.Parameters["@sm"].Value = service_MarkTextBox.Text;
    sqlCommand1.Parameters["@dt"].Value = datetDateTimePicker.Text;
    sqlCommand1.Parameters["@time"].Value = Convert.ToString(textBox1.Text);
    sqlCommand1.Parameters["@vr"].Value = fIOComboBox1.Text;
    sqlConnection1.Open();
    sqlCommand1.ExecuteNonQuery();
    sqlConnection1.Close();
    if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 1)
        MessageBox.Show("Запись уже была добавлена!", "Добавление записи  
невозможно ", MessageBoxButtons.OK, MessageBoxIcon.Error);
    else if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 2)
        MessageBox.Show("Врач занят в это время!", "Добавление записи невозможно ",
        MessageBoxButtons.OK, MessageBoxIcon.Error);
    else if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 3)
        MessageBox.Show("Вы не можете записаться на это время", "Добавление записи  
невозможно ", MessageBoxButtons.OK, MessageBoxIcon.Error);
    else
        MessageBox.Show("Запись добавлена!", "Добавление записи",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
}
catch
{
    MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
    MessageBoxIcon.Error);
}
}
}

```

#### 4. Форма «Расписание»

```

public partial class Form12 : Form
{
    public Form12()
    {
        InitializeComponent();
    }

    private void Form12_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_2". При необходимости она может быть перемещена или удалена.
        this.view_2TableAdapter.Fill(this.dentistryDataSet.View_2);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            if (radioButton1.Checked)
                view_2BindingSource.Filter = "FIO='" + comboBox1.Text + "'";
            else
                view_2BindingSource.Filter = "NameDay =' " +
                Convert.ToString(nameDayDateTimePicker.Value) + "'";
        }
        catch
        {
            MessageBox.Show("Повторите ввод", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {

```

```

        view_2BindingSource.Filter = "";
    }
}

```

## 5. Форма «Добавить запись в расписание»

```

public Form13()
{
    InitializeComponent();
}

private void employeeBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    this.Validate();
    this.employeeBindingSource.EndEdit();
    this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
}

private void Form13_Load(object sender, EventArgs e)
{
    // TODO: данная строка кода позволяет загрузить данные в таблицу
    "dentistryDataSet.TimeTable". При необходимости она может быть перемещена или удалена.
    this.timeTableTableAdapter.Fill(this.dentistryDataSet.TimeTable);
    // TODO: данная строка кода позволяет загрузить данные в таблицу
    "dentistryDataSet.Employee". При необходимости она может быть перемещена или удалена.
    this.employeeTableAdapter.Fill(this.dentistryDataSet.Employee);
}

private void button1_Click(object sender, EventArgs e)
{
    try
    {
        myCommand.Parameters["@fiosotr"].Value = fIOComboBox.Text;
        myCommand.Parameters["@NameD"].Value = dateTimePicker1.Text;
        myCommand.Parameters["@TimeR"].Value = textBox1.Text;
        mySqlConnection.Open();
        myCommand.ExecuteNonQuery();
        mySqlConnection.Close();
        MessageBox.Show("Изменение расписания", "Запись добавлена!",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch
    {
        MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}

private void button2_Click(object sender, EventArgs e)
{
    try
    {
        myCommand.Parameters["@fiosotr"].Value = fIOComboBox.Text;
        myCommand.Parameters["@NameD"].Value = dateTimePicker1.Text;
        myCommand.Parameters["@TimeR"].Value = textBox1.Text;
        mySqlConnection.Open();
        myCommand.ExecuteNonQuery();
        mySqlConnection.Close();
        MessageBox.Show("Изменение расписания", "Запись удалена!",
        MessageBoxButtons.OK, MessageBoxIcon.Information);
    }
    catch
    {

```

```

        MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
        MessageBoxIcon.Error);
    }
}
void formClosed(object sender, FormClosedEventArgs e)
{
    this.Show();
}

private void button4_Click(object sender, EventArgs e)
{
    Form14 form14 = new Form14();
    form14.Show();
}
}
}

```

## 6. Форма «Изменение расписания»

```

public partial class Form14 : Form
{
    public Form14()
    {
        InitializeComponent();
    }

    private void Form14_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_2". При необходимости она может быть перемещена или удалена.
        this.view_2TableAdapter.Fill(this.dentistryDataSet.View_2);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            if (radioButton1.Checked)
                view_2BindingSource.Filter = "FIO='" + comboBox1.Text + "'";
            else
                view_2BindingSource.Filter = "NameDay = '" +
                Convert.ToString(nameDayDateTimePicker.Value) + "'";
        }
        catch
        {
            MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        view_2BindingSource.Filter = "";
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form form13 = new Form13();
        form13.Show();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        Form form15 = new Form15();
        form15.Show();
    }
}

```

```

    }

    private void fIOTextBox_TextChanged(object sender, EventArgs e)
    {
    }

    private void nameDayDateTimePicker_ValueChanged(object sender, EventArgs e)
    {
    }

    private void button6_Click(object sender, EventArgs e)
    {
        view_2BindingSource.DataSource = view_2TableAdapter.GetData();
    }
}

```

## 7. Форма «Удалить запись»

```

public partial class Form15 : Form
{
    public Form15()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            sqlCommand1.Parameters["@fiosotr"].Value = fIOComboBox.Text;
            sqlCommand1.Parameters["@NameD"].Value = dateTimePicker1.Text;
            sqlCommand1.Parameters["@TimeR"].Value = textBox1.Text;
            sqlConnection1.Open();
            sqlCommand1.ExecuteNonQuery();
            sqlConnection1.Close();
            MessageBox.Show("Изменение расписания", "Запись добавлена!",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch
        {
            MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }

    private void button4_Click(object sender, EventArgs e)
    {
        Form14 form14 = new Form14();
        form14.Show();
    }
}

```

## 8. Форма «Удалить запись»

```

public partial class Form17 : Form
{
    public Form17()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
    }
}

```



```

        try
        {
            sqlCommand1.Parameters["@fiosotr"].Value = fIOComboBox.Text;
            sqlCommand1.Parameters["@dt"].Value = datetDateTimePicker.Text;
            sqlCommand1.Parameters["@time"].Value = textBox1.Text;
            sqlCommand1.Connection.Open();
            sqlCommand1.ExecuteNonQuery();
            sqlCommand1.Connection.Close();
            MessageBox.Show("Запись удалена!", "Удаление записи", MessageBoxButtons.OK,
            MessageBoxIcon.Information);
        }
        catch
        {
            MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
            MessageBoxIcon.Error);
        }
    }

    private void clientBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.clientBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
    }

    private void Form17_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Client". При необходимости она может быть перемещена или удалена.
        this.clientTableAdapter.Fill(this.dentistryDataSet.Client);
    }
}

```

## 9. Форма «Учет материалов»

```

public partial class Form18 : Form
{
    public Form18()
    {
        InitializeComponent();
    }

    private void materialBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.materialBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
    }

    private void Form18_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.ServThr". При необходимости она может быть перемещена или удалена.
        this.servThrTableAdapter.Fill(this.dentistryDataSet.ServThr);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Service". При необходимости она может быть перемещена или удалена.
        this.serviceTableAdapter.Fill(this.dentistryDataSet.Service);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Expenses". При необходимости она может быть перемещена или удалена.
        this.expensesTableAdapter.Fill(this.dentistryDataSet.Expenses);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Material". При необходимости она может быть перемещена или удалена.
    }
}

```

```

        this.materialTableAdapter.Fill(this.dentistryDataSet.Material);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            sqlCommand1.Parameters["@idserv"].Value = id_ThrComboBox.Text;
            sqlCommand1.Parameters["@namemat"].Value = comboBox2.Text;
            sqlCommand1.Parameters["@amount"].Value = amountTextBox.Text;
            sqlCommand1.Parameters["@unit"].Value = comboBox1.Text;
            sqlConnection1.Open();
            sqlCommand1.ExecuteNonQuery();
            sqlConnection1.Close();
            MessageBox.Show("Учет произведен!", "Учет материалов", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        }
        catch
        {
            MessageBox.Show("Повторите попытку", "Ошибка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
}

```

## 10. Форма «Зубная формула»

```

public partial class Form19 : Form
{
    public Form19()
    {
        InitializeComponent();
    }

    private void Form19_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Client". При необходимости она может быть перемещена или удалена.
        this.clientTableAdapter.Fill(this.dentistryDataSet.Client);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_3". При необходимости она может быть перемещена или удалена.
        this.view_3TableAdapter.Fill(this.dentistryDataSet.View_3);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            view_3BindingSource.Filter = "FIO Like'" + fIOComboBox.Text + "%'";
        }
        catch
        {
            MessageBox.Show("Ошибка. Повторите ввод");
        }
    }

    private void label1_Click(object sender, EventArgs e)
    {
    }

    private void button5_Click(object sender, EventArgs e)
    {
    }
}

```



```

        catch
        {
            MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
}

```

## 12. Форма «Добавить историю лечения»

```

public partial class Form22 : Form
{
    public Form22()
    {
        InitializeComponent();

        private void Form22_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "dentistryDataSet.Treatment". При необходимости она может быть перемещена или удалена.
            this.treatmentTableAdapter.Fill(this.dentistryDataSet.Treatment);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "dentistryDataSet.Employee". При необходимости она может быть перемещена или удалена.
            this.employeeTableAdapter.Fill(this.dentistryDataSet.Employee);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "dentistryDataSet.Client". При необходимости она может быть перемещена или удалена.
            this.clientTableAdapter.Fill(this.dentistryDataSet.Client);

        }

        private void fIOLabel1_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                // Console.WriteLine(Convert.ToInt32(sqlCommand1.Parameters["@a"].Value));
                // Console.WriteLine("FLAG");
                // Console.WriteLine(sqlCommand1.Parameters["@a"].Value);

                {
                    sqlCommand1.Parameters["@fiocl"].Value = fIOComboBox.Text;
                    sqlCommand1.Parameters["@fioemp"].Value = comboBox1.Text;
                    sqlCommand1.Parameters["@date"].Value = dateTHDateTimePicker.Text;
                    sqlCommand1.Parameters["@claim"].Value = claimTextBox.Text;
                    sqlCommand1.Parameters["@rec"].Value = recTextBox.Text;
                    sqlConnection1.Open();
                    sqlCommand1.ExecuteNonQuery();
                    sqlConnection1.Close();
                    if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 1)
                        MessageBox.Show("Запись уже была добавлена!", "Добавление записи
невозможно", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    else if (Convert.ToInt32(sqlCommand1.Parameters["@a"].Value) == 2)
                        MessageBox.Show("Ограничение по дате!", "Введите дату прошедшего
лечения", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    else
                        MessageBox.Show("Запись добавлена!", "Добавление записи",
                            MessageBoxButtons.OK, MessageBoxIcon.Information);
                }
            }
        }
    }
}

```

```

        catch
        {
            MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
        }
    }
}

```

### 13. Форма «Список сотрудников»

```

public partial class Form23 : Form
{
    public Form23()
    {
        InitializeComponent();
    }

    private void Form23_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Employee". При необходимости она может быть перемещена или удалена.
        this.employeeTableAdapter.Fill(this.dentistryDataSet.Employee);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_5". При необходимости она может быть перемещена или удалена.
        this.view_5TableAdapter.Fill(this.dentistryDataSet.View_5);
    }

    private void button5_Click(object sender, EventArgs e)
    {
        view_5BindingSource.Filter = "";
    }

    private void button1_Click(object sender, EventArgs e)
    {
        view_5BindingSource.Filter = "FIO='" + fIOComboBox.Text + "'";
    }
}

```

### 14. Форма «Материалы»

```

public partial class Form25 : Form
{
    public Form25()
    {
        InitializeComponent();
    }

    private void Form25_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_7". При необходимости она может быть перемещена или удалена.
        this.view_7TableAdapter.Fill(this.dentistryDataSet.View_7);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Material". При необходимости она может быть перемещена или удалена.
        this.materialTableAdapter.Fill(this.dentistryDataSet.Material);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {

```

```

        view_7BindingSource.Filter = "NameMat='" + nameMatComboBox.Text + "'";
    }
    catch
    {
        MessageBox.Show("Ошибка. Повторите ввод");
    }
}

private void button5_Click(object sender, EventArgs e)
{
    view_7BindingSource.Filter = "";
}

private void button2_Click(object sender, EventArgs e)
{
    Form form18 = new Form18();
    form18.Show();
}

private void button3_Click(object sender, EventArgs e)
{
    Form form26 = new Form26();
    form26.Show();
}

private void button6_Click(object sender, EventArgs e)
{
    view_7BindingSource.DataSource = view_7TableAdapter.GetData();
}
}

```

## 15. Форма «Услуги лечения»

```

public partial class Form26 : Form
{
    public Form26()
    {
        InitializeComponent();
    }

    private void servThrBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.servThrBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
    }

    private void Form26_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_11". При необходимости она может быть перемещена или удалена.
        this.view_11TableAdapter.Fill(this.dentistryDataSet.View_11);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Treatment". При необходимости она может быть перемещена или удалена.
        this.treatmentTableAdapter.Fill(this.dentistryDataSet.Treatment);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_8". При необходимости она может быть перемещена или удалена.
        this.view_8TableAdapter.Fill(this.dentistryDataSet.View_8);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.ServThr". При необходимости она может быть перемещена или удалена.
        this.servThrTableAdapter.Fill(this.dentistryDataSet.ServThr);
    }

    private void button1_Click(object sender, EventArgs e)

```

```

    {
        try
        {
            view_8BindingSource.Filter = "Id_Thr='" + id_ThrComboBox.Text + "'";
        }
        catch
        {
            MessageBox.Show("Ошибка. Повторите ввод");
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        view_8BindingSource.Filter = "";
    }
}

```

## 16. Форма «Услуги лечения»

```

public partial class Form27 : Form
{
    public Form27()
    {
        InitializeComponent();
    }

    private void Form27_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_11". При необходимости она может быть перемещена или удалена.
        this.view_11TableAdapter.Fill(this.dentistryDataSet.View_11);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Treatment". При необходимости она может быть перемещена или удалена.
        this.treatmentTableAdapter.Fill(this.dentistryDataSet.Treatment);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_8". При необходимости она может быть перемещена или удалена.
        this.view_8TableAdapter.Fill(this.dentistryDataSet.View_8);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            view_8BindingSource.Filter = "Id_Thr='" + id_ThrComboBox.Text + "'";
        }
        catch
        {
            MessageBox.Show("Ошибка. Повторите ввод");
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        view_8BindingSource.Filter = "";
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form form28 = new Form28();
        form28.Show();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        view_8BindingSource.DataSource = view_8TableAdapter.GetData();
    }
}

```

```

    }
}

```

## 17. Форма «Добавить услуги лечения»

```

public partial class Form28 : Form
{
    public Form28()
    {
        InitializeComponent();

        private void serviceBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.serviceBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.dentistryDataSet);

        }

        private void Form28_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "dentistryDataSet.Treatment". При необходимости она может быть перемещена или удалена.
            this.treatmentTableAdapter.Fill(this.dentistryDataSet.Treatment);
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "dentistryDataSet.Service". При необходимости она может быть перемещена или удалена.
            this.serviceTableAdapter.Fill(this.dentistryDataSet.Service);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                sqlCommand1.Parameters["@name"].Value = nameServComboBox.Text;
                sqlCommand1.Parameters["@id"].Value = id_TreatComboBox.Text;
                sqlConnection1.Open();
                sqlCommand1.ExecuteNonQuery();
                sqlConnection1.Close();
                MessageBox.Show("Запись добавлена!", "Добавление записи",
                MessageBoxButtons.OK, MessageBoxIcon.Information);

            }
            catch
            {
                MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
                MessageBoxIcon.Error);
            }
        }
    }
}

```

## 18. Форма «Взаиморасчет»

```

public partial class Form30 : Form
{
    public Form30()
    {
        InitializeComponent();

        private void Form30_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу
            "dentistryDataSet.Client". При необходимости она может быть перемещена или удалена.

```



```

        this.clientTableAdapter.Fill(this.dentistryDataSet.Client);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_10". При необходимости она может быть перемещена или удалена.
        this.view_10TableAdapter.Fill(this.dentistryDataSet.View_10);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try
        {
            if (radioButton1.Checked)
                view_10BindingSource.Filter = "FIO='" + fIOComboBox.Text + "'";
            else
                view_10BindingSource.Filter = "DateTH ='" +
Convert.ToString(nameDayDateTimePicker.Value) + "'";
        }
        catch
        {
            MessageBox.Show("Повторите ввод", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        view_10BindingSource.Filter = "";
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form form24 = new Form24();
        form24.Show();
    }
}

```

## 19. Форма «Взаиморасчет»

```

public partial class Form24 : Form
{
    private Word.Application wordapp;
    private Word.Document worddocument;
    private Word.Paragraphs wordparagraphs;
    private Word.Paragraph wordparagraph;
    public Form24()
    {
        InitializeComponent();
    }

    private void Form24_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Treatment". При необходимости она может быть перемещена или удалена.
        this.treatmentTableAdapter.Fill(this.dentistryDataSet.Treatment);
    }

    private void treatmentBindingNavigatorSaveItem_Click(object sender, EventArgs e)
    {
        this.Validate();
        this.treatmentBindingSource.EndEdit();
        this.tableAdapterManager.UpdateAll(this.dentistryDataSet);
    }

    private void button1_Click(object sender, EventArgs e)

```

```

{
    try
    {
        wordapp = new Word.Application();
        wordapp.Visible = true;
        Object template = Type.Missing;
        Object newTemplate = false;
        Object documentType = Word.WdNewDocumentType.wdNewBlankDocument;
        Object visible = true;
        worddocument = wordapp.Documents.Add(
            ref template, ref newTemplate, ref documentType, ref visible);
        template = @"C:\a1.doc";
        Object name = "Отчет об оказанных услугах";
        wordparagraphs = worddocument.Paragraphs;
        //Будем работать с первым параграфом
        wordparagraph = (Word.Paragraph)wordparagraphs[1];

        object oMissing = System.Reflection.Missing.Value;
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        wordparagraph = (Word.Paragraph)wordparagraphs[7];
        wordparagraph.Range.Text = "Оказанные услуги: ";
        sqlCommand2.Parameters["@id"].Value = Convert.ToInt32(id_TreatComboBox.Text);
        sqlConnection2.Open();
        //wordparagraph = (Word.Paragraph)wordparagraphs[3];
        var temp = new DataTable();
        // выполнить табличную функцию и вернуть таблицу в объект Reader
        // заполнить таблицу temp данными из Reader
        temp.Load(sqlCommand2.ExecuteReader());

        Object start = Word.WdUnits.wdStory;
        Object end = Word.WdMovementType.wdMove;
        wordapp.Selection.EndKey(ref start, ref end);
        //Word.Range wordrange = worddocument.Range(ref start, ref end);
        Object defaultTableBehavior =
        Word.WdDefaultTableBehavior.wdWord9TableBehavior;
        Object autoFitBehavior = Word.WdAutoFitBehavior.wdAutoFitWindow;

        var t1 = temp.Rows.Cast<DataRow>().Select(r => r.ItemArray).ToArray();

        Word.Table wordtable1 = worddocument.Tables.Add(wordapp.Selection.Range,
        temp.Rows.Count, temp.Columns.Count, ref defaultTableBehavior, ref autoFitBehavior);

        for (int i = 1; i <= temp.Rows.Count; i++)
            for (int j = 1; j <= temp.Columns.Count; j++)
                worddocument.Tables[1].Cell(i, j).Range.Text = t1[i - 1][j -
1].ToString();

        sqlConnection2.Close();

        sqlCommand3.Parameters["@id"].Value = Convert.ToInt32(id_TreatComboBox.Text);
        sqlConnection3.Open();
        sqlCommand3.ExecuteNonQuery();
        sqlConnection3.Close();
        string a = Convert.ToString(sqlCommand3.Parameters["@date"].Value);

        a = a.Substring(0, a.Length - 7);
        wordparagraph = (Word.Paragraph)wordparagraphs[1];
        wordparagraph.Range.Text = "Счет на оплату от: " + a;
        wordparagraph = (Word.Paragraph)wordparagraphs[2];
    }
}

```

```

        wordparagraph.Range.Text = "Врач: " +
Convert.ToString(sqlCommand3.Parameters["@vr"].Value);
        wordparagraph = (Word.Paragraph)wordparagraphs[3];
        wordparagraph.Range.Text = "Клиент: " +
Convert.ToString(sqlCommand3.Parameters["@cl"].Value);
        sqlCommand1.Parameters["@id"].Value = Convert.ToInt32(id_TreatComboBox.Text);

        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        worddocument.Paragraphs.Add(ref oMissing);
        sqlCommand1.Open();
        sqlCommand1.ExecuteNonQuery();
        wordparagraph = (Word.Paragraph)wordparagraphs[11];
        wordparagraph.Range.Text = "
Итого: " + sqlCommand1.ExecuteScalar().ToString();
        wordparagraph = (Word.Paragraph)wordparagraphs[12];
        wordparagraph.Range.Text = "Клиент _____
_____";
        wordparagraph = (Word.Paragraph)wordparagraphs[13];
        wordparagraph.Range.Text = "_____ подпись
расшифровка";
        sqlCommand1.Close();
    }
    catch
    {
        MessageBox.Show("Невозможно произвести взаиморасчет", "Невозможно произвести
взаиморасчет", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}
}

```

## 20. Форма «История лечения»

```

public partial class Form21 : Form
{
    public Form21()
    {
        InitializeComponent();
    }

    private void Form21_Load(object sender, EventArgs e)
    {
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_12". При необходимости она может быть перемещена или удалена.
        this.view_12TableAdapter.Fill(this.dentistryDataSet.View_12);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.DentalFormula". При необходимости она может быть перемещена или удалена.
        this.dentalFormulaTableAdapter.Fill(this.dentistryDataSet.DentalFormula);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.Client". При необходимости она может быть перемещена или удалена.
        this.clientTableAdapter.Fill(this.dentistryDataSet.Client);
        // TODO: данная строка кода позволяет загрузить данные в таблицу
        "dentistryDataSet.View_4". При необходимости она может быть перемещена или удалена.
        this.view_4TableAdapter.Fill(this.dentistryDataSet.View_4);
    }

    private void button1_Click(object sender, EventArgs e)
    {
        try

```

```

        {
            view_12BindingSource.Filter = "FIO Like'" + fIOComboBox.Text + "%'";
        }
        catch
        {
            MessageBox.Show("Ошибка. Повторите ввод");
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        view_12BindingSource.Filter = "";
    }

    private void button2_Click(object sender, EventArgs e)
    {
        Form form19 = new Form19();
        form19.Show();
    }

    private void button3_Click(object sender, EventArgs e)
    {
        Form form22 = new Form22();
        form22.Show();
    }

    private void button6_Click(object sender, EventArgs e)
    {
        view_12BindingSource.DataSource = view_12TableAdapter.GetData();
    }
}

```

## 21. Форма «Восстановление пароля»

```

public partial class Form31 : Form
{
    public Form31()
    {
        InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
        sqlCommand1.Parameters["@log"].Value = textBox1.Text;
        sqlConnection1.Open();
        sqlCommand1.ExecuteNonQuery();
        sqlConnection1.Close();
        int result = (int)sqlCommand1.Parameters["@pass"].Value;
        textBox2.Text = Convert.ToString(result);
    }
}

```