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

Институт
информационных технологий

Кафедра
информационных систем

Отчет по курсовой работе

по дисциплине «**Управление данными**»

на тему: **Проектирование БД ветеринарной клиники**

Студент **Румянцев М. С.**
группы ИДБ-22-07

_____
подпись

Руководитель **Быстрикова В.А.**
старший преподаватель

_____
подпись

Оглавление

ВВЕДЕНИЕ 3

ГЛАВА 1. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ 4

1.1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ 4

1.2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ 17

1.3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 22

1.4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 31

ГЛАВА 2. ОПИСАНИЕ ФУНКЦИОНИРОВАНИЯ БД 43

2.1. НАЗНАЧЕНИЕ И ПЕРЕЧЕНЬ ФУНКЦИЙ БД 43

2.2. ОПИСАНИЕ РАБОТЫ С ПРИЛОЖЕНИЕМ 44

ЗАКЛЮЧЕНИЕ 90

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 92

ПРИЛОЖЕНИЕ А. ЗАПОЛНЕНИЕ ТАБЛИЦ ДАННЫМИ 93

ПРИЛОЖЕНИЕ Б. SQL-ОПЕРАТОРЫ СОЗДАНИЯ ОСНОВНЫХ ОБЪЕКТОВ БД 98

ПРИЛОЖЕНИЕ В. КОД ПРОГРАММЫ 127

ВВЕДЕНИЕ

Исследовательский фокус направлен на работу ветеринарной клиники, где ключевой сферой является предоставление медицинских услуг для животных, включая диагностику, лечение и послеоперационный уход. Ключевые фигуры в этой сфере – это ветеринарный врач, генеральный директор, администратор и менеджер по запасам. Важность детального учета, эффективного управления и проведения анализа всех процессов в этой сфере невозможно переоценить.

Основная задача этого проекта заключается в разработке инновационной базы данных, специально предназначенной для автоматизации всех внутренних процессов, используемых в работе ветеринарной клиники. Создание такой базы данных позволит значительно повысить эффективность работы клиники, упрощая все основные операции и делая их более быстрыми.

Для успешной реализации проекта необходимо провести анализ сферы деятельности и выявить ключевые функции ветеринарной клиники. Затем необходимо приступить к логическому проектированию, где составляется ER-диаграмма для наглядного отображения сущностей и их взаимосвязей. Физическое проектирование включает в себя реализацию разработанной базы данных на выбранной системе управления базами данных с учетом всех требований. Завершающим этапом будет разработка пользовательского интерфейса для взаимодействия с базой данных.

Таким образом, целью проекта является разработка и внедрение базы данных для ветеринарной клиники, которая обеспечит упрощение учета, контроль всех этапов процессов и повышение эффективности работы организации.

ГЛАВА 1. ПРОЕКТИРОВАНИЕ СТРУКТУРЫ БАЗЫ ДАННЫХ

1.1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Предметную область можно определить, как сферу человеческой деятельности, выделенную и описанную согласно установленным критериям [1]. В описываемое понятие должны входить сведения об ее элементах, явлениях, отношениях и процессах, отражающих различные аспекты этой деятельности [1].

Одна из первых задач, с решением которых сталкивается разработчик программной системы – это изучение, осмысление и анализ предметной области. Дело в том, что предметная область сильно влияет на все аспекты проекта: требования к системе, взаимодействие с пользователем, модель хранения данных, реализацию и т.д.

Анализ предметной области позволяет выделить ее сущности, определить первоначальные требования к функциональности и определить границы проекта.

В данной работе в качестве предметной области рассматривается деятельность ветеринарной клиники.

Ветеринарная клиника в настоящее время сталкивается с значительным объемом информации о пациентах, что делает процессы управления данными более сложными. Учет медицинской истории, регистрация клиентов, запись результатов обследований, анализы и лечение животных - все это требует систематизации и эффективной обработки данных. Для успешного функционирования ветеринарной клиники важно иметь систему управления информацией, которая будет обеспечивать надежное хранение данных о пациентах, обеспечивать доступность информации для врачей и персонала, а также обеспечивать соблюдение конфиденциальности и безопасности данных. Разработка современной информационной системы, специально адаптированной для потребностей ветеринарной медицины, позволит

эффективно управлять данными и повысить уровень обслуживания пациентов.

Основные задачи, решаемые в предметной области "ветеринарная клиника", включают в себя:

- Учет медицинской истории пациента.
- Оказание услуг по осмотру и лечению животных.
- Регистрация клиентов и пациентов.
- Планирование и управление временем, включающее в себя составление расписания приемов, операций, прививок и т.д.
- Хранение и управление медицинскими данными, включающими в себя результаты анализов, врачебные выписки и др.
- Обеспечение конфиденциальности данных пациентов клиники и их владельцев.

В области ветеринарной медицины существует ряд программных продуктов, решающих различные задачи:

- Учет медицинской истории и клиентской базы: программное обеспечение, такое как “AVImark”, “ezyVet” и “VetBlue”, позволяет вести электронные медицинские карты пациентов, регистрировать клиентов и управлять данными о животных.
- Планирование и управление временем: ветеринарные клиники часто используют программные продукты для управления расписанием приемов и процедур, такие как “VetSpace”, “VetOffice” и “Vetter Software”.

- Управление лекарствами и инвентарем: системы учета склада и инвентаря, включая “Hippo Manager”, “eVetPractice” и “VetBadger”, помогают отслеживать запасы лекарств и медицинского оборудования.
- Финансовый учет: программы учета финансовых операций, такие как “VetBlue”, “Cornerstone” и “VetLinkSQL”, предоставляют возможность ведения учета счетов, оплаты услуг и управления финансами клиники.
- Обеспечение конфиденциальности данных: средства управления доступом и защиты данных, в том числе встроенные в медицинские программные продукты механизмы шифрования и аутентификации.

В области ветеринарной медицины существует несколько популярных программных продуктов, сайтов и интернет-сервисов:

1. “IDEXX Neo”: Это программное обеспечение для управления ветеринарной клиникой, которое включает в себя учет медицинской истории, планирование и управление временем, а также финансовый учет.

2. “eVetPractice”: Платформа для управления ветеринарной клиникой, включающая учет медицинской истории, клиентскую базу, планирование и управление временем.

3. “VetBlue”: Программное обеспечение для управления клинической практикой, включающее в себя учет медицинской истории, управление лекарствами и финансовый учет.

Кроме того, существуют онлайн-сервисы и интернет-платформы, такие как “VetConnectPlus” и “Vetter Software”, которые предлагают широкий спектр функций для управления ветеринарной клиникой, включая учет медицинской истории, планирование приемов, управление лекарствами и финансовый учет.

Среди отдельных конфигураций можно выделить: Конфигурацию "Ветеринарная клиника" от компании “Простой софт”.

Сервисы управления ветеринарной клиникой предназначены для оптимизации работы клиники, повышения эффективности и удобства как для врачей, так и для владельцев животных. Они обеспечивают автоматизацию различных процессов, таких как запись на прием, управление медицинскими картами животных, учет товаров и услуг, а также анализ данных для улучшения качества обслуживания.

Преимущества использования сервисов управления ветеринарной клиникой: состоят в повышении эффективности работы клиники; автоматизации рутинных процессов, что снижает нагрузку на персонал и позволяет сосредоточиться на оказании качественной помощи животным.

В качестве первого программного продукта рассмотрим БИТ.Айболит, программу для управления ветеринарной клиникой, созданную на базе 1С (https://www.1cbit.ru/1csoft/bit-aybolit) [2]. БИТ.Айболит – решение для комплексной автоматизации ветеринарных клиник, а также сетей ветклиник в едином информационном пространстве. Программа позволяет автоматизировать работу всех отделений клиники: от записи питомцев на прием, до шаблонов приема врача, а также настройки уведомлений о плановой вакцинации животных.

Данное ПО предоставляет следующие возможности:

- Заполнение регистрационных записей, записей на прием (рис. 1.1).
- Составление занятости врачей, исходя из записей на прием (рис. 1.1).
- Взаиморасчеты с клиентами (рис. 1.1).
- Создание единой регистратуры сети клиник (рис. 1.1).
- Добавление дополнительного владельца в карточку животного, или же дополнительного животного в карточку владельца (рис. 1.2).

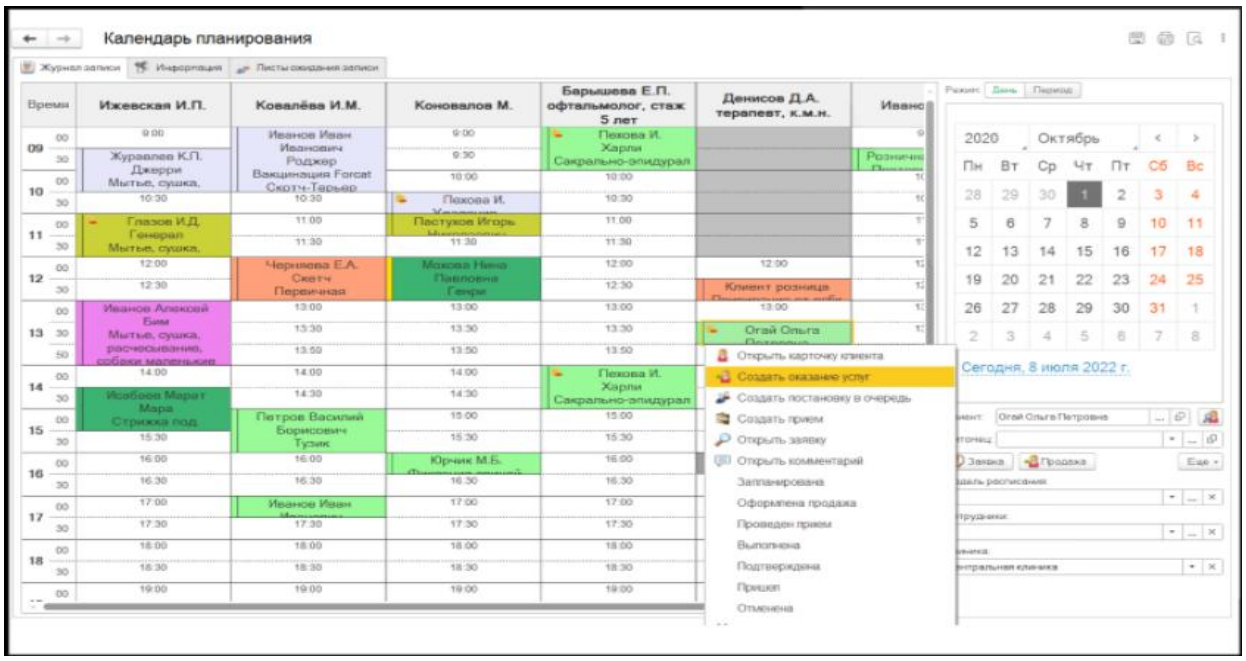

Рис. 1.1. Регистратура. Эффективная загрузка ветеринарной клиники

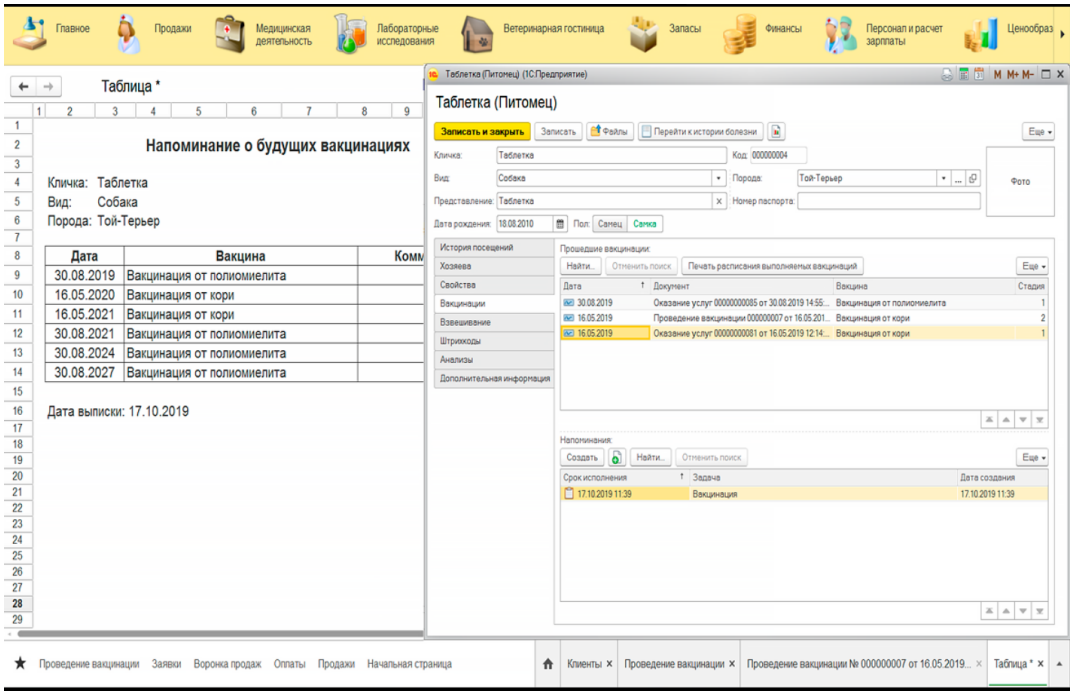

Рис. 1.2. Карточка пациента

- Автоматизация call-центра за счет совершения вызовов прямо из программы, записи разговоров и контроля специалистов, возможности доступа информации о клиенте и истории его посещений во время его звонка, ускоренного подбора времени записи, врача, услуги (рис. 1.3). Есть функция обратного звонка и очередь ожидания.

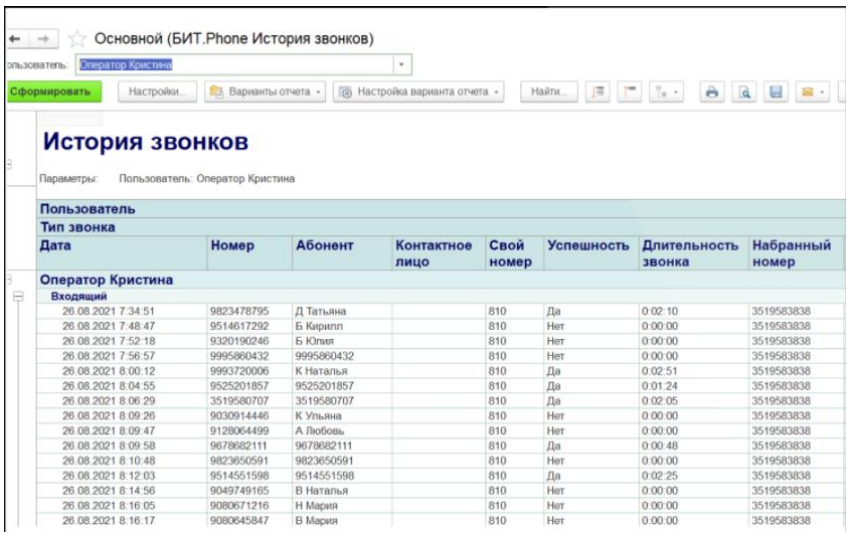

Рис. 1.3. Автоматизация call-центра

- Создание списков обзвона и отправки уведомлений.
- ПО позволяет сегментировать клиентов по определенным характеристикам, создавать системы лояльности и бонусные системы для клиентов, а также использования различных схем ценообразования.
- Просмотр истории посещений (рис. 1.4) и последующего анализа данных по посещениям клиента.

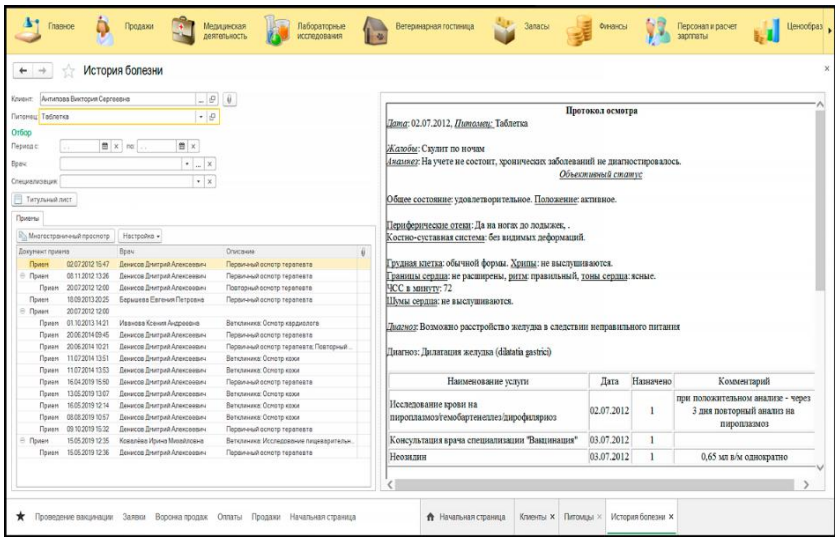

Рис. 1.4. История посещений клиента

- Наличие готовых шаблонов приема врача (рис. 1.5) и рецептов, настраиваемые стандарты лечения питомцев, электронная история болезни и быстрый доступ к результатам исследований.

- Фиксация медицинских показателей и отслеживание их в динамике, а также автоматическое напоминание клиентам о вакцинации путем уведомлений.

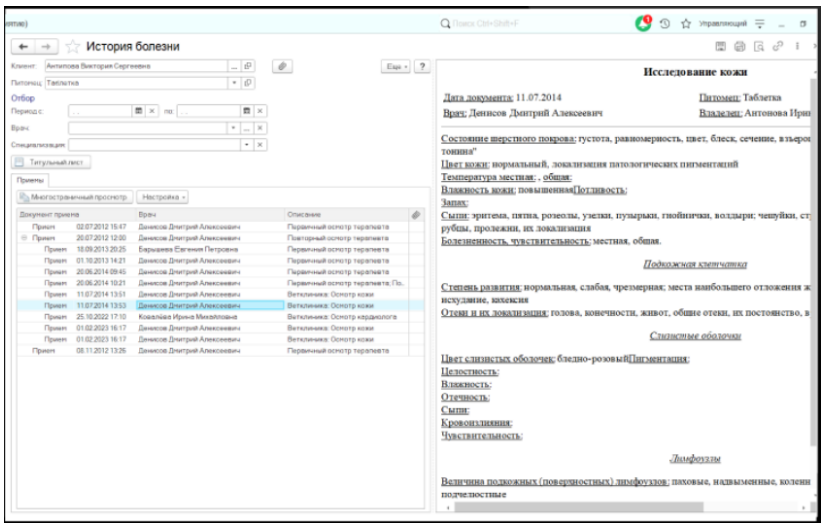

Рис. 1.5. Пример готового шаблона исследования

- Работа со стационаром для животных, оформляя поступления, перемещения, выписку животных. Есть поля для привязки договора, заключенного с владельцем животного. Помогает в планировании занятости стационара, расчете стоимости пребывания животного. Хранит назначения на процедуры и обследования животного. Все вышеперечисленное представлено на рис. 1.6.

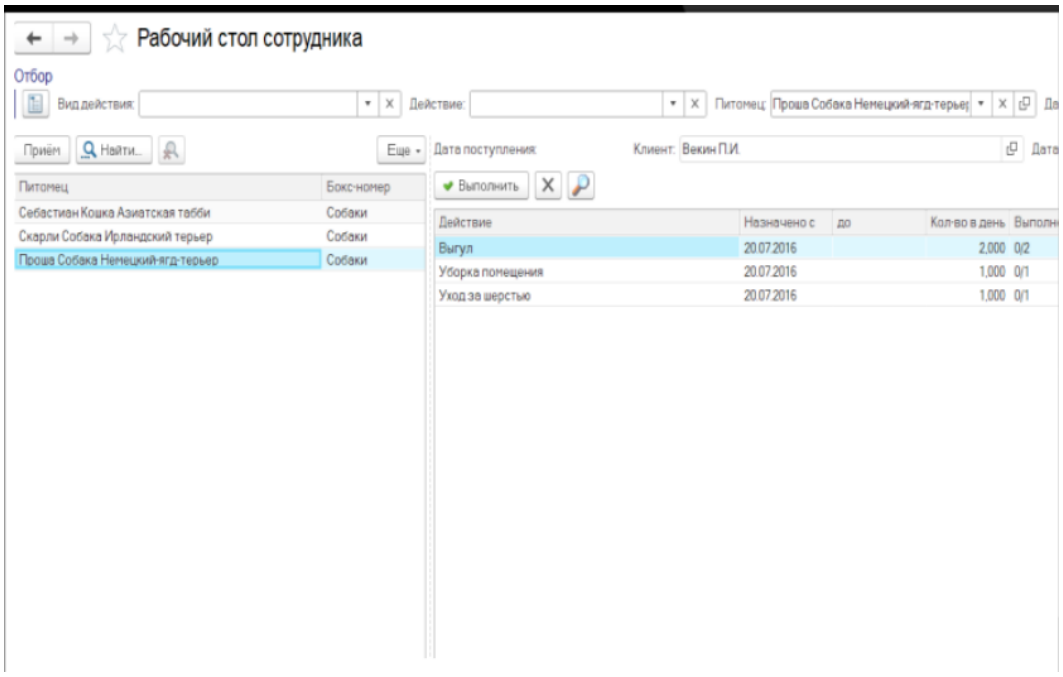

Рис. 1.6. Стационар

- Автоматическое заполнение графиков работы сотрудников, расчета зарплаты по индивидуальным схемам.
- Анализ популярности сотрудников для принятия эффективных управленческих решений.
- Контроль расхода материалов с подсчетом отклонения от постановленных норм (рис. 1.7), мониторинг остатков материалов для их своевременного пополнения запасов.
- Автоматическое списание расходных материалов в соответствии с нормой.

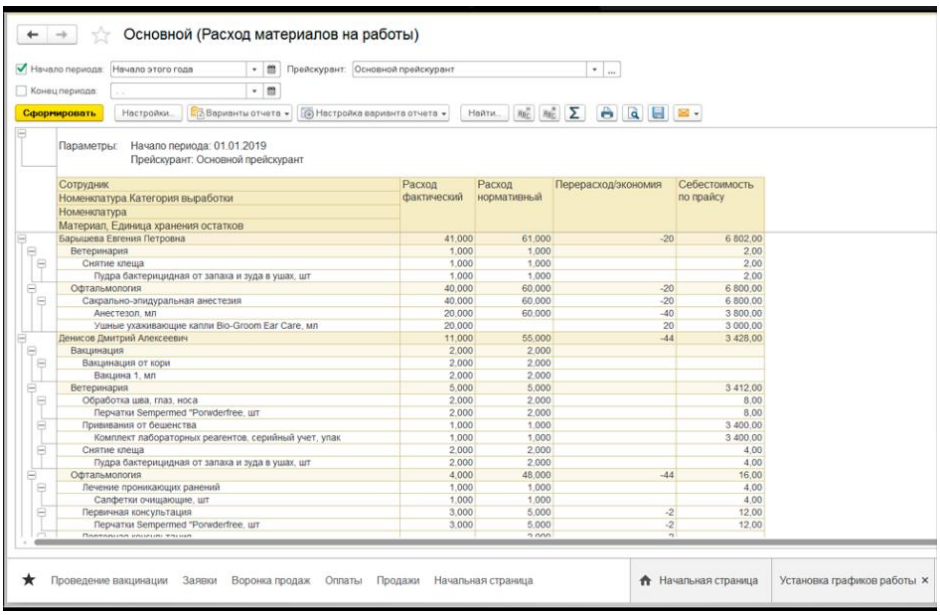

Рис. 1.7. Складской учет

В качестве второго программного продукта рассмотрим конфигурацию “Ветеринарная клиника” от компании “Простой софт” (https://prostoysoft.ru/Veterinary.htm) [3]. Программный продукт “Ветеринарная клиника” от компании “Простой софт” предназначен для автоматизации работы ветеринарных клиник.

- Учет клиентов с хранением их контактной информации. По каждому клиенту возможно посмотреть полную информацию о его питомцах-пациентах ветеринарной клиники (рис. 1.8).

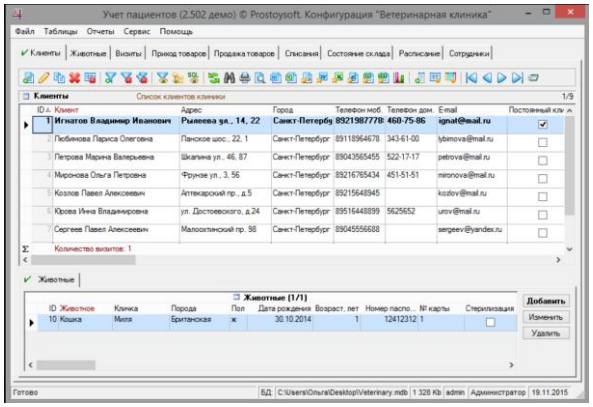

Рис. 1.8. Учет клиентов-владельцев животных

- Ведение списка животных-пациентов с полной информацией по ним. По каждому животному можно просмотреть историю его лечения, обследования и поставленные диагнозы (рис. 1.9).

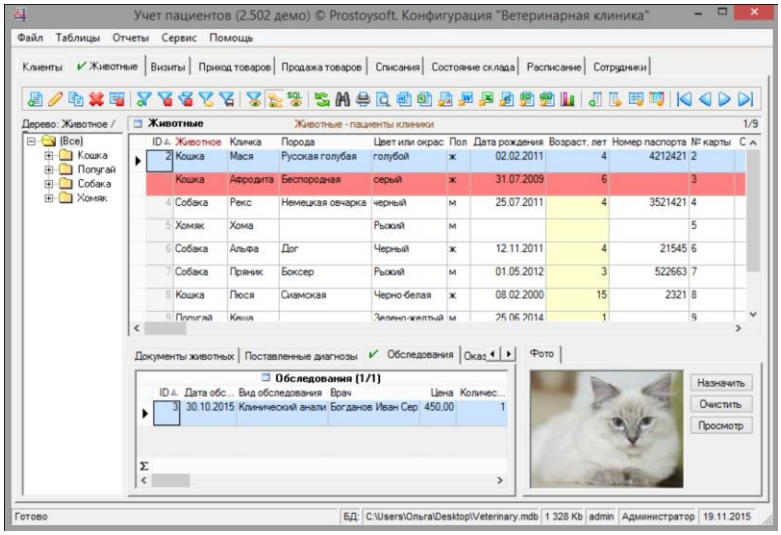

Рис. 1.9. Учет животных-пациентов

- Фиксация регистрации посещений, оказанных услуг, обследований и назначений по дальнейшему лечению животных-пациентов ветеринарной клиники (рис. 1.10).

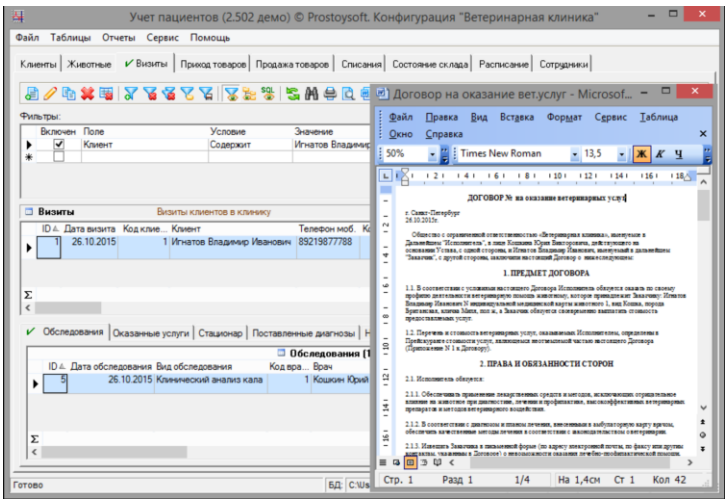

Рис. 1.10. Учет визитов и оказания услуг

- Учет основных складских операций – поступление и продажа товаров, списание расходных средств (рис. 1.11).

- Учет остатков на складе (рис. 1.12).

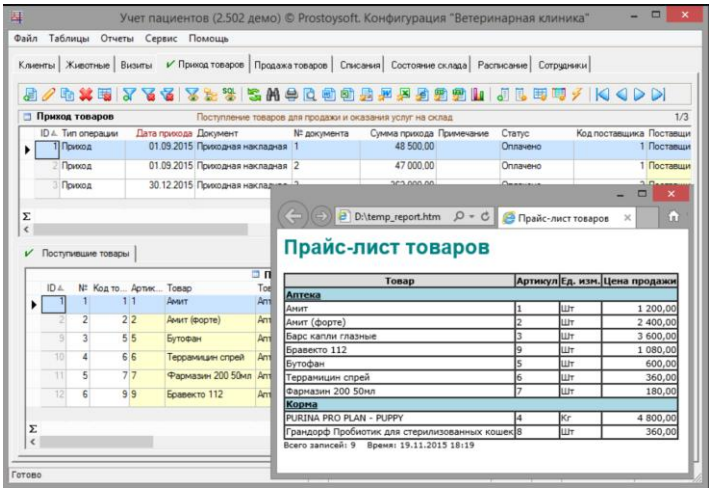

Рис. 1.11. Поступление товара на склад

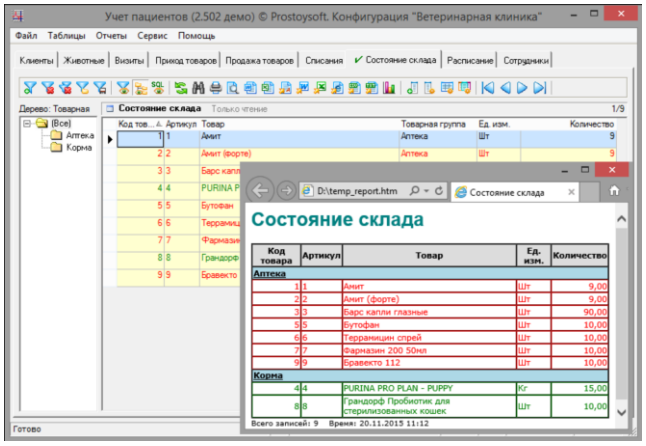

Рис. 1.12. Состояние склада

- Ведение расписания дежурств сотрудников ветеринарной клиники, предварительная запись клиентов на прием(рис. 1.13).

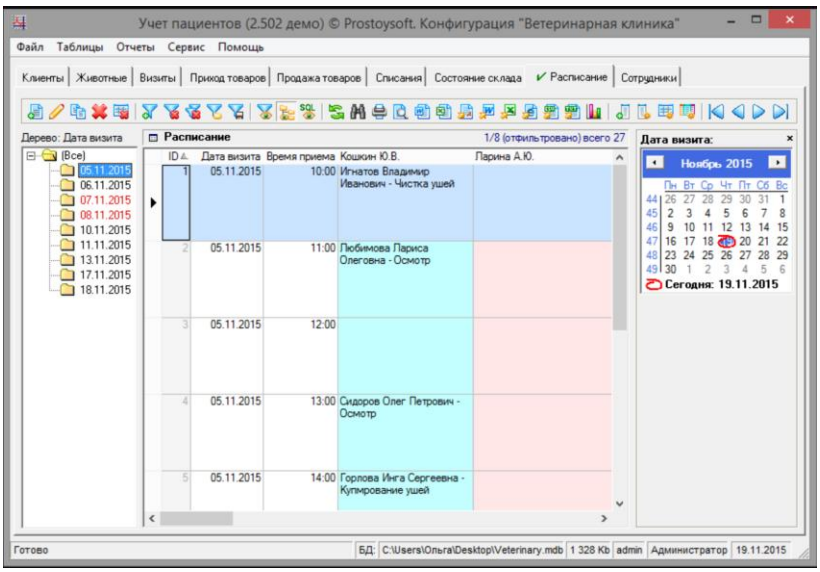

Рис. 1.13. Расписание работы персонала

- Формирование отчетов по оказанным услугам, ежедневным визитам пациентов в клинику состоянию склада и т.д. Возможность создавать новые отчеты.

Для всех рассмотренных программных продуктов можно выделить общие функции:

- учет клиентов-владельцев животных;
- учет животных-пациентов клиники;
- учет посещений ветеринарной клиники;
- складской учет;
- составление расписания работы персонала;
- формирование отчетов.

Особенностью программного продукта БИТ.Айболит является наличие возможности автоматизации call-центра, а также наличие CRM-системы, которая существенно упрощает анализ. Кроме того, отдельно можно выделить отдельный раздел, посвященный стационару для животных

В конфигурации “Ветеринарная клиника” от компании “Простой софт” можно выделить более простой и интуитивно понятный дизайн, что также

позволит предприятию сэкономить бюджет на мощностях устройств, предназначенных для работы с данной конфигурацией.

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

- учет клиентов-владельцев животных;
- учет животных-пациентов клиники;
- учет посещений ветеринарной клиники;
- складской учет;
- составление расписания работы персонала;
- учет персонала ветеринарной клиники;
- учет оказанных услуг.

1.2. КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Весь процесс проектирования базы данных (БД) состоит из трех основных фаз: концептуального, логического и физического проектирования [4].

Концептуальное проектирование – это формализованное описание предметной области, которое должно быть таким, чтобы, с одной стороны, можно было проанализировать правильность разрабатываемого проекта БД, а с другой не должна быть привязана к конкретной СУБД [4].

Цель этапа концептуализации - создать концептуальную модель данных, основанную на представлениях пользователей о тематической области [4]. Результатом этого этапа проектирования является определение содержимого базы данных – задач, которые она должна выполнить, и объектов, которые она должна хранить.

Концептуальное проектирование использует диаграмму вариантов использования, которая описывает, какой функционал разрабатываемой программной системы доступен каждой группе пользователей.

Назначение диаграммы вариантов использования заключается в формализации функциональных требований к системе с помощью действующих лиц (потенциальных пользователей системы) и вариантов использования (задач, которые позволяют действующему лицу получить доступ к разрабатываемому продукту) [5].

Последовательность составления диаграммы вариантов использования:

1. Выделить действующие лица.
2. Для каждого действующего лица выделить варианты использования. После этого следует составить общий список вариантов использования.
3. Связать между собой действующие лица и варианты использования.

4. Проанализировать в первом приближении наличие общих подзадач и разных вариантов использования, при наличии выделить их и установить отношения.

Проведем анализ задач базы данных ветеринарной клиники.

Генеральный директор использует базу данных для анализа общей эффективности клиники, отслеживания ключевых показателей успеха и поддержания информированности о финансовом состоянии и операционной деятельности. Он ведет учет услуг, изменяет расписание работы персонала и просматривает складской учет для оптимизации работы клиники и повышения прибыли.

Администратор клиники использует базу данных для управления расписанием сотрудников, заполнения клиентской базы и для обеспечения своевременного выполнения всех административных задач. Он ведет учет сотрудников, клиентов-владельцев животных и формирует расписание работы персонала для обеспечения качественного обслуживания и комфорта для пациентов и клиентов.

Менеджер по запасам использует базу данных для отслеживания запасов, анализа потребностей в материалах и оборудовании и для управления закупками. Он ведет складской учет, пополняет и списывает медикаменты со склада для обеспечения бесперебойного функционирования клиники и оптимального использования ресурсов.

Ветеринар использует базу данных для заполнения данных животных-пациентов, учета посещений ветеринарной клиники и создания записей о предоставленных услугах.

С проектируемой системой будут взаимодействовать следующие действующие лица:

- Генеральный директор;
- Администратор клиники;
- Менеджер по запасам;

- Ветеринар.

Рассмотрим варианты использования функций, которые имеют данные действующие лица.

Генеральный директор выполняет следующие функции:

- Добавление услуг;
- Изменение расписания работы персонала;
- Просмотр складского учета.

Администратор клиники выполняет следующие функции:

- Добавление данных клиентов-владельцев животных;
- Обновление данных клиентов-владельцев животных;
- Добавление данных сотрудников;
- Обновление данных сотрудников;
- Формирование расписания работы персонала.

Менеджер по запасам выполняет следующие функции:

- Ведение складского учета;
- Пополнение и списание медикаментов со склада.

Ветеринар выполняет следующие функции:

- Заполнение медкнижки животного-пациента
- Учет посещений ветеринарной клиники
- Оказание услуг на приеме;
- Удаление неоказанных услуг.

Диаграмма вариантов использования проиллюстрирована на рис. 1.14.

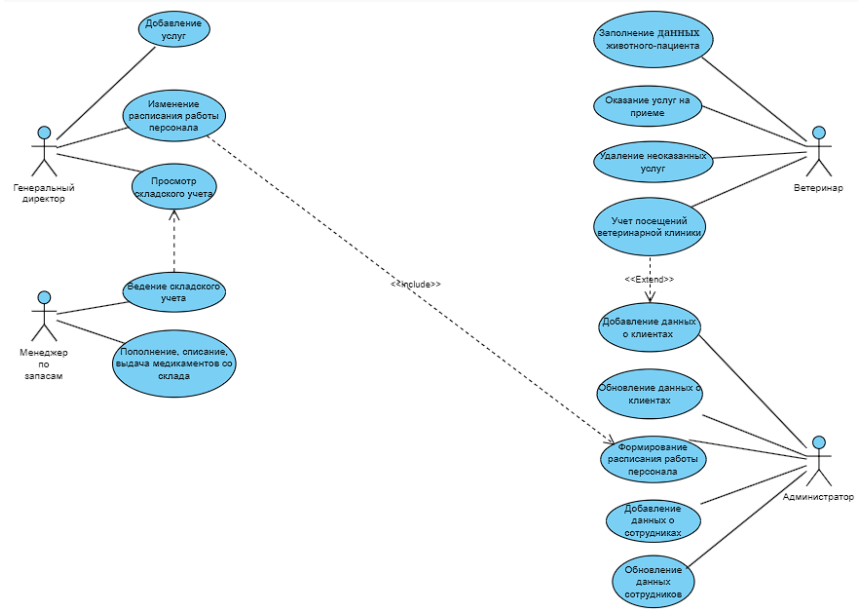

Рис. 1.14 Диаграмма вариантов использования

В первом приближении для решения выделенных задач необходимо хранение данных о следующих объектах:

1. Клиент – содержит информацию о каждом клиенте (уникальный идентификатор, ФИО, контактный телефон, адрес электронной почты, адрес).
2. Сотрудник – содержит информацию о каждом сотруднике (уникальный идентификатор, ФИО, должность, заработная плата, электронная почта, телефон).
3. Питомец – содержит информацию о животном (уникальный идентификатор, кличка, порода, пол).
4. Прием – содержит информацию о посещении ветеринарной клиники (уникальный идентификатор, дата и время, уникальный идентификатор сотрудника, уникальный идентификатор клиента, уникальный идентификатор питомца, уникальный идентификатор услуги).

5. Услуга – содержит информацию о предоставляемых клиникой услугах (уникальный идентификатор, название услуги).

6. Складская операция – содержит информацию об операциях, связанных с завозом/выдачей медикаментов, хранимых на складе (уникальный идентификатор, дата и время, уникальный идентификатор медикамента, операция, количество единиц, единицы измерения)

7. Медикамент – содержит информацию о медикаментах, хранящихся на складе (уникальный идентификатор, наименование, стоимость, количество на складе, количество заказанных, количество выданных за месяц)

8. График – содержит информацию о графике работы сотрудников (уникальный идентификатор, понедельник, вторник, среда, четверг, пятница, суббота, воскресение)

Данные можно классифицировать по частоте их изменения:

- условно-постоянные данные (например, данные о клиентах и сотрудниках);
- данные, которые оперативно обновляются при каждом решении задачи (например, данные об оказанных услугах и состоянии склада).

1.3. ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Логическое проектирование базы данных — процесс, при котором создается структура данных, независимая от конкретного СУБД, и представляющая собой перевод концептуальной модели данных в схему базы данных. На этом этапе определяются таблицы, ключи, индексы, ограничения целостности и другие структурные элементы, необходимые для обеспечения целостности и эффективности хранения и обработки данных.

ER-диаграмма (сущность-связь диаграмма) - это графический инструмент, используемый для семантического моделирования предметной области. Он помогает описать структуру данных в информационной системе, идентифицируя сущности (объекты или понятия) и их взаимосвязи [6].

Вот основные элементы ER-диаграммы и их роли в семантическом моделировании:

1. Сущности представляют объекты или понятия в предметной области. Каждая сущность имеет атрибуты, которые описывают её характеристики.

2. Атрибуты представляют конкретные характеристики сущностей. Они используются для описания свойств или параметров сущностей.

3. Связи определяют отношения между различными сущностями. Они указывают на то, как сущности взаимодействуют друг с другом.

4. Кардинальность определяет количество экземпляров одной сущности, которые могут быть связаны с определенным экземпляром другой сущности.

ER-диаграмма помогает визуализировать структуру данных и взаимосвязи между ними, что делает её полезным инструментом для анализа, проектирования и коммуникации в процессе разработки информационных систем.

В процессе проектирования баз данных сущности играют ключевую роль, связывая концептуальный и логический этапы. На концептуальном этапе проектировщики анализируют предметную область и выделяют сущности, их атрибуты и взаимосвязи, создавая абстрактную модель. Затем

на логическом этапе эта модель преобразуется в структуры данных, где сущности становятся основой для таблиц, а их атрибуты - для столбцов таблиц. Таким образом, сущности обеспечивают непрерывность между концептуальным представлением предметной области и конкретными структурами данных в базе данных.

Построим ER-диаграммы для выбранных сущностей базы данных ветеринарной клиники.

Рис. 1.15. Связь «Клиент владеет питомцем»

Связь, представленная на рисунке, 1.15 имеет тип 1:M, так как у клиента может быть несколько питомцев, а у питомца может только один хозяин (клиент). Сущность Питомец имеет обязательный класс принадлежности, так как у питомца обязательно должен быть хозяин (клиент). Сущность Клиент имеет обязательный класс принадлежности, так как у всех клиентов есть питомец.

Рис. 1.16. Связь «Сотрудник ведет прием»

Связь, представленная на рисунке 1.16, имеет тип 1:M, так как у сотрудника может быть несколько приемов, а на приеме может присутствовать только один сотрудник. Сущность Сотрудник имеет необязательный класс принадлежности, так как у сотрудника необязательно должен быть прием. Сущность Прием имеет обязательный класс принадлежности, так как на всех приемах присутствует сотрудник.

Рис. 1.17. Связь «Питомец посетил прием»

Связь, представленная на рисунке 1.17, имеет тип 1:M, так как у питомца может быть несколько приемов, а на приеме может присутствовать только один питомец. Сущность Питомец имеет обязательный класс принадлежности, так как у питомца обязательно должен быть прием. Сущность Прием имеет обязательный класс принадлежности, так как на всех приемах присутствует питомец.

Рис. 1.18. Связь «График управляет сотрудником»

Связь, представленная на рисунке 1.18, имеет тип 1:M, так как график может совпадать у нескольких сотрудников, а у сотрудника только один график работы. Сущность График имеет обязательный класс принадлежности, так как в графике обязательно присутствует сотрудник. Сущность Сотрудник имеет обязательный класс принадлежности, так как у всех сотрудников есть график.

Рис. 1.19. Связь «Услуга оказывается на приеме»

Связь, представленная на рисунке 1.19, имеет тип M:M, так как одна услуга может быть оказана на нескольких приемах, а на приеме может быть оказано несколько услуг. Сущность Услуга имеет необязательный класс принадлежности, так как услуга может быть и не оказана на приеме.

Сущность Прием имеет обязательный класс принадлежности, так как на каждом приеме оказана хотя бы одна услуга.

Рис. 1.20. Связь «Медикамент участвует в складской операции»

Связь, представленная на рисунке 1.20, имеет тип 1:M, так как один медикамент может участвовать в нескольких складских операциях, а складская операция производится только по одному медикаменту. Сущность Медикамент имеет обязательный класс принадлежности, так как медикамент обязательно должен быть задействован в складской операции. Сущность Складская операция имеет обязательный класс принадлежности, так как складская операция обязательно производится по какому-то из медикаментов.

Рис. 1.21. Связь «Сотрудник производит складскую операцию»

Связь, представленная на рисунке 1.21, имеет тип 1:M, так как один сотрудник производит нескольких складских операций, а складская операция производится только одним сотрудником. Сущность Сотрудник имеет необязательный класс принадлежности, так как сотрудник необязательно производит складские операции. Сущность Складская операция имеет обязательный класс принадлежности, так как складская операция обязательно производится сотрудником.

Рис. 1.22. Связь «Прием требует складской операции»

Связь, представленная на рисунке 1.22, имеет тип 1:M, так как складская операция относится только к одному приему, а для одного приема может потребоваться несколько складских операций. Сущность Складская операция имеет обязательный класс принадлежности, так как складская операция может быть связана с приемом. Сущность Прием имеет необязательный класс принадлежности, так как прием не всегда требует проведения складской операции.

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

1. Связь «Клиент владеет питомцем» удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

- Клиент (Id Клиента…)
- Питомец (Id Питомца, Id Клиента…)

2. Связь «Сотрудник ведет прием» удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

- Сотрудник (Id Сотрудника…)
- Прием (Id Приема, Id Сотрудника…)

3. Связь «Питомец посетил прием» удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

- Питомец (Id Питомца…)
- Прием (Id Приема, Id Питомца…)

4. Связь «График управляет сотрудником» удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

- График (Id Графика…)
- Сотрудник (Id Сотрудника, Id Графика…)

5. Связь «Услуга оказывается на приеме» удовлетворяет условиям правила 6, в соответствии с которым получаем 3 отношения:

- Услуга (Id Услуги…)
- Прием (Id Приема…)
- УслугаПриема (Id Приема, Id Услуги)

6. Связь «Медикамент участвует в складской операции» удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

- Медикамент (Id Медикамента…)
- СкладскаяОперация (Id СкладскойОперации, Id Медикамента…)

7. Связь «Сотрудник производит складскую операцию» удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

- Сотрудник (Id Сотрудника…)
- СкладскаяОперация (Id СкладскойОперации, Id Сотрудника…)

8. Связь «Складская операция производится для приема» удовлетворяет условиям правила 4, в соответствии с которым получаем 2 отношения:

- Прием (Id Приема…)
- СкладскаяОперация (Id СкладскойОперации, Id Приема…)

Добавим не ключевые атрибуты в каждое из предварительных отношений с условием, чтобы отношения отвечали требованиям третьей нормальной формы.

- Клиент (Id Клиента, ФИО, контактный телефон, адрес электронной почты, адрес)

- Сотрудник (Id Сотрудника, ФИО, должность, дата рождения, паспортные данные, адрес электронной почты, телефон, Id Графика)
- Питомец (Id Питомца, Id Клиента, кличка, порода, окрас, пол, возраст)
- Прием (Id Приема, Id Сотрудника, Id Питомца, дата и время, комментарий)
- Услуга (Id Услуги, название услуги)
- СкладскаяОперация (Id СкладскойОперации, Id Сотрудника, Id Медикамента, Id Приема, дата и время, операция, количество единиц, единицы измерения)
- Медикамент (Id Медикамента, наименование, стоимость, срок годности, количество, единицы измерения)
- График (Id Графика, понедельник, вторник, среда, четверг, пятница, суббота, воскресенье)
- УслугаПриема (Id Приема, Id Услуги)

Ограничения предметной области, или бизнес-правила, представляют собой правила, ограничения или условия, которым должна соответствовать информационная система или бизнес-процессы в конкретной предметной области. Эти правила могут определяться как внешними (например, законодательством, стандартами отрасли), так и внутренними требованиями бизнеса (например, политики компании, процедуры, логика бизнес-процессов).

Ограничения предметной области могут включать в себя различные аспекты, такие как:

- Требования к данным: например, форматы данных, допустимые значения, правила валидации.
- Бизнес-правила: условия, которым должны соответствовать бизнес-процессы, например, правила автоматического принятия решений или согласования.
- Ограничения безопасности и доступа: права доступа к данным и функционалу системы.
- Условия работы системы: ограничения по времени работы, максимальные нагрузки и т.д.

Бизнес-правила и ограничения для предметной области ветеринарной клиники:

- Сотрудник должен быть старше 18 лет, иметь уникальный номер телефона и адрес электронной почты.
- Медикаменты не могут быть выданы, если их количество на складе меньше требуемого для выдачи.
- Прием сотрудника должен быть назначен в соответствии с его графиком работы.
- Запись на прием должна быть создана не позднее чем за 24 часа до назначенного времени.
- Выдача медикаментов должна сопровождаться фиксацией в базе данных с указанием даты и количества выданных препаратов.
- Дата приема не может быть установлена ранее текущей даты и времени.
- Количество медикаментов на складе не может быть отрицательным.

- Ограничения на время приема. Приемы не должны пересекаться для одного врача, чтобы избежать конфликтов в расписании.
- Данные клиента, такие как ФИО, номер телефона, адрес электронной почты, домашний адрес – должны быть уникальными.
- Форма задания отрезков рабочего времени сотрудника должна быть стандартизирована.
- Складские операции должны автоматически отражаться в базе данных медикаментов соответствующим увеличением или уменьшением количества того или иного препарата на складе.
- Можно удалить информацию только о том приеме, что еще не произошел.

1.4. ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Физическое проектирование базы данных — это процесс определения способа хранения и доступа к данным в базе данных на физическом уровне, который выполняется с учетом особенностей выбранной СУБД. Этот процесс включает в себя:

- Определение основных отношений и структур данных, наиболее подходящих для конкретной СУБД.
- Организацию файлов и индексов, которые обеспечивают эффективный доступ к данным, с учетом специфики физических носителей информации.
- Разработку механизмов для обеспечения ограничений целостности и средств защиты данных.
- Описание способа физической реализации логического проекта базы данных, оптимизированного для достижения высокой производительности и надежности.

Физическое проектирование базы данных направлено на преобразование логической модели в эффективную физическую структуру, с учетом всех технических и функциональных требований системы.

Для физического проектирования базы данных ветеринарной клиники была выбрана СУБД SQL Server в сочетании со средой разработки SQL Server Management Studio.

СУБД SQL Server используются для создания, размещения, хранения и управления реляционными базами данных на специальных серверах или в облаке. Основными возможностями, которые предоставляет данная СУБД являются:

• создание и модификация таблиц, индексов, ограничений и триггеров,

• выполнение запросов для выборки, добавления, изменения и удаления данных,

• управление транзакциями и контроль доступа к данным,

• оптимизация запросов и производительности баз данных,

• работа с хранимыми процедурами и функциями,

• интеграция с другими продуктами Microsoft и сторонними приложениями.

SQL Server Management Studio (SSMS) — это интегрированная среда для управления инфраструктурой SQL Server. Основные возможности среды:

• подключение к службам SQL Server,

• обозреватель объектов,

• обозреватель шаблонов,

• редактор SQL кода, просмотр плана выполнения запроса,

• обозреватель решений,

• конструктор таблиц,

• конструктор баз данных,

• конструктор запросов и представлений, просмотр свойств объектов, мастер создания скриптов,

• присоединение и отсоединение баз данных,

• создание резервных копий баз данных

На этапе логического проектирования были выделены 9 отношений, связанных между собой. Описание отношений представлены в таблицах 1-9, таблицы, заполненные данными представлены в приложении А.

Таблица 1

Требования к таблице «Клиент»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Ограничение**
Id Клиента	int	Нет	Первичный	
ФИО	nvarchar(50)	Нет		
Контактный телефон	varchar (20)	Нет		Уникальный
Адрес электронной почты	nvarchar(50)	Нет		Уникальный
Адрес	nvarchar(100)	Нет		Уникальный

Таблица 2

Требования к таблице «Сотрудник»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Значение по умолчанию**	**Ограничение**	**Ссылка**
Id Сотрудника	Целое	Нет	Первичный			
ФИО	nvarchar (50)	Нет				
Должность	nvarchar (50)	Нет		Ветеринар		
Дата рождения	date	Нет			>18	
Паспортные данные	varchar (20)	Нет			Уникальный	
Адрес электронной почты	nvarchar (50)	Нет			Уникальный	
Телефон	varchar (20)	Нет			Уникальный	
Id Графика	int	Нет	Внешний			График (Id Графика)

Таблица 3

Требования к таблице «Питомец»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Значение по умолчанию**	**Ограничение**	**Ссылка**
Id Питомца	int	Нет	Первичный			
Id Клиента	int	Нет	Внешний			Клиент (Id Клиента)
Кличка	nvarchar (20)	Нет				
Порода	nvarchar (30)	Нет				
Окрас	nvarchar (20)	Нет				
Пол	nchar (1)	Нет		м	м, ж	

Возраст	int	Нет				

Таблица 4

Требования к таблице «Прием»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Ограничение**	**Ссылка**
Id Приема	int	Нет	Первичный		
Id Сотрудника	int	Нет	Внешний		Сотрудник (Id Сотрудника)
Id Питомца	int	Нет	Внешний		Питомец (Id Питомца)
Дата_время	datetime	Нет		datediff(day, getdate(), Дата_время) >=1	
Комментарий	nvarchar (1000)	Нет			

Таблица 5

Требования к таблице «Услуга»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Ограничение**
Id Услуги	int	Нет	Первичный	
Название	nvarchar(50)	Нет		Уникальный

Таблица 6

Требования к таблице «СкладскаяОперация»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Значение по умолчанию**	**Ограни-чение**	**Ссылка**
Id Складской Операции	int	Нет	Первичный			
Id Сотрудника	int	Нет	Внешний			Сотрудник (Id Сотрудника)
Id Медикамента	int	Нет	Внешний			Медикамент (Id Медикамента)
Id Приема	int	Да	Внешний			Прием (Id Приема)
Дата_время	datetime	Нет		getdate()		
Операция	nvarchar (10)	Нет		Выдача	Выдача, Списание, Получение	
Количество единиц	int	Нет				
Единица измерения	nvarchar (4)	Нет		шт	шт, гр, уп, амп, флак	

Таблица 7

Требования к таблице «Медикамент»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Значение по умолчанию**	**Ограничение**
Id Медикамента	int	Нет	Первичный		
Наименование	nvarchar (50)	Нет			
Стоимость	int	Нет			>0
Срок годности	int	Нет			>0
Количество	int	Нет		0	>=0
Единица измерения	nvarchar(4)	Нет		шт	шт, гр, уп, амп, флак

Таблица 8

Требования к таблице «График»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Значение по умолчанию**
Id Графика	int	Нет	Первичный	
Понедельник	nvarchar (20)	Да		“08:00-20:00”
Вторник	nvarchar (20)	Да		“08:00-20:00”
Среда	nvarchar (20)	Да		“08:00-20:00”
Четверг	nvarchar (20)	Да		“08:00-20:00”
Пятница	nvarchar (20)	Да		“08:00-20:00”
Суббота	nvarchar (20)	Да		“10:00-18:00”
Воскресенье	nvarchar (20)	Да		“10:00-18:00”

Таблица 9

Требования к таблице «УслугаПриема»

Столбец	**Тип данных**	**Нуль?**	**Ключ**	**Ссылка**
Id Приема	int	Нет	Первичный Внешний	Прием (Id Приема)
Id Услуги	int	Нет		Услуга (Id Услуги)

Схема получившейся базы данных представлена на рис. 1.23.

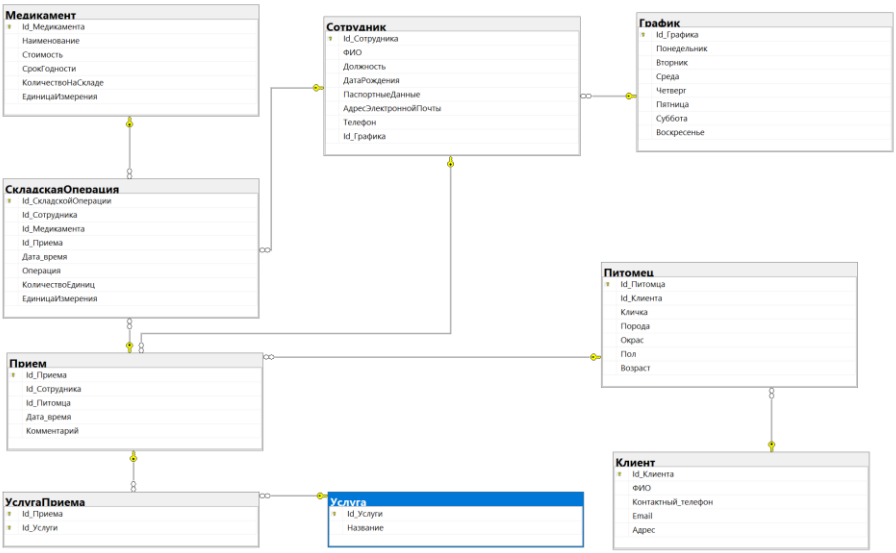

Рис. 1.23. Схема базы данных ветеринарной клиники

Помимо таблиц в БД также созданы представления и хранимые процедуры, для наилучшего взаимодействия с клиентским приложением.

У таблицы Питомец есть представление Питомцы_с_ФИО (Приложение Б, п. 1). Данное представление выводит информацию о питомцах, включая их идентификатор, кличку, породу, окрас, пол и возраст, а также ФИО их владельцев (клиентов).

У таблицы СкладскаяОперация есть представление Представление_СкладскаяОперация (Приложение Б, п. 2). Данное представление предоставляет данные о складских операциях, включая идентификатор операции, ФИО сотрудника, наименование медикамента, дату и время операции, тип операции, количество единиц и единицу измерения.

У таблицы Прием есть представление ПолнаяИнформацияОПриемах (Приложение Б, п. 3). В данном представлении содержится информация о приемах, включая идентификатор приема, ФИО сотрудника, кличку питомца, ФИО клиента, дату и время приема, комментарии, а также оказанные услуги (при наличии).

У таблицы Сотрудник есть представление V_Ветеринары_График (Приложение Б, п. 4). Данное представление отображает график работы сотрудников с должностью "ветеринар", включая их ФИО, должность и расписание по дням недели.

У таблицы Сотрудник также есть представление СотрудникиСкладаСГрафиком (Приложение Б, п. 5). Это представление выводит график работы сотрудников склада, включая заведующих складом, складских операторов, кладовщиков и специалистов по учету товаров. В нем отображаются ФИО сотрудников и их расписание по дням недели.

У таблицы Клиент есть процедура AddClient (Приложение Б, п. 6). Данная процедура используется для добавления нового клиента в базу данных. В процессе выполнения проверяется уникальность ФИО, контактного телефона, Email и адреса. Если данные проверки успешно пройдены, информация о клиенте добавляется в таблицу Клиент. В случае возникновения ошибок процедура обрабатывает их и передает сообщение в приложение.

У таблицы Клиент есть процедура UpdateClient (Приложение Б, п. 7). Данная процедура используется для обновления данных клиента в базе данных. В процессе выполнения проверяется уникальность нового контактного телефона, Email и адреса среди других клиентов. Если проверки успешно пройдены, обновление выполняется на основе уникального ФИО клиента. В случае, если клиент с указанным ФИО не найден, возвращается сообщение об ошибке. Все возникающие ошибки обрабатываются и передаются в приложение.

У таблицы Сотрудник есть процедура AddSotr (Приложение Б, п. 8). Данная процедура используется для добавления нового сотрудника в базу данных. В процессе выполнения проверяется уникальность ФИО, паспортных данных, адреса электронной почты и телефона. Также осуществляется проверка на достижение сотрудником возраста 18 лет и наличие графика с переданным идентификатором. Если все проверки успешно пройдены, данные сотрудника добавляются в таблицу. В случае возникновения ошибок процедура возвращает соответствующие сообщения.

У таблицы Сотрудник есть процедура UpdateSotrByFIO (Приложение Б, п. 9). Данная процедура позволяет обновить данные сотрудника по

уникальному ФИО. В ходе выполнения проверяется наличие сотрудника с указанным ФИО, достижение совершеннолетия, уникальность паспортных данных, адреса электронной почты и телефона, а также существование графика с переданным идентификатором. При успешном выполнении обновляются должность, дата рождения, паспортные данные, адрес электронной почты, телефон и идентификатор графика. В случае некорректных данных возвращаются соответствующие сообщения об ошибках.

У таблицы Питомец есть процедура ДобавитьПитомца (Приложение Б, п. 10). Эта процедура позволяет добавить нового питомца для клиента. В процессе выполнения осуществляется проверка на существование клиента по переданному ФИО, а также уникальность клички питомца в рамках одного клиента. Если клиент не найден или кличка уже существует у этого клиента, возвращается соответствующее сообщение об ошибке. При успешном выполнении в таблицу Питомец добавляется запись с информацией о питомце, включая кличку, породу, окрас, пол и возраст.

У таблицы Медикамент есть процедура ДобавитьМедикамент (Приложение Б, п. 11). Эта процедура добавляет новый медикамент в таблицу с выполнением ряда проверок. Проверяется уникальность наименования медикамента, корректность стоимости (должна быть больше 0), срока годности (должен быть больше 0), количества (не может быть отрицательным), а также единицы измерения (должна соответствовать одному из допустимых значений: шт, гр, уп, амп, флак). При успешной проверке всех условий запись добавляется в таблицу Медикамент. В случае несоблюдения условий возвращается соответствующее сообщение об ошибке.

У таблицы Медикамент есть процедура ОбновитьМедикамент (Приложение Б, п. 12). Эта процедура обновляет информацию о медикаменте с заданным наименованием. Проверяется существование медикамента, корректность новой стоимости (должна быть больше 0), нового количества

(не может быть отрицательным), новой единицы измерения (должна соответствовать одному из допустимых значений: шт, гр, уп, амп, флак). При этом срок годности является неизменяемым параметром, и попытка его изменения приведёт к ошибке. Если все условия выполнены, обновляются стоимость, количество и единица измерения медикамента. В случае нарушений возвращается соответствующее сообщение об ошибке.

У таблицы Прием есть процедура ДобавитьПриемИУслугу (Приложение Б, п. 13), которая обрабатывает добавление нового приёма и/или услуги к существующему приёму. Проверяется наличие клиента, питомца и сотрудника в базе данных, корректность времени приёма (оно должно быть в будущем, попадать в рабочие часы сотрудника и не пересекаться с другими приёмами для данного сотрудника или питомца). Если все условия соблюдены, создаётся новая запись о приёме с указанным комментарием. Для существующего приёма проверяется, не прошло ли более 24 часов с момента его проведения, и при соблюдении условий добавляется новая услуга, если она не была ранее добавлена. В случае нарушения условий возвращается соответствующее сообщение об ошибке.

У таблицы СкладскаяОперация есть процедура ДобавитьСкладскуюОперацию (Приложение Б, п. 14), которая добавляет новую запись о складской операции, связанной с медикаментами. Проверяется наличие сотрудника с заданным ФИО, его должность (должна быть связана со складскими операциями) и актуальность рабочего времени. Учитывается корректность параметров: операция может быть только "Выдача", "Списание" или "Получение"; количество единиц должно быть больше нуля; медикамент с указанным наименованием должен существовать. Для операций "Выдача" и "Списание" проверяются доступные остатки медикамента, а для "Получение" осуществляется их увеличение. Если операция связана с приёмом, проверяется существование соответствующего Id_Приема. В случае выполнения всех условий обновляется количество

медикамента в базе, и в таблицу СкладскаяОперация добавляется новая запись. В случае нарушений возвращается сообщение об ошибке.

У таблицы График есть процедура ДобавитьГрафик (Приложение Б, п. 15), которая добавляет новый график рабочего времени. Для каждого дня недели задаётся временной интервал в формате HH:MM-HH:MM или NULL (если день выходной). Проверяется корректность формата времени для каждого дня. Если формат некорректен, процедура завершает выполнение с соответствующим сообщением об ошибке. Дополнительно проверяется, существует ли уже в таблице График запись с аналогичными временными интервалами. Если запись найдена, добавление нового графика отклоняется, и возвращается сообщение об ошибке. В случае успешного выполнения записи добавляются в таблицу График.

Процедура УдалитьГрафик (Приложение Б, п. 16) удаляет график рабочего времени из таблицы График, принимая параметры Id_Графика и дни недели. Если передан Id_Графика, проверяется его существование и соответствие с введенными днями недели, а если Id_Графика не передан, процедура ищет его по дням недели. Также проверяется, используется ли график сотрудниками, и если используется, удаление невозможно. В случае успешной проверки удаляется соответствующий график.

Процедура ДобавитьУслугу (Приложение Б, п. 17) добавляет новую услугу в таблицу Услуга. Принимает параметр `Название` услуги. Вначале проверяется, существует ли уже услуга с таким названием в таблице. Если услуга с таким названием уже есть, выводится ошибка. Если услуги с таким названием нет, то выполняется вставка новой записи в таблицу и выводится сообщение об успешном добавлении услуги.

Процедура ОбновитьКомментарийПриема (Приложение Б, п. 18) обновляет комментарий для конкретного приема в таблице Прием. Она принимает параметры: ФИО_Клиента, Кличка_Питомца, ФИО_Сотрудника, Дата_время и новый текст комментария. Сначала проверяется существование клиента по ФИО, затем проверяется наличие питомца у этого клиента,

наличие сотрудника с указанным ФИО и запись о приеме с указанными данными. Если все проверки пройдены, то проверяется, прошло ли более 24 часов с момента приема — если да, изменение комментария невозможно. В случае успеха комментарий обновляется в таблице, и выводится сообщение об успешном обновлении.

Процедура УдалитьПрием (Приложение Б, п. 19) удаляет прием по данным клиента, питомца, сотрудника и времени приема. Она принимает параметры: ФИО_Клиента, Кличка_Питомца, ФИО_Сотрудника и Дата_время. Сначала проверяется существование клиента, питомца и сотрудника. Затем проверяется наличие записи о приеме с указанными данными. Также проверяется, что до приема осталось больше 24 часов и что прием не связан с услугами в таблице УслугаПриема. Если все условия выполнены, запись о приеме удаляется, и выводится сообщение об успешном удалении.

Процедура ОбновитьГрафик (Приложение Б, п. 20) обновляет график для заданного Id_Графика с проверками на существование записи с указанным Id и корректность формата времени для каждого дня недели. Она принимает параметры: Id_Графика, Понедельник, Вторник, Среда, Четверг, Пятница, Суббота, Воскресенье. Сначала проверяется, существует ли запись с данным Id. Затем проводится проверка формата времени для каждого дня недели и на уникальность графика. Если все условия выполнены, график обновляется с новыми значениями для каждого дня недели.

Процедура ОбновитьВозрастПитомца (Приложение Б, п. 21) обновляет возраст питомца для указанного клиента с проверками. Она принимает параметры: ФИО_Клиента, Кличка питомца и новый возраст. Вначале проверяется существование клиента по ФИО и питомца по кличке. Далее проверяется, что новый возраст больше текущего и является положительным числом. Если все проверки пройдены, возраст питомца обновляется в базе данных. В случае ошибок или несоответствий с данными процедура выдает

сообщения об ошибках. В случае успешного обновления, выводится сообщение об успешной операции.

Процедура УдалитьУслугуПриема (Приложение Б, п. 22) удаляет услугу, оказанную при приеме питомца. Она принимает параметры: ФИО клиента, кличку питомца, ФИО сотрудника, дату и время приёма, а также название оказанной услуги. В начале проверяется существование клиента, питомца, сотрудника и записи о приёме. Затем проверяется, прошло ли более суток с момента приёма и существует ли указанная услуга. Если все проверки пройдены, процедура удаляет соответствующую запись из таблицы УслугаПриема. В случае ошибок или несоответствий с данными, процедура выводит соответствующие сообщения об ошибках.

ГЛАВА 2. ОПИСАНИЕ ФУНКЦИОНИРОВАНИЯ БД

2.1. НАЗНАЧЕНИЕ И ПЕРЕЧЕНЬ ФУНКЦИЙ БД

Программа, предназначена для автоматизации и упрощения работы ветеринарной клиники. Интерфейс создан для четырех пользователей – генерального директора, администратора, менеджера по запасам и ветеринара. С помощью данного интерфейса сотрудники могут взаимодействовать с базой данных.

При реализации приложения для создания интерфейса программы были использованы WinForms в среде Visual Studio на языке C# [8].

Генеральный директор имеет такие функции, как:

- Добавление услуг;
- Изменение расписания работы персонала;
- Просмотр данных о складских операциях.

Администратор имеет такие функции, как:

- Просмотр клиентской базы;
- Добавление данных клиента;
- Обновление данных клиента;
- Просмотр базы сотрудников;
- Добавление данных сотрудника;
- Обновление данных сотрудника;
- Просмотр базы графиков;
- Добавление новых графиков;
- Обновление уже имеющихся графиков;
- Удаление графиков.

Менеджер по запасам имеет такие функции, как:

- Просмотр данных о складских операциях;

- Проведение складских операций;
- Просмотр данных о имеющихся медикаментах;
- Добавление новых медикаментов;
- Обновление данных о медикаменте.

Ветеринар имеет такие функции, как:

- Просмотр базы данных питомцев;
- Добавление данных о питомцах;
- Обновление возраста уже имеющегося в БД питомца;
- Просмотр информации о приемах и услугах;
- Добавление данных приема;
- Обновление комментария к приему;
- Удаление данных о приеме;
- Добавление услуги к приему;
- Удаление услуги приема.

2.2. ОПИСАНИЕ РАБОТЫ С ПРИЛОЖЕНИЕМ

Работа с программой «Ветеринарная клиника» начинается с главной страницы, где пользователь должен авторизоваться под пользователем и паролем (рис. 2.1). На форме создан выпадающий список с доступными пользователями. Происходит проверка правильности введенного пароля. На некорректно введенный пароль приложение отреагирует ошибкой (рис. 2.2). Код всех форм представлен в приложении В [8].

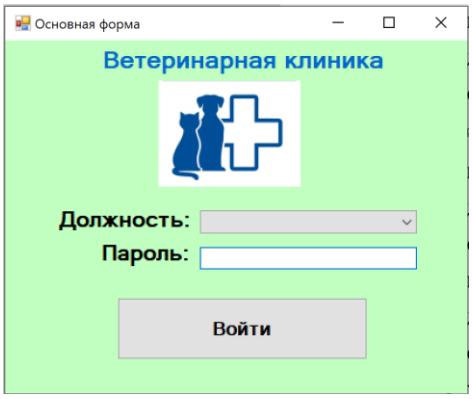

Рис. 2.1. Основная форма приложения

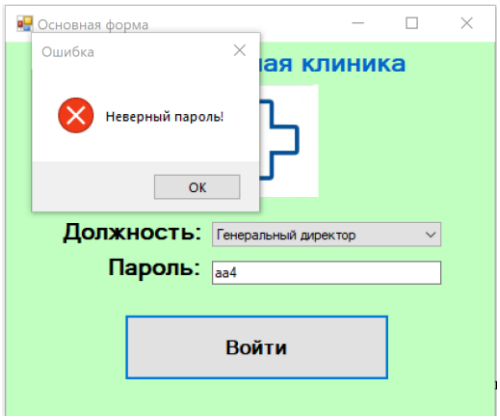

Рис. 2.2. Ошибка «Неверный пароль»

Войдем в приложение под пользователем «генеральный директор». Попадем в меню, где нам доступны такие варианты как: добавление услуги, просмотр и обновление графиков, просмотр складских операций (рис. 2.3). Каждая из вышеперечисленных кнопок ведет на отдельную форму, к которым мы обратимся позднее.

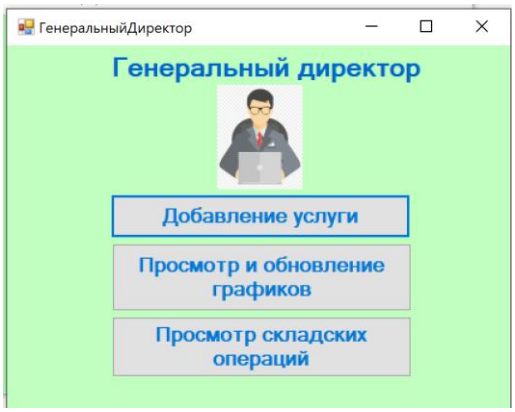

Рис. 2.3. Форма ГенеральныйДиректор

Из основной формы нам также доступен переход на форму Администратора. На данной форме у пользователя есть следующие кнопки

для перехода на дальнейшие формы: работа с клиентами, работа с сотрудниками, работа с графиками (рис. 2.4). К формам, на которые переводят эти кнопки, мы определимся позднее.

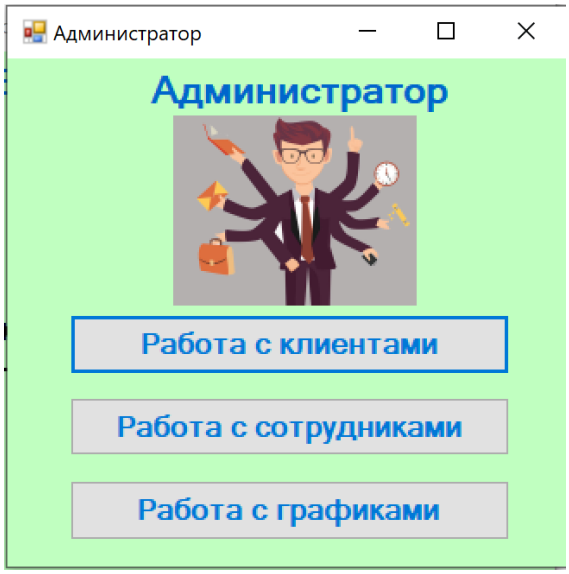

Рис. 2.4. Форма Администратор

Также на основной форме в качестве пользователя может быть выбран менеджер по запасам. Перейдя на форму менеджера по запасам пользователю будут доступны две кнопки: «Добавить складскую операцию» и «Редактировать медикаменты» (рис. 2.5). Формы, на которые данные кнопки осуществляют переход, будут рассмотрены позднее.

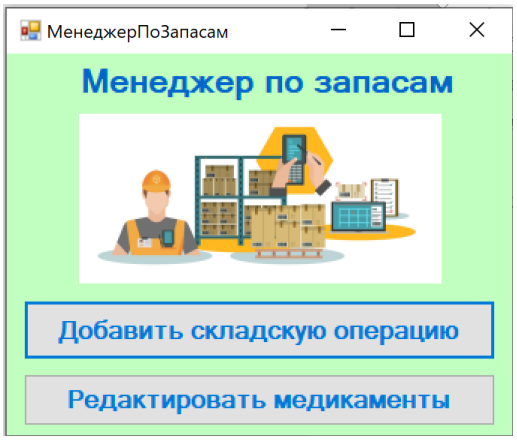

Рис. 2.5. Форма МенеджерПоЗапасам

Последней формой, на которую можно перейти с основной формы является форма Ветеринар. Пользователю на данной форме доступны две кнопки: «Работа с питомцами» и «Проведение приема» (рис. 2.6). Формы, на которые данные кнопки осуществляют переход, будут рассмотрены позднее.

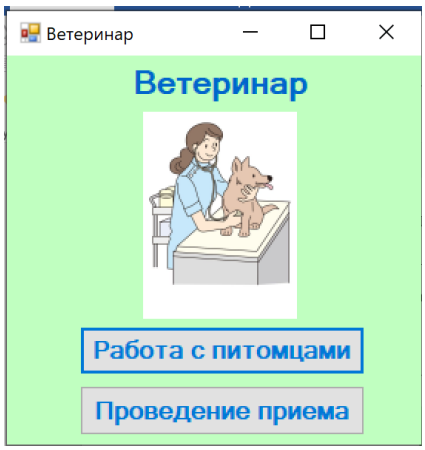

Рис. 2.6. Форма Ветеринар

Теперь перейдем к рассмотрению формы Сотрудники (рис. 2.7). Доступ к ней имеет администратор, по кнопке «Работа с сотрудниками». Заметим, что на данной форме реализовано введение даты рождения через календарь, кроме того, график работы сотрудника предложено выбрать из выпадающего списка. На данной форме реализована сортировка данных, а также поиск по ФИО. Используя данную форму, пользователь имеет возможность добавить нового сотрудника (рис. 2.8) или обновить данные существующего сотрудника (рис. 2.9). На рис. 2.10-2.19 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

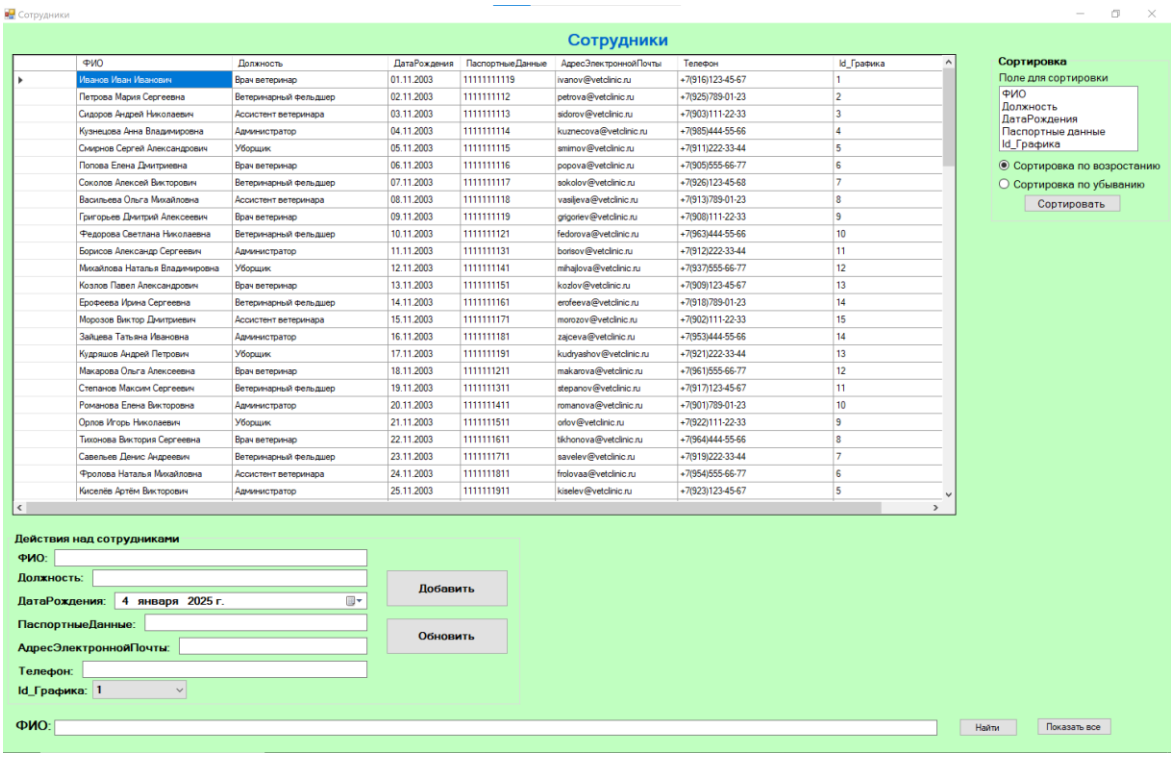

Рис. 2.7. Форма Сотрудники

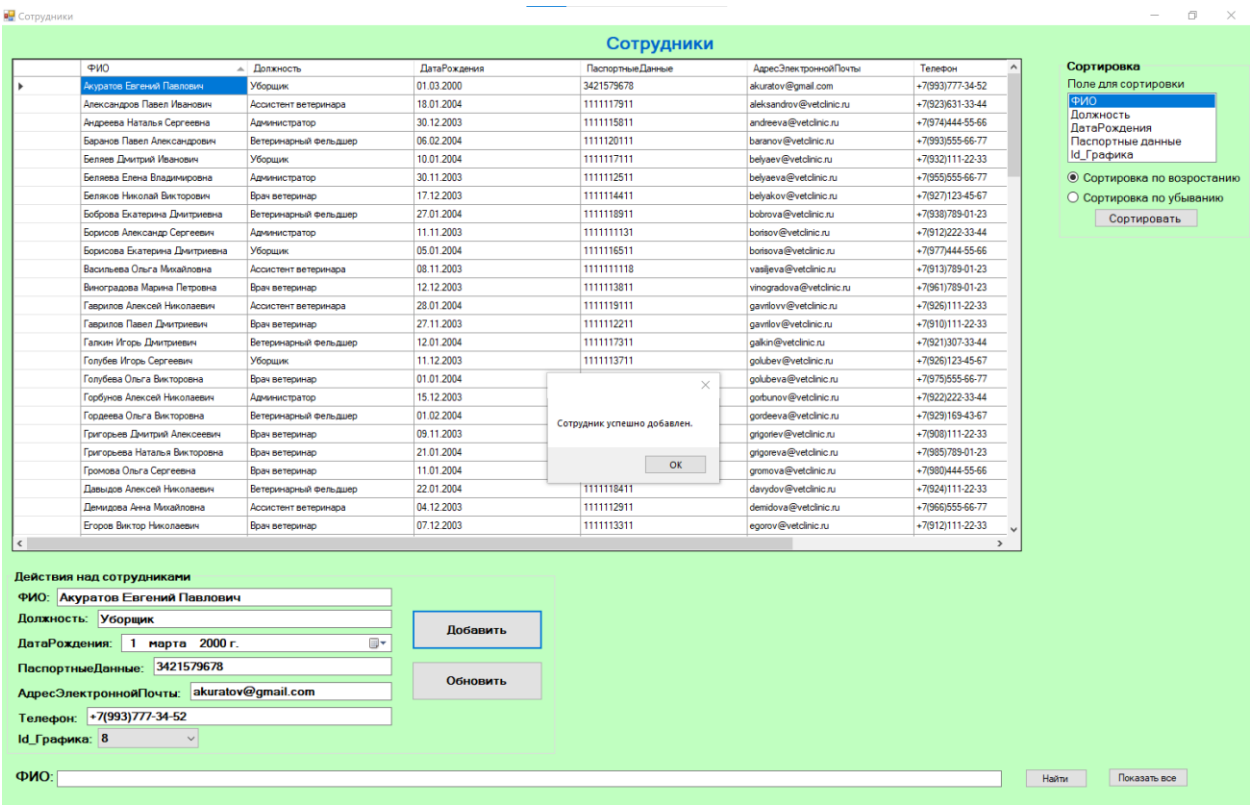

Рис. 2.8. Успешное добавление нового сотрудника

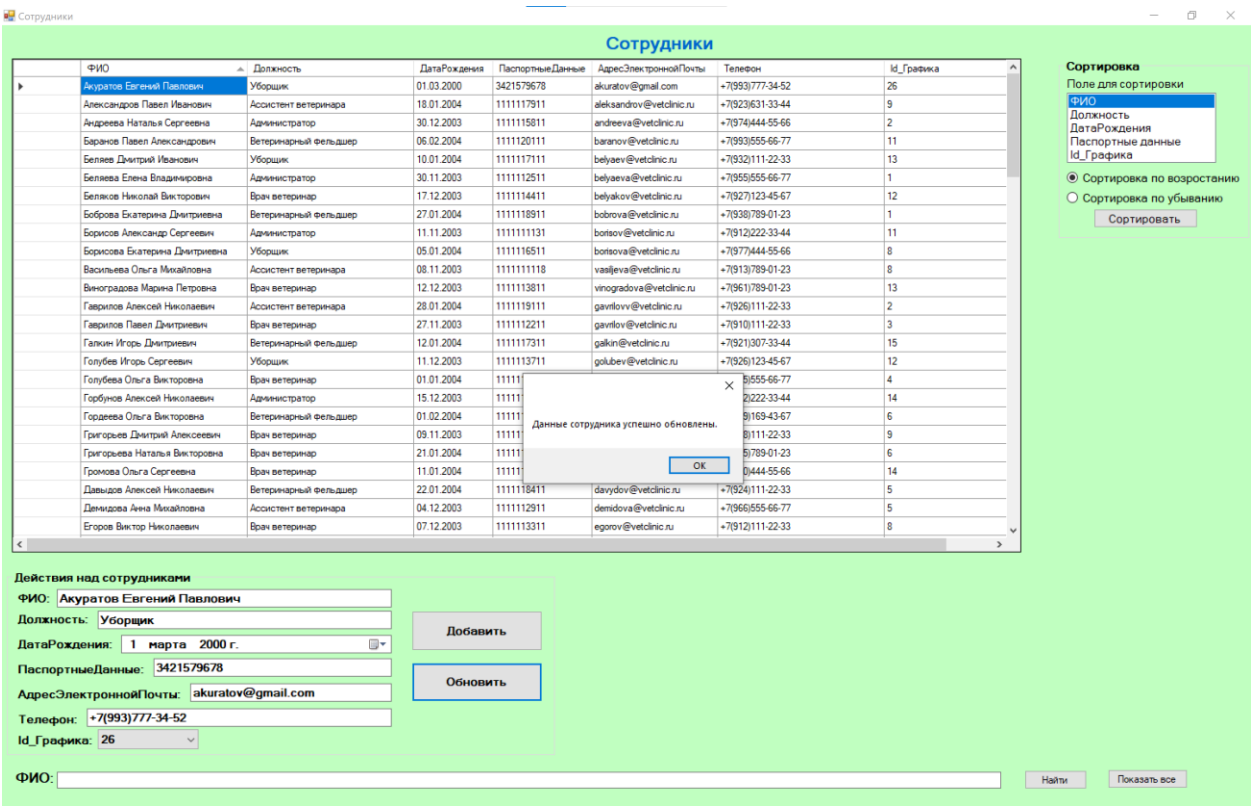

Рис. 2.9. Успешное обновление данных сотрудника

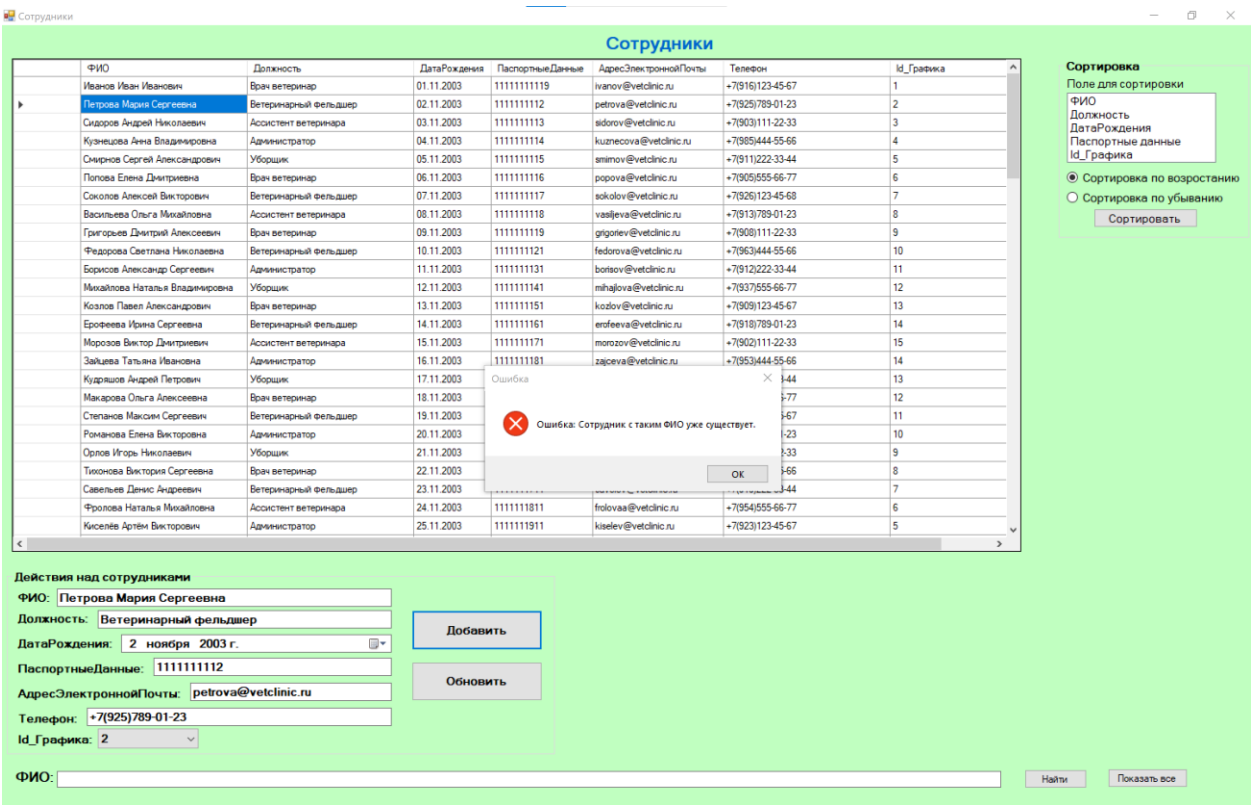

Рис. 2.10. Ошибка при добавлении сотрудника, сотрудник с таким ФИО уже существует

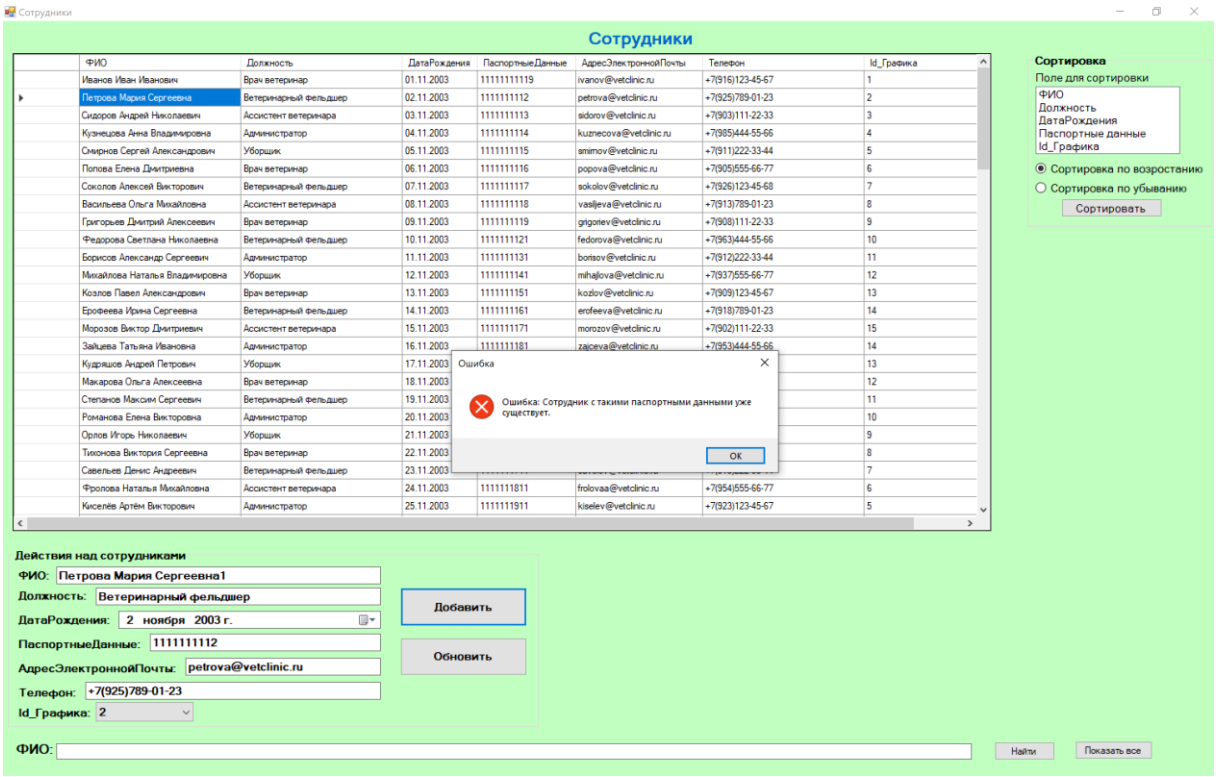

Рис. 2.11. Ошибка при добавлении сотрудника, сотрудник с такими паспортными данными уже существует

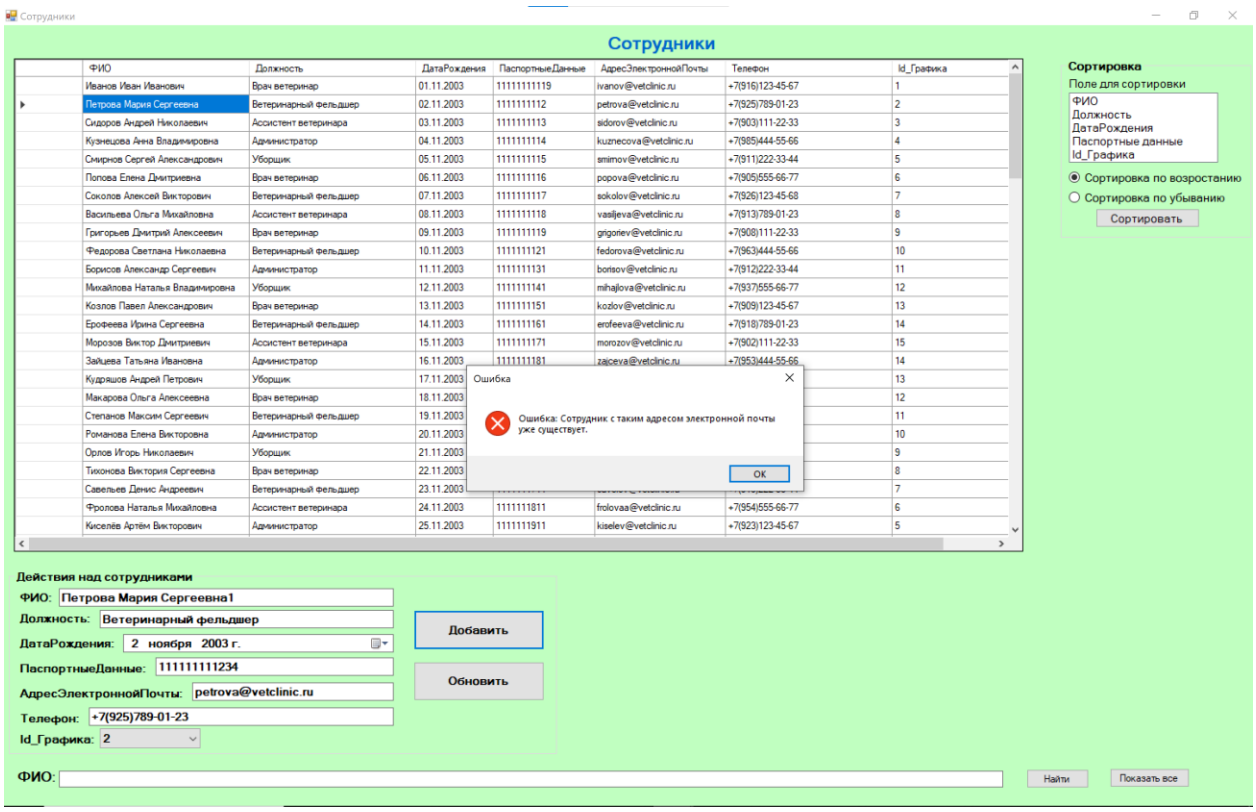

Рис. 2.12. Ошибка при добавлении сотрудника, сотрудник с таким адресом электронной почты уже существует

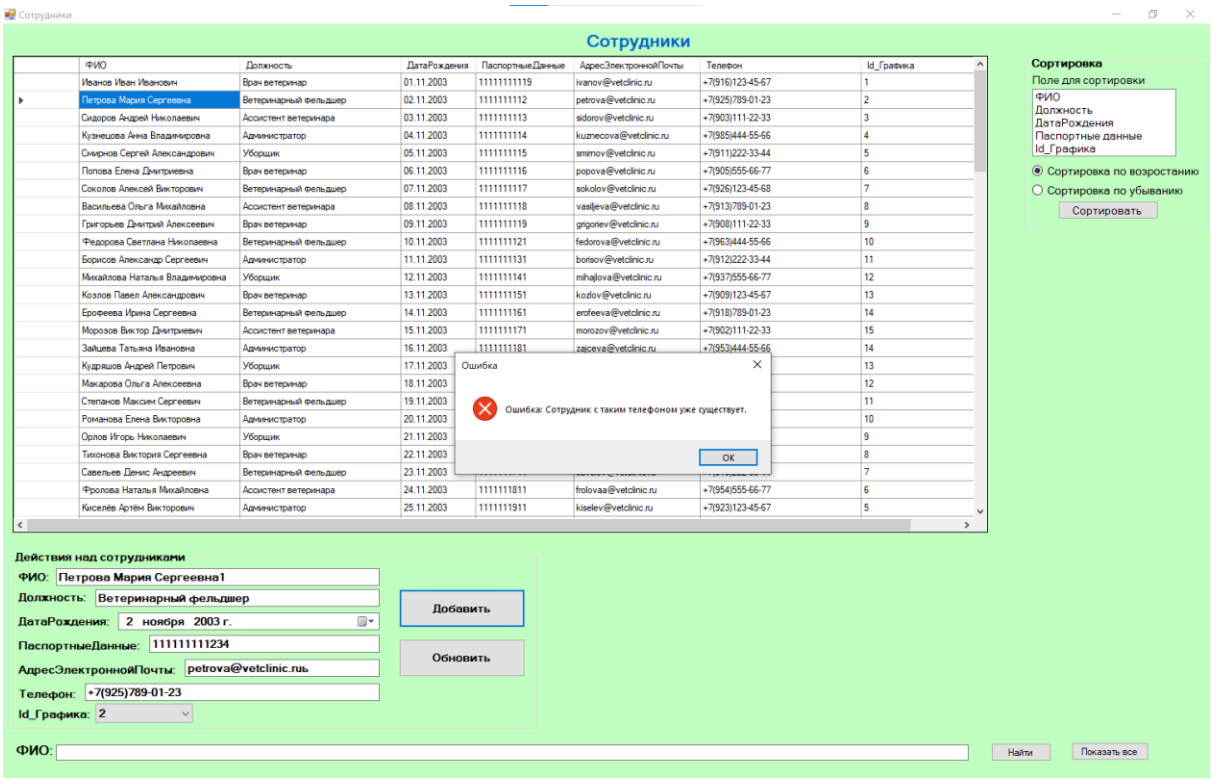

Рис. 2.13. Ошибка при добавлении сотрудника, сотрудник с таким номером телефона уже существует

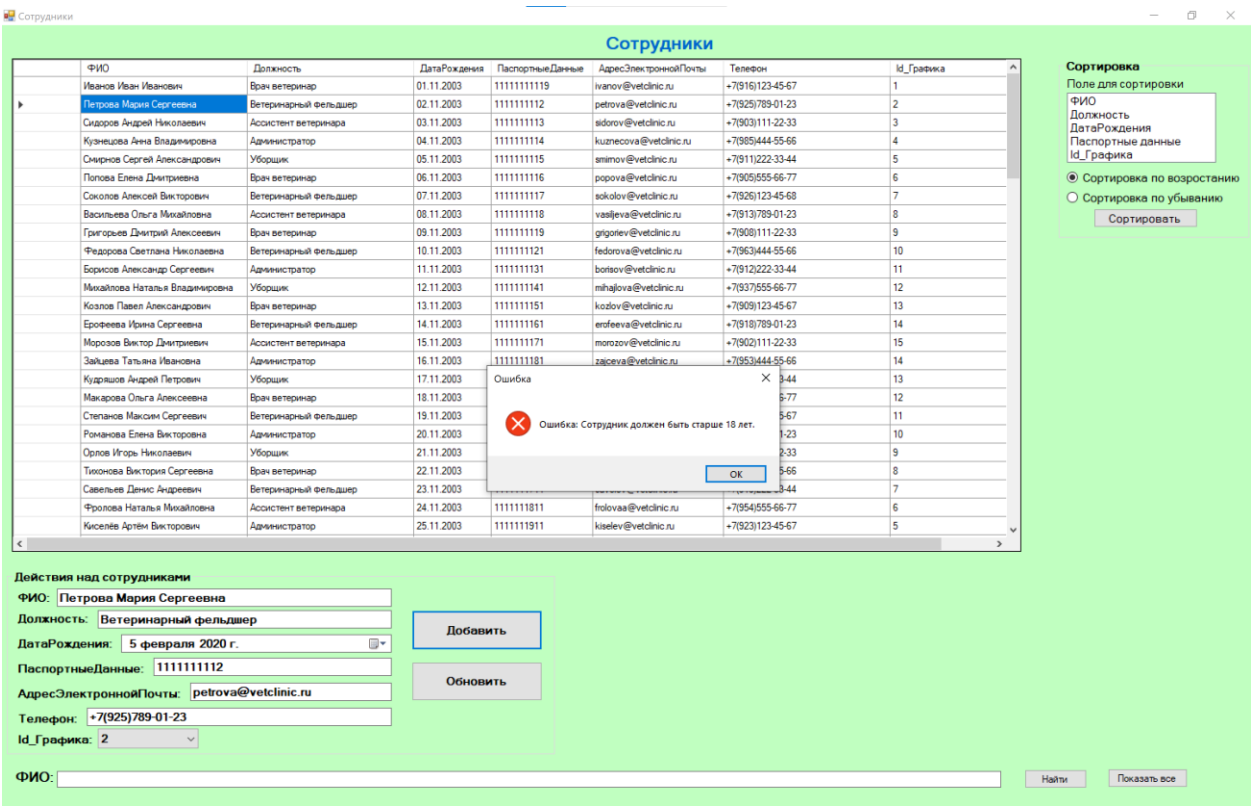

Рис. 2.14. Ошибка при добавлении сотрудника, возраст сотрудника меньше 18 лет

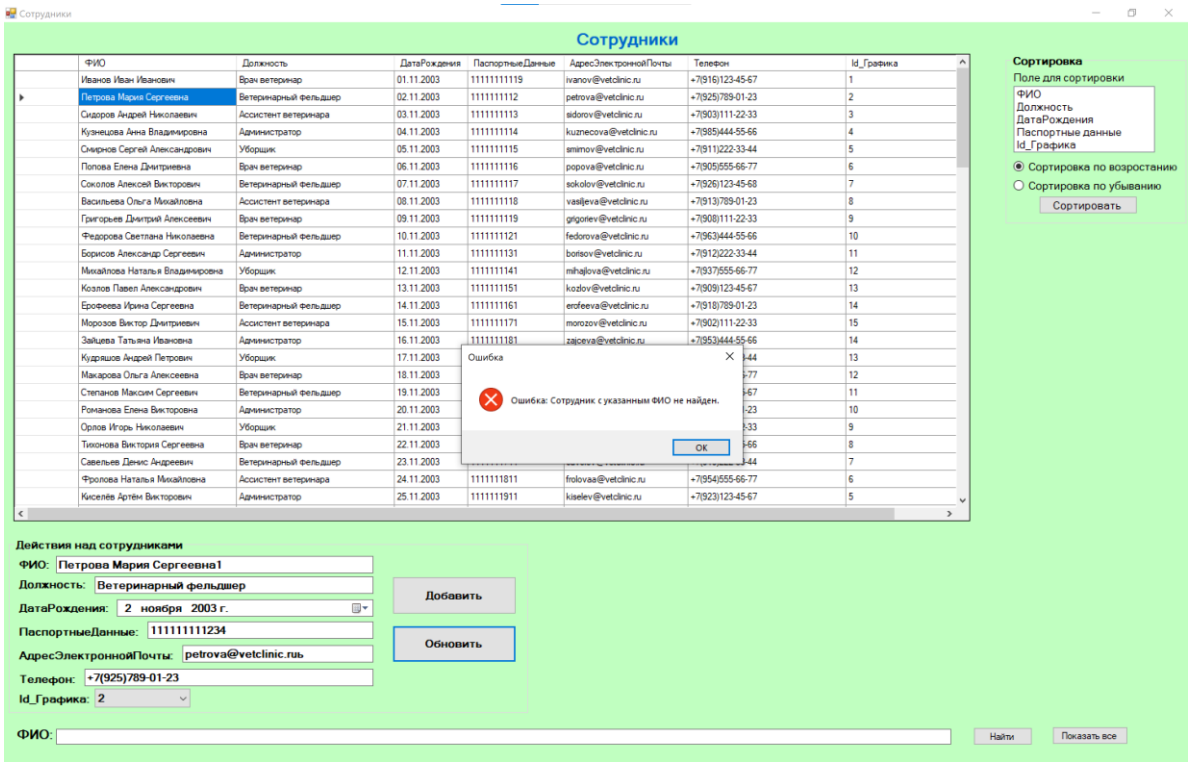

Рис. 2.15. Ошибка при обновлении сотрудника, сотрудник с таким ФИО не существует

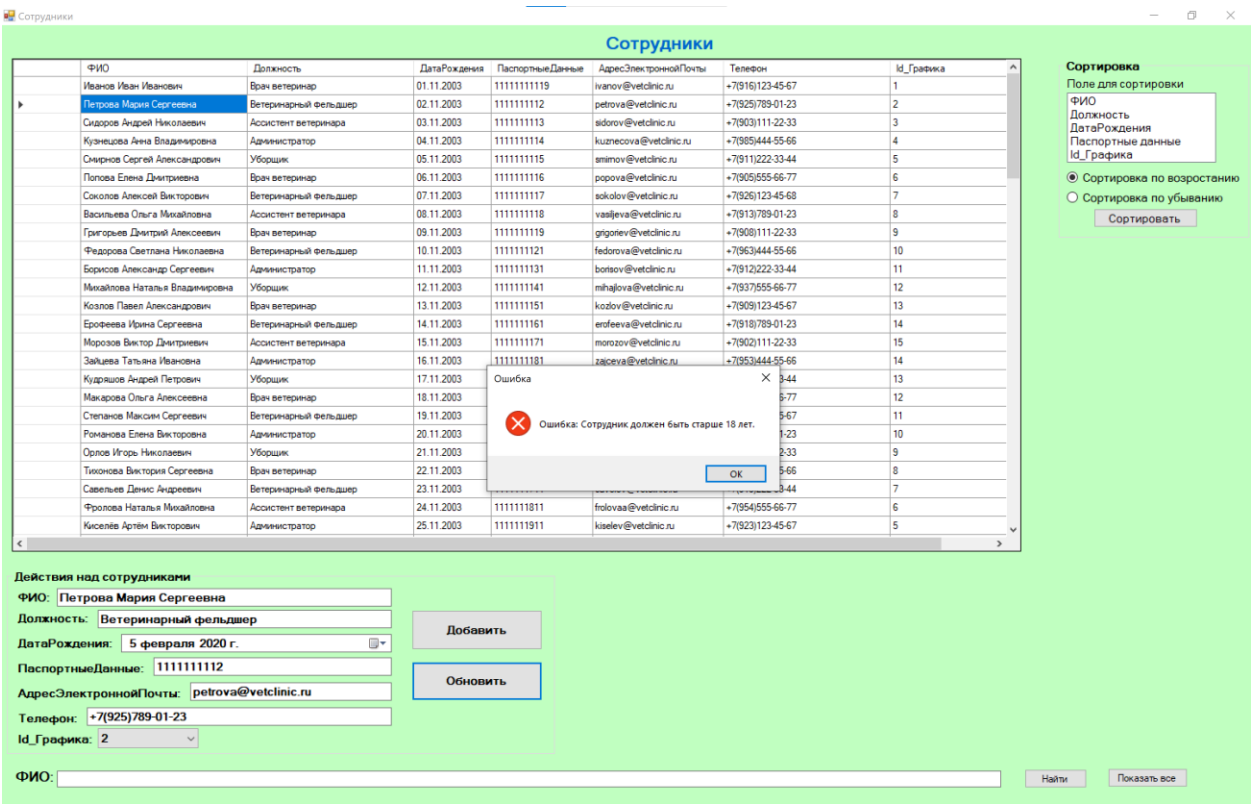

Рис. 2.16. Ошибка при обновлении сотрудника, возраст сотрудника меньше 18 лет

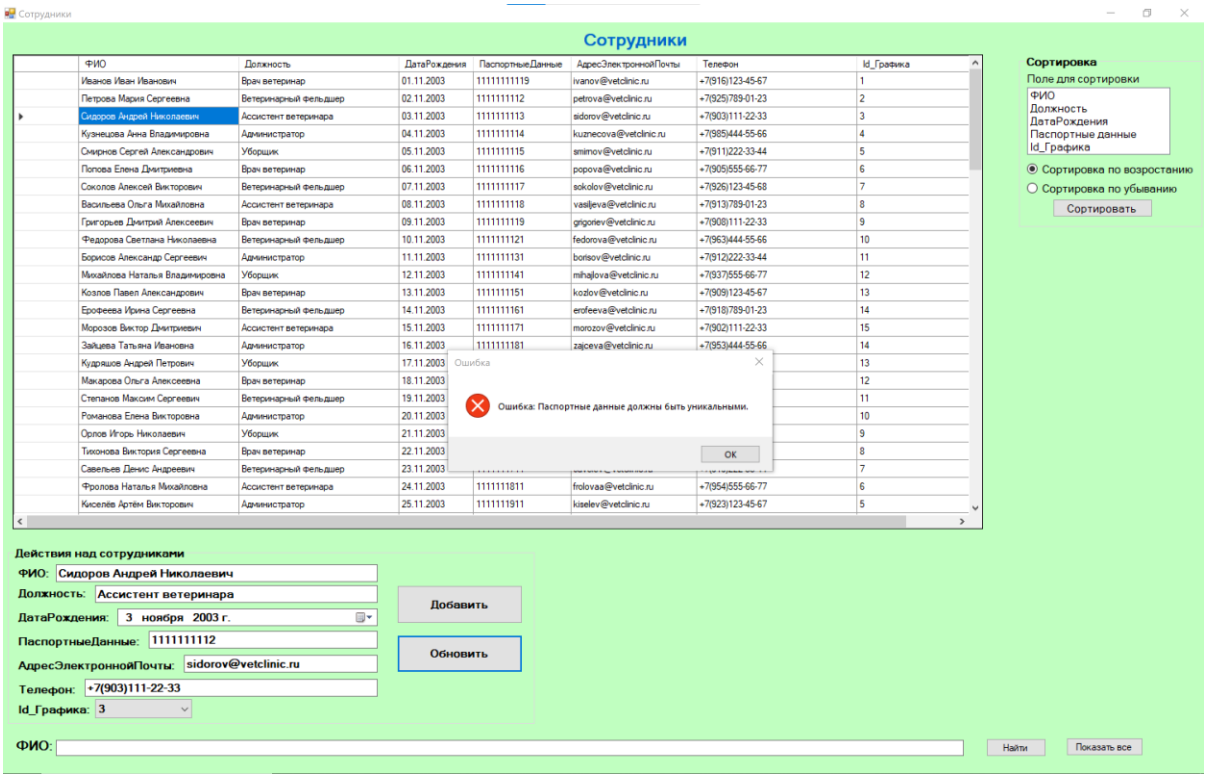

Рис. 2.17. Ошибка при обновлении сотрудника, неуникальные паспортные данные

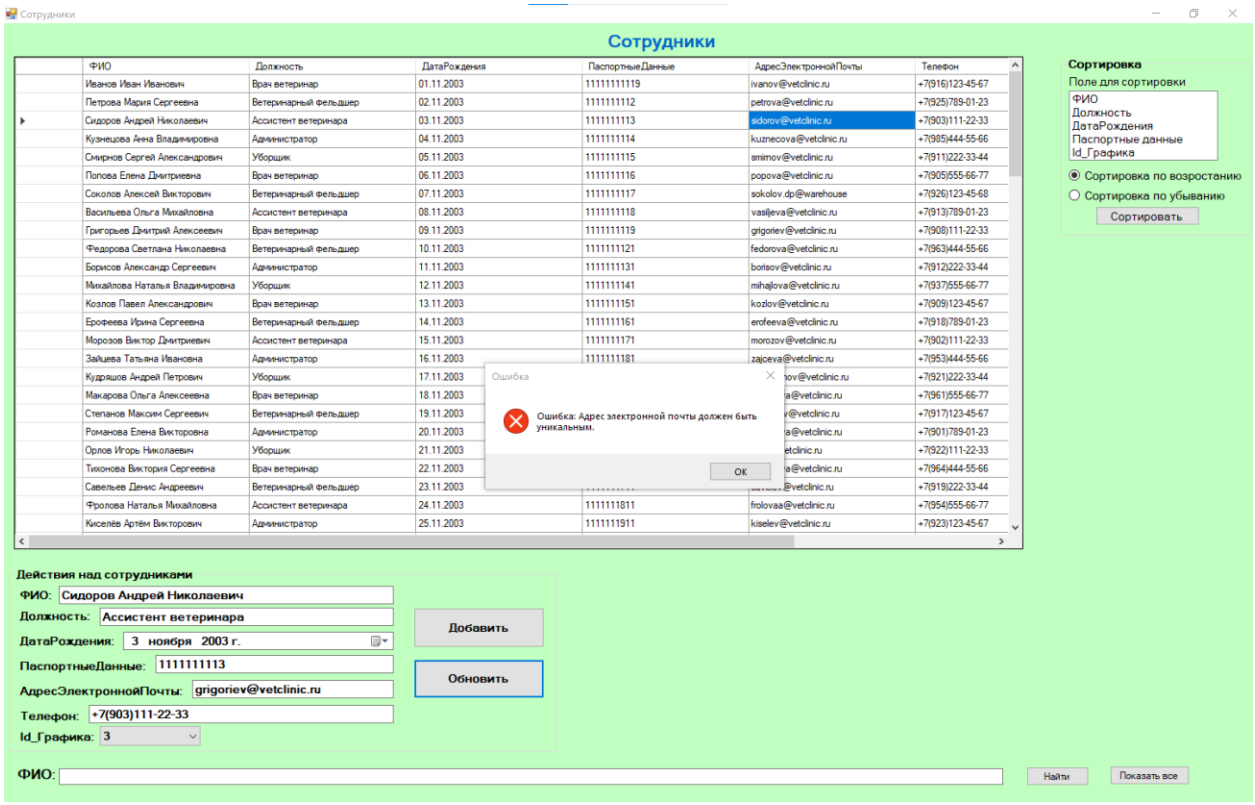

Рис. 2.18. Ошибка при обновлении сотрудника, неуникальный адрес электронной почты

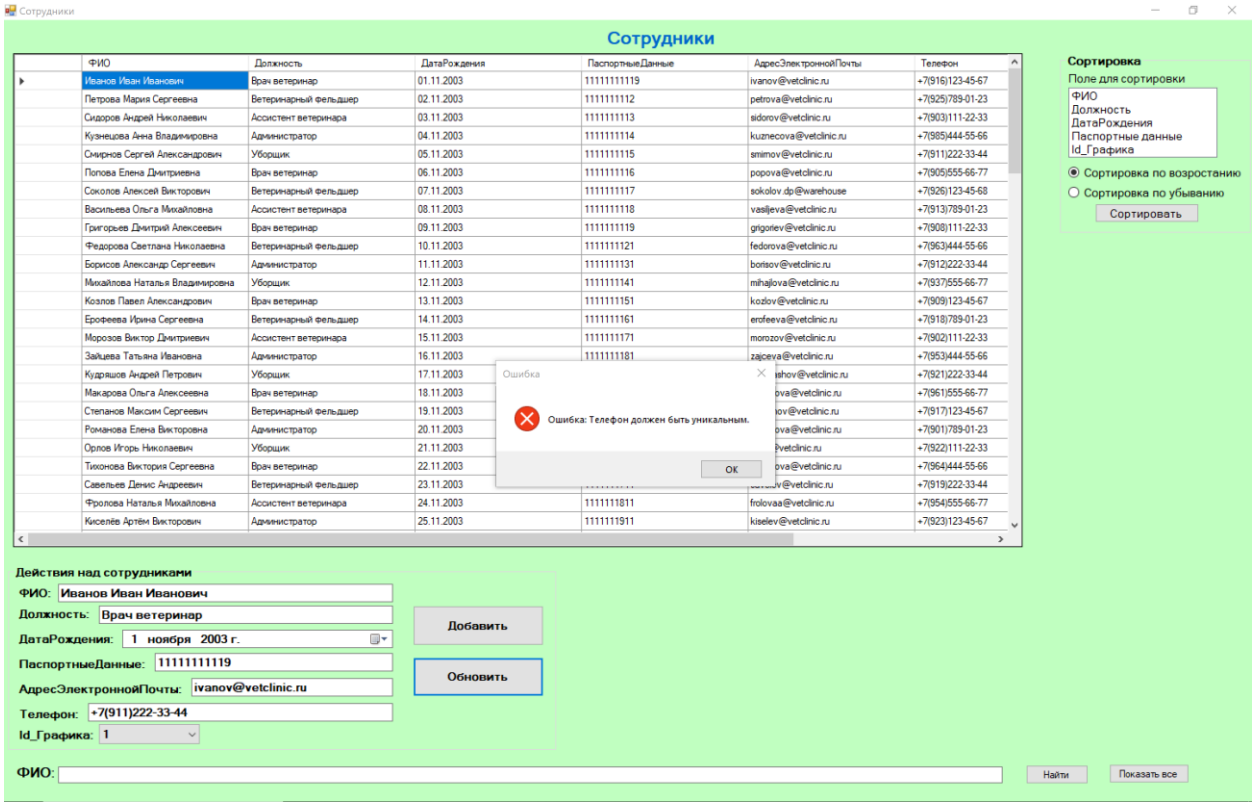

Рис. 2.19. Ошибка при обновлении сотрудника, неуникальный номер телефона

Теперь перейдем к рассмотрению формы Клиенты (рис. 2.20). Доступ к ней имеет администратор, по кнопке «Работа с клиентами». На данной форме реализована сортировка данных, а также поиск по ФИО. Используя данную форму, пользователь имеет возможность добавить нового клиента (рис. 2.21) или обновить данные существующего клиента (рис. 2.22). На рис. 2.23-2.27 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

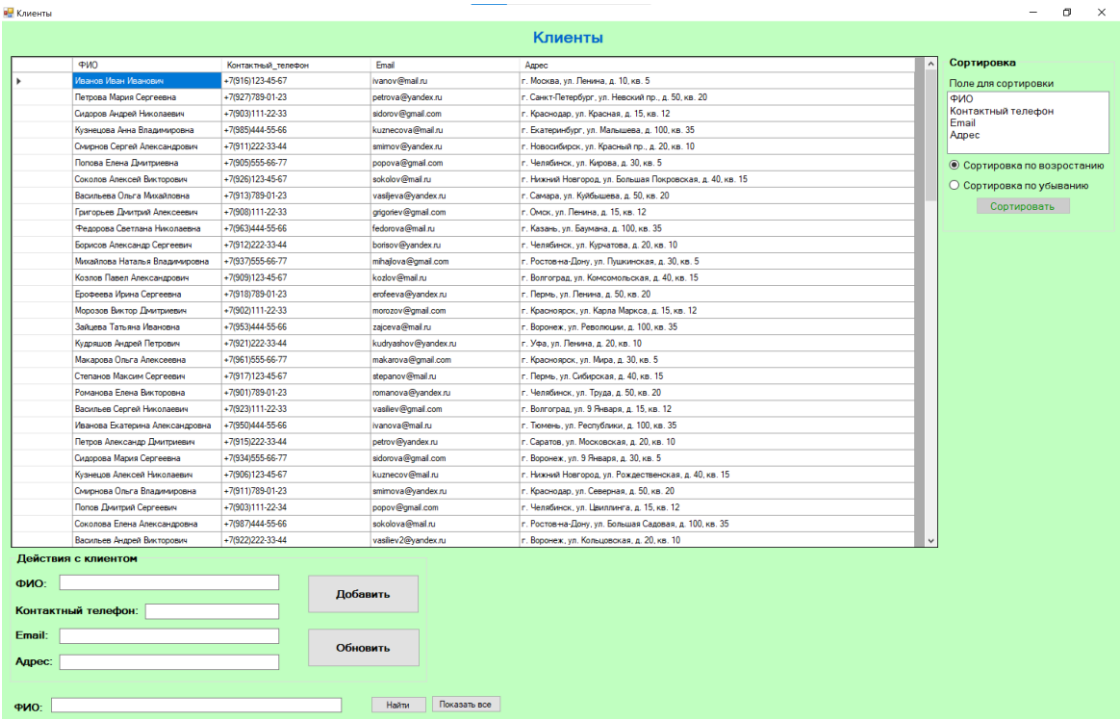

Рис. 2.20. Форма Клиенты

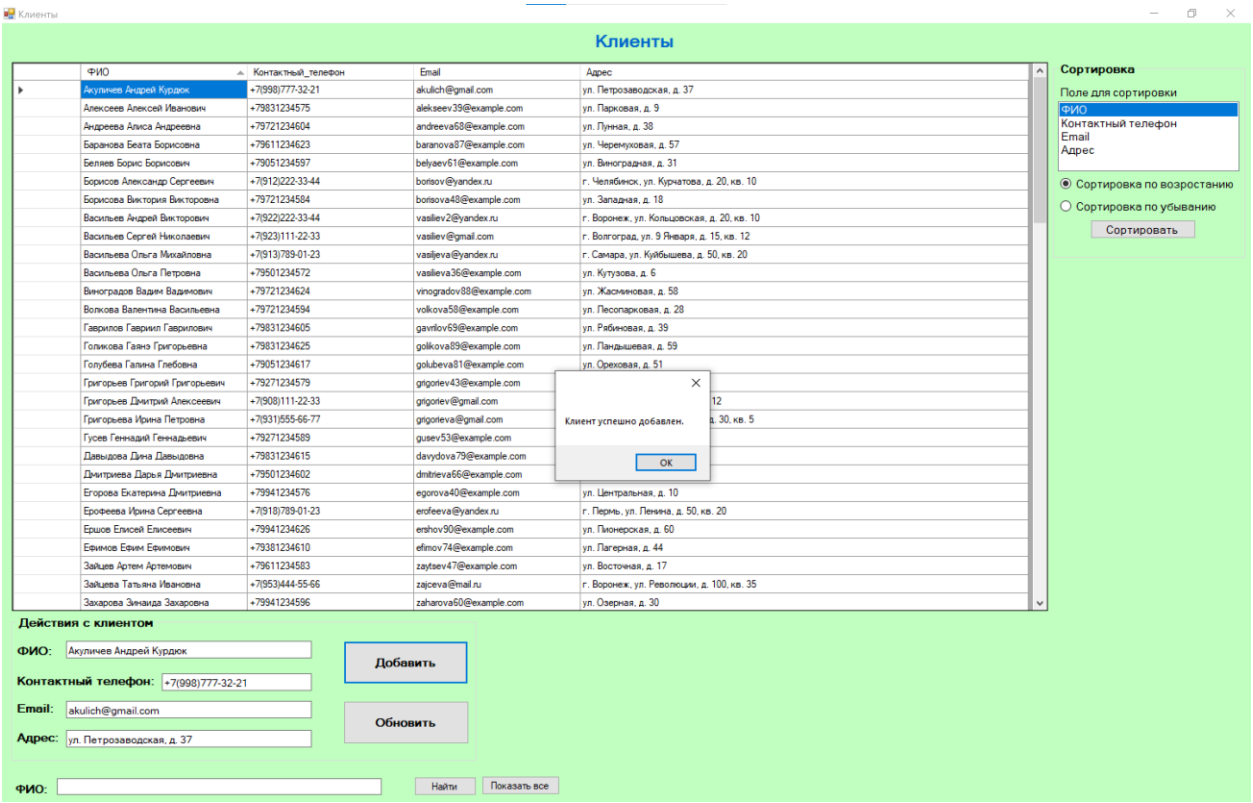

Рис. 2.21. Успешное добавление клиента

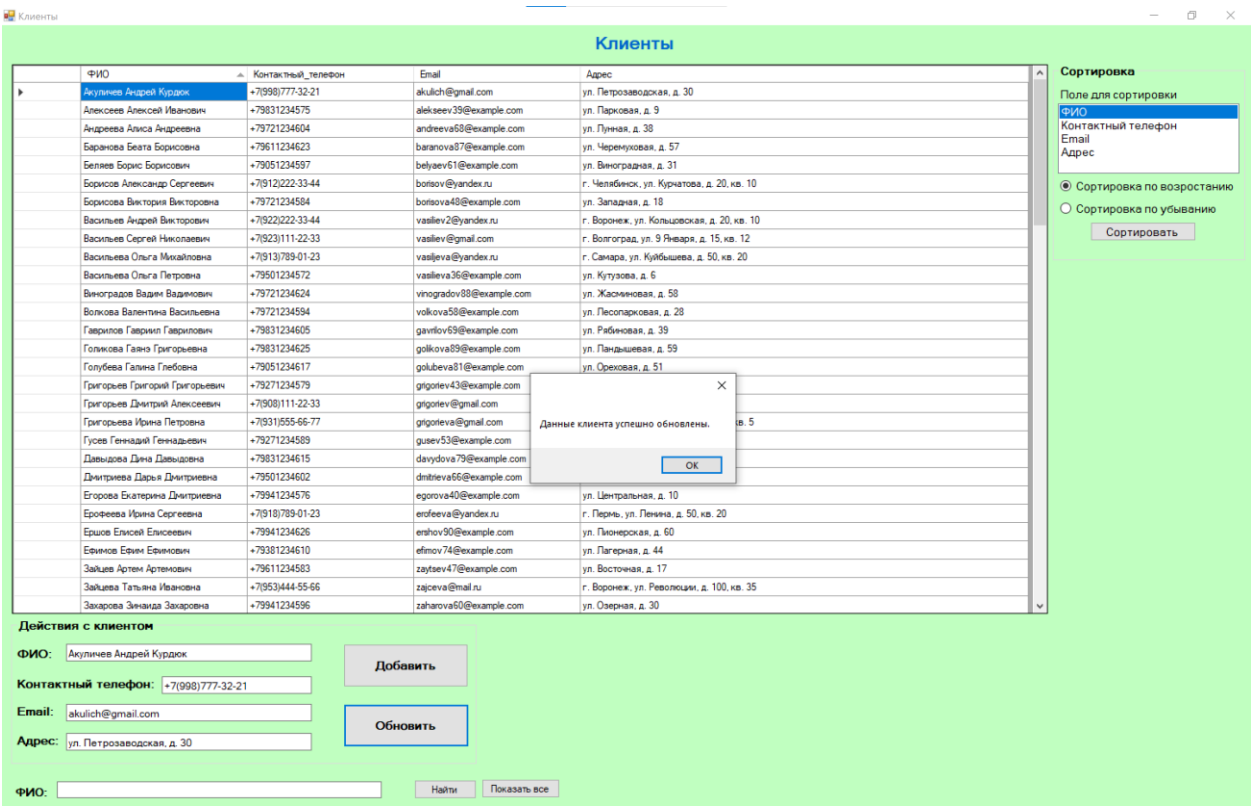

Рис. 2.22. Успешное обновление данных клиента

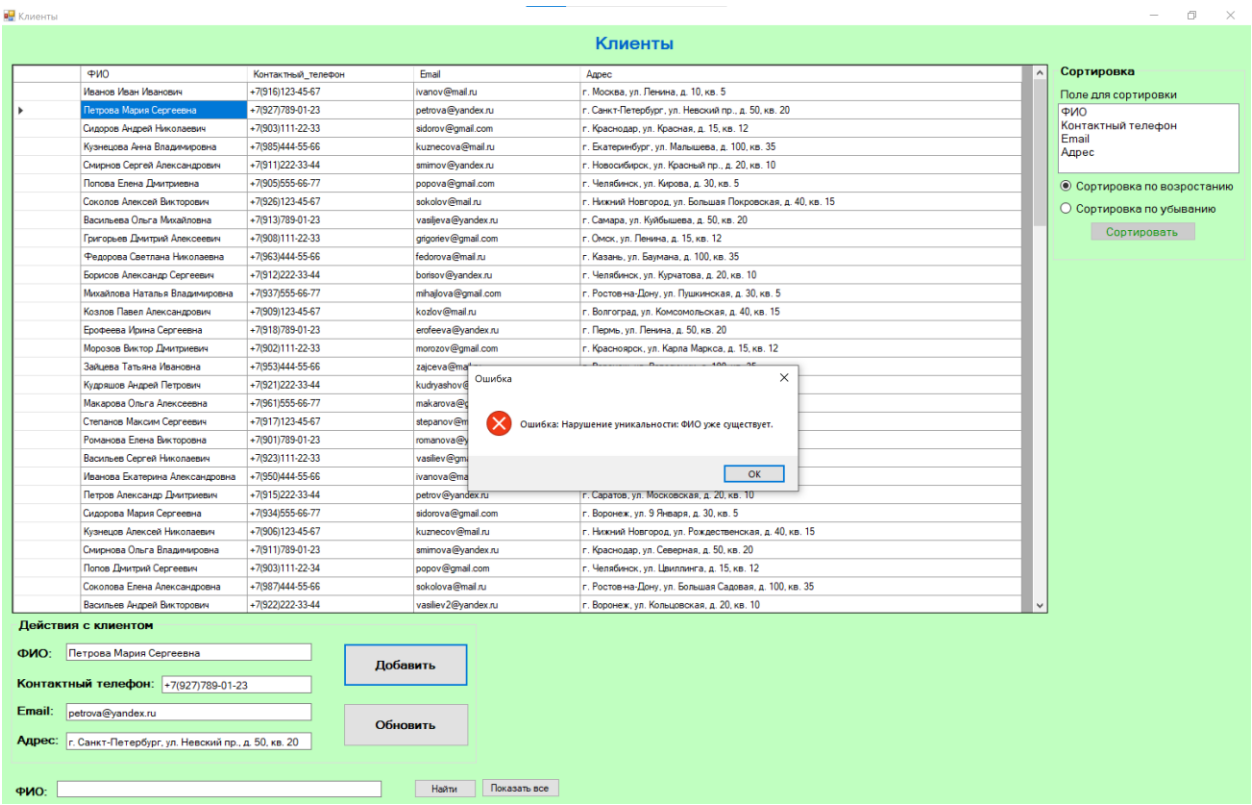

Рис. 2.23. Ошибка при добавлении клиента, клиент с данным ФИО уже существует

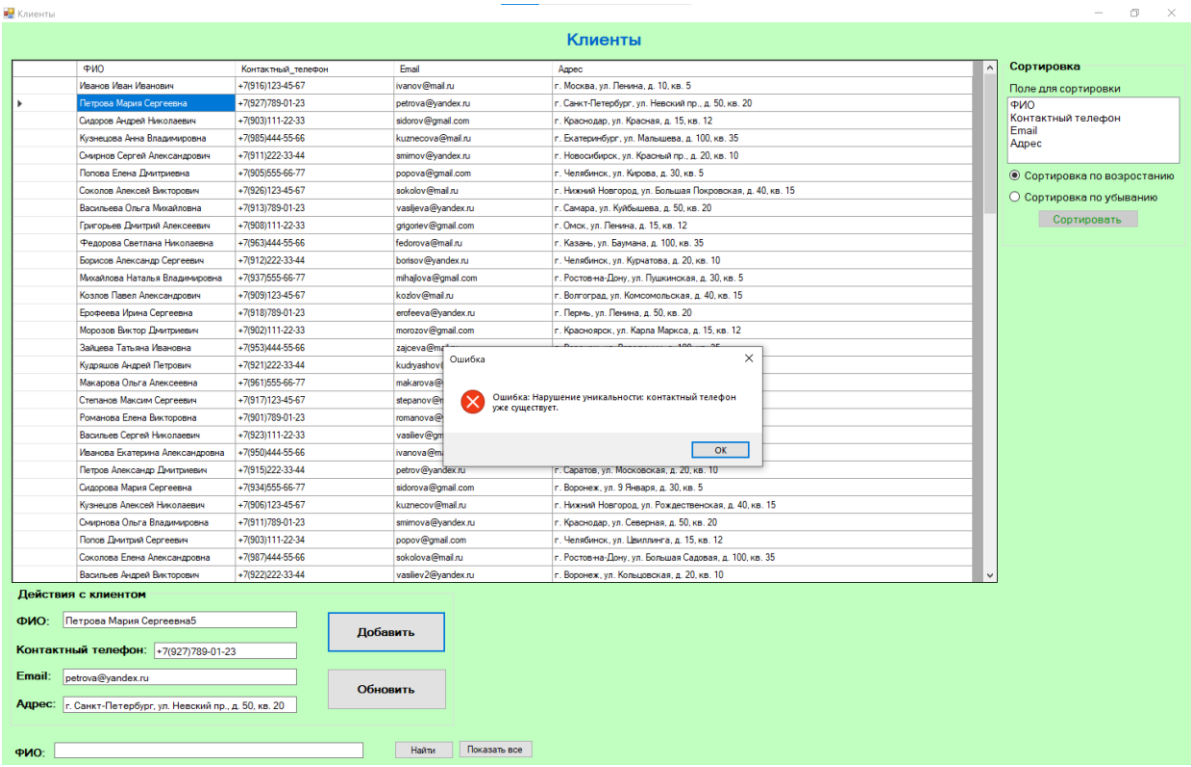

Рис. 2.24. Ошибка при добавлении клиента, клиент с данным номером телефона уже существует

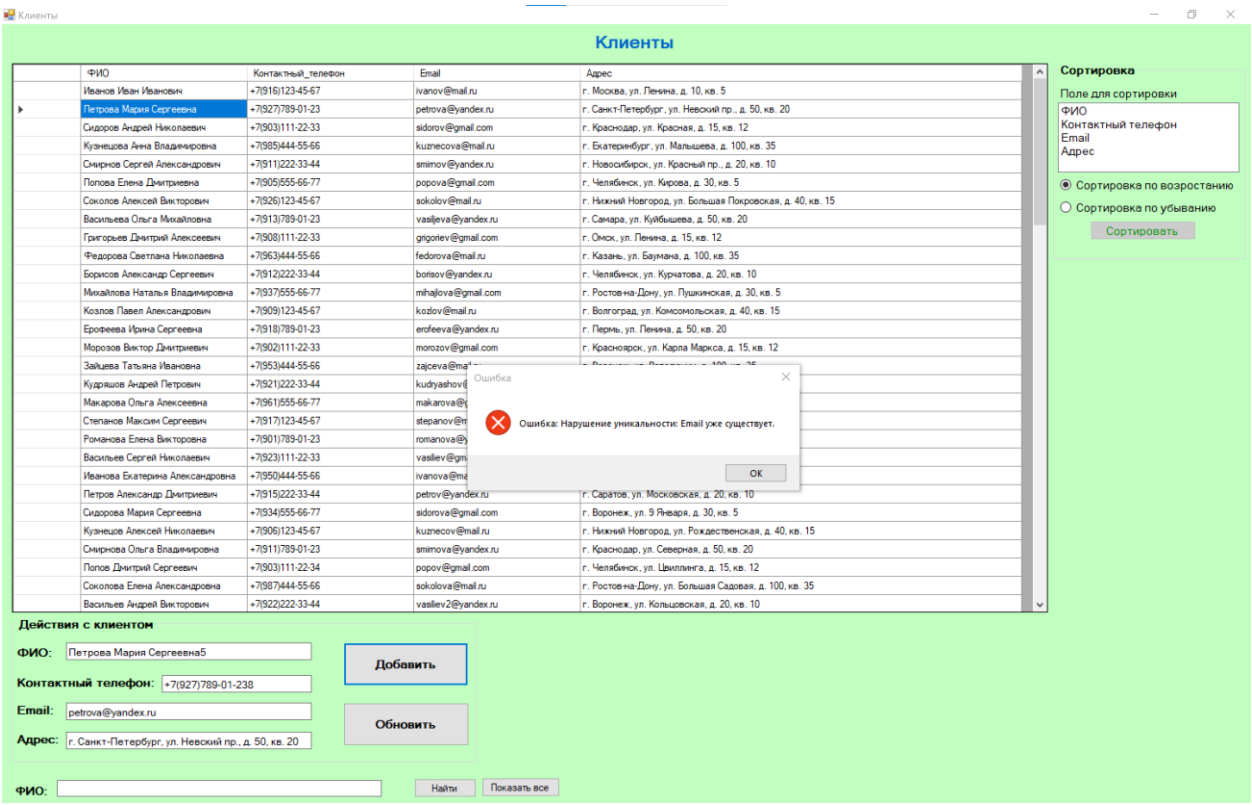

Рис. 2.25. Ошибка при добавлении клиента, клиент с данным адресом электронной почты уже существует

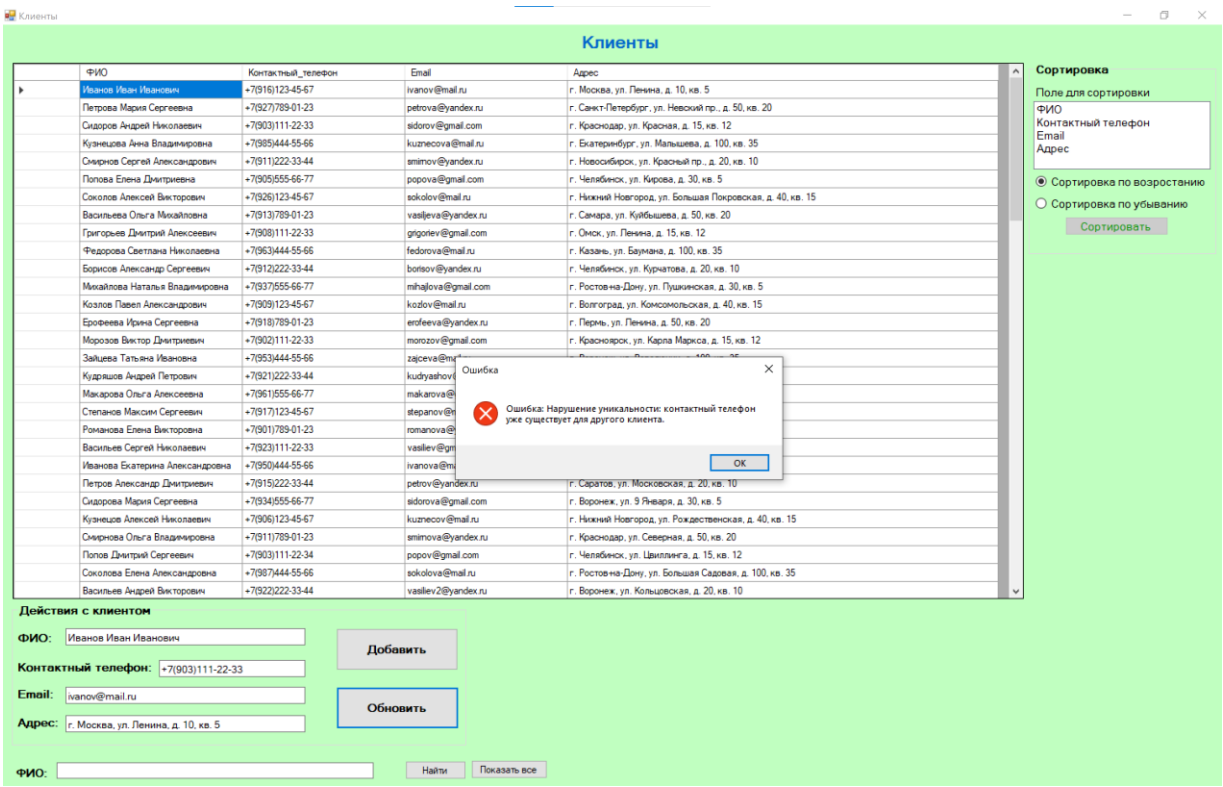

Рис. 2.26. Ошибка при обновлении клиента, клиент с данным номером телефона уже существует

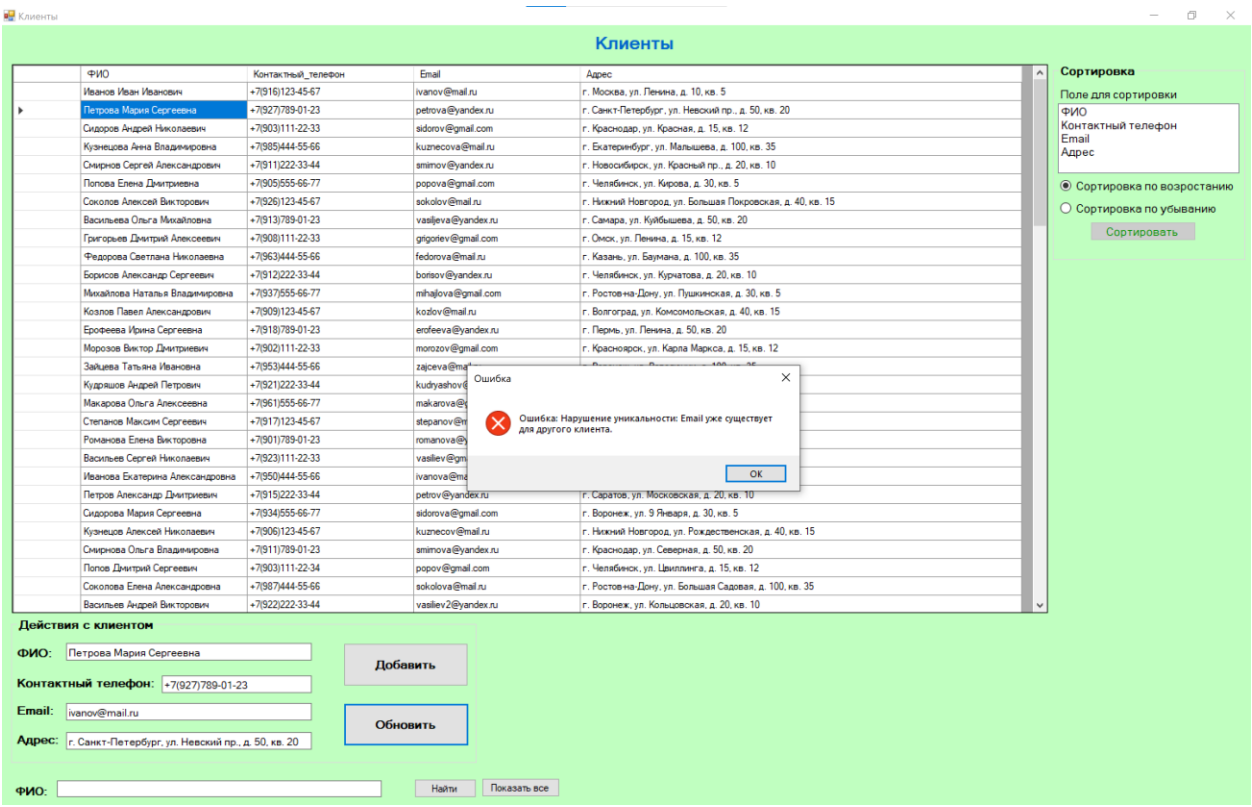

Рис. 2.27. Ошибка при обновлении клиента, клиент с таким адресом электронной почты уже существует

Теперь перейдем к рассмотрению формы График (рис. 2.28). Доступ к ней имеет администратор, по кнопке «Работа с сотрудниками», и генеральный директор, по кнопке «Просмотр и обновление графиков». На данной форме реализован поиск по Id_Графика. Используя данную форму, пользователь имеет возможность добавить новый график (рис. 2.29), обновить (рис. 2.30) или удалить существующий (рис. 2.31). На рис. 2.32-2.36 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

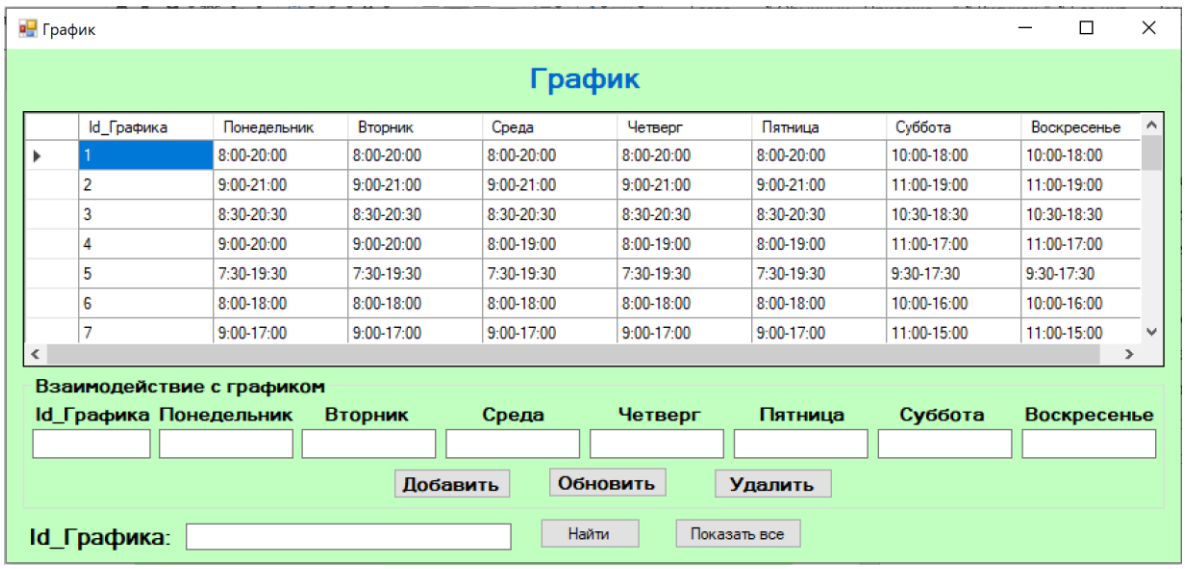

Рис. 2.28. Форма График

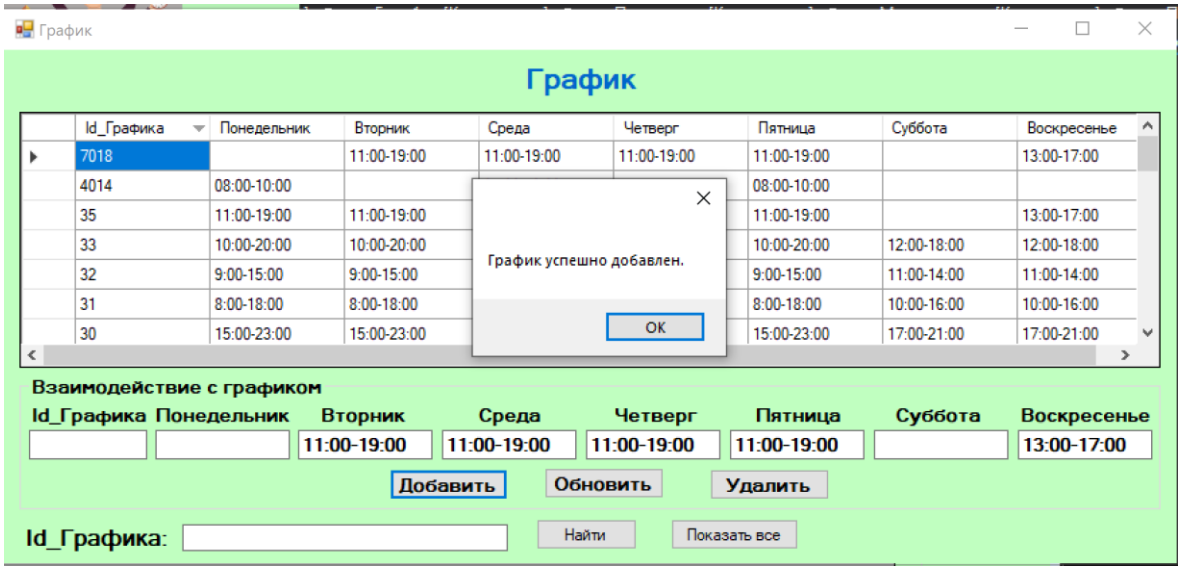

Рис. 2.29. Успешное добавление нового графика

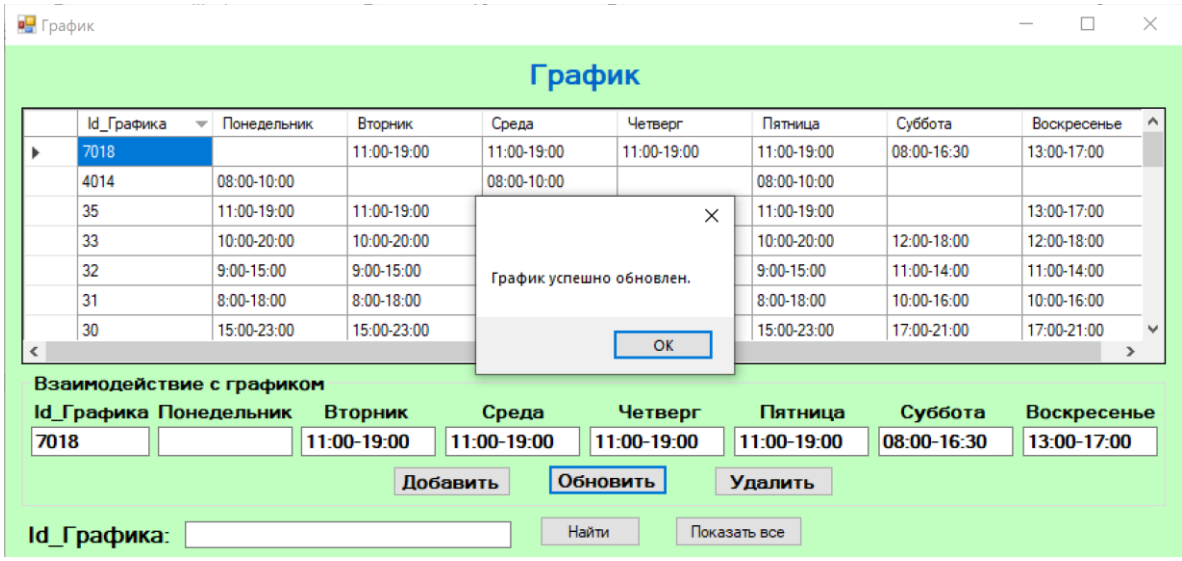

Рис. 2.30. Успешное обновление графика

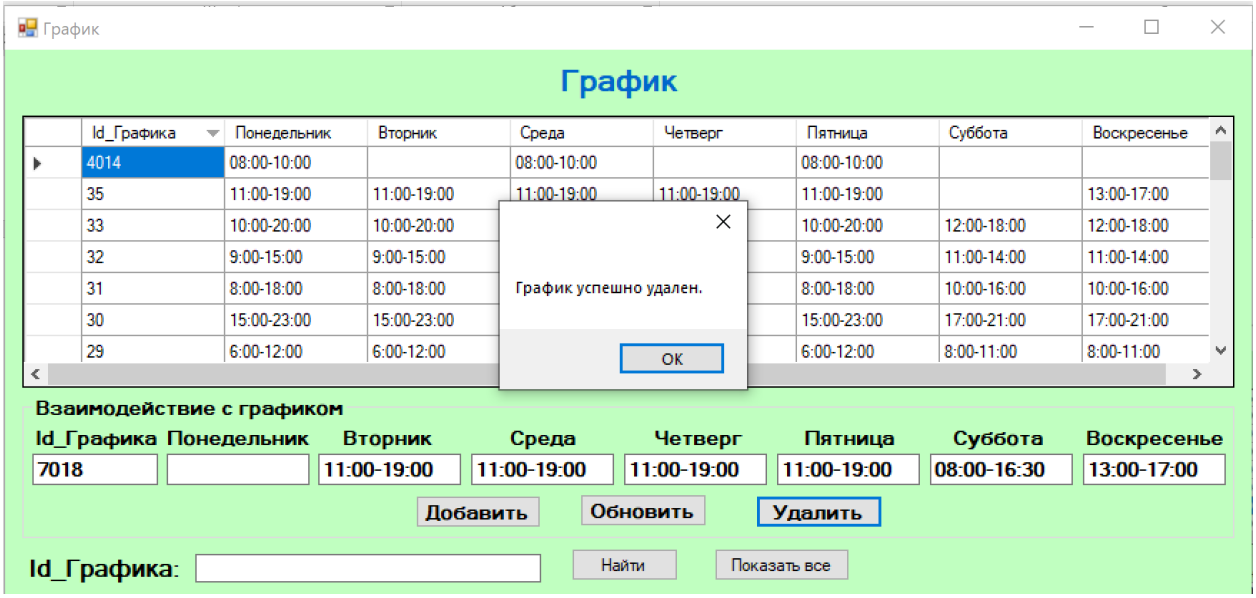

Рис. 2.31. Успешное удаление графика

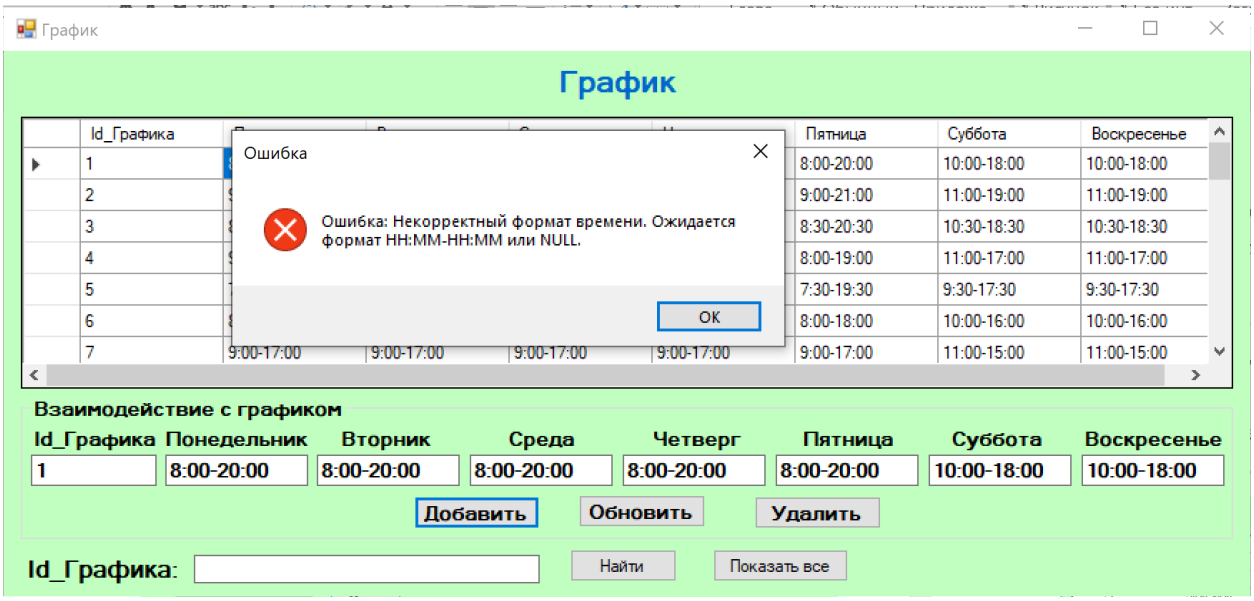

Рис. 2.32. Ошибка при добавлении графика, некорректный формат ввода

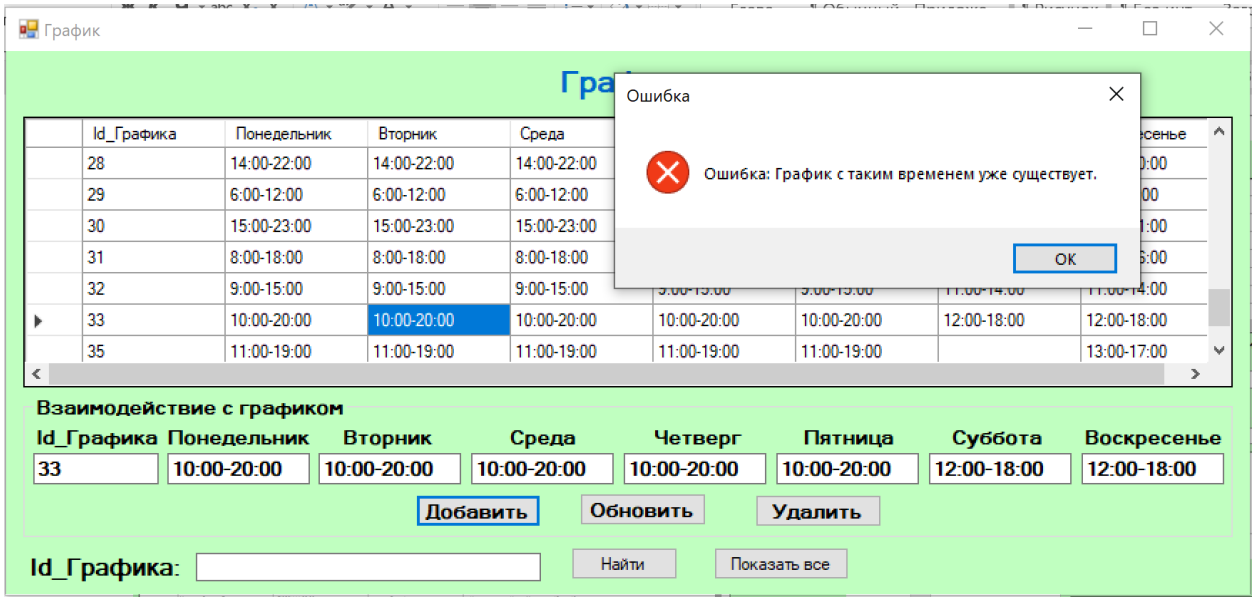

Рис. 2.33. Ошибка при добавлении графика, такой график уже существует

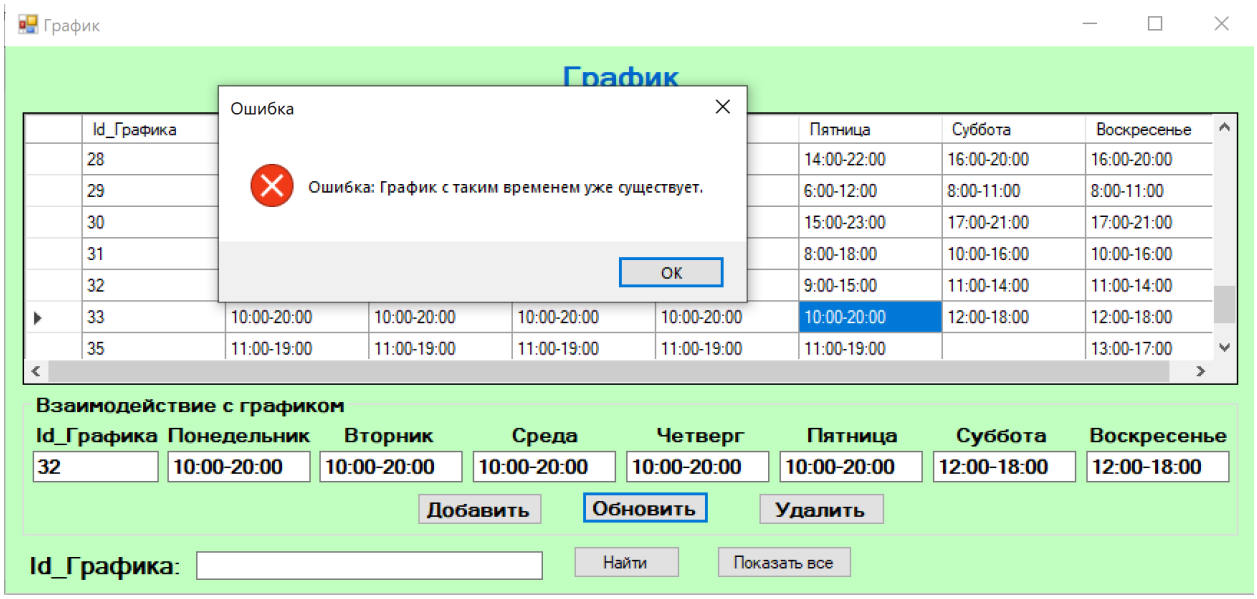

Рис. 2.34. Ошибка при обновлении графика, график с таким временем уже существует

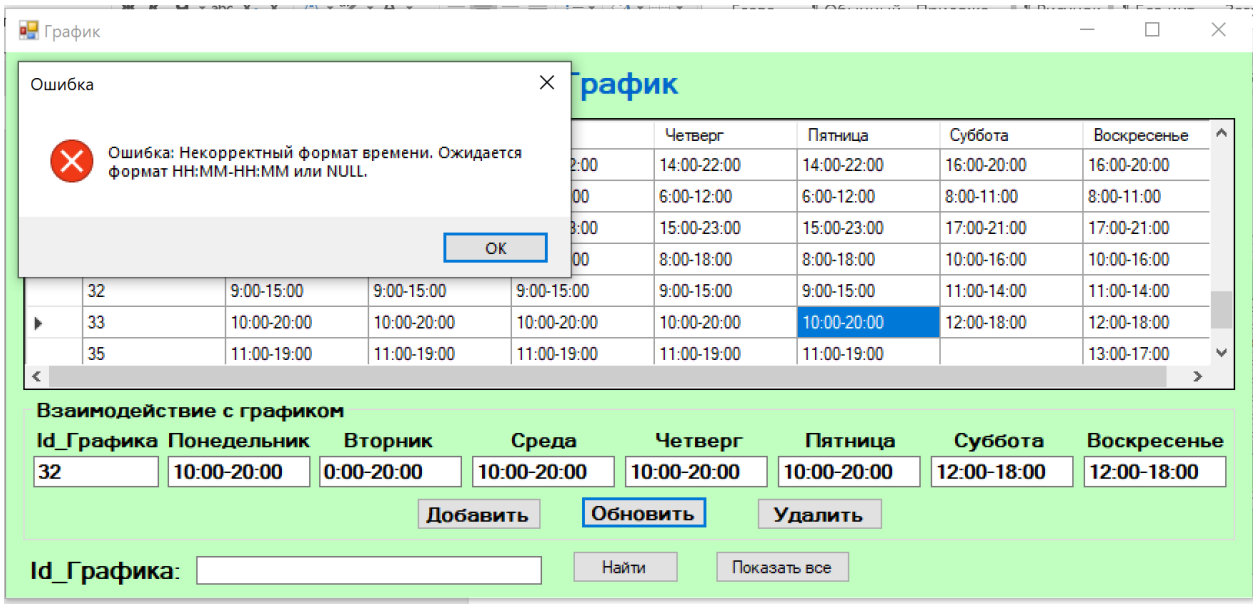

Рис. 2.35. Ошибка при обновлении графика, некорректный формат ввода

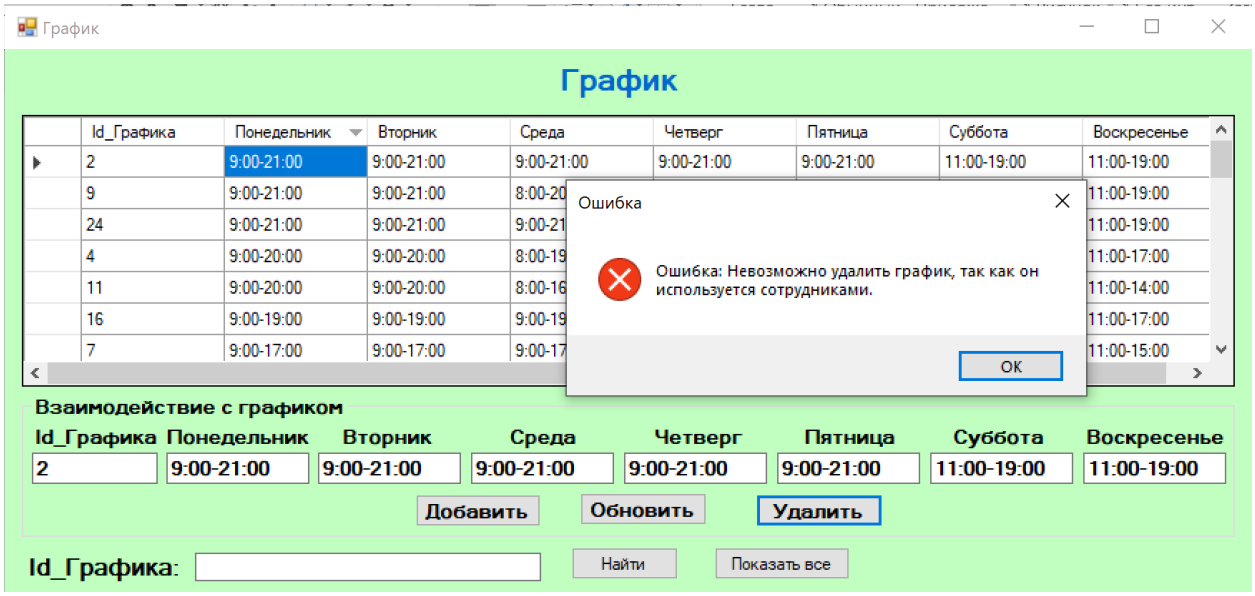

Рис. 2.36. Ошибка при удалении графика, график используется сотрудником

Теперь перейдем к рассмотрению формы Услуги (рис. 2.37). Доступ к ней имеет генеральный директор, по кнопке «Добавление услуги». На данной форме реализована сортировка данных, а также поиск по Названию. Используя данную форму, пользователь имеет возможность добавить новую услугу (рис. 2.38). На рис. 2.39-2.40 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

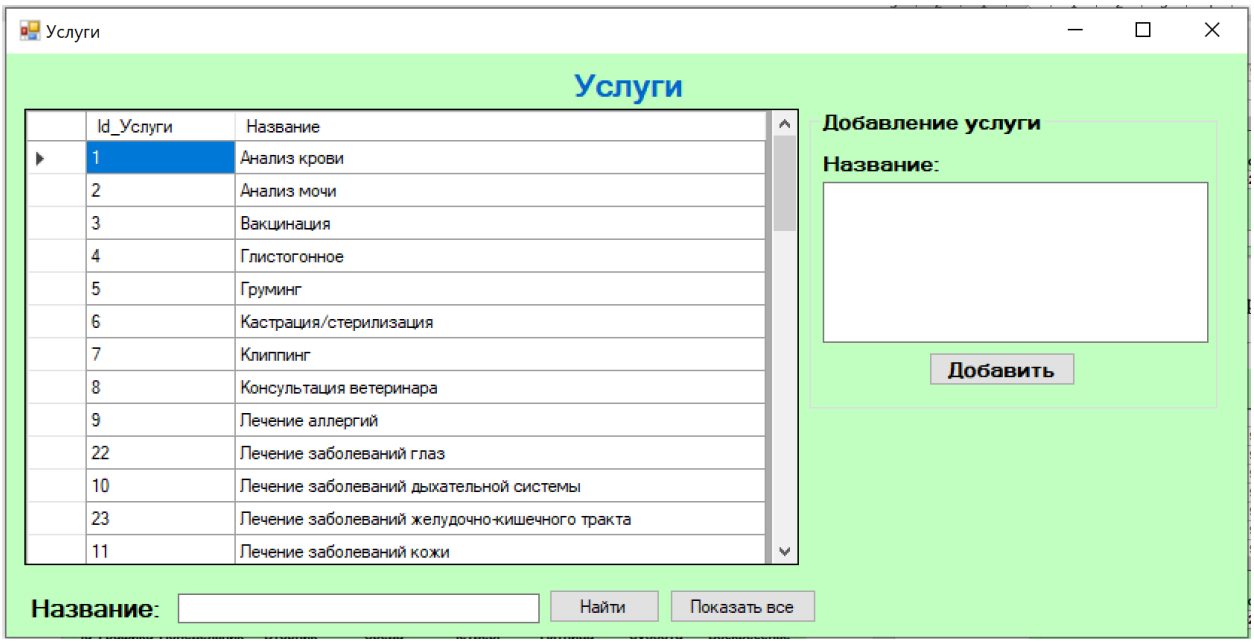

Рис. 2.37. Форма Услуги

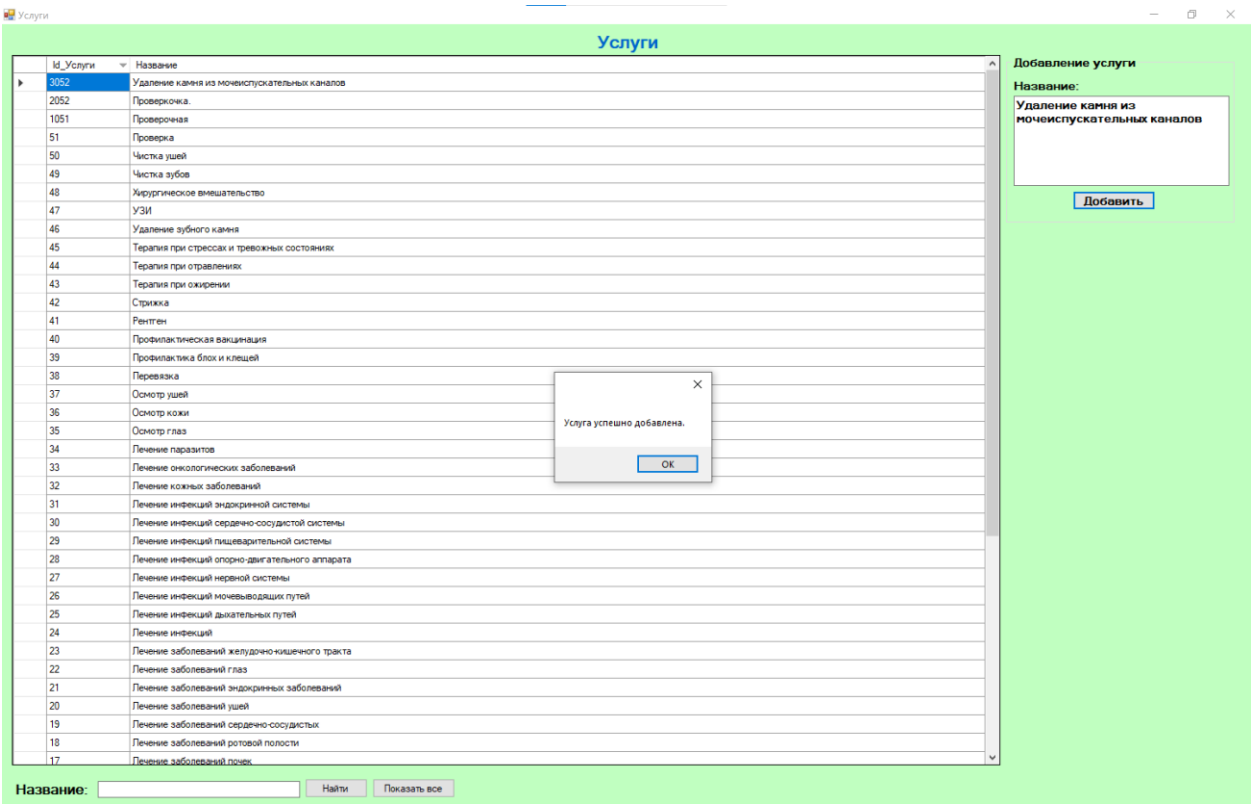

Рис. 2.38. Успешное добавление услуги

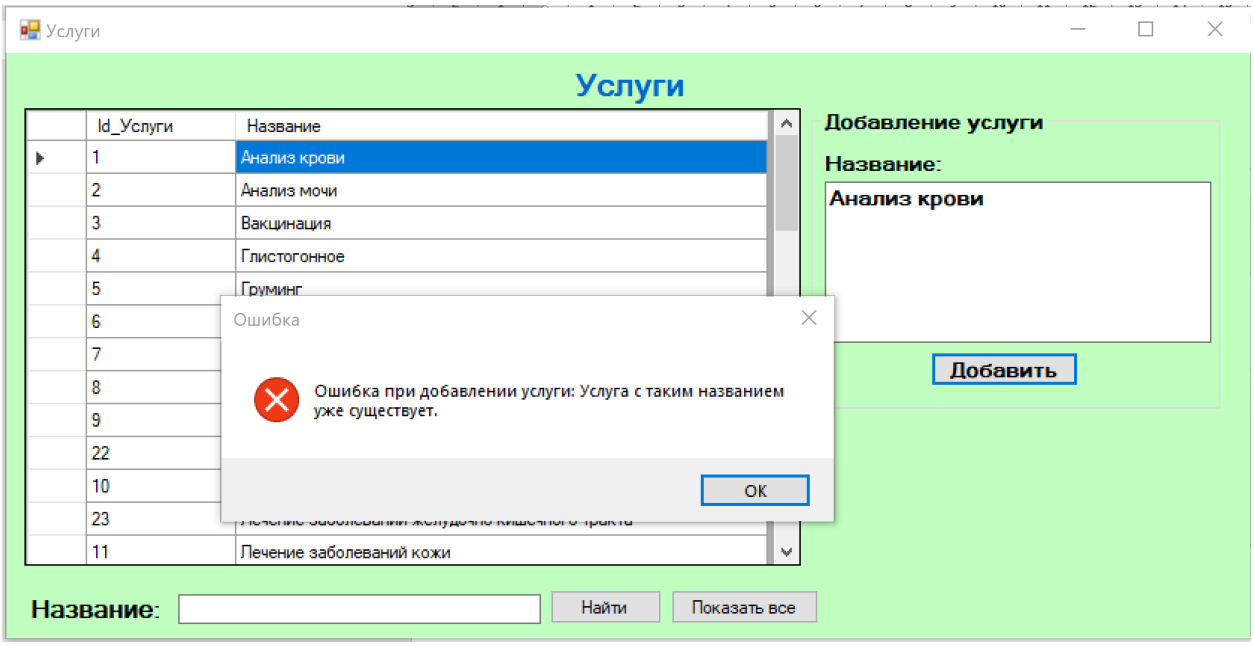

Рис. 2.39. Ошибка при добавлении услуги, услуга с таким названием уже существует

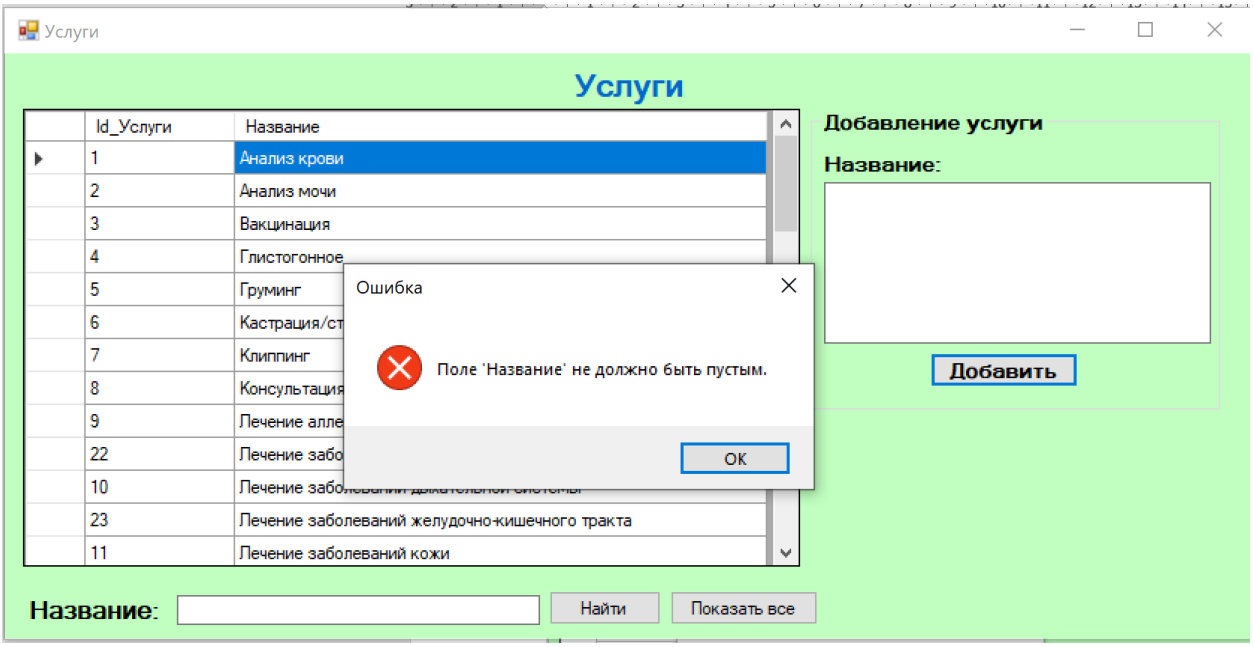

Рис. 2.40. Ошибка при добавлении услуги, поле Название не должно быть пустым

Теперь перейдем к рассмотрению формы СкладскиеОперации (рис. 2.41). Доступ к ней имеет генеральный директор, по кнопке «Просмотр складских операций», и менеджер по запасам, по кнопке «Добавить складскую операцию». На данной форме реализован поиск по

Наименованию_Медикамента, а также сортировка. Также использованы выпадающие списки для полей ФИО_Сотрудника и Операция, кроме того используются элементы numericUpDown для полей Id_Приема и Количество единиц, отдельный dataGried для отображения рабочего времени каждого из сотрудников. Используя данную форму, пользователь имеет возможность добавить новую складскую операцию (рис. 2.42). На рис. 2.43-2.48 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

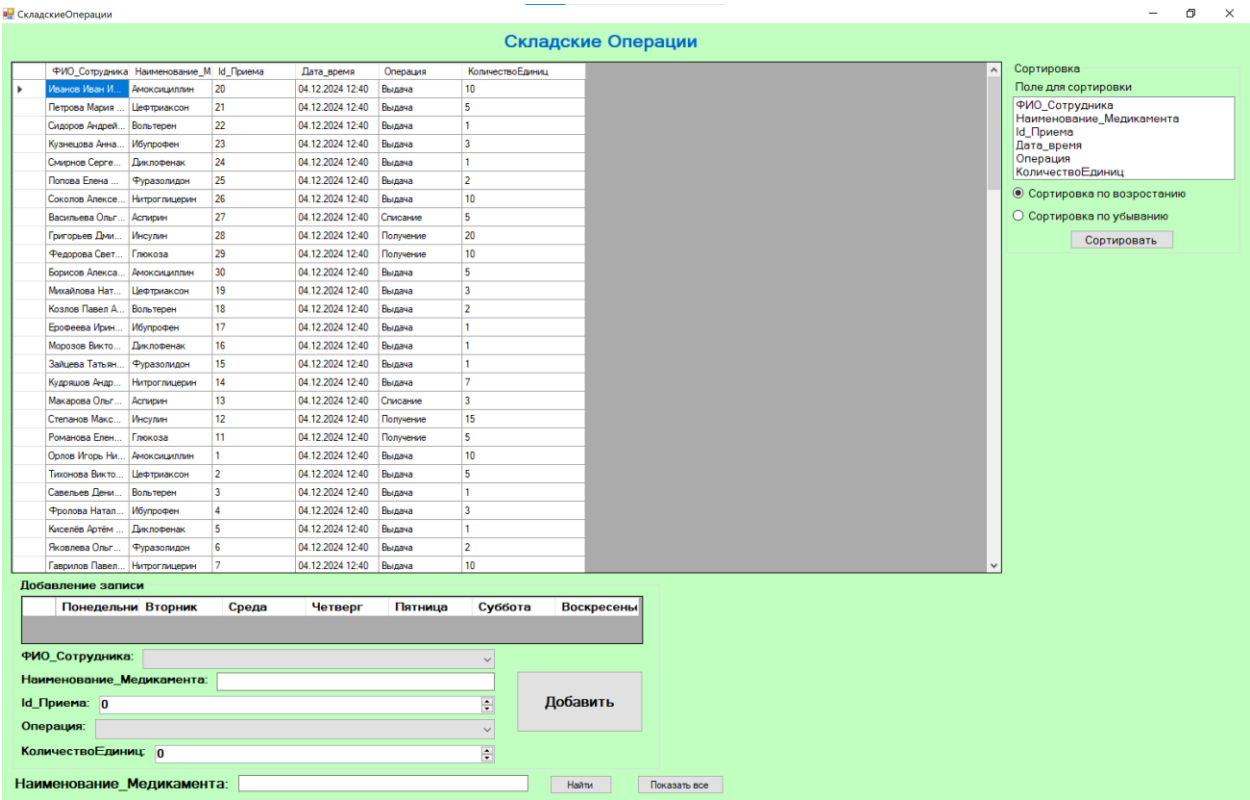

Рис. 2.41. Форма СкладскиеОперации

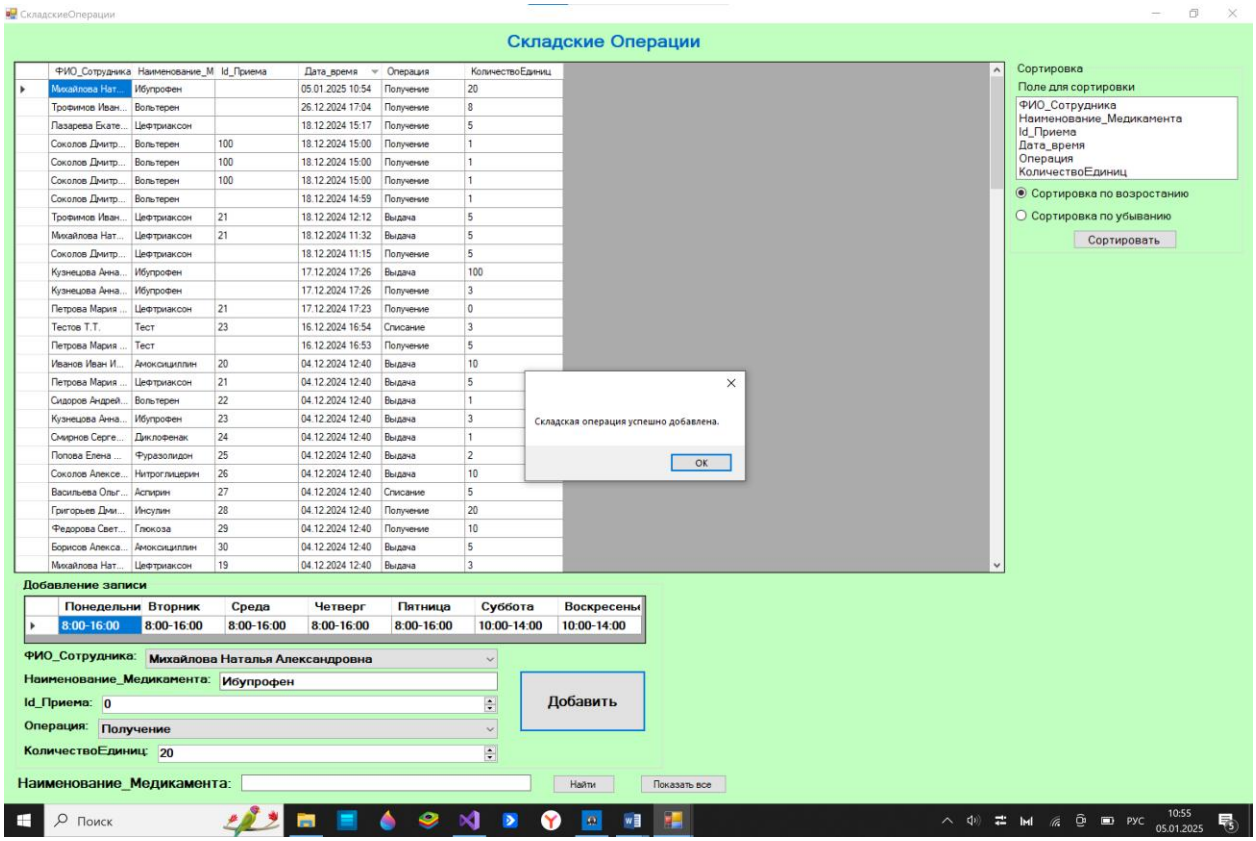

Рис. 2.42. Успешное добавление складской операции

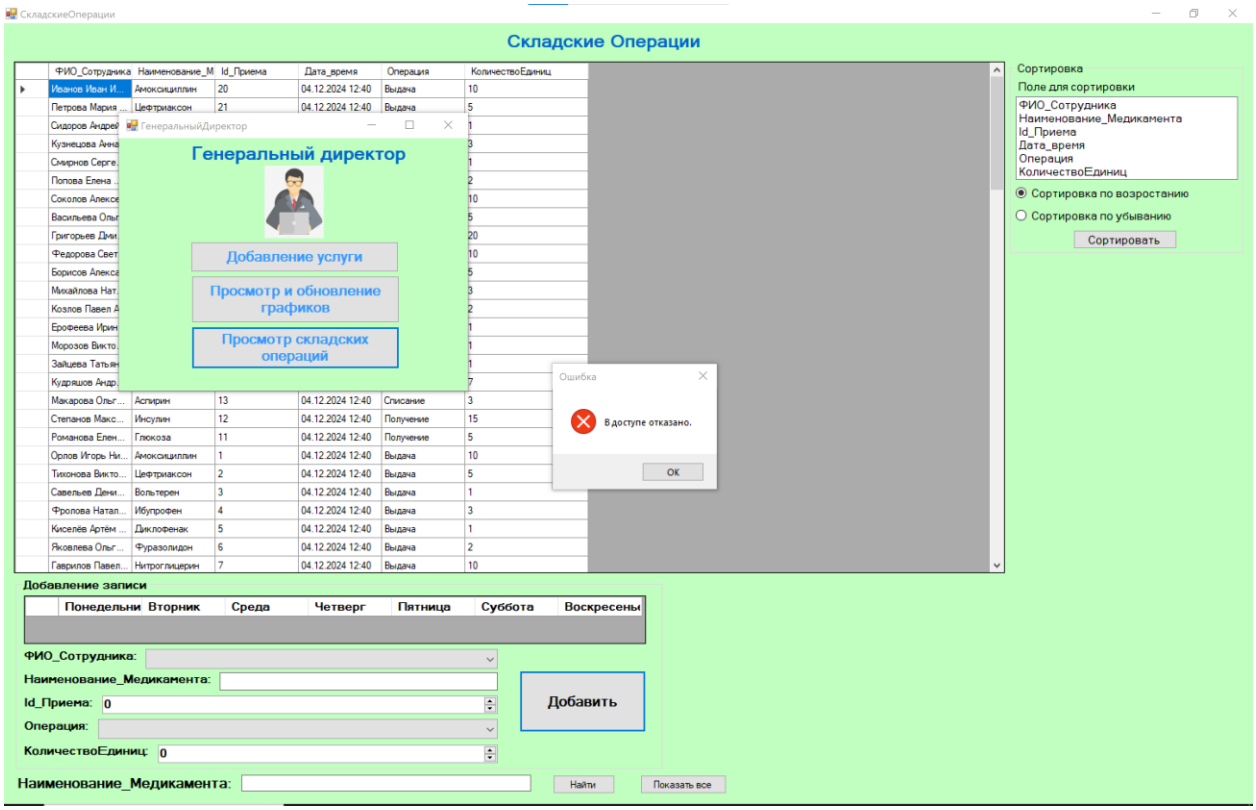

Рис. 2.43. Ошибка при добавлении складской операции, отказано в доступе

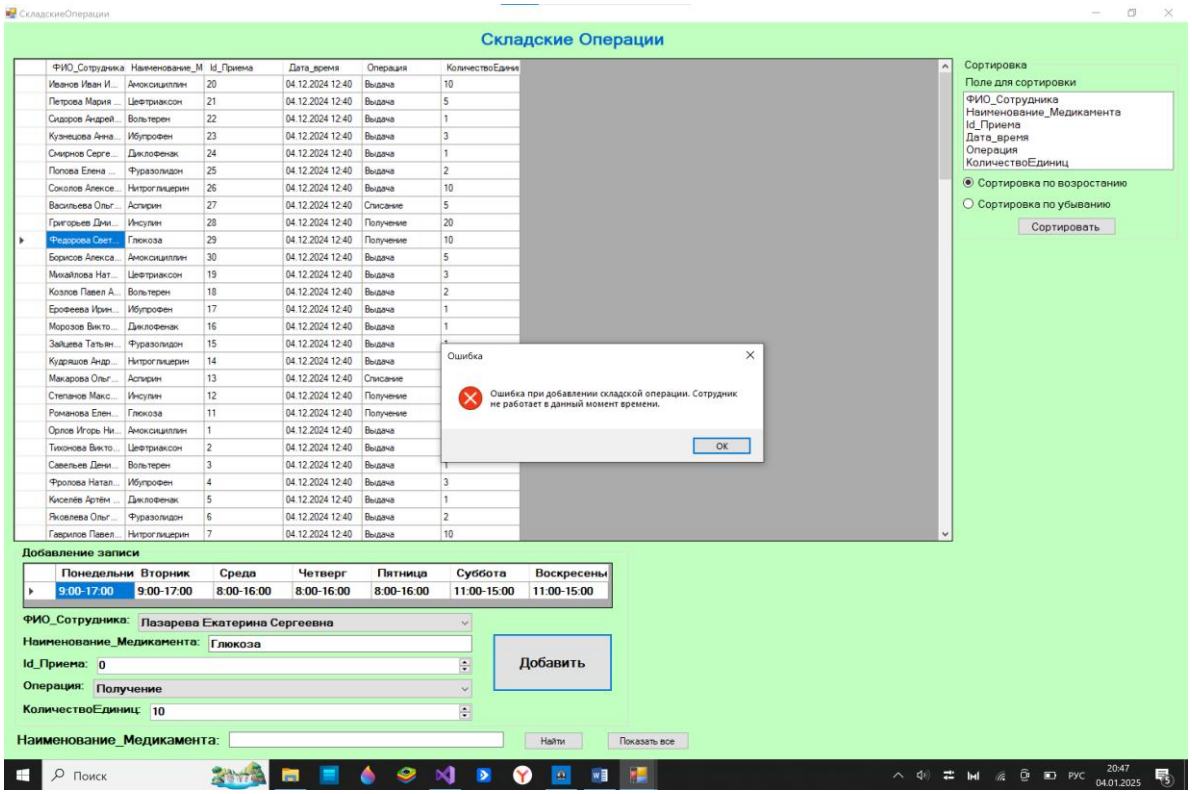

Рис. 2.44. Ошибка при добавлении складской операции, время не совпадает с графиком работы сотрудника

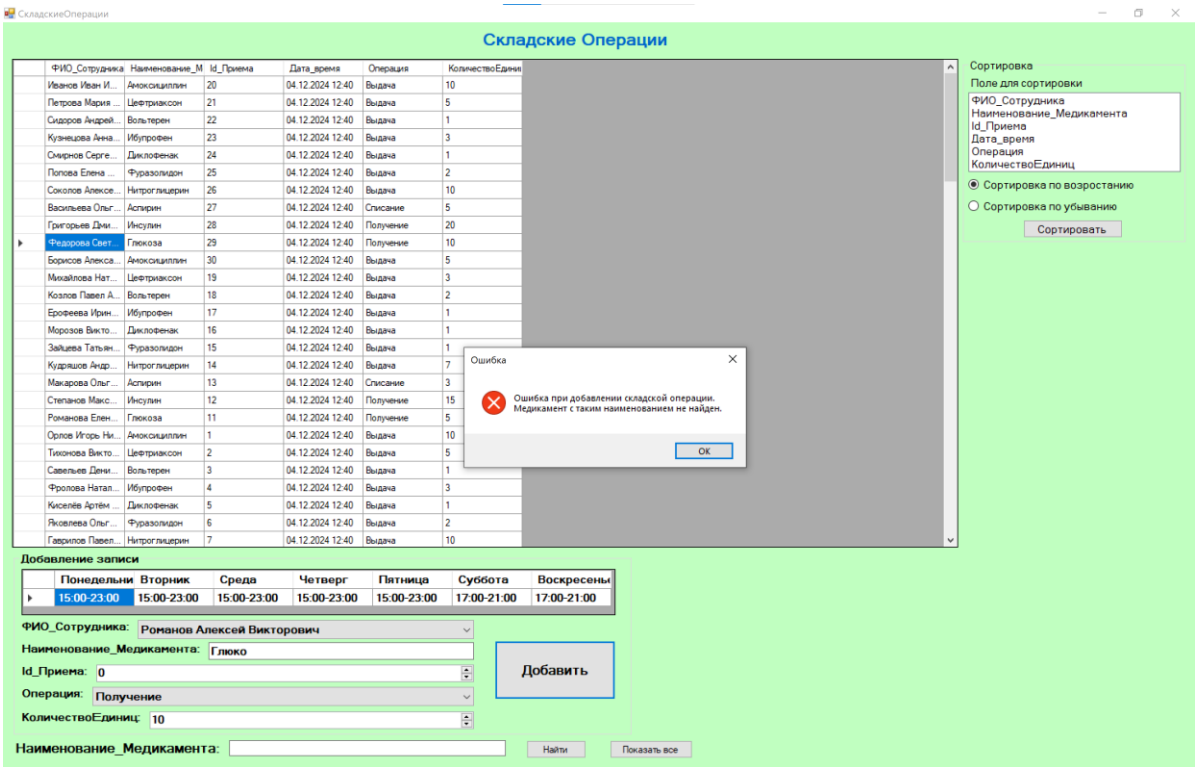

Рис. 2.45. Ошибка при добавлении складской операции, медикамент не найден

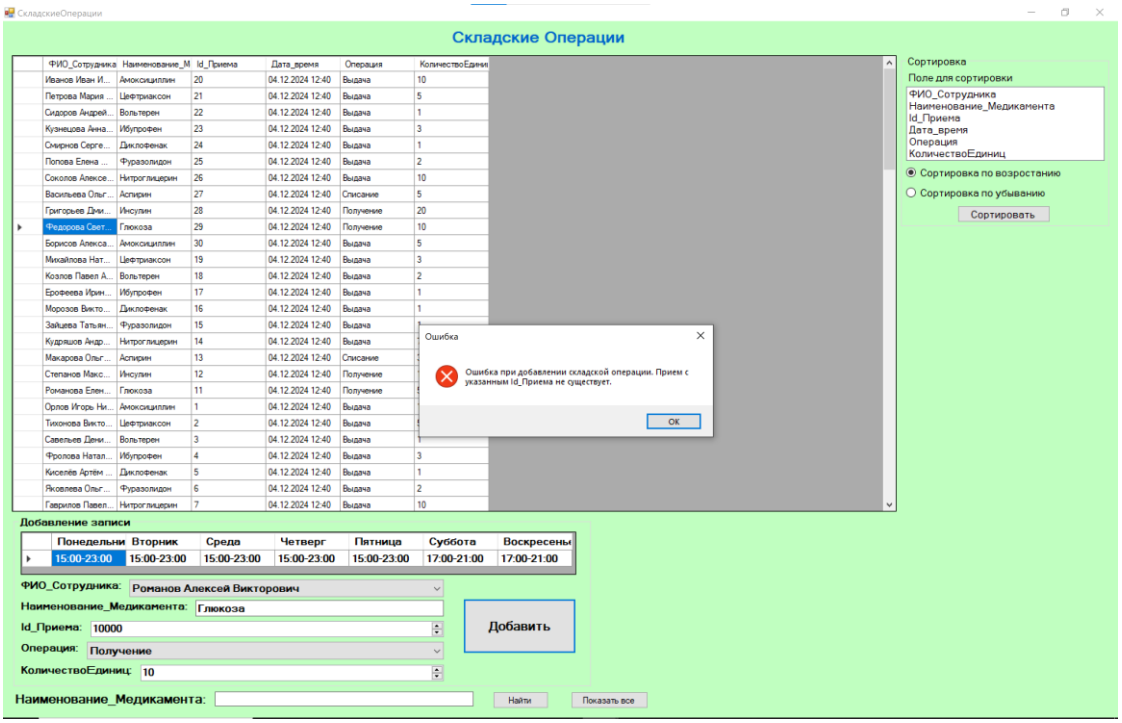

Рис. 2.46. Ошибка при добавлении складской операции, приема с указанным Id_Приема не существует

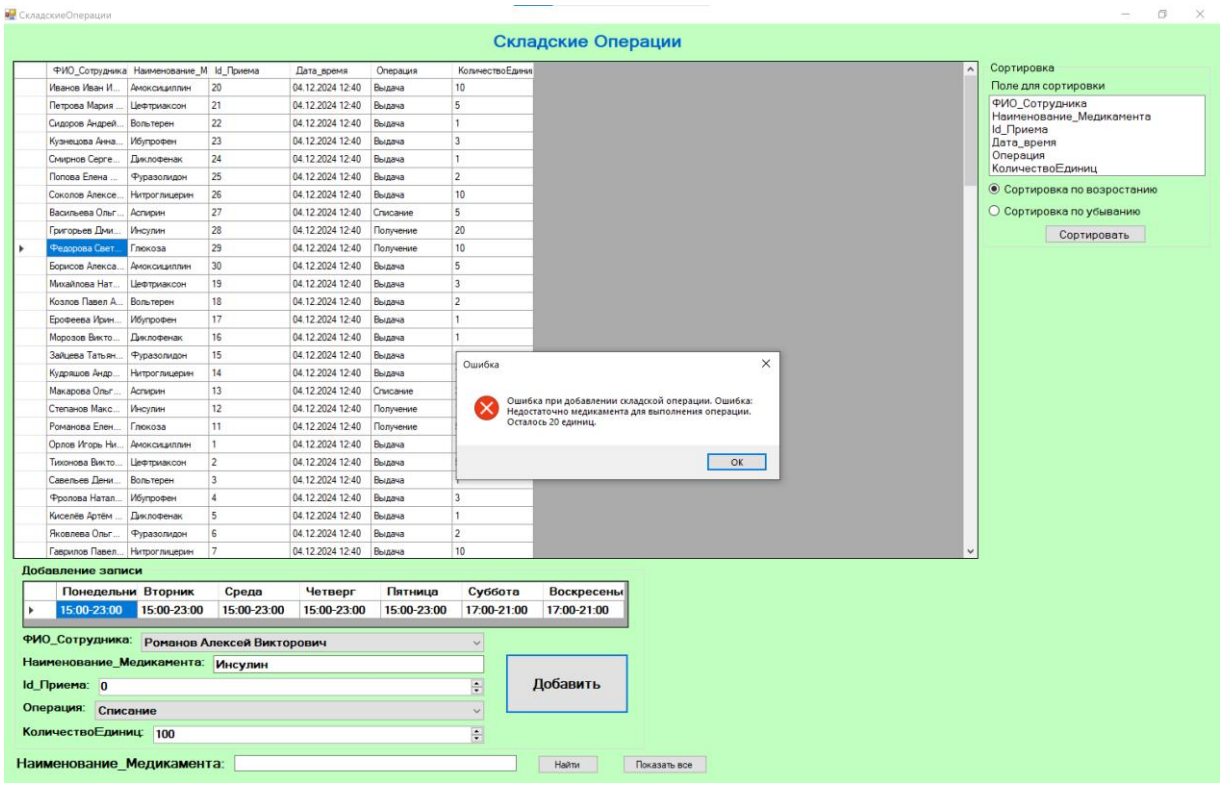

Рис. 2.47. Ошибка при добавлении складской операции, недостаточно медикамента на складе

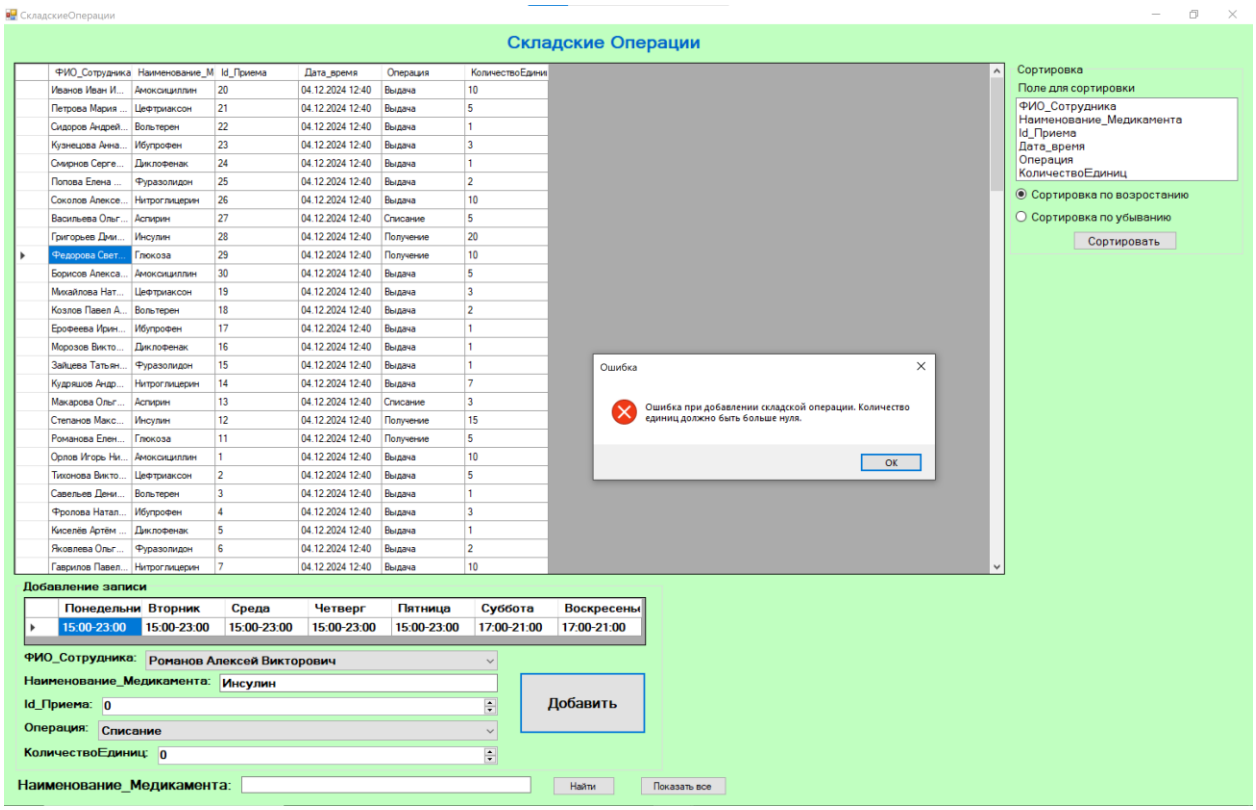

Рис. 2.48. Ошибка при добавлении складской операции, количество единиц должно быть больше нуля

Теперь перейдем к рассмотрению формы Медикамент (рис. 2.49). Доступ к ней имеет менеджер по запасам, по кнопке «Редактировать медикаменты». На данной форме реализован поиск по Наименованию, а также сортировка. Также использован выпадающий список для поля ЕдиницаИзмерения. Используя данную форму, пользователь имеет возможность добавить новый медикамент (рис. 2.50) или обновить имеющийся (рис. 2.51). На рис. 2.52-2.54 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

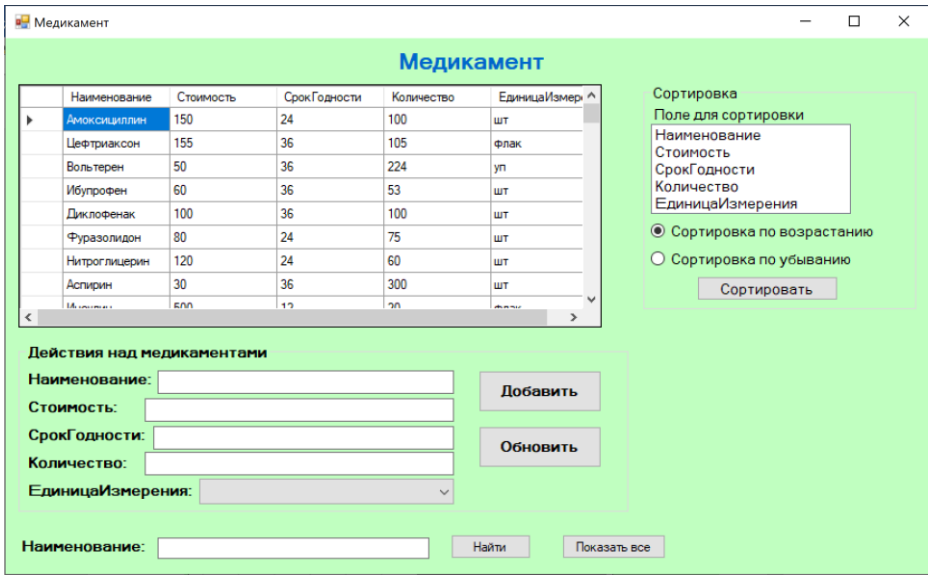

Рис. 2.49. Форма Медикамент

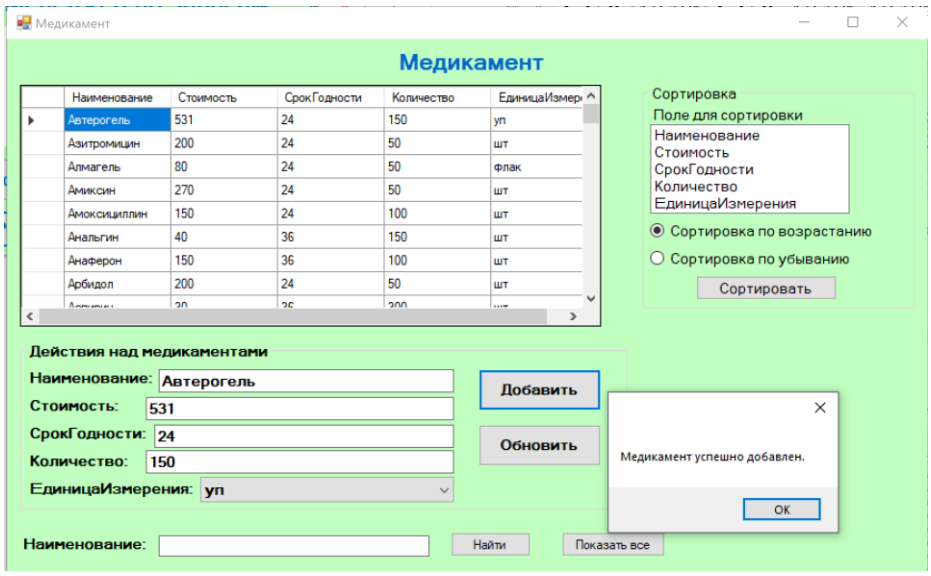

Рис. 2.50. Успешное добавление нового медикамента

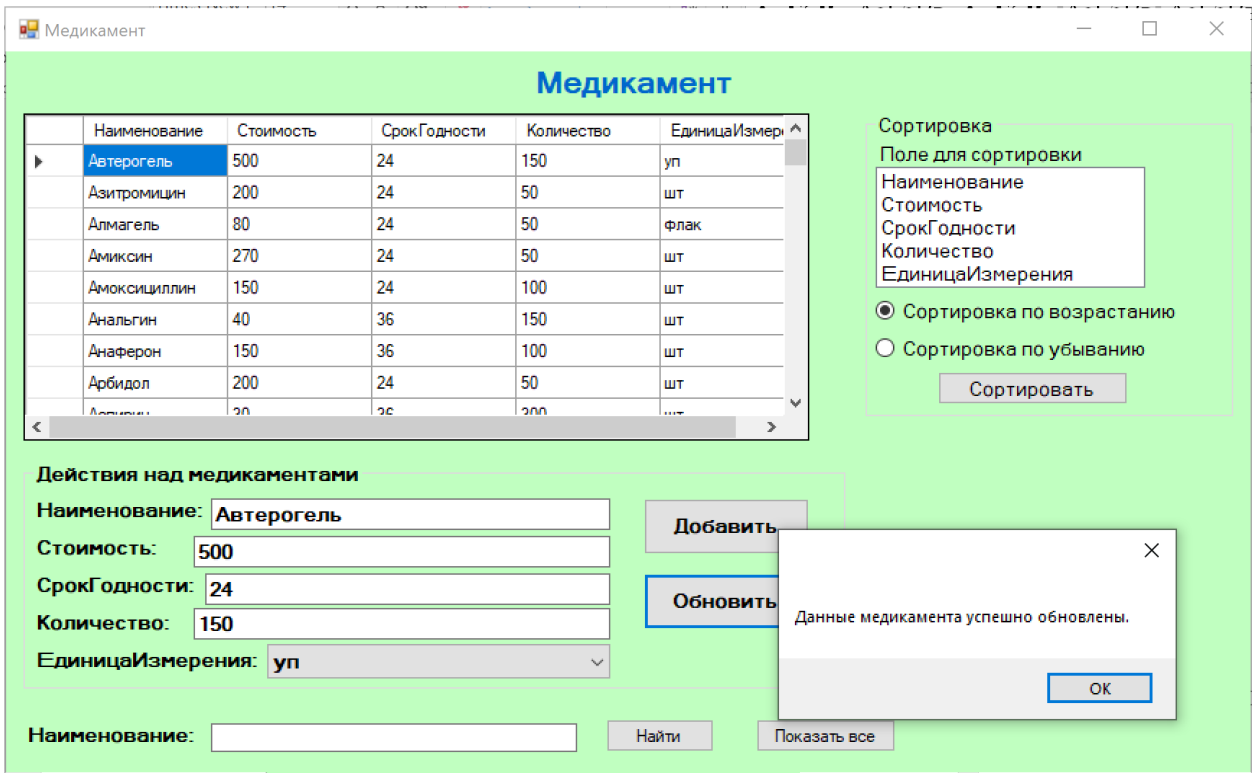

Рис. 2.51. Успешное обновление данных о медикаменте

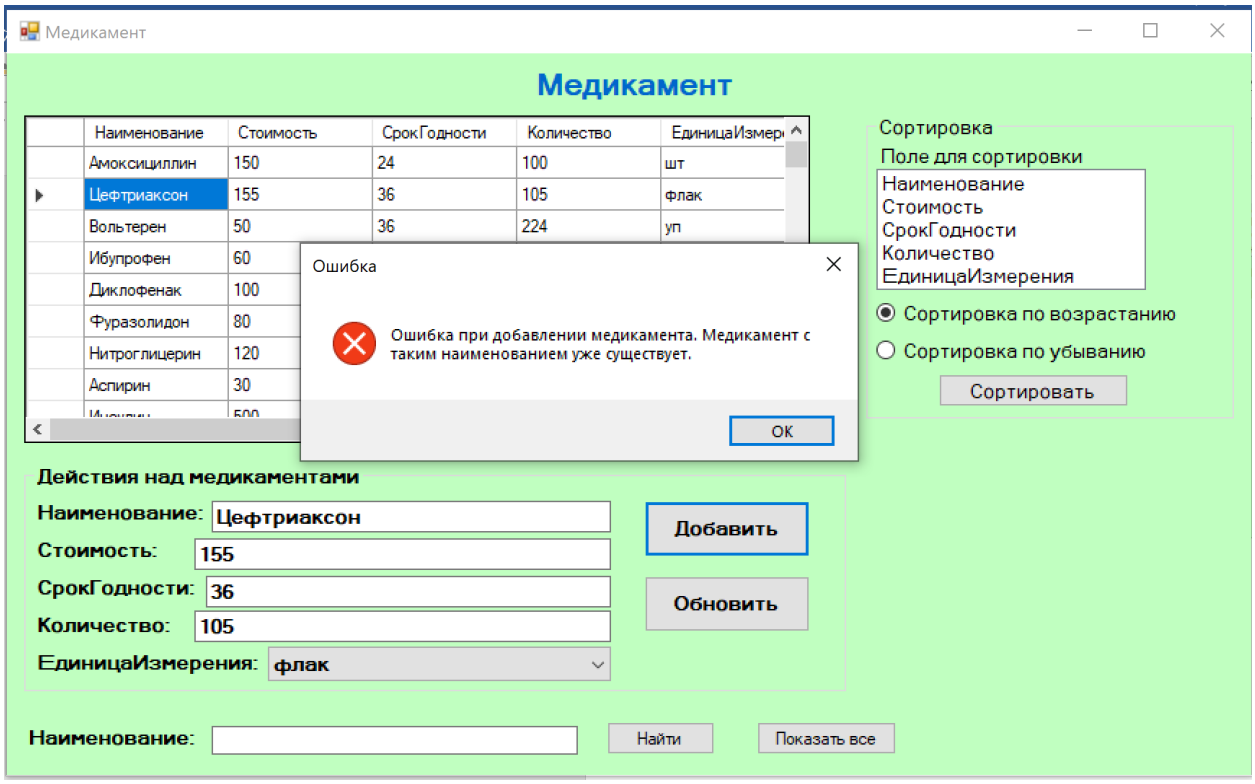

Рис. 2.52. Ошибка при добавлении медикамента, медикамент с таким наименованием не найден

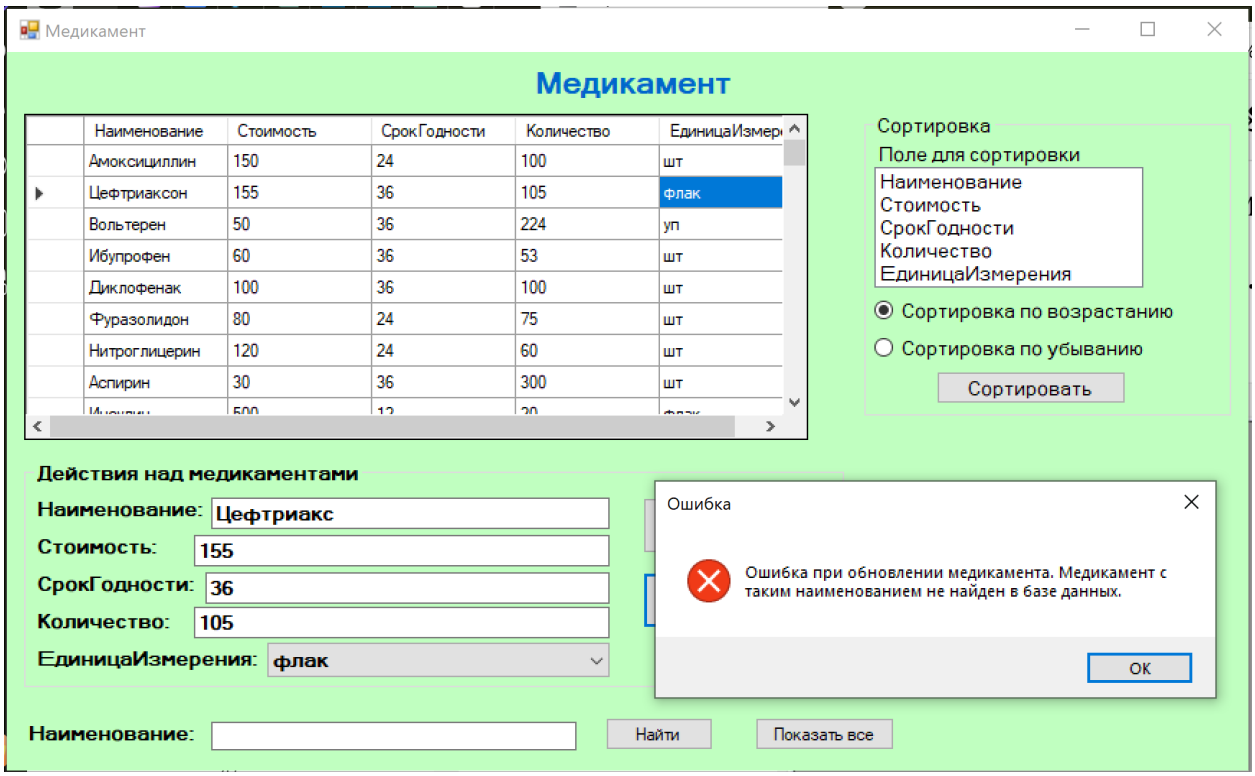

Рис. 2.53. Ошибка при обновлении медикамента, медикамент с таким наименованием не найден

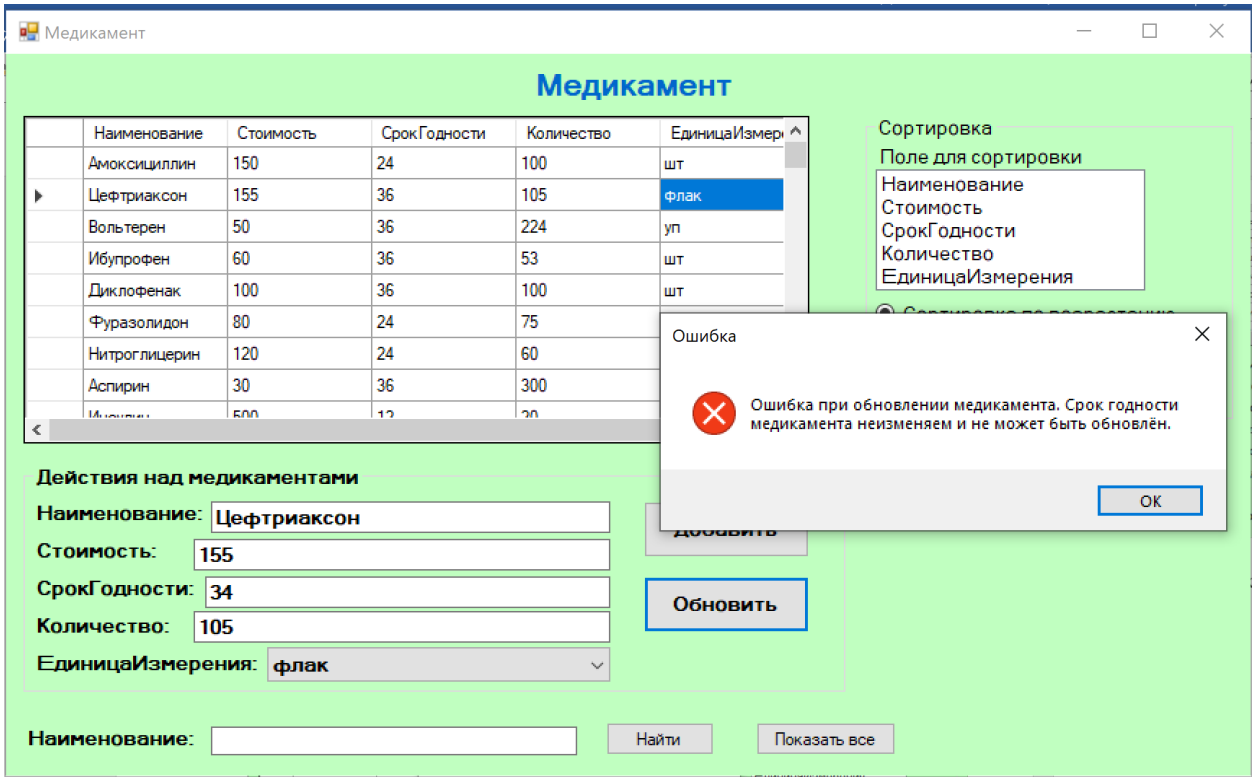

Рис. 2.54. Ошибка при обновлении медикамента, срок годности медикамента не должен быть обновлен

Теперь перейдем к рассмотрению формы Питомцы (рис. 2.55). Доступ к ней имеет ветеринар, по кнопке «Работа с питомцами». На данной форме реализован поиск по Кличке, а также сортировка. Также использован выпадающий список для поля Пол и поле numericUpDown для возраста питомца. Используя данную форму, пользователь имеет возможность добавить нового питомца (рис. 2.56) или обновить возраст имеющегося в базе (рис. 2.57). На рис. 2.58-2.63 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

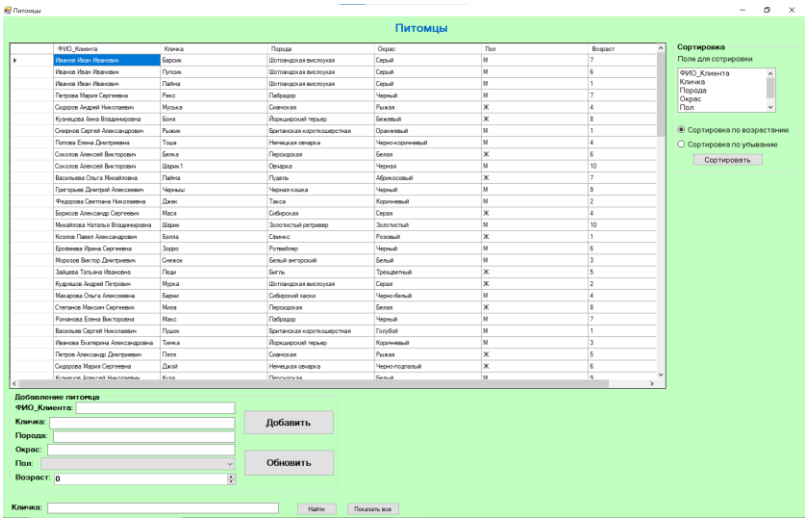

Рис. 2.55. Форма Питомцы

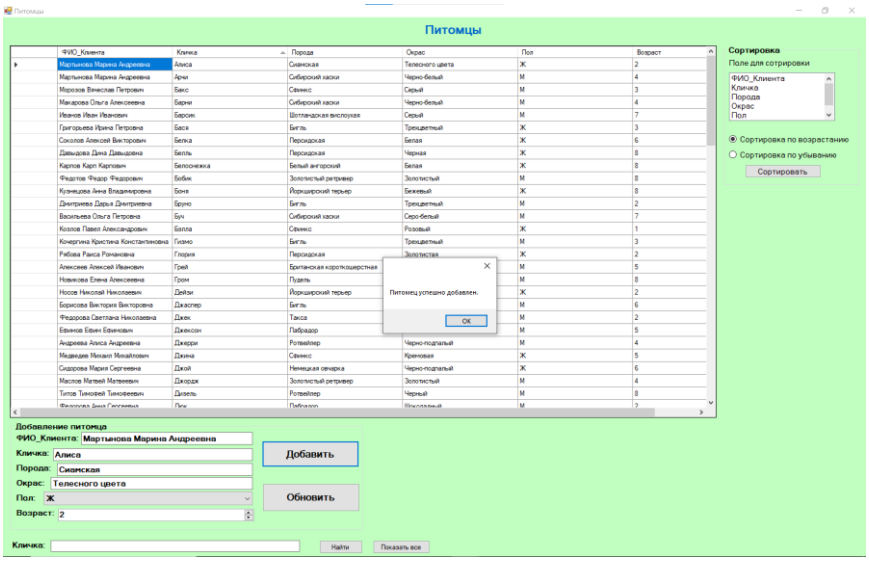

Рис. 2.56. Успешное добавление питомца

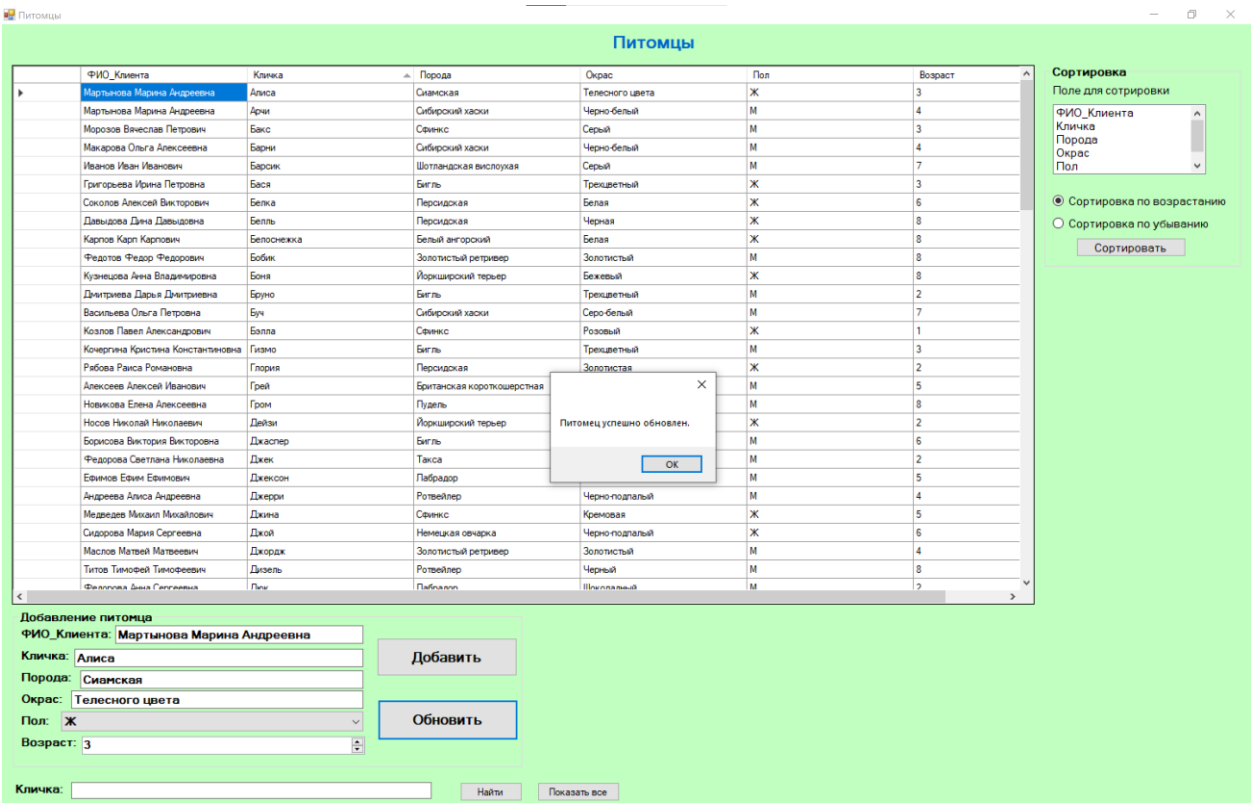

Рис. 2.57. Успешное обновление возраста питомца

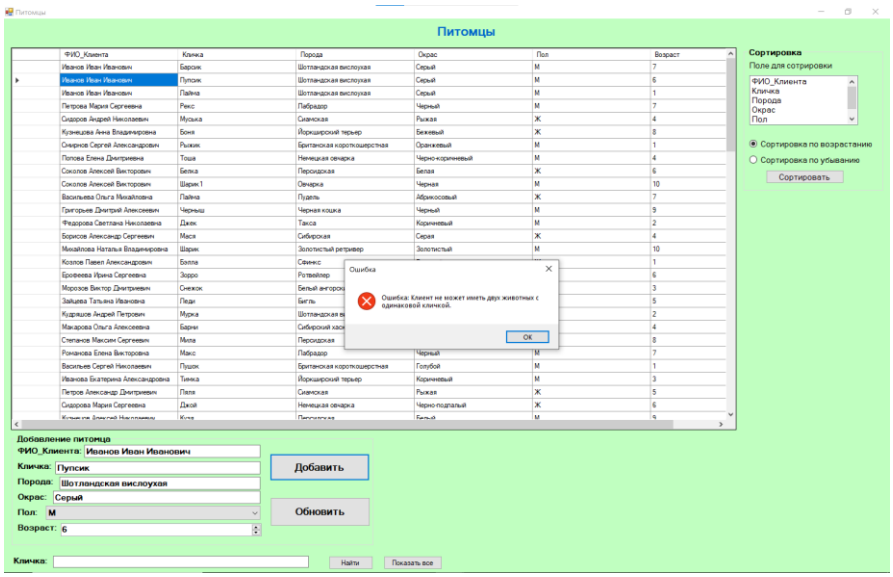

Рис. 2.58. Ошибка при добавлении питомца, один клиент не может иметь двух питомцев с одинаковой кличкой

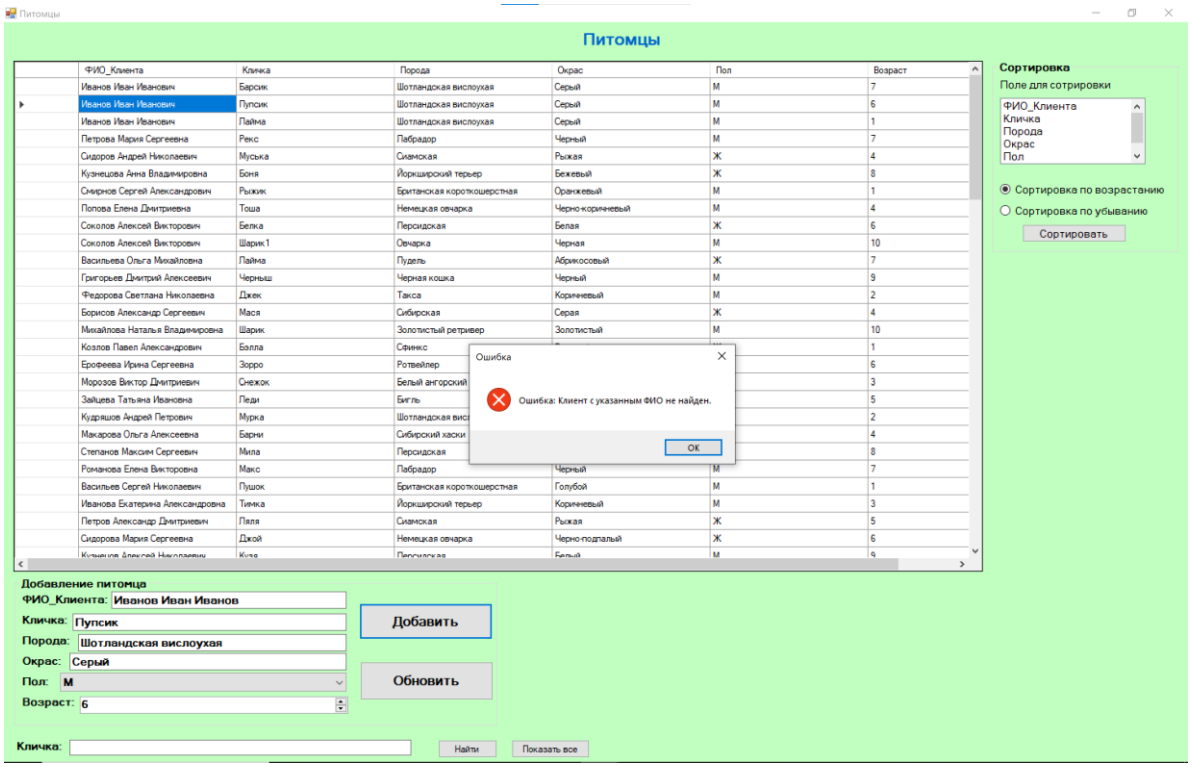

Рис. 2.59. Ошибка при добавлении питомца, клиент с указанным ФИО не найден

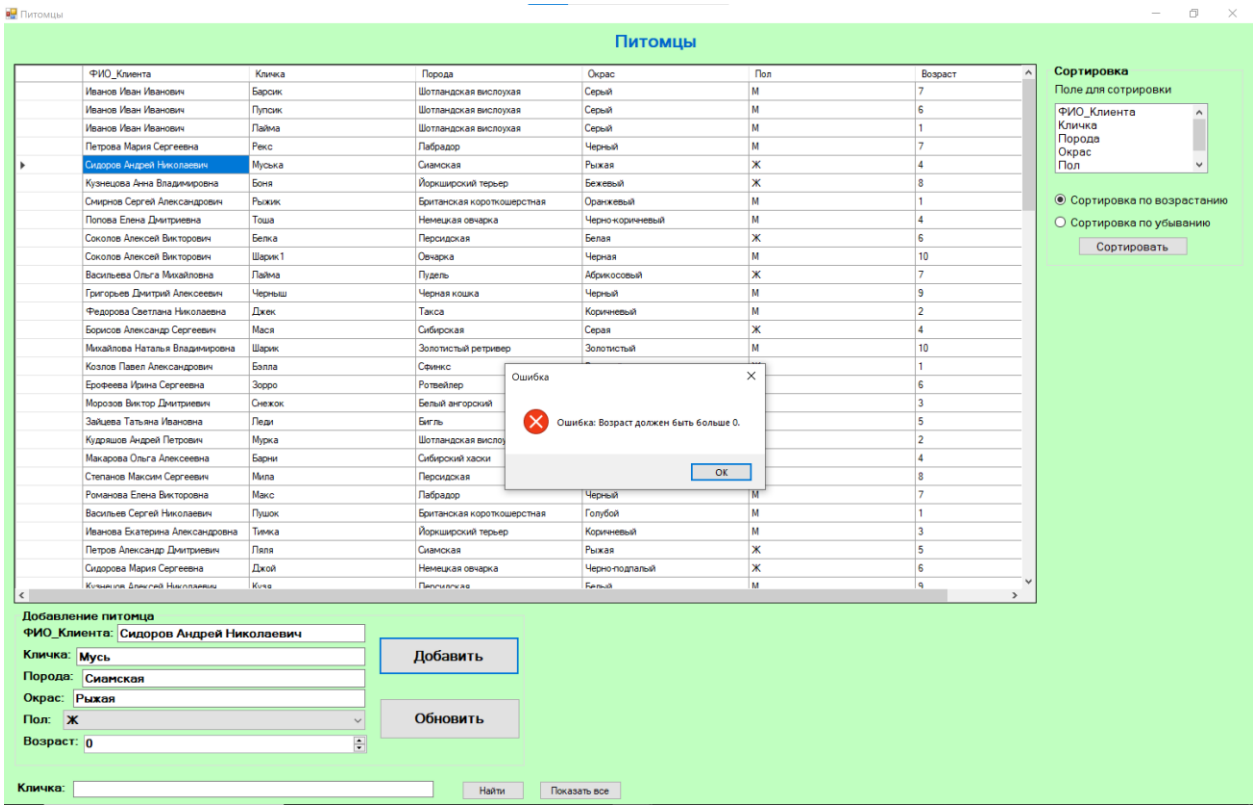

Рис. 2.60. Ошибка при добавлении питомца, возраст питомца должен быть больше 0

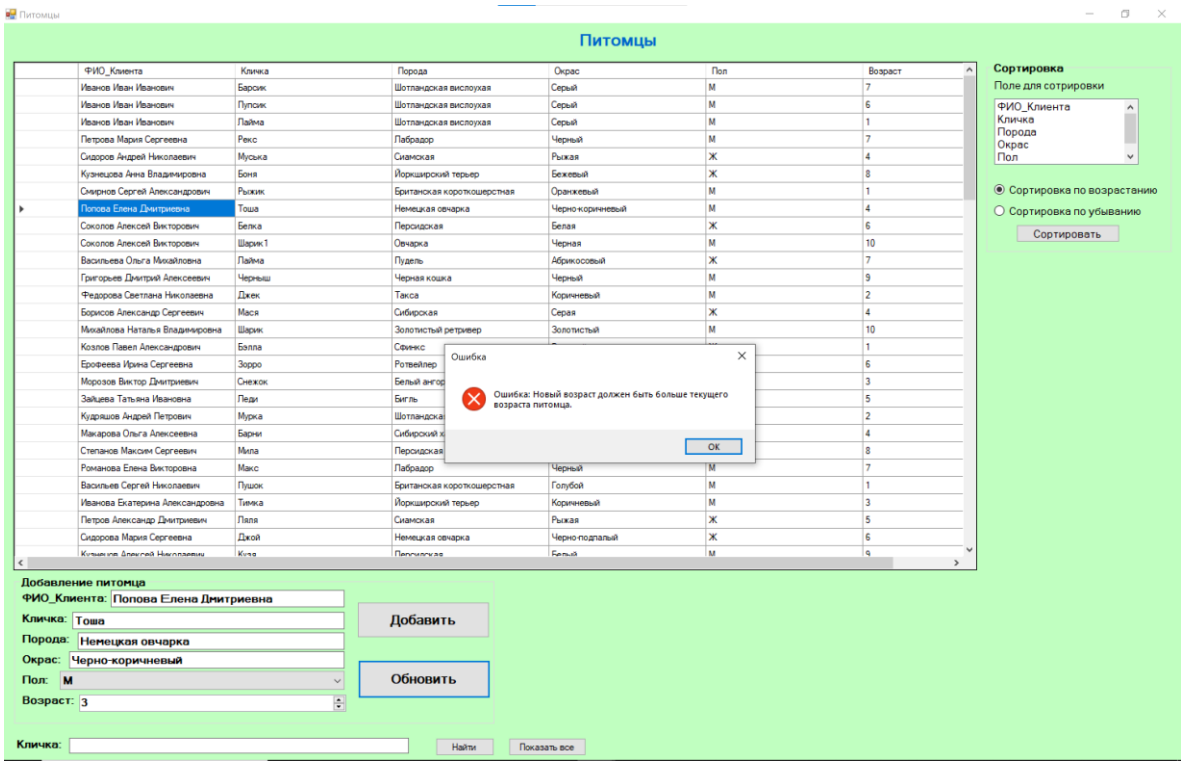

Рис. 2.61. Ошибка при обновлении возраста питомца, новый возраст должен быть больше текущего возраста питомца

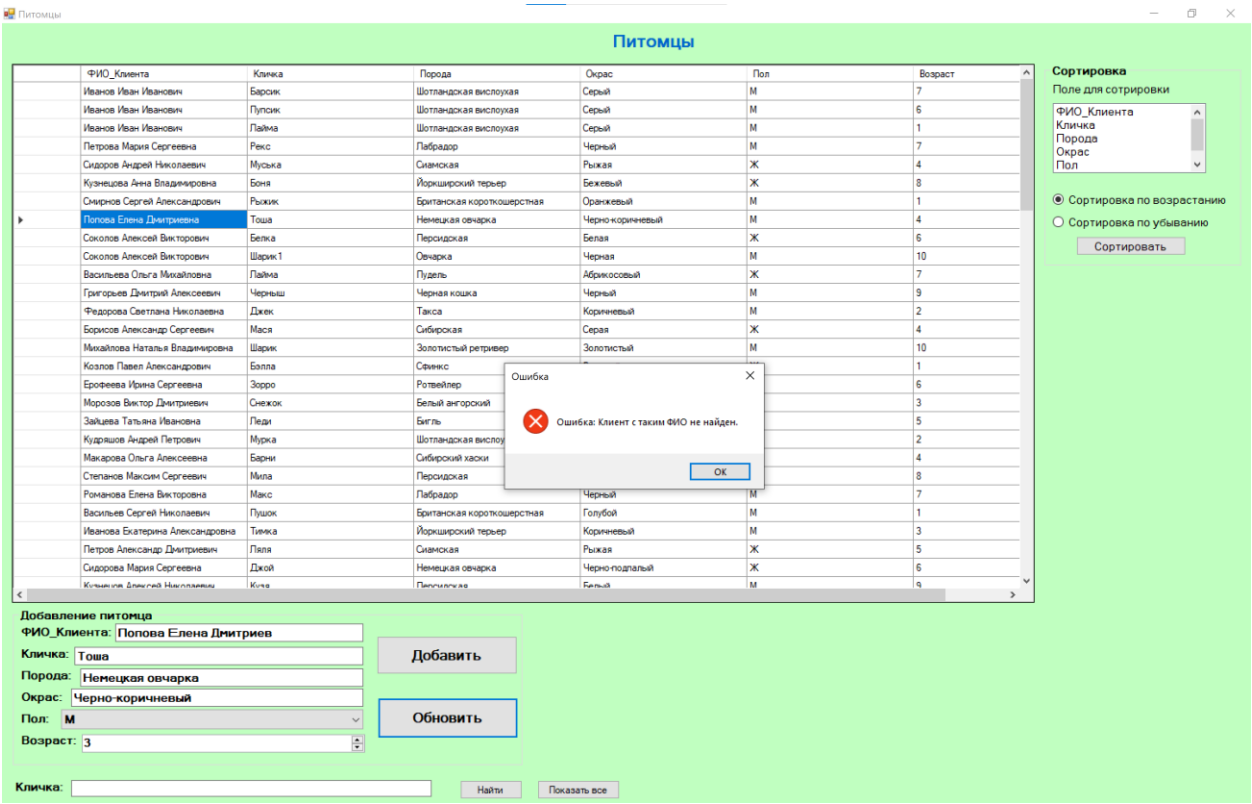

Рис. 2.62. Ошибка при обновлении возраста питомца, клиент с указанным ФИО не найден

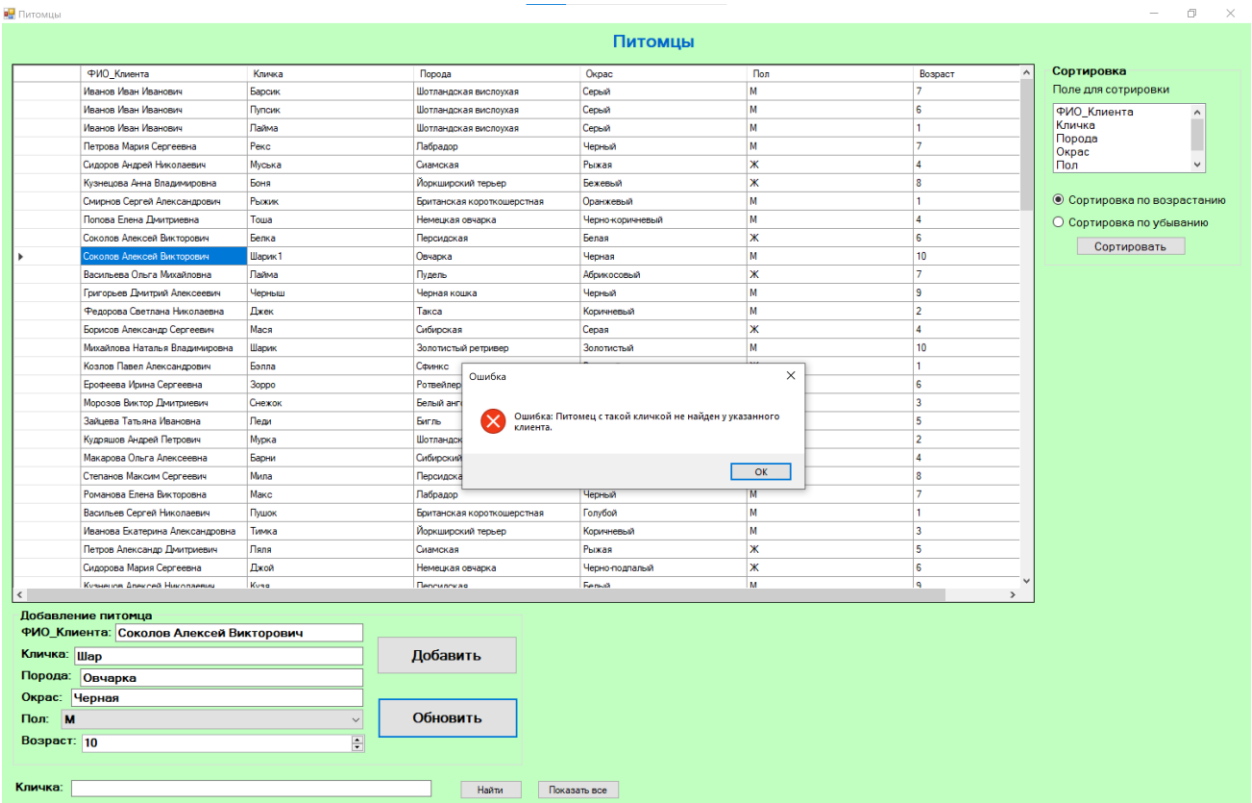

Рис. 2.63. Ошибка при обновлении возраста питомца, питомец с такой кличкой не найден у указанного клиента

Теперь перейдем к рассмотрению формы Прием (рис. 2.64). Доступ к ней имеет ветеринар, по кнопке «Проведение приема». На данной форме реализован поиск по Кличке_Питомца, а также сортировка. Также использованы выпадающие списки для полей ФИО_Сотрудника, Кличка_Питомца, ФИО_Клиента и Оказанная_Услуга, кроме того используется календарь для выбора Даты_Времени, а также отдельный dataGried для отображения рабочего времени каждого из сотрудников. Используя данную форму, пользователь имеет возможность добавить новый прием (рис. 2.65), добавить новую услугу (рис. 2.66), обновить комментарий приема (рис. 2.67), а также удалить прием (рис. 2.68) или услугу (рис. 2.69). На рис. 2.70-2.86 представлены выводы сообщений об ошибках, отслеживаемых при работе с БД.

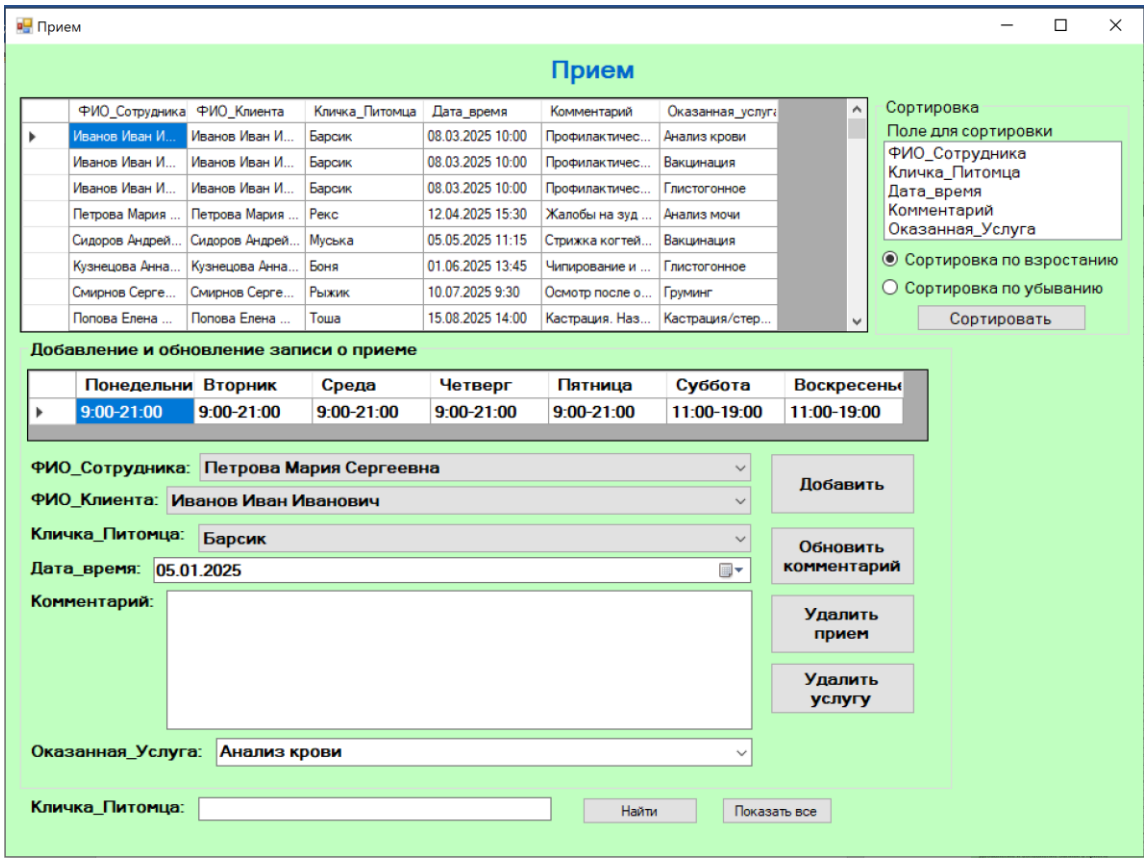

Рис. 2.64. Форма Прием

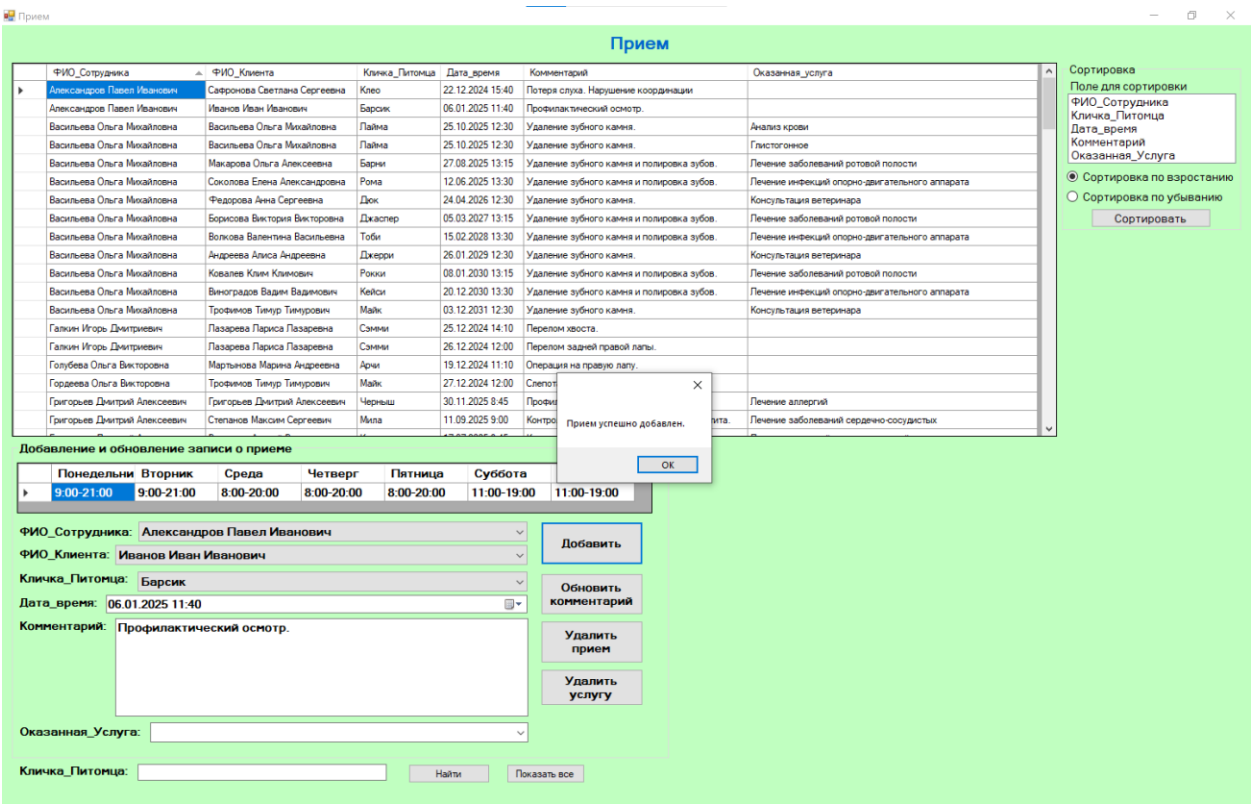

Рис. 2.65. Успешное добавление нового приема

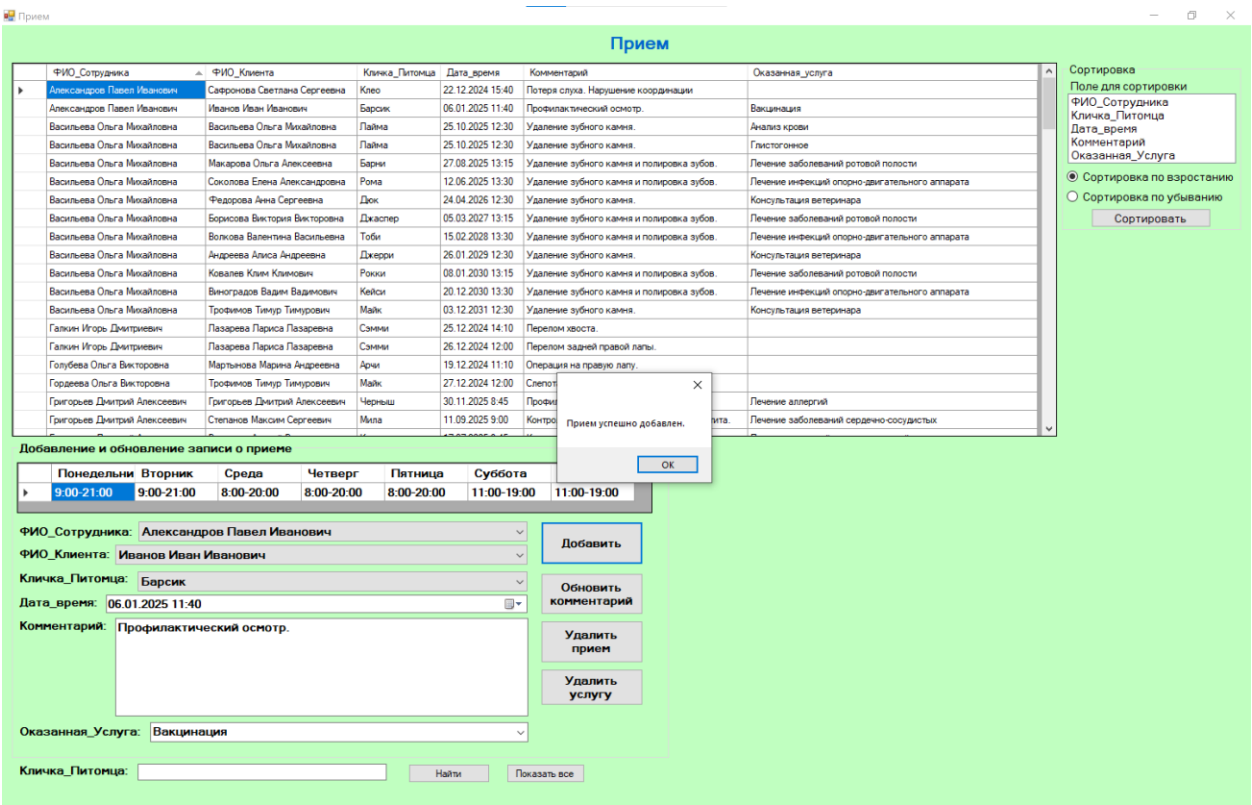

Рис. 2.66. Успешное добавление услуги к приему

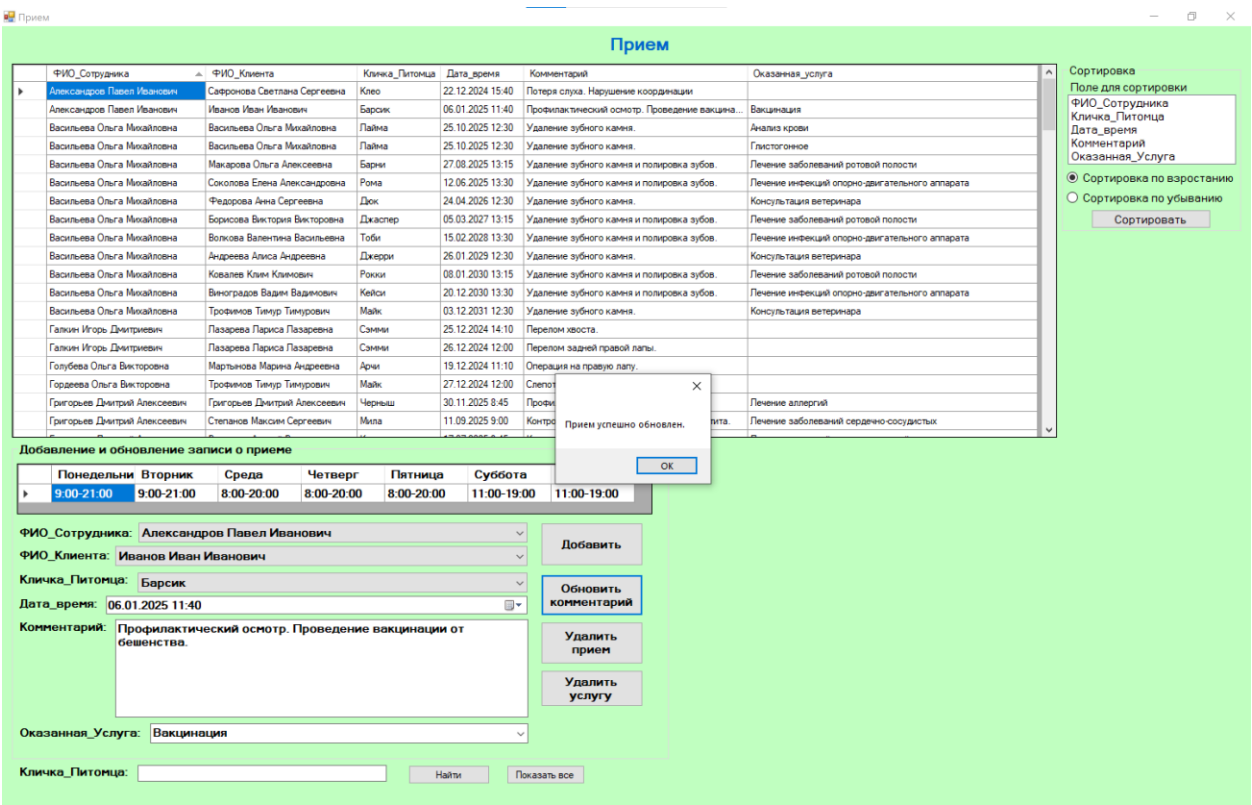

Рис. 2.67. Успешное обновление комментария приема

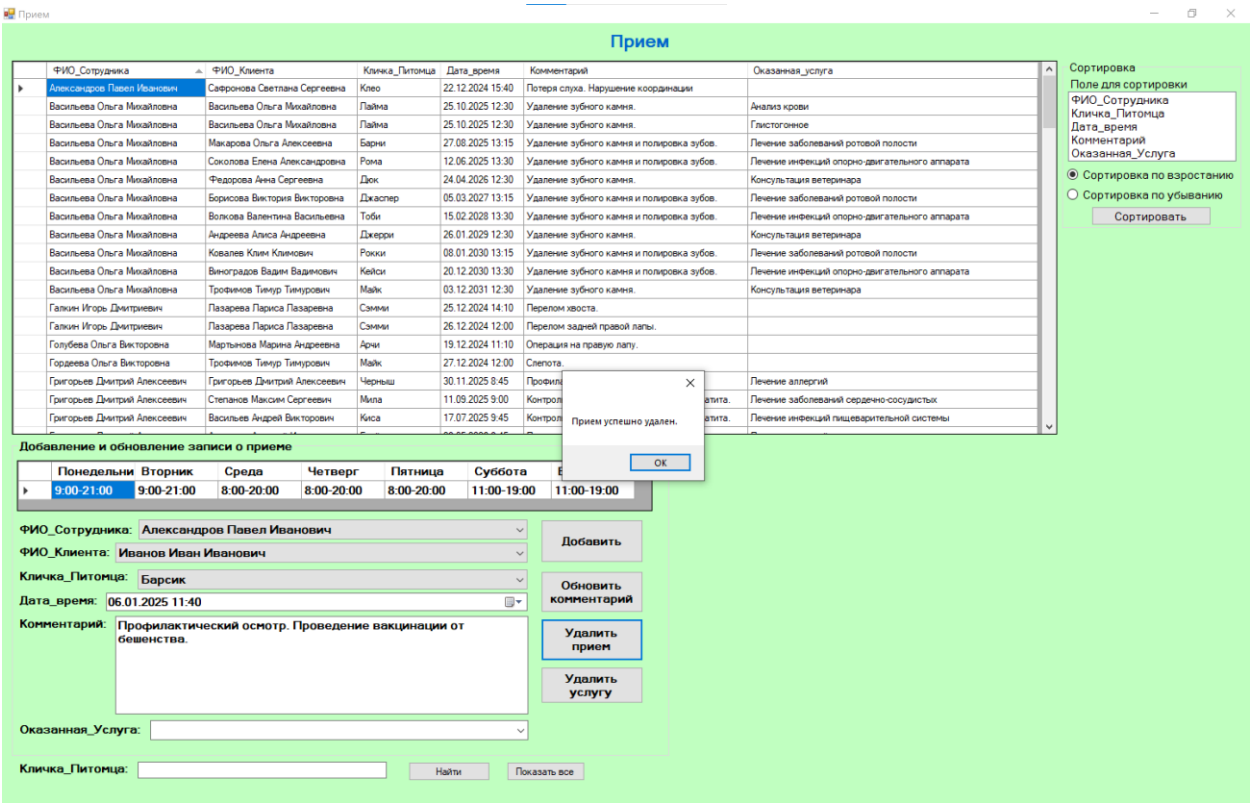

Рис. 2.68. Успешное удаление приема

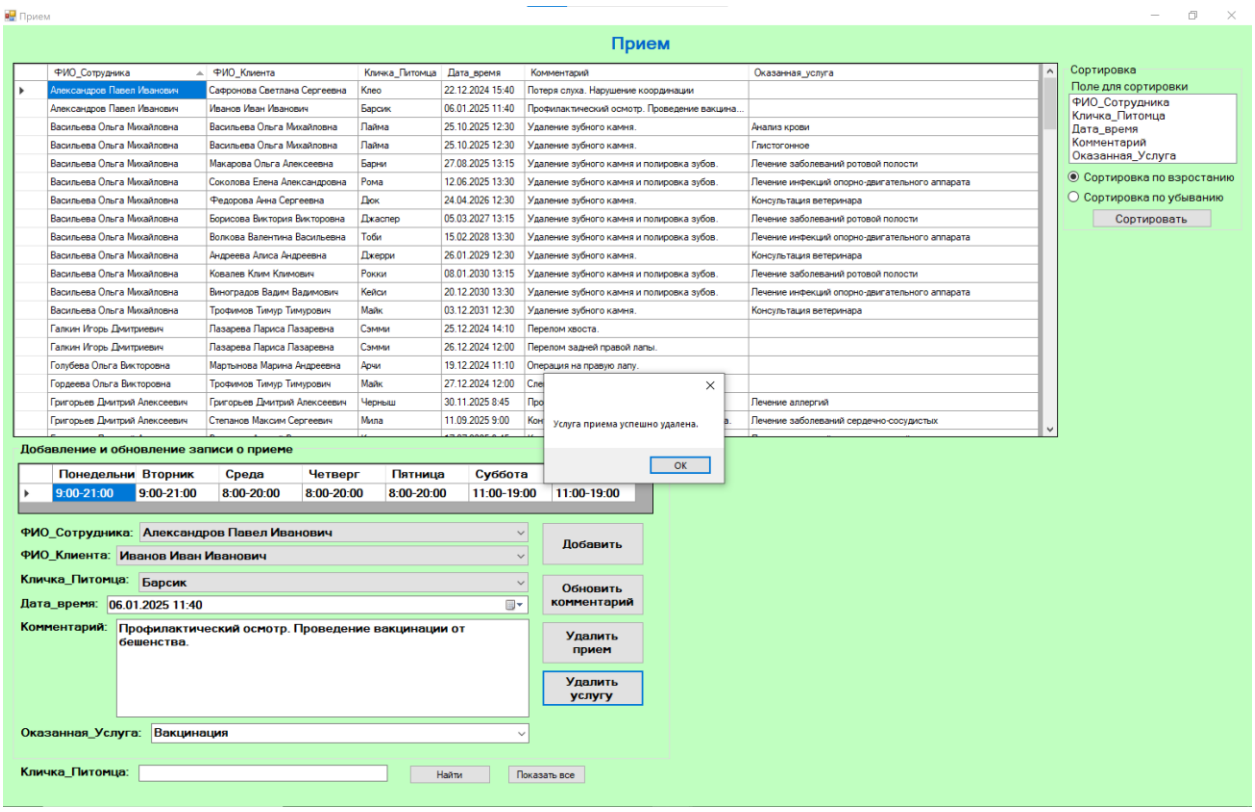

Рис. 2.69. Успешное удаление услуги приема

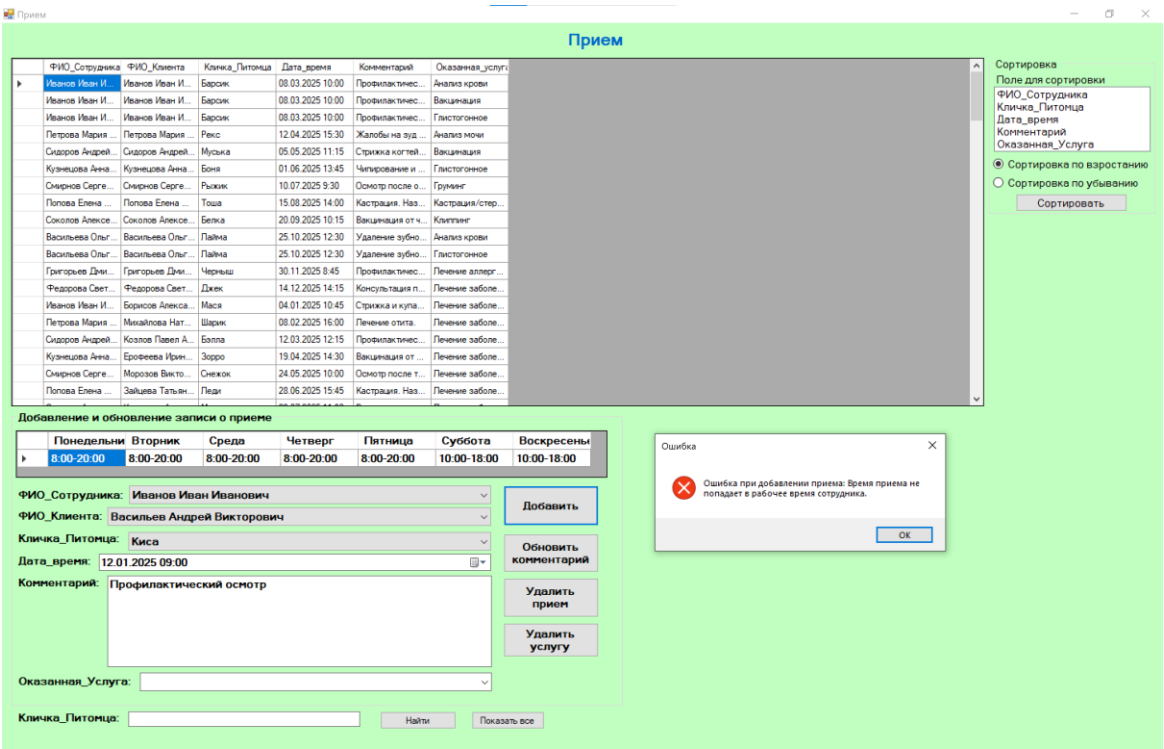

Рис. 2.70. Ошибка при добавлении приема, указано время, в которое сотрудник не работает

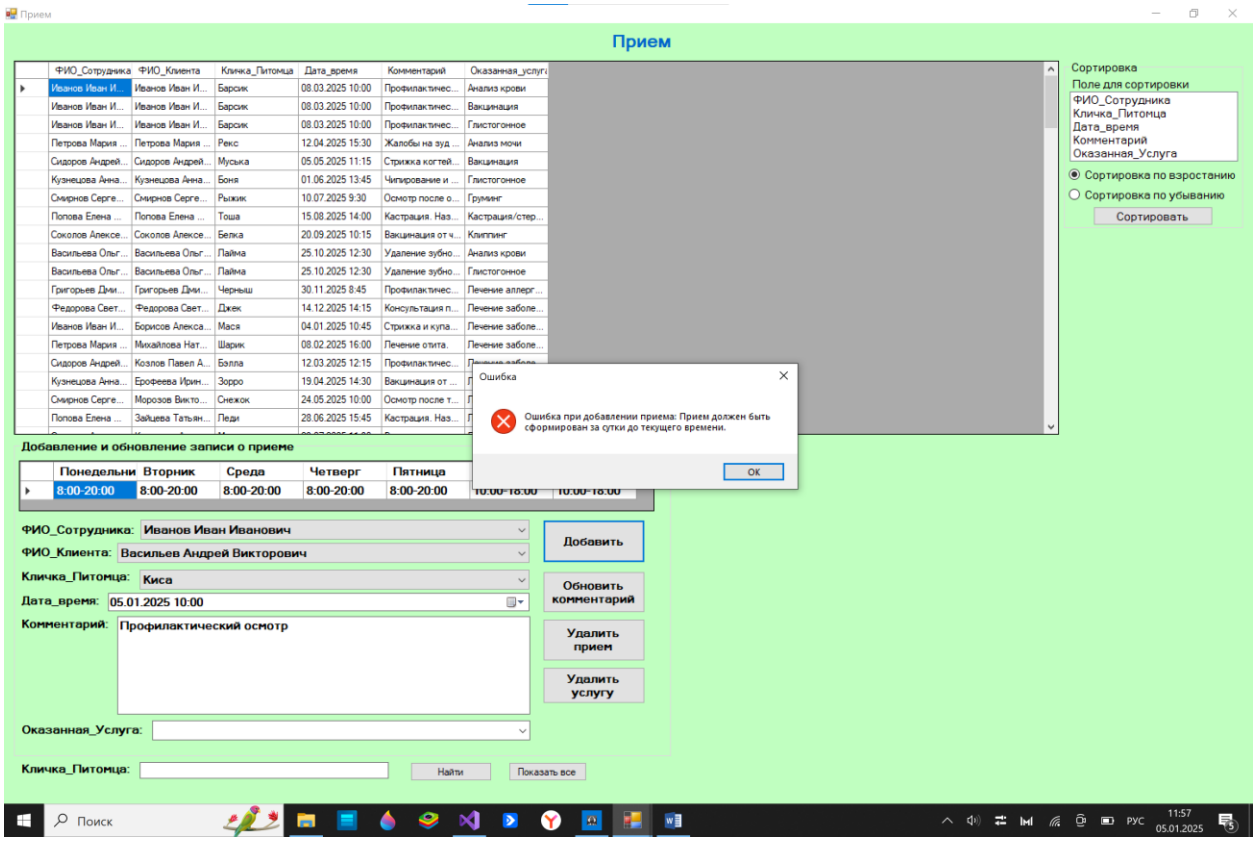

Рис. 2.71. Ошибка при добавлении приема, попытка добавить прием менее чем за сутки до текущей даты или «задним числом»

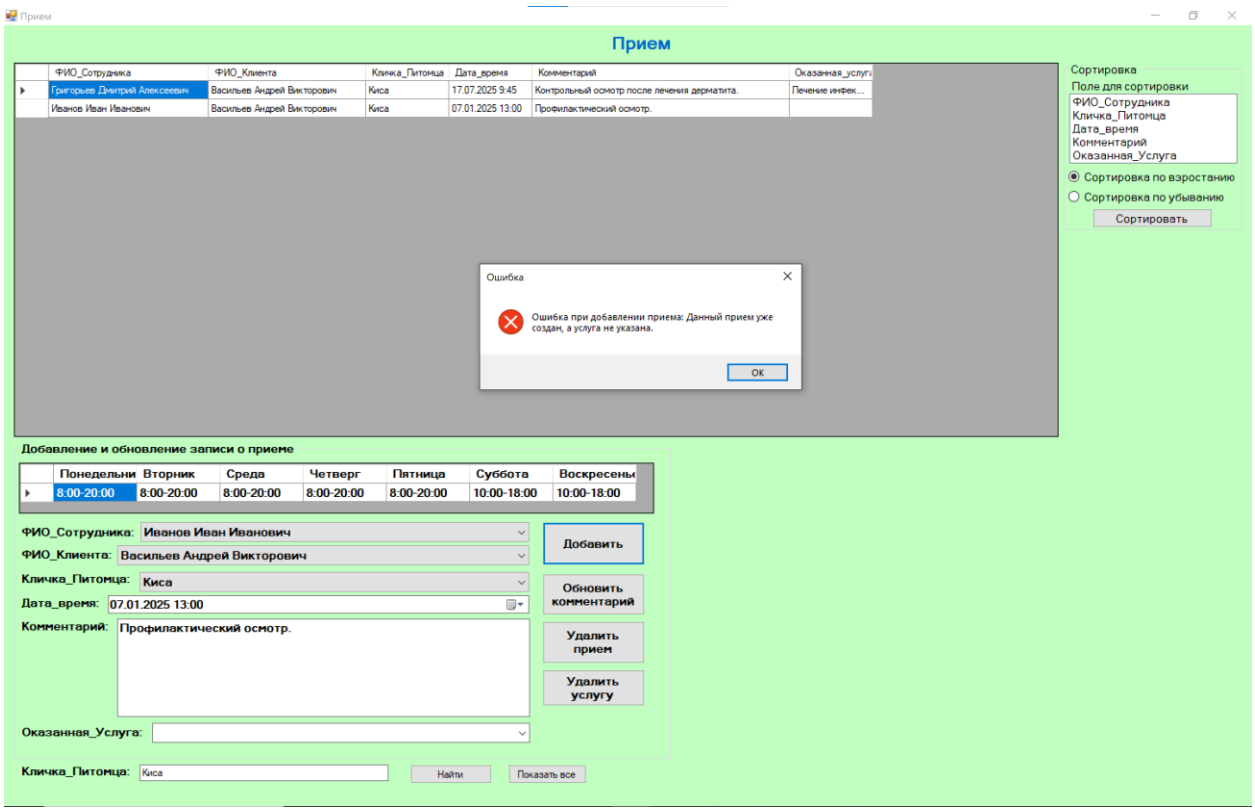

Рис. 2.72. Ошибка при добавлении приема, попытка создать уже существующий прием, не указав услугу

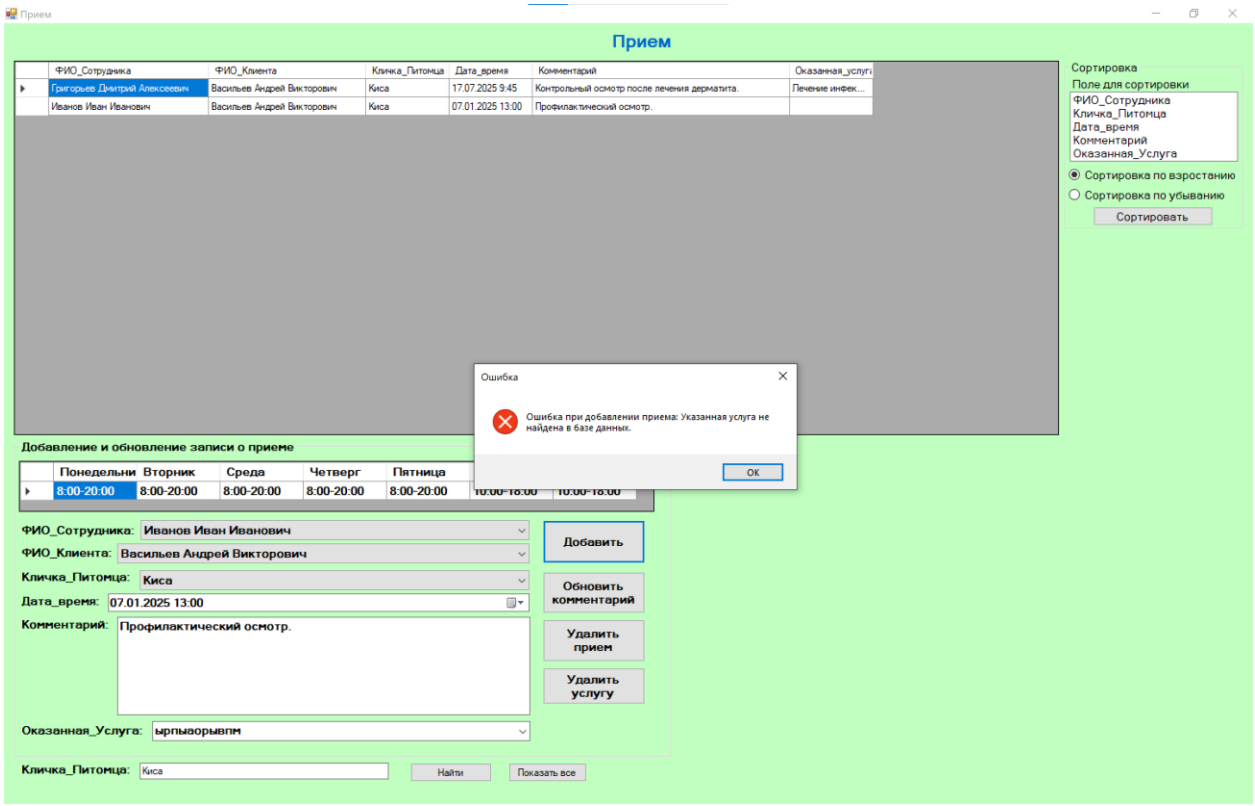

Рис. 2.73. Ошибка при добавлении услуги, услуга не существует в БД

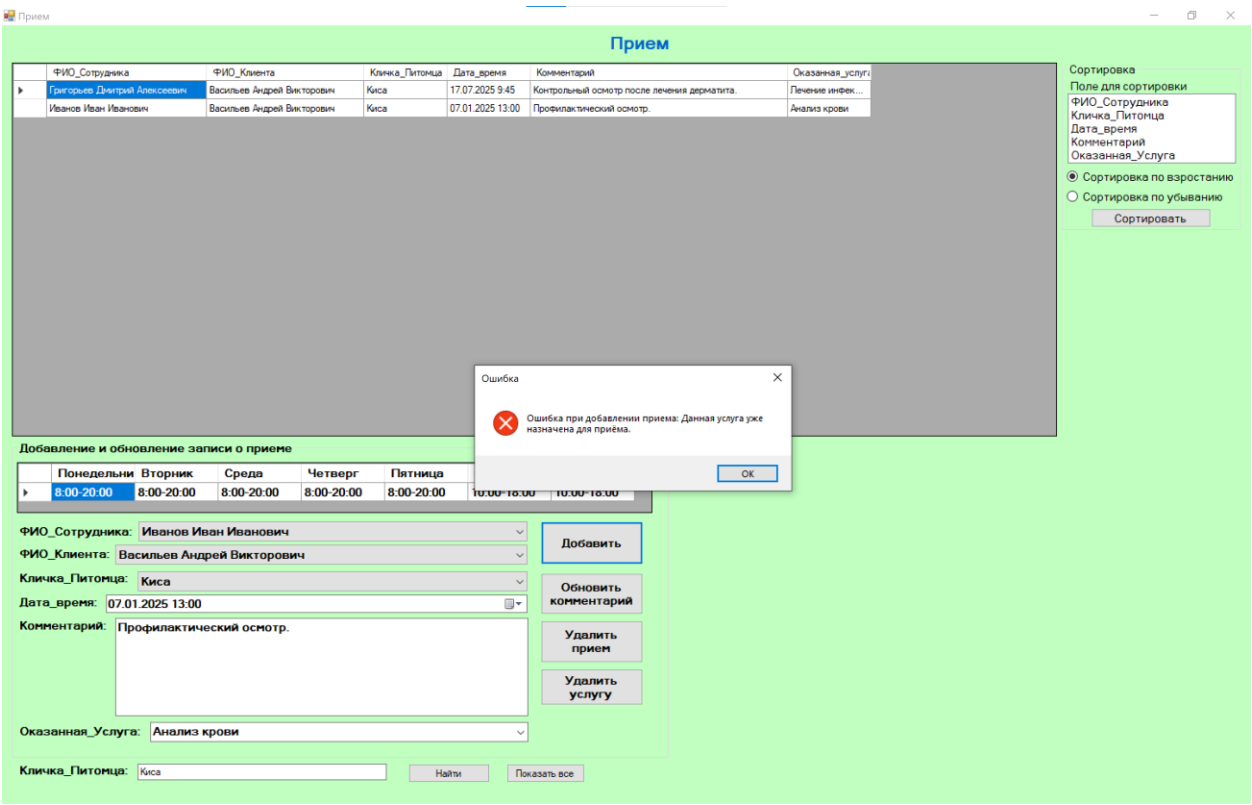

Рис. 2.74. Ошибка при добавлении услуги, повторное добавление одной и той же услуги к приему

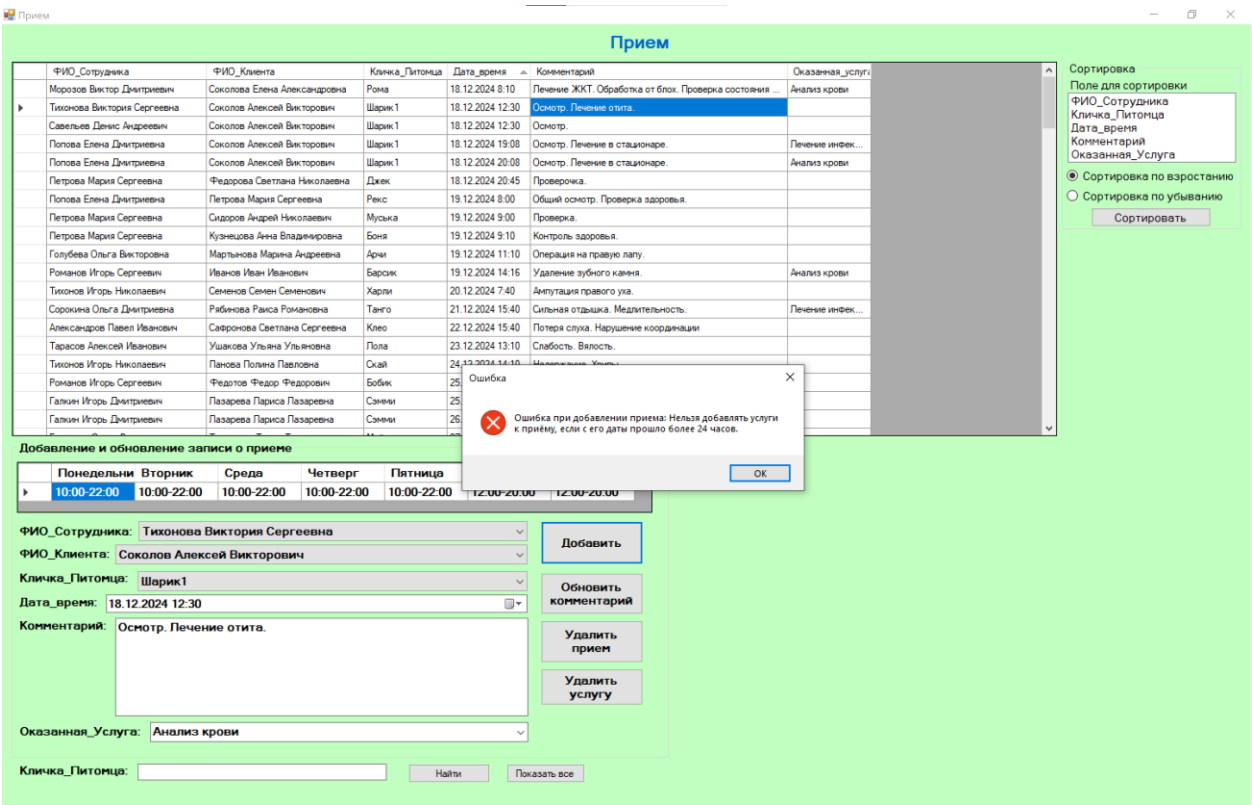

Рис. 2.75. Ошибка при добавлении услуги, со дня приема прошло более суток

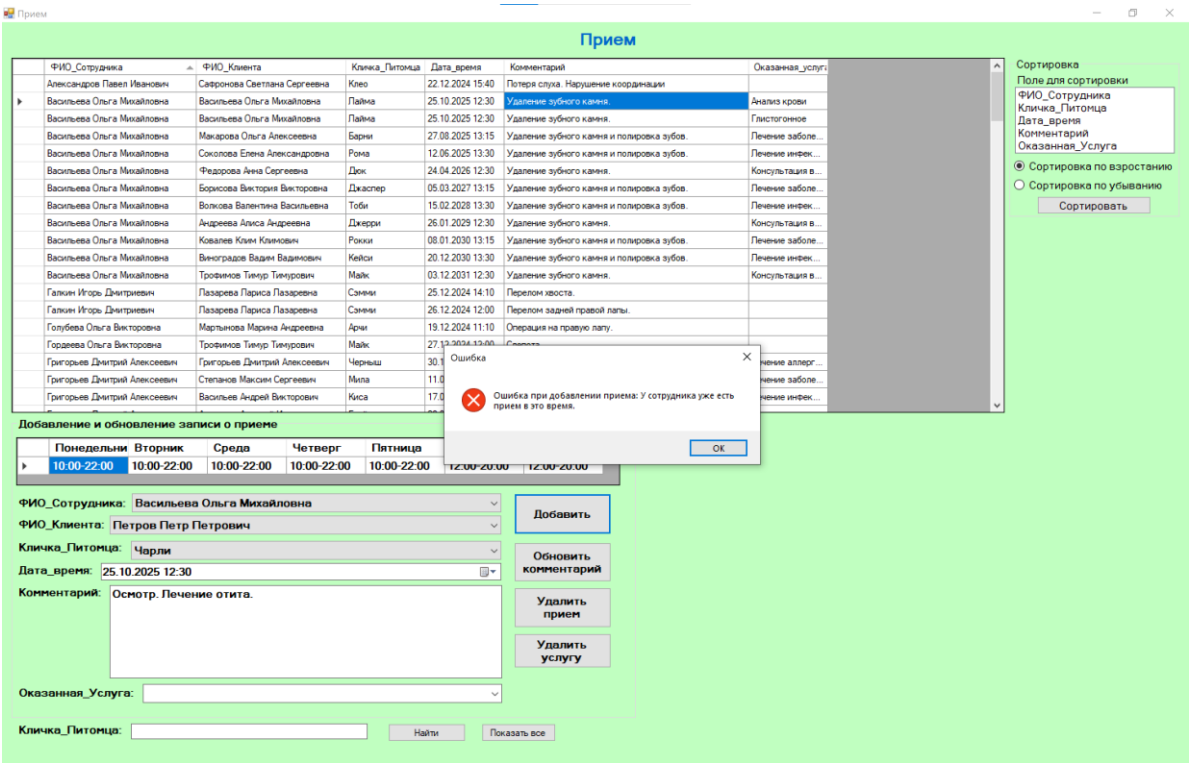

Рис. 2.76. Ошибка при добавлении приема, у сотрудника уже есть прием в это время

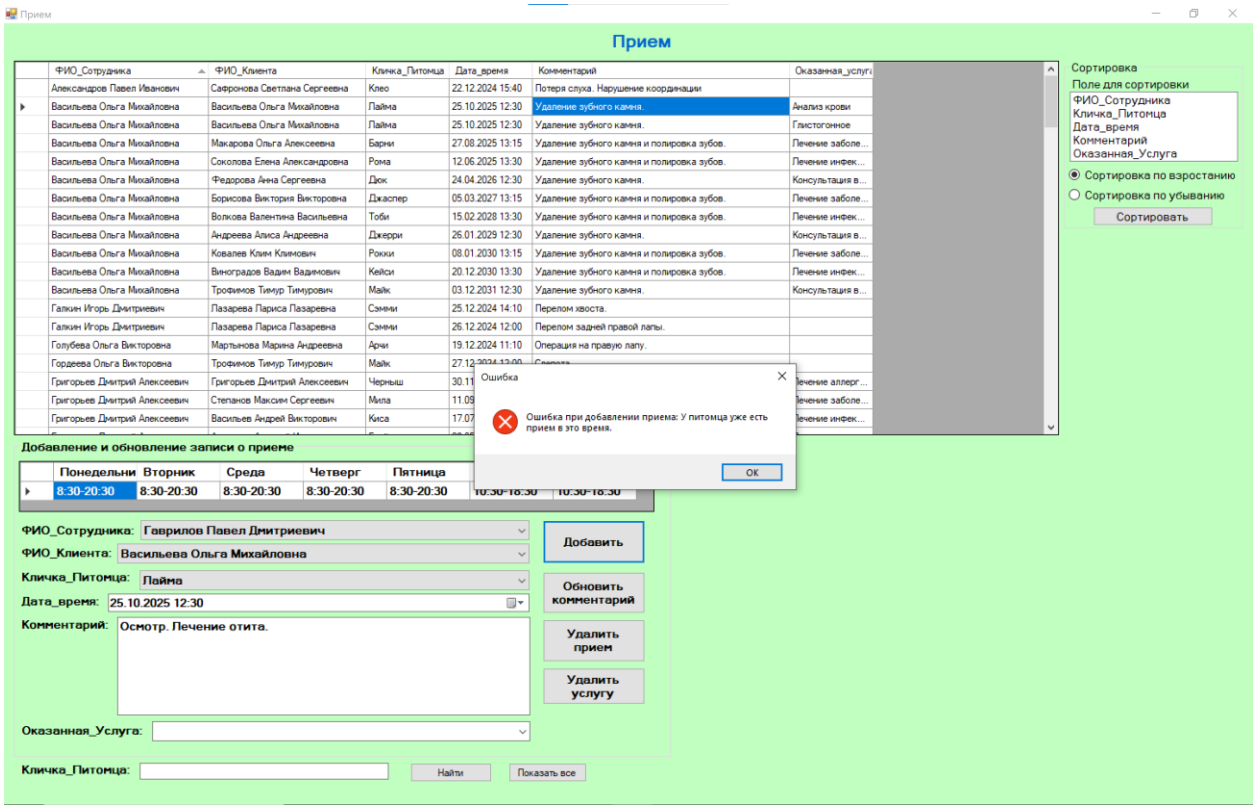

Рис. 2.77. Ошибка при добавлении приема, у питомца уже есть прием в это время

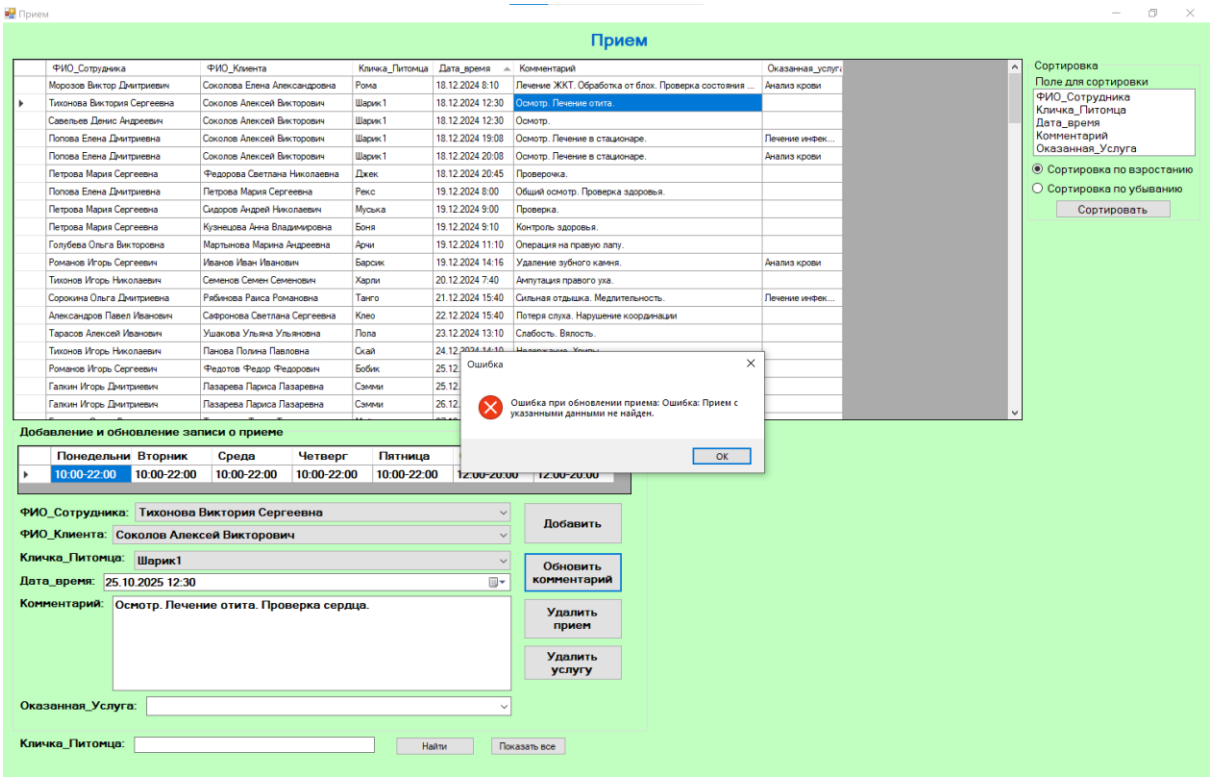

Рис. 2.78. Ошибка при обновлении комментария к приему, указанный прием не найден

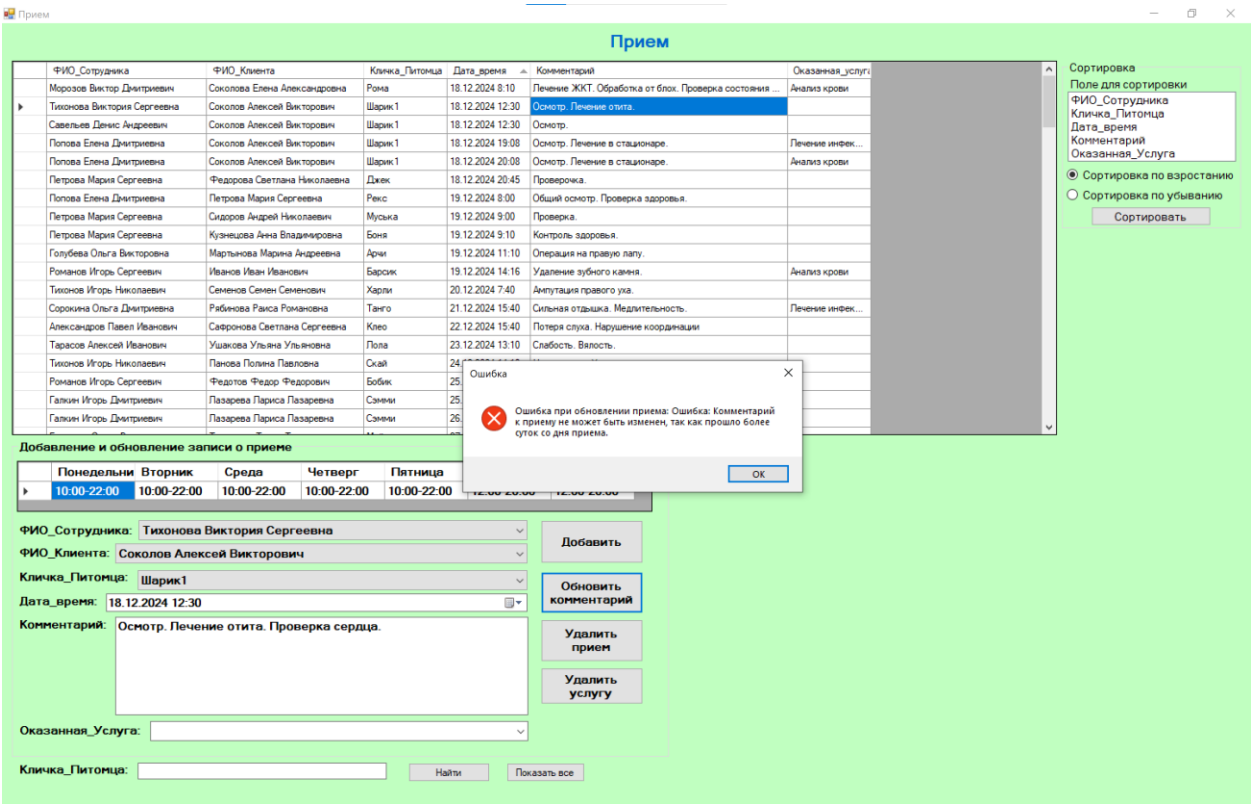

Рис. 2.79. Ошибка при обновлении комментария к приему, со дня приема прошло более суток

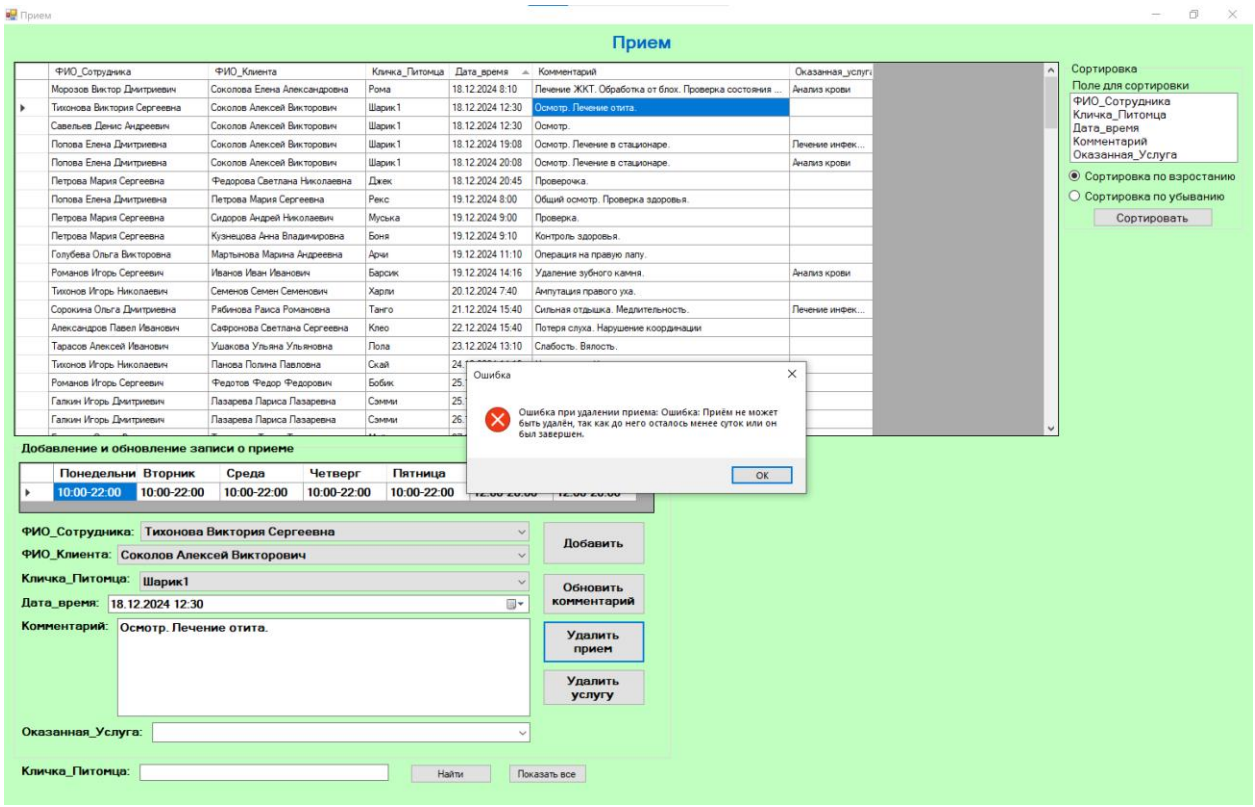

Рис. 2.80. Ошибка при удалении приема, до приема осталось менее суток или он был завершен

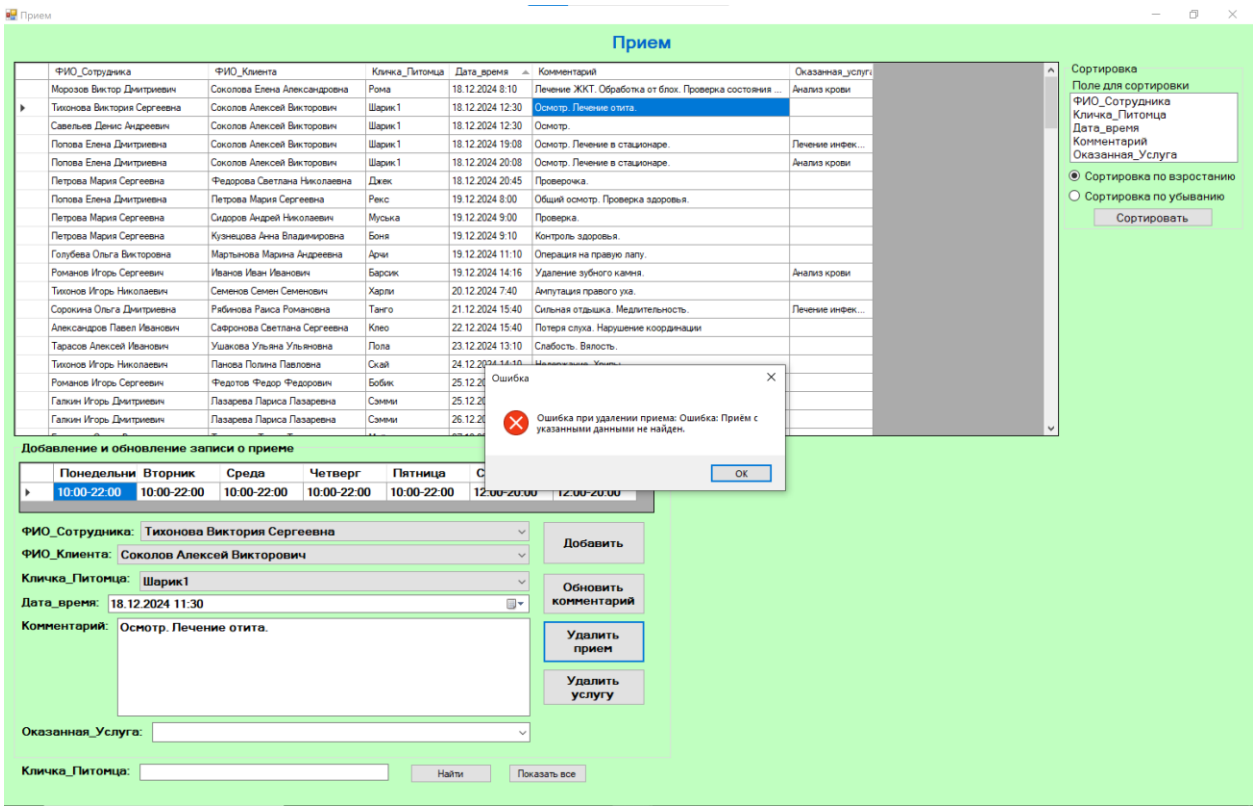

Рис. 2.81. Ошибка при удалении приема, прием не найден

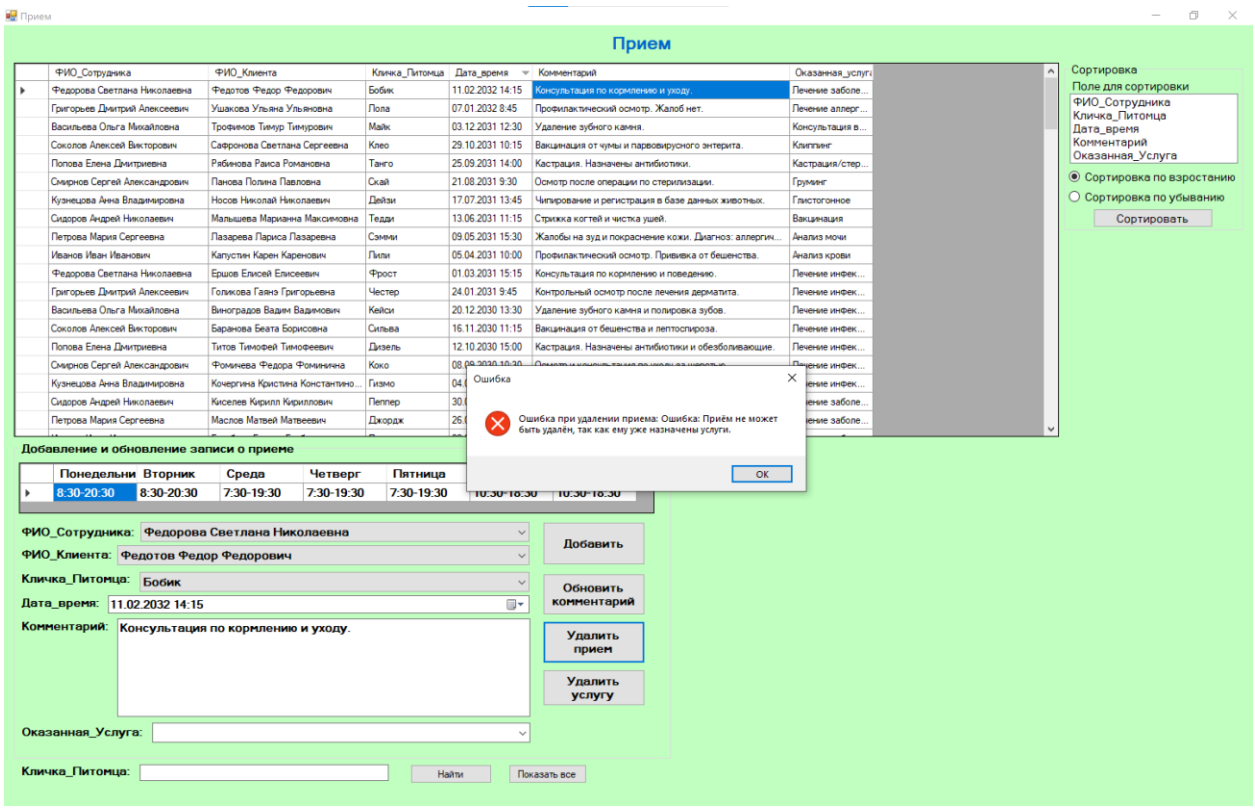

Рис. 2.82. Ошибка при удалении приема, приему назначена услуга

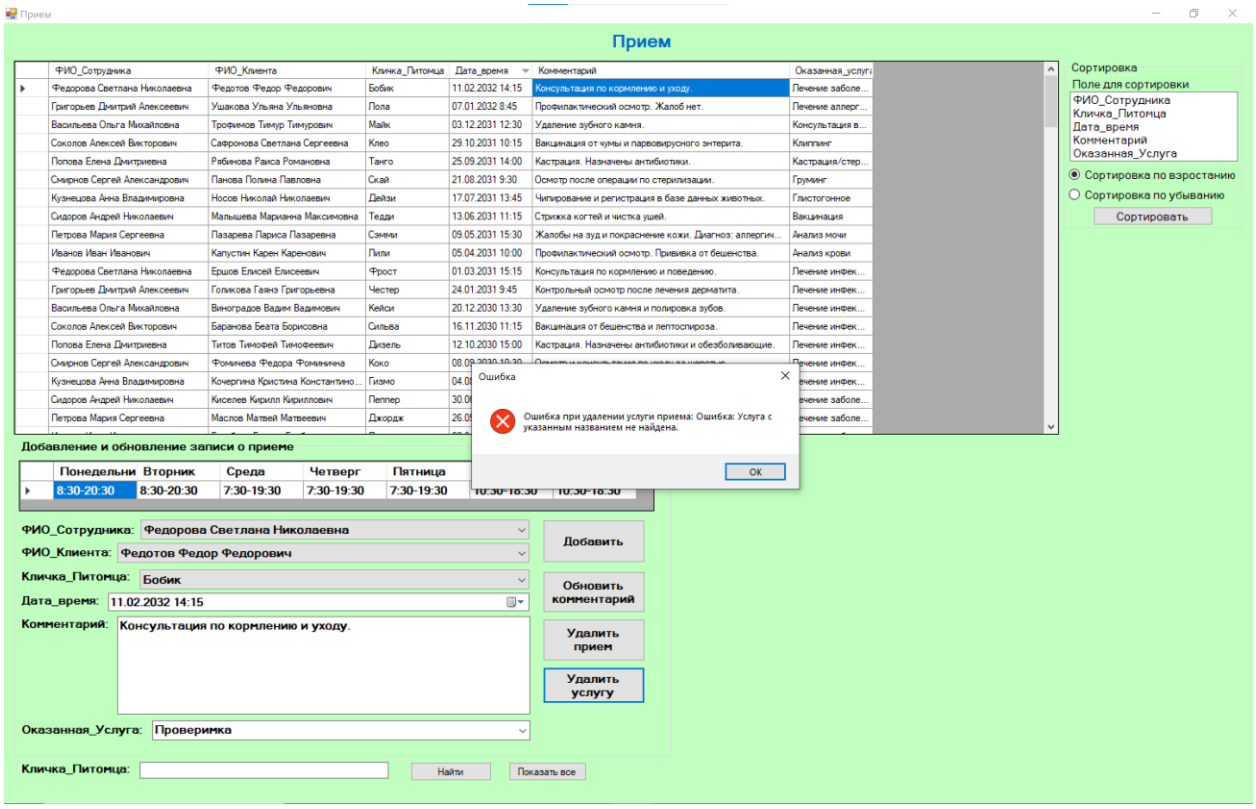

Рис. 2.83. Ошибка при удалении услуги, услуга с указанным названием не найдена

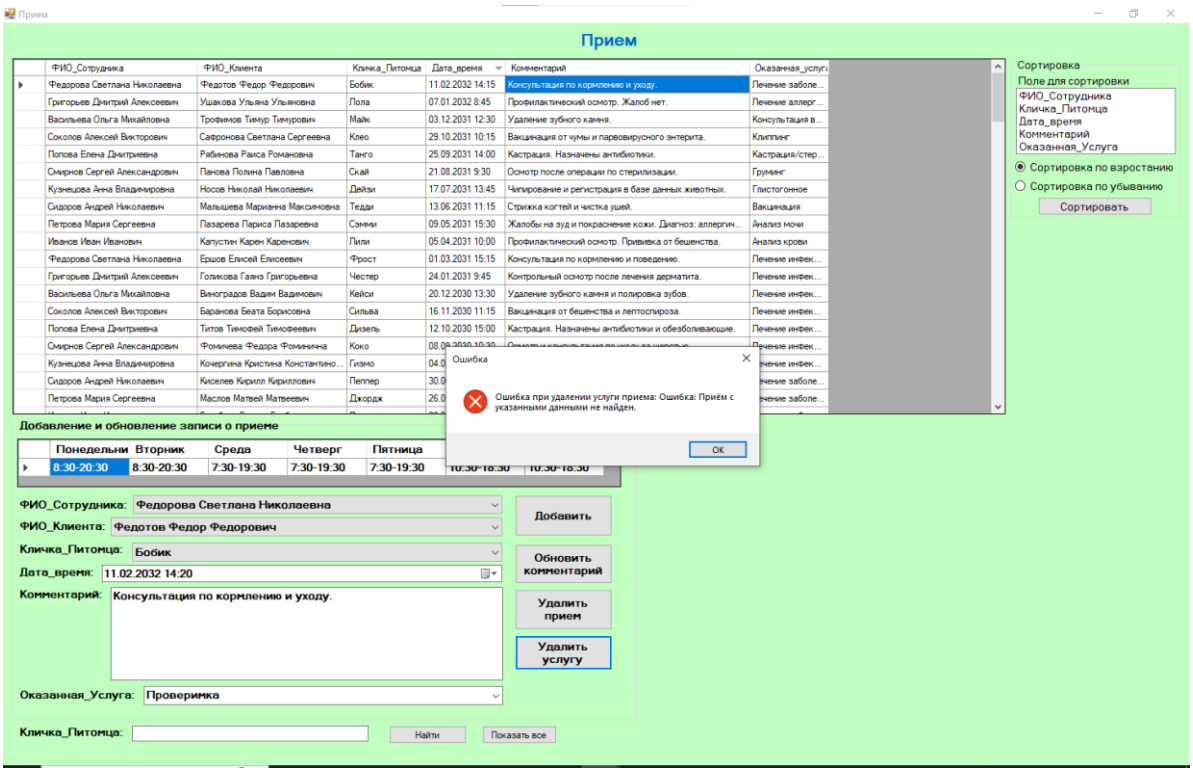

Рис. 2.84. Ошибка при удалении услуги, прием с указанными данными не найден

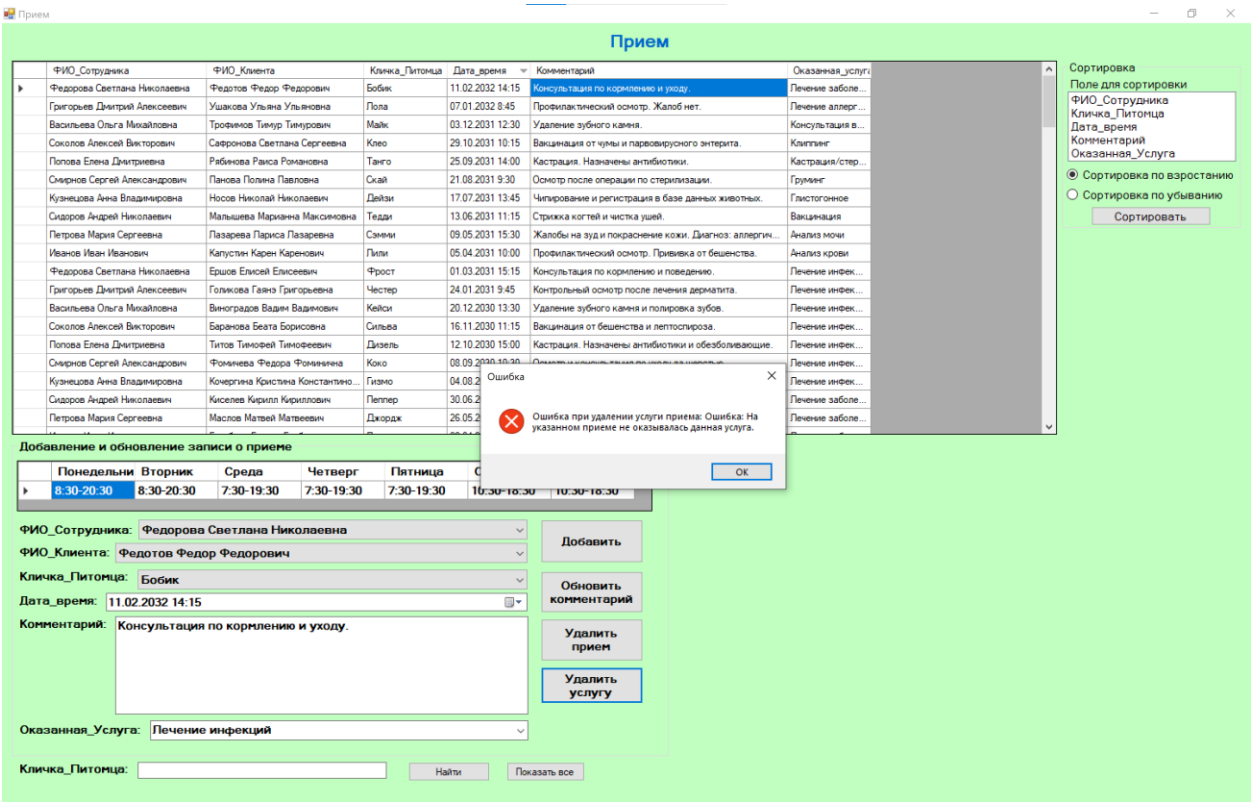

Рис. 2.85. Ошибка при удалении услуги, на приеме не оказывалась данная услуга

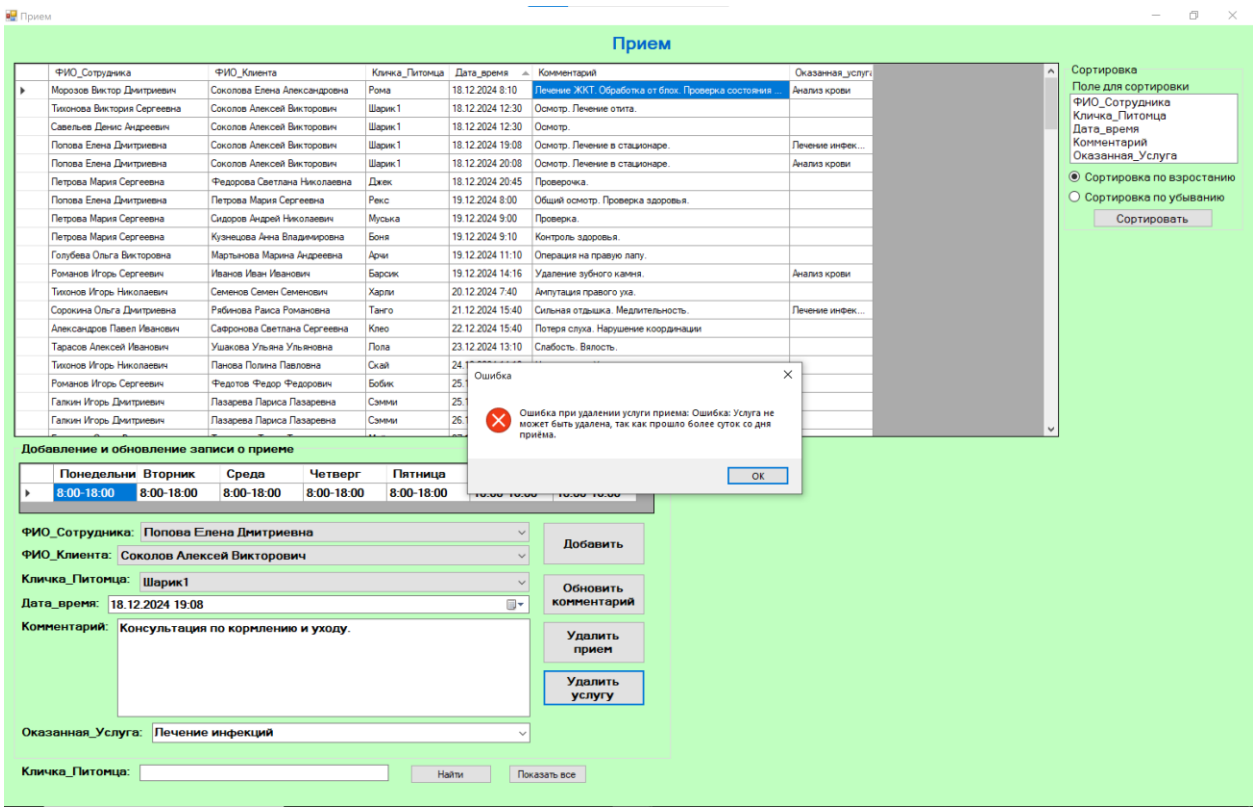

Рис. 2.86. Ошибка при удалении услуги, со дня приема прошло более суток

ЗАКЛЮЧЕНИЕ

Предметной областью данной самостоятельной работы является деятельность ветеринарной клиники.

Ветеринарная клиника в настоящее время сталкивается с значительным объемом информации о пациентах, что делает процессы управления данными более сложными. Учет медицинской истории, регистрация клиентов, запись результатов обследований, анализы и лечение животных - все это требует систематизации и эффективной обработки данных. Для успешного функционирования ветеринарной клиники важно иметь систему управления информацией, которая будет обеспечивать надежное хранение данных о пациентах, обеспечивать доступность информации для врачей и персонала, а также обеспечивать соблюдение конфиденциальности и безопасности данных. Разработка современной информационной системы, специально адаптированной для потребностей ветеринарной медицины, позволит эффективно управлять данными и повысить уровень обслуживания пациентов.

В ходе выполнения самостоятельной работы были получены следующие результаты.

Во время анализа предметной области были выделены основные задачи, решаемые в рамках ветеринарной клиники. Рассмотрены существующие программные решения, среди них: программный продукт « БИТ.Айболит» от компании 1С и конфигурация «Ветеринарная клиника» от компании «Простой софт». По результатам проведенного анализа был выделен перечень функций, которые будут реализованы в данной работе.

На этапе концептуального проектирования были выделены пользователи системы (генеральный директор, администратор клиники, менеджер по запасам и ветеринар) и определены их функции. Далее составлена диаграмма вариантов использования, по которой представлен список объектов, необходимых для решения выделенных задач.

В процессе выполнения логического проектирования были построены ER-диаграммы сущностей, определены связи между ними, получен набор предварительных отношений с указанием первичных и внешних ключей. Сформирована полная схема БД с добавлением неключевых атрибутов, состоящая из отношений Клиент, Сотрудник, Питомец, Прием, Услуга, СкладскаяОперация, Медикамент, График, УслугаПриема. Определены ограничения, действующие для ветеринарной клиники.

На этапе физического проектирования с помощью среды Microsoft SQL Server Management Studio были созданы связанные таблицы (схема БД) согласно приведенным требованиям к их структурам. Кроме того, каждая из таблиц была частично заполнена данными.

На заключительном этапе выполнения проекта была разработана программа для взаимодействия с пользователями – генеральным директором, администратором, менеджером по запасам и ветеринаром. При этом были использованы WinForms выполненные на языке C# [8]. В проекте были реализованы процедуры и представления, представленные на различных формах, с помощью которых были реализованы: добавление, изменение и удаление данных, а также их корректное отображение в рамках пользовательского интерфейса.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Сайт «Инфоурок» [Электронный ресурс] – Режим доступа: https://infourok.ru/1-organizaciya-i-metody-sbora-informacii-analiz-predmetnoj-oblasti-osnovnye-ponyatiya-sistemnogo-i-strukturnogo-analiza-7330430.html, свободный. Дата обращения: 20.04.2024 г.
2. Сайт «БИТ.Айболит» [Электронный ресурс] – Режим доступа: https://bit-medic.ru/bit-aybolit/, свободный. Дата обращения: 21.04.2024 г.
3. Сайт «Простой Софт» [Электронный ресурс] – Режим доступа: https://prostoysoft.ru/Veterinary.htm, свободный. Дата обращения: 22.04.2024 г.
4. Сайт «CYBERLENINKA» [Электронный ресурс] – Режим доступа: https://cyberleninka.ru/article/n/podhody-k-proektirovaniyu-baz-dannyh-dlya-avtomatizirovannyh-sistem, свободный. Дата обращения: 24.04.2024 г.
5. Сайт «Хабр» [Электронный ресурс] – Режим доступа: https://habr.com/ru/articles/566218/, свободный. Дата обращения: 26.04.2024 г.
6. Сайт «БлогЯПрактикума» [Электронный ресурс] – Режим доступа: https://practicum.yandex.ru/blog/chto-takoe-er-diagramma/, свободный. Дата обращения: 29.04.2024 г.
7. Сайт «Microsoft» [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/en-us/dotnet/desktop/winforms/controls/datagrid-control-overview-windows-forms?view=netframeworkdesktop-4.8, свободный. Дата обращения: 19.11.2024 г.
8. Сайт «Microsoft» [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/visualstudio/get-started/csharp/tutorial-windows-forms-picture-viewer-code?view=vs-2022&tabs=csharp, свободный. Дата обращения: 29.11.2024 г.

ПРИЛОЖЕНИЕ А

ЗАПОЛНЕНИЕ ТАБЛИЦ ДАННЫМИ

Id_Графика	Понедельник	Вторник	Среда	Четверг	Пятница	Суббота	Воскресенье
1	08:00-20:00	08:00-20:00	08:00-20:00	08:00-20:00	08:00-20:00	10:00-18:00	10:00-18:00
2	09:00-21:00	09:00-21:00	09:00-21:00	09:00-21:00	09:00-21:00	11:00-19:00	11:00-19:00
3	08:30-20:30	08:30-20:30	08:30-20:30	08:30-20:30	08:30-20:30	10:30-18:30	10:30-18:30
4	09:00-20:00	09:00-20:00	08:00-19:00	08:00-19:00	08:00-19:00	11:00-17:00	11:00-17:00
5	7:30-19:30	7:30-19:30	7:30-19:30	7:30-19:30	7:30-19:30	9:30-17:30	9:30-17:30
6	08:00-18:00	08:00-18:00	08:00-18:00	08:00-18:00	08:00-18:00	10:00-16:00	10:00-16:00
7	09:00-17:00	09:00-17:00	09:00-17:00	09:00-17:00	09:00-17:00	11:00-15:00	11:00-15:00
8	10:00-22:00	10:00-22:00	10:00-22:00	10:00-22:00	10:00-22:00	12:00-20:00	12:00-20:00
9	09:00-21:00	09:00-21:00	08:00-20:00	08:00-20:00	08:00-20:00	11:00-19:00	11:00-19:00
10	08:30-20:30	08:30-20:30	07:30-19:30	07:30-19:30	07:30-19:30	10:30-18:30	10:30-18:30

Рис. А. 1. Таблица «График»

Id_Клиента	ФИО	Контактный_телефон	Email	Адрес
1	Иванов Иван Иванович	+7(916)123-45-67	ivanov@mail.ru	г. Москва, ул. Ленина, д. 10, кв. 5
2	Петрова Мария Сергеевна	+7(927)789-01-23	petrova@yandex.ru	г. Санкт-Петербург, ул. Невский пр., д. 50, кв. 20
3	Сидоров Андрей Николаевич	+7(903)111-22-33	sidorov@gmail.com	г. Краснодар, ул. Красная, д. 15, кв. 12
4	Кузнецова Анна Владимировна	+7(985)444-55-66	kuznecova@mail.ru	г. Екатеринбург, ул. Малышева, д. 100, кв. 35
5	Смирнов Сергей Александрович	+7(911)222-33-44	smirnov@yandex.ru	г. Новосибирск, ул. Красный пр., д. 20, кв. 10
6	Попова Елена Дмитриевна	+7(905)555-66-77	popova@gmail.com	г. Челябинск, ул. Кирова, д. 30, кв. 5
7	Соколов Алексей Викторович	+7(926)123-45-67	sokolov@mail.ru	г. Нижний Новгород, ул. Большая Покровская, д. 40, кв. 15
8	Васильева Ольга Михайловна	+7(913)789-01-23	vasiljeva@yandex.ru	г. Самара, ул. Куйбышева, д. 50, кв. 20
9	Григорьев Дмитрий Алексеевич	+7(908)111-22-33	grigoriev@gmail.com	г. Омск, ул. Ленина, д. 15, кв. 12
10	Федорова Светлана Николаевна	+7(963)444-55-66	fedorova@mail.ru	г. Казань, ул. Баумана, д. 100, кв. 35
11	Борисов Александр Сергеевич	+7(912)222-33-44	borisov@yandex.ru	г. Челябинск, ул. Курчатова, д. 20, кв. 10
12	Михайлова Наталья Владимировна	+7(937)555-66-77	mihajlova@gmail.com	г. Ростов-на-Дону, ул. Пушкинская, д. 30, кв. 5
13	Козлов Павел Александрович	+7(909)123-45-67	kozlov@mail.ru	г. Волгоград, ул. Комсомольская, д. 40, кв. 15
14	Ерофеева Ирина Сергеевна	+7(918)789-01-23	erofeeva@yandex.ru	г. Пермь, ул. Ленина, д. 50, кв. 20
15	Морозов Виктор Дмитриевич	+7(902)111-22-33	morozov@gmail.com	г. Красноярск, ул. Карла Маркса, д. 15, кв. 12
16	Зайцева Татьяна Ивановна	+7(953)444-55-66	zajceva@mail.ru	г. Воронеж, ул. Революции, д. 100, кв. 35
17	Кудряшов Андрей Петрович	+7(921)222-33-44	kudryashov@yandex.ru	г. Уфа, ул. Ленина, д. 20, кв. 10
18	Макарова Ольга Алексеевна	+7(961)555-66-77	makarova@gmail.com	г. Красноярск, ул. Мира, д. 30, кв. 5
19	Степанов Максим Сергеевич	+7(917)123-45-67	stepanov@mail.ru	г. Пермь, ул. Сибирская, д. 40, кв. 15
20	Романова Елена Викторовна	+7(901)789-01-23	romanova@yandex.ru	г. Челябинск, ул. Труда, д. 50, кв. 20
21	Васильев Сергей Николаевич	+7(923)111-22-33	vasiliev@gmail.com	г. Волгоград, ул. 9 Января, д. 15, кв. 12
22	Иванова Екатерина Александровна	+7(950)444-55-66	ivanova@mail.ru	г. Тюмень, ул. Республики, д. 100, кв. 35
23	Петров Александр Дмитриевич	+7(915)222-33-44	petrov@yandex.ru	г. Саратов, ул. Московская, д. 20, кв. 10
24	Сидорова Мария Сергеевна	+7(934)555-66-77	sidorova@gmail.com	г. Воронеж, ул. 9 Января, д. 30, кв. 5
25	Кузнецов Алексей Николаевич	+7(906)123-45-67	kuznecov@mail.ru	г. Нижний Новгород, ул. Рождественская, д. 40, кв. 15
26	Смирнова Ольга Владимировна	+7(911)789-01-23	smirnova@yandex.ru	г. Краснодар, ул. Северная, д. 50, кв. 20
27	Попов Дмитрий Сергеевич	+7(903)111-22-34	popov@gmail.com	г. Челябинск, ул. Цвиллинга, д. 15, кв. 12
28	Соколова Елена Александровна	+7(987)444-55-66	sokolova@mail.ru	г. Ростов-на-Дону, ул. Большая Садовая, д. 100, кв. 35
29	Васильев Андрей Викторович	+7(922)222-33-44	vasiliev2@yandex.ru	г. Воронеж, ул. Кольцовская, д. 20, кв. 10
30	Григорьева Ирина Петровна	+7(931)555-66-77	grigorieva@gmail.com	г. Екатеринбург, ул. 8 Марта, д. 30, кв. 5

Рис. А. 2. Таблица «Клиент»

Id_M...	Наименование	Стоимость	СрокГодности	Количество	ЕдиницаИзмерения
1	Амоксициллин	150	24	100	шт
2	Цефтриаксон	155	36	105	флак
3	Вольтерен	50	36	224	уп
4	Ибупрофен	60	36	73	шт
5	Диклофенак	100	36	100	шт
6	Фуразолидон	80	24	75	шт
7	Нитроглицерин	120	24	60	шт
8	Аспирин	30	36	300	шт
9	Инсулин	500	12	20	флак
10	Глюкоза	40	36	0	уп
11	Спазмолгон	60	36	150	шт
12	Кот Альфа*	40	36	200	шт
13	Корвалол	50	36	100	флак
14	Анальгин	40	36	150	шт
15	Супрастин	70	36	100	шт
16	Лоратадин	100	36	50	шт
17	Кларитин	120	36	50	шт
18	Ципрофлоксацин	150	24	75	флак
19	Азитромицин	200	24	50	шт
20	Доксициклин	100	24	100	шт
21	Метронидазол	50	36	150	шт
22	Флуконазол	120	24	50	шт
23	Итраконазол	150	24	50	шт
24	Кетоконазол	80	24	75	шт
25	Омепразол	100	36	100	шт
26	Ранитидин	60	36	150	шт
27	Фамотидин	80	36	100	шт
28	Панкреатин	100	36	100	шт
29	Фестал	120	36	50	шт
30	Эрсефурил	150	24	75	шт

Рис. А. 3. Таблица «Медикамент»

Id_Питомца	Id_Клиента	Кличка	Порода	Окрас	Пол	Возраст
1	1	Барсик	Шотландская вислоухая	Серый	М	7
2	2	Рекс	Лабрадор	Черный	М	7
3	3	Муська	Сиамская	Рыжая	Ж	4
4	4	Боня	Йоркширский терьер	Бежевый	Ж	8
5	5	Рыжик	Британская короткошерстная	Оранжевый	М	1
6	6	Тоша	Немецкая овчарка	Черно-коричневый	М	4
7	7	Белка	Персидская	Белая	Ж	6
8	8	Лайма	Пудель	Абрикосовый	Ж	7
9	9	Черныш	Черная кошка	Черный	М	9
10	10	Джек	Такса	Коричневый	М	2
11	11	Мася	Сибирская	Серая	Ж	4
12	12	Шарик	Золотистый ретривер	Золотистый	М	10
13	13	Бэлла	Сфинкс	Розовый	Ж	1
14	14	Зорро	Ротвейлер	Черный	М	6
15	15	Снежок	Белый ангорский	Белый	М	3
16	16	Леди	Бигль	Трехцветный	Ж	5
17	17	Мурка	Шотландская вислоухая	Серая	Ж	2
18	18	Барни	Сибирский хаски	Черно-белый	М	4
19	19	Мила	Персидская	Белая	Ж	8
20	20	Макс	Лабрадор	Черный	М	7
21	21	Пушок	Британская короткошерстная	Голубой	М	1
22	22	Тимка	Йоркширский терьер	Коричневый	М	3
23	23	Ляля	Сиамская	Рыжая	Ж	5
24	24	Джой	Немецкая овчарка	Черно-подпалый	Ж	6
25	25	Кузя	Персидская	Белый	М	9
26	26	Филя	Пудель	Белый	М	2
27	27	Сэм	Сибирская	Серая	М	4
28	28	Рома	Золотистый ретривер	Золотистый	М	7
29	29	Киса	Шотландская вислоухая	Серая	Ж	1
30	30	Бася	Бигль	Трехцветный	Ж	3
31	31	Мурзик	Сфинкс	Кремовый	М	2
32	32	Чарли	Ротвейлер	Черно-подпалый	М	5
33	33	Флаффи	Белый ангорский	Белый	Ж	4
34	34	Рокси	Бигль	Трехцветный	Ж	6
35	35	Пушистик	Шотландская вислоухая	Черный	М	3
36	36	Буч	Сибирский хаски	Серо-белый	М	7
37	37	Мими	Персидская	Золотистая	Ж	8
38	38	Дюк	Лабрадор	Шоколадный	М	2
39	39	Грей	Британская короткошерстная	Серый	М	5
40	40	Чип	Йоркширский терьер	Серебристый	М	4

Рис. А. 4. Таблица «Питомец»

Id_Приема	Id_Сотрудника	Id_Питомца	Дата_время	Комментарий
1	1	1	2025-03-08 10:00:...	Профилактический осмотр. Прививка от бешенства. Витаминизация. Капельницы.
2	2	2	2025-04-12 15:30:...	Жалобы на зуд и покраснение кожи. Диагноз: аллергический дерматит.
3	3	3	2025-05-05 11:15:...	Стрижка когтей и чистка ушей.
4	4	4	2025-06-01 13:45:...	Чипирование и регистрация в базе данных животных.
5	5	5	2025-07-10 09:30:...	Осмотр после операции по стерилизации.
6	6	6	2025-08-15 14:00:...	Кастрация. Назначены антибиотики.
7	7	7	2025-09-20 10:15:...	Вакцинация от чумы и парвовирусного энтерита.
8	8	8	2025-10-25 12:30:...	Удаление зубного камня.
9	9	9	2025-11-30 08:45:...	Профилактический осмотр. Жалоб нет.
10	10	10	2025-12-14 14:15:...	Консультация по кормлению и уходу.
11	1	11	2025-01-04 10:45:...	Стрижка и купание.
12	2	12	2025-02-08 16:00:...	Лечение отита.
13	3	13	2025-03-12 12:15:...	Профилактическая дегельминтизация.
14	4	14	2025-04-19 14:30:...	Вакцинация от бешенства и лептоспироза.
15	5	15	2025-05-24 10:00:...	Осмотр после травмы лапы.
16	6	16	2025-06-28 15:45:...	Кастрация. Назначены антибиотики и обезболивающие.
17	7	17	2025-07-23 11:30:...	Вакцинация от калицивироза и ринотрахеита.
18	8	18	2025-08-27 13:15:...	Удаление зубного камня и полировка зубов.
19	9	19	2025-09-11 09:00:...	Контрольный осмотр после лечения дерматита.
20	10	20	2025-10-16 14:45:...	Консультация по кормлению и содержанию.
21	1	21	2025-11-21 11:00:...	Стрижка и тримминг.
22	2	22	2025-12-26 16:30:...	Лечение конъюнктивита.
23	3	23	2025-01-10 12:45:...	Профилактическая вакцинация.
24	4	24	2025-02-14 14:00:...	Чипирование и регистрация в базе данных животных.
25	5	25	2025-03-19 10:30:...	Осмотр и консультация по уходу за шерстью.
26	6	26	2025-04-23 15:00:...	Кастрация. Назначены антибиотики и обезболивающие.
27	7	27	2025-05-28 11:15:...	Вакцинация от бешенства и лептоспироза.
28	8	28	2025-06-12 13:30:...	Удаление зубного камня и полировка зубов.
29	9	29	2025-07-17 09:45:...	Контрольный осмотр после лечения дерматита.
30	10	30	2025-08-21 15:15:...	Консультация по кормлению и поведению.
31	1	31	2025-09-25 10:00:...	Профилактический осмотр. Прививка от бешенства.
32	2	32	2025-10-29 15:30:...	Жалобы на зуд и покраснение кожи. Диагноз: аллергический дерматит. Назначено лечение.
33	3	33	2025-11-02 11:15:...	Стрижка когтей и чистка ушей.
34	4	34	2025-12-06 13:45:...	Чипирование и регистрация в базе данных животных.
35	5	35	2026-01-10 09:30:...	Осмотр после операции по стерилизации.
36	6	36	2026-02-14 14:00:...	Кастрация. Назначены антибиотики.
37	7	37	2026-03-20 10:15:...	Вакцинация от чумы и парвовирусного энтерита.
38	8	38	2026-04-24 12:30:...	Удаление зубного камня.
39	9	39	2026-05-28 08:45:...	Профилактический осмотр. Жалоб нет.
40	10	40	2026-06-01 14:15:...	Консультация по кормлению и уходу.

1 для 125 Ячейка доступна только для чтения.

Рис. А. 5. Таблица «Прием»

Id_СкладскойОперации	Id_Сотрудника	Id_Медикамента	Id_Приема	Дата_время	Операция	КоличествоЕдиниц	ЕдиницаИзмерения
1	1	1	20	2024-12-04 12:40:...	Выдача	10	шт
2	2	2	21	2024-12-04 12:40:...	Выдача	5	амп
3	3	3	22	2024-12-04 12:40:...	Выдача	1	уп
4	4	4	23	2024-12-04 12:40:...	Выдача	3	шт
5	5	5	24	2024-12-04 12:40:...	Выдача	1	флак
6	6	6	25	2024-12-04 12:40:...	Выдача	2	уп
7	7	7	26	2024-12-04 12:40:...	Выдача	10	шт
8	8	8	27	2024-12-04 12:40:...	Списание	5	амп
9	9	9	28	2024-12-04 12:40:...	Получение	20	шт
10	10	10	29	2024-12-04 12:40:...	Получение	10	амп
11	11	1	30	2024-12-04 12:40:...	Выдача	5	шт
12	12	2	19	2024-12-04 12:40:...	Выдача	3	амп
13	13	3	18	2024-12-04 12:40:...	Выдача	2	уп
14	14	4	17	2024-12-04 12:40:...	Выдача	1	шт
15	15	5	16	2024-12-04 12:40:...	Выдача	1	флак
16	16	6	15	2024-12-04 12:40:...	Выдача	1	уп
17	17	7	14	2024-12-04 12:40:...	Выдача	7	шт
18	18	8	13	2024-12-04 12:40:...	Списание	3	амп
19	19	9	12	2024-12-04 12:40:...	Получение	15	шт
20	20	10	11	2024-12-04 12:40:...	Получение	5	амп
21	21	1	1	2024-12-04 12:40:...	Выдача	10	шт
22	22	2	2	2024-12-04 12:40:...	Выдача	5	амп
23	23	3	3	2024-12-04 12:40:...	Выдача	1	уп
24	24	4	4	2024-12-04 12:40:...	Выдача	3	шт
25	25	5	5	2024-12-04 12:40:...	Выдача	1	флак
26	26	6	6	2024-12-04 12:40:...	Выдача	2	уп
27	27	7	7	2024-12-04 12:40:...	Выдача	10	шт
28	28	8	8	2024-12-04 12:40:...	Списание	5	амп
29	29	9	9	2024-12-04 12:40:...	Получение	20	шт
30	30	10	10	2024-12-04 12:40:...	Получение	10	амп
31	31	1	31	2024-12-04 12:40:...	Выдача	8	шт
32	32	2	32	2024-12-04 12:40:...	Выдача	4	амп
33	33	3	33	2024-12-04 12:40:...	Выдача	2	уп
34	34	4	34	2024-12-04 12:40:...	Выдача	5	шт
35	35	5	35	2024-12-04 12:40:...	Выдача	1	флак
36	36	6	36	2024-12-04 12:40:...	Выдача	3	уп
37	37	7	37	2024-12-04 12:40:...	Выдача	12	шт
38	38	8	38	2024-12-04 12:40:...	Списание	6	амп
39	39	9	39	2024-12-04 12:40:...	Получение	25	шт
40	40	10	40	2024-12-04 12:40:...	Получение	12	амп

1 для 115 Ячейка доступна только для чтения.

Рис. А. 6. Таблица «СкладскаяОперация»

Id_Сотрудника	ФИО	Должность	ДатаРождения	ПаспортныеД...	АдресЭлектроннойПочты	Телефон	Id_Графика
1	Иванов Иван Иванович	Врач ветеринар	2003-11-01	11111111119	ivanov@vetclinic.ru	+7(916)123-45-67	1
2	Петрова Мария Сергеевна	Ветеринарный фельдшер	2003-11-02	1111111112	petrova@vetclinic.ru	+7(925)789-01-23	2
3	Сидоров Андрей Николаевич	Ассистент ветеринара	2003-11-03	1111111113	sidorov@vetclinic.ru	+7(903)111-22-33	3
4	Кузнецова Анна Владимировна	Администратор	2003-11-04	1111111114	kuznecova@vetclinic.ru	+7(985)444-55-66	4
5	Смирнов Сергей Александрович	Уборщик	2003-11-05	1111111115	smirnov@vetclinic.ru	+7(911)222-33-44	5
6	Попова Елена Дмитриевна	Врач ветеринар	2003-11-06	1111111116	popova@vetclinic.ru	+7(905)555-66-77	6
7	Соколов Алексей Викторович	Ветеринарный фельдшер	2003-11-07	1111111117	sokolov.dp@warehouse	+7(926)123-45-68	7
8	Васильева Ольга Михайловна	Ассистент ветеринара	2003-11-08	1111111118	vasiljeva@vetclinic.ru	+7(913)789-01-23	8
9	Григорьев Дмитрий Алексеевич	Врач ветеринар	2003-11-09	1111111119	grigoriev@vetclinic.ru	+7(908)111-22-33	9
10	Федорова Светлана Николаевна	Ветеринарный фельдшер	2003-11-10	1111111121	fedorova@vetclinic.ru	+7(963)444-55-66	10
11	Борисов Александр Сергеевич	Администратор	2003-11-11	1111111131	borisov@vetclinic.ru	+7(912)222-33-44	11
12	Михайлова Наталья Владимировна	Уборщик	2003-11-12	1111111141	mihajlova@vetclinic.ru	+7(937)555-66-77	12
13	Козлов Павел Александрович	Врач ветеринар	2003-11-13	1111111151	kozlov@vetclinic.ru	+7(909)123-45-67	13
14	Ерофеева Ирина Сергеевна	Ветеринарный фельдшер	2003-11-14	1111111161	erofeeva@vetclinic.ru	+7(918)789-01-23	14
15	Морозов Виктор Дмитриевич	Ассистент ветеринара	2003-11-15	1111111171	morozov@vetclinic.ru	+7(902)111-22-33	15
16	Зайцева Татьяна Ивановна	Администратор	2003-11-16	1111111181	zajceva@vetclinic.ru	+7(953)444-55-66	14
17	Кудряшов Андрей Петрович	Уборщик	2003-11-17	1111111191	kudryashov@vetclinic.ru	+7(921)222-33-44	13
18	Макарова Ольга Алексеевна	Врач ветеринар	2003-11-18	1111111211	makarova@vetclinic.ru	+7(961)555-66-77	12
19	Степанов Максим Сергеевич	Ветеринарный фельдшер	2003-11-19	1111111311	stepanov@vetclinic.ru	+7(917)123-45-67	11
20	Романова Елена Викторовна	Администратор	2003-11-20	1111111411	romanova@vetclinic.ru	+7(901)789-01-23	10
21	Орлов Игорь Николаевич	Уборщик	2003-11-21	1111111511	orlov@vetclinic.ru	+7(922)111-22-33	9
22	Тихонова Виктория Сергеевна	Врач ветеринар	2003-11-22	1111111611	tikhonova@vetclinic.ru	+7(964)444-55-66	8
23	Савельев Денис Андреевич	Ветеринарный фельдшер	2003-11-23	1111111711	savelev@vetclinic.ru	+7(919)222-33-44	7
24	Фролова Наталья Михайловна	Ассистент ветеринара	2003-11-24	1111111811	frolovaa@vetclinic.ru	+7(954)555-66-77	6
25	Киселёв Артём Викторович	Администратор	2003-11-25	1111111911	kiselev@vetclinic.ru	+7(923)123-45-67	5
26	Яковлева Ольга Сергеевна	Уборщик	2003-11-26	1111112111	yakovleva@vetclinic.ru	+7(962)789-01-23	4
27	Гаврилов Павел Дмитриевич	Врач ветеринар	2003-11-27	1111112211	gavrilov@vetclinic.ru	+7(910)111-22-33	3
28	Рябова Анастасия Ивановна	Ветеринарный фельдшер	2003-11-28	1111112311	ryabova@vetclinic.ru	+7(965)444-55-66	2
29	Лазарев Сергей Петрович	Ассистент ветеринара	2003-11-29	1111112411	lazarev@vetclinic.ru	+7(920)222-33-44	1
30	Беляева Елена Владимировна	Администратор	2003-11-30	1111112511	belyaeva@vetclinic.ru	+7(955)555-66-77	1
31	Крылов Николай Андреевич	Уборщик	2003-12-01	1111112611	krylov@vetclinic.ru	+7(939)184-22-33	2
32	Зуева Марина Сергеевна	Врач ветеринар	2003-12-02	1111112711	zueva@vetclinic.ru	+7(963)444-55-16	3
33	Никифоров Дмитрий Викторович	Ветеринарный фельдшер	2003-12-03	1111112811	nikiforov@vetclinic.ru	+7(918)222-33-44	4
34	Демидова Анна Михайловна	Ассистент ветеринара	2003-12-04	1111112911	demidova@vetclinic.ru	+7(966)555-66-77	5
35	Семёнов Алексей Иванович	Администратор	2003-12-05	1111113111	semenov@vetclinic.ru	+7(925)123-45-67	6
36	Лебедева Ольга Сергеевна	Уборщик	2003-12-06	1111113211	lebedeva@vetclinic.ru	+7(963)789-01-29	7
37	Егоров Виктор Николаевич	Врач ветеринар	2003-12-07	1111113311	egorov@vetclinic.ru	+7(912)111-22-33	8
38	Кудрявцева Анастасия Викторовна	Ветеринарный фельдшер	2003-12-08	1111113411	kudryavtseva@vetclinic.ru	+7(967)444-55-66	9
39	Фомин Павел Дмитриевич	Ассистент ветеринара	2003-12-09	1111113511	fomin@vetclinic.ru	+7(929)222-33-44	10
40	Терентьева Наталья Ивановна	Администратор	2003-12-10	1111113611	terenteva@vetclinic.ru	+7(956)555-66-77	11

Рис. А. 7. Таблица «Сотрудник»

Id_Услуги	Название
1	Анализ крови
2	Анализ мочи
3	Вакцинация
4	Глистогонное
5	Груминг
6	Кастрация/стерилизация
7	Клиппинг
8	Консультация ветеринара
9	Лечение аллергий
22	Лечение заболеваний глаз
10	Лечение заболеваний дыхательной системы
23	Лечение заболеваний желудочно-кишечного тракта
11	Лечение заболеваний кожи
12	Лечение заболеваний крови
13	Лечение заболеваний мочевыделительной системы
14	Лечение заболеваний нервной системы
15	Лечение заболеваний опорно-двигательного аппарата
16	Лечение заболеваний печени
17	Лечение заболеваний почек
18	Лечение заболеваний ротовой полости
19	Лечение заболеваний сердечно-сосудистых
20	Лечение заболеваний ушей
21	Лечение заболеваний эндокринных заболеваний
24	Лечение инфекций
25	Лечение инфекций дыхательных путей
26	Лечение инфекций мочевыводящих путей
27	Лечение инфекций нервной системы
28	Лечение инфекций опорно-двигательного аппарата
29	Лечение инфекций пищеварительной системы
30	Лечение инфекций сердечно-сосудистой системы
31	Лечение инфекций эндокринной системы
32	Лечение кожных заболеваний
33	Лечение онкологических заболеваний
34	Лечение паразитов
35	Осмотр глаз
36	Осмотр кожи
37	Осмотр ушей

Рис. А. 8. Таблица «Услуга»

Id_Приема	Id_Услуги
1	1
1	3
1	4
2	2
3	3
4	4
5	5
6	6
7	7
8	1
8	4
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16
17	17
18	18
19	19
20	20
21	21
22	22
23	23
24	24
25	25
26	26
27	27
28	28
29	29
30	30
31	1
32	2
33	3
34	4
35	5
36	6
37	7
1	для 111

Рис. А. 9. Таблица «УслугаПриема»

ПРИЛОЖЕНИЕ Б

SQL-ОПЕРАТОРЫ СОЗДАНИЯ ОСНОВНЫХ ОБЪЕКТОВ БД

1. Код представления Питомцы_С_ФИО:

```
CREATE VIEW Питомцы_с_ФИО AS
SELECT
   Питомец.Id_Питомца,
   Клиент.ФИО AS ФИО_Клиента,
   Питомец.Кличка,
   Питомец.Порода,
   Питомец.Окрас,
   Питомец.Пол,
   Питомец.Возраст
FROM
   Питомец
JOIN
   Клиент ON Питомец.Id_Клиента = Клиент.Id_Клиента;
```

2. Код представления Представление_СкладскаяОперация:

```
CREATE VIEW Представление_СкладскаяОперация AS
SELECT
   СкладскаяОперация.Id_СкладскойОперации,
   Сотрудник.ФИО AS ФИО_Сотрудника,
   Медикамент.Наименование AS Наименование_Медикамента,
   СкладскаяОперация.Id_Приема,
   СкладскаяОперация.Дата_время,
   СкладскаяОперация.Операция,
   СкладскаяОперация.КоличествоЕдиниц,
   СкладскаяОперация.ЕдиницаИзмерения
FROM
   СкладскаяОперация
JOIN
   Сотрудник ON СкладскаяОперация.Id_Сотрудника = Сотрудник.Id_Сотрудника
JOIN
   Медикамент ON СкладскаяОперация.Id_Медикамента = Медикамент.Id_Медикамента;
```

3. Код представления ПолнаяИнформацияОПриемах:

```
CREATE VIEW ПолнаяИнформацияОПриемах AS
SELECT
   p.Id_Приема,
   s.ФИО AS ФИО_Сотрудника,
   pet.Кличка AS Кличка_Питомца,
   c.ФИО AS ФИО_Клиента,
   p.Дата_время,
   p.Комментарий,
   u.Название AS Оказанная_услуга
FROM
```

```
   dbo.Прием AS p
INNER JOIN
   dbo.Сотрудник AS s ON p.Id_Сотрудника = s.Id_Сотрудника
INNER JOIN
   dbo.Питомец AS pet ON p.Id_Питомца = pet.Id_Питомца
INNER JOIN
   dbo.Клиент AS c ON pet.Id_Клиента = c.Id_Клиента
LEFT JOIN  -- Используем LEFT JOIN вместо INNER JOIN
   dbo.УслугаПриема AS up ON p.Id_Приема = up.Id_Приема
LEFT JOIN  -- Используем LEFT JOIN для связки с услугой
   dbo.Услуга AS u ON up.Id_Услуги = u.Id_Услуги;
```

4. Код представления V_Ветеринары_График:

```
CREATE VIEW V_Ветеринары_График AS
SELECT
   Сотрудник.ФИО,
   Сотрудник.Должность,
   График.Понедельник,
   График.Вторник,
   График.Среда,
   График.Четверг,
   График.Пятница,
   График.Суббота,
   График.Воскресенье
FROM
   Сотрудник
INNER JOIN
   График ON Сотрудник.Id_Графика = График.Id_Графика
WHERE
   LOWER(Сотрудник.Должность) LIKE N'%ветеринар%';
```

5. Код представления СотрудникиСкладаСГрафиком:

```
CREATE VIEW СотрудникиСкладаСГрафиком AS
SELECT
   s.ФИО,
   g.Понедельник,
   g.Вторник,
   g.Среда,
   g.Четверг,
   g.Пятница,
   g.Суббота,
   g.Воскресенье
FROM
   Сотрудник s
JOIN
   График g ON s.Id_Графика = g.Id_Графика
WHERE
   s.Должность IN (N'Заведующий складом', N'Складской оператор', N'Кладовщик',
N'Специалист по учету товаров');
```

6. Код хранимой процедуры AddClient:

```
CREATE PROCEDURE [dbo].[AddClient]
   @ФИО NVARCHAR(50),
   @Контактный_телефон VARCHAR(20),
   @Email NVARCHAR(50),
   @Адрес NVARCHAR(100)
AS
BEGIN
   BEGIN TRY
      -- Проверка уникальности ФИО
      IF EXISTS (SELECT 1 FROM Клиент WHERE ФИО = @ФИО)
      BEGIN
         RAISERROR(N'Нарушение уникальности: ФИО уже существует.', 16, 1);
         RETURN;
      END

      -- Проверка уникальности контактного телефона
      IF EXISTS (SELECT 1 FROM Клиент WHERE Контактный_телефон =
@Контактный_телефон)
      BEGIN
         RAISERROR(N'Нарушение уникальности: контактный телефон уже существует.',
16, 1);
         RETURN;
      END

      -- Проверка уникальности Email
      IF EXISTS (SELECT 1 FROM Клиент WHERE Email = @Email)
      BEGIN
         RAISERROR(N'Нарушение уникальности: Email уже существует.', 16, 1);
         RETURN;
      END

      -- Проверка уникальности адреса
      IF EXISTS (SELECT 1 FROM Клиент WHERE Адрес = @Адрес)
      BEGIN
         RAISERROR(N'Нарушение уникальности: адрес уже существует.', 16, 1);
         RETURN;
      END

      -- Вставка данных в таблицу
      INSERT INTO Клиент (ФИО, Контактный_телефон, Email, Адрес)
      VALUES (@ФИО, @Контактный_телефон, @Email, @Адрес);

   END TRY
   BEGIN CATCH
      -- Обработка ошибки
      DECLARE @ErrorMessage NVARCHAR(400);
      DECLARE @ErrorSeverity INT;
      DECLARE @ErrorState INT;

      -- Получение информации об ошибке
      SELECT @ErrorMessage = ERROR_MESSAGE(),
```

```
            @ErrorSeverity = ERROR_SEVERITY(),
            @ErrorState = ERROR_STATE();

        -- Ретрансляция ошибки в приложение
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END;
```

7. Код хранимой процедуры UpdateClient:

```
CREATE PROCEDURE [dbo].[UpdateClient]
    @ФИО NVARCHAR(50),                    -- ФИО клиента, по которому будет выполняться обновление
    @Контактный_телефон VARCHAR(20),  -- Новый контактный телефон
    @Email NVARCHAR(50),              -- Новый Email
    @Адрес NVARCHAR(100)              -- Новый адрес
AS
BEGIN
    BEGIN TRY
        -- Проверка уникальности нового контактного телефона
        IF EXISTS (SELECT 1 FROM Клиент WHERE Контактный_телефон = @Контактный_телефон AND ФИО != @ФИО)
        BEGIN
            RAISERROR(N'Нарушение уникальности: контактный телефон уже существует для другого клиента.', 16, 1);
            RETURN;
        END

        -- Проверка уникальности нового Email
        IF EXISTS (SELECT 1 FROM Клиент WHERE Email = @Email AND ФИО != @ФИО)
        BEGIN
            RAISERROR(N'Нарушение уникальности: Email уже существует для другого клиента.', 16, 1);
            RETURN;
        END

        -- Проверка уникальности нового адреса
        IF EXISTS (SELECT 1 FROM Клиент WHERE Адрес = @Адрес AND ФИО != @ФИО)
        BEGIN
            RAISERROR(N'Нарушение уникальности: адрес уже существует для другого клиента.', 16, 1);
            RETURN;
        END

        -- Обновление данных клиента по уникальному ФИО
        UPDATE Клиент
        SET
            Контактный_телефон = @Контактный_телефон,
            Email = @Email,
            Адрес = @Адрес
        WHERE ФИО = @ФИО;
```

```
        -- Проверка успешности обновления
        IF @@ROWCOUNT = 0
        BEGIN
            RAISERROR(N'Клиент с таким ФИО не найден.', 16, 1);
            RETURN;
        END
        ELSE
        BEGIN
            PRINT N'Данные клиента успешно обновлены.';
        END
    END TRY
    BEGIN CATCH
        -- Обработка ошибки
        DECLARE @ErrorMessage NVARCHAR(400);
        DECLARE @ErrorSeverity INT;
        DECLARE @ErrorState INT;

        -- Получение информации об ошибке
        SELECT @ErrorMessage = ERROR_MESSAGE(),
               @ErrorSeverity = ERROR_SEVERITY(),
               @ErrorState = ERROR_STATE();

        -- Ретрансляция ошибки в приложение
        RAISERROR (@ErrorMessage, @ErrorSeverity, @ErrorState);
    END CATCH
END;
```

8. Код хранимой процедуры AddSotr:

```
CREATE PROCEDURE [dbo].[AddSotr]
    @ФИО NVARCHAR(50),
    @Должность NVARCHAR(50) = N'Ветеринар',
    @ДатаРождения DATE,
    @ПаспортныеДанные VARCHAR(20),
    @АдресЭлектроннойПочты NVARCHAR(50),
    @Телефон VARCHAR(20),
    @Id_Графика INT
AS
BEGIN
    -- Проверка уникальности ФИО, паспортных данных, email и телефона
    IF EXISTS (SELECT 1 FROM Сотрудник WHERE ФИО = @ФИО)
    BEGIN
        RAISERROR(N'Сотрудник с таким ФИО уже существует.', 16, 1);
        RETURN;
    END
            -- Проверка достижения совершеннолетия
    IF @ДатаРождения > DATEADD(YEAR, -18, GETDATE())
    BEGIN
        RAISERROR (N'Сотрудник должен быть старше 18 лет.', 16, 1);
        RETURN;
    END;
```

```
   IF EXISTS (SELECT 1 FROM Сотрудник WHERE ПаспортныеДанные = @ПаспортныеДанные)
   BEGIN
      RAISERROR(N'Сотрудник с такими паспортными данными уже существует.', 16, 1);
      RETURN;
   END

   IF EXISTS (SELECT 1 FROM Сотрудник WHERE АдресЭлектроннойПочты = @АдресЭлектроннойПочты)
   BEGIN
      RAISERROR(N'Сотрудник с таким адресом электронной почты уже существует.', 16, 1);
      RETURN;
   END

   IF EXISTS (SELECT 1 FROM Сотрудник WHERE Телефон = @Телефон)
   BEGIN
      RAISERROR(N'Сотрудник с таким телефоном уже существует.', 16, 1);
      RETURN;
   END
         -- Проверяем, существует ли график с переданным Id_Графика
   IF NOT EXISTS (SELECT 1 FROM График WHERE Id_Графика = @Id_Графика)
   BEGIN
      PRINT N'Ошибка: График с таким Id не найден.';
      RETURN;
   END

   -- Добавляем нового сотрудника
   INSERT INTO Сотрудник (ФИО, Должность, ДатаРождения, ПаспортныеДанные, АдресЭлектроннойПочты, Телефон, Id_Графика)
   VALUES (@ФИО, @Должность, @ДатаРождения, @ПаспортныеДанные, @АдресЭлектроннойПочты, @Телефон, @Id_Графика);

   PRINT N'Сотрудник успешно добавлен.';
END;
```

9. Код хранимой процедуры UpdateSotrByFIO:

```
CREATE PROCEDURE [dbo].[UpdateSotrByFIO]
   @ФИО NVARCHAR(50),
   @Должность NVARCHAR(50),
   @ДатаРождения DATE,
   @ПаспортныеДанные VARCHAR(20),
   @АдресЭлектроннойПочты NVARCHAR(50),
   @Телефон VARCHAR(20),
   @Id_Графика INT
AS
BEGIN
   SET NOCOUNT ON;

   -- Проверка наличия сотрудника с указанным ФИО
```

```
IF NOT EXISTS (SELECT 1 FROM Сотрудник WHERE ФИО = @ФИО)
BEGIN
    RAISERROR (N'Сотрудник с указанным ФИО не найден.', 16, 1);
    RETURN;
END;

-- Проверка достижения совершеннолетия
IF @ДатаРождения > DATEADD(YEAR, -18, GETDATE())
BEGIN
    RAISERROR (N'Сотрудник должен быть старше 18 лет.', 16, 1);
    RETURN;
END;

-- Проверка уникальности ПаспортныхДанных
IF EXISTS (SELECT 1 FROM Сотрудник WHERE ПаспортныеДанные =
@ПаспортныеДанные AND ФИО != @ФИО)
BEGIN
    RAISERROR (N'Паспортные данные должны быть уникальными.', 16, 1);
    RETURN;
END;

-- Проверка уникальности АдресаЭлектронноймПочты
IF EXISTS (SELECT 1 FROM Сотрудник WHERE АдресЭлектронноймПочты =
@АдресЭлектронноймПочты AND ФИО != @ФИО)
BEGIN
    RAISERROR (N'Адрес электронной почты должен быть уникальным.', 16, 1);
    RETURN;
END;

-- Проверка уникальности Телефона
IF EXISTS (SELECT 1 FROM Сотрудник WHERE Телефон = @Телефон AND ФИО !=
@ФИО)
BEGIN
    RAISERROR (N'Телефон должен быть уникальным.', 16, 1);
    RETURN;
END;

-- Проверка существования Id_Графика
IF NOT EXISTS (SELECT 1 FROM График WHERE Id_Графика = @Id_Графика)
BEGIN
    RAISERROR (N'Указанный Id_Графика не существует.', 16, 1);
    RETURN;
END;

-- Обновление данных сотрудника
UPDATE Сотрудник
SET
    Должность = @Должность,
    ДатаРождения = @ДатаРождения,
    ПаспортныеДанные = @ПаспортныеДанные,
    АдресЭлектронноймПочты = @АдресЭлектронноймПочты,
    Телефон = @Телефон,
```

```
        Id_Графика = @Id_Графика
    WHERE ФИО = @ФИО;

    PRINT N'Данные сотрудника успешно обновлены.';
END;
```

10. Код хранимой процедуры ДобавитьПитомца:

```
CREATE PROCEDURE [dbo].[ДобавитьПитомца]
    @ФИО_Клиента NVARCHAR(50),
    @Кличка NVARCHAR(20),
    @Порода NVARCHAR(30),
    @Окрас NVARCHAR(20),
    @Пол NCHAR(1),
    @Возраст INT
AS
BEGIN
    SET NOCOUNT ON;

    DECLARE @Id_Клиента INT;

    -- Найти Id_Клиента по ФИО
    SELECT @Id_Клиента = Id_Клиента
    FROM Клиент
    WHERE ФИО = @ФИО_Клиента;

    -- Проверка существования клиента
    IF @Id_Клиента IS NULL
    BEGIN
        RAISERROR (N'Клиент с указанным ФИО не найден.', 16, 1);
        RETURN;
    END;

    -- Проверка уникальности клички в рамках одного клиента
    IF EXISTS (SELECT 1 FROM Питомец WHERE Кличка = @Кличка AND Id_Клиента =
@Id_Клиента)
    BEGIN
        RAISERROR (N'Клиент не может иметь двух животных с одинаковой кличкой.', 16,
1);
        RETURN;
    END;

    -- Добавление записи в таблицу Питомец
    INSERT INTO Питомец (Id_Клиента, Кличка, Порода, Окрас, Пол, Возраст)
    VALUES (@Id_Клиента, @Кличка, @Порода, @Окрас, @Пол, @Возраст);

    PRINT N'Питомец успешно добавлен.';
END;
```

11. Код хранимой процедуры ДобавитьМедикамент:

```
CREATE PROCEDURE [dbo].[ДобавитьМедикамент]
   @Наименование NVARCHAR(50),
   @Стоимость INT,
   @СрокГодности INT,
   @Количество INT,
   @ЕдиницаИзмерения NVARCHAR(4)
AS
BEGIN
   -- Проверка на уникальность наименования медикамента
   IF EXISTS (SELECT 1 FROM Медикамент WHERE Наименование = @Наименование)
   BEGIN
      RAISERROR(N'Медикамент с таким наименованием уже существует.', 16, 1);
      RETURN;
   END

   -- Проверка на корректность стоимости
   IF @Стоимость <= 0
   BEGIN
      RAISERROR(N'Стоимость медикамента должна быть больше 0.', 16, 1);
      RETURN;
   END

   -- Проверка на корректность срока годности
   IF @СрокГодности <= 0
   BEGIN
      RAISERROR(N'Срок годности медикамента должен быть больше 0.', 16, 1);
      RETURN;
   END

   -- Проверка на корректность количества
   IF @Количество < 0
   BEGIN
      RAISERROR(N'Количество медикамента не может быть отрицательным.', 16, 1);
      RETURN;
   END

   -- Проверка на корректность единицы измерения
   IF @ЕдиницаИзмерения NOT IN (N'шт', N'гр', N'уп', N'амп', N'флак')
   BEGIN
      RAISERROR(N'Некорректная единица измерения. Допустимые значения: шт, гр, уп,
амп, флак.', 16, 1);
      RETURN;
   END

   -- Вставка новой записи в таблицу Медикамент
   INSERT INTO Медикамент (Наименование, Стоимость, СрокГодности, Количество,
ЕдиницаИзмерения)
   VALUES (@Наименование, @Стоимость, @СрокГодности, @Количество,
@ЕдиницаИзмерения);
END;
```

12. Код хранимой процедуры ОбновитьМедикамент:

```
CREATE PROCEDURE [dbo].[ОбновитьМедикамент]
   @Наименование NVARCHAR(50),
   @Стоимость INT,
   @СрокГодности INT,
   @Количество INT,
   @ЕдиницаИзмерения NVARCHAR(4)
AS
BEGIN
   SET NOCOUNT ON;

   -- Проверка на существование медикамента с таким наименованием
   IF NOT EXISTS (SELECT 1 FROM Медикамент WHERE Наименование =
@Наименование)
   BEGIN
      RAISERROR(N'Медикамент с таким наименованием не найден в базе данных.', 16, 1);
      RETURN;
   END;

   -- Проверка на корректность новой стоимости
   IF @Стоимость <= 0
   BEGIN
      RAISERROR(N'Стоимость медикамента должна быть больше 0.', 16, 1);
      RETURN;
   END;

   -- Проверка на корректность нового количества
   IF @Количество < 0
   BEGIN
      RAISERROR(N'Количество медикамента не может быть отрицательным.', 16, 1);
      RETURN;
   END;

   -- Проверка на корректность новой единицы измерения
   IF @ЕдиницаИзмерения NOT IN (N'шт', N'гр', N'уп', N'амп', N'флак')
   BEGIN
      RAISERROR(N'Некорректная единица измерения. Допустимые значения: шт, гр, уп,
амп, флак.', 16, 1);
      RETURN;
   END;

   -- Проверка на попытку изменения срока годности
   DECLARE @ТекущийСрокГодности INT;
   SELECT @ТекущийСрокГодности = СрокГодности
   FROM Медикамент
   WHERE Наименование = @Наименование;

   IF @СрокГодности != @ТекущийСрокГодности
   BEGIN
      RAISERROR(N'Срок годности медикамента неизменяем и не может быть обновлён.',
16, 1);
      RETURN;
```

```
    END;

    -- Обновление записи о медикаменте
    UPDATE Медикамент
    SET
        Стоимость = @Стоимость,
        Количество = @Количество,
        ЕдиницаИзмерения = @ЕдиницаИзмерения
    WHERE Наименование = @Наименование;

    PRINT N'Медикамент успешно обновлён.';
END;
```

13. Код хранимой процедуры ДобавитьПриемИУслугу:

```
CREATE PROCEDURE [dbo].[ДобавитьПриемИУслугу]
    @ФИО_Сотрудника NVARCHAR(50),
    @ФИО_Клиента NVARCHAR(50),
    @Кличка_Питомца NVARCHAR(20),
    @Дата_время DATETIME,
    @Комментарий NVARCHAR(1000),
    @Оказанная_Услуга NVARCHAR(50) = NULL
AS
BEGIN
    SET NOCOUNT ON;

    -- Переменные для хранения Id
    DECLARE @Id_Сотрудника INT;
    DECLARE @Id_Клиента INT;
    DECLARE @Id_Питомца INT;
    DECLARE @Id_Приема INT;
    DECLARE @Id_Услуги INT;
          DECLARE @День_Недели NVARCHAR(20);
    DECLARE @РабочееВремя NVARCHAR(20);
    DECLARE @ВремяНачала TIME;
    DECLARE @ВремяОкончания TIME;


    -- Получение Id клиента
    SELECT @Id_Клиента = Id_Клиента
    FROM Клиент
    WHERE ФИО = @ФИО_Клиента;

    IF @Id_Клиента IS NULL
    BEGIN
        RAISERROR(N'Клиент с указанным ФИО не найден.', 16, 1);
        RETURN;
    END;

    -- Получение Id питомца
    SELECT @Id_Питомца = Id_Питомца
    FROM Питомец
```

```
WHERE Кличка = @Кличка_Питомца AND Id_Клиента = @Id_Клиента;

IF @Id_Питомца IS NULL
BEGIN
   RAISERROR(N'У данного клиента нет питомца с указанной кличкой.', 16, 1);
   RETURN;
END;

-- Получение Id сотрудника
SELECT @Id_Сотрудника = Id_Сотрудника
FROM Сотрудник
WHERE ФИО = @ФИО_Сотрудника;

IF @Id_Сотрудника IS NULL
BEGIN
   RAISERROR(N'Сотрудник с указанным ФИО не найден.', 16, 1);
   RETURN;
END;

-- Проверяем, существует ли приём с заданными параметрами
SELECT @Id_Приема = Id_Приема
FROM Прием
WHERE Id_Сотрудника = @Id_Сотрудника AND Id_Питомца = @Id_Питомца AND
Дата_время = @Дата_время;

-- Если приём уже существует
IF @Id_Приема IS NOT NULL
BEGIN
   -- Если услуга не указана
   IF @Оказанная_Услуга IS NULL
   BEGIN
      RAISERROR(N'Данный прием уже создан, а услуга не указана.', 16, 1);
      RETURN;
   END;

         -- Проверяем, прошло ли более 24 часов с момента приёма
         IF DATEDIFF(HOUR, @Дата_время, GETDATE()) > 24
         BEGIN
               RAISERROR(N'Нельзя добавлять услуги к приёму, если с его даты
прошло более 24 часов.', 16, 1);
               RETURN;
         END;

   -- Проверка существования услуги
   SELECT @Id_Услуги = Id_Услуги
   FROM Услуга
   WHERE Название = @Оказанная_Услуга;

   IF @Id_Услуги IS NULL
   BEGIN
      RAISERROR(N'Указанная услуга не найдена в базе данных.', 16, 1);
      RETURN;
```

```
        END;

        -- Проверяем, существует ли такая услуга для данного приёма
        IF EXISTS (SELECT 1 FROM УслугаПриема WHERE Id_Приема = @Id_Приема AND
Id_Услуги = @Id_Услуги)
        BEGIN
            RAISERROR(N'Данная услуга уже назначена для приёма.', 16, 1);
            RETURN;
        END;

        -- Добавляем услугу для существующего приёма
        INSERT INTO УслугаПриема (Id_Приема, Id_Услуги)
        VALUES (@Id_Приема, @Id_Услуги);

        PRINT N'Приём был создан ранее. Услуга была добавлена.';
        RETURN;
    END;

            -- Проверка, что Дата_время в будущем
    IF DATEDIFF(DAY, GETDATE(), @Дата_время) <= 1
    BEGIN
        RAISERROR(N'Прием должен быть сформирован за сутки до текущего времени.', 16,
1);
        RETURN;
    END;

            -- Получение Id сотрудника по ФИО
    SELECT @Id_Сотрудника = Id_Сотрудника
    FROM Сотрудник
    WHERE ФИО = @ФИО_Сотрудника AND Должность LIKE N'%ветеринар%';

    IF @Id_Сотрудника IS NULL
    BEGIN
        RAISERROR(N'Сотрудник с указанным ФИО не найден или не является
ветеринаром.', 16, 1);
        RETURN;
    END

            -- Определение дня недели и рабочего времени сотрудника
    SET @День_Недели = CASE DATEPART(WEEKDAY, @Дата_время)
        WHEN 1 THEN N'Воскресенье'
        WHEN 2 THEN N'Понедельник'
        WHEN 3 THEN N'Вторник'
        WHEN 4 THEN N'Среда'
        WHEN 5 THEN N'Четверг'
        WHEN 6 THEN N'Пятница'
        WHEN 7 THEN N'Суббота'
    END;

    SELECT @РабочееВремя = CASE @День_Недели
        WHEN N'Понедельник' THEN Понедельник
        WHEN N'Вторник' THEN Вторник
```

```
        WHEN N'Среда' THEN Среда
        WHEN N'Четверг' THEN Четверг
        WHEN N'Пятница' THEN Пятница
        WHEN N'Суббота' THEN Суббота
        WHEN N'Воскресенье' THEN Воскресенье
    END
    FROM График
    WHERE Id_Графика = (SELECT Id_Графика FROM Сотрудник WHERE Id_Сотрудника
= @Id_Сотрудника);

    -- Разделение рабочего времени на начало и окончание
    IF @РабочееВремя IS NULL OR CHARINDEX('-', @РабочееВремя) = 0
    BEGIN
        RAISERROR(N'Сотрудник не работает в этот день.', 16, 1);
        RETURN;
    END

    SET @ВремяНачала = CAST(LEFT(@РабочееВремя, CHARINDEX('-', @РабочееВремя)
- 1) AS TIME);
    SET @ВремяОкончания = CAST(RIGHT(@РабочееВремя, LEN(@РабочееВремя) -
CHARINDEX('-', @РабочееВремя)) AS TIME);

    -- Проверка, что время приема попадает в рабочее время!
    IF CAST(@Дата_время AS TIME) NOT BETWEEN @ВремяНачала AND
@ВремяОкончания
    BEGIN
        RAISERROR(N'Время приема не попадает в рабочее время сотрудника.', 16, 1);
        RETURN;
    END

            -- Проверка, что у сотрудника нет другого приема в это время
    IF EXISTS (SELECT 1 FROM Прием WHERE Id_Сотрудника = @Id_Сотрудника AND
Дата_время = @Дата_время)
    BEGIN
        RAISERROR(N'У сотрудника уже есть прием в это время.', 16, 1);
        RETURN;
    END

    -- Проверка, что у питомца нет другого приема в это время
    IF EXISTS (SELECT 1 FROM Прием WHERE Id_Питомца = @Id_Питомца AND
Дата_время = @Дата_время)
    BEGIN
        RAISERROR(N'У питомца уже есть прием в это время.', 16, 1);
        RETURN;
    END

    -- Если приём не существует, создаём новый
    INSERT INTO Прием (Id_Сотрудника, Id_Питомца, Дата_время, Комментарий)
    VALUES (@Id_Сотрудника, @Id_Питомца, @Дата_время, @Комментарий);

    SET @Id_Приема = SCOPE_IDENTITY();
```

```
    -- Если указана услуга, добавляем её в УслугаПриема
    IF @Оказанная_Услуга IS NOT NULL
    BEGIN
        -- Проверяем, существует ли указанная услуга
        SELECT @Id_Услуги = Id_Услуги
        FROM Услуга
        WHERE Название = @Оказанная_Услуга;

        IF @Id_Услуги IS NULL
        BEGIN
            RAISERROR(N'Указанная услуга не найдена в базе данных.', 16, 1);
            RETURN;
        END;

        INSERT INTO УслугаПриема (Id_Приема, Id_Услуги)
        VALUES (@Id_Приема, @Id_Услуги);

        PRINT N'Приём и услуга успешно добавлены.';
    END
    ELSE
    BEGIN
        PRINT N'Приём успешно добавлен.';
    END;
END;
```

14. Код хранимой процедуры ДобавитьСкладскуюОперацию:

```
CREATE PROCEDURE [dbo].[ДобавитьСкладскуюОперацию]
    @ФИО_Сотрудника NVARCHAR(100),
    @Наименование_Медикамента NVARCHAR(50),
    @Id_Приема INT = NULL,  -- Этот параметр может быть NULL для операций без привязки к приему
    @Операция NVARCHAR(10),
    @КоличествоЕдиниц INT
AS
BEGIN
    SET NOCOUNT ON;
            -- Проверка существования Id_Приема, если оно не NULL
    IF @Id_Приема IS NOT NULL
    BEGIN
        IF NOT EXISTS (SELECT 1 FROM Прием WHERE Id_Приема = @Id_Приема)
        BEGIN
            RAISERROR(N'Прием с указанным Id_Приема не существует.', 16, 1);
            RETURN;
        END
    END

    -- Проверка корректности операции
    IF @Операция NOT IN (N'Выдача', N'Списание', N'Получение')
    BEGIN
        RAISERROR(N'Неверная операция. Операция должна быть "Выдача", "Списание" или "Получение".', 16, 1);
```

```
        RETURN;
    END

    -- Проверка корректности количества единиц
    IF @КоличествоЕдиниц <= 0
    BEGIN
        RAISERROR(N'Количество единиц должно быть больше нуля.', 16, 1);
        RETURN;
    END

    -- Получаем Id_Сотрудника и проверяем должность
    DECLARE @Id_Сотрудника INT;
    DECLARE @Должность NVARCHAR(50);
    SELECT @Id_Сотрудника = Id_Сотрудника, @Должность = Должность
    FROM Сотрудник
    WHERE ФИО = @ФИО_Сотрудника;

    IF @Id_Сотрудника IS NULL
    BEGIN
        RAISERROR(N'Сотрудник с таким ФИО не найден.', 16, 1);
        RETURN;
    END

    IF @Должность NOT IN (N'Заведующий складом', N'Складской оператор',
N'Кладовщик', N'Специалист по учету товаров')
    BEGIN
        RAISERROR(N'Ошибка: Сотрудник не имеет права выполнять складские операции.',
16, 1);
        RETURN;
    END

    -- Получаем рабочее время сотрудника для текущего дня
    DECLARE @День_Недели NVARCHAR(20);
    DECLARE @РабочееВремя NVARCHAR(20);
    DECLARE @ВремяНачала TIME;
    DECLARE @ВремяОкончания TIME;

    -- Определение дня недели
    SET @День_Недели = CASE DATEPART(WEEKDAY, GETDATE())
        WHEN 1 THEN N'Воскресенье'
        WHEN 2 THEN N'Понедельник'
        WHEN 3 THEN N'Вторник'
        WHEN 4 THEN N'Среда'
        WHEN 5 THEN N'Четверг'
        WHEN 6 THEN N'Пятница'
        WHEN 7 THEN N'Суббота'
    END;

    -- Получаем рабочее время для этого дня
    SELECT @РабочееВремя = CASE @День_Недели
        WHEN N'Понедельник' THEN Понедельник
        WHEN N'Вторник' THEN Вторник
```

```
        WHEN N'Среда' THEN Среда
        WHEN N'Четверг' THEN Четверг
        WHEN N'Пятница' THEN Пятница
        WHEN N'Суббота' THEN Суббота
        WHEN N'Воскресенье' THEN Воскресенье
    END
    FROM График
    WHERE Id_Графика = (SELECT Id_Графика FROM Сотрудник WHERE Id_Сотрудника
= @Id_Сотрудника);

    -- Проверка, что сотрудник работает в этот день
    IF @РабочееВремя IS NULL OR CHARINDEX('-', @РабочееВремя) = 0
    BEGIN
        RAISERROR(N'Сотрудник не работает в этот день.', 16, 1);
        RETURN;
    END

    -- Разделение рабочего времени на начало и окончание
    SET @ВремяНачала = CAST(LEFT(@РабочееВремя, CHARINDEX('-', @РабочееВремя)
- 1) AS TIME);
    SET @ВремяОкончания = CAST(RIGHT(@РабочееВремя, LEN(@РабочееВремя) -
CHARINDEX('-', @РабочееВремя)) AS TIME);

    -- Проверка, что текущее время попадает в рабочее время сотрудника
    IF CAST(GETDATE() AS TIME) NOT BETWEEN @ВремяНачала AND
@ВремяОкончания
    BEGIN
        RAISERROR(N'Сотрудник не работает в данный момент времени.', 16, 1);
        RETURN;
    END

    -- Получаем Id_Медикамента и ЕдиницуИзмерения по наименованию
    DECLARE @Id_Медикамента INT;
    DECLARE @ЕдиницаИзмерения_Медикамента NVARCHAR(4);
    SELECT @Id_Медикамента = Id_Медикамента, @ЕдиницаИзмерения_Медикамента =
ЕдиницаИзмерения
    FROM Медикамент
    WHERE Наименование = @Наименование_Медикамента;

    IF @Id_Медикамента IS NULL
    BEGIN
        RAISERROR(N'Медикамент с таким наименованием не найден.', 16, 1);
        RETURN;
    END

    -- Проверка остатков для операций "Выдача" и "Списание"
    IF @Операция IN (N'Выдача', N'Списание')
    BEGIN
        DECLARE @КоличествоМедикамента INT;
        SELECT @КоличествоМедикамента = Количество FROM Медикамент WHERE
Id_Медикамента = @Id_Медикамента;
```

```
        IF @КоличествоМедикамента < @КоличествоЕдиниц
        BEGIN
            RAISERROR(N'Ошибка: Недостаточно медикамента для выполнения операции. Осталось %d единиц.', 16, 1, @КоличествоМедикамента);
            RETURN;
        END
    END

    -- Обновляем количество медикамента для операций "Выдача" и "Списание"
    IF @Операция IN (N'Выдача', N'Списание')
    BEGIN
        UPDATE Медикамент
        SET Количество = Количество - @КоличествоЕдиниц
        WHERE Id_Медикамента = @Id_Медикамента;
    END

    -- Обновляем количество медикамента для операции "Получение"
    IF @Операция = N'Получение'
    BEGIN
        UPDATE Медикамент
        SET Количество = Количество + @КоличествоЕдиниц
        WHERE Id_Медикамента = @Id_Медикамента;
    END

    -- Добавляем запись в таблицу СкладскаяОперация
    INSERT INTO СкладскаяОперация (Id_Сотрудника, Id_Медикамента, Id_Приема, Дата_время, Операция, КоличествоЕдиниц, ЕдиницаИзмерения)
    VALUES (@Id_Сотрудника, @Id_Медикамента, @Id_Приема, GETDATE(), @Операция, @КоличествоЕдиниц, @ЕдиницаИзмерения_Медикамента);

    -- Уведомление о успешной вставке
    PRINT N'Складская операция успешно добавлена.';
END;
```

15. Код хранимой процедуры ДобавитьГрафик:

```
CREATE PROCEDURE [dbo].[ДобавитьГрафик]
    @Понедельник NVARCHAR(20) = NULL,
    @Вторник NVARCHAR(20) = NULL,
    @Среда NVARCHAR(20) = NULL,
    @Четверг NVARCHAR(20) = NULL,
    @Пятница NVARCHAR(20) = NULL,
    @Суббота NVARCHAR(20) = NULL,
    @Воскресенье NVARCHAR(20) = NULL
AS
BEGIN
        IF NOT (
        (@Понедельник LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Понедельник IS NULL) AND
        (@Вторник LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Вторник IS NULL) AND
        (@Среда LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Среда IS NULL) AND
```

```
    (@Четверг LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Четверг IS NULL) AND
    (@Пятница LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Пятница IS NULL) AND
    (@Суббота LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Суббота IS NULL) AND
    (@Воскресенье LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Воскресенье IS NULL)
  )
  BEGIN
    RAISERROR(N'Некорректный формат времени. Ожидается формат HH:MM-HH:MM или NULL.', 16, 1);
    RETURN;
  END
  -- Проверка на существование одинакового графика в таблице
  IF EXISTS (SELECT 1 FROM График
        WHERE (Понедельник = @Понедельник OR (Понедельник IS NULL AND @Понедельник IS NULL)) AND
            (Вторник = @Вторник OR (Вторник IS NULL AND @Вторник IS NULL)) AND
            (Среда = @Среда OR (Среда IS NULL AND @Среда IS NULL)) AND
            (Четверг = @Четверг OR (Четверг IS NULL AND @Четверг IS NULL)) AND
            (Пятница = @Пятница OR (Пятница IS NULL AND @Пятница IS NULL)) AND
            (Суббота = @Суббота OR (Суббота IS NULL AND @Суббота IS NULL)) AND
            (Воскресенье = @Воскресенье OR (Воскресенье IS NULL AND @Воскресенье IS NULL)))
  BEGIN
    RAISERROR(N'График с таким временем уже существует.', 16, 1);
    RETURN;
  END

  -- Добавление записи в таблицу График
  INSERT INTO График (Понедельник, Вторник, Среда, Четверг, Пятница, Суббота, Воскресенье)
  VALUES (@Понедельник, @Вторник, @Среда, @Четверг, @Пятница, @Суббота, @Воскресенье);
END;
```

16. Код хранимой процедуры УдалитьГрафик:

```
CREATE PROCEDURE [dbo].[УдалитьГрафик]
  @Id_Графика INT = NULL,
  @Понедельник NVARCHAR(20) = NULL,
  @Вторник NVARCHAR(20) = NULL,
  @Среда NVARCHAR(20) = NULL,
  @Четверг NVARCHAR(20) = NULL,
  @Пятница NVARCHAR(20) = NULL,
  @Суббота NVARCHAR(20) = NULL,
  @Воскресенье NVARCHAR(20) = NULL
AS
BEGIN
```

```
-- Если передан Id_Графика, проверяем его существование
IF @Id_Графика IS NOT NULL
BEGIN
   IF NOT EXISTS (SELECT 1 FROM График WHERE Id_Графика = @Id_Графика)
   BEGIN
      RAISERROR(N'График с указанным Id не найден.', 16, 1);
      RETURN;
   END

   -- Проверяем соответствие Id_Графика и введенных дней недели
   IF EXISTS (
      SELECT 1 FROM График
      WHERE Id_Графика = @Id_Графика AND NOT (
         (ISNULL(Понедельник, '') = ISNULL(@Понедельник, '')) AND
         (ISNULL(Вторник, '') = ISNULL(@Вторник, '')) AND
         (ISNULL(Среда, '') = ISNULL(@Среда, '')) AND
         (ISNULL(Четверг, '') = ISNULL(@Четверг, '')) AND
         (ISNULL(Пятница, '') = ISNULL(@Пятница, '')) AND
         (ISNULL(Суббота, '') = ISNULL(@Суббота, '')) AND
         (ISNULL(Воскресенье, '') = ISNULL(@Воскресенье, ''))
      )
   )
   BEGIN
      RAISERROR(N'Id_Графика не совпадает с введенным графиком.', 16, 1);
      RETURN;
   END
END
ELSE
BEGIN
   -- Если Id_Графика не передан, ищем его по переданным дням недели
   SELECT @Id_Графика = Id_Графика
   FROM График
   WHERE
      (ISNULL(Понедельник, '') = ISNULL(@Понедельник, '')) AND
      (ISNULL(Вторник, '') = ISNULL(@Вторник, '')) AND
      (ISNULL(Среда, '') = ISNULL(@Среда, '')) AND
      (ISNULL(Четверг, '') = ISNULL(@Четверг, '')) AND
      (ISNULL(Пятница, '') = ISNULL(@Пятница, '')) AND
      (ISNULL(Суббота, '') = ISNULL(@Суббота, '')) AND
      (ISNULL(Воскресенье, '') = ISNULL(@Воскресенье, ''));

   -- Проверка наличия графика по дням недели
   IF @Id_Графика IS NULL
   BEGIN
      RAISERROR(N'График с таким временем не найден.', 16, 1);
      RETURN;
   END
END

-- Проверка, есть ли сотрудники с таким графиком
IF EXISTS (SELECT 1 FROM Сотрудник WHERE Id_Графика = @Id_Графика)
BEGIN
```

```
        RAISERROR(N'Невозможно удалить график, так как он используется сотрудниками.',
16, 1);
        RETURN;
    END

    -- Удаление графика
    DELETE FROM График WHERE Id_Графика = @Id_Графика;
END;
```

17. Код хранимой процедуры ДобавитьУслугу:

```
CREATE PROCEDURE [dbo].[ДобавитьУслугу]
    @Название NVARCHAR(50)
AS
BEGIN
    -- Проверка, что услуга с таким названием уже существует
    IF EXISTS (SELECT 1 FROM Услуга WHERE Название = @Название)
    BEGIN
        RAISERROR(N'Услуга с таким названием уже существует.', 16, 1);
        RETURN;
    END

    -- Добавление записи в таблицу Услуга
    INSERT INTO Услуга (Название)
    VALUES (@Название);

    -- Возвращение сообщения об успешном добавлении
    PRINT N'Услуга успешно добавлена';
END;
```

18. Код хранимой процедуры ОбновитьКомментарийПриема:

```
CREATE PROCEDURE [dbo].[ОбновитьКомментарийПриема]
    @ФИО_Клиента NVARCHAR(100),
    @Кличка_Питомца NVARCHAR(100),
    @ФИО_Сотрудника NVARCHAR(100),
    @Дата_время DATETIME,
    @Комментарий NVARCHAR(1000)
AS
BEGIN
    SET NOCOUNT ON;

    -- Проверка существования клиента
    DECLARE @Id_Клиента INT;
    SELECT @Id_Клиента = Id_Клиента
    FROM Клиент
    WHERE ФИО = @ФИО_Клиента;

    IF @Id_Клиента IS NULL
    BEGIN
        RAISERROR(N'Ошибка: Клиент с указанным ФИО не найден.', 16, 1);
```

```
    RETURN;
END;

-- Проверка существования питомца у клиента
DECLARE @Id_Питомца INT;
SELECT @Id_Питомца = Id_Питомца
FROM Питомец
WHERE Кличка = @Кличка_Питомца AND Id_Клиента = @Id_Клиента;

IF @Id_Питомца IS NULL
BEGIN
    RAISERROR(N'Ошибка: У клиента с указанным ФИО нет питомца с указанной кличкой.', 16, 1);
    RETURN;
END;

-- Проверка существования сотрудника
DECLARE @Id_Сотрудника INT;
SELECT @Id_Сотрудника = Id_Сотрудника
FROM Сотрудник
WHERE ФИО = @ФИО_Сотрудника;

IF @Id_Сотрудника IS NULL
BEGIN
    RAISERROR(N'Ошибка: Сотрудник с указанным ФИО не найден.', 16, 1);
    RETURN;
END;

-- Проверка существования записи о приеме
DECLARE @Id_Приема INT;
SELECT @Id_Приема = Id_Приема
FROM Прием
WHERE Id_Сотрудника = @Id_Сотрудника AND Id_Питомца = @Id_Питомца AND Дата_время = @Дата_время;

IF @Id_Приема IS NULL
BEGIN
    RAISERROR(N'Ошибка: Прием с указанными данными не найден.', 16, 1);
    RETURN;
END;

-- Проверка, прошло ли больше суток с момента приема
IF DATEDIFF(HOUR, @Дата_время, GETDATE()) > 24
BEGIN
    RAISERROR(N'Ошибка: Комментарий к приему не может быть изменен, так как прошло более суток со дня приема.', 16, 1);
    RETURN;
END;

-- Обновление комментария
UPDATE Прием
SET Комментарий = @Комментарий
```

```
    WHERE Id_Приема = @Id_Приема;

    PRINT N'Комментарий успешно обновлен.';
END;
```

19. Код хранимой процедуры УдалитьПрием:

```
CREATE PROCEDURE [dbo].[УдалитьПрием]
    @ФИО_Клиента NVARCHAR(100),
    @Кличка_Питомца NVARCHAR(100),
    @ФИО_Сотрудника NVARCHAR(100),
    @Дата_время DATETIME
AS
BEGIN
    SET NOCOUNT ON;

    -- Проверка существования клиента
    DECLARE @Id_Клиента INT;
    SELECT @Id_Клиента = Id_Клиента
    FROM Клиент
    WHERE ФИО = @ФИО_Клиента;

    IF @Id_Клиента IS NULL
    BEGIN
        RAISERROR(N'Ошибка: Клиент с указанным ФИО не найден.', 16, 1);
        RETURN;
    END;

    -- Проверка существования питомца у клиента
    DECLARE @Id_Питомца INT;
    SELECT @Id_Питомца = Id_Питомца
    FROM Питомец
    WHERE Кличка = @Кличка_Питомца AND Id_Клиента = @Id_Клиента;

    IF @Id_Питомца IS NULL
    BEGIN
        RAISERROR(N'Ошибка: У клиента с указанным ФИО нет питомца с указанной
кличкой.', 16, 1);
        RETURN;
    END;

    -- Проверка существования сотрудника
    DECLARE @Id_Сотрудника INT;
    SELECT @Id_Сотрудника = Id_Сотрудника
    FROM Сотрудник
    WHERE ФИО = @ФИО_Сотрудника;

    IF @Id_Сотрудника IS NULL
    BEGIN
        RAISERROR(N'Ошибка: Сотрудник с указанным ФИО не найден.', 16, 1);
        RETURN;
    END;

    -- Проверка существования записи о приёме
```

```
    DECLARE @Id_Приема INT;
    SELECT @Id_Приема = Id_Приема
    FROM Прием
    WHERE Id_Сотрудника = @Id_Сотрудника AND Id_Питомца = @Id_Питомца AND
Дата_время = @Дата_время;

    IF @Id_Приема IS NULL
    BEGIN
        RAISERROR(N'Ошибка: Приём с указанными данными не найден.', 16, 1);
        RETURN;
    END;

    -- Проверка, что до приёма осталось более суток
    IF DATEDIFF(HOUR, GETDATE(), @Дата_время) < 24
    BEGIN
        RAISERROR(N'Ошибка: Приём не может быть удалён, так как до него осталось менее
суток или он был завершен.', 16, 1);
        RETURN;
    END;

    -- Проверка, что приём не используется в таблице УслугаПриема
    IF EXISTS (SELECT 1 FROM УслугаПриема WHERE Id_Приема = @Id_Приема)
    BEGIN
        RAISERROR(N'Ошибка: Приём не может быть удалён, так как ему уже назначены
услуги.', 16, 1);
        RETURN;
    END;

    -- Удаление приёма
    DELETE FROM Прием WHERE Id_Приема = @Id_Приема;

    PRINT N'Приём успешно удалён.';
END;
```

20. Код хранимой процедуры ОбновитьГрафик:

```
CREATE PROCEDURE [dbo].[ОбновитьГрафик]
    @Id_Графика INT,
    @Понедельник NVARCHAR(20) = NULL,
    @Вторник NVARCHAR(20) = NULL,
    @Среда NVARCHAR(20) = NULL,
    @Четверг NVARCHAR(20) = NULL,
    @Пятница NVARCHAR(20) = NULL,
    @Суббота NVARCHAR(20) = NULL,
    @Воскресенье NVARCHAR(20) = NULL
AS
BEGIN
    -- Проверка существования записи с указанным Id
    IF NOT EXISTS (SELECT 1 FROM График WHERE Id_Графика = @Id_Графика)
    BEGIN
        RAISERROR(N'Запись с указанным Id не найдена.', 16, 1);
        RETURN;
```

```
END

-- Проверка формата времени для каждого дня
IF NOT (
    (@Понедельник LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Понедельник IS NULL) AND
    (@Вторник LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Вторник IS NULL) AND
    (@Среда LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Среда IS NULL) AND
    (@Четверг LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Четверг IS NULL) AND
    (@Пятница LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Пятница IS NULL) AND
    (@Суббота LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Суббота IS NULL) AND
    (@Воскресенье LIKE '[0-2][0-9]:[0-5][0-9]-[0-2][0-9]:[0-5][0-9]' OR @Воскресенье IS NULL)
)
BEGIN
    RAISERROR(N'Некорректный формат времени. Ожидается формат HH:MM-HH:MM или NULL.', 16, 1);
    RETURN;
END

-- Проверка на уникальность графика
IF EXISTS (
    SELECT 1 FROM График
    WHERE Id_Графика != @Id_Графика AND
        (Понедельник = @Понедельник OR (Понедельник IS NULL AND @Понедельник IS NULL)) AND
        (Вторник = @Вторник OR (Вторник IS NULL AND @Вторник IS NULL)) AND
        (Среда = @Среда OR (Среда IS NULL AND @Среда IS NULL)) AND
        (Четверг = @Четверг OR (Четверг IS NULL AND @Четверг IS NULL)) AND
        (Пятница = @Пятница OR (Пятница IS NULL AND @Пятница IS NULL)) AND
        (Суббота = @Суббота OR (Суббота IS NULL AND @Суббота IS NULL)) AND
        (Воскресенье = @Воскресенье OR (Воскресенье IS NULL AND @Воскресенье IS NULL))
)
BEGIN
    RAISERROR(N'График с таким временем уже существует.', 16, 1);
    RETURN;
END

-- Обновление записи
UPDATE График
SET Понедельник = @Понедельник,
    Вторник = @Вторник,
    Среда = @Среда,
    Четверг = @Четверг,
    Пятница = @Пятница,
    Суббота = @Суббота,
    Воскресенье = @Воскресенье
```

```
    WHERE Id_Графика = @Id_Графика;
END;
```

21. Код хранимой процедуры ОбновитьВозрастПитомца:

```
CREATE PROCEDURE ОбновитьВозрастПитомца
    @ФИО_Клиента NVARCHAR(50),
    @Кличка NVARCHAR(20),
    @Возраст INT
AS
BEGIN
    -- Объявляем переменные для хранения Id_Клиента и текущего возраста питомца
    DECLARE @Id_Клиента INT;
    DECLARE @ТекущийВозраст INT;

    -- Получаем Id_Клиента по ФИО_Клиента
    SELECT @Id_Клиента = Id_Клиента
    FROM Клиент
    WHERE ФИО = @ФИО_Клиента;

    -- Проверяем, найден ли клиент
    IF @Id_Клиента IS NULL
    BEGIN
        RAISERROR(N'Клиент с таким ФИО не найден.', 16, 1);
        RETURN;
    END

    -- Проверка, что питомец с заданной кличкой существует для этого клиента
    IF NOT EXISTS (
        SELECT 1
        FROM Питомец
        WHERE Id_Клиента = @Id_Клиента AND Кличка = @Кличка
    )
    BEGIN
        RAISERROR(N'Питомец с такой кличкой не найден у указанного клиента.', 16, 1);
        RETURN;
    END

    -- Получаем текущий возраст питомца
    SELECT @ТекущийВозраст = Возраст
    FROM Питомец
    WHERE Id_Клиента = @Id_Клиента AND Кличка = @Кличка;

    -- Проверка, что новый возраст больше текущего и не является отрицательным числом
    IF @Возраст <= 0
    BEGIN
        RAISERROR(N'Возраст не может быть меньше или равен нулю.', 16, 1);
        RETURN;
    END

    IF @Возраст <= @ТекущийВозраст
    BEGIN
```

```
        RAISERROR(N'Новый возраст должен быть больше текущего возраста питомца.', 16,
1);
        RETURN;
    END

    -- Обновляем возраст питомца
    UPDATE Питомец
    SET Возраст = @Возраст
    WHERE Id_Клиента = @Id_Клиента
      AND Кличка = @Кличка;

    -- Проверка, был ли обновлен питомец
    IF @@ROWCOUNT = 0
    BEGIN
        RAISERROR(N'Не удалось обновить возраст питомца. Проверьте данные.', 16, 1);
        RETURN;
    END

    -- Уведомление об успешном обновлении
    RAISERROR(N'Возраст питомца успешно обновлен.', 10, 1);
END;
```

22. Код хранимой процедуры УдалитьУслугуПриема:

```
CREATE PROCEDURE [dbo].[УдалитьУслугуПриема]
    @ФИО_Клиента NVARCHAR(100),
    @Кличка_Питомца NVARCHAR(100),
    @ФИО_Сотрудника NVARCHAR(100),
    @Дата_время DATETIME,
    @Оказанная_Услуга NVARCHAR(50)
AS
BEGIN
    SET NOCOUNT ON;

    -- Проверка существования клиента
    DECLARE @Id_Клиента INT;
    SELECT @Id_Клиента = Id_Клиента
    FROM Клиент
    WHERE ФИО = @ФИО_Клиента;

    IF @Id_Клиента IS NULL
    BEGIN
        RAISERROR(N'Ошибка: Клиент с указанным ФИО не найден.', 16, 1);
        RETURN;
    END;

    -- Проверка существования питомца у клиента
    DECLARE @Id_Питомца INT;
    SELECT @Id_Питомца = Id_Питомца
    FROM Питомец
    WHERE Кличка = @Кличка_Питомца AND Id_Клиента = @Id_Клиента;
```

```
IF @Id_Питомца IS NULL
BEGIN
    RAISERROR(N'Ошибка: У клиента с указанным ФИО нет питомца с указанной кличкой.', 16, 1);
    RETURN;
END;

-- Проверка существования сотрудника
DECLARE @Id_Сотрудника INT;
SELECT @Id_Сотрудника = Id_Сотрудника
FROM Сотрудник
WHERE ФИО = @ФИО_Сотрудника;

IF @Id_Сотрудника IS NULL
BEGIN
    RAISERROR(N'Ошибка: Сотрудник с указанным ФИО не найден.', 16, 1);
    RETURN;
END;

-- Проверка существования приёма
DECLARE @Id_Приема INT;
SELECT @Id_Приема = Id_Приема
FROM Прием
WHERE Id_Сотрудника = @Id_Сотрудника AND Id_Питомца = @Id_Питомца AND Дата_время = @Дата_время;

IF @Id_Приема IS NULL
BEGIN
    RAISERROR(N'Ошибка: Приём с указанными данными не найден.', 16, 1);
    RETURN;
END;

-- Проверка, что прошло не более суток со дня приёма
IF DATEDIFF(HOUR, @Дата_время, GETDATE()) > 24
BEGIN
    RAISERROR(N'Ошибка: Услуга не может быть удалена, так как прошло более суток со дня приёма.', 16, 1);
    RETURN;
END;

-- Проверка существования услуги
DECLARE @Id_Услуги INT;
SELECT @Id_Услуги = Id_Услуги
FROM Услуга
WHERE Название = @Оказанная_Услуга;

IF @Id_Услуги IS NULL
BEGIN
    RAISERROR(N'Ошибка: Услуга с указанным названием не найдена.', 16, 1);
    RETURN;
END;
```

```
    -- Проверка существования записи в таблице УслугаПриема
    IF NOT EXISTS (
        SELECT 1
        FROM УслугаПриема
        WHERE Id_Приема = @Id_Приема AND Id_Услуги = @Id_Услуги
    )
    BEGIN
        RAISERROR(N'Ошибка: На указанном приеме не оказывалась данная услуга.', 16, 1);
        RETURN;
    END;

    -- Удаление записи из таблицы УслугаПриема
    DELETE FROM УслугаПриема
    WHERE Id_Приема = @Id_Приема AND Id_Услуги = @Id_Услуги;

    PRINT N'Услуга приёма успешно удалена.';
END;
```

ПРИЛОЖЕНИЕ В

КОД ПРОГРАММЫ

1. Код программы «Основная Форма»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Итоговая_проектная_работа
{
   public partial class СтартоваяФорма : Form
   {
      public СтартоваяФорма()
      {
         InitializeComponent();
         comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
      }
      private void button1_Click_1(object sender, EventArgs e)
      {
         // Проверка, что выбрана роль в ComboBox и введен пароль
         if (string.IsNullOrWhiteSpace(comboBox1.Text) ||
string.IsNullOrWhiteSpace(textBox2.Text))
         {
            MessageBox.Show("Пожалуйста, выберите роль и введите пароль.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
         }

         // Проверка для каждой роли
         if ("Генеральный директор" == comboBox1.Text && textBox2.Text == "1234")
         {
            GlobalData.Password = textBox2.Text;
            Form ген_директор = new ГенеральныйДиректор();
            ген_директор.Show();
         }
         else if ("Администратор" == comboBox1.Text && textBox2.Text == "5678")
         {
            GlobalData.Password = textBox2.Text;
            Form администратор = new Администратор();
            администратор.Show();
         }
         else if ("Менеджер по запасам" == comboBox1.Text && textBox2.Text == "9012")
         {
```

```
            GlobalData.Password = textBox2.Text;
            Form менеджер_по_запасам = new МенеджерПоЗапасам();
            менеджер_по_запасам.Show();
        }
        else if ("Ветеринар" == comboBox1.Text && textBox2.Text == "3456")
        {
            GlobalData.Password = textBox2.Text;
            Form ветеринар = new Ветеринар();
            ветеринар.Show();
        }
        else
        {
            // Если ни одно из условий не выполнено, выводим сообщение об ошибке
            MessageBox.Show("Неверный пароль!", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }

  }
}
```

2. Код программы «Генеральный директор»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Итоговая_проектная_работа
{
  public partial class ГенеральныйДиректор : Form
  {
    public ГенеральныйДиректор()
    {
        InitializeComponent();
    }

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

    private void button1_Click(object sender, EventArgs e)
    {
        Form услуги = new Услуги();
        услуги.Show();
    }
```

```
        private void button3_Click(object sender, EventArgs e)
        {
            Form складскаяоперация = new СкладскиеОперации();
            складскаяоперация.Show();
        }
    }
}
```

3. Код программы «Администратор»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Итоговая_проектная_работа
{
    public partial class Администратор : Form
    {
        public Администратор()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form client = new Client();
            client.Show();
        }

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

        private void button3_Click(object sender, EventArgs e)
        {
            Form график = new График();
            график.Show();
        }
    }
}
```

4. Код программы «Менеджер по запасам»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Итоговая_проектная_работа
{
    public partial class МенеджерПоЗапасам : Form
    {
        public МенеджерПоЗапасам()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form складскаяоперация = new СкладскиеОперации();
            складскаяоперация.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form медикамент = new Медикамент();
            медикамент.Show();
        }
    }
}
```

5. Код программы «Ветеринар»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Итоговая_проектная_работа
{
    public partial class Ветеринар : Form
    {
        public Ветеринар()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            Form питомец = new Питомцы();
            питомец.Show();
```

```
        }

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

        private void button3_Click(object sender, EventArgs e)
        {
            Form услугиприема = new УслугаПриема();
            услугиприема.Show();
        }
    }
}
```

6. Код программы «Клиенты»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;


namespace Итоговая_проектная_работа
{
    public partial class Client : Form
    {
        public Client()
        {
            InitializeComponent();
        }

        private void клиентBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.клиентBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.ветеринарная_клиникаDataSet);

        }

        private void Клиенты_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Клиент". При необходимости она может быть перемещена или удалена.
            this.клиентTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Клиент);
```

```
        }

        private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
        {
            button1.Enabled = true;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.DataGridViewColumn Col = 
default(System.Windows.Forms.DataGridViewColumn);
            switch (listBox1.SelectedIndex)
            {
                case 0:
                    Col = dataGridViewTextBoxColumn2;
                    break;
                case 1:
                    Col = dataGridViewTextBoxColumn3;
                    break;
                case 2:
                    Col = dataGridViewTextBoxColumn4;
                    break;
                case 3:
                    Col = dataGridViewTextBoxColumn5;
                    break;
                //case 4:
                  //  Col = dataGridViewTextBoxColumn6;
            }
            if (radioButton1.Checked)
                клиентDataGridView.Sort(Col, 
System.ComponentModel.ListSortDirection.Ascending);
            else
                клиентDataGridView.Sort(Col, 
System.ComponentModel.ListSortDirection.Descending);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string searchText = textBox1.Text.Replace("'", "''"); // Экранируем одинарные кавычки
            клиентBindingSource.Filter = $"ФИО LIKE '%{searchText}%'"; // Используем 
интерполяцию строк и правильное расположение %
            //клиентBindingSource.Filter = "ФИО Like N'" + textBox1.Text + "%'";
        }

        private void button3_Click(object sender, EventArgs e)
        {
            клиентBindingSource.Filter = "";
        }

        private void button4_Click(object sender, EventArgs e)
        {
            try
```

```
        {
            // Проверяем, что все обязательные поля заполнены
            if (string.IsNullOrWhiteSpace(textBox2.Text))
                throw new ArgumentException("ФИО клиента не может быть пустым.");
            if (string.IsNullOrWhiteSpace(textBox3.Text))
                throw new ArgumentException("Контактный телефон не может быть пустым.");
            if (string.IsNullOrWhiteSpace(textBox4.Text))
                throw new ArgumentException("Email не может быть пустым.");
            if (string.IsNullOrWhiteSpace(textBox5.Text))
                throw new ArgumentException("Адрес не может быть пустым.");

            // Присваиваем значения параметрам
            sqlCommand.Parameters["@ФИО"].Value = textBox2.Text.Trim();
            sqlCommand.Parameters["@Контактный_телефон"].Value = textBox3.Text.Trim();
            sqlCommand.Parameters["@Email"].Value = textBox4.Text.Trim();
            sqlCommand.Parameters["@Адрес"].Value = textBox5.Text.Trim();

            // Открываем соединение с БД
            sqlConnection.Open();

            // Выполняем хранимую процедуру
            sqlCommand.ExecuteNonQuery();
            this.клиентTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Клиент);
            // Уведомляем пользователя об успешном добавлении
            MessageBox.Show("Клиент успешно добавлен.");
        }
        catch (ArgumentException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        catch (SqlException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            if (sqlConnection.State == ConnectionState.Open)
                sqlConnection.Close();
        }
    }



    private void клиентDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        // Проверяем, чтобы не был кликнут заголовок
        if (e.RowIndex >= 0)
        {
            // Получаем данные выбранной строки
```

```
            var row = клиентDataGridView.Rows[e.RowIndex];

            // Предположим, что у нас есть столбцы с именами "Id_Услуги", "Название"
            textBox2.Text = row.Cells[1].Value.ToString();
            textBox3.Text = row.Cells[2].Value.ToString();
            textBox4.Text = row.Cells[3].Value.ToString();
            textBox5.Text = row.Cells[4].Value.ToString();
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        try
        {
            if (string.IsNullOrWhiteSpace(textBox2.Text))
                throw new ArgumentException("ФИО клиента не может быть пустым.");
            if (string.IsNullOrWhiteSpace(textBox3.Text))
                throw new ArgumentException("Контактный телефон не может быть пустым.");
            if (string.IsNullOrWhiteSpace(textBox4.Text))
                throw new ArgumentException("Email не может быть пустым.");
            if (string.IsNullOrWhiteSpace(textBox5.Text))
                throw new ArgumentException("Адрес не может быть пустым.");

            sqlCommandUpdate.Parameters["@ФИО"].Value = textBox2.Text.Trim();
            sqlCommandUpdate.Parameters["@Контактный_телефон"].Value =
textBox3.Text.Trim();
            sqlCommandUpdate.Parameters["@Email"].Value = textBox4.Text.Trim();
            sqlCommandUpdate.Parameters["@Адрес"].Value = textBox5.Text.Trim();

            sqlConnection.Open();
            sqlCommandUpdate.ExecuteNonQuery();
            this.клиентTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Клиент);
            MessageBox.Show("Данные клиента успешно обновлены.");
        }
        catch (ArgumentException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        catch (SqlException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            if (sqlConnection.State == ConnectionState.Open)
                sqlConnection.Close();
        }
    }

}
```

}

7. Код программы «Сотрудники»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;


namespace Итоговая_проектная_работа
{
    public partial class Sotrudnik : Form
    {
        public Sotrudnik()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.DataGridViewColumn Col =
default(System.Windows.Forms.DataGridViewColumn);
            switch (listBox1.SelectedIndex)
            {
                case 0:
                    Col = dataGridViewTextBoxColumn2;
                    break;
                case 1:
                    Col = dataGridViewTextBoxColumn3;
                    break;
                case 2:
                    Col = dataGridViewTextBoxColumn4;
                    break;
                case 3:
                    Col = dataGridViewTextBoxColumn5;
                    break;
                case 4:
                    Col = dataGridViewTextBoxColumn6;
                    break;
            }
            if (radioButton1.Checked)
                сотрудникDataGridView.Sort(Col,
System.ComponentModel.ListSortDirection.Ascending);
            else
                сотрудникDataGridView.Sort(Col,
System.ComponentModel.ListSortDirection.Descending);
        }
```

```
private void сотрудникBindingNavigatorSaveItem_Click(object sender, EventArgs e)
{
    this.Validate();
    this.сотрудникBindingSource.EndEdit();
    this.tableAdapterManager.UpdateAll(this.ветеринарная_клиникаDataSet);

}

private void Sotrudnik_Load(object sender, EventArgs e)
{
    // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.График". При необходимости она может быть перемещена или удалена.
    this.графикTableAdapter.Fill(this.ветеринарная_клиникаDataSet.График);
    // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Сотрудник". При необходимости она может быть перемещена или удалена.
    this.сотрудникTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Сотрудник);
    id_ГрафикаComboBox.DropDownStyle = ComboBoxStyle.DropDownList;
}

private void button3_Click(object sender, EventArgs e)
{
    сотрудникBindingSource.Filter = "";
}

private void button2_Click(object sender, EventArgs e)
{
    string searchText = textBox1.Text.Replace("'", "''"); // Экранируем одинарные кавычки
    сотрудникBindingSource.Filter = $"ФИО LIKE '%{searchText}%'"; // Используем интерполяцию строк и правильное расположение %
    //клиентBindingSource.Filter = "ФИО Like N'" + textBox1.Text + "%'";
}

private void button4_Click(object sender, EventArgs e)
{
    try
    {
        if (string.IsNullOrWhiteSpace(id_ГрафикаComboBox.Text) || id_ГрафикаComboBox.Text == "0")
        {
            MessageBox.Show("Поле Id_Графика не должно быть пустым или равным нулю.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        int idГрафика;
        if (!int.TryParse(id_ГрафикаComboBox.Text, out idГрафика))
        {
            MessageBox.Show("Id_Графика должно быть числом.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            return;
        }

        if (idГрафика <= 0)
        {
            MessageBox.Show("Id_Графика должно быть больше нуля.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }
        if (string.IsNullOrWhiteSpace(textBox2.Text))
            throw new ArgumentException("ФИО не может быть пустым.");
        if (string.IsNullOrWhiteSpace(textBox3.Text))
            throw new ArgumentException("Должность не может быть пустой.");
        if (dateTimePicker1.Value >= DateTime.Now.AddYears(-18))
            throw new ArgumentException("Сотрудник должен быть старше 18 лет.");
        if (string.IsNullOrWhiteSpace(textBox5.Text))
            throw new ArgumentException("Паспортные данные не могут быть пустыми.");
        if (string.IsNullOrWhiteSpace(textBox6.Text))
            throw new ArgumentException("Адрес электронной почты не может быть
пустым.");
        if (string.IsNullOrWhiteSpace(textBox7.Text))
            throw new ArgumentException("Телефон не может быть пустым.");
        if (id_ГрафикаComboBox.SelectedValue == null)
            throw new ArgumentException("График должен быть выбран.");

        sqlCommand.Parameters["@ФИО"].Value = textBox2.Text.Trim();
        sqlCommand.Parameters["@Должность"].Value = textBox3.Text.Trim();
        sqlCommand.Parameters["@ДатаРождения"].Value = dateTimePicker1.Value;
        sqlCommand.Parameters["@ПаспортныеДанные"].Value = textBox5.Text.Trim();
        sqlCommand.Parameters["@АдресЭлектронноЙПочты"].Value =
textBox6.Text.Trim();
        sqlCommand.Parameters["@Телефон"].Value = textBox7.Text.Trim();
        sqlCommand.Parameters["@Id_Графика"].Value =
id_ГрафикаComboBox.SelectedValue;

        sqlConnection.Open();
        sqlCommand.ExecuteNonQuery();
        this.сотрудникTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Сотрудник);
        MessageBox.Show("Сотрудник успешно добавлен.");
    }
    catch (ArgumentException ex)
    {
        MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
    catch (SqlException ex)
    {
        MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }
    finally
    {
```

```
            if (sqlConnection.State == ConnectionState.Open)
                sqlConnection.Close();
        }
    }




    private void button5_Click(object sender, EventArgs e)
    {
        try
        {
            if (string.IsNullOrWhiteSpace(textBox2.Text))
                throw new ArgumentException("ФИО не может быть пустым.");
            if (string.IsNullOrWhiteSpace(textBox3.Text))
                throw new ArgumentException("Должность не может быть пустой.");
            if (dateTimePicker1.Value >= DateTime.Now.AddYears(-18))
                throw new ArgumentException("Сотрудник должен быть старше 18 лет.");
            if (string.IsNullOrWhiteSpace(textBox5.Text))
                throw new ArgumentException("Паспортные данные не могут быть пустыми.");
            if (string.IsNullOrWhiteSpace(textBox6.Text))
                throw new ArgumentException("Адрес электронной почты не может быть
пустым.");
            if (string.IsNullOrWhiteSpace(textBox7.Text))
                throw new ArgumentException("Телефон не может быть пустым.");
            if (id_ГрафикаComboBox.SelectedValue == null)
                throw new ArgumentException("График должен быть выбран.");

            // Присвоить значения входным параметрам процедуры с преобразованием типов
            sqlCommandUpdate.Parameters["@ФИО"].Value = textBox2.Text;
            sqlCommandUpdate.Parameters["@Должность"].Value = textBox3.Text;
            sqlCommandUpdate.Parameters["@ДатаРождения"].Value = dateTimePicker1.Value;
// Используем Value для даты
            sqlCommandUpdate.Parameters["@ПаспортныеДанные"].Value = textBox5.Text;
            sqlCommandUpdate.Parameters["@АдресЭлектронноЙПочты"].Value =
textBox6.Text;
            sqlCommandUpdate.Parameters["@Телефон"].Value = textBox7.Text;
            sqlCommandUpdate.Parameters["@Id_Графика"].Value =
id_ГрафикаComboBox.SelectedValue; // Используем SelectedValue

            // Открыть соединение с БД
            sqlConnection.Open();

            // Выполнить sql-выражение (хранимую процедуру)
            sqlCommandUpdate.ExecuteNonQuery();
            this.сотрудникTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Сотрудник);
            // Уведомить пользователя об успешном обновлении
            MessageBox.Show("Данные сотрудника успешно обновлены.");
        }
        catch (ArgumentException ex)
        {
```

```
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        catch (SqlException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }
    }



    private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
    {

    }

    private void сотрудникDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        // Проверяем, чтобы не был кликнут заголовок
        if (e.RowIndex >= 0)
        {
            // Получаем данные выбранной строки
            var row = сотрудникDataGridView.Rows[e.RowIndex];

            // Предположим, что у нас есть столбцы с именами "Id_Услуги", "Название"
            textBox2.Text = row.Cells[1].Value.ToString();
            textBox3.Text = row.Cells[2].Value.ToString();
            dateTimePicker1.Text = row.Cells[3].Value.ToString();
            textBox5.Text = row.Cells[4].Value.ToString();
            textBox6.Text = row.Cells[5].Value.ToString();
            textBox7.Text = row.Cells[6].Value.ToString();
            id_ГрафикаComboBox.Text = row.Cells[7].Value.ToString();
        }
    }
  }
}
```

8. Код программы «Питомцы»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;


namespace Итоговая_проектная_работа
{
    public partial class Питомцы : Form
    {
        public Питомцы()
        {
            InitializeComponent();
        }

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

this.питомцы_с_ФИОTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Питомцы_с_ФИО);
            comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.DataGridViewColumn Col = default(System.Windows.Forms.DataGridViewColumn);
            switch (listBox1.SelectedIndex)
            {
                case 0:
                    Col = dataGridViewTextBoxColumn2;
                    break;
                case 1:
                    Col = dataGridViewTextBoxColumn3;
                    break;
                case 2:
                    Col = dataGridViewTextBoxColumn4;
                    break;
                case 3:
                    Col = dataGridViewTextBoxColumn5;
                    break;
                case 4:
                    Col = dataGridViewTextBoxColumn6;
                    break;
                case 5:
                    Col = dataGridViewTextBoxColumn7;
                    break;
            }
```

```
            if (radioButton1.Checked)
                питомцы_с_ФИОDataGridView.Sort(Col,
System.ComponentModel.ListSortDirection.Ascending);
            else
                питомцы_с_ФИОDataGridView.Sort(Col,
System.ComponentModel.ListSortDirection.Descending);
        }

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

        private void button4_Click(object sender, EventArgs e)
        {
            string searchText = textBox7.Text.Replace("'", "''"); // Экранируем одинарные кавычки
            питомцы_с_ФИОBindingSource.Filter = $"Кличка LIKE '%{searchText}%'"; //
Используем интерполяцию строк и правильное расположение %
        }

        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                // Проверяем, что все необходимые поля заполнены
                if (string.IsNullOrWhiteSpace(textBox1.Text))
                    throw new ArgumentException("ФИО клиента не может быть пустым.");
                if (string.IsNullOrWhiteSpace(textBox2.Text))
                    throw new ArgumentException("Кличка не может быть пустой.");
                if (string.IsNullOrWhiteSpace(textBox3.Text))
                    throw new ArgumentException("Порода не может быть пустой.");
                if (string.IsNullOrWhiteSpace(textBox4.Text))
                    throw new ArgumentException("Окрас не может быть пустым.");
                if (comboBox1.SelectedItem == null)
                    throw new ArgumentException("Пол должен быть выбран.");
                if (numericUpDown1.Value <= 0)
                    throw new ArgumentException("Возраст должен быть больше 0.");

                // Присвоить значения входным параметрам процедуры с преобразованием типов
                sqlCommand.Parameters["@ФИО_Клиента"].Value = textBox1.Text.Trim();
                sqlCommand.Parameters["@Кличка"].Value = textBox2.Text.Trim();
                sqlCommand.Parameters["@Порода"].Value = textBox3.Text.Trim();
                sqlCommand.Parameters["@Окрас"].Value = textBox4.Text.Trim();
                sqlCommand.Parameters["@Пол"].Value = comboBox1.SelectedItem.ToString(); //
Пол из ComboBox
                sqlCommand.Parameters["@Возраст"].Value = (int)numericUpDown1.Value; //
Возраст из NumericUpDown

                // Открыть соединение с БД
                sqlConnection.Open();

                // Выполнить sql-выражение (хранимую процедуру)
```

```
            sqlCommand.ExecuteNonQuery();


this.питомцы_с_ФИОTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Питомцы_с_ФИ
О);

            // Уведомить пользователя об успешном обновлении данных питомца
            MessageBox.Show("Питомец успешно добавлен.");
        }
        catch (ArgumentException ex)
        {
            // Обработка ошибок заполнения полей
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        catch (SqlException ex)
        {
            // Выводим сообщение ошибки, которое возвращает хранимая процедура
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }
    }



    private void питомцы_с_ФИОDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        // Проверяем, чтобы не был кликнут заголовок
        if (e.RowIndex >= 0)
        {
            // Получаем данные выбранной строки
            var row = питомцы_с_ФИОDataGridView.Rows[e.RowIndex];

            // Предположим, что у нас есть столбцы с именами "Id_Услуги", "Название"
            textBox1.Text = row.Cells[1].Value.ToString();
            textBox2.Text = row.Cells[2].Value.ToString();
            textBox3.Text = row.Cells[3].Value.ToString();
            textBox4.Text = row.Cells[4].Value.ToString();
            comboBox1.Text = row.Cells[5].Value.ToString();
            numericUpDown1.Text = row.Cells[6].Value.ToString();
        }
    }
```

```
        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                // Проверяем, что все необходимые поля заполнены
                if (string.IsNullOrWhiteSpace(textBox1.Text))
                    throw new ArgumentException("ФИО клиента не может быть пустым.");
                if (string.IsNullOrWhiteSpace(textBox2.Text))
                    throw new ArgumentException("Кличка не может быть пустой.");
                if (numericUpDown1.Value <= 0)
                    throw new ArgumentException("Возраст должен быть больше 0.");

                // Присвоить значения входным параметрам процедуры с преобразованием типов
                sqlCommandUpdate.Parameters["@ФИО_Клиента"].Value = textBox1.Text.Trim();
                sqlCommandUpdate.Parameters["@Кличка"].Value = textBox2.Text.Trim();
                sqlCommandUpdate.Parameters["@Возраст"].Value  =  (int)numericUpDown1.Value;
// Возраст из NumericUpDown

                // Открыть соединение с БД
                sqlConnection.Open();

                // Выполнить sql-выражение (хранимую процедуру)
                sqlCommandUpdate.ExecuteNonQuery();


this.питомцы_с_ФИОTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Питомцы_с_ФИ
О);

                // Уведомить пользователя об успешном обновлении данных питомца
                MessageBox.Show("Питомец успешно обновлен.");
            }
            catch (ArgumentException ex)
            {
                // Обработка ошибок заполнения полей
                MessageBox.Show($"Ошибка:  {ex.Message}",  "Ошибка",  MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            catch (SqlException ex)
            {
                // Выводим сообщение ошибки, которое возвращает хранимая процедура
                MessageBox.Show($"Ошибка:  {ex.Message}",  "Ошибка",  MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            finally
            {
                // Закрыть соединение с БД
                if (sqlConnection.State == System.Data.ConnectionState.Open)
                {
                    sqlConnection.Close();
                }
            }
        }
```

```
        private BindingSource питомцыBindingSource = new BindingSource();
        // Метод для перезагрузки данных

        private void textBox1_TextChanged(object sender, EventArgs e)
        {

        }
    }
}
```

9. Код программы «Медикамент»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Итоговая_проектная_работа
{
    public partial class Медикамент : Form
    {
        public Медикамент()
        {
            InitializeComponent();
        }

        private void медикаментBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.медикаментBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.ветеринарная_клиникаDataSet);

        }

        private void Медикамент_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Медикамент". При необходимости она может быть перемещена или удалена.
            this.медикаментTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Медикамент);
            comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;

        }

        private void button1_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.DataGridViewColumn Col = default(System.Windows.Forms.DataGridViewColumn);
```

```
            switch (listBox1.SelectedIndex)
            {
                case 0:
                    Col = dataGridViewTextBoxColumn2;
                    break;
                case 1:
                    Col = dataGridViewTextBoxColumn3;
                    break;
                case 2:
                    Col = dataGridViewTextBoxColumn4;
                    break;
                case 3:
                    Col = dataGridViewTextBoxColumn5;
                    break;
                case 4:
                    Col = dataGridViewTextBoxColumn6;
                    break;
            }
            if (radioButton1.Checked)
                медикаментDataGridView.Sort(Col,
System.ComponentModel.ListSortDirection.Ascending);
            else
                медикаментDataGridView.Sort(Col,
System.ComponentModel.ListSortDirection.Descending);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            string searchText = textBox6.Text.Replace("'", "''"); // Экранируем одинарные кавычки
            медикаментBindingSource.Filter = $"Наименование LIKE '%{searchText}%'";
        }

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

        private void button2_Click(object sender, EventArgs e)
        {
            try
            {
                // Проверка на пустоту полей
                if (string.IsNullOrWhiteSpace(textBox1.Text))
                {
                    MessageBox.Show("Поле 'Наименование' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return; // Прерываем выполнение, если поле пустое
                }
                if (string.IsNullOrWhiteSpace(textBox2.Text))
                {
                    MessageBox.Show("Поле 'Стоимость' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
```

```
            return; // Прерываем выполнение, если поле пустое
        }
        if (string.IsNullOrWhiteSpace(textBox3.Text))
        {
            MessageBox.Show("Поле 'СрокГодности' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return; // Прерываем выполнение, если поле пустое
        }
        if (string.IsNullOrWhiteSpace(textBox4.Text))
        {
            MessageBox.Show("Поле 'Количество' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return; // Прерываем выполнение, если поле пустое
        }
        if (string.IsNullOrWhiteSpace(comboBox1.Text))
        {
            MessageBox.Show("Поле 'ЕдиницаИзмерения' не может быть пустым.",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return; // Прерываем выполнение, если поле пустое
        }

        // Проверка на правильность ввода стоимости, срока годности и количества
        if (!decimal.TryParse(textBox2.Text, out decimal стоимость))
        {
            MessageBox.Show("Введите корректную числовую стоимость.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return; // Прерываем выполнение, если значение не числовое
        }
        if (!int.TryParse(textBox3.Text, out int срокГодности))
        {
            MessageBox.Show("Введите корректный срок годности (целое число).",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return; // Прерываем выполнение, если значение не числовое
        }
        if (!int.TryParse(textBox4.Text, out int количество))
        {
            MessageBox.Show("Введите корректное количество (целое число).", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            return; // Прерываем выполнение, если значение не числовое
        }

        // Проверка на корректность данных
        if (стоимость <= 0)
        {
            MessageBox.Show("Стоимость медикамента должна быть больше 0.",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        if (срокГодности <= 0)
        {
```

```
                MessageBox.Show("Срок годности медикамента должен быть больше 0.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (количество < 0)
            {
                MessageBox.Show("Количество медикамента не может быть отрицательным.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Проверка на корректность единицы измерения
            if (comboBox1.SelectedItem == null || !new List<string> { "шт", "гр", "уп", "амп", "флак" }.Contains(comboBox1.SelectedItem.ToString()))
            {
                MessageBox.Show("Некорректная единица измерения. Допустимые значения: шт, гр, уп, амп, флак.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Присвоить значения входным параметрам процедуры с преобразованием типов
            sqlCommand.Parameters["@Наименование"].Value = textBox1.Text;
            sqlCommand.Parameters["@Стоимость"].Value = стоимость;  // Используем decimal для стоимости
            sqlCommand.Parameters["@СрокГодности"].Value = срокГодности;  // Используем int для срока годности
            sqlCommand.Parameters["@Количество"].Value = количество;  // Используем int для количества
            sqlCommand.Parameters["@ЕдиницаИзмерения"].Value = comboBox1.SelectedItem.ToString(); // Используем SelectedItem для ComboBox

            // Открыть соединение с БД
            sqlConnection.Open();

            // Выполнить sql-выражение (хранимую процедуру)
            sqlCommand.ExecuteNonQuery();
            this.медикаментTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Медикамент);
            // Уведомить пользователя об успешном добавлении медикамента
            MessageBox.Show("Медикамент успешно добавлен.");
        }
        catch (SqlException ex)
        {
            // Обработка ошибок: показать сообщение пользователю
            MessageBox.Show($"Ошибка при добавлении медикамента. {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
```

```
                sqlConnection.Close();
            }
        }
    }




    private void button3_Click(object sender, EventArgs e)
    {
        try
        {
            // Проверка на пустоту полей
            if (string.IsNullOrWhiteSpace(textBox1.Text))
            {
                MessageBox.Show("Поле 'Наименование' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если поле пустое
            }
            if (string.IsNullOrWhiteSpace(textBox2.Text))
            {
                MessageBox.Show("Поле 'Стоимость' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если поле пустое
            }
            if (string.IsNullOrWhiteSpace(textBox3.Text))
            {
                MessageBox.Show("Поле 'СрокГодности' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если поле пустое
            }
            if (string.IsNullOrWhiteSpace(textBox4.Text))
            {
                MessageBox.Show("Поле 'Количество' не может быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если поле пустое
            }
            if (string.IsNullOrWhiteSpace(comboBox1.Text))
            {
                MessageBox.Show("Поле 'ЕдиницаИзмерения' не может быть пустым.",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если поле пустое
            }

            // Проверка на правильность ввода стоимости, срока годности и количества
            if (!decimal.TryParse(textBox2.Text, out decimal стоимость))
            {
                MessageBox.Show("Введите корректную числовую стоимость.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если значение не числовое
            }
            if (!int.TryParse(textBox3.Text, out int срокГодности))
```

```
            {
                MessageBox.Show("Введите корректный срок годности (целое число).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если значение не числовое
            }
            if (!int.TryParse(textBox4.Text, out int количество))
            {
                MessageBox.Show("Введите корректное количество (целое число).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return; // Прерываем выполнение, если значение не числовое
            }

            // Проверка на корректность данных
            if (стоимость <= 0)
            {
                MessageBox.Show("Стоимость медикамента должна быть больше 0.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (срокГодности <= 0)
            {
                MessageBox.Show("Срок годности медикамента должен быть больше 0.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (количество < 0)
            {
                MessageBox.Show("Количество медикамента не может быть отрицательным.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Проверка на корректность единицы измерения
            if (comboBox1.SelectedItem == null || !new List<string> { "шт", "гр", "уп", "амп", "флак" }.Contains(comboBox1.SelectedItem.ToString()))
            {
                MessageBox.Show("Некорректная единица измерения. Допустимые значения: шт, гр, уп, амп, флак.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Присвоить значения входным параметрам процедуры с преобразованием типов
            sqlCommandUpdate.Parameters["@Наименование"].Value = textBox1.Text;
            sqlCommandUpdate.Parameters["@Стоимость"].Value = стоимость;  // Используем decimal для стоимости
            sqlCommandUpdate.Parameters["@СрокГодности"].Value = срокГодности;  // Используем int для срока годности
            sqlCommandUpdate.Parameters["@Количество"].Value = количество;  // Используем int для количества
```

```
            sqlCommandUpdate.Parameters["@ЕдиницаИзмерения"].Value =
comboBox1.SelectedItem.ToString(); // Используем SelectedItem для ComboBox

            // Открыть соединение с БД
            sqlConnection.Open();

            // Выполнить sql-выражение (хранимую процедуру) для обновления данных
            sqlCommandUpdate.ExecuteNonQuery();
            this.медикаментTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Медикамент);
            // Уведомить пользователя об успешном обновлении
            MessageBox.Show("Данные медикамента успешно обновлены.");
        }
        catch (SqlException ex)
        {
            // Обработка ошибок: показать сообщение пользователю
            MessageBox.Show($"Ошибка при обновлении медикамента. {ex.Message}",
"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }
    }




    private void медикаментDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        // Проверяем, чтобы не был кликнут заголовок
        if (e.RowIndex >= 0)
        {
            // Получаем данные выбранной строки
            var row = медикаментDataGridView.Rows[e.RowIndex];

            // Предположим, что у нас есть столбцы с именами "Id_Услуги", "Название"
            textBox1.Text = row.Cells[1].Value.ToString();
            textBox2.Text = row.Cells[2].Value.ToString();
            textBox3.Text = row.Cells[3].Value.ToString();
            textBox4.Text = row.Cells[4].Value.ToString();
            comboBox1.Text = row.Cells[5].Value.ToString();
        }
    }

    private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
    {
```

```
        }
    }
}
```

10. Код программы «Прием»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;


namespace Итоговая_проектная_работа
{
    public partial class Прием : Form
    {
        public Прием()
        {
            InitializeComponent();
        }

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

this.питомцы_с_ФИОTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Питомцы_с_ФИО);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Питомец". При необходимости она может быть перемещена или удалена.
            this.питомецTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Питомец);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Клиент". При необходимости она может быть перемещена или удалена.
            this.клиентTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Клиент);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Услуга". При необходимости она может быть перемещена или удалена.
            this.услугаTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Услуга);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.V_Ветеринары_График". При необходимости она может быть перемещена или удалена.

this.v_Ветеринары_ГрафикTableAdapter.Fill(this.ветеринарная_клиникаDataSet.V_Ветеринары_График);
            ветеринарная_клиникаDataSet.EnforceConstraints = false;
```

```
        // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.ПолнаяИнформацияОПриемах". При необходимости она может быть перемещена или удалена.

this.полнаяИнформацияОПриемахTableAdapter.Fill(this.ветеринарная_клиникаDataSet.ПолнаяИнформацияОПриемах);
        фИОComboBox.DropDownStyle = ComboBoxStyle.DropDownList;
        comboBox2.DropDownStyle = ComboBoxStyle.DropDownList;
        comboBox3.DropDownStyle = ComboBoxStyle.DropDownList;
    }

    private void button1_Click(object sender, EventArgs e)
    {
        System.Windows.Forms.DataGridViewColumn Col = default(System.Windows.Forms.DataGridViewColumn);
        switch (listBox1.SelectedIndex)
        {
            case 0:
                Col = dataGridViewTextBoxColumn2;
                break;
            case 1:
                Col = dataGridViewTextBoxColumn3;
                break;
            case 2:
                Col = dataGridViewTextBoxColumn4;
                break;
            case 3:
                Col = dataGridViewTextBoxColumn5;
                break;
            case 4:
                Col = dataGridViewTextBoxColumn6;
                break;
        }
        if (radioButton1.Checked)
            полнаяИнформацияОПриемахDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);
        else
            полнаяИнформацияОПриемахDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);
    }

    private void button4_Click(object sender, EventArgs e)
    {
        полнаяИнформацияОПриемахBindingSource.Filter = "";
    }

    private void button3_Click(object sender, EventArgs e)
    {
        string searchText = textBox3.Text.Replace("'", "''"); // Экранируем одинарные кавычки
        полнаяИнформацияОПриемахBindingSource.Filter = $"Кличка_Питомца LIKE '%{searchText}%'";
    }
```

```
private void button2_Click(object sender, EventArgs e)
{
    try
    {
        // Проверка на пустые поля ФИО сотрудника и кличку питомца
        if (string.IsNullOrWhiteSpace(фИОComboBox.Text) || string.IsNullOrWhiteSpace(comboBox3.Text))
        {
            MessageBox.Show("Пожалуйста, введите корректные ФИО сотрудника и кличку питомца.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        // Преобразование и проверка даты
        if (!DateTime.TryParse(dateTimePicker1.Text, out DateTime датаВремя))
        {
            MessageBox.Show("Некорректная дата/время. Пожалуйста, выберите правильное значение.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        // Проверка на пустое поле комментария
        if (string.IsNullOrWhiteSpace(textBox4.Text))
        {
            MessageBox.Show("Пожалуйста, введите комментарий.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        // Присвоить значения входным параметрам процедуры с преобразованием типов
        sqlCommand.Parameters["@ФИО_Сотрудника"].Value = фИОComboBox.Text;
        sqlCommand.Parameters["@ФИО_Клиента"].Value = comboBox2.Text;
        sqlCommand.Parameters["@Кличка_Питомца"].Value = comboBox3.Text;
        sqlCommand.Parameters["@Дата_время"].Value = датаВремя; // Используем DateTime для даты и времени
        sqlCommand.Parameters["@Комментарий"].Value = textBox4.Text;
        sqlCommand.Parameters["@Оказанная_Услуга"].Value = comboBox1.Text;
        if (string.IsNullOrWhiteSpace(comboBox1.Text))
        {
            sqlCommand.Parameters["@Оказанная_Услуга"].Value = DBNull.Value;
        }
        else
        {
            sqlCommand.Parameters["@Оказанная_Услуга"].Value = comboBox1.Text;
        }

        // Открыть соединение с БД
        sqlConnection.Open();

        // Выполнить sql-выражение (хранимую процедуру) для добавления нового приема
```

```
            sqlCommand.ExecuteNonQuery();

this.полнаяИнформацияОПриемахTableAdapter.Fill(this.ветеринарная_клиникаDataSet.ПолнаяИнформацияОПриемах);
            // Уведомить пользователя об успешном добавлении
            MessageBox.Show("Прием успешно добавлен.");
        }
        catch (SqlException ex)
        {
            //Обработка ошибок SQL(например, если сотрудник или питомец не существуют, или если данные не удовлетворяют ограничениям NOT NULL)
            string errorMessage = ex.Message;

            //Проверка на отсутствие сотрудника или питомца
            if (errorMessage.Contains("FK_Сотрудник_Прием") || errorMessage.Contains("FK_Питомец_Прием"))
            {
                MessageBox.Show("Ошибка: Сотрудник с таким ФИО или питомец с такой кличкой не найдены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            //Проверка на нарушение ограничения NOT NULL
            else if (errorMessage.Contains("NULL"))
            {
                MessageBox.Show("Ошибка: Все поля (ФИО сотрудника, кличка питомца, дата и комментарий) должны быть заполнены.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            //Общая обработка других ошибок
            else
            {
                MessageBox.Show($"Ошибка при добавлении приема: {errorMessage}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }

        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }
    }



    private void dateTimePicker1_ValueChanged(object sender, EventArgs e)
    {
        // Настраиваем DateTimePicker для отображения даты и времени
        dateTimePicker1.Format = DateTimePickerFormat.Custom;
```

```
        dateTimePicker1.CustomFormat = "dd.MM.yyyy HH:mm"; // Пользовательский формат
    }
    private void представление_ПриемDataGridView_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        // Проверяем, чтобы не был кликнут заголовок
        if (e.RowIndex >= 0)
        {
            // Получаем данные выбранной строки
            var row = полнаяИнформацияОПриемахDataGridView.Rows[e.RowIndex];

            фИОComboBox.Text = row.Cells[1].Value.ToString();
            //textBox1.Text = row.Cells[2].Value.ToString();
            //textBox2.Text = row.Cells[3].Value.ToString();
            dateTimePicker1.Text = row.Cells[4].Value.ToString();
            textBox4.Text = row.Cells[5].Value.ToString();
            comboBox1.Text = row.Cells[6].Value.ToString();
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        try
        {
            // Проверка на пустые поля ФИО сотрудника и кличку питомца
            if (string.IsNullOrWhiteSpace(фИОComboBox.Text) || string.IsNullOrWhiteSpace(comboBox3.Text))
            {
                MessageBox.Show("Пожалуйста, введите корректные ФИО сотрудника и кличку питомца.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Преобразование и проверка даты
            if (!DateTime.TryParse(dateTimePicker1.Text, out DateTime датаВремя))
            {
                MessageBox.Show("Некорректная дата/время. Пожалуйста, выберите правильное значение.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Проверка на пустое поле комментария
            if (string.IsNullOrWhiteSpace(textBox4.Text))
            {
                MessageBox.Show("Пожалуйста, введите комментарий.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Присвоить значения входным параметрам процедуры с преобразованием типов
```

```
            sqlCommandUpdate.Parameters["@ФИО_Сотрудника"].Value =
фИОComboBox.Text;
            sqlCommandUpdate.Parameters["@ФИО_Клиента"].Value = comboBox2.Text;
            sqlCommandUpdate.Parameters["@Кличка_Питомца"].Value = comboBox3.Text;
            sqlCommandUpdate.Parameters["@Дата_время"].Value = датаВремя; //
Используем DateTime для даты и времени
            sqlCommandUpdate.Parameters["@Комментарий"].Value = textBox4.Text;

            // Открыть соединение с БД
            sqlConnection.Open();

            // Выполнить sql-выражение (хранимую процедуру) для добавления нового
приема
            sqlCommandUpdate.ExecuteNonQuery();

this.полнаяИнформацияОПриемахTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Пол
наяИнформацияОПриемах);
            // Уведомить пользователя об успешном добавлении
            MessageBox.Show("Прием успешно обновлен.");
        }
        catch (SqlException ex)
        {
            // Обработка ошибок SQL (например, если сотрудник или питомец не
существуют, или если данные не удовлетворяют ограничениям NOT NULL)
            MessageBox.Show($"Ошибка при обновлении приема: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }
    }

    private void button6_Click(object sender, EventArgs e)
    {
        try
        {
            // Проверка на пустые поля ФИО сотрудника и кличку питомца
            if (string.IsNullOrWhiteSpace(фИОComboBox.Text) ||
string.IsNullOrWhiteSpace(comboBox3.Text))
            {
                MessageBox.Show("Пожалуйста, введите корректные ФИО сотрудника и
кличку питомца.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Преобразование и проверка даты
            if (!DateTime.TryParse(dateTimePicker1.Text, out DateTime датаВремя))
```

```
            {
                MessageBox.Show("Некорректная дата/время. Пожалуйста, выберите правильное значение.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Проверка на пустое поле комментария
            //if (string.IsNullOrWhiteSpace(textBox4.Text))
            //{
            //    MessageBox.Show("Пожалуйста, введите комментарий.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            //    return;
            //}

            // Присвоить значения входным параметрам процедуры с преобразованием типов
            sqlCommandDelete.Parameters["@ФИО_Сотрудника"].Value = фИОComboBox.Text;
            sqlCommandDelete.Parameters["@ФИО_Клиента"].Value = comboBox2.Text;
            sqlCommandDelete.Parameters["@Кличка_Питомца"].Value = comboBox3.Text;
            sqlCommandDelete.Parameters["@Дата_время"].Value = датаВремя;  // Используем DateTime для даты и времени
            //sqlCommandDelete.Parameters["@Комментарий"].Value = textBox4.Text;

            // Открыть соединение с БД
            sqlConnection.Open();

            // Выполнить sql-выражение (хранимую процедуру) для добавления нового приема
            sqlCommandDelete.ExecuteNonQuery();

this.полнаяИнформацияОПриемахTableAdapter.Fill(this.ветеринарная_клиникаDataSet.ПолнаяИнформацияОПриемах);
            // Уведомить пользователя об успешном добавлении
            MessageBox.Show("Прием успешно удален.");
        }
        catch (SqlException ex)
        {
            // Обработка ошибок SQL (например, если сотрудник или питомец не существуют, или если данные не удовлетворяют ограничениям NOT NULL)
            MessageBox.Show($"Ошибка при удалении приема: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
        }
    }
```

```
        private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs
e)
        {

        }

        private void фИОComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            string searchText = фИОComboBox.Text.Replace("'", "''"); // Экранируем одинарные
кавычки
            v_Ветеринары_ГрафикBindingSource.Filter = $"ФИО LIKE '%{searchText}%'";
        }

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

this.питомцы_с_ФИОTableAdapter.FillBy(this.ветеринарная_клиникаDataSet.Питомцы_с_Ф
ИО);
            }
            catch (System.Exception ex)
            {
                System.Windows.Forms.MessageBox.Show(ex.Message);
            }

        }

        private void comboBox3_SelectedIndexChanged(object sender, EventArgs e)
        {

        }

        private void comboBox2_SelectedIndexChanged(object sender, EventArgs e)
        {
                // Применение фильтрации по выбранному значению
                питомцысФИОBindingSource.Filter = $"ФИО_Клиента = '{comboBox2.Text}'";
        }

        private void button7_Click(object sender, EventArgs e)
        {
            try
            {
                // Проверка на пустые поля ФИО сотрудника и кличку питомца
                if (string.IsNullOrWhiteSpace(фИОComboBox.Text) ||
string.IsNullOrWhiteSpace(comboBox3.Text))
                {
                    MessageBox.Show("Пожалуйста, введите корректные ФИО сотрудника и
кличку питомца.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
```

```
            // Преобразование и проверка даты
            if (!DateTime.TryParse(dateTimePicker1.Text, out DateTime датаВремя))
            {
                MessageBox.Show("Некорректная дата/время. Пожалуйста, выберите правильное значение.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            // Проверка на пустое поле комментария
            //if (string.IsNullOrWhiteSpace(textBox4.Text))
            //{
            //    MessageBox.Show("Пожалуйста, введите комментарий.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            //    return;
            //}

            // Присвоить значения входным параметрам процедуры с преобразованием типов
            sqlCommandDelete2.Parameters["@ФИО_Сотрудника"].Value = фИОComboBox.Text;
            sqlCommandDelete2.Parameters["@ФИО_Клиента"].Value = comboBox2.Text;
            sqlCommandDelete2.Parameters["@Кличка_Питомца"].Value = comboBox3.Text;
            sqlCommandDelete2.Parameters["@Дата_время"].Value = датаВремя; // Используем DateTime для даты и времени
            //sqlCommandDelete.Parameters["@Комментарий"].Value = textBox4.Text;
            sqlCommandDelete2.Parameters["@Оказанная_Услуга"].Value = comboBox1.Text;

            // Открыть соединение с БД
            sqlConnection.Open();

            // Выполнить sql-выражение (хранимую процедуру) для добавления нового приема
            sqlCommandDelete2.ExecuteNonQuery();

this.полнаяИнформацияОПриемахTableAdapter.Fill(this.ветеринарная_клиникаDataSet.ПолнаяИнформацияОПриемах);
            // Уведомить пользователя об успешном добавлении
            MessageBox.Show("Услуга приема успешно удалена.");
        }
        catch (SqlException ex)
        {
            // Обработка ошибок SQL (например, если сотрудник или питомец не существуют, или если данные не удовлетворяют ограничениям NOT NULL)
            MessageBox.Show($"Ошибка при удалении услуги приема: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
        finally
        {
            // Закрыть соединение с БД
            if (sqlConnection.State == System.Data.ConnectionState.Open)
            {
                sqlConnection.Close();
            }
```

```
            }
        }
    }
}
```

11. Код программы «СкладскиеОперации»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Итоговая_проектная_работа
{
    public partial class СкладскиеОперации : Form
    {
        public СкладскиеОперации()
        {
            InitializeComponent();
        }

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

this.сотрудникиСкладаTableAdapter.Fill(this.ветеринарная_клиникаDataSet.СотрудникиСклада);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.СотрудникиСкладаСГрафиком". При необходимости она может быть перемещена или удалена.

this.сотрудникиСкладаСГрафикомTableAdapter.Fill(this.ветеринарная_клиникаDataSet.СотрудникиСкладаСГрафиком);
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Представление_СкладскаяОперация". При необходимости она может быть перемещена или удалена.

this.представление_СкладскаяОперацияTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Представление_СкладскаяОперация);

            numericUpDown1.TextChanged += numericUpDown1_TextChanged;

            comboBox1.DropDownStyle = ComboBoxStyle.DropDownList;
            фИОComboBox.DropDownStyle = ComboBoxStyle.DropDownList;
        }
```

```
        private void button1_Click(object sender, EventArgs e)
        {
            System.Windows.Forms.DataGridViewColumn Col = default(System.Windows.Forms.DataGridViewColumn);
            switch (listBox1.SelectedIndex)
            {
                case 0:
                    Col = dataGridViewTextBoxColumn2;
                    break;
                case 1:
                    Col = dataGridViewTextBoxColumn3;
                    break;
                case 2:
                    Col = dataGridViewTextBoxColumn4;
                    break;
                case 3:
                    Col = dataGridViewTextBoxColumn5;
                    break;
                case 4:
                    Col = dataGridViewTextBoxColumn6;
                    break;
                case 5:
                    Col = dataGridViewTextBoxColumn7;
                    break;
                case 6:
                    Col = dataGridViewTextBoxColumn8;
                    break;
            }
            if (radioButton1.Checked)
                представление_СкладскаяОперацияDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Ascending);
            else
                представление_СкладскаяОперацияDataGridView.Sort(Col, System.ComponentModel.ListSortDirection.Descending);
        }

        private void button4_Click(object sender, EventArgs e)
        {
            представление_СкладскаяОперацияBindingSource.Filter = "";
        }

        private void button3_Click(object sender, EventArgs e)
        {
            string searchText = textBox3.Text.Replace("'", "''"); // Экранируем одинарные кавычки
            представление_СкладскаяОперацияBindingSource.Filter = $"Наименование_Медикамента LIKE '%{searchText}%'";
        }

        private bool numericUpDown1Cleared = false; // Флаг для отслеживания пустого состояния

        private void numericUpDown1_TextChanged(object sender, EventArgs e)
```

```
        {
            if (string.IsNullOrWhiteSpace(numericUpDown1.Text))
            {
                numericUpDown1Cleared = true; // Устанавливаем флаг, если поле "очищено"
            }
            else
            {
                numericUpDown1Cleared = false; // Сбрасываем флаг, если есть текст
            }
        }


        private void button2_Click(object sender, EventArgs e)
        {
            if (GlobalData.Password != "1234")
            {
                try
                {
                    // Проверка на наличие обязательных данных
                    if (string.IsNullOrWhiteSpace(фИОComboBox.Text) || string.IsNullOrWhiteSpace(textBox2.Text) ||
                        string.IsNullOrWhiteSpace(comboBox1.Text))
                    {
                        MessageBox.Show("Пожалуйста, заполните все обязательные поля.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return;
                    }
                    if (numericUpDown1.Value == 0 || numericUpDown1Cleared)
                    {
                        sqlCommand.Parameters["@Id_Приема"].Value = DBNull.Value; // Передаем NULL в параметр
                    }
                    else
                    {
                        sqlCommand.Parameters["@Id_Приема"].Value = numericUpDown1.Value; // Передаем числовое значение
                    }
                    // Проверка на правильность введенного количества
                    if (!int.TryParse(numericUpDown2.Text, out int количествоЕдиниц) || количествоЕдиниц < 0)
                    {
                        MessageBox.Show("Пожалуйста, введите корректное количество единиц.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                        return;
                    }

                    // Присвоить значения входным параметрам процедуры с преобразованием типов
                    sqlCommand.Parameters["@ФИО_Сотрудника"].Value = фИОComboBox.Text;
                    sqlCommand.Parameters["@Наименование_Медикамента"].Value = textBox2.Text;
                    sqlCommand.Parameters["@Операция"].Value = comboBox1.Text;
```

```
                sqlCommand.Parameters["@КоличествоЕдиниц"].Value = количествоЕдиниц;

                // Открыть соединение с БД
                sqlConnection.Open();

                // Выполнить sql-выражение (хранимую процедуру) для добавления новой складской операции
                sqlCommand.ExecuteNonQuery();

this.представление_СкладскаяОперацияTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Представление_СкладскаяОперация);
                // Уведомить пользователя об успешном добавлении
                MessageBox.Show("Складская операция успешно добавлена.");
            }
            catch (SqlException ex)
            {
                // Обработка ошибок SQL: вывод ошибок, если они есть
                MessageBox.Show($"Ошибка при добавлении складской операции. {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                // Закрыть соединение с БД
                if (sqlConnection.State == System.Data.ConnectionState.Open)
                {
                    sqlConnection.Close();
                }
            }
        }
        else MessageBox.Show("В доступе отказано.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }



    private void представление_СкладскаяОперацияDataGridView_CellContentClick(object sender, DataGridViewCellEventArgs e)
    {
        // Проверяем, чтобы не был кликнут заголовок
        if (e.RowIndex >= 0)
        {
            // Получаем данные выбранной строки
            var row = представление_СкладскаяОперацияDataGridView.Rows[e.RowIndex];

            // Предположим, что у нас есть столбцы с именами "Id_Услуги", "Название"
            фИОComboBox.Text = row.Cells[1].Value.ToString();
            textBox2.Text = row.Cells[2].Value.ToString();
            numericUpDown1.Text = row.Cells[3].Value.ToString();
            comboBox1.Text = row.Cells[5].Value.ToString();
            numericUpDown2.Text = row.Cells[6].Value.ToString();
        }
    }
```

```
        private void фИОComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            // Применение фильтрации по выбранному значению
            сотрудникиСкладаСГрафикомBindingSource.Filter = $"ФИО = '{фИОComboBox.Text}'";
        }
    }
}
```

12. Код программы «Услуги»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Итоговая_проектная_работа
{
    public partial class Услуги : Form
    {
        public Услуги()
        {
            InitializeComponent();
        }

        private void услугаBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.услугаBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.ветеринарная_клиникаDataSet);

        }

        private void Услуги_Load(object sender, EventArgs e)
        {
            // TODO: данная строка кода позволяет загрузить данные в таблицу "ветеринарная_клиникаDataSet.Услуга". При необходимости она может быть перемещена или удалена.
            this.услугаTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Услуга);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            string searchText = textBox1.Text.Replace("'", "''"); // Экранируем одинарные кавычки
            услугаBindingSource.Filter = $"Название LIKE '%{searchText}%'";
        }
```

```
        private void button2_Click(object sender, EventArgs e)
        {
            услугаBindingSource.Filter = "";
        }

        private void button3_Click(object sender, EventArgs e)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(textBox2.Text))
                {
                    MessageBox.Show("Поле 'Название' не должно быть пустым.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                sqlCommand.Parameters["@Название"].Value = textBox2.Text;

                // Открыть соединение с БД
                sqlConnection.Open();

                // Выполнить sql-выражение (хранимую процедуру) и вернуть количество
измененных записей
                sqlCommand.ExecuteNonQuery();
                this.услугаTableAdapter.Fill(this.ветеринарная_клиникаDataSet.Услуга);
                // Уведомить пользователя об успешном добавлении
                MessageBox.Show("Услуга успешно добавлена.");
            }
            catch (SqlException ex)
            {
                // Вывод ошибок из хранимой процедуры
                MessageBox.Show($"Ошибка при добавлении услуги: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                // Закрыть соединение с БД
                if (sqlConnection.State == System.Data.ConnectionState.Open)
                {
                    sqlConnection.Close();
                }
            }
        }
    }
}
```

13. Код программы «График»:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Data.SqlClient;

namespace Итоговая_проектная_работа
{
    public partial class График : Form
    {
        public График()
        {
            InitializeComponent();
        }

        private void графикBindingNavigatorSaveItem_Click(object sender, EventArgs e)
        {
            this.Validate();
            this.графикBindingSource.EndEdit();
            this.tableAdapterManager.UpdateAll(this.ветеринарная_клиникаDataSet);

        }

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


        private void button1_Click(object sender, EventArgs e)
        {
            int.TryParse(textBox1.Text, out int searchValue); // Проверяем, что введённое значение числовое
            графикBindingSource.Filter = $"Id_Графика = {searchValue}";

        }

        private void button2_Click(object sender, EventArgs e)
        {
            графикBindingSource.Filter = "";
        }

        private void button3_Click(object sender, EventArgs e)
        {
            if (GlobalData.Password != "1234")
            {
                try
                {
                    // Задание параметров с использованием метода GetDbValue
```

```
                sqlCommandAdd.Parameters["@Понедельник"].Value =
GetDbValue(textBox2.Text);
                sqlCommandAdd.Parameters["@Вторник"].Value = GetDbValue(textBox3.Text);
                sqlCommandAdd.Parameters["@Среда"].Value = GetDbValue(textBox4.Text);
                sqlCommandAdd.Parameters["@Четверг"].Value = GetDbValue(textBox5.Text);
                sqlCommandAdd.Parameters["@Пятница"].Value = GetDbValue(textBox6.Text);
                sqlCommandAdd.Parameters["@Суббота"].Value = GetDbValue(textBox7.Text);
                sqlCommandAdd.Parameters["@Воскресенье"].Value =
GetDbValue(textBox8.Text);

                // Выполнение команды
                sqlConnection.Open();
                sqlCommandAdd.ExecuteNonQuery();
                this.графикTableAdapter.Fill(this.ветеринарная_клиникаDataSet.График);
                MessageBox.Show("График успешно добавлен.");
            }
            catch (SqlException ex)
            {
                MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlConnection.State == ConnectionState.Open)
                    sqlConnection.Close();
            }
        }
        else MessageBox.Show("В доступе отказано.", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }

    private object GetDbValue(string input)
    {
        if (string.IsNullOrWhiteSpace(input))
        {
            return DBNull.Value; // Пустая строка или null заменяется на DBNull
        }
        else
        {
            return input.Trim(); // Удаляем лишние пробелы
        }
    }
    private void графикDataGridView_CellContentClick(object sender,
DataGridViewCellEventArgs e)
    {
        // Проверяем, чтобы не был кликнут заголовок
        if (e.RowIndex >= 0)
        {
            // Получаем данные выбранной строки
            var row = графикDataGridView.Rows[e.RowIndex];

            // Предположим, что у нас есть столбцы с именами "Id_Услуги", "Название"
```

```
            textBox9.Text = row.Cells[0].Value.ToString();
            textBox2.Text = row.Cells[1].Value.ToString();
            textBox3.Text = row.Cells[2].Value.ToString();
            textBox4.Text = row.Cells[3].Value.ToString();
            textBox5.Text = row.Cells[4].Value.ToString();
            textBox6.Text = row.Cells[5].Value.ToString();
            textBox7.Text = row.Cells[6].Value.ToString();
            textBox8.Text = row.Cells[7].Value.ToString();
        }
    }

    private void button5_Click(object sender, EventArgs e)
    {
        if (GlobalData.Password != "1234")
        {
            try
            {
                if (string.IsNullOrWhiteSpace(textBox9.Text) || textBox9.Text == "0")
                {
                    MessageBox.Show("Поле Id_Графика не должно быть пустым или равным
нулю.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                int idГрафика;
                if (!int.TryParse(textBox9.Text, out idГрафика))
                {
                    MessageBox.Show("Id_Графика должно быть числом.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }

                if (idГрафика <= 0)
                {
                    MessageBox.Show("Id_Графика должно быть больше нуля.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                    return;
                }
                // Задание параметров с использованием метода GetDbValue
                sqlCommandDelete.Parameters["@Id_Графика"].Value = textBox9.Text;
                sqlCommandDelete.Parameters["@Понедельник"].Value =
GetDbValue(textBox2.Text);
                sqlCommandDelete.Parameters["@Вторник"].Value = GetDbValue(textBox3.Text);
                sqlCommandDelete.Parameters["@Среда"].Value = GetDbValue(textBox4.Text);
                sqlCommandDelete.Parameters["@Четверг"].Value = GetDbValue(textBox5.Text);
                sqlCommandDelete.Parameters["@Пятница"].Value = GetDbValue(textBox6.Text);
                sqlCommandDelete.Parameters["@Суббота"].Value = GetDbValue(textBox7.Text);
                sqlCommandDelete.Parameters["@Воскресенье"].Value =
GetDbValue(textBox8.Text);

                sqlConnection.Open();
                sqlCommandDelete.ExecuteNonQuery();
```

```
                this.графикTableAdapter.Fill(this.ветеринарная_клиникаDataSet.График);
                MessageBox.Show("График успешно удален.");
            }
            catch (SqlException ex)
            {
                MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            finally
            {
                if (sqlConnection.State == ConnectionState.Open)
                    sqlConnection.Close();
            }
        }
        else MessageBox.Show("В доступе отказано.", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
    }

    private void button4_Click(object sender, EventArgs e)
    {
        try
        {
            if (string.IsNullOrWhiteSpace(textBox9.Text) || textBox9.Text == "0")
            {
                MessageBox.Show("Поле Id_Графика не должно быть пустым или равным
нулю.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            int idГрафика;
            if (!int.TryParse(textBox9.Text, out idГрафика))
            {
                MessageBox.Show("Id_Графика должно быть числом.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            if (idГрафика <= 0)
            {
                MessageBox.Show("Id_Графика должно быть больше нуля.", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            // Задание параметров с использованием метода GetDbValue
            sqlCommandUpdate.Parameters["@Id_Графика"].Value = textBox9.Text;
            sqlCommandUpdate.Parameters["@Понедельник"].Value =
GetDbValue(textBox2.Text);
            sqlCommandUpdate.Parameters["@Вторник"].Value = GetDbValue(textBox3.Text);
            sqlCommandUpdate.Parameters["@Среда"].Value = GetDbValue(textBox4.Text);
            sqlCommandUpdate.Parameters["@Четверг"].Value = GetDbValue(textBox5.Text);
            sqlCommandUpdate.Parameters["@Пятница"].Value = GetDbValue(textBox6.Text);
            sqlCommandUpdate.Parameters["@Суббота"].Value = GetDbValue(textBox7.Text);
```

```
            sqlCommandUpdate.Parameters["@Воскресенье"].Value =
GetDbValue(textBox8.Text);

            sqlConnection.Open();
            sqlCommandUpdate.ExecuteNonQuery();
            this.графикTableAdapter.Fill(this.ветеринарная_клиникаDataSet.График);
            MessageBox.Show("График успешно обновлен.");
        }
        catch (SqlException ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
        finally
        {
            if (sqlConnection.State == ConnectionState.Open)
                sqlConnection.Close();
        }
    }
  }
}
```