МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет	*Кафедра*
вычислительной техники	«Вычислительная техника»

Специальность: 09.05.01 «Применение и эксплуатация автоматизированных систем специального назначения»

Специализация: «Эксплуатация вычислительных машин, комплексов, систем и сетей специального назначения»

ДИПЛОМНЫЙ ПРОЕКТ

на тему

КЛИЕНТ-СЕРВЕРНАЯ АВТОМАТИЗИРОВАННАЯ СИСТЕМА ДЛЯ МЕДИЦИНСКИХ ОРГАНИЗАЦИЙ

Студент 14.06.24 _____ Першина Валерия Владимировна
(подпись, дата) (ФИО полностью)

Руководитель _____ Зинкин С.А.
(подпись, дата) (фамилия, инициалы) 14.06.2024

Нормоконтролёр _____ 14.06.2024 Бычков А.С.
(подпись, дата) (фамилия, инициалы)

Рецензент нач. управления криптографии _____ 12.06.24 Арюпин А.Ю.
(должность, место работы) (подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 27.05.2024 г. № 11)

Заведующий кафедрой _____ Митрохин М.А.
(подпись) (фамилия, инициалы)

Работа защищена с отметкой хорошо

(протокол заседания ГЭК от 19.06.24 № 11)

Секретарь ГЭК _____ Е.И. Гурин
(подпись) (фамилия, инициалы)

Пенза 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет Вычислительной техники

Кафедра Вычислительная техника

«Утверждаю»
Заведующий кафедрой
Митрохин М.А.
«19» апреля 2024 г.

ЗАДАНИЕ
НА ДИПЛОМНЫЙ ПРОЕКТ

1. Студент Першина Валерия Владимировна гр. 19ВОЭ1 факультета вычислительной техники.

2.Тема работы: «Клиент-серверная автоматизированная система для медицинских организаций».

Тема утверждена приказом ПГУ № 405/о от 19 апреля 2024 г.

3. Руководитель работы д.т.н., профессор кафедры ВТ «ПГУ» Зинкин С.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Разработать информационную автоматизированную систему медицинского назначения. Информационная автоматизированная система медицинского назначения должна выполнять следующие функции: добавление, изменение и удаление товаров; добавление, изменение и удаление записи к мастеру; вести учет поставок медицинских принадлежностей и средств; вести учет данных о сотрудниках, клиентах, мастерах, фирмах-поставщиках, видах товаров и косметических услугах, создавать отчетную документацию. Информационная автоматизированная система медицинского назначения должна разрабатываться с помощью среды разработки Microsoft Visual Studio 2022. База данных должна разрабатываться для СУБД «PostgreSQL».

5. Перечень подлежащих разработке вопросов.

Требуется произвести анализ предметной области, определить функциональный состав программы, выбрать стек технологий для реализации программы, разработать алгоритм работы программы, разработать диаграммы: прецедентов, активностей, последовательностей, объектов, классов, с последующим написанием программы.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	19.04.2024 – 26.04.2024	
2.	Проектирование программного приложения	10	26.04.2024 – 3.05.2024	
3.	Разработка базы данных	20	3.05.2024 – 10.05.2024	
4.	Отладка и тестирование	10	10.05.2024 17.05.2024	
5.	Разработка экономической и бжд части	10	17.05.2024 – 24.05.2024	
6.	Оформление графической части	10	24.05.2024 31.05.2024	
7.	Оформление выпускной работы	30	31.05.2024 05.06.2024	

Дата выдачи задания 19 апреля 2024 года.

Руководитель дипломного проекта _____ Зинкин С.А.

Задание к исполнению принял студент _____ Першина В.В.

Работу к защите допустить
Декан факультета _____ Фионова Л.Р.
18.06.2024

Реферат

Пояснительная записка состоит из 314 листов, 96 рисунков, 47 таблиц и 5 приложений

ИНФОРМАЦИОННАЯ АВТОМАТИЗИРОВАННАЯ СИСТЕМА, САЛОН КРАСОТЫ, МЕДИЦИНСКИЕ ПРОЦЕДУРЫ, КОСМЕТОЛОГИЧЕСКИЕ УСЛУГИ, БАЗА ДАННЫХ, МАСТЕР, КЛИЕНТ

Объектом исследования является процесс контроля работы салона красоты и учета медицинских процедур.

Цель работы – разработка информационной автоматизированной системы салона красоты.

В процессе работы необходимо выполнить анализ предметной области, сформулировать техническое задание для разрабатываемой информационной-автоматизированной системы, разработать логическую и физическую модели данных для разработки базы данных.

В результате проделанной работы разработана информационная автоматизированная система салона красоты, состоящее из базы данных и клиент-серверного приложения для работы с ней.

База данных спроектирована с использованием CASE-средства OpenModelShere 3.2. База данных разработана для СУБД PostgreSQL, приложение реализовано в среде разработки Microsoft Visual Studio 2022.

Содержание

Перечень сокращений и обозначений 5
Введение 6
1 Анализ предметной области 9
1.1 Описание предметной области 9
1.2 Анализ аналогов 11
1.3 Цели и задачи разрабатываемого программного приложения 14
1.4 Анализ целевой аудитории 15
2 Техническое задание 16
2.1 Назначение разработки 16
2.2 Требования к программе 16
2.2.1 Требования к функциональным характеристикам 16
2.2.2 Требования к надежности 17
2.2.3 Требования к составу и параметрам технических средств 18
2.3 Требования к программной документации 19
2.4 Стадии и этапы разработки 19
2.5 Порядок контроля и приемки 20
3 Обоснование целесообразности выбранных заданных средств разработки. . 21
3.1 Язык разработки C# 21
3.2 CASE-средство OpenModelSphere 3.2 22
3.3 СУБД PostgreSQL 23
3.4 Среда разработки Microsoft Visual Studio 2022 23
4 Разработка информационной-автоматизированной системы 26
4.1Логическое проектирование базы данных 26
4.2 Физическое проектирование базы данных 41
4.3Описание модулей программы 47
4.4 Методика испытаний программы 51
5 Оценка затрат на разработку и внедрение информационной-автоматизированной системы салона красоты 56

5.1 Организация производства разработки информационной-автоматизированной системы....56
5.2 Расчет себестоимости программного обеспечения для информационной-автоматизированной системы салона красоты....59
5.3 Оценка эффективности информационной-автоматизированной системы салона красоты....65
5.4 Анализ рисков при разработке и внедрении информационной-автоматизированной системы салона красоты....66
6 Общие принципы обеспечения безопасности на рабочем месте оператора информационной-автоматизированной систесы салона красоты. 68
6.1 Актуальность вопросов безопасности на рабочем месте 69
6.2 Требования безопасности при работе с электроприборами....69
6.3 Влияние ПЭВМ на физиологию человека....69
6.4 Техника безопасности при работе с персональным компьютером 70
Заключение 72
Список использованных источников....73
Приложение А – UML-диаграммы....74
Приложение А.1 – Диаграмма развертывания....75
Приложение А.2 – Логическая модель данных 76
Приложениие А.3 – Логическая модель данных....77
Приложение Б – SQL-скрипт....78
Приложение В – Результаты испытаний 97
Приложение Г – Листинг программы....139
Приложение Д – Презентация 300

Перечень сокращений и обозначений

В настоящей работе применяют следующие сокращения и обозначения

БД – база данных

ГОСТ – государственный стандарт

ЕСПД – единая система программной документации

МРОТ – минимальный размер оплаты труда

НФ – нормальная форма

ОС – операционная система

ПО – программное обеспечение

ПЭВМ – персональная электронно-вычислительная машина

ЦНС – центральная нервная система

ЭВМ – электронно-вычислительная машина

FK (Foreign Key) – внешний ключ

ID (Identifier) – идентификатор объекта

PK (Primary Key) – первичный ключ

SQL (Structured Query Language) – язык структурированных запросов

UML (Unified Modeling Language) – унифицированный язык моделирования

Введение

Современные информационные технологии играют огромную роль в различных сферах человеческой деятельности, включая медицину, образования, бизнес, науку, искусство и многие другие. Информационные технологии относятся к области информатики и компьютерных наук.

Рассмотрим информационные автоматизированные системы с точки зрения эффективности и полезности использования. Большая их часть основана на базах данных, что удобно для хранения и работы с информацией. База данных – это совокупность данных, организованных по определенным правилам, предусматривающим общие принципы описания, хранения и манипулирования данными, не зависимая от прикладных программ [1]. Эти данные относятся к определенной предметной области и используются для решения многих задач разными пользователями.

Информационные автоматизированные системы применяются для автоматизации повторяющихся процессов, которые связаны с обработкой и хранением информации. Особенностями применения информационной-автоматизированной системы в экономике являются: возможность сбора и хранения важной информации, получение ее в короткий срок, выполнение сложных экономических расчетов без участия человека, самостоятельное ведение документации.

Таким образом, информационные автоматизированные системы играют ключевую роль в современной экономике, медицине, науке и технике, и их использование будет продолжать расширяться и углубляться в будущем, открывая новые возможности для исследований и инноваций.

Разработка информационной автоматизированной системы для работы салона красоты является актуальной и необходимой по следующим причинам:

- улучшение контроля и управления персоналом (информационная автоматизированная система позволяет автоматизировать учет и управление финансами, услугами мастеров, сроки выполнения работ);

- улучшение коммуникации (информационная автоматизированная система позволяет мастерам и клиентам быстро общаться в системе и оперативно решать все вопросы);
- упрощение рутинных процессов (информационная автоматизированная система позволяет автоматизировать процессы управления мастерами и клиентами, учета процедур, планирования посещений и других рутинных задач, это способствует экономии времени и ресурсов, а также улучшает уровень сервиса для клиентов);
- улучшение безопасности (информационная автоматизированная система может быть использована для надежного хранения персональных данных клиентов).

В целом, разработка информационной автоматизированной системы салона красоты является важным шагом к повышению эффективности управления, улучшению коммуникации и упрощению рутинных процессов в данном коллективе.

Информационная автоматизированная система — это совокупность устройств и программных средств, которые предназначены для автоматизации деятельности предприятия или организации и связаны с хранением, передачей, поиском и обработкой всех объемов информации. Такие системы призваны обеспечивать качественное удовлетворение информационных запросов многочисленного пользовательского сектора благодаря эффективному хранению, поиску и передаче информации согласно запросам [2].

Высокая востребованность сферы медицины и красоты сейчас сделала работу квалифицированных специалистов еще более необходимой. Ни один новый салон красоты не обходится без ведения отчетной и текущей документации. А значит, и наличие качественного программного обеспечения, для комфортной работы специалистов необходимо.

В представленной работе описывается информационная автоматизированная система салона красоты, позволяющая упростить и автоматизировать процесс работы мастеров.

Информационная автоматизированная система салона красоты включает в себя базу данных и клиент-серверное приложение обработки данных. При проектировании базы данных использовалось CASE - средство Open ModelSphere 3.2 для СУБД PostgreSQL, для реализации приложения была применена среда разработки Microsoft Visual Studio 2022.

1 Анализ предметной области

1.1 Описание предметной области

Информационная автоматизированная система салона красоты предназначена для упрощения контроля мастеров салона красоты, учета медицинских процедур, проводимых в салоне, упрощения записи клиентов к мастерам, а также для оптимизации отчетов салона.

Актуальностью создания информационной автоматизированной системы салона красоты является современный мир, который быстро развивается. Как следствие, открытие новых перспектив для роста некоторых сфер экономической деятельности. Сфера услуг, в частности салон красоты, является одной из таких сфер. Так как отрасль быстро развивается, для ее более эффективного развития необходимо использовать информационные-автоматизированные системы.

В салонах красоты существует потребность в таких системах, которые бы смогли обеспечивать хранение информации:

– о клиентах;

– о сотрудниках;

– о всех, оказываемых в салоне, услугах.

Особенность салонов красоты подразумевает возможность хранения информации обо всех услугах, в том числе и медицинских, и то, что в них входит, а также все финансовые операции, которые потом необходимы для налоговой службы. Это приводит к тому, что в базе данных должны храниться все оказанные услуги. Также база данных должна быть защищена, так как требуется конфиденциальное хранение информации, особенно, о медицинских услугах. Создание системы, которая позволила это реализовать, обеспечило бы более качественную и оперативную работу салона красоты.

В современном мире одним из ценных ресурсов на ряду с финансовыми, материальными, человеческими и другими становится эффективное

управление. Следовательно, повышение эффективности управленческой деятельности также становится направлением совершенствования деятельности предприятия в целом. Автоматизация – является наиболее рациональным способом повышения эффективности трудового процесса.

Трудности, которые возникают при решении задачи по автоматизации управленческого труда, связаны с его спецификой. Управленческий труд отличается сложностью и многообразием, наличием большого числа форм и видов. На первый взгляд, большая его часть вообще не поддается какой-либо формализации. Именно поэтому автоматизация управленческой деятельности изначально связывалась только с автоматизацией некоторых вспомогательных, рутинных операций. Однако, благодаря быстрому развитию информационных компьютерных технологий, совершенствование технической платформы и появление новых классов программных продуктов привело к изменению подходов к автоматизации управления сферой красоты и медицины.

Наиболее распространенным форматом отчетности салонов красоты является: отчет по продажам за конкретный период, отчет по поставкам, отчет об оказанных услугах и финансовый отчет.

Отчет по продажам за конкретный период – это документ, который соединяет все данные, связанные с продажами салона за выбранный период. На основании данного отчета можно выявить объем проданного материала, его количество и стоимость.

Отчет по поставкам – это документ, в котором отображается, вся поступившая в салон продукция / материалы. На основании данного отчета можно выявить дату поставки, фирму, которая поставила товар, наименование, цену, количество и стоимость товаров.

Отчет об оказанных услугах – это документ, в котором описывается оказанная услуга. Он является одним из подтверждений факта того, что услуга была произведена. На основании данного отчета можно выявить дату и время оказания услуги, а также имя мастера и стоимость услуги.

Финансовый отчет – это документ, который хранит информацию о финансовом положении салона. На основании данного отчета можно отследить доходы и расходы салона на конкретные услуги / материалы.

Целью данной работы является разработка информационной-автоматизированной системы, которая будет представлять собой клиент-серверное приложение со встроенной базой данных. На сервере будет храниться вся информация о сотрудниках, клиентах и всех услугах, как медицинских, так и бьюти. Клиентская часть будет отвечать за: регистрацию пользователя в системе, выбор услуги, выбор специалиста, запись к нему через приложение, рассылка напоминание о записи, формирование отчетов о поставках, о продажах, об оказанных услугах и финансовый отчет.

Данная информационная автоматизированная система может быть применена не только в салонах красоты, но и в других заведениях, которые оказывают подобные услуги. Спрос на сотрудников салонов растет равномерно по всем категориям. Вакансий для мастеров стало больше на 22%, для менеджеров по продажам - на 24%, а на сотрудников зоны ресепшен - на 28% [3]. Из всего выше сказанного можно сделать вывод, что данная предметная область актуальна, так как с каждым годом растет спрос на программное обеспечение в данной сфере.

1.2 Анализ аналогов

Для того, чтобы определиться с функционалом разрабатываемой информационной-автоматизированной системы необходимо проанализировать аналоги. Такими системами являются: EasyWeek, ПрофиГид, CleverBox CRM, Арника, 1С: Салон красоты, SONLINE, UNIVERSE-CRM, S2(SALESAPCRM), YCLIENTS, Rubitime.

EasyWeek – простая в освоении CRM. Отлично подходит для сферы услуг. В отличие от большинства CRM систем, здесь есть все необходимое для полноценного ведения бизнеса. Имеет виджеты для онлайн записи и синхронизацию с календарем. Лендинг компании – имеет возможность

настройки цветов и загрузку фото. После того, как клиент записался ему приходит уведомление по SMS. Система хранит информацию о клиентах и том, какими услугами они чаще всего пользуются. Формируются отчеты по работе мастеров и посещаемости клиентов. Есть возможность подключить различные дополнительные сервисы. Стоимость от 30 000 рублей в год.

ПрофиГид – CRM + мобильное приложение. Есть отдельное приложение для мастеров, а есть брендированное для клиентов. Есть возможность создавать расписание для мастеров. Push-уведомления для мастеров и клиентов в мобильном приложении. Есть клиентская база. Нет возможности формировать отчеты. Стоимость от 90 000 рублей за 3 сотрудников.

CleverBox CRM – облачная система для beauty-сферы. Есть календарь, где наглядно показано сколько визитов запланировано. Онлайн запись для клиентов с SMS оповещениями. Формирование клиентской базы с возможностью просматривать историю визитов. Система формирует отчеты по финансам и активности работы персонала. Есть интеграция с Viber и Telegram. Использование системы необходимо оплачивать в долларах, что сейчас сделать невозможно.

Арника – система для управления бизнесом в сфере услуг. Предназначена для работы как с большими салонами, так и с частными мастерами. Возможность онлайн-записи для клиентов с подтверждением на почту. Составляет график работы сотрудников. Карточки клиентов хранят историю визитов. Есть возможность формирования отчетов по выручке и расходу материалов. Система использует реферальную программу и запись разговоров. Цена от 102 000 рублей за год.

1С: Салон красоты – многофункциональная CRM, которая больше подходит крупным салонам и сетям. Учитывает посещаемость, не требует администратора при записи. Система следит за количеством отработанных часов сотрудником. Есть SMS-уведомления для клиентов. Активно используются инструменты по привлечению и удержанию клиентов.

Формирование отчетов как по клиентам и сотрудникам, так и по финансам. Стоимость от 138 000 рублей за год.

SONLINE существует с 2011 года. В CRM есть виджеты онлайн-записи и через приложение. Отдельный модуль по персоналу с графиком работы сотрудников. Для клиентов есть SMS рассылка и карточки со статистикой. Более 10 отчетов по финансам, сотрудникам и клиентам. Возможность оформления услуги в долг. Стоимость от 16 800 за 10 мастеров с возможностью подключения дополнительных платных модулей.

UNIVERSE-CRM – удобная CRM с бесплатным доступом на 3 месяца. Работает с 2003 года. Содержит в себе все необходимые модули для сферы бьюти-услуг. Формируется клиентская база, проводится ее анализ посещаемости. Отчеты в виде таблиц по клиентам, сотрудникам и финансам. Есть возможность добавить полное описание для услуги. Стоимость на каждого специалиста 2 600 рублей в год.

S2 – CRM подходит для разного бизнеса. Есть возможность отключения неиспользуемых функций. Клиенты не могут записаться самостоятельно, только через администратора системы, но SMS о дате и времени визита приходят. Формирует полную базу по клиентам и формирует самые важные отчеты, а также наглядные диаграммы и воронки продаж. Стоимость от 2 880 рублей в год за одного сотрудника.

YCLIENTS – программа для управления салоном красоты. Сервис работает с 2010 года. Огромное количество функционала для настройки системы. Данную CRM можно использовать в различных сферах от медицинских центров до фитнес-центров. Есть функция онлайн-записи и возможность адаптирования цветовой палитры. Для клиентов предусмотрено 10 видов уведомлений, для мастеров 4. База клиентов единая для всей сети и карточки клиента с историей посещения. CRM формирует отчет по мастерам, активности клиентов и товарообороту. Стоимость данной системы от 14 600 рублей за год при условии, что сотрудников меньше 5.

Rubitime – сервис онлайн-записи клиентов. Подходит для любого бизнеса в сфере услуг. Есть возможность онлайн записи, после чего записи вносятся в электронный журнал, а клиент получает SMS уведомление. Клиентская база хранит информацию о карточках клиента, истории посещений и сумме оказанных услуг. Сервис формирует отчет по сотрудникам и востребованности мастеров. Есть отдельное мобильное приложение и промокоды для клиентов. Стоимость данной системы от 4 036 рублей за год за одного сотрудника.

После проведения анализа аналогов разрабатываемой информационной-автоматизированной системы салона красоты можно сделать вывод, что данная система будет нацелена не на крупный бизнес, а на небольшие салоны с количеством сотрудников до 10 человек. В системе будут реализованы только те функции, которые действительно необходимы сотрудникам и клиентам.

1.3 Цели и задачи разрабатываемого программного приложения

Цели автоматизации работы салона красоты:

- увеличение скорости работы мастеров;
- обеспечение контроля всей деятельности мастера по предоставлению услуг в салоне красоты;
- структуризация деятельности мастеров.

В результате внедрения информационной-автоматизированной системы планируется достичь ряда задач, а именно:

- повысить эффективность работы мастеров;
- снизить лишнюю нагрузку на мастеров;
- повысить скорость регистрации клиентов.

1.4 Анализ целевой аудитории

На основании того, что все вышеописанные программные продукты предназначены для работы мастеров или большого штата сотрудников, продуктов, подходящих для универсальной работы салона красоты разной направленности, не было найдено. Как следствие, разработка информационной-автоматизированной системы салона красоты является актуальной как для малых частных компаний, так и для крупных сетей, у которых есть штат мастеров.

Делая вывод, основной целевой группой являются мастера и клиенты малых частных компаний и крупных сетей салонов красоты, но за счет простоты и универсальности рабочего пространства системы, его могут использовать все, кому необходимо фиксировать работу мастеров.

2 Техническое задание

2.1 Назначение разработки

Информационная автоматизированная система салона красоты должна быть предназначена для автоматизации основных направлений деятельности мастеров, клиентов и администраторов салона красоты.

Цель данного проекта – разработка базы данных и клиент-серверного приложения для информационной-автоматизированной системы салона красоты.

2.2 Требования к программе

2.2.1 Требования к функциональным характеристикам

Программное обеспечение должно выполнять следующие функции:

– формирование договоров на поставку;

– учет поставок;

– оформление заказа;

– учет сведений по проданным товарам;

– хранение справочных данных:

1) данные о товарах;

2) данные о сотрудниках;

3) данные о покупателях;

4) данные о мастерах;

5) данные о фирмах-поставщиках;

6) данные об услугах;

7) данные о видах товаров;

8) данные о перечне косметических услуг, которые необходимо оказать;

– формирование договоров на оказание услуг;

– формирование прайс-листа;

– формирование графика приема;

– формирование отчетных документов:

1) отчет по поставкам;

2) отчет по продажам;

3) отчет по оказанным услугам;

4) финансовый отчет.

Входными данными информационной-автоматизированной системы должны быть:

– реквизиты поставщика;

– счет-фактура;

– заказ;

– платежный документ;

– реквизиты мастера.

Выходными данными информационной-автоматизированной системы должны быть:

– заявка на поставку;

– платежное поручение;

– товарная накладная;

– прайс-лист;

– график приема;

– отчет по поставкам;

– отчет по продажам;

– отчет по оказанным услугам;

– финансовый отчет.

2.2.2 Требования к надежности

Информационная автоматизированная система салона красоты должна работать без перебоев, контролировать вводимые данных и в случае некорректного ввода данных должна сообщать об этом пользователю.

Информационная автоматизированная система салона красоты должна обеспечивать защиту данных от несанкционированного доступа на основе процедуры логирования пользователей.

При вводе входной информации должен осуществляться ее контроль на допустимость вводимых значений по следующим правилам:

– вводимые данные не должны противоречить заявленному типу данных;

– данные, имеющие формат номера телефона, должны заноситься по соответствующему формату;

– вводимые данные не должны противоречить законам логики.

При выполнении операции по изменению и удалению данных должна быть обеспечена целостность базы данных.

Конечный пользователь не должен иметь доступ к исходным данным приложения.

2.2.3 Требования к составу и параметрам технических средств

Информационная автоматизированная система должна быть предназначено для использования как на локальном компьютере, так и в сети. Рабочие станции должны иметь следующие минимальные технические характеристики:

– операционная система Windows 7 и выше;

– процессор Intel Core 1,9 ГГц и выше;

– свободное место на жестком диске 2 ГБ и выше;

– оперативную память 2 ГБ и выше;

– монитор с диагональю не менее 16 дюймов;

– клавиатура;

– мышь.

Сервер в сети должен иметь следующие технические характеристики:

– операционная система Windows 7 и выше;

– процессор Intel Core 1,9 ГГц и выше;

– свободное место на жестком диске 4 ГБ и выше;

– оперативную память 8 ГБ и выше;

– монитор с диагональю не менее 16 дюймов;

– клавиатура;

– мышь.

2.3 Требования к программной документации

Информационная автоматизированная система салона красоты должна сопровождаться документацией, содержащей:

– техническое задание;

– описание логического проектирования БД;

– описание физического проектирования БД;

– описание программы;

– программу и методику испытаний;

– описание применения.

Структура и содержание программной документации должны соседствовать требованиям стандартов Единой системы программной документации (ЕСПД).

2.4 Стадии и этапы разработки

Этапы разработки информационная автоматизированная система салона красоты представлены в таблице 1.

Таблица 1 — Стадии и этапы разработки

Стадии	Этапы	Объем работы в процентах	Срок выполнения	Исполнитель
1 Анализ предметной области	Анализ предметной области	10		Першина В.В.

Стадии	Этапы	Объем работы в процентах	Срок выполнения	Исполнитель
2 Проектирование программного приложения	-концептуальное проектирование БД -логическое проектирование БД -физическое проектирование БД	10		Першина В.В.
3 Разработка базы данных	Разработка базы данных	20		Першина В.В.
4 Отладка и тестирование	Отладка и тестирование	10		Першина В.В.
5 Разработка экономической части	Оценка затрат на разработку и внедрение программного приложения контроля выполнения задач	5		Першина В.В.
6 Разработка раздела БЖД	Оценка тяжести труда эксплуататора программного приложения	5		Першина В.В.
7 Оформление графической части	Оформление графической части	10		Першина В.В.
8 Оформление Выпускной работы	Оформление пояснительной записки	30		Першина В.В.

2.5 Порядок контроля и приемки

Для проверки выполнения системой всех ее функций в соответствии с требованием технического задания необходимо разработать тестовые наборы данных. Протестировать действия системы на возможные действия пользователя: запуск программы, заполнение полей данными, выбор пунктов меню, нажатие кнопок, работа с базой данных. Система соответствует

требованиям технического задания, если она выполняет все заданные функции при различных входных данных из тестовых наборов данных.

Проверка и приемка системы должна осуществляться при ее работоспособности и в присутствии заказчика, выполнения всех возложенных на нее функциональных возможностей, а также при наличии полной документации на программу (пояснительная записка с приложениями: текстом программы и результатом текстовых примеров).

3 Обоснование целесообразности использования выбранных средств разработки

На основании анализа требований к функциональным характеристикам информационной-автоматизированной системы салона красоты были выбраны наиболее актуальные средства разработки.

3.1 Язык разработки C#

Язык программирования – это набор правил, предназначенный для записи компьютерных программ. Далее из полученного текста специализированные программы (трансляторы, компоновщики и др.) практически без участия человека формируют код, предназначенный для процессора. В объектно-ориентированных языках центральное место отведено данным, а выполнение функций так или иначе организовано вокруг этих данных.

Когда суммарный вес специфических добавок превысил некоторый порог, появился новый язык программирования C# (произносится как си шарп, хотя разработчики предполагали название си-диез), вобравший в себя наиболее значимые черты своих предшественников и в большей степени отвечающий современным потребностям [4].

C# — это язык программирования, который был разработан сотрудником компании Microsoft Андерсом Хейлсбергом. Изначально его создавали для проектов под Windows, но сейчас это универсальный язык, который используется для создания игр, веб-сервисов, десктопных приложений и даже для разработки нейросей.

Язык C# является:

– кросс-платформенным (он запускается на любом аппарате);

–объектно-ориентированным (он состоит из классов и объектов, которые, в свою очередь, могут передавать свойства друг другу);

– постоянно развивающийся.

Язык C# разработан прежде всего для платформы .NET, которая является средой, объединяющей программные технологии, для разработки Web- и Windows-приложений (отсюда и название).

Основу среды .NET составляет CLR (Common Language Runtime) – общеязыковая среда исполнения, которая состоит из двух основных частей:

– ядра (набор служб, управляющих загрузкой приложения в память, собраны в библиотеке mscoree.dll);

– библиотеки базовых классов (главная сборка в библиотеке mscorlib.dll) [4].

3.2 CASE-средство OpenModelSphere 3.2

Для разработки базы данных используются специальные CASE-средства для разработки концептуальных, логических и реляционных моделей.

Популярным CASE-средством является OpenModelSphere 3.2 - бесплатные инструменты моделирования, доступные в виде программного 19 обеспечения с открытым исходным кодом, выпущенного под GPL (общедоступная лицензия GNU) [6].

Преимущества OpenModelSphere 3.2:

– обеспечивание поддержки прямого и обратного проектирования между UML и реляционными схемами;

– работает с Oracle, Informix, Microsoft SQL Server, Sybase, DB2, PostgreSQL;

– построение графической модели БД в нотациях: IDEF1X,IE, UML;

– создание таблицы связей БД, в т.ч. «многие ко многим»;

– бесплатное ПО.

Недостатками OpenModelSphere 3.2 являются:

– ограничены возможности по обеспечению многопользовательской среды;

– невозможность построения моделей и их синхронизации с БД на сервере.

3.3 СУБД PostgreSQL

Для функционирования базы данных необходимо выбрать систему управления базами данных.

СУБД – это набор программ, управляющие структурой БД и контролирующие доступ к данным, хранящимся в БД. Система управления служит посредником между пользователем и БД. Сама структура БД хранится в виде набора файлов, и единственный способ получить доступ к данным в этих файлах – через СУБД.

СУБД PostgreSQL – это свободная объектно-реляционная система управления базами данных.

Выделяется ряд особенностей PostgreSQL:

– поддерживается БД неограниченного размера;

– обладает легкой расширяемостью;

– расширяемая система встроенных языков программирования;

– функции являются блоками кода, исполняемыми на сервере, а не на клиенте БД;

–триггеры определяются как функции, инициируемые DML-операциями;

– механизм правил – есть механизм создания пользовательских обработчиков;

– наследование реализовано на уровне таблиц, которые могут наследовать характеристики и наборы полей от других таблиц [7].

3.4 Среда разработки Microsoft Visual Studio 2022

Microsoft Visual Studio 2022 — это современная интегрированная среда разработки (Integrated Development Environment, IDE), она предлагает набор

инструментов для разработки ПО для Windows, Android, iOS, а также веб-приложений и облачных сервисов.

К главным возможностям Microsoft Visual Studio 2022 можно отнести:

– Многоязычная поддержка.

– Редактирование и отладка кода.

– Интеграция с Azure.

– Совместная работа и управление версиями.

– Создание пользовательского интерфейса.

– Расширяемость [5].

В Visual Studio 2022 существует множество расширений, с помощью которых есть возможность настраивать среду разработки под конкретные требования и добавлять новые функции.

Visual Studio 2022 – предназначена как для учащихся, так и для тех, кто создает масштабные проекты, все потому, что она стала более быстрая, производительная и упрощенная. Также в ней были улучшены производительности многих основных возможностей.

В Visual Studio 2022 возможность IntelliCode может автоматически завершать целую строку кода за раз. IntelliCode теперь может обнаружить, когда выполняется общая задача и рекомендуется правильное быстрое действие, завершив его правильно по мере ввода.

VS 2022 для Windows теперь стал 64-разрядным приложением. Это означает, что можно открывать, изменять, запускать и отлаживать любые по объему и сложности решения, не беспокоясь о нехватке памяти. При выполнении повседневных дел, таких как написание кода и переключение ветвей, система реагирует быстрее и плавнее [5].

Одним из нововведений является возможность «Горячая перезагрузка» для .NET и C++, с помощью которой можно сэкономить время. Вносятся изменения и применяется горячая перезагрузка, она применит внесенные изменения к работающему приложению, чтобы можно было их увидеть сразу, без остановки для перестроения.

Горячая перезагрузка – это функция, которая позволяет изменять исходный код приложения и мгновенно применять внесенные изменения к работающему приложению. Ее цель – повысить продуктивность разработчиков, исключив перезапуски приложения между редактированиями. Горячая перезагрузка доступна в Visual Studio 2022 и программе командной строки dotnet watch. Горячая перезагрузка работает с большинством типов приложений .NET, а также с исходным кодом на C#, Visual Basic и C++ [5].

С VS 2022 доступен C# версии 10, в котором есть возможности присвоения и объявления в одном и том же деконструировании. С помощью данного изменения снимаются ограничения, которые были в прошлых версиях C#. В предыдущих версиях деконструирование могло присвоить все значения существующим переменным или инициализировать только что объявленные переменные.

4 Разработка информационной-автоматизированной системы

Диаграмма развертывания информационной-автоматизированной системы салона красоты представлена на рисунке 1.

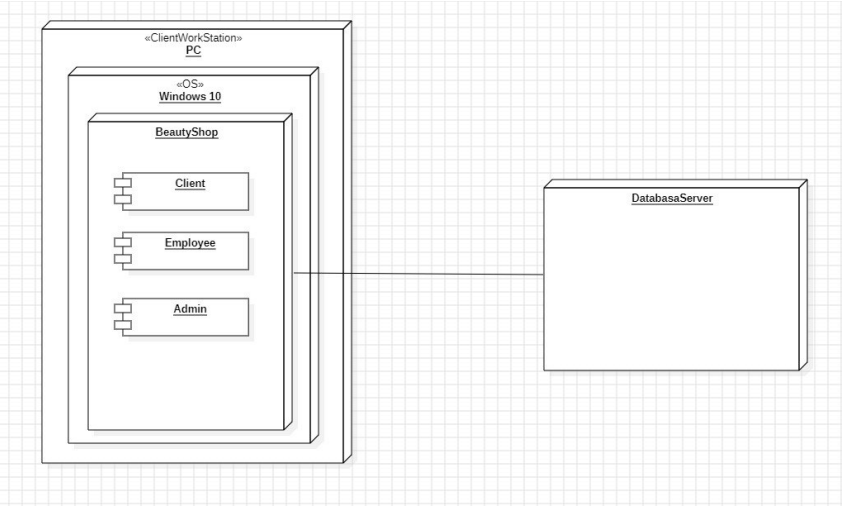


Рисунок 1 – Диаграмма развертывания

4.1 Логическое проектирование базы данных

Логическая модель базы данных приведена на рисунке 2 и содержит восемнадцать сущностей.

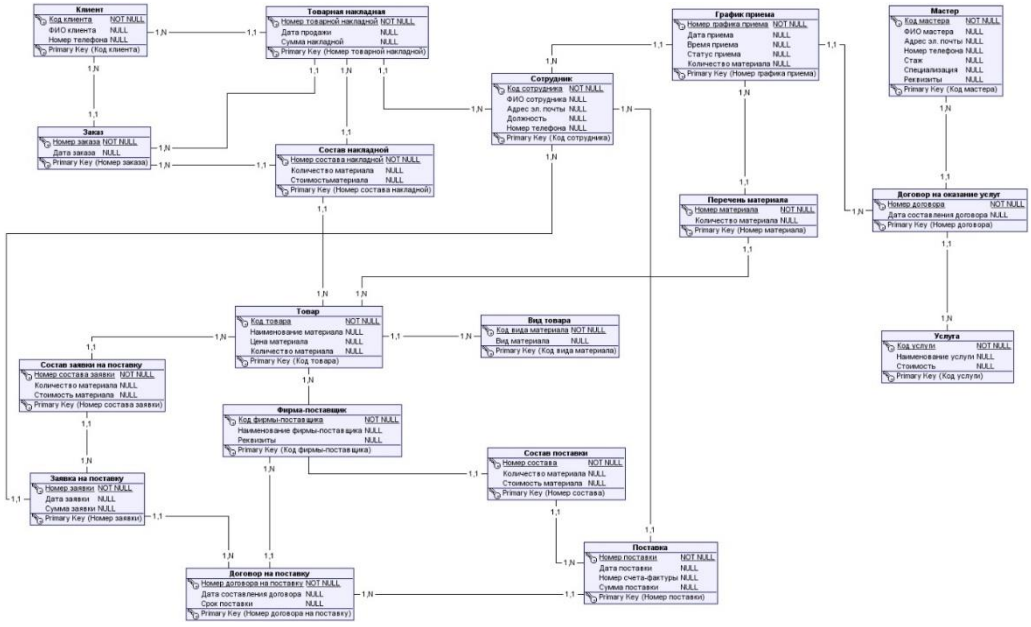


Рисунок 2 – Логическая модель данных

Описание атрибутов сущностей логической модели приведено в таблицах 2 – 19.

Таблица 2 – Описание атрибутов сущности «Клиент»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код клиента	ID клиента	PK	
ФИО клиента	ФИО клиента		
Номер телефона	Номер телефона клиента		

Таблица 3 – Описание атрибутов сущности «Заказ»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер заказа	ID заказа	PK	
Дата заказа	Дата заказа		

Таблица 4 - Описание атрибутов сущности «Товарная накладная»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер товарной накладной	ID товарной накладной	PK	
Дата заказа	Дата создания заказа		
Сумма накладной	Общая сумма в накладной		

Таблица 5 - Описание атрибутов сущности «Состав накладной»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер состава накладной	ID состава накладной	PK	

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Количество материалов	Количество материала в накладной		
Стоимость материалов	Стоимость материалов в накладной		

Таблица 6 – Описание атрибутов сущности «Сотрудник»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код сотрудника	ID сотрудника	PK	
ФИО сотрудника	ФИО сотрудника		
Адрес эл. Почты	Адрес электронной почты сотрудника		
Должность	Должность сотрудника		
Номер телефона	Номер телефона сотрудника		

Таблица 7 – Описание атрибутов сущности «График приема»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер графика приема	ID графика приема	PK	
Дата приема	Дата приема клиента		
Время приема	Время приема клиента		
Статус приема	Статус проведения процедуры		
Количество материалов	Количество, необходимого для процедуры, материала		

Таблица 8 – Описание атрибутов сущности «Перечень материалов»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер материала	ID материала	PK	
Количество материала	Количество материала		

Таблица 9 – Описание атрибутов сущности «Договор на оказание услуг»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер договора	ID договора	PK	
Дата составления договора	Дата оформления договора		

Таблица 10 – Описание атрибутов сущности «Мастер»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код мастера	ID мастера	PK	
ФИО мастера	ФИО мастера		
Адрес эл. Почты	Адрес электронной почты мастера		
Номер телефона	Номер телефона мастера		
Стаж	Стаж мастера		
Специализация	Специализация мастера		
Реквизиты	Реквизиты мастера для оплаты услуг		

Таблица 11 – Описание атрибутов сущности Услуга

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код услуги	ID услуги	PK	
Наименование услуги	Наименование услуги		

Таблица 12 – Описание атрибутов сущности «Состав заявки на поставку»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер состава заявки	ID состава заявки	PK	
Количество материала	Количество материала		
Стоимость материала	Стоимость материала		

Таблица 13 – Описание атрибутов сущности «Заявка на поставку»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер заявки	ID заявки	PK	
Дата заявки	Дата создания заявки		
Сумма заявки	Общая сумма заявки		

Таблица 14 – Описание атрибутов сущности «Фирма поставщик»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код фирмы-поставщика	ID фирмы-поставщика	PK	
Наименование фирмы-поставщика	Название фирмы-поставщика		
Реквизиты	Реквизиты фирмы-поставщика		

Таблица 15 – Описание атрибутов сущности «Товар»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код товара	ID товара	PK	
Наименование материала	Название материала		
Цена материала	Цена за материал		
Количество материалов	Количество материала		

Таблица 16 – Описание атрибутов сущности «Договор на поставку»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер договора на поставку	ID договора на поставку	PK	
Дата составления договора	Дата составления договора		

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Срок поставки	Срок поставки		

Таблица 17 – Описание атрибутов сущности «Вид товара»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код вида материала	ID материала	PK	
Вид материала	Вид материала		

Таблица 18 – Описание атрибутов сущности «Состав поставки»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Номер состава	ID состава	PK	
Количество материала	Количество материала		
Стоимость материала	Стоимость материала		

Таблица 19 – Описание атрибутов сущности «Поставка»

Имя атрибута	Назначение	Первичный ключ	Внешний ключ
Код поставки	ID поставки	PK	
Дата поставки	Дата поставки		
Номер счета-фактуры	Номер счета-фактуры		
Сумма поставки	Сумма поставки		

Процесс нормализации – это последовательный процесс приведения базы данных к эталонному виду, то есть переход от одной нормальной формы к следующей.

Отношение находится в 1 НФ, если все строки таблицы различные и все атрибуты являются атомарными.

Отношение находится во 2 НФ, если оно находится в 1 НФ и каждый не ключевой атрибут таблицы функционально полно зависит от первичного ключа.

Отношение находится в 3 НФ, если оно находится во 2 НФ и между не ключевыми атрибутами не существует транзитивных зависимостей.

Все атрибуты восемнадцати сущностей атомарны и находятся в 1 НФ. Для подтверждения данного фактора необходимо рассмотреть каждую сущность:

– сущность «Клиент» имеет ключевой атрибут «Код клиента». Атрибуты «ФИО клиента», «Номер телефона» данной сущности являются простыми и неделимыми;

– сущность «Заказ» имеет ключевой атрибут «Номер заказа». Атрибут «Дата заказа» данной сущности является простым и неделимым;

– сущность «Товарная накладная» имеет ключевой атрибут «Номер товарной накладной». Атрибуты «Дата продажи», «Сумма накладной» данной сущности являются простыми и неделимыми;

– сущность «Состав накладной» имеет ключевой атрибут «Номер состава накладной». Атрибуты «Количество материала», «Стоимость материала» данной сущности являются простыми и неделимыми;

– сущность «Сотрудник» имеет ключевой атрибут «Код сотрудника». Атрибуты «ФИО сотрудника», «Адрес эл. почты», «должность», «Номер телефона» данной сущности являются простыми и неделимыми;

– сущность «График приема» имеет ключевой атрибут «Номер графика приема». Атрибуты «Дата приема», «Время приема», «Статус приема», «Количество материала» данной сущности являются простыми и неделимыми;

– сущность «Перечень материала» имеет ключевой атрибут «Номер материала». Атрибут «Количество материала» данной сущности является простым и неделимым;

– сущность «Мастер» имеет ключевой атрибут «Код мастера». Атрибуты «ФИО мастера», «Адрес эл. почты», «Номер телефона», «Стаж», «Специализация», «Реквизиты» данной сущности являются простыми и неделимыми;

– сущность «Договор на оказание услуг» имеет ключевой атрибут «Номер договора». Атрибут «Дата составления договора» данной сущности является простым и неделимым;

– сущность «Услуга» имеет ключевой атрибут «Код услуги». Атрибуты «Наименование услуги», «Стоимость» данной сущности являются простыми и неделимыми;

– сущность «Состав заявки на поставку» имеет ключевой атрибут «Номер состава заявки». Атрибуты «Количество материала», «Стоимость материала» данной сущности являются простыми и неделимыми;

– сущность «Заявка на поставку» имеет ключевой атрибут «Номер заявки». Атрибуты «Дата заявки», «Сумма заявки» данной сущности являются простыми и неделимыми;

– сущность «Товар» имеет ключевой атрибут «Код товара». Атрибуты «Наименование материала», «Цена материала», «Количество материала» сущности являются простыми и неделимыми;

– сущность «Фирма-поставщик» имеет ключевой атрибут «Код фирмы-поставщика». Атрибуты «Наименование фирмы-поставщика», «Реквизиты» данной сущности являются простыми и неделимыми;

– сущность «Договор на поставку» имеет ключевой атрибут «Номер договора на поставку». Атрибуты «Дата составления договора», «Срок поставки» данной сущности являются простыми и неделимыми;

– сущность «Вид товара» имеет ключевой атрибут «Код вида товара». Атрибут «Вид материала» данной сущности является простым и неделимым;

– сущность «Состав поставки» имеет ключевой атрибут «Номер состава». Атрибуты «Количество материала», «Стоимость материала» данной сущности являются простыми и неделимыми;

– сущность «Поставка» имеет ключевой атрибут «Номер поставки». Атрибуты «Дата поставки», «Номер счета-фактуры», «Сумма поставки» данной сущности являются простыми и неделимыми.

Все отношения логической модели данных находятся во 2 НФ, так как их не ключевые атрибуты функционально полно зависят от первичных ключей.

Сущность «Клиент» находится во 2 НФ, так как имеет единственный ключевой атрибут «Код клиента». Не ключевые атрибуты «ФИО клиента», «Номер телефона» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Заказ» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер заказа». Не ключевой атрибут «Дата заказа» данной сущности функционально полно зависит от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Товарная накладная» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер товарной накладной». Не ключевые атрибуты «Дата продажи», «Сумма накладной» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Состав накладной» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер состава накладной». Не ключевые атрибуты «Количество материала», «Стоимость материала» данной сущности

функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Сотрудник» находится во 2 НФ, так как имеет единственный ключевой атрибут «Код сотрудника». Не ключевые атрибуты «ФИО сотрудника», «Адрес эл. почты», «должность», «Номер телефона» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «График приема» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер графика приема». Не ключевые атрибуты «Дата приема», «Время приема», «Статус приема», «Количество материала» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Перечень материала» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер материала». Не ключевой атрибут «Количество материала» данной сущности функционально полно зависит от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Мастер» находится во 2 НФ, так как имеет единственный ключевой атрибут «Код мастера». Не ключевые атрибуты «ФИО мастера», «Адрес эл. почты», «Номер телефона», «Стаж», «Специализация», «Реквизиты» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Договор на оказание услуг» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер договора». Не ключевой атрибут «Дата составления договора» данной сущности функционально полно зависит от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Услуга» находится во 2 НФ, так как имеет единственный ключевой атрибут «Код услуги». Не ключевые атрибуты «Наименование услуги», «Стоимость» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Состав заявки на поставку» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер состава заявки». Не ключевые атрибуты «Количество материала», «Стоимость материала» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Заявка на поставку» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер заявки». Не ключевые атрибуты «Дата заявки», «Сумма заявки» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Товар» находится во 2 НФ, так как имеет единственный ключевой атрибут «Код товара». Не ключевые атрибуты «Наименование материала», «Цена материала», «Количество материала» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Фирма-поставщик» находится во 2 НФ, так как имеет единственный ключевой атрибут «Код фирмы-поставщика». Не ключевые атрибуты «Наименование фирмы-поставщика», «Реквизиты» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Договор на поставку» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер договора на поставку». Не ключевые атрибуты «Дата составления договора», «Срок поставки» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Вид товара» находится во 2 НФ, так как имеет единственный ключевой атрибут «Код вида товара». Не ключевой атрибут «Вид материала» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Состав поставки» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер состава». Не ключевые атрибуты «Количество материала», «Стоимость материала» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

Сущность «Поставка» находится во 2 НФ, так как имеет единственный ключевой атрибут «Номер поставки». Не ключевые атрибуты «Номер поставки». Атрибуты «Дата поставки», «Номер счета-фактуры», «Сумма поставки» данной сущности функционально полно зависят от данного первичного ключа. Следовательно, можно сделать вывод о том, что данная сущность находится во 2 НФ.

В сущности «Клиент» не ключевые атрибуты «ФИО клиента», «Номер телефона» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Заказ» не ключевой атрибут «Дата заказа» данного отношения не зависит от других атрибутов. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Товарная накладная» не ключевые атрибуты «Дата продажи», «Сумма накладной» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Состав накладной» не ключевые атрибуты «Количество материала», «Стоимость материала» данного отношения не зависят друг от

друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Сотрудник» не ключевые атрибуты «ФИО сотрудника», «Адрес эл. почты», «должность», «Номер телефона» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «График приема» не ключевые атрибуты «Дата приема», «Время приема», «Статус приема», «Количество материала» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Перечень материала» не ключевой атрибут «Количество материала» данного отношения не зависит от других атрибутов. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Мастер» не ключевые атрибуты «ФИО мастера», «Адрес эл. почты», «Номер телефона», «Стаж», «Специализация», «Реквизиты» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Договор на оказание услуг» не ключевой атрибут «Дата составления договора» данного отношения не зависит от других атрибутов. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Услуга» не ключевые атрибуты «Наименование услуги», «Стоимость» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Состав заявки на поставку» не ключевые атрибуты «Количество материала», «Стоимость материала» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Заявка на поставку» не ключевые атрибуты «Дата заявки», «Сумма заявки» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Товар» не ключевые атрибуты «Наименование материала», «Цена материала», «Количество материала» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Фирма-поставщик» не ключевые атрибуты «Наименование фирмы-поставщика», «Реквизиты» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Договор на поставку» не ключевые атрибуты «Дата составления договора», «Срок поставки» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Вид товара» не ключевой атрибут «Вид материала», данного отношения не зависит от других атрибутов. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Состав поставки» не ключевые атрибуты «Количество материала», «Стоимость материала» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

В сущности «Поставка» не ключевые атрибуты «Номер поставки». Атрибуты «Дата поставки», «Номер счета-фактуры», «Сумма поставки» данного отношения не зависят друг от друга. Следовательно, можно сделать вывод о том, что данная сущность логической модели находится в 3 НФ.

Таким образом, была разработана логическая модель данных.

4.2 Физическое проектирование

На основании разработанной логической модели данных была построена физическая модель данных. Физическая модель данных разработана для СУБД PostgreSQL и представлена на рисунке 3.

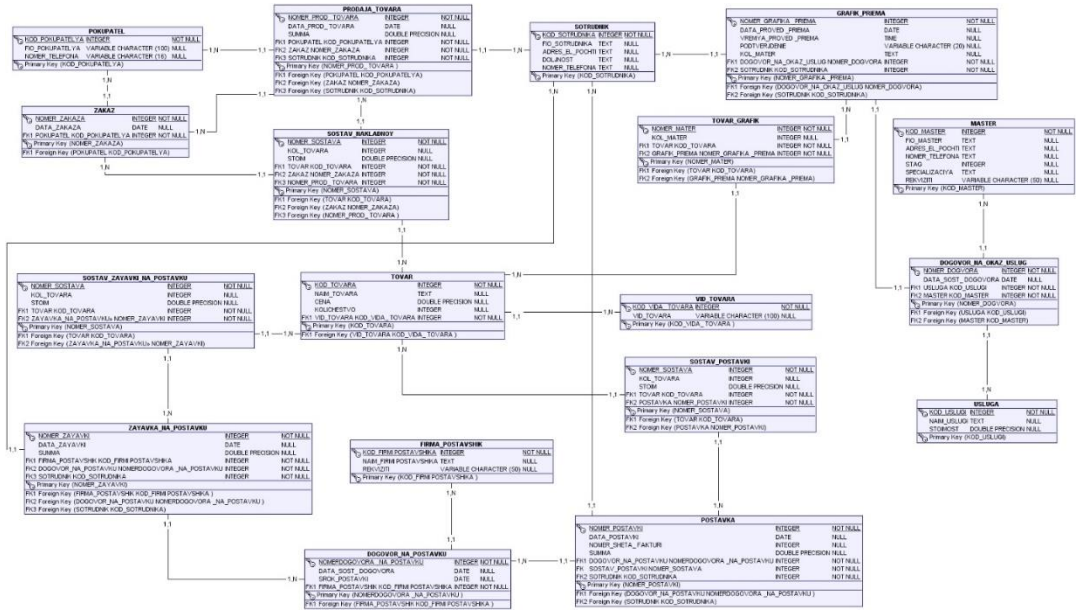


Рисунок 3 – Физическая модель данных

Физическая модель данных содержит восемнадцать таблиц. Описание таблиц физической модели приведено в таблицах 20 - 37.

Таблица 20 – Описание полей таблицы «POKUPATEL»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Kod_pokupatelya	Kod_pokupatelya	Не допустимо	PK	INTEGER
Fio_pokupatelya	Fio_pokupatelya	Допустимо		VARCHAR (100)
Nomer_telefona	Nomer_telefona	Допустимо		VARCHAR (16)

Таблица 21 – Описание полей таблицы «ZAKAZ»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_zakaza	Nomer_zakaza	Не допустимо	PK	INTEGER
Data_zakaza	Data_zakaza	допустимо		DATE

Таблица 22 – Описание полей таблицы «PRODAJA_TOVARA»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_prod_tovara	Nomer_prod_tovara	Не допустимо	PK	INTEGER
Data_prod_tovara	Data_prod_tovara	допустимо		DATE
summa	summa	допустимо		DOUBLE PRECISION

Таблица 23 – Описание полей таблицы «SOSTAV_NAKLADNOY»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_sostava	Nomer_sostava	Не допустимо	PK	INTEGER
Kol-tovara	Kol-tovara	допустимо		INTEGER
stoim	stoim	допустимо		DOUBLE PRECISION

Таблица 24 – Описание полей таблицы «SOTRUDNIK»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Kod_sotrudnika	Kod_sotrudnika	Не допустимо	PK	INTEGER
Fio_sotrudnika	Fio_sotrudnika	допустимо		TEXT
Adres_el_pochti	Adres_el_pochti	допустимо		TEXT
doljnost	doljnost	допустимо		TEXT
Nomer_telefona	Nomer_telefona	допустимо		TEXT

Таблица 25 – Описание полей таблицы «GRAFIK_PRIEMA»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_grafika_priema	Nomer_grafika_priema	Не допустимо	PK	INTEGER
Data_provedenia_priema	Data_provedenia_priema	допустимо		DATE
Vremia_proved_priema	Vremia_proved_priema	допустимо		TIME
podtverjdenie	podtverjdenie	допустимо		VARCHAR(20)
Kod_mater	Kod_mater	допустимо		TEXT

Таблица 26 – Описание полей таблицы «TOVAR_GRAFIK»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_mater	Nomer_mater	Не допустимо	PK	INTEGER
Kol_mater	Kol_mater	допустимо		INTEGER

Таблица 27 – Описание полей таблицы «MASTER»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Kod_mastera	Kod_mastera	Не допустимо	PK	INTEGER
Fio_mastera	Fio_mastera	допустимо		TEXT
Adres_el_pochti	Adres_el_pochti	допустимо		TEXT
Nomer_telefona	Nomer_telefona	допустимо		TEXT
stag	stag	допустимо		INTEGER
specializaciya	specializaciya	допустимо		TEXT
rekviziti	rekviziti	допустимо		VARCHAR(50)

Таблица 28 – Описание полей таблицы «DOGOVOR_NA_OKAZ_USLUG»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_dogovora	Nomer_dogovora	Не допустимо	PK	INTEGER
Data_sost_dogovora	Data_sost_dogovora	допустимо		DATE

Таблица 29 – Описание полей таблицы «USLUGA»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Kod_uslugi	Kod_uslugi	Не допустимо	PK	INTEGER
Naim_uslugi	Naim_uslugi	допустимо		TEXT
stoimost	stoimost	допустимо		DOUBLE PRECISION

Таблица 30 – Описание полей таблицы «SOSTAV_ZAYAVKINAPOSTAVKU»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_sostava	Nomer_sostava	Не допустимо	PK	INTEGER
Kol_tovara	Kol_tovara	допустимо		INTEGER
stoim	stoim	допустимо		DOUBLE PRECISION

Таблица 31 – Описание полей таблицы «ZAYAVKA_NA_POSTRAVKU»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_zayavki	Nomer_zayavki	Не допустимо	PK	INTEGER
Data_zayavki	Data_zayavki	допустимо		DATE
Summa	Summa	допустимо		DOUBLE PRECISION

Таблица 32 – Описание полей таблицы «TOVAR»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Kod_tovara	Kod_tovara	Не допустимо	PK	INTEGER
Naim_tovara	Naim_tovara	допустимо		TEXT
cena	cena	допустимо		DOUBLE PRECISION
kolichestvo	kolichestvo	допустимо		INTEGER

Таблица 33 – Описание полей таблицы «FIRMA_POSTAVSHIK»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Kod_firmipostavshika	Kod_firmipostavshika	Не допустимо	PK	INTEGER
Naim_firmipostavshika	Naim_firmipostavshika	допустимо		TEXT
rekviziti	rekviziti	допустимо		VARCHAR(50)

Таблица 34 – Описание полей таблицы «DOGOVOR_NA_POSTAVKU»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomerdogovora_na_postavku	Nomerdogovora_na_postavku	Не допустимо	PK	INTEGER
Data_sost_dogovora	Data_sost_dogovora	допустимо		DATA
Srok_postavki	Srok_postavki	допустимо		DATE

Таблица 35 – Описание полей таблицы «VID_TOVARA»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Kod_vida_tovara	Kod_vida_tovara	Не допустимо	PK	INTEGER
Vid_tovara	Vid_tovara	допустимо		VARCHAR (100)

Таблица 36 – Описание полей таблицы «SOSTAV_POSTAVKI»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_sostava	Nomer_sostava	Не допустимо	PK	INTEGER
Kol_tovara	Kol_tovara	допустимо		INTEGER
stoim	stoim	допустимо		DOUBLE PRECISION

Таблица 37 – Описание полей таблицы «POSTAVKA»

Имя атрибута	Физическое имя	Нулевое значение	Ключевое поле	Тип данных
Nomer_postavki	Nomer_postavki	Не допустимо	PK	INTEGER
Data_postavki	Data_postavki	допустимо		DATA
Nomer_sheta_fakturi	Nomer_sheta_fakturi	допустимо		INTEGER
summa	summa	допустимо		DOUBLE PRECISION

Таким образом, была разработана физическая модель данных для СУБД PostgreSQL.

4.3 Описание модулей программы

Программа реализована с помощью языка программирования C# в среде разработки Microsoft Visual Studio 2022 и имеет модульную структуру, представленную на рисунке.

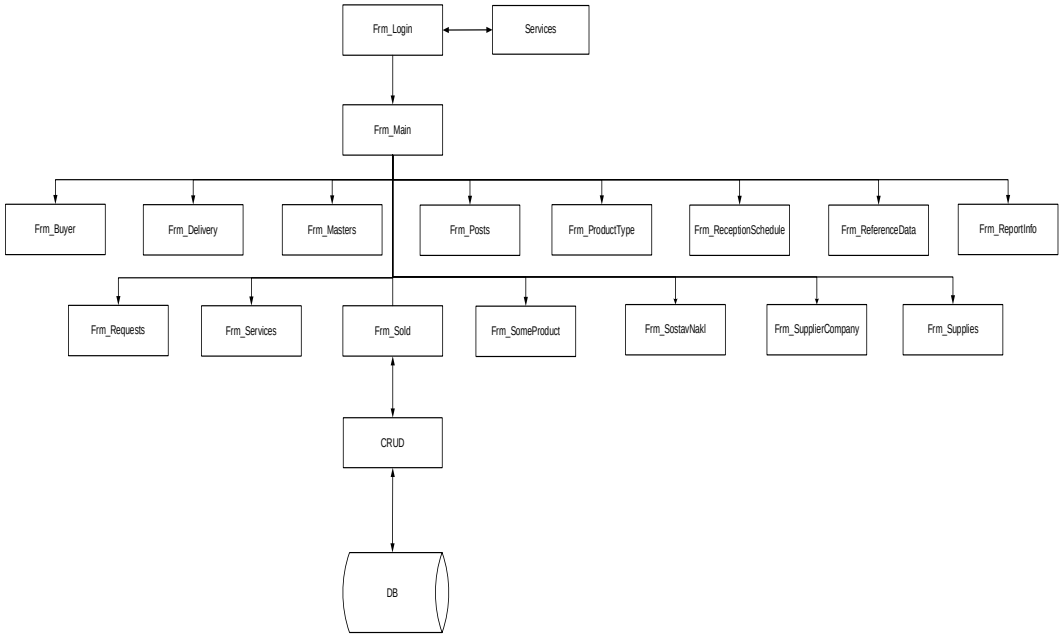


Рисунок 4 — Структура модулей программы

Описание модулей представлено в таблице 38.

Таблица 38 – Описание модулей программы

Название модуля	Описание модуля
Frm_Login	Модуль экрана авторизации, обеспечивающей разграничение доступных видов функциональности в зависимости от роли пользователя.
Services	Модуль обеспечения работы функции авторизации пользователей.
MainForm	Модуль главного экрана программы. Предназначен для обеспечения доступа ко всем функциям программы, обеспечивающий вывод информации о товарах. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о приме, а также выбора информации о товарах, продажах, записях, справочной информации и создании отчетов по поставкам, продажам, оказанным услугам, финансовой отчетности.

Название модуля	Описание модуля
Frm_Buyer	Модуль вывода справочной информации о клиентах. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о клиентах.
Frm_Delivery	Модуль обеспечивающий вывод информации о поставках товаров. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о поставках товаров.
Frm_Masters	Модуль вывода справочной информации о мастерах. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о мастерах.
Frm_Posts	Модуль вывода справочной информации о поставщиках. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о поставщиках.
Frm_ProductType	Модуль вывода справочной информации о материалах. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о материалах.
Frm_ReceptionSchedule	Модуль обеспечивающий вывод информации о приемах. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о приемах.
Frm_ReferenceData	Модуль вывода справочной информации о сотрудниках. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о сотрудниках.

Название модуля	Описание модуля
Frm_ReportInfo	Модуль, обеспечивающий формирование отчетов. Предназначен для обеспечения работы создания отчетности по поставкам, продажам, оказанным услугам и финансовой отчетности.
Frm_Requests	Модуль обеспечивающий вывод информации о заявках на поставку. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о заявках на поставку.
Frm_Services	Модуль вывода справочной информации о услугах. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о услугах.
Frm_Sold	Модуль обеспечивающий вывод информации о заказах. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о заказах.
Frm_SomeProduct	Модуль обеспечивающий вывод информации о товарных накладных. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о товарных накладных.
Frm_SostavNakl	Модуль обеспечивающий вывод информации о составе накладных. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о составе накладных.
Frm_SupplierCompany	Модуль обеспечивающий вывод информации о поставщиках. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о поставщиках.

Название модуля	Описание модуля
Frm_Supplies	Модуль обеспечивающий вывод информации о поставщиках. Предназначен для обеспечения работы функций добавления, изменения и удаления данных о поставщиках.
CRUD	Модуль обеспечивающий обмен данными с базой данных.
DB	Модуль базы данных

4.4 Методика испытаний программы

Функциональное тестирование предполагает проверку всех реализованных функций на предмет соответствия их техническому заданию.

После входа в систему открывается окно логирования (рисунок В.1). Для входа необходимо ввести логин и пароль в соответствующие поля формы (рисунок В.2). Для проверки корректности пароля, который был введен, необходимо активировать чекбокс «Показать пароль» (рисунок В.3). После этого необходимо нажать кнопку «Войти», далее откроется главное окно системы (рисунок В.4). Затем в поля «Наименование материла», «Цена материала», «Количество материала» вводим тестовые данные и нажимаем кнопку «Добавить», появляется окно успешного добавления данных (рисунок В.5), нажимаем кнопку «ОК». На главной форме отображается добавленный товар в поле формы «Данные» (рисунок В.6). Для изменения данных товара необходимо выбрать желаемый товар, ввести данных для изменения в выбранные поля, нажать кнопку «Изменить» (рисунок В.7) после этого появится окно успешного обновления записи (рисунок В.8), нажимаем кнопку «ОК». Изменения применены (рисунок В.9). Для удаления записи необходимо выбрать строку и нажать кнопку «Удалить», появится окно подтверждения удаления данных (рисунок В.10), нажать кнопку «Да», появится окно подтверждающее, что данные удалены (рисунок В.11), нажать кнопку «ОК».

Появляется главная форма, на которой отсутствует запись, которую удалили (рисунок В.12). Для поиска товара необходима ввести несколько символов, относящихся к желаемому товару и нажать кнопку «Поиск», в поле «Данные» отобразятся элементы, которые были выбраны на основе строки поиска (рисунок В.13). Для перехода на вкладку «Продажи» необходимо выбрать ее в поле меню в левой части главной формы. Для добавления новой записи необходимо выбрать ФИО покупателя и дату желаемого приема, нажать кнопку «Добавить» (рисунок В.14), в окне успешного добавления данных нажать кнопку «ОК» (рисунок В.15). Для изменения данных необходимо выбрать строку для изменения, внести изменения и нажать кнопку «Изменить» (рисунок В.16), в окне подтверждения кнопку «ОК» (рисунок В.17). Для удаления записи необходимо выбрать запись, нажать кнопку «Удалить», подтвердить удаление (рисунок В.18), нажать кнопку «Да», в информационном окне нажать кнопку «ОК» (рисунок В.19). Форма после удаления записи (рисунок В.20). Для поиска необходимо в строку «Поиск» ввести символы, по которым будет совершен поиск и нажать кнопку «Поиск» (рисунок В.21). Результат поиска (рисунок В.22). Далее выбрать пункт меню «Товарные накладные» (рисунок В.23). Для добавления записи необходимо заполнить поля «ФИО Сотрудника», «ФИО Покупателя», «Заказ», «Стоимость» (рисунок В.24) и нажать кнопку «Добавить», после чего появится новая запись (рисунок В.25). Для изменения данных необходимо выбрать желаемую запись и изменить данные, нажать кнопку «Изменить» (рисунок В.26). Измененные данные (рисунок В.27). Для удаления записи необходимо выбрать ее, нажать кнопку «Удалить», подтвердить удаление (рисунок В.28). Для поиска по записям необходимо ввести несколько символов, которые содержатся в желаемой записи и нажать кнопку «Поиск» (рисунок В.29). Для просмотра детальной информации необходимо нажать кнопку «Детальный вид», откроется форма детальной информации (рисунок В.30). Данную информацию также можно добавить (рисунок В.31), изменить (рисунок В.32), удалить (рисунок В.33), найти нужную запись через поиск (рисунок В.34). Для

выхода из окна детальной информации необходимо нажать пункт меню «Назад». Для просмотра информации о поставках, выбрать пункт меню «Поставки», откроется главное окно формы «Поставки» (рисунок В.35). Для добавления записи необходимо в поле «Фирма-поставщик» выбрать фирму и выбрать «Дату поставки» и «Срок поставки», нажать кнопку добавить (рисунок В.36). Для изменения данных необходимо выбрать строку и внести изменения в желаемые поля, нажать кнопку «Изменить» (рисунок В.37). Для удаления записи необходимо выбрать строку, нажать кнопку «Удалить», подтвердить удаление (рисунок В.38). Для поиска записи необходимо в строку поиска ввести символы, которые содержатся в искомой записи, после чего нажать кнопку «Поиск» (рисунок В.39). Для формирования заявки на поставку необходимо перейти в пункт меню «Договоры на поставку» (рисунок В.40). ввести данные в поля «Сумма заявки», «Код фирмы-поставщика», «Заявки», «ФИО сотрудника» (рисунок В.41). Для изменения данных необходимо выбрать заявку, внести изменения и нажать кнопку «Изменить» (рисунок В.42). Для удаления записи необходимо выбрать ее, нажать кнопку «Удалить», подтвердить удаление (рисунок В.43). Для поиска, ввести в строку поиска символы, которые относятся к искомой записи, нажать кнопку «Поиск» (рисунок В.44). Перейти в пункт меню «Поставка товаров» (рисунок В.45). Для добавления записи необходимо ввести данные в поля «Сума заявки», «Номер счета-фактуры», «Заявки», «ФИО Сотрудника» и нажать кнопку «Добавить» (рисунок В.46). Для изменения данных необходимо выбрать запись, внести изменения и нажать кнопку «Изменить» (рисунок В.47). Для удаления записи необходимо выбрать ее, нажать кнопку «Удалить» и подтвердить удаление (рисунок В.48). Для поиска записи в поле поиска ввести символы, которые относятся к искомой записи, нажать кнопку «Поиск» (рисунок В.49). Выйти на главную форму, выбрать пункт меню «Записи» откроется форма «Договор на оказание услуг» (рисунок В.50). Для добавления записи необходимо ввести данные в поля «ФИО Сотрудника», «Заказ» нажать кнопку «Добавить» (рисунок В.51). Для изменения данных нужно выбрать

запись, внести изменения в желаемые поля, нажать кнопку «Изменить» (рисунок В.52). Для удаления записи необходимо выбрать запись, нажать кнопку «Удалить», подтвердить удаление (рисунок В.53). Для поиска, необходимо в строку поиска ввести символы, которые относятся к искомой записи, нажать кнопку «Поиск» (рисунок В.54). Выбрать пункт меню «График приема (рисунок В.55). Для добавления записи необходимо ввести данные в поля «ФИО Сотрудника», «Номер договора», «Дата приема», выбрать статус «В процессе» (рисунок В.56). для изменения записи необходимо выбрать запись, внести в желаемые поля изменения и нажать кнопку «Изменить» (рисунок В.57). Для удаления необходимо выбрать запись, нажать кнопку «Удалить» и подтвердить удаление (рисунок В.58). Для поиска необходимо выбрать запись, ввести в поисковую строку символы, которые относятся к искомой записи и нажать кнопку «Поиск» (рисунок В.59). Перейти на главную форму. Выбрать пункт меню «Справочные данные», откроется форма «Сотрудники» (рисунок В.60). Данные о сотрудниках можно добавлять (рисунок В.61), изменять (рисунок В.62), удалять (рисунок В.63), осуществлять поиск (рисунок В.64). Выбрать пункт меню «Клиенты» (рисунок В.65). Данные о клиентах можно добавлять (рисунок В.66), изменять (рисунок В.67), удалять (рисунок В.68), осуществлять поиск (рисунок В.69). Выбрать пункт меню «Мастер» (рисунок В.70). Данные о мастерах можно добавлять (рисунок В.71), изменять (рисунок В.72), удалять (рисунок В.73), осуществлять поиск (рисунок В.74). Выбрать пункт меню «Фирмы-поставщики» (рисунок В.75). Данные о фирмах-поставщиках можно добавлять (рисунок В.76), изменять (рисунок В.77), удалять (рисунок В.78), осуществлять поиск (рисунок В.79). Выбрать пункт меню «Услуги» (рисунок В.80). Данные об услугах можно добавлять (рисунок В.81), изменять (рисунок В.82), удалять (рисунок В.83), осуществлять поиск (рисунок В.84). Выбрать пункт меню «Виды товаров» (рисунок В.85). Данные о видах товаров можно добавлять (рисунок В.86), изменять (рисунок В.87), удалять (рисунок В.88), осуществлять поиск (рисунок В.89). Вернуться на главную форму и перейти в

пункт меню «Отчеты» (рисунок В.90). Выбрать «По поставкам» (рисунок В.91). Выбрать «По продажам» (рисунок В.92). Выбрать «Оказанным услугам» (рисунок В.93). Выбрать «Финансовый отчет» (рисунок В.94). Вернуться на главную форму и перейти в пункт меню «Данные салона» (рисунок В.95). Вернуться на главную форму и перейти в пункт меню «О программе» (рисунок В.96).

5 Оценка затрат на разработку и внедрение информационной-автоматизированной системы салона красоты

5.1 Организация производства разработки информационной-автоматизированной системы

Разработка информационной-автоматизированной системы осуществлена на базе собственной площади общей площадью 30 кв.м. Содержание помещения в месяц составляет 15000 рублей.

Для разработки системы приобретено оборудование (вычислительная техника), лицензионное программное обеспечение, дополнительные материалы.

Данные по потребности в оборудовании и лицензионном программном обеспечении представлены соответственно в таблицах 39 и 40.

Таблица 39 – Потребность в оборудовании

Оборудование	Источник	Количество	Ориентировочная цена за единицу, рублей	Потребляемая мощность, Ватт
Ноутбук INFINIX Inbook Y2 Plus 11TH XL29 71008301573, 15.6"	ООО «Ситилинк»	3	39990	65

Таблица 40 – Потребность в лицензионном программном обеспечении

Лицензионное программное обеспечение	Источник	Цена, рублей	Срок действия лицензии, дней
Microsoft Windows 10	Входит в комплект «Ноутбук Huawei MateBook D 14 NbD-WDI9, 14"»	-	бессрочная
Microsoft Visual Studio 2022	Интернет-магазин techspot: https://www.techspot.com/downloads/7493-visual-studio-2022.html	-	Бессрочная
Антивирус Kaspersky Plus 2023	Интернет-магазин Kaspersky URL: https://www.kaspersky.ru/	1990	365
СУБД PostgreSQL	Интернет-магазин Postgresql URL: https://www.postgresql.org/download/	-	бессрочная
OpenModelSphere3.2	Интернет-магазин Open Model Sphere URL: http://www.modelsphere.com/org/download.html	-	бессрочная

Перечень дополнительных материалов, источники их приобретения, объем, и ориентировочные цены приведены в таблице 41.

Таблица 41 – Потребность в дополнительных материалах

Материал	Источник	Количество, штук	Ориентировочная цена за единицу, рублей
Стол компьютерный МЕБЕЛЬ-М	ООО «Леруа Мерлен восток»	3	2800

Материал	Источник	Количество, штук	Ориентировочная цена за единицу, рублей
Simple 110x75x60 см ЛДСП цвет белый шагрень			
Настольная светодиодная лампа CH-360 10 Вт, диммер, цвет белый	ООО «Леруа Мерлен восток»	3	3900
Офисное кресло Ridberg Ch-695 ткань цвет серый	ООО «Леруа Мерлен восток»	3	4200
Канцтовары	Гипермаркет «Канцлер»	3	1000

Разработка информационной-автоматизированной системы салона красоты осуществлялась командой разработчиков, представленных в таблице 42.

Таблица 42 – Состав коллектива разработчиков

Специалист	Количество, человек	Описание занятости
Руководитель проекта	1	-управление проектом -набор команды исполнителей -управление ресурсами проекта -контроль соблюдения требований заказчика -приемка и внедрение проекта -анализ предметной области

Специалист	Количество, человек	Описание занятости
Программист	1	-создание базы данных -написание, отладка и первичное тестирование программного кода -разработка и утверждение программной документации -приемка и внедрение программного кода
Тестировщик	1	-тестирование серверной части -тестирование клиентской части -общее тестирование ифнормационной-автоматизированной системы

Данные по основной заработной плате разработчиков информационной-автоматизированной системы и их количественному составу приведены в таблице 43.

Таблица 43 – Сведения о заработной плате

Специалист	Количество, человек	Сумма заработной платы, рублей
Руководитель проекта	1	100000
Программист	1	80000
Тестировщик	1	50000

5.2 Расчет себестоимости программного обеспечения для информационной-автоматизированной системы салона красоты

Себестоимость разработки информационной-автоматизированной системы включает следующие расходы:

– на приобретение оборудования;

– на приобретение лицензионного программного обеспечения;

– на дополнительные материалы;

– на заработную плату;

– на обслуживание программы.

Затраты на оборудование в период разработки составят 119970 рублей и представлены в таблице 39.

Затраты на программное обеспечение в период разработки составят 1990 рублей и представлены в таблице 40.

Затраты на дополнительные материалы в период разработки составят 39600 рублей и представлены в таблице 41.

Затраты на заработную плату включают затраты на основную заработную плату разработчиков программного приложения и отчисления на страховые взносы.

С учетом выходных дней при 5-днейвной рабочей неделе количество рабочих дней в месяце в среднем составляет 21 день. На разработку всего проекта необходимо 3 месяца.

Таким образом, затраты на заработную плату разработчиков составят 230000 рублей в месяц, 690000 рублей за весь период создания проекта.

Отчисления на страховые взносы:

-обязательные отчисления по нормам, установленным законодательством государственного социального страхования;

– в Пенсионный фонд РФ;

– фонд обязательного медицинского страхования;

– фонд социального страхования РФ.

От затрат на оплату труда работников, включаемых в себестоимость продукции (программного обеспечения), по элементу «Затраты на оплату труда» (кроме тех видов оплаты, на которые страховые взносы не начисляются).

Отчисления на страховые взносы производится согласно «Налоговому кодексу Российской Федерации» от 14.07.2022 № 239-ФЗ [8].

С января 2023 года введена единая базовая ставка по страховым взносам. Действующие тарифы на страховые взносы работодателей установлены статьей 425 НК РФ и составляют 30%.

Так как компания осуществляет деятельность в области IT, то она может претендовать на льготную ставку (подп. 3 п. 1 ст. 427 НК РФ) равную 7,6%.

Таким образом, с выплат в пределах МРОТ (минимального размера оплаты труда) уплачиваю взносы – 30%, с суммы свыше МРОТ уплачиваются взносы – 7,6%. МРОТ на 2024 год составляет 19242 рубля.

Данные по суммам отчислений на страховые взносы приведены в таблице 44.

Таблица 44 – Отчисления на страховые взносы

Сотрудник	Заработная плата, рублей	Взнос по единой ставке, рублей	Сумма, подлежащая пониженной ставке по взносу, рублей	Взнос по пониженной ставке, рублей	Общая сумма взноса с сотрудника за месяц, рублей
Руководитель проекта	100000	5772,6	80758	6137,6	11910,2
Программист	80000	5772,6	60758	4617,6	10390,2
Тестировщик	50000	5772,6	30758	2337,6	8110,2

Затраты на заработную плату и отчисления на страховые взносы представлены в таблице 45.

Таблица 45 – Затраты на заработную плату

Сотрудник	Заработная плата, рублей	Общая сумма отчислений, рублей	Всего за 1 месяц, рублей	Всего за 3 месяца, рублей
Руководитель проекта	100000	11910,2	111910,2	335730,6
Программист	80000	10390,2	90390,2	271170,6
Тестировщик	50000	8110,2	58110,2	174330,6

Затраты на заработную плату команды проекта по созданию информационной-автоматизированной системы составят 781231,8 рублей из которых общая сумма отчислений на страховые взносы разработчиков составит 30410,6 рублей, а заработная плата составит 750821,2 рублей.

Затраты на обслуживание включают следующие статьи:

– амортизация оборудования, составляющая 20% от стоимости оборудования;

– обслуживание оборудования, составляющее 15% от амортизационных отчислений;

– затраты на электроэнергию, включающие затраты на электроэнергию персональных компьютеров, оргтехники и бытовых приборов;

– расходы на содержание помещения;

– накладные расходы.

Амортизация оборудования составляет 20% от стоимости оборудования. Было приобретено 3 ноутбука по 39990 рублей. Таким образом затраты на амортизацию оборудования в месяц составят:

$$\frac{(3 * 39990) * 0{,}2}{12} = 1999{,}5 \text{ руб}$$

Обслуживание оборудования составляет 15% от амортизационных отчислений.

Таким образом затраты на обслуживание оборудования составят:

$$1999{,}5 * 0{,}15 = 299{,}93 \quad \text{руб.}$$

Предположительно в день по 8 часов будут работать 3 ноутбука:

$$65 * 8 * 63 * 3 = 123 \quad \text{кВт.}$$

Таким образом, за 63 дня разработки программного приложения будет использовано 123 кВт электроэнергии на питание ноутбуков, а также 0,5 кВт на прочие приборы.

$$123 + 0{,}5 = 123{,}5 \text{ кВт.}$$

Накладные расходы составляют 75% от заработной платы. Таким образом, затраты на накладные расходы составляют:

$$750821{,}2 * 0{,}75 = 563115{,}9 \text{ руб.}$$

Результат расчетов по статьям затрат на обслуживание приведены в таблице 46.

Таблица 46 – Затраты на обслуживание

Статья затрат	Показатель отчислений	Сумма отчислений в месяц, рублей	Итоговая сумма отчислений за период разработки, рублей
Амортизация оборудования	20%	1999,5	5998,5
Обслуживание оборудования	15%	299,9	899,8
Затраты на электроэнергию	1,23 на 1 кВт/ч	367,8	1103,4
Накладные расходы	75%	563115,9	1689347,7
Аренда офиса и расходы на его содержание	-	15000	45000
Итого, рублей		580783,1	1742349,4

Себестоимость разработки определяется как сумма затрат, включающая все статьи затрат на реализацию проекта, представленное в таблице 47.

Таблица 47 – Общая сумма затрат

Статья затрат	Сумма отчислений, рублей
Приобретение оборудования	119970
Приобретение лицензионного программного обеспечения	1990
Затраты на заработную плату	781231,8
Статья затрат	Сумма отчислений, рублей
Затраты на дополнительные материалы	39600
Затраты на обслуживание	1742349,4
Итого:	2685141,2

Таким образом, сумма затрат на разработку проекта составит 2685141,2 рублей и будет соответствовать себестоимости информационной-автоматизированной системы салона красоты.

5.3 Оценка эффективности информационной - автоматизированной системы салона красоты

Необходимость и целесообразность реализации инвестиционных проектов осуществляется путем оценки их экономической эффективности.

Экономическая эффективность инвестиционных проектов характеризуется системой показателей, отражающих соотношение результатов (притока, поступлений, дохода) и затрат (оттока, выплат) от операционной, инвестиционной и финансовой деятельности.

За счет внедрения и использования информационной-автоматизированной системы салона красоты также ожидаются следующие неисчисляемые эффекты:

– увеличение скорости записи клиентов;

– минимизация ошибок, вызванных человеческим фактором;

– повышение эффективности работы мастеров;

– упрощение формирования отчетности разного вида;

– снижение нагрузки на сотрудников.

5.4 Анализ рисков при разработке и внедрении информационной-автоматизированной системы салона красоты

Риски взаимодействия между заказчиком и исполнителем – это риски, связанные с отсутствием коммуникации между исполнителем и заказчиком или их представителями.

Риски планирования – это риски, связанные с отсутствием навыков планирования сроков выполняемых работ, как менеджером, так и исполнителями. Часто слишком мало времени закладывается на этап тестирования и исправления ошибок в исходном коде, предпроектного обследования. Для минимизации риска, проектный руководитель следил за четким выполнением сроков на каждом этапе проекта.

Недостаточное обсуждение задач или архитектуры может негативно сказаться на разрабатываемой информационной-автоматизированной системе. Для уменьшения последствий от таких рисков на начальном этапе проекта разработано соглашение и обозначена в договоре частота проведения встреч по проекту, куратора проекта со стороны заказчика и исполнителя, а также состав рабочей группы.

Функциональные риски – это риски, связанные с неверным выполнением функций, возложенных на систему. Для минимизации данных рисков необходим тщательный анализ требований заказчика, разработка моделей и привлечение к работе грамотных специалистов.

Риск смены сотрудников возникает, когда проект покидают ключевые сотрудники, которые владеют максимально полной информацией о проекте. В

некоторых случаях уход данных сотрудников может полностью остановить разработку проекта. Для того чтобы минимизировать последствия от ухода ключевых сотрудников проекта, руководитель должен поддерживать высокий уровень сотрудничества и обмена информацией в коллективе, планировать работу сотрудников так, чтобы они могли заменить друг друга, а также выстраивает систему внутреннего обучения.

Риски хищения исходного кода возникают, если при уходе разработчики забирают созданный ими проект.

Дополнительной мотивацией для всех сотрудников оставаться на своем рабочем месте является премирование, которое может быть материальным и нематериальным.

Риск привлечения к ответственности, в случае использования компанией нелицензионного ПО, это может удешевить себестоимость проекта, но и может понести последствия. Для минимизации данного риска, каждая компания должна закупать только лицензионное программное обеспечение.

При предпроектном анализе в обязательном порядке должны быть раскрыты все риски, а также пути их минимизации. В конце должен быть составлен обобщающий отчет по всем рискам.

6 Общие принципы обеспечения безопасности на рабочем месте оператора информационной-автоматизированной системы салона красоты

6.1 Актуальность вопросов безопасности человека на рабочем месте

С каждым годом человек все больше исследует и использует IT технологии в своей жизни. Это стало возможно благодаря распространению ПЭВМ (персональных электронных вычислительных машин). Они стали употребляться почти на каждом рабочем месте, именно поэтому необходимо помнить про безопасность оператора ПЭВМ, сохранении его здоровья и работоспособности.

Использование ЭВМ связано с распознаванием печатного и рукописного текста, анализ изображений на экране, долгим нахождением в одной позе, и др., что может пагубно сказаться на здоровье человека, если не придерживаться нормами использования ЭВМ.

На основании ГОСТ 12.0.003-74[8] вредные производственные факторы подразделяются на 4 категории:

– физические, к ним относятся недостаточная освещенность рабочего места, повышенная яркость света, отклонения от нормы температуры воздуха, высокая влажность воздуха, повышенный уровень шума, и др.;

– психофизиологические, к ним относятся нервно-психические и физические перегрузки, перенапряжение ушей и глаз;

– биологические, к ним относятся микроорганизмы и макроорганизмы, которые могут вызвать различные заболевания;

– химические, к ним относятся превышения в рабочей зоне аммиака, фенола, озона, оксида углерода и др.

6.2 Техника безопасности при работе с электроприборами

Персональная электронно-вычислительная машина – это электроприбор, следовательно, работа с ним требует знаний правил техники безопасности:

– при работе с электрооборудованием следует использовать заземленные розетки;

– после выключения электроприборов из сети, нельзя тянуть их за шнур и находиться рядом с мокрыми руками;

– неисправность у прибора нельзя чинить до момента отключения от сети;

– до включения прибора в сеть необходимо проверить сохранность шнура (наличие заломов и повреждения изоляции);

– не оставлять включенные приборы без присмотра.

В состав ПЭВМ входят монитор, блок питания, процессор – все они создают электромагнитные поля. Такие поля создают еще распределительные щиты, электропроводка, линии электропередач.

6.3 Влияние ПЭВМ на физиологию человека

Работа за компьютером подразумевает длительное напряжение на зрительные органы и опорно-двигательный аппарат. Если не контролировать время пребывания за ЭВМ, то это может привести к нарушению зрения, искривлению позвоночника, а также нарушению кровообращения в организме.

Нагрузка на зрительный аппарат может вызвать нарушения в работе ЦНС (центральной нервной системы). При длительной работе за компьютером снижается фокусировка и острота зрения, увеличивается чувствительность глаз, они могут слезиться и краснеть.

Длительное нахождения человека в одной позе приводит к быстрому развитию утомляемости.

При работе за компьютером стоит обратить внимание на:

– высоту рабочей зоны и стула;

– наличие опорной спинки и подлокотников;

– угол наклона головы при работе;

– угол наклона экрана компьютера.

Огромное влияние на здоровье человека оказывает уровень шума в помещении. Согласно СанПиН 2.2.3670-20 "Санитарно-эпидемиологические требования к условиям труда" [9], эквивалентный уровень шума не должен превышать 50 дБА. Если шум свыше нормы будет длительно присутствовать на рабочем месте сотрудника, то будут ухудшены его условия труда и нанесен вред здоровью. А именно:

– снижается концентрация внимания;

– повышается усталость;

– снижается работоспособность сотрудника.

Для того, чтобы снизить уровень шума в помещении используется шумоизоляция.

6.4 Техника безопасности при работе с персональным компьютером

На основе типовой инструкции по охране труда при работе на ПК (персональном компьютере) ТОИ Р-45-084-01 [10], которая была утверждена Приказом Минсвязи Российской Федерации №162 от 02.07.2001, деятельность на ЭВМ делится на 3 группы:

– группа А – чтение информации с монитора подразумевает два перерыва по 15 минут, без учета обеденного перерыва;

– группа Б – набор текста подразумевает перерывы по 10 минут раз в час;

– группа В – творческий труд подразумевает перерывы по 15 минут раз в час.

Выделенное время рекомендуется уделять физической нагрузке, такой как зарядка или гимнастика для глаз.

На основании всего вышесказанного, можно сделать вывод, что для сохранения здоровья сотрудников и ограничения от негативного влияния ПЭВМ необходимо использовать только сертифицированную и качественную технику. Обеспечивать сотрудникам рациональный режим труда и перерывов.

Необходимо отметить, что сертифицированная техника не является полной гарантией безопасности условий труда, необходимо регулярно проводить проверку всей используемой техники.

Заключение

В ходе выполнения дипломного проекта была изучена предметная область, а именно работа сотрудников салона красоты. На основе ее анализа было сформулировано техническое задание для разработки информационной-автоматизированной системы салона красоты. Далее были выбраны средства разработки, с помощью которых была разработана информационная автоматизированная система.

Информационная автоматизированная система салона красоты реализована в соответствии с техническим заданием.

В разделе экономики была произведена оценка затрат на разработку и внедрение приложения.

В разделе безопасность жизнедеятельности были проанализированы и описаны общие принципы обеспечения безопасности на рабочем месте оператора программного приложения контроля выполнения задач.

Задание на дипломное проектирование выполнено в полном объеме.

Список использованных источников

1. Карпова И.П. – «Базы данных. Курс лекций и материалы для практических заданий». – Учебное пособие. – 2011. – 240с.
2. Норенков И.П. – «Автоматизированные-информационные системы». – 2010. – 342с.
3. Статистика по салонам красоты [Электронный ресурс] – URL - https://rg.ru/2023/03/20/dela-idut-krasivo.html?ysclid=lx1jo1wz7q208872500 (дата обращения: 08.02.2024).
4. Котов О.М. – «Язык C#: краткое описание и введение в технологии программирования». – 2014. – 208с.
5. Пьянова Е.А., Антонов Е.В., Климов О.А., Гурин И.А. – «Обзор новых возможностей Visual Studio 2022». – 2022. – 3с.
6. Еременко А.В. – «Проектирование баз данных с помощью OpenModelSphere». – 2011. – 73с.
7. Лузанов П., Рогов Е., Лёвшин И. – «POSTGRES первое знакомство». – 2023. – 178с.
8. ГОСТ 12.0.0003-74 ССБТ. Библиографическая запись. Опасные и вредные производственные факторы. Классификация (с Изменением № 1). Изд-во стандартов, 2002. – 5 с.
9. СанПиН 2.2.3670-20. Санитарно-эпидемиологические требования к условиям труда. Изд-во стандартов, 2012. – 47с.
10. Типовая инструкция по охране труда при работе на персональном компьютере. – 2001. – 3 с.

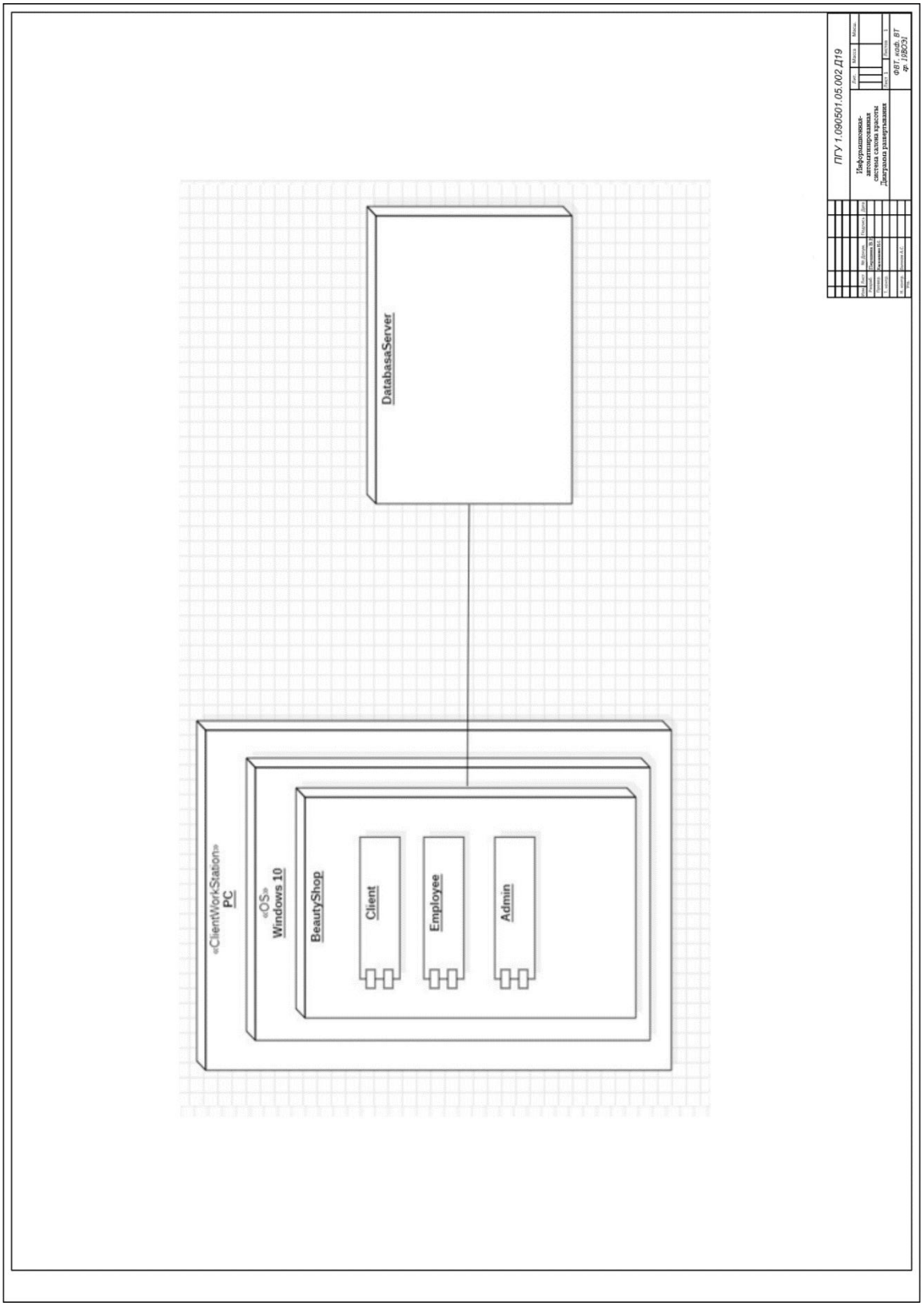
Приложение А

(обязательное)

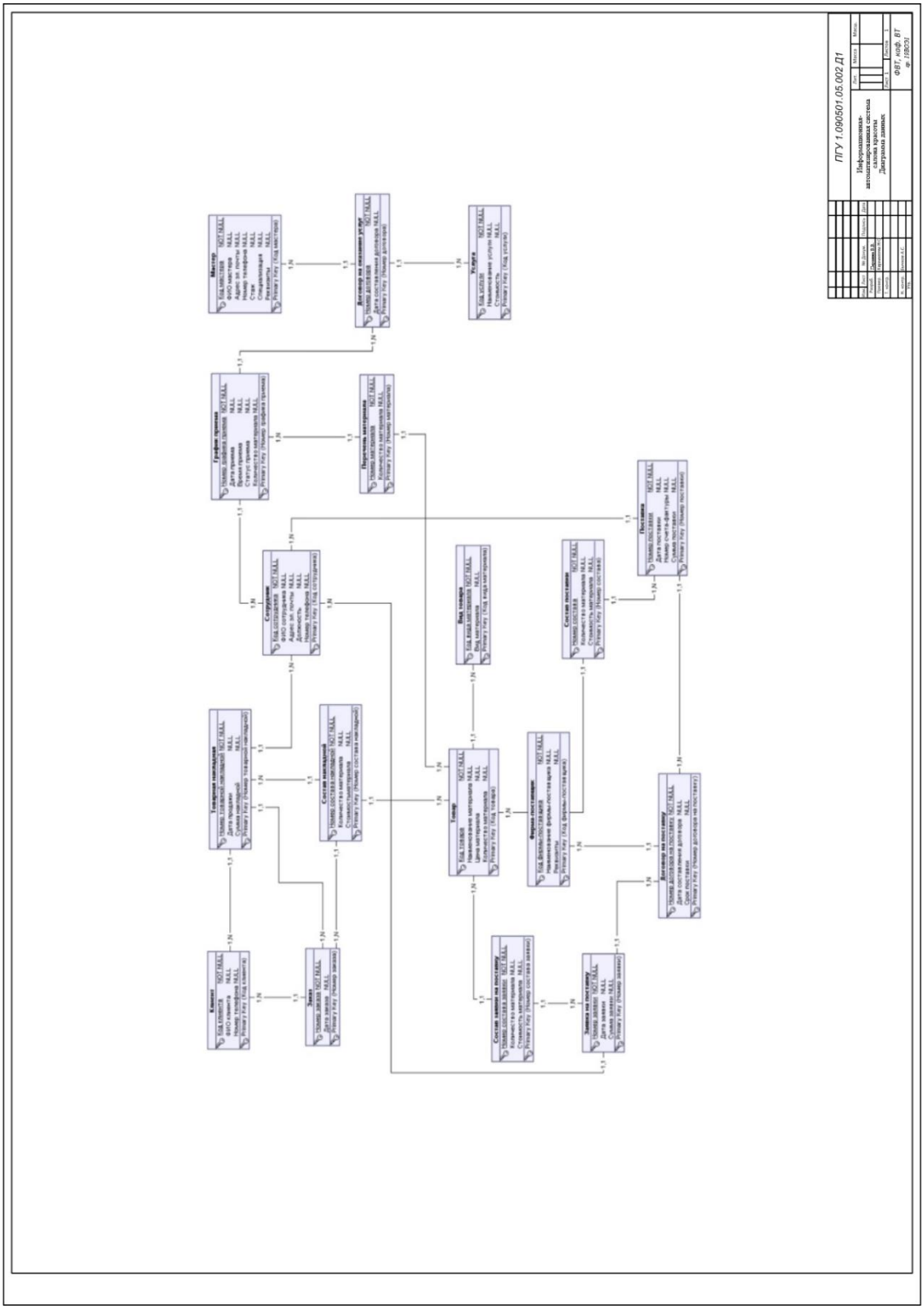
Информационная автоматизированная система салона красоты

UML-диаграммы

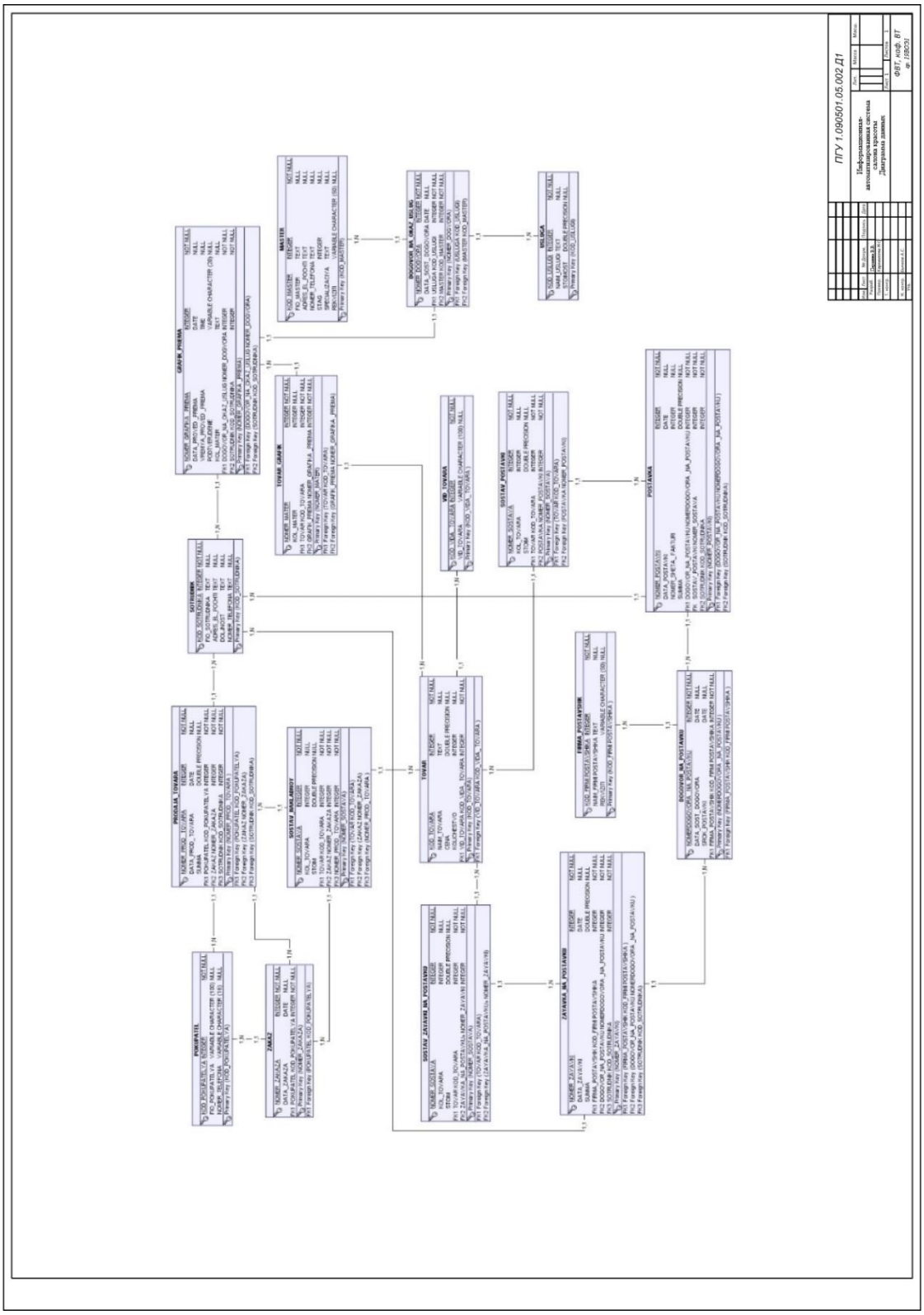
Приложение А.1 – Диаграмма развертывания



Приложение А.2 – Логическая модель данных



Приложение А.3 – Физическая модель данных



Приложение Б

(обязательное)

Информационная автоматизированная система салона красоты

SQL-скрипт

```
-- Table: public.DOGOVOR_NA_OKAZ_USLUG


-- DROP TABLE IF EXISTS public."DOGOVOR_NA_OKAZ_USLUG";


CREATE TABLE IF NOT EXISTS public."DOGOVOR_NA_OKAZ_USLUG"
(
    "NOMER_DOGVORA" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    "DATA_SOST_DOGOVORA" date,
    "USLUGA_ID" integer NOT NULL,
    "MASTER_ID" integer NOT NULL,
    CONSTRAINT "DOGOVOR_NA_OKAZ_USLUG_pkey" PRIMARY KEY
("NOMER_DOGVORA"),
    CONSTRAINT "MASTER_FK" FOREIGN KEY ("MASTER_ID")
        REFERENCES public."MASTER" ("KOD_MASTER") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "USLUGA_FK" FOREIGN KEY ("USLUGA_ID")
        REFERENCES public."USLUGA" ("KOD_USLUGI") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."DOGOVOR_NA_OKAZ_USLUG"
    OWNER to postgres;
```

```
-- Table: public.DOGOVOR_NA_POSTAVKU


-- DROP TABLE IF EXISTS public."DOGOVOR_NA_POSTAVKU";


CREATE TABLE IF NOT EXISTS public."DOGOVOR_NA_POSTAVKU"

(

    "NOMERDOGOVORA_NA_POSTAVKU" integer NOT NULL GENERATED ALWAYS
AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647
CACHE 1 ),

    "DATA_SOST_DOGOVORA " date,

    "SROK_POSTAVKI" date,

    "FIRMA_ID" integer NOT NULL,

    CONSTRAINT "DOGOVOR_NA_POSTAVKU_pkey" PRIMARY KEY
("NOMERDOGOVORA_NA_POSTAVKU"),

    CONSTRAINT "FIRMA_FK" FOREIGN KEY ("FIRMA_ID")

        REFERENCES public."FIRMA_POSTAVSHIK"
("KOD_FIRMI_POSTAVSHIKA") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."DOGOVOR_NA_POSTAVKU"

    OWNER to postgres;
```

```
-- Table: public.FIRMA_POSTAVSHIK

-- DROP TABLE IF EXISTS public."FIRMA_POSTAVSHIK";

CREATE TABLE IF NOT EXISTS public."FIRMA_POSTAVSHIK"
(
    "KOD_FIRMI_POSTAVSHIKA" integer NOT NULL GENERATED ALWAYS AS
IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647
CACHE 1 ),
    "NAIM_FIRMI_POSTAVSHIKA" text COLLATE pg_catalog."default",
    "REKVIZITI" character varying(50) COLLATE
pg_catalog."default",
    CONSTRAINT "FIRMA_POSTAVSHIK_pkey" PRIMARY KEY
("KOD_FIRMI_POSTAVSHIKA")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."FIRMA_POSTAVSHIK"
    OWNER to postgres;

-- Table: public.GRAFIK_PRIEMA

-- DROP TABLE IF EXISTS public."GRAFIK_PRIEMA";

CREATE TABLE IF NOT EXISTS public."GRAFIK_PRIEMA"
(
```

```
    "NOMER_GRAFIKA_PRIEMA" integer NOT NULL GENERATED ALWAYS AS
IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647
CACHE 1 ),

    "DATA_PROVED_PRIEMA" date,

    "VREMYA_PROVED_PRIEMA" time with time zone,

    "PODTVERJDENIE" character varying(20) COLLATE
pg_catalog."default",

    "DOGOVOR_NA_OKAZANIE_ID" integer NOT NULL,

    "SOTRUDNIK_ID" integer NOT NULL,

    CONSTRAINT "GRAFIK_PRIEMA_pkey" PRIMARY KEY
("NOMER_GRAFIKA_PRIEMA"),

    CONSTRAINT "DOGOVOR_NA_OKAZ_FK" FOREIGN KEY
("DOGOVOR_NA_OKAZANIE_ID")

        REFERENCES public."DOGOVOR_NA_OKAZ_USLUG"
("NOMER_DOGVORA") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "SOTRUDNIK_FK" FOREIGN KEY ("SOTRUDNIK_ID")

        REFERENCES public."SOTRUDNIK" ("KOD_SOTRUDNIKA") MATCH
SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."GRAFIK_PRIEMA"

    OWNER to postgres;
```

```
-- Table: public.MASTER

-- DROP TABLE IF EXISTS public."MASTER";

CREATE TABLE IF NOT EXISTS public."MASTER"
(
    "KOD_MASTER" integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    "FIO_MASTER" text COLLATE pg_catalog."default",
    "ADRES_EL_POCHTI" text COLLATE pg_catalog."default",
    "NOMER_TELEFONA" text COLLATE pg_catalog."default",
    "STAG" integer,
    "SPECIALIZACIYA" text COLLATE pg_catalog."default",
    "REKVIZITI" character varying(50) COLLATE pg_catalog."default",
    CONSTRAINT "MASTER_pkey" PRIMARY KEY ("KOD_MASTER")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."MASTER"
    OWNER to postgres;



-- Table: public.POKUPATEL

-- DROP TABLE IF EXISTS public."POKUPATEL";
```

```
CREATE TABLE IF NOT EXISTS public."POKUPATEL"

(

    "KOD_POKUPATEL" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    "FIO_POKUPATEL" character varying(100) COLLATE
pg_catalog."default",

    "NOMER_TELEFONA" character varying(16) COLLATE
pg_catalog."default",

    CONSTRAINT "POKUPATEL_pkey" PRIMARY KEY ("KOD_POKUPATEL")

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."POKUPATEL"

    OWNER to postgres;



-- Table: public.POSTAVKA


-- DROP TABLE IF EXISTS public."POSTAVKA";


CREATE TABLE IF NOT EXISTS public."POSTAVKA"

(

    "NOMER_POSTAVKI" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    "DATA_POSTAVKI" date,

    "NOMER_SHETA_FAKTURI" integer,

    "SUMMA" double precision,

    "DOGOVOR_NA_POSTAVKU_ID" integer NOT NULL,
```

```
    "SOTRUDNIK_ID" integer NOT NULL,

    CONSTRAINT "POSTAVKA_pkey" PRIMARY KEY ("NOMER_POSTAVKI"),

    CONSTRAINT "NOMER_POST_FK" FOREIGN KEY
("DOGOVOR_NA_POSTAVKU_ID")

        REFERENCES public."DOGOVOR_NA_POSTAVKU"
("NOMERDOGOVORA_NA_POSTAVKU") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "SOTRUDNIK_FK" FOREIGN KEY ("SOTRUDNIK_ID")

        REFERENCES public."SOTRUDNIK" ("KOD_SOTRUDNIKA") MATCH
SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."POSTAVKA"

    OWNER to postgres;



-- Table: public.PRODAJA_TOVARA


-- DROP TABLE IF EXISTS public."PRODAJA_TOVARA";


CREATE TABLE IF NOT EXISTS public."PRODAJA_TOVARA"

(
```

```
    "NOMER_PROD_TOVARA" integer NOT NULL GENERATED ALWAYS AS
IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647
CACHE 1 ),

    "DATA_PROD_TOVARA" date,

    "SUMMA" double precision,

    "POKUPATEL_ID" integer NOT NULL,

    "ZAKAZ_ID" integer NOT NULL,

    "SOTRUDNIK_ID" integer NOT NULL,

    CONSTRAINT "PRODAJA_TOVARA_pkey" PRIMARY KEY
("NOMER_PROD_TOVARA")

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."PRODAJA_TOVARA"

    OWNER to postgres;




-- Table: public.SOSTAV_NAKLADNOY

-- DROP TABLE IF EXISTS public."SOSTAV_NAKLADNOY";

CREATE TABLE IF NOT EXISTS public."SOSTAV_NAKLADNOY"

(

    "NOMER_SOSTAVA" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    "KOL_TOVARA" integer,

    "STOIM" double precision,

    "TOVAR_ID" integer NOT NULL,

    "ZAKAZ_ID" integer NOT NULL,

    "NOMER_PROD_ID" integer NOT NULL,
```

```
    CONSTRAINT "SOSTAV_NAKLADNOY_pkey" PRIMARY KEY
("NOMER_SOSTAVA"),

    CONSTRAINT "PRODAJA_TOVARA_FK" FOREIGN KEY ("NOMER_PROD_ID")

        REFERENCES public."PRODAJA_TOVARA" ("NOMER_PROD_TOVARA")
MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "TOVAR_FK" FOREIGN KEY ("TOVAR_ID")

        REFERENCES public."TOVAR" ("KOD_TOVARA") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "ZAKAZ_FK" FOREIGN KEY ("ZAKAZ_ID")

        REFERENCES public."ZAKAZ" ("NOMER_ZAKAZA") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)

TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."SOSTAV_NAKLADNOY"

    OWNER to postgres;


-- Table: public.SOSTAV_POSTAVKI


-- DROP TABLE IF EXISTS public."SOSTAV_POSTAVKI";


CREATE TABLE IF NOT EXISTS public."SOSTAV_POSTAVKI"
```

```
(
    "NOMER_SOSTAVA" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    "KOL_TOVARA" integer,
    "STOIM" double precision,
    "TOVAR_ID" integer NOT NULL,
    "POSTAVKA_ID" integer NOT NULL,
    CONSTRAINT "SOSTAV_POSTAVKI_pkey" PRIMARY KEY
("NOMER_SOSTAVA"),
    CONSTRAINT "POSTAVKA_FK" FOREIGN KEY ("POSTAVKA_ID")
        REFERENCES public."POSTAVKA" ("NOMER_POSTAVKI") MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "TOVAR_FK" FOREIGN KEY ("TOVAR_ID")
        REFERENCES public."TOVAR" ("KOD_TOVARA") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."SOSTAV_POSTAVKI"
    OWNER to postgres;



-- Table: public.SOSTAV_ZAYAVKI_NA_POSTAVKU
```

```
-- DROP TABLE IF EXISTS public."SOSTAV_ZAYAVKI_NA_POSTAVKU";


CREATE TABLE IF NOT EXISTS public."SOSTAV_ZAYAVKI_NA_POSTAVKU"

(

    "NOMER_SOSTAVA" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    "KOL_TOVARA" integer,

    "STOIM" double precision,

    "TOVAR_ID" integer NOT NULL,

    "ZAYAVKA_NA_POSTAVKU_ID" integer NOT NULL,

    CONSTRAINT "SOSTAV_ZAYAVKI_NA_POSTAVKU_pkey" PRIMARY KEY
("NOMER_SOSTAVA"),

    CONSTRAINT "TOVAR_FK" FOREIGN KEY ("TOVAR_ID")

        REFERENCES public."TOVAR" ("KOD_TOVARA") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "ZAKAZ_NA_POST_FK" FOREIGN KEY
("ZAYAVKA_NA_POSTAVKU_ID")

        REFERENCES public."ZAYAVKA_NA_POSTAVKU" ("NOMER_ZAYAVKI")
MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."SOSTAV_ZAYAVKI_NA_POSTAVKU"
```

```
    OWNER to postgres;



-- Table: public.SOTRUDNIK

-- DROP TABLE IF EXISTS public."SOTRUDNIK";

CREATE TABLE IF NOT EXISTS public."SOTRUDNIK"
(
    "KOD_SOTRUDNIKA" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    "FIO_SOTRUDNIKA" text COLLATE pg_catalog."default",
    "ADRES_EL_POCHTI" text COLLATE pg_catalog."default",
    "DOLJNOST" text COLLATE pg_catalog."default",
    "NOMER_TELEFONA" text COLLATE pg_catalog."default",
    CONSTRAINT "SOTRUDNIK_pkey" PRIMARY KEY ("KOD_SOTRUDNIKA")
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."SOTRUDNIK"
    OWNER to postgres;



-- Table: public.TOVAR

-- DROP TABLE IF EXISTS public."TOVAR";
```

```
CREATE TABLE IF NOT EXISTS public."TOVAR"
(
    "KOD_TOVARA" integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    "NAIM_TOVARA" text COLLATE pg_catalog."default",
    "CENA" double precision,
    "KOLICHESTVO" integer,
    "VID_TOVARA_ID" integer NOT NULL,
    CONSTRAINT "TOVAR_pkey" PRIMARY KEY ("KOD_TOVARA"),
    CONSTRAINT "VID_TOVARA_FK" FOREIGN KEY ("VID_TOVARA_ID")
        REFERENCES public."VID_TOVARA" ("KOD_VIDA_TOVARA") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)

TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."TOVAR"
    OWNER to postgres;



-- Table: public.TOVAR_GRAFIK

-- DROP TABLE IF EXISTS public."TOVAR_GRAFIK";
```

```
CREATE TABLE IF NOT EXISTS public."TOVAR_GRAFIK"

(

    "NOMER_MATER" integer NOT NULL GENERATED ALWAYS AS IDENTITY (
INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    "KOL_MATER" integer,

    "TOVAR_ID" integer NOT NULL,

    "GRAFIK_PRIEMA_ID" integer NOT NULL,

    CONSTRAINT "TOVAR_GRAFIK_pkey" PRIMARY KEY ("NOMER_MATER"),

    CONSTRAINT "GRAFIK_PRIEMA_FK" FOREIGN KEY ("GRAFIK_PRIEMA_ID")

        REFERENCES public."GRAFIK_PRIEMA" ("NOMER_GRAFIKA_PRIEMA")
MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "TOVAR_FK" FOREIGN KEY ("TOVAR_ID")

        REFERENCES public."TOVAR" ("KOD_TOVARA") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."TOVAR_GRAFIK"

    OWNER to postgres;



-- Table: public.USLUGA
```

```
-- DROP TABLE IF EXISTS public."USLUGA";


CREATE TABLE IF NOT EXISTS public."USLUGA"

(

    "KOD_USLUGI" integer NOT NULL GENERATED ALWAYS AS IDENTITY (
INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    "STOIMOST" double precision,

    "NAIM_USLUGI" text COLLATE pg_catalog."default",

    CONSTRAINT "USLUGA_pkey" PRIMARY KEY ("KOD_USLUGI")

)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."USLUGA"

    OWNER to postgres;


-- Table: public.VID_TOVARA


-- DROP TABLE IF EXISTS public."VID_TOVARA";


CREATE TABLE IF NOT EXISTS public."VID_TOVARA"

(

    "KOD_VIDA_TOVARA" integer NOT NULL GENERATED ALWAYS AS
IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647
CACHE 1 ),

    "VID_TOVARA" character varying(100) COLLATE
pg_catalog."default",

    CONSTRAINT "VID_TOVARA_pkey" PRIMARY KEY ("KOD_VIDA_TOVARA")

)
```

```
TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."VID_TOVARA"
    OWNER to postgres;



-- Table: public.ZAKAZ


-- DROP TABLE IF EXISTS public."ZAKAZ";


CREATE TABLE IF NOT EXISTS public."ZAKAZ"
(
    "NOMER_ZAKAZA" integer NOT NULL GENERATED ALWAYS AS IDENTITY ( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),
    "DATA_ZAKAZA" date,
    "POKUPATEL_ID" integer NOT NULL,
    CONSTRAINT "ZAKAZ_pkey" PRIMARY KEY ("NOMER_ZAKAZA"),
    CONSTRAINT "POKUPATEL_FK" FOREIGN KEY ("POKUPATEL_ID")
        REFERENCES public."POKUPATEL" ("KOD_POKUPATEL") MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)


TABLESPACE pg_default;


ALTER TABLE IF EXISTS public."ZAKAZ"
    OWNER to postgres;
```

```
-- Table: public.ZAYAVKA_NA_POSTAVKU

-- DROP TABLE IF EXISTS public."ZAYAVKA_NA_POSTAVKU";

CREATE TABLE IF NOT EXISTS public."ZAYAVKA_NA_POSTAVKU"

(

    "NOMER_ZAYAVKI" integer NOT NULL GENERATED ALWAYS AS IDENTITY
( INCREMENT 1 START 1 MINVALUE 1 MAXVALUE 2147483647 CACHE 1 ),

    "DATA_ZAYAVKI" date,

    "SUMMA" double precision,

    "KOD_FIRMI_POSTAVSHIKA_ID" integer NOT NULL,

    "NOMERDOGOVORA_ID" integer NOT NULL,

    "SOTRUDNIK_ID" integer NOT NULL,

    CONSTRAINT "ZAYAVKA_NA_POSTAVKU_pkey" PRIMARY KEY
("NOMER_ZAYAVKI"),

    CONSTRAINT "FIRMA_FK" FOREIGN KEY ("KOD_FIRMI_POSTAVSHIKA_ID")

        REFERENCES public."FIRMA_POSTAVSHIK"
("KOD_FIRMI_POSTAVSHIKA") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "NOMER_FK" FOREIGN KEY ("NOMERDOGOVORA_ID")

        REFERENCES public."DOGOVOR_NA_POSTAVKU"
("NOMERDOGOVORA_NA_POSTAVKU") MATCH SIMPLE

        ON UPDATE NO ACTION

        ON DELETE NO ACTION

        NOT VALID,

    CONSTRAINT "SOTRUDNIK_FK" FOREIGN KEY ("SOTRUDNIK_ID")

        REFERENCES public."SOTRUDNIK" ("KOD_SOTRUDNIKA") MATCH
SIMPLE
```

```
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
TABLESPACE pg_default;

ALTER TABLE IF EXISTS public."ZAYAVKA_NA_POSTAVKU"
    OWNER to postgres;
```

Приложение В

(обязательное)

Информационная автоматизированная система салона красоты

Результаты испытаний

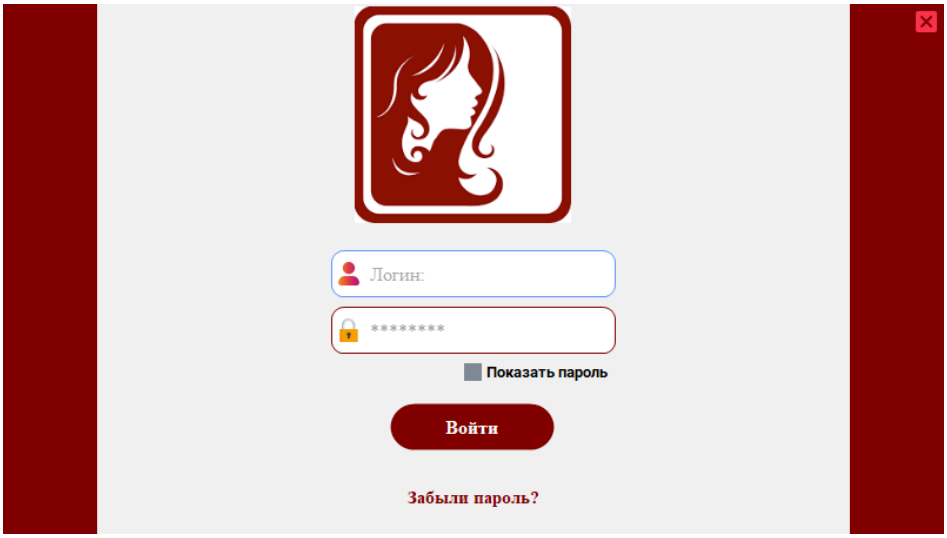


Рисунок В.1 – Окно логирования

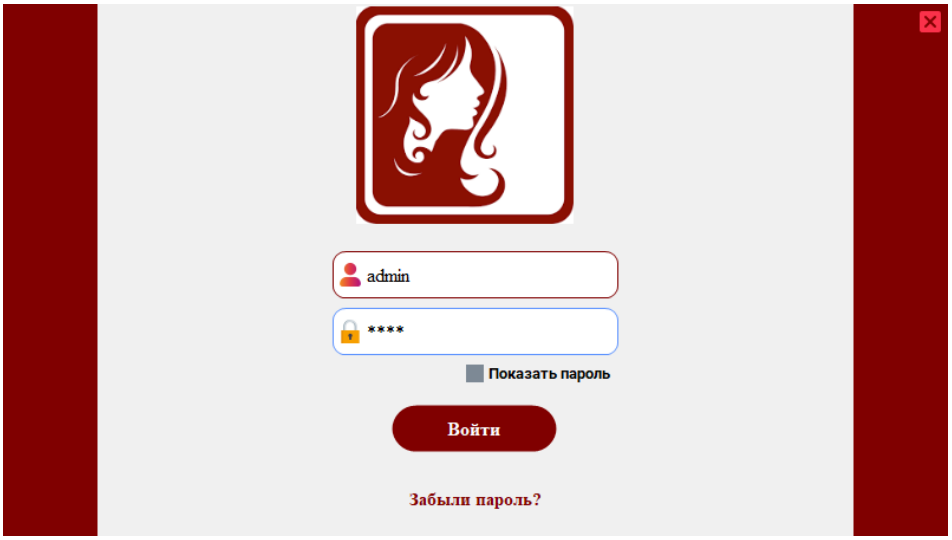


Рисунок В.2 – Ввод логина и пароля

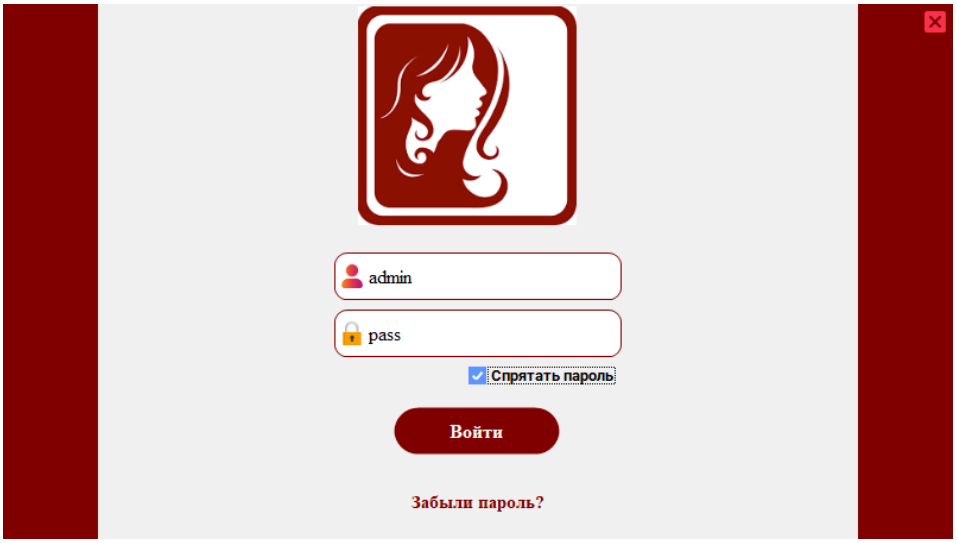


Рисунок В.3 – Проверка корректности пароля

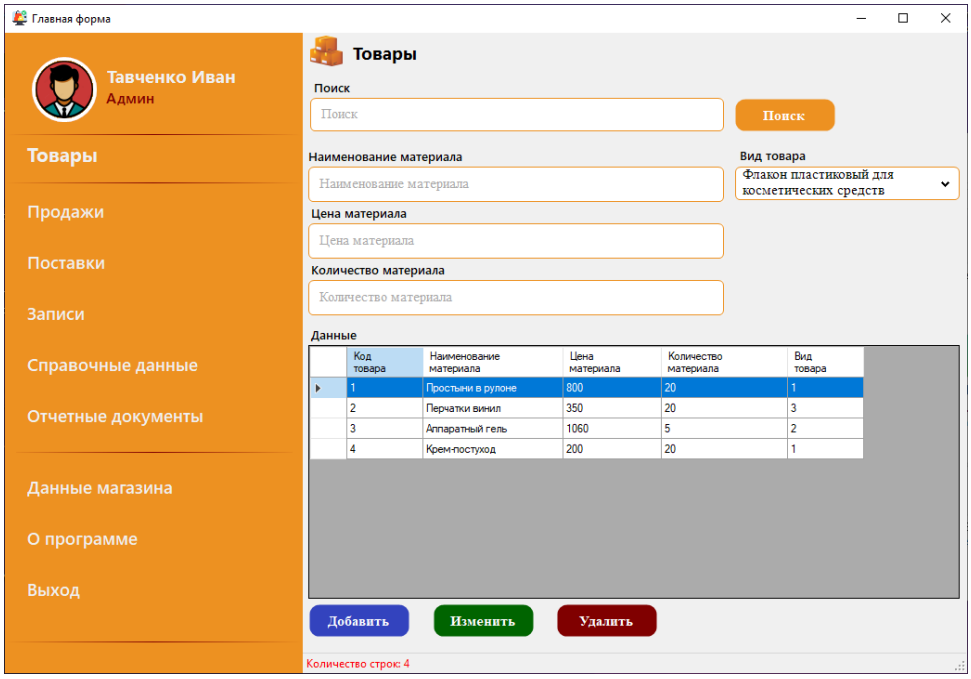


Рисунок В.4 – Главная форма системы

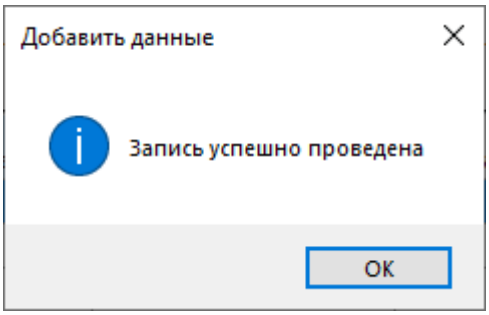


Рисунок В.5 – Окно успешного добавления данных

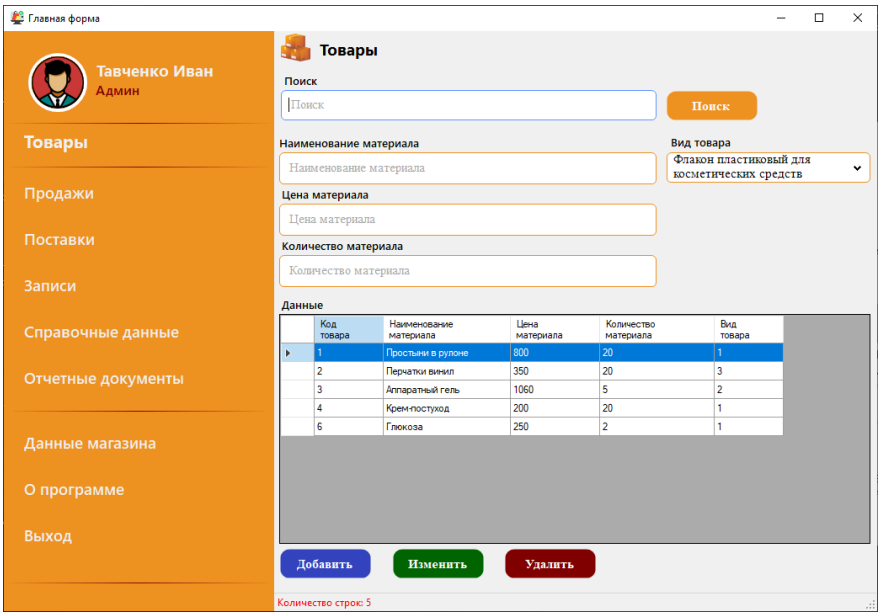


Рисунок В.6 – Добавление материала

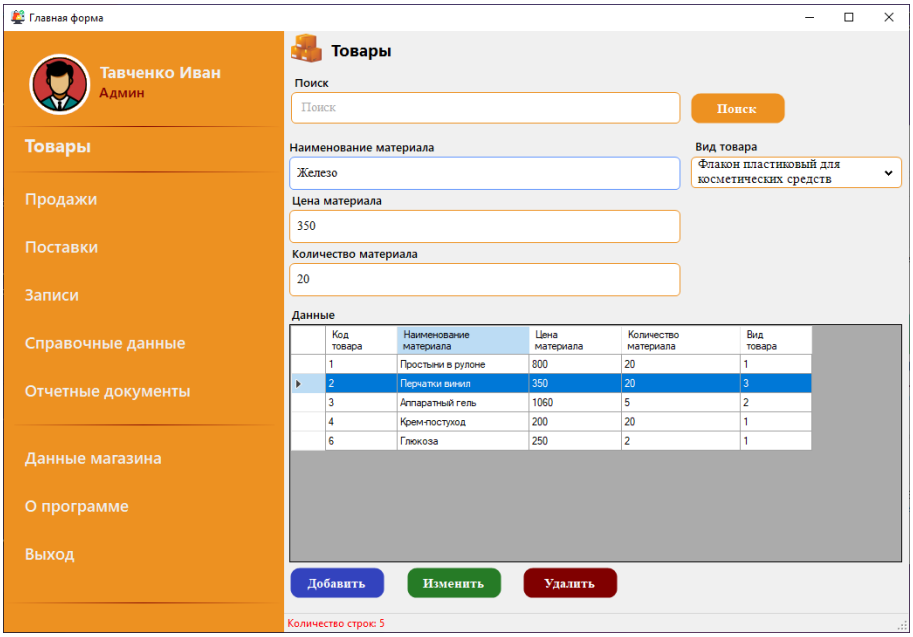


Рисунок В.7 – Ввод данных для изменения

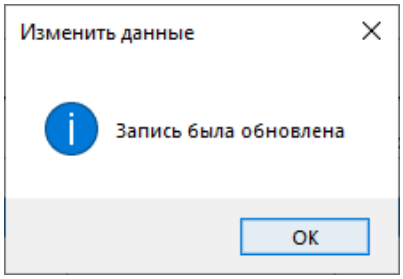


Рисунок В.8 – Успешное изменение данных

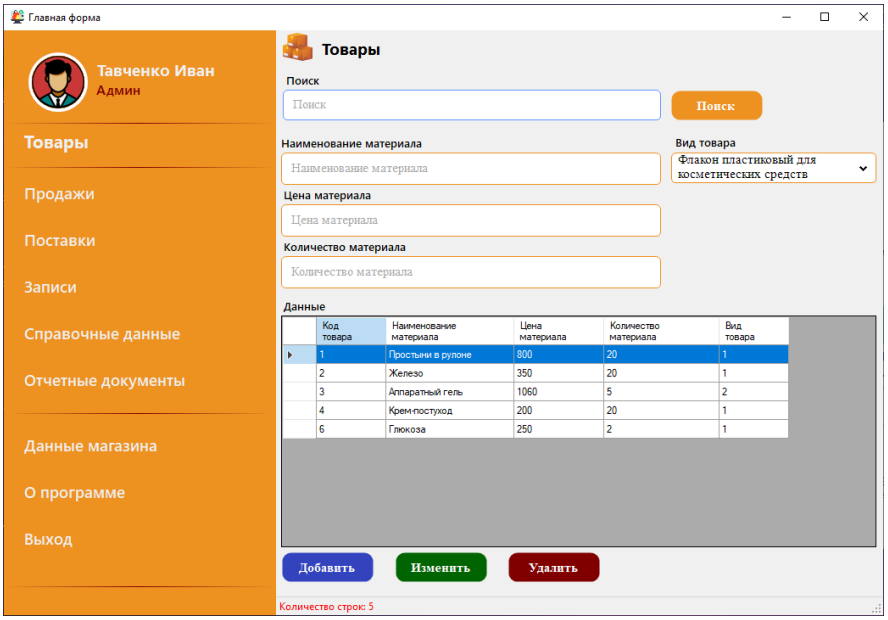


Рисунок В.9 – Примененные изменения

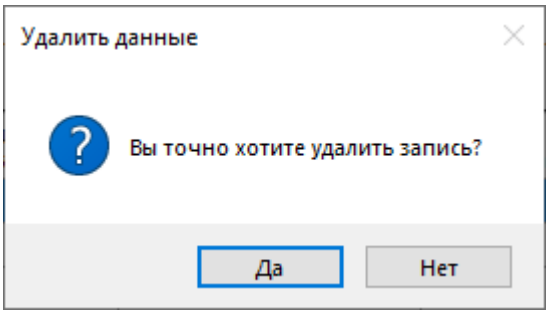


Рисунок В.10 – Подтверждение удаления данных

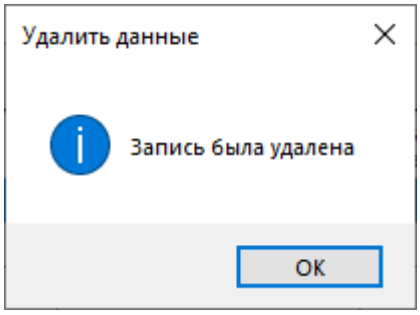


Рисунок В.11 – Окно удаления данных

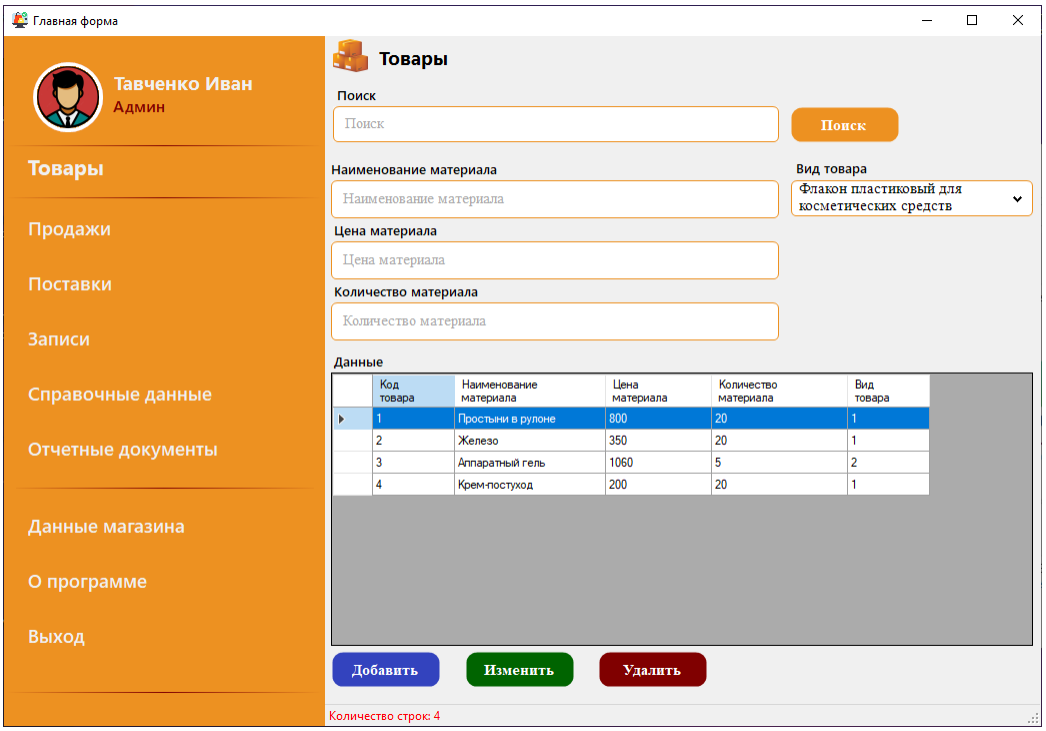


Рисунок В.12 – Форма после удаления

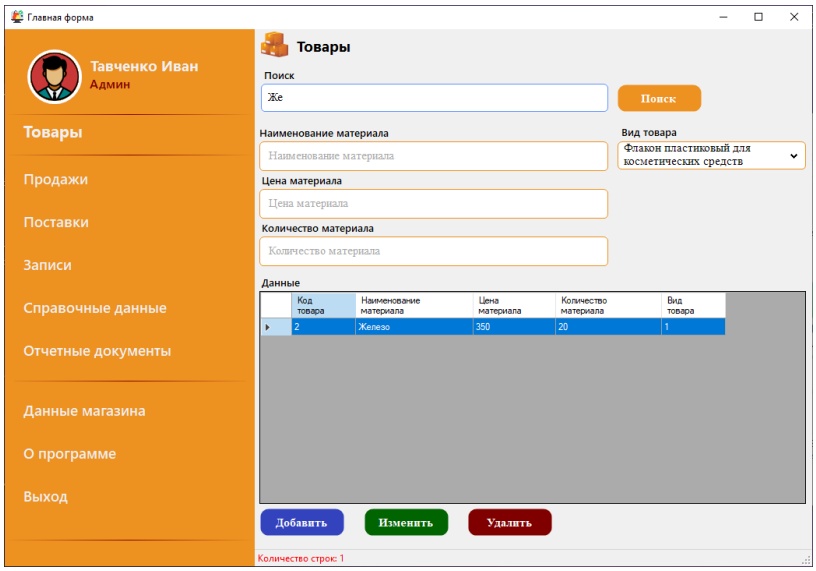


Рисунок В.13 – Проверка поиска элемента

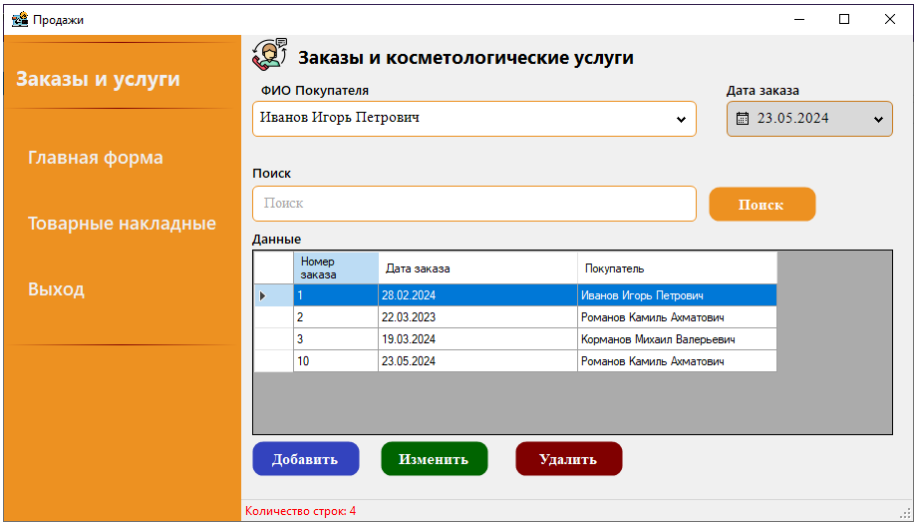


Рисунок В.14 – Добавление записи

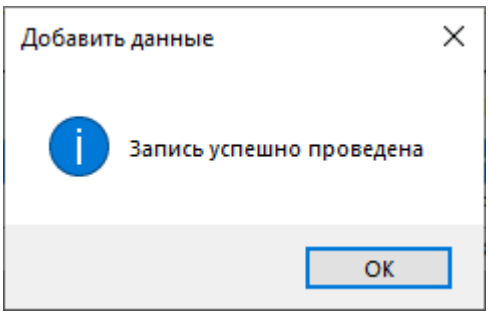


Рисунок В.15 – Подтверждение успешного добавления записи

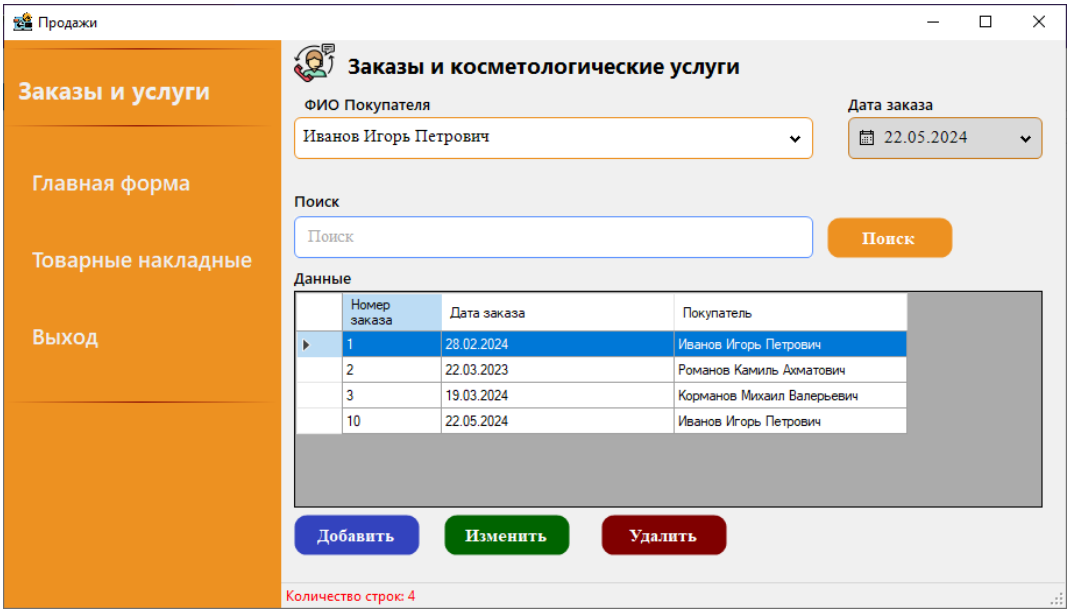


Рисунок В.16 – Изменение данных

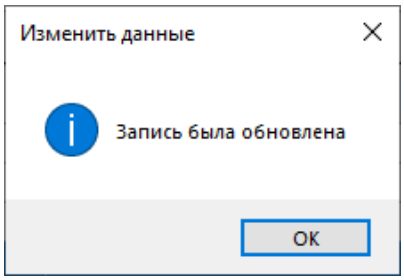


Рисунок В.17 – Подтверждение обновления данных

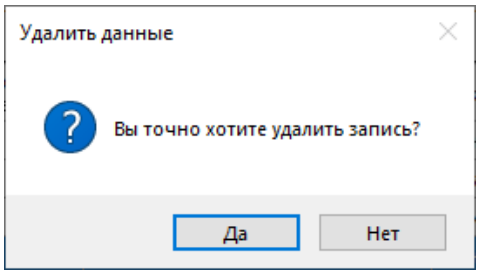


Рисунок В.18 – Подтверждение удаления

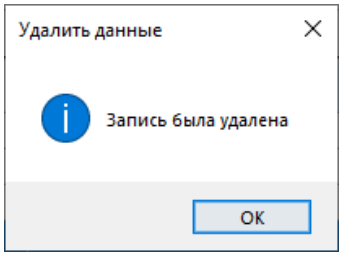


Рисунок В.19 – Оповещение удаления записи

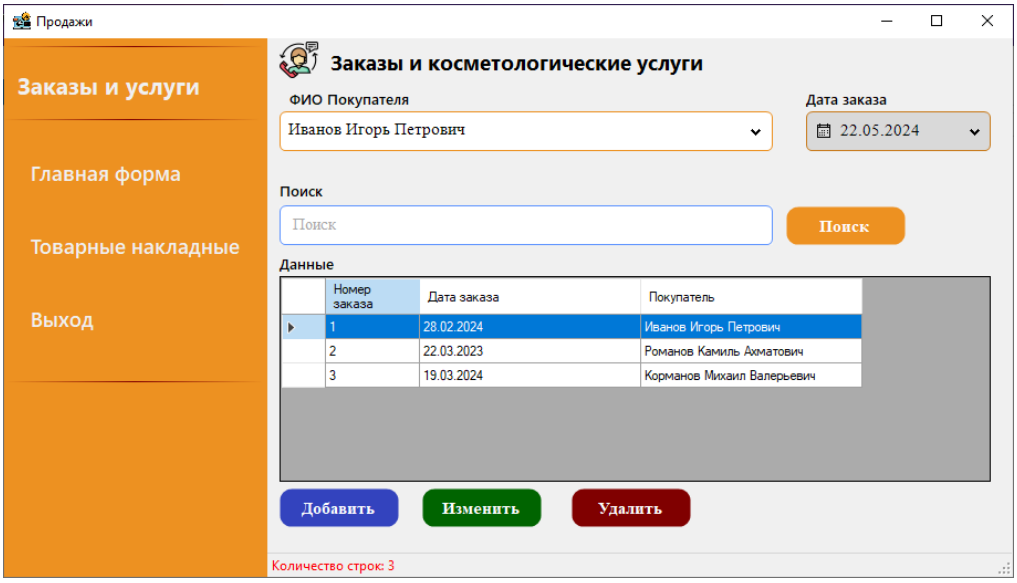


Рисунок В.20 – Форма после удаления

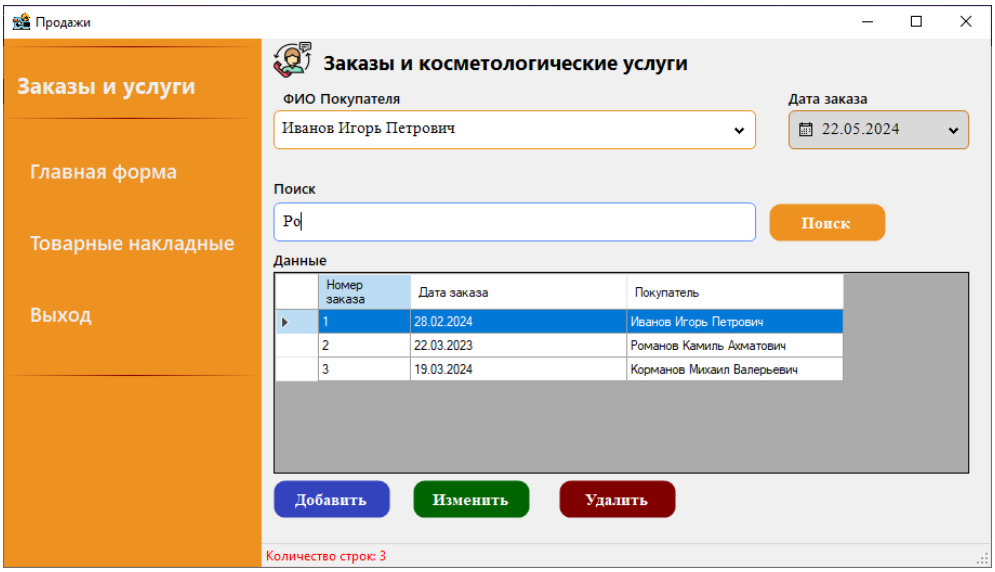


Рисунок В.21 – Поиск записи

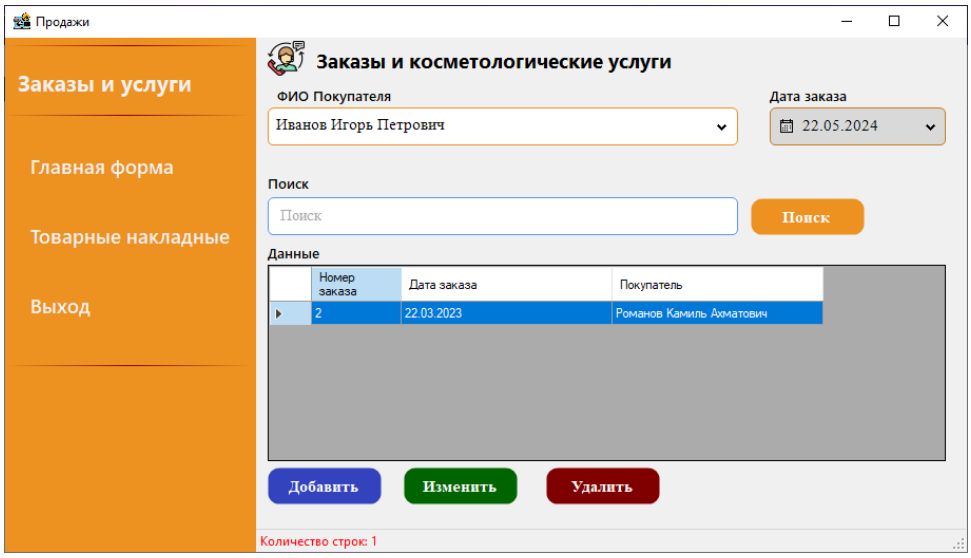


Рисунок В.22 – Результат поиска

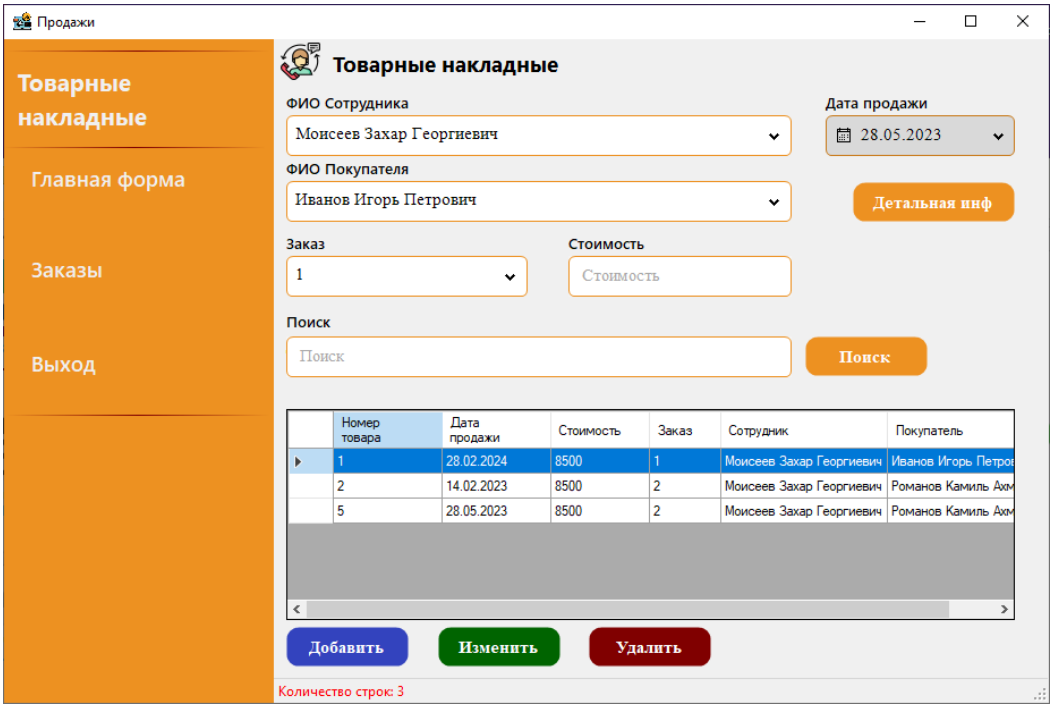


Рисунок В.23 – Вкладка «Товарные накладные»

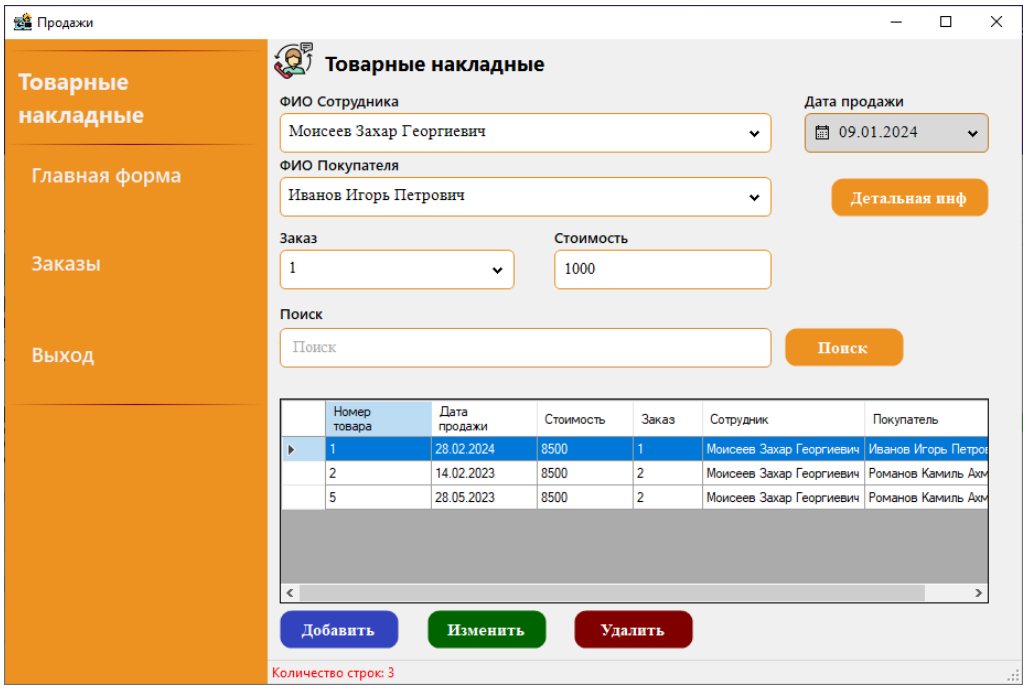


Рисунок В.24 – Ввод данных для добавления

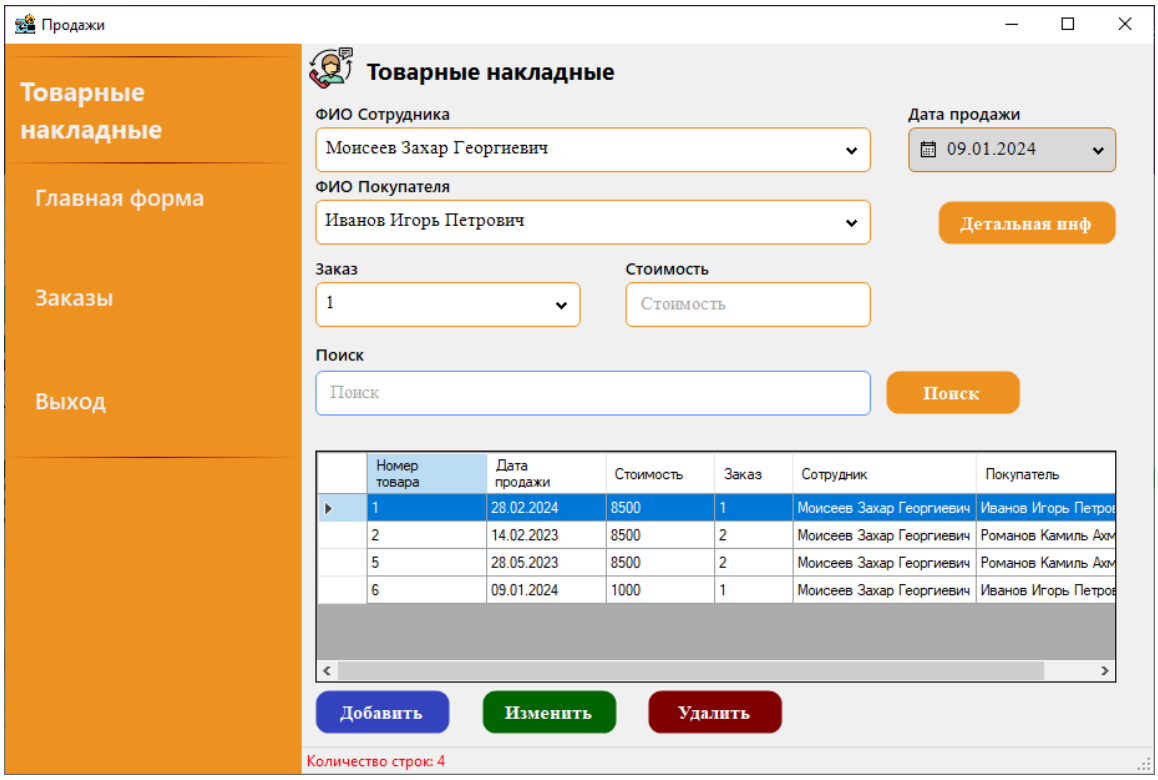


Рисунок В.25 – Новая запись добавлена

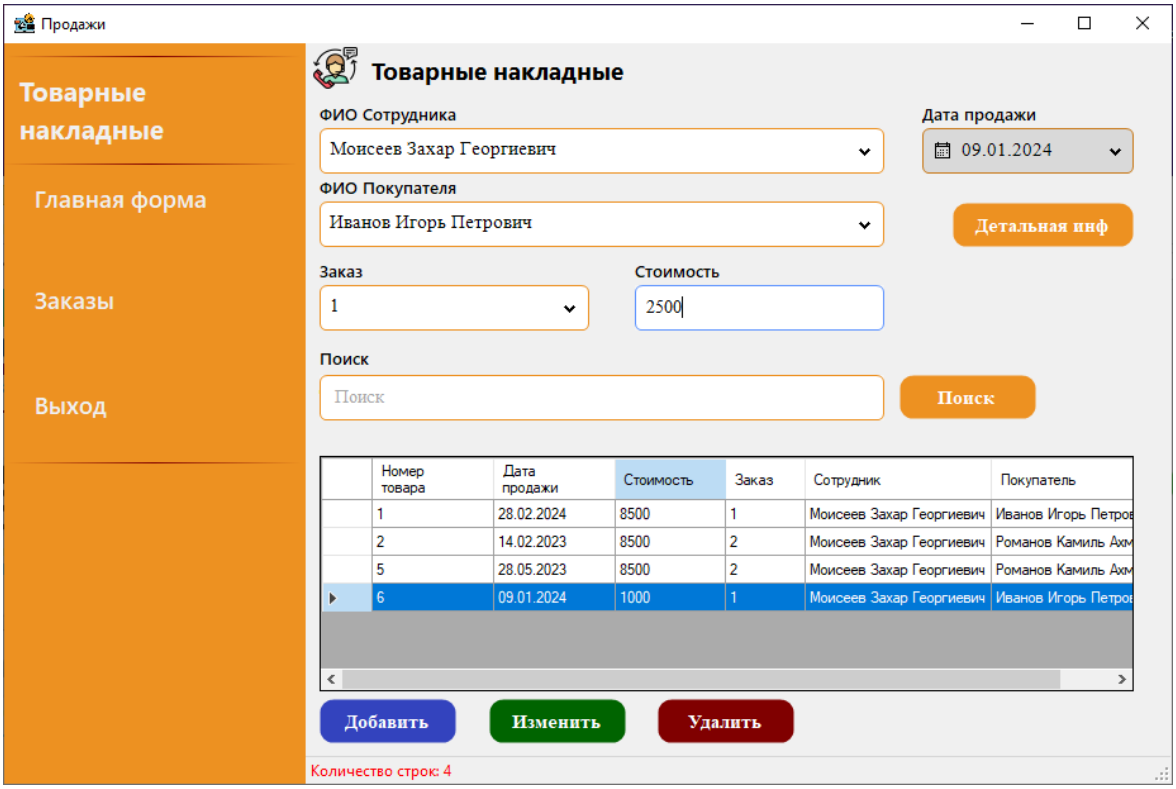


Рисунок В.26 – Изменение данных

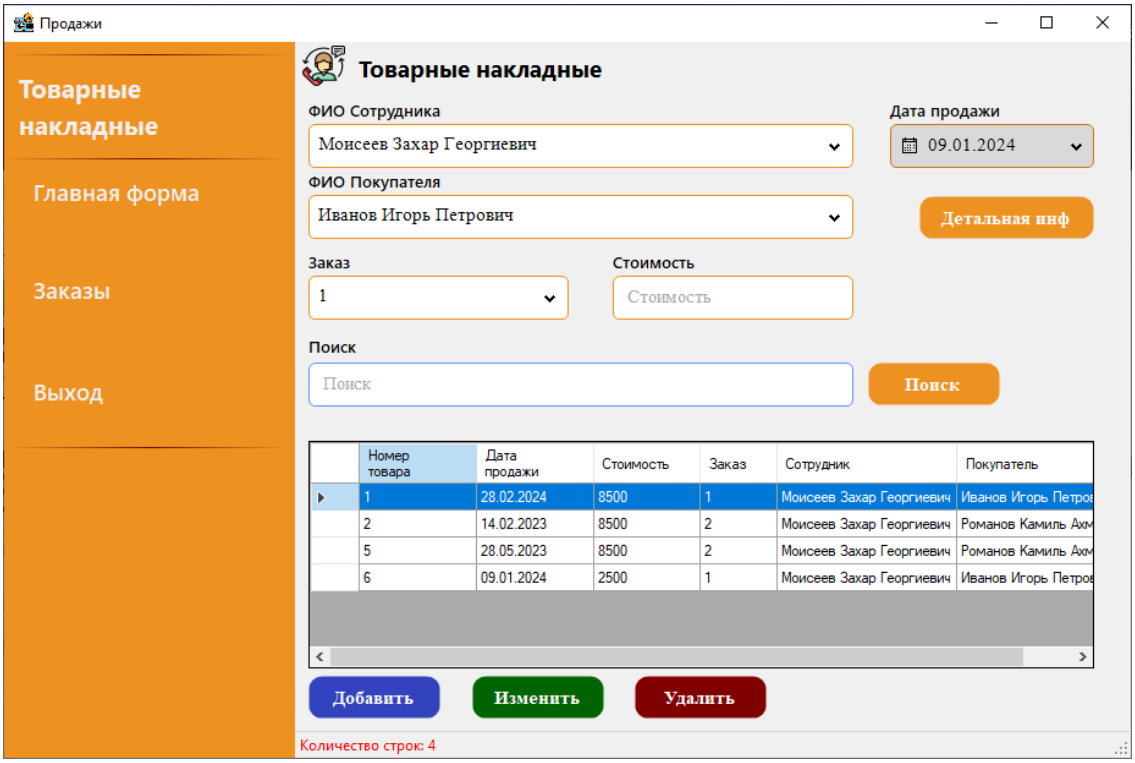


Рисунок В.27 – Измененные данные

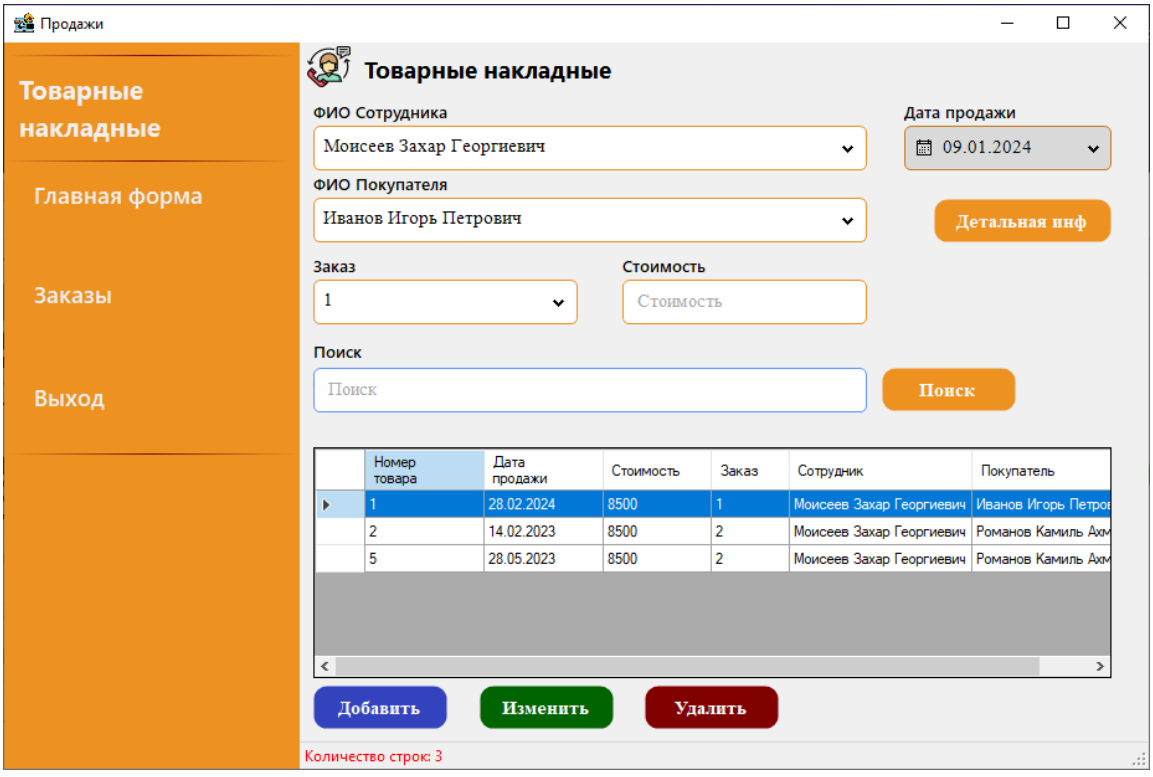


Рисунок В.28 – Удаление записи

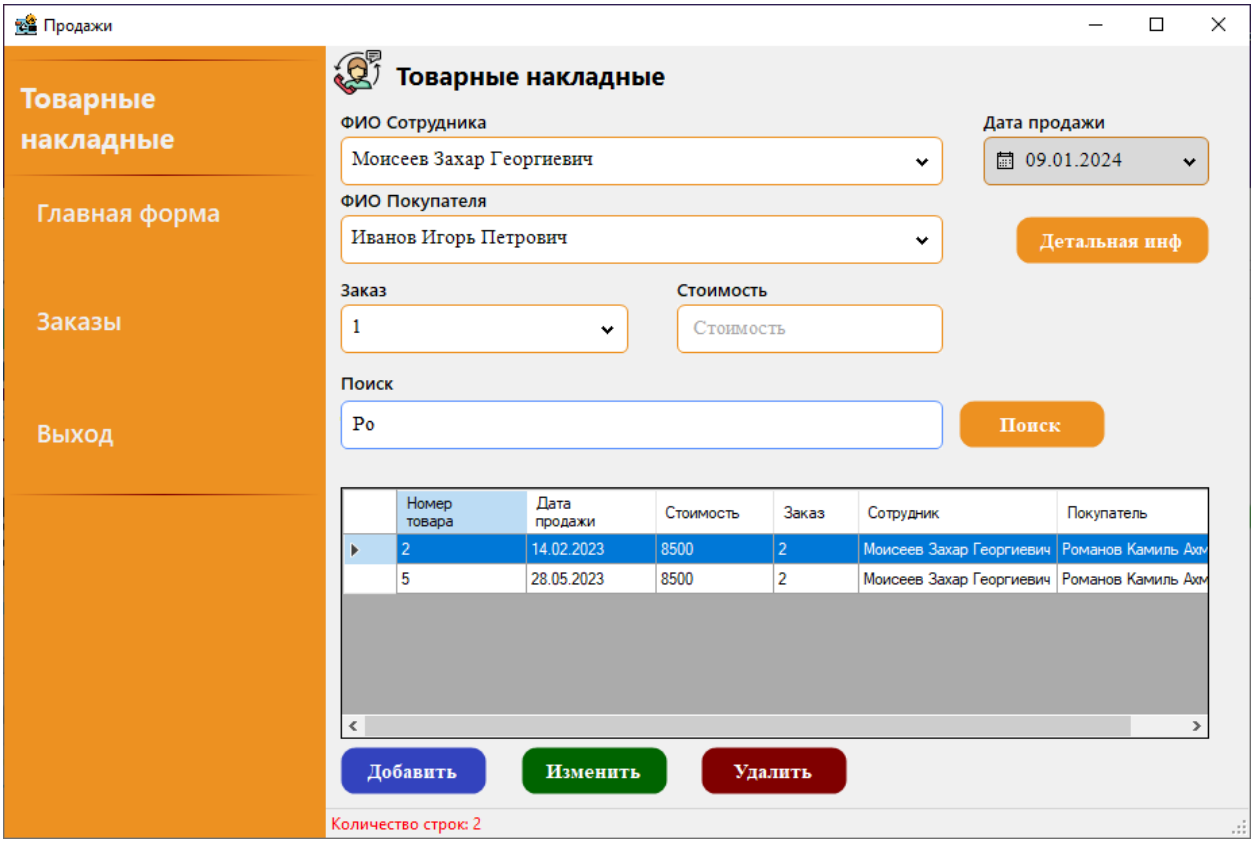


Рисунок В.29 – Поиск по записям

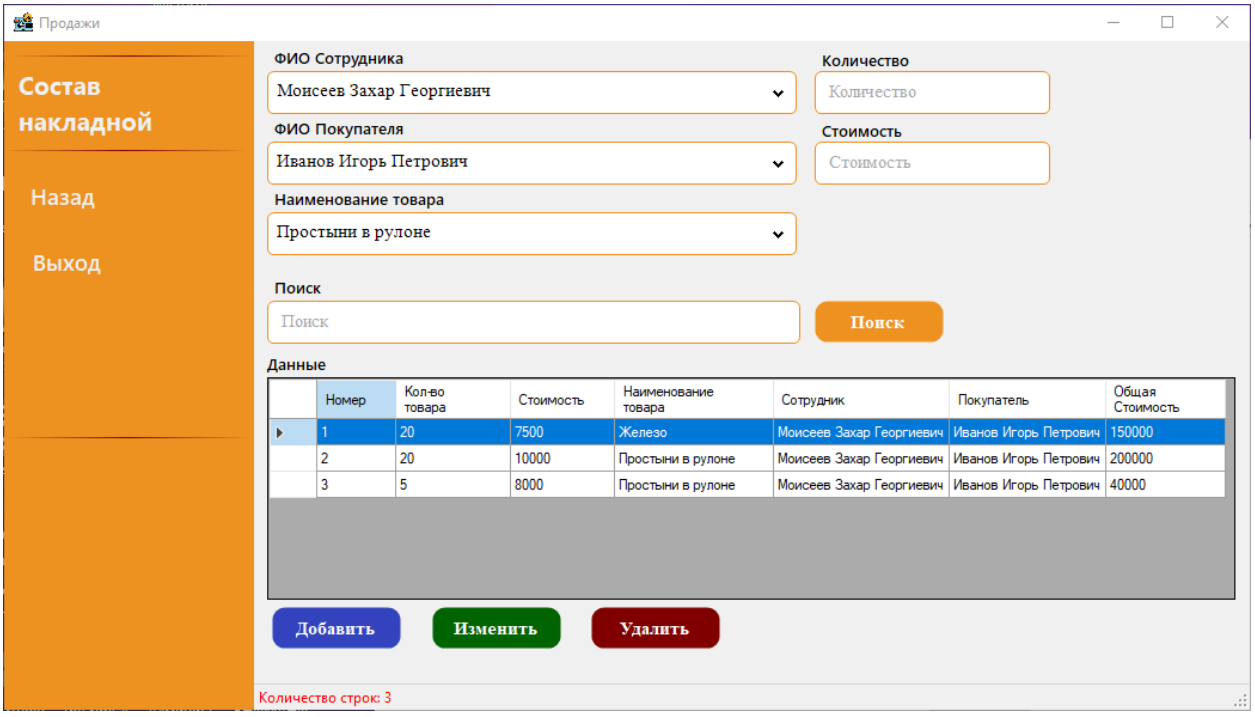


Рисунок В.30 – Форма детальной информации

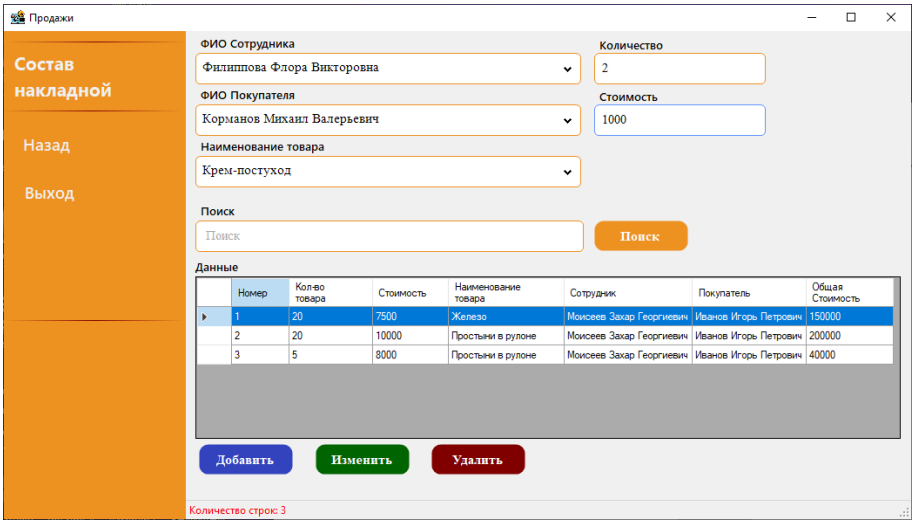


Рисунок В.31 – Добавление информации

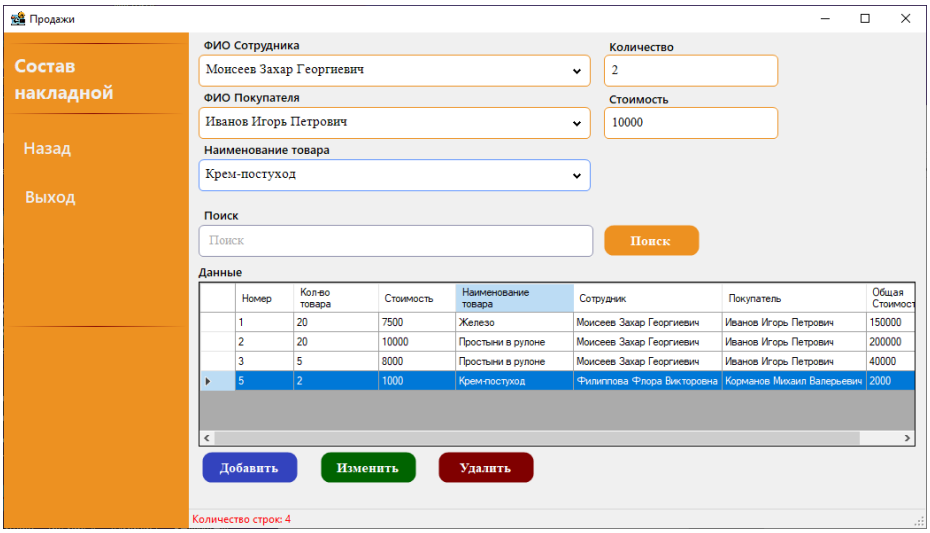


Рисунок В.32 – Изменение информации

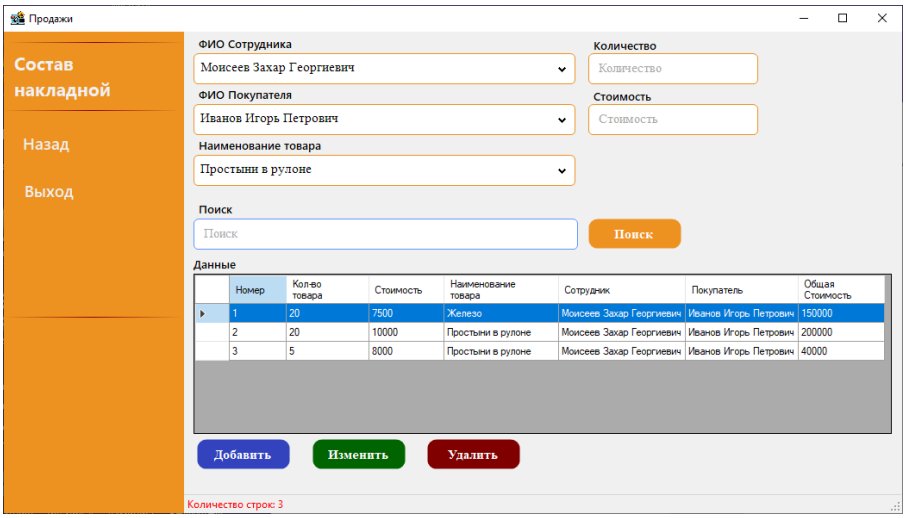


Рисунок В.33 – Удаление записи

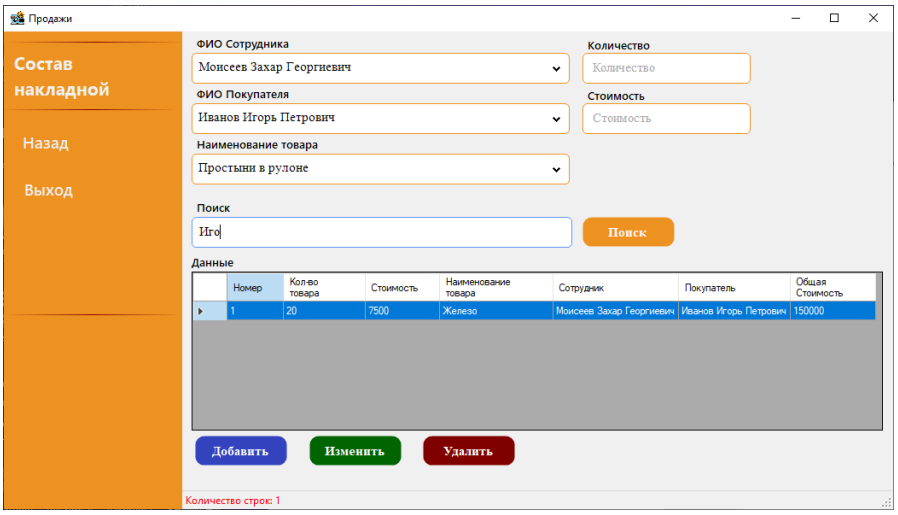


Рисунок В.34 – Поиск записи

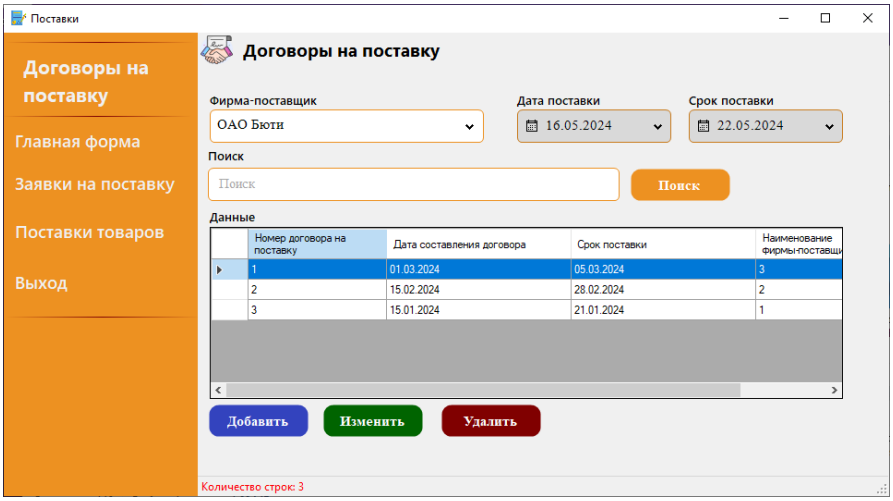


Рисунок В.35 – Главное окно «Поставки»

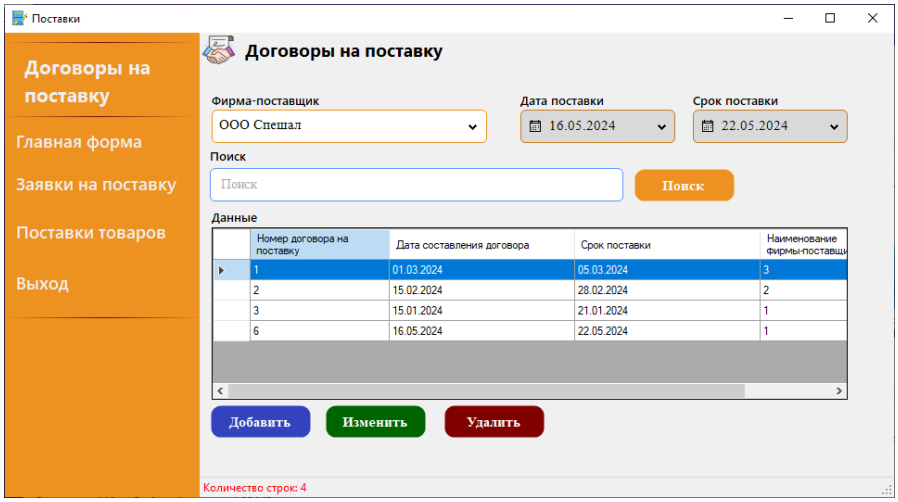


Рисунок В.36 – Добавление договора на поставку

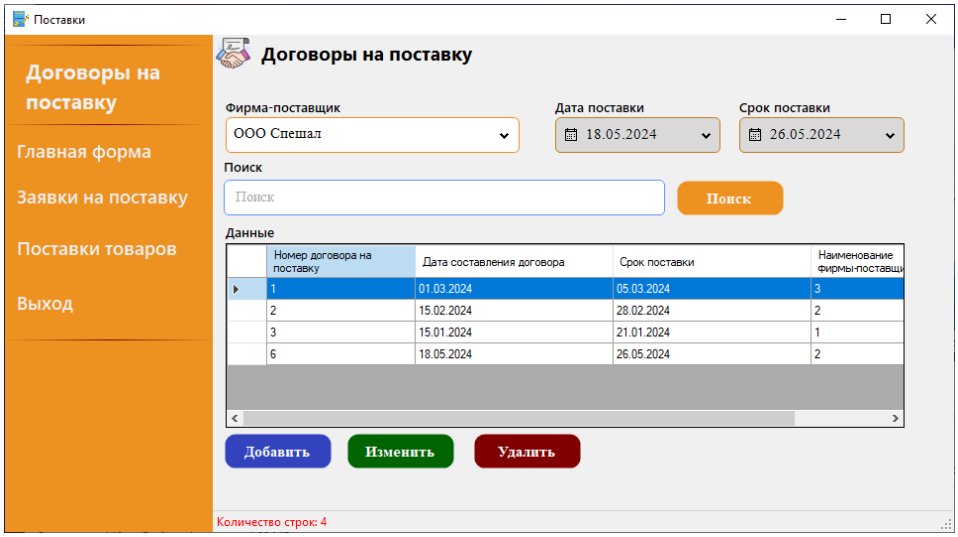


Рисунок В.37 – Внесение изменений в договор на поставку

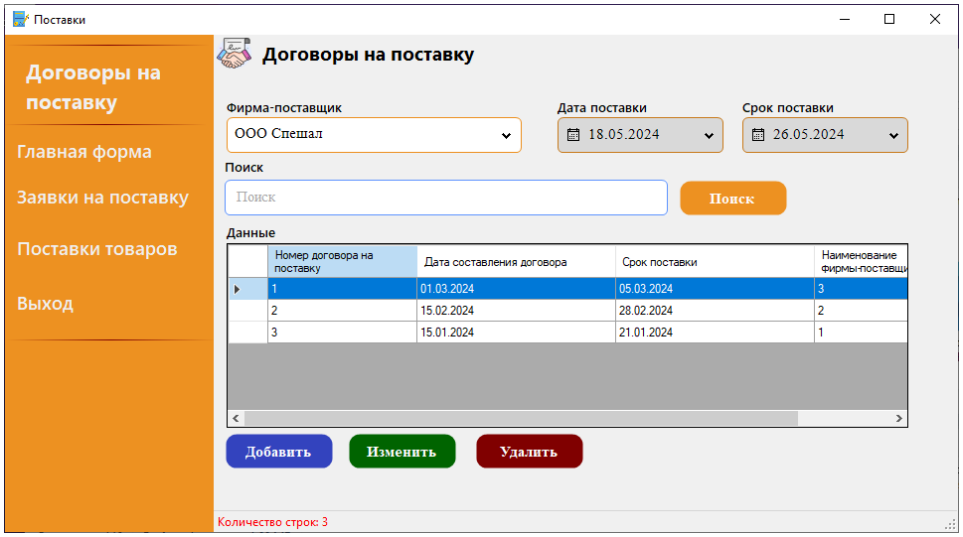


Рисунок В.38 – Удаления записи

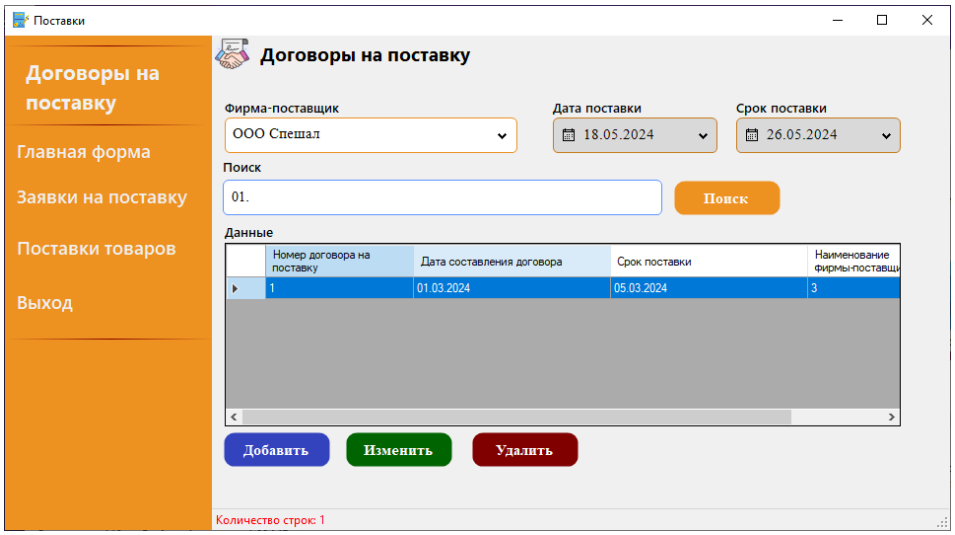


Рисунок В.39 – Поиск договора на поставку

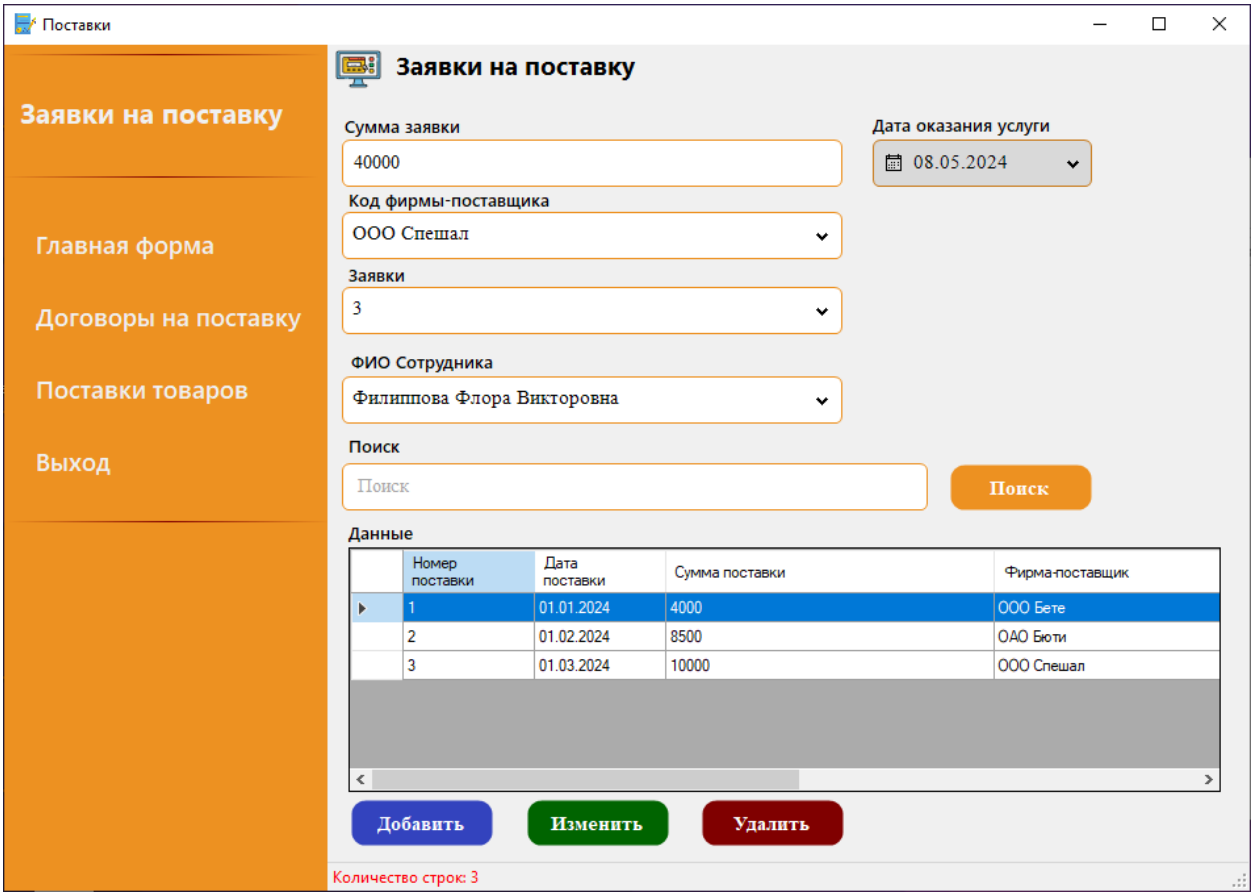


Рисунок В.40 – Окно заявки на поставку

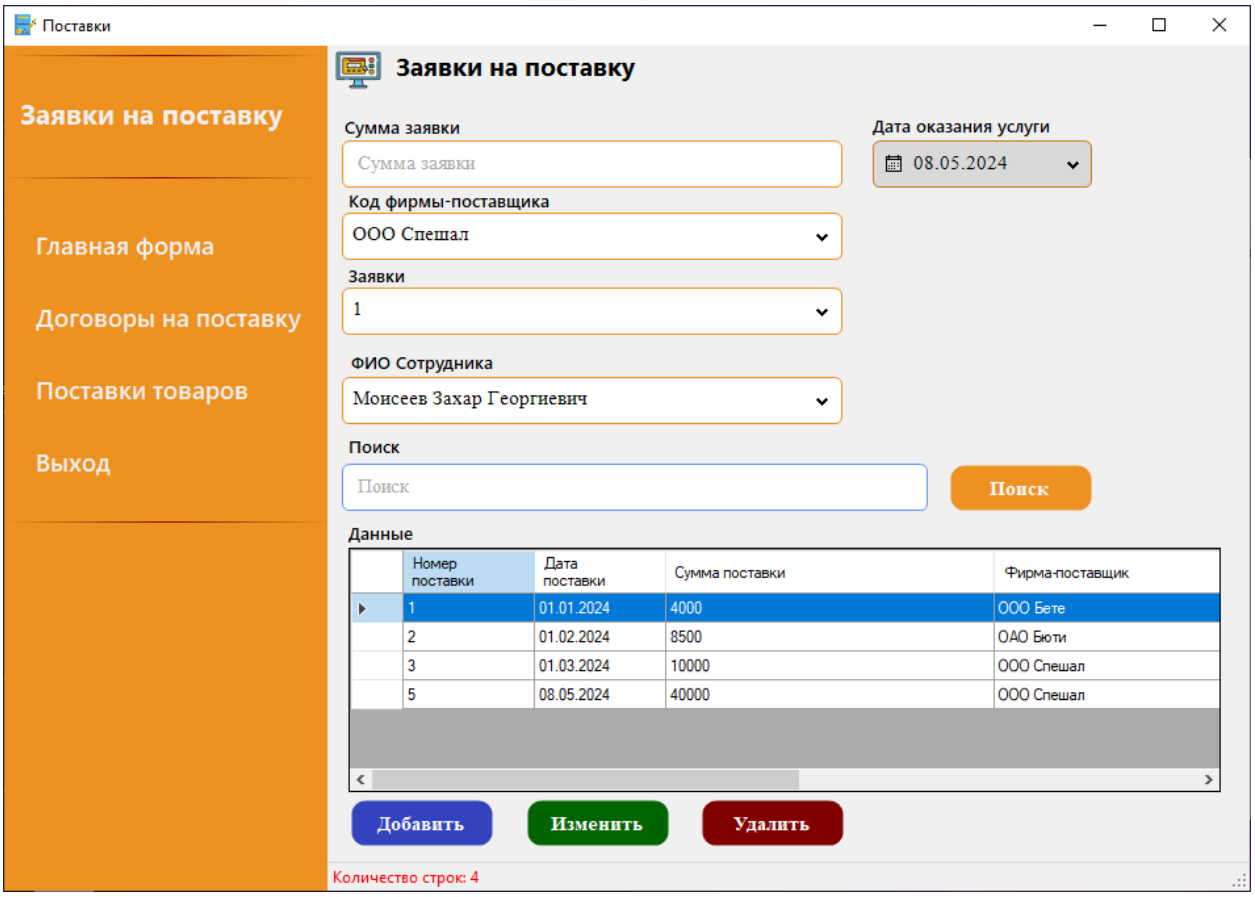


Рисунок В.41 – Добавление заявки на поставку

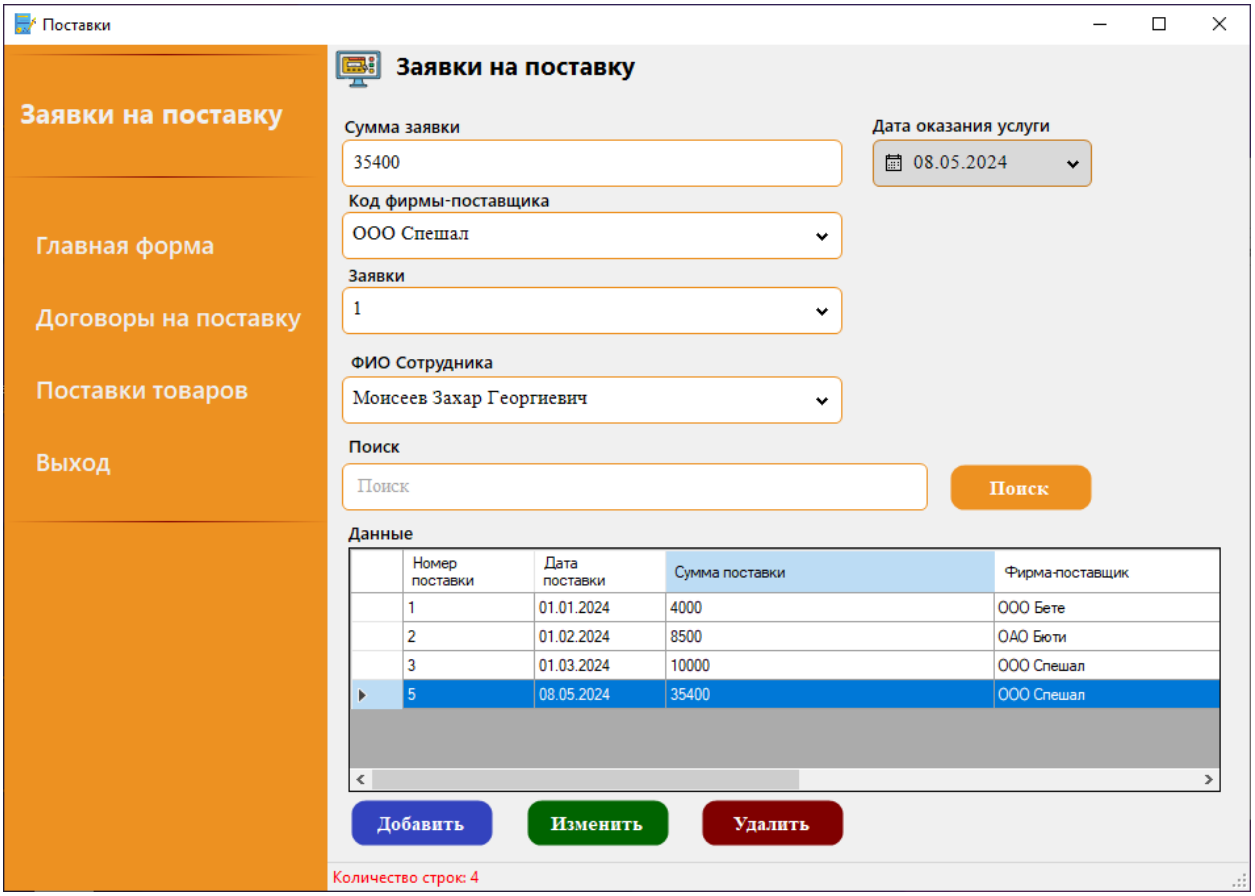


Рисунок В.42 – Изменение данных

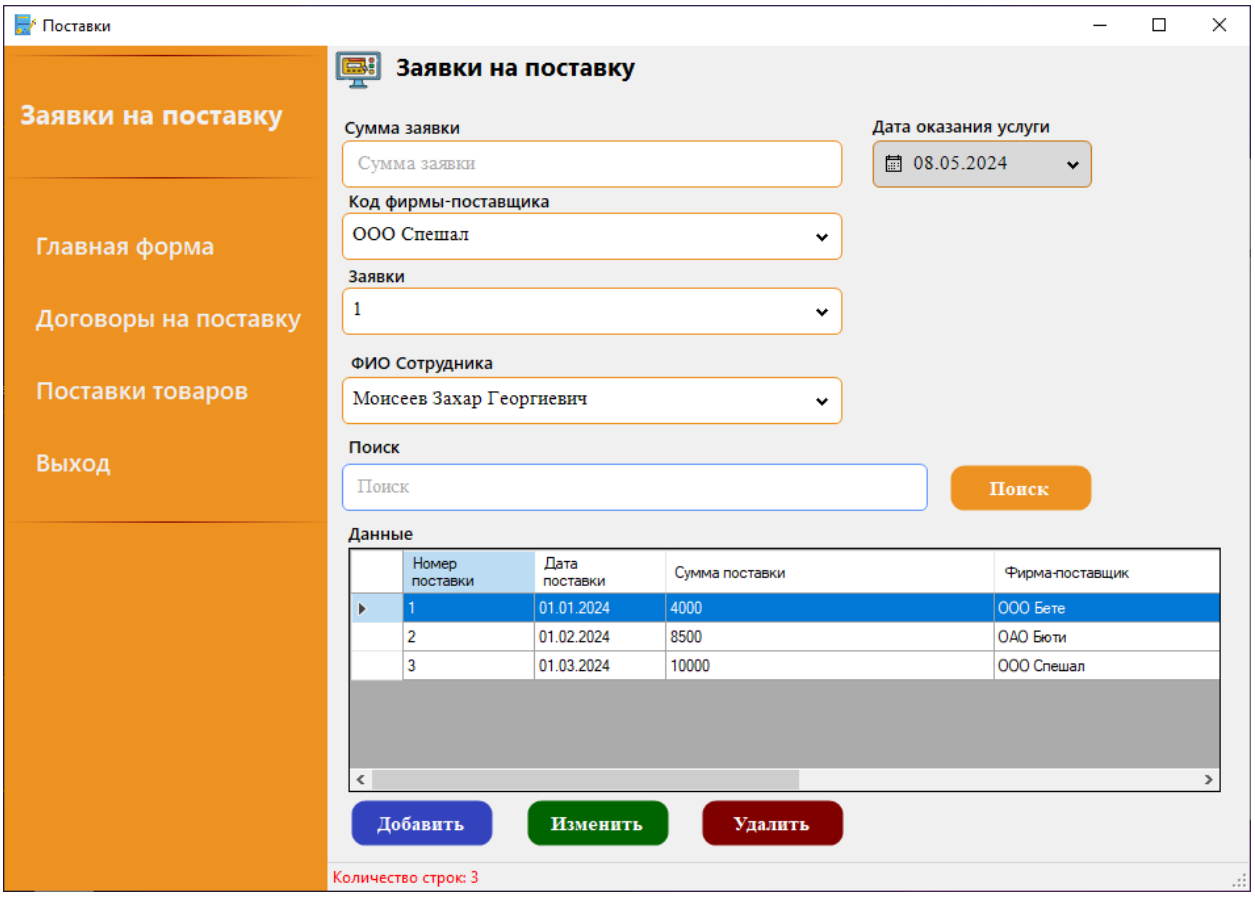


Рисунок В.43 – Удаление записи

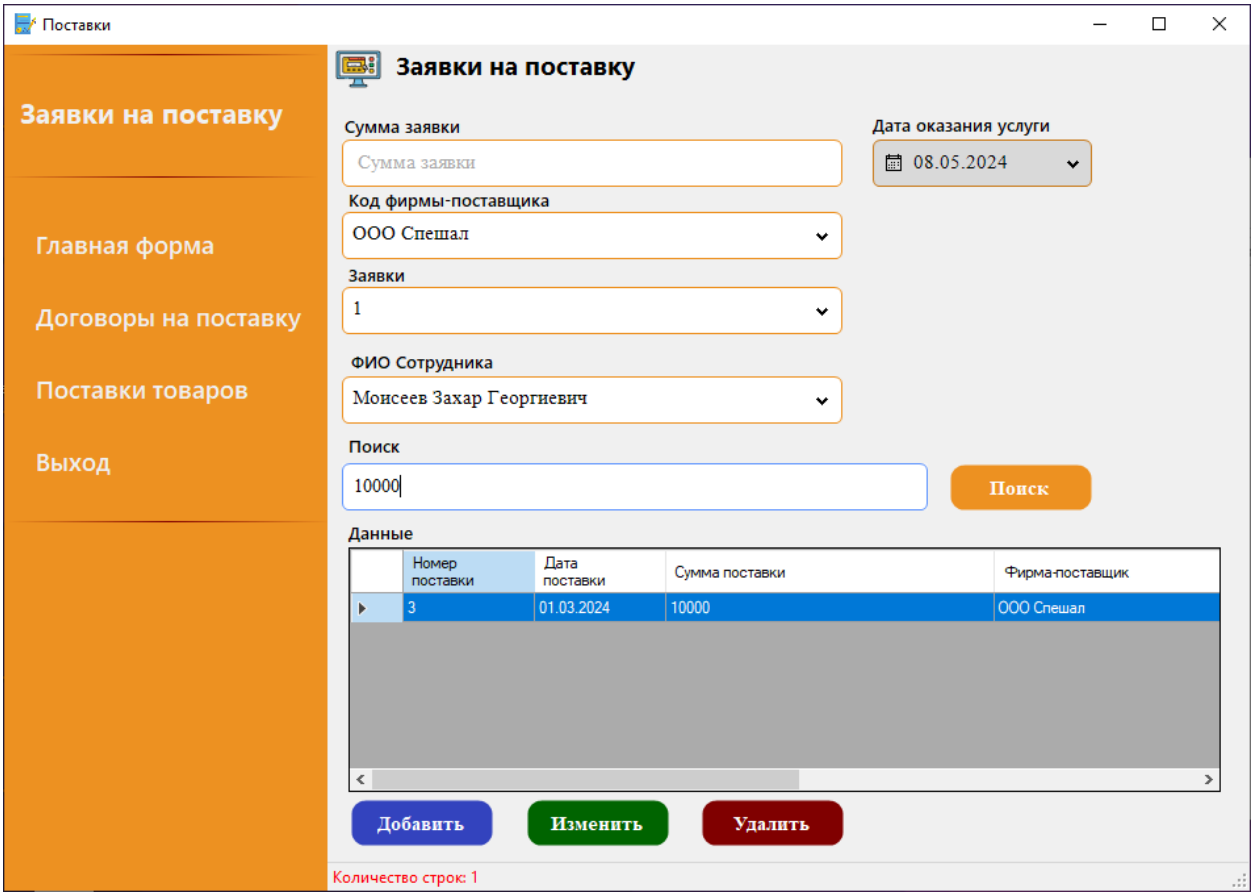


Рисунок В.44 – Поиск записи

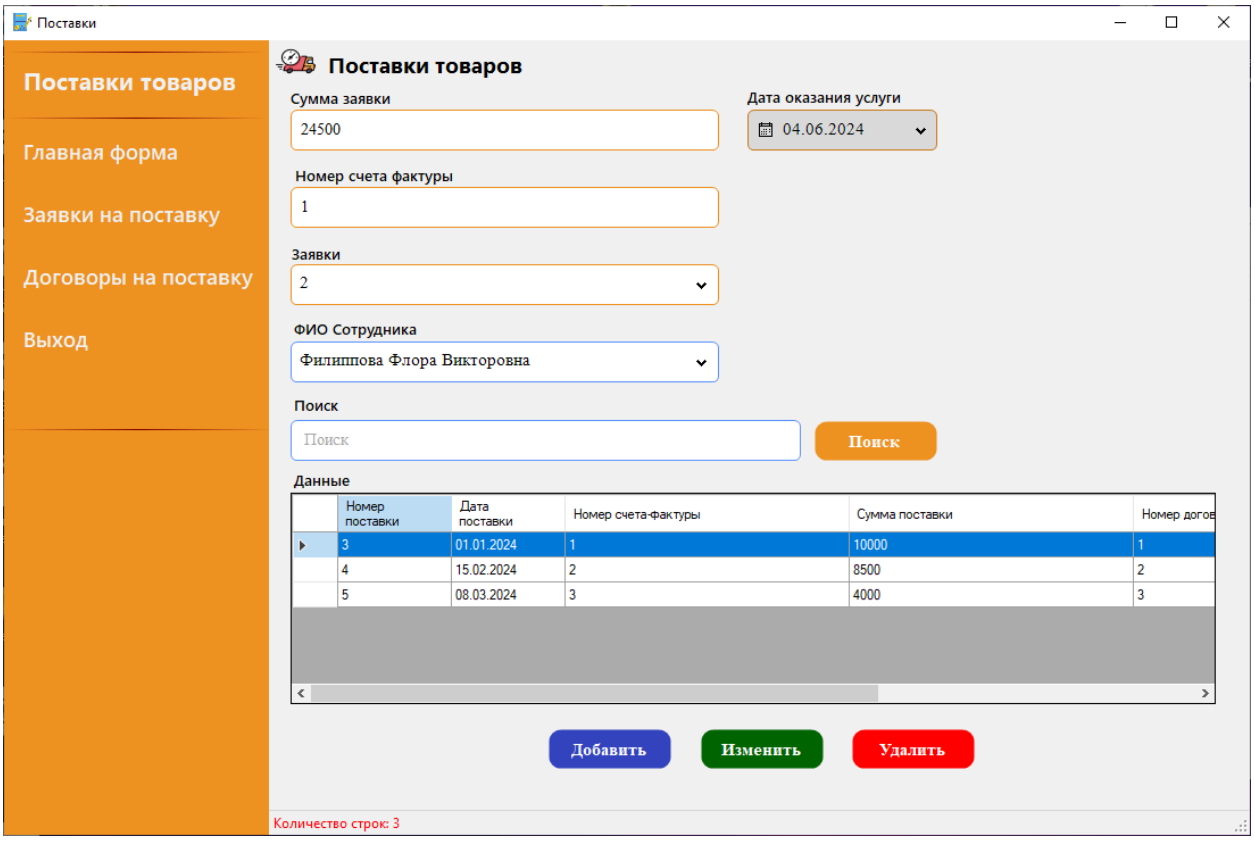


Рисунок В.45 – Окно «Поставка товаров»

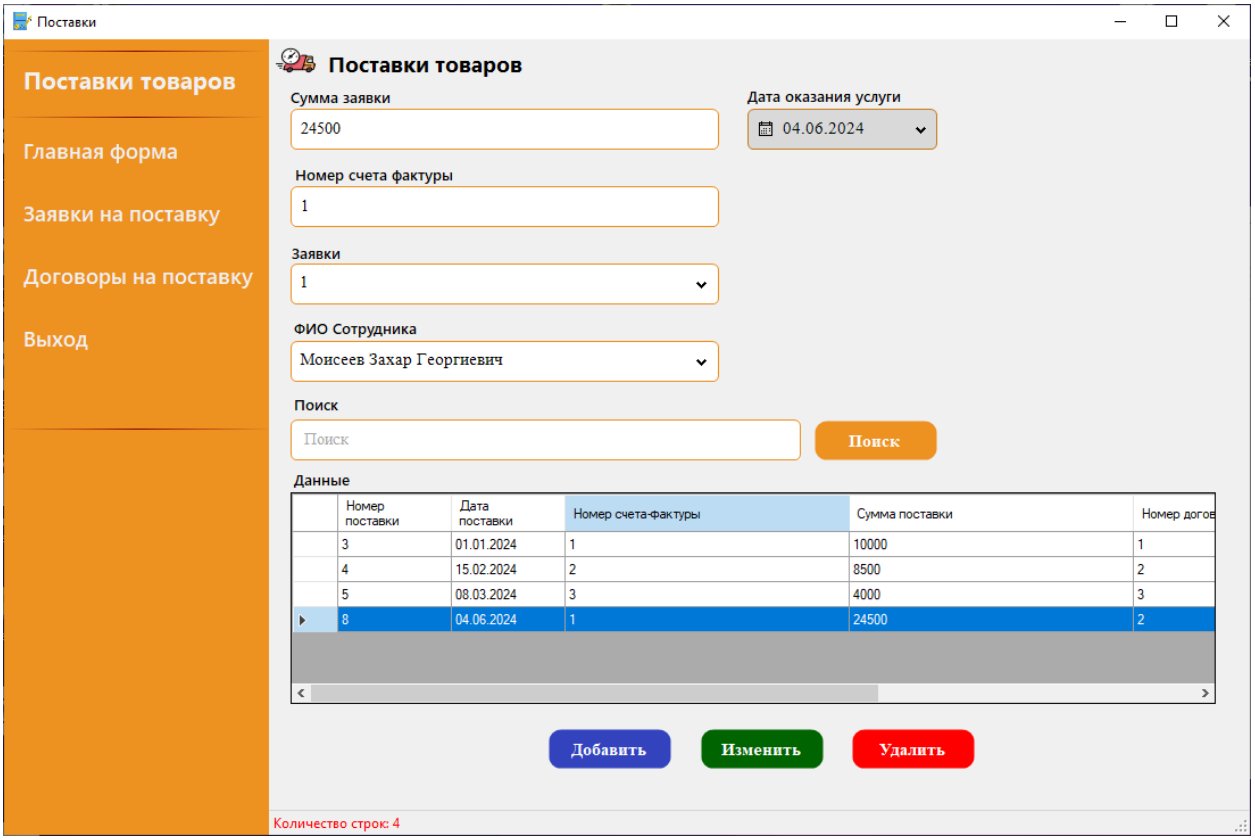


Рисунок В.46 – Добавление записи

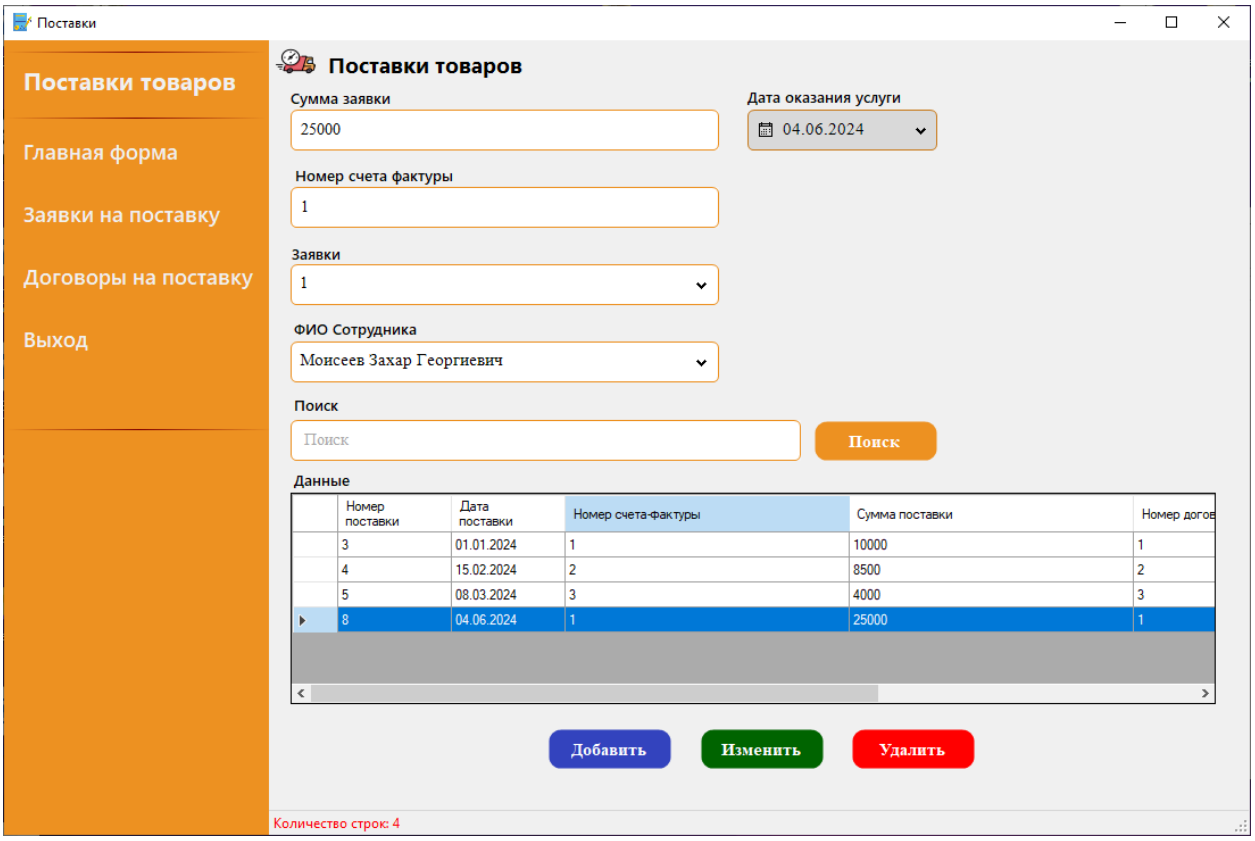


Рисунок В.47 – Изменение записи

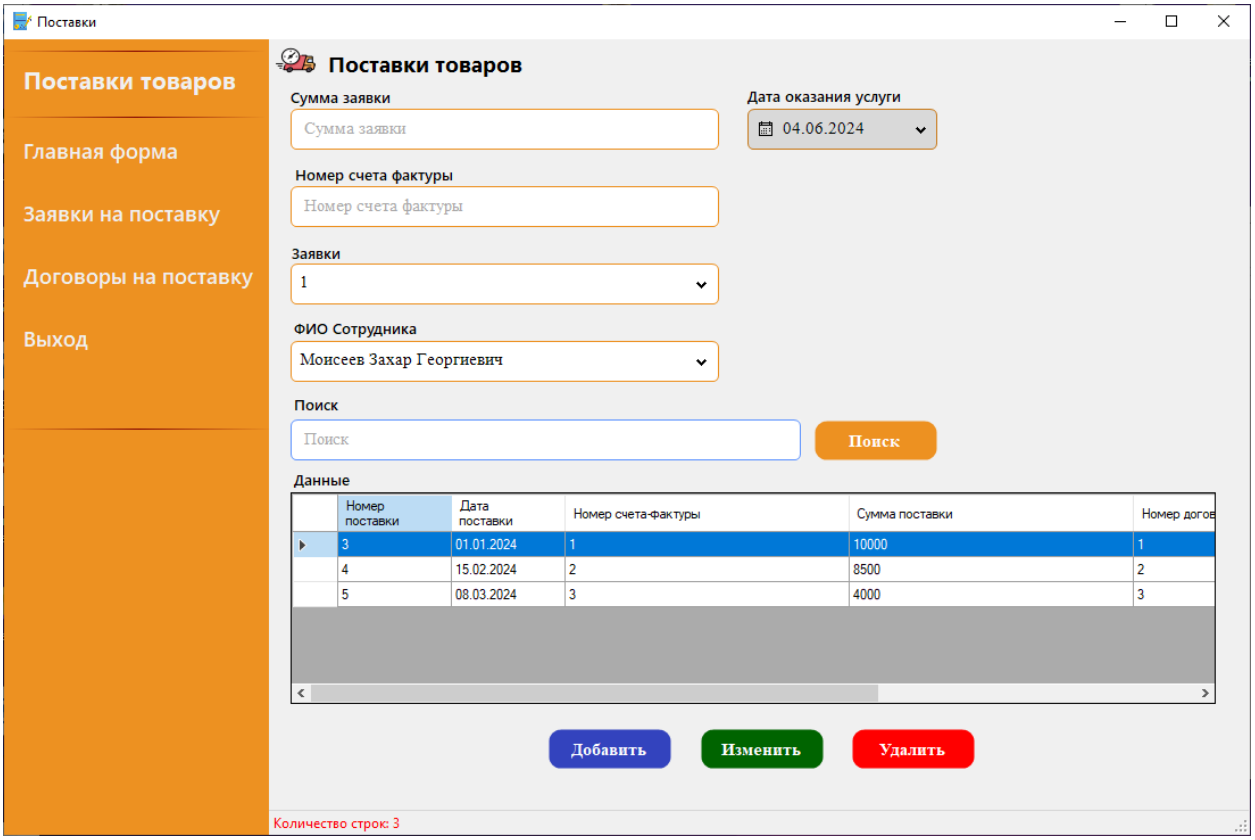


Рисунок В.48 – Удаление записи

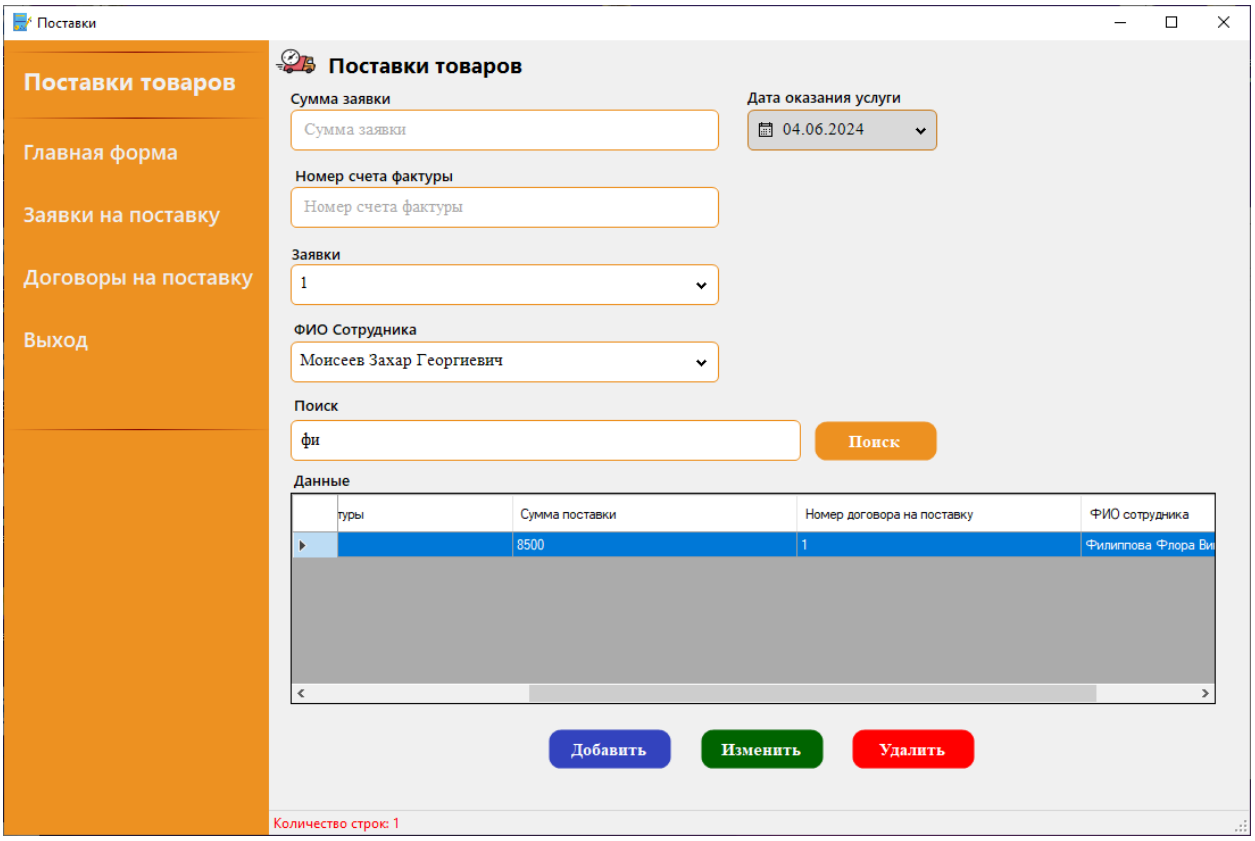


Рисунок В.49 – Поиск записи

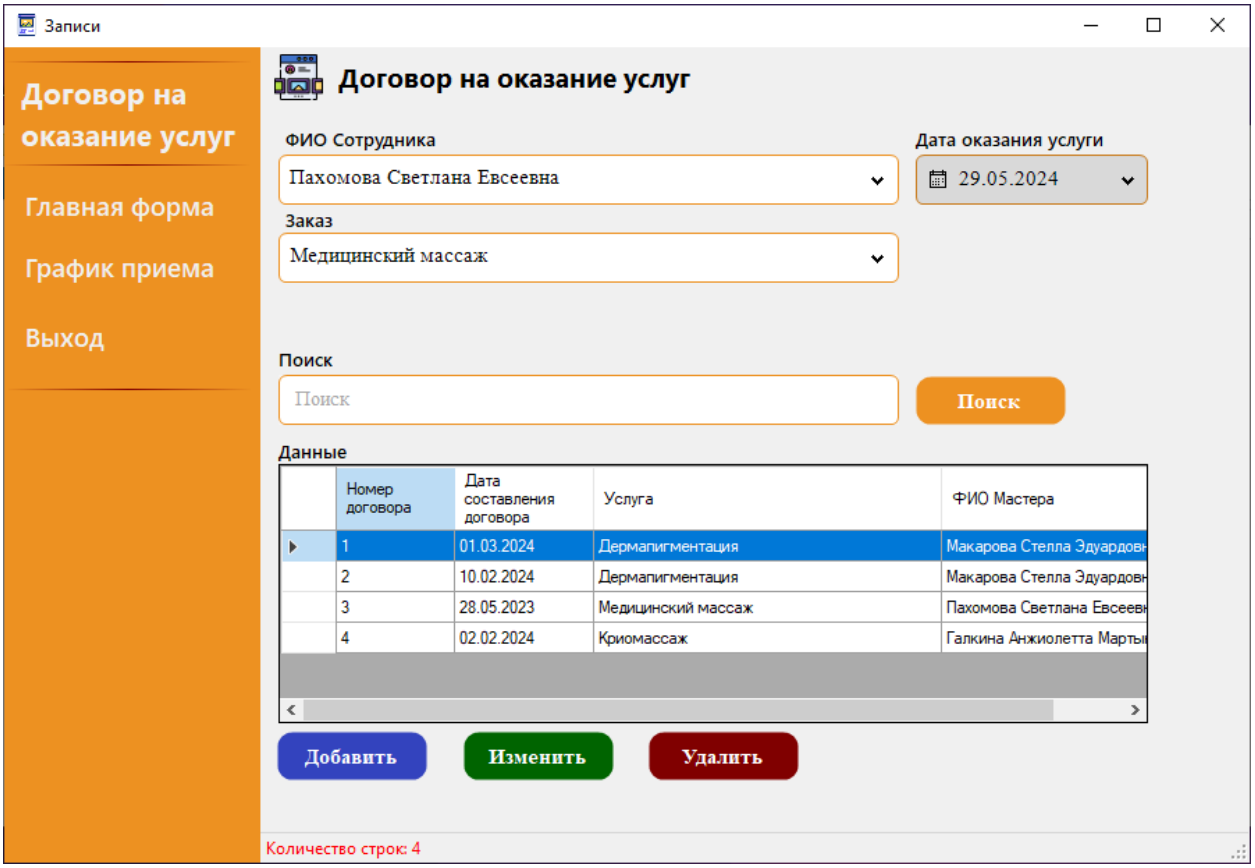


Рисунок В.50 - Форма «Записи»

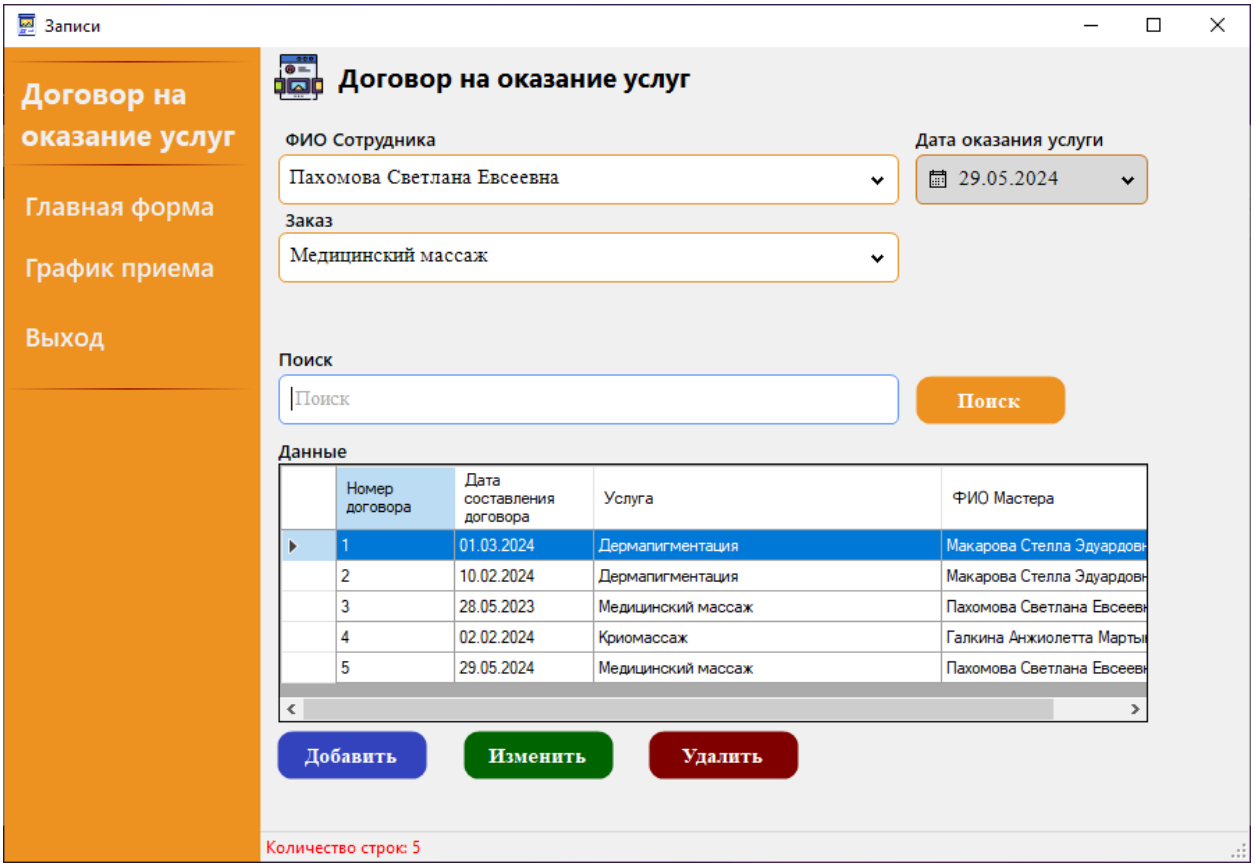


Рисунок В.51 – Добавленная запись

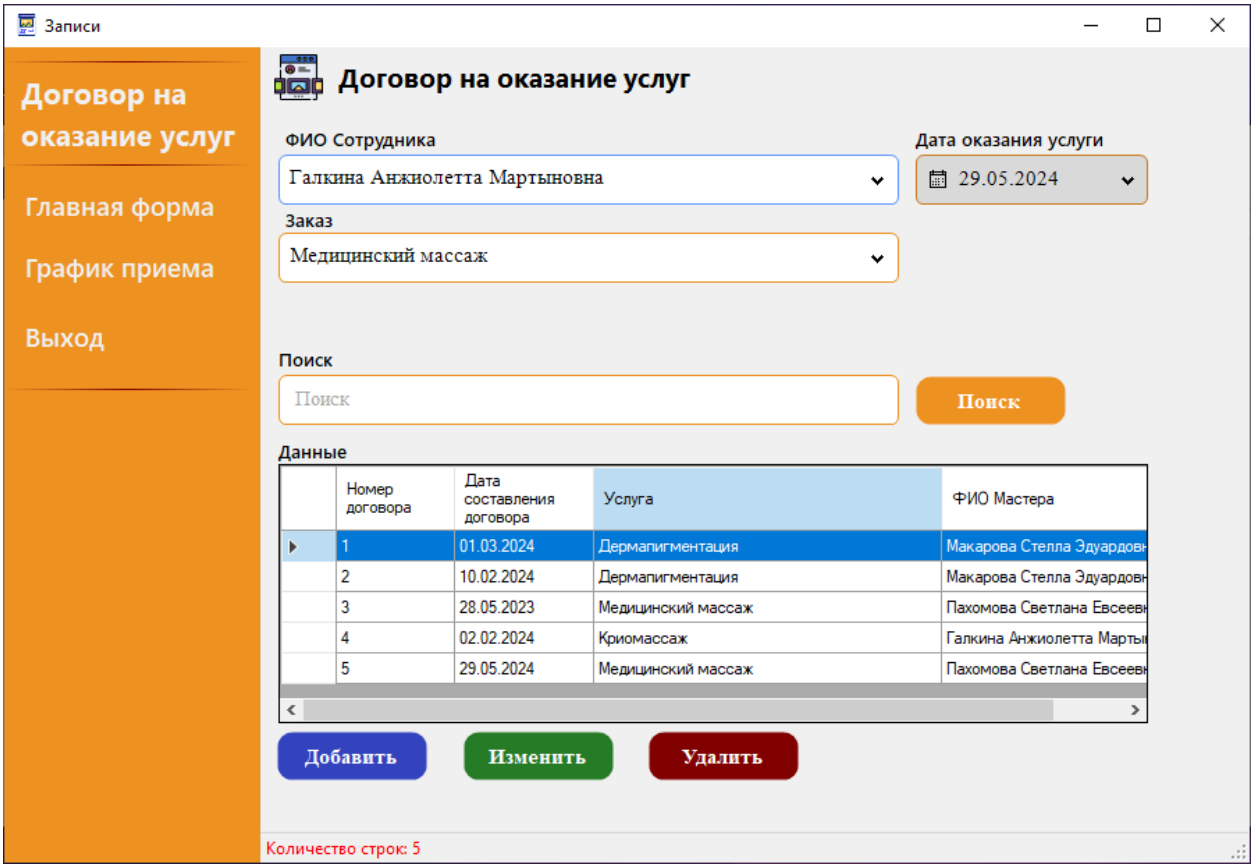


Рисунок В.52 – Изменение данных записи

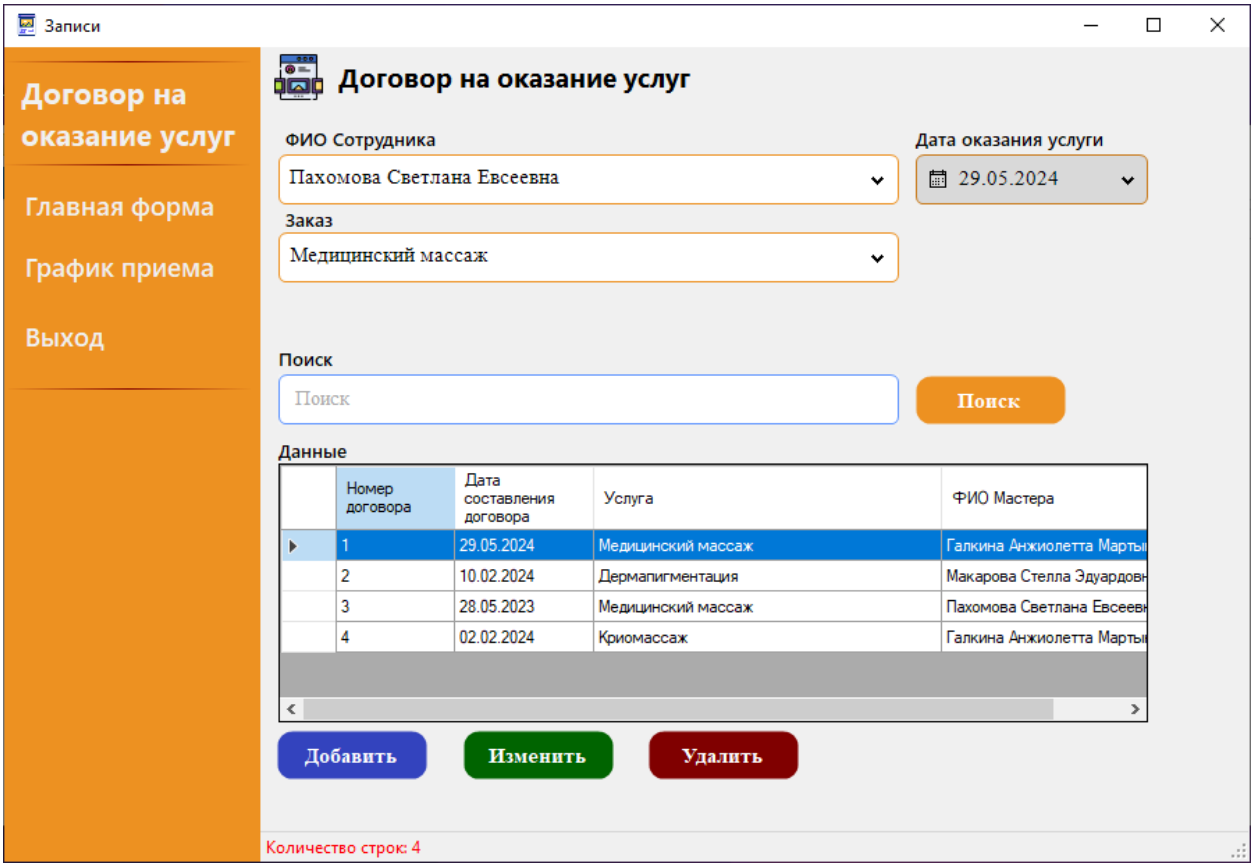


Рисунок В.53 – Удаление записи

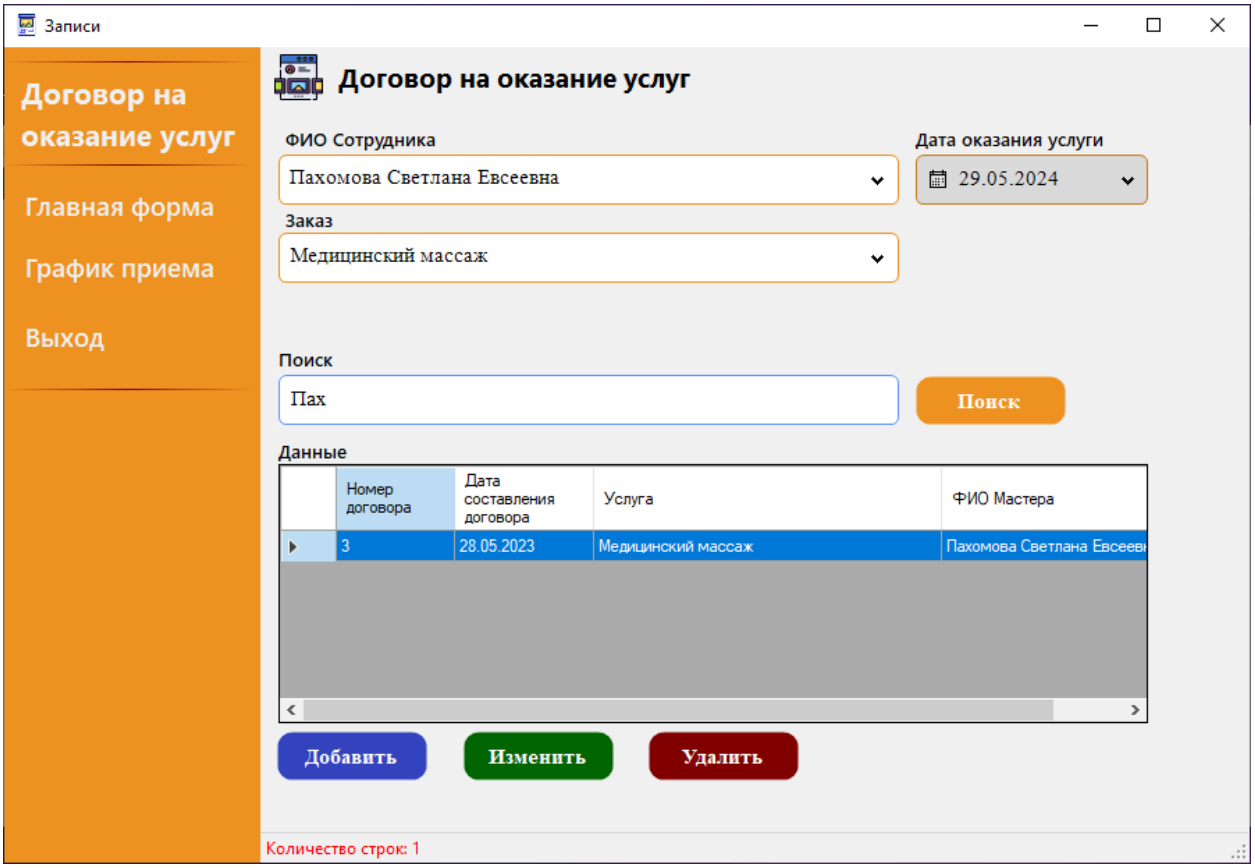


Рисунок В.54 – Поиск записи

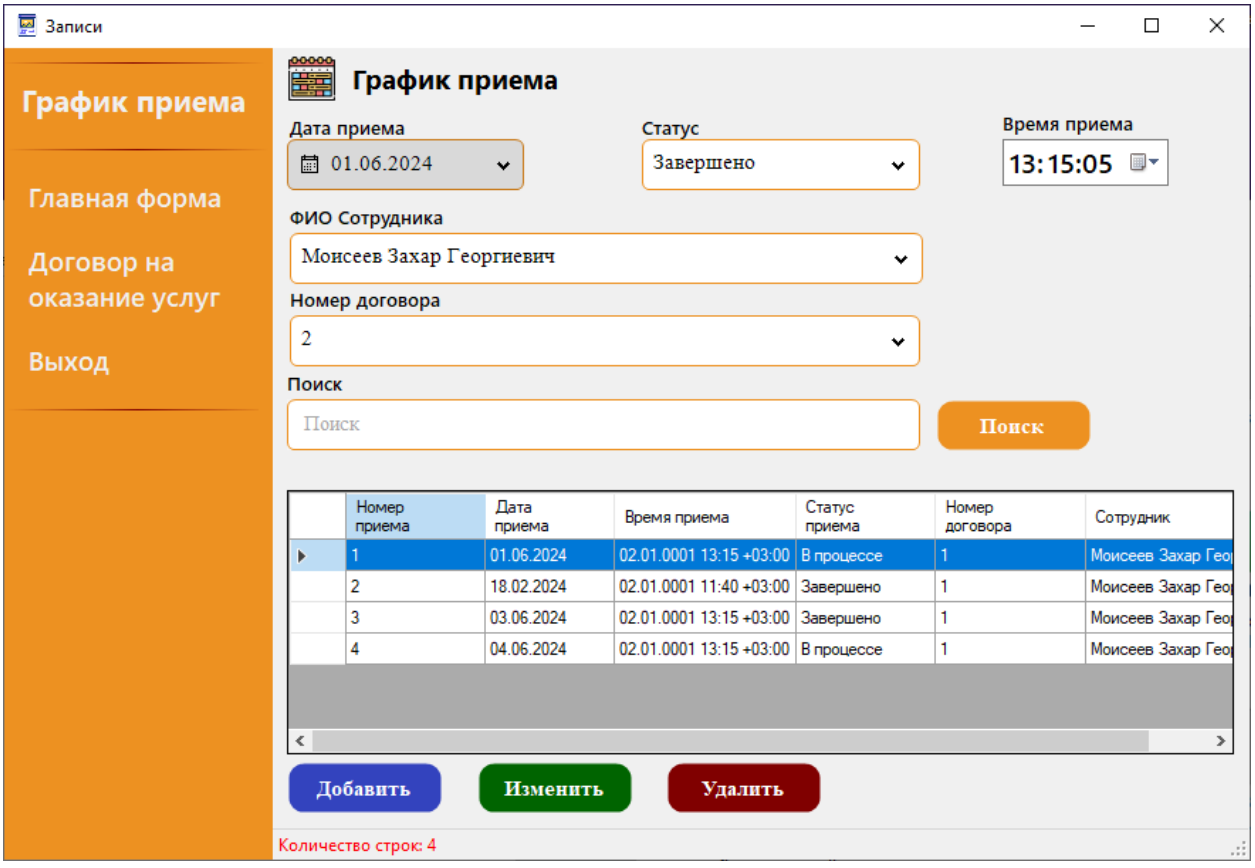


Рисунок В.55 – Меню «График приема»

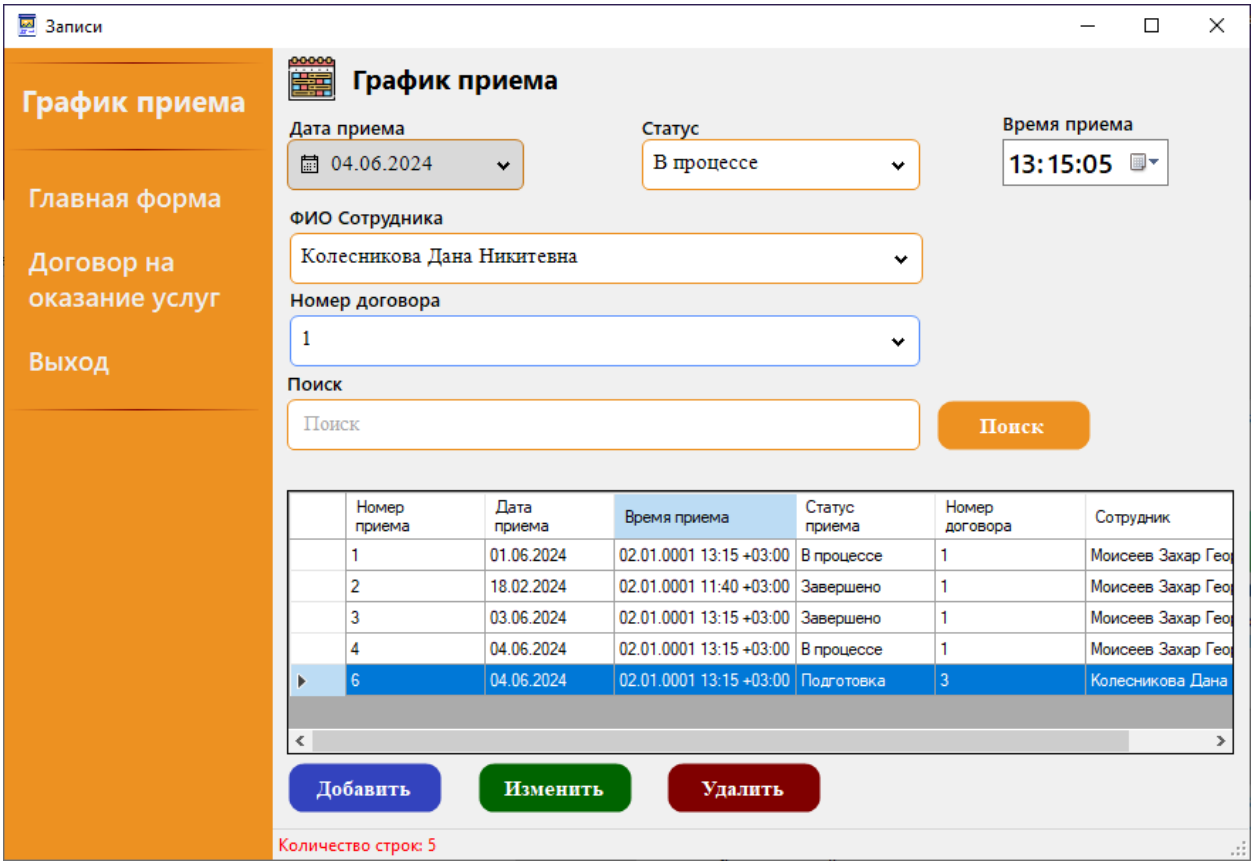


Рисунок В.56 – Добавление записи

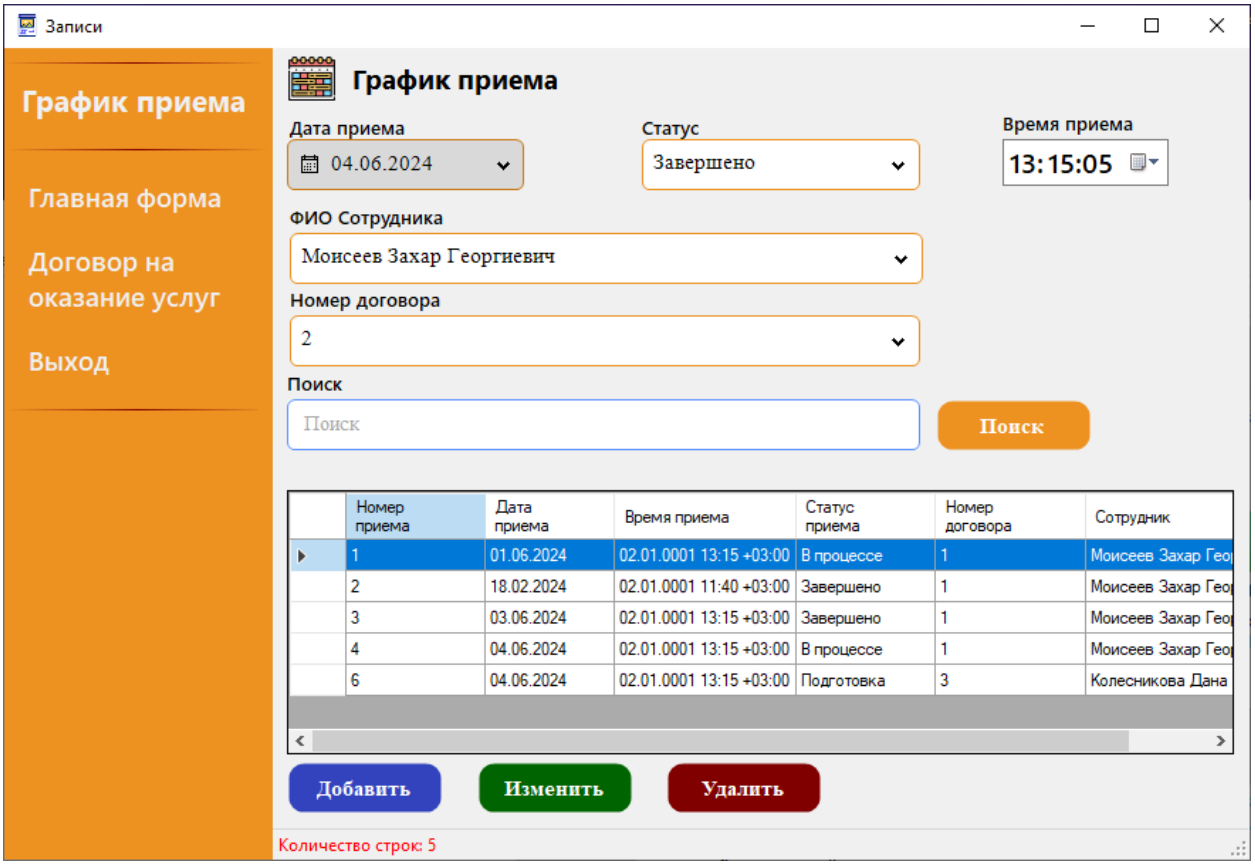


Рисунок В.57 – Изменение данных

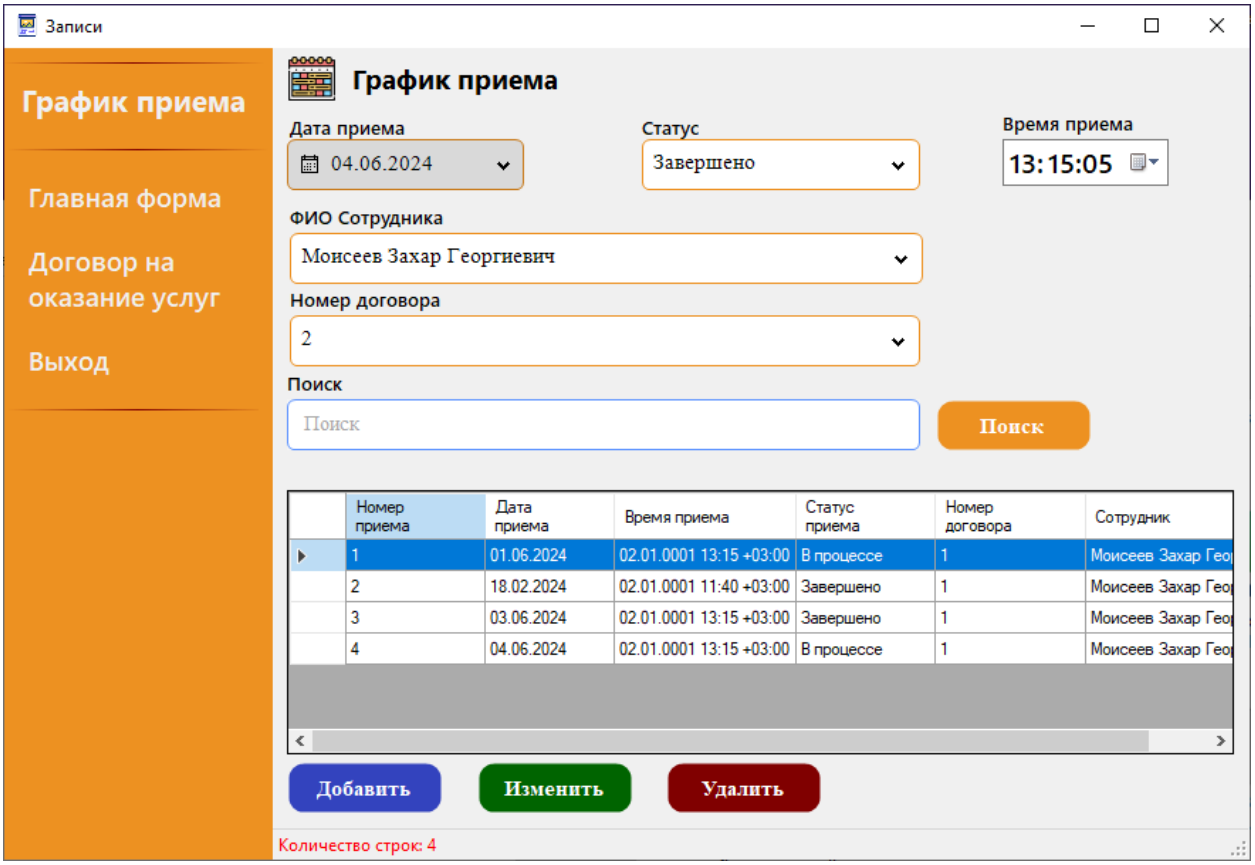


Рисунок В.58 – Удаление записи

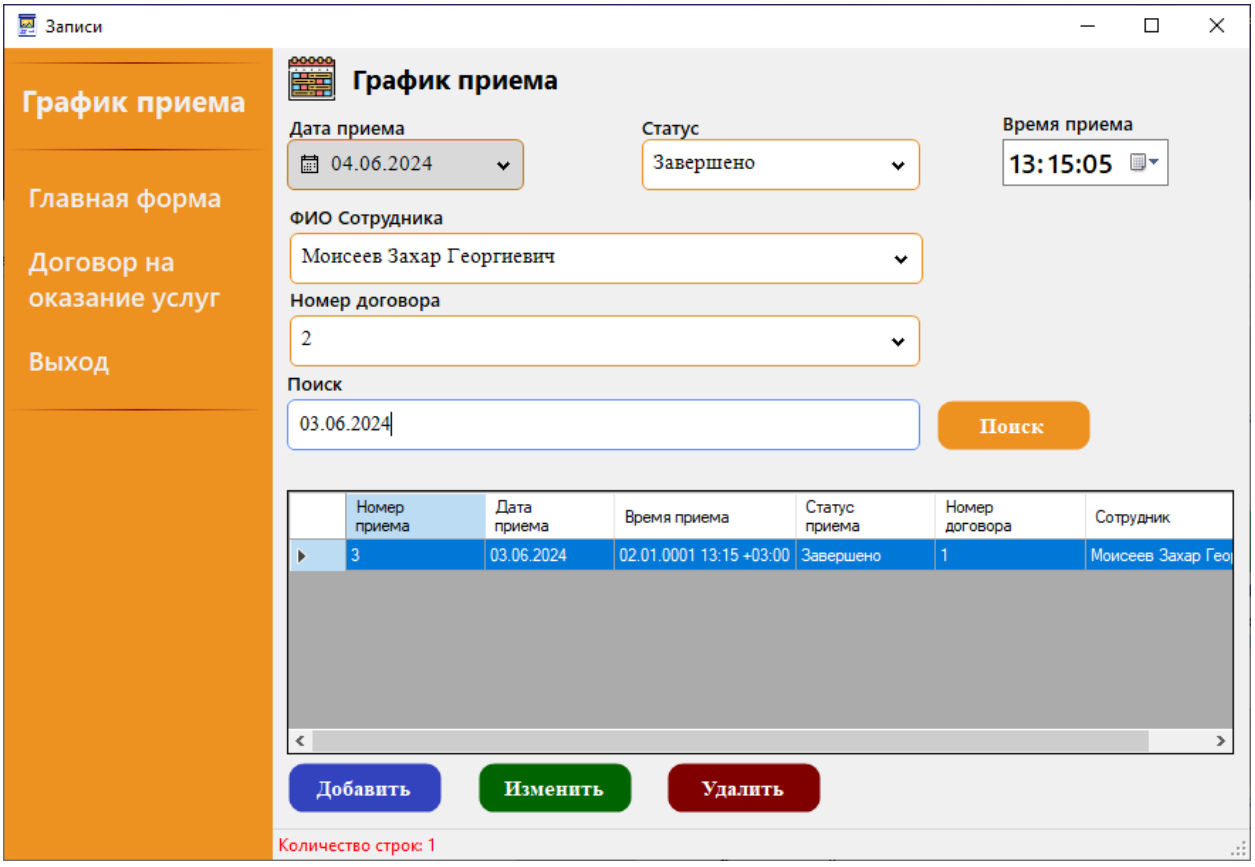


Рисунок В.59 – Поиск информации

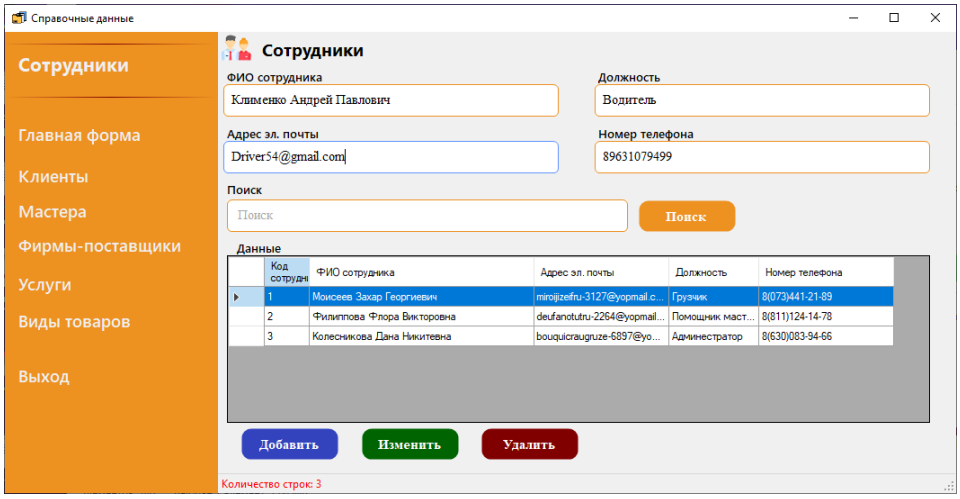


Рисунок В.60 – Форма «Справочные данные»

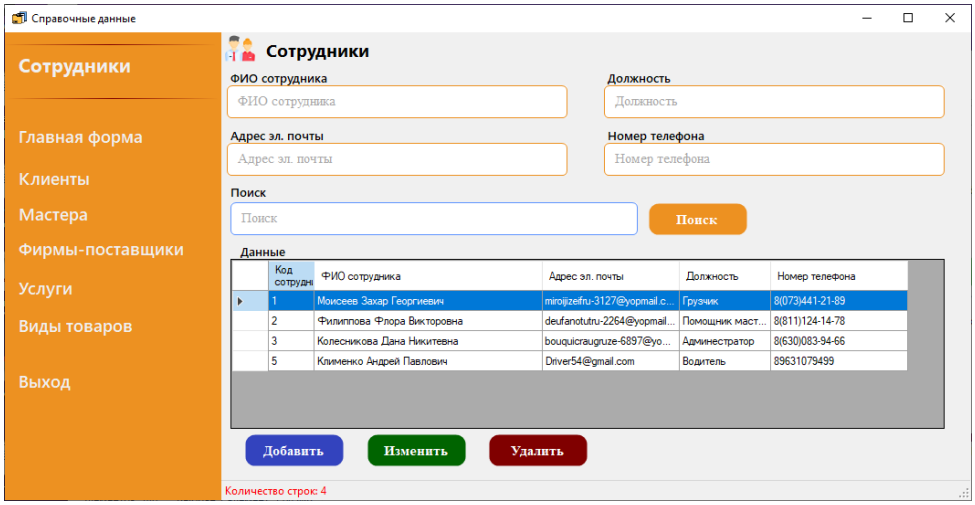


Рисунок В.61 – Добавление данных

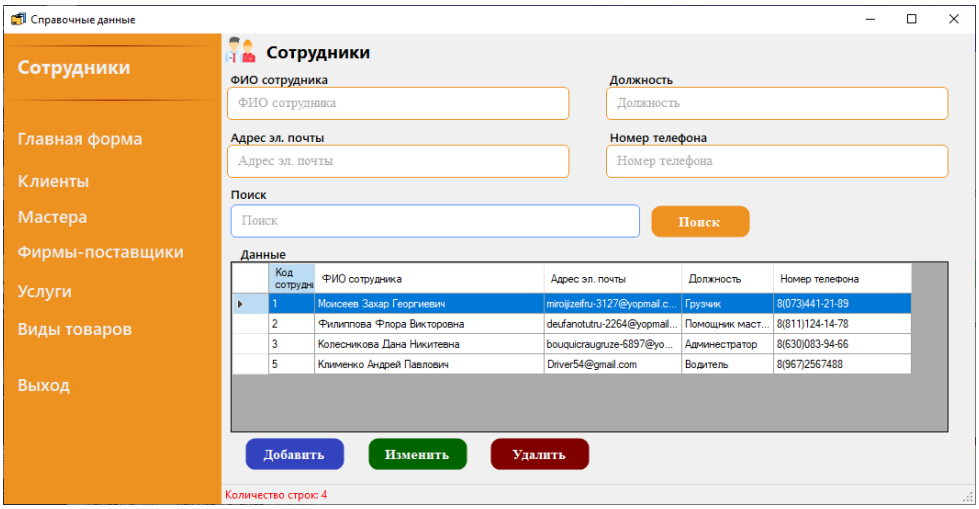


Рисунок В.62 – Изменение данных

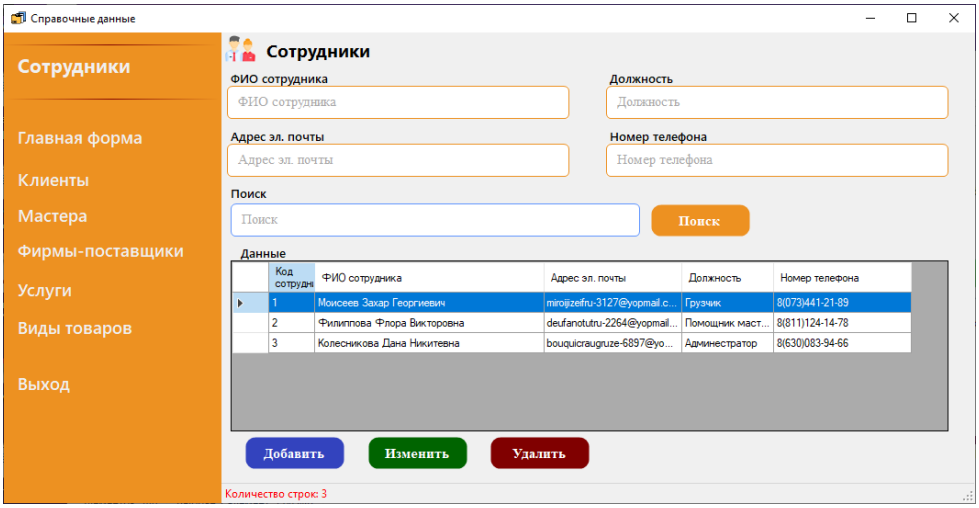


Рисунок В.63 -Удаление записи

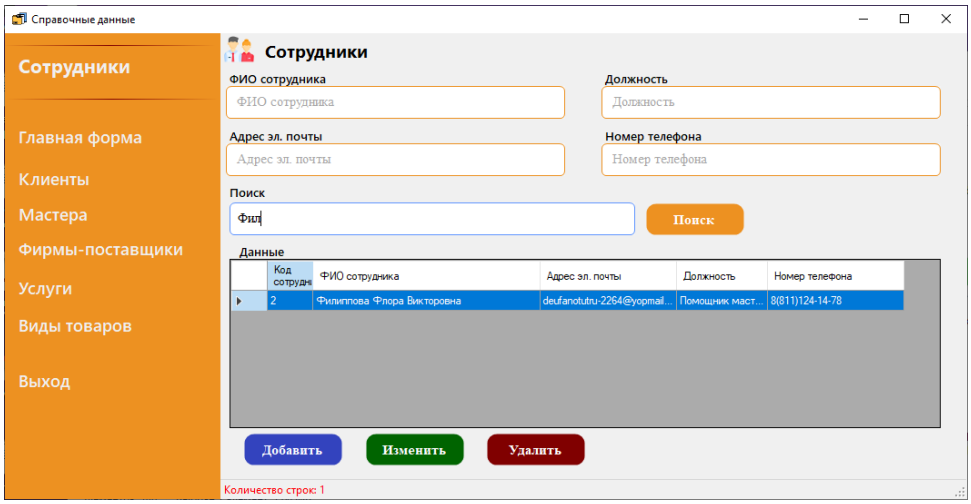


Рисунок В.64 – Поиск записи

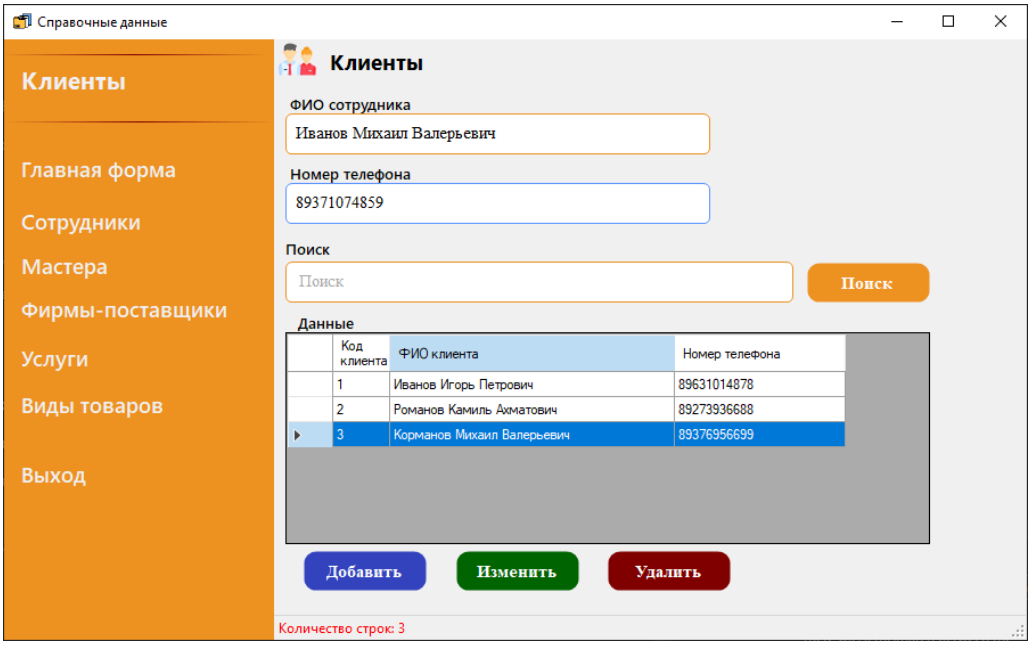


Рисунок В.65 – Меню «Клиенты»

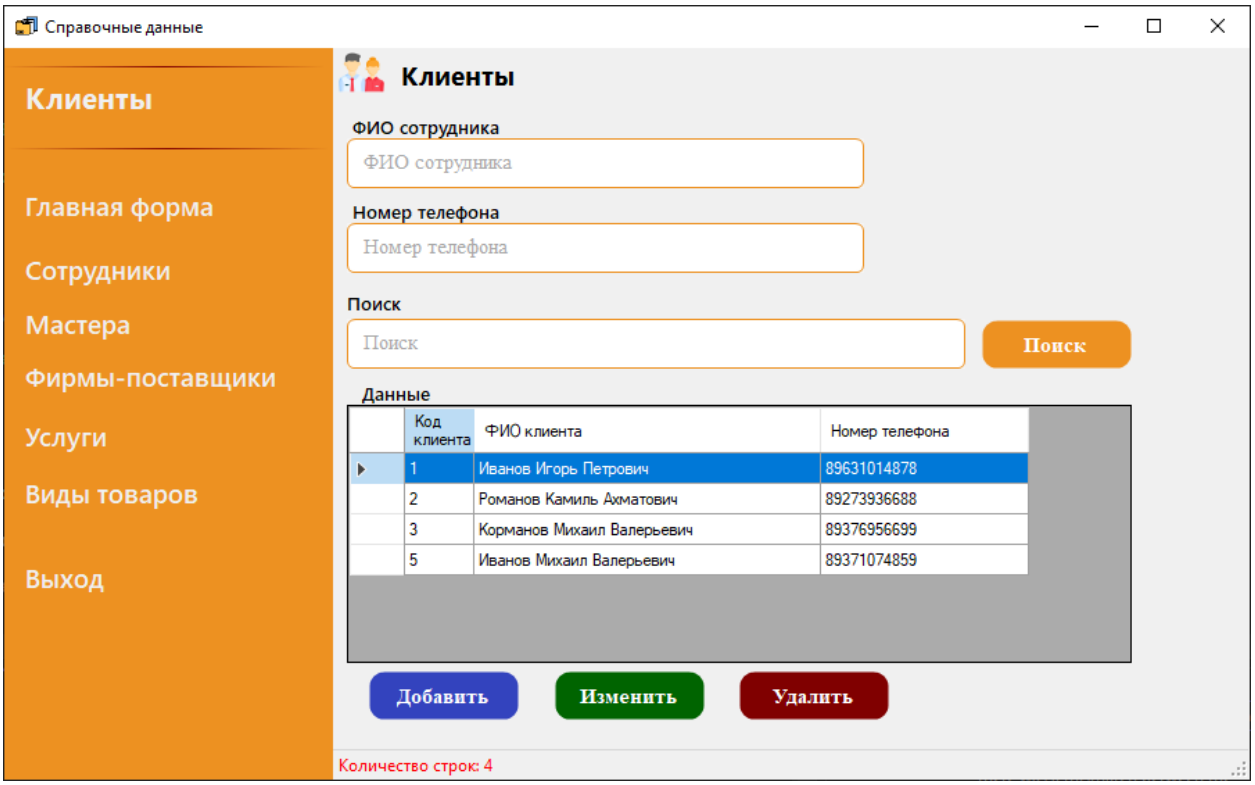


Рисунок В.66 – Добавление данных

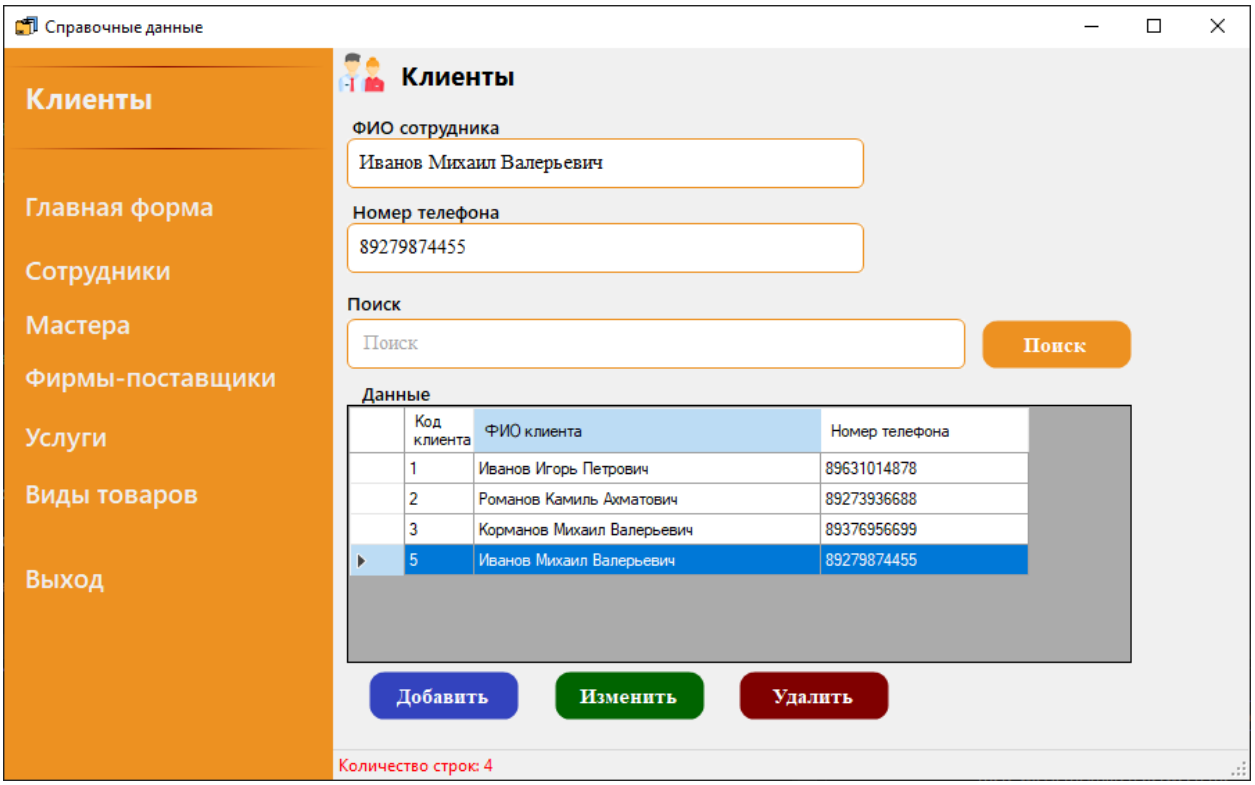


Рисунок В.67 – Изменение данных

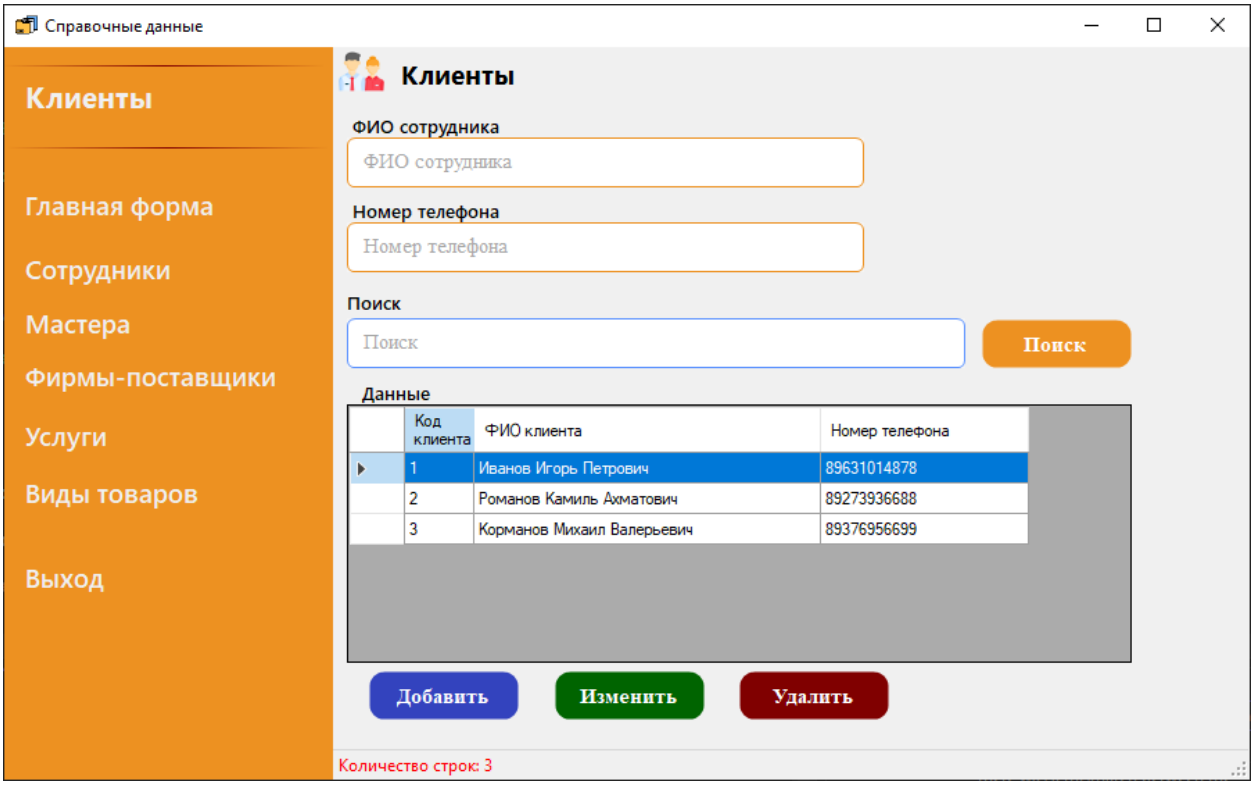


Рисунок В.68 – Удаление записи

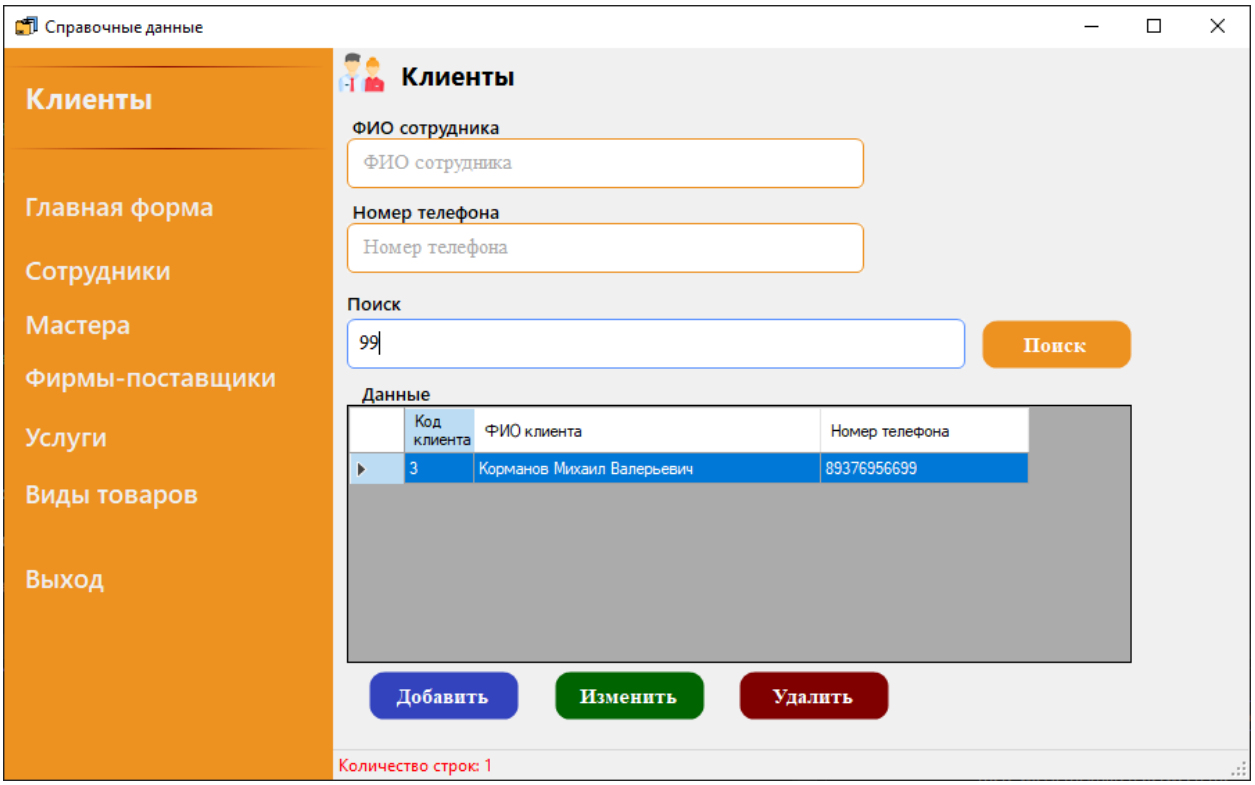


Рисунок В.69 – Поиск данных

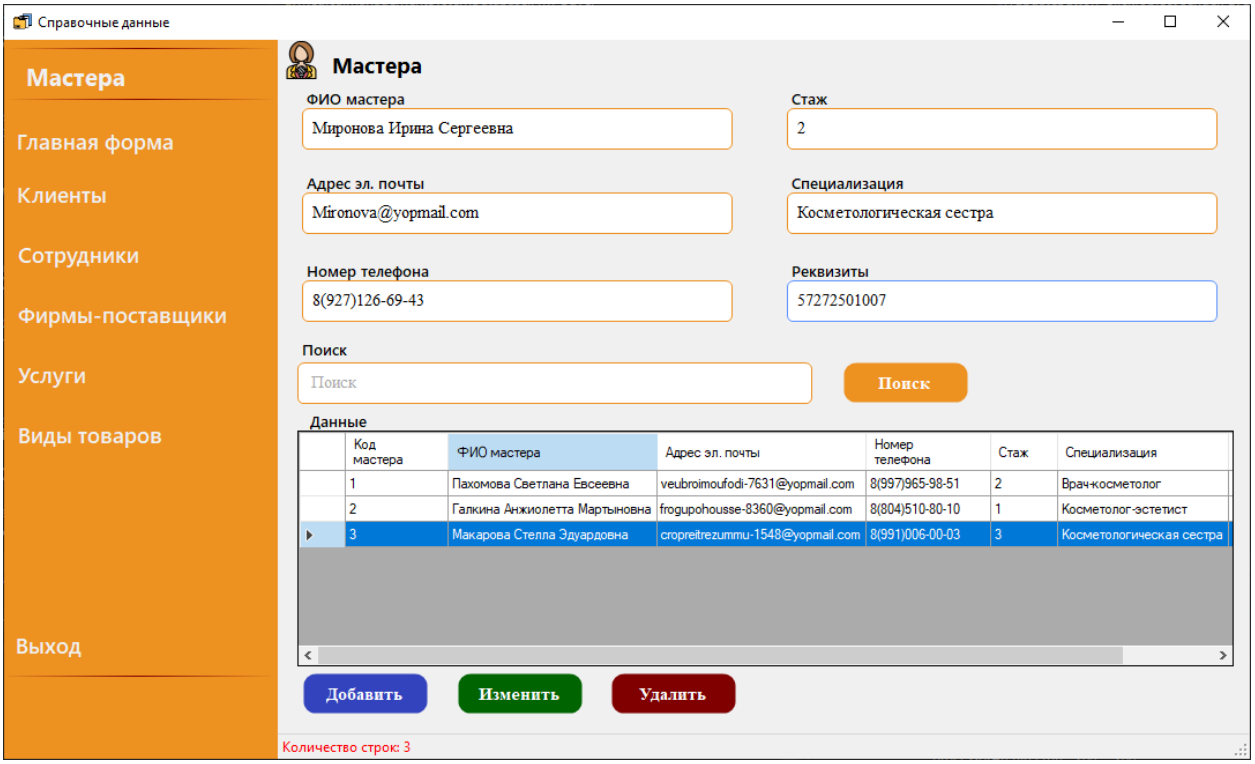


Рисунок В.70 – Меню «Мастера»

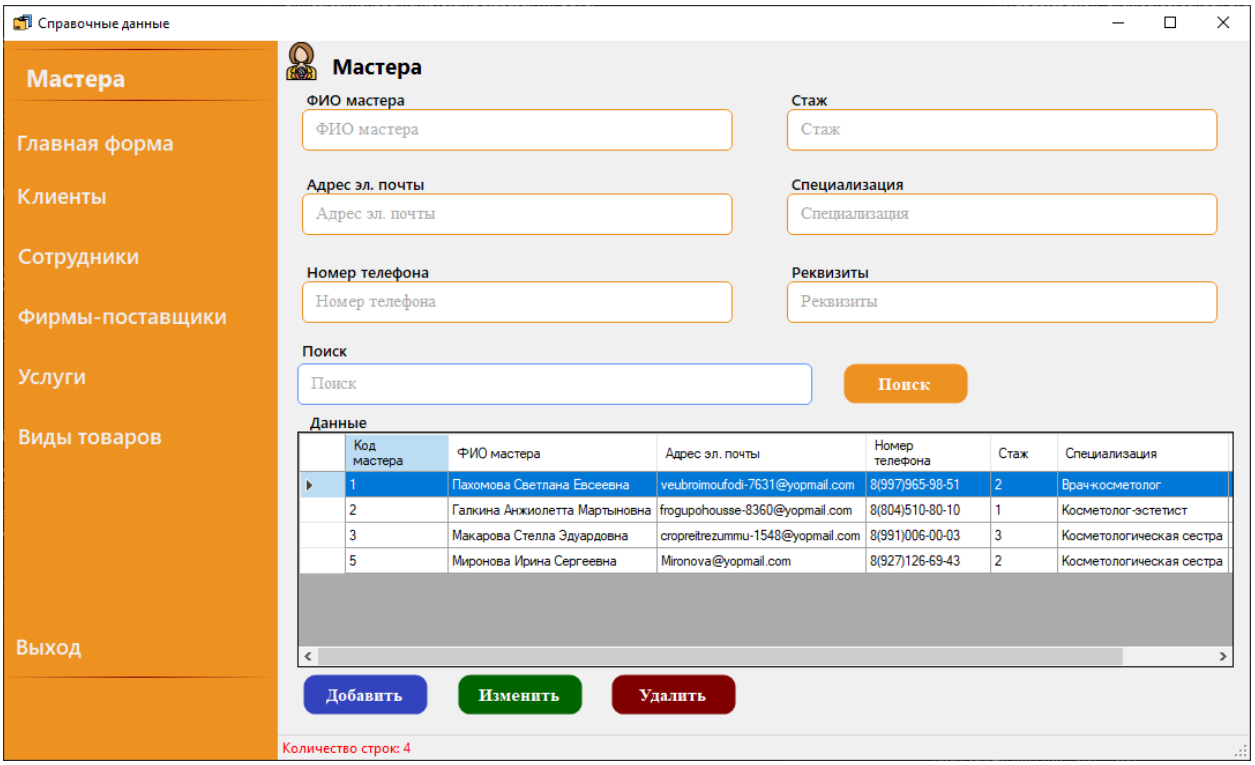


Рисунок В.71 – Добавление данных мастера

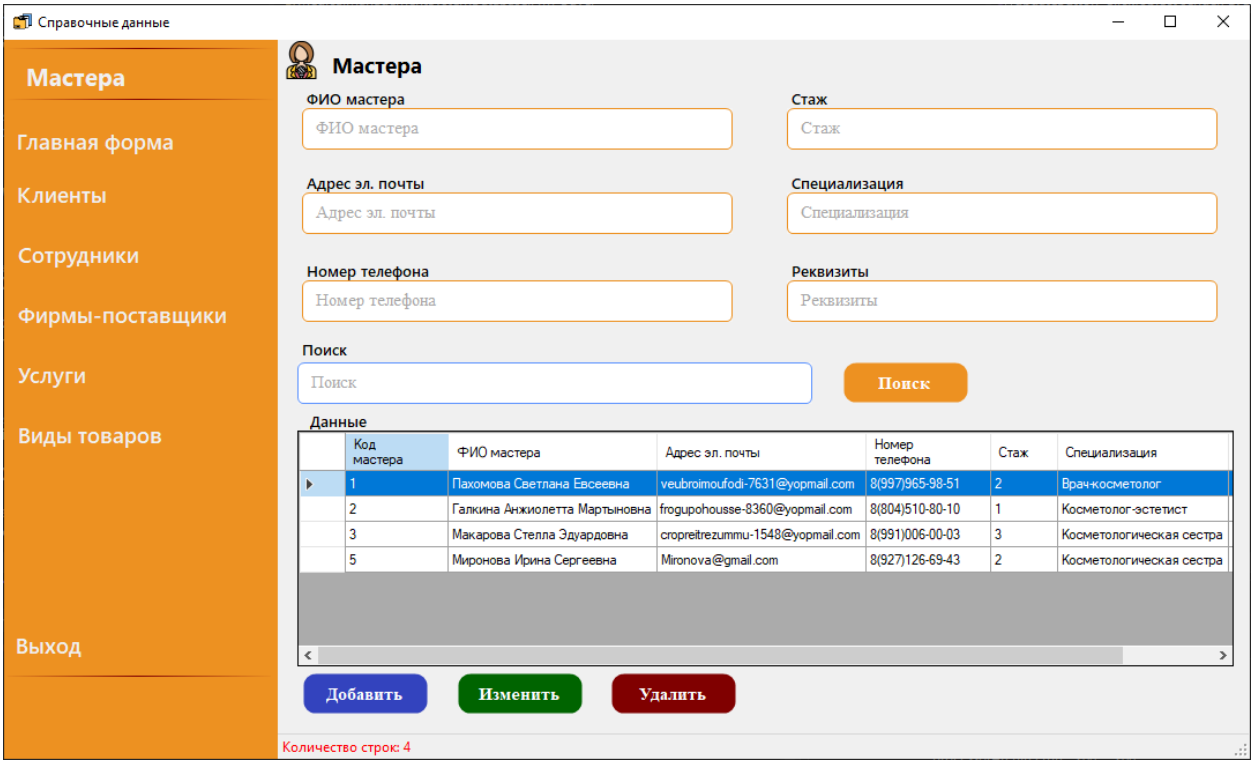


Рисунок В.72 – Изменение данных мастера

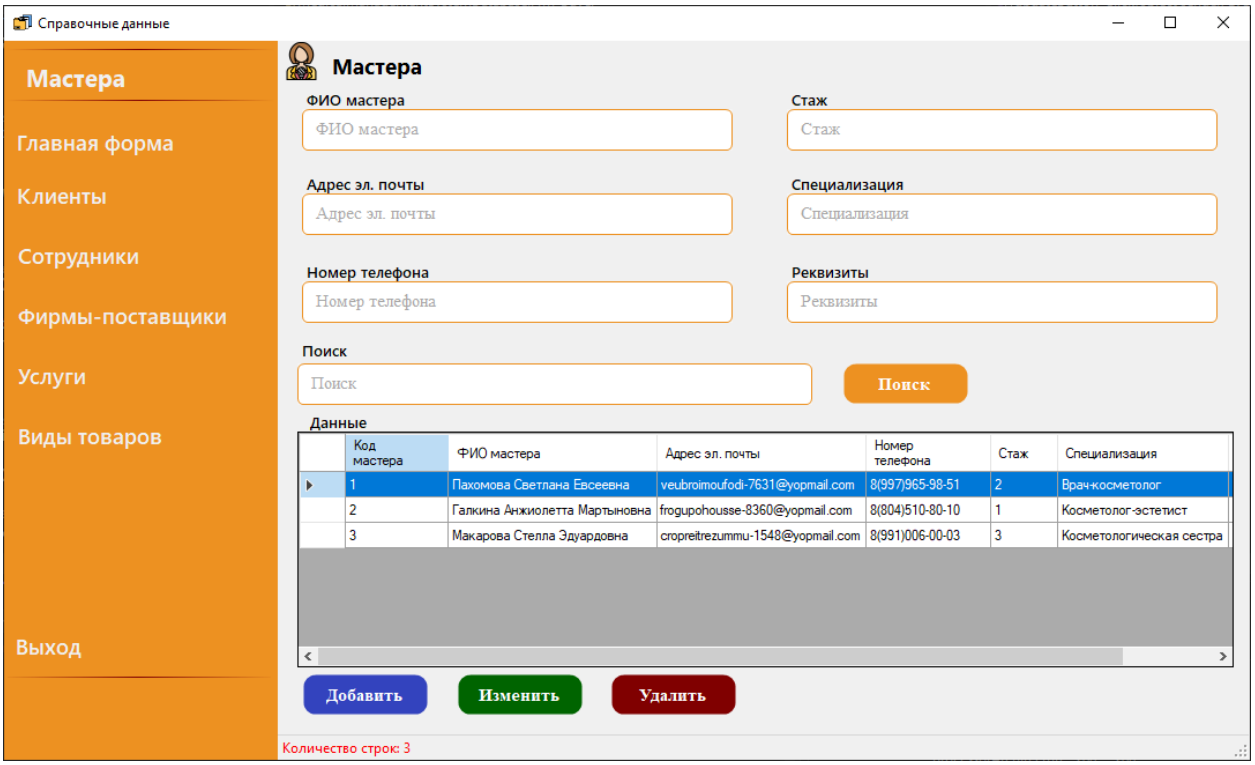


Рисунок В.73 – Удаление записи

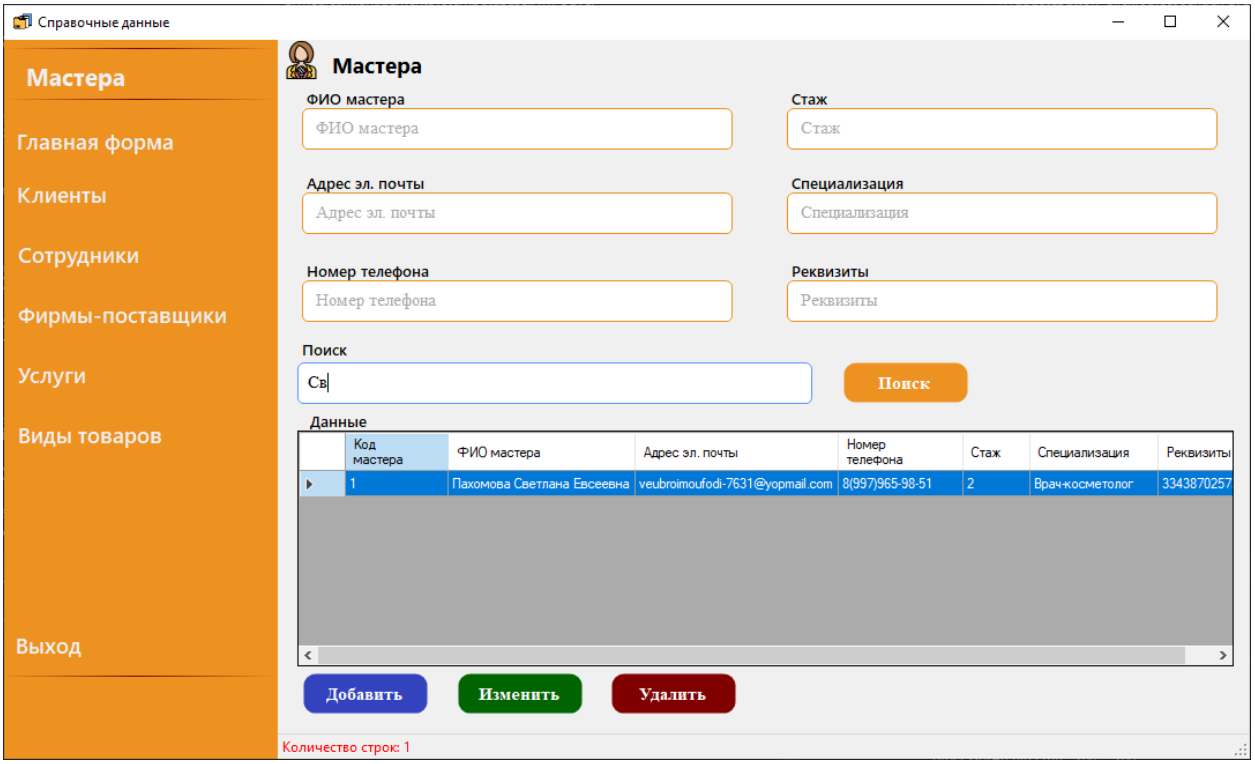


Рисунок В.74 – Поиск данных

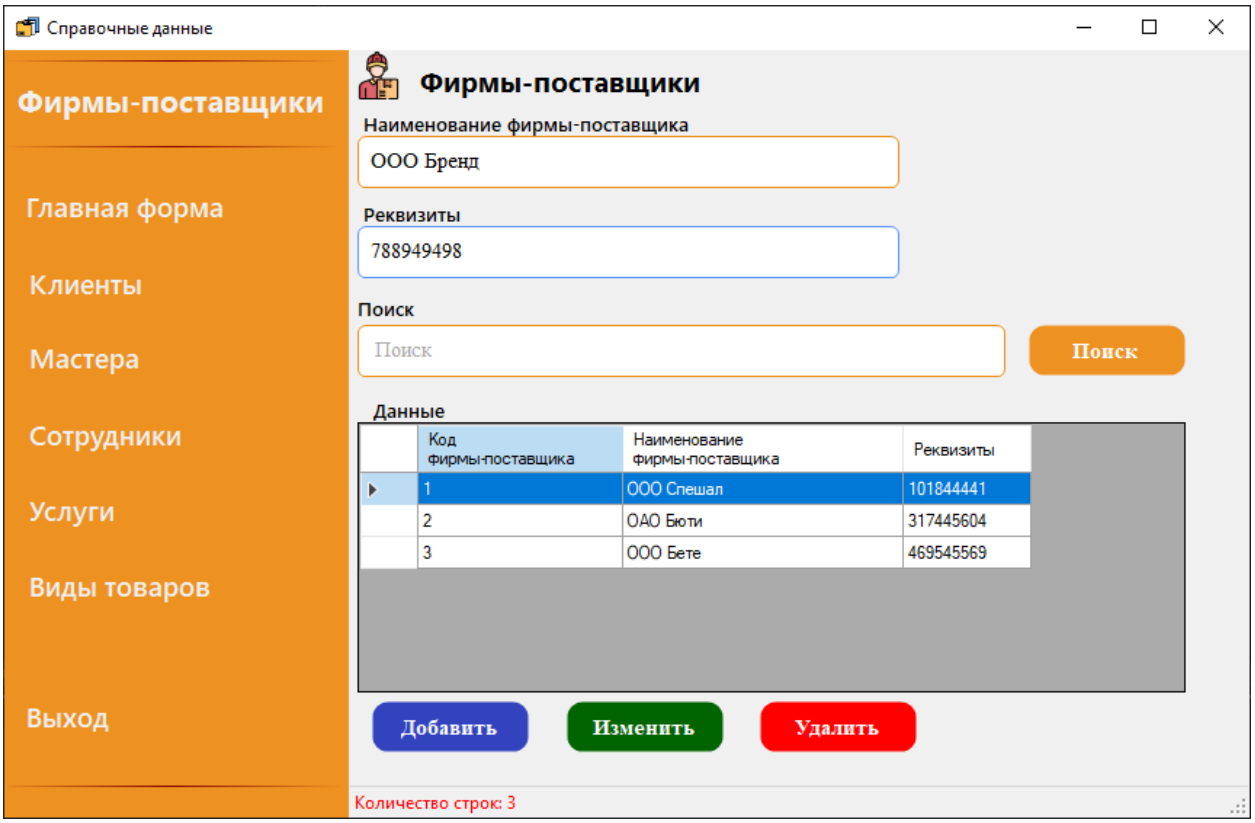


Рисунок В.75 – Меню «Фирмы-поставщики»

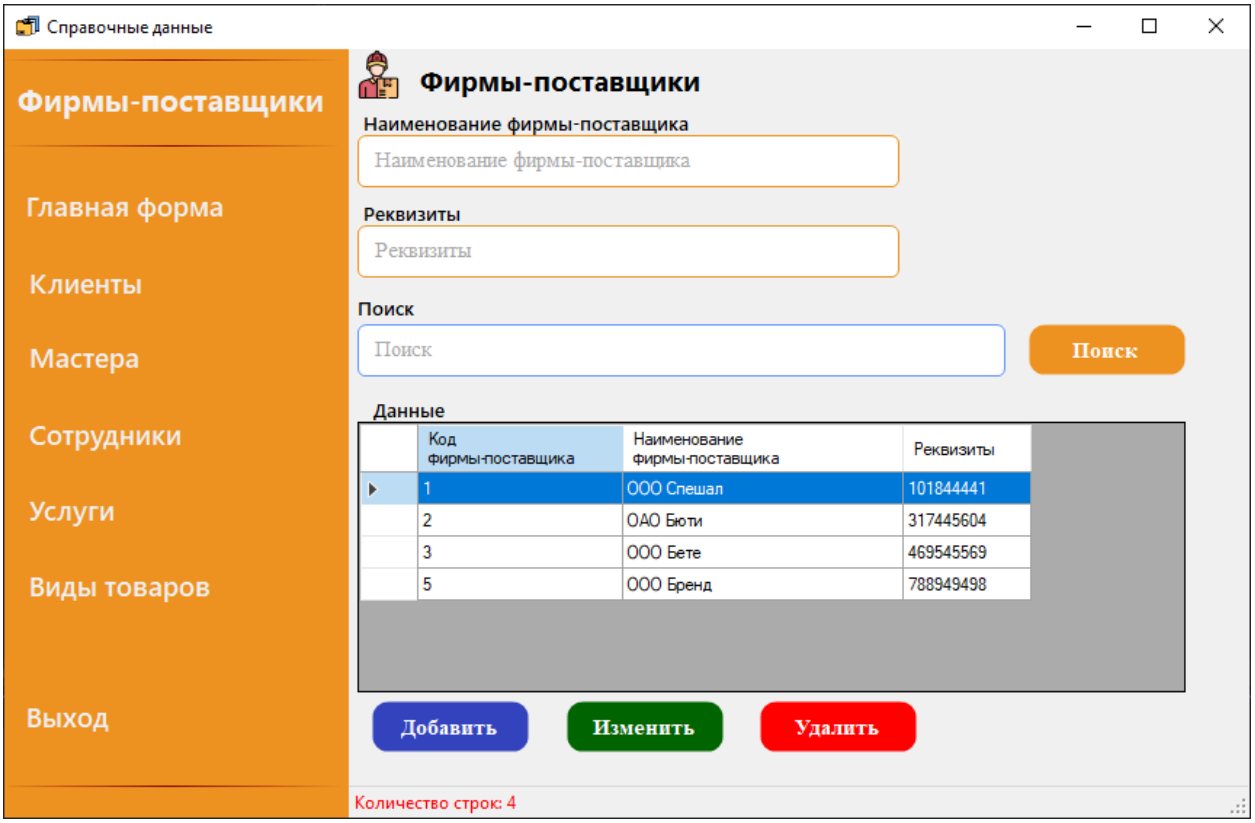


Рисунок В.76 – Добавление данных фирмы-поставщика

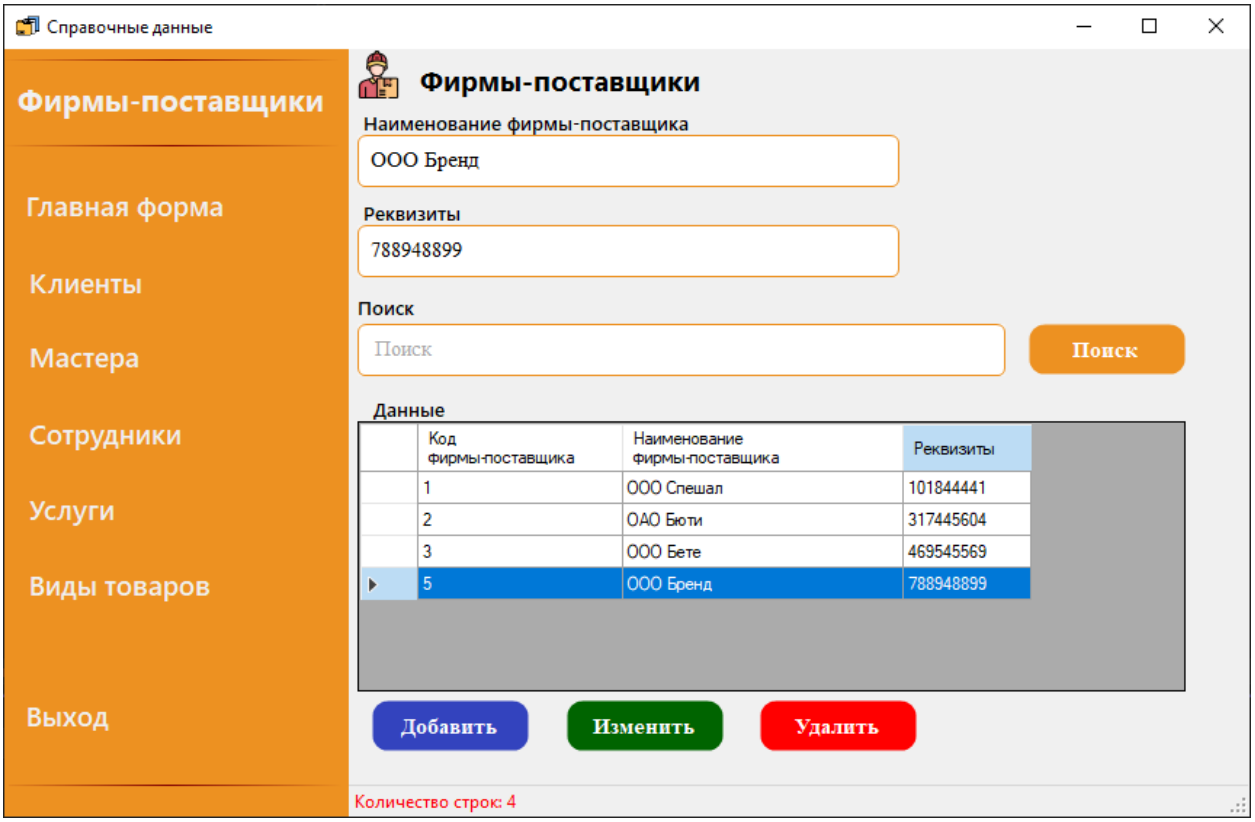


Рисунок В.77 – Изменение данных фирмы-поставщика

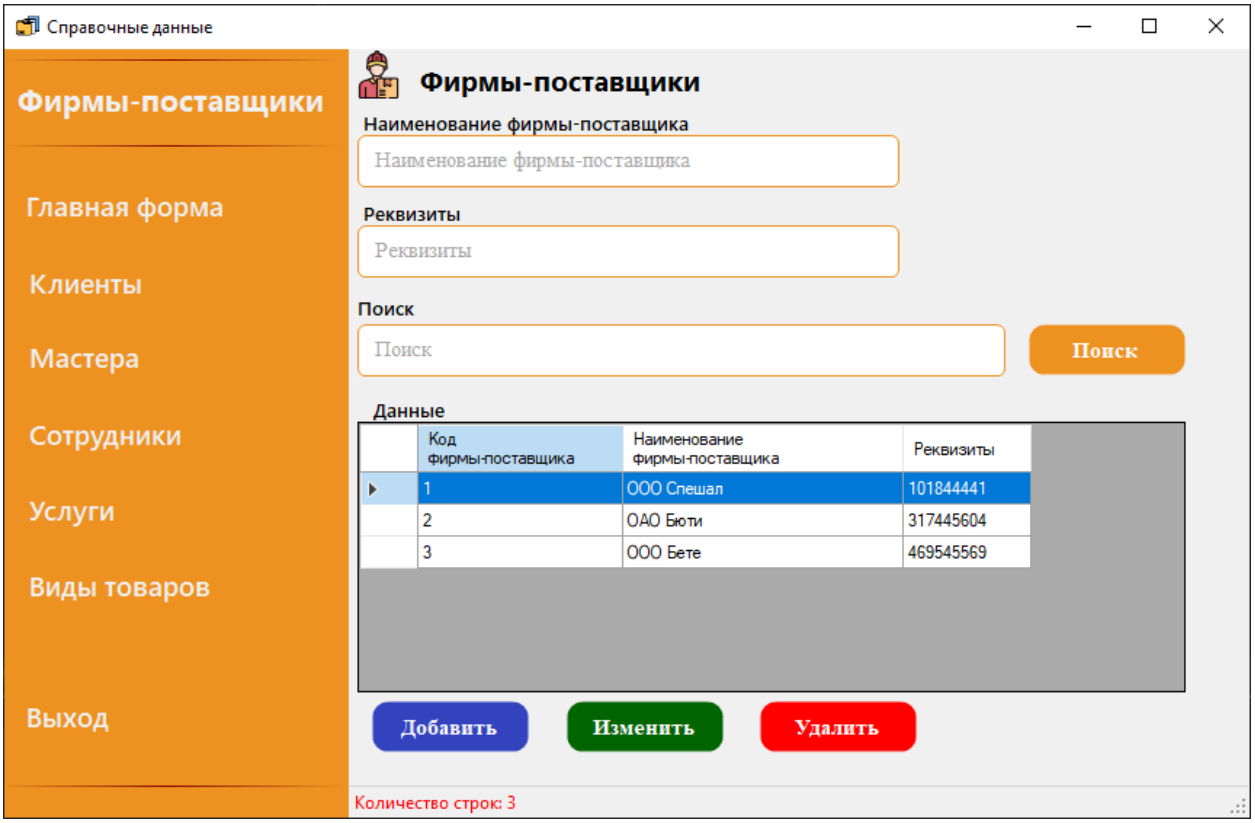


Рисунок В.78 – Удаление записи

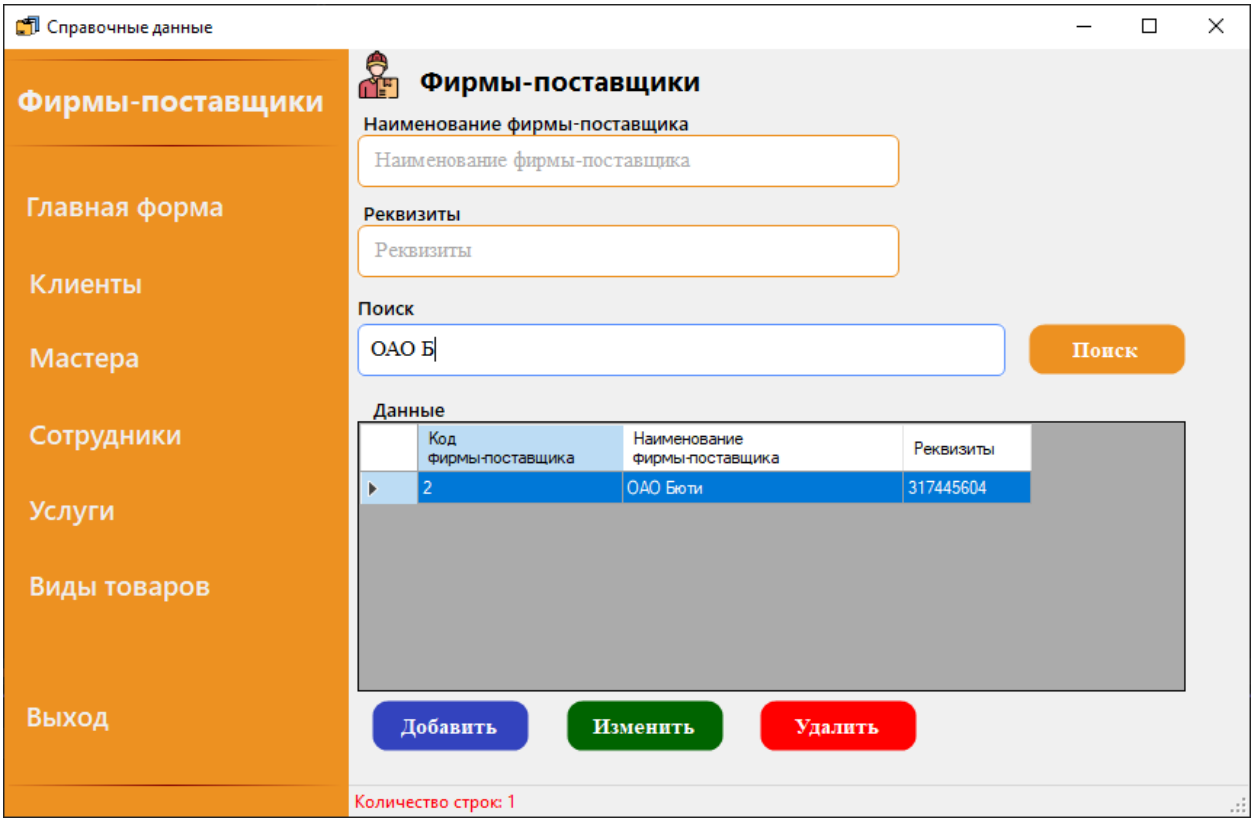


Рисунок В.79 – Поиск данных

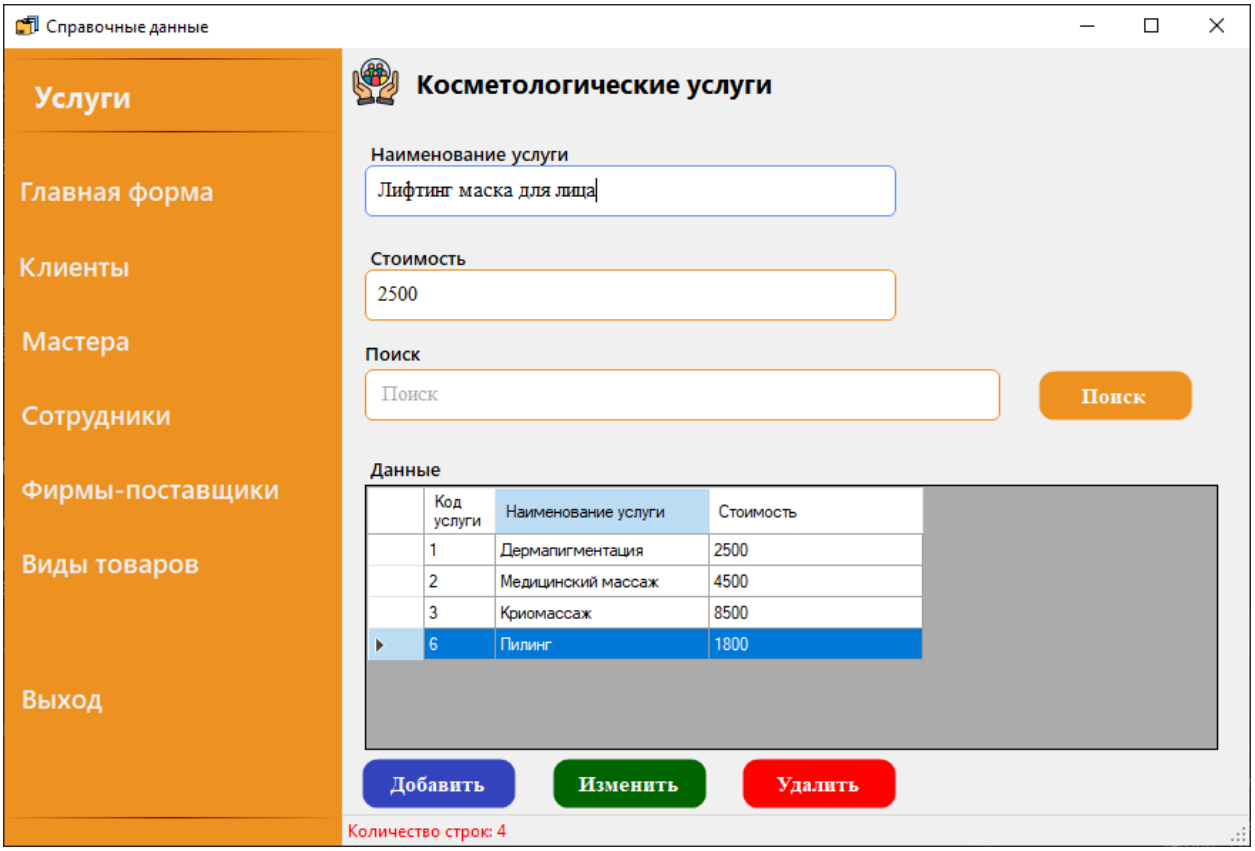


Рисунок В.80 – Меню «Услуги»

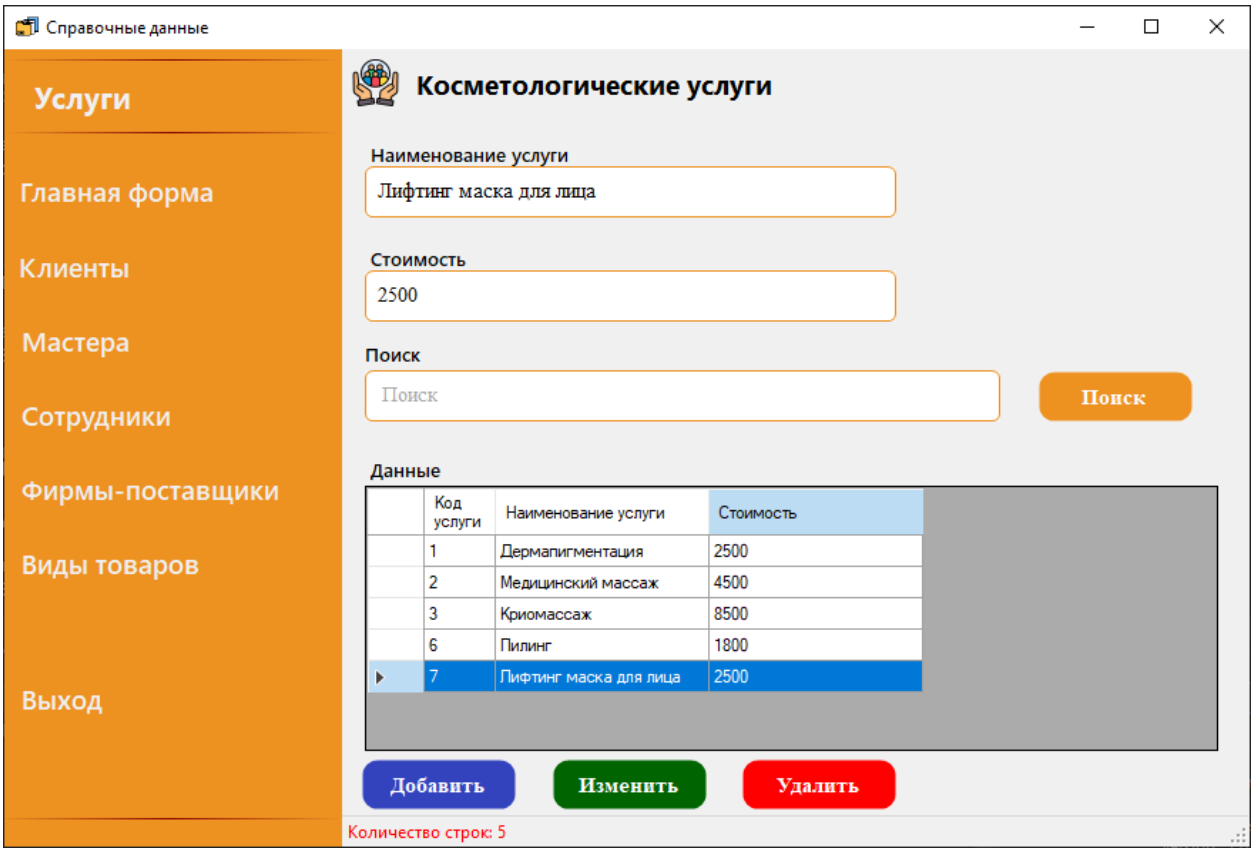


Рисунок В.81 – Добавление данных

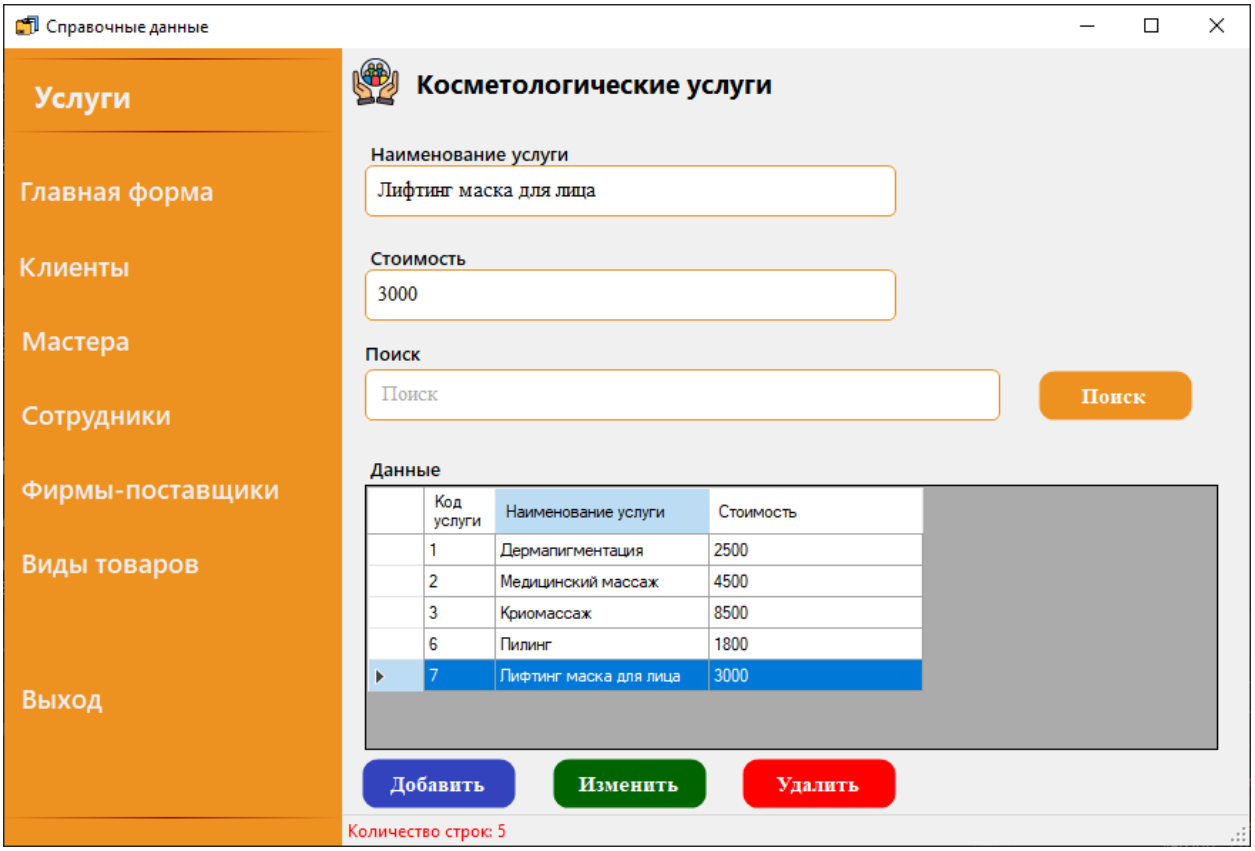


Рисунок В.82 – Изменение данных

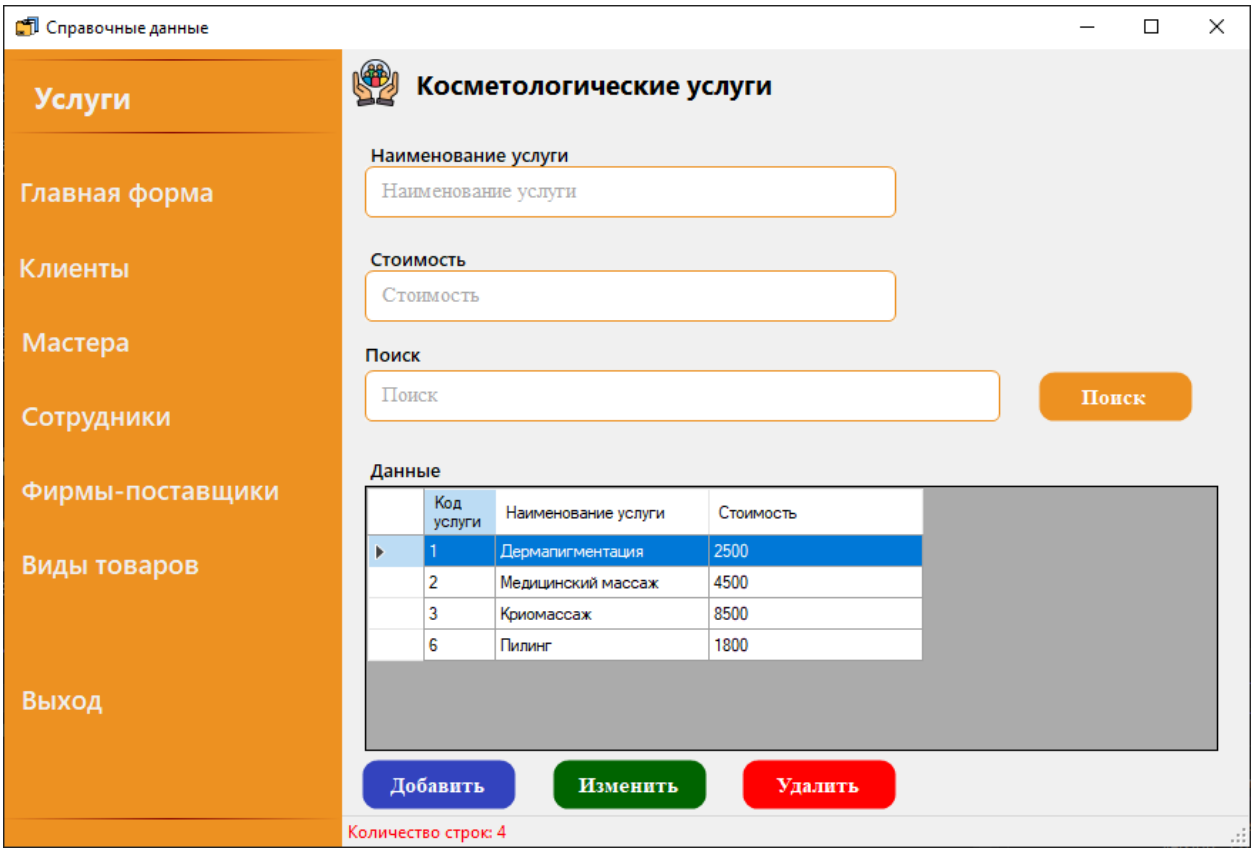


Рисунок В.83 – Удаление данных

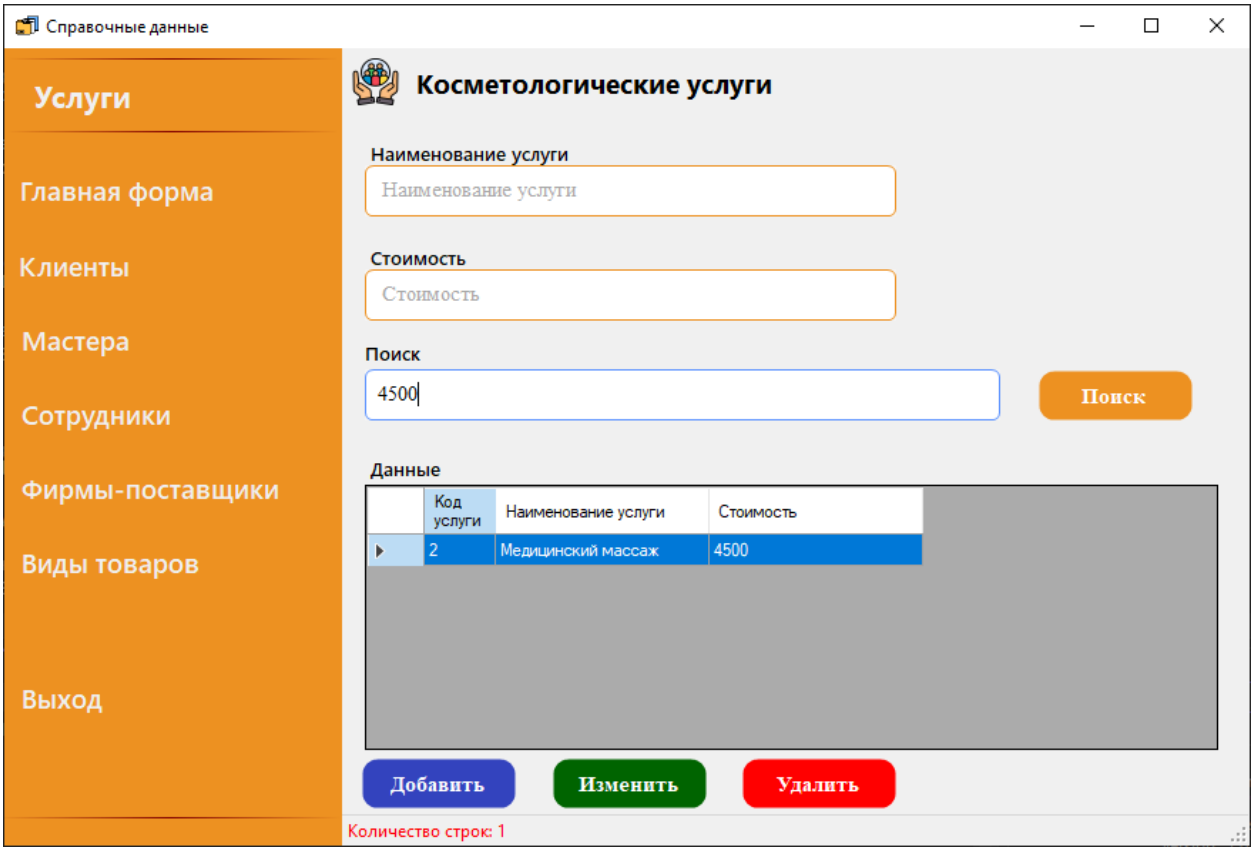


Рисунок В.84 – Поиск данных

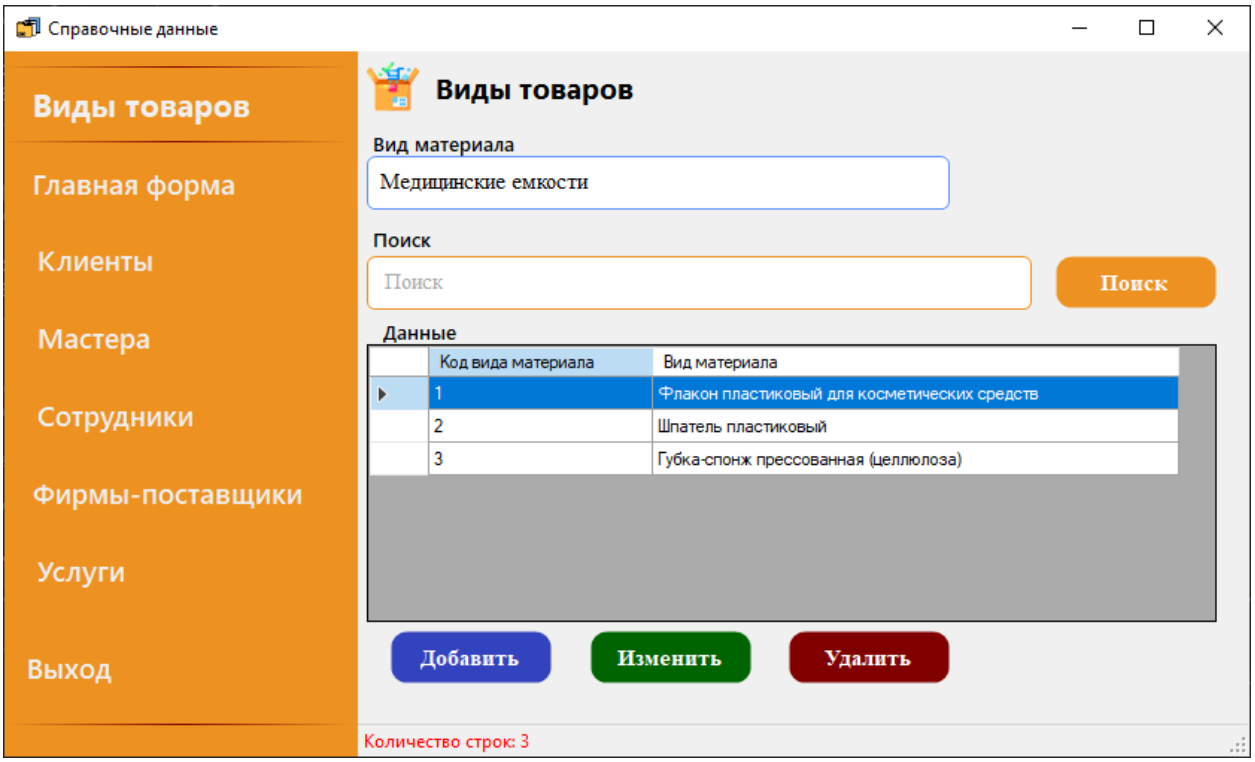


Рисунок В.85 – Меню «Виды товаров»

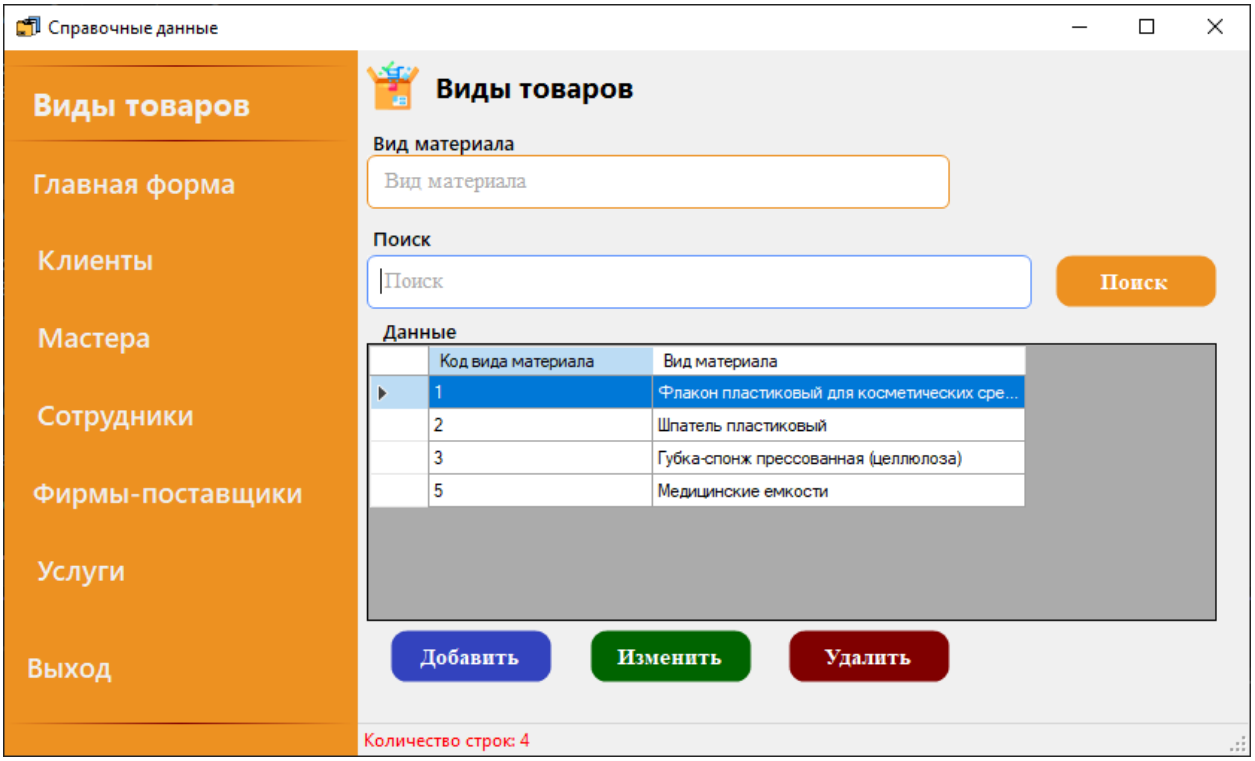


Рисунок В.86 – Добавление данных

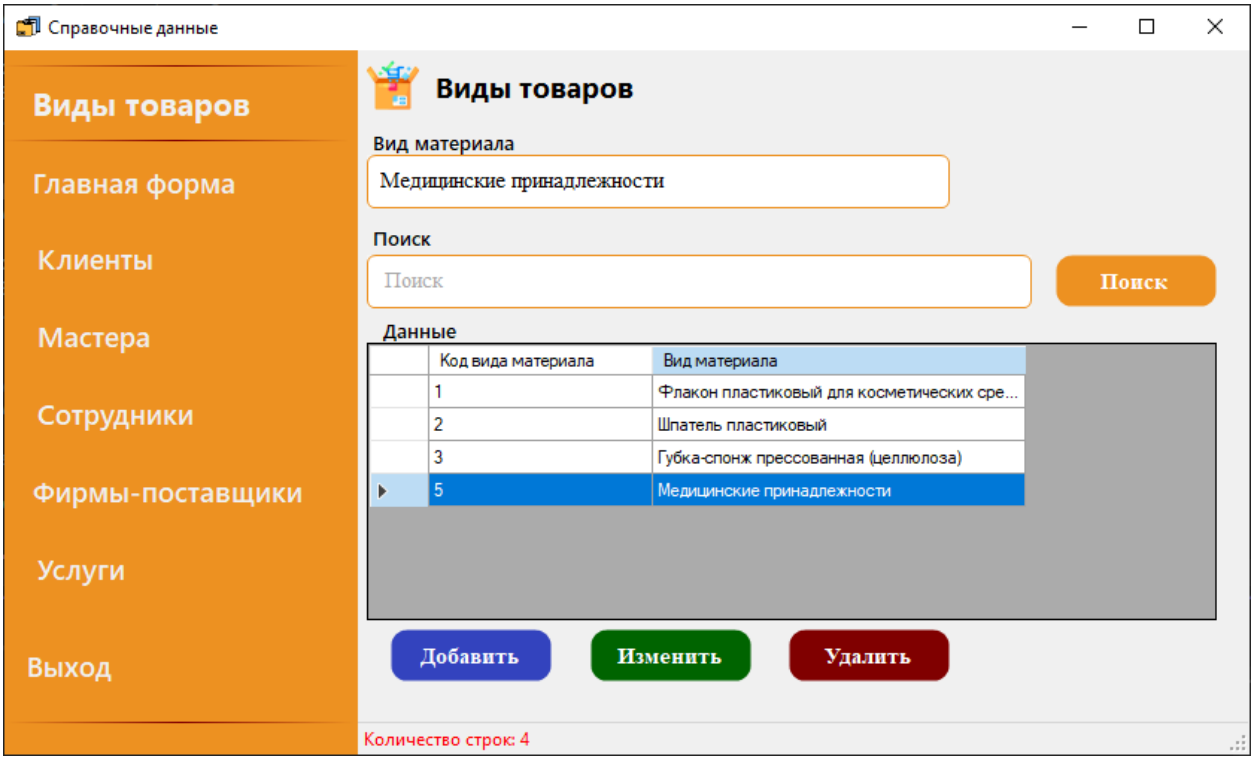


Рисунок В.87 – Изменение данных

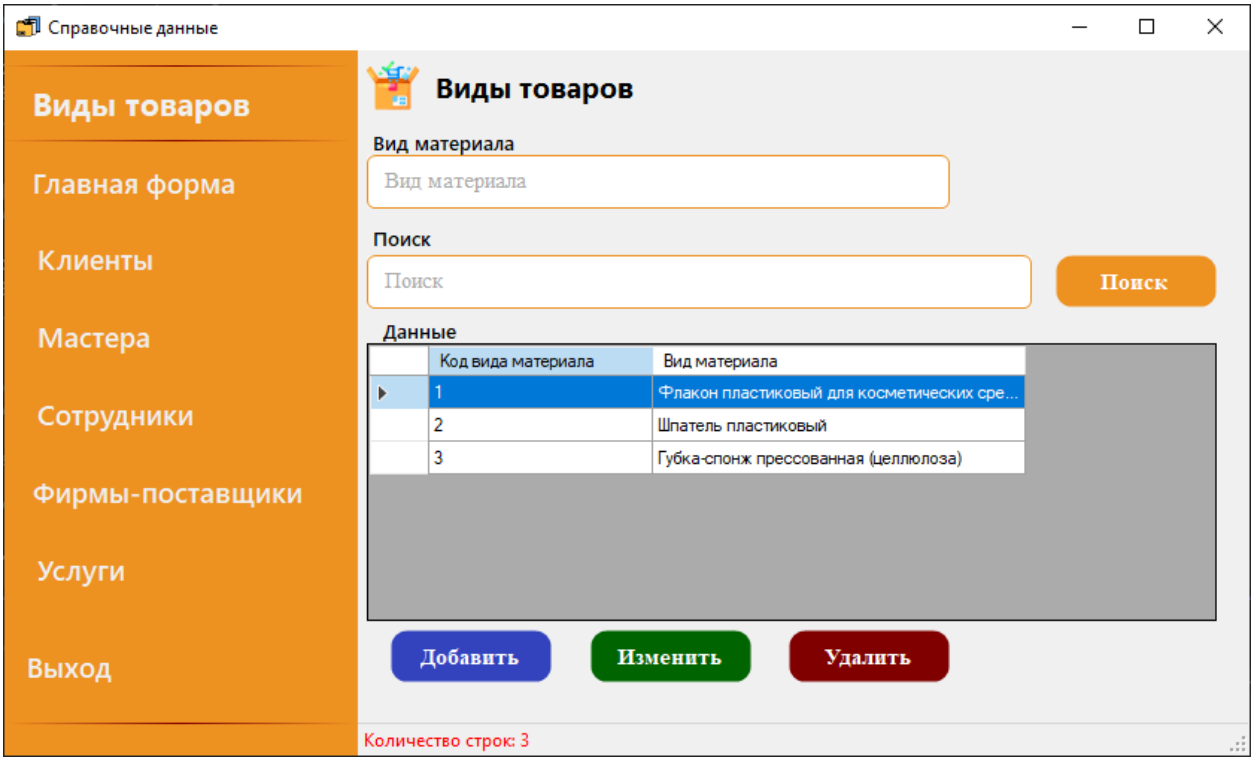


Рисунок В.88 – Удаление данных

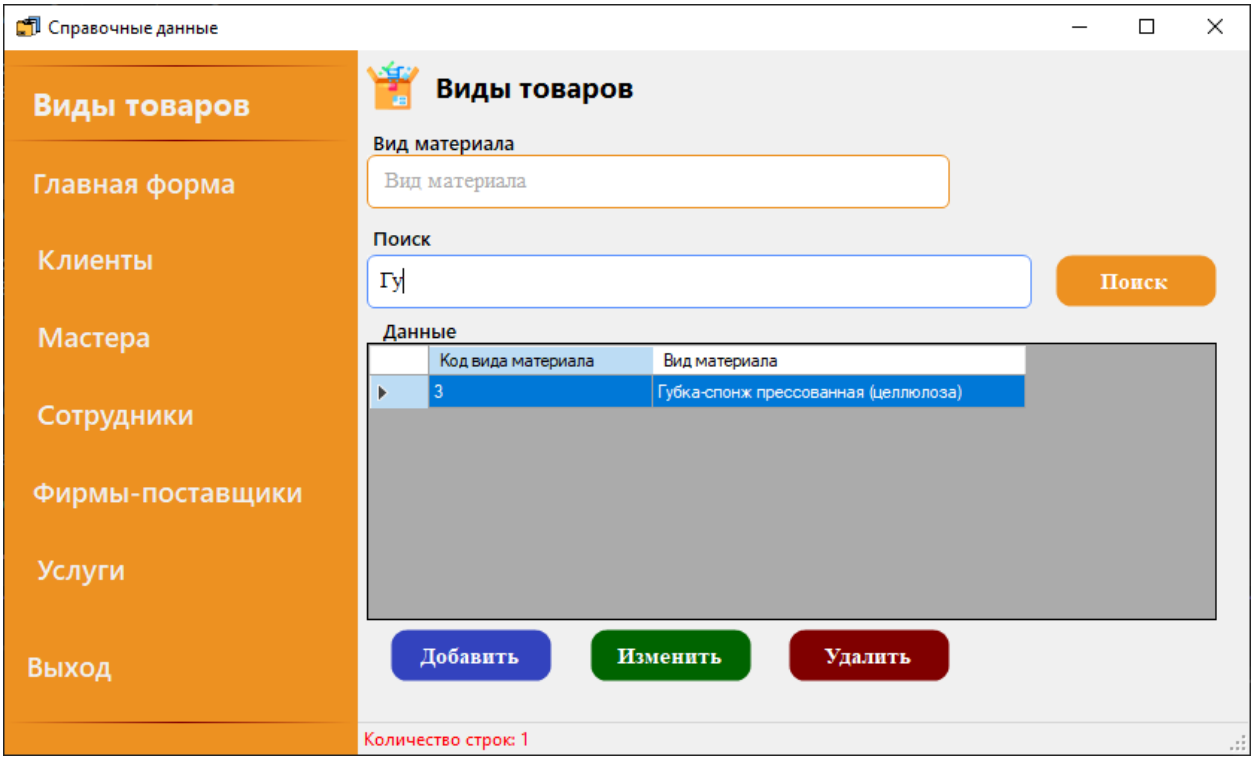


Рисунок В.89 – Поиск данных

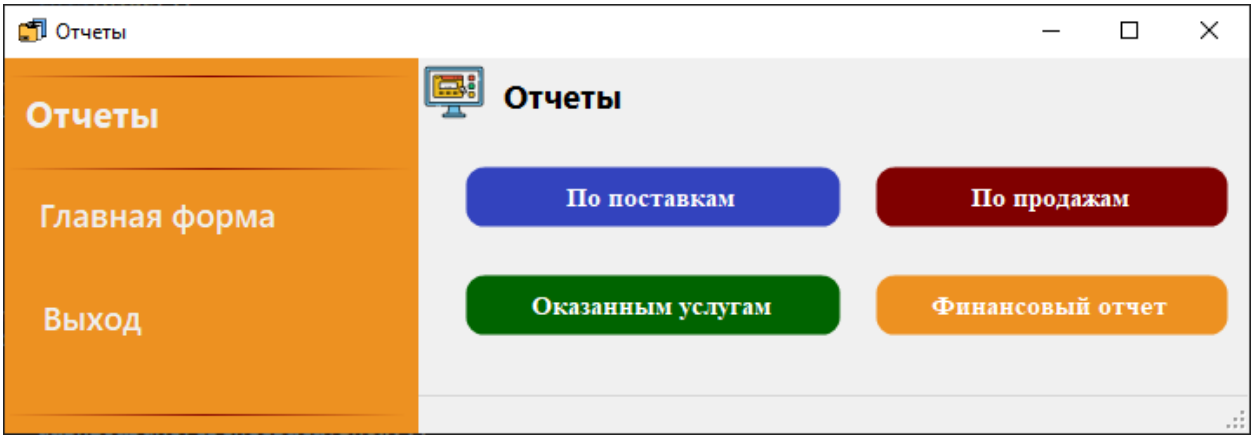


Рисунок В.90 – Меню «Отчеты»

Поставки

Период: с 01.04.2024 по 30.04.2024

Дата поставки	Фирма - поставщик	Код	Наименование материала	Цена, руб.	Количество	Стоимость
24.04.2024	ООО Бюти	8	Перчатки винил “Lite”	100	20	2000
24.04.2024	ООО Бюти	5	Простыни в рулоне “Lite”	1080	5	5400
24.04.2024	ООО Бете	4	Аппаратный гель “X01”	10000	1	10000

Материально-ответственное лицо _____ Моисеев Захар Георгиевич

(подпись) (Ф.И.О)

Рисунок В.91 – Отчет по Поставкам

Продажи за 24.04.2024

№	Наименование материала	Количество	Стоимость
1	Перчатки винил “Lite”	20	2000
2	Простыни в рулоне “Lite”	5	5400
3	Аппаратный гель “X01”	1	10000

Итого 17400

Сотрудник Продавец-консультант _____ Моисеев Захар Георгиевич

(должность) (подпись) (Ф.И.О)

Рисунок В.92 – Отчет по Продажам

Оказанные услуги

Период: с 01.04.2024 по 30.04.2024

Дата приема	Время приема	Мастер	Оказанная услуга	Стоимость, руб.	Ответственный сотрудник
07.04.2024	11:40	Пахомова Светлана Евсеевна	Медицинский массаж	4500	Колесникова Дана Никитевна
07.04.2024	11:45	Макарова Стелла Эдуардовна	Дермапигментация	2500	Филиппова Флора Викторовна
07.04.2024	11:50	Галкина Анжиолетта Мартыновна	Криомассаж	8500	Колесникова Дана Никитевна

Рисунок В.93 – Отчет по Оказанным услугам

Финансовый отчет

с 01.01.2024 по 01.04.2024

Организация _____ Белла _____

Наименование операции	Итог
Доходы	
Продажа и оказание услуг № 1	134000
Продажа и оказание услуг № 2	96000
Продажа и оказание услуг № 3	115900
Итого	**345900**
Расходы	
Поставка товаров № 1	92000
Поставка товаров № 2	84000
Поставка товаров № 3	117300
Итого	**293300**
Финансовый результат	**52600**

Финансовый результат за период с 01.01.2024 по 01.04.2024 является прибыль в размере 52600 р.

Рисунок В.94 – Финансовый отчет

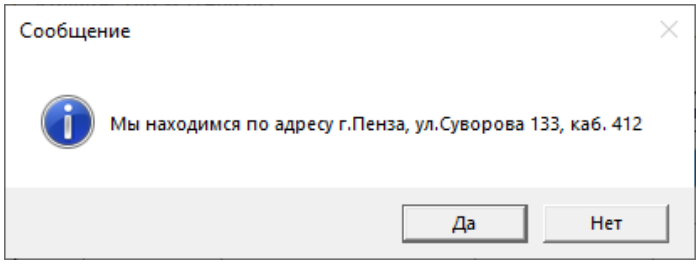


Рисунок В.95 – Сообщение «Данные салона»

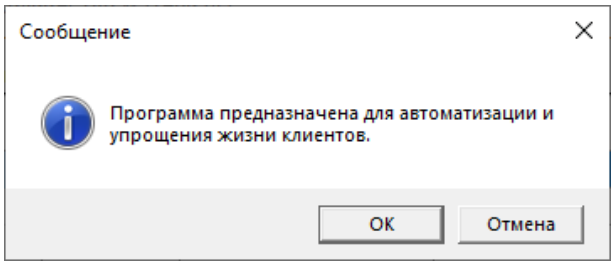


Рисунок В.96 – Сообщение «О программе»

Приложение Г

(обязательное)

Информационная автоматизированная система салона красоты

Листинг программы

```
using Npgsql;
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 MedicalManagementSystemDatabase
{
    public partial class Frm_Buyer : Form
    {
        private string id = "";
        private int intRow = 0;
        public Frm_Buyer()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }
        private void resetMe()
        {
            this.id = string.Empty;

            fullnameTextBox.Text = "";
            phoneTextBox.Text = "";


            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }

        }

        private void loadData(string keyword)
        {

            CRUD.sql = "";

            CRUD.sql = "SELECT * FROM public.\"POKUPATEL\"\r\n " +
                "WHERE CONCAT(CAST(\"KOD_POKUPATEL\" as varchar),
' ', \"FIO_POKUPATEL\", " +
```

```
                " ' ', \"NOMER_TELEFONA\") " +
                "LIKE @keyword::varchar ORDER BY \"KOD_POKUPATEL\"
ASC ";


            string strKeyword = string.Format("%{0}%", keyword);

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;

            dgv1.Columns[0].HeaderText = "Код клиента";
            dgv1.Columns[1].HeaderText = "ФИО клиента";
            dgv1.Columns[2].HeaderText = "Номер телефона";

            dgv1.Columns[0].Width = 50;
            dgv1.Columns[1].Width = 250;
            dgv1.Columns[2].Width = 150;

        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
```

```
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardBuyers = new Frm_Buyer();
            dashBoardBuyers.Show();
            this.Hide();
        }

        private void Frm_Buyer_Load(object sender, EventArgs e)
        {
            loadData("");
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            CRUD.cmd.Parameters.AddWithValue("ФИО клиента",
fullnameTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Номер телефона",
phoneTextBox.Text.Trim());
            //


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void button3_Click(object sender, EventArgs e)
        {
            var dashBoardMasters = new Frm_Masters();
            dashBoardMasters.Show();
            this.Hide();
        }
```

```
        private void button4_Click(object sender, EventArgs e)
        {
            var dashBoardSupplierCompany = new
Frm_SupplierCompany();
            dashBoardSupplierCompany.Show();
            this.Hide();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            var dashBoardServices = new Frm_Services();
            dashBoardServices.Show();
            this.Hide();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            var dashBoardProductType = new Frm_ProductType();
            dashBoardProductType.Show();
            this.Hide();
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            var dashBoardReferenceData = new Frm_ReferenceData();
            dashBoardReferenceData.Show();
            this.Hide();
        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста введите данные",
"Добавить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            CRUD.sql = $"INSERT INTO public.\"POKUPATEL\"( " +
                $" \"FIO_POKUPATEL\", \"NOMER_TELEFONA\")" +
                $" VALUES('{fullnameTextBox.Text}',
'{phoneTextBox.Text}')";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");
```

```
            resetMe();
        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }


            CRUD.sql = $"UPDATE public.\"POKUPATEL\" SET " +
                $" \"FIO_POKUPATEL\"='{fullnameTextBox.Text}'," +
                $" \"NOMER_TELEFONA\"='{phoneTextBox.Text}' " +
                $"WHERE \"KOD_POKUPATEL\" = @id::integer";

            execute(CRUD.sql, "Изменить");

            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();
        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }
```

```
            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"POKUPATEL\" WHERE
\"KOD_POKUPATEL\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }
        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;
```

```
                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);

                fullnameTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);
                phoneTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
            }
        }
    }
}
using Npgsql;

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 MedicalManagementSystemDatabase

{

    public partial class Frm_Delivery : Form

    {

        private string id = "";

        private int intRow = 0;

        public Frm_Delivery()

        {

            InitializeComponent();

            loadData("");

            resetMe();
```

```
        }

        private void loadData(string keyword)
        {
            CRUD.sql = "";

            CRUD.sql = "SELECT \"KOD_SOTRUDNIKA\",
\"FIO_SOTRUDNIKA\" FROM public.\"SOTRUDNIK\"";

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            DataTable dtPer = CRUD.PerformCRUD(CRUD.cmd);

            personComboBox.DataSource = dtPer;
            personComboBox.DisplayMember = "FIO_SOTRUDNIKA";
            personComboBox.ValueMember = "KOD_SOTRUDNIKA";



            CRUD.sql = "";

            CRUD.sql = "SELECT \"NOMERDOGOVORA_NA_POSTAVKU\" FROM
public.\"DOGOVOR_NA_POSTAVKU\"";

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            DataTable dtZak = CRUD.PerformCRUD(CRUD.cmd);

            zakComboBox.DataSource = dtZak;
```

```
            zakComboBox.DisplayMember =
"NOMERDOGOVORA_NA_POSTAVKU";
            zakComboBox.ValueMember = "NOMERDOGOVORA_NA_POSTAVKU";


            CRUD.sql = "";


            CRUD.sql = "select p.\"NOMER_POSTAVKI\",\r\n\t " +
                "  p.\"DATA_POSTAVKI\",\r\n\t  " +
                " p.\"NOMER_SHETA_FAKTURI\",\r\n\t " +
                "  p.\"SUMMA\",\r\n\t  " +
                " dp.\"NOMERDOGOVORA_NA_POSTAVKU\",\r\n\t " +
                "  s.\"FIO_SOTRUDNIKA\"\r\n " +
                " from public.\"POSTAVKA\" p\r\n " +
                " join public.\"DOGOVOR_NA_POSTAVKU\" dp\r\n\t " +
                " on dp.\"NOMERDOGOVORA_NA_POSTAVKU\" =
p.\"DOGOVOR_NA_POSTAVKU_ID\"\r\n " +
                " join public.\"SOTRUDNIK\" s\r\n  " +
                "  on s.\"KOD_SOTRUDNIKA\" = p.\"SOTRUDNIK_ID\" "
+
            "WHERE CONCAT(CAST(p.\"NOMER_POSTAVKI\" as varchar), '
', p.\"DATA_POSTAVKI\", ' ', p.\"NOMER_SHETA_FAKTURI\", '
',p.\"SUMMA\", ' '," +
            " dp.\"NOMERDOGOVORA_NA_POSTAVKU\", ' ',
s.\"FIO_SOTRUDNIKA\") " +
            "LIKE @keyword::varchar ORDER BY p.\"NOMER_POSTAVKI\"
ASC";



            string strKeyword = string.Format("%{0}%", keyword);


            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
```

```
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);

            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;

            dgv1.Columns[0].HeaderText = "Номер поставки";
            dgv1.Columns[1].HeaderText = "Дата поставки";
```

```
            dgv1.Columns[2].HeaderText = "Номер счета-фактуры";
            dgv1.Columns[3].HeaderText = "Сумма поставки";
            dgv1.Columns[4].HeaderText = "Номер договора на
поставку";
            dgv1.Columns[5].HeaderText = "ФИО сотрудника";



            dgv1.Columns[0].Width = 100;
            dgv1.Columns[1].Width = 100;
            dgv1.Columns[2].Width = 250;
            dgv1.Columns[3].Width = 250;
            dgv1.Columns[4].Width = 250;
            dgv1.Columns[5].Width = 250;

        }

        private void resetMe()
        {
            this.id = string.Empty;

            sumTextBox.Text = "";
            numTextBox.Text = "";

            if (zakComboBox.Items.Count > 0)
            {
                zakComboBox.SelectedIndex = 0;
            }
```

```
        if (personComboBox.Items.Count > 0)
        {
            personComboBox.SelectedIndex = 0;
        }

        updateButton.Text = "Изменить";
        deleteButton.Text = "Удалить";

        keywordTextBox.Clear();

        if (keywordTextBox.CanSelect)
        {
            keywordTextBox.Select();
        }
    }

    private void button7_Click(object sender, EventArgs e)
    {
        var dashBoard = new Frm_Main();
        dashBoard.Show();
        this.Hide();
    }

    private void button10_Click(object sender, EventArgs e)
    {
        var dashBoardLogin = new Frm_Login();
        dashBoardLogin.Show();
        this.Hide();
```

```
        }


        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }


        private void Frm_Requests_Load(object sender, EventArgs e)
        {
            loadData("");
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }


        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();



            if (zakComboBox.SelectedItem != null &&
zakComboBox.SelectedValue != null)
            {
                if (zakComboBox.SelectedValue is int)
                {
```

```
                    CRUD.cmd.Parameters.AddWithValue("Номер счета-
фактуры", zakComboBox.SelectedValue);
                }
                else if (zakComboBox.SelectedValue is DataRowView)
                {
                    CRUD.cmd.Parameters.AddWithValue("Номер счета-
фактуры", Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]));
                }
            }

            if (personComboBox.SelectedItem != null &&
personComboBox.SelectedValue != null)
            {
                if (personComboBox.SelectedValue is int)
                {
                    CRUD.cmd.Parameters.AddWithValue("ФИО
Сотрудника", personComboBox.SelectedValue);
                }
                else if (personComboBox.SelectedValue is
DataRowView)
                {
                    CRUD.cmd.Parameters.AddWithValue("ФИО
Сотрудника", Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]));
                }
            }

            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
```

```
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);
                numTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
                sumTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);

                zakComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);
                personComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[5].Value);

            }
        }

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

```
            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");


            int zakComboBoxValue = 0;

            int personComboBoxValue = 0;



            if (zakComboBox.SelectedValue is int)

            {

                zakComboBoxValue = (int)zakComboBox.SelectedValue;

            }

            else if (zakComboBox.SelectedValue is DataRowView)

            {

                zakComboBoxValue =
Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]);

            }


            if (personComboBox.SelectedValue is int)

            {

                personComboBoxValue =
(int)personComboBox.SelectedValue;

            }

            else if (personComboBox.SelectedValue is DataRowView)

            {

                personComboBoxValue =
Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]);

            }
```

```
            CRUD.sql = $"INSERT INTO public.\"POSTAVKA\"(" +

                $"\"DATA_POSTAVKI\", \"NOMER_SHETA_FAKTURI\",
\"SUMMA\", \"DOGOVOR_NA_POSTAVKU_ID\", \"SOTRUDNIK_ID\")" +

                        $" VALUES ('{formattedDate}',
'{numTextBox.Text}', '{sumTextBox.Text}',  '{zakComboBoxValue}',
'{personComboBoxValue}')";


            execute(CRUD.sql, "Добавить");


            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


            loadData("");

            resetMe();

        }



        private void updateButton_Click(object sender, EventArgs
e)

        {

            if (dataGridView1.Rows.Count == 0)

            {

                return;

            }


            if (string.IsNullOrEmpty(this.id))

            {

                MessageBox.Show("Пожалуйста выберите строку,
которую хотите изменить", "Изменить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);

                return;
```

```
            }


            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");


            int zakComboBoxValue = 0;

            int personComboBoxValue = 0;



            if (zakComboBox.SelectedValue is int)

            {

                zakComboBoxValue = (int)zakComboBox.SelectedValue;

            }

            else if (zakComboBox.SelectedValue is DataRowView)

            {

                zakComboBoxValue =
Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]);

            }


            if (personComboBox.SelectedValue is int)

            {

                personComboBoxValue =
(int)personComboBox.SelectedValue;

            }

            else if (personComboBox.SelectedValue is DataRowView)

            {

                personComboBoxValue =
Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]);
```

```
            }



            CRUD.sql = $"UPDATE public.\"POSTAVKA\" " +
                       $"SET " +
                       $"\"DATA_POSTAVKI\"='{formattedDate}', " +
$"\"NOMER_SHETA_FAKTURI\"='{numTextBox.Text}', " +
                       $"\"SUMMA\"='{sumTextBox.Text}', " +
$"\"DOGOVOR_NA_POSTAVKU_ID\"='{zakComboBoxValue}', " +
                       $"\"SOTRUDNIK_ID\"='{personComboBoxValue}'"
+
                       $"WHERE \"NOMER_POSTAVKI\" = @id::integer";

            execute(CRUD.sql, "Изменить");

            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);

            loadData("");
            resetMe();
        }



        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
```

```
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"POSTAVKA\" WHERE
\"NOMER_POSTAVKI\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();

            }
```

```
        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();

        }

        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardDelivery = new Frm_Delivery();
            dashBoardDelivery.Show();
            this.Hide();
        }

        private void button3_Click(object sender, EventArgs e)
        {
            var dashBoardSupplies = new Frm_Supplies();
```

```
            dashBoardSupplies.Show();
            this.Hide();
        }

        private void button8_Click(object sender, EventArgs e)
        {
            var dashBoardRequests = new Frm_Requests();
            dashBoardRequests.Show();
            this.Hide();
        }
    }
}
using Npgsql;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using static MedicalManagementSystemDatabase.Frm_Login;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_Main : Form
    {
        private string id = "";
        private int intRow = 0;

        public Frm_Main() //*****
        {
```

```
            InitializeComponent();
            loadData("");
            resetMe();

            label1.Text = parser.Log.name;
            label2.Text = parser.Log.role;

            Name = label1.Text;

            picPhoto.Image = Image.FromFile(parser.Log.photo);

            if (parser.Log.role == "Директор")
            {
                button6.Visible = false;
                button9.Visible = false;
            }
            else if (parser.Log.role == "Пользователь")
            {
                button2.Visible = false;
                button3.Visible = false;
                button5.Visible = false;
                button7.Visible = false;
                button4.Location = new Point(3, 1);
                insertButton.Visible = false;
                updateButton.Visible = false;
                deleteButton.Visible = false;

            }
```

```
        }

        private void resetMe()
        {
            this.id = string.Empty;

            NameMaterialTextBox.Text = "";
            priceTextBox.Text = "";
            kolTextBox.Text = "";

            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }

        }

        private void loadData(string keyword)
        {
            CRUD.sql = "";

            CRUD.sql = "select * FROM public.\"VID_TOVARA\"";
```

```
            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            DataTable ddt = CRUD.PerformCRUD(CRUD.cmd);

            tovarComboBox.DataSource = ddt;
            tovarComboBox.DisplayMember = "VID_TOVARA";
            tovarComboBox.ValueMember = "KOD_VIDA_TOVARA";

            CRUD.sql = "";

            CRUD.sql = "SELECT * FROM public.\"TOVAR\"\r\nORDER BY
\"KOD_TOVARA\" ASC ";

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            CRUD.sql = "";

            CRUD.sql = "SELECT * FROM public.\"TOVAR\"\r\n " +
               "WHERE CONCAT(CAST(\"KOD_TOVARA\" as varchar), ' ',
\"NAIM_TOVARA\") LIKE @keyword::varchar " +
               "ORDER BY \"KOD_TOVARA\" ASC";

            string strKeyword = string.Format("%{0}%", keyword);

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);
```

```
            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);



            if (dt.Rows.Count > 0)

            {

                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());

            }

            else

            {

                intRow = 0;

            }


            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();


            DataGridView dgv1 = dataGridView1;


            dgv1.MultiSelect = false;

            dgv1.AutoGenerateColumns = true;

            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;


            dgv1.DataSource = dt;


            dgv1.Columns[0].HeaderText = "Код товара";

            dgv1.Columns[1].HeaderText = "Наименование материала";

            dgv1.Columns[2].HeaderText = "Цена материала";

            dgv1.Columns[3].HeaderText = "Количество материала";
```

```
            dgv1.Columns[4].HeaderText = "Вид товара";


            dgv1.Columns[0].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[4].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;


            for (int i = 0; i <= dgv1.Columns.Count - 1; i++)
            {
                // Store Auto Sized Widths
                int colw = dgv1.Columns[i].Width;


                // Remove AutoSizing
                dgv1.Columns[i].AutoSizeMode =
DataGridViewAutoSizeColumnMode.None;


                // Set Width to calculated AutoSize value:
                dgv1.Columns[i].Width = colw;
            }


        }


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

```
            loadData("");
        }

        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }
        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();
            //
            CRUD.cmd.Parameters.AddWithValue("Наименование
материала", NameMaterialTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Цена материала",
priceTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Количество
материала", kolTextBox.Text.Trim());
            //
            NpgsqlParameter npgsqlVidTovara =
CRUD.cmd.Parameters.AddWithValue("Вид товара",
tovarComboBox.SelectedValue);

            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }
```

```
private void button10_Click(object sender, EventArgs e)
{
    var dashBoardLogin = new Frm_Login();
    dashBoardLogin.Show();
    this.Hide();
}

private void button2_Click(object sender, EventArgs e)
{
    var dashBoardSold = new Frm_Sold();
    dashBoardSold.Show();
    this.Hide();
}

private void button3_Click(object sender, EventArgs e)
{
    var dashBoardSupplies = new Frm_Supplies();
    dashBoardSupplies.Show();
    this.Hide();
}

private void button4_Click(object sender, EventArgs e)
{
    var dashBoardPosts = new Frm_Posts();
    dashBoardPosts.Show();
    this.Hide();
```

```
        }

        private void button5_Click(object sender, EventArgs e)
        {
            var dashBoardReferenceData = new Frm_ReferenceData();
            dashBoardReferenceData.Show();
            this.Hide();
        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoardReportInfo = new Frm_ReportInfo();
            dashBoardReportInfo.Show();
            this.Hide();
        }

        private void button9_Click(object sender, EventArgs e)
        {
            MessageBox.Show(
                "Программа предназначена для автоматизации и
упрощения жизни клиентов.",
                "Сообщение",
                MessageBoxButtons.OKCancel,
                MessageBoxIcon.Information,
                MessageBoxDefaultButton.Button1,
                MessageBoxOptions.DefaultDesktopOnly);
        }
```

```
        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;


                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);


                NameMaterialTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);
                priceTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
                kolTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);
                tovarComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);
            }
        }


        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
```

```
            }

            resetMe();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            {
                MessageBox.Show(
                    "Мы находимся по адресу г.Пенза, ул.Суворова
133, каб. 412",
                    "Сообщение",
                    MessageBoxButtons.YesNo,
                    MessageBoxIcon.Information,
                    MessageBoxDefaultButton.Button1,
                    MessageBoxOptions.DefaultDesktopOnly);
            }
        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            if
(string.IsNullOrEmpty(NameMaterialTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста введите данные",
"Добавить данные", MessageBoxButtons.OK,
                MessageBoxIcon.Exclamation);
                return;
            }
```

```
            CRUD.sql = $"insert into public.\"TOVAR\"
(\"NAIM_TOVARA\", \"CENA\", \"KOLICHESTVO\",
\"VID_TOVARA_ID\")\r\n" +
                $" VALUES('{NameMaterialTextBox.Text}', " +
            $"'{priceTextBox.Text}', '{kolTextBox.Text}',
'{tovarComboBox.SelectedValue}')";
            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,
                MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
```

```
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", $"Изменить данные", MessageBoxButtons.OK,

                    MessageBoxIcon.Exclamation);

                return;

            }


            CRUD.sql = $"UPDATE public.\"TOVAR\" SET
\"VID_TOVARA_ID\" = '{tovarComboBox.SelectedValue}', " +

                $"\"NAIM_TOVARA\" = '{NameMaterialTextBox.Text}',
\"CENA\" = '{priceTextBox.Text}', \"KOLICHESTVO\" =
'{kolTextBox.Text}' " +

                $"WHERE \"KOD_TOVARA\" = @id::integer";




            execute(CRUD.sql, "Изменить");


            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,

            MessageBoxIcon.Information);


            loadData("");


            resetMe();

        }


        private void deleteButton_Click(object sender, EventArgs
e)

        {

            if (dataGridView1.Rows.Count == 0)

            {
```

```
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"TOVAR\" WHERE
\"KOD_TOVARA\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();

            }
```

```
        }
    }
}
using Npgsql;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using static MedicalManagementSystemDatabase.Frm_Login;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_Masters : Form
    {
        private string id = "";
        private int intRow = 0;

        public Frm_Masters()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }

        private void resetMe()
        {
            this.id = string.Empty;
```

```
            fullnameTextBox.Text = "";
            mailTextBox.Text = "";
            phoneTextBox.Text = "";
            rankTextBox.Text = "";
            spehTextBox.Text = "";
            reqTextBox.Text = "";



            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }

        }

        private void loadData(string keyword)
        {

            CRUD.sql = "";

            CRUD.sql = "SELECT * FROM public.\"MASTER\"\r\n " +
                "WHERE CONCAT(CAST(\"KOD_MASTER\" as varchar), '
', \"FIO_MASTER\", ' ', \"ADRES_EL_POCHTI\", ' ',
\"NOMER_TELEFONA\"," +
```

```
                " ' ', \"STAG\", ' ', \"SPECIALIZACIYA\", ' ',
\"REKVIZITI\") " +
                "LIKE @keyword::varchar ORDER BY \"KOD_MASTER\"
ASC ";



            string strKeyword = string.Format("%{0}%", keyword);


            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);


            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);



            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
            {
                intRow = 0;
            }


            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();


            DataGridView dgv1 = dataGridView1;
```

```
            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;


            dgv1.DataSource = dt;


            dgv1.Columns[0].HeaderText = "Код мастера";
            dgv1.Columns[1].HeaderText = "ФИО мастера";
            dgv1.Columns[2].HeaderText = "Адрес эл. почты";
            dgv1.Columns[3].HeaderText = "Номер телефона";
            dgv1.Columns[4].HeaderText = "Стаж";
            dgv1.Columns[5].HeaderText = "Специализация";
            dgv1.Columns[6].HeaderText = "Реквизиты";


            dgv1.Columns[0].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[4].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[5].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[6].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
```

```
}

private void button7_Click(object sender, EventArgs e)
{
    var dashBoard = new Frm_Main();
    dashBoard.Show();
    this.Hide();
}

private void button10_Click(object sender, EventArgs e)
{
    var dashBoardLogin = new Frm_Login();
    dashBoardLogin.Show();
    this.Hide();
}

private void button2_Click(object sender, EventArgs e)
{
    var dashBoardBuyers = new Frm_Buyer();
    dashBoardBuyers.Show();
    this.Hide();
}

private void Frm_Masters_Load(object sender, EventArgs e)
{
    loadData("");
}
private void execute(string mySQL, string param)
```

```
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            CRUD.cmd.Parameters.AddWithValue("ФИО сотрудника",
fullnameTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Адрес эл. почты",
mailTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Номер телефона",
phoneTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Стаж",
rankTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Специализация",
spehTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Реквизиты",
reqTextBox.Text.Trim());
            //

            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }
```

```
        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)

        {

            if (e.RowIndex != -1)

            {

                DataGridView dgv1 = dataGridView1;


                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);


                fullnameTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);

                mailTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);

                phoneTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);

                rankTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);

                spehTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[5].Value);

                reqTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[6].Value);

            }


        }


        private void insertButton_Click(object sender, EventArgs
e)

        {

            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))

            {
```

```
                MessageBox.Show("Пожалуйста введите данные",
"Добавить данные", MessageBoxButtons.OK,

                    MessageBoxIcon.Exclamation);

                return;

            }


            CRUD.sql = $"INSERT INTO public.\"MASTER\"( " +

                $" \"FIO_MASTER\", \"ADRES_EL_POCHTI\",
\"NOMER_TELEFONA\", \"STAG\", \"SPECIALIZACIYA\", \"REKVIZITI\")"
+

                $" VALUES('{fullnameTextBox.Text}', " +

                $"'{mailTextBox.Text}','{phoneTextBox.Text}'," +

                $" '{rankTextBox.Text}', '{spehTextBox.Text}',
'{reqTextBox.Text}')";


            execute(CRUD.sql, "Добавить");


            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,

    MessageBoxIcon.Information);


            loadData("");


            resetMe();


        }


        private void updateButton_Click(object sender, EventArgs
e)

        {
```

```
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }


            CRUD.sql = $"UPDATE public.\"MASTER\" SET " +
                $" \"FIO_MASTER\"='{fullnameTextBox.Text}'," +
                $" \"ADRES_EL_POCHTI\"='{mailTextBox.Text}'," +
                $" \"NOMER_TELEFONA\"='{phoneTextBox.Text}'," +
                $" \"STAG\"='{rankTextBox.Text}'," +
                $" \"SPECIALIZACIYA\"='{spehTextBox.Text}'," +
                $" \"REKVIZITI\"='{reqTextBox.Text}'" +
```

```
                $"WHERE \"KOD_MASTER\" = @id::integer";


            execute(CRUD.sql, "Изменить");


            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);


            loadData("");


            resetMe();


        }


        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }


            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }
```

```
            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"MASTER\" WHERE
\"KOD_MASTER\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }

        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
```

```
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();

        }

        private void button3_Click(object sender, EventArgs e)
        {
            var dashBoardReferenceData = new Frm_ReferenceData();
            dashBoardReferenceData.Show();
            this.Hide();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            var dashBoardSupplierCompany = new
Frm_SupplierCompany();
            dashBoardSupplierCompany.Show();
            this.Hide();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            var dashBoardServices = new Frm_Services();
            dashBoardServices.Show();
            this.Hide();
```

```
        }


        private void button6_Click(object sender, EventArgs e)
        {
            var dashBoardProductType = new Frm_ProductType();
            dashBoardProductType.Show();
            this.Hide();
        }


    }
}
using System;
using System.IO;
using System.Windows.Forms;
using System.Xml;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_Login : Form
    {
        public Frm_Login()
        {
            InitializeComponent();
        }

        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        private void Guna2Button1_Click(object sender, EventArgs
e)
        {
            var username = Guna2TextBox1.Text;
            var password = Guna2TextBox2.Text;

            if (parser.parsLoader(username, password))
            {
                if (parser.Log.role == "Админ")
                {
                    var dashBoard = new Frm_Main();
```

```
                    dashBoard.Show();
                    this.Hide();
                }
                else if (parser.Log.role == "Директор")
                {
                    var dashBoard = new Frm_Main();
                    dashBoard.Show();
                    this.Hide();
                }
                else if (parser.Log.role == "Пользователь")
                {
                    var dashBoard = new Frm_Main();
                    dashBoard.Show();
                    this.Hide();
                }
            }
            else
            {
                MessageBox.Show("Неверный Логин/Пароль, Обратитесь
к амдминестратору", "Ограничение", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }



        }

        private void Guna2CheckBox1_CheckedChanged(object sender,
EventArgs e)
        {
            if (Guna2CheckBox1.Checked == true)
            {
                Guna2CheckBox1.Text = "Спрятать пароль";
                Guna2TextBox2.PasswordChar = '\0';
            }
            else
            {
                Guna2CheckBox1.Text = "Показать пароль";
                Guna2TextBox2.PasswordChar = '*';
            }
        }

        private void Label1_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Обратитесь к амдминестратору для
создания нового аккаунта", "Ограничение", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }

        public class parser
        {
            private static logindata log;
```

```
            public static logindata Log { get => log; set => log =
value; }

            public static bool parsLoader(String login, String
password)
            {
                logindata loginU = new logindata();
                var filepath = "loader.xml";
                string name, pwd; // Новые переменные имени и
пароля

                // Объявляем и забиваем файл в документ
                XmlDocument xd = new XmlDocument();
                FileStream fs = new FileStream(filepath,
FileMode.Open);
                xd.Load(fs);

                XmlNodeList list =
xd.GetElementsByTagName("user"); // Создаем и заполняем лист по
тегу "user"
                for (int i = 0; i < list.Count; i++)
                {
                    // XmlElement id =
(XmlElement)xd.GetElementsByTagName("user")[i];          //
Забиваем id в переменную
                    XmlElement user =
(XmlElement)xd.GetElementsByTagName("login")[i];       // Забиваем
login в переменную
                    XmlElement pass =
(XmlElement)xd.GetElementsByTagName("password")[i];    // Забиваем
password в переменную
                    XmlElement role =
(XmlElement)xd.GetElementsByTagName("role")[i];    // Забиваем role
в переменную
                    XmlElement fullname =
(XmlElement)xd.GetElementsByTagName("name")[i];    // Забиваем name
в переменную
                    XmlElement photo =
(XmlElement)xd.GetElementsByTagName("photo")[i];    // Забиваем
photo в переменную


                    name = user.InnerText;
                    if (name == login)
                    {
                        pwd = pass.InnerText;
                        if (pwd == password)
                        {
                            loginU.role = role.InnerText;
                            loginU.name = fullname.InnerText;
                            loginU.photo = photo.InnerText;
```

```
                                Log = loginU;

                                fs.Close();
                                return true;
                            }
                        }
                    }
                    // Закрываем поток
                    fs.Close();
                    return false;
                }
            }
        }
    }
```

```
using Npgsql;

using System;

using System.Data;

using System.Windows.Forms;


namespace MedicalManagementSystemDatabase

{

    public partial class Frm_Posts : Form

    {

        private string id = "";

        private int intRow = 0;

        public Frm_Posts()

        {

            InitializeComponent();

            loadData("");

            resetMe();

        }


        private void loadData(string keyword)
```

```
        {
            CRUD.sql = "";

            CRUD.sql = "SELECT \"KOD_MASTER\", \"FIO_MASTER\" FROM
public.\"MASTER\"";

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            DataTable dtPer = CRUD.PerformCRUD(CRUD.cmd);

            personComboBox.DataSource = dtPer;
            personComboBox.DisplayMember = "FIO_MASTER";
            personComboBox.ValueMember = "KOD_MASTER";

            CRUD.sql = "";

            CRUD.sql = "SELECT \"KOD_USLUGI\", \"NAIM_USLUGI\"
FROM public.\"USLUGA\"";

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            DataTable dtZak = CRUD.PerformCRUD(CRUD.cmd);

            zakComboBox.DataSource = dtZak;
            zakComboBox.DisplayMember = "NAIM_USLUGI";
            zakComboBox.ValueMember = "KOD_USLUGI";

            CRUD.sql = "";
```

```
            CRUD.sql = "select pu.\"NOMER_DOGVORA\", \r\n " +
            "      pu.\"DATA_SOST_DOGOVORA\", \r\n\t " +
            "  u.\"NAIM_USLUGI\", \r\n\t " +
            "  m.\"FIO_MASTER\"\r\n " +
            " from public.\"DOGOVOR_NA_OKAZ_USLUG\" pu\r\n " +
            " join public.\"USLUGA\" u\r\n  " +
            "  on pu.\"USLUGA_ID\" = u.\"KOD_USLUGI\"\r\n " +
            " join public.\"MASTER\" m\r\n  " +
            "  on pu.\"MASTER_ID\" = m.\"KOD_MASTER\" " +
            "WHERE CONCAT(CAST(pu.\"NOMER_DOGVORA\" as varchar), '
', pu.\"DATA_SOST_DOGOVORA\") " +
            "LIKE @keyword::varchar ORDER BY pu.\"NOMER_DOGVORA\"
ASC";



            string strKeyword = string.Format("%{0}%", keyword);


            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);


            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
```

```
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;

            dgv1.Columns[0].HeaderText = "Номер договора";
            dgv1.Columns[1].HeaderText = "Дата составления
договора";
            dgv1.Columns[2].HeaderText = "Услуга";
            dgv1.Columns[3].HeaderText = "ФИО Мастера";

            dgv1.Columns[0].Width = 85;
            dgv1.Columns[1].Width = 100;
            dgv1.Columns[2].Width = 250;
            dgv1.Columns[3].Width = 200;

        }
```

```
private void resetMe()
{
    this.id = string.Empty;



    if (zakComboBox.Items.Count > 0)
    {
        zakComboBox.SelectedIndex = 0;
    }

    if (personComboBox.Items.Count > 0)
    {
        personComboBox.SelectedIndex = 0;
    }

    updateButton.Text = "Изменить";
    deleteButton.Text = "Удалить";

    keywordTextBox.Clear();

    if (keywordTextBox.CanSelect)
    {
        keywordTextBox.Select();
    }
}
```

```
        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        private void Frm_Posts_Load(object sender, EventArgs e)
        {
            loadData("");
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
```

```
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            if (zakComboBox.SelectedItem != null &&
zakComboBox.SelectedValue != null)
            {
                if (zakComboBox.SelectedValue is int)
                {
                    CRUD.cmd.Parameters.AddWithValue("Услуга",
zakComboBox.SelectedValue);
                }
                else if (zakComboBox.SelectedValue is DataRowView)
                {
                    CRUD.cmd.Parameters.AddWithValue("Услуга",
Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]));
                }
            }

            if (personComboBox.SelectedItem != null &&
personComboBox.SelectedValue != null)
            {
                if (personComboBox.SelectedValue is int)
                {
                    CRUD.cmd.Parameters.AddWithValue("ФИО
Мастера", personComboBox.SelectedValue);
```

```
                }
                else if (personComboBox.SelectedValue is
DataRowView)
                {
                    CRUD.cmd.Parameters.AddWithValue("ФИО
Мастера", Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]));
                }
            }

            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);

                zakComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
```

```
                personComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);


            }

        }


        private void insertButton_Click(object sender, EventArgs
e)

        {

            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");


            int zakComboBoxValue = 0;

            int personComboBoxValue = 0;


            if (zakComboBox.SelectedValue is int)

            {

                zakComboBoxValue = (int)zakComboBox.SelectedValue;

            }

            else if (zakComboBox.SelectedValue is DataRowView)

            {

                zakComboBoxValue =
Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]);

            }


            if (personComboBox.SelectedValue is int)

            {

                personComboBoxValue =
(int)personComboBox.SelectedValue;
```

```
            }
            else if (personComboBox.SelectedValue is DataRowView)
            {
                personComboBoxValue =
Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]);
            }

            CRUD.sql = $"INSERT INTO
public.\"DOGOVOR_NA_OKAZ_USLUG\"(\"DATA_SOST_DOGOVORA\",
\"USLUGA_ID\", \"MASTER_ID\")" +
                        $" VALUES ('{formattedDate}',
{zakComboBoxValue}, {personComboBoxValue})";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);

            loadData("");
            resetMe();
        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }
```

```
            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите изменить", "Изменить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
            }

            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");

            int zakComboBoxValue = 0;
            int personComboBoxValue = 0;

            if (zakComboBox.SelectedValue is int)
            {
                zakComboBoxValue = (int)zakComboBox.SelectedValue;
            }
            else if (zakComboBox.SelectedValue is DataRowView)
            {
                zakComboBoxValue =
Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]);
            }

            if (personComboBox.SelectedValue is int)
            {
                personComboBoxValue =
(int)personComboBox.SelectedValue;
```

```
            }
            else if (personComboBox.SelectedValue is DataRowView)
            {
                personComboBoxValue =
Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]);
            }

            CRUD.sql = $"UPDATE public.\"DOGOVOR_NA_OKAZ_USLUG\" "
+
                       $"SET \"DATA_SOST_DOGOVORA\" =
'{formattedDate}', \"USLUGA_ID\" = {zakComboBoxValue},
\"MASTER_ID\" = {personComboBoxValue} " +
                       $"WHERE \"NOMER_DOGVORA\" = @id::integer";

            execute(CRUD.sql, "Изменить");

            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);

            loadData("");
            resetMe();
        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
```

```
            }


            if (string.IsNullOrEmpty(this.id))

            {

                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,

                    MessageBoxIcon.Exclamation);

                return;

            }


            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",

                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)

            {

                CRUD.sql = "DELETE FROM
public.\"DOGOVOR_NA_OKAZ_USLUG\" WHERE \"NOMER_DOGVORA\" =
@id::integer";


                execute(CRUD.sql, "Удалить");


                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,

        MessageBoxIcon.Information);


                loadData("");


                resetMe();


            }
```

```
        }


        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();


        }


        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardReceptionSchedule = new
Frm_ReceptionSchedule();
            dashBoardReceptionSchedule.Show();
            this.Hide();
        }
    }
}

using Npgsql;
using System;
using System.Data;
```

```
using System.Drawing;
using System.Windows.Forms;
using static MedicalManagementSystemDatabase.Frm_Login;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_ProductType : Form
    {
        private string id = "";
        private int intRow = 0;

        public Frm_ProductType()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }

        private void resetMe()
        {
            this.id = string.Empty;

            fullnameTextBox.Text = "";


            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }

        }

        private void loadData(string keyword)
        {

            CRUD.sql = "";

            CRUD.sql = "SELECT * FROM public.\"VID_TOVARA\"\r\n "
+
                "WHERE CONCAT(CAST(\"KOD_VIDA_TOVARA\" as
varchar), ' ', \"VID_TOVARA\") " +
                "LIKE @keyword::varchar ORDER BY
\"KOD_VIDA_TOVARA\" ASC ";


            string strKeyword = string.Format("%{0}%", keyword);
```

```
            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;

            dgv1.Columns[0].HeaderText = "Код вида материала";
            dgv1.Columns[1].HeaderText = "Вид материала";

            dgv1.Columns[0].Width = 150;
            dgv1.Columns[1].Width = 250;


        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }
```

```
        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardBuyers = new Frm_Buyer();
            dashBoardBuyers.Show();
            this.Hide();
        }

        private void Frm_ProductType_Load(object sender, EventArgs
e)
        {
            loadData("");
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            CRUD.cmd.Parameters.AddWithValue("Вид материала",
fullnameTextBox.Text.Trim());
            //


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);

                fullnameTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);
            }

        }
```

```
        private void insertButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста введите данные",
"Добавить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            CRUD.sql = $"INSERT INTO
public.\"VID_TOVARA\"(\"VID_TOVARA\") " +
                $"VALUES('{fullnameTextBox.Text}')";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }
```

```
            CRUD.sql = $"UPDATE public.\"VID_TOVARA\" SET " +
                $"\"VID_TOVARA\"='{fullnameTextBox.Text}' " +
                $"WHERE \"KOD_VIDA_TOVARA\" = @id::integer";

            execute(CRUD.sql, "Изменить");

            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"VID_TOVARA\"
WHERE \"KOD_VIDA_TOVARA\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }

        }
```

```
        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();

        }

        private void button3_Click(object sender, EventArgs e)
        {
            var dashBoardMasters = new Frm_Masters();
            dashBoardMasters.Show();
            this.Hide();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            var dashBoardSupplierCompany = new
Frm_SupplierCompany();
            dashBoardSupplierCompany.Show();
            this.Hide();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            var dashBoardServices = new Frm_Services();
            dashBoardServices.Show();
            this.Hide();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            var dashBoardReferenceData = new Frm_ReferenceData();
            dashBoardReferenceData.Show();
            this.Hide();
        }

    }
}
using Npgsql;

using System;
```

```
using System.Data;
using System.Windows.Forms;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_ReceptionSchedule : Form
    {
        private string id = "";
        private int intRow = 0;
        public Frm_ReceptionSchedule()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }

        private void resetMe()
        {
            this.id = string.Empty;



            if (zakComboBox.Items.Count > 0)
            {
                zakComboBox.SelectedIndex = 0;
            }

            if (skinComboBox.Items.Count > 0)
            {
```

```
                skinComboBox.SelectedIndex = 0;
            }

            if (personComboBox.Items.Count > 0)
            {
                personComboBox.SelectedIndex = 0;
            }

            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }
        }
        private void Frm_ReceptionSchedule_Load(object sender,
EventArgs e)
        {
            loadData("");
        }

        private void loadData(string keyword)
        {
            {
```

```
                CRUD.sql = "";

                CRUD.sql = "SELECT \"NOMER_DOGVORA\" FROM
public.\"DOGOVOR_NA_OKAZ_USLUG\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable dtDog = CRUD.PerformCRUD(CRUD.cmd);

                zakComboBox.DataSource = dtDog;
                zakComboBox.DisplayMember = "NOMER_DOGVORA";
                zakComboBox.ValueMember = "NOMER_DOGVORA";

                CRUD.sql = "";

                CRUD.sql = "SELECT \"KOD_SOTRUDNIKA\",
\"FIO_SOTRUDNIKA\" FROM public.\"SOTRUDNIK\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable dtPer = CRUD.PerformCRUD(CRUD.cmd);

                personComboBox.DataSource = dtPer;
                personComboBox.DisplayMember = "FIO_SOTRUDNIKA";
                personComboBox.ValueMember = "KOD_SOTRUDNIKA";

                CRUD.sql = "";
```

```
                CRUD.sql = "select
gp.\"NOMER_GRAFIKA_PRIEMA\",\r\n\t " +
                    "  gp.\"DATA_PROVED_PRIEMA\",\r\n\t " +
                    "  gp.\"VREMYA_PROVED_PRIEMA\",\r\n\t  " +
                    " gp.\"PODTVERJDENIE\",\r\n\t  " +
                    " du.\"USLUGA_ID\",\r\n\t  " +
                    " s.\"FIO_SOTRUDNIKA\"\r\nFROM
public.\"GRAFIK_PRIEMA\"" +
                    " gp\r\njoin public.\"DOGOVOR_NA_OKAZ_USLUG\"
du\r\non gp.\"DOGOVOR_NA_OKAZANIE_ID\" = du.\"NOMER_DOGVORA\"\r\n"
+
                    " join public.\"SOTRUDNIK\" s\r\non
s.\"KOD_SOTRUDNIKA\" = gp.\"SOTRUDNIK_ID\"" +
                    "WHERE CONCAT(CAST(\"NOMER_GRAFIKA_PRIEMA\" as
varchar), ' ', \"DATA_PROVED_PRIEMA\") LIKE @keyword::varchar " +
                    "ORDER BY \"NOMER_GRAFIKA_PRIEMA\" ASC";


                string strKeyword = string.Format("%{0}%",
keyword);


                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
                CRUD.cmd.Parameters.Clear();
                CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);


                DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


                if (dt.Rows.Count > 0)
                {
                    intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
```

```
                }
                else
                {
                    intRow = 0;
                }

                toolStripStatusLabel1.Text = "Количество строк: "
+ intRow.ToString();

                DataGridView dgv1 = dataGridView1;

                dgv1.MultiSelect = false;
                dgv1.AutoGenerateColumns = true;
                dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

                dgv1.DataSource = dt;

                dgv1.Columns[0].HeaderText = "Номер приема";
                dgv1.Columns[1].HeaderText = "Дата приема";
                dgv1.Columns[2].HeaderText = "Время приема";
                dgv1.Columns[3].HeaderText = "Статус приема";
                dgv1.Columns[4].HeaderText = "Номер договора";
                dgv1.Columns[5].HeaderText = "Сотрудник";

                dgv1.Columns[0].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
                dgv1.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
```

```
                dgv1.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[4].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[5].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;


            }


        }


        private void execute(string mySQL, string param)

        {

            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);

            addParameters(param);

            CRUD.PerformCRUD(CRUD.cmd);

        }


        private void addParameters(string str)

        {

            CRUD.cmd.Parameters.Clear();


            NpgsqlParameter npgsqlSkin =
CRUD.cmd.Parameters.AddWithValue("Статус приема",
skinComboBox.SelectedItem.ToString());

            NpgsqlParameter npgsqlDog =
CRUD.cmd.Parameters.AddWithValue("Номер договора",
zakComboBox.SelectedItem.ToString());
```

```
            NpgsqlParameter npgsqlPer =
CRUD.cmd.Parameters.AddWithValue("Сотрудник",
personComboBox.SelectedValue);


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }


        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }


        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }


        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }
```

```
        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;


                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);
                skinComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);
                zakComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);
                personComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[5].Value);


            }
        }


        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardHandbookDiagnosis = new Frm_Sold();
            dashBoardHandbookDiagnosis.Show();
            this.Hide();
        }


        private void button10_Click_1(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
```

```
            dashBoardLogin.Show();

            this.Hide();

        }


        private void insertButton_Click_1(object sender, EventArgs
e)

        {

            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");

            string formattedTime =
dateTimePicker2.Value.ToString("HH:mm:ss");



            CRUD.sql = $"INSERT INTO public.\"GRAFIK_PRIEMA\"( " +

                       $"\"DATA_PROVED_PRIEMA\",
\"VREMYA_PROVED_PRIEMA\", \"PODTVERJDENIE\",
\"DOGOVOR_NA_OKAZANIE_ID\", \"SOTRUDNIK_ID\") " +

                       $"VALUES ('{formattedDate}',
'{formattedTime}', " +

                       $"'{skinComboBox.SelectedItem}',
'{zakComboBox.SelectedValue}', '{personComboBox.SelectedValue}')";



            execute(CRUD.sql, "Добавить");



            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);



            loadData("");



            resetMe();

        }
```

```
        private void updateButton_Click_1(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите изменить", "Изменить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
            }

            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");
            string formattedTime =
dateTimePicker2.Value.ToString("HH:mm:ss");

            CRUD.sql = $"UPDATE public.\"GRAFIK_PRIEMA\" " +
                       $"SET
\"DATA_PROVED_PRIEMA\"='{formattedDate}', " +

$"\"VREMYA_PROVED_PRIEMA\"='{formattedTime}'," +
                       $"
\"PODTVERJDENIE\"='{skinComboBox.SelectedItem}'," +
                       $"
\"DOGOVOR_NA_OKAZANIE_ID\"='{zakComboBox.SelectedValue}'," +
```

```
                        $"
\"SOTRUDNIK_ID\"='{personComboBox.SelectedValue}' " +
                        $" WHERE \"NOMER_GRAFIKA_PRIEMA\" =
@id::integer";


            execute(CRUD.sql, "Изменить");


            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


            loadData("");


            resetMe();
        }



        private void deleteButton_Click_1(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }


            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите удалить", "Удалить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
```

```
            }


            if (MessageBox.Show("Вы уверены, что хотите удалить
запись?", "Удалить данные",

                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)

            {

                CRUD.sql = "DELETE FROM public.\"GRAFIK_PRIEMA\"
WHERE \"NOMER_GRAFIKA_PRIEMA\" = @id::integer";


                execute(CRUD.sql, "Удалить");


                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


                loadData("");


                resetMe();

            }

        }



        private void searchButton_Click_1(object sender, EventArgs
e)

        {

            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))

            {

                loadData("");

            }

            else
```

```
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();
        }

        private void button7_Click_1(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            var dashBoardPosts = new Frm_Posts();
            dashBoardPosts.Show();
            this.Hide();
        }

        private void button10_Click_2(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }
    }
```

```
}

using Npgsql;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using static MedicalManagementSystemDatabase.Frm_Login;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_ReferenceData : Form
    {
        private string id = "";
        private int intRow = 0;

        public Frm_ReferenceData()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }

        private void resetMe()
        {
            this.id = string.Empty;

            fullnameTextBox.Text = "";
            mailTextBox.Text = "";
            rankTextBox.Text = "";
            phoneTextBox.Text = "";


            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }

        }

        private void loadData(string keyword)
        {

            CRUD.sql = "";

            CRUD.sql = "SELECT * FROM public.\"SOTRUDNIK\"\r\n " +
```

```
                "WHERE CONCAT(CAST(\"KOD_SOTRUDNIKA\" as varchar),
' ', \"FIO_SOTRUDNIKA\", ' ', \"ADRES_EL_POCHTI\", ' ',
\"DOLJNOST\"," +
                " ' ', \"NOMER_TELEFONA\") " +
                "LIKE @keyword::varchar ORDER BY
\"KOD_SOTRUDNIKA\" ASC ";


            string strKeyword = string.Format("%{0}%", keyword);

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;

            dgv1.Columns[0].HeaderText = "Код сотрудника";
            dgv1.Columns[1].HeaderText = "ФИО сотрудника";
            dgv1.Columns[2].HeaderText = "Адрес эл. почты";
            dgv1.Columns[3].HeaderText = "Должность";
            dgv1.Columns[4].HeaderText = "Номер телефона";

            dgv1.Columns[0].Width = 50;
            dgv1.Columns[1].Width = 250;
            dgv1.Columns[2].Width = 150;
            dgv1.Columns[3].Width = 100;
            dgv1.Columns[4].Width = 150;
```

```
        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardBuyers = new Frm_Buyer();
            dashBoardBuyers.Show();
            this.Hide();
        }

        private void Frm_ReferenceData_Load(object sender,
EventArgs e)
        {
            loadData("");
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            CRUD.cmd.Parameters.AddWithValue("ФИО сотрудника",
fullnameTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Адрес эл. почты",
mailTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Должность",
rankTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Номер телефона",
phoneTextBox.Text.Trim());
            //


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
```

```
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);

                fullnameTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);
                mailTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
                rankTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);
                phoneTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);
            }

        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста введите данные",
"Добавить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            CRUD.sql = $"INSERT INTO public.\"SOTRUDNIK\"( " +
                $" \"FIO_SOTRUDNIKA\", \"ADRES_EL_POCHTI\",
\"DOLJNOST\", \"NOMER_TELEFONA\")" +
                $" VALUES('{fullnameTextBox.Text}', " +
                $"'{mailTextBox.Text}','{rankTextBox.Text}'," +
                $" '{phoneTextBox.Text}')";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");
```

```
            resetMe();

        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }


            CRUD.sql = $"UPDATE public.\"SOTRUDNIK\" SET " +
                $"\"FIO_SOTRUDNIKA\"='{fullnameTextBox.Text}', " +
                $"\"ADRES_EL_POCHTI\"='{mailTextBox.Text}', " +
                $" \"DOLJNOST\"='{rankTextBox.Text}'," +
                $" \"NOMER_TELEFONA\"='{phoneTextBox.Text}' " +
                $"WHERE \"KOD_SOTRUDNIKA\" = @id::integer";

            execute(CRUD.sql, "Изменить");

            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

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

```
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"SOTRUDNIK\" WHERE
\"KOD_SOTRUDNIKA\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }

        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();

        }

        private void button3_Click(object sender, EventArgs e)
        {
```

```
            var dashBoardMasters = new Frm_Masters();
            dashBoardMasters.Show();
            this.Hide();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            var dashBoardSupplierCompany = new
Frm_SupplierCompany();
            dashBoardSupplierCompany.Show();
            this.Hide();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            var dashBoardServices = new Frm_Services();
            dashBoardServices.Show();
            this.Hide();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            var dashBoardProductType = new Frm_ProductType();
            dashBoardProductType.Show();
            this.Hide();
        }

    }
}
using System.Windows.Forms;
using System.Diagnostics;
using ServiceStack.MiniProfiler.Data;
using Reprter;
using System;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_ReportInfo : Form
    {
        public Frm_ReportInfo()
        {
            InitializeComponent();
        }

        private void button7_Click(object sender, System.EventArgs
e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }
```

```
        private void button10_Click(object sender,
System.EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void insertButton_Click(object sender,
System.EventArgs e)
        {
            string fileP =
@"F:\Диплом\MedicalManagementSystemDatabase\MedicalManagementSyste
mDatabase\Doc\Posts.docx";
            try { System.Diagnostics.Process.Start(fileP); }
            catch (Exception ex) { MessageBox.Show("Ошибка чтения
файла" + ex.Message); }
        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            string fileP =
@"F:\Диплом\MedicalManagementSystemDatabase\MedicalManagementSyste
mDatabase\Doc\Sold.docx";
            try { System.Diagnostics.Process.Start(fileP); }
            catch (Exception ex) { MessageBox.Show("Ошибка чтения
файла" + ex.Message); }
        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            string fileP =
@"F:\Диплом\MedicalManagementSystemDatabase\MedicalManagementSyste
mDatabase\Doc\Services.docx";
            try { System.Diagnostics.Process.Start(fileP); }
            catch (Exception ex) { MessageBox.Show("Ошибка чтения
файла" + ex.Message); }
        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            string fileP =
@"F:\Диплом\MedicalManagementSystemDatabase\MedicalManagementSyste
mDatabase\Doc\FinRep.docx";
            try { System.Diagnostics.Process.Start(fileP); }
            catch (Exception ex) { MessageBox.Show("Ошибка чтения
файла" + ex.Message); }
        }
    }
```

```
}
using Npgsql;

using System;

using System.Data;

using System.Windows.Forms;


namespace MedicalManagementSystemDatabase

{

    public partial class Frm_Requests : Form

    {

        private string id = "";

        private int intRow = 0;

        public Frm_Requests()

        {

            InitializeComponent();

            loadData("");

            resetMe();

        }


        private void loadData(string keyword)

        {

            CRUD.sql = "";


            CRUD.sql = "SELECT \"KOD_SOTRUDNIKA\",
\"FIO_SOTRUDNIKA\" FROM public.\"SOTRUDNIK\"";


            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);


            DataTable dtPer = CRUD.PerformCRUD(CRUD.cmd);
```

```
            personComboBox.DataSource = dtPer;
            personComboBox.DisplayMember = "FIO_SOTRUDNIKA";
            personComboBox.ValueMember = "KOD_SOTRUDNIKA";

            CRUD.sql = "";

            CRUD.sql = "SELECT \"KOD_FIRMI_POSTAVSHIKA\",
\"NAIM_FIRMI_POSTAVSHIKA\" FROM public.\"FIRMA_POSTAVSHIK\"";

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            DataTable dtFir = CRUD.PerformCRUD(CRUD.cmd);

            firmaComboBox.DataSource = dtFir;
            firmaComboBox.DisplayMember =
"NAIM_FIRMI_POSTAVSHIKA";
            firmaComboBox.ValueMember = "KOD_FIRMI_POSTAVSHIKA";

            CRUD.sql = "";

            CRUD.sql = "SELECT \"NOMERDOGOVORA_NA_POSTAVKU\" FROM
public.\"DOGOVOR_NA_POSTAVKU\"";

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

            DataTable dtZak = CRUD.PerformCRUD(CRUD.cmd);

            zakComboBox.DataSource = dtZak;
```

```
            zakComboBox.DisplayMember =
"NOMERDOGOVORA_NA_POSTAVKU";
            zakComboBox.ValueMember = "NOMERDOGOVORA_NA_POSTAVKU";


            CRUD.sql = "";


            CRUD.sql = "select zp.\"NOMER_ZAYAVKI\",\r\n\t " +
                "  zp.\"DATA_ZAYAVKI\",\r\n\t " +
                "  zp.\"SUMMA\",\r\n\t " +
                "  fp.\"NAIM_FIRMI_POSTAVSHIKA\",\r\n\t " +
                "  dp.\"NOMERDOGOVORA_NA_POSTAVKU\",\r\n\t " +
                "  s.\"FIO_SOTRUDNIKA\"\r\n\t " +
                "  from public.\"ZAYAVKA_NA_POSTAVKU\" zp\r\n\t  "
+
                " join public.\"FIRMA_POSTAVSHIK\" fp\r\n\t  " +
                " on zp.\"KOD_FIRMI_POSTAVSHIKA_ID\" =
fp.\"KOD_FIRMI_POSTAVSHIKA\"\r\n\t  " +
                " join public.\"DOGOVOR_NA_POSTAVKU\" dp\r\n\t  "
+
                " on dp.\"NOMERDOGOVORA_NA_POSTAVKU\" =
zp.\"NOMERDOGOVORA_ID\"\r\n\t  " +
                " join public.\"SOTRUDNIK\" s\r\n\t  " +
                " on s.\"KOD_SOTRUDNIKA\" = zp.\"SOTRUDNIK_ID\" "
+
            "WHERE CONCAT(CAST(zp.\"NOMER_ZAYAVKI\" as varchar), '
', zp.\"DATA_ZAYAVKI\") " +
            "LIKE @keyword::varchar ORDER BY zp.\"NOMER_ZAYAVKI\"
ASC";



            string strKeyword = string.Format("%{0}%", keyword);
```

```
            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);

            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;
```

```
            dgv1.Columns[0].HeaderText = "Номер поставки";
            dgv1.Columns[1].HeaderText = "Дата поставки";
            dgv1.Columns[2].HeaderText = "Сумма поставки";
            dgv1.Columns[3].HeaderText = "Фирма-поставщик";
            dgv1.Columns[4].HeaderText = "Заявки";
            dgv1.Columns[5].HeaderText = "ФИО сотрудника";


            dgv1.Columns[0].Width = 100;
            dgv1.Columns[1].Width = 100;
            dgv1.Columns[2].Width = 250;
            dgv1.Columns[3].Width = 250;
            dgv1.Columns[4].Width = 250;
            dgv1.Columns[5].Width = 250;

        }

        private void resetMe()
        {
            this.id = string.Empty;

            sumTextBox.Text = "";

            if (firmaComboBox.Items.Count > 0)
            {
                firmaComboBox.SelectedIndex = 0;
            }
```

```
        if (zakComboBox.Items.Count > 0)
        {
            zakComboBox.SelectedIndex = 0;
        }

        if (personComboBox.Items.Count > 0)
        {
            personComboBox.SelectedIndex = 0;
        }

        updateButton.Text = "Изменить";
        deleteButton.Text = "Удалить";

        keywordTextBox.Clear();

        if (keywordTextBox.CanSelect)
        {
            keywordTextBox.Select();
        }
    }

    private void button7_Click(object sender, EventArgs e)
    {
        var dashBoard = new Frm_Main();
        dashBoard.Show();
        this.Hide();
    }
```

```
        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        private void Frm_Requests_Load(object sender, EventArgs e)
        {
            loadData("");
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();
```

```
            if (firmaComboBox.SelectedItem != null &&
firmaComboBox.SelectedValue != null)

            {

                if (firmaComboBox.SelectedValue is int)

                {

                    CRUD.cmd.Parameters.AddWithValue("Номер счета-
фактуры", firmaComboBox.SelectedValue);

                }

                else if (firmaComboBox.SelectedValue is
DataRowView)

                {

                    CRUD.cmd.Parameters.AddWithValue("Номер счета-
фактуры", Convert.ToInt32((firmaComboBox.SelectedValue as
DataRowView)[firmaComboBox.ValueMember]));

                }

            }


            if (zakComboBox.SelectedItem != null &&
zakComboBox.SelectedValue != null)

            {

                if (zakComboBox.SelectedValue is int)

                {

                    CRUD.cmd.Parameters.AddWithValue("Номер счета-
фактуры", zakComboBox.SelectedValue);

                }

                else if (zakComboBox.SelectedValue is DataRowView)

                {

                    CRUD.cmd.Parameters.AddWithValue("Номер счета-
фактуры", Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]));

                }
```

```
            }


            if (personComboBox.SelectedItem != null &&
personComboBox.SelectedValue != null)

            {

                if (personComboBox.SelectedValue is int)

                {

                    CRUD.cmd.Parameters.AddWithValue("ФИО
Сотрудника", personComboBox.SelectedValue);

                }

                else if (personComboBox.SelectedValue is
DataRowView)

                {

                    CRUD.cmd.Parameters.AddWithValue("ФИО
Сотрудника", Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]));

                }

            }


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))

            {

                CRUD.cmd.Parameters.AddWithValue("id", this.id);

            }

        }




        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)

        {

            if (e.RowIndex != -1)
```

```
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);
                sumTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);

                firmaComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);
                zakComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);
                personComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[5].Value);

            }
        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");

            int zakComboBoxValue = 0;
            int personComboBoxValue = 0;
            int firmaComboBoxValue = 0;

            if (firmaComboBox.SelectedValue is int)
            {
```

```
                firmaComboBoxValue =
(int)firmaComboBox.SelectedValue;

            }

            else if (firmaComboBox.SelectedValue is DataRowView)

            {

                firmaComboBoxValue =
Convert.ToInt32((firmaComboBox.SelectedValue as
DataRowView)[firmaComboBox.ValueMember]);

            }


            if (zakComboBox.SelectedValue is int)

            {

                zakComboBoxValue = (int)zakComboBox.SelectedValue;

            }

            else if (zakComboBox.SelectedValue is DataRowView)

            {

                zakComboBoxValue =
Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]);

            }


            if (personComboBox.SelectedValue is int)

            {

                personComboBoxValue =
(int)personComboBox.SelectedValue;

            }

            else if (personComboBox.SelectedValue is DataRowView)

            {

                personComboBoxValue =
Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]);
```

```
            }


            CRUD.sql = $"INSERT INTO
public.\"ZAYAVKA_NA_POSTAVKU\"(\"DATA_ZAYAVKI\", \"SUMMA\",
\"KOD_FIRMI_POSTAVSHIKA_ID\", " +
                $" \"NOMERDOGOVORA_ID\", \"SOTRUDNIK_ID\")" +
                    $" VALUES ('{formattedDate}',
'{sumTextBox.Text}', '{firmaComboBoxValue}', '{zakComboBoxValue}',
'{personComboBoxValue}')";


            execute(CRUD.sql, "Добавить");


            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


            loadData("");
            resetMe();
        }



        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }


            if (string.IsNullOrEmpty(this.id))
            {
```

```
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите изменить", "Изменить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
            }

            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");

            int zakComboBoxValue = 0;
            int personComboBoxValue = 0;
            int firmaComboBoxValue = 0;

            if (firmaComboBox.SelectedValue is int)
            {
                firmaComboBoxValue =
(int)firmaComboBox.SelectedValue;
            }
            else if (firmaComboBox.SelectedValue is DataRowView)
            {
                firmaComboBoxValue =
Convert.ToInt32((firmaComboBox.SelectedValue as
DataRowView)[firmaComboBox.ValueMember]);
            }

            if (zakComboBox.SelectedValue is int)
            {
                zakComboBoxValue = (int)zakComboBox.SelectedValue;
            }
            else if (zakComboBox.SelectedValue is DataRowView)
```

```
            {
                zakComboBoxValue =
Convert.ToInt32((zakComboBox.SelectedValue as
DataRowView)[zakComboBox.ValueMember]);
            }

            if (personComboBox.SelectedValue is int)
            {
                personComboBoxValue =
(int)personComboBox.SelectedValue;
            }
            else if (personComboBox.SelectedValue is DataRowView)
            {
                personComboBoxValue =
Convert.ToInt32((personComboBox.SelectedValue as
DataRowView)[personComboBox.ValueMember]);
            }

            CRUD.sql = $"UPDATE public.\"ZAYAVKA_NA_POSTAVKU\" " +
                       $"SET " +
                       $"\"DATA_ZAYAVKI\"='{formattedDate}', " +
                       $"\"SUMMA\"='{sumTextBox.Text}', " +
$"\"KOD_FIRMI_POSTAVSHIKA_ID\"='{firmaComboBoxValue}', " +
$"\"NOMERDOGOVORA_ID\"='{zakComboBoxValue}', " +
                       $"\"SOTRUDNIK_ID\"='{personComboBoxValue}'
" +
                       $"WHERE \"NOMER_ZAYAVKI\" = @id::integer";
```

```
            execute(CRUD.sql, "Изменить");


            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


            loadData("");
            resetMe();
        }



        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }


            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }


            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
```

```
            {
                CRUD.sql = "DELETE FROM
public.\"ZAYAVKA_NA_POSTAVKU\" WHERE \"NOMER_ZAYAVKI\" =
@id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();

            }
        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }
```

```
            resetMe();


        }


        private void button2_Click(object sender, EventArgs e)

        {

            var dashBoardDelivery = new Frm_Delivery();

            dashBoardDelivery.Show();

            this.Hide();

        }


        private void button3_Click(object sender, EventArgs e)

        {

            var dashBoardSupplies = new Frm_Supplies();

            dashBoardSupplies.Show();

            this.Hide();

        }

    }

}
using Npgsql;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using static MedicalManagementSystemDatabase.Frm_Login;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_Services : Form
    {
        private string id = "";
        private int intRow = 0;

        public Frm_Services()
        {
            InitializeComponent();
```

```
            loadData("");
            resetMe();

        }

        private void resetMe()
        {
            this.id = string.Empty;

            nameTextBox.Text = "";
            suldTextBox.Text = "";


            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }

        }

        private void loadData(string keyword)
        {

            CRUD.sql = "";

            CRUD.sql = "SELECT \"KOD_USLUGI\", \"NAIM_USLUGI\",
\"STOIMOST\" FROM public.\"USLUGA\"\r\n " +
                "WHERE CONCAT(CAST(\"KOD_USLUGI\" as varchar), '
', \"NAIM_USLUGI\", ' ', \"STOIMOST\") " +
                "LIKE @keyword::varchar ORDER BY \"KOD_USLUGI\"
ASC ";


            string strKeyword = string.Format("%{0}%", keyword);

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
```

```
            }
            else
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;

            dgv1.Columns[0].HeaderText = "Код услуги";
            dgv1.Columns[1].HeaderText = "Наименование услуги";
            dgv1.Columns[2].HeaderText = "Стоимость";

            dgv1.Columns[0].Width = 50;
            dgv1.Columns[1].Width = 150;
            dgv1.Columns[2].Width = 150;

        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardBuyers = new Frm_Buyer();
            dashBoardBuyers.Show();
            this.Hide();
        }

        private void Frm_Services_Load(object sender, EventArgs e)
        {
            loadData("");
```

```
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            CRUD.cmd.Parameters.AddWithValue("Наименование
услуги", nameTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Стоимость",
suldTextBox.Text.Trim());
            //


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);

                nameTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);
                suldTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
            }

        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(nameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста введите данные",
"Добавить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
```

```
                return;
            }

            CRUD.sql = $"INSERT INTO public.\"USLUGA\"( " +
                $" \"NAIM_USLUGI\", \"STOIMOST\")" +
                $" VALUES('{nameTextBox.Text}', " +
                $"'{suldTextBox.Text}')";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (string.IsNullOrEmpty(nameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }


            CRUD.sql = $"UPDATE public.\"USLUGA\" SET " +
                $"\"NAIM_USLUGI\"='{nameTextBox.Text}', " +
                $"\"STOIMOST\"='{suldTextBox.Text}' " +
                $"WHERE \"KOD_USLUGI\" = @id::integer";

            execute(CRUD.sql, "Изменить");
```

```
            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"USLUGA\" WHERE
\"KOD_USLUGI\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }

        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
```

```
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();

        }

        private void button3_Click(object sender, EventArgs e)
        {
            var dashBoardMasters = new Frm_Masters();
            dashBoardMasters.Show();
            this.Hide();
        }

        private void button4_Click(object sender, EventArgs e)
        {
            var dashBoardSupplierCompany = new
Frm_SupplierCompany();
            dashBoardSupplierCompany.Show();
            this.Hide();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            var dashBoardReferenceData = new Frm_ReferenceData();
            dashBoardReferenceData.Show();
            this.Hide();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            var dashBoardProductType = new Frm_ProductType();
            dashBoardProductType.Show();
            this.Hide();
        }

    }
}
using Npgsql;
using System;
using System.Data;
using System.Windows.Forms;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_Sold : Form
    {
        private string id = "";
```

```
        private int intRow = 0;
        public Frm_Sold()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }

        private void resetMe()
        {
            this.id = string.Empty;

            if (fullComboBox.Items.Count > 0)
            {
                fullComboBox.SelectedIndex = 0;
            }

            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }
        }
        private void Frm_Sold_Load(object sender, EventArgs e)
        {
            loadData("");
        }
        private void loadData(string keyword)
        {
            {
                CRUD.sql = "SELECT \"KOD_POKUPATEL\",
\"FIO_POKUPATEL\" FROM public.\"POKUPATEL\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable ddt = CRUD.PerformCRUD(CRUD.cmd);

                fullComboBox.DataSource = ddt;
                fullComboBox.DisplayMember = "FIO_POKUPATEL";
                fullComboBox.ValueMember = "KOD_POKUPATEL";

                CRUD.sql = "";

                CRUD.sql = "select z.\"NOMER_ZAKAZA\",
z.\"DATA_ZAKAZA\", p.\"FIO_POKUPATEL\" from public.\"ZAKAZ\" z " +
                    "join public.\"POKUPATEL\" p ON
z.\"POKUPATEL_ID\" = p.\"KOD_POKUPATEL\"" +
```

```
                "WHERE CONCAT(CAST(\"NOMER_ZAKAZA\" as varchar),
' ', \"DATA_ZAKAZA\", ' ', p.\"FIO_POKUPATEL\") LIKE
@keyword::varchar " +
                "ORDER BY \"NOMER_ZAKAZA\" ASC";

                string strKeyword = string.Format("%{0}%",
keyword);

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
                CRUD.cmd.Parameters.Clear();
                CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

                DataTable dt = CRUD.PerformCRUD(CRUD.cmd);

                if (dt.Rows.Count > 0)
                {
                    intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
                }
                else
                {
                    intRow = 0;
                }

                toolStripStatusLabel1.Text = "Количество строк: "
+ intRow.ToString();

                DataGridView dgv1 = dataGridView1;

                dgv1.MultiSelect = false;
                dgv1.AutoGenerateColumns = true;
                dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

                dgv1.DataSource = dt;

                dgv1.Columns[0].HeaderText = "Номер заказа";
                dgv1.Columns[1].HeaderText = "Дата заказа";
                dgv1.Columns[2].HeaderText = "Покупатель";

                dgv1.Columns[0].Width = 85;
                dgv1.Columns[1].Width = 200;
                dgv1.Columns[2].Width = 200;

            }

        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
```

```
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            NpgsqlParameter npgsqlBuyer =
CRUD.cmd.Parameters.AddWithValue("Покупатель",
fullComboBox.SelectedItem.ToString());

            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            {

                string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");

                CRUD.sql = $"INSERT INTO public.\"ZAKAZ\"
(\"DATA_ZAKAZA\", \"POKUPATEL_ID\") VALUES ('{formattedDate}',
'{fullComboBox.SelectedValue}')";

                execute(CRUD.sql, "Добавить");
```

```
                MessageBox.Show("Запись успешно проведена",
"Добавить данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);

                //updateButton.Text = "Update (" + this.id + ")";
                //deleteButton.Text = "Delete (" + this.id + ")";

            }
        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");

            CRUD.sql = $"UPDATE public.\"ZAKAZ\" SET
\"DATA_ZAKAZA\" = '{formattedDate}', \"POKUPATEL_ID\" =
'{fullComboBox.SelectedValue}' WHERE \"NOMER_ZAKAZA\" =
@id::integer";

            execute(CRUD.sql, "Изменить");
```

```
            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();
        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"ZAKAZ\" WHERE
\"NOMER_ZAKAZA\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }
        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
```

```
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardHandbookDiagnosis = new
Frm_SomeProduct();
            dashBoardHandbookDiagnosis.Show();
            this.Hide();
        }
    }
}
using Npgsql;

using System;

using System.Data;

using System.Windows.Forms;


namespace MedicalManagementSystemDatabase

{

    public partial class Frm_SomeProduct : Form

    {

        private string id = "";

        private int intRow = 0;

        public Frm_SomeProduct()

        {

            InitializeComponent();

            loadData("");

            resetMe();

        }


        private void resetMe()
```

```
{
    this.id = string.Empty;

    stoimTextBox.Text = "";

    if (zakComboBox.Items.Count > 0)
    {
        zakComboBox.SelectedIndex = 0;
    }

    if (fullComboBox.Items.Count > 0)
    {
        fullComboBox.SelectedIndex = 0;
    }

    if (personComboBox.Items.Count > 0)
    {
        personComboBox.SelectedIndex = 0;
    }

    updateButton.Text = "Изменить";
    deleteButton.Text = "Удалить";

    keywordTextBox.Clear();

    if (keywordTextBox.CanSelect)
    {
        keywordTextBox.Select();
```

```
            }
        }
        private void Frm_Sold_Load(object sender, EventArgs e)
        {
            loadData("");
        }
        private void loadData(string keyword)
        {
            {
                CRUD.sql = "SELECT \"KOD_POKUPATEL\",
\"FIO_POKUPATEL\" FROM public.\"POKUPATEL\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable ddt = CRUD.PerformCRUD(CRUD.cmd);

                fullComboBox.DataSource = ddt;
                fullComboBox.DisplayMember = "FIO_POKUPATEL";
                fullComboBox.ValueMember = "KOD_POKUPATEL";

                CRUD.sql = "";

                CRUD.sql = "SELECT \"KOD_SOTRUDNIKA\",
\"FIO_SOTRUDNIKA\" FROM public.\"SOTRUDNIK\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable dtPer = CRUD.PerformCRUD(CRUD.cmd);
```

```
                personComboBox.DataSource = dtPer;
                personComboBox.DisplayMember = "FIO_SOTRUDNIKA";
                personComboBox.ValueMember = "KOD_SOTRUDNIKA";

                CRUD.sql = "";

                CRUD.sql = "SELECT \"NOMER_ZAKAZA\" FROM
public.\"ZAKAZ\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable dtZak = CRUD.PerformCRUD(CRUD.cmd);

                zakComboBox.DataSource = dtZak;
                zakComboBox.DisplayMember = "NOMER_ZAKAZA";
                zakComboBox.ValueMember = "NOMER_ZAKAZA";

                CRUD.sql = "";

                CRUD.sql = "SELECT pn.\"NOMER_PROD_TOVARA\", \r\n
" +
                    "     pn.\"DATA_PROD_TOVARA\", \r\n   " +
                    "    pn.\"SUMMA\", \r\n   " +
                    "    pn.\"ZAKAZ_ID\", \r\n   " +
                    "    s.\"FIO_SOTRUDNIKA\", \r\n   " +
                    "    p.\"FIO_POKUPATEL\" \r\nFROM
public.\"PRODAJA_TOVARA\" pn\r\n" +
                    "JOIN public.\"SOTRUDNIK\" s ON
s.\"KOD_SOTRUDNIKA\" = pn.\"SOTRUDNIK_ID\"\r\n" +
```

```
                    "JOIN public.\"ZAKAZ\" z ON z.\"POKUPATEL_ID\"
= pn.\"ZAKAZ_ID\"\r\n" +

                    "JOIN public.\"POKUPATEL\" p ON
z.\"POKUPATEL_ID\" = p.\"KOD_POKUPATEL\"" +

                    "WHERE CONCAT(CAST(\"NOMER_PROD_TOVARA\" as
varchar), ' ', \"DATA_PROD_TOVARA\", ' ', pn.\"SUMMA\", " +

                    "' ', pn.\"ZAKAZ_ID\", ' ',
s.\"FIO_SOTRUDNIKA\", ' ', p.\"FIO_POKUPATEL\") LIKE
@keyword::varchar " +

                    "ORDER BY \"NOMER_PROD_TOVARA\" ASC";


                string strKeyword = string.Format("%{0}%",
keyword);


                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                CRUD.cmd.Parameters.Clear();

                CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);


                DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


                if (dt.Rows.Count > 0)

                {

                    intRow =
Convert.ToInt32(dt.Rows.Count.ToString());

                }

                else

                {

                    intRow = 0;

                }
```

```
                toolStripStatusLabel1.Text = "Количество строк: "
+ intRow.ToString();


                DataGridView dgv1 = dataGridView1;


                dgv1.MultiSelect = false;

                dgv1.AutoGenerateColumns = true;

                dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;


                dgv1.DataSource = dt;


                dgv1.Columns[0].HeaderText = "Номер товара";

                dgv1.Columns[1].HeaderText = "Дата продажи";

                dgv1.Columns[2].HeaderText = "Стоимость";

                dgv1.Columns[3].HeaderText = "Заказ";

                dgv1.Columns[4].HeaderText = "Сотрудник";

                dgv1.Columns[5].HeaderText = "Покупатель";


                dgv1.Columns[0].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[4].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[5].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
```

```
            }


        }


        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }


        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("Стоимость",
stoimTextBox.Text.Trim());


            NpgsqlParameter npgsqlZak =
CRUD.cmd.Parameters.AddWithValue("Заказ",
zakComboBox.SelectedItem.ToString());
            NpgsqlParameter npgsqlBuyer =
CRUD.cmd.Parameters.AddWithValue("Покупатель",
fullComboBox.SelectedItem.ToString());
            NpgsqlParameter npgsqlPer =
CRUD.cmd.Parameters.AddWithValue("Сотрудник",
personComboBox.SelectedValue);


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
```

```
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
```

```
                DataGridView dgv1 = dataGridView1;


                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);
                stoimTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
                zakComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);
                personComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);
                fullComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[5].Value);


                //updateButton.Text = "Update (" + this.id + ")";
                //deleteButton.Text = "Delete (" + this.id + ")";


            }
        }


        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardHandbookDiagnosis = new Frm_Sold();
            dashBoardHandbookDiagnosis.Show();
            this.Hide();
        }


        private void button10_Click_1(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
```

```
            this.Hide();
        }

        private void NextButton_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_SostavNakl();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void insertButton_Click_1(object sender, EventArgs
e)
        {
            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");

            CRUD.sql = $"INSERT INTO public.\"PRODAJA_TOVARA\" " +
                       $"(\"DATA_PROD_TOVARA\", \"SUMMA\",
\"ZAKAZ_ID\", \"POKUPATEL_ID\", \"SOTRUDNIK_ID\") " +
                       $"VALUES ('{formattedDate}',
'{stoimTextBox.Text}', " +
                       $"'{zakComboBox.SelectedValue}',
'{fullComboBox.SelectedValue}',
'{personComboBox.SelectedValue}')";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);

            loadData("");
```

```
            resetMe();
        }



        private void updateButton_Click_1(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }


            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите изменить", "Изменить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
            }


            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");


            CRUD.sql = $"UPDATE public.\"PRODAJA_TOVARA\" " +
                       $"SET \"DATA_PROD_TOVARA\" =
'{formattedDate}', " +
                       $"\"SUMMA\" = '{stoimTextBox.Text}', " +
                       $"\"ZAKAZ_ID\" =
'{zakComboBox.SelectedValue}', " +
```

```
                        $"\"POKUPATEL_ID\" =
'{fullComboBox.SelectedValue}', " +
                        $"\"SOTRUDNIK_ID\" =
'{personComboBox.SelectedValue}' " +
                        $" WHERE \"NOMER_PROD_TOVARA\" =
@id::integer";


            execute(CRUD.sql, "Изменить");


            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


            loadData("");


            resetMe();
        }



        private void deleteButton_Click_1(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }


            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите удалить", "Удалить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
```

```
                return;
            }

            if (MessageBox.Show("Вы уверены, что хотите удалить
запись?", "Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM public.\"PRODAJA_TOVARA\"
WHERE \"NOMER_PROD_TOVARA\" = @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }
        }


        private void searchButton_Click_1(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
```

```
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();
        }
    }
}
using Npgsql;
using System;
using System.Data;
using System.Windows.Forms;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_SostavNakl : Form
    {
        private string id = "";
        private int intRow = 0;
        public Frm_SostavNakl()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }

        private void resetMe()
```

```
        {
            this.id = string.Empty;

            kolTextBox.Text = "";
            stoimTextBox.Text = "";

            if (zakComboBox.Items.Count > 0)
            {
                zakComboBox.SelectedIndex = 0;
            }

            if (fullComboBox.Items.Count > 0)
            {
                fullComboBox.SelectedIndex = 0;
            }

            if (personComboBox.Items.Count > 0)
            {
                personComboBox.SelectedIndex = 0;
            }

            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
```

```
                keywordTextBox.Select();
            }
        }
        private void Frm_Sold_Load(object sender, EventArgs e)
        {
            loadData("");
        }
        private void loadData(string keyword)
        {
            {
                CRUD.sql = "SELECT \"KOD_POKUPATEL\",
\"FIO_POKUPATEL\" FROM public.\"POKUPATEL\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable ddt = CRUD.PerformCRUD(CRUD.cmd);

                fullComboBox.DataSource = ddt;
                fullComboBox.DisplayMember = "FIO_POKUPATEL";
                fullComboBox.ValueMember = "KOD_POKUPATEL";

                CRUD.sql = "";

                CRUD.sql = "SELECT \"KOD_SOTRUDNIKA\",
\"FIO_SOTRUDNIKA\" FROM public.\"SOTRUDNIK\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable dtPer = CRUD.PerformCRUD(CRUD.cmd);
```

```
                personComboBox.DataSource = dtPer;
                personComboBox.DisplayMember = "FIO_SOTRUDNIKA";
                personComboBox.ValueMember = "KOD_SOTRUDNIKA";

                CRUD.sql = "";

                CRUD.sql = "SELECT \"KOD_TOVARA\", \"NAIM_TOVARA\"
FROM public.\"TOVAR\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable dtZak = CRUD.PerformCRUD(CRUD.cmd);

                zakComboBox.DataSource = dtZak;
                zakComboBox.DisplayMember = "NAIM_TOVARA";
                zakComboBox.ValueMember = "KOD_TOVARA";

                CRUD.sql = "";

                CRUD.sql = "SELECT sn.\"NOMER_SOSTAVA\", \r\n   "
+
                    " sn.\"KOL_TOVARA\", \r\n " +
                    "   sn.\"STOIM\", \r\n  " +
                    "  t.\"NAIM_TOVARA\", \r\n  " +
                    "  s.\"FIO_SOTRUDNIKA\", \r\n   " +
                    " p.\"FIO_POKUPATEL\", \r\n   " +
                    " SUM(sn.\"KOL_TOVARA\" * sn.\"STOIM\") AS
\"Общая Стоимость\"\r\n" +
```

```
                    " FROM public.\"SOSTAV_NAKLADNOY\" sn\r\n" +

                    " JOIN public.\"SOTRUDNIK\" s ON
s.\"KOD_SOTRUDNIKA\" = sn.\"NOMER_PROD_ID\"\r\n" +

                    " JOIN public.\"ZAKAZ\" z ON
z.\"POKUPATEL_ID\" = sn.\"ZAKAZ_ID\"\r\n" +

                    " JOIN public.\"POKUPATEL\" p ON
z.\"POKUPATEL_ID\" = p.\"KOD_POKUPATEL\"\r\n" +

                    " JOIN public.\"TOVAR\" t ON sn.\"TOVAR_ID\" =
t.\"KOD_TOVARA\"\r\n" +

                    " GROUP BY sn.\"NOMER_SOSTAVA\",
sn.\"KOL_TOVARA\", sn.\"STOIM\", t.\"NAIM_TOVARA\",
s.\"FIO_SOTRUDNIKA\", p.\"FIO_POKUPATEL\"" +

                    " HAVING CONCAT(CAST(\"NOMER_SOSTAVA\" as
varchar), ' ', \"NAIM_TOVARA\") LIKE @keyword::varchar " +

                    " ORDER BY \"NOMER_SOSTAVA\" ASC";


                string strKeyword = string.Format("%{0}%",
keyword);


                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                CRUD.cmd.Parameters.Clear();

                CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);


                DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


                if (dt.Rows.Count > 0)

                {

                    intRow =
Convert.ToInt32(dt.Rows.Count.ToString());

                }

                else
```

```
                {
                    intRow = 0;
                }

                toolStripStatusLabel1.Text = "Количество строк: "
+ intRow.ToString();

                DataGridView dgv1 = dataGridView1;

                dgv1.MultiSelect = false;
                dgv1.AutoGenerateColumns = true;
                dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

                dgv1.DataSource = dt;

                dgv1.Columns[0].HeaderText = "Номер";
                dgv1.Columns[1].HeaderText = "Кол-во товара";
                dgv1.Columns[2].HeaderText = "Стоимость";
                dgv1.Columns[3].HeaderText = "Наименование
товара";
                dgv1.Columns[4].HeaderText = "Сотрудник";
                dgv1.Columns[5].HeaderText = "Покупатель";

                dgv1.Columns[0].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
                dgv1.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
                dgv1.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
```

```
                dgv1.Columns[3].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[4].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

                dgv1.Columns[5].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;


            }


        }


        private void execute(string mySQL, string param)

        {

            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);

            addParameters(param);

            CRUD.PerformCRUD(CRUD.cmd);

        }


        private void addParameters(string str)

        {

            CRUD.cmd.Parameters.Clear();


            CRUD.cmd.Parameters.AddWithValue("Кол-во товара",
kolTextBox.Text.Trim());

            CRUD.cmd.Parameters.AddWithValue("Стоимость",
stoimTextBox.Text.Trim());


            NpgsqlParameter npgsqlZak =
CRUD.cmd.Parameters.AddWithValue("Наименование товара",
zakComboBox.SelectedItem.ToString());
```

```
            NpgsqlParameter npgsqlBuyer =
CRUD.cmd.Parameters.AddWithValue("Покупатель",
fullComboBox.SelectedItem.ToString());

            NpgsqlParameter npgsqlPer =
CRUD.cmd.Parameters.AddWithValue("Сотрудник",
personComboBox.SelectedValue);


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))

            {

                CRUD.cmd.Parameters.AddWithValue("id", this.id);

            }

        }


        private void button7_Click(object sender, EventArgs e)

        {

            var dashBoard = new Frm_Main();

            dashBoard.Show();

            this.Hide();

        }


        private void button10_Click(object sender, EventArgs e)

        {

            var dashBoardLogin = new Frm_Login();

            dashBoardLogin.Show();

            this.Hide();

        }


        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
```

```
        {
            Application.Exit();
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);
                kolTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);
                stoimTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);

                zakComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);
                personComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[4].Value);
                fullComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[5].Value);

                //updateButton.Text = "Update (" + this.id + ")";
                //deleteButton.Text = "Delete (" + this.id + ")";

            }
        }
```

```
        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardHandbookDiagnosis = new Frm_Sold();
            dashBoardHandbookDiagnosis.Show();
            this.Hide();
        }

        private void button10_Click_1(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void NextButton_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_SostavNakl();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void insertButton_Click_1(object sender, EventArgs
e)
        {

            CRUD.sql = $"INSERT INTO public.\"SOSTAV_NAKLADNOY\" "
+
                       $"(\"KOL_TOVARA\", \"STOIM\", \"TOVAR_ID\",
\"ZAKAZ_ID\", \"NOMER_PROD_ID\") " +
```

```
                        $"VALUES ('{kolTextBox.Text}',
'{stoimTextBox.Text}', " +
                        $"'{zakComboBox.SelectedValue}',
'{fullComboBox.SelectedValue}',
'{personComboBox.SelectedValue}')";


            execute(CRUD.sql, "Добавить");


            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


            loadData("");


            resetMe();
        }



        private void updateButton_Click_1(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }


            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите изменить", "Изменить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
```

```
            }



            CRUD.sql = $"UPDATE public.\"SOSTAV_NAKLADNOY\" " +
                       $"SET \"KOL_TOVARA\" = '{kolTextBox.Text}',
" +
                       $"\"STOIM\" = '{stoimTextBox.Text}', " +
                       $"\"TOVAR_ID\" =
'{zakComboBox.SelectedValue}', " +
                       $"\"ZAKAZ_ID\" =
'{fullComboBox.SelectedValue}', " +
                       $"\"NOMER_PROD_ID\" =
'{personComboBox.SelectedValue}' " +
                       $" WHERE \"NOMER_SOSTAVA\" = @id::integer";


            execute(CRUD.sql, "Изменить");


            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);


            loadData("");


            resetMe();
        }



        private void deleteButton_Click_1(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
```

```
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите строку,
которую хотите удалить", "Удалить данные", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы уверены, что хотите удалить
запись?", "Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM
public.\"SOSTAV_NAKLADNOY\" WHERE \"NOMER_SOSTAVA\" =
@id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK, MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }
        }
```

```
        private void searchButton_Click_1(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();
        }

        private void button2_Click_1(object sender, EventArgs e)
        {
            var dashBoardHandbookDiagnosis = new
Frm_SomeProduct();
            dashBoardHandbookDiagnosis.Show();
            this.Hide();
        }

        private void button10_Click_2(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
```

```
            this.Hide();

        }

    }

}

using Npgsql;
using System;
using System.Data;
using System.Drawing;
using System.Windows.Forms;
using static MedicalManagementSystemDatabase.Frm_Login;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_SupplierCompany : Form
    {
        private string id = "";
        private int intRow = 0;

        public Frm_SupplierCompany()
        {
            InitializeComponent();
            loadData("");
            resetMe();

        }

        private void resetMe()
        {
            this.id = string.Empty;

            fullnameTextBox.Text = "";
            reqTextBox.Text = "";


            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
                keywordTextBox.Select();
            }

        }

        private void loadData(string keyword)
        {
```

```
            CRUD.sql = "";

            CRUD.sql = "SELECT * FROM
public.\"FIRMA_POSTAVSHIK\"\r\n " +
                "WHERE CONCAT(CAST(\"KOD_FIRMI_POSTAVSHIKA\" as
varchar), ' ', \"NAIM_FIRMI_POSTAVSHIKA\", ' ', \"REKVIZITI\")" +
                "LIKE @keyword::varchar ORDER BY
\"KOD_FIRMI_POSTAVSHIKA\" ASC ";


            string strKeyword = string.Format("%{0}%", keyword);

            CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
            CRUD.cmd.Parameters.Clear();
            CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

            DataTable dt = CRUD.PerformCRUD(CRUD.cmd);


            if (dt.Rows.Count > 0)
            {
                intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
            }
            else
            {
                intRow = 0;
            }

            toolStripStatusLabel1.Text = "Количество строк: " +
intRow.ToString();

            DataGridView dgv1 = dataGridView1;

            dgv1.MultiSelect = false;
            dgv1.AutoGenerateColumns = true;
            dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

            dgv1.DataSource = dt;

            dgv1.Columns[0].HeaderText = "Код фирмы-поставщика";
            dgv1.Columns[1].HeaderText = "Наименование фирмы-
поставщика";
            dgv1.Columns[2].HeaderText = "Реквизиты";

            dgv1.Columns[0].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
            dgv1.Columns[1].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;
```

```
            dgv1.Columns[2].AutoSizeMode =
DataGridViewAutoSizeColumnMode.AllCells;

        }

        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Main();
            dashBoard.Show();
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardBuyers = new Frm_Buyer();
            dashBoardBuyers.Show();
            this.Hide();
        }

        private void Frm_SupplierCompany_Load(object sender,
EventArgs e)
        {
            loadData("");
        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();

            CRUD.cmd.Parameters.AddWithValue("Наименование фирмы-
поставщика", fullnameTextBox.Text.Trim());
            CRUD.cmd.Parameters.AddWithValue("Реквизиты",
reqTextBox.Text.Trim());
            //

            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
```

```
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);

                fullnameTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[1].Value);
                reqTextBox.Text =
Convert.ToString(dgv1.CurrentRow.Cells[2].Value);
            }

        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста введите данные",
"Добавить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            CRUD.sql = $"INSERT INTO public.\"FIRMA_POSTAVSHIK\"(
" +
                $" \"NAIM_FIRMI_POSTAVSHIKA\", \"REKVIZITI\")" +
                $" VALUES('{fullnameTextBox.Text}', " +
                $" '{reqTextBox.Text}')";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

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

```
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (string.IsNullOrEmpty(fullnameTextBox.Text.Trim()))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }


            CRUD.sql = $"UPDATE public.\"FIRMA_POSTAVSHIK\" SET "
+
                $"
\"NAIM_FIRMI_POSTAVSHIKA\"='{fullnameTextBox.Text}'," +
                $" \"REKVIZITI\"='{reqTextBox.Text}'" +
                $"WHERE \"KOD_FIRMI_POSTAVSHIKA\" = @id::integer";

            execute(CRUD.sql, "Изменить");

            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
```

```
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM
public.\"FIRMA_POSTAVSHIK\" WHERE \"KOD_FIRMI_POSTAVSHIKA\" =
@id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();
            }

        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }

            resetMe();

        }

        private void button3_Click(object sender, EventArgs e)
        {
            var dashBoardReferenceData = new Frm_ReferenceData();
            dashBoardReferenceData.Show();
            this.Hide();
        }

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

```
        {
            var dashBoardMasters = new Frm_Masters();
            dashBoardMasters.Show();
            this.Hide();
        }

        private void button5_Click(object sender, EventArgs e)
        {
            var dashBoardServices = new Frm_Services();
            dashBoardServices.Show();
            this.Hide();
        }

        private void button6_Click(object sender, EventArgs e)
        {
            var dashBoardProductType = new Frm_ProductType();
            dashBoardProductType.Show();
            this.Hide();
        }

    }
}
using Npgsql;
using System;
using System.Data;
using System.Windows.Forms;

namespace MedicalManagementSystemDatabase
{
    public partial class Frm_Supplies : Form
    {
        private string id = "";
        private int intRow = 0;

        public Frm_Supplies()
        {
            InitializeComponent();
            loadData("");
            resetMe();
        }

        private void resetMe()
        {
            this.id = string.Empty;

            updateButton.Text = "Изменить";
            deleteButton.Text = "Удалить";

            keywordTextBox.Clear();

            if (keywordTextBox.CanSelect)
            {
```

```
                keywordTextBox.Select();
            }
        }
        private void loadData(string keyword)
        {
            {
                CRUD.sql = "";

                CRUD.sql = "SELECT \"KOD_FIRMI_POSTAVSHIKA\",
\"NAIM_FIRMI_POSTAVSHIKA\" FROM public.\"FIRMA_POSTAVSHIK\"";

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);

                DataTable dtFir = CRUD.PerformCRUD(CRUD.cmd);

                firmaComboBox.DataSource = dtFir;
                firmaComboBox.DisplayMember =
"NAIM_FIRMI_POSTAVSHIKA";
                firmaComboBox.ValueMember =
"KOD_FIRMI_POSTAVSHIKA";

                CRUD.sql = "";

                CRUD.sql = "SELECT * FROM
public.\"DOGOVOR_NA_POSTAVKU\"" +
                "WHERE CONCAT(CAST(\"NOMERDOGOVORA_NA_POSTAVKU\"
as varchar)) " +
                "LIKE @keyword::varchar ORDER BY
\"NOMERDOGOVORA_NA_POSTAVKU\" ASC";

                string strKeyword = string.Format("%{0}%",
keyword);

                CRUD.cmd = new NpgsqlCommand(CRUD.sql, CRUD.con);
                CRUD.cmd.Parameters.Clear();
                CRUD.cmd.Parameters.AddWithValue("keyword",
strKeyword);

                DataTable dt = CRUD.PerformCRUD(CRUD.cmd);

                if (dt.Rows.Count > 0)
                {
                    intRow =
Convert.ToInt32(dt.Rows.Count.ToString());
                }
                else
                {
                    intRow = 0;
                }

                toolStripStatusLabel1.Text = "Количество строк: "
+ intRow.ToString();
```

```
                DataGridView dgv1 = dataGridView1;

                dgv1.MultiSelect = false;
                dgv1.AutoGenerateColumns = true;
                dgv1.SelectionMode =
DataGridViewSelectionMode.FullRowSelect;

                dgv1.DataSource = dt;

                dgv1.Columns[0].HeaderText = "Номер договора на
поставку";
                dgv1.Columns[1].HeaderText = "Дата составления
договора";
                dgv1.Columns[2].HeaderText = "Срок поставки";
                dgv1.Columns[3].HeaderText = "Наименование фирмы-
поставщика";


                dgv1.Columns[0].Width = 150;
                dgv1.Columns[1].Width = 200;
                dgv1.Columns[2].Width = 200;
                dgv1.Columns[3].Width = 200;

            }

        }
        private void execute(string mySQL, string param)
        {
            CRUD.cmd = new NpgsqlCommand(mySQL, CRUD.con);
            addParameters(param);
            CRUD.PerformCRUD(CRUD.cmd);
        }

        private void addParameters(string str)
        {
            CRUD.cmd.Parameters.Clear();


            if (str == "Изменить" || str == "Удалить" &&
!string.IsNullOrEmpty(this.id))
            {
                CRUD.cmd.Parameters.AddWithValue("id", this.id);
            }
        }



        private void button7_Click(object sender, EventArgs e)
        {
            var dashBoard = new Frm_Requests();
            dashBoard.Show();
```

```
            this.Hide();
        }

        private void button10_Click(object sender, EventArgs e)
        {
            var dashBoardLogin = new Frm_Login();
            dashBoardLogin.Show();
            this.Hide();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var dashBoardDelivery = new Frm_Delivery();
            dashBoardDelivery.Show();
            this.Hide();
        }

        private void Guna2PictureBox2_Click(object sender,
EventArgs e)
        {
            Application.Exit();
        }

        private void Frm_Supplies_Load(object sender, EventArgs e)
        {
            loadData("");
        }

        private void insertButton_Click(object sender, EventArgs
e)
        {
            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");
            string formattedDateEnd =
DateTimePicker2.Value.ToString("yyyy-MM-dd");


            CRUD.sql = $"INSERT INTO
public.\"DOGOVOR_NA_POSTAVKU\"(\"DATA_SOST_DOGOVORA \",
\"SROK_POSTAVKI\", \"FIRMA_ID\")" +
                $" VALUES ('{formattedDate}',
'{formattedDateEnd}', '{firmaComboBox.SelectedIndex}')";

            execute(CRUD.sql, "Добавить");

            MessageBox.Show("Запись успешно проведена", "Добавить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();
```

```
        }

        private void updateButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }

            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите изменить", "Изменить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }
            string formattedDate =
DateTimePicker1.Value.ToString("yyyy-MM-dd");
            string formattedDateEnd =
DateTimePicker2.Value.ToString("yyyy-MM-dd");


            CRUD.sql = $"UPDATE public.\"DOGOVOR_NA_POSTAVKU\" " +
                $"SET \"DATA_SOST_DOGOVORA \"='{formattedDate}'," 
+
                $" \"SROK_POSTAVKI\"='{formattedDateEnd}', " +
                $" \"FIRMA_ID\"='{firmaComboBox.SelectedIndex}' "
+
                $"WHERE \"NOMERDOGOVORA_NA_POSTAVKU\" =
@id::integer";

            execute(CRUD.sql, "Изменить");

            MessageBox.Show("Запись была обновлена", "Изменить
данные", MessageBoxButtons.OK,
    MessageBoxIcon.Information);

            loadData("");

            resetMe();

        }

        private void deleteButton_Click(object sender, EventArgs
e)
        {
            if (dataGridView1.Rows.Count == 0)
            {
                return;
            }
```

```
            if (string.IsNullOrEmpty(this.id))
            {
                MessageBox.Show("Пожалуйста выберите стоку которую
хотите удалить", "Удалить данные", MessageBoxButtons.OK,
                    MessageBoxIcon.Exclamation);
                return;
            }

            if (MessageBox.Show("Вы точно хотите удалить запись?",
"Удалить данные",
                MessageBoxButtons.YesNo, MessageBoxIcon.Question,
MessageBoxDefaultButton.Button1) == DialogResult.Yes)
            {
                CRUD.sql = "DELETE FROM
public.\"DOGOVOR_NA_POSTAVKU\" WHERE \"NOMERDOGOVORA_NA_POSTAVKU\"
= @id::integer";

                execute(CRUD.sql, "Удалить");

                MessageBox.Show("Запись была удалена", "Удалить
данные", MessageBoxButtons.OK,
        MessageBoxIcon.Information);

                loadData("");

                resetMe();

            }
        }

        private void searchButton_Click(object sender, EventArgs
e)
        {
            if (string.IsNullOrEmpty(keywordTextBox.Text.Trim()))
            {
                loadData("");
            }
            else
            {
                loadData(keywordTextBox.Text.Trim());
            }
        }

        private void dataGridView1_CellClick(object sender,
DataGridViewCellEventArgs e)
        {
            if (e.RowIndex != -1)
            {
                DataGridView dgv1 = dataGridView1;

                this.id =
Convert.ToString(dgv1.CurrentRow.Cells[0].Value);
```

```
                firmaComboBox.SelectedItem =
Convert.ToString(dgv1.CurrentRow.Cells[3].Value);
            }

        }
    }
}
namespace MedicalManagementSystemDatabase
{
    public class logindata
    {
        public string role;
        public string name;
        public string photo;
    }
}
using System;
using System.Windows.Forms;

namespace MedicalManagementSystemDatabase
{
    internal static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Frm_Login());
        }
    }
}
using Npgsql;
using System;
using System.Data;
using System.Windows.Forms;

namespace MedicalManagementSystemDatabase
{
    internal class CRUD
    {
        private static string getConnectionString()
        {
            string host = "Host=localhost;";
            string port = "Port=5432;";
            string db = "Database=BeautyShop;";
            string user = "Username=postgres;";
            string pass = "Password=root;";
```

```
            string conString = string.Format("{0}{1}{2}{3}{4}",
host, port, db, user, pass);

            return conString;

        }

        public static NpgsqlConnection con = new
NpgsqlConnection(getConnectionString());
        public static NpgsqlCommand cmd = default(NpgsqlCommand);
        public static string sql = string.Empty;

        public static DataTable PerformCRUD(NpgsqlCommand com)
        {
            NpgsqlDataAdapter da = default(NpgsqlDataAdapter);
            DataTable dt = new DataTable();

            try
            {
                da = new NpgsqlDataAdapter();
                da.SelectCommand = com;
                da.Fill(dt);

                return dt;
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка соеденения с БД " +
ex.Message, "Ошибка соеденения ",
                        MessageBoxButtons.OK, MessageBoxIcon.Error);
                dt = null;
            }

            return dt;
        }
    }
}
```

Приложение Д

(обязательное)

Информационная автоматизированная система салона красоты

Презентация

Клиент – серверная автоматизированная система для медицинских организаций

АВТОР РАБОТЫ: СТУДЕНТКА ГРУППЫ 19ВОЭ1

ПЕРШИНА В.В.

НАУЧНЫЙ РУКОВОДИТЕЛЬ: Д.Т.Н., ПРОФЕССОР КАФЕДРЫ ВТ

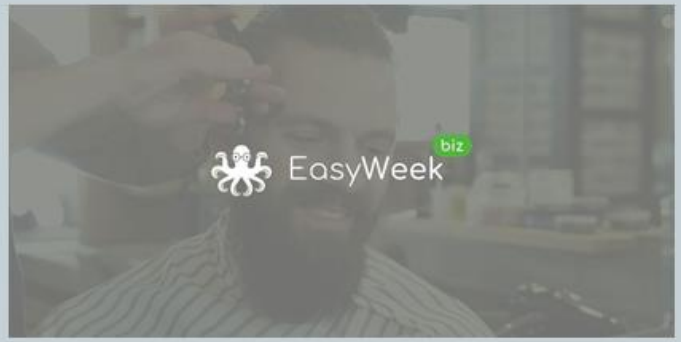
ЗИНКИН С.А.

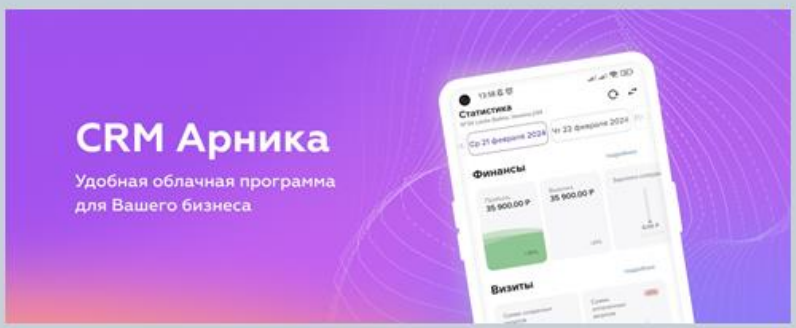
Цель

- Целью работы является создание клиент – серверной автоматизированной системы для медицинских организаций. Объектом исследования является процесс контроля работы салона красоты и учета медицинских процедур.

Аналоги

- EasyWeek
- ПрофиГид
- Арника



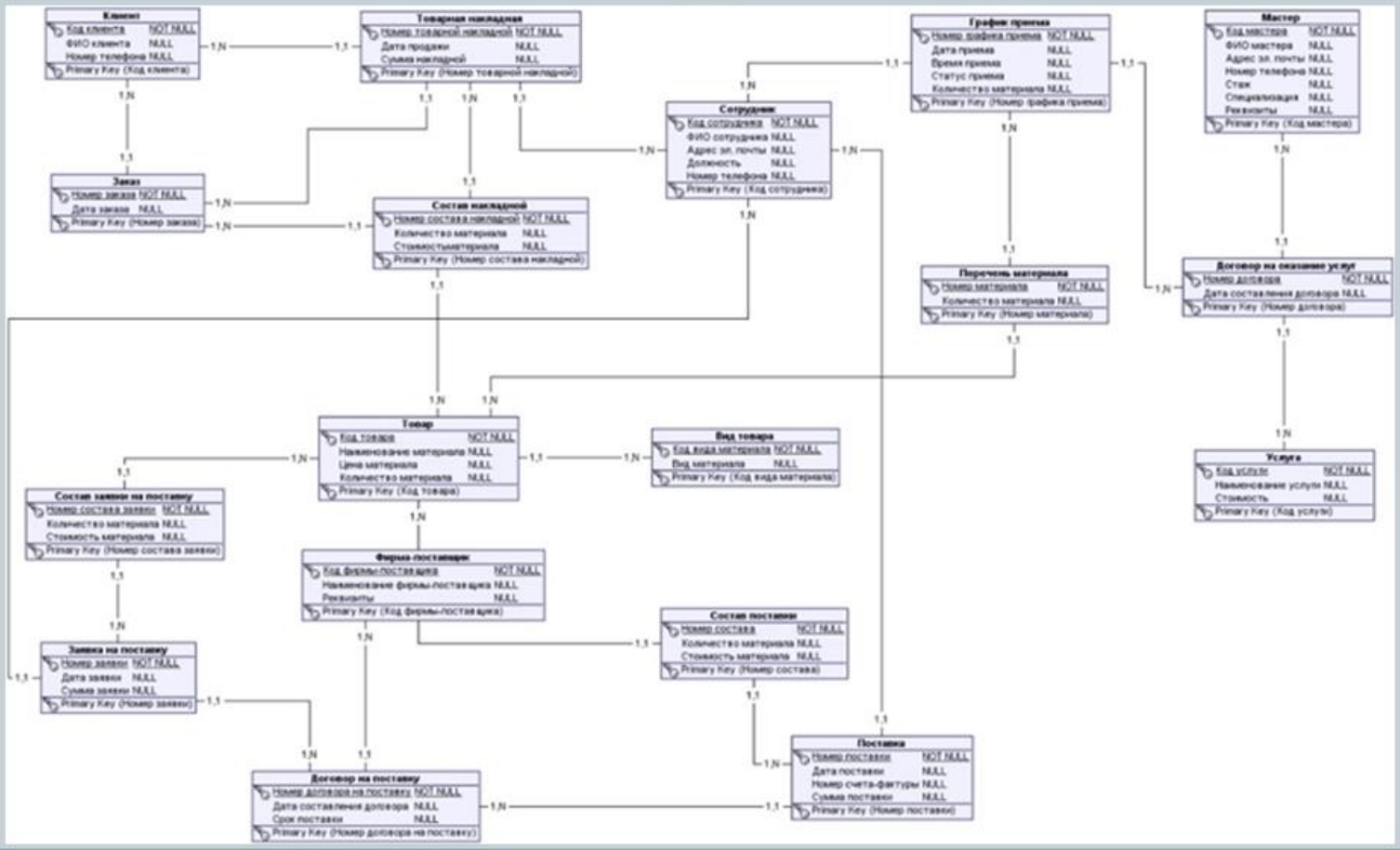


ПрофиГид

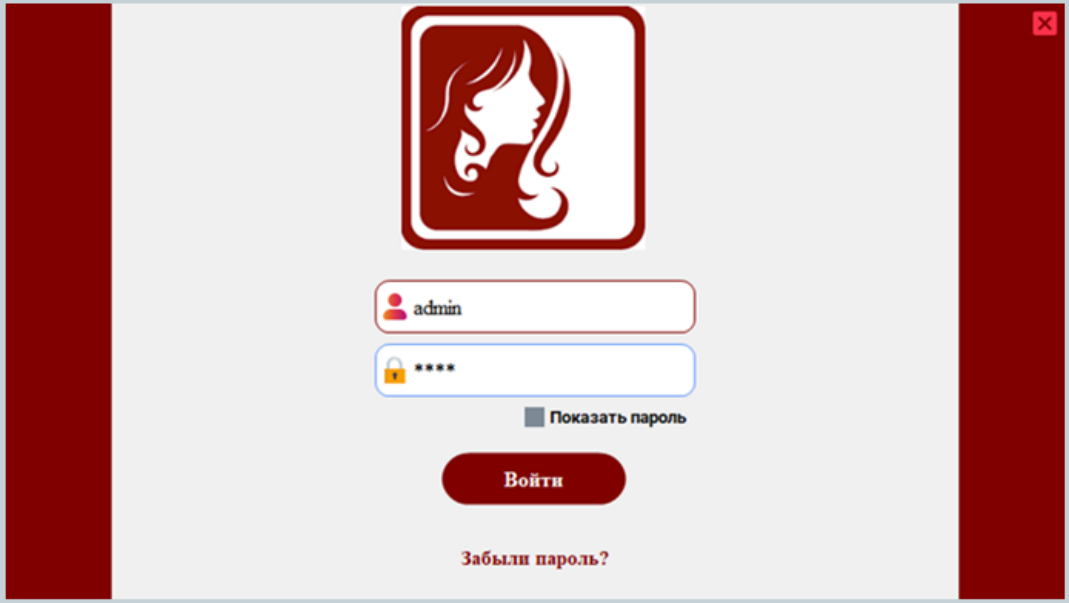
В результате внедрения информационной-автоматизированной системы планируется достичь ряда задач, а именно:

- ПОВЫСИТЬ ЭФФЕКТИВНОСТЬ РАБОТЫ МАСТЕРОВ;
- СНИЗИТЬ ЛИШНЮЮ НАГРУЗКУ НА МАСТЕРОВ;
- ПОВЫСИТЬ СКОРОСТЬ РЕГИСТРАЦИИ КЛИЕНТОВ.

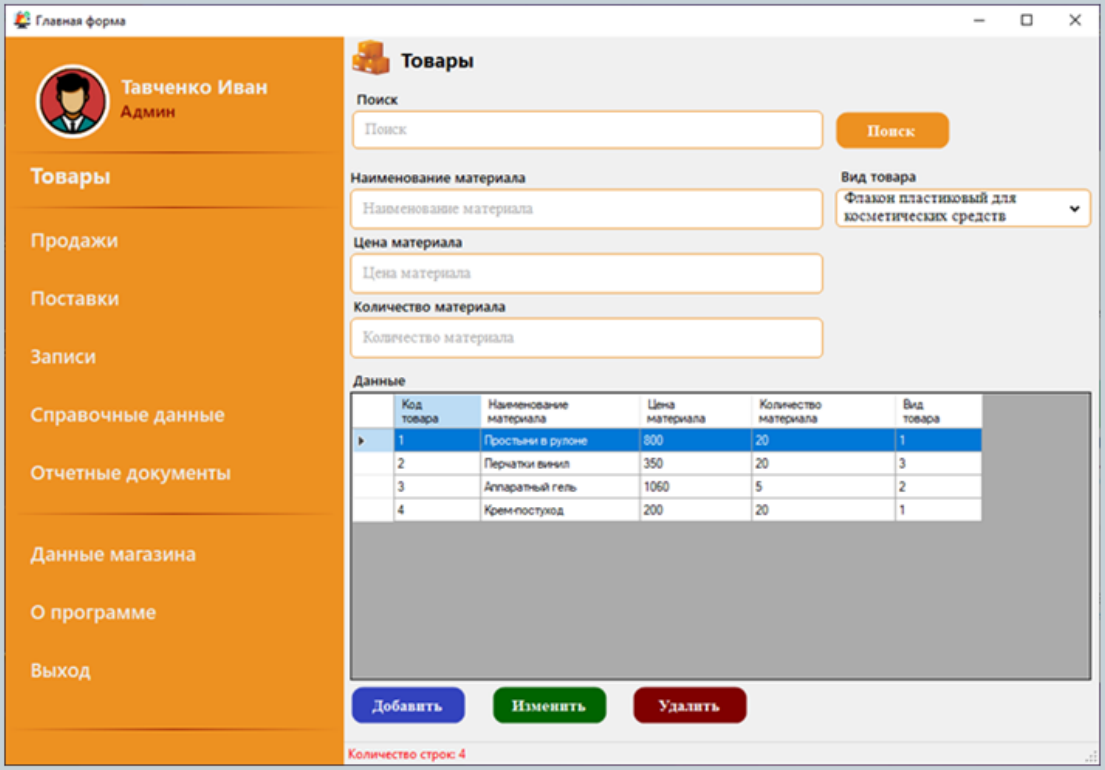
Схема базы данных



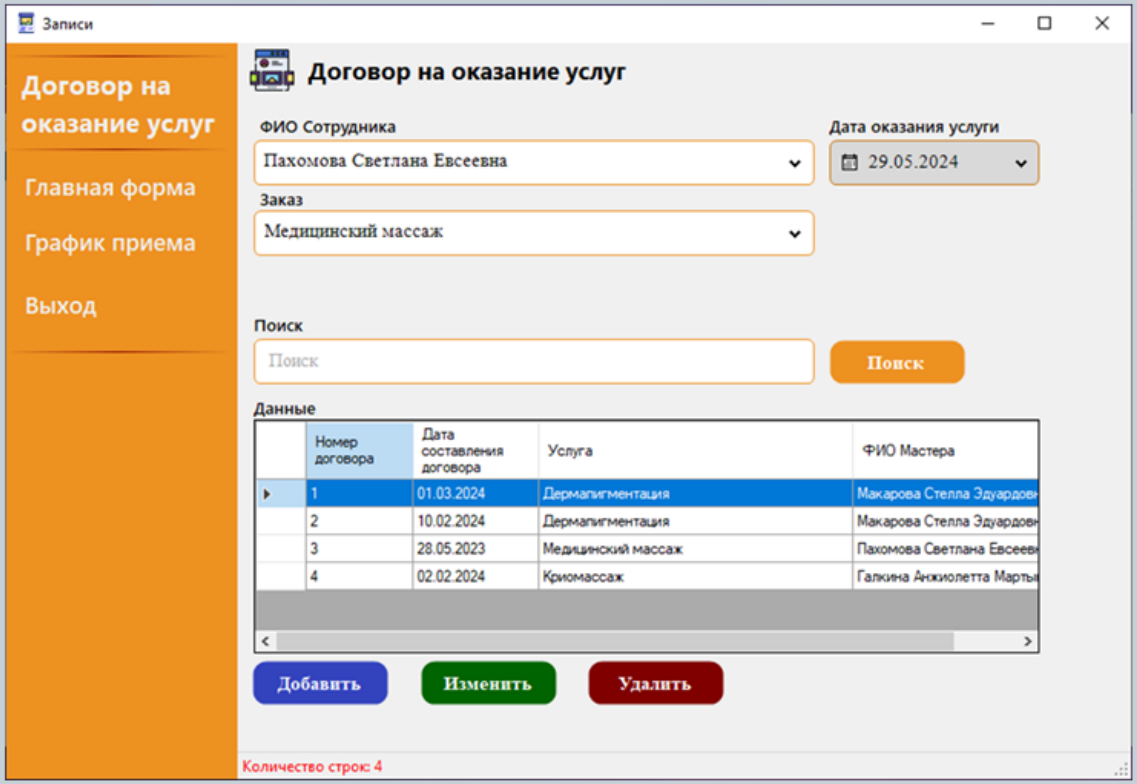
Процесс аутентификации пользователя



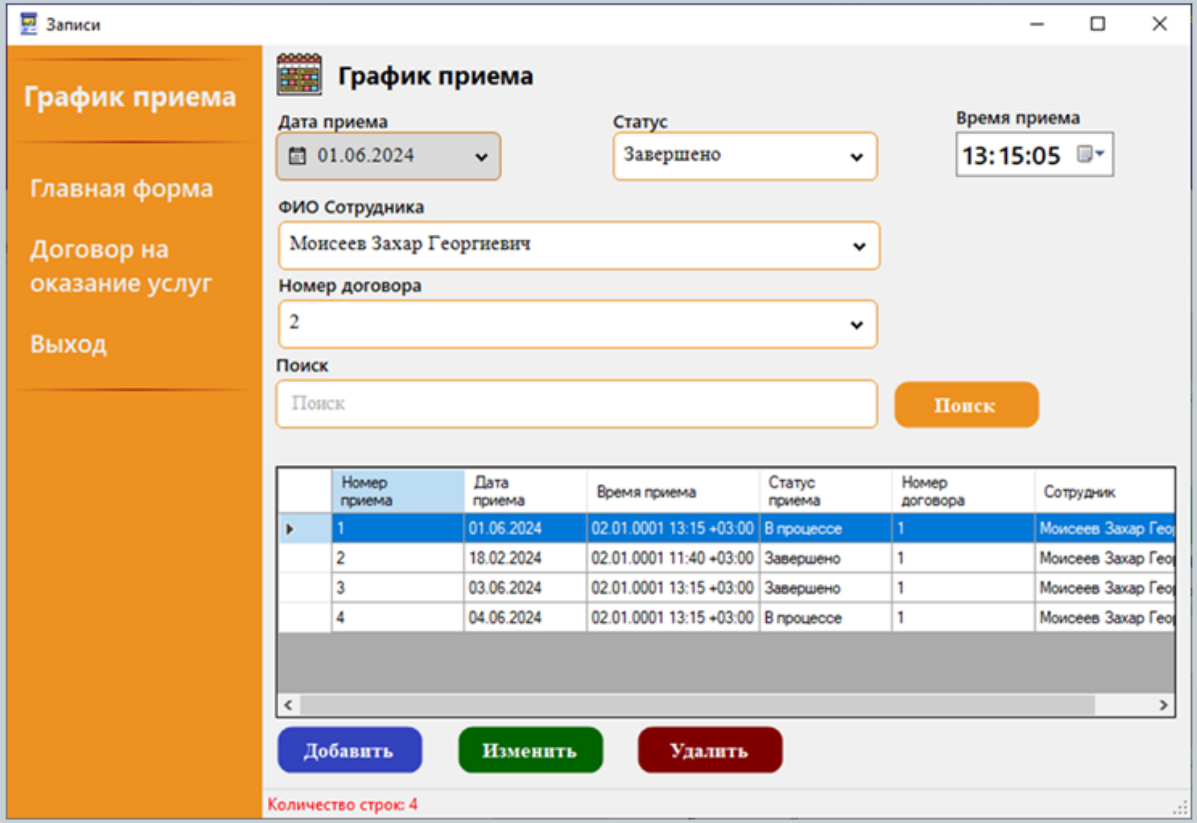
Главная форма системы



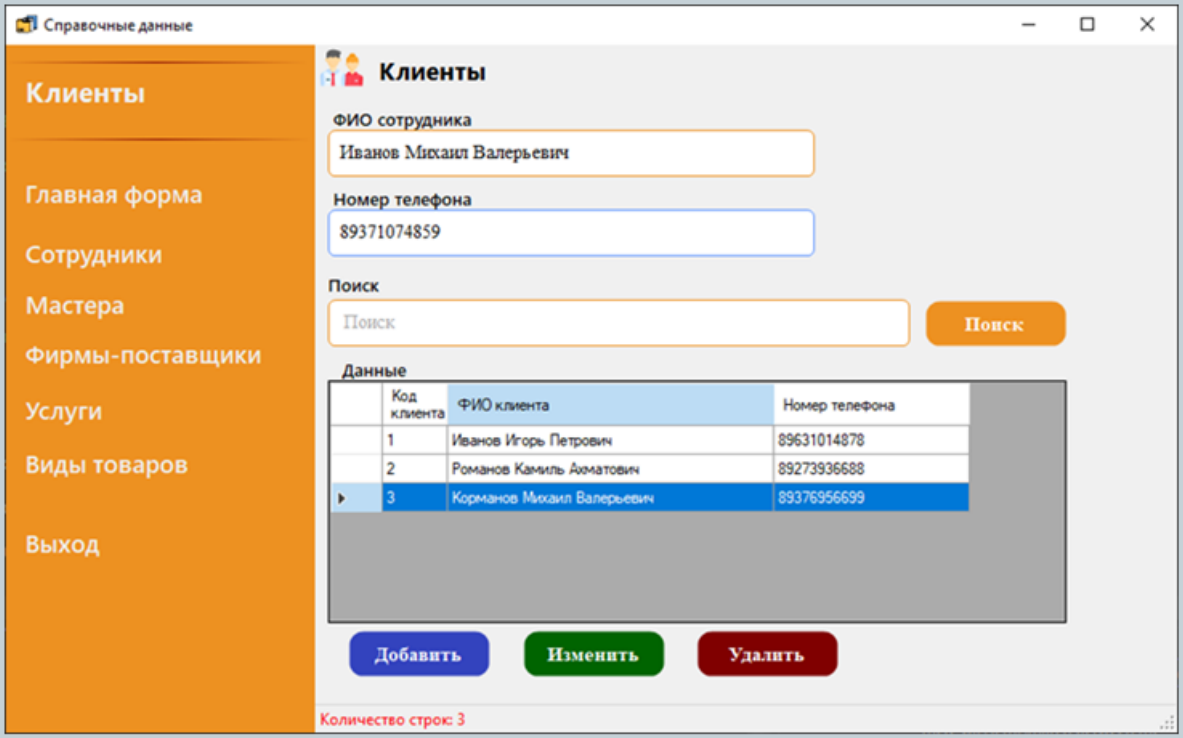
Форма «Записи»



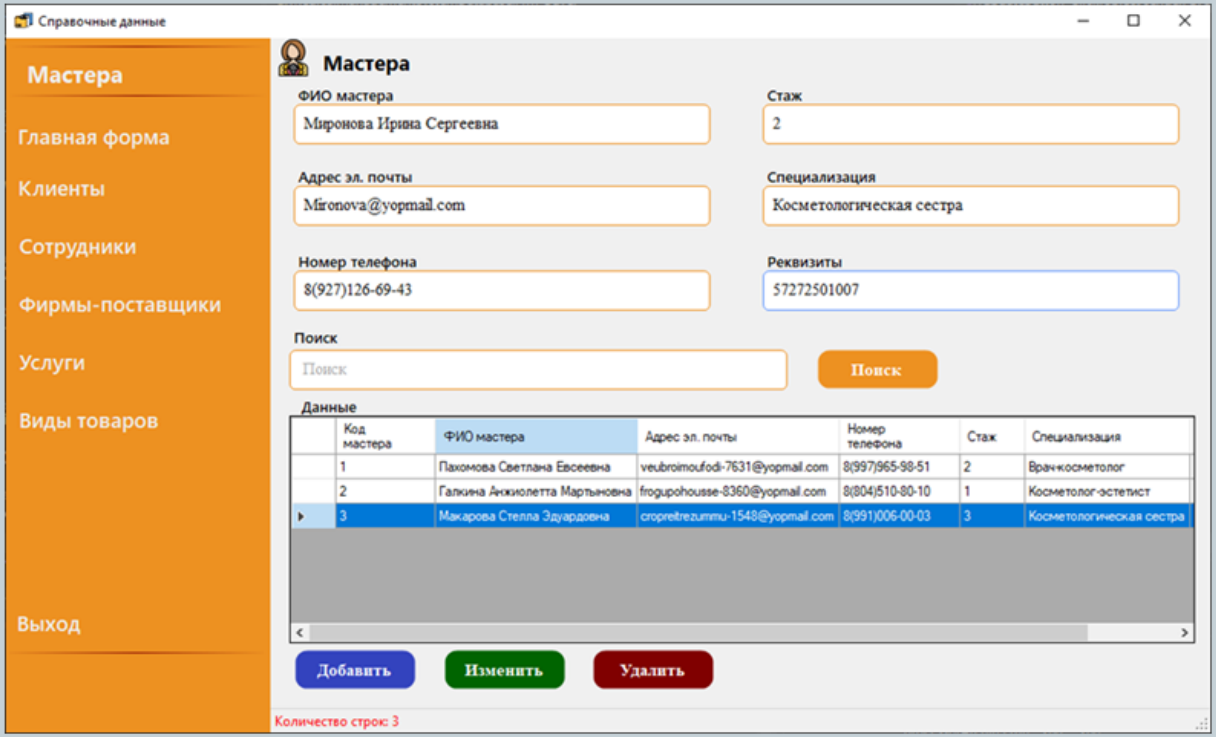
Меню «График приема»



Меню «Клиенты»



Меню «Мастера»



Заключение

- Разработанная система способна выполнять все обозначенные целевые функции и может быть использована в салонах красоты, с уклоном в области косметологии, и в частных медицинских организациях.