

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	2
1. СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА СУБД.....	4
1.1. Выбор СУБД	5
2. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА	7
2.1. Исследование предметной области	7
2.2. Постановка задачи	8
3. ПЛАНИРОВАНИЕ РАЗРАБОТКИ БД.....	9
3.1. Определение требований к системе.....	9
3.2. Сбор и анализ требований пользователей.....	9
3.3. Концептуальная модель базы данных	10
3.4. Преобразование концептуальной модели данных в реляционную	
модель данных	14
4. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ	20
4.1. Разграничение прав доступа.....	20
4.2. Реализация запросов в Microsoft SQL Server.....	30
5. РАЗРАБОТКА ПРИЛОЖЕНИЯ.....	34
5.1. Определение дизайна и разработка графического интерфейса .	34
5.2. Графический интерфейс приложения предприятия.....	35
5.3. Выполнение механизмов приложения	44
5.4. Реализация запросов.....	59
ЗАКЛЮЧЕНИЕ.....	64
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	65
ПРИЛОЖЕНИЕ А	66

ВВЕДЕНИЕ

В современном информационном обществе базы данных становятся неотъемлемой частью работы в различных областях, начиная от бизнеса и заканчивая научными исследованиями. С каждым днем объемы данных, требующих хранения, обработки и защиты, растут, что подчеркивает важность эффективного управления базами данных. В этом контексте междисциплинарный подход к разработке и защите баз данных приобретает особую значимость.

Целью данной курсовой работы является создание эффективной базы данных и системы управления для магазина электроники. Эта система будет оптимизирована для управления продуктами, заказами и клиентскими данными.

В ходе работы будет освоены основы работы с базами данных с использованием языка программирования C#. Основное внимание будет уделено использованию SQL-запросов.

Исследование взаимосвязи баз данных с языком программирования C# представляет огромный интерес для студентов, так как открывает новые горизонты в области информационных технологий. Эта работа не только поможет в получении глубокого понимания возможностей C# в работе с данными, но и станет надежным фундаментом для дальнейшего обучения. Полученные знания смогут быть успешно применены в будущих проектах и на практике.

1. СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА СУБД

Microsoft SQL Server – мощная коммерческая СУБД, предлагающая широкий спектр функциональности, высокую производительность и глубокую интеграцию с языком программирования C# и платформой .NET, что делает ее идеальным выбором для разработки корпоративных приложений.

MySQL – популярная открытая СУБД, обеспечивающая высокую скорость работы и масштабируемость при правильной настройке, однако интеграция с платформой .NET может быть менее глубокой по сравнению с коммерческими аналогами.

MariaDB – ответвление от MySQL с открытым исходным кодом, обеспечивающее совместимость с MySQL, но также добавляющее новые функции и улучшенную производительность, что делает ее хорошим вариантом для разработчиков, ориентированных на открытые технологии.

MongoDB – популярная это NoSQL база данных, ориентированная на работу с документами, что позволяет ей эффективно обрабатывать неструктурированные данные и адаптироваться к изменяющимся потребностям проекта.

PostgreSQL - это мощная и расширяемая объектно-реляционная СУБД с открытым исходным кодом, которая предоставляет богатый набор функциональности для эффективного управления данными. В основном используется для приложений, где требуется сложная обработка данных.

SQLite – легковесная встраиваемая СУБД, которая не требует отдельного сервера и подходит для небольших приложений и мобильных устройств, благодаря своей простоте и минимальным требованиям к ресурсам.

Oracle – коммерческая СУБД с широким функциональным спектром и высокой степенью надежности, нацеленная на крупные предприятия и критически важные системы, однако требует значительных затрат на лицензирование и обслуживание.

1.1. Выбор СУБД

В данном курсовом проекте было решено использовать СУБД Microsoft SQL Server по следующим причинам:

— Тесно интегрирована с платформой .NET и языком программирования C#, что обеспечивает простоту и эффективность взаимодействия между приложением на C# и базой данных. Такие возможности делают работу с базой данных интуитивно понятной и позволяют сосредоточиться на разработке логики приложения.

— Предоставляет разнообразные инструменты для управления базами данных, оптимизации запросов и обеспечения безопасности данных, что упрощает разработку, тестирование и поддержку приложений на C#. Благодаря интегрированным средствам администрирования баз данных пользователи могут эффективно управлять объектами баз данных, выполнять сложные запросы и анализировать их производительность. Инструменты для оптимизации запросов позволяют находить и устранять узкие места в производительности, обеспечивая более быстрое выполнение операций.

— Обладает высокой производительностью и масштабируемостью, что делает его подходящим выбором для разработки как небольших приложений, так и крупных корпоративных систем. Благодаря своей архитектуре SQL Server способен эффективно обрабатывать большие объемы данных и поддерживать множество одновременных подключений без снижения производительности. Оптимизация производительности также включает в себя продвинутые функции, такие как индексирование, кэширование данных, оптимизация запросов и параллельная обработка, что позволяет максимизировать скорость выполнения операций и минимизировать время отклика.

— Предлагает продвинутую систему безопасности, обеспечивающую многоуровневую защиту данных. Авторизация в SSMS включает использование ролей и разрешений, которые могут быть назначены на уровне

сервера, базы данных и отдельных объектов, таких как таблицы и схемы. Это гарантирует, что пользователи имеют доступ только к тем данным и операциям, которые необходимы для их работы, что повышает безопасность системы.

2. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

2.1. Исследование предметной области

Предметной областью является магазин электроники. Основным направлением деятельности является предоставление товаров, связанных с электроникой. В рамках этого магазина выделяются три основные роли: администратор, персонал и клиент. Администратор обладает полным доступом ко всем функциям и данным базы данных, что позволяет ему эффективно управлять всеми аспектами магазина, включая анализ финансовых данных. Также администратор осуществляет контроль над персоналом. Персонал имеет доступ к функциям управления товарами, таким как добавление, редактирование и удаление продуктов, обеспечивая актуальность и полноту ассортимента. Кроме этого, персонал обладает возможностью просматривать таблицу с активными и закрытыми заказами, а также выполнять запросы на добавление цены и скидок. Клиент, в свою очередь, может просматривать предложенные товары и заказывать их.

Данная база обеспечивает:

- хранение информации о продукции и заказах;
- удобное заполнение или редактирование товара;
- учёт и контроль складских запасов и поставок товаров;
- обеспечение безопасности данных и защиту от несанкционированного доступа;
- масштабируемость и высокую производительность при обработке данных;
- анализ финансовых данных с помощью запросов.

2.2. Постановка задачи

Для успешной реализации проекта магазина электроники необходимо создать информационную систему, которая обеспечит эффективное управление товарами и облегчит взаимодействие с клиентами. Основные цели разработки включают в себя:

Информационная система должна нести в себе следующие возможности:

- удовлетворение потребностей клиентов. Главной задачей информационной системы является обеспечение клиентов всей необходимой информацией о товарах;

- повышение эффективности управления ассортиментом. Система должна обеспечить персоналу удобные инструменты для добавления, удаления и редактирования товаров, обеспечивая актуальность ассортимента.

- возможность создания запросов. Это позволит администратору и персоналу магазина получать в удобном формате данные о продажах, остатках товаров и финансовых показателях.

Клиентская часть приложения должна иметь в себе:

- приложение должно быстро обеспечить доступ к информации о продуктах;

- интерфейс должен быть легким в освоении и приятным для использования, чтобы пользователи чувствовали себя комфортно во время работы с приложением. К этому также относиться интуитивно-понятная и простая навигация.

3. ПЛАНИРОВАНИЕ РАЗРАБОТКИ БД

Содержание данного этапа – разработка стратегического плана, в процессе которого осуществляется предварительное планирование конкретной системы управления базами данных. Общая информационная модель, созданная на этом шаге, должна быть вновь проанализирована и, если нужно, изменена на этапе разработки проекта реализации.

3.1. Определение требований к системе

Для обеспечения оптимальной работы приложения следует удостовериться, что компьютер соответствует следующим минимальным системным требованиям:

- операционная система: Windows 7 и выше;
- процессор: минимум 2 ядра;
- объём оперативной памяти: не менее 4096 Мб или выше;
- версия .NET Core: 8.0 или выше;
- установленный Microsoft SQL Server;
- наличие необходимых драйверов для корректной работы ПО;

Требуется наличие монитора для взаимодействия с приложением. А также необходимо наличие клавиатуры, мыши и сетевого порта для обеспечения работы и доступа к сети.

3.2. Сбор и анализ требований пользователей

Этот этап является предварительным шагом в концептуальном проектировании базы данных для магазина электроники. На данном этапе проводится анализ основных материальных объектов, таких как товары,

клиенты, заказы.

3.3. Концептуальная модель базы данных

Информационная система магазина электроники ведет учет данных о поставщиках, товарах, клиентах, заказах и деталях заказов. Она предназначена для управления всеми процессами, связанными с продажей электроники и управлением складскими запасами.

Объектом исследования является система учета продаж и управления товарными запасами в магазине электроники.

К предмету исследования можно отнести следующие процессы в университете электроники:

- процесс регистрации клиента;
- процесс входа клиента, персонала или администратора;
- процесс создания новой позиции товара;
- процесс редактирования существующего товара;
- процесс удаления позиции товара;
- процесс оценивания финансов;
- процесс заказа товара клиентом.

Субъекты системы - пользователи (администраторы, персонал и клиенты магазина). Администраторы имеют полный доступ ко всем функциям системы, включая управление пользователями и настройку параметров системы. Персонал могут добавлять и редактировать информацию о товарах, обрабатывать заказы и взаимодействовать с клиентами. Клиенты имеют доступ к каталогу товаров, могут оформлять заказы.

При внесении данных о заказах в базу данных фиксируются следующие параметры:

- Код заказа: уникальный идентификатор заказа.
- Идентификатор клиента: уникальный идентификатор клиента,

оформившего заказ.

- Дата оформления заказа: дата, когда заказ был оформлен.
- Общая сумма заказа: общая стоимость всех товаров в заказе.
- Статус выполнения заказа: текущий статус заказа (например,

активный или завершенный).

Характеристики каждой категории данных:

Поставщики:

- Код поставщика: уникальный идентификатор поставщика.
- Название: название компании-поставщика.
- Адрес: физический адрес поставщика.
- Телефон: контактный телефон поставщика для связи.

Товары:

- Код товара: уникальный идентификатор товара.
- Название: наименование товара.
- Бренд: торговая марка товара.
- Категория: категория, к которой относится товар (например,

смартфоны, ноутбуки).

- Цена: стоимость товара.
- Количество на складе: текущее количество товара на складе.
- Идентификатор поставщика: идентификатор поставщика, который

поставляет данный товар.

Клиенты:

- Код клиента: уникальный идентификатор клиента.
- Имя: имя клиента.
- Фамилия: фамилия клиента.
- Электронная почта: уникальный адрес электронной почты клиента

для связи и уведомлений.

- Телефон: контактный телефон клиента.
- Адрес: почтовый адрес клиента для доставки заказов.

- Логин: уникальное имя пользователя для входа в систему.
- Пароль: защищенный пароль для доступа к учетной записи.
- Статус: статус клиента в системе (администратор, сотрудник, клиент).

Заказы:

- Код заказа: уникальный идентификатор заказа.
- Идентификатор клиента: идентификатор клиента, который оформил заказ.
- Дата заказа: дата оформления заказа.
- Общая сумма: общая стоимость всех товаров в заказе.
- Статус: статус выполнения заказа (например, активный или завершённый).
- id_товара: уникальный идентификатор товара.

Связи между сущностями составлены следующим образом:

- Связь между Клиенты и Заказы:

Один ко многим. Клиент может сделать несколько заказов.

- Связь между Товары и Заказы:

Один ко многим. Один товар могут заказать несколько клиентов.

- Связь между Поставщики и Товары:

Один ко многим. Каждый поставщик может поставлять различные товары.

Выделенные сущности представлены на рисунке 1.

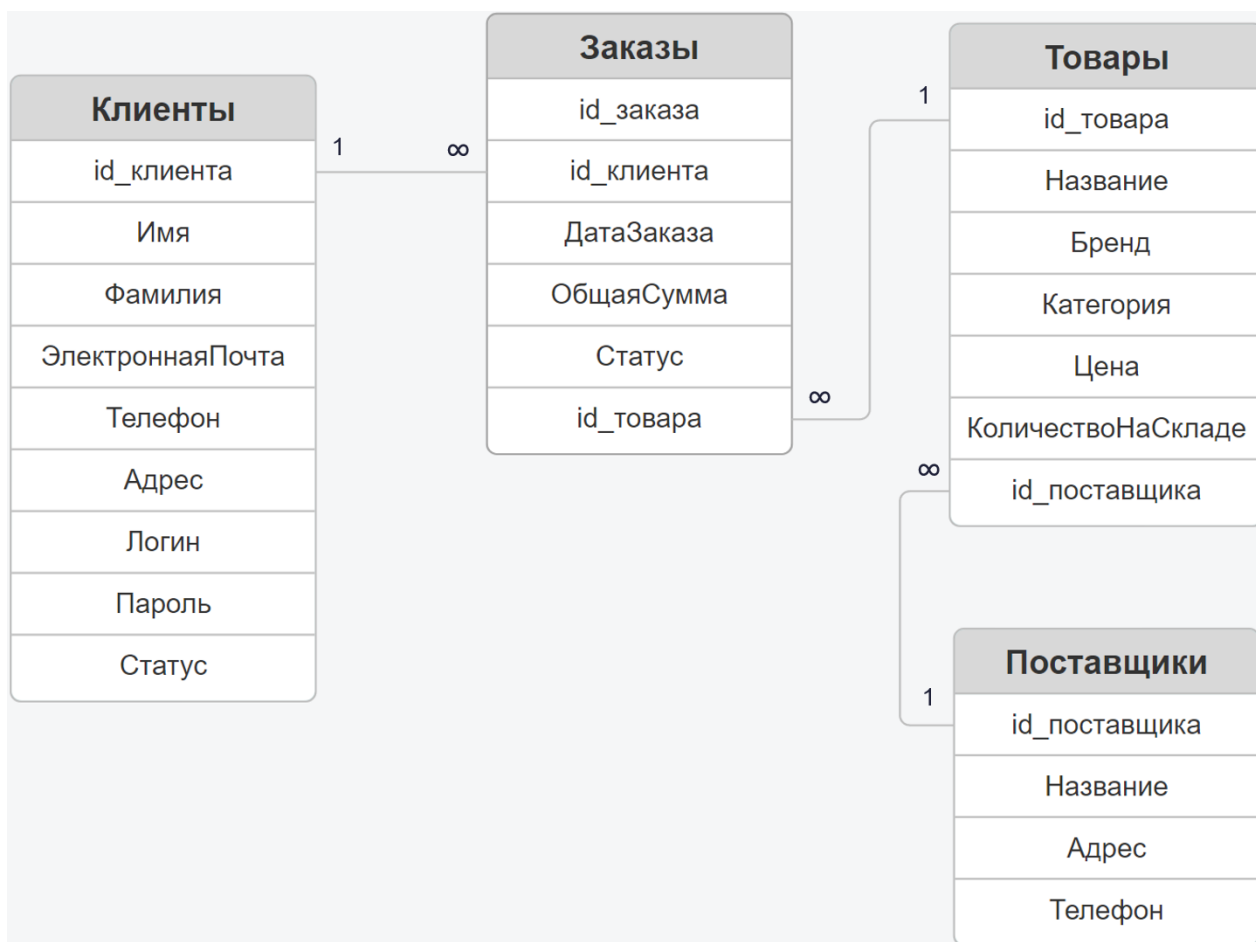


Рисунок 1 – ER-диаграмма базы данных разрабатываемой системы

ER-модель обеспечивает наглядное и интуитивно понятное представление о структуре данных в информационной системе. Она позволяет охватить все сущности и связи между ними, обеспечивая полноту представления о данных. Благодаря четкому определению сущностей, их атрибутов и взаимосвязей, ER-модель упрощает управление данными и помогает оптимизировать структуру базы данных. Использование ER-модели упрощает процесс проектирования базы данных, помогая выявить потенциальные ошибки и проблемы еще на начальных этапах разработки. Это сокращает время и ресурсы, затрачиваемые на исправление ошибок впоследствии.

3.4. Преобразование концептуальной модели данных в реляционную модель данных

Следующим шагом будет создание логической модели хранения данных на основе концептуального проектирования.

Существуют разные типы моделей баз данных, каждая из которых подходит для определенных задач. Классическими примерами таких моделей являются иерархическая, сетевая, реляционная и объектно-ориентированная.

На данный момент наиболее распространённой моделью данных является реляционная. Преимуществами этой модели являются ясная и простая логическая структура, крепкая теоретическая база, поддержка целостности данных, а также наличие широкого выбора качественных языковых и программных средств.

В реляционной модели данных важным элементом является теория нормальных форм. Нормальные формы представляют собой определённые состояния, которым должны соответствовать отношения в базе данных. Процесс приведения реляционной базы данных в соответствие с этими нормальными формами называется нормализацией. Её целью является устранение избыточности данных, формирование ясной и логической структуры хранения информации, а также обеспечение целостности данных. В рамках данного курсового проекта принято решение довести базу данных до третьей нормальной формы, так как это наиболее высокий уровень нормализации, который обычно требуется для большинства приложений.

В базе данных для магазина электроники применены принципы нормализации.

- Первая нормальная форма: все таблицы имеют простые, неделимые атрибуты. Например, таблица "Поставщики" содержит такие атрибуты, как id_поставщика, Название, Адрес и Телефон, каждый из которых содержит только одно значение.
- Вторая нормальная форма: все неключевые атрибуты полностью зависят

от первичного ключа. В таблице "Товары" атрибуты Название, Бренд, Категория, Цена и КоличествоНаСкладе зависят от первичного ключа id_товара. Точно так же атрибут id_поставщика является внешним ключом, указывающим на таблицу "Поставщики".

- Третья нормальная форма: в таблице "Клиенты" атрибуты Имя, Фамилия, ЭлектроннаяПочта, Телефон, Адрес, Логин, Пароль и Статус зависят только от первичного ключа id_клиента и не зависят друг от друга, что исключает транзитивные зависимости.

Создадим все необходимые ключевые атрибуты, определим отношения между ними. После завершения этапа преобразования концептуальной модели в логическую, получим следующую структуру логических сущностей.

Таблица 1 – Объект «Заказы» и формат его атрибутов

Название атрибута	Тип данных	Принимает null	Ключ
id_заказа	int	Нет	Главный
id_клиента	int	Нет	Внешний
ДатаЗаказа	date	Нет	
ОбщаяСумма	decimal(10, 2)	Нет	
Статус	nvarchar(10)	Да	
Id_товара	int	Да	

Таблица 2 – Объект «Клиенты» и формат его атрибутов

Название атрибута	Тип данных	Принимает null	Ключ
id_клиента	int	Нет	Главный
Имя	nvarchar(100)	Да	
Фамилия	nvarchar(100)	Да	
ЭлектроннаяПочта	nvarchar(255)	Нет	
Телефон	nvarchar(20)	Да	
Адрес	nvarchar(225)	Да	
Логин	nvarchar(50)	Нет	
Пароль	nvarchar(100)	Нет	
Статус	nvarchar(10)	Да	

Таблица 3 – Объект «Поставщики» и формат его атрибутов

Название атрибута	Тип данных	Принимает null	Ключ
id_поставщика	int	Нет	Главный
Название	nvarchar(255)	Нет	

Адрес	nvarchar(255)	Да	
Телефон	nvarchar(20)	Да	

Таблица 4 – Объект «Товары» и формат его атрибутов

Название атрибута	Тип данных	Принимает null	Ключ
id_товара	int	Нет	Главный
Название	nvarchar(255)	Нет	
Бренд	nvarchar(100)	Да	
Категория	nvarchar(100)	Да	
Цена	decimal(10, 2)	Нет	
КоличествоНа Складе	int	Нет	
id_поставщика	int	Да	Внешний

Заполнение созданных таблиц данными (Рисунки 2-5).

	id_заказа	id_клиента	ДатаЗаказа	ОбщаяСумма	Статус	id_товара
▶*	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 2 – Данные таблицы Заказы

	id_клиента	Имя	Фамилия	Электронная...	Телефон	Адрес	Логин	Пароль	Статус
▶	1	Александр	Лебедев	AlexanderLebe...	89117231129	Ул. Белая, 43	adm	adm	client
	2	Михаил	Габкий	gibok@mail.ru	89002998030	Ул. Карпинско...	miha	911miha	client
	3	Сергей	Бричкин	sergeyBrich@y...	89112312211	Ул. Школьная	serg	serg123	client
	4	Иван	Ксенонов	xenon72@gmai...	89110223327	Просп. Энгель...	xenonivan	4894024	client
	5	Сергей	Петров	airPush873@m...	88113842292	Просп. Науки, ...	petrovS	984484petrov	client
	6	Артём	Сорокин	sorokin1976@g...	88003849863	Ул. Хлопина, 9	arSorokin	7877	client
	7	Анна	Васильева	annaVasilka@y...	89118334219	Ул. Политехни...	annaVasilka92	933842V	client
	8	Никита	Широков	dhindi@gmail....	89223742312	Ул. Грусти, 18	shirokovnikita	842494Yt8389	client
	9	Сергей	Поляков	polyakov2003@...	89118173633	Просп. Науки, ...	waterBottle	38394822442	client
	10	Кира	Орлова	87KiraOrlova@...	88337331296	Ул. Шишкина, 4	KirkaOrlova	orlova8492	client
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 3 – Данные таблицы Клиенты

	id_поставщика	Название	Адрес	Телефон
▶	1	Белокосмос	Ул. Разбитых ...	89002931221
	2	EuroStars	Просп. Энгель...	89117663411
	3	ZipLock	Просп. Науки, ...	89003942219
•	NULL	NULL	NULL	NULL

Рисунок 4– Данные таблицы Поставщики

	id_товара	Название	Бренд	Категория	Цена	КоличествоН...	id_поставщика
▶	1	USB зарядка дл...	Samsung	Провода	146,35	91	1
	2	Телефон A71	Samsung	Телефоны	15300,50	7	1
	3	iPhone 15	Apple	Телефоны	149999,00	4	2
	4	iPhone XR	Apple	Телефоны	40000,00	9	2
	5	GeoMetrix A7	LG	Телефоны	7999,00	0	2
	6	Компьютерная...	Logitech	Компьютерная...	3999,00	15	1
	7	Монитор 24" д...	LG	Мониторы	9999,00	4	1
	8	Монитор 28" д...	LG	Мониторы	23999,00	2	1
	9	Телевизор 24" ...	Samsung	Телевизоры	78999,00	1	1
	10	Игровой комп...	DNS	Компьютеры	84999,00	5	1
	11	USB переходник	Ardire	Провода	99,99	45	1
	12	Блендер U76	LHZ	Блендеры	2999,00	16	1
	13	Коврик для мы...	Razer	Компьютерная...	2500,00	19	1
	14	iPhone 13	Apple	Телефоны	78999,00	3	2
	15	Клавиатура Z71	HyperX	Компьютерная...	7999,00	35	1
	16	HDMI переход...	LHZ	Провода	199,85	29	1
	17	Наушники HH...	HyperX	Компьютерная...	6499,00	11	1
	18	Моноблок	Acer	Компьютеры	44999,00	5	1
	19	Веб-камера 4K...	GoodQuality	Компьютерная...	11999,00	3	1
	20	Микрофон Sat...	MSI	Компьютерная...	11999,00	17	1
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 5 – Данные таблицы Товары

После создания таблиц и последующего заполнения их данными, необходимо создать связи между таблицами в диаграмме (Рисунок 6).

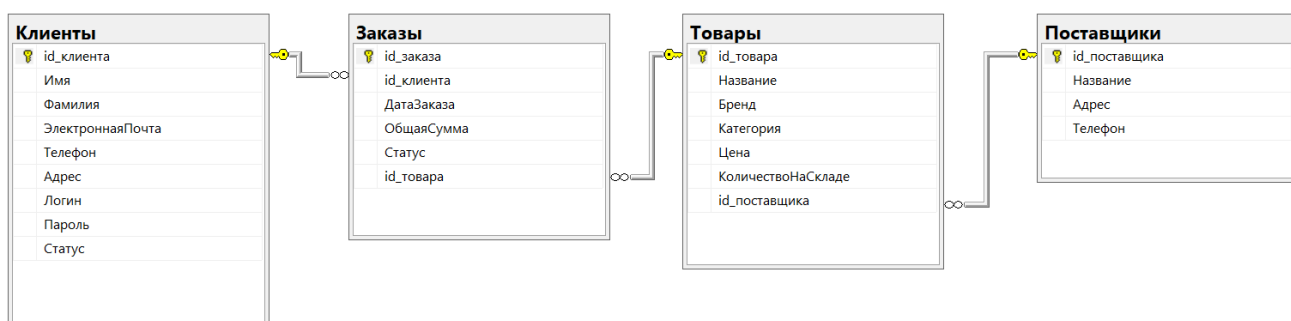


Рисунок 6 – Реляционная модель БД разрабатываемой системы

4. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ

4.1. Разграничение прав доступа

Система авторизации пользователей с различными уровнями доступа позволяет эффективно управлять доступом к данным и ресурсам, обеспечивая безопасность и конфиденциальность информации. Эта система предотвращает как умышленные, так и случайные ошибки, тем самым защищая целостность данных.

Microsoft SQL Server использует хеширование паролей для защиты учетных данных. При этом применяется алгоритм SHA-512, который преобразует пароль в фиксированную строку символов. Этот процесс гарантирует, что пароль пользователя хранится в зашифрованном виде в базе данных. Хеширование паролей обеспечивает безопасность даже в случае получения злоумышленником доступа к файлам базы данных или их копиям. В результате, даже при возможном нарушении безопасности данных, реальный пароль остается недоступным для несанкционированного использования.

Для реализации системы авторизации пользователей нужно создать соответствующие роли: Администратор, Персонал и Клиент. Администраторы будут обладать полными правами на управление базой данных, включая создание, изменение и удаление таблиц и данных. Персонал сможет добавлять и изменять записи в таблицах товаров и заказов, а также использовать фильтрацию данных для выполнения своих задач. Клиенты, в свою очередь, будут иметь возможность только просматривать информацию, без возможности внесения изменений. Для подключения к SQL Server, пользователь должен ввести логин и пароль при подключении. Этот процесс аутентификации позволяет серверу проверить учетные данные пользователя и разрешить или отклонить доступ к базе данных в зависимости от предоставленных данных.

Для того, чтобы создать новое имя для входа, необходимо раскрыть папку Безопасность, выбрать папку Имена для входа и нажать на соответствующую кнопку (Рисунок 7).

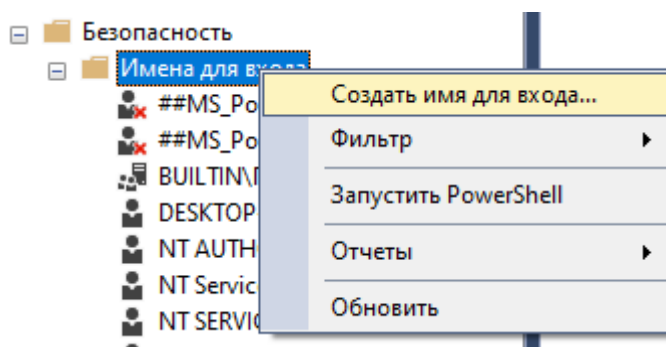


Рисунок 7 – Создание имени для входа

После нажатия на кнопку появляется окно, в котором задаём имя администратора и пароль для этой роли (Рисунок 8).

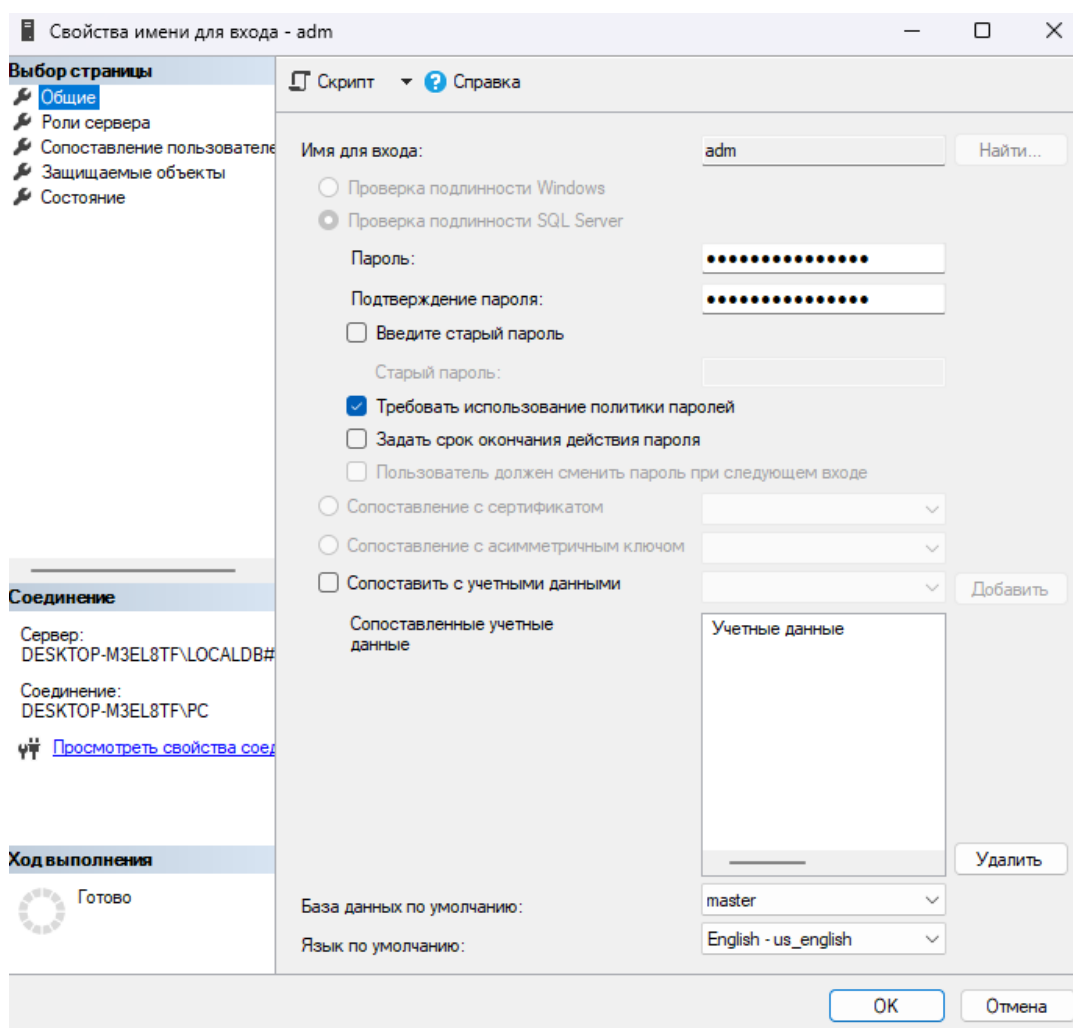


Рисунок 8 – Создание Администратора

После установки пароля нужно указать роль сервера для данного пользователя. Администратору подойдёт наивысшая роль sysadmin – так он сможет выполнять любые действия на сервере (Рисунок 9).

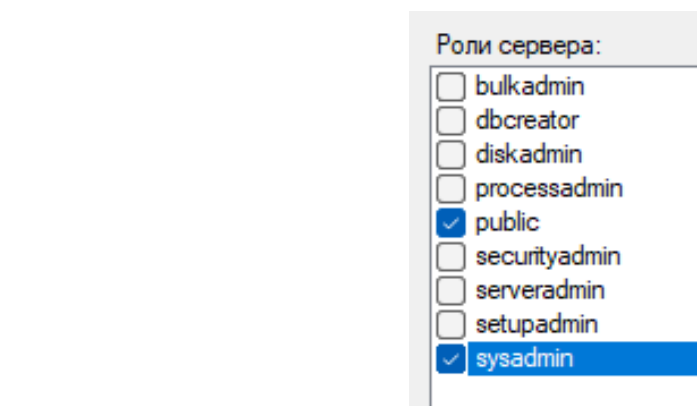


Рисунок 9 – Присвоение роли Администратору для управления всей базой данных

Подобным образом создаётся роль Персонал (Рисунок 10).

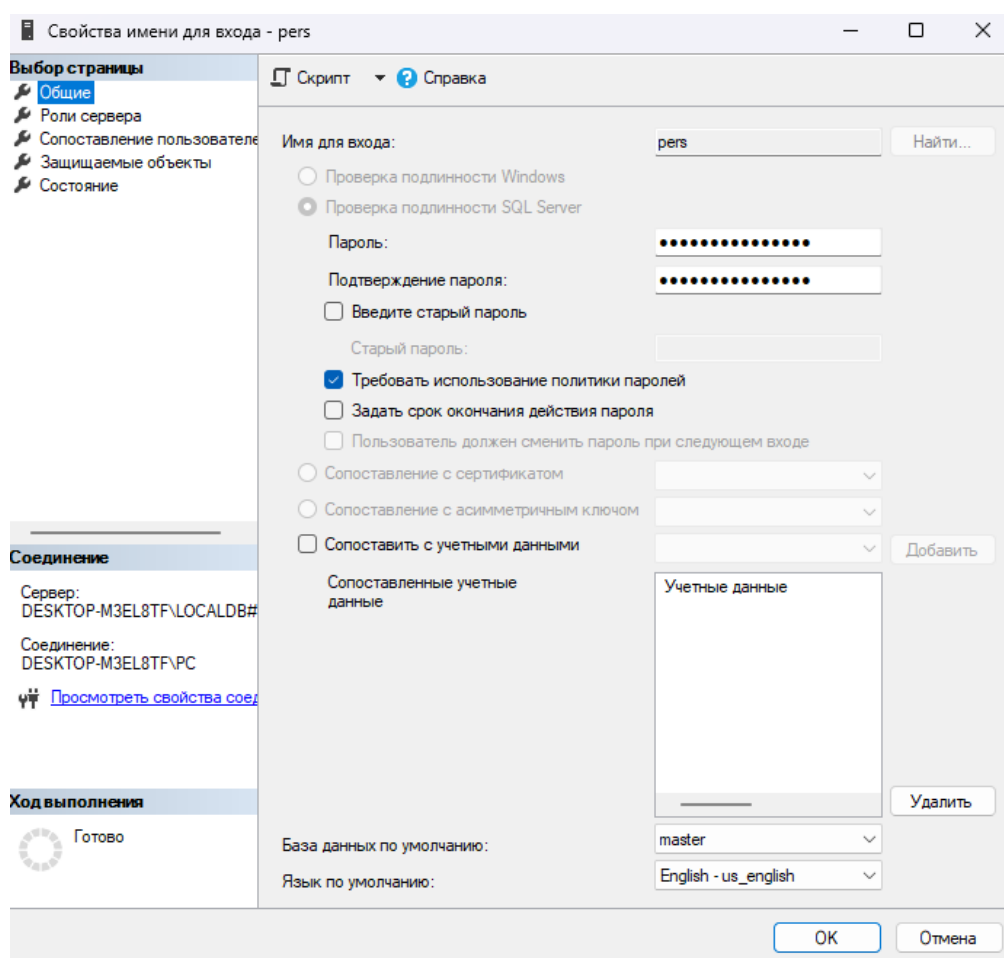


Рисунок 10 – Создание Персонала

Использование операции GRANT для добавления прав пользователю Персонал (Рисунок 11). Таким образом, Персонал получает полный доступ к

таблице Заказы и Товары, включая возможности чтения, добавления, удаления и изменения данных.

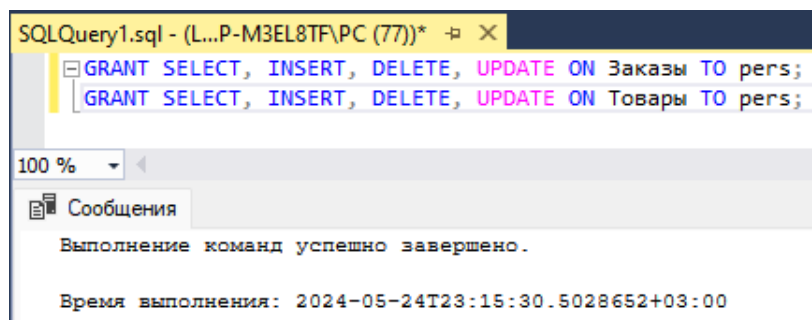


Рисунок 11 – Добавление прав для Персонала

Аналогичным образом создаётся Клиент (Рисунок 12).

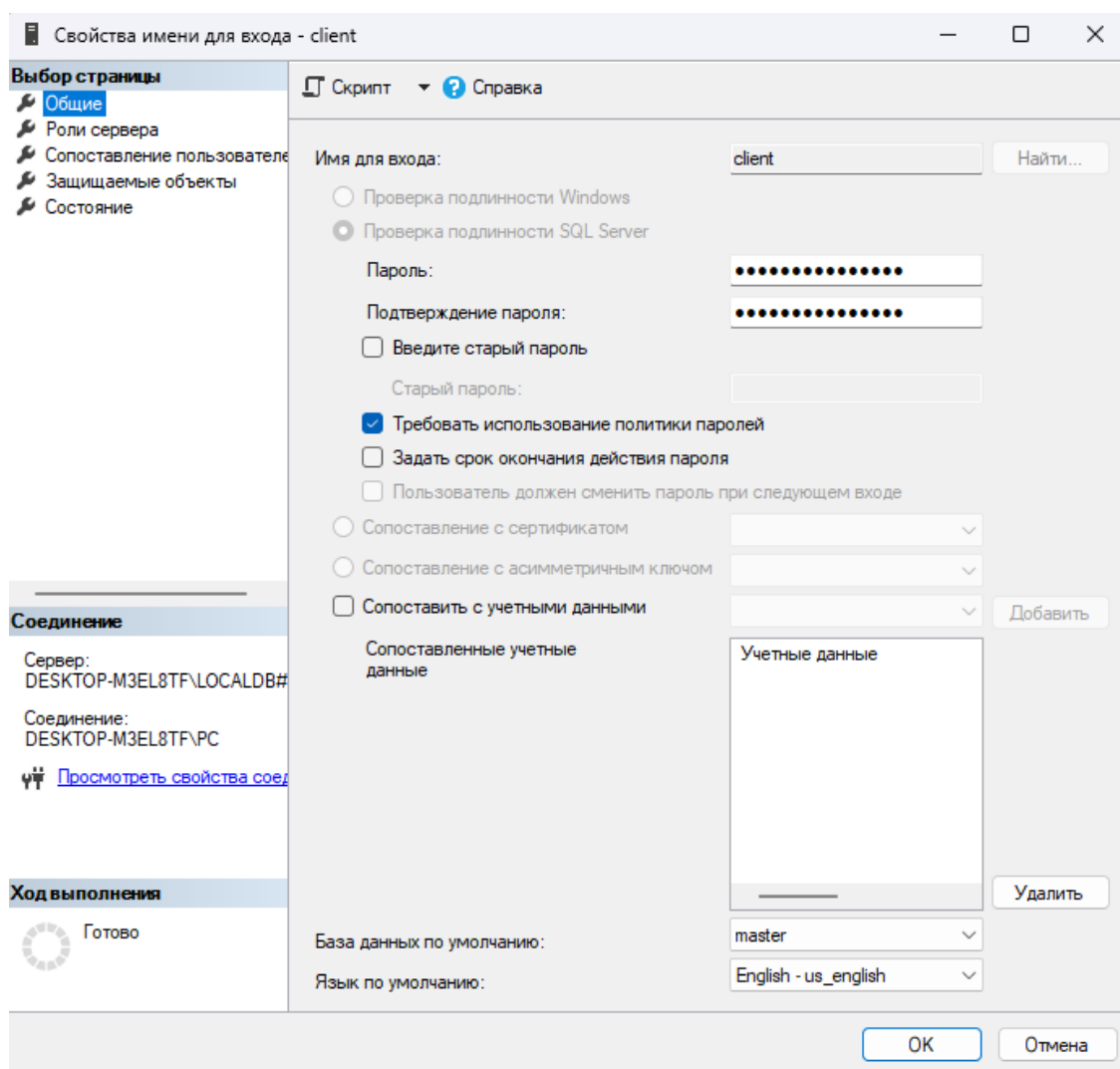


Рисунок 12 – Создание Клиента

Добавление прав Клиенту с помощью операции GRANT, позволяющие просматривать таблицу Товары (Рисунок 13).

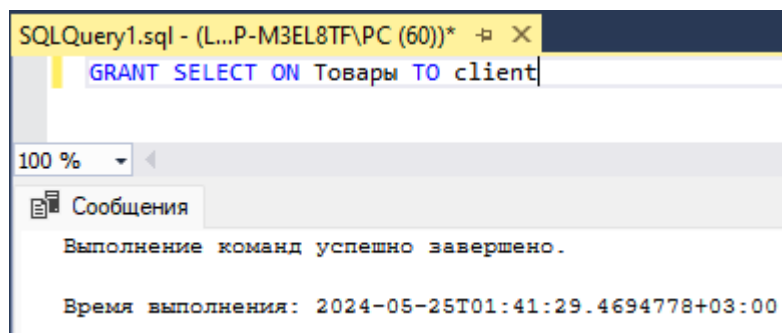


Рисунок 13 – Добавление прав для Клиента

После настройки всех ролей в Microsoft SQL Server можно увидеть созданных пользователей и проверить их права (Рисунок 14).

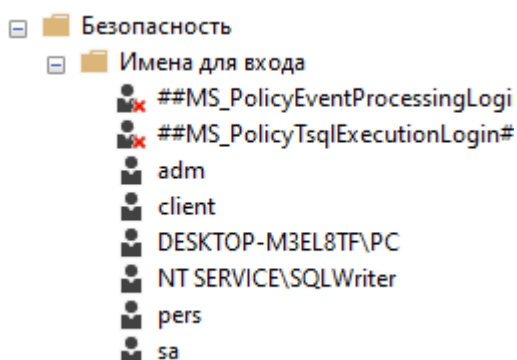


Рисунок 14 – Список пользователей

Проверка прав пользователя Администратор. Для проверки работы серверных ролей требуется выбрать проверку подлинности SQL Server и ввести имя для входа с паролем (Рисунок 15).

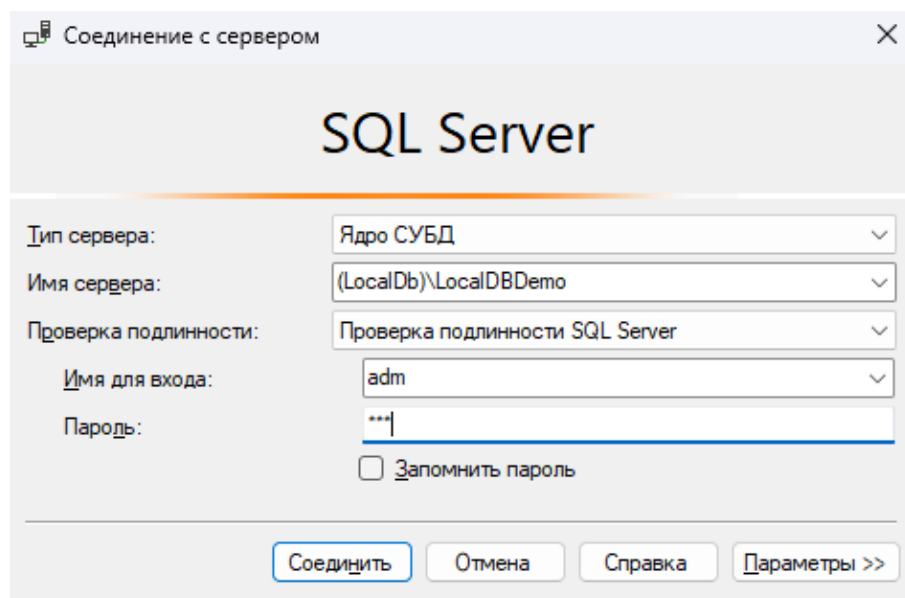


Рисунок 15 – Подключение к БД как Администратор

Администратор видит все таблицы и может полностью управлять ими (Рисунок 16).

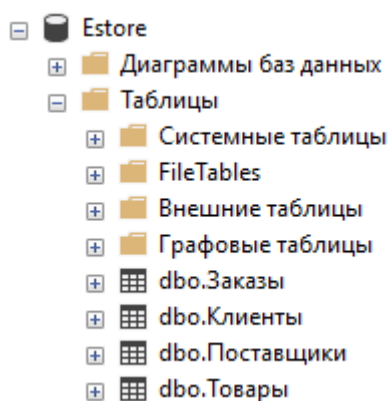


Рисунок 16 – Вид от лица Администратора

Проверка прав пользователя Персонал (Рисунки 17-20).

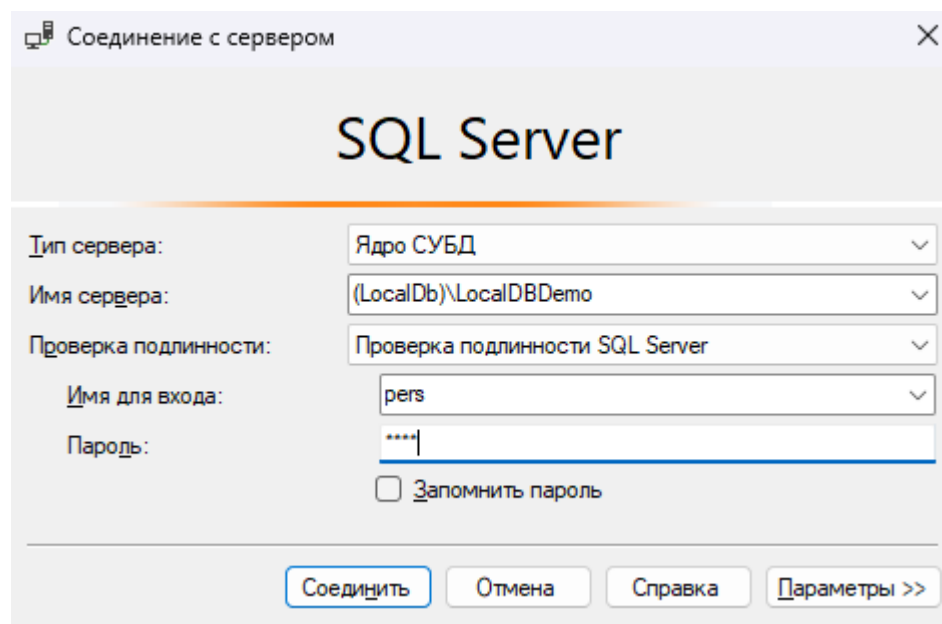


Рисунок 17 – Подключение к БД как Персонал

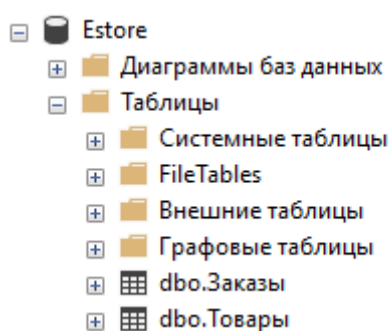


Рисунок 18 – Вид от лица Персонала

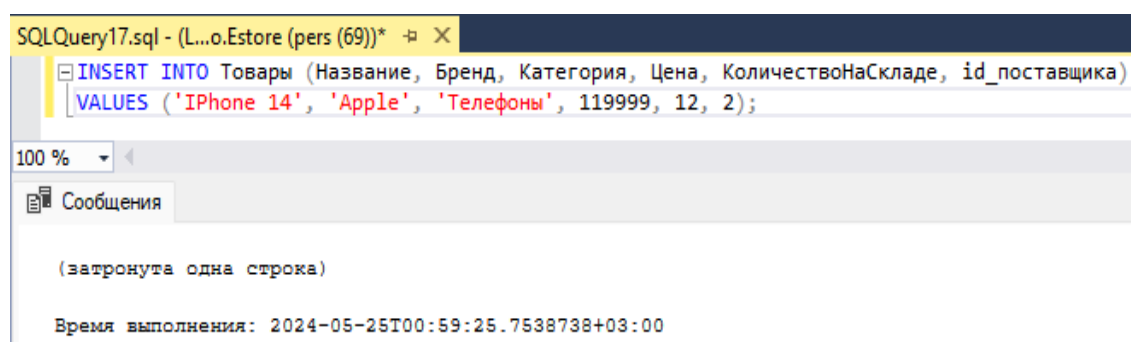


Рисунок 19 – Запрос на добавление нового товара

	id_товара	Название	Бренд	Категория	Цена	КоличествоH...	id_поставщика
▶	1	USB зарядка для телефона	Samsung	Провода	146,35	91	1
	2	Телефон A71	Samsung	Телефоны	15300,50	7	1
	3	iPhone 15	Apple	Телефоны	149999,00	4	2
	4	iPhone XR	Apple	Телефоны	40000,00	9	2
	5	GeoMetrix A7	LG	Телефоны	7999,00	0	2
	6	Компьютерная мышь	Logitech	Компьютерная периферия	3999,00	15	1
	7	Монитор 24" дюйма	LG	Мониторы	9999,00	4	1
	8	Монитор 28" дюйма	LG	Мониторы	23999,00	2	1
	9	Телевизор 24" дюйма	Samsung	Телевизоры	78999,00	1	1
	10	Игровой компьютер	DNS	Компьютеры	84999,00	5	1
	11	USB переходник	Ardire	Провода	99,99	45	1
	12	Блендер U76	LHZ	Блендеры	2999,00	16	1
	13	Коврик для мыши	Razer	Компьютерная периферия	2500,00	19	1
	14	iPhone 13	Apple	Телефоны	78999,00	3	2
	15	Клавиатура Z71	HyperX	Компьютерная периферия	7999,00	35	1
	16	HDMI переходник	LHZ	Провода	199,85	29	1
	17	Наушники HHZ Super	HyperX	Компьютерная периферия	6499,00	11	1
	18	Моноблок	Acer	Компьютеры	44999,00	5	1
	19	Веб-камера 4K Ultra	GoodQuality	Компьютерная периферия	11999,00	3	1
	20	Микрофон Saturn11	MSI	Компьютерная периферия	11999,00	17	1
	21	iPhone 14	Apple	Телефоны	119999,00	12	2
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 20 – Успешно добавленный товар с помощью запроса

Проверка прав пользователя Клиента (Рисунки 21-24).

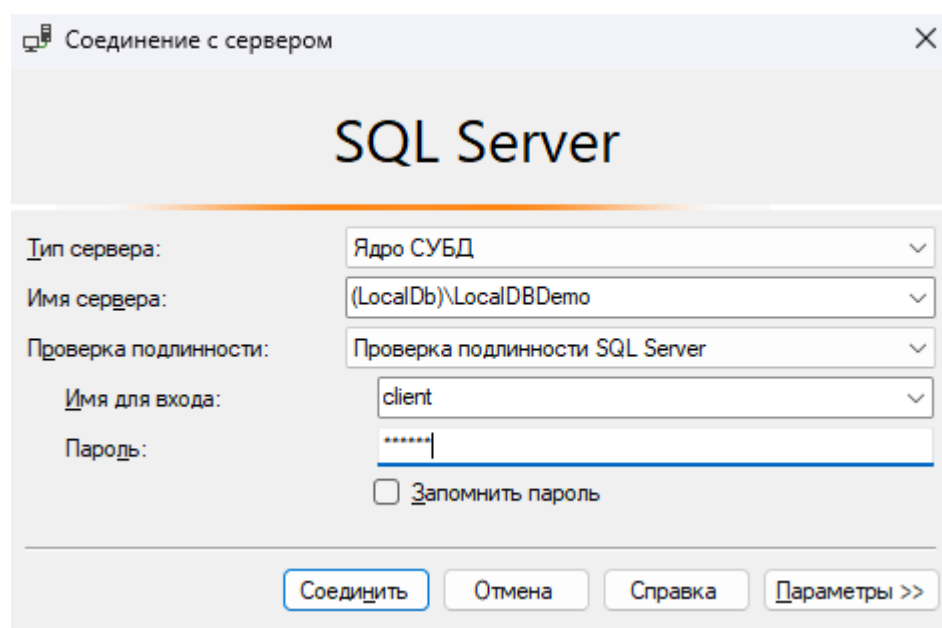


Рисунок 21 – Подключение к БД как Клиент

Для последующей авторизации в клиентском приложении необходимо обновить таблицу клиентов, добавив в статусе adm или pers (Рисунок 25).

	id_клиента	Имя	Фамилия	Электронная...	Телефон	Адрес	Логин	Пароль	Статус
►	1	Александр	Лебедев	AlexanderLebe...	89117231129	Ул. Белая, 43	adm	adm	adm
	2	Михаил	Габкий	gibok@mail.ru	89002998030	Ул. Карпинско...	miha	911miha	client
	3	Сергей	Бричкин	sergeyBrich@y...	89112312211	Ул. Школьная	serg	serg123	pers
	4	Иван	Ксенонов	xenon72@gmai...	89110223327	Просп. Энгель...	xenonivan	4894024	client
	5	Сергей	Петров	airPush873@m...	88113842292	Просп. Науки, ...	petrovS	984484petrov	client
	6	Артём	Сорокин	sorokin1976@g...	88003849863	Ул. Хлопина, 9	arSorokin	7877	client
	7	Анна	Васильева	annaVasilka@y...	89118334219	Ул. Политехни...	annaVasilka92	933842V	pers
	8	Никита	Широков	dhindi@gmail....	89223742312	Ул. Грусти, 18	shirokovnikita	842494Yt8389	client
	9	Сергей	Поляков	polyakov2003@...	89118173633	Просп. Науки, ...	waterBottle	38394822442	client
	10	Кира	Орлова	87KiraOrlova@...	88337331296	Ул. Шишкина, 4	KirkaOrlova	orlova8492	client
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Рисунок 25 – Обновление таблицы Клиенты

4.2. Реализация запросов в Microsoft SQL Server

Запросы обеспечивают функциональность, необходимую для эффективного взаимодействия с данными, хранящимися в базе.

Создание транзакции, которая добавляет новую запись в таблицу «Поставщики» (Рисунок 26).

SQLQuery1.sql - (L...-M3EL8TF\PC (105))*				
<pre> BEGIN TRANSACTION SELECT * FROM Поставщики; INSERT INTO Поставщики VALUES('SpbLight', 'Просп. Петра, 21', 89003831103); SELECT * FROM Поставщики; COMMIT TRANSACTION </pre>				
100 %				
<div> <div>Результаты</div> <div>Сообщения</div> </div>				
	id_поставщика	Название	Адрес	Телефон
1	1	Белокосмос	Ул. Разбитых фонарей, 31	89002931221
2	2	EuroStars	Просп. Энгельса, 7	89117663411
3	3	ZipLock	Просп. Науки, 28	89003942219
4	4	Piter52	Ул. Петербургская, 52	89528125252
	id_поставщика	Название	Адрес	Телефон
1	1	Белокосмос	Ул. Разбитых фонарей, 31	89002931221
2	2	EuroStars	Просп. Энгельса, 7	89117663411
3	3	ZipLock	Просп. Науки, 28	89003942219
4	4	Piter52	Ул. Петербургская, 52	89528125252
5	5	SpbLight	Просп. Петра, 21	89003831103

Рисунок 26 – Результат работы транзакции

Создание представления, которое объединяет информацию о товарах и их поставщиках, показывая данные о товарах вместе с именем, адресом и телефоном их поставщиков (Рисунки 27-28).

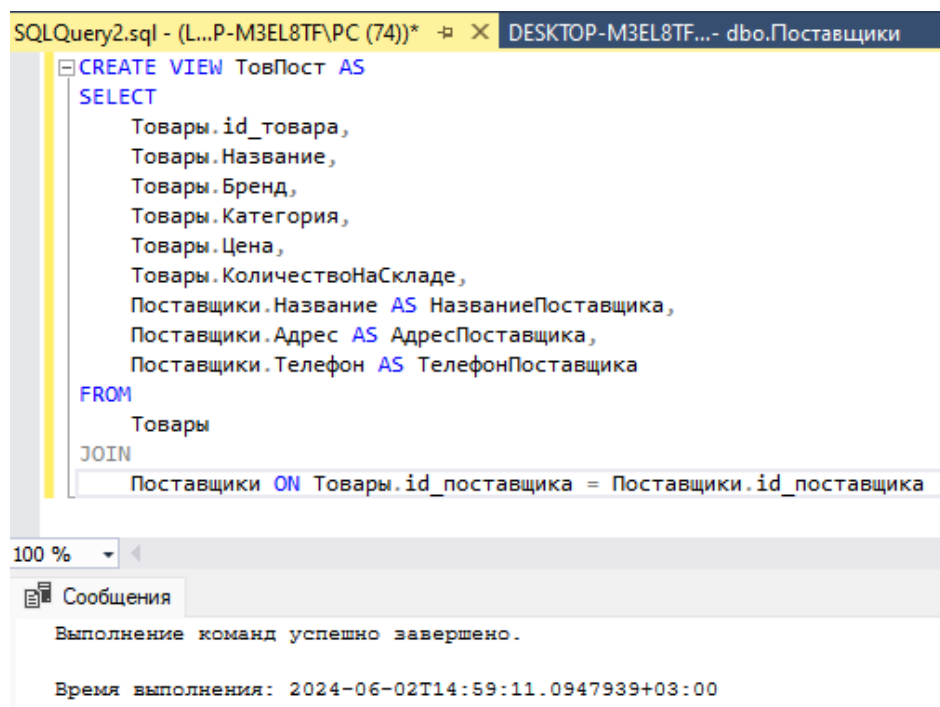


Рисунок 27 – Создание представления с помощью запроса

id_товара	Название	Бренд	Категория	Цена	КоличествоНаСкладе	НазваниеПоставщика	АдресПоставщика	ТелефонПоставщика
1	USB зарядка для телефона	Samsung	Провода	132.17	90	Белокосмос	Ул. Разбитых фонарей, 31	89002931221
2	Телефон A71	Samsung	Телефоны	13770.90	7	Белокосмос	Ул. Разбитых фонарей, 31	89002931221
3	iPhone 15	Apple	Телефоны	134999.55	4	EuroStars	Просп. Энгельса, 7	89117663411
4	iPhone XR	Apple	Телефоны	36000.45	9	EuroStars	Просп. Энгельса, 7	89117663411
5	GeoMetrix A7	LG	Телефоны	7199.55	0	EuroStars	Просп. Энгельса, 7	89117663411

Рисунок 28 – Результат работы представления

Создание транзакции, которая удаляет товар из базы данных и записывает информацию об этом удалении во временную таблицу. Результатом выполнения транзакции является вывод сообщения о удаленном товаре и его даты удаления (Рисунок 29).

SQLQuery3.sql - (L...P-M3EL8TF\PC (59))* X DESKTOP-M3EL8TF\...tore - dbc

```
BEGIN TRANSACTION;

DECLARE @УдаленныйТовар NVARCHAR(100);

CREATE TABLE #Результат (
    Сообщение NVARCHAR(255),
    Дата DATETIME
);

SELECT @УдаленныйТовар = Название
FROM Товары
WHERE id_товара = 22;

DELETE FROM Товары
WHERE id_товара = 22;

INSERT INTO #Результат (Сообщение, Дата)
VALUES ('Удален товар: ' + @УдаленныйТовар, GETDATE());

COMMIT TRANSACTION;

SELECT * FROM #Результат;

DROP TABLE #Результат;
```

100 %

Результаты Сообщения

	Сообщение	Дата
1	Удален товар: Планшет Luna51	2024-06-02 15:18:57.487

Рисунок 29 – Результат работы транзакции

5. РАЗРАБОТКА ПРИЛОЖЕНИЯ

В ходе разработки приложения для управления данными и обеспечения функциональности информационной системы был выбран Microsoft SQL Server (MSSQL). MSSQL предоставляет широкие возможности для работы с данными, также обеспечивает удобный и интуитивно понятный интерфейс для администрирования базы данных.

Для серверной части информационной системы выбрана операционная система Windows Server, которая обеспечивает стабильность и масштабируемость, необходимые для надежной работы MSSQL. СУБД MSSQL станет основой для работы с данными, предоставляя мощные инструменты для хранения, обработки и анализа информации.

Клиентская часть системы, построенная по архитектуре клиент-сервер, разрабатывается на языке программирования C#. Этот выбор обоснован высокой производительностью, простотой в использовании и широкими возможностями для разработки многофункциональных приложений. Использование языка C# позволяет обеспечить эффективное взаимодействие между клиентом и сервером.

Для разработки клиентской части приложения используется Microsoft Visual Studio 2022, это современная среда разработки, которая предоставляет все необходимые инструменты для создания, тестирования и отладки приложений.

5.1. Определение дизайна и разработка графического интерфейса

Графический интерфейс для работы с базами данных выполняет важную роль в упрощении и повышении эффективности различных задач, связанных с администрированием и управлением данными:

Интуитивно понятная навигация: Графический интерфейс предоставляет

пользователям понятные и легко доступные меню, панели инструментов, которые облегчают навигацию по различным функциям и возможностям базы данных.

Удобство администрирования: Администраторы базы данных могут легко управлять пользователями, назначать права доступа и выполнять другие административные задачи через графический интерфейс, что снижает вероятность ошибок и повышает эффективность.

Визуализация данных: предоставляет возможность представлять данные в наглядной форме, что облегчает их анализ и способствует более обоснованному принятию решений.

Таким образом, графический интерфейс значительно упрощает взаимодействие пользователей с базами данных, делая их более доступными и удобными для широкого круга пользователей.

5.2. Графический интерфейс приложения предприятия

Для клиентов и персонала разработан простейший и понятный дизайн-интерфейс. Создано несколько форм, главная из них является MainForm, в которой и будет происходить основное управление базой данных (Рисунок 30).

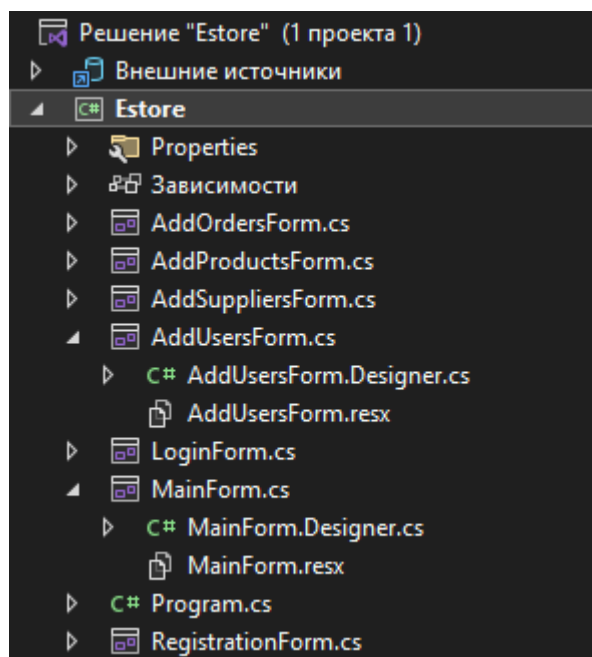


Рисунок 30 – Элементы приложения

При запуске приложения отображается окно входа в систему, где также предлагается зарегистрироваться новому пользователю (Рисунок 31).

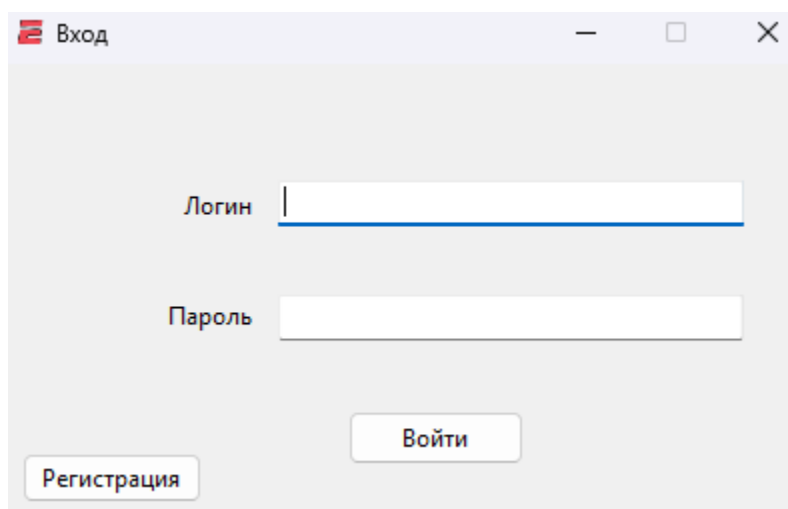


Рисунок 31 – Начальное окно входа

Если ввести неправильный или пустой логин или пароль, то выведет окно с ошибкой (Рисунок 32).

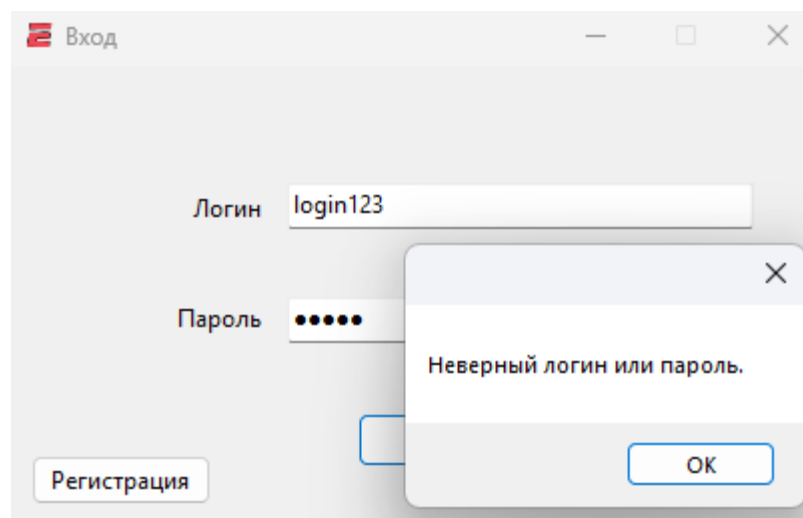


Рисунок 32 – Ошибка при входе в систему

При нажатии на кнопку Регистрация откроется форма регистрации нового пользователя в системе (Рисунок 33).

Рисунок 33 – Регистрация нового пользователя

Процесс регистрации нового пользователя в систему (Рисунки 34-35).

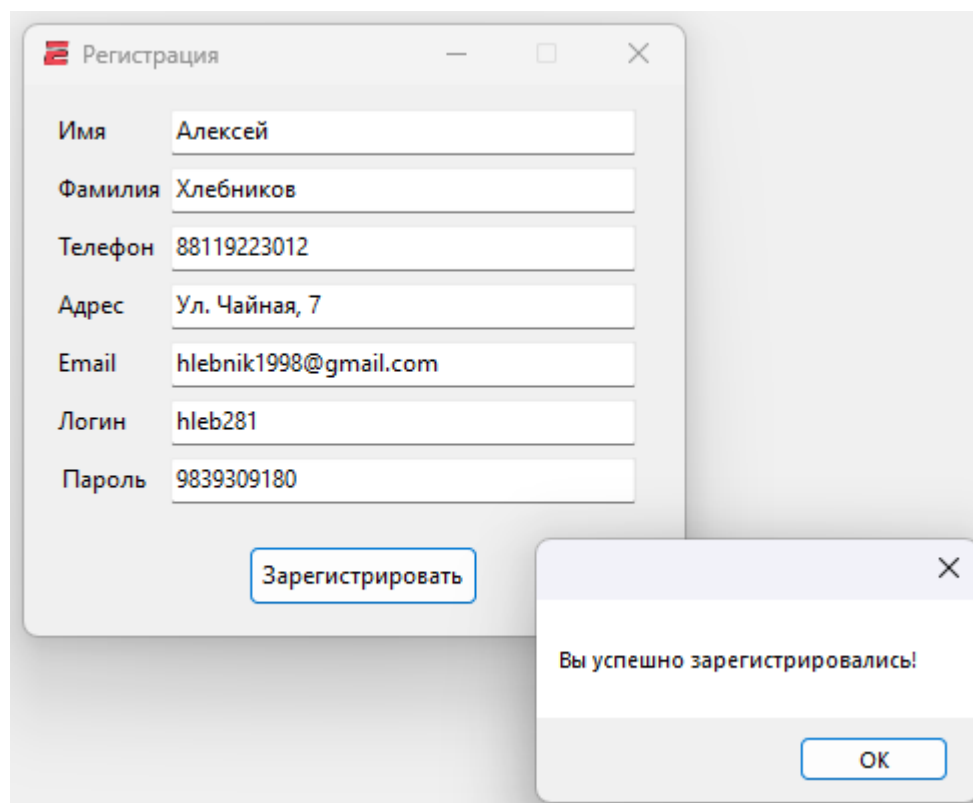


Рисунок 34 – Успешное прохождение регистрации нового пользователя

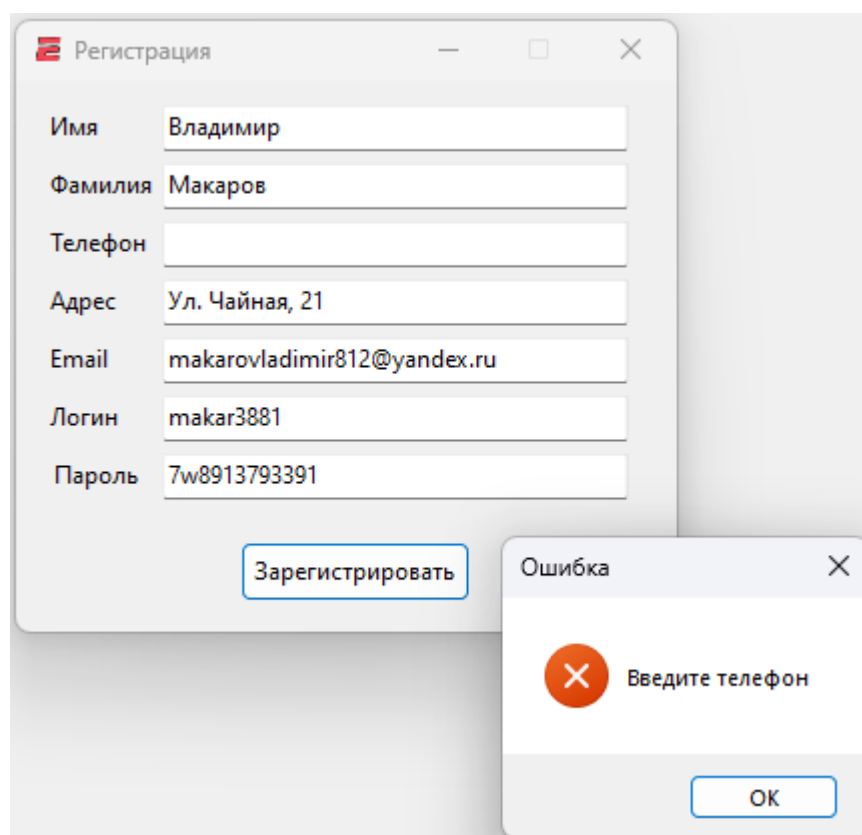


Рисунок 35 – Сообщение с ошибкой при не заполнении какого-либо поля

Авторизация в систему под разными уровнями доступа (Рисунки 36-38).

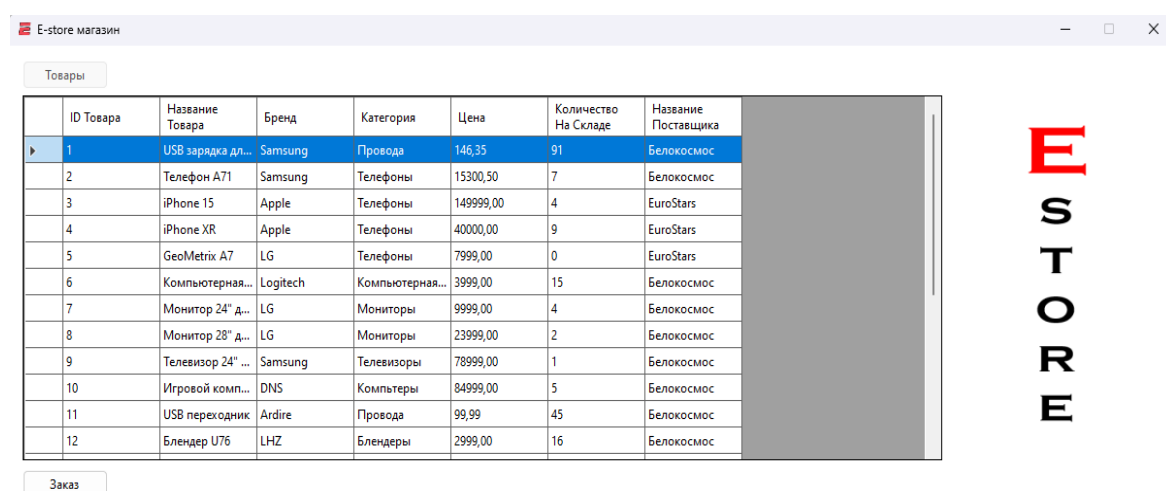


Рисунок 36 – Авторизация в систему как Клиент

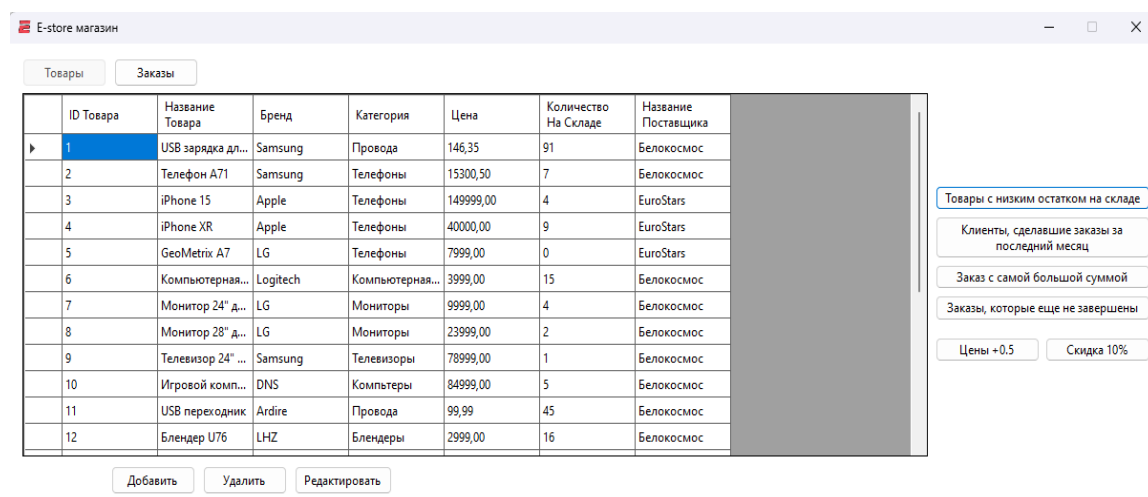


Рисунок 37 – Авторизация в систему как Персонал

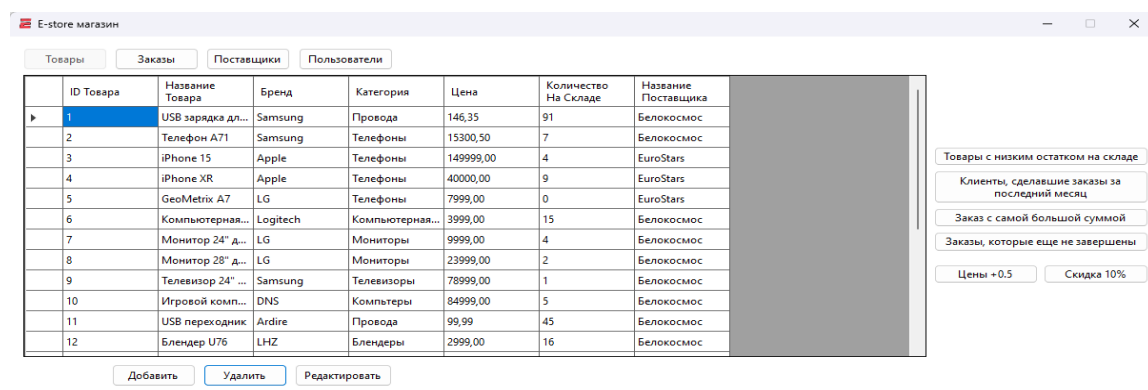


Рисунок 38 – Авторизация в систему как Администратор

Представление всех таблиц и формы графического интерфейса (Рисунки 39-50).

E-store магазин

Товары Заказы Поставщики Пользователи

	ID Товара	Название Товара	Бренд	Категория	Цена	Количество На Складе	Название Поставщика
▶	1	USB зарядка дл...	Samsung	Провода	146,35	91	Белокосмос
	2	Телефон A71	Samsung	Телефоны	15300,50	7	Белокосмос
	3	iPhone 15	Apple	Телефоны	149999,00	4	EuroStars
	4	iPhone XR	Apple	Телефоны	40000,00	9	EuroStars
	5	GeoMetrix A7	LG	Телефоны	7999,00	0	EuroStars
	6	Компьютерная...	Logitech	Компьютерная...	3999,00	15	Белокосмос
	7	Монитор 24" д...	LG	Мониторы	9999,00	4	Белокосмос
	8	Монитор 28" д...	LG	Мониторы	23999,00	2	Белокосмос
	9	Телевизор 24" ...	Samsung	Телевизоры	78999,00	1	Белокосмос
	10	Игровой комп...	DNS	Компьютеры	84999,00	5	Белокосмос
	11	USB переходник	Ardire	Провода	99,99	45	Белокосмос
	12	Блендер U76	LHZ	Блендеры	2999,00	16	Белокосмос

Рисунок 39 – Таблица Товары

E-store магазин

Товары Заказы Поставщики Пользователи

	ID Заказа	Название Товара	Имя Клиента	Дата Заказа	Общая Сумма	Статус Заказа
*						

Рисунок 40 – Таблица Заказы

E-store магазин				
<div>Товары</div> <div>Заказы</div> <div>Поставщики</div> <div>Пользователи</div>				
	id_поставщика	Название	Адрес	Телефон
▶	1	Белокосмос	Ул. Разбитых ...	89002931221
	2	EuroStars	Просп. Энгель...	89117663411
	3	ZipLock	Просп. Науки, ...	89003942219
*				

Рисунок 41 – Таблица Поставщики

E-store магазин									
<div>Товары</div> <div>Заказы</div> <div>Поставщики</div> <div>Пользователи</div>									
	id_клиента	Имя	Адрес	Фамилия	Телефон	ЭлектроннаяПоч	Логин	Пароль	Статус
▶	1	Александр	Ул. Белая, 43	Лебедев	89117231129	AlexanderLebe...	adm	adm	adm
	2	Михаил	Ул. Карпинско...	Габкий	89002998030	gibok@mail.ru	miha	911miha	client
	3	Сергей	Ул. Школьная	Бричкин	89112312211	sergeyBrich@y...	serg	serg123	pers
	4	Иван	Просп. Энгель...	Ксенонов	89110223327	xenon72@gmai...	xenonivan	4894024	client
	5	Сергей	Просп. Науки, ...	Петров	88113842292	airPush873@m...	petrov5	984484petrov	client
	6	Артём	Ул. Хлопина, 9	Сорокин	88003849863	sorokin1976@g...	arSorokin	7877	client
	7	Анна	Ул. Политехни...	Васильева	89118334219	annaVasilka@y...	annaVasilka92	933842V	pers
	8	Никита	Ул. Грусти, 18	Широков	89223742312	dhindi@gmail....	shirokovnikita	842494Yt8389	client
	9	Сергей	Просп. Науки, ...	Поляков	89118173633	polyakov2003@...	waterBottle	38394822442	client
	10	Кира	Ул. Шишкина, 4	Орлова	88337331296	87KiraOrlova@...	KirkaOrlovaa	orlova8492	client
	11	Георгий	Ул. Моряков	Михалков	89501119327	mihalkovG74@...	mihalkovG	83938913	client
	12	Алексей	Ул. Чайная, 7	Хлебников	88119223012	hlebnik1998@g...	hleb281	9839309180	client

Рисунок 42 – Таблица Пользователи

Добавление товара

Название

Бренд

Категория

Цена

Количество

Поставщик

Белокосмос

Добавить

Рисунок 43 – Окно добавления нового товара

Редактирование товара #1

Название	USB зарядка для телефона
Бренд	Samsung
Категория	Провода
Цена	146,35
Количество	91
Поставщик	Белокосмос

Изменить

Рисунок 44 – Окно редактирования существующего товара

Добавление заказа

Клиент	Лебедев
Дата заказа	26 мая 2024 г.
Сумма заказа	
Статус заказа	active
Товар	1

Добавить

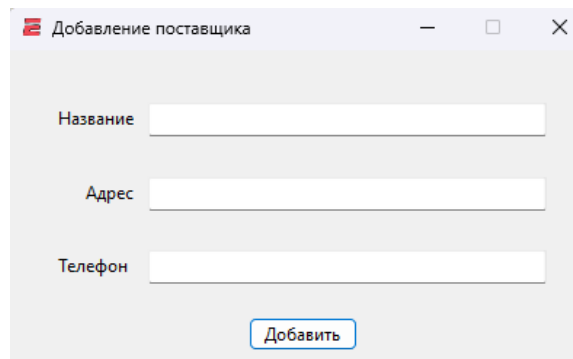
Рисунок 45 – Окно добавления нового заказа

Редактирование заказа #1

Клиент	Лебедев
Дата заказа	26 мая 2024 г.
Сумма заказа	146,35
Статус заказа	active
Товар	1

Изменить

Рисунок 46 – Окно редактирования существующего заказа



Добавление поставщика

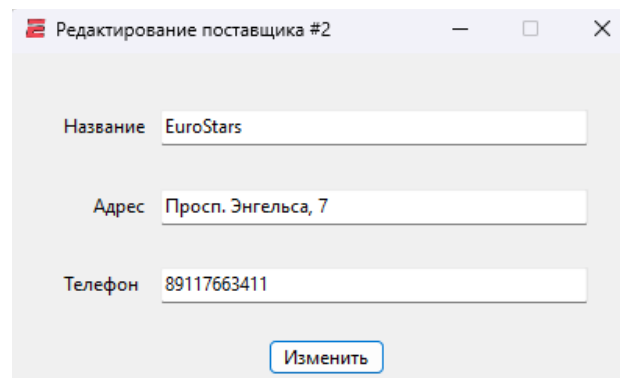
Название

Адрес

Телефон

Добавить

Рисунок 47 – Окно добавления нового поставщика



Редактирование поставщика #2

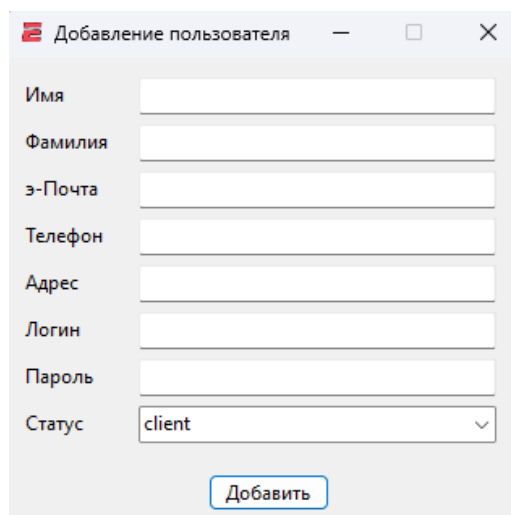
Название EuroStars

Адрес Просп. Энгельса, 7

Телефон 89117663411

Изменить

Рисунок 48 – Окно редактирования существующего поставщика



Добавление пользователя

Имя

Фамилия

э-Почта

Телефон

Адрес

Логин

Пароль

Статус client

Добавить

Рисунок 49 – Окно добавления нового пользователя

Имя	Анна
Фамилия	Васильева
э-Почта	annaVasilka@yandex.ru
Телефон	89118334219
Адрес	Ул. Политехническая, 73
Логин	annaVasilka92
Пароль	933842V
Статус	pers

Изменить

Рисунок 50 – Окно редактирования существующего пользователя

5.3. Выполнение механизмов приложения

Для эффективного функционирования и удобства использования приложения были разработаны механизмы, позволяющие добавлять товары, поставщиков и пользователей в базу данных. Также реализованы функции для управления заказами клиентами. Приложение предоставляет возможность редактирования информации о товарах и поставщиках, что помогает поддерживать актуальность данных. Кроме того, предусмотрены инструменты для анализа продаж. Все эти механизмы делают работу магазина электроники более организованной и эффективной.

Далее будут рассмотрены все вышеперечисленные механизмы для клиента:

После авторизации за клиента появляется окно, в котором клиент может увидеть список товаров, а также кнопку Заказать (Рисунок 51).

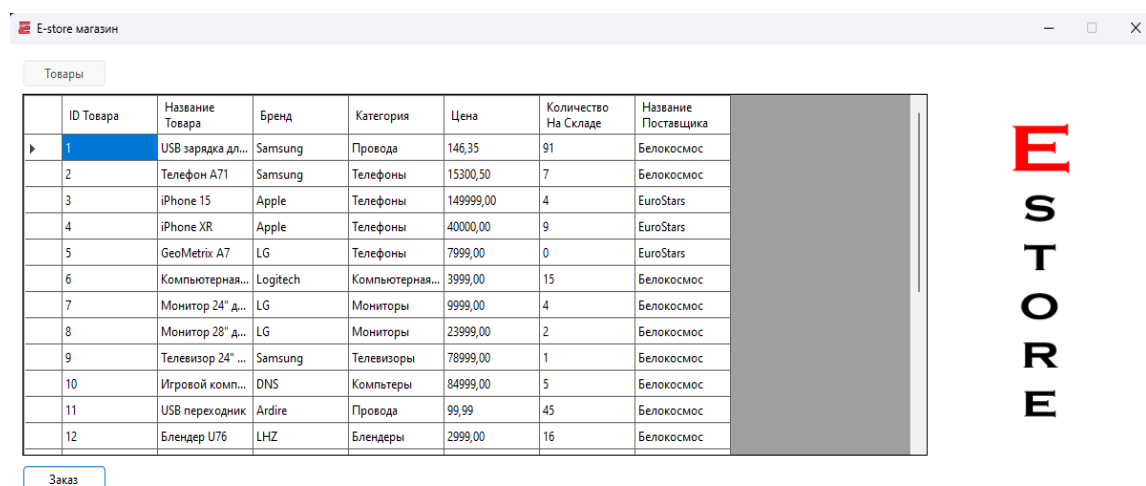


Рисунок 51 – Окно приложения от лица Клиента

Предположим, что клиент желает оформить заказ понравившиеся товара. Для этого клиенту необходимо выделить всю строку товара и нажать на кнопку «Заказ», а после этого ему будет предложено подтвердить заказ (Рисунок 52).

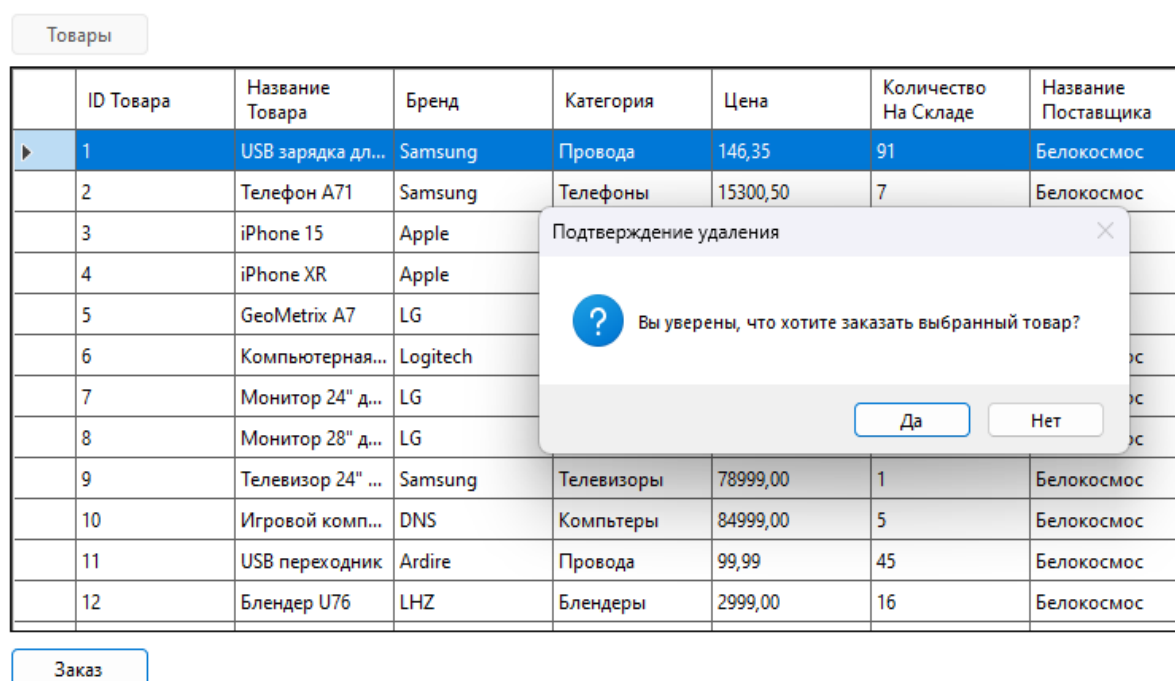


Рисунок 52 – Подтверждение заказа товара

Если товар был в наличии, то успешно создаётся заказ в таблице Заказы от данного клиента, а количество на складе уменьшится на единицу (Рисунки 53-54).

E-store магазин						
<div>Товары</div> <div>Заказы</div> <div>Поставщики</div> <div>Пользователи</div>						
	ID Заказа	Имя Клиента	Название Товара	Дата Заказа	Общая Сумма	Статус Заказа
▶	1	Михаил Габкий	USB зарядка для телефона	26.05.2024	146,35	active
*						

Рисунок 53 – Созданный активный заказ в таблице Заказы

	ID Товара	Название Товара	Бренд	Категория	Цена	Количество На Складе	Название Поставщика
▶	1	USB зарядка дл...	Samsung	Провода	146,35	90	Белокосмос
	2	Телефон A71	Samsung	Телефоны	15300,50	7	Белокосмос

Рисунок 54 – Уменьшение количества товара в таблице Товары

Если выбрать не всю строку товара и нажать на кнопку Заказ, то ничего не произойдёт. Также при заказе товара, которого нет в наличии, будет выведена ошибка и товар не будет заказан (Рисунок 55).

	4	iPhone XR	Apple	Телефоны	40000,00	9	EuroStars
▶	5	GeoMetrix A7	LG	Телефоны	7999,00	0	EuroStars
	6	Компьютерная...	Logitech	Компьютерная...	3999,00		
	7	Монитор 24" д...	LG	Мониторы	9999,00		
	8	Монитор 28" д...	LG	Мониторы	23999,00		
	9	Телевизор 24" ...	Samsung	Телевизоры	78999,00		
	10	Игровой комп...	DNS	Компьютеры	84999,00		
	11	USB переходник	Ardire	Провода	99,99		

Данного товара нет в наличии.

OK

Рисунок 55 – Ошибка при заказе товара

Далее будут рассмотрены механизмы для персонала:

После авторизации за персонал появляется окно, в котором персонал может добавить, удалить и отредактировать любой товар или заказ. Кроме этого, персоналу доступны кнопки в правой стороне интерфейса для анализа данных и финансов, а также обновление цен товаров (Рисунок 56).

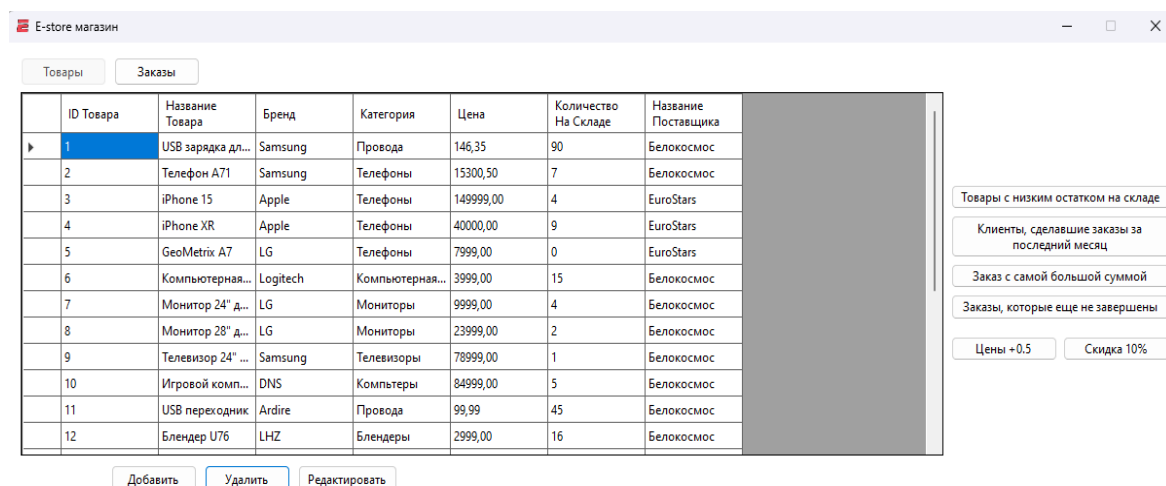


Рисунок 56 – Окно приложения от лица Персонала

Персонал может добавить товар путём нажатия на кнопку **Добавить**, при этом необходимо находиться в таблице **Товары**. После этого заполняются все поля и нажимается кнопка **Добавить** (Рисунки 57-59).

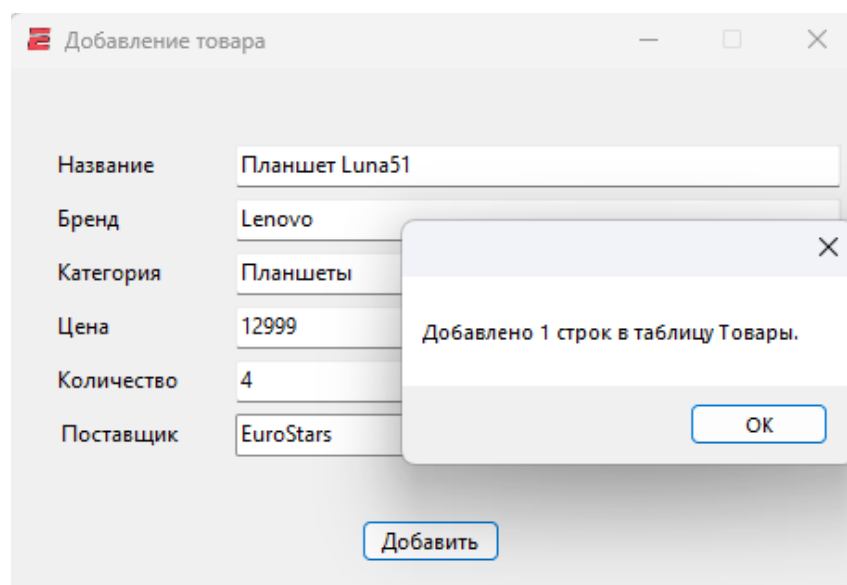


Рисунок 57 – Добавление нового товара

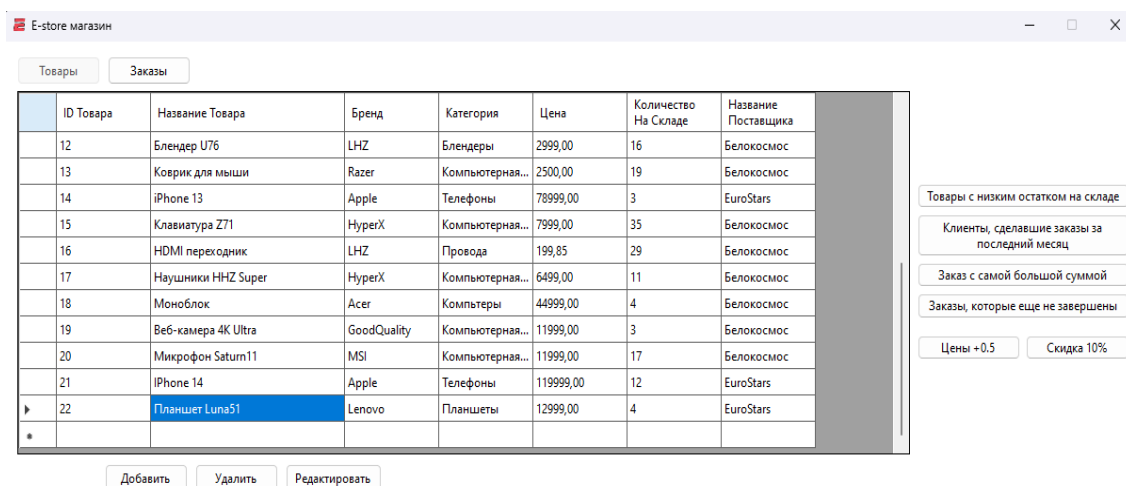


Рисунок 58 – Результат добавления нового товара

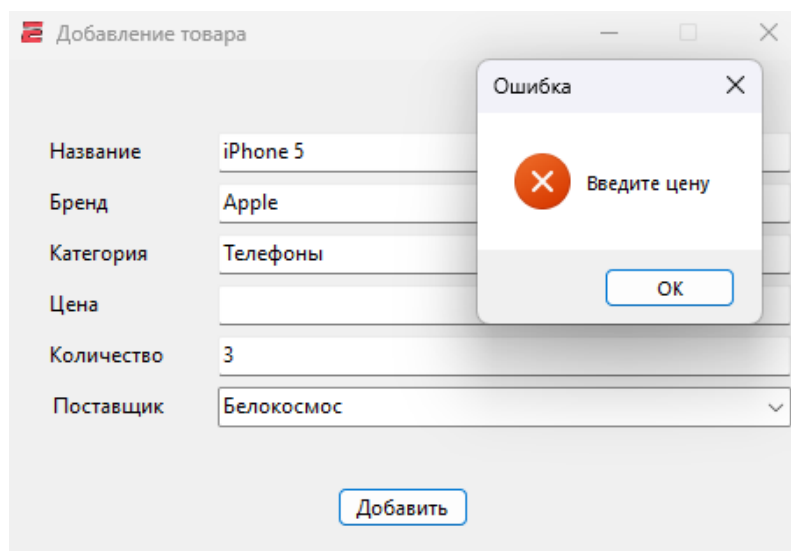


Рисунок 59 – Сообщение с ошибкой при не заполнении какого-либо поля

Персонал также владеет функционалом удаления и редактированием существующего товара. Для этого необходимо выделить всю строку, если этого не сделать, то высветится сообщение с ошибкой (Рисунки 60-62).

E-store магазин

Товары

Заказы

	ID Товара	Название Товара	Бренд	Категория	Цена	Количество На Складе	Название Поставщика
	1	USB зарядка для телефона	Samsung	Провода	146,35	90	Белокосмос
	2	Телефон A71	Samsung	Телефоны	15300,50	7	Белокосмос
▶	3	iPhone 15	Apple	Телефоны	149999,00	4	EuroStars
	4	iPhone XR	Apple	Телефоны	40000,00	9	EuroStars
	5	GeoMetrix A7	LG				roStars
	6	Компьютерная мышь	Logitech				локосмос
	7	Монитор 24" дюйма	LG				локосмос
	8	Монитор 28" дюйма	LG				локосмос
	9	Телевизор 24" дюйма	Samsung				локосмос
	10	Игровой компьютер	DNS				локосмос
	11	USB переходник	Ardire	Провода	99,99	45	Белокосмос
	12	Блендер U76	LHZ	Блендеры	2999,00	16	Белокосмос

Добавить

Удалить

Редактировать

Подтверждение удаления

?

Вы уверены, что хотите удалить выбранный товар

Да

Нет

Рисунок 60 – Окно удаления товара

Редактирование товара #3

Название

iPhone 15

Бренд

Apple

Категория

Телефоны

Цена

149999,00

Количество

4

Поставщик

Белокосмос

Изменить

Рисунок 61 – Окно редактирования товара

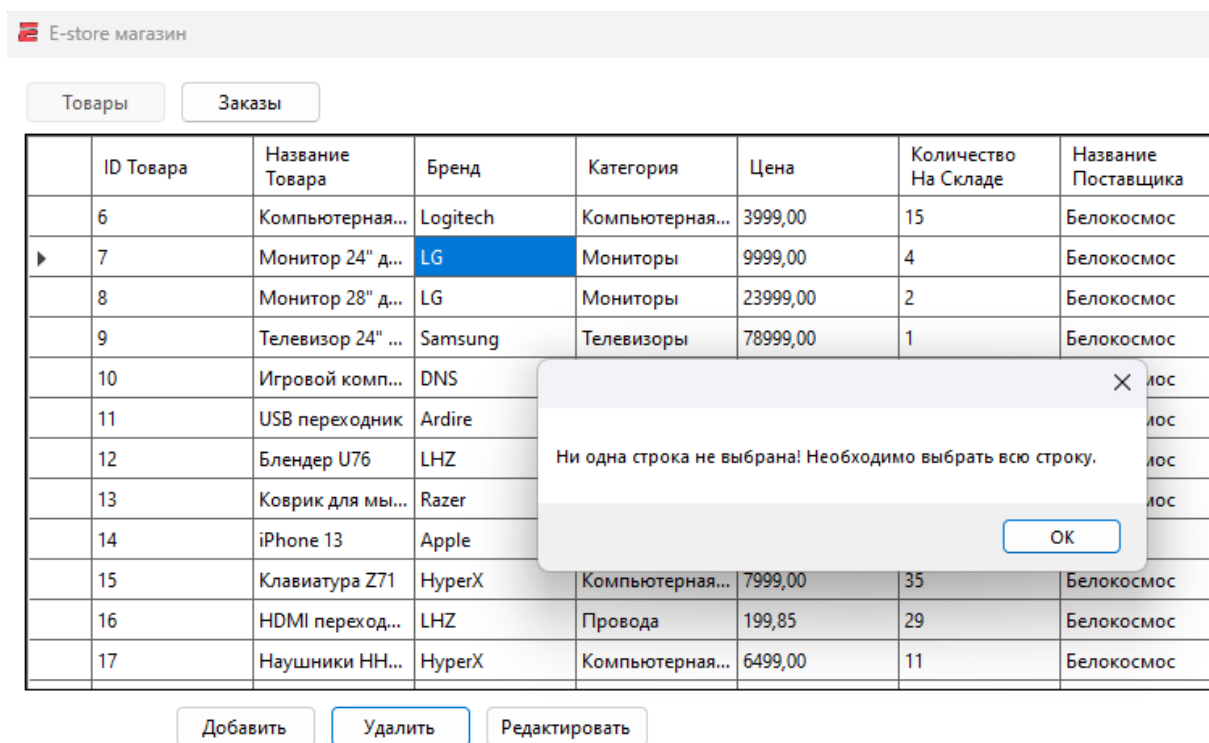


Рисунок 62 – Сообщение с ошибкой удаления или редактирования товара

Аналогичным способом добавляется новый заказ, однако уже нужно находиться в таблице Заказы. После этого заполняются все поля и нажимается кнопка Добавить (Рисунки 63-65).

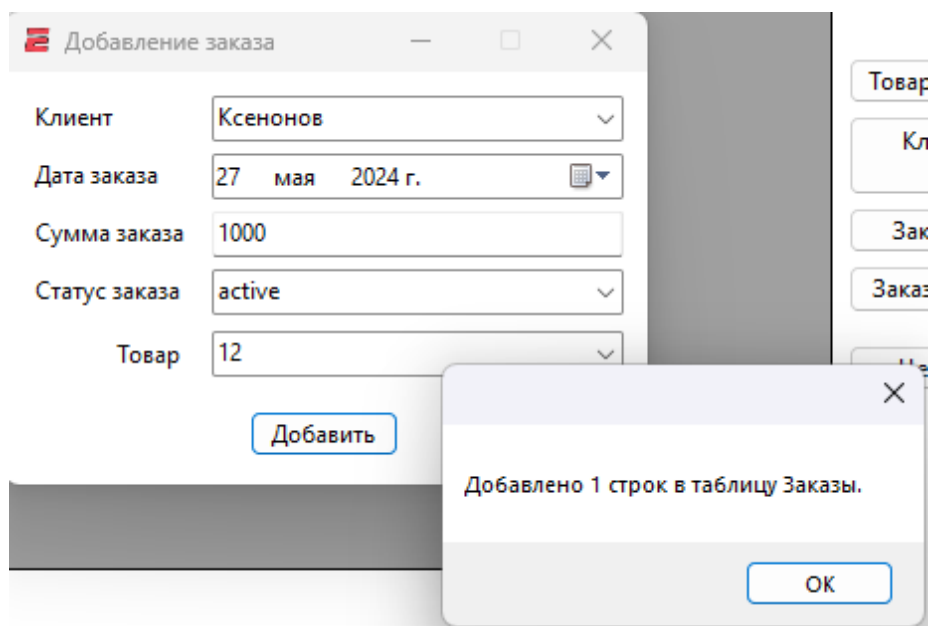


Рисунок 63 – Добавления нового заказа

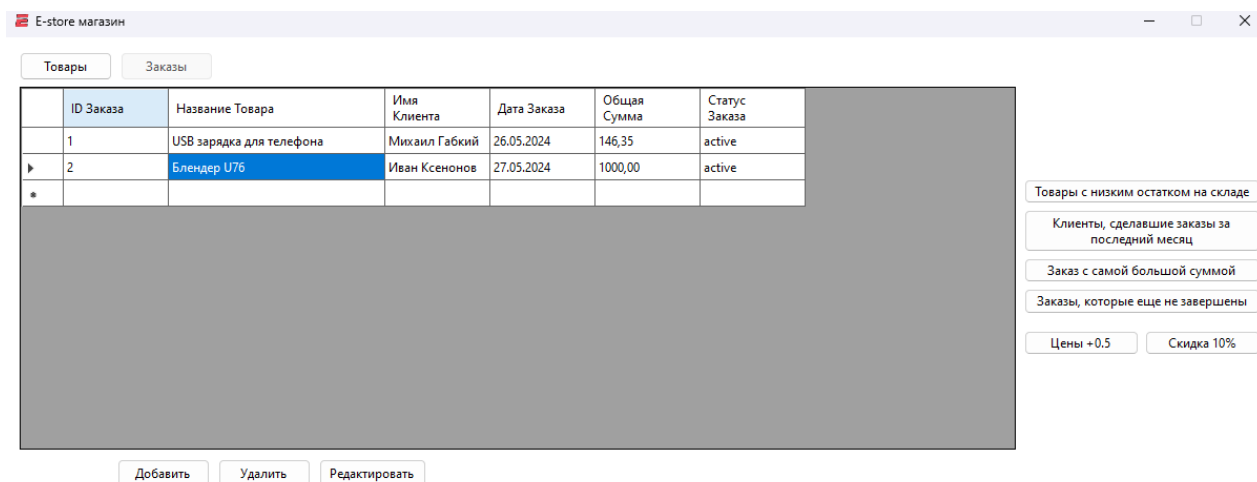


Рисунок 64 – Результат добавления нового заказа

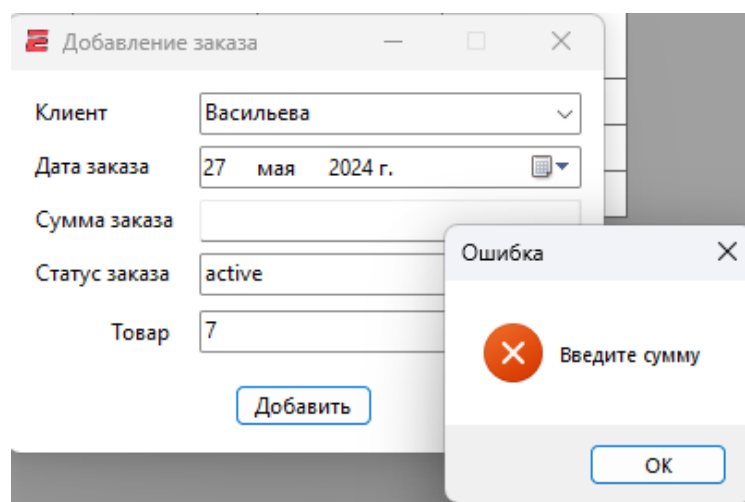


Рисунок 65 – Сообщение с ошибкой при не заполнении какого-либо поля

Процесс удаления и редактирования существующего заказа (Рисунки 66-68).

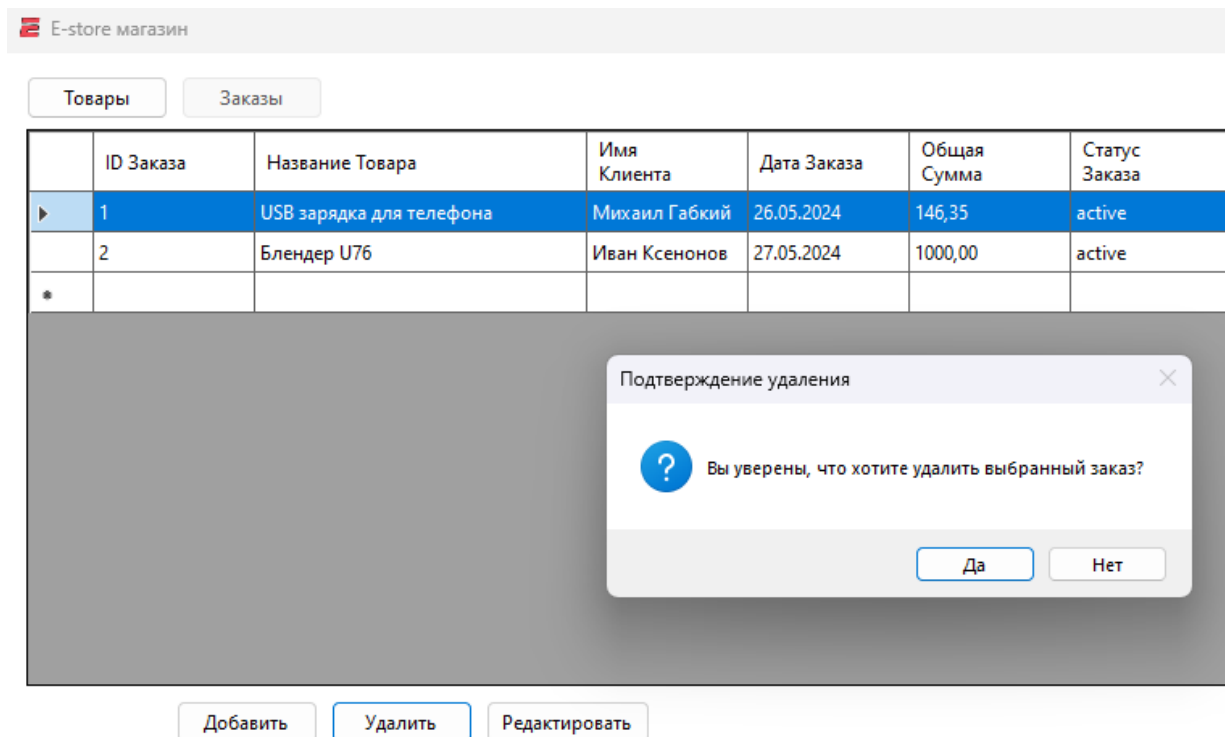


Рисунок 66 – Окно удаления заказа

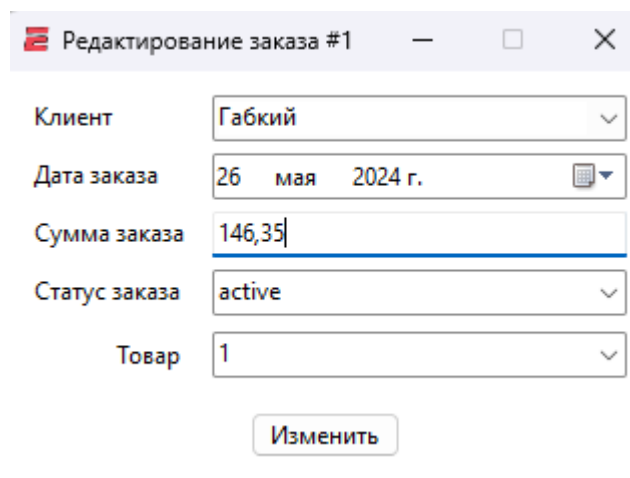


Рисунок 67 – Окно редактирования заказа

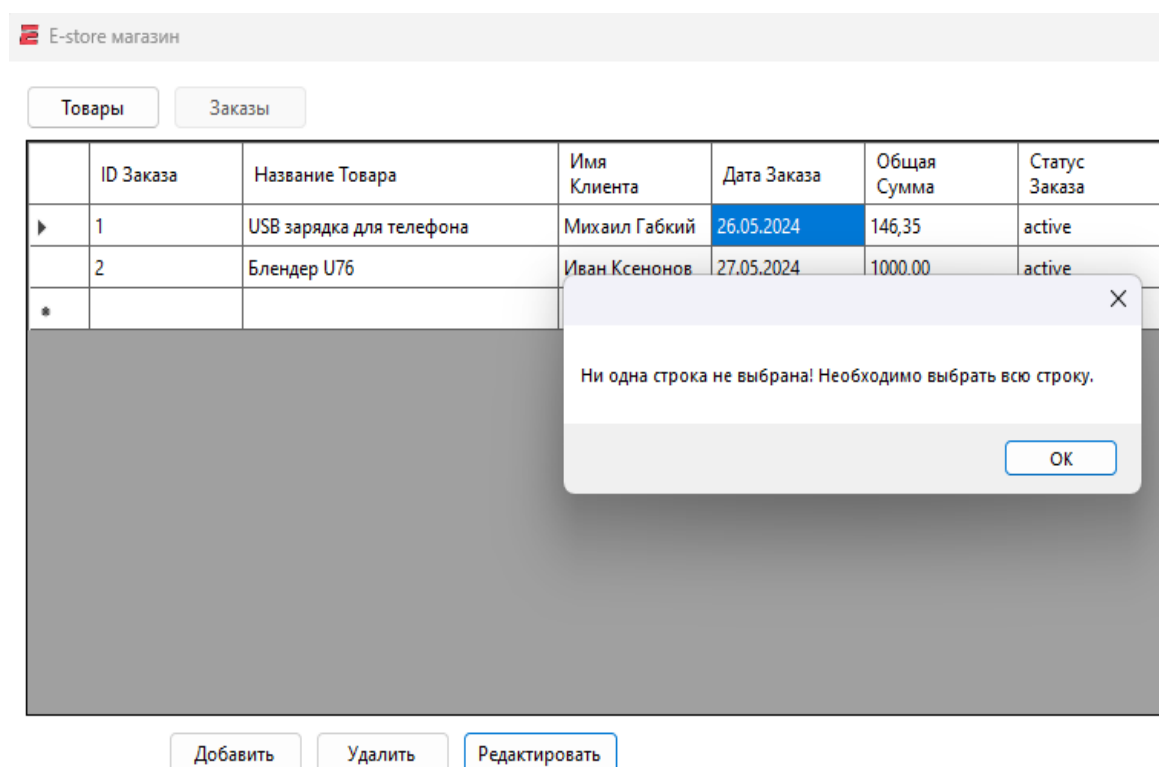


Рисунок 68 – Сообщение с ошибкой удаления или редактирования заказа

Далее будут рассмотрены механизмы для администратора:

После авторизации за администратора появляется окно, в котором администратор может осуществлять полный контроль над всеми таблицами, помимо товаров и клиентов, администратору доступны таблицы поставщики и пользователи, в которой хранятся данные всех клиентов и персонала (Рисунок 69).

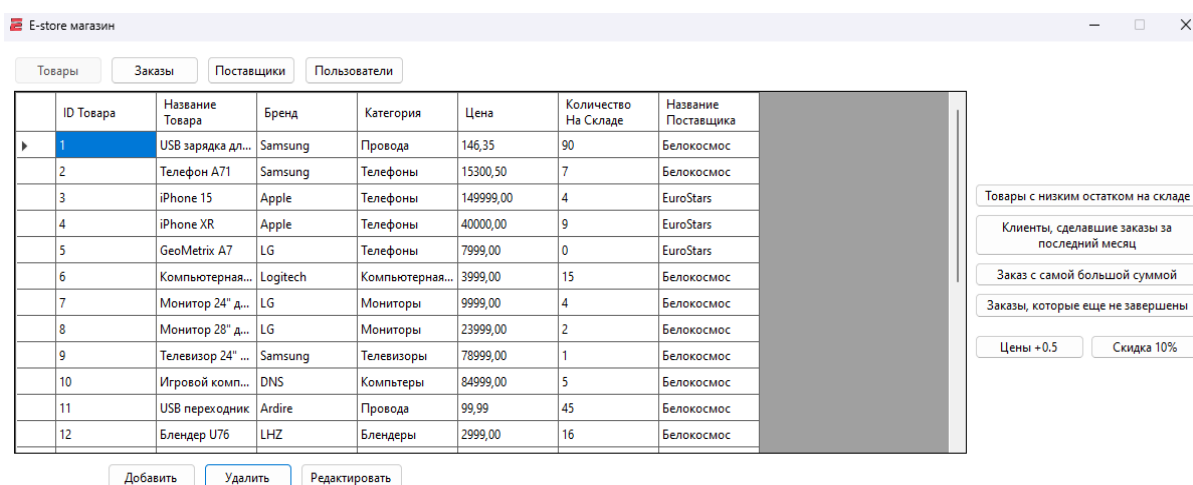


Рисунок 69 – Окно приложения от лица Администратора

Администратор может добавить не только новый товар и заказы, но и поставщиков и пользователей. При этом необходимо находиться в таблице Поставщики или Пользователи. После этого заполняются все поля и нажимается кнопка Добавить (Рисунки 70-75).

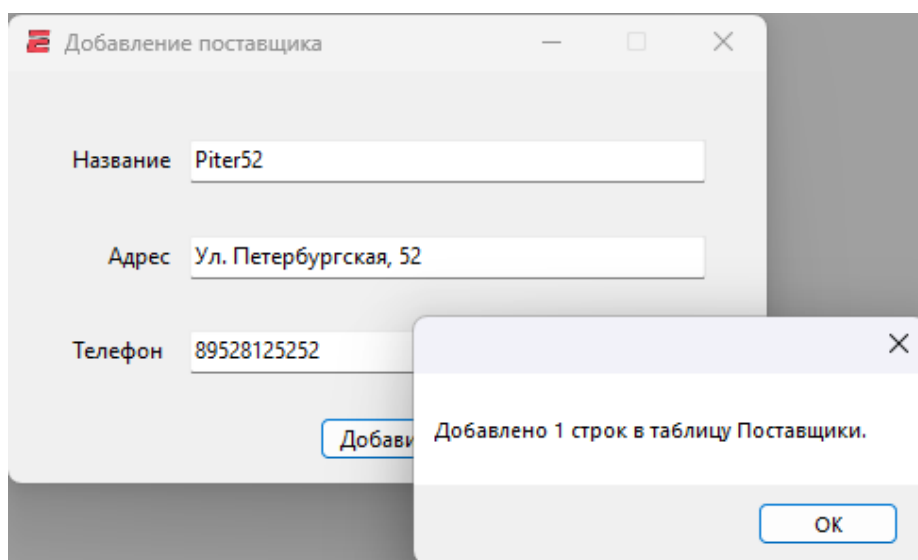


Рисунок 70 – Добавление нового поставщика



Рисунок 71 – Результат добавления нового поставщика

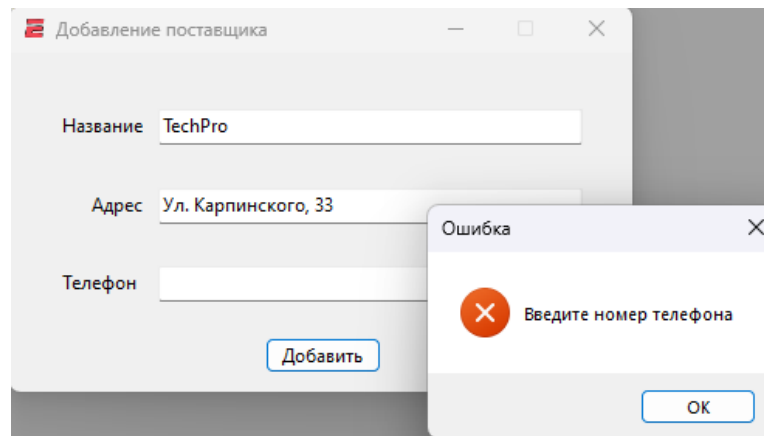


Рисунок 72 – Сообщение с ошибкой при не заполнении какого-либо поля

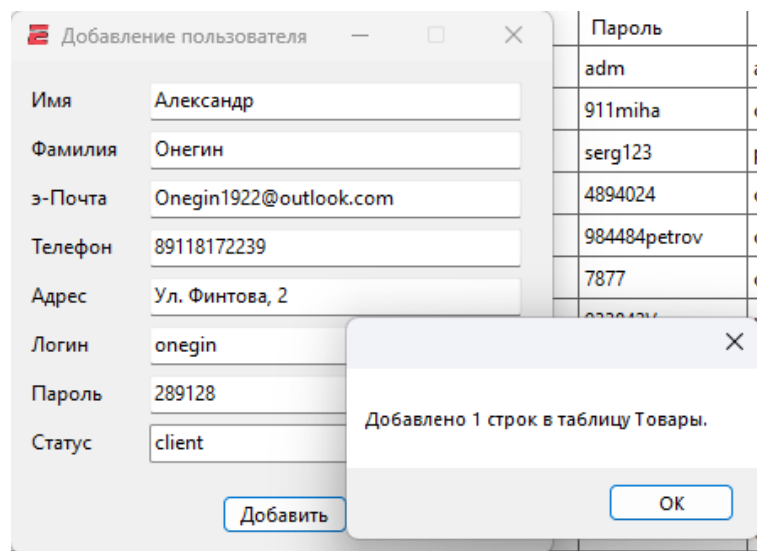


Рисунок 73 – Добавление нового пользователя

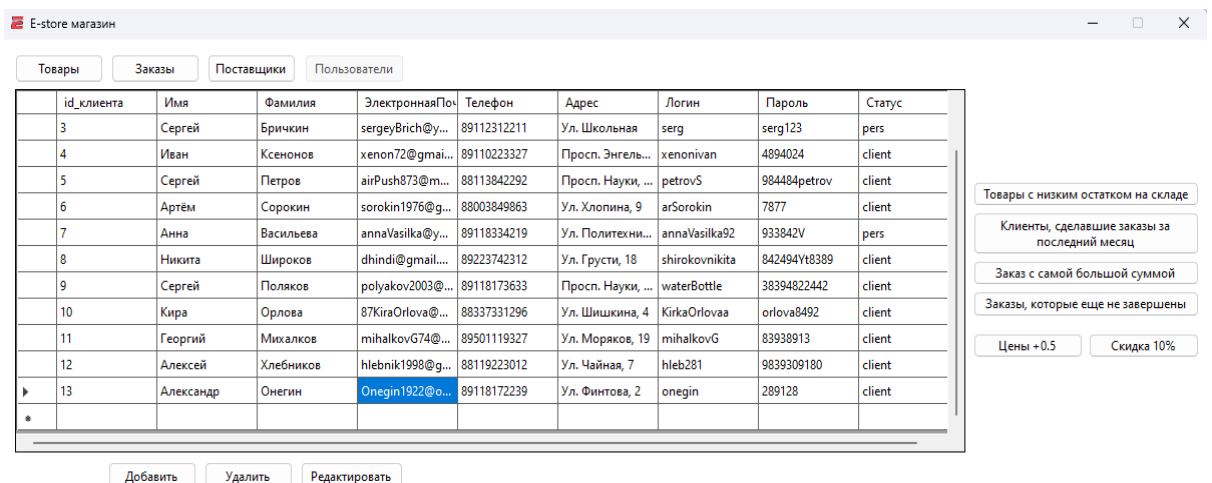


Рисунок 74 – Результат добавления нового пользователя

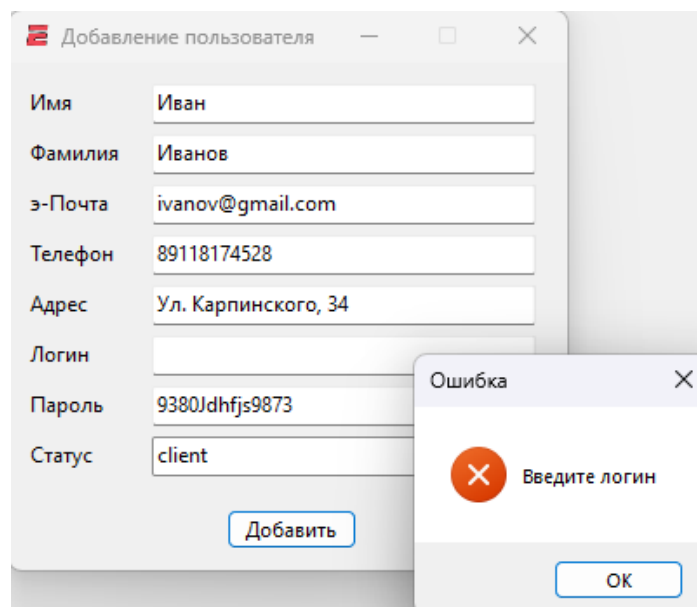


Рисунок 75 – Сообщение с ошибкой при не заполнении какого-либо поля

Процесс удаления и редактирования существующего поставщика (Рисунки 76-78).

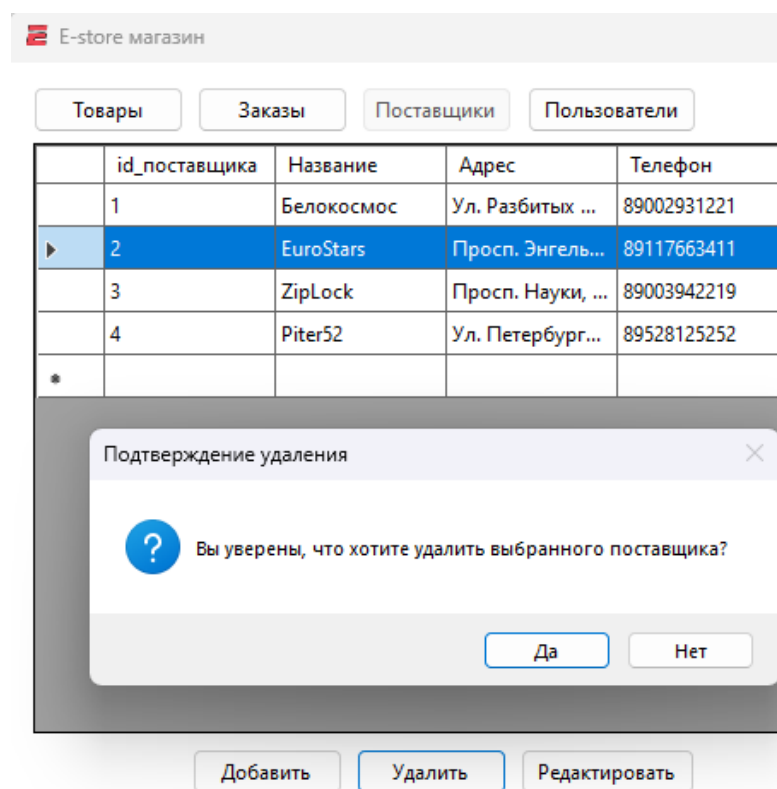


Рисунок 76 – Окно удаления поставщика

Редактирование поставщика #2

Название: EuroStars

Адрес: Просп. Энгельса, 7

Телефон: 89117663411

Изменить

Рисунок 77 – Окно редактирования поставщика

E-store магазин

Товары Заказы Поставщики Пользователи

	id_поставщика	Название	Адрес	Телефон
	1	Белокосмос	Ул. Разбитых ...	89002931221
	2	EuroStars	Просп. Энгель...	89117663411
	3	ZipLock	Просп. Науки, ...	89003942219
▶	4	Piter52	Ул. Петербург...	89528125252
*				

Ни одна строка не выбрана! Необходимо выбрать всю строку.

ОК

Добавить Удалить Редактировать

Рисунок 78 – Сообщение с ошибкой удаления или редактирования поставщика

Процесс удаления и редактирования существующего пользователя (Рисунки 79-81).

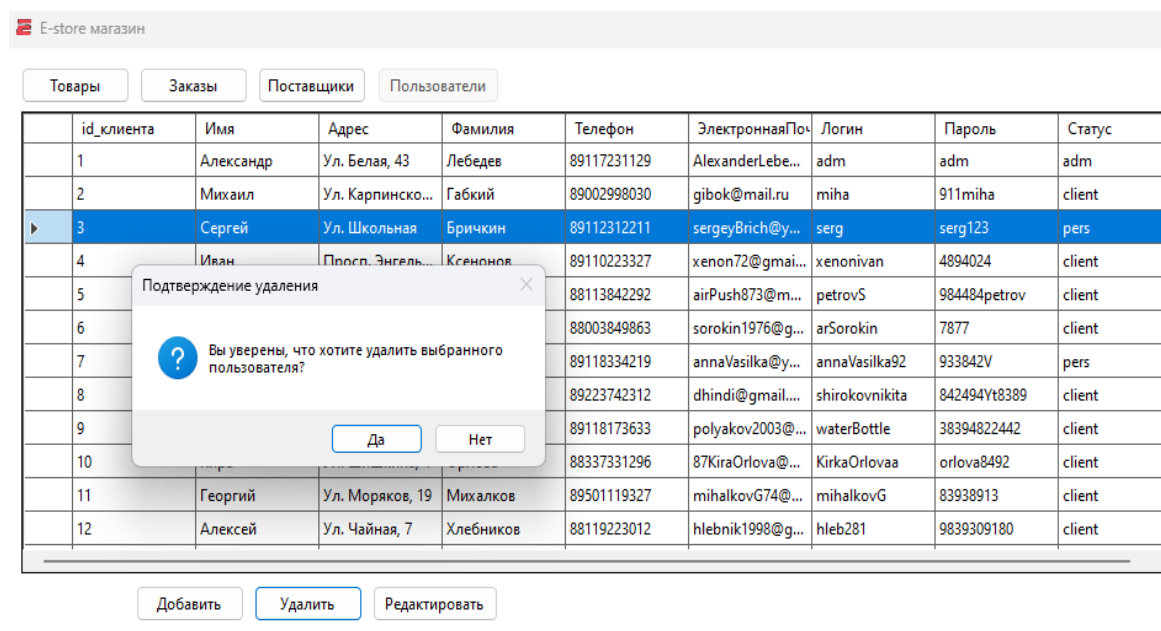


Рисунок 79 – Окно удаления пользователя

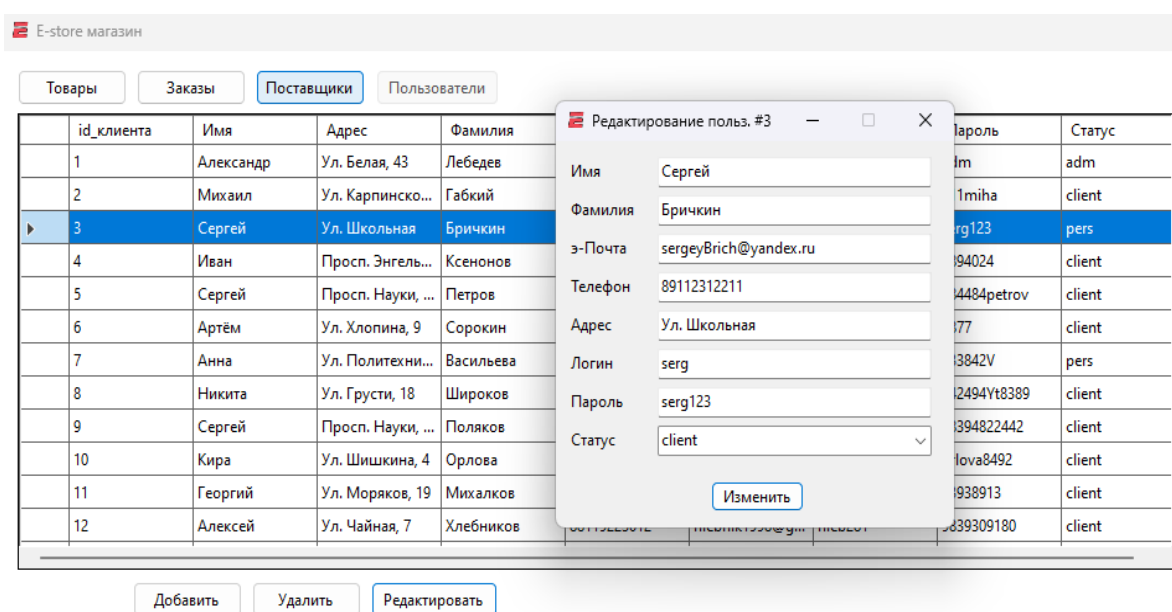


Рисунок 80 – Окно редактирования пользователя

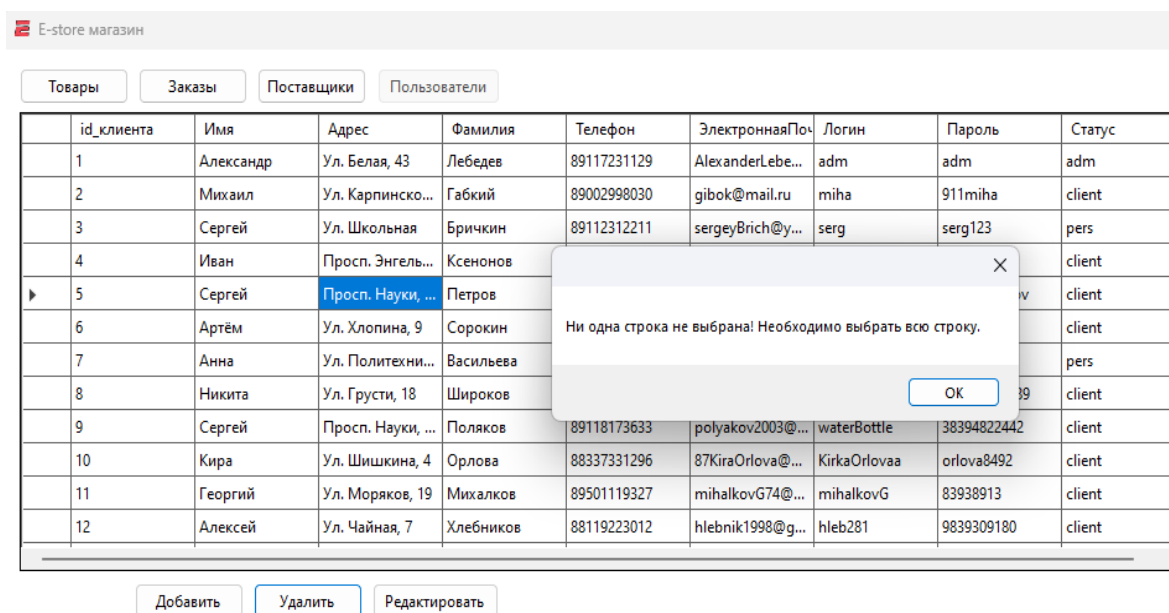


Рисунок 81 – Сообщение с ошибкой удаления или редактирования пользователя

5.4. Реализация запросов

В рамках курсового проекта были разработаны различные запросы для эффективного управления базой данных магазина электроники. Запросы реализованы с использованием SQL и интегрированы в интерфейс приложения, обеспечивая удобное взаимодействие с базой данных.

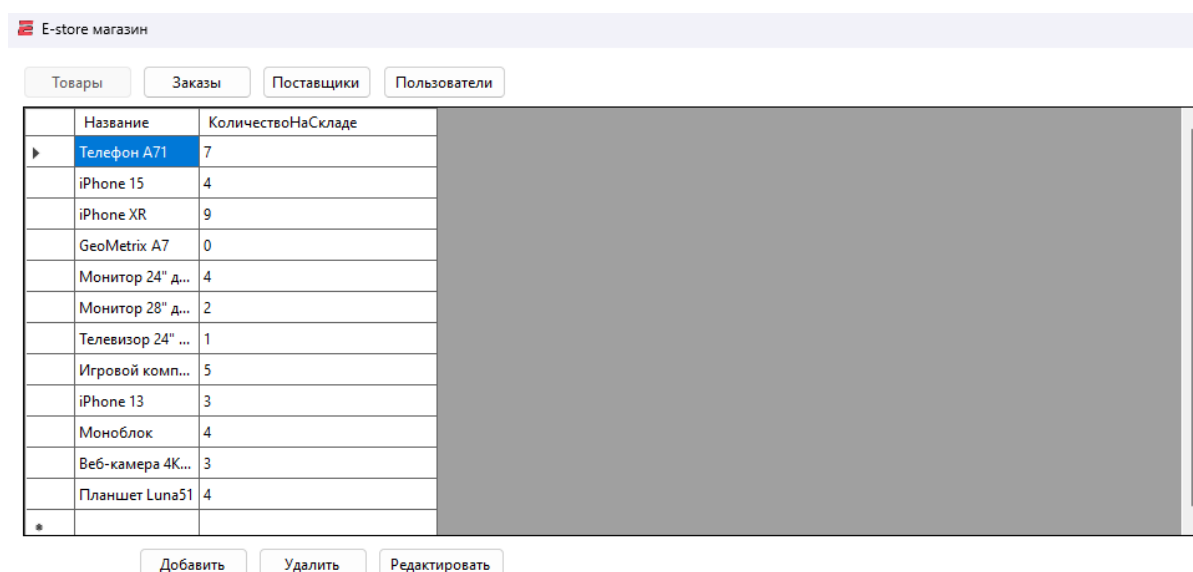
Запросы играют важную роль в анализе данных и финансов. С их помощью можно анализировать данные о продажах. Финансовые запросы позволяют оценивать доходы, расходы и прибыль.

Также разработаны запросы для управления ценами на товары в базе данных магазина электроники. Один из таких запросов позволяет увеличить цену всех товаров на 0.5 единицы. Другой запрос реализует применение скидки в 10 процентов на все товары.

Запрос 1: Товары с низким остатком на складе.

Код SQL запроса: `SELECT Название, КоличествоНаСкладе FROM Товары WHERE КоличествоНаСкладе < 10;`

Результат такого запроса приведён на рисунке 82.



	Название	КоличествоНаСкладе
▶	Телефон A71	7
	iPhone 15	4
	iPhone XR	9
	GeoMetrix A7	0
	Монитор 24" д...	4
	Монитор 28" д...	2
	Телевизор 24" ...	1
	Игровой комп...	5
	iPhone 13	3
	Моноблок	4
	Веб-камера 4К...	3
	Планшет Luna51	4
*		

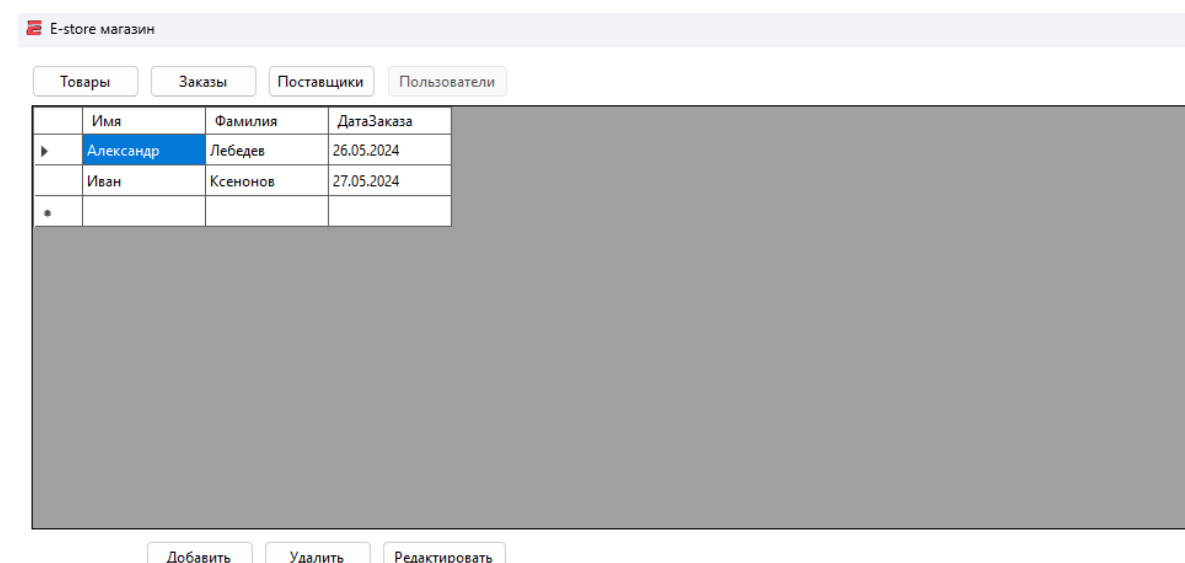
Добавить Удалить Редактировать

Рисунок 82 – Результат запроса 1

Запрос 2: Клиенты, сделавшие заказы за последний месяц.

Код SQL запроса: `SELECT Клиенты.Имя, Клиенты.Фамилия, Заказы.ДатаЗаказа FROM Клиенты JOIN Заказы ON Клиенты.id_клиента = Заказы.id_клиента WHERE Заказы.ДатаЗаказа BETWEEN DATEADD(month, -1, GETDATE()) AND GETDATE();`

Результат такого запроса приведён на рисунке 83.



	Имя	Фамилия	ДатаЗаказа
▶	Александр	Лебедев	26.05.2024
	Иван	Ксенонов	27.05.2024
*			

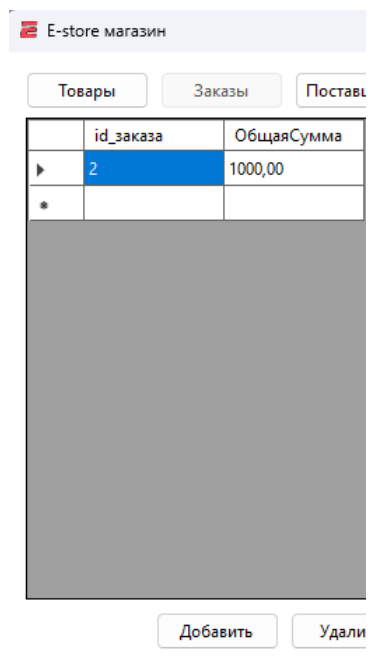
Добавить Удалить Редактировать

Рисунок 83 – Результат запроса 2

Запрос 3: Заказ с самой большой суммой.

Код SQL запроса: `SELECT TOP 1 id_заказа, ОбщаяСумма FROM Заказы ORDER BY ОбщаяСумма DESC;`

Результат такого запроса приведён на рисунке 84.



The screenshot shows a web application titled 'E-store магазин'. It has three tabs: 'Товары', 'Заказы', and 'Поставки'. The 'Заказы' tab is active, displaying a table with two columns: 'id_заказа' and 'ОбщаяСумма'. The first row is highlighted in blue and contains the values '2' and '1000,00'. Below the table is a large grey rectangular area, and at the bottom are two buttons: 'Добавить' and 'Удали'.

	id_заказа	ОбщаяСумма
▶	2	1000,00
*		

Рисунок 84 – Результат запроса 3

Запрос 4: Заказы, которые еще не завершены.

Код SQL запроса: `SELECT id_заказа, id_клиента, ДатаЗаказа, ОбщаяСумма FROM Заказы WHERE Статус = 'active';`

Результат такого запроса приведён на рисунке 85.

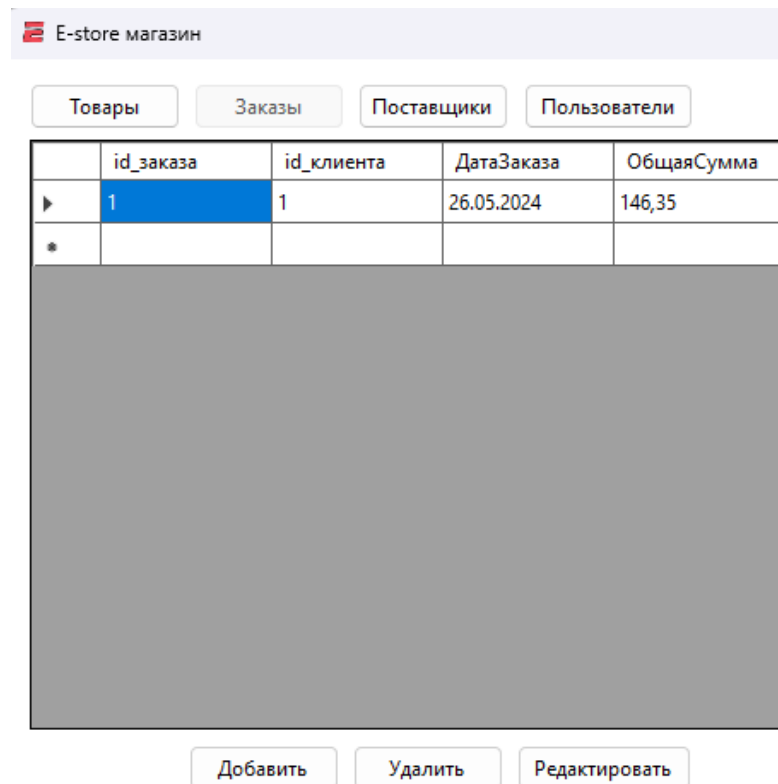


Рисунок 85 – Результат запроса 4

Запрос 5: Цены +0.5.

Код SQL запроса: UPDATE Товары SET Цена = Цена + 0.5;

Результат такого запроса приведён на рисунке 86.

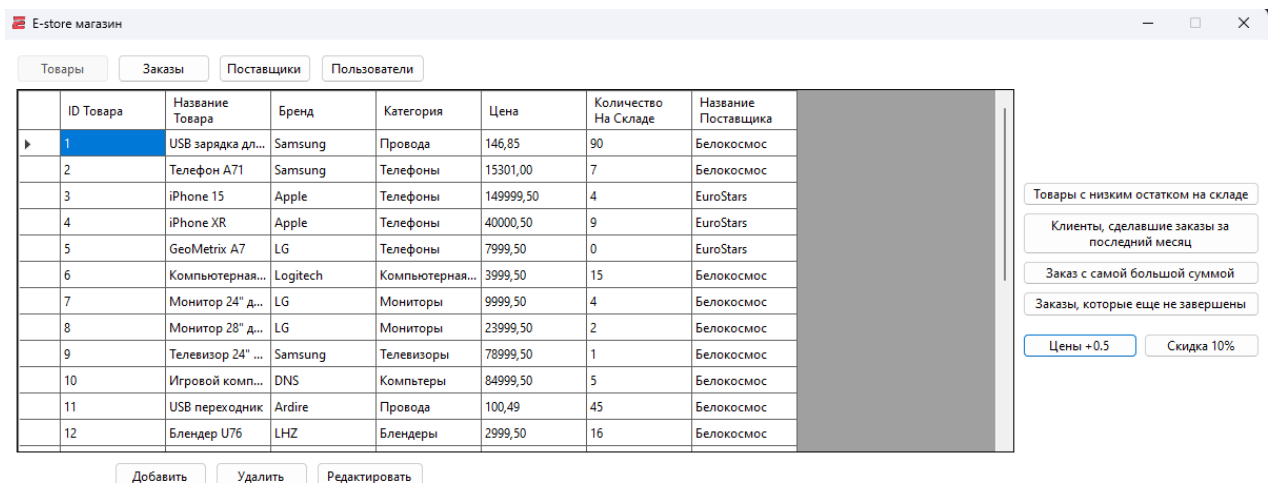
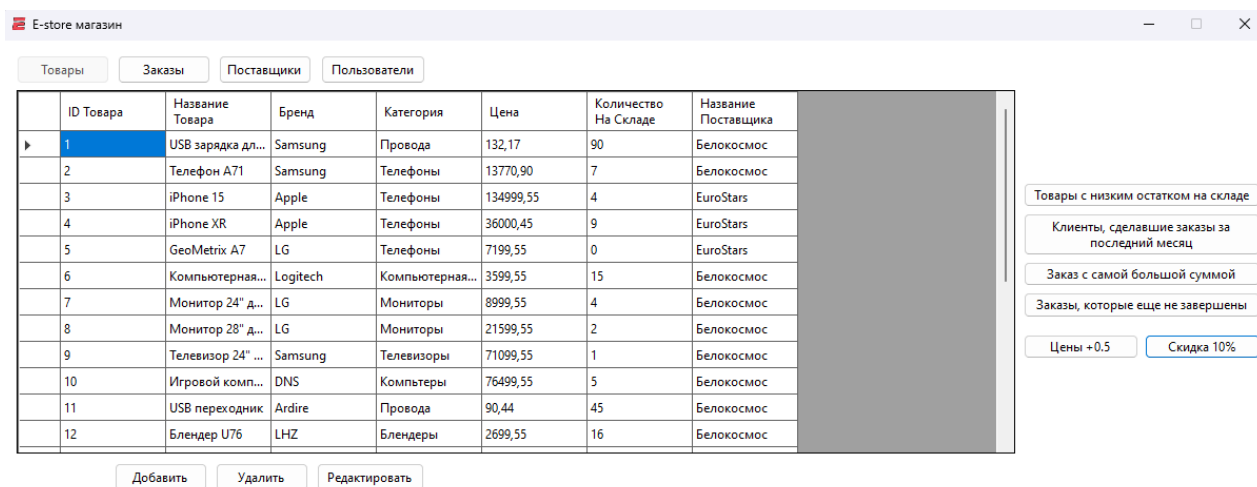


Рисунок 86 – Результат запроса 5

Запрос 6: Скидка 10%.

Код SQL запроса: UPDATE Товары SET Цена = Цена * 0.9;

Результат такого запроса приведён на рисунке 87.



The screenshot shows a web application titled 'E-store магазин'. At the top, there are four tabs: 'Товары' (selected), 'Заказы', 'Поставщики', and 'Пользователи'. Below the tabs is a table with 8 columns: 'ID Товара', 'Название Товара', 'Бренд', 'Категория', 'Цена', 'Количество На Складе', and 'Название Поставщика'. The table contains 12 rows of data. To the right of the table is a sidebar with several filter buttons: 'Товары с низким остатком на складе', 'Клиенты, сделавшие заказы за последний месяц', 'Заказ с самой большой суммой', 'Заказы, которые еще не завершены', 'Цены +0.5', and 'Скидка 10%'. At the bottom of the table, there are three buttons: 'Добавить', 'Удалить', and 'Редактировать'.

ID Товара	Название Товара	Бренд	Категория	Цена	Количество На Складе	Название Поставщика
1	USB зарядка дл...	Samsung	Провода	132,17	90	Белокосмос
2	Телефон A71	Samsung	Телефоны	13770,90	7	Белокосмос
3	iPhone 15	Apple	Телефоны	134999,55	4	EuroStars
4	iPhone XR	Apple	Телефоны	36000,45	9	EuroStars
5	GeoMetrix A7	LG	Телефоны	7199,55	0	EuroStars
6	Компьютерная...	Logitech	Компьютерная...	3599,55	15	Белокосмос
7	Монитор 24" д...	LG	Мониторы	8999,55	4	Белокосмос
8	Монитор 28" д...	LG	Мониторы	21599,55	2	Белокосмос
9	Телевизор 24" ...	Samsung	Телевизоры	71099,55	1	Белокосмос
10	Игровой комп...	DNS	Компьютеры	76499,55	5	Белокосмос
11	USB переходник	Ardire	Провода	90,44	45	Белокосмос
12	Блендер U76	LHZ	Блендеры	2699,55	16	Белокосмос

Рисунок 87 – Результат запроса 6

ЗАКЛЮЧЕНИЕ

В результате курсового проекта была разработана информационная система для управления данными магазина электроники, основанная на применении Microsoft SQL Server (MSSQL). Данный проект включал в себя все этапы разработки: от постановки задачи и анализа требований до реализации и тестирования готового приложения. Основной целью проекта было создание эффективного и удобного инструмента для автоматизации управления магазином, что включало работу с товарами, поставщиками, пользователями и заказами.

Одной из ключевых задач проекта стало создание базы данных, которая обеспечивала бы надежное хранение и быстрый доступ к необходимой информации. Для этого была выбрана СУБД Microsoft SQL Server.

Важным аспектом проекта стало создание графического интерфейса пользователей на языке программирования C#. Этот интерфейс обеспечивает интуитивно понятное взаимодействие пользователей с системой, облегчая выполнение повседневных задач. Интерфейс был разработан с учетом различных уровней доступа, что позволяет разграничивать права и возможности пользователей в зависимости от их роли.

Таким образом, созданная информационная система позволяет автоматизировать управление магазином электроники, обеспечивая надежное хранение данных, удобное взаимодействие пользователей с системой и безопасное распределение прав доступа.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Эндрю С., Дженнифер Г. Изучаем С# – 2022. – 769 с.
2. Фленов М. Е. Библия С# – Издательство ВHV, 2020 г. – 512 с.
3. Бураков П.В., Петров В. Ю. Введение в системы баз данных учебное пособие - Санкт-Петербург, 2010. - 129 с.: ил.
4. Язык SQL в примерах и задачах – Толстобров А. П., Фертиков В. В., 2023г., – 260 с.
5. Программирование на платформе Microsoft .NET Framework 4.5 на языке С#. 4-е изд., Джеффри Рихтер, 2024, 896 с.
6. Трунин В. Путь программиста T-SQL – 2020. – 204 с.
7. Алексей В. Программирование на С# для начинающих. Основные сведения – 2018. – 586 с.
8. Попова- Коварцева Д.А., Сопченко Е.В. Основы проектирования баз данных : учебное пособие / Д. А. Попова-Коварцева. - Самара : СамГУ, 2019. - 112 с.
9. [С# и Windows Forms | Первое приложение с .NET CLI \(metanit.com\)](https://metanit.com/beginner/sql/1.php)
10. [Все операции с БД в графическом приложении в С# \(metanit.com\)](https://metanit.com/beginner/sql/2.php)

ПРИЛОЖЕНИЕ А

Листинг кода главного окна (MainForm):

```
using System;
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Data;
using System.Reflection.Emit;
using static System.ComponentModel.Design.ObjectSelectorEditor;
using System.Diagnostics;
using System.Xml.Linq;
using static System.Runtime.InteropServices.JavaScript.JSType;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Estore
{
    public partial class MainForm : Form
    {
        private SqlConnection connection = new SqlConnection(@"Data Source=(LocalDb)\LocalDBDemo;Initial Catalog=Estore;Integrated
Security=True;");
        private int id_client;
        public MainForm(int id_client, int status_client)
        {
            InitializeComponent();
            this.id_client = id_client;
            if (status_client == 3)
            {
                btnAdd.Visible = false;
                btnDelete.Visible = false;
                btnEdit.Visible = false;
                btnSuppliers.Visible = false;
                btnUsers.Visible = false;
                btnOrders.Visible = false;
                btnQuery1.Visible = false;
                btnQuery3.Visible = false;
                button1.Visible = false;
                button2.Visible = false;
                button3.Visible = false;
                button4.Visible = false;
            }
            if (status_client == 2)
            {
                pictureBox1.Visible = false;
                btnSuppliers.Visible = false;
                btnUsers.Visible = false;
                btnOrder.Visible = false;
            }
        }
    }
}
```

```

    }
    if (status_client == 1) // adm
    {
        pictureBox1.Visible = false;
        btnOrder.Visible = false;
    }

    }

    private int selectID = 0;
    private void btnProducts_Click(object sender, EventArgs e)
    {
        btnProducts.Enabled = false;
        btnOrders.Enabled = true;

        btnUsers.Enabled = true;
        btnSuppliers.Enabled = true;
        showTableProducts();
    }

    private void btnUsers_Click(object sender, EventArgs e)
    {
        btnProducts.Enabled = true;
        btnOrders.Enabled = true;
        btnUsers.Enabled = false;
        btnSuppliers.Enabled = true;
        showTableUsers();
    }

    private void btnOrders_Click(object sender, EventArgs e)
    {
        btnProducts.Enabled = true;
        btnOrders.Enabled = false;
        btnUsers.Enabled = true;
        btnSuppliers.Enabled = true;
        showTableOrders();
    }

    private void btnSuppliers_Click(object sender, EventArgs e)
    {
        btnProducts.Enabled = true;
        btnOrders.Enabled = true;
        btnUsers.Enabled = true;
        btnSuppliers.Enabled = false;
        showTableSuppliers();
    }

    private void btnAdd_Click(object sender, EventArgs e)
    {
        if (btnSuppliers.Enabled == false)
        {

```

```

AddSuppliersForm addSuppliersForm = new AddSuppliersForm(this, 0);

addSuppliersForm.FormClosed += (s, args) => Application.Exit();
addSuppliersForm.Show();
return;
}
if (btnProducts.Enabled == false)
{
AddProductsForm addProductsForm = new AddProductsForm(this, 0);

addProductsForm.FormClosed += (s, args) => Application.Exit();
addProductsForm.Show();
return;
}
if (btnUsers.Enabled == false)
{
AddUsersForm addUsersForm = new AddUsersForm(this, 0);

addUsersForm.FormClosed += (s, args) => Application.Exit();
addUsersForm.Show();
return;
}
if (btnOrders.Enabled == false)
{
AddOrdersForm addOrdersForm = new AddOrdersForm(this, 0);

addOrdersForm.FormClosed += (s, args) => Application.Exit();
addOrdersForm.Show();
return;
}

MessageBox.Show("Добавление данных");
}

private void btnEdit_Click(object sender, EventArgs e)
{
if (btnUsers.Enabled == false)
{
if (dataGridView.SelectedRows.Count > 0)
{
DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
int id_user = (int)selectedRow.Cells["id_клиента"].Value;
AddUsersForm addUsersForm = new AddUsersForm(this, id_user);

addUsersForm.FormClosed += (s, args) => Application.Exit();
addUsersForm.Show();
return;
}
else
{

```

```

MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
return;
}
}

if (btnSuppliers.Enabled == false)
{
if (dataGridView.SelectedRows.Count > 0)
{
DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
int id_supplier = (int)selectedRow.Cells["id_поставщика"].Value;
AddSuppliersForm addSuppliersForm = new AddSuppliersForm(this, id_supplier);

addSuppliersForm.FormClosed += (s, args) => Application.Exit();
addSuppliersForm.Show();
return;
}
else
{
MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
return;
}

}

if (btnProducts.Enabled == false)
{
if (dataGridView.SelectedRows.Count > 0)
{
DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
int id_product = (int)selectedRow.Cells["ID Товара"].Value;
AddProductsForm addProductsForm = new AddProductsForm(this, id_product);

addProductsForm.FormClosed += (s, args) => Application.Exit();
addProductsForm.Show();
return;
}
else
{
MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
return;
}
}

if (btnOrders.Enabled == false)
{
if (dataGridView.SelectedRows.Count > 0)
{
DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
int id_orders = (int)selectedRow.Cells["ID Заказа"].Value;
AddOrdersForm addOrdersForm = new AddOrdersForm(this, id_orders);

addOrdersForm.FormClosed += (s, args) => Application.Exit();

```

```

addOrdersForm.Show();
return;
}
else
{
    MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
    return;
}
}
MessageBox.Show("Редактирование данных");
}

private void btnDelete_Click(object sender, EventArgs e)
{
    if (btnSuppliers.Enabled == false)
    {
        if (dataGridView.SelectedRows.Count > 0)
        {
            var results = MessageBox.Show("Вы уверены, что хотите удалить выбранного поставщика?", "Подтверждение удаления",
MessageBoxButtons.YesNo, MessageBoxIcon.Question);
            if (results == DialogResult.Yes)
            {
                DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
                int id_suppler = (int)selectedRow.Cells["id_поставщика"].Value;

                connection.Open();

                string query = "DELETE FROM Поставщики WHERE id_поставщика = @id_sup";

                using (SqlCommand command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@id_sup", id_suppler);
                    int result;
                    try
                    {
                        result = command.ExecuteNonQuery();
                    }
                    catch (Exception m)
                    {
                        connection.Close();
                        MessageBox.Show(m.Message);
                        return;
                    }
                }

                if (result > 0)
                {
                    MessageBox.Show("Успешное удаление.");
                    connection.Close();
                    showTableSuppliers();
                }
                else

```

```

    {
        connection.Close();
        MessageBox.Show("Не удалось удалить.");
    }
}

}

}

else
{
    MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
}

return;
}

if (btnOrders.Enabled == false)
{
    if (dataGridView.SelectedRows.Count > 0)
    {
        var results = MessageBox.Show("Вы уверены, что хотите удалить выбранный заказ?", "Подтверждение удаления",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (results == DialogResult.Yes)
        {
            DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
            int id_order = (int)selectedRow.Cells["id_заказа"].Value;

            connection.Open();

            string query = "DELETE FROM Заказы WHERE id_заказа = @id_ord";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@id_ord", id_order);
                int result;
                try
                {
                    result = command.ExecuteNonQuery();
                }
                catch (Exception m)
                {
                    connection.Close();
                    MessageBox.Show(m.Message);
                    return;
                }
            }

            if (result > 0)
            {
                MessageBox.Show("Успешное удаление.");
            }
        }
    }
}

```

```

        connection.Close();
        showTableOrders();
    }
    else
    {
        connection.Close();
        MessageBox.Show("Не удалось удалить.");
    }
}

}

}
else
{
    MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
}
return;
}

if (btnUsers.Enabled == false)
{
    if (dataGridView.SelectedRows.Count > 0)
    {
        var results = MessageBox.Show("Вы уверены, что хотите удалить выбранного пользователя?", "Подтверждение удаления",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (results == DialogResult.Yes)
        {
            DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
            int id_user = (int)selectedRow.Cells["id_клиента"].Value;

            connection.Open();

            string query = "DELETE FROM Клиенты WHERE id_клиента = @id_usr";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@id_usr", id_user);
                int result;
                // Выполнение команды
                try
                {
                    result = command.ExecuteNonQuery();
                }
                catch (Exception m)
                {
                    connection.Close();
                    MessageBox.Show(m.Message);
                }
                return;
            }

```



```

        if (result > 0)
        {
            MessageBox.Show("Успешное удаление.");
            connection.Close();
            showTableUsers();
        }
        else
        {
            connection.Close();
            MessageBox.Show("Не удалось удалить.");
        }
    }

}

else
{
    MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
}

return;
}

if (btnProducts.Enabled == false)
{
    if (dataGridView.SelectedRows.Count > 0)
    {
        var results = MessageBox.Show("Вы уверены, что хотите удалить выбранный товар", "Подтверждение удаления",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (results == DialogResult.Yes)
        {
            DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
            int id_product = (int)selectedRow.Cells["id_товара"].Value;

            connection.Open();

            string query = "DELETE FROM Товары WHERE id_товара = @id_pdr";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@id_pdr", id_product);
                int result;

                try
                {
                    result = command.ExecuteNonQuery();
                }
                catch (Exception m)
                {
                    connection.Close();

```

```

        MessageBox.Show(m.Message);
        return;
    }

    if (result > 0)
    {
        MessageBox.Show("Успешное удаление.");
        connection.Close();
        showTableProducts();
    }
    else
    {
        connection.Close();
        MessageBox.Show("Не удалось удалить.");
    }
}

}

else
{
    MessageBox.Show("Ни одна строка не выбрана! Необходимо выбрать всю строку.");
}
return;
}
MessageBox.Show("Удаление данных");
}

private void btnOrder_Click(object sender, EventArgs e)
{
    if (dataGridView.SelectedRows.Count > 0)
    {
        var results = MessageBox.Show("Вы уверены, что хотите заказать выбранный товар?", "Подтверждение удаления",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question);
        if (results == DialogResult.Yes)
        {
            DataGridViewRow selectedRow = dataGridView.SelectedRows[0];
            int id_product = (int)selectedRow.Cells["ID Товара"].Value;

            int countProduct = 0;
            try
            {
                connection.Open();
                string query2 = $"SELECT КоличествоНаСкладе FROM Товары WHERE id_товара = {id_product}";
                using (SqlCommand command = new SqlCommand(query2, connection))
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {

```

```

countProduct = reader.GetInt32(0);
}
}
connection.Close();
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка: " + ex.Message);
}

if (countProduct <= 0)
{
    MessageBox.Show("Данного товара нет в наличии.");
    return;
}

decimal sum = (decimal)selectedRow.Cells["Цена"].Value;
string query = "INSERT INTO Заказы (id_клиента, ДатаЗаказа, ОбщаяСумма, Статус, id_товара) VALUES (@User, @Date, @Sum,
'active', @product)";
using (SqlCommand command = new SqlCommand(query, connection))
{
    try
    {
        connection.Open();

        command.Parameters.AddWithValue("@Date", DateTime.Now);
        command.Parameters.AddWithValue("@Sum", sum);
        command.Parameters.AddWithValue("@User", id_client);
        command.Parameters.AddWithValue("@product", id_product);

        int rowsAffected = command.ExecuteNonQuery();
        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}

string queryalter = $"UPDATE Товары SET КоличествоНаСкладе = КоличествоНаСкладе - 1 WHERE id_товара = {id_product}";
using (SqlCommand command = new SqlCommand(queryalter, connection))
{
    try
    {
        connection.Open();

        int rowsAffected = command.ExecuteNonQuery();

        connection.Close();
        showTableProducts();
    }
    catch (Exception ex)

```

```

    {
        MessageBox.Show("Ошибка: " + ex.Message);
        connection.Close();
    }

}

}

}

}

private void MainForm_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void MainForm_Load(object sender, EventArgs e)
{
    showTableProducts();
}

public void showTableProducts()
{
    connection.Open();

    string query = "SELECT \r\n    Товары.id_товара AS 'ID Товара',\r\n    Товары.Название AS 'Название Товара',\r\n    Товары.Бренд
AS 'Бренд',\r\n    Товары.Категория AS 'Категория',\r\n    Товары.Цена AS 'Цена',\r\n    Товары.КоличествоНаСкладе AS 'Количество На
Складе',\r\n    Поставщики.Название AS 'Название Поставщика'\r\nFROM \r\n    Товары\r\nLEFT JOIN \r\n    Поставщики ON
Товары.id_поставщика = Поставщики.id_поставщика;";

    using (SqlCommand command = new SqlCommand(query, connection))
    {
        DataTable dataTable = new DataTable();

        using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
        {
            dataAdapter.Fill(dataTable);
        }

        dataGridView.DataSource = dataTable;
    }
    connection.Close();
}

public void showTableSuppliers()
{
    connection.Open();

    string query = "SELECT * FROM Поставщики";

    using (SqlCommand command = new SqlCommand(query, connection))

```

```

{
    DataTable dataTable = new DataTable();

    using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
    {
        dataAdapter.Fill(dataTable);
    }

    dataGridView.DataSource = dataTable;
}
connection.Close();
}

public void showTableOrders()
{
    connection.Open();

    string query = "SELECT \r\n  Заказы.id_заказа AS 'ID Заказа',\r\n  Клиенты.Имя + ' ' + Клиенты.Фамилия AS 'Имя Клиента',\r\n
Товары.Название AS 'Название Товара',\r\n  Заказы.ДатаЗаказа AS 'Дата Заказа',\r\n  Заказы.ОбщаяСумма AS 'Общая Сумма',\r\n
Заказы.Статус AS 'Статус Заказа'\r\nFROM \r\n  Заказы\r\nJOIN \r\n  Клиенты ON Заказы.id_клиента = Клиенты.id_клиента\r\nJOIN \r\n
Товары ON Заказы.id_товара = Товары.id_товара;";

    using (SqlCommand command = new SqlCommand(query, connection))
    {
        DataTable dataTable = new DataTable();

        using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
        {
            dataAdapter.Fill(dataTable);
        }

        dataGridView.DataSource = dataTable;
    }
    connection.Close();
}

public void showTableUsers()
{
    connection.Open();

    string query = "SELECT * FROM Клиенты";

    using (SqlCommand command = new SqlCommand(query, connection))
    {
        DataTable dataTable = new DataTable();

        using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
        {
            dataAdapter.Fill(dataTable);
        }
    }
}

```

```

dataGridView.DataSource = dataTable;
}
connection.Close();
}

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

private void dataGridView_CellClick(object sender, DataGridViewCellEventArgs e)
{
}

private void btnQuery1_Click(object sender, EventArgs e)
{
connection.Open();

string query = "SELECT Название, КоличествоНаСкладе FROM Товары WHERE КоличествоНаСкладе < 10;";

using (SqlCommand command = new SqlCommand(query, connection))
{
DataTable dataTable = new DataTable();

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
{
dataAdapter.Fill(dataTable);
}

dataGridView.DataSource = dataTable;
}
connection.Close();
}

private void button1_Click_1(object sender, EventArgs e)
{
connection.Open();

string query = "SELECT Клиенты.Имя, Клиенты.Фамилия, Заказы.ДатаЗаказа FROM Клиенты JOIN Заказы ON
Клиенты.id_клиента = Заказы.id_клиента WHERE Заказы.ДатаЗаказа BETWEEN DATEADD(month, -1, GETDATE()) AND GETDATE();";

using (SqlCommand command = new SqlCommand(query, connection))
{
DataTable dataTable = new DataTable();

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
{
dataAdapter.Fill(dataTable);
}
}
}

```

```

dataGridView.DataSource = dataTable;
}
connection.Close();
}

private void btnQuery3_Click(object sender, EventArgs e)
{
connection.Open();

string query = "SELECT TOP 1 id_заказа, ОбщаяСумма FROM Заказы ORDER BY ОбщаяСумма DESC;";

using (SqlCommand command = new SqlCommand(query, connection))
{
DataTable dataTable = new DataTable();

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
{
dataAdapter.Fill(dataTable);
}

dataGridView.DataSource = dataTable;
}
connection.Close();
}

private void button2_Click(object sender, EventArgs e)
{
connection.Open();

string query = "SELECT id_заказа, id_клиента, ДатаЗаказа, ОбщаяСумма FROM Заказы WHERE Статус = 'active'";

using (SqlCommand command = new SqlCommand(query, connection))
{
DataTable dataTable = new DataTable();

using (SqlDataAdapter dataAdapter = new SqlDataAdapter(command))
{
dataAdapter.Fill(dataTable);
}

dataGridView.DataSource = dataTable;
}
connection.Close();
}

private void button3_Click(object sender, EventArgs e)
{
string queryalter = "UPDATE Товары SET Цена = Цена + 0.5;";
using (SqlCommand command = new SqlCommand(queryalter, connection))
{

```

```

try
{
connection.Open();

int rowsAffected = command.ExecuteNonQuery();

MessageBox.Show("Обновление цен успешно выполнено!");
connection.Close();
showTableProducts();
}
catch (Exception ex)
{
MessageBox.Show("Ошибка: " + ex.Message);
connection.Close();
}

}
}

private void button4_Click(object sender, EventArgs e)
{
string queryalter = "UPDATE Товары SET Цена = Цена * 0.9;";
using (SqlCommand command = new SqlCommand(queryalter, connection))
{
try
{
// Открытие подключения
connection.Open();

// Выполнение запроса
int rowsAffected = command.ExecuteNonQuery();

MessageBox.Show("Обновление цен успешно выполнено!");
connection.Close();
showTableProducts();
}
catch (Exception ex)
{
MessageBox.Show("Ошибка: " + ex.Message);
connection.Close();
}
}
}

private void pictureBox1_Click(object sender, EventArgs e)
{

}

private void dataGridView1_CellContentClick(object sender, DataGridViewCellEventArgs e)
{

```



```

}
}
}

```

Листинг кода окна регистрации (RegistrationForm):

```

using static System.Windows.Forms.VisualStyles.VisualStyleElement;
using System.Data.SqlClient;

namespace Estore
{
    public partial class RegistrationForm : Form
    {
        private SqlConnection connection = new SqlConnection(@"Data Source=(LocalDb)\LocalDBDemo;Initial
Catalog=Estore;Integrated Security=True;");
        public RegistrationForm()
        {
            InitializeComponent();
            this.FormClosed += RegistrationForm_FormClosed;
        }
        private void RegisterButton_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(nameTextBox.Text))
            {
                MessageBox.Show("Введите имя", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (string.IsNullOrEmpty(surnameTextBox.Text))
            {
                MessageBox.Show("Введите фамилию", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (string.IsNullOrEmpty(emailTextBox.Text))
            {
                MessageBox.Show("Введите почту", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (string.IsNullOrEmpty(phoneTextBox.Text))
            {
                MessageBox.Show("Введите телефон", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (string.IsNullOrEmpty(addressTextBox.Text))
            {
                MessageBox.Show("Введите адрес", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            if (string.IsNullOrEmpty(usernameTextBox.Text))
            {
                MessageBox.Show("Введите логин", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

```

```

    }
    if (string.IsNullOrEmpty(passwordTextBox.Text))
    {
        MessageBox.Show("Введите пароль", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    string name = nameTextBox.Text;
    string surname = surnameTextBox.Text;
    string mail = emailTextBox.Text;
    string phone = phoneTextBox.Text;
    string address = addressTextBox.Text;
    string login = usernameTextBox.Text;
    string password = passwordTextBox.Text;
    string query = "INSERT INTO Клиенты (Имя, Фамилия, ЭлектроннаяПочта, Телефон, Адрес, Логин, Пароль, Статус)
VALUES (@Name, @Surname, @Mail, @Phone, @Address, @Login, @Password, 'client');";
    using (SqlCommand command = new SqlCommand(query, connection))
    {
        try
        {
            connection.Open();

            command.Parameters.AddWithValue("@Name", name);
            command.Parameters.AddWithValue("@Surname", surname);
            command.Parameters.AddWithValue("@Mail", mail);
            command.Parameters.AddWithValue("@Phone", phone);
            command.Parameters.AddWithValue("@Address", address);
            command.Parameters.AddWithValue("@Login", login);
            command.Parameters.AddWithValue("@Password", password);

            int rowsAffected = command.ExecuteNonQuery();
            MessageBox.Show("Вы успешно зарегистрировались!");
            connection.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка: " + ex.Message);
            connection.Close();
            return;
        }
    }

    LoginForm loginForm = new LoginForm();
    this.Hide();
    loginForm.Show();
}

private void RegistrationForm_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

```

```

private void nameTextBox_TextChanged(object sender, EventArgs e)
{

}

private void RegistrationForm_Load(object sender, EventArgs e)
{

}

private void label4_Click(object sender, EventArgs e)
{

}
}
}

```

Листинг кода окна авторизации (LoginForm):

```

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

namespace Estore
{
    public partial class LoginForm : Form
    {
        private string connectionStr = @"Data Source=(LocalDb)\LocalDBDemo;Initial Catalog=Estore;Integrated Security=True;";
        public LoginForm()
        {
            InitializeComponent();
            this.FormClosed += LoginForm_FormClosed;
        }

        private void LoginButton_Click(object sender, EventArgs e)
        {
            string login = usernameTextBox.Text;
            string password = passwordTextBox.Text;

            if (ValidateUser(login, password, out int userId, out int userStatus))
            {

                MainForm mainForm = new MainForm(userId, userStatus);
            }
        }
    }
}

```

```

        this.Hide();

        mainForm.FormClosed += (s, args) => Application.Exit();
        mainForm.Show();
    }
    else
    {
        MessageBox.Show("Неверный логин или пароль.");
    }
}

private void LoginForm_FormClosed(object sender, FormClosedEventArgs e)
{
    Application.Exit();
}

private void RegisterButton_Click(object sender, EventArgs e)
{
    RegistrationForm registrationForm = new RegistrationForm();

    this.Hide();

    registrationForm.FormClosed += (s, args) => Application.Exit();
    registrationForm.Show();
}

private bool ValidateUser(string login, string password, out int userId, out int userStatus)
{
    userId = 0;
    userStatus = 0;

    using (SqlConnection connection = new SqlConnection(connectionStr))
    {
        try
        {
            connection.Open();
            string query = "SELECT id_клиента, Пароль, Статус FROM Клиенты WHERE Логин = @login";

            using (SqlCommand command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@login", login);

                using (SqlDataReader reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        int id = reader.GetInt32(0);
                        string storedPassword = reader.GetString(1);
                        string status = reader.GetString(2);

```

```

        if (password == storedPassword)
        {
            userId = id;
            userStatus = status switch
            {
                "adm" => 1,
                "pers" => 2,
                "client" => 3,
                _ => 0
            };
            return true;
        }
    }
}

catch (Exception ex)
{
    MessageBox.Show("Ошибка при подключении к базе данных: " + ex.Message);
}

return false;
}

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

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

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

```

Листинг кода окна добавления пользования (AddUsersForm):

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Drawing;

```

```

using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static Estore.AddProductsForm;

namespace Estore
{
    public partial class AddUsersForm : Form
    {
        SqlConnection connection = new SqlConnection(@"Data Source=(LocalDb)\LocalDBDemo;Initial Catalog=Estore;Integrated
Security=True;");

        private MainForm mainForm;
        private int id_redact = 0;
        public AddUsersForm(MainForm mForm, int id)
        {
            InitializeComponent();
            comboStatus.SelectedIndex = 0;
            mainForm = mForm;
            id_redact = id;
            if (id != 0)
            {
                btnAdd.Text = "Изменить";
                this.Text = $"Редактирование польз. #{id}";
                try
                {
                    connection.Open();
                    string query = "SELECT Имя, Фамилия, ЭлектроннаяПочта, Телефон, Адрес, Логин, Пароль FROM Клиенты WHERE
id_клиента = " + id;

                    using (SqlCommand command = new SqlCommand(query, connection))
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            string userName = reader.GetString(0);
                            string userSurname = reader.GetString(1);
                            string userEmail = reader.GetString(2);
                            string userPhone = reader.GetString(3);
                            string userAddress = reader.GetString(4);
                            string userLogin = reader.GetString(5);
                            string userPassword = reader.GetString(6);
                            textName.Text = userName;
                            textSurname.Text = userSurname;
                            textMail.Text = userEmail;
                            textPhone.Text = userPhone;
                            textAddress.Text = userAddress;
                            textLogin.Text = userLogin;
                            textPassword.Text = userPassword;
                        }
                    }
                }
            }
        }
    }
}

```

```

        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при загрузке страницы: " + ex.Message);
    }
}

private void textAddress_KeyPress(object sender, KeyPressEventArgs e)
{
}

private void AddUsersForm_FormClosing(object sender, FormClosingEventArgs e)
{
    if (e.CloseReason == CloseReason.UserClosing)
    {
        e.Cancel = true;
        this.Hide();
    }
}

private void btnAdd_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textName.Text))
    {
        MessageBox.Show("Введите имя", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (string.IsNullOrEmpty(textSurname.Text))
    {
        MessageBox.Show("Введите фамилию", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (string.IsNullOrEmpty(textMail.Text))
    {
        MessageBox.Show("Введите почту", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (string.IsNullOrEmpty(textPhone.Text))
    {
        MessageBox.Show("Введите телефон", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
    if (string.IsNullOrEmpty(textAddress.Text))
    {
        MessageBox.Show("Введите адрес", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
}

```

```

if (string.IsNullOrEmpty(textLogin.Text))
{
    MessageBox.Show("Введите логин", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}
if (string.IsNullOrEmpty(textPassword.Text))
{
    MessageBox.Show("Введите пароль", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    return;
}
string name = textName.Text;
string surname = textSurname.Text;
string mail = textMail.Text;
string phone = textPhone.Text;
string address = textAddress.Text;
string login = textLogin.Text;
string password = textPassword.Text;
string status = comboStatus.Text;
if (id_redact != 0)
{
    string queryalter = @"
UPDATE Клиенты
SET Имя = @Имя,
    Фамилия = @Фамилия,
    ЭлектроннаяПочта = @ЭлектроннаяПочта,
    Телефон = @Телефон,
    Адрес = @Адрес,
    Логин = @Логин,
    Пароль = @Пароль,
    Статус = @Статус
WHERE id_клиента = @id_клиента";
    using (SqlCommand command = new SqlCommand(queryalter, connection))
    {
        try
        {
            connection.Open();
            command.Parameters.AddWithValue("@Имя", name);
            command.Parameters.AddWithValue("@Фамилия", surname);
            command.Parameters.AddWithValue("@ЭлектроннаяПочта", mail);
            command.Parameters.AddWithValue("@Телефон", phone);
            command.Parameters.AddWithValue("@Адрес", address);
            command.Parameters.AddWithValue("@Логин", login);
            command.Parameters.AddWithValue("@Пароль", password);
            command.Parameters.AddWithValue("@Статус", status);
            command.Parameters.AddWithValue("@id_клиента", id_redact);

            int rowsAffected = command.ExecuteNonQuery();

            MessageBox.Show($"Изменено {rowsAffected} строк в таблице Клиенты.");
            connection.Close();
        }
    }
}

```



```

        catch (Exception ex)
        {
            MessageBox.Show("Ошибка: " + ex.Message);
        }
    }
    mainForm.showTableUsers();
    Close();
    return;
}

string query = "INSERT INTO Клиенты (Имя, Фамилия, ЭлектроннаяПочта, Телефон, Адрес, Логин, Пароль, Статус)
VALUES (@Name, @Surname, @Mail, @Phone, @Address, @Login, @Password, @Status);";
using (SqlCommand command = new SqlCommand(query, connection))
{
    try
    {
        connection.Open();

        command.Parameters.AddWithValue("@Name", name);
        command.Parameters.AddWithValue("@Surname", surname);
        command.Parameters.AddWithValue("@Mail", mail);
        command.Parameters.AddWithValue("@Phone", phone);
        command.Parameters.AddWithValue("@Address", address);
        command.Parameters.AddWithValue("@Login", login);
        command.Parameters.AddWithValue("@Password", password);
        command.Parameters.AddWithValue("@Status", status);

        int rowsAffected = command.ExecuteNonQuery();

        MessageBox.Show($"Добавлено {rowsAffected} строк в таблицу Товары.");
        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}
mainForm.showTableUsers();
Close();

}

private void textPhone_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}

private void textName_TextChanged(object sender, EventArgs e)

```

```

    {

    }

    private void textAddress_TextChanged(object sender, EventArgs e)
    {

    }

    }
}

```

Листинг кода окна добавления поставщика (AddSuppliersForm):

```

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

namespace Estore
{
    public partial class AddSuppliersForm : Form
    {
        private int id_redact = 0;
        private MainForm mainForm;
        public AddSuppliersForm(MainForm mForm, int id)
        {
            InitializeComponent();
            id_redact = id;
            mainForm = mForm;
            if (id != 0)
            {
                btnAdd.Text = "Изменить";
                this.Text = $"Редактирование поставщика #{id}";
                try
                {
                    connection.Open();
                    string query = "SELECT Название, Адрес, Телефон FROM Поставщики WHERE id_поставщика = " + id;
                    using (SqlCommand command = new SqlCommand(query, connection))
                    using (SqlDataReader reader = command.ExecuteReader())
                    {
                        while (reader.Read())
                        {
                            string supplierName = reader.GetString(0);
                            string supplierAddress = reader.GetString(1);
                            string supplierPhone = reader.GetString(2);

```

```

        textName.Text = supplierName;
        textAddress.Text = supplierAddress;
        textPhone.Text = supplierPhone;
    }
}
connection.Close();
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка при загрузке поставщика: " + ex.Message);
}
}
}
SqlConnection connection = new SqlConnection(@"Data Source=(LocalDb)\LocalDBDemo;Initial Catalog=Estore;Integrated
Security=True;");
private void label3_Click(object sender, EventArgs e)
{
}

private void AddSuppliersForm_FormClosing(object sender, FormClosingEventArgs e)
{
    if (e.CloseReason == CloseReason.UserClosing)
    {
        e.Cancel = true;
        this.Hide();
    }
}

private void btnAdd_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textName.Text))
    {
        MessageBox.Show("Введите имя", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (string.IsNullOrEmpty(textAddress.Text))
    {
        MessageBox.Show("Введите адрес", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    // Проверяем наличие данных в поле textPhone
    if (string.IsNullOrEmpty(textPhone.Text))
    {
        MessageBox.Show("Введите номер телефона", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    string name = textName.Text;
    string address = textAddress.Text;

```

```

string phone = textPhone.Text;
if (id_redact != 0)
{
    string queryalter = @"
UPDATE Поставщики
SET Название = @Название,
    Адрес = @Адрес,
    Телефон = @Телефон
WHERE id_поставщика = @id_поставщика";

    using (SqlCommand command = new SqlCommand(queryalter, connection))
    {
        try
        {
            connection.Open();

            command.Parameters.AddWithValue("@Название", name);
            command.Parameters.AddWithValue("@Адрес", address);
            command.Parameters.AddWithValue("@Телефон", phone);
            command.Parameters.AddWithValue("@id_поставщика", id_redact);

            int rowsAffected = command.ExecuteNonQuery();

            MessageBox.Show($"Изменено {rowsAffected} строк в таблице Поставщики.");
            connection.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка: " + ex.Message);
        }
    }
    mainForm.showTableSuppliers();

    Close();
    return;
}
string query = "INSERT INTO Поставщики (Название, Адрес, Телефон) VALUES (@Name, @Address, @Phone)";

using (SqlCommand command = new SqlCommand(query, connection))
{
    try
    {
        connection.Open();

        command.Parameters.AddWithValue("@Name", name);
        command.Parameters.AddWithValue("@Address", address);
        command.Parameters.AddWithValue("@Phone", phone);

        int rowsAffected = command.ExecuteNonQuery();

        MessageBox.Show($"Добавлено {rowsAffected} строк в таблицу Поставщики.");
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}

```

```

        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}
mainForm.showTableSuppliers();
Close();
}

private void textPhone_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
    {
        e.Handled = true;
    }
}

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

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

```

Листинг кода окна добавления товара (AddProductsForm):

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Estore
{
    public partial class AddProductsForm : Form
    {
        private int id_redact = 0;
        SqlConnection connection = new SqlConnection(@"Data Source=(LocalDb)\LocalDBDemo;Initial Catalog=Estore;Integrated

```

```

Security=True;");

private MainForm mainForm;

public AddProductsForm(MainForm mForm, int id)
{
    InitializeComponent();
    LoadSuppliers();
    mainForm = mForm;
    comboSupp.SelectedIndex = 0;
    id_redact = id;
    if (id != 0)
    {
        btnAdd.Text = "Изменить";
        this.Text = $"Редактирование товара #{id}";
        try
        {

            connection.Open();
            string query = "SELECT Название, Бренд, Категория, Цена, КоличествоНаСкладе FROM Товары WHERE id_товара
= " + id;

            using (SqlCommand command = new SqlCommand(query, connection))
            using (SqlDataReader reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    string productName = reader.GetString(0);
                    string productBrend = reader.GetString(1);
                    string productCategory = reader.GetString(2);
                    decimal productPrice = reader.GetDecimal(3);
                    int productCount = reader.GetInt32(4);
                    textName.Text = productName;
                    textBrend.Text = productBrend;
                    textCategory.Text = productCategory;
                    textPrice.Text = Convert.ToString(productPrice);
                    textCount.Text = Convert.ToString(productCount);

                }
            }
            connection.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show("Ошибка при загрузке страницы: " + ex.Message);
        }
    }
}

private void LoadSuppliers()
{
    comboSupp.Items.Clear();

```

```

try
{

    connection.Open();
    string query = "SELECT id_поставщика, Название FROM Поставщики";
    using (SqlCommand command = new SqlCommand(query, connection))
    using (SqlDataReader reader = command.ExecuteReader())
    {
        while (reader.Read())
        {
            int supplierId = reader.GetInt32(0);
            string supplierName = reader.GetString(1);
            comboSupp.Items.Add(new ComboBoxItem(supplierName, supplierId));
        }
    }
    connection.Close();
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка при загрузке поставщиков: " + ex.Message);
}
}

private void btnAdd_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textName.Text))
    {
        MessageBox.Show("Введите имя", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (string.IsNullOrEmpty(textBrend.Text))
    {
        MessageBox.Show("Введите бренд", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (string.IsNullOrEmpty(textCategory.Text))
    {
        MessageBox.Show("Введите категорию", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (string.IsNullOrEmpty(textPrice.Text))
    {
        MessageBox.Show("Введите цену", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }

    if (string.IsNullOrEmpty(textCount.Text))
    {
        MessageBox.Show("Введите количество", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return;
    }
}

```

```

    }
    string name = textName.Text;
    string brend = textBrend.Text;
    string category = textCategory.Text;
    double price = Convert.ToDouble(textPrice.Text);
    int count = Convert.ToInt32(textCount.Text);
    if (id_redact != 0)
    {
        string queryalter = @"
UPDATE Товары
SET Название = @Название,
    Бренд = @Бренд,
    Категория = @Категория,
    Цена = @Цена,
    КоличествоНаСкладе = @КоличествоНаСкладе,
    id_поставщика = @id_поставщика
WHERE id_товара = @id_товара";
        using (SqlCommand command = new SqlCommand(queryalter, connection))
        {
            try
            {
                connection.Open();

                command.Parameters.AddWithValue("@Название", name);
                command.Parameters.AddWithValue("@Бренд", brend);
                command.Parameters.AddWithValue("@Категория", category);
                command.Parameters.AddWithValue("@Цена", price);
                command.Parameters.AddWithValue("@КоличествоНаСкладе", count);
                ComboBoxItem selectedItem = (ComboBoxItem)comboSupp.SelectedItem;
                int selectedSupplierId = selectedItem.Value;
                command.Parameters.AddWithValue("@id_поставщика", selectedSupplierId);
                command.Parameters.AddWithValue("@id_товара", id_redact);

                int rowsAffected = command.ExecuteNonQuery();

                MessageBox.Show($"Изменено {rowsAffected} строк в таблице Товары.");
                connection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка: " + ex.Message);
            }
        }
        mainForm.showTableProducts();
        Close();
        return;
    }
}

```

```

string query = "INSERT INTO Товары (Название, Бренд, Категория, Цена, КоличествоНаСкладе, id_поставщика) VALUES

```



```

(@Name, @Brend, @Category, @Price, @Count, @Supp);"
using (SqlCommand command = new SqlCommand(query, connection))
{
    try
    {
        connection.Open();

        command.Parameters.AddWithValue("@Name", name);
        command.Parameters.AddWithValue("@Brend", brend);
        command.Parameters.AddWithValue("@Category", category);
        command.Parameters.AddWithValue("@Price", price);
        command.Parameters.AddWithValue("@Count", count);
        ComboBoxItem selectedItem = (ComboBoxItem)comboSupp.SelectedItem;
        int selectedSupplierId = selectedItem.Value;
        command.Parameters.AddWithValue("@Supp", selectedSupplierId);

        // Выполнение запроса
        int rowsAffected = command.ExecuteNonQuery();

        MessageBox.Show($"Добавлено {rowsAffected} строк в таблицу Товары.");
        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка: " + ex.Message);
    }
}
mainForm.showTableProducts();
Close();
}

public class ComboBoxItem
{
    public string Text { get; set; }
    public int Value { get; set; }

    public ComboBoxItem(string text, int value)
    {
        Text = text;
        Value = value;
    }

    public override string ToString()
    {
        return Text;
    }
}

private void AddProductsForm_FormClosing(object sender, FormClosingEventArgs e)
{
    if (e.CloseReason == CloseReason.UserClosing)

```

```

        {
            e.Cancel = true;
            this.Hide();
        }
    }

    private void textCount_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsControl(e.KeyChar) && !char.IsDigit(e.KeyChar))
        {
            e.Handled = true;
        }
    }

    private void textPrice_KeyPress(object sender, KeyPressEventArgs e)
    {
        if (!char.IsDigit(e.KeyChar) && e.KeyChar != ',' && e.KeyChar != '\b')
        {
            e.Handled = true;
        }
        else if (e.KeyChar == ',' && (sender as System.Windows.Forms.TextBox).Text.Contains(","))
        {
            e.Handled = true;
        }
    }

    private void AddProductsForm_Load(object sender, EventArgs e)
    {
        // ...
    }
}

```

Листинг кода окна добавления заказа (AddOrdersForm):

```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Diagnostics;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;
using static Estore.AddProductsForm;

namespace Estore
{
    public partial class AddOrdersForm : Form
    {
        // ...
    }
}

```

```

{
    SqlConnection connection = new SqlConnection(@"Data Source=(LocalDb)\LocalDBDemo;Initial Catalog=Estore;Integrated
Security=True;");
    private MainForm mainForm;
    private int id_redact = 0;
    public AddOrdersForm(MainForm mForm, int id)
    {
        InitializeComponent();
        mainForm = mForm;
        LoadProduct();
        comboProduct.SelectedIndex = 0;
        LoadUsers();
        comboStatus.SelectedIndex = 0;
        comboUsers.SelectedIndex = 0;
        id_redact = id;
        if (id != 0)
        {
            btnAdd.Text = "Изменить";
            this.Text = $"Редактирование заказа #{id}";
            try
            {
                connection.Open();
                string query = "SELECT ДатаЗаказа, ОбщаяСумма FROM Заказы WHERE id_заказа = " + id;
                using (SqlCommand command = new SqlCommand(query, connection))
                using (SqlDataReader reader = command.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        DateTime orderdate = reader.GetDateTime(0);
                        decimal orderprice = reader.GetDecimal(1);
                        dateOrder.Value = orderdate;
                        textSum.Text = orderprice.ToString();
                    }
                }
                connection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка при загрузке страницы: " + ex.Message);
            }
        }
    }
    private void LoadUsers()
    {
        comboUsers.Items.Clear();

        try
        {

```

```

connection.Open();
string query = "SELECT id_клиента, Фамилия FROM Клиенты";
using (SqlCommand command = new SqlCommand(query, connection))
using (SqlDataReader reader = command.ExecuteReader())
{
    while (reader.Read())
    {
        int supplierId = reader.GetInt32(0);
        string supplierName = reader.GetString(1);
        comboUsers.Items.Add(new ComboBoxItem(supplierName, supplierId));
    }
}
connection.Close();
}
catch (Exception ex)
{
    MessageBox.Show("Ошибка при загрузке клиентов: " + ex.Message);
}
}

private void LoadProduct()
{
    comboProduct.Items.Clear();

    try
    {
        connection.Open();
        string query = "SELECT id_товара FROM Товары";
        using (SqlCommand command = new SqlCommand(query, connection))
        using (SqlDataReader reader = command.ExecuteReader())
        {
            while (reader.Read())
            {
                int supplierId = reader.GetInt32(0);
                comboProduct.Items.Add(supplierId);
            }
        }
        connection.Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show("Ошибка при загрузке клиентов: " + ex.Message);
    }
}

private void btnAdd_Click(object sender, EventArgs e)
{
    if (string.IsNullOrEmpty(textSum.Text))
    {
        MessageBox.Show("Введите сумму", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

```

```

        return;
    }
    double summ = Convert.ToDouble(textSum.Text);
    DateTime date = dateOrder.Value;
    string status = comboStatus.Text;
    int product = Convert.ToInt32(comboProduct.Text);

    if (id_redact != 0)
    {
        string queryalter = @"
UPDATE Заказы
SET id_клиента = @id_клиента,
    ДатаЗаказа = @ДатаЗаказа,
    ОбщаяСумма = @ОбщаяСумма,
    Статус = @Статус,
    id_товара = @Product
WHERE id_заказа = @id_заказа";
        using (SqlCommand command = new SqlCommand(queryalter, connection))
        {
            try
            {
                connection.Open();

                command.Parameters.AddWithValue("@ДатаЗаказа", date);
                command.Parameters.AddWithValue("@ОбщаяСумма", summ);
                command.Parameters.AddWithValue("@Статус", status);
                command.Parameters.AddWithValue("@Product", product);
                ComboBoxItem selectedItem = (ComboBoxItem)comboUsers.SelectedItem;
                int selectedUserId = selectedItem.Value;
                command.Parameters.AddWithValue("@id_клиента", selectedUserId);
                command.Parameters.AddWithValue("@id_заказа", id_redact);

                int rowsAffected = command.ExecuteNonQuery();

                MessageBox.Show($"Добавлено {rowsAffected} строк в таблицу Заказы.");
                connection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка: " + ex.Message);
            }
        }
        mainForm.showTableOrders();
        Close();
        return;
    }
}

```

```

        string query = "INSERT INTO Заказы (id_клиента, ДатаЗаказа, ОбщаяСумма, Статус, id_товара) VALUES (@User, @Date,
@Sum, @Status, @Product);";
        using (SqlCommand command = new SqlCommand(query, connection))
        {
            try
            {
                connection.Open();

                command.Parameters.AddWithValue("@Date", date);
                command.Parameters.AddWithValue("@Sum", summ);
                command.Parameters.AddWithValue("@Status", status);
                command.Parameters.AddWithValue("@Product", product);
                ComboBoxItem selectedItem = (ComboBoxItem)comboUsers.SelectedItem;
                int selectedUserId = selectedItem.Value;
                command.Parameters.AddWithValue("@User", selectedUserId);

                int rowsAffected = command.ExecuteNonQuery();

                MessageBox.Show($"Добавлено {rowsAffected} строк в таблицу Заказы.");
                connection.Close();
            }
            catch (Exception ex)
            {
                MessageBox.Show("Ошибка: " + ex.Message);
            }
        }
        mainForm.showTableOrders();
        Close();
    }
    public class ComboBoxItem
    {
        public string Text { get; set; }
        public int Value { get; set; }

        public ComboBoxItem(string text, int value)
        {
            Text = text;
            Value = value;
        }

        public override string ToString()
        {
            return Text;
        }
    }

    private void AddOrdersForm_FormClosing(object sender, FormClosingEventArgs e)
    {
        if (e.CloseReason == CloseReason.UserClosing)
        {
            e.Cancel = true;
        }
    }

```

```

        this.Hide();
    }
}

private void textSum_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsDigit(e.KeyChar) && e.KeyChar != ',' && e.KeyChar != '\b')
    {
        e.Handled = true;
    }
    else if (e.KeyChar == ',' && (sender as System.Windows.Forms.TextBox).Text.Contains(","))
    {
        e.Handled = true;
    }
}

private void label4_Click(object sender, EventArgs e)
{
}

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

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

```

Код класса Program:

```

namespace Estore
{
    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 LoginForm());
        }
    }
}

```