**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙФЕДЕРАЦИИ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПБПУ»)

**Институт среднего профессионального образования**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отделение информационных технологий

Специальность: 09.02.07 Информационные системы и программирование

Дисциплина: МДК 11.01 Технологии разработки и защиты баз данных

Группа: 32919/2

**КУРСОВОЙ ПРОЕКТ**

**на тему:**

**«Проектирование и реализация базы данных**

**интернет-магазина продажи электроники»**

Студент Бронин Евгений Александрович

(ФИО) (подпись)

Руководитель Тузова Диана Анатольевна

(ФИО) (подпись)

Оценка

(оценка) (подпись) (ФИО)

Санкт - Петербург

2024

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ**

**РОССИЙСКОЙФЕДЕРАЦИИ**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

федеральное государственное автономное образовательное учреждение

высшего образования

«Санкт-Петербургский политехнический университет Петра Великого»

(ФГАОУ ВО «СПБПУ»)

**Институт среднего профессионального образования**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Отделение информационных технологий

**Задание на курсовое проектирование**

**по МДК 11.01 «Технологии разработки и защиты баз данных»**

Специальность: 09.02.07 Информационные системы и программирование

**студенту** Бронину Е.А. группа\_\_32919/2\_\_\_\_

(фамилия, инициалы)

**Тема:** **Проектирование и реализация базы данных интернет-магазина продажи электроники**

**Исходные данные к проекту:**

1. Литература по описанию объекта информатизации
2. Литература по инструментальным средствам разработки прикладных программ
3. Лекции, стандарты.
4. Интернет – сайты по предметной области

**Перечень подлежащих разработке вопросов:**

1. Аналитический обзор программных средств, технологий, аналогов ПП для создания базы данных и интерфейсов пользователей.
2. Проектирование реляционной базы данных (инфологическая и даталогическая модели с указанием связей (с пояснениями) между таблицами).
3. Разработка механизмов управления данными в БД (добавление, удаление и обновление данных).
4. Организация выборки информации из разработанной базы данных (сформулировать не менее 5 запросов всех типов, реализуемых средствами выбранной СУБД).
5. Проработка технологии доступа к базе данных (определение круга пользователей базы данных и оценка возможности разграничения полномочий пользователей).
6. Разработка ПО для решения поставленной задачи.
7. Разработка алгоритмов программных модулей.
8. Тестирование работы приложения.
9. Оформление документации (руководство системного программиста).
10. Оформление пояснительной записки.

**Руководитель Д.А. Тузова**

(подпись, дата) (инициалы, фамилия)

**Задание принял к выполнению:**

**Студент Бронин Е.А.**

(подпись, дата) (инициалы, фамилия)

**содержание**

[ВВЕДЕНИЕ 3](#_Toc167092892)

[1. Сравнительная характеристика СУБД 4](#_Toc167092893)

[1.1. СУБД PostgreSQL 4](#_Toc167092894)

[1.2. СУБД MongoDB 5](#_Toc167092895)

[1.3. СУБД MS SQL Server 9](#_Toc167092896)

[1.4. Выбор СУБД 10](#_Toc167092897)

[2. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ 11](#_Toc167092898)

[2.1. Описание предметной области 11](#_Toc167092899)

[2.2. Разработка таблиц в Microsoft SQL Server 12](#_Toc167092900)

[2.3. Нормализация таблиц 14](#_Toc167092901)

[2.4. Создание таблиц в БД 14](#_Toc167092902)

[2.5. Реализация запросов в Microsoft SQL Server 17](#_Toc167092903)

[3. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА БАЗЫ ДАННЫХ 22](#_Toc167092904)

[3.1. Разграничение прав доступа и защита целостности данных 22](#_Toc167092905)

[3.2. Функционал форм для работы с БД 25](#_Toc167092906)

[ЗАКЛЮЧЕНИЕ 36](#_Toc167092907)

[СПИСОК ИCПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 37](#_Toc167092908)

# **ВВЕДЕНИЕ**

Эта курсовая работа сосредоточена на разработке базы данных и пользовательского интерфейса для системы управления интернет-магазином. Главной целью проекта является создание эффективной и удобной системы для хранения и изменения данных о покупателях, товарах, заказах и категориях товаров.

Для достижения этой цели, необходимо выполнить несколько задач. Во-первых, провести анализ различных систем управления базами данных (СУБД), их преимуществ и недостатков, и выбрать наиболее подходящую СУБД для данного проекта.

Затем, необходимо создать ER-модель базы данных, включающую все необходимые таблицы и связи между ними. После этого, нужно настроить первичные и внешние ключи, а также связи между таблицами, что позволит эффективно хранить и манипулировать данными.

Кроме того, в рамках курсовой работы, необходимо создать не только структуру базы данных, но и различные запросы, функции и хранимые процедуры, обеспечивающие более удобный и эффективный доступ к данным.

Для обеспечения удобного доступа к данным и возможности их изменения, должен быть разработан пользовательский интерфейс.

В целом, данная курсовая работа представляет собой сложный проект, включающий в себя множество различных задач и требующий знаний в области баз данных и программирования. Успешное выполнение проекта позволит создать удобную и эффективную систему для системы управления интернет-магазином продажи электроники, что существенно облегчит работу и повысит качество предоставляемых услуг.

1. **Сравнительная характеристика СУБД**
   1. **СУБД PostgreSQL**

PostgreSQL является бесплатной и открытой реляционной системой управления базами данных (СУБД), написанной на языке Си. Она поддерживает множество типов данных и богатую функциональность, а также различные операционные системы, включая Windows, Linux, OS X, FreeBSD, NetBSD, HP-UX, OpenBSD, Solaris и Unix.

При обновлении данных в PostgreSQL, разработчики вставляют новый столбец и строку, что приводит к увеличению количества столбцов и строк и размера базы данных. Однако, это способствует более удобному чтению и пониманию данных.

PostgreSQL поддерживает организацию таблиц на основе индексов, но в ранних версиях не предусмотрено автоматическое обновление индексов. Решение также позволяет искать много индексов в одном поиске.

Пользователи PostgreSQL могут загружать JSON-файлы, а также индексировать их и выполнять частичные обновления. Кроме того, поддерживаются геопространственные данные, многомерные массивы и другие типы данных.

PostgreSQL изолирует процессы, рассматривая их как отдельные процессы операционной системы. Каждая база данных имеет свою собственную память и запускает свой собственный процесс. Это упрощает управление и мониторинг, но может занять много времени и вычислительных ресурсов при масштабировании нескольких баз данных.

Временные таблицы в PostgreSQL делятся на локальные и глобальные и настраиваются с помощью гибких переменных.

СУБД PostgreSQL предлагает множество инструментов для масштабирования и оптимизации программного обеспечения, включая кластеризацию, интеграцию с ИИ, совместную работу и отслеживание проблем.

Для дефрагментации базы данных, PostgreSQL предоставляет функцию, которая сканирует все таблицы уровня данных, чтобы найти пустые строки и удалить ненужные элементы. Это освобождает дисковое пространство, но может загрузить процессор и повлиять на производительность приложения.

Пользователи PostgreSQL могут создавать разделы LIST и RANGE, где индекс раздела создается вручную. Разработчикам необходимо определить дочерний и родительский столбцы, прежде чем назначать для них раздел.

Некоторые разработчики отмечают, что процесс установки PostgreSQL может быть немного сложным, поскольку он требует дополнительных настроек.

* 1. **СУБД MongoDB**

MongoDB является бесплатной документо-ориентированной системой управления базами данных. Она не требует описания схемы таблиц и считается одним из классических примеров NoSQL-систем. MongoDB использует JSON-подобные документы и схему базы данных, что обеспечивает гибкость и легкость в работе с разнообразными и сложно структурированными данными. Система написана на языке C++ и имеет открытый исходный код, что делает ее доступной для любого разработчика.

MongoDB подходит для операционных систем семейства Linux, Windows и macOS. В MongoDB реализована система, при которой данные хранятся в «табличных» документах форматов, близких к JSON. Информация записывается в виде пар из ключей и значений, что позволяет размещать очень разнообразную и сложно структурированную информацию.

Вместо традиционных понятий таблиц и полей в MongoDB используются свои сущности, которые отличаются от принятых в реляционных базах данных, но во многом схожи. Id - это идентификатор, который помогает обращаться к тем или иным данным, по нему записи отличают друг от друга. Поле - это одна запись в документе, которой соответствует определенный набор данных. Документ - это файл в формате BSON: название расшифровывается как binary JSON, или бинарный JSON. Отличие от стандартного JSON в том, что формат работает еще быстрее, но файлы в нем занимают меньше места. В документе хранятся однотипные строго структурированные записи: ключи и значения. Документ может хранить сложную по структуре информацию, он более гибкий — для отдельных записей можно опускать те или иные поля. Максимальный размер документа — 16 Мб. Для сохранения данных большего размера используется технология GridFS.

Коллекция - это набор из документов. В одном наборе могут храниться документы с разнообразными данными — это главное отличие от таблиц. Коллекция разнородна, документы внутри нее могут различаться структурой, размером, значениями и связями. Поэтому MongoDB считается отличным выбором для хранения слабо структурированной информации.

Встроенный документ - это структура, которая встраивается в другую структуру. В базах данных, которые управляются SQL, есть операция JOIN, объединяющая между собой сведения из разных таблиц. В MongoDB другая структура: обычно все находится внутри одной коллекции, а операции JOIN не предусмотрено. Для связей и разделений используются встроенные документы — структуры встраиваются друг в друга.

База данных - это общее хранилище, где находятся коллекции, в которых, соответственно, расположены документы. У баз в MongoDB есть интересная особенность: когда база создана, но в нее ничего не записано, она не существует. Это отличает ее от пустых реляционных баз, которые существуют, даже если в них ничего нет.

Особенности MongoDB:

Индексация. Данные внутри базы можно индексировать — так система быстрее найдет к ним доступ. Для данных создаются индексы, специальные объекты, которые хранят указатели на ту или иную информацию: обычно это конкретные значения и ссылки на них. Благодаря индексам можно реализовать поиск по базе, он не будет выглядеть как простой перебор всех данных.

Репликация. Так называется возможность работы с репликами — копиями базы данных. Можно иметь несколько связанных друг с другом копий БД на разных серверах, и одна из них будет главной, а остальные — вспомогательными. Изменения сначала будут вноситься в главную базу, а потом передаваться другим. В итоге, если с основной копией что-то случится, любая существующая реплика сможет ее заменить. Замена происходит автоматически.

Сегментирование. Это разделение базы данных на отдельные сегменты и их распределение по разным серверам, что позволяет балансировать нагрузку на мощности. Принцип разбиения определяет администратор, поэтому распределение по разным частям серверного кластера тоже можно спланировать.

Запросы «ad hoc». Такие запросы еще называют специальными. Одна из особенностей MongoDB — гибкая поддержка разнообразных запросов. СУБД принимает запросы на поиск по разным полям, работает с функциями JavaScript и может возвращать пользовательские функции в ответ на запрос. Она поддерживает регулярные выражения. MongoDB позволяет получить в качестве ответа диапазон или случайное значение — запросы могут быть в том числе очень сложными.

Grid File System. Сокращенно эта технология называется GridFS. По сути, это то же сегментирование, но в рамках документа. Массивные данные хранятся в двух коллекциях: files и chunks: files — коллекция, в которой находятся сведения о файлах. Это их имена и метаданные, содержащие информацию об объеме и других параметрах; chunks — коллекция, где хранятся сами файлы, но не целиком, а разбитые на небольшие сегменты. Размер каждого сегмента обычно 256 Кб, но эта цифра может меняться. Благодаря такому разделению на маленькие составные части в базе можно хранить даже очень большие файлы. Это позволяет обойти максимальный размер документа в 16 Мб.

Преимущества MongoDB:

Быстрая работа. Благодаря структуре документов, поддержке индексации и другим особенностям MongoDB работает быстро, без задержек. База производительна, часто используется во внутренних системах, на сайтах — там, где скорость имеет значение.

Гибкость. MongoDB позволяет хранить разнородные данные, поэтому она гибкая и легко адаптируется под разнообразные задачи. Например, если информация не укладывается в общую структуру или собирается разными способами и сильно различается, MongoDB все равно позволит успешно хранить ее и получать доступ.

Легкая масштабируемость. Возможность быстро и легко масштабировать систему. Если информация в базе должна измениться, например, планируется добавить новое поле, не нужно радикально перекраивать структуру всей БД. Это возможно благодаря документно-ориентированной модели и особому формату данных.

Отсутствие сложных соединений. В MongoDB не нужно описывать схему таблиц. Конструкции вроде JOIN не используются. В реляционных базах JOIN соединяет между собой данные из разных таблиц, но при документно-ориентированной модели необходимость в таких соединениях отсутствует.

Возможность работы на нескольких серверах. Это возможно благодаря механизмам репликации и сегментирования. Первый позволяет создавать работающие копии базы, которым в любой момент можно передать управление; второй помогает грамотно распределять нагрузку. Такие возможности делают работу системы стабильнее и снижают риск отказа базы в важный момент.

Недостатки MongoDB:

Отсутствие хранимых процедур и функций. Поддержка хранимых процедур в MongoDB не предусмотрена, и это не дает в полной мере автоматизировать работу с БД. Пользовательские функции, или UDF, похожи на хранимые процедуры, но различаются особенностями сохранения и вызова. Их можно реализовать самостоятельно, они сохраняются с помощью специальной команды и позднее используются снова.

Неполное соответствие ACID. ACID — это набор принципов для баз данных, соответствие которым делает систему стабильной и предсказуемой. Принципов всего четыре: атомарность, согласованность, изолированность, устойчивость. MongoDB, в отличие от распространенных реляционных СУБД, соответствует им не полностью.

Сложности с транзакциями. Документно-ориентированный подход дает много плюсов, но из-за него сначала было затруднительно работать с транзакциями.

Трудности при работе с жестко связанными данными. У разных документов внутри коллекции может быть совершенно различная структура, и сами коллекции не обязаны походить друг на друга. Это упрощает хранение слабо связанной информации, но если данные имеют жесткие связи между собой, то подход перестает быть удобным.

* 1. **СУБД MS SQL Server**

Инфраструктура SQL Server включает в себя множество дополнительных инструментов, таких как службы отчетов, системы интеграции и аналитика, которые имеют большое значение для компаний, управляющих несколькими командами.

Хотя СУБД имеет бесплатную версию для разработчиков и малых предприятий, она имеет ограничения, такие как поддержка только одного процессора, 1 ГБ максимальной памяти и максимальный размер базы данных 10 ГБ. Полная версия сервера стоит $931.

SQL Server написана на языке С++ с закрытым исходным кодом и поддерживает как Linux, так и Windows.

Одной из ключевых особенностей SQL Server является наличие трех ядер, которые отвечают за обновления строк. Кроме того, система имеет механизм in-memory, который позволяет анализировать качество обновленной базы данных с помощью сборщика мусора.

SQL Server предлагает широкие функциональные возможности для временного управления таблицами, автоматизированного управления индексами и оптимизированных для памяти таблиц. Она также обеспечивает полную поддержку документов JSON и имеет множество дополнительных функций для данных GPS, пользовательских типов и иерархической информации.

Преимущества SQL Server включают масштабируемость системы, высокую производительность, автоматизацию рутинных административных задач, удобный поиск и поддержку работы с другими решениями Майкрософт.

Однако, у SQL Server есть и недостатки, такие как зависимость от операционной среды, поскольку она работает только с системой Windows.

* 1. **Выбор СУБД**

Был проведен сравнительный анализ вышеперечисленных СУБД, их особенностей, преимуществ и недостатков. Опираясь на данную предметную область, среди PostgreSQL, MongoDB и MS SQL Server, последняя имеет следующие преимущества:

* высокая производительность и надежность;
* широкие функциональные возможности для управления таблицами;
* автоматизированное управление индексами;
* масштабируемость системы;
* автоматизация рутинных административных задач;
* удобная интеграция с другими решениями Майкрософт.

Эти и другие возможности SQL Server повлияли на выбор данной СУБД для реализации БД «интернет-магазина продажи электроники».

1. **РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ**
   1. **Описание предметной области**

ElectroHub — это современный интернет-магазин электроники, начавший свою деятельность в цифровой коммерции. На данный момент компания предлагает широкий ассортимент электронных товаров, но уже зарекомендовала себя как надежный и эффективный партнер для покупателей в регионе.

Компания ElectroHub специализируется на высококачественных электронных товарах, предлагая оптимальные условия для покупки и доставки любых видов электроники, будь то ноутбуки или домашняя электроника. ElectroHub сотрудничает с ведущими производителями электроники, такими как Samsung, Apple и другими, и постоянно обновляет свой ассортимент, отслеживая новейшие тенденции в области электроники.

ElectroHub разработала систему управления электроникой, которая обеспечивает эффективное управление продажами и запасами электроники в интернет-магазине. Это мощный инструмент, созданный с учетом потребностей современных интернет-магазинов, стремящихся к оптимизации своих внутренних процессов и максимизации прибыли.

В ElectroHub уделяют большое внимание деталям и индивидуальному подходу к каждому клиенту. Команда опытных специалистов готова помочь вам с настройкой системы, обучением персонала и предоставлением поддержки на всех этапах работы с ElectroHub. Мы верим, что с помощью нашего инструмента вы сможете достичь новых высот в управлении своим бизнесом, повысить эффективность и увеличить прибыль. ElectroHub создает атмосферу надежности и доверия, обеспечивая спокойствие клиентов и их уверенность в качестве своих покупок.

* 1. **Разработка таблиц в Microsoft SQL Server**

Для базы данных по предметной области информационной системы интернет-магазина необходимы следующие сущности:

* Users (Пользователи) с атрибутами: UserID, Username, PasswordHash, UserRole;
* Products (Товары) с атрибутами: ProductID, ProductName, ProductDescription, ProductPrice, ProductQuantity, CategoryID, VendorID;
* Orders (Заказы) с атрибутами: OrderID, OrderDate, OrderStatus, CustomerID;
* Customers (Покупатели) с атрибутами: CustomerID, CustomerName, CustomerAddress, CustomerPhone;
* OrderItems (Детали заказа) с атрибутами: OrderItemID, OrderID, ProductID, Quantity, TotalAmount;
* Vendors (Поставщики) с атрибутами: VendorID, VendorName, VendorAddress, VendorPhone;
* ProductCategories (Категории товаров) с атрибутами: CategoryID, CategoryName.

По выделенным сущностям была разработана ER-модель (Рис. 1).



Рис. 1 – ER-модель БД система интернет-магазина

* 1. **Нормализация таблиц**

Данная модель соответствует третьей форме нормализации. В ней отсутствуют связи многие-ко-многим, выделены ключи и установлены связи.

* 1. **Создание таблиц в БД**

Скрипт создания таблиц БД:

CREATE TABLE Vendors (

VendorID int PRIMARY KEY IDENTITY,

VendorName nvarchar(100),

VendorAddress nvarchar(100),

VendorPhone nvarchar(20)

);

CREATE TABLE ProductCategories (

CategoryID int PRIMARY KEY IDENTITY,

CategoryName nvarchar(100)

);

CREATE TABLE Products (

ProductID int PRIMARY KEY IDENTITY,

ProductName nvarchar(100),

ProductDescription nvarchar(MAX),

ProductPrice decimal,

ProductQuantity int,

CategoryID int FOREIGN KEY REFERENCES ProductCategories(CategoryID),

VendorID int

);

CREATE TABLE Customers (

CustomerID int PRIMARY KEY IDENTITY,

CustomerName nvarchar(100),

CustomerAddress nvarchar(100),

CustomerPhone nvarchar(20)

);

CREATE TABLE Orders (

OrderID int PRIMARY KEY IDENTITY,

OrderDate datetime,

OrderStatus nvarchar(50),

CustomerID int FOREIGN KEY REFERENCES Customers(CustomerID)

);

CREATE TABLE OrderItems (

OrderItemID int PRIMARY KEY IDENTITY,

OrderID int FOREIGN KEY REFERENCES Orders(OrderID),

ProductID int FOREIGN KEY REFERENCES Products(ProductID),

Quantity int,

TotalAmount decimal

);

CREATE TABLE Users (

UserID int PRIMARY KEY IDENTITY,

Username nvarchar(50),

PasswordHash nvarchar(50),

UserRole nvarchar(50)

);

Заполнение таблиц:

INSERT INTO Vendors (VendorName, VendorAddress, VendorPhone)

VALUES

('Samsung Electronics', 'Москва, ул. Ленина, 1', '8-811-111-1111'),

('Xiaomi Inc.', 'чайнатаун пекин', '8-822-222-2222'),

('Apple Inc.', 'Калифорния (не роллы)', '8-833-333-3333');

INSERT INTO ProductCategories (CategoryName)

VALUES

('Смартфоны'),

('Ноутбуки'),

('Планшеты'),

('Телевизоры');

INSERT INTO Customers (CustomerName, CustomerAddress, CustomerPhone)

VALUES

('Иван Иванов', 'Иваново, ул. Ивана Фомина, 10', '8-926-123-4567'),

('Евгений Бронин', 'Санкт-Петербург, прос. Энгельса, д. 129, к.4', '8-812-111-2222'),

('Ольга Ольгина', 'Екатеринбург, ул. Ольги, 35', '8-343-333-4444');

INSERT INTO Orders (OrderDate, OrderStatus, CustomerID)

VALUES

('2024-05-09', 'В обработке', 1),

('2024-05-10', 'Доставка', 2),

('2024-05-11', 'Оплачено', 3);

INSERT INTO Products (ProductName, ProductDescription, ProductPrice, ProductQuantity, CategoryID, VendorID)

VALUES

('Samsung Galaxy S22', 'Смартфон с камерой 50 Мп', 59999.99, 100, 1, 1),

('Xiaomi RedmiBook 15', 'Ноутбук с процессором Intel Core i5', 39999.99, 50, 2, 2),

('Apple iPad Air 4', 'Планшет с экраном 10.9 дюймов', 49999.99, 200, 3, 3);

INSERT INTO OrderItems (OrderID, ProductID, Quantity, TotalAmount)

VALUES

(1, 1, 2, 119999.99),

(1, 2, 1, 39999.99),

(2, 3, 3, 149999.99);

INSERT INTO Users (Username, PasswordHash, UserRole)

VALUES

('admin', 'admin', 'Администратор'),

('user', 'user', 'Пользователь');

Спроектированную ER модель, перенес в базу данных и создал диаграмму таблиц (Рис. 2).

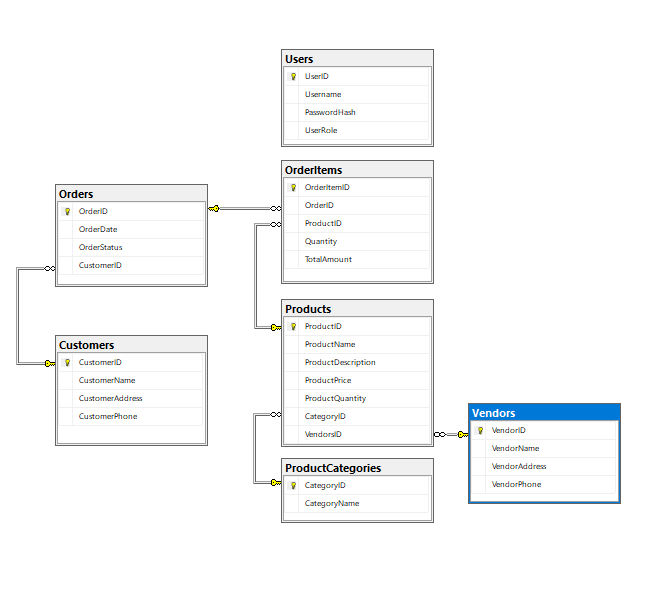


Рис. 2 – Диаграмма БД

* 1. **Реализация запросов в Microsoft SQL Server**

Для БД интернет-магазина будут актуальны следующие запросы:

1. функция: получение имени клиента по его ID (Рис. 3-4);
2. функция: возвращает среднюю цену продуктов в указанной категории (Рис. 5-6);
3. процедура: добавление нового клиента (Рис. 7-8);
4. транзакция: для добавления нового заказа вместе с деталями заказа (Рис. 9-13);
5. транзакция: для обновления статуса заказа (Рис. 14-16).

Создаем функцию (1) (Рис. 3):

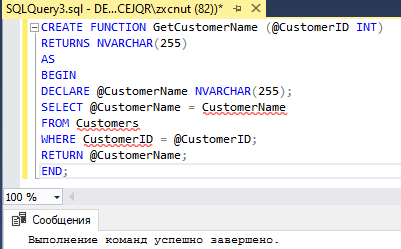


Рис. 3 – Функция для получения имени клиента по его ID

Пример использования функции (Рис. 4):

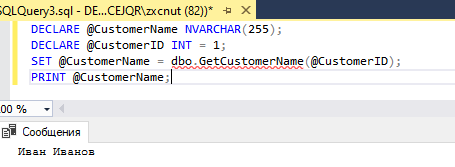


Рис. 4 – Использование функции

Создаем функцию (2) (Рис. 5):

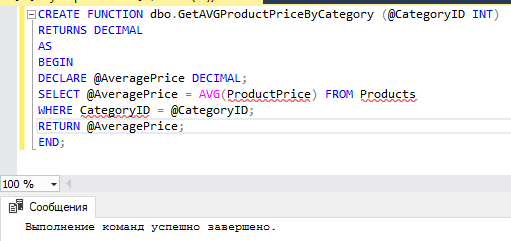


Рис. 5 – Функция для подсчёта средней цены продуктов в категории

Пример использования функции (Рис. 6):

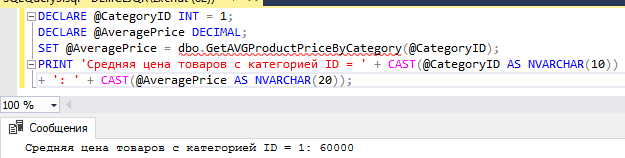


Рис. 6 – Пример использования функции

Создаем процедуру (3) (Рис. 7):

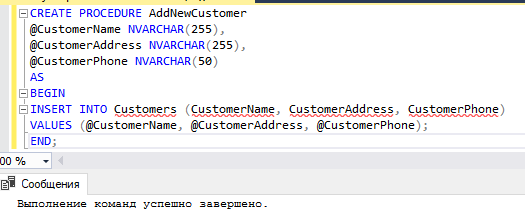


Рис. 7 – Код процедуры

Пример использования процедуры (Рис.8)

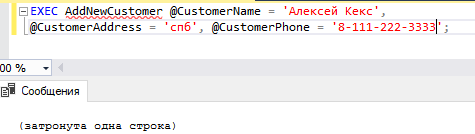


Рис. 8 – Пример использования процедуры

Создаем транзакцию (4) (Рис. 9):

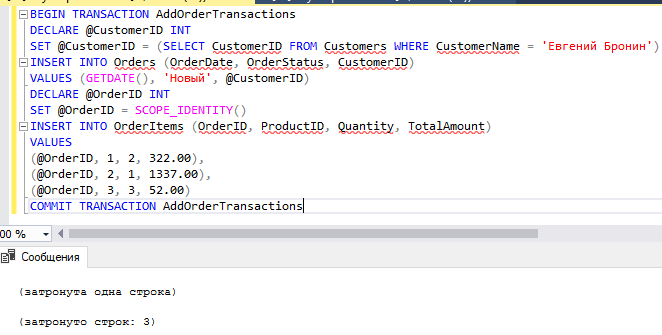


Рис. 9 – Создание транзакции

Таблица Orders до транзакции (Рис. 10)

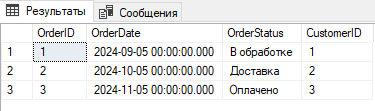


Рис. 10 – Таблица Orders

Таблица OrderDetails до транзакции (Рис. 11)

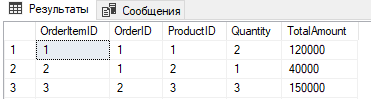


Рис. 11 – Таблица OrderDetails

Транзакция прошла успешно (Рис. 12):

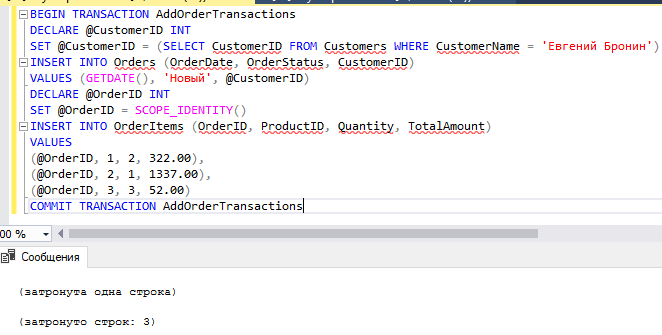


Рис. 12 – Успешное прохождение транзакции

Таблица Orders после транзакции (Рис. 13)

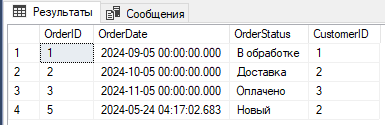


Рис. 13 – Таблица Orders

Таблица OrderDetails после транзакции (Рис. 14)

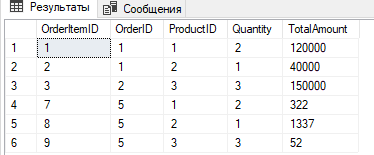


Рис. 14 – Таблица OrderDetails

Создаем транзакцию (5) (Рис. 15):

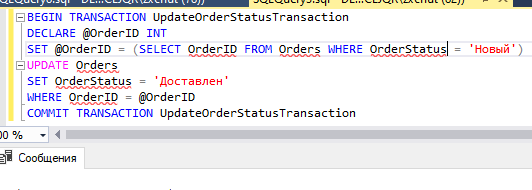


Рис. 15 – Создал транзакцию

Таблица Orders до транзакции (Рис. 16)

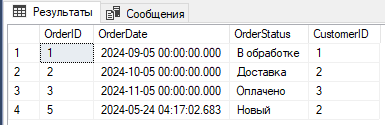


Рис. 16 – Таблица Orders

Транзакция успешно выполнена (Рис. 17)

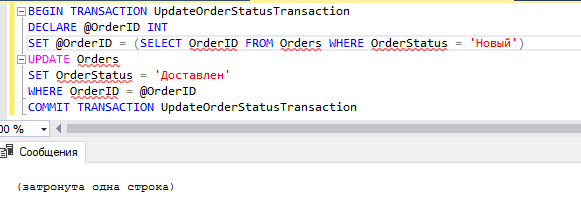


Рис. 17 – Успешное выполнение транзакции

Таблица Orders после транзакции (Рис. 18):

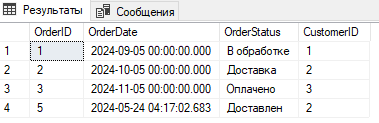


Рис. 18 – Таблица Orders

1. **РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА БАЗЫ ДАННЫХ**
   1. **Разграничение прав доступа и защита целостности данных**

В системе для пользователей предусмотрены 2 типа ролей, «Admin» и «User»

При входе в приложение пользователь вводит свой логин и пароль (Рис. 19).

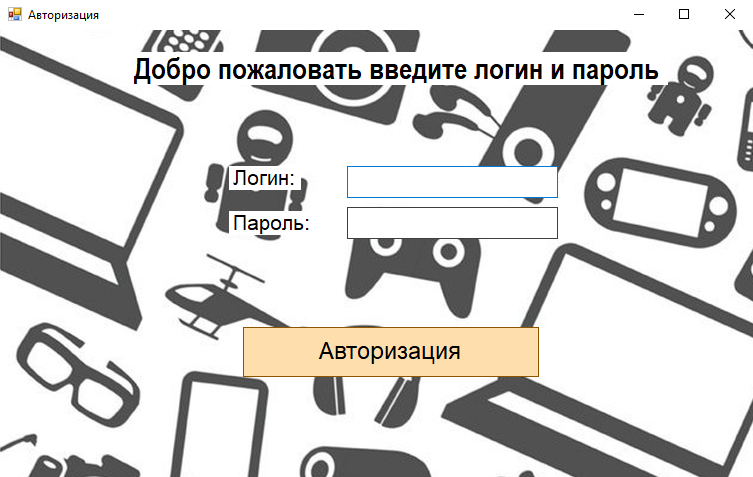


Рис. 19 – форма входа

Либо же, если логин или пароль неверный, то выскакивает ошибка (Рис. 20).

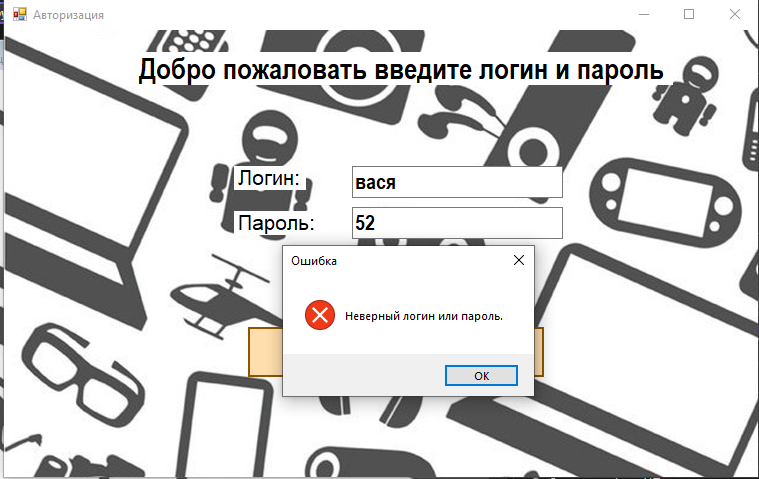


Рис. 20 – форма входа (Обработана ошибка)

User может только просматривать данные в приложении БД (Рис. 21).

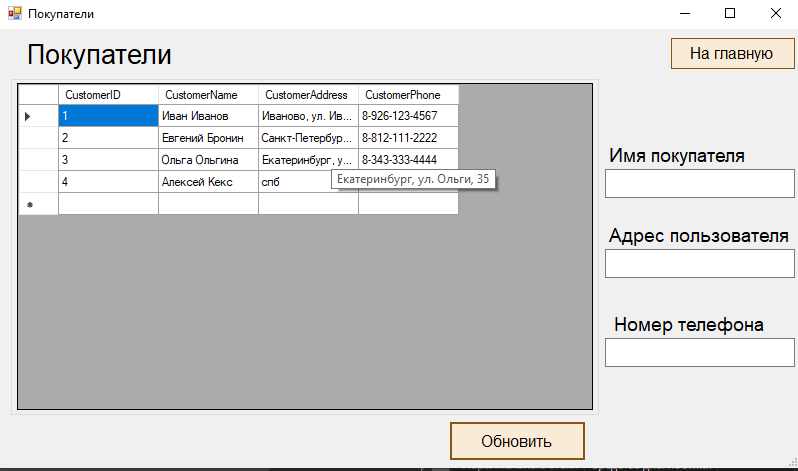


Рис. 21 – Вид пользователя на форму «Поставщики»

Admin имеет доступ к полному функционалу программы Администратор может просматривать, редактировать, добавлять и удалять данные из базы (Рис. 22).

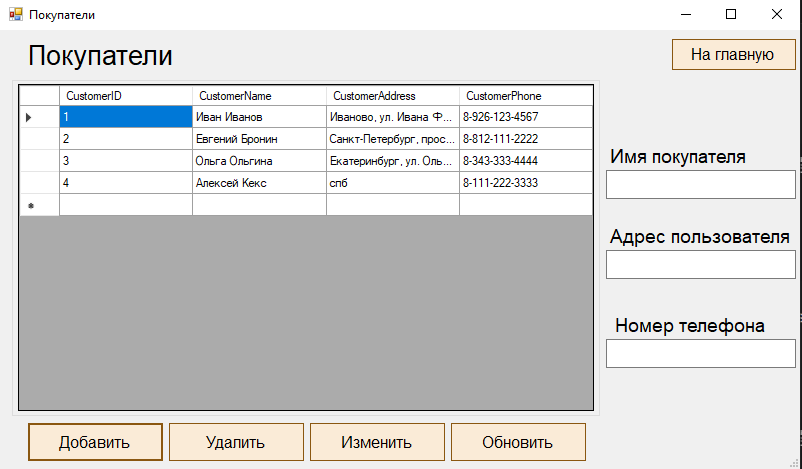


Рис.22 – Вид администратора на форму «Поставщики»

* 1. **Функционал форм для работы с БД**

При входе в приложение пользователь вводит свой логин и пароль (Рис. 23).

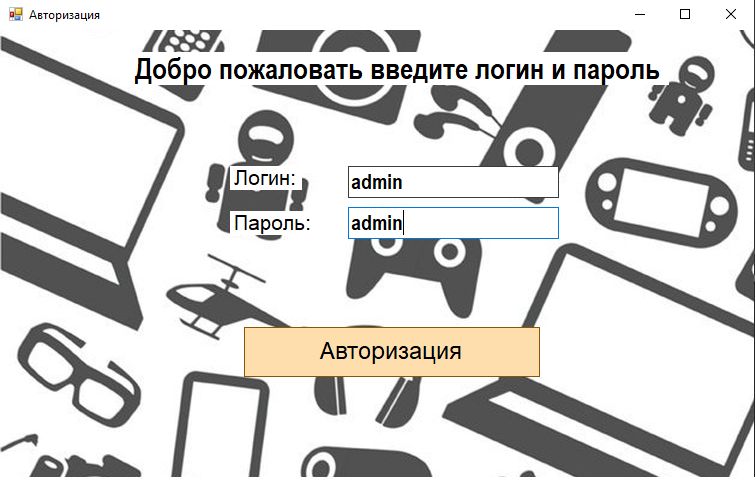


Рис. 23 – форма для входа в учетную запись

Затем пользователя встречает главное меню с возможностью выбрать таблицу для просмотра (Рис. 24).

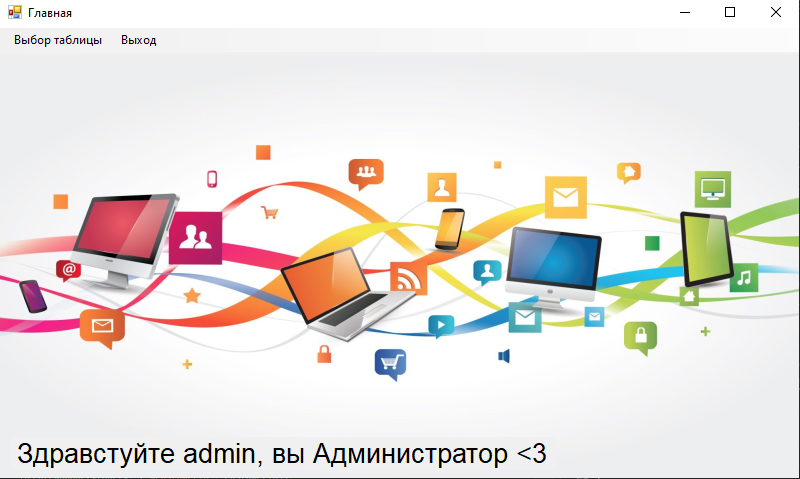


Рис. 24 – Главная форма

После этого можно открыть любую таблицу из БД, выйти из приложения, либо открыть специальную таблицу «Пользователи», доступную только для администратора (Рис. 25).

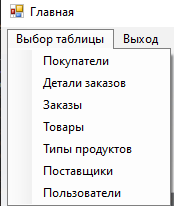


Рис. 25 – главная форма

Откроем таблицу «Поставщики» и попробуем сначала добавить, затем изменить и удалить запись (Рис. 26).

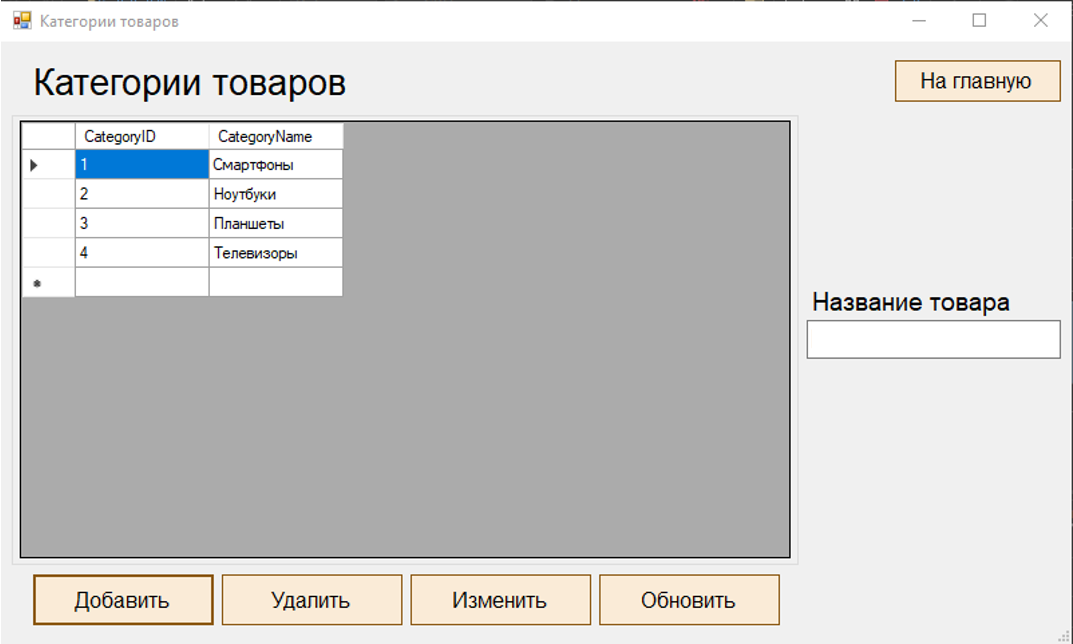


Рис. 26 – таблица Категории товаров

Для того чтобы добавить новую запись, пользователь должен ввести данные в соответствующие поля, а затем нажать кнопку добавить (Рис. 27).

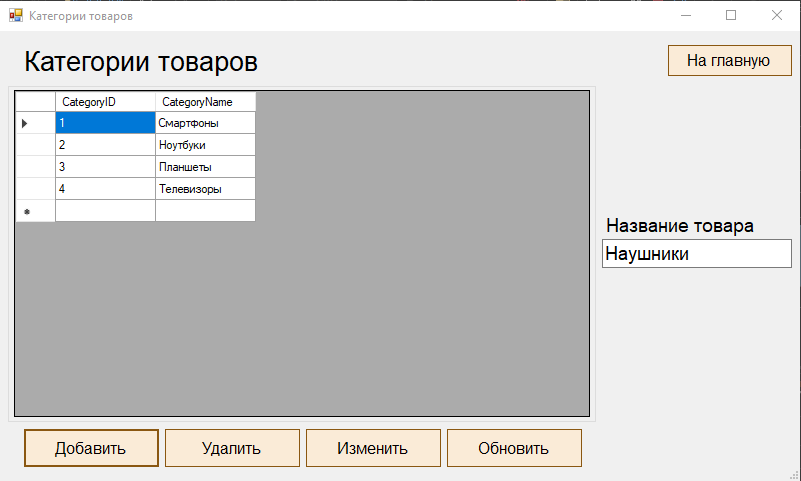


Рис. 27 – форма добавления новой категории товара

Категория товара успешно добавлена (Рис. 28).

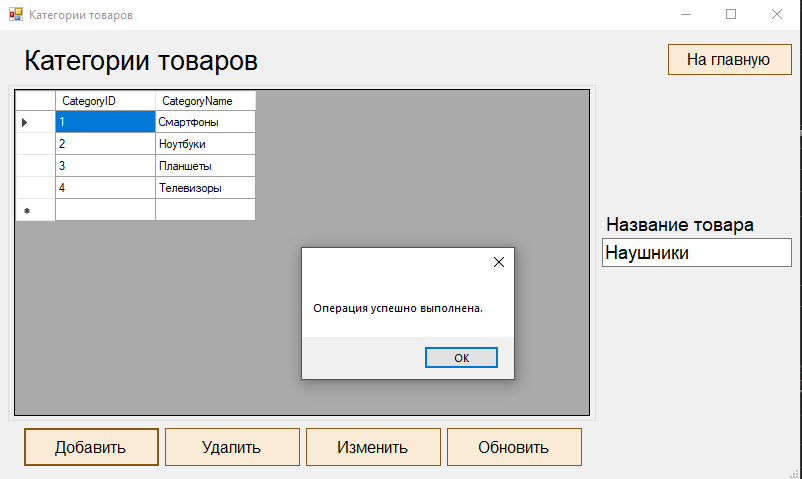


Рис. 28 – уведомление об успешном добавлении категории товара

Категория товара появилась в таблице в БД и отображается в приложении (Рис. 29).

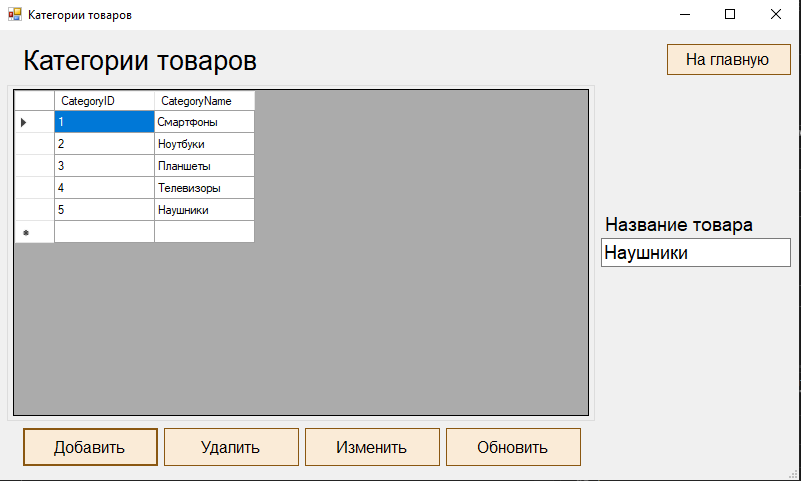


Рис. 29 – таблица Категории товаров

Попробуем изменить данные категории товара, для того чтобы изменить нужно нажать на соответствующего категории, его данные перенесутся в соответствующие поля, а затем изменить нестыковки и нажать на кнопку изменить (Рис. 30).

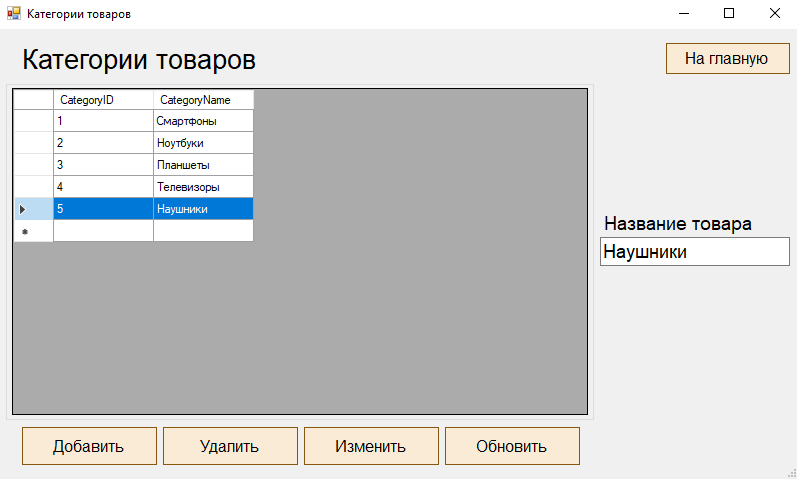


Рис. 30– форма изменение категории товара

Категория товара успешно изменена (Рис. 31).

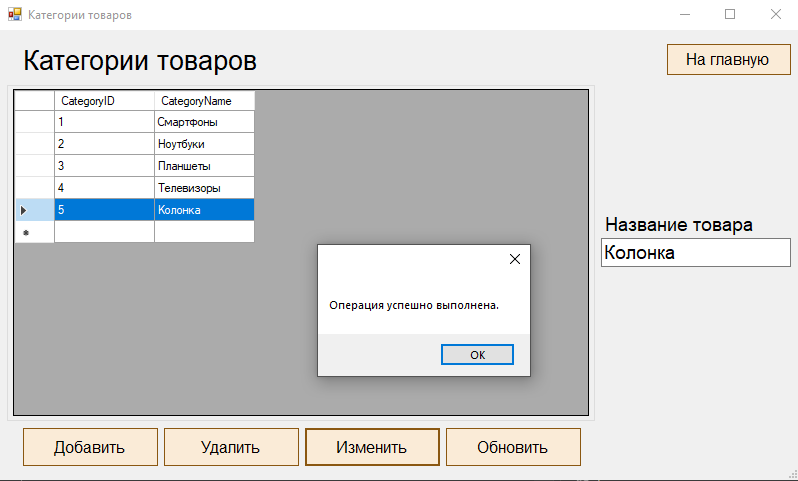


Рис. 31– измененная таблица поставщики

При удалении категории товара аналогично нужно выбрать его, затем нажать кнопку удалить (Рис. 32).

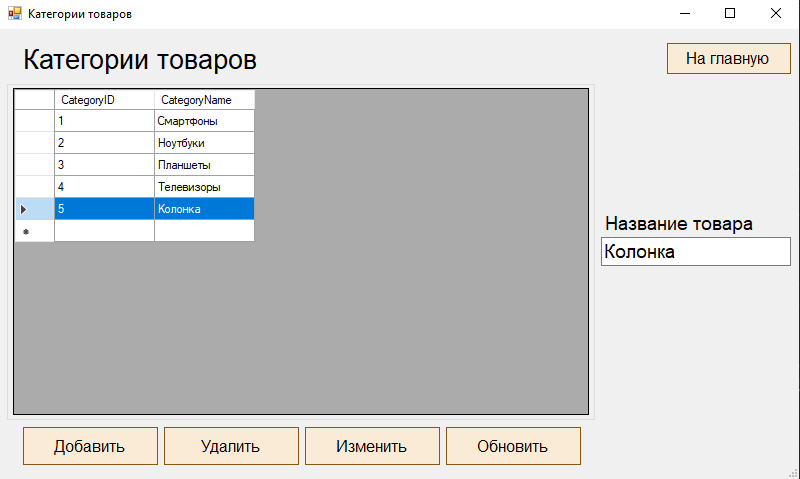


Рис. 32 – форма с выбором удаления категории товара

Выберем категорию товара и удалим её (Рис. 33).

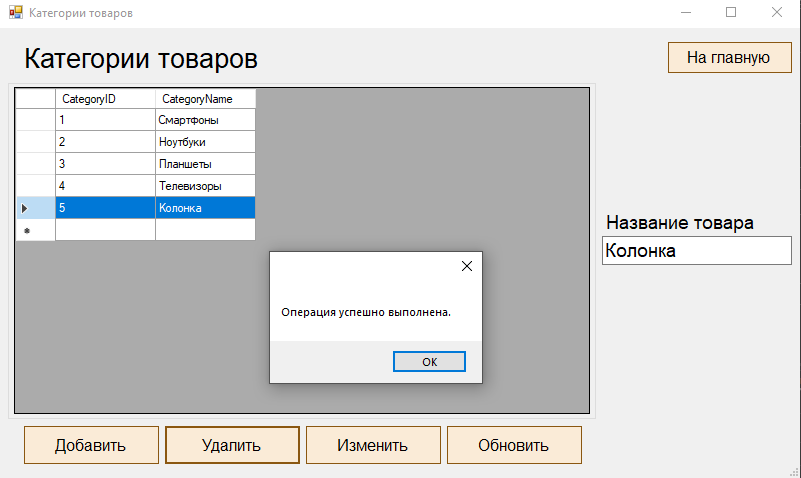


Рис. 33 – форма с выбором удаления категории товара

Удаление прошло успешно (Рис. 34).

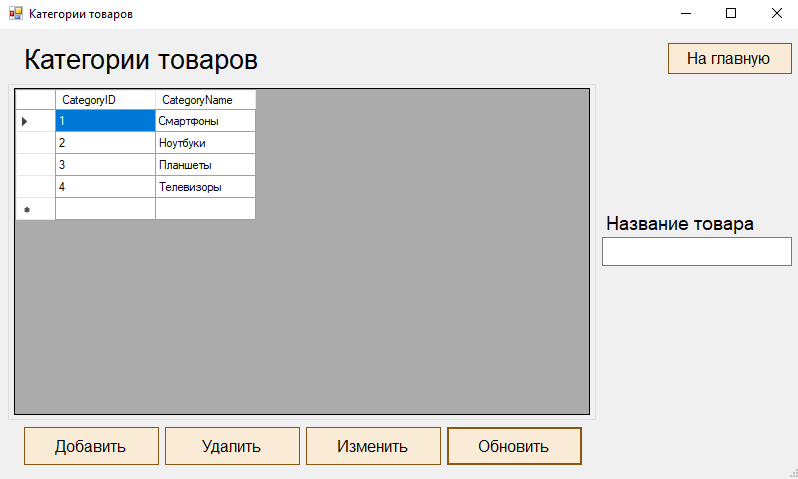


Рис. 34 – таблица категории товаров

При нажатию на кнопку «На главную» или закрытии формы дополнительных форм пользователь оказывается на главной форме (Рис. 35).

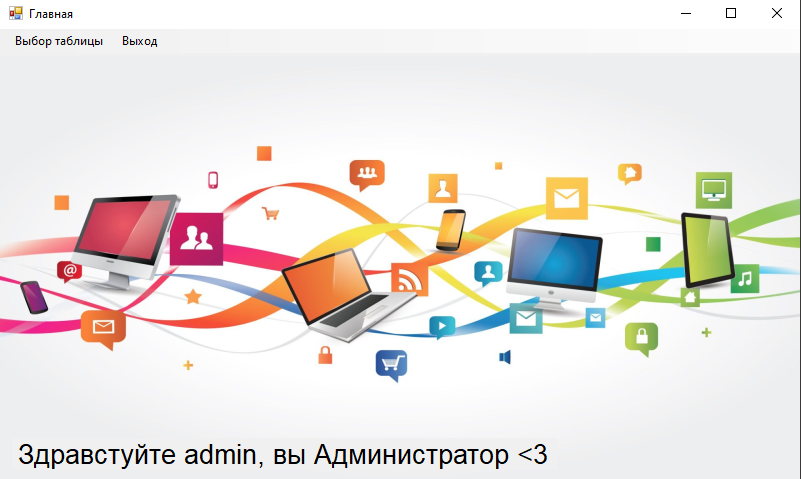


Рис. 35 – главная форма

Так как мы администратор, откроем таблицу «Пользователи» (Рис. 36).

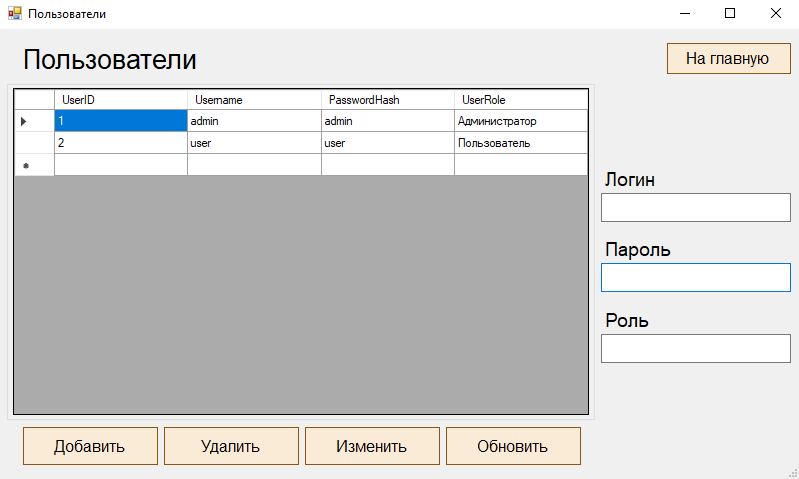


Рис. 36 – таблица пользователи

Попробуем добавить нового пользователя и войти в его учетную запись (Рис. 37).

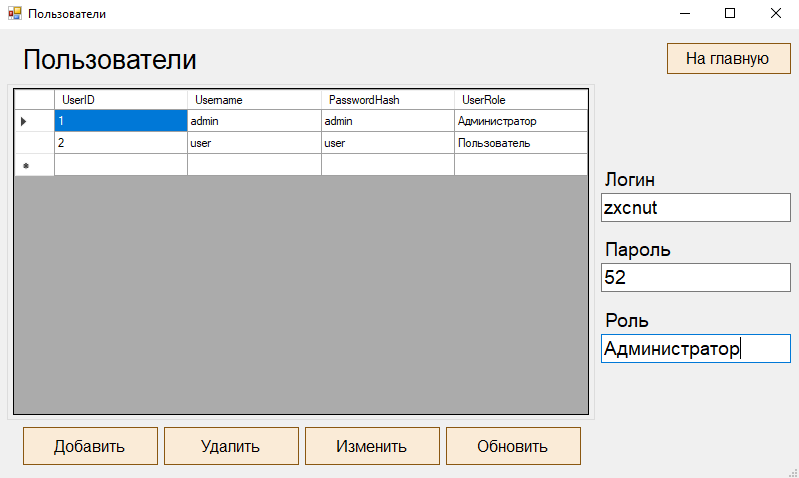


Рис. 37 – форма добавления нового пользователя

Пользователь успешно добавлен (Рис. 38)

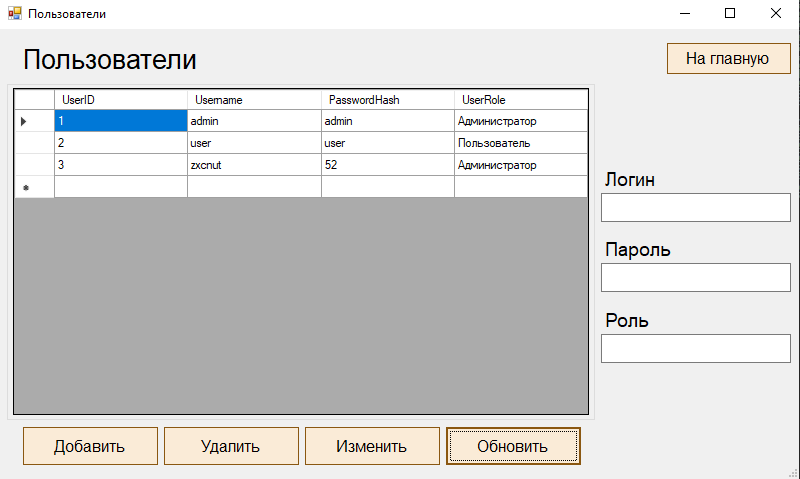


Рис. 38– таблица пользователи с новым пользователем

Попробуем войти в его учетную запись (Рис. 39).

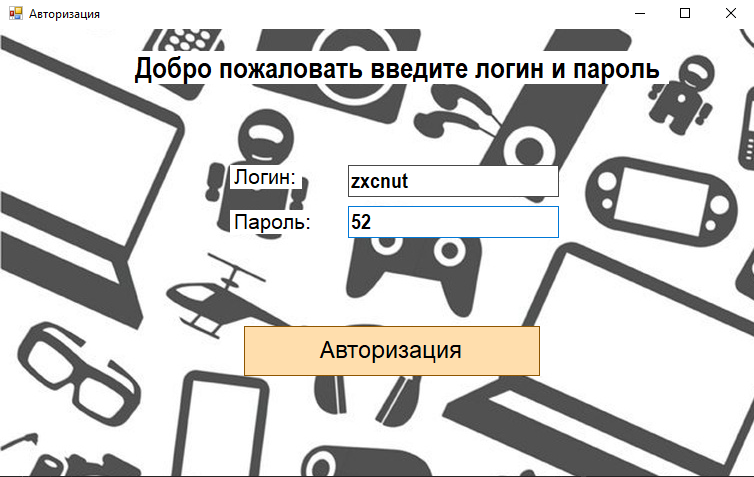


Рис. 39 – форма входа, попытка войти в нового пользователя

Мы успешно вошли, так как это обычный пользователь, а не админ, он не может зайти на таблицу «Пользователи» (Рис. 40).



Рис. 40 – главная форма

Рассмотрит вид форм от его лица (Рис. 41-42).

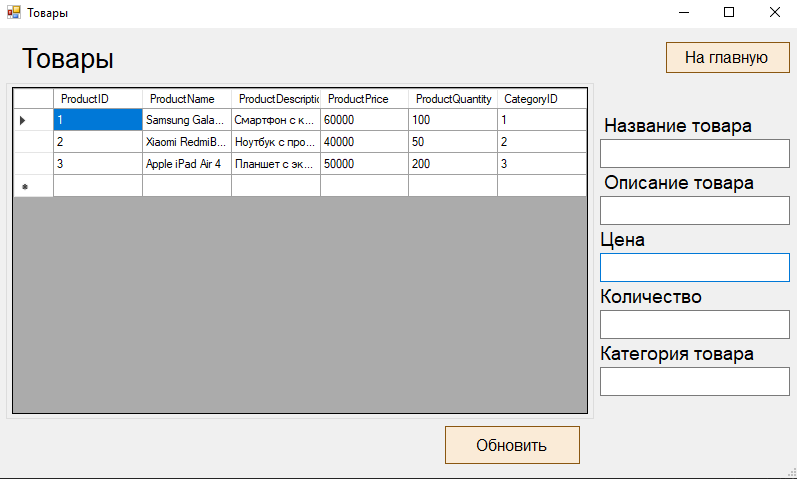


Рис. 41 –форма Товары

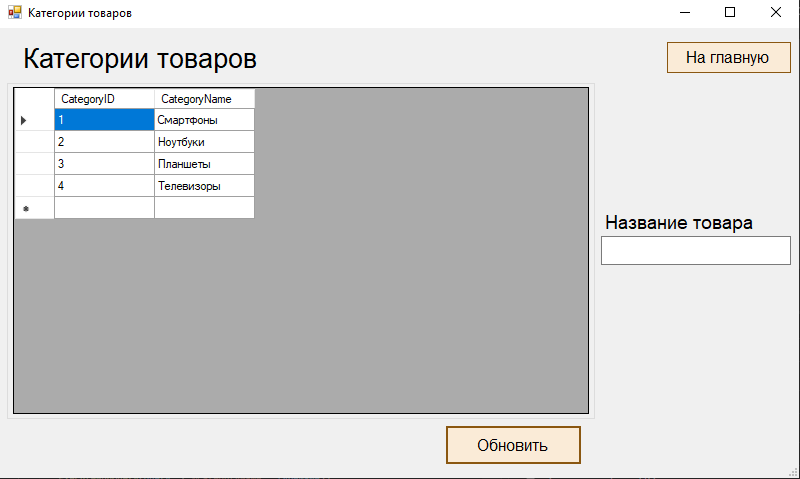


Рис. 42 – форма Категории товаров

Как мы видим, пользователь с ролью User имеет право только обновлять таблицы, но не вносить в них изменения

Код приложения представлен в приложении.

# **ЗАКЛЮЧЕНИЕ**

Результатом выполнения данной курсовой работы стала реализованная база данных по предметной области «интернет-магазин продажи электроники», а также реализована форма для работы с этой БД.

В ходе выполнения курсовой работы было произведено подробное изучение выбранной СУБД – Microsoft SQL Server. Также была реализована ER-модель по предметной области, после чего она была перенесена в СУБД, где были настроены первичные и внешние ключи, а также связи между таблицами.

В СУБД были реализованы запросы для работы с базой данных. С помощью программы Visual Studio был создан удобный пользовательский интерфейс, позволяющий производить различные манипуляции с базой данных. Данные манипуляции реализуют просмотр, добавление, изменение и удаление записей. Также был реализован класс для удобного использования запросов к базе данных, класс для определения роли пользователя в системе.

В результате выполнения курсовой работы была получена работающая база данных, которая может быть использована для хранения и обработки информации об электронных товарах, заказах и клиентах интернет-магазина. Разработанная форма для работы с БД обеспечивает удобный и интуитивно понятный интерфейс для пользователей, что позволяет им эффективно работать с базой данных.

**СПИСОК ИCПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Гаст, Х. Объектно-ориентированное проектирование: концепции и программый код / Х. Гаст. - М.: Диалектика, 2018. - 1040 c.
2. Пирогов, В. Информационные системы и базы данных: организация и проектирование: Учебное пособие / В. Пирогов. - СПб. : BHV, 2009. - 528 c.
3. Перлова, О. Н. Проектирование и разработка информационных систем: Учебник / О. Н. Перлова. - М. : Академия, 2018. - 272 c.
4. Липаев, В.В. Сертификация программных средств / В.В. Липаев. - М.: Синтег, 2010. – 348 с.
5. Техническая документация [Офиц. сайт]. URL: <http://tdocs.su>
6. Соловьев, И. В. Проектирование информационных систем. Фундаментальный курс: Учебное пособие для высшей школы / И. В. Соловьев, А.А. Майоров; Под ред. В.П. Савиных. - М. : Академический проспект, 2009. - 398 c.

Гвоздева, Т. В. Проектирование информационных систем: технология автоматизированного проектирования. Лабораторный практикум

1. Сравнение баз данных MY SQL, POSTGRESQL, SQL SERVER, Р. Раянов: <https://vc.ru/u/155908-ruslan-rayanov/540619-sravnenie-baz-dannyh-my-sql-postgresql-sql-server>

**Приложение А**

**Код формы Покупатели(Customers)**

public partial class Customers : Form

{

private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

//для вц

//private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private SqlConnection connection;

public Customers()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

LoadCustomers();

}

private void LoadCustomers()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT CustomerID, CustomerName, CustomerAddress, CustomerPhone FROM Customers";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

private void Customers\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

string name = textBox1.Text;

string address = textBox2.Text;

string phone = textBox3.Text;

string query = "INSERT INTO Customers (CustomerName, CustomerAddress, CustomerPhone) VALUES (@CustomerName, @CustomerAddress, @CustomerPhone)";

ExecuteNonQuery(query, ("@CustomerName", name), ("@CustomerAddress", address), ("@CustomerPhone", phone));

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int customerID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["CustomerID"].Value);

string query = "DELETE FROM Customers WHERE CustomerID = @CustomerID";

ExecuteNonQuery(query, ("@CustomerID", customerID));

}

else

{

MessageBox.Show("Выберите клиента для удаления.");

}

}

private void button4\_Click(object sender, EventArgs e)

{

LoadCustomers();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int customerID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["CustomerID"].Value);

string name = textBox1.Text;

string address = textBox2.Text;

string phone = textBox3.Text;

string query = "UPDATE Customers SET CustomerName = @CustomerName, CustomerAddress = @CustomerAddress, CustomerPhone = @CustomerPhone WHERE CustomerID = @CustomerID";

ExecuteNonQuery(query, ("@CustomerName", name), ("@CustomerAddress", address), ("@CustomerPhone", phone), ("@CustomerID", customerID));

}

else

{

MessageBox.Show("Выберите клиента для обновления.");

}

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadCustomers(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["CustomerName"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["CustomerAddress"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["CustomerPhone"].Value.ToString();

}

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

private void Customers\_FormClosed(object sender, FormClosedEventArgs e)

{

Main.Instance.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

}

**Код формы Авторизация(Form1)**

public partial class Form1 : Form

{

string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

string login = textBox1.Text.Trim();

string password = textBox2.Text.Trim();

if (string.IsNullOrEmpty(login) || string.IsNullOrEmpty(password))

{

MessageBox.Show("Пожалуйста, введите логин и пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

using (SqlConnection connection = new SqlConnection(connectionString))

{

string query = "SELECT UserRole FROM Users WHERE Username = @Username and PasswordHash = @PasswordHash";

SqlCommand command = new SqlCommand(query, connection);

command.Parameters.AddWithValue("@Username", login);

command.Parameters.AddWithValue("@PasswordHash", password);

connection.Open();

object role = command.ExecuteScalar();

if (role != null)

{

string userRole = (string)role;

UserManager.SetCurrentUser(login, userRole);

Main form2 = new Main();

form2.Show();

this.Hide();

}

else

{

MessageBox.Show("Неверный логин или пароль.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**Код формы Главная(Main)**

public partial class Main : Form

{

public static Main Instance { get; private set; }

public Main()

{

InitializeComponent();

Instance = this;

}

private void Main\_Load(object sender, EventArgs e)

{

string currentRole = UserManager.CurrentUser.Role;

string userName = UserManager.CurrentUser.Username;

label1.Text = $"Здравстуйте {userName}, вы {currentRole} <3";

if (currentRole != "Администратор")

{

пользователиToolStripMenuItem.Enabled = false;

}

}

private void покупателиToolStripMenuItem\_Click(object sender, EventArgs e)

{

Customers category = new Customers();

this.Hide();

category.ShowDialog();

}

private void деталиЗаказовToolStripMenuItem\_Click(object sender, EventArgs e)

{

OrderItems category = new OrderItems();

this.Hide();

category.ShowDialog();

}

private void заказыToolStripMenuItem\_Click(object sender, EventArgs e)

{

Orders category = new Orders();

this.Hide();

category.ShowDialog();

}

private void товарыToolStripMenuItem\_Click(object sender, EventArgs e)

{

Products category = new Products();

this.Hide();

category.ShowDialog();

}

private void типыПродуктовToolStripMenuItem\_Click(object sender, EventArgs e)

{

ProductCategories category = new ProductCategories();

this.Hide();

category.ShowDialog();

}

private void поставщикиToolStripMenuItem\_Click(object sender, EventArgs e)

{

Vendors category = new Vendors();

this.Hide();

category.ShowDialog();

}

private void пользователиToolStripMenuItem\_Click(object sender, EventArgs e)

{

Users category = new Users();

this.Hide();

category.ShowDialog();

}

private void Main\_FormClosed(object sender, FormClosedEventArgs e)

{

Environment.Exit(Environment.ExitCode);

}

private void выходToolStripMenuItem\_Click(object sender, EventArgs e)

{

Environment.Exit(Environment.ExitCode);

}

}

**Код формы Детали заказов(OrderItems)**

public partial class OrderItems : Form

{

private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private SqlConnection connection;

public OrderItems()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

LoadOrderItems();

}

private void LoadOrderItems()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT OrderItemID, OrderID, ProductID, Quantity, TotalAmount FROM OrderItems";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["OrderItemID"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["ProductID"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["Quantity"].Value.ToString();

textBox4.Text = dataGridView1.SelectedRows[0].Cells["TotalAmount"].Value.ToString();

}

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

int orderID = Convert.ToInt32(textBox1.Text);

int productID = Convert.ToInt32(textBox2.Text);

int quantity = Convert.ToInt32(textBox3.Text);

decimal amount = Convert.ToDecimal(textBox4.Text);

string query = "INSERT INTO OrderItems (OrderItemID, ProductID, Quantity, TotalAmount) VALUES (@OrderItemID, @ProductID, @Quantity, @TotalAmount)";

ExecuteNonQuery(query, ("@OrderItemID", orderID), ("@ProductID", productID), ("@Quantity", quantity), ("@TotalAmount", amount));

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int detailID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["OrderItemID"].Value);

string query = "DELETE FROM OrderItems WHERE OrderItemID = @DetailID";

ExecuteNonQuery(query, ("@DetailID", detailID));

}

else

{

MessageBox.Show("Выберите деталь заказа для удаления.");

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int detailID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["OrderItemID"].Value);

int orderID = Convert.ToInt32(textBox1.Text);

int productID = Convert.ToInt32(textBox2.Text);

int quantity = Convert.ToInt32(textBox3.Text);

decimal amount = Convert.ToDecimal(textBox4.Text);

string query = "UPDATE OrderItems SET OrderItemID = @OrderItemID, ProductID = @ProductID, Quantity = @Quantity, TotalAmount = @TotalAmount WHERE OrderItemID = @DetailID";

ExecuteNonQuery(query, ("@OrderItemID", orderID), ("@ProductID", productID), ("@Quantity", quantity), ("@TotalAmount", amount), ("@DetailID", detailID));

}

else

{

MessageBox.Show("Выберите деталь заказа для обновления.");

}

}

private void button4\_Click(object sender, EventArgs e)

{

LoadOrderItems();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadOrderItems(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void OrderItems\_FormClosed(object sender, FormClosedEventArgs e)

{

Main.Instance.Show();

}

}

**Код формы Заказы(Orders)**

public partial class Orders : Form

{

private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private SqlConnection connection;

public Orders()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

}

private void Orders\_FormClosed(object sender, FormClosedEventArgs e)

{

Main.Instance.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

DateTime date = dateTimePicker1.Value;

string status = textBox1.Text;

int customerID;

if (!int.TryParse(textBox2.Text, out customerID))

{

MessageBox.Show("Невозможно преобразовать идентификатор заказчика в число.");

return;

}

string query = "INSERT INTO Orders (OrderDate, OrderStatus, CustomerID) VALUES (@OrderDate, @OrderStatus, @CustomerID)";

ExecuteNonQuery(query, ("@OrderDate", date), ("@OrderStatus", status), ("@CustomerID", customerID));

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int orderID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["OrderID"].Value);

string query = "DELETE FROM Orders WHERE OrderID = @OrderID";

string query1 = "DELETE FROM OrderItems WHERE OrderID = @OrderID";

ExecuteNonQuery(query1, ("@OrderID", orderID));

ExecuteNonQuery(query, ("@OrderID", orderID));

}

else

{

MessageBox.Show("Выберите заказ для удаления.");

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int orderID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["OrderID"].Value);

DateTime date = dateTimePicker1.Value;

string status = textBox2.Text;

int customerID = Convert.ToInt32(textBox2.Text);

string query = "UPDATE Orders SET OrderDate = @OrderDate, OrderStatus = @OrderStatus, CustomerID = @CustomerID WHERE OrderID = @OrderID";

ExecuteNonQuery(query, ("@OrderDate", date), ("@OrderStatus", status), ("@CustomerID", customerID), ("@OrderID", orderID));

}

else

{

MessageBox.Show("Выберите заказ для обновления.");

}

}

private void button4\_Click(object sender, EventArgs e)

{

LoadOrders();

textBox1.Text = "";

textBox2.Text = "";

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

string dateString = dataGridView1.SelectedRows[0].Cells["OrderDate"].Value.ToString();

DateTime date;

if (DateTime.TryParseExact(dateString, "yyyy-MM-dd", CultureInfo.InvariantCulture, DateTimeStyles.None, out date))

{

dateTimePicker1.Value = date;

}

textBox1.Text = dataGridView1.SelectedRows[0].Cells["OrderStatus"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["CustomerID"].Value.ToString();

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadOrders(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void LoadOrders()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT OrderID, OrderDate, OrderStatus, CustomerID FROM Orders";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

private void Orders\_Load(object sender, EventArgs e)

{

LoadOrders();

// Устанавливаем формат даты для txtOrderDate

dateTimePicker1.Format = DateTimePickerFormat.Custom;

dateTimePicker1.CustomFormat = "yyyy-MM-dd";

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

}

**Код формы Категории товаров(ProductCategories)**

public partial class ProductCategories : Form

{

private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private SqlConnection connection;

public ProductCategories()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string typeName = textBox1.Text;

if (!string.IsNullOrWhiteSpace(typeName))

{

string query = "INSERT INTO ProductCategories (CategoryName) VALUES (@CategoryName)";

ExecuteNonQuery(query, ("@CategoryName", typeName));

LoadProductCategories();

MessageBox.Show("Категория успешно добавлена.");

}

else

{

MessageBox.Show("Введите название категории.");

}

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int productTypeID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["CategoryID"].Value);

string query = "DELETE FROM ProductCategories WHERE CategoryID = @CategoryID";

ExecuteNonQuery(query, ("@CategoryID", productTypeID));

}

else

{

MessageBox.Show("Выберите тип продукта для удаления.");

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int productTypeID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["CategoryID"].Value);

string typeName = textBox1.Text;

string query = "UPDATE ProductCategories SET CategoryName = @CategoryName WHERE CategoryID = @CategoryID";

ExecuteNonQuery(query, ("@CategoryName", typeName), ("@CategoryID", productTypeID));

}

else

{

MessageBox.Show("Выберите тип продукта для обновления.");

}

}

private void button4\_Click(object sender, EventArgs e)

{

LoadProductCategories();

textBox1.Text = "";

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadProductCategories(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["CategoryName"].Value.ToString();

}

}

private void ProductCategories\_FormClosed(object sender, FormClosedEventArgs e)

{

Main.Instance.Show();

}

private void ProductCategories\_Load(object sender, EventArgs e)

{

LoadProductCategories();

}

private void LoadProductCategories()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT CategoryID, CategoryName FROM ProductCategories";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

**Код формы Товары(Products)**

public partial class Products : Form

{

private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private SqlConnection connection;

public Products()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

LoadProducts();

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void button1\_Click(object sender, EventArgs e)

{

string name = textBox1.Text;

string description = textBox2.Text;

decimal price = Convert.ToDecimal(textBox3.Text);

int quantity = Convert.ToInt32(textBox4.Text);

int productTypeID = Convert.ToInt32(textBox5.Text);

string query = "INSERT INTO Products (ProductName, ProductDescription, ProductPrice, ProductQuantity, CategoryID) VALUES (@ProductName, @ProductDescription, @ProductPrice, @ProductQuantity, @CategoryID)";

ExecuteNonQuery(query, ("@ProductName", name), ("@ProductDescription", description), ("@ProductPrice", price), ("@ProductQuantity", quantity), ("@CategoryID", productTypeID));

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int productID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["ProductID"].Value);

string query = "DELETE FROM Products WHERE ProductID = @ProductID";

ExecuteNonQuery(query, ("@ProductID", productID));

}

else

{

MessageBox.Show("Выберите продукт для удаления.");

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int productID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["ProductID"].Value);

string name = textBox1.Text;

string description = textBox2.Text;

decimal price = Convert.ToDecimal(textBox3.Text);

int quantity = Convert.ToInt32(textBox4.Text);

int productTypeID = Convert.ToInt32(textBox5.Text);

string query = "UPDATE Products SET ProductName = @ProductName, ProductDescription = @ProductDescription, ProductPrice = @ProductPrice, ProductQuantity = @ProductQuantity, CategoryID = @CategoryID WHERE ProductID = @ProductID";

ExecuteNonQuery(query, ("@ProductName", name), ("@ProductDescription", description), ("@ProductPrice", price), ("@ProductQuantity", quantity), ("@CategoryID", productTypeID), ("@ProductID", productID));

}

else

{

MessageBox.Show("Выберите продукт для обновления.");

}

}

private void button4\_Click(object sender, EventArgs e)

{

LoadProducts();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

textBox4.Text = "";

textBox5.Text = "";

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["ProductName"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["ProductDescription"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["ProductPrice"].Value.ToString();

textBox4.Text = dataGridView1.SelectedRows[0].Cells["ProductQuantity"].Value.ToString();

textBox5.Text = dataGridView1.SelectedRows[0].Cells["CategoryID"].Value.ToString();

}

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

private void LoadProducts()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT ProductID, ProductName, ProductDescription, ProductPrice, ProductQuantity, CategoryID FROM Products";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void Products\_FormClosed(object sender, FormClosedEventArgs e)

{

Main.Instance.Show();

}

}

**Код формы Пользователи(Products)**

public partial class Users : Form

{

private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private SqlConnection connection;

public Users()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

LoadUsers();

}

private void button1\_Click(object sender, EventArgs e)

{

string login = textBox1.Text;

string password = textBox2.Text;

string role = textBox3.Text;

string query = "INSERT INTO Users (Username, PasswordHash, UserRole) VALUES (@Username, @PasswordHash, @UserRole)";

ExecuteNonQuery(query, ("@Username", login), ("@PasswordHash", password), ("@UserRole", role));

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int userID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["UserID"].Value);

string query = "DELETE FROM Users WHERE UserID = @UserID";

ExecuteNonQuery(query, ("@UserID", userID));

}

else

{

MessageBox.Show("Выберите пользователя для удаления.");

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int userID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["UserID"].Value);

string login = textBox1.Text;

string password = textBox2.Text;

string role = textBox3.Text;

string query = "UPDATE Users SET Username = @Username, PasswordHash = @PasswordHash, UserRole = @UserRole WHERE UserID = @UserID";

ExecuteNonQuery(query, ("@Username", login), ("@PasswordHash", password), ("@UserRole", role), ("@UserID", userID));

}

else

{

MessageBox.Show("Выберите пользователя для обновления.");

}

}

private void button4\_Click(object sender, EventArgs e)

{

LoadUsers();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["Username"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["PasswordHash"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["UserRole"].Value.ToString();

}

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

private void Users\_FormClosed(object sender, FormClosedEventArgs e)

{

Main.Instance.Show();

}

private void LoadUsers()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT UserID, Username, PasswordHash, UserRole FROM Users";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadUsers(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

}

**Код формы Поставщики(Vendors)**

public partial class Vendors : Form

{

private string connectionString = @"Data Source= DESKTOP-56CEJQR; Initial catalog=kursacBronin; Integrated Security=True";

private SqlConnection connection;

public Vendors()

{

InitializeComponent();

connection = new SqlConnection(connectionString);

LoadVendors();

}

private void dataGridView1\_SelectionChanged(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

textBox1.Text = dataGridView1.SelectedRows[0].Cells["VendorName"].Value.ToString();

textBox2.Text = dataGridView1.SelectedRows[0].Cells["VendorAddress"].Value.ToString();

textBox3.Text = dataGridView1.SelectedRows[0].Cells["VendorPhone"].Value.ToString();

}

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.AutoSizeMode = DataGridViewAutoSizeColumnMode.Fill;

}

}

private void button1\_Click(object sender, EventArgs e)

{

string name = textBox1.Text;

string address = textBox2.Text;

string phone = textBox3.Text;

if (string.IsNullOrEmpty(textBox1.Text) || string.IsNullOrEmpty(textBox2.Text) || string.IsNullOrEmpty(textBox3.Text))

{

MessageBox.Show("Ошибка, недопустимые значения, заполните все поля!");

}

string query = "INSERT INTO Vendors (VendorName, VendorAddress, VendorPhone) VALUES (@VendorName, @VendorAddress, @VendorPhone)";

ExecuteNonQuery(query, ("@VendorName", name), ("@VendorAddress", address), ("@VendorPhone", phone));

}

private void button3\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int supplierID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["VendorID"].Value);

string query = "DELETE FROM Vendors WHERE VendorID = @VendorID";

ExecuteNonQuery(query, ("@VendorID", supplierID));

}

else

{

MessageBox.Show("Выберите поставщика для удаления.");

}

}

private void button2\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count > 0)

{

int supplierID = Convert.ToInt32(dataGridView1.SelectedRows[0].Cells["VendorID"].Value);

string name = textBox1.Text;

string address = textBox2.Text;

string phone = textBox3.Text;

string query = "UPDATE Vendors SET VendorName = @VendorName, VendorAddress = @VendorAddress, VendorPhone = @VendorPhone WHERE VendorID = @VendorID";

ExecuteNonQuery(query, ("@VendorName", name), ("@VendorAddress", address), ("@VendorPhone", phone), ("@VendorID", supplierID));

}

else

{

MessageBox.Show("Выберите поставщика для обновления.");

}

}

private void button4\_Click(object sender, EventArgs e)

{

LoadVendors();

textBox1.Text = "";

textBox2.Text = "";

textBox3.Text = "";

}

private void ExecuteNonQuery(string query, params (string, object)[] parameters)

{

using (SqlConnection connection = new SqlConnection(connectionString))

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

{

foreach (var parameter in parameters)

{

command.Parameters.AddWithValue(parameter.Item1, parameter.Item2);

}

try

{

connection.Open();

command.ExecuteNonQuery();

MessageBox.Show("Операция успешно выполнена.");

LoadVendors(); // Перезагружаем данные после выполнения операции

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при выполнении операции: " + ex.Message);

}

}

}

private void LoadVendors()

{

string currentRole = UserManager.CurrentUser.Role;

if (currentRole != "Администратор")

{

button1.Hide();

button2.Hide();

button3.Hide();

}

string query = "SELECT VendorID, VendorName, VendorAddress, VendorPhone FROM Vendors";

SqlDataAdapter adapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

private void button5\_Click(object sender, EventArgs e)

{

this.Close();

}

private void Vendors\_FormClosed(object sender, FormClosedEventArgs e)

{

Main.Instance.Show();

}

}

**Код класса Users**

internal class User

{

public string Username { get; set; }

public string Role { get; set; }

public User(string username, string role)

{

this.Username = username;

this.Role = role;

}

}

**Код класса UsersManager**

internal class UserManager

{

public static User CurrentUser { get; private set; }

public static void SetCurrentUser(string username, string role)

{

CurrentUser = new User(username, role);

}

}