СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 2](#_Toc167713745)

[1. СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА СУБД 4](#_Toc167713746)

[1.1. Выбор СУБД 5](#_Toc167713747)

[2. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И ПОСТАНОВКА 7](#_Toc167713748)

[2.1. Исследование предметной области 7](#_Toc167713749)

[2.2. Постановка задачи 8](#_Toc167713750)

[3. ПЛАНИРОВАНИЕ РАЗРАБОТКИ БД 9](#_Toc167713751)

[3.1. Определение требований к системе 9](#_Toc167713752)

[3.2. Сбор и анализ требований пользователей 9](#_Toc167713753)

[3.3. Концептуальная модель базы данных 10](#_Toc167713754)

[3.4. Преобразование концептуальной модели данных в реляционную модель данных 14](#_Toc167713755)

[4. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ 20](#_Toc167713756)

[4.1. Разграничение прав доступа 20](#_Toc167713757)

[4.2. Реализация запросов в Microsoft SQL Server 30](#_Toc167713758)

[5. РАЗРАБОТКА ПРИЛОЖЕНИЯ 34](#_Toc167713759)

[5.1. Определение дизайна и разработка графического интерфейса 34](#_Toc167713760)

[5.2. Графический интерфейс приложения предприятия 35](#_Toc167713761)

[5.3. Выполнение механизмов приложения 44](#_Toc167713762)

[5.4. Реализация запросов 59](#_Toc167713763)

[ЗАКЛЮЧЕНИЕ 64](#_Toc167713764)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 65](#_Toc167713765)

[ПРИЛОЖЕНИЕ А 66](#_Toc167713766)

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. 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.

1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ
   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.
  1. 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 редактирование 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.

1. 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.

* 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.

* 1. 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.

* 1. 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, связанных 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 завершенный).
* 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.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

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.

* 1. 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 |  |
| КоличествоНаСкладе | int | Нет |  |
| id\_поставщика | int | Да | Внешний |

Заполнение созданных таблиц данными (Рисунки 2-5).



Рисунок 2 – Данные таблицы Заказы

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 3 – Данные таблицы Клиенты

Изображение выглядит как текст, снимок экрана, Шрифт, линия

Автоматически созданное описание

Рисунок 4– Данные таблицы Поставщики

Изображение выглядит как текст, число, меню, Шрифт

Автоматически созданное описание

Рисунок 5 – Данные таблицы Товары

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

Изображение выглядит как текст, линия, Шрифт, снимок экрана

Автоматически созданное описание

Рисунок 6 – Реляционная модель БД разрабатываемой системы

1. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ
   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).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 7 – Создание имени для входа

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 9 – Присвоение роли Администратору для управления всей базы данных

Подобным образом создаётся роль Персонал (Рисунок 10).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 11 – Добавление прав для Персонала

Аналогичным образом создаётся Клиент (Рисунок 12).

Изображение выглядит как текст, снимок экрана, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 12 – Создание Клиента

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

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 13 – Добавление прав для Клиента

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

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 15 – Подключение к БД как Администратор

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

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 16 – Вид от лица Администратора

Проверка прав пользователя Персонал (Рисунки 17-20).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 17 – Подключение к БД как Персонал

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 18 – Вид от лица Персонала

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Автоматически созданное описание

Рисунок 19 – Запрос на добавление нового товара

Изображение выглядит как текст, число, меню

Автоматически созданное описание

Рисунок 20 – Успешно добавленный товар с помощью запроса

Проверка прав пользователя Клиента (Рисунки 21-24).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 21 – Подключение к БД как Клиент

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 22 – Вид от лица Клиента

Изображение выглядит как текст, Шрифт, программное обеспечение, веб-страница

Автоматически созданное описание

Рисунок 23 – Запрос на добавление записи в недоступной таблице для редактирования

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 24 – Сообщение с ошибкой при удалении строчки через контекстное меню

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

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 25 – Обновление таблицы Клиенты

* 1. 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).

Изображение выглядит как текст, электроника, снимок экрана, дисплей

Автоматически созданное описание

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).

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 27 – Создание представления с помощью запроса

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

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).

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Рисунок 29 – Результат работы транзакции

1. 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.

* 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.

* 1. 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).

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 30 – Элементы приложения

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

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 31 – Начальное окно входа

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, Шрифт

Автоматически созданное описание

Рисунок 32 – Ошибка при входе в систему

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

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 33 – Регистрация нового пользователя

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 34 – Успешное прохождение регистрации нового пользователя

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

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

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

Изображение выглядит как текст, число, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 36 – Авторизация в систему как Клиент

Изображение выглядит как текст, число, программное обеспечение, снимок экрана

Автоматически созданное описание

Рисунок 37 – Авторизация в систему как Персонал

Изображение выглядит как текст, число, программное обеспечение, линия

Автоматически созданное описание

Рисунок 38 – Авторизация в систему как Администратор

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

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 39 – Таблица Товары

Изображение выглядит как снимок экрана, текст, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 40 – Таблица Заказы

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 41 – Таблица Поставщики

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 42 – Таблица Пользователи

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 43 – Окно добавления нового товара

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 44 – Окно редактирования существующего товара

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 45 – Окно добавления нового заказа

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 46 – Окно редактирования существующего заказа

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 47 – Окно добавления нового поставщика

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 48 – Окно редактирования существующего поставщика

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 49 – Окно добавления нового пользователя

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 50 – Окно редактирования существующего пользователя

* 1. 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).

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

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).

Изображение выглядит как текст, линия, Шрифт, число

Автоматически созданное описание

Рисунок 53 – Созданный активный заказ в таблице Заказы

Изображение выглядит как снимок экрана, текст, линия, Шрифт

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, программное обеспечение, число

Автоматически созданное описание

Рисунок 57 – Добавление нового товара

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Рисунок 58 – Результат добавления нового товара

Изображение выглядит как текст, снимок экрана, программное обеспечение, Операционная система

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 60 – Окно удаления товара

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 61 – Окно редактирования товара

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 63 – Добавления нового заказа

Изображение выглядит как текст, программное обеспечение, число, Значок на компьютере

Автоматически созданное описание

Рисунок 64 – Результат добавления нового заказа

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

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

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

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 66 – Окно удаления заказа

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 67 – Окно редактирования заказа

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

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 доступы таблицы поставщики и пользователи, в которой хранятся данные всех клиентов и персонала (Рисунок 69).

Изображение выглядит как текст, число, программное обеспечение, снимок экрана

Автоматически созданное описание

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).

Изображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Рисунок 70 – Добавление нового поставщика

Изображение выглядит как текст, программное обеспечение, Значок на компьютере, число

Автоматически созданное описание

Рисунок 71 – Результат добавления нового поставщика

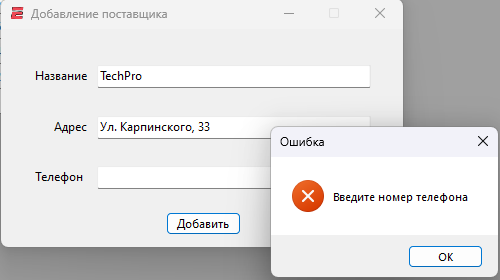


Рисунок 72 – Сообщение с ошибкой при не заполнении какого-либо поля

Изображение выглядит как текст, снимок экрана, число, дисплей

Автоматически созданное описание

Рисунок 73 – Добавление нового пользователя

Изображение выглядит как текст, снимок экрана, число, линия

Автоматически созданное описание

Рисунок 74 – Результат добавления нового пользователя

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

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

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

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок 76 – Окно удаления поставщика

Изображение выглядит как текст, снимок экрана, Шрифт, число

Автоматически созданное описание

Рисунок 77 – Окно редактирования поставщика

Изображение выглядит как текст, снимок экрана, дисплей, число

Автоматически созданное описание

Рисунок 78 – Сообщение с ошибкой удаления или редактирования поставщика

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

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

Рисунок 79 – Окно удаления пользователя

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 80 – Окно редактирования пользователя

Изображение выглядит как текст, снимок экрана, число, Шрифт

Автоматически созданное описание

Рисунок 81 – Сообщение с ошибкой удаления или редактирования пользователя

* 1. 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.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

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.

Изображение выглядит как текст, снимок экрана, число, программное обеспечение

Автоматически созданное описание

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 C# – 2022. – 769 с.
2. Фленов М. Е. Библия C# – Издательство BHV, 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 C#. 4-е изд., Джеффри Рихтер, 2024, 896 с.
6. Трунин В. Путь программиста T-SQL – 2020. – 204 с.
7. Алексей В. Программирование на C# 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. [C# и Windows Forms | Первое приложение с .NET CLI (metanit.com)](https://metanit.com/sharp/windowsforms/1.3.php?ysclid=lwhy4t7l5464092168)
10. [Все операции с БД в графическом приложении в C# (metanit.com)](https://metanit.com/sharp/adonet/3.5.php?ysclid=lwhy7uzy2d959569040)

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\_suppler = (int)selectedRow.Cells["id\_поставщика"].Value;

AddSuppliersForm addSuppliersForm = new AddSuppliersForm(this, id\_suppler);

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 удаленеи.");

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 dataGridView\_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.");

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());

}

}

}