**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: 32919/4

**КУРСОВОЙ ПРОЕКТ**

**на тему:**

**«Проектирование и реализация базы данных кофейни»**

Студент Алексеева Полина Александровна

(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** Алексеева П.А. группа\_\_32919/4\_\_\_\_

(фамилия, инициалы)

**Тема:** **Проектирование и реализация** **базы данных кофейни**

**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 П.А. Алексеева**

(подпись, дата) (инициалы, фамилия)

**содержание**

[ВВЕДЕНИЕ 4](#_Toc168999627)

[1. СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА СУБД 6](#_Toc168999628)

[1.1. Сравнение СУБД 6](#_Toc168999629)

[1.2. Аналоги 13](#_Toc168999630)

[1.3. Выбор СУБД 13](#_Toc168999631)

[2. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ 14](#_Toc168999632)

[2.1. Описание предметной области 14](#_Toc168999633)

[2.2. Диаграмма прецедентов использования 15](#_Toc168999634)

[2.3. Инфологическая модель 17](#_Toc168999635)

[2.4. Разработка таблиц в Microsoft SQL Server 18](#_Toc168999636)

[2.5. Даталогическая модель и связи между таблицами 20](#_Toc168999637)

[2.6. Создание и заполнение базы данных 21](#_Toc168999638)

[2.7. Реализация запросов в Microsoft SQL Server 24](#_Toc168999639)

[3. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА БАЗЫ ДАННЫХ 29](#_Toc168999640)

[3.1. Разграничение прав доступа 29](#_Toc168999641)

[ЗАКЛЮЧЕНИЕ 38](#_Toc168999642)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 39](#_Toc168999643)

[ПРИЛОЖЕНИЕ А 40](#_Toc168999644)

**ВВЕДЕНИЕ**

На сегодняшний день информационные технологии проникают практически во все сферы жизни. Их внедрение ориентировано на ускорение и увеличение производительности труда. Они упрощают обработку документов, ускоряют поиск нужных данных и дают возможность сотрудникам незамедлительно решать определенные проблемы.

Цель этого курсового проекта заключается в создании базы данных, позволяющей автоматизировать работу кофейни и управлять различными аспектами её деятельности.

В кофейню ежедневно приходят десятки посетителей, и успешная работа заведения зависит от эффективного управления заказами, сотрудниками и запасами. Разрабатываемая база данных должна автоматизировать процессы в кофейне и выполнять такие задачи, как регистрация заказов и сотрудников, возможность просматривать информацию о них, управление инвентарем, а также проведение анализа продаж и работы персонала.

Кофейня является развивающимся и перспективным объектом с молодой амбициозной командой. В приоритете для коллектива, который организует работу заведения, — обеспечение комфортных условий для клиентов, высокий уровень обслуживания, вкусные и качественные напитки и блюда, а также создание уютной и приятной атмосферы.

Посетители могут наслаждаться этим удивительным местом в любое время года, наслаждаясь разнообразными кофейными напитками, свежей выпечкой и приятной обстановкой. Летом они могут посидеть на террасе, а в остальное время года наслаждаться уютной атмосферой внутри кофейни. В кофейне также проводятся различные мероприятия и мастер-классы, направленные на развитие кулинарных и баристских навыков.

Таким образом, разработка базы данных для кофейни направлена на оптимизацию её работы и улучшение качества обслуживания клиентов, что является залогом успешного и стабильного развития заведения.

1. **СРАВНИТЕЛЬНАЯ ХАРАКТЕРИСТИКА СУБД**
   1. **Сравнение СУБД**

База данных — именованная совокупность данных, отражающая состояние объектов и их отношений в рассматриваемой предметной области.

БД можно классифицировать по характеру информации: фактографические и документальные.

Фактографическая база данных содержит информацию, относящуюся непосредственно к предметной области. Предоставляется в виде фактов, например, дата производства продукта, список зачисленных студентов т. д. В ответ на запрос пользователю выдается требуемая информация об интересующем его объекте или сообщение о том, что искомая информация отсутствует в БД.

В документальных БД единицей хранения является какой-либо документ (например, текст закона или статьи), и пользователю в ответ на его запрос выдается либо ссылка на документ, либо сам документ.

Система управления базами данных (СУБД) — комплекс программных и аппаратных средств, предназначенных для создания, ведения и использования баз данных многими пользователями.

СУБД различают по используемой модели данных. К числу классических относят:

* иерархическая;
* сетевая;
* реляционная;
* постреляционная;
* многомерная;
* объектно— ориентированная.

Иерархическая модель данных (рис 1) представляет собой упорядоченный граф. При описании структуры каждый из типов дерева состоит из корневого дерева и подчиненных типов записей. Основное правило контроля целостности: потомок не может существовать без родителя, а у некоторых родителей может не быть потомков.

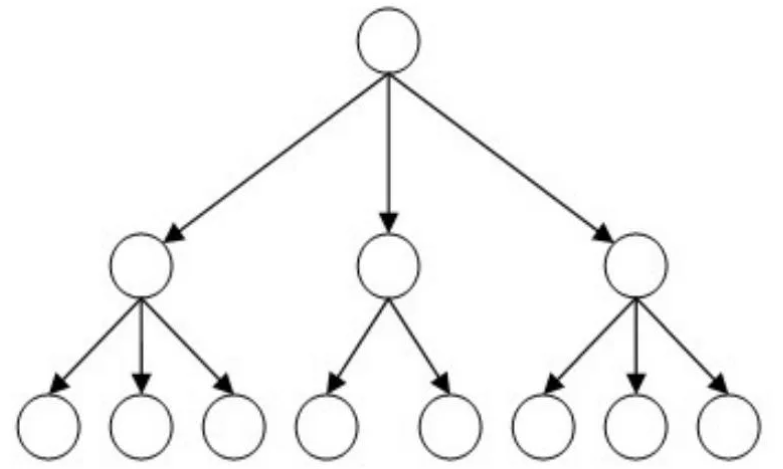


Рисунок 1 Иерархическая модель данных

Сетевая модель данных (рис 2)позволяет отображать разнообразные связи элементов данных в виде произвольного графа. Для описания схем сетевой базы данных используется 2 группы типа запись и связь.

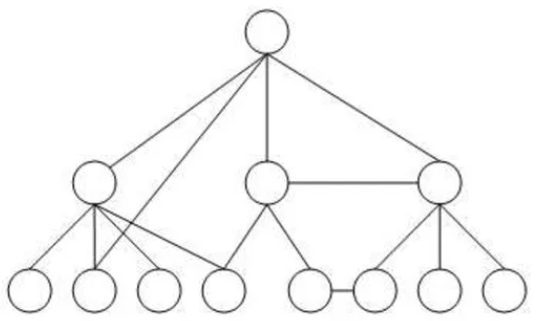


Рисунок 2 Сетевая модель данных

Реляционная модель данных (рис 3)состоит из взаимосвязанных между собой таблиц (отношений), имеет двумерную структуру. Строки называются картежами (так как значения в строке зависит от значения первичного ключа), а столбцы — атрибутами.

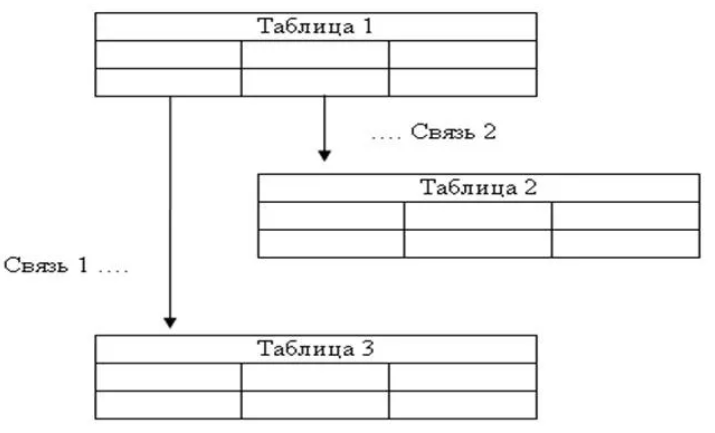


Рисунок 3 Реляционнаямодель данных

Постреляционная модель данных (рис 4)— расширенная табличная модель, снимающая ограничения неделимости данных, то есть допускаем многозначные поля.

Многомерная модель данныхпозволяет клиентам задавать аналитические вопросы, связанные с рынком или бизнес-тенденциями, в отличие от реляционных баз данных, которые позволяют клиентам получать доступ к данным в форме запросов.

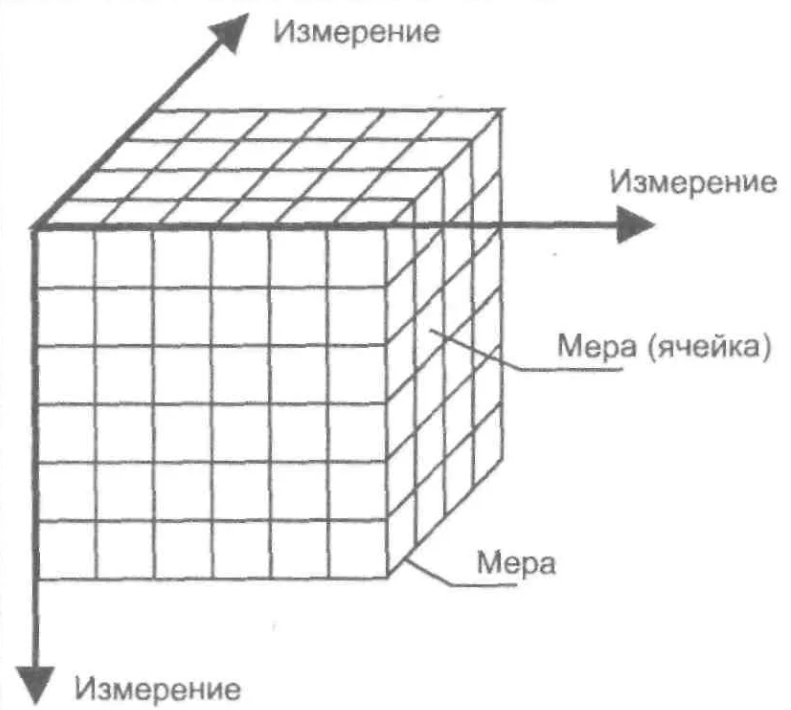


Рисунок 4 Многомернаямодель данных

Объектно—ориентированная модель данных (рис 5) подразумевает хранение данных и методов их обработки в одном объекте (таблице). Данная модель базируется на принципах объектно-ориентированного программирования (ООП) и реляционной модели.

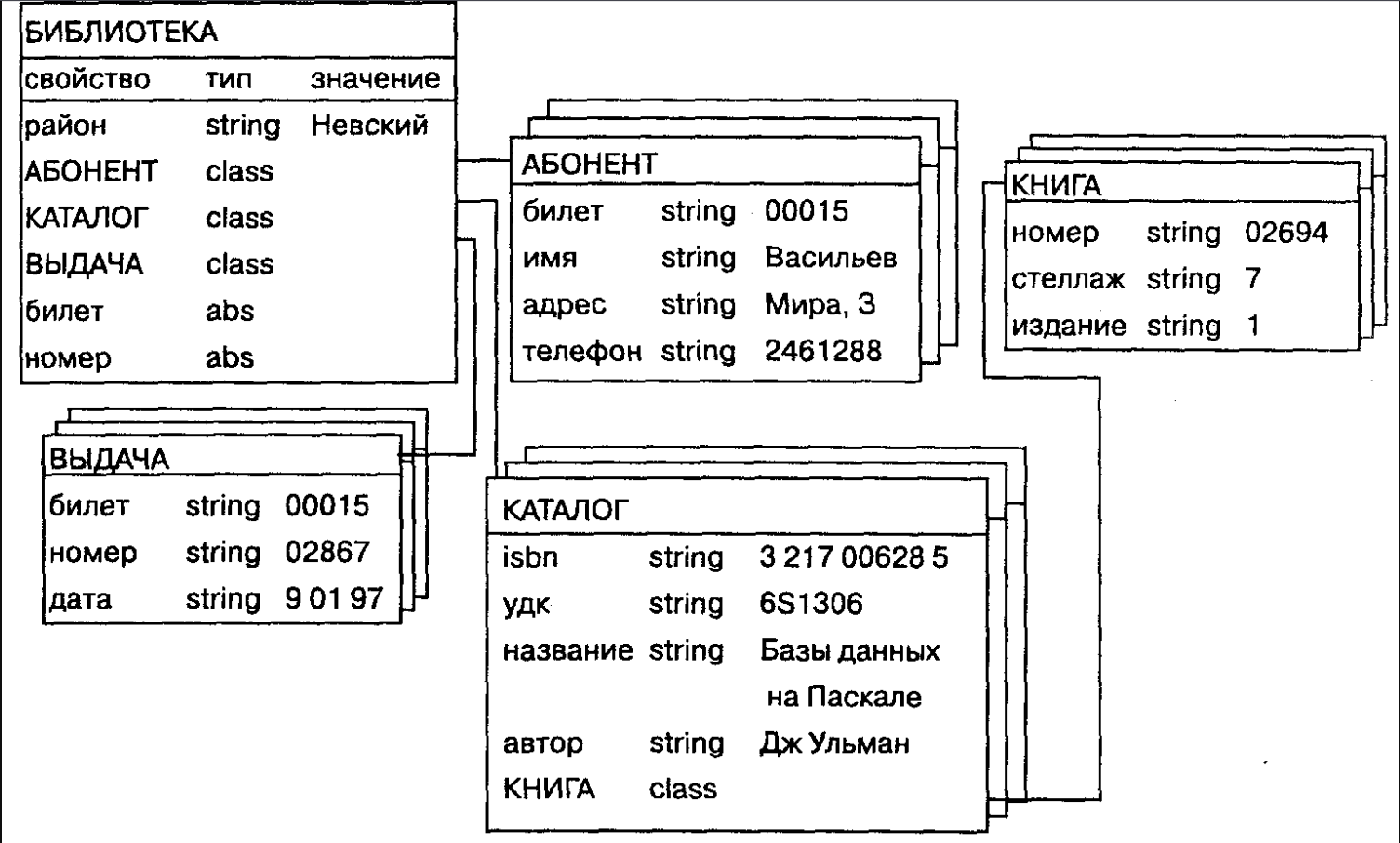


Рисунок 5 Объектно—ориентированная модель данных

MySQL— свободная реляционная система управления базами данных. MySQL является решением для малых и средних приложений. Обычно эта СУБД используется в качестве сервера, к которому обращаются локальные или удалённые клиенты, однако в дистрибутив входит библиотека внутреннего сервера, позволяющая включать MySQL в автономные программы. MySQL имеет API и коннекторы для языков Delphi, C, C++, Эйфель, Java, Лисп, Perl, PHP, Python, Ruby, Smalltalk, Компонентный Паскаль, Tcl и Lua, библиотеки для языков платформы .NET, а также обеспечивает поддержку для ODBC посредством ODBC-драйвера MyODBC.

MariaDB — ответвление от системы управления базами данных MySQL. Толчком к созданию стала необходимость обеспечения свободного статуса СУБД, в противовес политике лицензирования MySQL компанией Oracle. Система лицензирования MariaDB обязывает участников, желающих добавить свой код в основную ветку СУБД, обмениваться своими авторскими правами с MariaDB Foundation для охраны лицензии и возможности создавать критические исправления для MySQL. MariaDB намерен поддерживать высокую совместимость с MySQL, обеспечивая точное соответствие с API и командами MySQL. В состав MariaDB включена подсистема хранения данных XtraDB для возможности замены InnoDB, как основной подсистемы хранения.

Oracle Database или Oracle RDBMS — объектно-реляционная система управления базами данных компании Oracle. Это база данных, обычно используемая для выполнения рабочих нагрузок онлайн-обработки транзакций (OLTP), хранилища данных (DW) и смешанных (OLTP & DW) баз данных. Oracle Database доступна нескольким поставщикам услуг непосредственно, в облаке или в виде гибридной облачной установки. Она может быть запущена на сторонних серверах, а также на оборудовании Oracle.

PostgreSQL — свободная объектно-реляционная система управления базами данных (СУБД). Она поддерживает большую часть стандарта SQL и предлагает множество современных функций:

* сложные запросы;
* внешние ключи;
* триггеры;
* изменяемые представления;
* транзакционная целостность;
* многоверсионность

Кроме того, пользователи могут всячески расширять возможности PostgreSQL, например создавая свои:

* типы данных;
* функции;
* операторы;
* агрегатные функции;
* методы индексирования;
* процедурные языки.

А благодаря свободной лицензии, PostgreSQL разрешается бесплатно использовать, изменять и распространять всем и для любых целей — личных, коммерческих или учебных.

SQL Server Management Studio (SSMS) — утилита для Microsoft SQL Server для конфигурирования, управления и администрирования компонентов базы данных. Данная утилита содержит редактор скриптов и графическую программу, которая работает с объектами и настройками сервера. Главным инструментом SQL Server Management Studio является Object Explorer, который позволяет пользователю просматривать, извлекать объекты сервера, а также управлять ими.

Далее приводим таблицу сравнения этих СУБД:

Таблица 1. Сравнение популярных СУБД

| **Наименование** | **Тип** | **Операционная система** | **Достоинства** | **Недостатки** |
| --- | --- | --- | --- | --- |
| 1.MySQL | реляционная | Linux, Windows, macOS, IRIX | возможности бесплатной версии хватит для решения всех или почти всех задач маленькой компании;  простой синтаксис и умеренная сложность;  совместимость с облачными решениями. | проблемы масштабируемости;  частично открыт исходный код;  ограниченное соответствие стандартам SQL (MySQL не поддерживает некоторые стандартные функции SQL. С другой стороны, MySQL имеет некоторые расширения и отдельные функции); |
| 2. MariaDB | реляционная | Linux, Windows, macOS, IRIX | шифрование;  широкая функциональность;  высокая производительность. | недостаточно стабильна и надёжна;  пробелы между версиями обновлений MySQL и MariaDB. |
| 3.Oracle | объектно-реляционная | Linux, Microsoft Windows, Solaris, HP, IBM, Mac OS | инновации для ежедневного рабочего процесса;  сильная техническая поддержка и документация;  большая вместимость. | высокая стоимость;  ресурсоемкая технология;  высокий порог вхождения. |
| 4.PostgreSQL | объектно-реляционная | Linux, Microsoft Windows, iOS, Android | надёжность;  отличная масштабируемость;  поддержка пользовательских типов данных;  легко интегрируемые сторонние инструменты;  поддержка с открытым исходным кодом и на основе сообщества;  высокопроизводительные и надёжные механизмы транзакций и репликации;  расширяемая система встроенных языков программирования. | противоречивая документация;  инструменты отчетности и аудита оставляют желать лучшего;  снижение производительности со временем. |
| 5.Microsoft SQL Server | реляционная | Windows | разнообразие версий;  комплексное решение для обработки бизнес-данных;  внушительная документация;  поддержка облачных решений. | привязка к платформе Microsoft Windows;  высокая стоимость;  высокая требовательность к аппаратным ресурсам;  неясные и плавающие условия лицензии;  сложный процесс настройки;  восстановление данных после аварийного отключения питания обязательно требует участия специалиста |

## 

## Аналоги

В открытом доступе не нашлось адаптированных СУБД для кофейни, но есть только документация. Проанализировав, заметим, что вся она написана для Access, вследствие чего, проект будет чем-то новым и эксклюзивным.

## Выбор СУБД

Для своего курсового проекта выбираем СУБД Microsoft SQL Server, так как начали изучать эту среду только в этом году и хотелось бы углубить свои знания. Плюсом является то, что при возникновении проблем с написанием работы смогу обратиться к документации (как отмечалось ранее, она внушительная). Так же работаем на Microsoft Windows, поэтому привязка к платформе не страшна.

1. РЕАЛИЗАЦИЯ БАЗЫ ДАННЫХ

## Описание предметной области

Под кофейней обычно понимается пространство, где люди могут наслаждаться напитками и едой, а также отдыхать и общаться. Ряд исследователей рассматривают кофейню как важный социальный институт, который предоставляет возможность обновления образа жизни, расширения кругозора, общения и проведения досуга.

Кофейня — это общественное заведение, создаваемое с целью удовлетворения потребностей клиентов в качественных напитках и еде, обеспечивая при этом уютную и комфортную атмосферу. Целью кофейни является предоставление высококачественного сервиса, создание благоприятных условий для отдыха и общения клиентов.

Основными задачами кофейни являются:

* организация комфортного и приятного отдыха для посетителей;
* обеспечение высокого уровня обслуживания и качества продукции;
* создание условий для личностного и профессионального роста сотрудников, их обучения и развития навыков;
* обеспечение клиентов качественным питанием и разнообразными напитками;
* поддержание чистоты и уюта в зале для посетителей и на рабочем месте сотрудников;
* воспитание культуры потребления кофе и других напитков, привлечение клиентов к изучению и дегустации различных видов кофе;
* создание условий для проведения мероприятий, мастер-классов, дегустаций и других событий, способствующих развитию и привлечению клиентов.

Каждая кофейня имеет свой штат сотрудников. Бариста — специалист, отвечающий за приготовление кофе и других напитков, а также за обслуживание клиентов. Он находится в зале, взаимодействует с клиентами, помогает им выбрать напиток, готовит и подает заказ, становясь настоящим профессионалом своего дела. Также на него ложится большая ответственность за поддержание высокого уровня сервиса и качества продукции.

Таким образом, создание и управление базой данных для кофейни направлено на оптимизацию всех процессов внутри заведения. Это включает в себя регистрацию клиентов и сотрудников, возможность просмотра информации о них, управление заказами и запасами, проведение анализа продаж и работы персонала. Это позволит повысить эффективность работы кофейни, улучшить качество обслуживания клиентов и создать приятную атмосферу для всех посетителей.

## Диаграмма прецедентов использования

Прецеденты Клиент:

1. **Просмотр меню**: Клиент может просматривать все доступные позиции меню.
2. **Поиск по меню**: Клиент может искать конкретные позиции в меню.
3. **Заказ напитков и блюд**: Клиент может сделать заказ.

Прецеденты Администратора:

1. **Добавление сотрудников**: Администратор может добавлять новых сотрудников в систему.
2. **Обновление информации о сотрудниках**: Администратор может обновлять информацию о текущих сотрудниках.
3. **Удаление сотрудников**: Администратор может удалять сотрудников из системы.
4. **Добавление позиций меню:** Администратор может добавлять новые позиции в меню.
5. **Обновление позиций меню:** Администратор может обновлять информацию о текущих позициях меню.
6. **Удаление позиций меню:** Администратор может удалять позиции из меню.

Эта диаграмма прецедентов помогает визуализировать функциональные требования системы и понимание того, какие действия могут выполнять клиенты и администраторы.

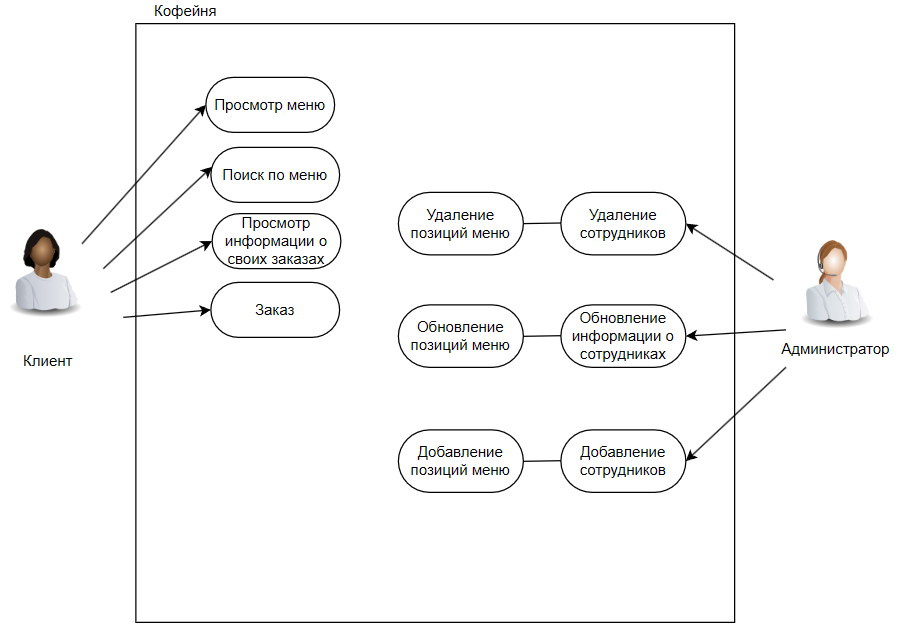


Рисунок 6 – Диаграмма предцендентов

## Инфологическая модель

Эта модель представляет собой базу данных, которая позволяет эффективно управлять информацией о клиентах, сотрудниках, меню, заказах, поставщиках и поставках в кофейне.

«Клиенты» имеют следующие атрибуты: Индивидуальный номер (client\_id), Имя, Фамилия, Отчество, email, Телефон.

Сущность «Сотрудники» имеют следующие атрибуты: Индивидуальный номер (employee\_id), Имя, Фамилия, Должность, email, Телефон.

Атрибутами сущности «Меню» являются: Индивидуальный номер (menu\_id), Название, Описание и Цена.

«Заказы» имеют только два атрибута: Индивидуальный номер (order\_id) , client\_id, employee\_id, Дата и время, Общая сумма.

**«Заказанные позиции»** имеют следующие атрибуты: индивидуальный номер (ordered\_item\_id), order\_id, menu\_id, количество, цена за единицу.

Описание предметной области, выполненное с использованием специальных языковых средств представлено на рисунке 7.

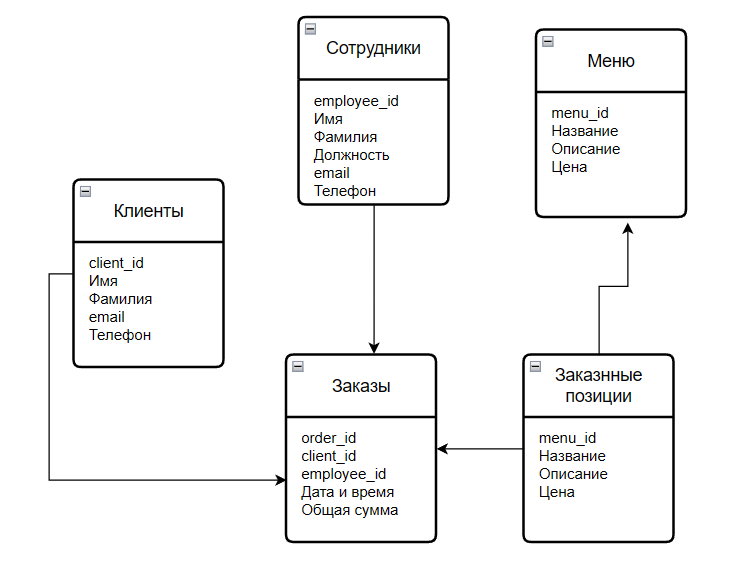


Рисунок 7 - Инфологическая модель

## Разработка таблиц в Microsoft SQL Server

Для начала необходимо создать базу данных «Кофейни» в Microsoft SQL Server. После ее создания перейдем к добавлению таблиц, в которых в дальнейшем будет записана вся информация.

Таблица «Клиенты» представлена на рисунке 8. Первичным ключом является атрибут «client\_id».

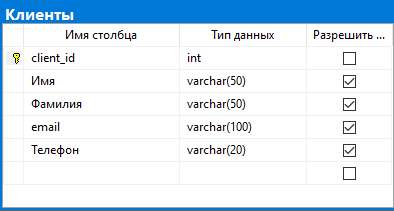


Рисунок 8 - Таблица "Клиенты"

Таблица «Сотрудники» (Рисунок 9). Первичный ключ – employee\_id.

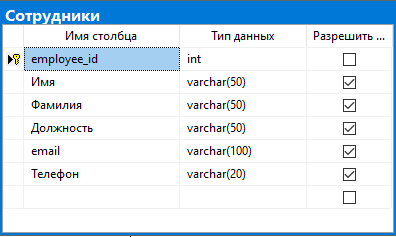


Рисунок 9 - Таблица "Сотрудники"

Таблица «Заказы» (Рисунок 10). Первичный ключом является атрибут «order\_id».

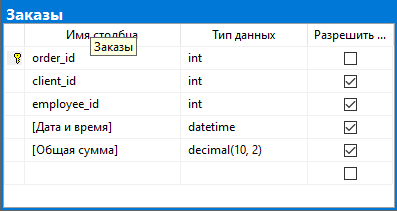


Рисунок 10 - Таблица "Заказы"

Таблица «Меню» представлена на рисунке 11. Первичный ключ – menu\_id.

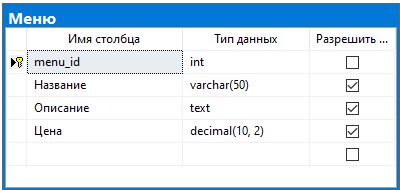


Рисунок 11 - Таблица "Меню"

Таблица «Заказные позиции» (Рисунок 12). Первичный ключ – order\_item\_id.

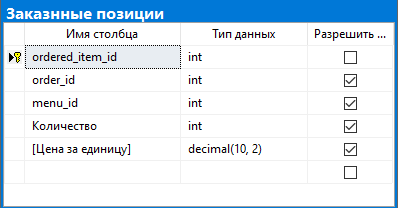


Рисунок 12 - Таблица "Заказные позиции"

## Даталогическая модель и связи между таблицами

После создания таблиц приступаем к созданию диаграммы базы данных «Кофейня», в которой настроим связи между сущностями (Рисунок 13).

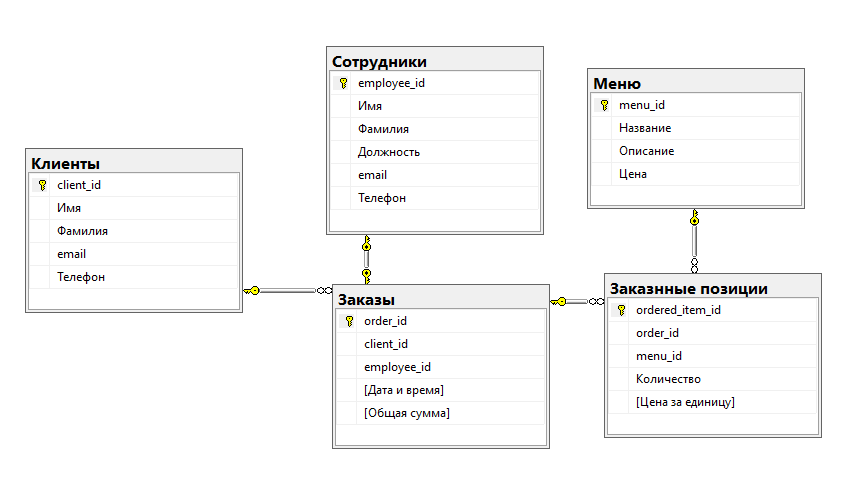


Рисунок 13 - Даталогическая модель

## **Создание и заполнение базы данных**

Заполнение таблицы «Клиенты»:

Запрос на заполнение таблицы представлен на рисунке 14. Результат выполнения запроса представлен на рисунке 15.

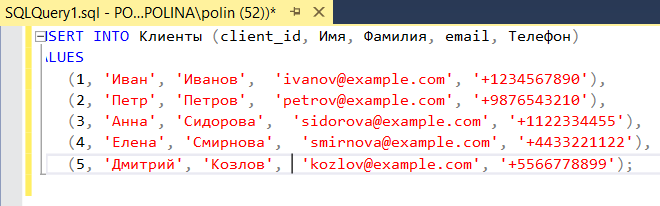


Рисунок 14 – Запрос

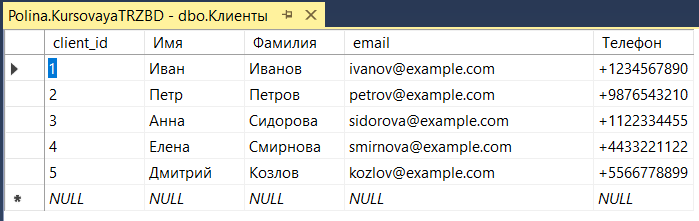


Рисунок 15 – Заполненная таблица

Заполнение таблицы «Меню»:

Запрос на заполнение таблицы представлен на рисунке 16. Результат выполнения запроса представлен на рисунке 17.

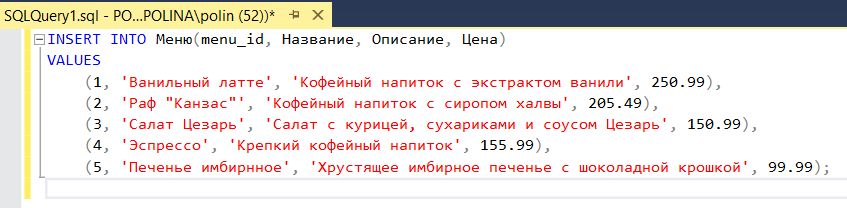


Рисунок 16 – Запрос

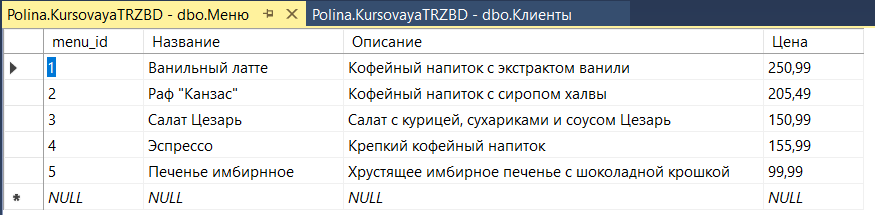


Рисунок 17 – Заполненная таблица

Заполнение таблицы «Сотрудники»:

Запрос на заполнение таблицы представлен на рисунке 18. Результат выполнения запроса представлен на рисунке 19.

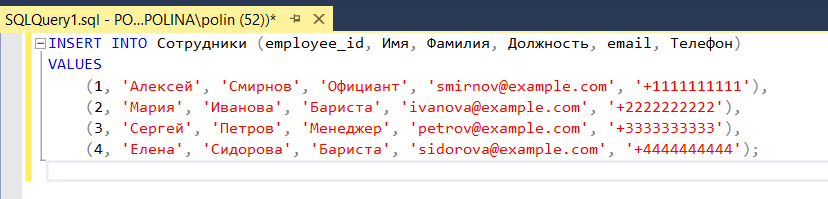


Рисунок 18 – Запрос

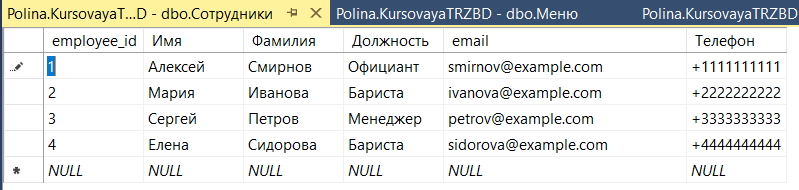


Рисунок 19 – Заполненная таблица

Заполнение таблицы «Заказные позиции»:

Запрос на заполнение таблицы представлен на рисунке 20. Результат выполнения запроса представлен на рисунке 21.

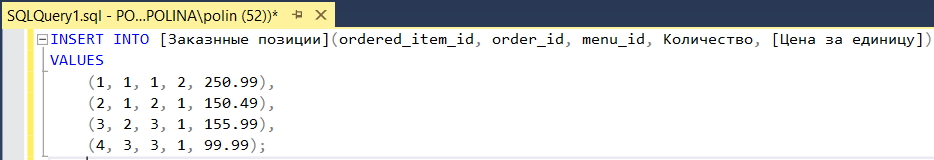


Рисунок 20 – Запрос

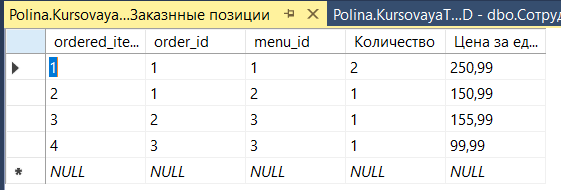


Рисунок 21 – Заполненная таблица

Заполнение таблицы «Заказы»:

Запрос на заполнение таблицы представлен на рисунке 22. Результат выполнения запроса представлен на рисунке 23.

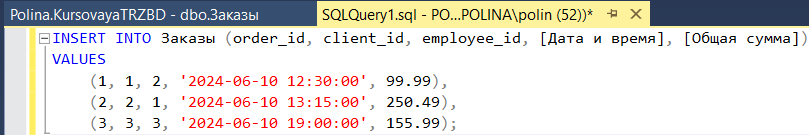


Рисунок 22 – Запрос

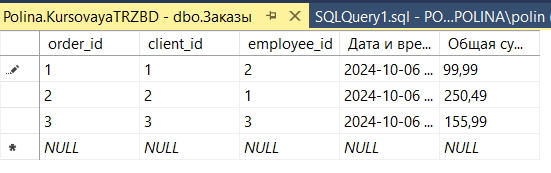


Рисунок 23 – Заполненная таблица

## Реализация запросов в Microsoft SQL Server

* Получение списка меню.

Текст запроса:

SELECT \* FROM Меню

Результат запроса представлен на рисунке 24.

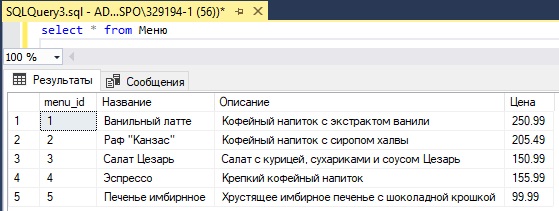


Рисунок 25 - Меню

* Получение списка сотрудников определенного имени (Имя – «Алексей»).

Текст запроса:

SELECT \* FROM Сотрудники

WHERE Имя = 'Алексей';

Результат запроса представлен на рисунке 26.

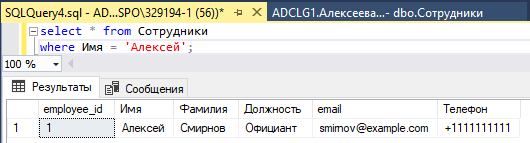


Рисунок 26 – Сотрудник с именем Алексей

* Добавление нового продукта в меню.

Текст запроса:

INSERT INTO Меню (menu\_id, Название, Описание, Цена)

VALUES

(6, 'Матча', 'Зеленый чай с молоком', 315.99);

select \* from Меню

where menu\_id = 6;

Результат запроса представлен на рисунке 27.

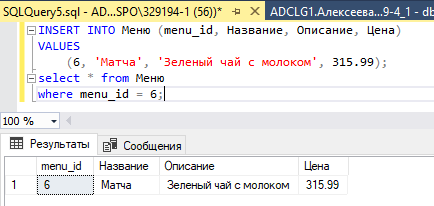


Рисунок 27 - Добавление нового продукта

* Удаление позиции меню из таблицы «Меню», где menu\_id = 6.

Текст запроса:

DELETE FROM Меню

WHERE menu\_id = 6;

Результат запроса представлен на рисунках 28 - 29.

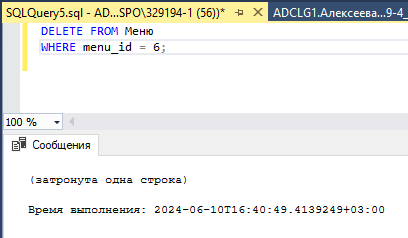


Рисунок 28 - Удаление позиции меню

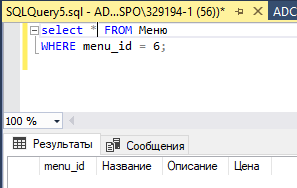


Рисунок 29 - Вывод продукта с menu\_id "6"

* Удаление клиента по его номеру и удаление его заказов, где client\_id = 5;

Текст запроса:

BEGIN TRANSACTION;

-- Объявляем переменную client\_to\_delete

DECLARE @client\_to\_delete INT;

SET @client\_to\_delete = 5;

-- Удаляем все заказы, связанные с клиентом

DELETE FROM Заказы WHERE client\_id = @client\_to\_delete;

-- Удаляем клиента

DELETE FROM Клиенты WHERE client\_id = @client\_to\_delete;

COMMIT;

Результат запроса представлен на рисунках 30 - 31.

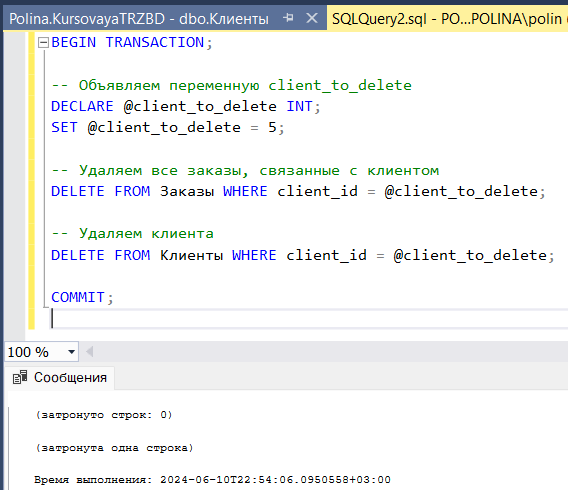


Рисунок 30 – Удаление клиента

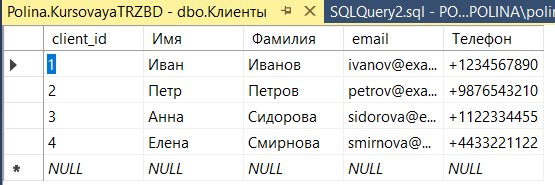


Рисунок 31 – Результат запроса

1. РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА БАЗЫ ДАННЫХ

## Разграничение прав доступа

В созданном приложении права доступа делятся на две роли: роль Администратора и роль Пользователя.

Пользователь, имея свои права, может просматривать каждую вкладку приложения и пользоваться поиском на странице «Меню», то есть он может находить нужную ему информацию.

Права администратора включают в себя права пользователя, но помимо этого он имеет и другие права. Администратор может: удалять, изменять и создавать записи для таблицы «Меню» и другие, а также сохранять и обновлять данные таблицы.

Для того чтобы разграничить права доступа для приложения, понадобилось создать новую таблицу «register» в базу данных «Кофейня». Таблица «register» представлена на рисунках 32-33.

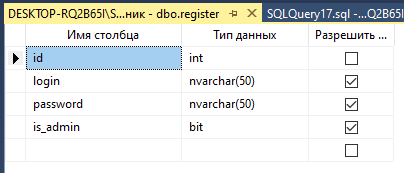


Рисунок 32 - Таблица "register"

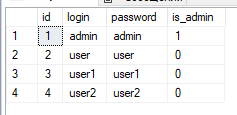


Рисунок 33 - Данные таблицы "register"

Демонстрация роли пользователя представлена на рисунках 34 – 35.

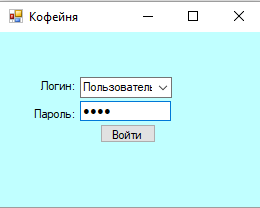


Рисунок 34 - Вход через пользователя

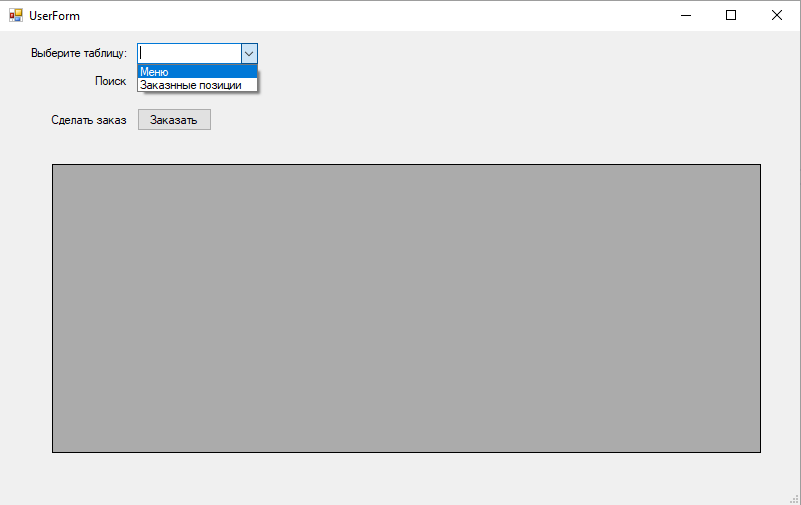


Рисунок 35 – Демонстрация формы

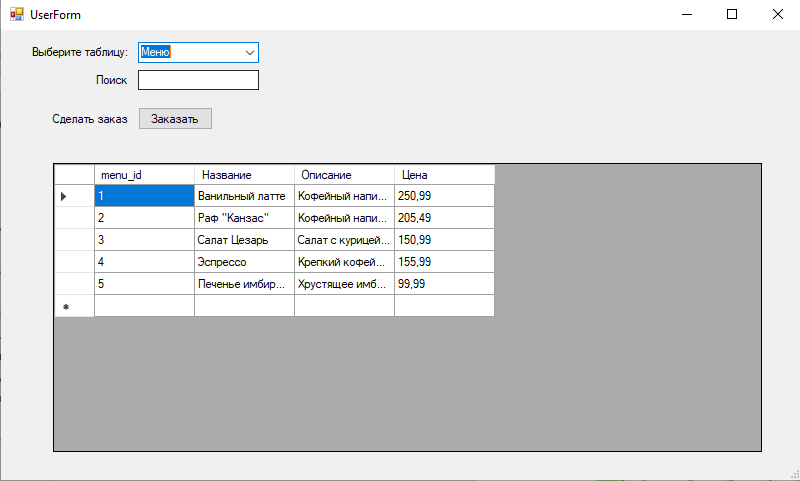


Рисунок 36 – Таблица Меню

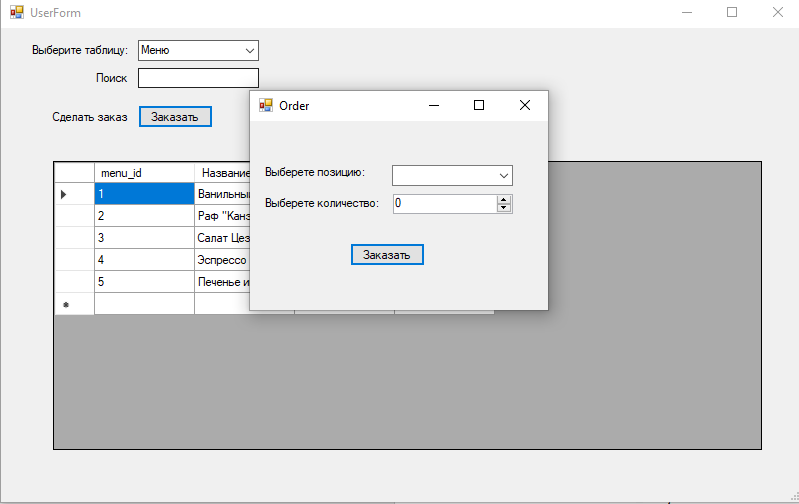


Рисунок 37 – Возможность заказа

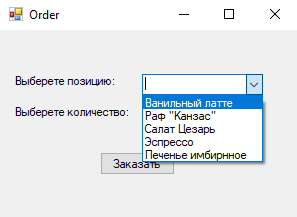


Рисунок 38 – Выбор позиций

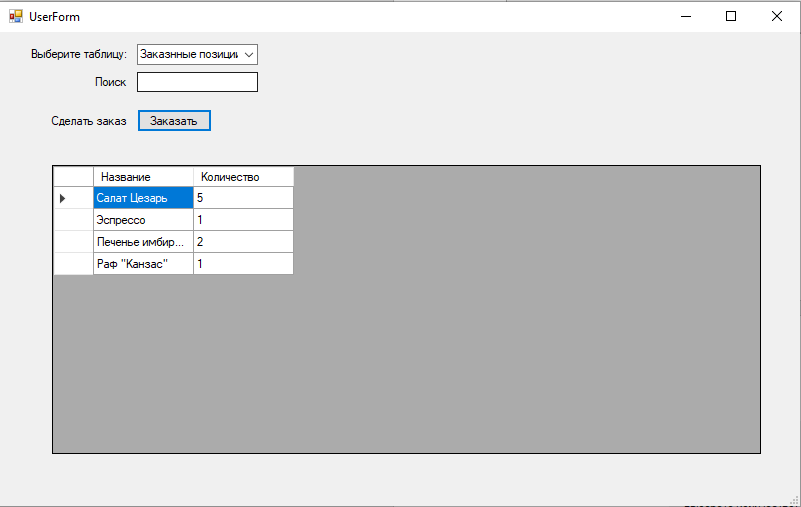


Рисунок 39 – Демонстрация заказа

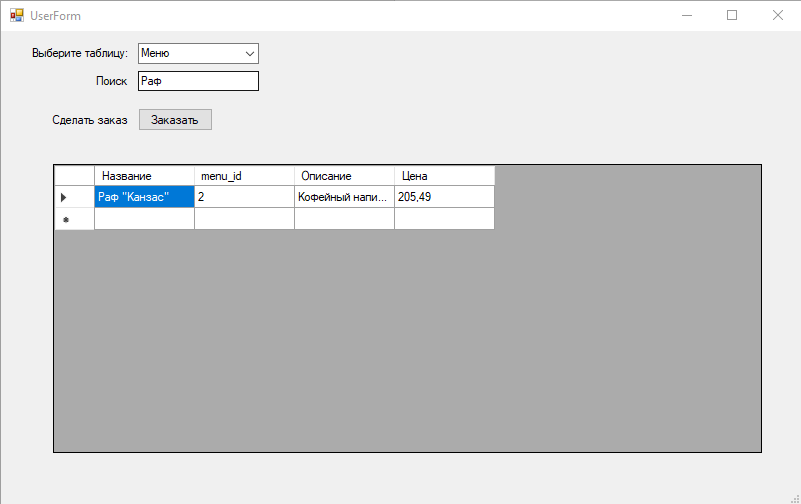


Рисунок 40 – Демонстрация поиска

Демонстрация роли администратора представлена на рисунках 41 – 49.

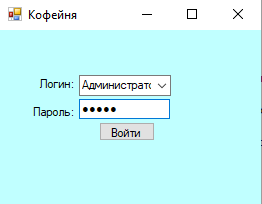


Рисунок 41 - Вход через администратора

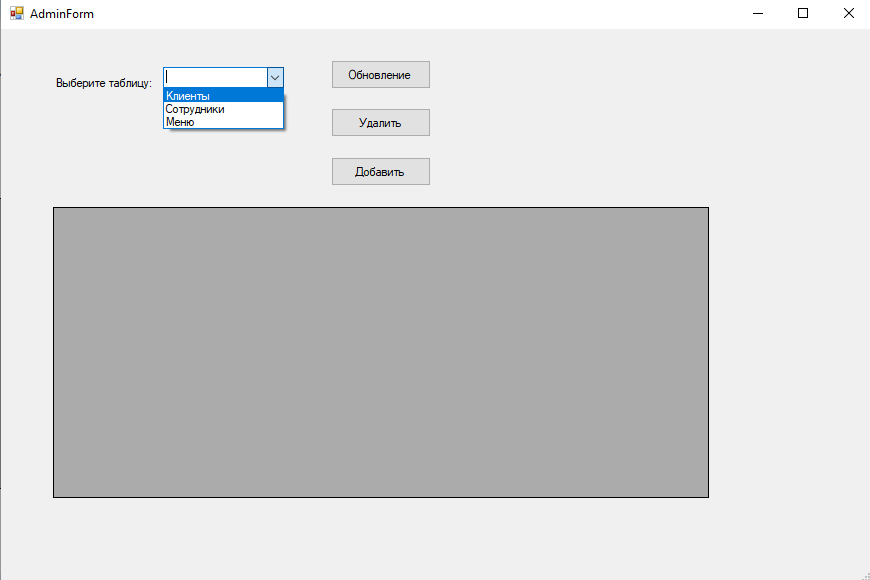


Рисунок 42 – Демонстрация формы

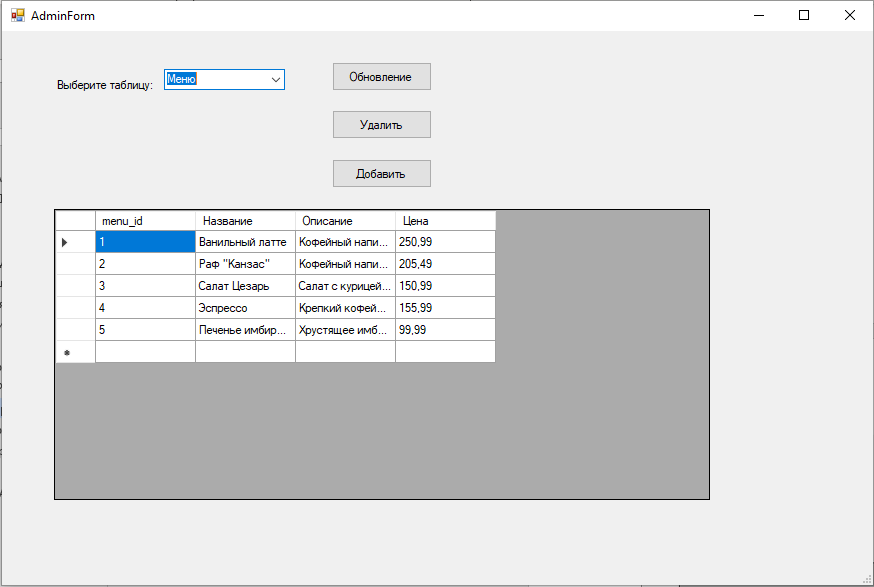


Рисунок 43 – Таблица Меню

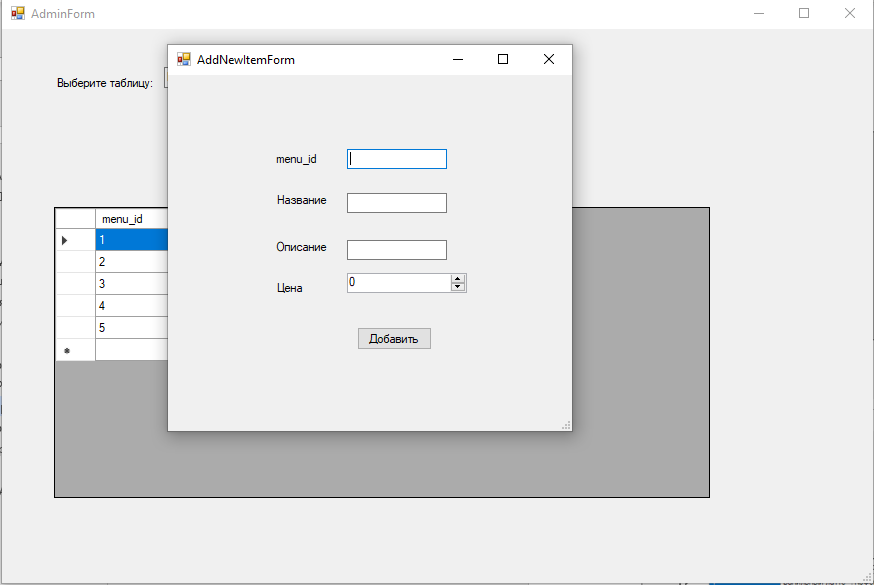


Рисунок 44 – Возможность добавление записей

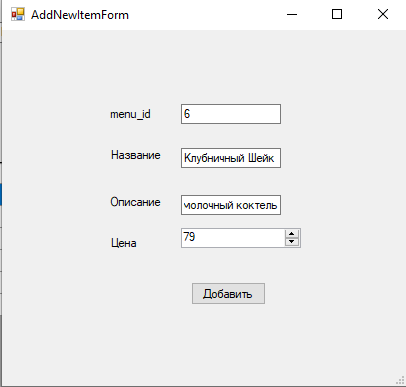


Рисунок 45 – Параметры новой строки

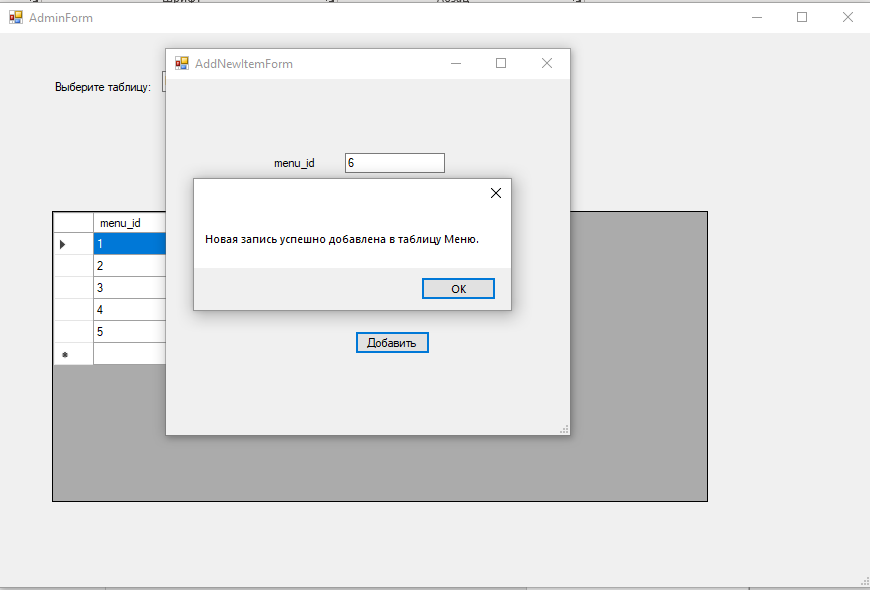


Рисунок 46 – Результат

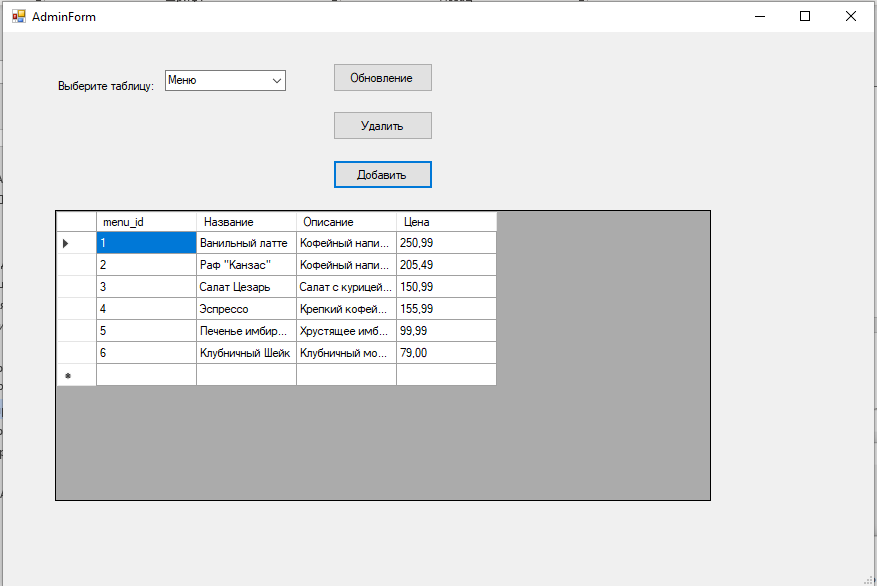


Рисунок 47 – Результат

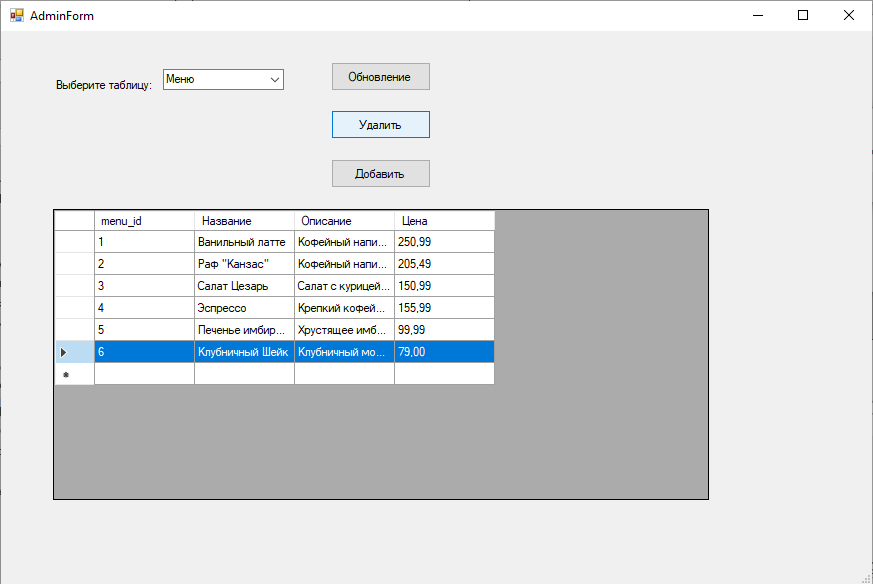


Рисунок 48 – Возможность удаления строк

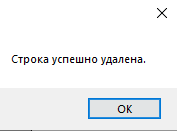


Рисунок 49 – Результат

ЗАКЛЮЧЕНИЕ

Разработка и внедрение базы данных кофейни представляют собой неотъемлемую часть современного кофейного бизнеса. База данных обеспечивает оперативный доступ к информации о клиентах и сотрудниках, упрощает взаимодействие между всеми участниками кофейного бизнеса и способствует повышению эффективности работы кофейни.

Курсовой проект по разработке базы данных кофейни должен учитывать потребности различных пользователей и обеспечивать удобный и безопасный доступ к информации. Важно также обеспечить надежность и безопасность системы, чтобы данные были защищены от потери. Поэтому важно уделить внимание созданию аккаунтов с разграничением прав доступа.

Наконец, внедрение базы данных кофейни способствует повышению качества обслуживания в кофейне в целом. Благодаря более оперативной обратной связи между персоналом и клиентами можно быстрее выявлять проблемные моменты и принимать меры для их решения, что способствует более эффективному функционированию кофейни.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Орлов, В.В. Технологии разработки программных продуктов. / В.В. Орлов. – СПб: Питер, 2009. – 437 с.

2 Малыхина, М.П. Базы данных: основы, проектирование, использование / М.П. Малыхина. - СПб.: BHV, 2007. - 528 c.

3 Баловсяк, Н. Видеосамоучитель создания реферата, курсовой, диплома на компьютере / Н. Баловсяк. – СПб: Питер, 2010. – 288 с.

4 Характеристика БД [Электронный ресурс]. – URL <https://studme.org/205585/informatika/sravnitelnaya_harakteristika_modeley>

5 Техническая документация [Электронный ресурс]. [Офиц. сайт]. URL: [http://tdocs.su](http://tdocs.su/)

6 Коваленко, В. В. Проектирование информационных систем: Учебное пособие / В. В. Коваленко. - М. : Форум, 2015. - 976 c.

7 Перлова, О. Н. Проектирование и разработка информационных систем: Учебник / О. Н. Перлова. - М. : Академия, 2018. - 272 c.

8 Павловская Т. А. - С#. Программирование на языке высокого уровня. Учебник для вузов. — СПб.: Питер, 2009. — 432 с: ил.

9 Иванова, Г.С. Технология программирования: Учебник для вузов / Г.С. Иванова – М.: Изд-во МГТУ им. Н.Э. Баумана, 2013. – 320 с.

10 Соловьев, И. В. Проектирование информационных систем. Фундаментальный курс / И.В. Соловьев, А. А. Майоров. - М. : Академический проект, 2009. - 398 c.

11 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.

ПРИЛОЖЕНИЕ А

Коды форм и классов для приложения на C#.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace BaseDate

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

using System;

using System.Collections.Generic;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace BaseDate

{

public partial class UserForm : Form

{

private string connectionString = @"Data Source=ADCLG1; Initial Catalog=Алексеева\_319-4\_1; Integrated Security=True";

private List<OrderItem> orderItems = new List<OrderItem>();

public UserForm()

{

InitializeComponent();

comboBox1.Items.AddRange(new string[] { "Меню", "Заказнные позиции" });

}

private void comboBox1\_SelectedIndexChanged\_1(object sender, EventArgs e)

{

LoadTableData();

}

private void textBox1\_TextChanged\_1(object sender, EventArgs e)

{

SearchTableData();

}

private void button1\_Click\_1(object sender, EventArgs e)

{

// Переход на форму Order

Order orderForm = new Order(this);

orderForm.Show();

}

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

{

// Переход на другую форму

Form2 anotherForm = new Form2();

anotherForm.Show();

this.Hide();

}

public void AddOrderItem(string menuItem, int quantity)

{

orderItems.Add(new OrderItem { Название = menuItem, Количество = quantity });

UpdateOrderGrid();

}

private void UpdateOrderGrid()

{

dataGridView1.DataSource = null;

dataGridView1.DataSource = orderItems;

}

private void LoadTableData()

{

if (comboBox1.SelectedItem == null)

return;

string tableName = comboBox1.SelectedItem.ToString();

string query = $"SELECT \* FROM {tableName}";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

dataAdapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных из таблицы {tableName}: {ex.Message}");

}

}

private void SearchTableData()

{

if (comboBox1.SelectedItem == null || string.IsNullOrEmpty(textBox1.Text))

return;

string tableName = comboBox1.SelectedItem.ToString();

string searchValue = textBox1.Text;

string query = $"SELECT \* FROM {tableName} WHERE ";

if (tableName == "Меню")

{

query += $"Название LIKE '%{searchValue}%' OR Описание LIKE '%{searchValue}%'";

}

else if (tableName == "Заказнные позиции")

{

query += $"Имя LIKE '%{searchValue}%' OR Детали LIKE '%{searchValue}%'";

}

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

dataAdapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при поиске данных в таблице {tableName}: {ex.Message}");

}

}

}

public class OrderItem

{

public string Название { get; set; }

public int Количество { get; set; }

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace BaseDate

{

public partial class Order : Form

{

private string connectionString = @"Data Source=ADCLG1; Initial Catalog=Алексеева\_319-4\_1; Integrated Security=True";

private UserForm userForm;

public Order(UserForm userForm)

{

InitializeComponent();

this.userForm = userForm;

LoadMenuItems();

}

private void LoadMenuItems()

{

string query = "SELECT Название FROM Меню";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader["Название"].ToString());

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке меню: {ex.Message}");

}

}

private void button1\_Click\_1(object sender, EventArgs e)

{

if (comboBox1.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, выберите позицию из меню.");

return;

}

string menuItem = comboBox1.SelectedItem.ToString();

int quantity = (int)numericUpDown1.Value;

userForm.AddOrderItem(menuItem, quantity);

this.Close();

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace BaseDate

{

public partial class Order : Form

{

private string connectionString = @"Data Source=ADCLG1; Initial Catalog=Алексеева\_319-4\_1; Integrated Security=True";

private UserForm userForm;

public Order(UserForm userForm)

{

InitializeComponent();

this.userForm = userForm;

LoadMenuItems();

}

private void LoadMenuItems()

{

string query = "SELECT Название FROM Меню";

try

{

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlCommand command = new SqlCommand(query, connection);

connection.Open();

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

comboBox1.Items.Add(reader["Название"].ToString());

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке меню: {ex.Message}");

}

}

private void button1\_Click\_1(object sender, EventArgs e)

{

if (comboBox1.SelectedItem == null)

{

MessageBox.Show("Пожалуйста, выберите позицию из меню.");

return;

}

string menuItem = comboBox1.SelectedItem.ToString();

int quantity = (int)numericUpDown1.Value;

userForm.AddOrderItem(menuItem, quantity);

this.Close();

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

using System.Collections.Generic; // Добавьте эту директиву

namespace BaseDate

{

public partial class AddNewItemForm : Form

{

private string connectionString = @"Data Source=ADCLG1; Initial Catalog=Алексеева\_319-4\_1; Integrated Security=True";

private string tableName;

private List<string> columnNames;

public AddNewItemForm(string tableName, List<string> columnNames)

{

InitializeComponent();

this.tableName = tableName;

this.columnNames = columnNames;

CreateControlsForColumns();

}

private void CreateControlsForColumns()

{

// Преобразование списка в массив строк

string[] columnNamesArray = columnNames.ToArray();

// Создание элементов управления для каждого столбца таблицы

foreach (string columnName in columnNamesArray)

{

Label label = new Label();

label.Text = columnName;

TextBox textBox = new TextBox();

textBox.Name = $"textBox{columnName}"; // Уникальное имя для каждого текстового поля

// Добавьте другие элементы управления, если необходимо, для разных типов данных

flowLayoutPanel1.Controls.Add(label);

flowLayoutPanel1.Controls.Add(textBox);

}

}

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

{

// Получение значений из элементов управления

List<string> values = new List<string>();

foreach (Control control in flowLayoutPanel1.Controls)

{

if (control is TextBox)

{

values.Add(((TextBox)control).Text);

}

// Добавьте обработку других типов элементов управления

}

// Создание SQL-запроса на добавление новой строки в таблицу

string columns = string.Join(", ", columnNames);

string parameters = string.Join(", ", values.ConvertAll(value => $"@{value}"));

string insertQuery = $"INSERT INTO {tableName} ({columns}) VALUES ({parameters})";

using (SqlConnection connection = new SqlConnection(connectionString))

{

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

{

// Параметризация запроса

for (int i = 0; i < values.Count; i++)

{

command.Parameters.AddWithValue($"@{values[i]}", values[i]);

}

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Новая строка успешно добавлена.");

}

else

{

MessageBox.Show("Не удалось добавить новую строку.");

}

}

catch (SqlException ex)

{

MessageBox.Show($"Ошибка SQL: {ex.Message}");

}

}

}

}

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

{

// Получение значений из текстовых полей

string menuId = textBox1.Text;

string name = textBox2.Text;

string description = textBox3.Text;

string price = numericUpDown1.Text;

// Создание SQL-запроса на добавление новой записи в таблицу "Меню"

string insertQuery = "INSERT INTO Меню (menu\_id, Название, Описание, Цена) VALUES (@menuId, @name, @description, @price)";

using (SqlConnection connection = new SqlConnection(connectionString))

{

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

{

// Параметризация запроса

command.Parameters.AddWithValue("@menuId", menuId);

command.Parameters.AddWithValue("@name", name);

command.Parameters.AddWithValue("@description", description);

command.Parameters.AddWithValue("@price", price);

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Новая запись успешно добавлена в таблицу Меню.");

}

else

{

MessageBox.Show("Не удалось добавить новую запись в таблицу Меню.");

}

}

catch (SqlException ex)

{

MessageBox.Show($"Ошибка SQL: {ex.Message}");

}

}

}

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace BaseDate

{

public partial class Form1 : Form

{

bool See = true;

public Form1()

{

// Заполнение ComboBox значениями ролей

InitializeComponent();

}

private void Form1\_Load\_1(object sender, EventArgs e)

{

button1.Enabled = false;

textBox1.MaxLength = 50;

textBox1.UseSystemPasswordChar = true;

}

private void comboBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

e.KeyChar = '\0'; // Запретить ввод пользователем

}

private void textBox1\_TextChanged\_1(object sender, EventArgs e)

{

if (comboBox1.Text.Length != 0 && textBox1.Text.Length != 0)

button1.Enabled = true;

else

button1.Enabled = false;

}

private void button1\_Click\_1(object sender, EventArgs e)

{

string adminPassword = "admin"; // Пароль для администратора

string userPassword = "user"; // Пароль для пользователя

if (comboBox1.Text == "Администратор" && textBox1.Text == adminPassword)

{

// Действия для администратора

AdminForm adminForm = new AdminForm();

this.Hide();

adminForm.ShowDialog();

this.Show();

return;

}

else if (comboBox1.Text == "Пользователь" && textBox1.Text == userPassword)

{

// Действия для пользователя

UserForm userForm = new UserForm();

this.Hide();

userForm.ShowDialog();

this.Show();

return;

}

MessageBox.Show("Неверно введен пароль", "Ошибка входа");

comboBox1.Text = "";

textBox1.Text = "";

}

}

}

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace BaseDate

{

public partial class Form2 : Form

{

public Form2()

{

InitializeComponent();

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

using System.Collections.Generic; // Добавьте эту директиву

namespace BaseDate

{

public partial class AdminForm : Form

{

private string connectionString = @"Data Source=ADCLG1; Initial Catalog=Алексеева\_319-4\_1; Integrated Security=True";

public AdminForm()

{

InitializeComponent();

// Заполнение comboBox1 значениями таблиц

comboBox1.Items.AddRange(new string[] { "Клиенты", "Сотрудники", "Меню" });

}

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

{

// Логика удаления выбранной записи из таблицы

DeleteSelectedItem();

}

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

{

// Получение имени выбранной таблицы

string selectedTable = comboBox1.SelectedItem.ToString();

// Создание пустого списка имен столбцов

List<string> columnNames = new List<string>();

// Открытие формы для добавления новых значений в таблицу

AddNewItemForm addItemForm = new AddNewItemForm(selectedTable, columnNames);

addItemForm.ShowDialog();

// Обновление данных в dataGridView1 после закрытия формы добавления новых значений

LoadTableData();

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

// Загрузка данных выбранной таблицы при изменении выбора в comboBox1

LoadTableData();

}

private void LoadTableData()

{

if (comboBox1.SelectedItem == null)

return;

string tableName = comboBox1.SelectedItem.ToString();

string query = $"SELECT \* FROM {tableName}";

using (SqlConnection connection = new SqlConnection(connectionString))

{

SqlDataAdapter dataAdapter = new SqlDataAdapter(query, connection);

DataTable dataTable = new DataTable();

dataAdapter.Fill(dataTable);

dataGridView1.DataSource = dataTable;

}

}

private void DeleteSelectedItem()

{

if (dataGridView1.SelectedRows.Count > 0)

{

int selectedIndex = dataGridView1.SelectedRows[0].Index;

// Получение значения первого столбца выбранной строки

object valueToDelete = dataGridView1.Rows[selectedIndex].Cells[0].Value;

// Создание SQL-запроса на удаление строки по значению первого столбца

string deleteQuery = $"DELETE FROM Меню WHERE menu\_id = @ValueToDelete";

using (SqlConnection connection = new SqlConnection(connectionString))

{

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

{

// Параметризация запроса

command.Parameters.AddWithValue("@ValueToDelete", valueToDelete);

try

{

connection.Open();

int rowsAffected = command.ExecuteNonQuery();

if (rowsAffected > 0)

{

MessageBox.Show("Строка успешно удалена.");

LoadTableData(); // Перезагрузка данных в dataGridView1 после удаления

}

else

{

MessageBox.Show("Не удалось удалить строку.");

}

}

catch (SqlException ex)

{

MessageBox.Show($"Ошибка SQL: {ex.Message}");

}

}

}

}

else

{

MessageBox.Show("Пожалуйста, выберите строку для удаления.");

}

}

}

}