АННОТАЦИЯ

Курсовой проект на тему: «Лабиринт», состоит из программного продукта, пояснительной записки и графической части.

Программный продукт предназначен для автоматизации процессов управления товарами в кофейни, включая учет продукции, управление персоналом, формирование отчетов и статистики. Система обеспечивает разграничение прав доступа между администраторами и обычными пользователями, позволяет осуществлять поиск и сортировку товаров по различным критериям, а также ведение базы данных сотрудников.

Программный продукт разработан в среде программирования C++ Builder с использованием языка программирования C++ и технологии ADO для работы с базами данных Microsoft Access. Система включает семь основных модулей: регистрацию и авторизацию пользователей, управление товарами, управление сотрудниками, формирование статистики и отчетов, просмотр каталога товаров и добавление новых позиций.

Пояснительная записка выполнена на 95 листах, содержит четыре раздела и одно приложение.

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc199765328)

[1. НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ 6](#_Toc199765329)

[2. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ 7](#_Toc199765330)

[2.1. Постановка задачи 7](#_Toc199765331)

[2.2. Описание организации данных 7](#_Toc199765332)

[2.3. Описание программных средств 8](#_Toc199765333)

[2.4. Проектирование интерфейса 9](#_Toc199765334)

[2.5. Описание логической структуры программы 15](#_Toc199765335)

[3. ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ 18](#_Toc199765336)

[4. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА 48](#_Toc199765337)

[ЗАКЛЮЧЕНИЕ 59](#_Toc199765338)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 61](#_Toc199765339)

[ПРИЛОЖЕНИЕ 1 (ОБЯЗАТЕЛЬНОЕ) – ЛИСТИНГ ПРОГРАММЫ 62](#_Toc199765340)

ВВЕДЕНИЕ

История торговли насчитывает тысячелетия, эволюционируя от простого товарообмена до современных систем электронной коммерции. В древних цивилизациях Месопотамии и Египта учет товаров велся на глиняных табличках, в Средние века купцы использовали специальные книги для фиксации информации о запасах и продажах.

Промышленная революция принесла механические кассовые аппараты, но настоящий прорыв произошел с появлением компьютеров и специализированного программного обеспечения во второй половине XX века, что позволило автоматизировать множество торговых процессов.

Несмотря на доступность современных технологий, многие малые и средние предприятия все еще используют устаревшие методы учета или простые электронные таблицы, что снижает их эффективность и конкурентоспособность.

Данный курсовой проект представляет собой разработку информационной системы управления товарами магазина, созданную в среде C++ Builder с использованием базы данных Microsoft Access. Система предоставляет различные функциональные возможности в зависимости от уровня доступа пользователя (администратор или обычный пользователь), включая управление товарами, контроль запасов, обработку заказов и формирование отчетов.

Цели курсового проектирования:

* закрепить, углубить и систематизировать теоретические знания и

практические навыки, полученные в ходе изучения учебного предмета;

* сформировать навыки разработки алгоритмов решения задач;
* разработка программы, проведение ее отладки, тестирование и верификация, разработка удобного интерфейса пользователя, системы меню (навигации), системы помощи.

Пояснительная записка содержит следующие разделы:

1. «Назначение и область применения» описывается назначение программы, краткая характеристика области применения программы. Проводится анализ предметной области и инфологическое проектирование разрабатываемой базы данных. Приводится обзор существующих подходов к решению задачи, их достоинства и недостатки, отличие разработанной программы от аналогов. Рассматриваются существующие аналоги.
2. «Технические характеристики» – рассматриваются следующие вопросы: формирование требований к системе в целом; требования к функциям (раздел «Постановка задачи»); требования к организации данных в проекте, описание их организации, структуры, построение концептуальной и логической модели данных, способа кодирования (при необходимости) и предварительной подготовки (раздел «Описание организации данных»); обоснование выбора программного обеспечения, используемого для разработки программного средства (раздел «Описание программных средств»); проектирование программного средства. (разделы «Описание физической структуры базы данных», «Проектирование интерфейса», «Описание логической структуры программы»).
3. «Тестирование и анализ полученных результатов». Описывается проверка работы программы на различных тестах и приводится анализ ожидаемых и полученных результатов, выводы по результатам тестирования.
4. «Руководство по использованию программного средства» – описывается последовательность действий пользователя, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведены описание функций, формата и возможных вариантов команд, с помощью которых пользователь управляет выполнением программы, а также ответы программы на эти команды.

# НАЗНАЧЕНИЕ И ОБЛАСТЬ ПРИМЕНЕНИЯ

Несмотря Разрабатываемый программный продукт представляет собой информационную систему управления товарами магазина, созданную с использованием C++ Builder и базы данных Microsoft Access. Система обеспечивает автоматизацию учета товаров, предоставляя удобный инструмент для работы с информацией о продуктах, их наличии, ценах и категориях.

Система предназначена для использования в розничной торговле, преимущественно в продуктовых магазинах малого и среднего масштаба. Её функционал позволяет эффективно управлять ассортиментом товаров, контролировать запасы и формировать заказы.

На рынке программного обеспечения представлены различные системы для автоматизации торговли, такие как «1С:Розница», «АТОЛ Frontol» и другие. Эти комплексные решения объединяют функции управления товарами, кассовыми операциями, складским учетом и аналитикой. Однако такие системы часто имеют избыточный функционал для небольших магазинов, требуют значительных затрат на внедрение и обучение персонала.

Разработанная информационная система предназначена для автоматизации основных бизнес-процессов в сфере розничной торговли продуктами. Она обеспечивает эффективное управление товарным ассортиментом, учет наличия товаров на складе и взаимодействие с клиентами.

Минимальные системные требования:

* ОС: Windows 10 или новее
* Процессор: Двухъядерный процессор с тактовой частотой 2,80 ГГц от Intel или AMD
* Оперативная память: 4 GB ОЗУ (И более)
* Видеокарта: NVIDIA GeForce 8600/9600GT, ATI/AMD Radeon HD2600/3600
* DirectX: версии 11
* Место на диске: 1 GB

1. ТЕХНИЧЕСКИЕ ХАРАКТЕРИСТИКИ
   1. Постановка задачи

Разработать программное средство для автоматизации работы кофейни. Необходимо реализовать модуль авторизации администратора с проверкой учетных данных (логин и пароль). Обеспечить возможность управления товарами, включая добавление нового товара в базу данных с указанием названия, категории, цены и количества на складе, редактирование информации о существующих товарах, удаление товаров из базы данных.

Реализовать функции для работы со списком товаров:

* поиск товаров по названию или другим параметрам;
* сортировка списка товаров по различным полям (наименование, цена, категория);
* фильтрация товаров по категориям и наличию.

Разработать модуль регистрации и авторизации пользователей с сохранением учетных данных в базе данных.

Реализовать функции для работы с каталогом товаров:

* просмотр списка доступных товаров, поиск товаров по различным параметрам;
* сортировка и фильтрация списка товаров;
* формирование заказа из выбранных товаров.

Также необходимо обеспечить удобный пользовательский интерфейс с интуитивно понятной навигацией. Обеспечить защиту от некорректного ввода данных и обработку возможных ошибок. Разработать модуль формирования и просмотра отчетов о состоянии товаров и продажах.

* 1. Описание организации данных

Данные приложения необходимо хранить в базе данных. Структура данных таблицы «Пользователи» представлена в таблице 2.1.

Таблица 2.1 – Структура данных таблицы «Пользователи»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| Код | Счетчик | Хранит уникальный номер пользователя |
| Логин | Короткий текст | Хранит логин пользователя |
| Пароль | Короткий текст | Хранит пароль пользователя |

Структура данных таблицы «Товары» представлена в таблице 2.2.

Таблица 2.2 – Структура данных таблицы «Товары»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| ТоварID | Счетчик | Хранит уникальный номер товара |
| Название | Короткий текст | Хранит название товара |
| Категория | Короткий текст | Хранит категория товара |
| Цена | Денежный | Хранит цену товара |
| ВНаличии | Логический | Хранит информацию о наличии товара |
| Описание | Короткий текст | Подробное описание товара |

Структура данных таблицы «Сотрудники» представлена в таблице 2.3.

Таблица 2.3 – Структура данных таблицы «Сотрудники»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| СотрудникID | Счетчик | Хранит уникальный номер сотрудника |
| Фамилия | Короткий текст | Хранит фамилию сотрудника |
| Имя | Короткий текст | Хранит имя сотрудника |
| Отчество | Короткий текст | Хранит отчество сотрудника |
| ДатаРождения | Дата и время | Хранит дату рождения сотрудника |
| Должность | Короткий текст | Хранит должность сотрудника |
| ДатаПриема | Дата и время | Хранит дату приема на работу сотрудника |
| Зарплата | Числовой | Хранит информацию о зарплате сотрудника |
| Телефон | Короткий текст | Хранит номер телефона сотрудника |
| Email | Короткий текст | Хранит информацию о почте сотрудника |
| Адрес | Короткий текст | Хранит адрес проживания сотрудника |
| Примечание | Короткий текст | Хранит короткую информацию о сотруднике |

Структура данных таблицы «Заказы» представлена в таблице 2.4.

Таблица 2.4 – Структура данных таблицы «Заказы»

|  |  |  |
| --- | --- | --- |
| Наименование столбца | Тип данных | Описание |
| ID | Счетчик | Хранит уникальный номер заказа |
| Дата | Дата и время | Хранит информацию о дате заказа |
| Детали | Длинный текст | Хранит детали заказа |
| Сумма | Числовой | Хранит сумму заказа |

* 1. Описание программных средств

В качестве выбранного языка программирования используется С++.

C++ – это язык программирования, созданный в 1979 году Бьёрном Страуструпом как расширение языка C. Он позволяет писать как низкоуровневый код для прямого управления аппаратными ресурсами, так и высокоуровневые абстракции для удобной разработки сложных программ. Основные преимущества C++:

* скомпилированный код работает быстро и эффективно использует ресурсы компьютера;
* прямой доступ к памяти позволяет точно контролировать, как программа использует оперативную память;
* объектно-ориентированное программирование поддерживает классы, наследование, полиморфизм и другие ООП-концепции;
* программы на C++ можно скомпилировать для различных операционных систем.

В качестве среды разработки был использован C++ Builder. C++Builder – одна из ведущих сред разработки для создания   
Internet-приложений, «настольных» и распределенных приложений, а также приложений, основанных на модели клиент/сервер. C++Builder сочетает простоту среды быстрой разработки приложений, или RAD-среды (Rapid Application Development — RAD), с мощью и производительностью языка C++, совместимого со стандартом ANSI.

* 1. Проектирование интерфейса

Данный курсовой проект состоит из нескольких модулей, каждый из которых имеет свою реализацию. Форма авторизации, которая отображается при запуске приложения, рис. 2.1., содержит следующие элементы:

* поле ввода имени;
* поле ввода пароля;
* кнопка входа;
* кнопка регистрации.

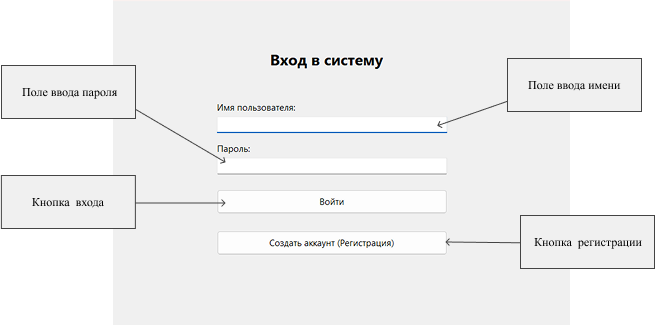


Рис. 2.1. Форма авторизации

Форма регистрации, рис. 2.2., содержит следующие элементы:

* поле ввода имени;
* поле ввода пароля;
* поле для повторного ввода пароля;
* кнопка регистрации;
* кнопка входа.

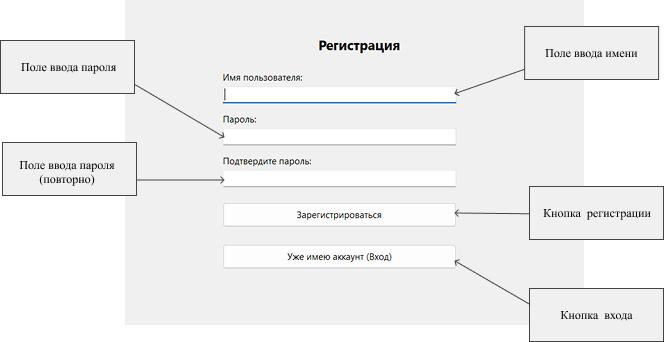


Рис. 2.2. Форма регистрации

Основная форма администратора, рис. 2.3., содержит следующие интерактивные элементы:

* поле поиска;
* кнопка поиска;
* выпадающий список сортировки;
* кнопка обновления;
* поле ввода названия;
* поле ввода цены;
* поле ввода описания;
* поле ввода количества;
* выпадающий список категорий;
* флажок наличия;
* кнопка добавления;
* кнопка редактирования;
* кнопка для удаления выбранного товара из базы данных;
* кнопка очистки полей;
* кнопка сброса;
* кнопка отмены;
* кнопка статистики;
* кнопка управления;
* кнопка корзины;
* кнопка выхода.

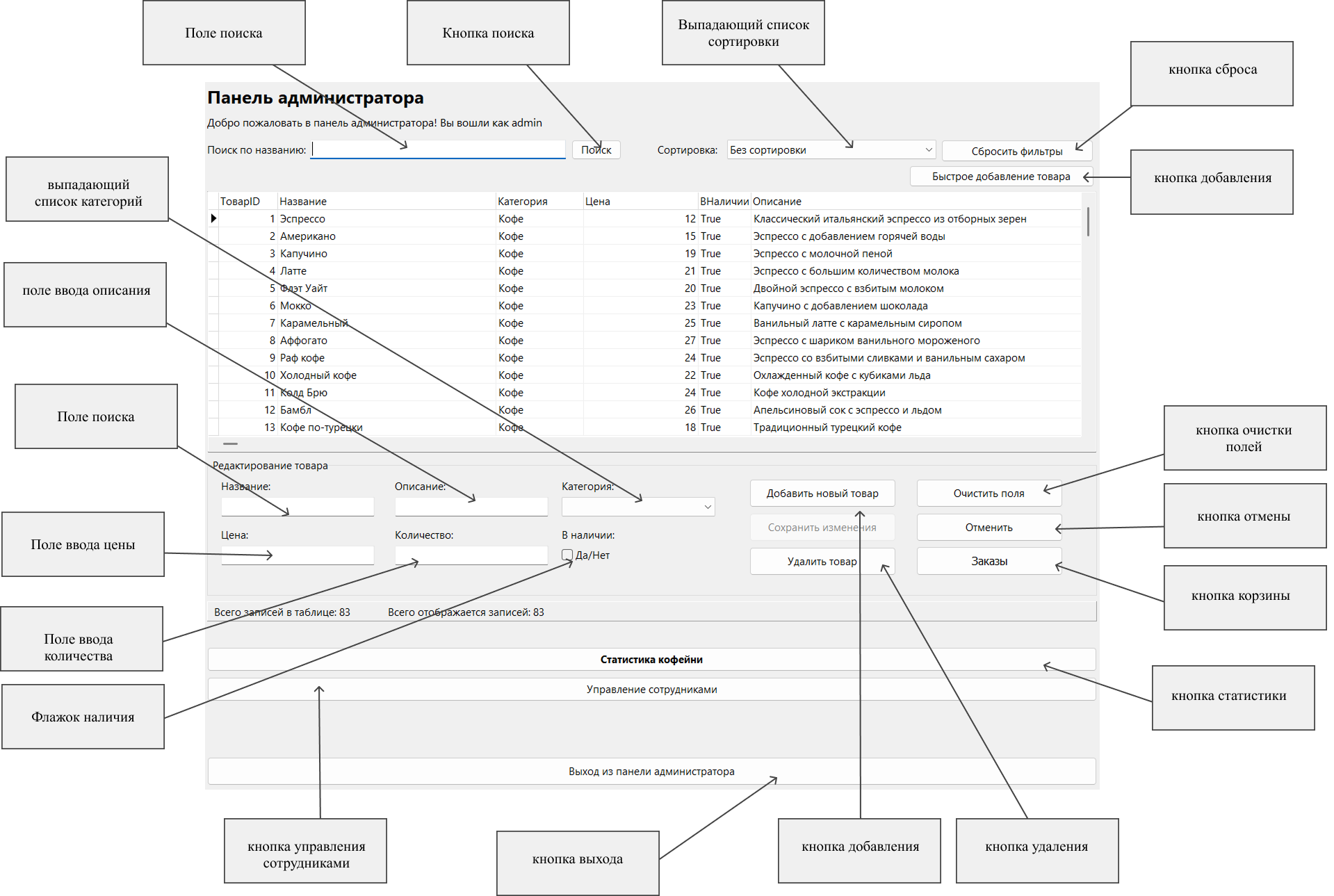
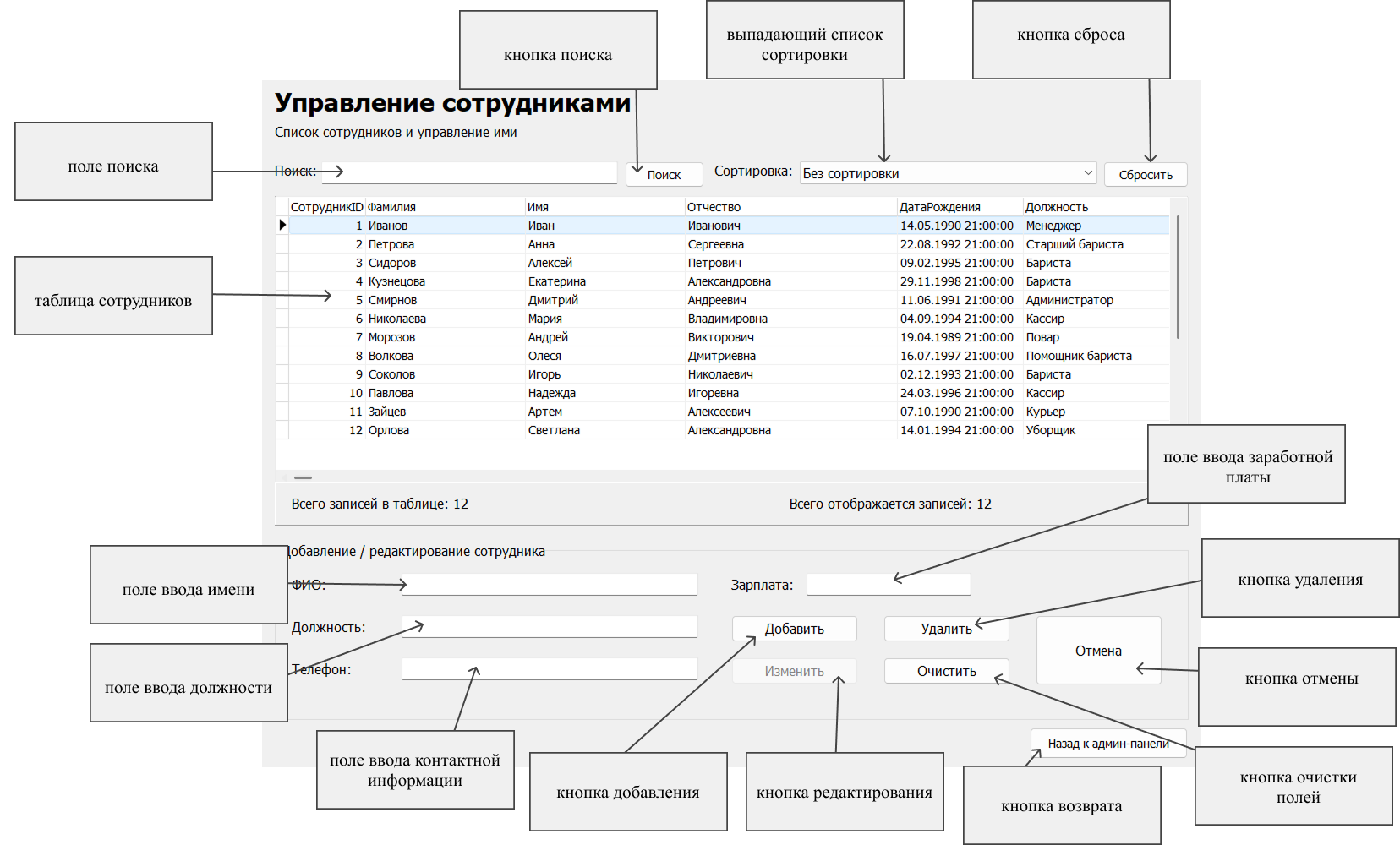


Рис. 2.3. Форма администратора

Форма управления сотрудниками, рис. 2.4., содержит следующие интерактивные элементы:

* таблица сотрудников;
* поле поиска;
* кнопка поиска;
* выпадающий список сортировки;
* поле ввода имени;
* поле ввода фамилии;
* поле ввода должности;
* поле ввода контактной информации;
* поле ввода заработной платы;
* кнопка добавления;
* кнопка редактирования;
* кнопка удаления;
* кнопка очистки полей;
* кнопка сброса;
* кнопка возврата.

  
Рис. 2.4. Форма управления сотрудниками

Форма управления заказами, рис. 2.5., содержит следующие интерактивные элементы:

* таблица;
* выпадающий список;
* выпадающий список категорий;
* поле ввода количества;
* кнопка добавления в корзину;
* скрытый компонент для хранения информации о корзине;
* кнопка возврата.

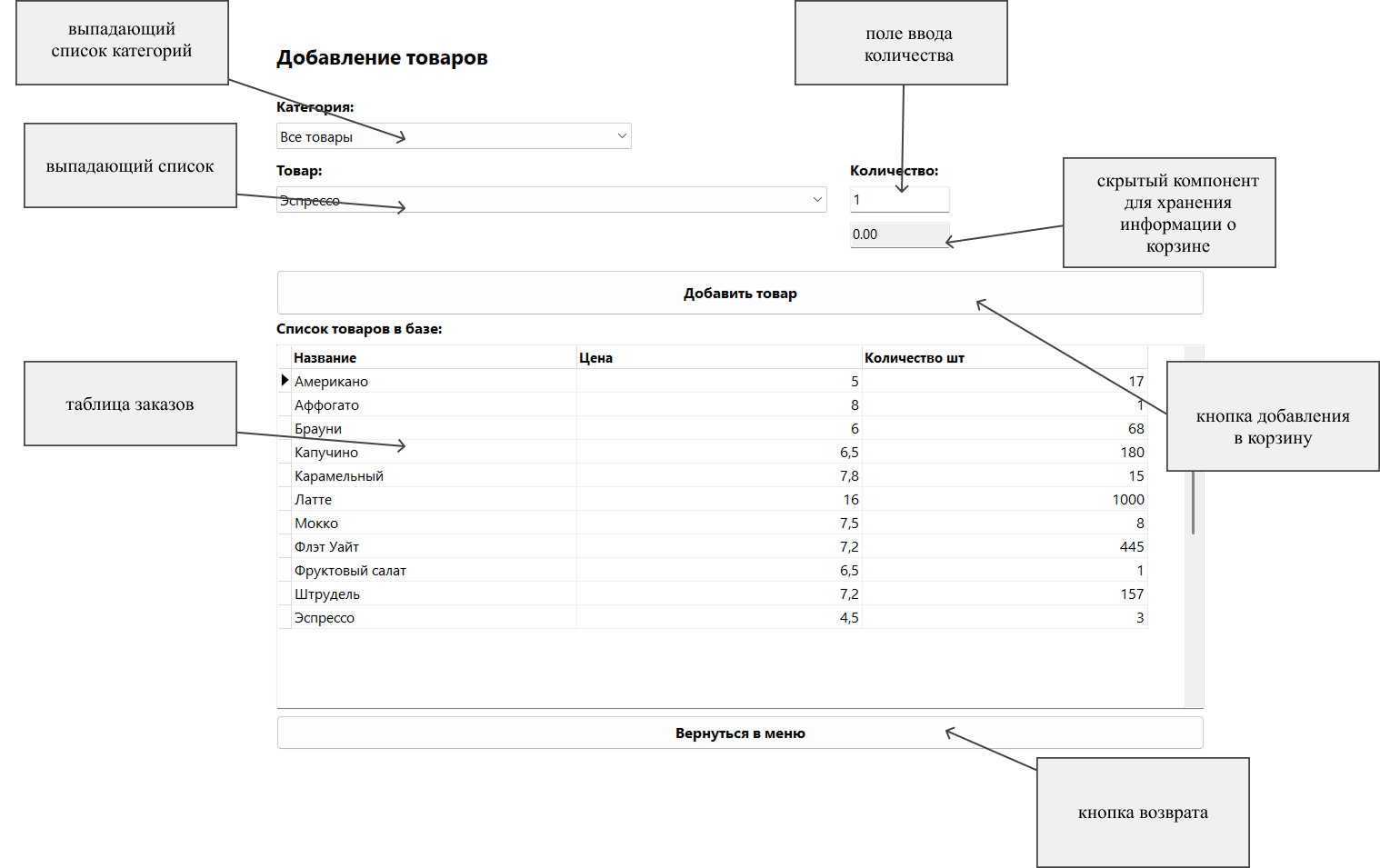


Рис. 2.5. Форма управления заказами

Форма статистики, рис. 2.6., содержит следующие интерактивные элементы:

* поле поиска;
* таблица статистики;
* выпадающий список сортировки;
* кнопка сортировки;
* кнопка поиска;
* кнопка экспорта;
* кнопка возврата.

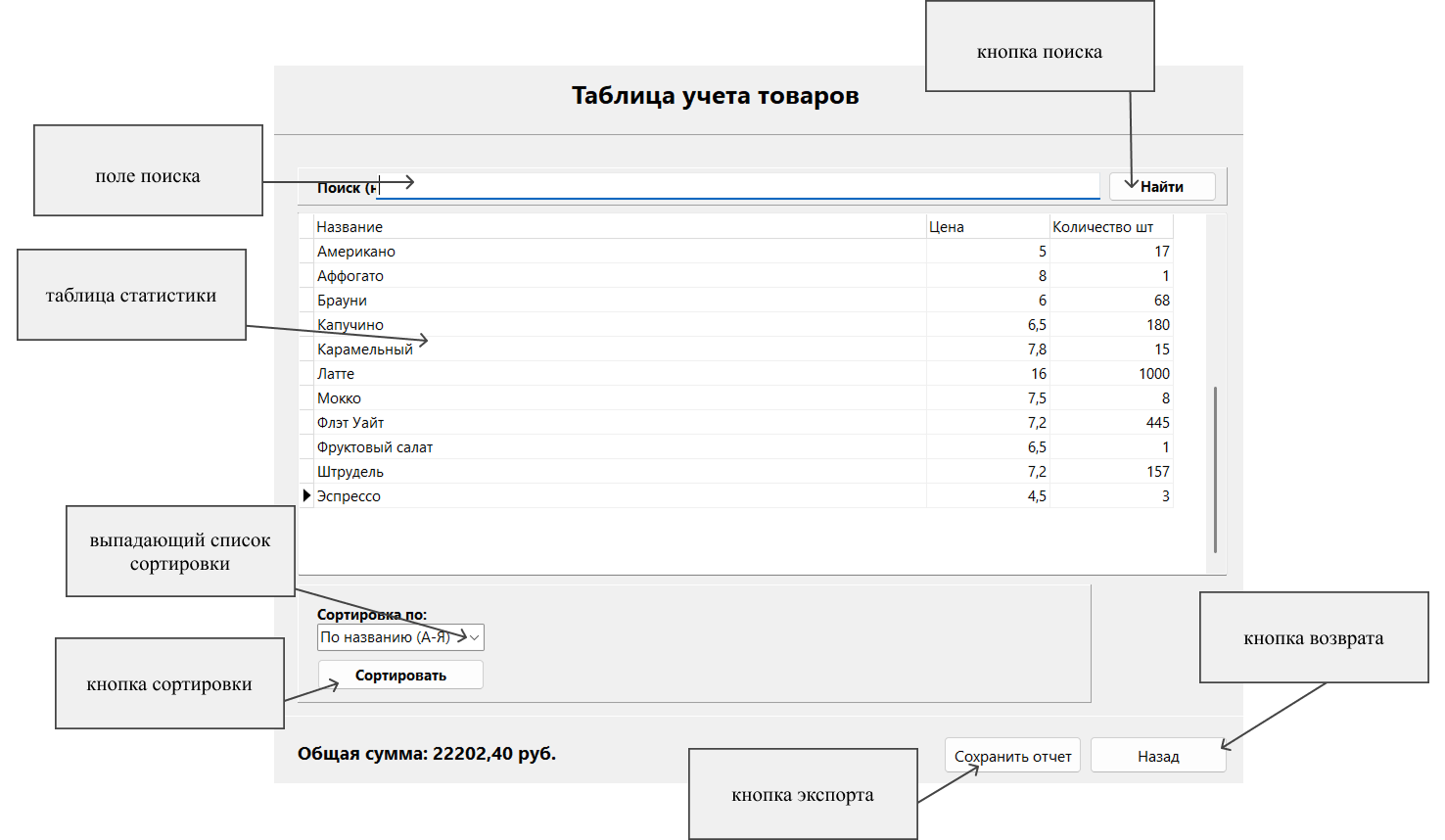


Рис. 2.6. Форма статистики

Форма управления заказами, рис. 2.7., содержит следующие интерактивные элементы:

* таблица;
* поле поиска;
* кнопка поиска;
* кнопка возврата;
* кнопка очистки фильтров;
* выпадающий список категорий.

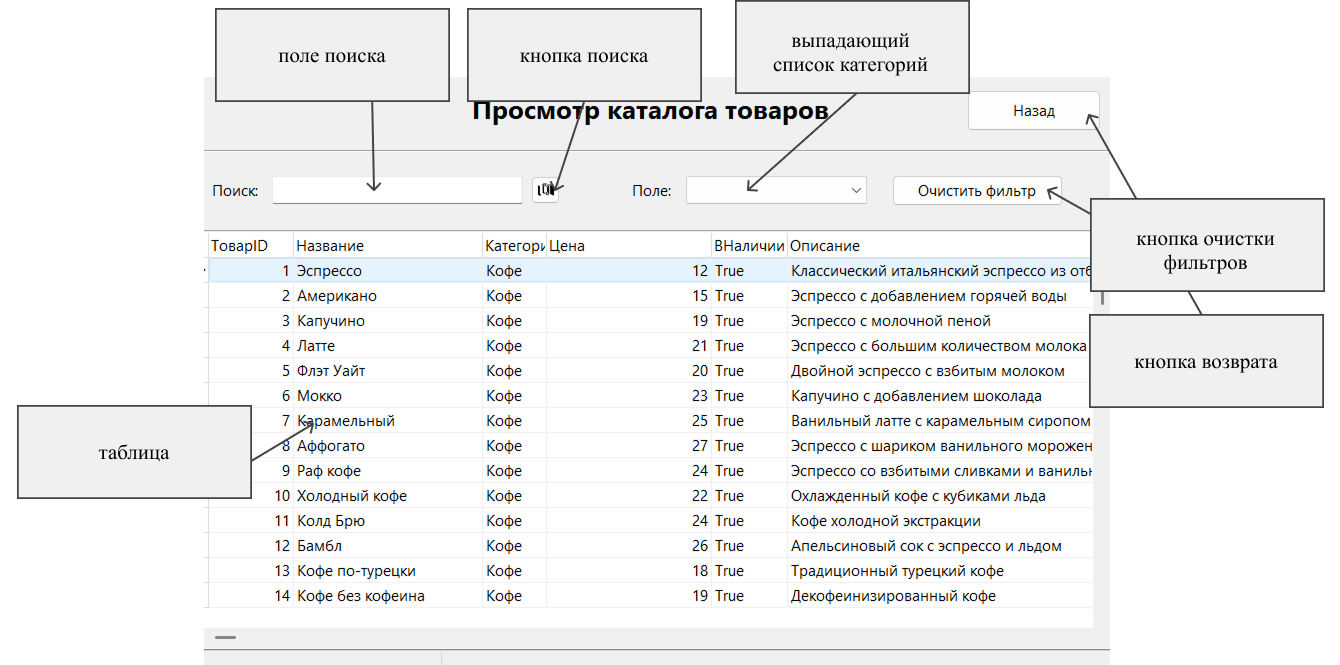


Рис. 2.7. Форма управления заказами

* 1. Описание логической структуры программы

Для работы приложения были реализованы следующие функции:

* FormCreate() – функция инициализации формы и подключения к базе данных;
* Button1Click() – функция обработки нажатия кнопки "Войти" для проверки учетных данных;
* Button2Click() – функция обработки нажатия кнопки "Регистрация" для перехода к форме регистрации;
* FormKeyPress() – функция обработки нажатия клавиши Enter для быстрого входа в систему;
* Edit2KeyPress() – функция обработки нажатия клавиши Enter в поле пароля.
* FormCreate() – функция инициализации формы и подключения к базе данных;
* Button1Click() – функция обработки нажатия кнопки "Зарегистрироваться" для добавления нового пользователя;
* Button2Click() – функция обработки нажатия кнопки "Вернуться" для возврата к форме авторизации.
* FormShow() – функция инициализации формы при отображении;
* Button1Click() – функция обработки нажатия кнопки поиска товаров;
* Button2Click() – функция обработки нажатия кнопки обновления данных;
* Button4Click() – функция обработки нажатия кнопки выхода из системы;
* Edit1Change() – функция обработки изменения текста в поле поиска;
* ComboBoxSortChange() – функция обработки изменения выбора в списке сортировки;
* ButtonAddClick() – функция обработки нажатия кнопки добавления нового товара;
* ButtonEditClick() – функция обработки нажатия кнопки редактирования товара;
* ButtonDeleteClick() – функция обработки нажатия кнопки удаления товара;
* ButtonClearClick() – функция обработки нажатия кнопки очистки полей ввода;
* ButtonCancelClick() – функция обработки нажатия кнопки отмены операции;
* DBGrid1DblClick() – функция обработки двойного клика по строке в таблице товаров;
* ButtonEmployeesClick() – функция обработки нажатия кнопки перехода к форме сотрудников;
* ButtonStatisticsClick() – функция обработки нажатия кнопки перехода к форме статистики;
* ButtonCartClick() – функция обработки нажатия кнопки перехода к форме заказов;
* UpdateRecordCount() – функция обновления счетчика записей;
* ApplyFilter() – функция применения фильтров к списку товаров;
* ApplySort() – функция применения сортировки к списку товаров;
* RefreshData() – функция обновления данных в таблице товаров;
* ClearInputFields() – функция очистки полей ввода;
* LoadProductToFields() – функция загрузки данных выбранного товара в поля ввода;
* ValidateInputs() – функция проверки корректности введенных данных;
* AddProductQuick() – функция быстрого добавления товара по базовым параметрам.
* FormCreate() – функция инициализации формы и подключения к базе данных;
* EditSearchChange() – функция обработки изменения текста в поле поиска;
* ComboBoxFilterChange() – функция обработки изменения выбора в списке фильтрации;
* ComboBoxSortChange() – функция обработки изменения выбора в списке сортировки;
* ButtonClearFilterClick() – функция обработки нажатия кнопки сброса фильтров;
* ButtonBackClick() – функция обработки нажатия кнопки возврата;
* SpeedButtonSearchClick() – функция обработки нажатия кнопки поиска;
* FormKeyPress() – функция обработки нажатия клавиши Enter для быстрого поиска;
* ApplyFilter() – функция применения фильтров к списку товаров;
* UpdateStatusBar() – функция обновления информации в строке состояния;
* RefreshData() – функция обновления данных в таблице товаров;
* ApplySort() – функция применения сортировки к списку товаров.
* FormShow() – функция инициализации формы при отображении;
* ButtonBackClick() – функция обработки нажатия кнопки возврата;
* ButtonAddToCartClick() – функция обработки нажатия кнопки добавления товара в корзину;
* ComboBoxProductsChange() – функция обработки изменения выбора в списке товаров;
* LoadProductsList() – функция загрузки списка доступных товаров;
* GetProductPrice() – функция получения цены выбранного товара;
* AddProductToDatabase() – функция добавления товара в базу данных.
* FormShow() – функция инициализации формы при отображении;
* ApplyFilterButtonClick() – функция обработки нажатия кнопки применения фильтров;
* ResetFilterButtonClick() – функция обработки нажатия кнопки сброса фильтров;
* ExportButtonClick() – функция обработки нажатия кнопки экспорта данных;
* PrintButtonClick() – функция обработки нажатия кнопки печати отчета;
* RefreshButtonClick() – функция обработки нажатия кнопки обновления данных;
* BackButtonClick() – функция обработки нажатия кнопки возврата;
* PeriodComboBoxChange() – функция обработки изменения выбора периода;
* ReportTypeComboBoxChange() – функция обработки изменения типа отчета;
* CategoryComboBoxChange() – функция обработки изменения категории;
* UpdateStatisticsData() – функция обновления статистических данных;
* GenerateReport() – функция формирования отчета по текущим данным.

1. ТЕСТИРОВАНИЕ И АНАЛИЗ ПОЛУЧЕННЫХ РЕЗУЛЬТАТОВ

В процессе разработки информационной системы управления товарами магазина был использован метод тестирования «черного ящика». Программа была протестирована на следующих тестах:

1. Тест №1. Проверка функциональности авторизации
2. Тест №2. Проверка функциональности регистрации нового пользователя.
3. Тест №3. Проверка разграничения прав доступа между администратором и обычным пользователем.
4. Тест №4. Проверка функциональности добавления нового товара.
5. Тест №5. Проверка функциональности редактирования существующего товара.
6. Тест №6. Проверка функциональности удаления товара.
7. Тест №7. Проверка функциональности поиска товаров.
8. Тест №8. Проверка функциональности сортировки списка товаров.
9. Тест №9. Проверка функциональности добавления товара в корзину.
10. Тест №10. Проверка функциональности страницы статистики кофейни.

Тест №1. Проверка функционала авторизации. Запустить приложение. Ввести логин «admin» и пароль «1111». Нажать кнопку «Войти». Открылась форма администратора с полным доступом к функциям управления товарами. При вводе некорректных данных система выдает соответствующее предупреждение, рис. 3.1., рис. 3.2., рис. 3.3.

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.1. Открылась форма авторизации

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.2. Ввод логина и пароля администратора

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.3. Открылась форма администратора

Тест №2. Проверка функциональности регистрации нового пользователя. Запустить приложение. На форме авторизации нажать кнопку «Регистрация». Открылась форма регистрации с полями для ввода логина, пароля и подтверждения пароля. Ввести новый логин «testuser», пароль «123456» и подтверждение пароля «123456». Нажать кнопку «Зарегистрироваться». Система создает нового пользователя и перенаправляет на форму просмотра товаров с правами обычного пользователя. При попытке зарегистрировать пользователя с существующим логином система выдает соответствующее предупреждение, рис. 3.4., рис. 3.5., рис. 3.6., рис. 3.7., рис. 3.8., рис. 3.9., рис. 3.10.

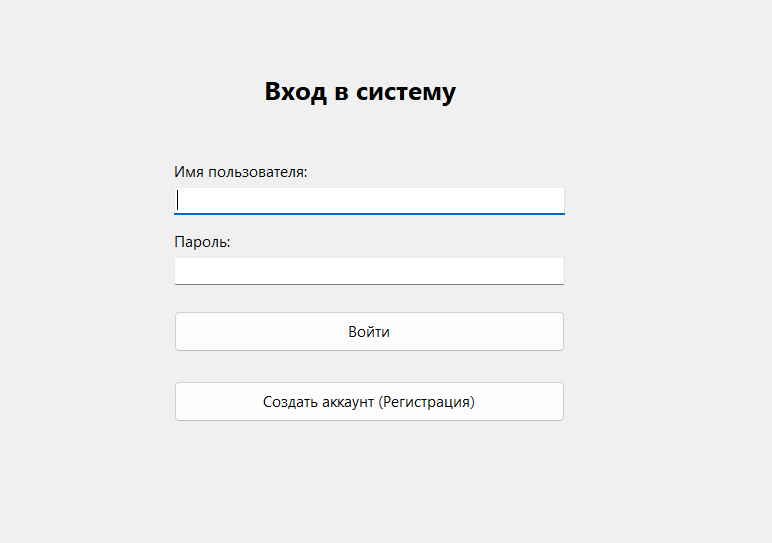


Рис. 3.4. Открылась форма авторизации

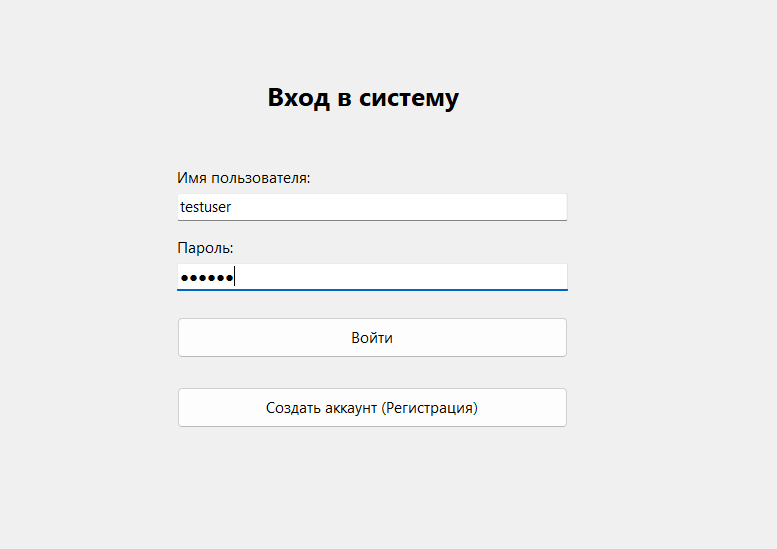


Рис. 3.5. Ввод логина и пароля пользователя

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.6. Сообщение о неизвестном пользователе

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.7. Переход на страницу регистрации

Изображение выглядит как текст, снимок экрана, Шрифт, число

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 3.8. Ввод данных для регистрации

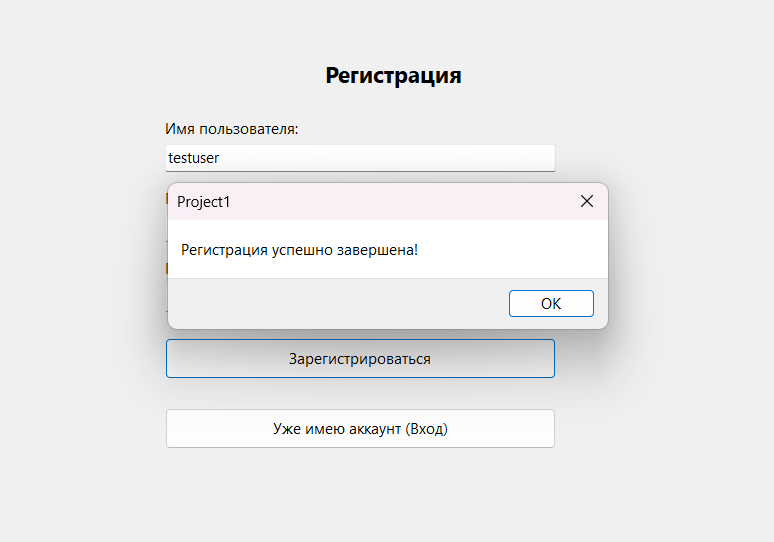


Рис. 3.9. Сообщение о успешной регистрации

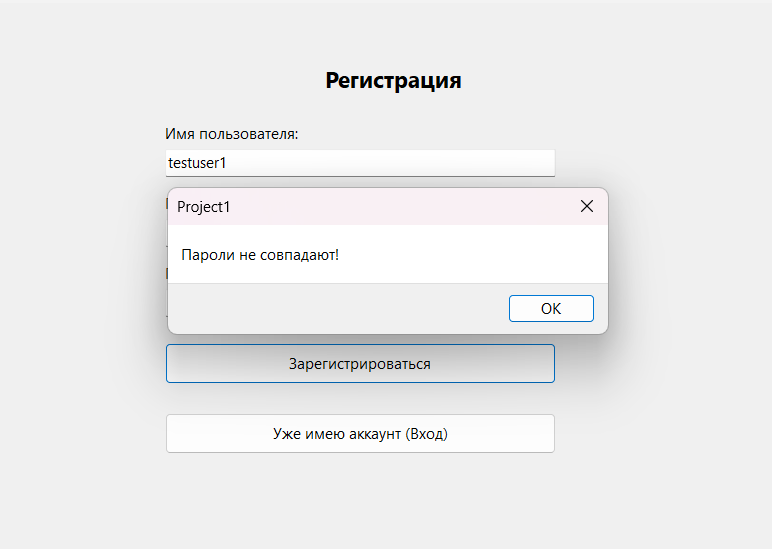


Рис. 3.10. Сообщение об ошибке

Тест №3. Проверка разграничения прав доступа между администратором и обычным пользователем. Войти в систему с учетными данными администратора. На форме администратора заполнить поля ввода: название товара – «Тестовое», категория – «Напитки», цена – «10», количество – «10». Нажать кнопку «Добавить». Новый товар успешно добавлен в базу данных и отображается в таблице товаров, рис. 3.11., рис. 3.12.

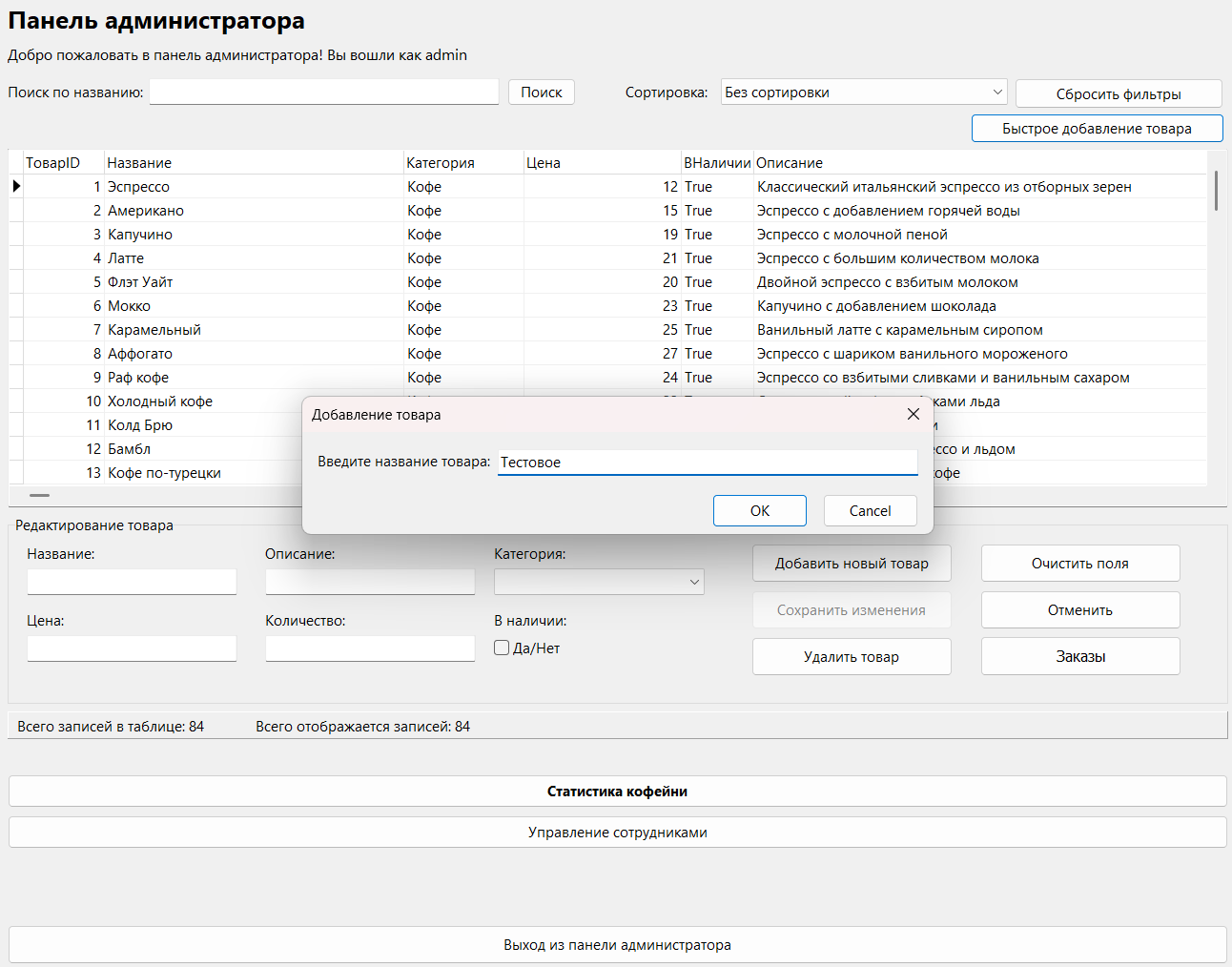


Рис. 3.11. Добавить товар

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.12. Товар отображается в таблице

Тест №4. Проверка функциональности добавления нового товара. Войти в систему с учетными данными администратора. В таблице товаров выбрать «Тестовый товар». Данные товара автоматически загружаются в поля ввода. Изменить цену на «12» и количество на «15». Нажать кнопку «Редактировать». Изменения успешно сохранены в базе данных и отображаются в таблице товаров, рис. 3.13., рис. 3.14., рис. 3.15.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.13. Загрузились данные о товаре

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.14. Изменения были внесены

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.15. Данные товара обновились

Тест №5. . Проверка функциональности редактирования существующего товара. Войти в систему с учетными данными администратора. В таблице товаров выбрать «Тестовый товар». Нажать кнопку «Удалить». Система запрашивает подтверждение удаления. Нажать «Да». Товар успешно удален из базы данных и больше не отображается в таблице товаров, рис. 3.16., рис. 3.17., рис. 3.18.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.16. Выбор товара

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.17. Нажатие кнопки «Удалить товар»

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.18. Товар успешно удален

Тест №6. Проверка функциональности удаления товара. Войти в систему с любыми учетными данными. В поле поиска ввести «Кофе». Нажать кнопку поиска. В таблице отображаются только товары, в названии которых присутствует слово «Кофе». При вводе несуществующего товара таблица товаров будет пустая, рис. 3.19., рис. 3.20.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.19. Поле поиск заполнено

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.20. Введено несуществующее название

Тест №7. Проверка функциональности поиска товаров. Войти в систему с любыми учетными данными. В выпадающем списке сортировки выбрать «По цене по возрастанию». Товары в таблице автоматически сортируются по цене от минимальной к максимальной. Выбрать «По названию от А до Я». Товары в таблице автоматически сортируются по алфавиту, рис. 3.21., рис. 3.22., рис. 3.23., рис. 3.24.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.21. Выбор сортировки «По цене по возрастанию»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.22. Товар был отсортирован по цене по возрастанию

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.23. Выбор сортировки «По названию от А до Я»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.24. Товар был отсортирован по названию от А до Я

Тест №8. Проверка функциональности сортировки списка товаров. Войти в систему с любыми учетными данными. В выпадающем списке сортировки выбрать «По цене по убыванию)». Товары в таблице автоматически сортируются по цене от минимальной к максимальной. Выбрать «По названию от Я до А». Товары в таблице автоматически сортируются по алфавиту, рис. 3.25., рис. 3.26., рис. 3.27., рис. 3.28.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.Рис. 3.25. Выбор сортировки «По цене по убыванию»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.26. Товар был отсортирован по цене по убыванию

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.27. Выбор сортировки «По названию от Я до А»

Изображение выглядит как текст, снимок экрана, Шрифт, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.28. Товар был отсортирован от Я до А

Тест №9. Проверка функциональности добавления товара в корзину. Запустить приложение. Войти как администратор. Перейти к форме добавления товаров. Выбрать товар «Капучино» из списка или ввести новое название. Указать количество «5». Выбрать категорию «Кофе». Система автоматически подставляет цену. Нажать «Добавить товар». Товар добавляется в базу данных, таблица обновляется. Если товар существует - увеличивается количество. Поля очищаются для следующего товара. При некорректных данных система выдает предупреждения, рис. 3.29., рис. 3.30., рис. 3.31.

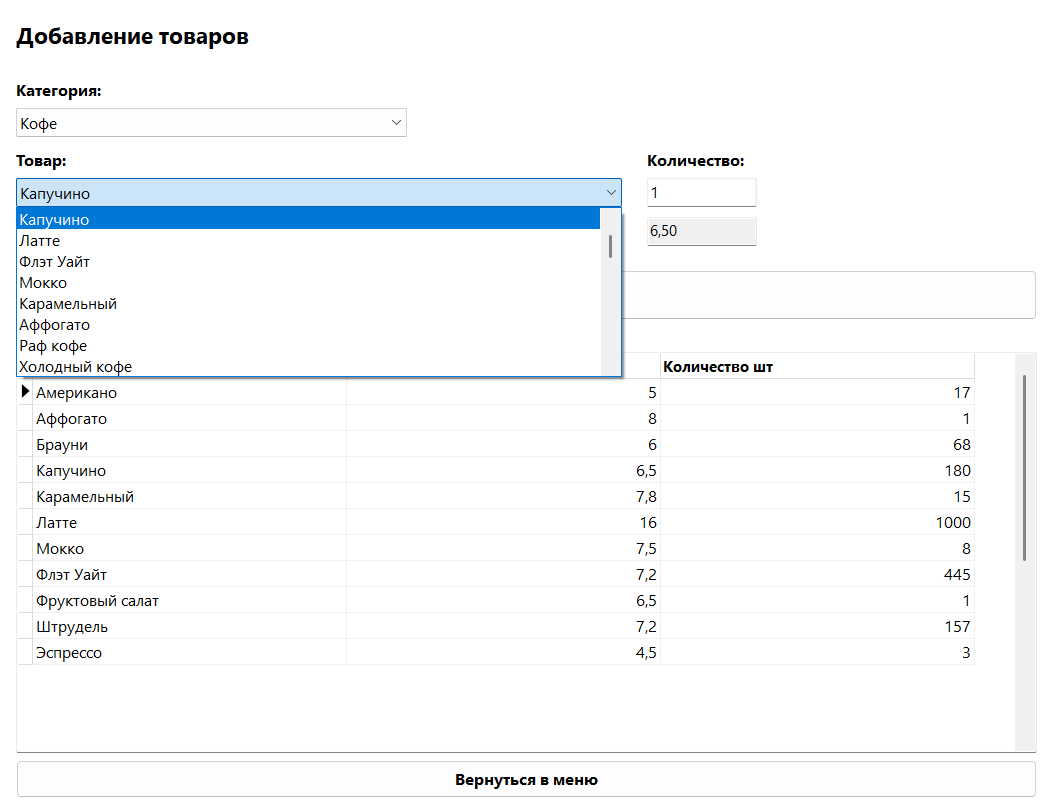


Рис. 3.29. Выбран товар из списка

Изображение выглядит как текст, снимок экрана, число, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.30. Указываем количество товара

Изображение выглядит как текст, снимок экрана, число, Параллельный

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.31. Обновилось количество товара

Тест №10. Проверка функциональности страницы статистики кофейни. Запустить приложение. Войти как администратор. Перейти к странице отчетов и статистики. Система отображает таблицу с товарами и их характеристиками. Проверить корректность расчета общей суммы товаров на складе. Нажать кнопку «Сохранить отчет» - система генерирует файл с полной статистикой товаров. Проверить, что в отчете содержится актуальная информация о товарах, ценах и общей сумме на складе, рис. 3.32., рис. 3.33.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.32. Сохраняем отчет

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 3.33. Проверяем отчет

1. РУКОВОДСТВО ПО ИСПОЛЬЗОВАНИЮ ПРОГРАММНОГО СРЕДСТВА

При запуске программы открывается форма авторизации с полями для ввода логина и пароля пользователя, рис. 4.1. Для входа в систему необходимо ввести свои учетные данные и нажать кнопку «Войти». Если у пользователя еще нет учетной записи, можно перейти к форме регистрации, нажав кнопку «Регистрация».

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4.1. Форма авторизации

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4.2. Форма регистрации

После успешной авторизации администратора открывается форма управления товарами с расширенным функционалом, рис. 4.3. В верхней части формы расположена таблица со списком всех товаров. При выборе товара в таблице его данные автоматически загружаются в поля ввода в нижней части формы.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4.3. Форма администратора

Для добавления нового товара необходимо заполнить поля «Название», «Цена», «Описание», «Количество», выбрать категорию из выпадающего списка и нажать кнопку «Добавить», рис. 4.4. Система проверит корректность введенных данных и добавит новый товар в базу.

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4.4. Добавление нового товара

Для редактирования существующего товара нужно выбрать его в таблице, внести необходимые изменения в поля ввода и нажать кнопку «Редактировать», рис. 4.5. Изменения будут сохранены в базе данных.

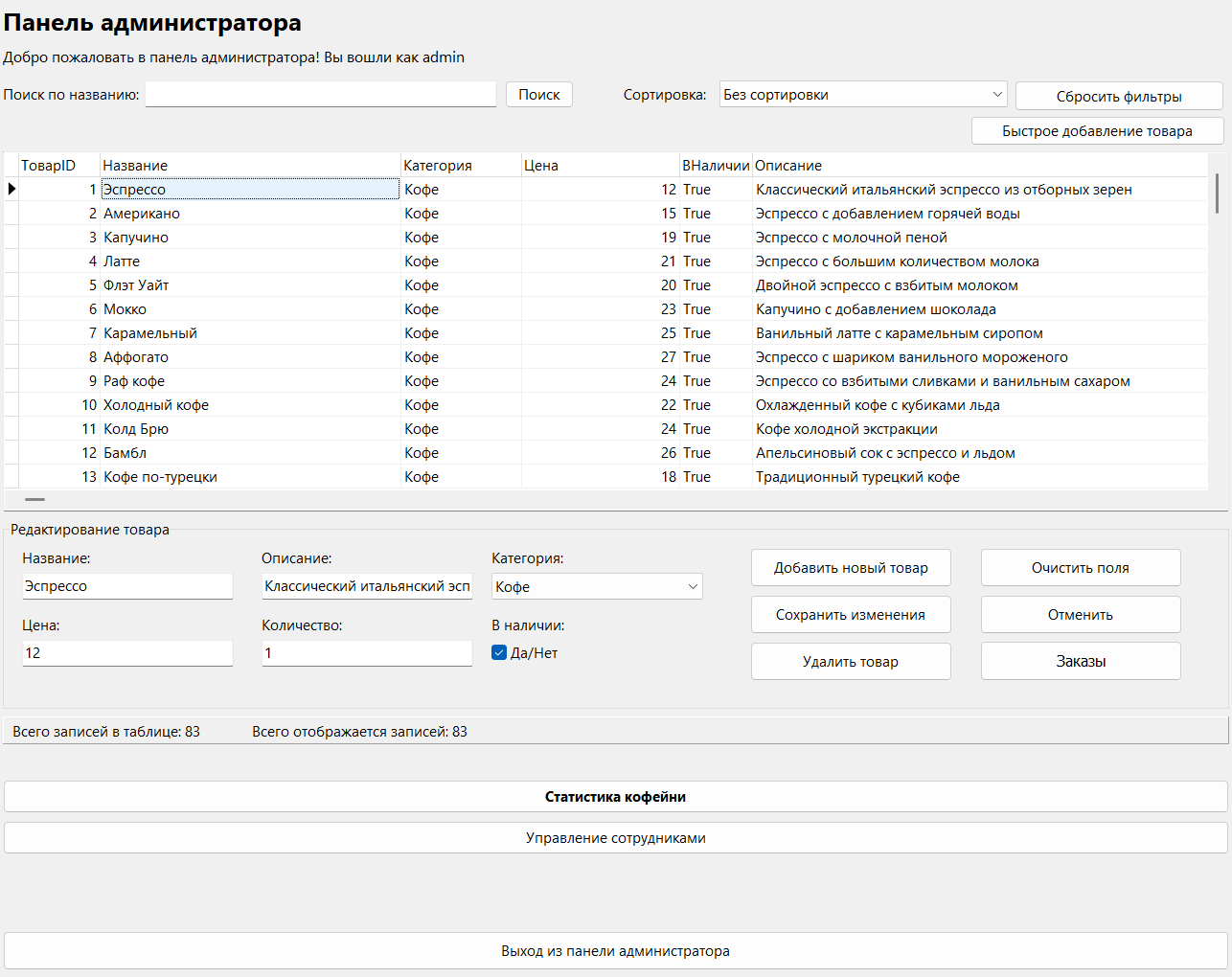


Рис. 4.5. Редактирование товара

Для удаления товара следует выбрать его в таблице и нажать кнопку «Удалить», рис. 4.6. Система запросит подтверждение операции. После подтверждения товар будет удален из базы данных.

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

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4.6. Удаление товара

Для поиска товаров можно использовать поле поиска в верхней части формы, рис. 4.7. После ввода поискового запроса необходимо нажать кнопку поиска или клавишу «Enter». В таблице отобразятся только товары, соответствующие запросу.

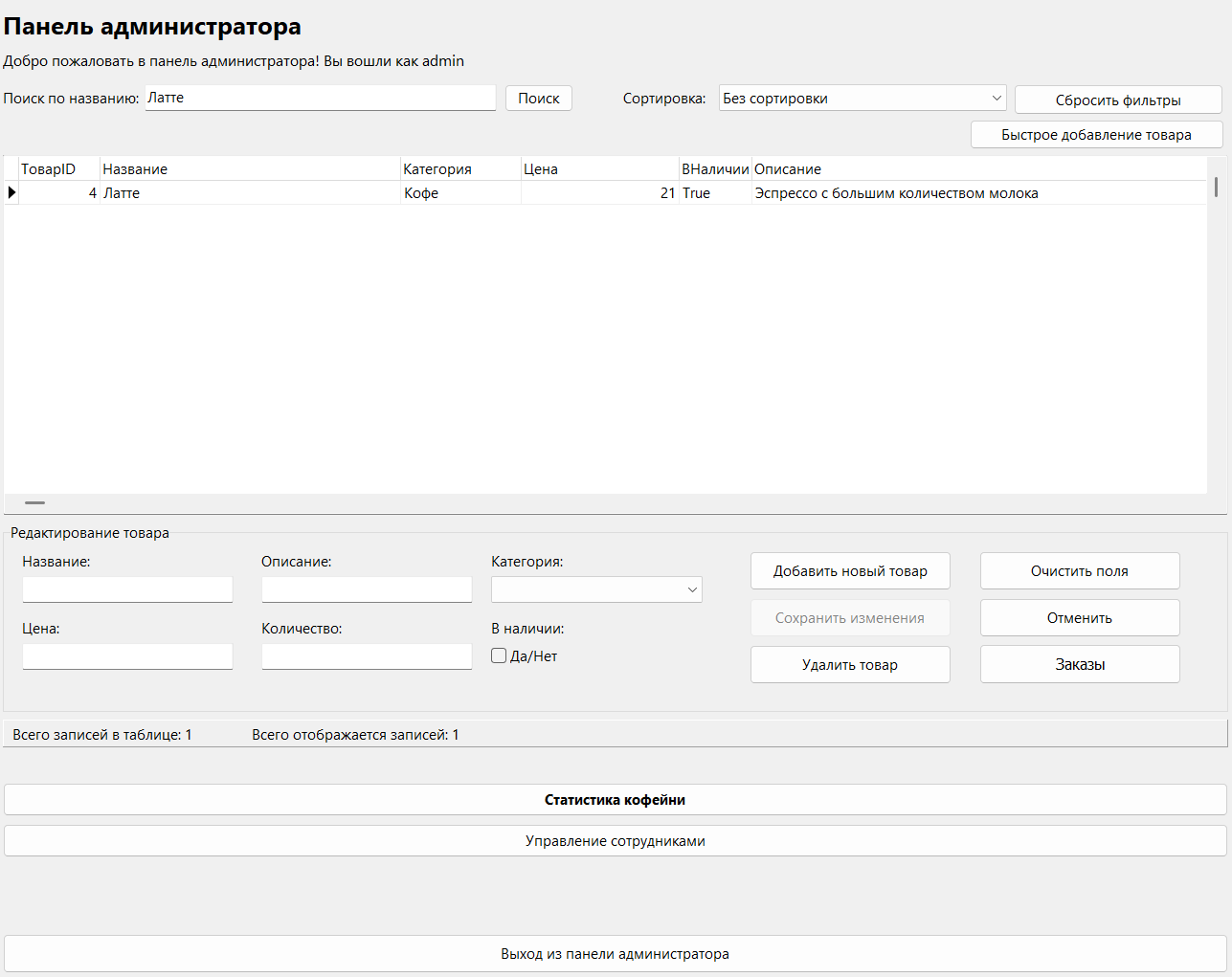


Рис. 4.7. Поиск товаров

Для сортировки списка товаров можно использовать выпадающий список «Сортировка», рис. 4.8. Доступны варианты сортировки по названию, цене и категории в порядке возрастания или убывания.

Изображение выглядит как текст, снимок экрана, число, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Рис. 4.8. Сортировка товаров

Для просмотра статистики продаж администратор может использовать форму статистики, доступную по нажатию кнопки «Статистика», рис. 4.9. На этой форме можно выбирать различные параметры отображения статистической информации и экспортировать отчеты.

Изображение выглядит как текст, снимок экрана, дисплей, число

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 4.9. Форма статистики

Обычный пользователь после авторизации попадает на форму просмотра товаров с ограниченным функционалом, рис. 4.10. Здесь доступны только функции просмотра, поиска, сортировки и фильтрации товаров, без возможности изменения данных.

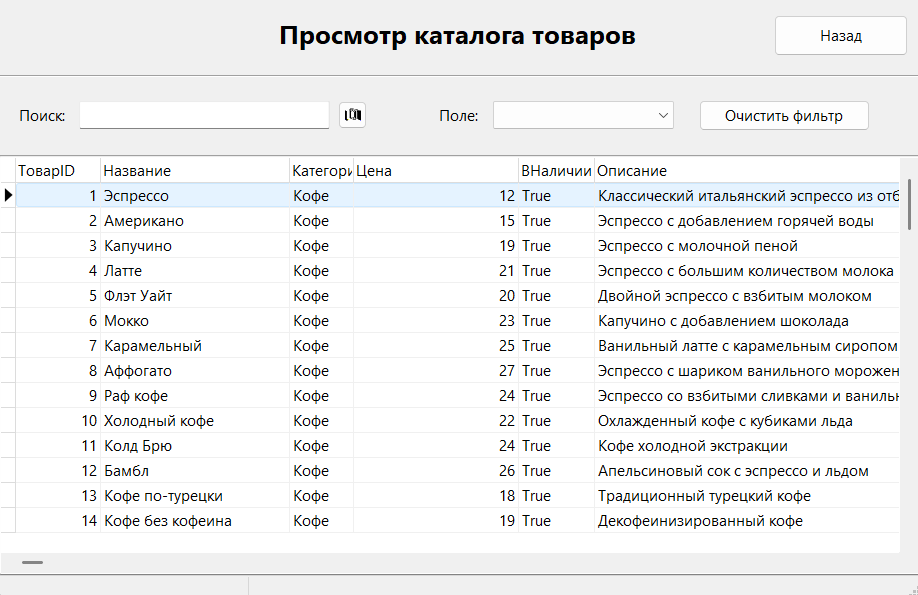


Рис. 4.10. Форма просмотра товаров для обычного пользователя

Для завершения работы с программой необходимо нажать кнопку «Выход» на соответствующей форме, рис. 4.11. Система завершит текущий сеанс и вернется к форме авторизации.

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

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рис. 4.11. Завершение работы с программой

ЗАКЛЮЧЕНИЕ

В ходе разработки курсового проекта по программированию на тему «Программное средство для автоматизации работы кофейни» были выполнены следующие этапы:

1. Проведен анализ существующих систем управления товарами и базами данных, выявлены их достоинства и недостатки.
2. Разработана архитектура приложения, определены основные модули и инструменты для реализации требуемой функциональности.
3. Написан код и реализованы основные функции, такие как авторизация пользователей, управление товарами (добавление, редактирование, удаление), поиск, сортировка и фильтрация товаров, формирование заказов и построение статистики продаж.
4. Проведены тестирование и отладка разработанного приложения с использованием методологии «черного ящика».

Для пользователя реализованы следующие функции:

* + возможность регистрации и авторизации в системе;
  + возможность просмотра каталога товаров;
  + возможность поиска товаров по названию;
  + возможность сортировки товаров по различным параметрам;
  + возможность фильтрации товаров по категориям и наличию;
  + возможность формирования заказов из выбранных товаров.

Для администратора дополнительно реализованы следующие функции:

* + возможность добавления, редактирования и удаления товаров;
  + возможность управления данными о сотрудниках;
  + возможность просмотра и анализа статистики продаж;
  + возможность формирования и экспорта отчетов.

В результате работы была разработана полнофункциональная информационная система управления кофейни, которая позволяет эффективно организовать учет товаров и взаимодействие с клиентами. Система обеспечивает разграничение прав доступа между различными категориями пользователей и предоставляет удобный интерфейс для работы с данными.

Проект разработан с использованием среды C++ Builder и базы данных Microsoft Access, что обеспечивает высокую производительность, надежность и простоту сопровождения системы. Использование технологии ADO для работы с базой данных позволяет при необходимости относительно легко перейти на другие СУБД без существенного изменения кода приложения.

В перспективе система может быть расширена дополнительными функциями, такими как интеграция с системами онлайн-оплаты, автоматическое формирование заказов поставщикам при снижении запасов товаров до критического уровня, реализация модуля лояльности для постоянных клиентов и другими возможностями, повышающими эффективность управления торговым предприятием.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

Архангельский А. Я. Программирование в C++ Builder / А.Я. Архангельский.А. Я.осква: Бином, 2010. – 1304 с.

Архангельский А.Я. Приемы программирования в C++ Builder / А.Я. Архангельский. – Москва: Бином, 2010. – 992 с.

Герберт Шилдт. C++: базовый курс / Шилдт Герберт. – Москва: Вильямс, 2019. – 624 с.

Культин Н.Б. C++ Builder. Освой самостоятельно / Н.Б. Культин. – Санкт-Петербург: БХВ-Петербург, 2015. – 464 с.

Макдональд М. Microsoft Access 2010. Самоучитель / М. Макдональд. – Санкт-Петербург: БХВ-Петербург, 2011. – 432 с.

Прата С. Язык программирования C++. Лекции и упражнения / С. Прата. – Москва: Вильямс, 2020. – 1248 с.

Страуструп Б. Язык программирования C++ / Б. Страуструп. – Москва: Бином, 2017. – 1136 с.

cplusplus.com [Электронный ресурс] / Справочник по языку C++. – Режим доступа: <https://cplusplus.com/reference/>– Дата доступа: 15.05.2025.

docs.microsoft.com [Электронный ресурс] / Документация по Microsoft Access. – Режим доступа: <https://docs.microsoft.com/ru-ru/office/client-developer/access/access-home>– Дата доступа: 28.05.2025.

embarcadero.com [Электронный ресурс] / Официальная документация C++ Builder. – Режим доступа: <https://www.embarcadero.com>– Дата доступа: 25.05.2025.

ПРИЛОЖЕНИЕ 1 (ОБЯЗАТЕЛЬНОЕ) – ЛИСТИНГ ПРОГРАММЫ

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit2.h"

#include "Unit6.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

    : TForm(Owner)

{

}

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

    if (!ADOConnection1->Connected) {

        ADOConnection1->ConnectionString =

            "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Users\\тихон\\OneDrive\\Рабочий стол\\Users.mdb;Persist Security Info=False";

        try {

            ADOConnection1->Connected = true;

        } catch (Exception &E) {

            ShowMessage("Ошибка подключения к базе данных: " + E.Message);

        }

    }

    try {

        ADOQuery1->Close();

        ADOQuery1->SQL->Clear();

        ADOQuery1->SQL->Add("SELECT \* FROM Пользователи");

        ADOQuery1->Open();

    } catch (Exception &E) {

        ShowMessage("Ошибка при загрузке данных: " + E.Message);

    }

}

void \_\_fastcall TForm1::Button1Click(TObject \*Sender)

{

    if (Edit1->Text == "" || Edit2->Text == "" || Edit3->Text == "") {

        ShowMessage("Все поля должны быть заполнены!");

        return;

    }

    if (Edit2->Text.Length() < 6) {

        ShowMessage("Пароль должен содержать не менее 6 символов!");

        Edit2->SetFocus();

        return;

    }

    if (Edit2->Text != Edit3->Text) {

        ShowMessage("Пароли не совпадают!");

        return;

    }

    try {

        ADOQuery1->Close();

        ADOQuery1->SQL->Clear();

        ADOQuery1->SQL->Add("SELECT \* FROM Пользователи WHERE Логин = :username");

        ADOQuery1->Parameters->ParamByName("username")->Value = Edit1->Text;

        ADOQuery1->Open();

        if (!ADOQuery1->IsEmpty()) {

            if (MessageDlg("Пользователь с таким именем уже существует! Хотите войти в систему?",

                mtConfirmation, TMsgDlgButtons() << mbYes << mbNo, 0) == mrYes) {

                Form2->Edit1->Text = Edit1->Text;

                Form2->Show();

                Form1->Hide();

            }

            return;

        }

        ADOQuery1->Close();

        ADOQuery1->SQL->Clear();

        ADOQuery1->SQL->Add("INSERT INTO Пользователи (Логин, Пароль) VALUES (:username, :password)");

        ADOQuery1->Parameters->ParamByName("username")->Value = Edit1->Text;

        ADOQuery1->Parameters->ParamByName("password")->Value = Edit2->Text;

        ADOQuery1->ExecSQL();

        ShowMessage("Регистрация успешно завершена!");

        Form6->IsAdmin = false;

        Form6->Show();

        Form1->Hide();

    } catch (Exception &E) {

        ShowMessage("Ошибка при регистрации: " + E.Message);

    }

}

void \_\_fastcall TForm1::Button2Click(TObject \*Sender)

{

    Form2->Show();

    Form1->Hide();

}

#include <vcl.h>

#pragma hdrstop

#include "Unit2.h"

#include "Unit1.h"

#include "Unit3.h"

#include "Unit6.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm2 \*Form2;

\_\_fastcall TForm2::TForm2(TComponent\* Owner)

    : TForm(Owner)

{

}

void \_\_fastcall TForm2::FormCreate(TObject \*Sender)

{

    if (!ADOConnection1->Connected) {

        ADOConnection1->ConnectionString =

            "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=C:\\Users\\тихон\\OneDrive\\Рабочий стол\\Users.mdb;Persist Security Info=False";

        try {

            ADOConnection1->Connected = true;

        } catch (Exception &E) {

            ShowMessage("Ошибка подключения к базе данных: " + E.Message);

        }

    }

}

void \_\_fastcall TForm2::Button1Click(TObject \*Sender)

{

    if (Edit1->Text == "" || Edit2->Text == "") {

        ShowMessage("Введите имя пользователя и пароль!");

        return;

    }

    if (Edit1->Text == "admin" && Edit2->Text == "1111") {

        Form3->Show();

        Form2->Hide();

        return;

    }

    if (Edit2->Text.Length() < 6) {

        ShowMessage("Пароль должен содержать не менее 6 символов!");

        Edit2->SetFocus();

        return;

    }

    try {

        ADOQuery1->Close();

        ADOQuery1->SQL->Clear();

        ADOQuery1->SQL->Add("SELECT \* FROM Пользователи WHERE Логин = :username");

        ADOQuery1->Parameters->ParamByName("username")->Value = Edit1->Text;

        ADOQuery1->Open();

        if (ADOQuery1->IsEmpty()) {

            if (MessageDlg("Пользователь с таким именем не найден. Хотите зарегистрироваться?",

                mtConfirmation, TMsgDlgButtons() << mbYes << mbNo, 0) == mrYes) {

                Form1->Edit1->Text = Edit1->Text;

                Form1->Show();

                Form2->Hide();

            }

        } else {

            if (ADOQuery1->FieldByName("Пароль")->AsString == Edit2->Text) {

                ShowMessage("Вход выполнен успешно!");

                Form6->IsAdmin = false;

                Form6->Show();

                Form2->Hide();

            } else {

                ShowMessage("Неверный пароль!");

            }

        }

    } catch (Exception &E) {

        ShowMessage("Ошибка при проверке учетных данных: " + E.Message);

    }

}

void \_\_fastcall TForm2::Button2Click(TObject \*Sender)

{

    Form1->Show();

    Form2->Hide();

}

void \_\_fastcall TForm2::FormKeyPress(TObject \*Sender, System::WideChar &Key)

{

    if (Key == VK\_RETURN) {

        Button1Click(Sender);

        Key = 0;

    }

}

void \_\_fastcall TForm2::Edit2KeyPress(TObject \*Sender, System::WideChar &Key)

{

    if (Key == VK\_RETURN) {

        Button1Click(Sender);

        Key = 0;

    }

}

#include <vcl.h>

#pragma hdrstop

#include "Unit3.h"

#include "Unit2.h"

#include "Unit4.h"

#include "Unit5.h"

#include "Unit7.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm3 \*Form3;

extern PACKAGE TForm7 \*Form7;

\_\_fastcall TForm3::TForm3(TComponent\* Owner)

    : TForm(Owner)

{

    CurrentProductID = -1;

    EditMode = false;

}

void \_\_fastcall TForm3::FormShow(TObject \*Sender)

{

    UpdateRecordCount();

    EditMode = false;

    ButtonEdit->Enabled = false;

    this->ComboBoxSort->ItemIndex = 0;

    if (ComboBoxCategory->Items->Count == 0) {

        try {

            TADOQuery\* tempQuery = new TADOQuery(this);

            tempQuery->Connection = ADOConnection1;

            tempQuery->SQL->Text = "SELECT DISTINCT Категория FROM Товары WHERE Категория IS NOT NULL ORDER BY Категория";

            tempQuery->Open();

            while (!tempQuery->Eof) {

                String category = tempQuery->FieldByName("Категория")->AsString;

                if (!category.IsEmpty() && ComboBoxCategory->Items->IndexOf(category) == -1) {

                    ComboBoxCategory->Items->Add(category);

                }

                tempQuery->Next();

            }

            tempQuery->Close();

            delete tempQuery;

        } catch (Exception &e) {

        }

    }

}

void \_\_fastcall TForm3::UpdateRecordCount()

{

    if (ADOQuery1->Active) {

        int totalRecords = 0;

        int visibleRecords = 0;

        ADOQuery1->DisableControls();

        ADOQuery1->First();

        while (!ADOQuery1->Eof) {

            totalRecords++;

            visibleRecords++;

            ADOQuery1->Next();

        }

        ADOQuery1->First();

        ADOQuery1->EnableControls();

        Label5->Caption = "Всего записей в таблице: " + IntToStr(totalRecords);

        Label6->Caption = "Всего отображается записей: " + IntToStr(visibleRecords);

    } else {

        Label5->Caption = "Всего записей в таблице: 0";

        Label6->Caption = "Всего отображается записей: 0";

    }

}

void \_\_fastcall TForm3::Button1Click(TObject \*Sender)

{

    ApplyFilter();

}

void \_\_fastcall TForm3::Edit1Change(TObject \*Sender)

{

    ApplyFilter();

}

void \_\_fastcall TForm3::ApplyFilter()

{

    if (ADOQuery1->Active) {

        String searchText = Edit1->Text.Trim();

        if (!searchText.IsEmpty()) {

            ADOQuery1->Filtered = false;

            ADOQuery1->Filter = "Название LIKE '%" + searchText + "%'";

            ADOQuery1->Filtered = true;

        } else {

            ADOQuery1->Filtered = false;

        }

        UpdateRecordCount();

    }

}

void \_\_fastcall TForm3::ComboBoxSortChange(TObject \*Sender)

{

    ApplySort();

}

void \_\_fastcall TForm3::ApplySort()

{

    if (ADOQuery1->Active) {

        switch (this->ComboBoxSort->ItemIndex) {

            case 0:

                ADOQuery1->Sort = "";

                break;

            case 1:

                ADOQuery1->Sort = "Название ASC";

                break;

            case 2:

                ADOQuery1->Sort = "Название DESC";

                break;

            case 3:

                ADOQuery1->Sort = "Цена ASC";

                break;

            case 4:

                ADOQuery1->Sort = "Цена DESC";

                break;

            case 5:

                ADOQuery1->Sort = "ВНаличии ASC";

                break;

            case 6:

                ADOQuery1->Sort = "ВНаличии DESC";

                break;

        }

        RefreshData();

    }

}

void \_\_fastcall TForm3::Button2Click(TObject \*Sender)

{

    if (ADOQuery1->Active) {

        Edit1->Text = "";

        this->ComboBoxSort->ItemIndex = 0;

        ADOQuery1->Filtered = false;

        ADOQuery1->Filter = "";

        ADOQuery1->Sort = "";

        RefreshData();

    }

}

void \_\_fastcall TForm3::RefreshData()

{

    if (ADOQuery1->Active) {

        TBookmark bookmark;

        if (!ADOQuery1->IsEmpty()) {

            bookmark = ADOQuery1->Bookmark;

        }

        ADOQuery1->Refresh();

        if (!ADOQuery1->IsEmpty() && bookmark.Length > 0) {

            try {

                ADOQuery1->Bookmark = bookmark;

            } catch (...) {

                ADOQuery1->First();

            }

        }

        UpdateRecordCount();

    }

}

void \_\_fastcall TForm3::ClearInputFields()

{

    EditName->Text = "";

    EditPrice->Text = "";

    EditDescription->Text = "";

    EditQuantity->Text = "";

    CheckBoxInStock->Checked = false;

    ComboBoxCategory->ItemIndex = -1;

    EditMode = false;

    CurrentProductID = -1;

    ButtonEdit->Enabled = false;

}

void \_\_fastcall TForm3::Button4Click(TObject \*Sender)

{

    Form2->Show();

    Form3->Hide();

}

void \_\_fastcall TForm3::ButtonEmployeesClick(TObject \*Sender)

{

    Form4->Show();

}

void \_\_fastcall TForm3::ButtonStatisticsClick(TObject \*Sender)

{

    Form5->Show();

}

void \_\_fastcall TForm3::ButtonCartClick(TObject \*Sender)

{

    Form7->Show();

}

#include <vcl.h>

#pragma hdrstop

#include "Unit4.h"

#include "Unit3.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm4 \*Form4;

\_\_fastcall TForm4::TForm4(TComponent\* Owner)

    : TForm(Owner)

{

    CurrentEmployeeID = -1;

    EditMode = false;

}

void \_\_fastcall TForm4::FormShow(TObject \*Sender)

{

    UpdateRecordCount();

    EditMode = false;

    ButtonEdit->Enabled = false;

    this->ComboBoxSort->ItemIndex = 0;

}

void \_\_fastcall TForm4::UpdateRecordCount()

{

    if (ADOQuery1->Active) {

        int totalRecords = 0;

        int visibleRecords = 0;

        ADOQuery1->DisableControls();

        ADOQuery1->First();

        while (!ADOQuery1->Eof) {

            totalRecords++;

            visibleRecords++;

            ADOQuery1->Next();

        }

        ADOQuery1->First();

        ADOQuery1->EnableControls();

        Label5->Caption = "Всего записей в таблице: " + IntToStr(totalRecords);

        Label6->Caption = "Всего отображается записей: " + IntToStr(visibleRecords);

    } else {

        Label5->Caption = "Всего записей в таблице: 0";

        Label6->Caption = "Всего отображается записей: 0";

    }

}

void \_\_fastcall TForm4::ButtonSearchClick(TObject \*Sender)

{

    ApplyFilter();

}

void \_\_fastcall TForm4::EditSearchChange(TObject \*Sender)

{

    ApplyFilter();

}

void \_\_fastcall TForm4::ApplyFilter()

{

    if (ADOQuery1->Active) {

        String searchText = EditSearch->Text.Trim();

        if (!searchText.IsEmpty()) {

            ADOQuery1->Filtered = false;

            ADOQuery1->Filter = "Имя LIKE '%" + searchText + "%'";

            ADOQuery1->Filtered = true;

        } else {

            ADOQuery1->Filtered = false;

        }

        UpdateRecordCount();

    }

}

void \_\_fastcall TForm4::ComboBoxSortChange(TObject \*Sender)

{

    ApplySort();

}

void \_\_fastcall TForm4::ApplySort()

{

    if (ADOQuery1->Active) {

        switch (this->ComboBoxSort->ItemIndex) {

            case 0:

                ADOQuery1->Sort = "";

                break;

            case 1:

                ADOQuery1->Sort = "Имя ASC";

                break;

            case 2:

                ADOQuery1->Sort = "Имя DESC";

                break;

            case 3:

                ADOQuery1->Sort = "Должность ASC";

                break;

            case 4:

                ADOQuery1->Sort = "Должность DESC";

                break;

            case 5:

                if (ADOQuery1->FindField("Зарплата") != nullptr) {

                    ADOQuery1->Sort = "Зарплата ASC";

                } else {

                    ShowMessage("Поле 'Зарплата' не найдено в таблице");

                    this->ComboBoxSort->ItemIndex = 0;

                }

                break;

            case 6:

                if (ADOQuery1->FindField("Зарплата") != nullptr) {

                    ADOQuery1->Sort = "Зарплата DESC";

                } else {

                    ShowMessage("Поле 'Зарплата' не найдено в таблице");

                    this->ComboBoxSort->ItemIndex = 0;

                }

                break;

        }

        RefreshData();

    }

}

void \_\_fastcall TForm4::ButtonResetClick(TObject \*Sender)

{

    if (ADOQuery1->Active) {

        EditSearch->Text = "";

        this->ComboBoxSort->ItemIndex = 0;

        ADOQuery1->Filtered = false;

        ADOQuery1->Filter = "";

        ADOQuery1->Sort = "";

        RefreshData();

    }

}

void \_\_fastcall TForm4::RefreshData()

{

    if (ADOQuery1->Active) {

        TBookmark bookmark;

        if (!ADOQuery1->IsEmpty()) {

            bookmark = ADOQuery1->Bookmark;

        }

        ADOQuery1->Refresh();

        if (!ADOQuery1->IsEmpty() && bookmark.Length > 0) {

            try {

                ADOQuery1->Bookmark = bookmark;

            } catch (...) {

                ADOQuery1->First();

            }

        }

        UpdateRecordCount();

    }

}

void \_\_fastcall TForm4::ClearInputFields()

{

    EditName->Text = "";

    EditPosition->Text = "";

    EditPhone->Text = "";

    EditSalary->Text = "";

    EditMode = false;

    CurrentEmployeeID = -1;

    ButtonEdit->Enabled = false;

}

void \_\_fastcall TForm4::ButtonBackClick(TObject \*Sender)

{

    Form3->Show();

    Form4->Hide();

}

#include <vcl.h>

#pragma hdrstop

#include "Unit5.h"

#include "Unit3.h"

#include <memory>

#include <algorithm>

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm5 \*Form5;

\_\_fastcall TForm5::TForm5(TComponent\* Owner)

    : TForm(Owner)

{

    this->TotalSum = 0;

    Orders.clear();

    try {

        if (ComboBoxSort->Items->Count == 0) {

            ComboBoxSort->Items->Add("По названию (А-Я)");

            ComboBoxSort->Items->Add("По названию (Я-А)");

            ComboBoxSort->Items->Add("По цене (возрастание)");

            ComboBoxSort->Items->Add("По цене (убывание)");

            ComboBoxSort->Items->Add("По количеству (возрастание)");

            ComboBoxSort->Items->Add("По количеству (убывание)");

            ComboBoxSort->Items->Add("По дате (новые-старые)");

            ComboBoxSort->Items->Add("По дате (старые-новые)");

            ComboBoxSort->ItemIndex = 0;

        }

        ButtonDelete->Visible = false;

        BitBtnSave->Caption = "Сохранить отчет";

        LabelSearch->Caption = "Поиск (название, дата):";

    } catch (...) {

    }

}

\_\_fastcall TForm5::~TForm5()

{

}

void \_\_fastcall TForm5::FormCreate(TObject \*Sender)

{

    try {

        Orders.clear();

        TotalSum = 0;

        String exePath = ExtractFilePath(Application->ExeName);

        String imagePath = exePath + "images\\coffee\_bg.jpg";

        try {

            if (FileExists(imagePath)) {

                ImageBackground->Picture->LoadFromFile(imagePath);

            }

        } catch (Exception &e) {

        }

        ComboBoxSort->Items->Clear();

        ComboBoxSort->Items->Add("По названию (А-Я)");

        ComboBoxSort->Items->Add("По названию (Я-А)");

        ComboBoxSort->Items->Add("По цене (возрастание)");

        ComboBoxSort->Items->Add("По цене (убывание)");

        ComboBoxSort->Items->Add("По количеству (возрастание)");

        ComboBoxSort->Items->Add("По количеству (убывание)");

        ComboBoxSort->Items->Add("По категории (А-Я)");

        ComboBoxSort->Items->Add("По категории (Я-А)");

        ComboBoxSort->ItemIndex = 0;

        ButtonDelete->Visible = false;

        if (!ADOConnection1->Connected) {

            String dbPath = exePath + "ProductsDB.mdb";

            ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath + ";Persist Security Info=False";

            ADOConnection1->LoginPrompt = false;

            try {

                ADOConnection1->Connected = true;

            } catch (Exception &e) {

                ShowMessage("Ошибка при подключении к базе данных: " + e.Message);

            }

        }

        ADOQuery1->Connection = ADOConnection1;

        ADOQuery1->SQL->Text = "SELECT \* FROM [Товары] ORDER BY [Название]";

        DataSource1->DataSet = ADOQuery1;

        DBGrid1->DataSource = DataSource1;

        if (DBGrid1->Columns->Count == 0) {

            DBGrid1->Columns->Clear();

            TColumn \*colName = DBGrid1->Columns->Add();

            colName->FieldName = "Название";

            colName->Title->Caption = "Название товара";

            colName->Width = 420;

            TColumn \*colPrice = DBGrid1->Columns->Add();

            colPrice->FieldName = "Цена";

            colPrice->Title->Caption = "Цена (руб.)";

            colPrice->Width = 120;

            TColumn \*colQuantity = DBGrid1->Columns->Add();

            colQuantity->FieldName = "Количество шт";

            colQuantity->Title->Caption = "Количество";

            colQuantity->Width = 120;

            TColumn \*colCategory = DBGrid1->Columns->Add();

            colCategory->FieldName = "Категория";

            colCategory->Title->Caption = "Категория";

            colCategory->Width = 150;

        }

    } catch (Exception &e) {

        ShowMessage("Ошибка при инициализации формы: " + e.Message);

    }

}

void \_\_fastcall TForm5::FormShow(TObject \*Sender)

{

    try {

        if (!ADOConnection1->Connected) {

            String exePath = ExtractFilePath(Application->ExeName);

            String dbPath = exePath + "ProductsDB.mdb";

            ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath + ";Persist Security Info=False";

            try {

                ADOConnection1->Connected = true;

            } catch (Exception &e) {

                ShowMessage("Ошибка подключения к БД: " + e.Message);

            }

        }

        if (ImageBackground->Picture->Graphic == NULL) {

            String exePath = ExtractFilePath(Application->ExeName);

            String imagePath = exePath + "images\\coffee\_bg.jpg";

            try {

                if (FileExists(imagePath)) {

                    ImageBackground->Picture->LoadFromFile(imagePath);

                }

            } catch (Exception &e) {

            }

        }

        if (ComboBoxSort->Items->Count == 0) {

            ComboBoxSort->Items->Add("По названию (А-Я)");

            ComboBoxSort->Items->Add("По названию (Я-А)");

            ComboBoxSort->Items->Add("По цене (возрастание)");

            ComboBoxSort->Items->Add("По цене (убывание)");

            ComboBoxSort->Items->Add("По количеству (возрастание)");

            ComboBoxSort->Items->Add("По количеству (убывание)");

            ComboBoxSort->Items->Add("По категории (А-Я)");

            ComboBoxSort->Items->Add("По категории (Я-А)");

            ComboBoxSort->ItemIndex = 0;

        }

        DBGrid1->Visible = true;

        ButtonDelete->Visible = false;

        DataSource1->DataSet = ADOQuery1;

        DBGrid1->DataSource = DataSource1;

        if (ADOQuery1->Active) {

            ADOQuery1->Close();

        }

        ADOQuery1->SQL->Text = "SELECT \* FROM [Товары] ORDER BY [Название]";

        try {

            ADOQuery1->Open();

            if (DBGrid1->Columns->Count == 0) {

                DBGrid1->Columns->Clear();

                TColumn \*colName = DBGrid1->Columns->Add();

                colName->FieldName = "Название";

                colName->Title->Caption = "Название товара";

                colName->Width = 420;

                TColumn \*colPrice = DBGrid1->Columns->Add();

                colPrice->FieldName = "Цена";

                colPrice->Title->Caption = "Цена (руб.)";

                colPrice->Width = 120;

                TColumn \*colQuantity = DBGrid1->Columns->Add();

                colQuantity->FieldName = "Количество шт";

                colQuantity->Title->Caption = "Количество";

                colQuantity->Width = 120;

                TColumn \*colCategory = DBGrid1->Columns->Add();

                colCategory->FieldName = "Категория";

                colCategory->Title->Caption = "Категория";

                colCategory->Width = 150;

            }

        } catch (Exception &e) {

            ShowMessage("Ошибка при открытии таблицы: " + e.Message);

        }

    } catch (Exception &e) {

        ShowMessage("Ошибка при отображении формы: " + e.Message);

    }

}

void \_\_fastcall TForm5::BitBtnBackClick(TObject \*Sender)

{

    Form3->Show();

    Form5->Hide();

}

#include <vcl.h>

#pragma hdrstop

#include "Unit6.h"

#include "Unit2.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm6 \*Form6;

\_\_fastcall TForm6::TForm6(TComponent\* Owner)

    : TForm(Owner)

{

    IsAdmin = false;

}

void \_\_fastcall TForm6::FormCreate(TObject \*Sender)

{

    ADOConnection1->ConnectionString =

        "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=.\\Users.mdb;Persist Security Info=False";

    try {

        ADOConnection1->Connected = true;

        ComboBoxFilter->Items->Add("Все поля");

        ComboBoxFilter->Items->Add("Наименование");

        ComboBoxFilter->Items->Add("Категория");

        ComboBoxFilter->Items->Add("Цена");

        ComboBoxFilter->ItemIndex = 0;

        ComboBoxSort->Items->Add("Без сортировки");

        ComboBoxSort->Items->Add("По названию (от А до Я)");

        ComboBoxSort->Items->Add("По названию (от Я до А)");

        ComboBoxSort->Items->Add("По цене (по возрастанию)");

        ComboBoxSort->Items->Add("По цене (по убыванию)");

        ComboBoxSort->Items->Add("По количеству (по возрастанию)");

        ComboBoxSort->Items->Add("По количеству (по убыванию)");

        ComboBoxSort->ItemIndex = 0;

        RefreshData();

        if (IsAdmin) {

            Caption = "Управление товарами (Администратор)";

            Label1->Caption = "Управление товарами";

        } else {

            Caption = "Каталог товаров";

            Label1->Caption = "Просмотр каталога товаров";

        }

    } catch (Exception &E) {

        ShowMessage("Ошибка подключения к базе данных: " + E.Message);

    }

}

void \_\_fastcall TForm6::RefreshData()

{

    try {

        ADOQuery1->Close();

        ADOQuery1->SQL->Clear();

        ADOQuery1->SQL->Add("SELECT \* FROM [Товары]");

        try {

            ADOQuery1->Open();

            DBGrid1->Columns->Clear();

            TColumn\* columnID = DBGrid1->Columns->Add();

            columnID->FieldName = "ID";

            columnID->Title->Caption = "№";

            columnID->Width = 50;

            TColumn\* columnName = DBGrid1->Columns->Add();

            columnName->FieldName = "Название";

            columnName->Title->Caption = "Наименование товара";

            columnName->Width = 200;

            TColumn\* columnCategory = DBGrid1->Columns->Add();

            columnCategory->FieldName = "Категория";

            columnCategory->Title->Caption = "Категория";

            columnCategory->Width = 150;

            TColumn\* columnPrice = DBGrid1->Columns->Add();

            columnPrice->FieldName = "Цена";

            columnPrice->Title->Caption = "Цена";

            columnPrice->Width = 100;

            TColumn\* columnQuantity = DBGrid1->Columns->Add();

            columnQuantity->FieldName = "Количество шт";

            columnQuantity->Title->Caption = "Количество";

            columnQuantity->Width = 100;

            DBGrid1->DataSource = DataSource1;

            DataSource1->DataSet = ADOQuery1;

            ApplySort();

            UpdateStatusBar();

        } catch (Exception &E) {

            ShowMessage("Ошибка при открытии таблицы: " + E.Message);

            ADOQuery1->Close();

            ADOQuery1->SQL->Clear();

            ADOQuery1->SQL->Add("SELECT \* FROM Products");

            try {

                ADOQuery1->Open();

                UpdateStatusBar();

            } catch (Exception &E2) {

                ShowMessage("Не удалось открыть таблицу. Проверьте подключение к базе данных.");

            }

        }

    } catch (Exception &E) {

        ShowMessage("Ошибка при загрузке данных: " + E.Message);

    }

}

void \_\_fastcall TForm6::ApplySort()

{

    if (ADOQuery1->Active) {

        String currentSQL = ADOQuery1->SQL->Text;

        String orderByClause = "";

        String nameField = "[Название]";

        String priceField = "[Цена]";

        String quantityField = "[Количество шт]";

        if (!ADOQuery1->FindField("Название") && ADOQuery1->FindField("Наименование")) {

            nameField = "[Наименование]";

        }

        if (!ADOQuery1->FindField("Количество шт") && ADOQuery1->FindField("Количество")) {

            quantityField = "[Количество]";

        }

        switch (ComboBoxSort->ItemIndex) {

            case 0:

                break;

            case 1:

                orderByClause = " ORDER BY " + nameField + " ASC";

                break;

            case 2:

                orderByClause = " ORDER BY " + nameField + " DESC";

                break;

            case 3:

                orderByClause = " ORDER BY " + priceField + " ASC";

                break;

            case 4:

                orderByClause = " ORDER BY " + priceField + " DESC";

                break;

            case 5:

                orderByClause = " ORDER BY " + quantityField + " ASC";

                break;

            case 6:

                orderByClause = " ORDER BY " + quantityField + " DESC";

                break;

        }

        if (!orderByClause.IsEmpty()) {

            int orderByPos = currentSQL.UpperCase().Pos(" ORDER BY ");

            if (orderByPos > 0) {

                currentSQL = currentSQL.SubString(1, orderByPos - 1);

            }

            currentSQL += orderByClause;

            ADOQuery1->Close();

            ADOQuery1->SQL->Clear();

            ADOQuery1->SQL->Add(currentSQL);

            try {

                ADOQuery1->Open();

            } catch (Exception &E) {

                ShowMessage("Ошибка при сортировке: " + E.Message);

                RefreshData();

            }

        }

    }

}

void \_\_fastcall TForm6::UpdateStatusBar()

{

    if (ADOQuery1->Active) {

        int recordCount = ADOQuery1->RecordCount;

        StatusBar1->Panels->Items[0]->Text = "Всего записей: " + IntToStr(recordCount);

    } else {

        StatusBar1->Panels->Items[0]->Text = "Нет данных";

    }

}

void \_\_fastcall TForm6::ButtonBackClick(TObject \*Sender)

{

    Form2->Show();

    Form6->Hide();

}

void \_\_fastcall TForm6::EditSearchChange(TObject \*Sender)

{

    ApplyFilter();

}

void \_\_fastcall TForm6::ApplyFilter()

{

    if (ADOQuery1->Active) {

        String searchText = EditSearch->Text.Trim();

        if (!searchText.IsEmpty()) {

            ADOQuery1->Filtered = false;

            ADOQuery1->Filter = "Название LIKE '%" + searchText + "%'";

            ADOQuery1->Filtered = true;

        } else {

            ADOQuery1->Filtered = false;

        }

        UpdateStatusBar();

    }

}

void \_\_fastcall TForm6::ComboBoxSortChange(TObject \*Sender)

{

    ApplySort();

}

void \_\_fastcall TForm6::ButtonClearFilterClick(TObject \*Sender)

{

    EditSearch->Text = "";

    ComboBoxSort->ItemIndex = 0;

    ADOQuery1->Filtered = false;

    RefreshData();

}

#include <vcl.h>

#pragma hdrstop

#include "Unit7.h"

#include "Unit3.h"

#include "Unit5.h"

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm7 \*Form7;

extern PACKAGE TForm3 \*Form3;

\_\_fastcall TForm7::TForm7(TComponent\* Owner)

    : TForm(Owner)

{

    TotalAmount = 0;

}

void \_\_fastcall TForm7::FormShow(TObject \*Sender)

{

    Label1->Caption = "Добавление товаров";

    MemoCart->Visible = false;

    ButtonSaveOrder->Visible = false;

    ButtonClear->Visible = false;

    LabelTotalCaption->Visible = false;

    LabelTotal->Visible = false;

    Label2->Caption = "Товар:";

    Label3->Caption = "Количество:";

    Label4->Caption = "Категория:";

    Label5->Caption = "Цена:";

    if (!FindComponent("EditPrice")) {

        TEdit \*EditPrice = new TEdit(this);

        EditPrice->Parent = this;

        EditPrice->Name = "EditPrice";

        EditPrice->Left = EditQuantity->Left;

        EditPrice->Top = EditQuantity->Top + EditQuantity->Height + 10;

        EditPrice->Width = EditQuantity->Width;

        EditPrice->Height = EditQuantity->Height;

        EditPrice->Text = "0.00";

        EditPrice->ReadOnly = true;

        EditPrice->Color = clBtnFace;

    } else {

        TEdit \*EditPrice = dynamic\_cast<TEdit\*>(FindComponent("EditPrice"));

        if (EditPrice) {

            EditPrice->ReadOnly = true;

            EditPrice->Color = clBtnFace;

        }

    }

    ButtonAddToCart->Caption = "Добавить товар";

    try {

        if (!ADOConnection1->Connected) {

            String exePath = ExtractFilePath(Application->ExeName);

            String dbPath = exePath + "ProductsDB.mdb";

            ADOConnection1->ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=" + dbPath + ";Persist Security Info=False";

            try {

                ADOConnection1->Connected = true;

            } catch (Exception &e) {

                ShowMessage("Ошибка подключения к базе данных: " + e.Message);

                return;

            }

        }

        TADOQuery \*CheckTable = new TADOQuery(this);

        CheckTable->Connection = ADOConnection1;

        try {

            CheckTable->SQL->Text = "SELECT \* FROM [Товары]";

            CheckTable->Open();

            CheckTable->Close();

        } catch (Exception &e) {

            try {

                CheckTable->SQL->Text =

                    "CREATE TABLE [Товары] ("

                    "[ID] COUNTER PRIMARY KEY, "

                    "[Название] TEXT(100), "

                    "[Цена] CURRENCY, "

                    "[Количество шт] INTEGER, "

                    "[Категория] TEXT(50))";

                CheckTable->ExecSQL();

            } catch (Exception &e2) {

                ShowMessage("Ошибка при создании таблицы 'Товары': " + e2.Message);

            }

        }

        delete CheckTable;

        LoadProductsList();

        ComboBoxCategory->ItemIndex = 0;

        ADOQuery1->Close();

        ADOQuery1->SQL->Text = "SELECT \* FROM [Товары] ORDER BY [Название]";

        try {

            ADOQuery1->Open();

            if (DBGrid1->Columns->Count == 0) {

                DBGrid1->Columns->Clear();

                TColumn \*colName = DBGrid1->Columns->Add();

                colName->FieldName = "Название";

                colName->Title->Caption = "Название товара";

                colName->Width = 250;

                TColumn \*colPrice = DBGrid1->Columns->Add();

                colPrice->FieldName = "Цена";

                colPrice->Title->Caption = "Цена (руб.)";

                colPrice->Width = 100;

                TColumn \*colQuantity = DBGrid1->Columns->Add();

                colQuantity->FieldName = "Количество шт";

                colQuantity->Title->Caption = "Количество";

                colQuantity->Width = 100;

                TColumn \*colCategory = DBGrid1->Columns->Add();

                colCategory->FieldName = "Категория";

                colCategory->Title->Caption = "Категория";

                colCategory->Width = 150;

            }

        } catch (Exception &e) {

            ShowMessage("Ошибка при открытии таблицы 'Товары': " + e.Message);

        }

    } catch (Exception &e) {

        ShowMessage("Ошибка при инициализации базы данных: " + e.Message);

    }

    ComboBoxProducts->SetFocus();

}

void \_\_fastcall TForm7::LoadProductsList()

{

    ComboBoxProducts->Clear();

    const char\* productNames[] = {

        "Эспрессо", "Американо", "Капучино", "Латте", "Флэт Уайт",

        "Мокко", "Карамельный", "Аффогато", "Раф кофе", "Холодный кофе",

        "Колд Брю", "Бамбл", "Кофе по-турецки", "Кофе без кофеина",

        "Зеленый чай", "Черный чай", "Фруктовый чай", "Чай с имбирем и лимоном",

        "Чай масала", "Улун", "Пуэр", "Матча", "Ромашковый чай", "Мятный чай",

        "Круассан классический", "Круассан с миндалем", "Круассан с шоколадом",

        "Булочка с корицей", "Фисташковый рулет", "Маффин с черникой",

        "Маффин шоколадный", "Пирог с яблоком"

    };

    int productCount = sizeof(productNames) / sizeof(productNames[0]);

    for (int i = 0; i < productCount; i++) {

        ComboBoxProducts->Items->Add(String(productNames[i]));

    }

    ComboBoxCategory->Clear();

    ComboBoxCategory->Items->Add("Все товары");

    ComboBoxCategory->Items->Add("Кофе");

    ComboBoxCategory->Items->Add("Чай");

    ComboBoxCategory->Items->Add("Выпечка");

    ComboBoxCategory->Items->Add("Десерты");

    ComboBoxCategory->Items->Add("Напитки");

    ComboBoxCategory->ItemIndex = 0;

}

void \_\_fastcall TForm7::ButtonBackClick(TObject \*Sender)

{

    Form3->Show();

    Form7->Hide();

}

void \_\_fastcall TForm7::ButtonAddToCartClick(TObject \*Sender)

{

    if (ComboBoxProducts->ItemIndex == -1) {

        ShowMessage("Выберите товар из списка!");

        return;

    }

    if (EditQuantity->Text.Trim() == "" || StrToIntDef(EditQuantity->Text, 0) <= 0) {

        ShowMessage("Введите корректное количество товара!");

        EditQuantity->SetFocus();

        return;

    }

    String productName = ComboBoxProducts->Text;

    int quantity = StrToInt(EditQuantity->Text);

    double price = GetProductPrice(productName);

    String category = "Общая";

    if (ComboBoxCategory->ItemIndex > 0) {

        category = ComboBoxCategory->Text;

    }

    if (AddProductToDatabase(productName, category, price, quantity)) {

        ShowMessage("Товар успешно добавлен в базу данных!");

        ComboBoxProducts->ItemIndex = -1;

        EditQuantity->Text = "";

        ComboBoxCategory->ItemIndex = 0;

        try {

            ADOQuery1->Refresh();

        } catch (Exception &e) {

            ShowMessage("Ошибка при обновлении данных: " + e.Message);

        }

    } else {

        ShowMessage("Ошибка при добавлении товара в базу данных!");

    }

}

double \_\_fastcall TForm7::GetProductPrice(String productName)

{

    if (productName.Pos("Эспрессо") > 0) return 120.0;

    if (productName.Pos("Американо") > 0) return 130.0;

    if (productName.Pos("Капучино") > 0) return 150.0;

    if (productName.Pos("Латте") > 0) return 160.0;

    if (productName.Pos("Мокко") > 0) return 170.0;

    if (productName.Pos("чай") > 0 || productName.Pos("Чай") > 0) return 80.0;

    if (productName.Pos("Круассан") > 0) return 90.0;

    if (productName.Pos("Маффин") > 0) return 110.0;

    if (productName.Pos("Пирог") > 0) return 140.0;

    return 100.0;

}

bool \_\_fastcall TForm7::AddProductToDatabase(String name, String category, double price, int quantity)

{

    try {

        TADOQuery \*insertQuery = new TADOQuery(this);

        insertQuery->Connection = ADOConnection1;

        insertQuery->SQL->Text =

            "INSERT INTO [Товары] ([Название], [Цена], [Количество шт], [Категория]) "

            "VALUES (:name, :price, :quantity, :category)";

        insertQuery->Parameters->ParamByName("name")->Value = name;

        insertQuery->Parameters->ParamByName("price")->Value = price;

        insertQuery->Parameters->ParamByName("quantity")->Value = quantity;

        insertQuery->Parameters->ParamByName("category")->Value = category;

        insertQuery->ExecSQL();

        delete insertQuery;

        return true;

    } catch (Exception &e) {

        ShowMessage("Ошибка при добавлении товара: " + e.Message);

        return false;

    }

}

void \_\_fastcall TForm7::ComboBoxProductsChange(TObject \*Sender)

{

    if (ComboBoxProducts->ItemIndex != -1) {

        String selectedProduct = ComboBoxProducts->Text;

        double price = GetProductPrice(selectedProduct);

        TEdit \*EditPrice = dynamic\_cast<TEdit\*>(FindComponent("EditPrice"));

        if (EditPrice) {

            EditPrice->Text = FloatToStrF(price, ffFixed, 15, 2);

        }

    }

}