Приложение А

Министерство образования и науки Нижегородской области

Государственное бюджетное профессиональное

образовательное учреждение

«Заволжский автомоторный техникум»

**Листинг**

Автоматизируемой информационной

системы «Магазин напольных покрытий»

Разработал:

студент группы ИС-21Б

Осипов И.Е.

г. Заволжье

2024– 2025 уч. год

**Листинг программы**

**Authorization.cs** – форма авторизации

using System;

using System.Security.Cryptography;

using System.Text;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace flooring\_shop

{

public partial class Authorization : Form

{

private bool passwordVisible = false;

public Authorization()

{

InitializeComponent();

Pwdtxt.PasswordChar = '•';

EyeBtn.Text = "👁";

}

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

{

string login = Logintxt.Text;

string password = Pwdtxt.Text;

DatabaseConnection dbConnection = new DatabaseConnection();

if (dbConnection.OpenConnection())

{

string query = $"SELECT UserRole, UserSurname, UserName, UserPatronymic FROM User WHERE UserLogin='{login}' AND UserPassword='{password}'";

MySqlDataReader reader = dbConnection.ExecuteQuery(query);

if (reader.HasRows)

{

reader.Read();

int role = reader.GetInt32("UserRole");

string surname = reader["UserSurname"].ToString();

string name = reader["UserName"].ToString();

string patronymic = reader["UserPatronymic"].ToString();

// Форматируем ФИО: фамилия полностью, имя и отчество — инициалы

string userFullName = $"{surname} {name[0]}.{patronymic[0]}.";

switch (role)

{

case 1:

Administrator adminForm = new Administrator();

adminForm.SetAdminInfo(userFullName);

this.Hide();

adminForm.Show();

break;

case 2:

SellerForm sellerForm = new SellerForm();

sellerForm.SetUserInfo(userFullName); // Передаем информацию о пользователе

this.Hide();

sellerForm.Show();

break;

case 3:

CommoditySpecialist commodityForm = new CommoditySpecialist();

commodityForm.SetUserInfomerch(userFullName);

this.Hide();

commodityForm.Show();

break;

default:

MessageBox.Show("Доступ запрещен.");

break;

}

}

else

{

MessageBox.Show("Неверный логин или пароль.");

}

reader.Close();

dbConnection.CloseConnection();

}

else

{

MessageBox.Show("Ошибка подключения к базе данных.");

}

}

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

{

if (!char.IsLetterOrDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

}

}

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

{

if (!char.IsLetterOrDigit(e.KeyChar) && !char.IsControl(e.KeyChar))

{

e.Handled = true;

}

}

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

{

Application.Exit();

}

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

{

passwordVisible = !passwordVisible;

Pwdtxt.PasswordChar = passwordVisible ? '\0' : '•';

EyeBtn.Text = passwordVisible ? "👁" : "👁";

}

private string ComputeSha256Hash(string rawData)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

}

}

**Administrator.cs** – форма администратора

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Net;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace flooring\_shop

{

public partial class Administrator : Form

{

public Administrator()

{

InitializeComponent();

}

public void SetAdminInfo(string userFullName)

{

AdminName.Text = $"Администратор - {userFullName}";

}

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

{

}

private void Administrator\_FormClosing(object sender, FormClosingEventArgs e)

{

Authorization authForm = new Authorization();

authForm.Show();

}

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

{

AdminUsersForm adminuser = new AdminUsersForm();

this.Hide();

adminuser.Show();

}

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

{

AdministratorWord adminword = new AdministratorWord();

this.Hide();

adminword.Show();

}

}

}

**SellerForm.cs** – форма продавца

using System;

using System.Windows.Forms;

namespace flooring\_shop

{

public partial class SellerForm : Form

{

public SellerForm()

{

InitializeComponent();

}

// Метод для установки информации о пользователе

public void SetUserInfo(string userFullName)

{

seller.Text = ("Продавец - " + userFullName); // Устанавливаем текст в Label

}

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

{

HistorySeller orderseller = new HistorySeller();

this.Hide();

orderseller.Show();

}

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

{

SellerOrder orderseller = new SellerOrder();

this.Hide();

orderseller.Show();

}

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

{

}

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

{

Authorization authForm = new Authorization();

authForm.Show();

}

}

}

**CmoditySecialist.cs** – форма товароведа

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace flooring\_shop

{

public partial class CommoditySpecialist : Form

{

private DatabaseConnection dbConnection;

private List<Product> selectedProducts = new List<Product>(); // Список выбранных товаров

private DataTable originalDataTable; // Оригинальные данные из базы

public void SetUserInfomerch(string userFullName)

{

label1.Text = ("Товаровед - " + userFullName); // Устанавливаем текст в Label

}

public CommoditySpecialist()

{

InitializeComponent();

dbConnection = new DatabaseConnection();

InitializeSortComboBox(); // Инициализируем ComboBox для сортировки

InitializeDataGridView(); // Настраиваем DataGridView

LoadCategories(); // Загружаем категории

LoadProducts(); // Загружаем товары при запуске формы

}

// Загрузка категорий в ComboBox

private void LoadCategories()

{

try

{

if (dbConnection.OpenConnection())

{

string query = "SELECT DISTINCT ProdCategoryName FROM ProdCategory";

MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection());

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

// Добавляем пункт "Все категории"

DataRow allCategoriesRow = dataTable.NewRow();

allCategoriesRow["ProdCategoryName"] = "Все категории";

dataTable.Rows.InsertAt(allCategoriesRow, 0);

CmbFilter.DataSource = dataTable;

CmbFilter.DisplayMember = "ProdCategoryName"; // Отображаемое значение

CmbFilter.SelectedIndex = 0; // Выбираем "Все категории" по умолчанию

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке категорий: " + ex.Message);

}

}

// Загрузка товаров из базы данных

public void LoadProducts()

{

try

{

if (dbConnection.OpenConnection())

{

string query = @"

SELECT

ProductArticleNumber AS 'Артикул',

ProductName AS 'Название',

ProductUnit AS 'Единица',

ProductCost AS 'Цена',

ProductQuantityInStock AS 'Остаток',

ProductDescription AS 'Описание',

prodcategory.ProdCategoryName AS 'Категория'

FROM Product INNER JOIN prodcategory ON ProductCategory = ProdCategoryID";

using (MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection()))

{

originalDataTable = new DataTable();

adapter.Fill(originalDataTable);

DGVCMMSpecProd.DataSource = originalDataTable;

}

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке данных: " + ex.Message);

}

}

// Инициализация ComboBox для сортировки

private void InitializeSortComboBox()

{

CmbSort.Items.AddRange(new string[]

{

"Без сортировки",

"Цена по возрастанию",

"Цена по убыванию",

"Название по алфавиту (А-Я)",

"Название по алфавиту (Я-А)"

});

CmbSort.SelectedIndex = 0;

}

// Настройка DataGridView

private void InitializeDataGridView()

{

DGVCMMSpecProd.ReadOnly = true;

DGVCMMSpecProd.AllowUserToDeleteRows = false;

DGVCMMSpecProd.AllowUserToAddRows = false;

DGVCMMSpecProd.AllowUserToResizeRows = false;

DGVCMMSpecProd.AllowUserToResizeColumns = false;

DGVCMMSpecProd.AutoGenerateColumns = true;

}

// Обновление данных в DataGridView с учетом фильтрации, поиска и сортировки

private void UpdateDataGridView()

{

if (originalDataTable == null) return;

// Копируем оригинальные данные

DataTable filteredDataTable = originalDataTable.Copy();

// Применяем фильтр по категории

string selectedCategory = CmbFilter.Text;

if (selectedCategory != "Все категории")

{

// Фильтруем по названию категории

filteredDataTable.DefaultView.RowFilter = $"[Категория] = '{selectedCategory}'";

}

else

{

// Если выбрано "Все категории", сбрасываем фильтр

filteredDataTable.DefaultView.RowFilter = "";

}

// Применяем поиск

string searchText = TxtSearch.Text.Trim().ToLower();

if (!string.IsNullOrEmpty(searchText))

{

// Добавляем условие поиска

string searchFilter = $"[Название] LIKE '%{searchText}%' OR [Артикул] LIKE '%{searchText}%'";

if (string.IsNullOrEmpty(filteredDataTable.DefaultView.RowFilter))

{

filteredDataTable.DefaultView.RowFilter = searchFilter;

}

else

{

filteredDataTable.DefaultView.RowFilter += $" AND ({searchFilter})";

}

}

// Применяем сортировку

string sortBy = CmbSort.Text;

switch (sortBy)

{

case "Цена по возрастанию":

filteredDataTable.DefaultView.Sort = "[Цена] ASC";

break;

case "Цена по убыванию":

filteredDataTable.DefaultView.Sort = "[Цена] DESC";

break;

case "Название по алфавиту (А-Я)":

filteredDataTable.DefaultView.Sort = "[Название] ASC";

break;

case "Название по алфавиту (Я-А)":

filteredDataTable.DefaultView.Sort = "[Название] DESC";

break;

default:

filteredDataTable.DefaultView.Sort = "";

break;

}

// Обновляем DataGridView

DGVCMMSpecProd.DataSource = filteredDataTable.DefaultView.ToTable();

}

// Метод для получения ID категории по её названию

private int GetCategoryIdByName(string categoryName)

{

if (dbConnection.OpenConnection())

{

string query = "SELECT ProdCategoryID FROM ProdCategory WHERE ProdCategoryName = @CategoryName";

using (MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection()))

{

command.Parameters.AddWithValue("@CategoryName", categoryName);

object result = command.ExecuteScalar();

dbConnection.CloseConnection();

if (result != null && int.TryParse(result.ToString(), out int categoryId))

{

return categoryId;

}

}

}

return -1; // Если категория не найдена

}

// Обработчик изменения выбора в ComboBox для фильтрации

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

{

UpdateDataGridView();

}

// Обработчик изменения текста в TextBox для поиска

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

{

UpdateDataGridView();

}

// Обработчик изменения выбора в ComboBox для сортировки

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

{

UpdateDataGridView();

}

// Обработчик нажатия кнопки "Добавить в заказ"

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

{

if (DGVCMMSpecProd.SelectedRows.Count > 0)

{

foreach (DataGridViewRow row in DGVCMMSpecProd.SelectedRows)

{

// Создаем объект Product и заполняем его данными из DataGridView

Product product = new Product

{

Article = row.Cells["Артикул"].Value.ToString(),

Name = row.Cells["Название"].Value.ToString(),

Price = Convert.ToDecimal(row.Cells["Цена"].Value),

QuantityInStock = Convert.ToInt32(row.Cells["Остаток"].Value), // Количество на складе

QuantityOrdered = 1 // По умолчанию 1 шт.

};

// Проверяем, есть ли товар уже в списке

Product existingProduct = selectedProducts.FirstOrDefault(p => p.Article == product.Article);

if (existingProduct != null)

{

// Увеличиваем количество, если товар уже есть в списке

existingProduct.QuantityOrdered += 1;

}

else

{

// Добавляем товар в список, если его там нет

selectedProducts.Add(product);

}

}

// Показываем сообщение о добавлении

MessageBox.Show("Товары добавлены в заказ.", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Выберите товары для добавления в заказ\n(с помощью выделения строки).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

// Обработчик нажатия кнопки "Добваить товар"

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

{

AddProductForm orderForm = new AddProductForm();

orderForm.ShowDialog();

this.Hide();

// Очищаем список после оформления заказа

selectedProducts.Clear();

}

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

{

DeleteProductForm deleteForm = new DeleteProductForm(this);

deleteForm.ShowDialog();

this.Hide();

}

// Обработчик закрытия формы

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

{

Authorization authForm = new Authorization();

authForm.Show();

}

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

{

LoadProducts();

}

private void DGVCMMSpecProd\_CellDoubleClick(object sender, DataGridViewCellEventArgs e)

{

if (e.RowIndex >= 0)

{

DataGridViewRow row = DGVCMMSpecProd.Rows[e.RowIndex];

string article = row.Cells["Артикул"].Value.ToString();

string name = row.Cells["Название"].Value.ToString();

decimal price = Convert.ToDecimal(row.Cells["Цена"].Value);

int quantity = Convert.ToInt32(row.Cells["Остаток"].Value);

string description = row.Cells["Описание"].Value.ToString();

EditProductForm editForm = new EditProductForm(article, name, price, quantity, description);

if (editForm.ShowDialog() == DialogResult.OK)

{

LoadProducts(); // Обновляем данные после редактирования

}

}

}

}

}

**SellerOrder.cs** – форма продавца с товарами

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace flooring\_shop

{

public partial class SellerOrder : Form

{

private DatabaseConnection dbConnection;

private List<Product> selectedProducts = new List<Product>(); // Список выбранных товаров

private DataTable originalDataTable; // Оригинальные данные из базы

public SellerOrder()

{

InitializeComponent();

dbConnection = new DatabaseConnection();

InitializeSortComboBox(); // Инициализируем ComboBox для сортировки

InitializeDataGridView(); // Настраиваем DataGridView

LoadCategories(); // Загружаем категории

LoadProducts(); // Загружаем товары при запуске формы

}

// Загрузка категорий в ComboBox

private void LoadCategories()

{

try

{

if (dbConnection.OpenConnection())

{

string query = "SELECT DISTINCT ProdCategoryName FROM ProdCategory";

MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection());

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

// Добавляем пункт "Все категории"

DataRow allCategoriesRow = dataTable.NewRow();

allCategoriesRow["ProdCategoryName"] = "Все категории";

dataTable.Rows.InsertAt(allCategoriesRow, 0);

CmbFilter.DataSource = dataTable;

CmbFilter.DisplayMember = "ProdCategoryName"; // Отображаемое значение

CmbFilter.SelectedIndex = 0; // Выбираем "Все категории" по умолчанию

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке категорий: " + ex.Message);

}

}

// Загрузка товаров из базы данных

public void LoadProducts()

{

try

{

if (dbConnection.OpenConnection())

{

string query = @"

SELECT

ProductArticleNumber AS 'Артикул',

ProductName AS 'Название',

ProductUnit AS 'Единица',

ProductCost AS 'Цена',

ProductQuantityInStock AS 'Остаток',

ProductDescription AS 'Описание',

prodcategory.ProdCategoryName AS 'Категория'

FROM Product INNER JOIN prodcategory ON ProductCategory = ProdCategoryID";

using (MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection()))

{

originalDataTable = new DataTable();

adapter.Fill(originalDataTable);

DGVProdOrderSeller.DataSource = originalDataTable;

}

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке данных: " + ex.Message);

}

}

// Инициализация ComboBox для сортировки

private void InitializeSortComboBox()

{

CmbSort.Items.AddRange(new string[]

{

"Без сортировки",

"Цена по возрастанию",

"Цена по убыванию",

"Название по алфавиту (А-Я)",

"Название по алфавиту (Я-А)"

});

CmbSort.SelectedIndex = 0;

}

// Настройка DataGridView

private void InitializeDataGridView()

{

DGVProdOrderSeller.ReadOnly = true;

DGVProdOrderSeller.AllowUserToDeleteRows = false;

DGVProdOrderSeller.AllowUserToAddRows = false;

DGVProdOrderSeller.AllowUserToResizeRows = false;

DGVProdOrderSeller.AllowUserToResizeColumns = false;

DGVProdOrderSeller.AutoGenerateColumns = true;

}

// Обновление данных в DataGridView с учетом фильтрации, поиска и сортировки

private void UpdateDataGridView()

{

if (originalDataTable == null) return;

// Копируем оригинальные данные

DataTable filteredDataTable = originalDataTable.Copy();

// Применяем фильтр по категории

string selectedCategory = CmbFilter.Text;

if (selectedCategory != "Все категории")

{

// Фильтруем по названию категории

filteredDataTable.DefaultView.RowFilter = $"[Категория] = '{selectedCategory}'";

}

else

{

// Если выбрано "Все категории", сбрасываем фильтр

filteredDataTable.DefaultView.RowFilter = "";

}

// Применяем поиск

string searchText = TxtSearch.Text.Trim().ToLower();

if (!string.IsNullOrEmpty(searchText))

{

// Добавляем условие поиска

string searchFilter = $"[Название] LIKE '%{searchText}%' OR [Артикул] LIKE '%{searchText}%'";

if (string.IsNullOrEmpty(filteredDataTable.DefaultView.RowFilter))

{

filteredDataTable.DefaultView.RowFilter = searchFilter;

}

else

{

filteredDataTable.DefaultView.RowFilter += $" AND ({searchFilter})";

}

}

// Применяем сортировку

string sortBy = CmbSort.Text;

switch (sortBy)

{

case "Цена по возрастанию":

filteredDataTable.DefaultView.Sort = "[Цена] ASC";

break;

case "Цена по убыванию":

filteredDataTable.DefaultView.Sort = "[Цена] DESC";

break;

case "Название по алфавиту (А-Я)":

filteredDataTable.DefaultView.Sort = "[Название] ASC";

break;

case "Название по алфавиту (Я-А)":

filteredDataTable.DefaultView.Sort = "[Название] DESC";

break;

default:

filteredDataTable.DefaultView.Sort = "";

break;

}

// Обновляем DataGridView

DGVProdOrderSeller.DataSource = filteredDataTable.DefaultView.ToTable();

}

// Метод для получения ID категории по её названию

private int GetCategoryIdByName(string categoryName)

{

if (dbConnection.OpenConnection())

{

string query = "SELECT ProdCategoryID FROM ProdCategory WHERE ProdCategoryName = @CategoryName";

using (MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection()))

{

command.Parameters.AddWithValue("@CategoryName", categoryName);

object result = command.ExecuteScalar();

dbConnection.CloseConnection();

if (result != null && int.TryParse(result.ToString(), out int categoryId))

{

return categoryId;

}

}

}

return -1; // Если категория не найдена

}

// Обработчик изменения выбора в ComboBox для фильтрации

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

{

UpdateDataGridView();

}

// Обработчик изменения текста в TextBox для поиска

private void TxtSearch\_TextChanged(object sender, EventArgs e)

{

UpdateDataGridView();

}

// Обработчик изменения выбора в ComboBox для сортировки

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

{

UpdateDataGridView();

}

// Обработчик нажатия кнопки "Добавить в заказ"

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

{

if (DGVProdOrderSeller.SelectedRows.Count > 0)

{

foreach (DataGridViewRow row in DGVProdOrderSeller.SelectedRows)

{

// Создаем объект Product и заполняем его данными из DataGridView

Product product = new Product

{

Article = row.Cells["Артикул"].Value.ToString(),

Name = row.Cells["Название"].Value.ToString(),

Price = Convert.ToDecimal(row.Cells["Цена"].Value),

QuantityInStock = Convert.ToInt32(row.Cells["Остаток"].Value), // Количество на складе

QuantityOrdered = 1 // По умолчанию 1 шт.

};

// Проверяем, есть ли товар уже в списке

Product existingProduct = selectedProducts.FirstOrDefault(p => p.Article == product.Article);

if (existingProduct != null)

{

// Увеличиваем количество, если товар уже есть в списке

existingProduct.QuantityOrdered += 1;

}

else

{

// Добавляем товар в список, если его там нет

selectedProducts.Add(product);

}

}

// Показываем сообщение о добавлении

MessageBox.Show("Товары добавлены в заказ.", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Выберите товары для добавления в заказ\n(с помощью выделения строки).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

// Обработчик нажатия кнопки "Оформить заказ"

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

{

if (selectedProducts.Count > 0)

{

// Открываем форму SellerOrderGo с выбранными товарами

SellerOrderGo orderForm = new SellerOrderGo(selectedProducts, this);

orderForm.ShowDialog();

}

else

{

MessageBox.Show("Добавьте товары в заказ\n(с помощью выделения строки).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

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

{

HistorySeller orderseller = new HistorySeller();

this.Hide();

orderseller.Show();

}

// Обработчик закрытия формы

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

{

Authorization authForm = new Authorization();

authForm.Show();

}

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

{

LoadProducts();

}

}

}

**SellerOrderGo.cs** – форма оформления товара

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using Word = Microsoft.Office.Interop.Word;

namespace flooring\_shop

{

public partial class SellerOrderGo : Form

{

private List<Product> selectedProducts; // Список выбранных товаров

// Конструктор, принимающий только список товаров

public SellerOrderGo(List<Product> products, SellerOrder parentForm)

{

this.parentForm = parentForm;

InitializeComponent();

selectedProducts = products;

// Инициализация DataGridView (добавление столбцов)

InitializeDataGridView();

LoadProducts(); // Загружаем товары в DataGridView

UpdateTotalPrice(); // Обновляем общую стоимость

}

// Инициализация DataGridView (добавление столбцов)

private void InitializeDataGridView()

{

// Очистка существующих столбцов

CartDataGridView.Columns.Clear();

// Добавление столбцов

CartDataGridView.Columns.Add("ArticleColumn", "Артикул");

CartDataGridView.Columns.Add("NameColumn", "Название");

CartDataGridView.Columns.Add("PriceColumn", "Цена за шт.");

CartDataGridView.Columns.Add("QuantityColumn", "Количество");

CartDataGridView.Columns.Add("TotalColumn", "Сумма");

// Настройка свойств DataGridView

CartDataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

CartDataGridView.AllowUserToAddRows = false;

CartDataGridView.AllowUserToDeleteRows = false;

CartDataGridView.ReadOnly = false; // Разрешаем редактирование ячеек по умолчанию

// Запрещаем редактирование всех столбцов, кроме "Количество"

foreach (DataGridViewColumn column in CartDataGridView.Columns)

{

column.ReadOnly = true; // Все столбцы доступны только для чтения

}

CartDataGridView.Columns["QuantityColumn"].ReadOnly = false; // Разрешаем редактирование только столбца "Количество"

// Обработка изменения значения в столбце "Количество"

CartDataGridView.CellEndEdit += CartDataGridView\_CellEndEdit;

}

// Загрузка товаров в DataGridView

private void LoadProducts()

{

CartDataGridView.Rows.Clear();

foreach (var product in selectedProducts)

{

CartDataGridView.Rows.Add(

product.Article,

product.Name,

product.Price.ToString("C"), // Форматируем как валюту

product.QuantityOrdered,

(product.Price \* product.QuantityOrdered).ToString("C") // Сумма

);

}

}

// Обновление общей стоимости

private void UpdateTotalPrice()

{

decimal totalPrice = selectedProducts.Sum(p => p.Price \* p.QuantityOrdered);

decimal discount = 0;

// Скидка 7% при сумме заказа от 10000 руб

if (totalPrice >= 10000)

{

discount = totalPrice \* 0.07m;

DiscountLabel.Text = $"Скидка 7%: -{discount:C}";

DiscountLabel.Visible = true;

}

else

{

DiscountLabel.Visible = false;

}

TotalPriceLabel.Text = $"Итого: {(totalPrice - discount):C}";

}

// Обработка изменения количества товара

private void CartDataGridView\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 3) // Проверяем, что редактировался столбец "Количество"

{

var row = CartDataGridView.Rows[e.RowIndex];

string article = row.Cells["ArticleColumn"].Value.ToString();

int newQuantity = Convert.ToInt32(row.Cells["QuantityColumn"].Value);

// Находим товар в списке

Product product = selectedProducts.FirstOrDefault(p => p.Article == article);

if (product == null)

{

MessageBox.Show("Товар не найден в списке.");

return;

}

// Проверка на отрицательное количество

if (newQuantity < 0)

{

MessageBox.Show("Количество не может быть отрицательным.");

row.Cells["QuantityColumn"].Value = product.QuantityOrdered; // Возвращаем старое значение

return;

}

// Проверка на превышение остатка на складе

if (newQuantity > product.QuantityInStock)

{

MessageBox.Show("Количество превышает остаток на складе.");

row.Cells["QuantityColumn"].Value = product.QuantityOrdered; // Возвращаем старое значение

return;

}

// Обновляем количество и сумму

product.QuantityOrdered = newQuantity;

row.Cells["TotalColumn"].Value = (product.Price \* newQuantity).ToString("C"); // Обновляем сумму

UpdateTotalPrice(); // Пересчитываем общую стоимость

}

}

// Сохранение заказа в базе данных

private void SaveOrderToDatabase()

{

try

{

DatabaseConnection dbConnection = new DatabaseConnection();

if (dbConnection.OpenConnection())

{

// Создаем запись в таблице Order

string orderQuery = @"

INSERT INTO `Order` (OrderDate, OrderDeliveryDate, UserID, OrderCode)

VALUES (@OrderDate, @OrderDeliveryDate, @UserID, @OrderCode)";

MySqlCommand orderCommand = new MySqlCommand(orderQuery, dbConnection.GetConnection());

orderCommand.Parameters.AddWithValue("@OrderDate", DateTime.Now);

orderCommand.Parameters.AddWithValue("@OrderDeliveryDate", DateTime.Now.AddDays(7)); // Пример: доставка через 7 дней

orderCommand.Parameters.AddWithValue("@UserID", 1); // Замените на ID текущего пользователя

orderCommand.Parameters.AddWithValue("@OrderCode", new Random().Next(1000, 9999)); // Генерация случайного кода заказа

orderCommand.ExecuteNonQuery();

// Получаем ID созданного заказа

long orderId = orderCommand.LastInsertedId;

// Создаем записи в таблице OrderProduct

foreach (var product in selectedProducts)

{

string orderProductQuery = @"

INSERT INTO OrderProduct (OrderID, ProductArticleNumber, ProdNumCount)

VALUES (@OrderID, @ProductArticleNumber, @ProdNumCount)";

MySqlCommand orderProductCommand = new MySqlCommand(orderProductQuery, dbConnection.GetConnection());

orderProductCommand.Parameters.AddWithValue("@OrderID", orderId);

orderProductCommand.Parameters.AddWithValue("@ProductArticleNumber", product.Article);

orderProductCommand.Parameters.AddWithValue("@ProdNumCount", product.QuantityOrdered);

orderProductCommand.ExecuteNonQuery();

// Обновляем остаток на складе

string updateStockQuery = $"UPDATE Product SET ProductQuantityInStock = ProductQuantityInStock - {product.QuantityOrdered} WHERE ProductArticleNumber = '{product.Article}'";

MySqlCommand updateStockCommand = new MySqlCommand(updateStockQuery, dbConnection.GetConnection());

updateStockCommand.ExecuteNonQuery();

}

dbConnection.CloseConnection();

MessageBox.Show("Заказ успешно оформлен!");

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при оформлении заказа: " + ex.Message);

}

}

// Создание чека в Word

private void CreateReceipt()

{

try

{

// Создаем экземпляр Word и новый документ

Word.Application wordApp = new Word.Application();

Word.Document wordDoc = wordApp.Documents.Add();

wordApp.Visible = true;

// 1. Заголовок магазина (центрированный, жирный, 16pt)

Word.Paragraph title = wordDoc.Content.Paragraphs.Add();

title.Range.Text = "Магазин напольных покрытий «ПолыРоскоши»";

title.Range.Font.Bold = 1;

title.Range.Font.Size = 16;

title.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

title.Range.InsertParagraphAfter();

// 2. Подзаголовок "Чек заказа" (центрированный, жирный, 14pt)

Word.Paragraph subtitle = wordDoc.Content.Paragraphs.Add();

subtitle.Range.Text = "КАССОВЫЙ ЧЕК";

subtitle.Range.Font.Bold = 1;

subtitle.Range.Font.Size = 14;

subtitle.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

subtitle.Range.InsertParagraphAfter();

// 3. Информация о заказе (дата/время)

Word.Paragraph orderInfo = wordDoc.Content.Paragraphs.Add();

orderInfo.Range.Text = $"Дата: {DateTime.Now:dd.MM.yyyy HH:mm:ss}\n" +

$"Номер заказа: {new Random().Next(1000, 9999)}";

orderInfo.Range.Font.Size = 12;

orderInfo.Range.InsertParagraphAfter();

// 4. Горизонтальная линия-разделитель

Word.Paragraph divider = wordDoc.Content.Paragraphs.Add();

divider.Range.Text = new string('-', 50);

divider.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

divider.Range.InsertParagraphAfter();

// 5. Таблица с товарами (4 колонки)

Word.Table table = wordDoc.Tables.Add(

Range: wordDoc.Range(wordDoc.Content.End - 1),

NumRows: selectedProducts.Count + 1, // +1 для заголовков

NumColumns: 4);

// Настройка стиля таблицы

table.Borders.Enable = 1; // Включаем все границы

table.Range.Font.Size = 12;

table.AllowAutoFit = true;

// Заголовки таблицы (жирный шрифт)

table.Cell(1, 1).Range.Text = "ТОВАР";

table.Cell(1, 2).Range.Text = "ЦЕНА";

table.Cell(1, 3).Range.Text = "КОЛ-ВО";

table.Cell(1, 4).Range.Text = "СУММА";

for (int i = 1; i <= 4; i++)

{

table.Cell(1, i).Range.Font.Bold = 1;

table.Cell(1, i).Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphCenter;

}

// Заполнение таблицы данными

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

{

var product = selectedProducts[i];

table.Cell(i + 2, 1).Range.Text = product.Name;

table.Cell(i + 2, 2).Range.Text = product.Price.ToString("C");

table.Cell(i + 2, 3).Range.Text = product.QuantityOrdered.ToString();

table.Cell(i + 2, 4).Range.Text = (product.Price \* product.QuantityOrdered).ToString("C");

// Выравнивание для числовых колонок

table.Cell(i + 2, 2).Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphRight;

table.Cell(i + 2, 3).Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphCenter;

table.Cell(i + 2, 4).Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphRight;

}

// 6. Итоговая информация (после таблицы)

decimal totalSum = selectedProducts.Sum(p => p.Price \* p.QuantityOrdered);

decimal discount = totalSum >= 10000 ? totalSum \* 0.07m : 0;

decimal toPay = totalSum - discount;

Word.Paragraph totalInfo = wordDoc.Content.Paragraphs.Add();

totalInfo.Range.Text = $"\nПодитог: {totalSum:C}";

totalInfo.Range.Font.Size = 12;

totalInfo.Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphRight;

totalInfo.Range.InsertParagraphAfter();

if (discount > 0)

{

Word.Paragraph discountInfo = wordDoc.Content.Paragraphs.Add();

discountInfo.Range.Text = $"Скидка 7%: -{discount:C}";

discountInfo.Range.Font.Size = 12;

discountInfo.Range.Font.Color = Word.WdColor.wdColorRed;

discountInfo.Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphRight;

discountInfo.Range.InsertParagraphAfter();

}

Word.Paragraph totalToPay = wordDoc.Content.Paragraphs.Add();

totalToPay.Range.Text = $"ИТОГО К ОПЛАТЕ: {toPay:C}";

totalToPay.Range.Font.Bold = 1;

totalToPay.Range.Font.Size = 14;

totalToPay.Range.Font.Color = Word.WdColor.wdColorDarkBlue;

totalToPay.Range.ParagraphFormat.Alignment =

Word.WdParagraphAlignment.wdAlignParagraphRight;

totalToPay.Range.InsertParagraphAfter();

// 7. Нижний колонтитул (благодарность)

Word.Paragraph footer = wordDoc.Content.Paragraphs.Add();

footer.Range.Text = "\nСпасибо за покупку!\n" +

"Телефон для справок: +7 (123) 456-78-90";

footer.Range.Font.Italic = 1;

footer.Range.Font.Size = 11;

footer.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

// 8. Автосохранение с уникальным именем

string desktopPath = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

string fileName = Path.Combine(desktopPath, $"Чек\_{DateTime.Now:yyyyMMdd\_HHmmss}.docx");

wordDoc.SaveAs(fileName);

MessageBox.Show($"Чек сохранен: {fileName}", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании чека:\n{ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// Кнопка "Оформить заказ"

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

{

DialogResult result = MessageBox.Show("Оформить заказ?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

// Сохраняем заказ в базе данных

SaveOrderToDatabase();

// Спрашиваем, нужно ли распечатать чек

DialogResult printResult = MessageBox.Show("Распечатать чек?", "Чек", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (printResult == DialogResult.Yes)

{

CreateReceipt();

}

this.Close();

}

}

// Кнопка "Вернуться к товарам"

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

{

DialogResult result = MessageBox.Show("Убрать товары с заказа?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

selectedProducts.Clear(); // Очищаем список товаров

this.Close(); // Закрываем форму

}

else

{

this.Hide();

}

}

private SellerOrder parentForm;

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

{

parentForm.LoadProducts();

}

}

}

**HistorySeller.cs** – форма оформления товара

using System;

using System.Collections.Generic;

using System.Data;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

using Word = Microsoft.Office.Interop.Word;

namespace flooring\_shop

{

public partial class HistorySeller : Form

{

private DatabaseConnection dbConnection;

private List<Product> selectedProductsForReturn = new List<Product>(); // Список выбранных товаров для возврата

public HistorySeller()

{

InitializeComponent();

dbConnection = new DatabaseConnection();

InitializeDataGridView(); // Настраиваем DataGridView

LoadOrderHistory(); // Загружаем историю заказов при запуске формы

}

// Настройка DataGridView

private void InitializeDataGridView()

{

DGVHistoryOrder.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

DGVHistoryOrder.AllowUserToAddRows = false;

DGVHistoryOrder.ReadOnly = true; // Запрещаем редактирование

}

// Загрузка истории заказов

private void LoadOrderHistory()

{

try

{

if (dbConnection.OpenConnection())

{

// Запрос для получения истории заказов

string query = @"

SELECT

`Order`.OrderID AS 'Номер заказа',

OrderProduct.ProductArticleNumber AS 'Артикул',

Product.ProductName AS 'Товар',

Product.ProductCost AS 'Цена',

Product.ProductQuantityInStock AS 'Остаток',

OrderProduct.ProdNumCount AS 'Количество',

(Product.ProductCost \* OrderProduct.ProdNumCount) AS 'Сумма'

FROM OrderProduct

JOIN `Order` ON OrderProduct.OrderID = `Order`.OrderID

JOIN Product ON OrderProduct.ProductArticleNumber = Product.ProductArticleNumber";

MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection());

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

DGVHistoryOrder.DataSource = dataTable;

DGVHistoryOrder.Rows[0].Selected = false;

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке истории заказов: " + ex.Message);

}

}

// Метод для возврата товара

public void ReturnProduct(int orderId, string productName, int returnedQuantity)

{

try

{

if (dbConnection.OpenConnection())

{

// Увеличиваем количество товара на складе

string updateStockQuery = $"UPDATE Product SET ProductQuantityInStock = ProductQuantityInStock + {returnedQuantity} WHERE ProductName = '{productName}'";

MySqlCommand updateStockCommand = new MySqlCommand(updateStockQuery, dbConnection.GetConnection());

updateStockCommand.ExecuteNonQuery();

// Получаем текущее количество товара в заказе

string getCurrentQuantityQuery = $"SELECT ProdNumCount FROM OrderProduct WHERE OrderID = {orderId} AND ProductArticleNumber = (SELECT ProductArticleNumber FROM Product WHERE ProductName = '{productName}')";

MySqlCommand getCurrentQuantityCommand = new MySqlCommand(getCurrentQuantityQuery, dbConnection.GetConnection());

int currentQuantity = Convert.ToInt32(getCurrentQuantityCommand.ExecuteScalar());

// Вычисляем новое количество товара в заказе

int newQuantity = currentQuantity - returnedQuantity;

if (newQuantity > 0)

{

// Обновляем количество в заказе

string updateOrderQuery = $"UPDATE OrderProduct SET ProdNumCount = {newQuantity} WHERE OrderID = {orderId} AND ProductArticleNumber = (SELECT ProductArticleNumber FROM Product WHERE ProductName = '{productName}')";

MySqlCommand updateOrderCommand = new MySqlCommand(updateOrderQuery, dbConnection.GetConnection());

updateOrderCommand.ExecuteNonQuery();

}

else

{

// Удаляем товар из заказа, если количество стало нулевым

string deleteOrderQuery = $"DELETE FROM OrderProduct WHERE OrderID = {orderId} AND ProductArticleNumber = (SELECT ProductArticleNumber FROM Product WHERE ProductName = '{productName}')";

MySqlCommand deleteOrderCommand = new MySqlCommand(deleteOrderQuery, dbConnection.GetConnection());

deleteOrderCommand.ExecuteNonQuery();

}

dbConnection.CloseConnection();

MessageBox.Show("Товар возвращен на склад!");

// Обновляем историю заказов

LoadOrderHistory(); // Обновляем интерфейс

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при возврате товара: " + ex.Message);

}

}

// Метод для получения цены товара

private decimal GetProductPrice(string productName)

{

decimal price = 0;

try

{

if (dbConnection.OpenConnection())

{

string query = $"SELECT ProductCost FROM Product WHERE ProductName = '{productName}'";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

object result = command.ExecuteScalar();

if (result != null)

{

price = Convert.ToDecimal(result);

}

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при получении цены товара: " + ex.Message);

}

return price;

}

// Метод для создания чека в формате Word

private void CreateReceipt(int orderId, string productName, int quantity, decimal totalPrice)

{

try

{

// Создаем новый документ Word

Word.Application wordApp = new Word.Application();

Word.Document wordDoc = wordApp.Documents.Add();

wordApp.Visible = true; // Делаем Word видимым

// Добавляем текст в документ

Word.Paragraph paragraph = wordDoc.Content.Paragraphs.Add();

paragraph.Range.Text = $"Чек по заказу №{orderId}\n\n";

paragraph.Range.Text += $"ИНН: {new Random().Next(100000, 999999)}\n"; // Генерация случайного ИНН

paragraph.Range.Text += $"Товар: {productName}\n";

paragraph.Range.Text += $"Количество: {quantity}\n";

paragraph.Range.Text += $"Общая сумма: {totalPrice:C}\n"; // Форматируем как валюту

paragraph.Range.InsertParagraphAfter();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при создании чека: " + ex.Message);

}

}

// Обработчик нажатия кнопки для перехода на форму SellerOrder

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

{

SellerOrder sellerOrder = new SellerOrder();

this.Hide();

sellerOrder.Show();

}

// Обработчик закрытия формы

private void HistorySeller\_FormClosed\_1(object sender, FormClosedEventArgs e)

{

Authorization authorizationForm = new Authorization();

authorizationForm.Show();

}

// Обработчик загрузки формы

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

{

LoadOrderHistory(); // Обновляем историю заказов при каждой загрузке формы

}

// Обработчик нажатия кнопки "Оформить возврат"

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

{

try

{

if (DGVHistoryOrder.SelectedRows.Count > 0)

{

// Список для хранения выбранных товаров и их заказов

List<(int OrderId, Product Product)> selectedProductsForReturn = new List<(int, Product)>();

foreach (DataGridViewRow row in DGVHistoryOrder.SelectedRows)

{

if (row.Cells["Артикул"].Value != null &&

row.Cells["Товар"].Value != null &&

row.Cells["Цена"].Value != null &&

row.Cells["Остаток"].Value != null &&

row.Cells["Количество"].Value != null)

{

int orderId = Convert.ToInt32(row.Cells["Номер заказа"].Value); // Получаем orderId из выбранной строки

string article = row.Cells["Артикул"].Value.ToString();

string name = row.Cells["Товар"].Value.ToString();

decimal price = Convert.ToDecimal(row.Cells["Цена"].Value);

int quantityInStock = Convert.ToInt32(row.Cells["Остаток"].Value);

int quantityOrdered = Convert.ToInt32(row.Cells["Количество"].Value);

Product product = new Product

{

Article = article,

Name = name,

Price = price,

QuantityInStock = quantityInStock,

QuantityOrdered = quantityOrdered

};

// Добавляем товар и его заказ в список

selectedProductsForReturn.Add((orderId, product));

}

}

// Открываем форму BackProduct с выбранными товарами и их заказами

BackProduct backProductForm = new BackProduct(selectedProductsForReturn, this);

backProductForm.ShowDialog();

// Очищаем список после закрытия формы

selectedProductsForReturn.Clear();

}

else

{

MessageBox.Show("Выберите товары для возврата.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch (Exception ex)

{

MessageBox.Show($"Произошла ошибка: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

if (DGVHistoryOrder.SelectedRows.Count > 0)

{

// Получаем данные из выбранной строки

DataGridViewRow selectedRow = DGVHistoryOrder.SelectedRows[0];

int orderId = Convert.ToInt32(selectedRow.Cells["Номер заказа"].Value);

string productName = selectedRow.Cells["Товар"].Value.ToString();

int quantity = Convert.ToInt32(selectedRow.Cells["Количество"].Value);

decimal price = GetProductPrice(productName);

// Создаем чек

CreateReceipt(orderId, productName, quantity, price \* quantity);

}

else

{

MessageBox.Show("Выберите заказ для создания чека.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

}

**EditProductForm.cs** – форма изменения продукта

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;

using MySql.Data.MySqlClient;

namespace flooring\_shop

{

public partial class EditProductForm : Form

{

private DatabaseConnection dbConnection;

private string articleNumber;

public EditProductForm(string article, string name, decimal price, int quantity, string description)

{

InitializeComponent();

dbConnection = new DatabaseConnection();

articleNumber = article;

ArticleTxt.Text = article;

ArticleTxt.Text = name;

PriceTxt.Text = price.ToString();

QuantityTxt.Text = quantity.ToString();

DescriptionTxt.Text = description;

}

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

{

if (!decimal.TryParse(PriceTxt.Text, out decimal price) || price <= 0)

{

MessageBox.Show("Введите корректную цену.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (!int.TryParse(QuantityTxt.Text, out int quantity) || quantity < 0)

{

MessageBox.Show("Введите корректное количество.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (MessageBox.Show("Применить изменения?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

try

{

if (dbConnection.OpenConnection())

{

string query = @"

UPDATE Product

SET ProductCost = @Price,

ProductQuantityInStock = @Quantity,

ProductDescription = @Description

WHERE ProductArticleNumber = @Article";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

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

command.Parameters.AddWithValue("@Quantity", quantity);

command.Parameters.AddWithValue("@Description", DescriptionTxt.Text);

command.Parameters.AddWithValue("@Article", articleNumber);

command.ExecuteNonQuery();

dbConnection.CloseConnection();

MessageBox.Show("Изменения сохранены.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.DialogResult = DialogResult.OK;

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при сохранении изменений: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

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

{

this.Close();

}

private void EditProductForm\_FormClosing(object sender, FormClosingEventArgs e)

{

this.Close();

}

}

}

**DeleteProductForm.cs** – форма удаления продукта

using MySql.Data.MySqlClient;

using System;

using System.Data;

using System.Windows.Forms;

namespace flooring\_shop

{

public partial class DeleteProductForm : Form

{

private DatabaseConnection dbConnection;

private DataTable originalDataTable;

private CommoditySpecialist parentForm;

public DeleteProductForm(CommoditySpecialist parent)

{

InitializeComponent();

this.parentForm = parent;

dbConnection = new DatabaseConnection();

InitializeSortComboBox();

InitializeDataGridView();

LoadCategories();

LoadProducts();

}

private void InitializeSortComboBox()

{

CmbSort.Items.AddRange(new string[]

{

"Без сортировки",

"Цена по возрастанию",

"Цена по убыванию",

"Название по алфавиту (А-Я)",

"Название по алфавиту (Я-А)"

});

CmbSort.SelectedIndex = 0;

}

private void InitializeDataGridView()

{

DGVCMMSpecProd.ReadOnly = true;

DGVCMMSpecProd.AllowUserToDeleteRows = false;

DGVCMMSpecProd.AllowUserToAddRows = false;

DGVCMMSpecProd.AllowUserToResizeRows = false;

DGVCMMSpecProd.AllowUserToResizeColumns = false;

DGVCMMSpecProd.AutoGenerateColumns = true;

}

private void LoadCategories()

{

try

{

if (dbConnection.OpenConnection())

{

string query = "SELECT DISTINCT ProdCategoryName FROM ProdCategory";

MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection());

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

DataRow allCategoriesRow = dataTable.NewRow();

allCategoriesRow["ProdCategoryName"] = "Все категории";

dataTable.Rows.InsertAt(allCategoriesRow, 0);

CmbFilter.DataSource = dataTable;

CmbFilter.DisplayMember = "ProdCategoryName";

CmbFilter.SelectedIndex = 0;

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке категорий: " + ex.Message);

}

}

public void LoadProducts()

{

try

{

if (dbConnection.OpenConnection())

{

string query = @"

SELECT

ProductArticleNumber AS 'Артикул',

ProductName AS 'Название',

ProductCost AS 'Цена',

ProductQuantityInStock AS 'Остаток',

ProductDescription AS 'Описание',

prodcategory.ProdCategoryName AS 'Категория'

FROM Product

INNER JOIN prodcategory ON ProductCategory = ProdCategoryID";

using (MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection()))

{

originalDataTable = new DataTable();

adapter.Fill(originalDataTable);

DGVCMMSpecProd.DataSource = originalDataTable;

}

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке данных: " + ex.Message);

}

}

private void UpdateDataGridView()

{

if (originalDataTable == null) return;

DataTable filteredDataTable = originalDataTable.Copy();

string selectedCategory = CmbFilter.Text;

if (selectedCategory != "Все категории")

{

filteredDataTable.DefaultView.RowFilter = $"[Категория] = '{selectedCategory}'";

}

else

{

filteredDataTable.DefaultView.RowFilter = "";

}

string searchText = TxtSearch.Text.Trim().ToLower();

if (!string.IsNullOrEmpty(searchText))

{

string searchFilter = $"[Название] LIKE '%{searchText}%' OR [Артикул] LIKE '%{searchText}%'";

if (string.IsNullOrEmpty(filteredDataTable.DefaultView.RowFilter))

{

filteredDataTable.DefaultView.RowFilter = searchFilter;

}

else

{

filteredDataTable.DefaultView.RowFilter += $" AND ({searchFilter})";

}

}

string sortBy = CmbSort.Text;

switch (sortBy)

{

case "Цена по возрастанию":

filteredDataTable.DefaultView.Sort = "[Цена] ASC";

break;

case "Цена по убыванию":

filteredDataTable.DefaultView.Sort = "[Цена] DESC";

break;

case "Название по алфавиту (А-Я)":

filteredDataTable.DefaultView.Sort = "[Название] ASC";

break;

case "Название по алфавиту (Я-А)":

filteredDataTable.DefaultView.Sort = "[Название] DESC";

break;

default:

filteredDataTable.DefaultView.Sort = "";

break;

}

DGVCMMSpecProd.DataSource = filteredDataTable.DefaultView.ToTable();

}

private void DeleteProduct(string articleNumber)

{

try

{

if (dbConnection.OpenConnection())

{

string query = $"DELETE FROM Product WHERE ProductArticleNumber = '{articleNumber}'";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

command.ExecuteNonQuery();

dbConnection.CloseConnection();

MessageBox.Show("Товар успешно удален.");

LoadProducts(); // Обновляем данные после удаления

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении товара: " + ex.Message);

}

}

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

{

UpdateDataGridView();

}

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

{

UpdateDataGridView();

}

private void TxtSearch\_TextChanged(object sender, EventArgs e)

{

UpdateDataGridView();

}

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

{

parentForm.LoadProducts();

this.Close();

}

private void DGVCMMSpecProd\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

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

{

if (DGVCMMSpecProd.SelectedRows.Count > 0)

{

string articleNumber = DGVCMMSpecProd.SelectedRows[0].Cells["Артикул"].Value.ToString();

string productName = DGVCMMSpecProd.SelectedRows[0].Cells["Название"].Value.ToString();

DialogResult firstConfirm = MessageBox.Show($"Вы уверены, что хотите удалить товар '{productName}'?", "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (firstConfirm == DialogResult.Yes)

{

DialogResult secondConfirm = MessageBox.Show("Точно удалить этот товар?", "Окончательное подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Warning);

if (secondConfirm == DialogResult.Yes)

{

DeleteProduct(articleNumber);

parentForm.LoadProducts(); // Обновляем данные на родительской форме

}

}

}

else

{

MessageBox.Show("Выберите товар для удаления.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**BackProduct.cs** – форма возвращения продукта

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace flooring\_shop

{

public partial class BackProduct : Form

{

private List<(int OrderId, Product Product)> selectedProducts;

private HistorySeller parentForm;

public BackProduct(List<(int OrderId, Product Product)> products, HistorySeller parentForm)

{

InitializeComponent();

this.selectedProducts = products;

this.parentForm = parentForm;

InitializeDataGridView(); // Настраиваем DataGridView

LoadProducts(); // Загружаем товары в DataGridView

}

// Настройка DataGridView

private void InitializeDataGridView()

{

BackProdDataGridView.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;

BackProdDataGridView.AllowUserToAddRows = false;

BackProdDataGridView.ReadOnly = false; // Разрешаем редактирование

// Добавляем столбцы

BackProdDataGridView.Columns.Add("Номер заказа", "Номер заказа");

BackProdDataGridView.Columns.Add("Артикул", "Артикул");

BackProdDataGridView.Columns.Add("Товар", "Товар");

BackProdDataGridView.Columns.Add("Цена", "Цена");

BackProdDataGridView.Columns.Add("Остаток", "Остаток");

BackProdDataGridView.Columns.Add("Количество", "Количество");

BackProdDataGridView.Columns.Add("Сумма", "Сумма");

// Запрещаем редактирование всех столбцов, кроме "Количество"

foreach (DataGridViewColumn column in BackProdDataGridView.Columns)

{

if (column.HeaderText != "Количество")

{

column.ReadOnly = true;

}

}

// Обработка изменения значения в столбце "Количество"

BackProdDataGridView.CellEndEdit += BackProdDataGridView\_CellEndEdit;

}

// Загрузка товаров в DataGridView

private void LoadProducts()

{

BackProdDataGridView.Rows.Clear();

foreach (var (orderId, product) in selectedProducts)

{

BackProdDataGridView.Rows.Add(

orderId, // Номер заказа

product.Article, // Артикул товара

product.Name, // Название товара

product.Price, // Цена

product.QuantityInStock, // Остаток на складе

product.QuantityOrdered, // Количество в заказе

product.Price \* product.QuantityOrdered // Сумма

);

}

}

// Обработка изменения количества товара

private void BackProdDataGridView\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 5) // Проверяем, что редактировался столбец "Количество"

{

var row = BackProdDataGridView.Rows[e.RowIndex];

string article = row.Cells["Артикул"].Value?.ToString(); // Проверка на null

if (article == null) return;

if (!int.TryParse(row.Cells["Количество"].Value?.ToString(), out int newQuantity))

{

MessageBox.Show("Введите корректное количество.");

return;

}

// Находим товар в списке

var selectedProduct = selectedProducts.FirstOrDefault(p => p.Product.Article == article);

if (selectedProduct.Product == null)

{

MessageBox.Show("Товар не найден в списке.");

return;

}

// Проверка на отрицательное количество

if (newQuantity < 0)

{

MessageBox.Show("Количество не может быть отрицательным.");

row.Cells["Количество"].Value = selectedProduct.Product.QuantityOrdered; // Возвращаем старое значение

return;

}

// Проверка на превышение количества в заказе

if (newQuantity > selectedProduct.Product.QuantityOrdered)

{

MessageBox.Show("Количество не может превышать количество в заказе.");

row.Cells["Количество"].Value = selectedProduct.Product.QuantityOrdered; // Возвращаем старое значение

return;

}

// Обновляем количество и сумму

selectedProduct.Product.QuantityOrdered = newQuantity;

row.Cells["Сумма"].Value = (selectedProduct.Product.Price \* newQuantity).ToString("C"); // Обновляем сумму

}

}

// Обработчик нажатия кнопки "Вернуть товар"

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

{

foreach (var (orderId, product) in selectedProducts)

{

// Возвращаем товар на склад и обновляем количество в заказе

parentForm.ReturnProduct(orderId, product.Name, product.QuantityOrdered);

}

MessageBox.Show("Товары успешно возвращены на склад.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.Close();

}

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

{

DialogResult result = MessageBox.Show("Убрать товары с возврата?", "Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question);

if (result == DialogResult.Yes)

{

selectedProducts.Clear(); // Очищаем список товаров

this.Close(); // Закрываем форму

}

}

}

}

**AdminUsersForm.cs** – форма пользователей

using MySql.Data.MySqlClient;

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 flooring\_shop

{

public partial class AdminUsersForm : Form

{

private DatabaseConnection dbConnection;

public AdminUsersForm()

{

InitializeComponent();

dbConnection = new DatabaseConnection();

LoadUsers();

}

private void LoadUsers()

{

try

{

if (dbConnection.OpenConnection())

{

string query = @"

SELECT

UserID,

UserSurname AS 'Фамилия',

UserName AS 'Имя',

UserPatronymic AS 'Отчество',

UserLogin AS 'Логин',

RoleName AS 'Роль'

FROM User

JOIN Role ON UserRole = RoleID";

MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection());

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

DGVUsers.DataSource = dataTable;

DGVUsers.Columns["UserID"].Visible = false;

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке пользователей: " + ex.Message);

}

}

private void AdminUsersForm\_FormClosing(object sender, FormClosingEventArgs e)

{

Authorization auth = new Authorization();

auth.Show();

}

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

{

if (DGVUsers.SelectedRows.Count == 1)

{

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

AdminEditUserForm editUserForm = new AdminEditUserForm(userId);

if (editUserForm.ShowDialog() == DialogResult.OK)

{

LoadUsers(); // Обновляем список после редактирования

}

}

else if (DGVUsers.SelectedRows.Count > 1)

{

MessageBox.Show("Можно редактировать только одного пользователя одновременно.", "Предупреждение", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

MessageBox.Show("Выберите пользователя для редактирования.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

AdminAddUserForm addUserForm = new AdminAddUserForm();

if (addUserForm.ShowDialog() == DialogResult.OK)

{

LoadUsers(); // Обновляем список после добавления

}

}

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

{

if (DGVUsers.SelectedRows.Count > 0)

{

string message = "Вы точно хотите удалить следующих пользователей:\n";

foreach (DataGridViewRow row in DGVUsers.SelectedRows)

{

message += $"{row.Cells["Фамилия"].Value} {row.Cells["Имя"].Value}\n";

}

if (MessageBox.Show(message, "Подтверждение удаления", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

try

{

if (dbConnection.OpenConnection())

{

foreach (DataGridViewRow row in DGVUsers.SelectedRows)

{

int userId = Convert.ToInt32(row.Cells["UserID"].Value);

string query = $"DELETE FROM User WHERE UserID = {userId}";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

command.ExecuteNonQuery();

}

dbConnection.CloseConnection();

LoadUsers(); // Обновляем список после удаления

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при удалении пользователей: " + ex.Message);

}

}

}

else

{

MessageBox.Show("Выберите пользователей для удаления.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**AdministratorWord.cs** – форма формирования отчета

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;

using MySql.Data.MySqlClient;

using Word = Microsoft.Office.Interop.Word;

namespace flooring\_shop

{

public partial class AdministratorWord : Form

{

private DatabaseConnection dbConnection;

public AdministratorWord()

{

InitializeComponent();

dbConnection = new DatabaseConnection();

LoadPeriods();

}

private void LoadPeriods()

{

PeriodCB.Items.AddRange(new string[] {

"За сегодня",

"За неделю",

"За месяц",

"За квартал",

"За год",

"За все время"

});

PeriodCB.SelectedIndex = 0;

}

private void CreateWordReport(DataTable data, DateTime startDate, DateTime endDate, decimal totalProfit)

{

try

{

Word.Application wordApp = new Word.Application();

Word.Document wordDoc = wordApp.Documents.Add();

wordApp.Visible = true;

// Заголовок

Word.Paragraph titleParagraph = wordDoc.Content.Paragraphs.Add();

titleParagraph.Range.Text = "Отчет о прибыли с продаж";

titleParagraph.Range.Font.Bold = 1;

titleParagraph.Range.Font.Size = 16;

titleParagraph.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

titleParagraph.Range.InsertParagraphAfter();

// Период

Word.Paragraph periodParagraph = wordDoc.Content.Paragraphs.Add();

periodParagraph.Range.Text = $"Период: {startDate.ToShortDateString()} - {endDate.ToShortDateString()}";

periodParagraph.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

periodParagraph.Range.InsertParagraphAfter();

// Дата формирования

Word.Paragraph dateParagraph = wordDoc.Content.Paragraphs.Add();

dateParagraph.Range.Text = $"Дата формирования: {DateTime.Now.ToShortDateString()}";

dateParagraph.Alignment = Word.WdParagraphAlignment.wdAlignParagraphCenter;

dateParagraph.Range.InsertParagraphAfter();

// Пустая строка

Word.Paragraph emptyParagraph = wordDoc.Content.Paragraphs.Add();

emptyParagraph.Range.Text = "";

emptyParagraph.Range.InsertParagraphAfter();

// Таблица с данными

Word.Table table = wordDoc.Tables.Add(wordDoc.Range(wordDoc.Content.End - 1), data.Rows.Count + 1, data.Columns.Count);

table.Borders.Enable = 1;

// Заголовки таблицы

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

{

table.Cell(1, i + 1).Range.Text = data.Columns[i].ColumnName;

table.Cell(1, i + 1).Range.Font.Bold = 1;

}

// Данные таблицы

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

{

for (int j = 0; j < data.Columns.Count; j++)

{

table.Cell(i + 2, j + 1).Range.Text = data.Rows[i][j].ToString();

}

}

// Итоговая прибыль

Word.Paragraph totalParagraph = wordDoc.Content.Paragraphs.Add();

totalParagraph.Range.Text = $"Общая прибыль за период: {totalProfit:C}";

totalParagraph.Range.Font.Bold = 1;

totalParagraph.Range.Font.Size = 14;

totalParagraph.Alignment = Word.WdParagraphAlignment.wdAlignParagraphRight;

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при создании Word-документа: " + ex.Message);

}

}

private void AdministratorWord\_FormClosing(object sender, FormClosingEventArgs e)

{

Authorization authForm = new Authorization();

authForm.Show();

}

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

{

try

{

DateTime startDate, endDate = DateTime.Now;

switch (PeriodCB.SelectedIndex)

{

case 0: // За сегодня

startDate = DateTime.Today;

break;

case 1: // За неделю

startDate = DateTime.Today.AddDays(-7);

break;

case 2: // За месяц

startDate = DateTime.Today.AddMonths(-1);

break;

case 3: // За квартал

startDate = DateTime.Today.AddMonths(-3);

break;

case 4: // За год

startDate = DateTime.Today.AddYears(-1);

break;

default: // За все время

startDate = new DateTime(2000, 1, 1);

break;

}

if (dbConnection.OpenConnection())

{

string query = @"

SELECT

`Order`.OrderID AS 'Номер заказа',

`Order`.OrderDate AS 'Дата заказа',

SUM(Product.ProductCost \* OrderProduct.ProdNumCount) AS 'Сумма заказа'

FROM `Order`

JOIN OrderProduct ON `Order`.OrderID = OrderProduct.OrderID

JOIN Product ON OrderProduct.ProductArticleNumber = Product.ProductArticleNumber

WHERE `Order`.OrderDate BETWEEN @StartDate AND @EndDate

GROUP BY `Order`.OrderID, `Order`.OrderDate";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

command.Parameters.AddWithValue("@StartDate", startDate);

command.Parameters.AddWithValue("@EndDate", endDate);

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

decimal totalProfit = 0;

foreach (DataRow row in dataTable.Rows)

{

totalProfit += Convert.ToDecimal(row["Сумма заказа"]);

}

CreateWordReport(dataTable, startDate, endDate, totalProfit);

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при формировании отчета: " + ex.Message);

}

}

}

}

**AdminEditUserForm.cs** – форма изменения пользователя

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace flooring\_shop

{

public partial class AdminEditUserForm : Form

{

private DatabaseConnection dbConnection;

private int userId;

private bool passwordVisible = false;

public AdminEditUserForm(int userId)

{

InitializeComponent();

this.userId = userId;

dbConnection = new DatabaseConnection();

LoadRoles();

LoadUserData();

NewPasswordTxt.PasswordChar = '•';

EyeBtn.Text = "👁";

}

private void LoadRoles()

{

try

{

if (dbConnection.OpenConnection())

{

string query = "SELECT RoleID, RoleName FROM Role";

MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection());

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

RoleComboBox.DataSource = dataTable;

RoleComboBox.DisplayMember = "RoleName";

RoleComboBox.ValueMember = "RoleID";

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке ролей: " + ex.Message);

}

}

private void LoadUserData()

{

try

{

if (dbConnection.OpenConnection())

{

string query = $"SELECT UserSurname, UserName, UserPatronymic, UserLogin, UserRole FROM User WHERE UserID = {userId}";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

MySqlDataReader reader = command.ExecuteReader();

if (reader.Read())

{

SurnameTxt.Text = reader["UserSurname"].ToString();

NameTxt.Text = reader["UserName"].ToString();

PatronymicTxt.Text = reader["UserPatronymic"].ToString();

LoginTxt.Text = reader["UserLogin"].ToString();

RoleComboBox.SelectedValue = reader["UserRole"];

}

reader.Close();

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке данных пользователя: " + ex.Message);

}

}

private string ComputeSha256Hash(string rawData)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

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

{

passwordVisible = !passwordVisible;

NewPasswordTxt.PasswordChar = passwordVisible ? '\0' : '•';

EyeBtn.Text = passwordVisible ? "👁" : "👁";

}

private void AdminEditUserForm\_FormClosing\_1(object sender, FormClosingEventArgs e)

{

this.Close();

}

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

{

if (string.IsNullOrWhiteSpace(SurnameTxt.Text) ||

string.IsNullOrWhiteSpace(NameTxt.Text) ||

string.IsNullOrWhiteSpace(PatronymicTxt.Text) ||

string.IsNullOrWhiteSpace(LoginTxt.Text))

{

MessageBox.Show("Заполните все обязательные поля.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

if (dbConnection.OpenConnection())

{

// Проверка уникальности логина (кроме текущего пользователя)

string checkQuery = $"SELECT COUNT(\*) FROM User WHERE UserLogin = '{LoginTxt.Text}' AND UserID != {userId}";

MySqlCommand checkCommand = new MySqlCommand(checkQuery, dbConnection.GetConnection());

int count = Convert.ToInt32(checkCommand.ExecuteScalar());

if (count > 0)

{

MessageBox.Show("Пользователь с таким логином уже существует.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Обновление данных пользователя

string updateQuery = @"

UPDATE User

SET UserSurname = @Surname,

UserName = @Name,

UserPatronymic = @Patronymic,

UserLogin = @Login,

UserRole = @Role

WHERE UserID = @UserID";

// Если введен новый пароль, добавляем его к запросу

if (!string.IsNullOrWhiteSpace(NewPasswordTxt.Text))

{

string hashedPassword = ComputeSha256Hash(NewPasswordTxt.Text);

updateQuery = updateQuery.Replace("UserRole = @Role", "UserRole = @Role, UserPassword = @Password");

}

MySqlCommand updateCommand = new MySqlCommand(updateQuery, dbConnection.GetConnection());

updateCommand.Parameters.AddWithValue("@Surname", SurnameTxt.Text);

updateCommand.Parameters.AddWithValue("@Name", NameTxt.Text);

updateCommand.Parameters.AddWithValue("@Patronymic", PatronymicTxt.Text);

updateCommand.Parameters.AddWithValue("@Login", LoginTxt.Text);

updateCommand.Parameters.AddWithValue("@Role", RoleComboBox.SelectedValue);

updateCommand.Parameters.AddWithValue("@UserID", userId);

if (!string.IsNullOrWhiteSpace(NewPasswordTxt.Text))

{

string hashedPassword = ComputeSha256Hash(NewPasswordTxt.Text);

updateCommand.Parameters.AddWithValue("@Password", hashedPassword);

}

updateCommand.ExecuteNonQuery();

dbConnection.CloseConnection();

MessageBox.Show("Данные пользователя успешно обновлены.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.DialogResult = DialogResult.OK;

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при обновлении данных пользователя: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**AdminAddUserForm.cs** – форма добавления пользователя

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using MySql.Data.MySqlClient;

namespace flooring\_shop

{

public partial class AdminAddUserForm : Form

{

private DatabaseConnection dbConnection;

private bool passwordVisible = false;

public AdminAddUserForm()

{

InitializeComponent();

dbConnection = new DatabaseConnection();

LoadRoles();

PasswordTxt.PasswordChar = '•';

EyeBtn.Text = "👁";

}

private void LoadRoles()

{

try

{

if (dbConnection.OpenConnection())

{

string query = "SELECT RoleID, RoleName FROM Role";

MySqlDataAdapter adapter = new MySqlDataAdapter(query, dbConnection.GetConnection());

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

RoleComboBox.DataSource = dataTable;

RoleComboBox.DisplayMember = "RoleName";

RoleComboBox.ValueMember = "RoleID";

dbConnection.CloseConnection();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при загрузке ролей: " + ex.Message);

}

}

private string ComputeSha256Hash(string rawData)

{

using (SHA256 sha256Hash = SHA256.Create())

{

byte[] bytes = sha256Hash.ComputeHash(Encoding.UTF8.GetBytes(rawData));

StringBuilder builder = new StringBuilder();

for (int i = 0; i < bytes.Length; i++)

{

builder.Append(bytes[i].ToString("x2"));

}

return builder.ToString();

}

}

private void AdminAddUserForm\_FormClosing(object sender, FormClosingEventArgs e)

{

this.Close();

}

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

{

passwordVisible = !passwordVisible;

PasswordTxt.PasswordChar = passwordVisible ? '\0' : '•';

EyeBtn.Text = passwordVisible ? "👁" : "👁";

}

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

{

if (string.IsNullOrWhiteSpace(SurnameTxt.Text) ||

string.IsNullOrWhiteSpace(NameTxt.Text) ||

string.IsNullOrWhiteSpace(PatronymicTxt.Text) ||

string.IsNullOrWhiteSpace(LoginTxt.Text) ||

string.IsNullOrWhiteSpace(PasswordTxt.Text))

{

MessageBox.Show("Заполните все поля.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

if (dbConnection.OpenConnection())

{

// Проверка уникальности логина

string checkQuery = $"SELECT COUNT(\*) FROM User WHERE UserLogin = '{LoginTxt.Text}'";

MySqlCommand checkCommand = new MySqlCommand(checkQuery, dbConnection.GetConnection());

int count = Convert.ToInt32(checkCommand.ExecuteScalar());

if (count > 0)

{

MessageBox.Show("Пользователь с таким логином уже существует.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Хеширование пароля

string hashedPassword = ComputeSha256Hash(PasswordTxt.Text);

// Добавление пользователя

string insertQuery = @"

INSERT INTO User (UserSurname, UserName, UserPatronymic, UserLogin, UserPassword, UserRole)

VALUES (@Surname, @Name, @Patronymic, @Login, @Password, @Role)";

MySqlCommand insertCommand = new MySqlCommand(insertQuery, dbConnection.GetConnection());

insertCommand.Parameters.AddWithValue("@Surname", SurnameTxt.Text);

insertCommand.Parameters.AddWithValue("@Name", NameTxt.Text);

insertCommand.Parameters.AddWithValue("@Patronymic", PatronymicTxt.Text);

insertCommand.Parameters.AddWithValue("@Login", LoginTxt.Text);

insertCommand.Parameters.AddWithValue("@Password", hashedPassword);

insertCommand.Parameters.AddWithValue("@Role", RoleComboBox.SelectedValue);

insertCommand.ExecuteNonQuery();

dbConnection.CloseConnection();

MessageBox.Show("Пользователь успешно добавлен.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.DialogResult = DialogResult.OK;

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при добавлении пользователя: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**AddproductForm.cs** – форма добавления продукта

using MySql.Data.MySqlClient;

using System;

using System.Text;

using System.Windows.Forms;

namespace flooring\_shop

{

public partial class AddProductForm : Form

{

private DatabaseConnection dbConnection;

public AddProductForm()

{

InitializeComponent();

dbConnection = new DatabaseConnection();

LoadUnits();

GenerateArticleNumber();

}

private void LoadUnits()

{

cmbUnit.Items.AddRange(new string[] { "шт", "уп", "рул" });

cmbUnit.SelectedIndex = 0;

}

private void GenerateArticleNumber()

{

Random random = new Random();

string letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";

string numbers = "0123456789";

StringBuilder article = new StringBuilder();

article.Append(letters[random.Next(letters.Length)]);

article.Append(letters[random.Next(letters.Length)]);

article.Append('-');

for (int i = 0; i < 4; i++)

{

article.Append(numbers[random.Next(numbers.Length)]);

}

// Проверяем уникальность артикула

while (IsArticleNumberExists(article.ToString()))

{

article.Clear();

article.Append(letters[random.Next(letters.Length)]);

article.Append(letters[random.Next(letters.Length)]);

article.Append('-');

for (int i = 0; i < 4; i++)

{

article.Append(numbers[random.Next(numbers.Length)]);

}

}

txtArticleNumber.Text = article.ToString();

}

private bool IsArticleNumberValid(string articleNumber)

{

return System.Text.RegularExpressions.Regex.IsMatch(articleNumber, @"^[A-Z]{2}-\d{4}$");

}

private bool IsArticleNumberExists(string articleNumber)

{

try

{

if (dbConnection.OpenConnection())

{

string query = $"SELECT COUNT(\*) FROM Product WHERE ProductArticleNumber = '{articleNumber}'";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

int count = Convert.ToInt32(command.ExecuteScalar());

dbConnection.CloseConnection();

return count > 0;

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при проверке артикула: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

return false;

}

private void txtArticleNumber\_TextChanged(object sender, EventArgs e)

{

string article = txtArticleNumber.Text;

if (string.IsNullOrEmpty(article))

{

errorProvider1.Clear();

return;

}

// Проверка формата артикула

if (!System.Text.RegularExpressions.Regex.IsMatch(article, @"^[A-Z]{2}-\d{4}$"))

{

errorProvider1.SetError(txtArticleNumber, "Формат: XX-9999 (X - заглавная буква, 9 - цифра)");

return;

}

// Проверка уникальности артикула

if (IsArticleNumberExists(article))

{

errorProvider1.SetError(txtArticleNumber, "Артикул уже существует");

}

else

{

errorProvider1.Clear();

}

}

private void AddProductForm\_FormClosing\_1(object sender, FormClosingEventArgs e)

{

this.Close();

}

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

{

GenerateArticleNumber();

}

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

{

// Проверка валидности артикула

if (!IsArticleNumberValid(txtArticleNumber.Text))

{

MessageBox.Show("Артикул должен состоять из двух заглавных букв, тире и четырех цифр (например, BL-3245).", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Проверка уникальности артикула

if (IsArticleNumberExists(txtArticleNumber.Text))

{

MessageBox.Show("Товар с таким артикулом уже существует.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Проверка остальных полей

if (string.IsNullOrWhiteSpace(txtProductName.Text) ||

!decimal.TryParse(txtPrice.Text, out decimal price) || price <= 0 ||

!int.TryParse(txtQuantity.Text, out int quantity) || quantity < 0)

{

MessageBox.Show("Проверьте правильность введенных данных.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

if (dbConnection.OpenConnection())

{

string query = @"

INSERT INTO Product

(ProductArticleNumber, ProductName, ProductUnit, ProductCost, ProductQuantityInStock, ProductDescription, ProductCategory, ProductManufacturer)

VALUES (@Article, @Name, @Unit, @Price, @Quantity, @Description, 1, 1)";

MySqlCommand command = new MySqlCommand(query, dbConnection.GetConnection());

command.Parameters.AddWithValue("@Article", txtArticleNumber.Text);

command.Parameters.AddWithValue("@Name", txtProductName.Text);

command.Parameters.AddWithValue("@Unit", cmbUnit.SelectedItem.ToString());

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

command.Parameters.AddWithValue("@Quantity", quantity);

command.Parameters.AddWithValue("@Description", txtDescription.Text);

command.ExecuteNonQuery();

dbConnection.CloseConnection();

MessageBox.Show("Товар успешно добавлен.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

this.DialogResult = DialogResult.OK;

this.Close();

}

}

catch (Exception ex)

{

MessageBox.Show("Ошибка при добавлении товара: " + ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

**DatabaseConnection.cs** – класс подключения к базе данных

using System;

using System.Collections.Generic;

using System.Data.SqlClient;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

namespace flooring\_shop

{

public class DatabaseConnection

{

private MySqlConnection connection;

public DatabaseConnection()

{

string connectionString = "server=localhost;user=root;password=root;database=flooring\_shop;";

connection = new MySqlConnection(connectionString);

}

public MySqlConnection GetConnection()

{

return connection;

}

public bool OpenConnection()

{

if (connection.State == System.Data.ConnectionState.Open)

{

return true; // Соединение уже открыто

}

try

{

connection.Open();

return true;

}

catch (MySqlException ex)

{

Console.WriteLine(ex.Message);

return false;

}

}

public bool CloseConnection()

{

if (connection.State == System.Data.ConnectionState.Closed)

{

return true; // Соединение уже закрыто

}

try

{

connection.Close();

return true;

}

catch (MySqlException ex)

{

Console.WriteLine(ex.Message);

return false;

}

}

public MySqlDataReader ExecuteQuery(string query)

{

MySqlCommand command = new MySqlCommand(query, connection);

return command.ExecuteReader();

}

}

}

**Product.cs** – класс продукта

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace flooring\_shop

{

public class Product

{

public string Article { get; set; }

public string Name { get; set; }

public decimal Price { get; set; }

public int QuantityInStock { get; set; }

public int QuantityOrdered{ get; set; }

public string Description { get; set; }

public string Category { get; set; }

}

}

**Program.cs** – класс программы

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace flooring\_shop

{

static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Authorization());

}

}

}