**Лабораторная работа №7**

**Тема: Отладка отдельных модулей программного проекта. Организация обработки исключений.**

**Выполнил:** Кузнецов Богдан ПР 23/2

**Цель:** создать программный проект в среде Visual Studio на языке C#, реализовать организацию обработки исключений для обеспечения надежной работы приложения, а также разработать функционал для оформления предзаказов с использованием базы данных MS SQL Server.

Ход работы

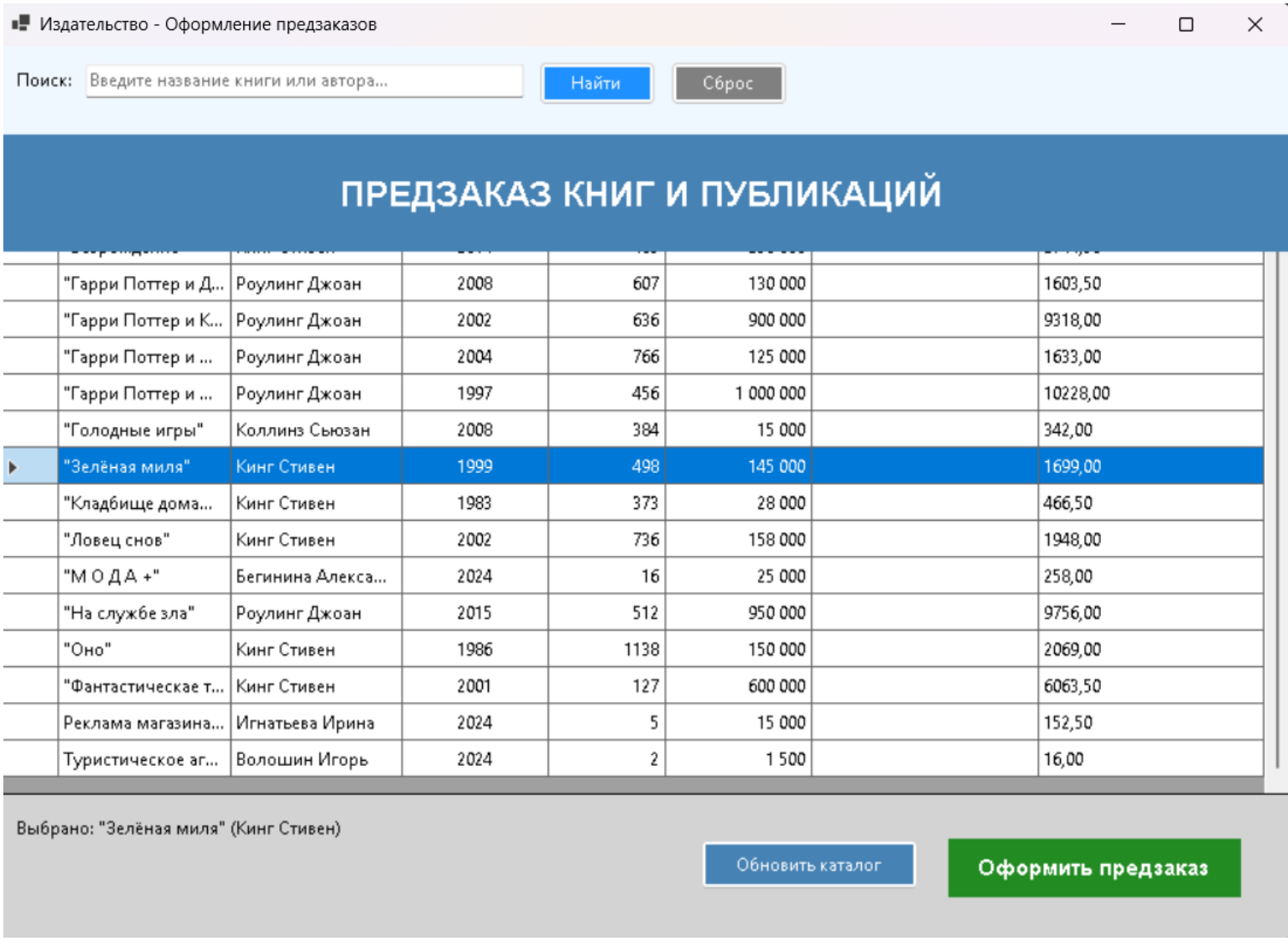


Рисунок 1 — Выбор книги

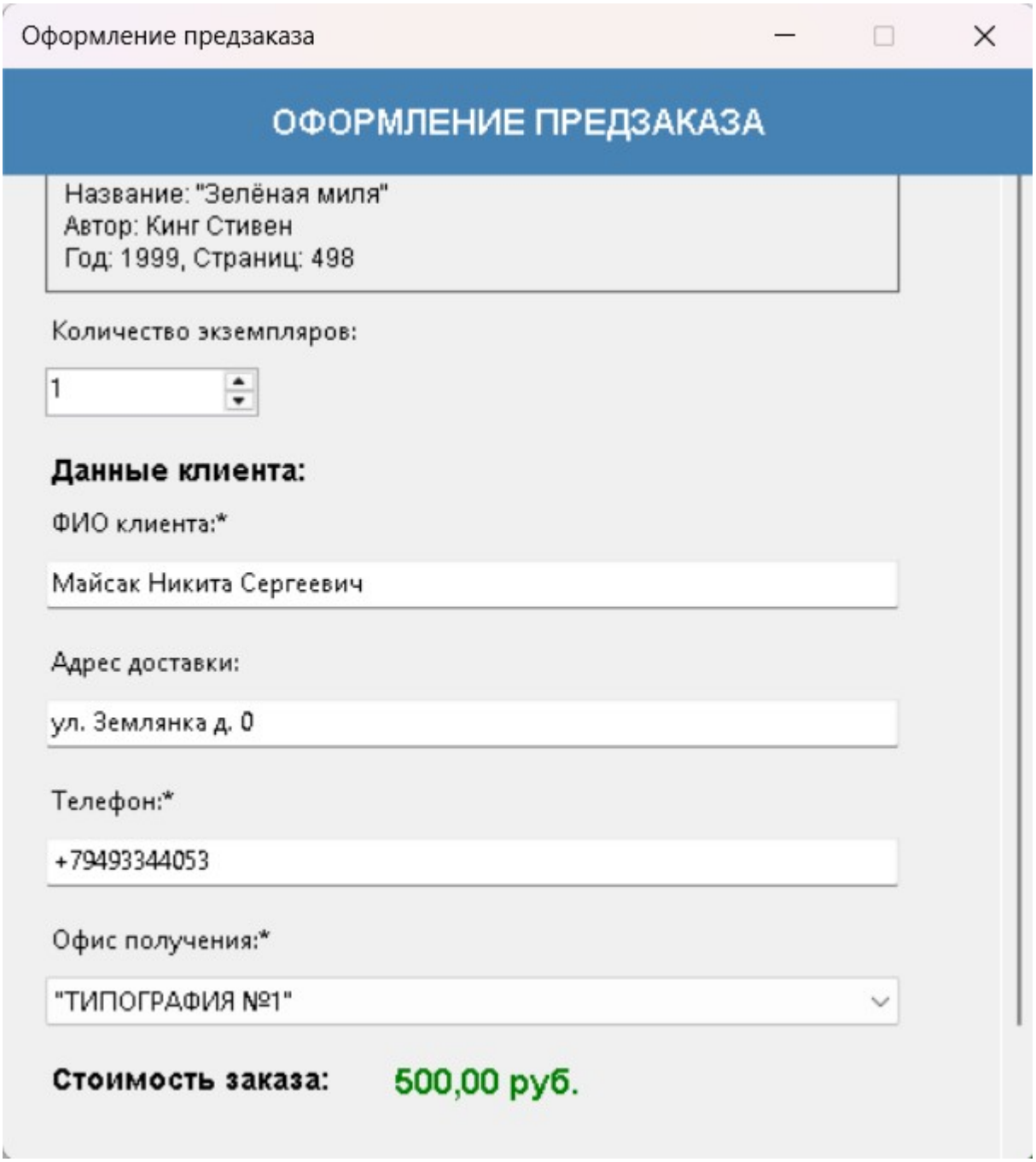


Рисунок 2 — Ввод данных клиента

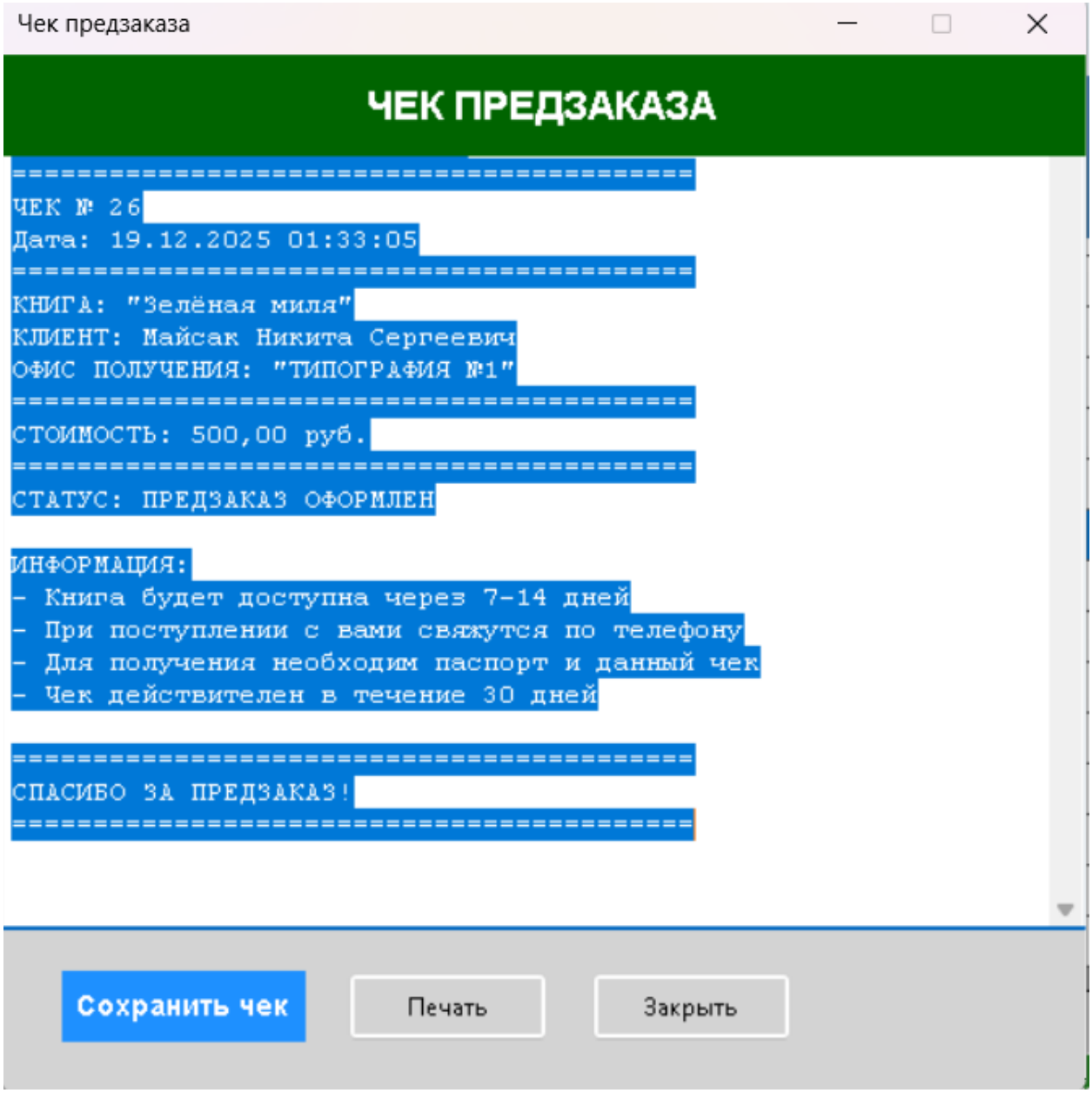


Рисунок 3 — Формирование чека

Листинг кода:

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.IO;

namespace PublishingApp
{
    public partial class FormReceipt : Form
    {
        private DatabaseHelper dbHelper;
```

```
private int orderId;
private TextBox txtReceipt;
private Models.Order currentOrder;

public FormReceipt(DatabaseHelper dbHelper, int orderId)
{
    this.dbHelper = dbHelper;
    this.orderId = orderId;
    InitializeComponent();
    LoadReceipt();
}

private void InitializeComponent()
{
    this.Text = "Чек предзаказа";
    this.Size = new Size(550, 550);
    this.StartPosition = FormStartPosition.CenterParent;
    this.FormBorderStyle = FormBorderStyle.FixedDialog;
    this.MaximizeBox = false;

    // Панель заголовка
    var panelHeader = new Panel();
    panelHeader.Dock = DockStyle.Top;
    panelHeader.Height = 50;
    panelHeader.BackColor = Color.DarkGreen;

    var lblHeader = new Label();
    lblHeader.Text = "ЧЕК ПРЕДЗАКАЗА";
    lblHeader.Font = new Font("Arial", 14, FontStyle.Bold);
    lblHeader.ForeColor = Color.White;
    lblHeader.Dock = DockStyle.Fill;
    lblHeader.TextAlign = ContentAlignment.MiddleCenter;
    panelHeader.Controls.Add(lblHeader);

    // Текстовое поле для чека
    txtReceipt = new TextBox();
    txtReceipt.Dock = DockStyle.Fill;
```

```
txtReceipt.Multiline = true;
txtReceipt.ReadOnly = true;
txtReceipt.Font = new Font("Courier New", 10);
txtReceipt.ScrollBars = ScrollBars.Vertical;
txtReceipt.Margin = new Padding(10);
txtReceipt.BackColor = Color.White;

// Панель для кнопок
var panelBottom = new Panel();
panelBottom.Dock = DockStyle.Bottom;
panelBottom.Height = 80;
panelBottom.BackColor = Color.LightGray;

// Кнопка сохранения
var btnSave = new Button();
btnSave.Text = "Сохранить чек";
btnSave.Size = new Size(120, 35);
btnSave.BackColor = Color.DodgerBlue;
btnSave.ForeColor = Color.White;
btnSave.FlatStyle = FlatStyle.Flat;
btnSave.FlatAppearance.BorderSize = 0;
btnSave.Font = new Font("Arial", 10, FontStyle.Bold);
btnSave.Location = new Point(30, 20);
btnSave.Click += BtnSave_Click;

// Кнопка печати
var btnPrint = new Button();
btnPrint.Text = "Печать";
btnPrint.Size = new Size(100, 35);
btnPrint.Location = new Point(170, 20);
btnPrint.Click += BtnPrint_Click;

// Кнопка закрытия
var btnClose = new Button();
btnClose.Text = "Закрыть";
btnClose.Size = new Size(100, 35);
btnClose.Location = new Point(290, 20);
```

```
    btnClose.Click += (s, e) => this.Close();

    // Добавляем кнопки на панель
    panelBottom.Controls.Add(btnSave);
    panelBottom.Controls.Add(btnPrint);
    panelBottom.Controls.Add(btnClose);

    // Добавление на форму
    this.Controls.Add(panelHeader);
    this.Controls.Add(txtReceipt);
    this.Controls.Add(panelBottom);
}

private void LoadReceipt()
{
    try
    {
        currentOrder = dbHelper.GetOrderDetails(orderId);
        if (currentOrder != null)
        {
            string receiptText = GenerateReceipt(currentOrder);
            txtReceipt.Text = receiptText;
        }
        else
        {
            txtReceipt.Text = "Ошибка: заказ не найден";
        }
    }
    catch (Exception ex)
    {
        txtReceipt.Text = $"Ошибка загрузки чека: {ex.Message}";
    }
}

private string GenerateReceipt(Models.Order order)
{
    return $@"========================================
```

ИЗДАТЕЛЬСТВО

'Предзаказы книг'

==========

ЧЕК № {order.Id}

Дата: {order.OrderDate:dd.MM.yyyy HH:mm:ss}

==========

КНИГА: {order.BookTitle}

КЛИЕНТ: {order.CustomerName}

ОФИС ПОЛУЧЕНИЯ: {order.OfficeName}

==========

СТОИМОСТЬ: {order.Price:F2} руб.

==========

СТАТУС: ПРЕДЗАКАЗ ОФОРМЛЕН

ИНФОРМАЦИЯ:

- Книга будет доступна через 7-14 дней

- При поступлении с вами свяжутся по телефону

- Для получения необходим паспорт и данный чек

- Чек действителен в течение 30 дней

==========

СПАСИБО ЗА ПРЕДЗАКАЗ!

==========";

}

```
private void BtnSave_Click(object sender, EventArgs e)
{
   try
   {
      SaveFileDialog saveFileDialog = new SaveFileDialog();
      saveFileDialog.Filter = "Текстовые файлы (*.txt)|*.txt|Все файлы (*.*)|*.*";
      saveFileDialog.FileName = $"Чек_предзаказа_№{orderId}_{DateTime.Now:yyyyMMdd_HHmmss}.txt";
      saveFileDialog.DefaultExt = "txt";
      saveFileDialog.AddExtension = true;
      saveFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
      saveFileDialog.Title = "Сохранить чек предзаказа";
```

```
            if (saveFileDialog.ShowDialog() == DialogResult.OK)
            {
                string filePath = saveFileDialog.FileName;
                File.WriteAllText(filePath, txtReceipt.Text, System.Text.Encoding.UTF8);

                MessageBox.Show($"Чек успешно сохранен!\n\nПуть: {filePath}",
                    "Сохранение завершено",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка сохранения файла: {ex.Message}",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void BtnPrint_Click(object sender, EventArgs e)
    {
        try
        {
            PrintDialog printDialog = new PrintDialog();
            if (printDialog.ShowDialog() == DialogResult.OK)
            {
                MessageBox.Show("Чек отправлен на печать", "Печать",
                    MessageBoxButtons.OK, MessageBoxIcon.Information);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка печати: {ex.Message}", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }
  }
}
```

```
using Microsoft.Data.SqlClient;
using System;
using System.Drawing;
using System.Windows.Forms;
using System.Linq;

namespace PublishingApp
{
    public partial class MainForm : Form
    {
        private DatabaseHelper dbHelper;
        private DataGridView dataGridViewBooks;
        private TextBox txtSearch;
        private Button btnSearch;
        private Button btnReset;
        private Button btnOrder;
        private Label lblSelectedInfo;

        public MainForm()
        {
            InitializeComponent();
            InitializeDatabase();
            LoadBooks();
        }

        private void InitializeComponent()
        {
            this.Text = "Издательство - Оформление предзаказов";
            this.Size = new Size(900, 650);
            this.StartPosition = FormStartPosition.CenterScreen;

            // Панель заголовка
            var panelHeader = new Panel();
            panelHeader.Dock = DockStyle.Top;
            panelHeader.Height = 80;
            panelHeader.BackColor = Color.SteelBlue;
```

```
var lblHeader = new Label();
lblHeader.Text = "ПРЕДЗАКАЗ КНИГ И ПУБЛИКАЦИЙ";
lblHeader.Font = new Font("Arial", 18, FontStyle.Bold);
lblHeader.ForeColor = Color.White;
lblHeader.Dock = DockStyle.Fill;
lblHeader.TextAlign = ContentAlignment.MiddleCenter;
panelHeader.Controls.Add(lblHeader);

// Панель поиска
var panelSearch = new Panel();
panelSearch.Dock = DockStyle.Top;
panelSearch.Height = 60;
panelSearch.BackColor = Color.AliceBlue;
panelSearch.Padding = new Padding(10);

var lblSearch = new Label();
lblSearch.Text = "Поиск:";
lblSearch.Location = new Point(10, 15);
lblSearch.AutoSize = true;

txtSearch = new TextBox();
txtSearch.Location = new Point(60, 12);
txtSearch.Size = new Size(300, 25);
txtSearch.PlaceholderText = "Введите название книги или автора...";
txtSearch.KeyPress += TxtSearch_KeyPress;

btnSearch = new Button();
btnSearch.Text = "Найти";
btnSearch.Location = new Point(370, 10);
btnSearch.Size = new Size(80, 30);
btnSearch.BackColor = Color.DodgerBlue;
btnSearch.ForeColor = Color.White;
btnSearch.Click += BtnSearch_Click;

btnReset = new Button();
btnReset.Text = "Сброс";
```

```
btnReset.Location = new Point(460, 10);
btnReset.Size = new Size(80, 30);
btnReset.BackColor = Color.Gray;
btnReset.ForeColor = Color.White;
btnReset.Click += BtnReset_Click;

panelSearch.Controls.AddRange(new Control[] { lblSearch, txtSearch, btnSearch, btnReset });

// DataGridView для книг
dataGridViewBooks = new DataGridView();
dataGridViewBooks.Dock = DockStyle.Fill;
dataGridViewBooks.ReadOnly = true;
dataGridViewBooks.AutoSizeColumnsMode = DataGridViewAutoSizeColumnsMode.Fill;
dataGridViewBooks.SelectionMode = DataGridViewSelectionMode.FullRowSelect;
dataGridViewBooks.MultiSelect = false;
dataGridViewBooks.AllowUserToAddRows = false;
dataGridViewBooks.SelectionChanged += DataGridViewBooks_SelectionChanged;
// Важно: подписываемся на событие создания колонок
dataGridViewBooks.DataBindingComplete += DataGridViewBooks_DataBindingComplete;

// Панель информации и кнопок
var panelBottom = new Panel();
panelBottom.Dock = DockStyle.Bottom;
panelBottom.Height = 100;
panelBottom.BackColor = Color.LightGray;
panelBottom.Padding = new Padding(10);

lblSelectedInfo = new Label();
lblSelectedInfo.Name = "lblSelectedInfo";
lblSelectedInfo.Text = "Выберите книгу для оформления предзаказа";
lblSelectedInfo.Location = new Point(10, 15);
lblSelectedInfo.AutoSize = true;

btnOrder = new Button();
btnOrder.Text = "Оформить предзаказ";
btnOrder.Size = new Size(200, 40);
btnOrder.Font = new Font("Arial", 11, FontStyle.Bold);
```

```
            btnOrder.BackColor = Color.ForestGreen;
            btnOrder.ForeColor = Color.White;
            btnOrder.FlatStyle = FlatStyle.Flat;
            btnOrder.FlatAppearance.BorderSize = 0;
            btnOrder.Location = new Point(650, 30);
            btnOrder.Click += BtnOrder_Click;
            btnOrder.Enabled = false;

            // Кнопка обновления
            var btnRefresh = new Button();
            btnRefresh.Text = "Обновить каталог";
            btnRefresh.Size = new Size(150, 35);
            btnRefresh.Location = new Point(480, 30);
            btnRefresh.BackColor = Color.SteelBlue;
            btnRefresh.ForeColor = Color.White;
            btnRefresh.Click += (s, e) => LoadBooks();

            panelBottom.Controls.AddRange(new Control[] { lblSelectedInfo, btnOrder, btnRefresh });

            // Добавление элементов на форму
            this.Controls.Add(panelHeader);
            this.Controls.Add(panelSearch);
            this.Controls.Add(dataGridViewBooks);
            this.Controls.Add(panelBottom);
        }

        private void InitializeDatabase()
        {
            try
            {
                dbHelper = new DatabaseHelper();
                if (!dbHelper.TestConnection())
                {
                    MessageBox.Show("Подключение к базе данных не установлено.\nПриложение будет работать в ограниченном режиме.",
                        "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }
```

```
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка инициализации базы данных: {ex.Message}",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void LoadBooks()
    {
        try
        {
            var books = dbHelper.GetBooks();
            dataGridViewBooks.DataSource = books;

            // Не форматируем здесь! Форматирование будет в DataBindingComplete
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка загрузки книг: {ex.Message}",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    // Событие вызывается ПОСЛЕ того как все колонки созданы
    private void DataGridViewBooks_DataBindingComplete(object sender, DataGridViewBindingCompleteEventArgs
e)
    {
        FormatDataGridView();
    }

    private void FormatDataGridView()
    {
        // Проверяем, что у нас есть колонки
        if (dataGridViewBooks.Columns == null || dataGridViewBooks.Columns.Count == 0)
            return;
```

```
        try
        {
            // Скрываем технические колонки
            if (dataGridViewBooks.Columns.Contains("Id"))
                dataGridViewBooks.Columns["Id"].Visible = false;

            if (dataGridViewBooks.Columns.Contains("AuthorId"))
                dataGridViewBooks.Columns["AuthorId"].Visible = false;

            // Устанавливаем заголовки и форматирование
            SetColumnHeader("Title", "Название");
            SetColumnHeader("AuthorName", "Автор");
            SetColumnHeader("ReleaseYear", "Год издания");
            SetColumnHeader("Pages", "Страниц");
            SetColumnHeader("Circulation", "Тираж");

            // Специальное форматирование для ReleaseYear
            if (dataGridViewBooks.Columns.Contains("ReleaseYear"))
            {
                dataGridViewBooks.Columns["ReleaseYear"].Width = 100;
                dataGridViewBooks.Columns["ReleaseYear"].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleCenter;
            }

            // Специальное форматирование для Pages
            if (dataGridViewBooks.Columns.Contains("Pages"))
            {
                dataGridViewBooks.Columns["Pages"].Width = 80;
                dataGridViewBooks.Columns["Pages"].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
            }

            // Специальное форматирование для Circulation
            if (dataGridViewBooks.Columns.Contains("Circulation"))
            {
                dataGridViewBooks.Columns["Circulation"].Width = 100;
                dataGridViewBooks.Columns["Circulation"].DefaultCellStyle.Alignment =
DataGridViewContentAlignment.MiddleRight;
```

```
            dataGridViewBooks.Columns["Circulation"].DefaultCellStyle.Format = "N0";
        }
    }
    catch (Exception ex)
    {
        // Игнорируем ошибки форматирования
        Console.WriteLine($"Ошибка форматирования DataGridView: {ex.Message}");
    }
}

private void SetColumnHeader(string columnName, string headerText)
{
    if (dataGridViewBooks.Columns.Contains(columnName))
    {
        dataGridViewBooks.Columns[columnName].HeaderText = headerText;
    }
}

private void DataGridViewBooks_SelectionChanged(object sender, EventArgs e)
{
    if (dataGridViewBooks.SelectedRows.Count > 0 && dataGridViewBooks.SelectedRows[0].DataBoundItem is
Models.Book book)
    {
        lblSelectedInfo.Text = $"Выбрано: {book.Title} ({book.AuthorName})";
        btnOrder.Enabled = true;
    }
    else
    {
        lblSelectedInfo.Text = "Выберите книгу для оформления предзаказа";
        btnOrder.Enabled = false;
    }
}

private void TxtSearch_KeyPress(object sender, KeyPressEventArgs e)
{
    if (e.KeyChar == (char)Keys.Enter)
    {
```

```
        SearchBooks();
        e.Handled = true;
    }
}

private void BtnSearch_Click(object sender, EventArgs e)
{
    SearchBooks();
}

private void BtnReset_Click(object sender, EventArgs e)
{
    txtSearch.Text = "";
    LoadBooks();
}

private void SearchBooks()
{
    if (string.IsNullOrWhiteSpace(txtSearch.Text))
    {
        LoadBooks();
        return;
    }

    try
    {
        var allBooks = dbHelper.GetBooks();
        var searchText = txtSearch.Text.ToLower();

        var filteredBooks = allBooks.Where(b =>
            (b.Title != null && b.Title.ToLower().Contains(searchText)) ||
            (b.AuthorName != null && b.AuthorName.ToLower().Contains(searchText)) ||
            b.ReleaseYear.ToString().Contains(searchText)
        ).ToList();

        dataGridViewBooks.DataSource = filteredBooks;
        // Форматирование произойдет автоматически в DataBindingComplete
```

```
            MessageBox.Show($"Найдено книг: {filteredBooks.Count}", "Результаты поиска",
                MessageBoxButtons.OK, MessageBoxIcon.Information);
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка поиска: {ex.Message}", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    private void BtnOrder_Click(object sender, EventArgs e)
    {
        if (dataGridViewBooks.SelectedRows.Count == 0)
        {
            MessageBox.Show("Выберите книгу для предзаказа", "Внимание",
                MessageBoxButtons.OK, MessageBoxIcon.Warning);
            return;
        }

        try
        {
            if (dataGridViewBooks.SelectedRows[0].DataBoundItem is Models.Book selectedBook)
            {
                using (var formOrder = new FormOrder(dbHelper, selectedBook))
                {
                    if (formOrder.ShowDialog() == DialogResult.OK)
                    {
                        MessageBox.Show("Предзаказ успешно оформлен!\n\nЧек сохранен.",
                            "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);
                    }
                }
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка: {ex.Message}", "Ошибка",
```

```
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}


using System;

namespace PublishingApp.Models
{
    public class Author
    {
        public int Id { get; set; }
        public string Surname { get; set; }
        public string Name { get; set; }
        public string Address { get; set; }
        public string FullName => $"{Surname} {Name}";
    }

    public class Book
    {
        public int Id { get; set; }
        public string Title { get; set; }
        public int AuthorId { get; set; }
        public string AuthorName { get; set; }
        public int ReleaseYear { get; set; }
        public int Pages { get; set; }
        public int Circulation { get; set; }
        public string Type { get; set; }
        public decimal Price => CalculatePrice();

        private decimal CalculatePrice()
        {
            decimal basePrice = Pages * 0.5m + Circulation * 0.01m;
            return Math.Round(basePrice, 2);
        }
```

```
    public override string ToString()
    {
        return $"{Title} ({AuthorName}, {ReleaseYear}) - {Pages} стр., тираж: {Circulation}";
    }
}

public class Customer
{
    public int Id { get; set; }
    public string Name { get; set; }
    public int Type { get; set; } = 1;
    public string Address { get; set; }
    public string Phone { get; set; }

    public bool IsValid()
    {
        return !string.IsNullOrWhiteSpace(Name) &&
               !string.IsNullOrWhiteSpace(Phone);
    }
}



public class Office
{
    public int Id { get; set; }
    public string Name { get; set; }
    public string Address { get; set; }
    public string Phone { get; set; }

    public override string ToString()
    {
        return $"{Name} ({Address})";
    }
}

public class Order
```

```
    {
        public int Id { get; set; }
        public string OrderName { get; set; }
        public int BookId { get; set; }
        public string BookTitle { get; set; }
        public int CustomerId { get; set; }
        public string CustomerName { get; set; }
        public int OfficeId { get; set; }
        public string OfficeName { get; set; }
        public DateTime OrderDate { get; set; } = DateTime.Now;
        public DateTime? CompletionDate { get; set; }
        public decimal Price { get; set; }

        public static string GenerateOrderName()
        {
            return $"Предзаказ-{DateTime.Now:yyyyMMdd-HHmmss}";
        }
    }
}
```

```
using System;
using System.Windows.Forms;
using System.Drawing;
using System.Linq;

namespace PublishingApp
{
    public partial class FormOrder : Form
    {
        private DatabaseHelper dbHelper;
        private Models.Book selectedBook;
        private ComboBox comboBoxOffices;
        private TextBox txtCustomerName, txtCustomerAddress, txtCustomerPhone;
        private NumericUpDown numQuantity;
        private Label lblTotalPrice;
```

```
private Label lblBookInfo;
private Button btnConfirm;
private Button btnCancel;

public FormOrder(DatabaseHelper dbHelper, Models.Book book)
{
    this.dbHelper = dbHelper;
    this.selectedBook = book;
    InitializeComponent();
    LoadOffices();
    CalculatePrice();
}

private void InitializeComponent()
{
    this.Text = "Оформление предзаказа";
    this.Size = new Size(500, 550);
    this.StartPosition = FormStartPosition.CenterParent;
    this.FormBorderStyle = FormBorderStyle.FixedDialog;
    this.MaximizeBox = false;

    // Панель заголовка
    var panelHeader = new Panel();
    panelHeader.Dock = DockStyle.Top;
    panelHeader.Height = 50;
    panelHeader.BackColor = Color.SteelBlue;

    var lblHeader = new Label();
    lblHeader.Text = "ОФОРМЛЕНИЕ ПРЕДЗАКАЗА";
    lblHeader.Font = new Font("Arial", 12, FontStyle.Bold);
    lblHeader.ForeColor = Color.White;
    lblHeader.Dock = DockStyle.Fill;
    lblHeader.TextAlign = ContentAlignment.MiddleCenter;
    panelHeader.Controls.Add(lblHeader);

    // Основной контейнер
    var panelMain = new Panel();
```

```
panelMain.Dock = DockStyle.Fill;
panelMain.Padding = new Padding(20);
panelMain.AutoScroll = true;

int yPos = 20;

// Информация о книге
var lblBookTitle = new Label();
lblBookTitle.Text = "Выбранная книга:";
lblBookTitle.AutoSize = true;
lblBookTitle.Font = new Font("Arial", 10, FontStyle.Bold);
lblBookTitle.Location = new Point(20, yPos);
yPos += 25;

lblBookInfo = new Label();
lblBookInfo.AutoSize = false;
lblBookInfo.Size = new Size(400, 60);
lblBookInfo.Location = new Point(20, yPos);
lblBookInfo.Font = new Font("Arial", 9);
lblBookInfo.BorderStyle = BorderStyle.FixedSingle;
lblBookInfo.Padding = new Padding(5);
yPos += 70;

// Количество
var lblQuantity = new Label();
lblQuantity.Text = "Количество экземпляров:";
lblQuantity.AutoSize = true;
lblQuantity.Location = new Point(20, yPos);
yPos += 25;

numQuantity = new NumericUpDown();
numQuantity.Location = new Point(20, yPos);
numQuantity.Size = new Size(100, 25);
numQuantity.Minimum = 1;
numQuantity.Maximum = 10;
numQuantity.Value = 1;
numQuantity.ValueChanged += (s, e) => CalculatePrice();
```

```
yPos += 40;

// Поля ввода данных клиента
var lblCustomerTitle = new Label();
lblCustomerTitle.Text = "Данные клиента:";
lblCustomerTitle.AutoSize = true;
lblCustomerTitle.Font = new Font("Arial", 10, FontStyle.Bold);
lblCustomerTitle.Location = new Point(20, yPos);
yPos += 25;

var lblCustomerName = new Label();
lblCustomerName.Text = "ФИО клиента:*";
lblCustomerName.AutoSize = true;
lblCustomerName.Location = new Point(20, yPos);
yPos += 25;

txtCustomerName = new TextBox();
txtCustomerName.Size = new Size(400, 25);
txtCustomerName.Location = new Point(20, yPos);
txtCustomerName.MaxLength = 100;
yPos += 40;

var lblCustomerAddress = new Label();
lblCustomerAddress.Text = "Адрес доставки:";
lblCustomerAddress.AutoSize = true;
lblCustomerAddress.Location = new Point(20, yPos);
yPos += 25;

txtCustomerAddress = new TextBox();
txtCustomerAddress.Size = new Size(400, 25);
txtCustomerAddress.Location = new Point(20, yPos);
txtCustomerAddress.MaxLength = 200;
yPos += 40;

var lblCustomerPhone = new Label();
lblCustomerPhone.Text = "Телефон:*";
lblCustomerPhone.AutoSize = true;
```

```
lblCustomerPhone.Location = new Point(20, yPos);
yPos += 25;

txtCustomerPhone = new TextBox();
txtCustomerPhone.Size = new Size(400, 25);
txtCustomerPhone.Location = new Point(20, yPos);
txtCustomerPhone.MaxLength = 20;
yPos += 40;

// Выбор офиса
var lblOffice = new Label();
lblOffice.Text = "Офис получения:*";
lblOffice.AutoSize = true;
lblOffice.Location = new Point(20, yPos);
yPos += 25;

comboBoxOffices = new ComboBox();
comboBoxOffices.Size = new Size(400, 25);
comboBoxOffices.Location = new Point(20, yPos);
comboBoxOffices.DropDownStyle = ComboBoxStyle.DropDownList;
yPos += 40;

// Стоимость
var lblPrice = new Label();
lblPrice.Text = "Стоимость заказа:";
lblPrice.AutoSize = true;
lblPrice.Font = new Font("Arial", 10, FontStyle.Bold);
lblPrice.Location = new Point(20, yPos);
yPos += 25;

lblTotalPrice = new Label();
lblTotalPrice.AutoSize = true;
lblTotalPrice.Font = new Font("Arial", 12, FontStyle.Bold);
lblTotalPrice.ForeColor = Color.Green;
lblTotalPrice.Location = new Point(180, yPos - 25);
yPos += 40;
```

```
// Кнопки
btnCancel = new Button();
btnCancel.Text = "Отмена";
btnCancel.Size = new Size(120, 35);
btnCancel.Location = new Point(250, yPos);
btnCancel.Click += (s, e) => this.DialogResult = DialogResult.Cancel;

btnConfirm = new Button();
btnConfirm.Text = "Оформить заказ";
btnConfirm.Size = new Size(150, 35);
btnConfirm.Location = new Point(80, yPos);
btnConfirm.BackColor = Color.ForestGreen;
btnConfirm.ForeColor = Color.White;
btnConfirm.Font = new Font("Arial", 10, FontStyle.Bold);
btnConfirm.Click += BtnConfirm_Click;

// Добавление элементов на панель
panelMain.Controls.Add(lblBookTitle);
panelMain.Controls.Add(lblBookInfo);
panelMain.Controls.Add(lblQuantity);
panelMain.Controls.Add(numQuantity);
panelMain.Controls.Add(lblCustomerTitle);
panelMain.Controls.Add(lblCustomerName);
panelMain.Controls.Add(txtCustomerName);
panelMain.Controls.Add(lblCustomerAddress);
panelMain.Controls.Add(txtCustomerAddress);
panelMain.Controls.Add(lblCustomerPhone);
panelMain.Controls.Add(txtCustomerPhone);
panelMain.Controls.Add(lblOffice);
panelMain.Controls.Add(comboBoxOffices);
panelMain.Controls.Add(lblPrice);
panelMain.Controls.Add(lblTotalPrice);
panelMain.Controls.Add(btnCancel);
panelMain.Controls.Add(btnConfirm);

// Добавление на форму
this.Controls.Add(panelHeader);
```

```
    this.Controls.Add(panelMain);
}

private void LoadOffices()
{
    try
    {
        var offices = dbHelper.GetOffices();
        comboBoxOffices.DataSource = offices;
        comboBoxOffices.DisplayMember = "Name";
        comboBoxOffices.ValueMember = "Id";

        if (offices.Count > 0)
            comboBoxOffices.SelectedIndex = 0;
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка загрузки офисов: {ex.Message}",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private void UpdateBookInfo()
{
    lblBookInfo.Text = $"Название: {selectedBook.Title}\n" +
                       $"Автор: {selectedBook.AuthorName}\n" +
                       $"Год: {selectedBook.ReleaseYear}, Страниц: {selectedBook.Pages}\n" +
                       $"Тираж: {selectedBook.Circulation:N0} экз.";
}

private void CalculatePrice()
{
    // Базовая цена (например, 500 руб за книгу)
    decimal basePrice = 500m;
    // Учет тиража (чем меньше тираж, тем дороже)
    decimal circulationFactor = selectedBook.Circulation > 10000 ? 1.0m : 1.5m;
    // Учет количества страниц
```

```
    decimal pageFactor = selectedBook.Pages > 500 ? 1.2m : 1.0m;

    int quantity = (int)numQuantity.Value;
    decimal totalPrice = basePrice * circulationFactor * pageFactor * quantity;

    lblTotalPrice.Text = $"{totalPrice:F2} руб.";
}

private void BtnConfirm_Click(object sender, EventArgs e)
{
    try
    {
        // Валидация данных
        if (string.IsNullOrWhiteSpace(txtCustomerName.Text))
            throw new Exception("Введите ФИО клиента");

        if (string.IsNullOrWhiteSpace(txtCustomerPhone.Text))
            throw new Exception("Введите телефон клиента");

        if (comboBoxOffices.SelectedItem == null)
            throw new Exception("Выберите офис получения");

        // Создание клиента
        var customer = new Models.Customer
        {
            Name = txtCustomerName.Text,
            Address = txtCustomerAddress.Text,
            Phone = txtCustomerPhone.Text
        };

        int customerId = dbHelper.CreateCustomer(customer);

        // Создание заказа
        int quantity = (int)numQuantity.Value;
        decimal totalPrice = decimal.Parse(lblTotalPrice.Text.Replace(" руб.", ""));

        var order = new Models.Order
```

```
            {
                OrderName = $"Предзаказ: {selectedBook.Title} ({quantity} шт.)",
                BookId = selectedBook.Id,
                CustomerId = customerId,
                OfficeId = (int)comboBoxOffices.SelectedValue,
                OrderDate = DateTime.Now,
                Price = totalPrice
            };

            int orderId = dbHelper.CreateOrder(order);

            // Показать чек
            using (var formReceipt = new FormReceipt(dbHelper, orderId))
            {
                formReceipt.ShowDialog();
            }

            this.DialogResult = DialogResult.OK;
            this.Close();
        }
        catch (Exception ex)
        {
            MessageBox.Show($"Ошибка оформления заказа: {ex.Message}",
                "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        }
    }

    protected override void OnLoad(EventArgs e)
    {
        base.OnLoad(e);
        UpdateBookInfo();
        CalculatePrice();
    }
  }
}
```

```
using System;
using System.Windows.Forms;

namespace PublishingApp
{
    static class Program
    {
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

```
using System;
using Microsoft.Data.SqlClient;
using System.Collections.Generic;
using PublishingApp.Models;
using System.Data;
using System.Windows.Forms;

namespace PublishingApp
{
    public class DatabaseHelper : IDisposable
    {
        // ОСНОВНАЯ СТРОКА ПОДКЛЮЧЕНИЯ:
        private string connectionString = @"Data Source=rezonchic_;Initial Catalog=publishing;Integrated
Security=True;TrustServerCertificate=True;Connect Timeout=30";

        public DatabaseHelper()
        {
            // Простая инициализация
        }
```

```
public DatabaseHelper(string customConnectionString)
{
    connectionString = customConnectionString;
}

public List<Book> GetBooks()
{
    var books = new List<Book>();
    try
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();

            string query = @"
                SELECT p.id_Publication, p.Name, p.Author,
                    a.Surname + ' ' + a.Name as AuthorName,
                    p.ReleaseYear, p.VolumeOfSheets, p.Circulation
                FROM Publications p
                LEFT JOIN Authors a ON p.Author = a.id_Author
                ORDER BY p.Name";

            using (var command = new SqlCommand(query, connection))
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    books.Add(new Book
                    {
                        Id = (int)reader["id_Publication"],
                        Title = reader["Name"].ToString(),
                        AuthorId = reader["Author"] != DBNull.Value ? (int)reader["Author"] : 0,
                        AuthorName = reader["AuthorName"].ToString(),
                        ReleaseYear = (int)reader["ReleaseYear"],
                        Pages = (int)reader["VolumeOfSheets"],
                        Circulation = (int)reader["Circulation"]
```

```
                });
            }
        }
    }
    catch (Exception ex)
    {
        throw new Exception($"Ошибка при загрузке книг: {ex.Message}");
    }
    return books;
}

public List<Office> GetOffices()
{
    var offices = new List<Office>();
    try
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string query = "SELECT id_Office, Office, Address, Phone FROM Offices ORDER BY Office";

            using (var command = new SqlCommand(query, connection))
            using (var reader = command.ExecuteReader())
            {
                while (reader.Read())
                {
                    offices.Add(new Office
                    {
                        Id = (int)reader["id_Office"],
                        Name = reader["Office"].ToString(),
                        Address = reader["Address"].ToString(),
                        Phone = reader["Phone"].ToString()
                    });
                }
            }
        }
```

```
        }
        catch (Exception ex)
        {
            throw new Exception($"Ошибка при загрузке офисов: {ex.Message}");
        }
        return offices;
    }

    public int CreateOrder(Order order)
    {
        try
        {
            using (var connection = new SqlConnection(connectionString))
            {
                connection.Open();
                string query = @"
                    INSERT INTO Orders (Name, Type, Publication, Office, Customer,
                                DateOfAdmission, DateOfCompletion, Price)
                    VALUES (@Name, 1, @Publication, @Office, @Customer,
                            @DateOfAdmission, @DateOfCompletion, @Price);
                    SELECT SCOPE_IDENTITY();";

                using (var command = new SqlCommand(query, connection))
                {
                    command.Parameters.AddWithValue("@Name", order.OrderName ?? $"Заказ от
{DateTime.Now:dd.MM.yyyy}");
                    command.Parameters.AddWithValue("@Publication", order.BookId);
                    command.Parameters.AddWithValue("@Office", order.OfficeId);
                    command.Parameters.AddWithValue("@Customer", order.CustomerId);
                    command.Parameters.AddWithValue("@DateOfAdmission", order.OrderDate);

                    if (order.CompletionDate.HasValue)
                        command.Parameters.AddWithValue("@DateOfCompletion", order.CompletionDate.Value);
                    else
                        command.Parameters.AddWithValue("@DateOfCompletion", DBNull.Value);

                    command.Parameters.AddWithValue("@Price", order.Price);
```

```
            return Convert.ToInt32(command.ExecuteScalar());
        }
    }
  }
  catch (Exception ex)
  {
    throw new Exception($"Ошибка при создании заказа: {ex.Message}");
  }
}

public int CreateCustomer(Customer customer)
{
  try
  {
    using (var connection = new SqlConnection(connectionString))
    {
      connection.Open();
      string query = @"
        INSERT INTO Customers (Name, Type, Address, Phone)
        VALUES (@Name, 1, @Address, @Phone);
        SELECT SCOPE_IDENTITY();";

      using (var command = new SqlCommand(query, connection))
      {
        command.Parameters.AddWithValue("@Name", customer.Name);
        command.Parameters.AddWithValue("@Address",
          string.IsNullOrEmpty(customer.Address) ? (object)DBNull.Value : customer.Address);
        command.Parameters.AddWithValue("@Phone",
          string.IsNullOrEmpty(customer.Phone) ? (object)DBNull.Value : customer.Phone);

        return Convert.ToInt32(command.ExecuteScalar());
      }
    }
  }
  catch (Exception ex)
  {
```

```
        throw new Exception($"Ошибка при создании клиента: {ex.Message}");
    }
}

public Order GetOrderDetails(int orderId)
{
    try
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();
            string query = @"
                SELECT o.id_Order, o.Name as OrderName, o.DateOfAdmission,
                    o.Price, p.Name as BookTitle, c.Name as CustomerName,
                    ofc.Office as OfficeName
                FROM Orders o
                LEFT JOIN Publications p ON o.Publication = p.id_Publication
                LEFT JOIN Customers c ON o.Customer = c.id_Customer
                LEFT JOIN Offices ofc ON o.Office = ofc.id_Office
                WHERE o.id_Order = @OrderId";

            using (var command = new SqlCommand(query, connection))
            {
                command.Parameters.AddWithValue("@OrderId", orderId);

                using (var reader = command.ExecuteReader())
                {
                    if (reader.Read())
                    {
                        return new Order
                        {
                            Id = (int)reader["id_Order"],
                            OrderName = reader["OrderName"].ToString(),
                            BookTitle = reader["BookTitle"].ToString(),
                            CustomerName = reader["CustomerName"].ToString(),
                            OfficeName = reader["OfficeName"].ToString(),
                            OrderDate = (DateTime)reader["DateOfAdmission"],
```

```
                        Price = (decimal)reader["Price"]
                    };
                }
            }
        }
    }
    catch (Exception ex)
    {
        throw new Exception($"Ошибка при получении деталей заказа: {ex.Message}");
    }
    return null;
}

public bool TestConnection()
{
    try
    {
        using (var connection = new SqlConnection(connectionString))
        {
            connection.Open();
            return connection.State == System.Data.ConnectionState.Open;
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка подключения: {ex.Message}",
            "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
        return false;
    }
}
public void Dispose()
{
    // Ресурсы для освобождения при необходимости
}
}
}
```