Код PartnersListForm:

```
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 ПДЭ
{
    public partial class PartnersListForm : Form
    {
        private List<Partner> _partners;

        public PartnersListForm()
        {
            InitializeComponent();
            LoadPartners();
        }

        private void LoadPartners()
        {
            try
            {
                _partners = DatabaseService.GetPartners();
                dataGridViewPartners.AutoGenerateColumns = true;
                dataGridViewPartners.DataSource = _partners;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки партнёров: {ex.Message}",
                    "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void btnAdd_Click(object sender, EventArgs e)
        {
            var editForm = new PartnerEditForm();
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadPartners(); // Обновить список
            }
        }

        private void btnEdit_Click(object sender, EventArgs e)
        {
            if (dataGridViewPartners.SelectedRows.Count == 0)
            {
                MessageBox.Show("Выберите партнёра для редактирования.", "Внимание",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            var partner = (Partner)dataGridViewPartners.SelectedRows[0].DataBoundItem;
            var editForm = new PartnerEditForm(partner);
            if (editForm.ShowDialog() == DialogResult.OK)
            {
                LoadPartners();
            }
        }

        private void btnHistory_Click(object sender, EventArgs e)
```

```
        {
            if (dataGridViewPartners.SelectedRows.Count == 0)
            {
                MessageBox.Show("Выберите партнёра для просмотра истории.", "Внимание",
                    MessageBoxButtons.OK, MessageBoxIcon.Warning);
                return;
            }

            var partner = (Partner)dataGridViewPartners.SelectedRows[0].DataBoundItem;
            var historyForm = new ServiceHistoryForm(partner.Id, partner.Name);
            historyForm.ShowDialog();
        }

        private void dataGridViewPartners_CellDoubleClick(object sender, DataGridViewCellEventArgs e)
        {
            if (e.RowIndex >= 0)
                btnEdit_Click(sender, e);
        }
    }
}
```

Код PartnerEditForm:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Imaging;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace ПДЭ
{
    public partial class PartnerEditForm : Form
    {
        private Partner _partner;
        private bool _isEditMode;

        public PartnerEditForm()
        {
            InitializeComponent();
            _isEditMode = false;
            Text = "Добавление партнёра";
        }

        public PartnerEditForm(Partner partner) : this()
        {
            _partner = partner;
            _isEditMode = true;
            Text = "Редактирование партнёра";
            LoadData();
        }

        private void LoadData()
        {
            txtName.Text = _partner.Name;
            cmbType.SelectedItem = _partner.PartnerType;
            txtManager.Text = _partner.ManagerName;
            txtEmail.Text = _partner.Email;
            txtPhone.Text = _partner.Phone;
            txtAddress.Text = _partner.LegalAddress;
```

```
    txtInn.Text = _partner.Inn;
    numRating.Value = _partner.Rating;
}

private void btnSave_Click(object sender, EventArgs e)
{
    if (!ValidateInput()) return;

    var partner = new Partner
    {
        Id = _isEditMode ? _partner.Id : 0,
        Name = txtName.Text.Trim(),
        PartnerType = cmbType.SelectedItem.ToString(),
        ManagerName = txtManager.Text.Trim(),
        Email = txtEmail.Text.Trim(),
        Phone = txtPhone.Text.Trim(),
        LegalAddress = txtAddress.Text.Trim(),
        Inn = txtInn.Text.Trim(),
        Rating = (int)numRating.Value
    };

    try
    {
        if (!DatabaseService.IsInnUnique(partner.Inn, _isEditMode ? (int?)partner.Id : null))
        {
            MessageBox.Show("Партнёр с таким ИНН уже существует.", "Ошибка",
                MessageBoxButtons.OK, MessageBoxIcon.Error);
            return;
        }

        DatabaseService.SavePartner(partner);
        DialogResult = DialogResult.OK;
        Close();
    }
    catch (Exception ex)
    {
        MessageBox.Show($"Ошибка сохранения: {ex.Message}", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Error);
    }
}

private bool ValidateInput()
{
    if (string.IsNullOrWhiteSpace(txtName.Text))
    {
        MessageBox.Show("Укажите наименование партнёра.", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }

    if (cmbType.SelectedItem == null)
    {
        MessageBox.Show("Выберите тип партнёра.", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }

    if (txtInn.Text.Length != 10 && txtInn.Text.Length != 12)
    {
        MessageBox.Show("ИНН должен содержать 10 или 12 цифр.", "Ошибка",
            MessageBoxButtons.OK, MessageBoxIcon.Warning);
        return false;
    }

    return true;
}
```

```
        private void btnCancel_Click(object sender, EventArgs e)
        {
            DialogResult = DialogResult.Cancel;
            Close();
        }
    }
}
```

Код ServiceHistoryForm:

```
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 ПДЭ
{
    public partial class ServiceHistoryForm : Form
    {
        private int _partnerId;
        private List<ServiceHistoryItem> _history;

        public ServiceHistoryForm(int partnerId, string partnerName)
        {
            InitializeComponent();
            _partnerId = partnerId;
            Text = $"История услуг — {partnerName}";
            LoadHistory();
        }

        private void LoadHistory(int partnerId)
        {
            try
            {
                var history = DatabaseService.GetServiceHistory(partnerId);
                dataGridViewHistory.AutoGenerateColumns = false;
                dataGridViewHistory.DataSource = history;
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка загрузки истории: {ex.Message}", "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void ServiceHistoryForm_Load(object sender, EventArgs e)
        {

        }

        private void LoadHistory()
        {
            try
            {
                _history = DatabaseService.GetServiceHistory(_partnerId);
                dataGridViewHistory.AutoGenerateColumns = true;
                dataGridViewHistory.DataSource = _history;
            }
            catch (Exception ex)
```

```
            {
                MessageBox.Show($"Ошибка загрузки истории: {ex.Message}", "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }

        private void button1_Click(object sender, EventArgs e)
        {
            try
            {
                if (_history == null || !_history.Any())
                {
                    MessageBox.Show("Нет данных для расчёта.", "Информация",
                        MessageBoxButtons.OK, MessageBoxIcon.Information);
                    return;
                }

                decimal totalCost = 0;

                foreach (var item in _history)
                {
                    decimal costPerUnit = DatabaseService.CalculateServiceCost(item.ServiceCode);
                    totalCost += costPerUnit * item.Quantity;
                }

                MessageBox.Show(
                    $"Общая себестоимость оказанных услуг для партнёра:\n{totalCost:N2} руб.",
                    "Себестоимость",
                    MessageBoxButtons.OK,
                    MessageBoxIcon.Information);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Ошибка расчёта: {ex.Message}", "Ошибка",
                    MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
        }
    }
}
```

Код DatabaseService.cs:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Configuration;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ПДЭ
{
    public class User
    {
        public int ID_сущности { get; set; }
        public string Логин { get; set; }
        public int? ID_партнера { get; set; }

    }

    public static class DatabaseService
    {
        private static string ConnectionString =>
            ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;
```

```
        // Получить всех партнёров
        public static List<Partner> GetPartners()
        {
            var partners = new List<Partner>();
            string query = @"
                SELECT partner_id, name, partner_type, manager_name, email, phone,
                       legal_address, inn, rating
                FROM partners
                ORDER BY name";

            using (var conn = new SqlConnection(ConnectionString))
            using (var cmd = new SqlCommand(query, conn))
            {
                conn.Open();
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        partners.Add(new Partner
                        {
                            Id = reader.GetInt32(0),
                            Name = reader.GetString(1),
                            PartnerType = reader.GetString(2),
                            ManagerName = reader.GetString(3),
                            Email = reader.IsDBNull(4) ? "" : reader.GetString(4),
                            Phone = reader.IsDBNull(5) ? "" : reader.GetString(5),
                            LegalAddress = reader.GetString(6),
                            Inn = reader.GetString(7),
                            Rating = reader.GetInt32(8)
                        });
                    }
                }
            }
            return partners;
        }

        // Получить историю услуг по партнёру
        public static List<ServiceHistoryItem> GetServiceHistory(int partnerId)
{
            var history = new List<ServiceHistoryItem>();
            string query = @"
                SELECT s.service_name, s.service_code, h.quantity, h.execution_date
                FROM partner_service_facts h
                JOIN services s ON h.service_code = s.service_code
                WHERE h.partner_id = @partnerId
                ORDER BY h.execution_date DESC";

            using (var conn = new SqlConnection(ConnectionString))
            using (var cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@partnerId", partnerId);
                conn.Open();
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        history.Add(new ServiceHistoryItem
                        {
                            ServiceName = reader.GetString(0),
                            ServiceCode = reader.GetString(1), // ← ДОЛЖНО БЫТЬ!
                            Quantity = reader.GetInt32(2),
                            ExecutionDate = reader.GetDateTime(3)
                        });
                    }
                }
            }
```

```
            return history;
        }

        // Сохранить или обновить партнёра
        public static void SavePartner(Partner partner)
        {
            string query;
            if (partner.Id == 0)
            {
                // INSERT
                query = @"
                    INSERT INTO partners (name, partner_type, manager_name, email, phone,
                                          legal_address, inn, rating)
                    VALUES (@name, @partnerType, @managerName, @email, @phone,
                            @legalAddress, @inn, @rating)";
            }
            else
            {
                // UPDATE
                query = @"
                    UPDATE partners
                    SET name = @name, partner_type = @partnerType, manager_name = @managerName,
                        email = @email, phone = @phone, legal_address = @legalAddress,
                        inn = @inn, rating = @rating
                    WHERE partner_id = @id";
            }

            using (var conn = new SqlConnection(ConnectionString))
            using (var cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@name", partner.Name);
                cmd.Parameters.AddWithValue("@partnerType", partner.PartnerType);
                cmd.Parameters.AddWithValue("@managerName", partner.ManagerName);
                cmd.Parameters.AddWithValue("@email", string.IsNullOrEmpty(partner.Email) ? (object)DBNull.Value :
partner.Email);
                cmd.Parameters.AddWithValue("@phone", string.IsNullOrEmpty(partner.Phone) ? (object)DBNull.Value :
partner.Phone);
                cmd.Parameters.AddWithValue("@legalAddress", partner.LegalAddress);
                cmd.Parameters.AddWithValue("@inn", partner.Inn);
                cmd.Parameters.AddWithValue("@rating", partner.Rating);
                if (partner.Id != 0)
                    cmd.Parameters.AddWithValue("@id", partner.Id);

                conn.Open();
                cmd.ExecuteNonQuery();
            }
        }

        // Проверка уникальности ИНН (кроме текущего партнёра)
        public static bool IsInnUnique(string inn, int? excludeId = null)
        {
            string query = "SELECT COUNT(*) FROM partners WHERE inn = @inn";
            if (excludeId.HasValue)
                query += " AND partner_id != @excludeId";

            using (var conn = new SqlConnection(ConnectionString))
            using (var cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@inn", inn);
                if (excludeId.HasValue)
                    cmd.Parameters.AddWithValue("@excludeId", excludeId.Value);
                conn.Open();
                int count = (int)cmd.ExecuteScalar();
                return count == 0;
            }
        }
```

```
        // Расчёт себестоимости одной услуги
        public static decimal CalculateServiceCost(string serviceCode)
        {
            // Получаем норму времени (условно = 1 час на любую услугу)
            // Часовая ставка сотрудника (условно 500 руб/час)
            const decimal hourlyRate = 500m;

            // Трудозатраты = 1 час * ставка
            decimal laborCost = hourlyRate;

            // Стоимость материалов
            decimal materialCost = 0;

            string query = @"
                SELECT sm.quantity_per_unit, m.overuse_percent
                FROM service_materials sm
                JOIN materials m ON sm.material_id = m.material_id
                WHERE sm.service_code = @serviceCode";

            using (var conn = new SqlConnection(ConnectionString))
            using (var cmd = new SqlCommand(query, conn))
            {
                cmd.Parameters.AddWithValue("@serviceCode", serviceCode);
                conn.Open();
                using (var reader = cmd.ExecuteReader())
                {
                    while (reader.Read())
                    {
                        decimal quantity = reader.GetDecimal(0);
                        decimal overuse = reader.GetDecimal(1);
                        // Условно: цена материала = 100 руб за единицу
                        const decimal materialPrice = 100m;
                        decimal actualQuantity = quantity * (1 + overuse);
                        materialCost += actualQuantity * materialPrice;
                    }
                }
            }

            return laborCost + materialCost;
        }
    }

    public class Partner
    {
        public int Id { get; set; }
        public string Name { get; set; }
        public string PartnerType { get; set; }
        public string ManagerName { get; set; }
        public string Email { get; set; }
        public string Phone { get; set; }
        public string LegalAddress { get; set; }
        public string Inn { get; set; }
        public int Rating { get; set; }
    }

    public class ServiceHistoryItem
    {
        public string ServiceName { get; set; }
        public string ServiceCode { get; set; } // ← ДОЛЖНО БЫТЬ!
        public int Quantity { get; set; }
        public DateTime ExecutionDate { get; set; }
    }

}
```