Приложение А

Министерство образования и науки Нижегородской области

Государственное бюджетное профессиональное

образовательное учреждение

«Заволжский автомоторный техникум»

**Листинг**

Автоматизируемой информационной

системы проката автомобилей на базе платформы .NET

Разработал:

студент группы ИС-21Б

Кожевников С.Р.

г. Заволжье

2024– 2025 уч. год

**Листинг программы**

**AuthForm.cs** – форма авторизации

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 Course

{

public partial class AuthForm : Form

{

public AuthForm()

{

InitializeComponent();

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!(char.IsLetterOrDigit(e.KeyChar) ||

char.IsPunctuation(e.KeyChar) ||

char.IsSeparator(e.KeyChar) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

}

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

{

var userData = Connection.GetUser(textBox1.Text, textBox2.Text);

if (userData.Rows.Count == 1)

{

var role = userData.Rows[0].ItemArray[userData.Columns["RoleName"].Ordinal].ToString();

var id = userData.Rows[0].ItemArray[userData.Columns["RoleId"].Ordinal].ToString();

var workerId = userData.Rows[0].ItemArray[userData.Columns["WorkerId"].Ordinal].ToString();

User.Role = role;

User.Id = id;

User.WorkerId = workerId;

if (User.Role == "Сотрудник")

{

var f = new ListForm("product");

this.Visible = false;

f.ShowDialog();

ClearForm();

this.Visible = true;

}

else

{

var f = new MenuForm();

this.Visible = false;

f.ShowDialog();

ClearForm();

this.Visible = true;

}

}

else

{

MessageBox.Show("Неверно введены логин или пароль","",MessageBoxButtons.OK,MessageBoxIcon.Exclamation);

ClearForm();

}

}

private void ClearForm()

{

textBox1.Text = "";

textBox2.Text = "";

textBoxUnderline1.BackColor = Color.DarkRed;

textBoxUnderline2.BackColor = Color.DarkRed;

button1.Enabled = false;

}

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

{

if (textBox1.Text.Length > 0)

{

textBoxUnderline1.BackColor = Color.DarkGreen;

}

else if (textBox1.Text.Length == 0)

{

textBoxUnderline1.BackColor = Color.DarkRed;

}

button1.Enabled = textBox1.Text.Length > 0 && textBox2.Text.Length > 0;

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.Length > 0)

{

textBoxUnderline2.BackColor = Color.DarkGreen;

}

else if (textBox2.Text.Length == 0)

{

textBoxUnderline2.BackColor = Color.DarkRed;

}

button1.Enabled = textBox1.Text.Length > 0 && textBox2.Text.Length > 0;

}

private void hidePasswordButton1\_Click(object sender, EventArgs e)

{

textBox2.UseSystemPasswordChar = !textBox2.UseSystemPasswordChar;

}

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

{

if (MessageBox.Show("Выйти?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

e.Cancel = true;

}

}

}

}

**Connection.cs** – класс для взаимодействия с базой данных

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using MySql.Data.MySqlClient;

namespace Course

{

static class Connection

{

static private string connStr = "host=127.0.0.1;user=root;pwd=root;database=db35";

static private MySqlConnection conn = new MySqlConnection(connStr);

static public void Open()

{

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

{

conn.Open();

}

}

static public void Close()

{

if (conn.State != ConnectionState.Closed)

{

conn.Close();

}

}

static public DataTable GetUser(string login, string password)

{

login = login.Replace("\"", "\\\"");

var a = MD5.Create().ComputeHash(Encoding.UTF8.GetBytes(password));

password = BitConverter.ToString(a).Replace("-", string.Empty).ToLower();

Connection.Open();

var cmd = new MySqlCommand($"select \* from `user` inner join worker inner join role on WorkerId=UserWorkerId and RoleId=UserRoleId where UserLogin=\"{login}\" and UserPassword=\"{password}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

return dt;

}

static public string[] GetCategories()

{

Connection.Open();

var cmd = new MySqlCommand("select CategoryName from category", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var categories = new string[] {};

for (var i = 0; i < dt.Rows.Count; i++)

{

categories = categories.Append(dt.Rows[i].ItemArray[0].ToString()).ToArray();

}

return categories;

}

static public string[] GetSuppliers()

{

Connection.Open();

var cmd = new MySqlCommand("select SupplierName from supplier", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var suppliers = new string[] { };

for (var i = 0; i < dt.Rows.Count; i++)

{

suppliers = suppliers.Append(dt.Rows[i].ItemArray[0].ToString()).ToArray();

}

return suppliers;

}

static public KeyValuePair<string, string>[] GetWorkers()

{

Connection.Open();

var cmd = new MySqlCommand("select WorkerId, WorkerSurname, WorkerName, WorkerPatronymic from worker", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var workers = new KeyValuePair<string, string>[] { };

for (var i = 0; i < dt.Rows.Count; i++)

{

workers = workers.Append(new KeyValuePair<string, string>(dt.Rows[i].ItemArray[0].ToString(), $"{dt.Rows[i].ItemArray[1].ToString()} {dt.Rows[i].ItemArray[2].ToString()[0]}.{dt.Rows[i].ItemArray[3].ToString()[0]}.")).ToArray();

}

return workers;

}

static public string[] GetRoles()

{

Connection.Open();

var cmd = new MySqlCommand("select RoleName from role", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var roles = new string[] { };

for (var i = 0; i < dt.Rows.Count; i++)

{

roles = roles.Append(dt.Rows[i].ItemArray[0].ToString()).ToArray();

}

return roles;

}

static public Dictionary<string, string> GetProductById(string id)

{

Connection.Open();

var cmd = new MySqlCommand($"select \* from product where ProductId=\"{id.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var product = new Dictionary<string, string>();

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

{

product.Add(dt.Columns[i].ColumnName, dt.Rows[0].ItemArray[i].ToString());

}

return product;

}

static public string GetCategoryById(string id)

{

Connection.Open();

var cmd = new MySqlCommand($"select CategoryName from category where CategoryId=\"{id.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var category = dt.Rows[0].ItemArray[0].ToString();

return category;

}

static public string GetSupplierById(string id)

{

Connection.Open();

var cmd = new MySqlCommand($"select SupplierName from supplier where SupplierId=\"{id.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var supplier = dt.Rows[0].ItemArray[0].ToString();

return supplier;

}

static public string GetWorkerById(string id)

{

Connection.Open();

var cmd = new MySqlCommand($"select WorkerSurname, WorkerName, WorkerPatronymic from worker where WorkerId=\"{id.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var worker = dt.Rows[0].ItemArray[0].ToString() + " " + dt.Rows[0].ItemArray[1].ToString()[0] + "." + dt.Rows[0].ItemArray[2].ToString()[0] + ".";

return worker;

}

static public string GetRoleById(string id)

{

Connection.Open();

var cmd = new MySqlCommand($"select RoleName from role where RoleId=\"{id.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var role = dt.Rows[0].ItemArray[0].ToString();

return role;

}

static public string GetCategoryId(string category)

{

Connection.Open();

var cmd = new MySqlCommand($"select CategoryId from category where CategoryName=\"{category.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var id = dt.Rows[0].ItemArray[0].ToString();

return id;

}

static public string GetSupplierId(string supplier)

{

Connection.Open();

var cmd = new MySqlCommand($"select SupplierId from supplier where SupplierName=\"{supplier.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var id = dt.Rows[0].ItemArray[0].ToString();

return id;

}

static public string GetRoleId(string role)

{

Connection.Open();

var cmd = new MySqlCommand($"select RoleId from role where RoleName=\"{role.Replace("\"", "\\\"")}\"", conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

var dt = new DataTable();

da.Fill(dt);

Connection.Close();

var id = dt.Rows[0].ItemArray[0].ToString();

return id;

}

static public DataTable SelectTable(string tableName,

string search = "",

string filter = "",

string sort = "",

string sortMode = "asc",

int page = -1)

{

var cmdText = $"select \* from `{tableName}`";

//joining fk-s

switch (tableName)

{

case "order":

{

cmdText += "inner join worker " +

"on WorkerId=OrderWorkerId";

break;

}

case "orderitem":

{

cmdText += " inner join `order` " +

"inner join `product` " +

"inner join `worker` " +

"on OrderId=OrderItemOrderId " +

"and ProductId=OrderItemProductId " +

"and OrderWorkerId=WorkerId";

break;

}

case "product":

{

cmdText += " inner join category " +

"inner join supplier " +

"on ProductCategoryId=CategoryId " +

"and ProductSupplierId=SupplierId";

break;

}

case "user":

{

cmdText += " inner join worker " +

"inner join role " +

"on WorkerId=UserWorkerId " +

"and RoleId=UserRoleId";

break;

}

default:

{

break;

}

}

if (search != "")

{

cmdText += " where";

//get search column

switch (tableName)

{

case "user":

{

cmdText += " UserLogin";

break;

}

case "worker":

{

cmdText += " WorkerSurname";

break;

}

default:

{

cmdText += $" {tableName[0].ToString().ToUpper() + tableName.Substring(1)}Name";

break;

}

}

cmdText += $" like \"%{search}%\"";

}

if (filter != "")

{

cmdText += search != "" ? " and" : " where";

cmdText += Connection.GetFilter(filter);

}

if (sort != "")

{

cmdText += " order by";

switch (sort)

{

case "Названию":

{

cmdText += " ProductName";

break;

}

case "Стоимости":

{

cmdText += " ProductCost";

break;

}

}

cmdText += " " + sortMode;

}

else if (tableName == "product")

{

cmdText += " order by ProductId";

}

else if (tableName == "user")

{

cmdText += " order by UserId";

}

Connection.Open();

var cmd = new MySqlCommand(cmdText, conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

return dt;

}

static private string GetFilter(string condition)

{

var filter = "";

switch (condition.Substring(0, condition.IndexOf(' ') - 1))

{

case "Категория":

{

filter += $" CategoryName = \"{condition.Substring(condition.IndexOf(' ') + 1)}\"";

break;

}

case "Поставщик":

{

filter += $" SupplierName = \"{condition.Substring(condition.IndexOf(' ') + 1)}\"";

break;

}

}

return filter;

}

static private string GetTableIdColumn(string tableName)

{

return tableName[0].ToString().ToUpper() + tableName.Substring(1) + "Id";

}

static public bool InsertObject(string tableName, Dictionary<string, string> obj)

{

foreach (string key in obj.Keys.ToArray())

{

if (key == GetTableIdColumn(tableName))

{

continue;

}

else

{

obj[key] = $"\"{obj[key].Replace("\"", "\\\"")}\"";

}

}

var cmdStr = $"insert into `{tableName}`({GetTableIdColumn(tableName)}, {string.Join(", ", obj.Keys)}) values (null, {string.Join(", ", obj.Values)})";

if (tableName == "orderitem")

{

cmdStr = $"insert into `{tableName}`({string.Join(", ", obj.Keys)}) values ({string.Join(", ", obj.Values)})";

}

Connection.Open();

var tr = conn.BeginTransaction();

var cmd = new MySqlCommand(cmdStr, conn, tr);

var res = cmd.ExecuteNonQuery();

if (res != 1)

{

tr.Rollback();

}

else

{

tr.Commit();

}

Connection.Close();

return res == 1;

}

static public bool UpdateObject(string tableName, Dictionary<string, string> obj)

{

var pairs = new string[] { };

foreach (string key in obj.Keys)

{

if (key == GetTableIdColumn(tableName))

{

continue;

}

else

{

pairs = pairs.Append($"{key}=\"{obj[key].Replace("\"", "\\\"")}\"").ToArray();

}

}

var cmdStr = "";

if (tableName == "orderitem")

{

cmdStr = $"update `{tableName}` set {string.Join(", ", pairs)} where OrderItemOrderId={obj["OrderItemOrderId"]} and OrderItemProductId={obj["OrderItemProductId"]}";

}

else

{

cmdStr = $"update `{tableName}` set {string.Join(", ", pairs)} where {GetTableIdColumn(tableName)}={obj[GetTableIdColumn(tableName)]}";

}

Connection.Open();

var tr = conn.BeginTransaction();

var cmd = new MySqlCommand(cmdStr, conn, tr);

var res = cmd.ExecuteNonQuery();

if (res != 1)

{

tr.Rollback();

}

else

{

tr.Commit();

}

Connection.Close();

return res == 1;

}

static public bool UpdateObject(string tableName, Dictionary<string, string> obj, MySqlTransaction tr)

{

var pairs = new string[] { };

foreach (string key in obj.Keys)

{

if (key == GetTableIdColumn(tableName))

{

continue;

}

else

{

pairs = pairs.Append($"{key}=\"{obj[key].Replace("\"", "\\\"")}\"").ToArray();

}

}

var cmdStr = $"update `{tableName}` set {string.Join(", ", pairs)} where {GetTableIdColumn(tableName)}={obj[GetTableIdColumn(tableName)]}";

if (tableName == "orderitem")

{

cmdStr = $"update `{tableName}` set {string.Join(", ", pairs)} where OrderItemOrderId={obj["OrderItemOrderId"]} and OrderItemProductId={obj["OrderItemProductId"]}";

}

Connection.Open();

var cmd = new MySqlCommand(cmdStr, conn, tr);

var res = cmd.ExecuteNonQuery();

Connection.Close();

return res >= 1;

}

static public bool DeleteObject(string tableName, string id)

{

var cmdStr = $"delete from `{tableName}` where {GetTableIdColumn(tableName)}=\"{id.Replace("\"", "\\\"")}\"";

Connection.Open();

var tr = conn.BeginTransaction();

var cmd = new MySqlCommand(cmdStr, conn, tr);

int res;

try

{

res = cmd.ExecuteNonQuery();

}

catch (MySqlException)

{

tr.Rollback();

Connection.Close();

return false;

}

if (res < 1)

{

tr.Rollback();

}

else

{

tr.Commit();

}

Connection.Close();

return res == 1;

}

static public DataTable GetOrderItems(string orderId)

{

var cmdText = "select \* from `orderitem`";

cmdText += " inner join `product` on ProductId=OrderItemProductId";

cmdText += $" where OrderItemOrderId=\"{orderId}\"";

Connection.Open();

var cmd = new MySqlCommand(cmdText, conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

return dt;

}

static public string GetLastOrderId()

{

var cmdText = "select max(OrderId) from `order`";

Connection.Open();

var cmd = new MySqlCommand(cmdText, conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

return dt.Rows[0].ItemArray[0].ToString();

}

static public void AddItemToOrder(string orderId, Dictionary<string, string> item)

{

var cmdText = $"select OrderItemQuantity from `orderitem` where OrderItemOrderId=\"{orderId}\" and OrderItemProductId=\"{item["OrderItemProductId"]}\"";

Connection.Open();

var cmd = new MySqlCommand(cmdText, conn);

var da = new MySqlDataAdapter(cmd);

cmd.ExecuteNonQuery();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

if (dt.Rows.Count > 0)

{

item.Add("OrderItemOrderId", orderId);

item["OrderItemQuantity"] = (Convert.ToInt32(item["OrderItemQuantity"]) + Convert.ToInt32(dt.Rows[0].ItemArray[0].ToString())).ToString();

item.Remove("OrderItemCost");

Connection.UpdateObject("orderitem", item);

}

else

{

item.Add("OrderItemOrderId", orderId);

Connection.InsertObject("orderitem", item);

}

}

static public void DeleteOrderItem(string orderId, string productId)

{

var cmdStr = $"delete from `orderitem` where OrderItemOrderId=\"{orderId.Replace("\"", "\\\"")}\" and OrderItemProductId=\"{productId.Replace("\"", "\\\"")}\"";

Connection.Open();

var tr = conn.BeginTransaction();

var cmd = new MySqlCommand(cmdStr, conn, tr);

int res;

try

{

res = cmd.ExecuteNonQuery();

}

catch (MySqlException)

{

tr.Rollback();

Connection.Close();

return;

}

if (res < 1)

{

tr.Rollback();

}

else

{

tr.Commit();

}

Connection.Close();

}

static public bool ConfirmOrder(Dictionary<string, string> order)

{

var orderId = order["OrderId"];

Connection.Open();

var tr = conn.BeginTransaction();

var items = GetOrderItems(orderId);

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

{

var product = Connection.GetProductById(items.Rows[i].ItemArray[items.Columns["OrderItemProductId"].Ordinal].ToString());

if (Convert.ToInt32(product["ProductQuantity"]) < Convert.ToInt32(items.Rows[i].ItemArray[items.Columns["OrderItemQuantity"].Ordinal].ToString()))

{

Connection.Open();

tr.Rollback();

Connection.Close();

return false;

}

product["ProductQuantity"] = (Convert.ToInt32(product["ProductQuantity"]) - Convert.ToInt32(items.Rows[i].ItemArray[items.Columns["OrderItemQuantity"].Ordinal].ToString())).ToString();

product["ProductExpirationDate"] = DateTime.Parse(product["ProductExpirationDate"]).ToString("yyyy-MM-dd");

Connection.UpdateObject("product", product, tr);

}

Connection.UpdateObject("order", order, tr);

Connection.Open();

tr.Commit();

Connection.Close();

return true;

}

static public string GetProductRevenue(string productId, DateTime dateFrom, DateTime dateTo)

{

var cmdText = $"SELECT sum(OrderItemCost \* OrderItemQuantity) FROM db35.orderitem inner join `order` on OrderId = OrderItemOrderId where OrderItemProductId = \"{productId}\" and OrderDate >= \"{dateFrom.ToString("yyyy-MM-dd")}\" and OrderDate <= \"{dateTo.ToString("yyyy-MM-dd")}\"";

Connection.Open();

var cmd = new MySqlCommand(cmdText, conn);

var da = new MySqlDataAdapter(cmd);

var res = cmd.ExecuteScalar().ToString();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

try

{

var sum = Convert.ToDouble(res);

return Math.Round(sum, 2, MidpointRounding.AwayFromZero).ToString();

}

catch

{

return "0";

}

}

static public DataTable GetOrderItemsForReport(string orderId)

{

var cmdText = $"select ProductName, OrderItemQuantity, OrderItemCost, (OrderItemQuantity \* OrderItemCost) as TotalCost" +

$" from orderitem inner join product on OrderItemProductId = ProductId " +

$"where OrderItemOrderId = \"{orderId}\"";

Connection.Open();

var cmd = new MySqlCommand(cmdText, conn);

var da = new MySqlDataAdapter(cmd);

var res = cmd.ExecuteScalar().ToString();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

return dt;

}

static public string GetOrderTotalCost(string orderId)

{

var cmdText = $"select sum(OrderItemQuantity \* OrderItemCost) as TotalCost from orderitem where OrderItemOrderId = \"{orderId}\"";

Connection.Open();

var cmd = new MySqlCommand(cmdText, conn);

var da = new MySqlDataAdapter(cmd);

var res = cmd.ExecuteScalar().ToString();

Connection.Close();

var dt = new DataTable();

da.Fill(dt);

try

{

var sum = Convert.ToDouble(res);

return Math.Round(sum, 2, MidpointRounding.AwayFromZero).ToString();

}

catch

{

return "0";

}

}

}

}

**DateSelectForm.cs** – форма выбора даты

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 Course

{

public partial class DateSelectForm : Form

{

public DateSelectForm()

{

InitializeComponent();

dateTimePicker1.MaxDate = DateTime.Now.Date;

dateTimePicker2.MaxDate = DateTime.Now.Date;

dateTimePicker2.Value = DateTime.Now.Date;

}

private void dateTimePicker2\_ValueChanged(object sender, EventArgs e)

{

if (dateTimePicker1.Value.Date > dateTimePicker2.Value.Date)

dateTimePicker1.Value = dateTimePicker2.Value.Date;

dateTimePicker1.MaxDate = dateTimePicker2.Value.Date;

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

if (dateTimePicker2.Value.Date < dateTimePicker1.Value.Date)

dateTimePicker2.Value = dateTimePicker1.Value.Date;

dateTimePicker2.MinDate = dateTimePicker1.Value.Date;

}

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

{

this.DialogResult = DialogResult.OK;

this.Close();

}

}

}

**DictionaryForm.cs** – форма элемента справочника

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 Course

{

public partial class DictionaryForm : Form

{

private string tableName;

private Dictionary<string, string> entity = null;

public DictionaryForm(string tableName)

{

InitializeComponent();

this.tableName = tableName;

}

public DictionaryForm(string tableName, Dictionary<string, string> entity)

{

InitializeComponent();

this.tableName = tableName;

textBox2.Text = entity[$"{tableName[0].ToString().ToUpper() + tableName.Substring(1)}Name"];

this.entity = entity;

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("ёйцукенгшщзхъфывапролджэячсмитьбю1234567890- ".Contains(e.KeyChar.ToString().ToLower()) ||

char.IsPunctuation(e.KeyChar) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.Length > 0)

{

label3.BackColor = Color.DarkGreen;

}

else

{

label3.BackColor = Color.DarkRed;

}

button1.Enabled = label3.BackColor == Color.DarkGreen;

}

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

{

var entitySelected = entity != null;

if (entity == null)

{

entity = new Dictionary<string, string>();

}

entity[tableName[0].ToString().ToUpper() + tableName.Substring(1) + "Name"] = textBox2.Text;

if (entitySelected)

{

//update

if (MessageBox.Show("Редактировать запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.UpdateObject(tableName, entity))

{

MessageBox.Show("Запись редактирована", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось редактировать запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

else

{

//insert

if (MessageBox.Show("Добавить запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.InsertObject(tableName, entity))

{

MessageBox.Show("Запись добавлена", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось добавить запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

}

}

**ListForm.cs** – форма списка

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.ComponentModel;

using System.Data;

using System.Diagnostics;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Course

{

public partial class ListForm : Form

{

private string tableName;

private string orderId = null;

private bool fillingOrder = false;

public ListForm(string tableName, string orderId = null)

{

InitializeComponent();

this.tableName = tableName;

if (tableName == "product" && orderId != null)

{

this.orderId = orderId;

fillingOrder = true;

}

}

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

{

if (User.Role == "Сотрудник")

{

flowLayoutPanel2.Visible = false;

contextMenuStrip1.Items.Remove(contextMenuStrip1.Items["добавитьToolStripMenuItem"]);

contextMenuStrip1.Items.Remove(contextMenuStrip1.Items["редактироватьToolStripMenuItem"]);

contextMenuStrip1.Items.Remove(contextMenuStrip1.Items["удалитьToolStripMenuItem"]);

}

if (User.Role == "Менеджер")

{

contextMenuStrip1.Items.Remove(contextMenuStrip1.Items["удалитьToolStripMenuItem"]);

}

if (tableName != "product")

{

panel2.Visible = false;

flowLayoutPanel2.Visible = false;

contextMenuStrip1.Items.Remove(contextMenuStrip1.Items["добавитьВЗаказToolStripMenuItem"]);

}

if (tableName == "order")

{

panel1.Visible = false;

contextMenuStrip1.Items.Remove(contextMenuStrip1.Items["добавитьToolStripMenuItem"]);

}

FillCombos();

FillDGVData();

}

private void FillCombos()

{

if (panel2.Visible)

{

var categories = Connection.GetCategories();

for (var i = 0; i < categories.Length; i++)

{

categories[i] = "Категория: " + categories[i];

}

var suppliers = Connection.GetSuppliers();

for (var i = 0; i < suppliers.Length; i++)

{

suppliers[i] = "Поставщик: " + suppliers[i];

}

comboBox2.DataSource = new string[] { "Не фильтровать" }.Concat(categories.Concat(suppliers).ToArray()).ToArray();

comboBox1.DataSource = new string[] { "Не сортировать", "Названию", "Стоимости" };

comboBox2.SelectedIndex = 0;

comboBox1.SelectedIndex = 0;

}

}

private void FillDGVData()

{

dataGridView1.DataSource = Connection.SelectTable(tableName,

textBox1.Text,

comboBox2.SelectedIndex > 0 ? comboBox2.SelectedValue.ToString():"",

comboBox1.SelectedIndex > 0 ? comboBox1.SelectedValue.ToString():"",

button5.Text == "↓"?"desc":"asc");

for (var i = 0; i < dataGridView1.Columns.Count; i++)

{

dataGridView1.Columns[i].SortMode = DataGridViewColumnSortMode.NotSortable;

dataGridView1.Columns[i].Visible = false;

}

switch (tableName)

{

case "category":

{

dataGridView1.Columns["CategoryId"].Visible = true;

dataGridView1.Columns["CategoryName"].Visible = true;

dataGridView1.Columns["CategoryId"].HeaderText = "#";

dataGridView1.Columns["CategoryName"].HeaderText = "Наименование";

break;

}

case "supplier":

{

dataGridView1.Columns["SupplierId"].Visible = true;

dataGridView1.Columns["SupplierName"].Visible = true;

dataGridView1.Columns["SupplierId"].HeaderText = "#";

dataGridView1.Columns["SupplierName"].HeaderText = "Наименование";

break;

}

case "order":

{

dataGridView1.Columns["OrderId"].Visible = true;

dataGridView1.Columns["OrderDate"].Visible = true;

dataGridView1.Columns["OrderStatus"].Visible = true;

dataGridView1.Columns["OrderId"].HeaderText = "#";

dataGridView1.Columns["OrderDate"].HeaderText = "Дата заказа";

dataGridView1.Columns["OrderStatus"].HeaderText = "Статус";

SetFioColumn(1);

break;

}

case "worker":

{

dataGridView1.Columns["WorkerId"].Visible = true;

dataGridView1.Columns["WorkerPhone"].Visible = true;

dataGridView1.Columns["WorkerId"].HeaderText = "#";

dataGridView1.Columns["WorkerPhone"].HeaderText = "Телефон";

SetFioColumn(1);

break;

}

case "user":

{

dataGridView1.Columns["UserId"].Visible = true;

dataGridView1.Columns["UserLogin"].Visible = true;

dataGridView1.Columns["RoleName"].Visible = true;

dataGridView1.Columns["UserId"].HeaderText = "#";

dataGridView1.Columns["UserLogin"].HeaderText = "Логин";

dataGridView1.Columns["RoleName"].HeaderText = "Роль";

SetFioColumn(1);

break;

}

case "product":

{

dataGridView1.Columns["ProductId"].Visible = true;

dataGridView1.Columns["ProductName"].Visible = true;

dataGridView1.Columns["ProductQuantity"].Visible = true;

dataGridView1.Columns["CategoryName"].Visible = true;

dataGridView1.Columns["SupplierName"].Visible = true;

dataGridView1.Columns["ProductId"].HeaderText = "#";

dataGridView1.Columns["ProductName"].HeaderText = "Наименование";

dataGridView1.Columns["ProductQuantity"].HeaderText = "Кол-во";

dataGridView1.Columns["CategoryName"].HeaderText = "Категория";

dataGridView1.Columns["SupplierName"].HeaderText = "Поставщик";

SetImageColumn(1);

SetTotalCostColumn(5);

break;

}

}

}

private void SetFioColumn(int columnIndex)

{

DataGridViewTextBoxColumn col = new DataGridViewTextBoxColumn();

col.Name = "WorkerFio";

col.HeaderText = "ФИО";

try

{

dataGridView1.Columns.Remove("WorkerFio");

}

catch

{

;

}

dataGridView1.Columns.Insert(columnIndex, col);

for (int i = 0; i < dataGridView1.Rows.Count; i++)

{

dataGridView1[columnIndex, i].Value = $"{dataGridView1["WorkerSurname", i].Value} {dataGridView1["WorkerName", i].Value.ToString()[0]}.{dataGridView1["WorkerPatronymic", i].Value.ToString()[0]}.";

}

}

private void SetImageColumn(int columnIndex)

{

var col = new DataGridViewImageColumn();

col.Name = "Image";

col.HeaderText = "Изображение";

col.ImageLayout = DataGridViewImageCellLayout.Stretch;

try

{

dataGridView1.Columns.Remove("Image");

}

catch

{

;

}

dataGridView1.Columns.Insert(columnIndex, col);

for (var i = 0; i < dataGridView1.Rows.Count; i++)

{

dataGridView1.Rows[i].Height = 80;

if (dataGridView1["ProductImage", i].Value.ToString() != "")

{

try

{

var imgPath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\" + dataGridView1["ProductImage", i].Value.ToString();

dataGridView1.Rows[i].Cells[columnIndex].Value = new Bitmap(imgPath);

}

catch (Exception)

{

dataGridView1.Rows[i].Cells[columnIndex].Value = Properties.Resources.picture;

}

}

else

{

dataGridView1.Rows[i].Cells[columnIndex].Value = Properties.Resources.picture;

}

}

}

private void SetTotalCostColumn(int columnIndex)

{

DataGridViewTextBoxColumn col = new DataGridViewTextBoxColumn();

col.Name = "TotalCost";

col.HeaderText = "Стоимость";

col.ValueType = typeof(float);

try

{

dataGridView1.Columns.Remove("TotalCost");

}

catch

{

;

}

dataGridView1.Columns.Insert(columnIndex, col);

for (var i = 0; i < dataGridView1.Rows.Count; i++)

{

if (dataGridView1["ProductDiscount", i].Value.ToString() != "0")

{

dataGridView1[columnIndex, i].Value = Math.Round(Convert.ToSingle(dataGridView1["ProductCost", i].Value.ToString()) \* (Convert.ToSingle((100 - Convert.ToInt32(dataGridView1["ProductDiscount", i].Value.ToString()))) / 100), 2, MidpointRounding.AwayFromZero).ToString();

}

else

{

dataGridView1[columnIndex, i].Value = dataGridView1["ProductCost", i].Value.ToString();

}

}

}

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

{

DialogResult res = DialogResult.None;

switch (tableName)

{

case "product":

{

var f = new ProductForm();

res = f.ShowDialog();

break;

}

case "user":

{

var f = new UserForm();

res = f.ShowDialog();

break;

}

case "worker":

{

var f = new WorkerForm();

res = f.ShowDialog();

break;

}

case "supplier":

{

var f = new DictionaryForm(tableName);

res = f.ShowDialog();

break;

}

case "category":

{

var f = new DictionaryForm(tableName);

res = f.ShowDialog();

break;

}

}

if (res == DialogResult.OK)

{

FillDGVData();

}

}

private void button5\_Click(object sender, EventArgs e)

{

if (button5.Text == "↑")

{

button5.Text = "↓";

}

else

{

button5.Text = "↑";

}

if (comboBox1.SelectedIndex > 0)

{

FillDGVData();

}

}

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

{

if (textBox1.Text.Length >= 2 || textBox1.Text == "")

{

FillDGVData();

}

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if ((!(("ёйцукенгшщзхъфывапролджэячсмитьбю1234567890- ".Contains(e.KeyChar.ToString().ToLower()) ||

char.IsPunctuation(e.KeyChar) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete)) && tableName != "user") ||

(!((char.IsLetterOrDigit(e.KeyChar) ||

char.IsPunctuation(e.KeyChar) ||

"- ".Contains(e.KeyChar.ToString()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete)) && tableName == "user"))

{

e.Handled = true;

return;

}

}

private void comboBox2\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0)

return;

using (var brush = new SolidBrush(e.ForeColor))

{

e.DrawBackground();

e.Graphics.DrawString(comboBox2.Items[e.Index].ToString(), e.Font, brush, e.Bounds);

e.DrawFocusRectangle();

}

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox2.SelectedIndex >= 0)

{

FillDGVData();

}

}

private void comboBox1\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0)

return;

using (var brush = new SolidBrush(e.ForeColor))

{

e.DrawBackground();

e.Graphics.DrawString(comboBox1.Items[e.Index].ToString(), e.Font, brush, e.Bounds);

e.DrawFocusRectangle();

}

}

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

{

if (comboBox1.SelectedIndex >= 0)

{

FillDGVData();

}

}

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

{

if (dataGridView1.SelectedRows.Count <= 0)

{

return;

}

DialogResult res = DialogResult.None;

var rowIndex = dataGridView1.SelectedRows[0].Index;

var entity = new Dictionary<string, string>();

switch (tableName)

{

case "product":

{

entity.Add("ProductId", dataGridView1["ProductId", rowIndex].Value.ToString());

entity.Add("ProductName", dataGridView1["ProductName", rowIndex].Value.ToString());

entity.Add("ProductQuantity", dataGridView1["ProductQuantity", rowIndex].Value.ToString());

entity.Add("ProductCost", dataGridView1["ProductCost", rowIndex].Value.ToString());

entity.Add("ProductDiscount", dataGridView1["ProductDiscount", rowIndex].Value.ToString());

entity.Add("ProductCategoryId", dataGridView1["ProductCategoryId", rowIndex].Value.ToString());

entity.Add("ProductSupplierId", dataGridView1["ProductSupplierId", rowIndex].Value.ToString());

entity.Add("ProductExpirationDate", dataGridView1["ProductExpirationDate", rowIndex].Value.ToString());

entity.Add("ProductImage", dataGridView1["ProductImage", rowIndex].Value.ToString());

var f = new ProductForm(entity);

res = f.ShowDialog();

break;

}

case "order":

{

entity.Add("OrderId", dataGridView1["OrderId", rowIndex].Value.ToString());

entity.Add("OrderWorkerId", dataGridView1["OrderWorkerId", rowIndex].Value.ToString());

entity.Add("OrderDate", dataGridView1["OrderDate", rowIndex].Value.ToString());

entity.Add("OrderStatus", dataGridView1["OrderStatus", rowIndex].Value.ToString());

var f = new OrderForm(entity);

res = f.ShowDialog();

break;

}

case "user":

{

entity.Add("UserId", dataGridView1["UserId", rowIndex].Value.ToString());

entity.Add("UserLogin", dataGridView1["UserLogin", rowIndex].Value.ToString());

entity.Add("UserRoleId", dataGridView1["UserRoleId", rowIndex].Value.ToString());

entity.Add("UserWorkerId", dataGridView1["UserWorkerId", rowIndex].Value.ToString());

var f = new UserForm(entity);

res = f.ShowDialog();

break;

}

case "worker":

{

entity.Add("WorkerId", dataGridView1["WorkerId", rowIndex].Value.ToString());

entity.Add("WorkerSurname", dataGridView1["WorkerSurname", rowIndex].Value.ToString());

entity.Add("WorkerName", dataGridView1["WorkerName", rowIndex].Value.ToString());

entity.Add("WorkerPatronymic", dataGridView1["WorkerPatronymic", rowIndex].Value.ToString());

entity.Add("WorkerPhone", dataGridView1["WorkerPhone", rowIndex].Value.ToString());

var f = new WorkerForm(entity);

res = f.ShowDialog();

break;

}

case "supplier":

{

entity.Add("SupplierId", dataGridView1["SupplierId", rowIndex].Value.ToString());

entity.Add("SupplierName", dataGridView1["SupplierName", rowIndex].Value.ToString());

var f = new DictionaryForm(tableName, entity);

res = f.ShowDialog();

break;

}

case "category":

{

entity.Add("CategoryId", dataGridView1["CategoryId", rowIndex].Value.ToString());

entity.Add("CategoryName", dataGridView1["CategoryName", rowIndex].Value.ToString());

var f = new DictionaryForm(tableName, entity);

res = f.ShowDialog();

break;

}

}

if (res == DialogResult.OK)

{

FillDGVData();

}

}

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

{

if (dataGridView1.SelectedRows.Count <= 0)

{

return;

}

DialogResult res = DialogResult.None;

var rowIndex = dataGridView1.SelectedRows[0].Index;

var entity = new Dictionary<string, string>();

res = MessageBox.Show("Удалить запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Exclamation);

if (res == DialogResult.OK)

{

if (Connection.DeleteObject(tableName, dataGridView1[tableName[0].ToString().ToUpper() + tableName.Substring(1) + "Id", rowIndex].Value.ToString()))

{

MessageBox.Show("Запись успешно удалена", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

FillDGVData();

}

else

{

MessageBox.Show("Не удалось удалить запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

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

{

ReportBuilder.ProductQuantityReport(Connection.SelectTable(tableName,

textBox1.Text,

comboBox2.SelectedIndex > 0 ? comboBox2.SelectedValue.ToString() : "",

comboBox1.SelectedIndex > 0 ? comboBox1.SelectedValue.ToString() : "",

button5.Text == "↓" ? "desc" : "asc"));

}

private void button6\_Click(object sender, EventArgs e)

{

var f = new OrderForm(orderId);

var res = f.ShowDialog();

if (res == DialogResult.OK)

{

orderId = null;

}

}

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

{

if (dataGridView1.SelectedRows.Count <= 0)

{

return;

}

if (orderId == null)

{

var order = new Dictionary<string, string>();

order.Add("OrderWorkerId", User.WorkerId);

order.Add("OrderDate", DateTime.Now.Date.ToString("yyyy-MM-dd"));

order.Add("OrderStatus", "Новый");

Connection.InsertObject("order", order);

this.orderId = Connection.GetLastOrderId();

}

if (!this.fillingOrder)

{

button6.Visible = true;

}

var rowIndex = dataGridView1.SelectedRows[0].Index;

var orderItem = new Dictionary<string, string>();

orderItem.Add("OrderItemProductId", dataGridView1["ProductId", rowIndex].Value.ToString());

orderItem.Add("OrderItemQuantity", "1");

orderItem.Add("OrderItemCost", dataGridView1["TotalCost", rowIndex].Value.ToString().Replace(",", "."));

Connection.AddItemToOrder(orderId, orderItem);

}

private void button4\_Click(object sender, EventArgs e)

{

var f = new DateSelectForm();

var res = f.ShowDialog();

if (res != DialogResult.OK)

{

return;

}

ReportBuilder.ProductRevenueReport(Connection.SelectTable(tableName,

textBox1.Text,

comboBox2.SelectedIndex > 0 ? comboBox2.SelectedValue.ToString() : "",

comboBox1.SelectedIndex > 0 ? comboBox1.SelectedValue.ToString() : "",

button5.Text == "↓" ? "desc" : "asc"),

f.dateTimePicker1.Value.Date,

f.dateTimePicker2.Value.Date);

}

}

}

**MenuForm.cs** – форма меню

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 Course

{

public partial class MenuForm : Form

{

public MenuForm()

{

InitializeComponent();

}

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

{

var f = new ListForm("product");

this.Visible = false;

f.ShowDialog();

this.Visible = true;

}

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

{

var f = new ListForm("order");

this.Visible = false;

f.ShowDialog();

this.Visible = true;

}

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

{

var f = new ListForm("category");

this.Visible = false;

f.ShowDialog();

this.Visible = true;

}

private void button4\_Click(object sender, EventArgs e)

{

var f = new ListForm("supplier");

this.Visible = false;

f.ShowDialog();

this.Visible = true;

}

private void button5\_Click(object sender, EventArgs e)

{

var f = new ListForm("worker");

this.Visible = false;

f.ShowDialog();

this.Visible = true;

}

private void button6\_Click(object sender, EventArgs e)

{

var f = new ListForm("user");

this.Visible = false;

f.ShowDialog();

this.Visible = true;

}

}

}

**OrderForm.cs** – форма заказа

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 static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Course

{

public partial class OrderForm : Form

{

private Dictionary<string, string> order = new Dictionary<string, string>();

public OrderForm()

{

InitializeComponent();

comboBox2.DisplayMember = "Value";

comboBox2.ValueMember = "Key";

FillCombos();

comboBox2.SelectedValue = User.WorkerId;

dateTimePicker1.Value = DateTime.Now;

order.Add("OrderWorkerId", comboBox2.SelectedValue.ToString());

order.Add("OrderDate", dateTimePicker1.Value.ToString("yyyy-MM-dd"));

order.Add("OrderStatus", "Новый");

Connection.InsertObject("order", order);

}

public OrderForm(string orderId)

{

InitializeComponent();

comboBox2.DisplayMember = "Value";

comboBox2.ValueMember = "Key";

FillCombos();

comboBox2.SelectedValue = User.WorkerId;

dateTimePicker1.Value = DateTime.Now;

order.Add("OrderId", orderId);

order.Add("OrderWorkerId", comboBox2.SelectedValue.ToString());

order.Add("OrderDate", dateTimePicker1.Value.ToString("yyyy-MM-dd"));

order.Add("OrderStatus", "Новый");

if (User.Role == "Сотрудник")

{

comboBox2.Enabled = false;

dateTimePicker1.Enabled = false;

}

FillDGV();

}

public OrderForm(Dictionary<string, string> order)

{

InitializeComponent();

comboBox2.DisplayMember = "Value";

comboBox2.ValueMember = "Key";

FillCombos();

comboBox2.SelectedValue = order["OrderWorkerId"];

dateTimePicker1.Value = DateTime.Parse(order["OrderDate"]).Date;

if (order["OrderStatus"] == "Проведён")

{

button1.Text = "Сформировать чек";

dataGridView1.ContextMenuStrip = null;

comboBox2.Enabled = false;

dateTimePicker1.Enabled = false;

}

this.order = order;

FillDGV();

}

private void FillCombos()

{

comboBox2.DataSource = Connection.GetWorkers();

}

private void FillDGV()

{

dataGridView1.DataSource = Connection.GetOrderItems(order["OrderId"]);

for (int i = 0; i < dataGridView1.Columns.Count; i++)

{

dataGridView1.Columns[i].Visible = false;

}

dataGridView1.Columns["OrderItemCost"].Visible = true;

dataGridView1.Columns["OrderItemQuantity"].Visible = true;

dataGridView1.Columns["ProductName"].Visible = true;

dataGridView1.Columns["ProductName"].HeaderText = "Товар";

dataGridView1.Columns["OrderItemQuantity"].HeaderText = "Количество";

dataGridView1.Columns["OrderItemCost"].HeaderText = "Стоимость";

SwitchButton();

}

private void SwitchButton()

{

button1.Enabled = label11.BackColor == Color.DarkGreen && label14.BackColor == Color.DarkGreen && dataGridView1.Rows.Count > 0;

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox2.SelectedIndex >= 0)

{

label11.BackColor = Color.DarkGreen;

}

else

{

label11.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void comboBox2\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0)

return;

using (var brush = new SolidBrush(e.ForeColor))

{

e.DrawBackground();

e.Graphics.DrawString(((KeyValuePair<string, string>)comboBox2.Items[e.Index]).Value.ToString(), e.Font, brush, e.Bounds);

e.DrawFocusRectangle();

}

}

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

{

}

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

{

button1.Enabled = false;

switch (button1.Text)

{

case "Провести":

{

order["OrderWorkerId"] = comboBox2.SelectedValue.ToString();

order["OrderDate"] = dateTimePicker1.Value.Date.ToString("yyyy-MM-dd");

order["OrderStatus"] = "Проведён";

if (Connection.ConfirmOrder(order))

{

button1.Text = "Сформировать чек";

dataGridView1.ContextMenuStrip = null;

comboBox2.Enabled = false;

dateTimePicker1.Enabled = false;

}

else

{

order["OrderStatus"] = "Новый";

}

break;

}

case "Сформировать чек":

{

//make report

ReportBuilder.OrderItemsReport(order["OrderId"], dateTimePicker1.Value.Date, ((KeyValuePair<string, string>)comboBox2.SelectedItem).Value);

break;

}

}

button1.Enabled = true;

}

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

{

var f = new ListForm("product", order["OrderId"]);

this.Visible = false;

f.ShowDialog();

this.Visible = true;

FillDGV();

}

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

{

if (order["OrderStatus"] == "Проведён")

{

return;

}

if (dataGridView1.SelectedRows.Count < 1)

{

return;

}

if (MessageBox.Show("Удалить запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Exclamation) != DialogResult.OK)

{

return;

}

var rowIndex = dataGridView1.SelectedRows[0].Index;

Connection.DeleteOrderItem(dataGridView1["OrderItemOrderId", rowIndex].Value.ToString(), dataGridView1["OrderItemProductId", rowIndex].Value.ToString());

FillDGV();

}

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

{

if (order["OrderStatus"] == "Проведён")

{

this.DialogResult = DialogResult.OK;

}

}

}

}

**ProductForm.cs** – форма товара

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Security.Permissions;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Course

{

public partial class ProductForm : Form

{

Dictionary<string, string> product = null;

private string imagePath = "";

public ProductForm()

{

InitializeComponent();

FillCombos();

pictureBox1.Image = Properties.Resources.picture;

}

public ProductForm(Dictionary<string, string> product)

{

InitializeComponent();

FillCombos();

textBox2.Text = product["ProductName"];

textBox3.Text = product["ProductQuantity"];

textBox4.Text = product["ProductCost"];

textBox5.Text = product["ProductDiscount"];

dateTimePicker1.Value = DateTime.Parse(product["ProductExpirationDate"]);

comboBox1.SelectedItem = Connection.GetCategoryById(product["ProductCategoryId"]);

comboBox2.SelectedItem = Connection.GetSupplierById(product["ProductSupplierId"]);

Directory.CreateDirectory(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\");

try

{

pictureBox1.Image = new Bitmap(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\" + product["ProductImage"]);

imagePath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\" + product["ProductImage"];

}

catch (Exception)

{

pictureBox1.Image = Properties.Resources.picture;

}

this.product = product;

}

private void FillCombos()

{

comboBox1.DataSource = Connection.GetCategories();

comboBox2.DataSource = Connection.GetSuppliers();

}

private void SwitchButton()

{

button1.Enabled = label3.BackColor == Color.DarkGreen &&

label5.BackColor == Color.DarkGreen &&

label7.BackColor == Color.DarkGreen &&

label9.BackColor == Color.DarkGreen &&

label14.BackColor == Color.DarkGreen &&

label11.BackColor == Color.DarkGreen &&

comboBoxUnderline1.BackColor == Color.DarkGreen;

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("ёйцукенгшщзхъфывапролджэячсмитьбю1234567890- ".Contains(e.KeyChar.ToString().ToLower()) ||

char.IsPunctuation(e.KeyChar) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.Length > 0)

{

label3.BackColor = Color.DarkGreen;

}

else

{

label3.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("1234567890".Contains(e.KeyChar.ToString()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

if (textBox3.Text.Length > 0)

{

label5.BackColor = Color.DarkGreen;

}

else

{

label5.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void textBox4\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("1234567890.".Contains(e.KeyChar.ToString()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

if ((textBox4.Text.Length == 0 || textBox4.Text.Contains('.')) && e.KeyChar == '.')

{

e.Handled = true;

return;

}

if (textBox4.Text.Length > 3 && textBox4.Text[textBox4.Text.Length - 3] == '.')

{

e.Handled = true;

return;

}

}

private void textBox4\_TextChanged(object sender, EventArgs e)

{

if (textBox4.Text.Length > 0)

{

label7.BackColor = Color.DarkGreen;

}

else

{

label7.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void textBox5\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("1234567890".Contains(e.KeyChar.ToString()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox2.SelectedIndex >= 0)

{

label11.BackColor = Color.DarkGreen;

}

else

{

label11.BackColor = Color.DarkRed;

}

SwitchButton();

}

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

{

if (comboBox1.SelectedIndex >= 0)

{

comboBoxUnderline1.BackColor = Color.DarkGreen;

}

else

{

comboBoxUnderline1.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void textBox5\_TextChanged(object sender, EventArgs e)

{

if (textBox5.Text.Length > 0)

{

label9.BackColor = Color.DarkGreen;

}

else

{

label9.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void comboBox2\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0)

return;

using (var brush = new SolidBrush(e.ForeColor))

{

e.DrawBackground();

e.Graphics.DrawString(comboBox2.Items[e.Index].ToString(), e.Font, brush, e.Bounds);

e.DrawFocusRectangle();

}

}

private void comboBox1\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0)

return;

using (var brush = new SolidBrush(e.ForeColor))

{

e.DrawBackground();

e.Graphics.DrawString(comboBox1.Items[e.Index].ToString(), e.Font, brush, e.Bounds);

e.DrawFocusRectangle();

}

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

var file = new OpenFileDialog();

file.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\";

file.Filter = "jpg images|\*.jpg";

file.Multiselect = false;

var res = file.ShowDialog();

if (res == DialogResult.OK)

{

pictureBox1.Image.Dispose();

imagePath = file.FileName;

pictureBox1.Image = new Bitmap(file.FileName);

}

file.Dispose();

}

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

{

var productSelected = product != null;

//copy image to res

int i = 1;

var newImg = imagePath.Substring(imagePath.LastIndexOf("\\") + 1);

while (File.Exists(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\" + newImg))

{

newImg = imagePath.Substring(imagePath.LastIndexOf("\\") + 1, imagePath.Substring(imagePath.LastIndexOf("\\") + 1).Length - 4) + $" ({i}).jpg";

i++;

}

try

{

File.Copy(imagePath,

Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\" + newImg,

true);

imagePath = Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\SetupInstaller\\Resources\\ProductImages\\" + newImg;

}

catch (Exception ex)

{

MessageBox.Show("Не удалось установить изображение:\n" + imagePath + "\n" + ex.Message, "", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (imagePath != "")

{

imagePath = imagePath.Substring(imagePath.LastIndexOf("\\") + 1);

}

if (product == null)

{

product = new Dictionary<string, string>();

}

product["ProductName"] = textBox2.Text;

product["ProductQuantity"] = textBox3.Text;

product["ProductCost"] = textBox4.Text;

product["ProductDiscount"] = textBox5.Text;

product["ProductExpirationDate"] = dateTimePicker1.Value.ToString("yyyy-MM-dd");

product["ProductCategoryId"] = Connection.GetCategoryId(comboBox1.SelectedItem.ToString());

product["ProductSupplierId"] = Connection.GetSupplierId(comboBox2.SelectedItem.ToString());

product["ProductImage"] = imagePath;

if (productSelected)

{

//update

if (MessageBox.Show("Редактировать запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.UpdateObject("product", product))

{

MessageBox.Show("Запись редактирована", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось редактировать запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

else

{

//insert

if (MessageBox.Show("Добавить запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.InsertObject("product", product))

{

MessageBox.Show("Запись добавлена", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось добавить запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

}

}

**Program.cs** – класс, инициализирующий приложение

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Course

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new AuthForm());

}

}

}

**ReportBuilder.cs** – класс сборки отчётов

using System;

using System.Collections.Generic;

using System.Data;

using System.IO;

using System.Linq;

using System.Runtime.InteropServices;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using Excel = Microsoft.Office.Interop.Excel;

namespace Course

{

class ReportBuilder

{

static void ReleaseObject(object obj)

{

try

{

Marshal.FinalReleaseComObject(obj);

obj = null;

}

catch (Exception)

{

obj = null;

}

finally

{

GC.Collect();

}

}

public static void ProductQuantityReport(DataTable data)

{

var excelApp = new Excel.Application();

Excel.Workbooks workbooks = excelApp.Workbooks;

Excel.Workbook workbook = workbooks.Add(Type.Missing);

Excel.Sheets worksheets = workbook.Worksheets;

Excel.Worksheet worksheet = worksheets.Add();

worksheet.Cells[1, "A"] = "Остатки товаров";

worksheet.Cells[2, "A"] = "#";

worksheet.Cells[2, "B"] = "Наименование";

worksheet.Cells[2, "C"] = "Количество";

var table = data;

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

{

worksheet.Cells[i + 3, "A"] = table.Rows[i].ItemArray[table.Columns["ProductId"].Ordinal].ToString();

worksheet.Cells[i + 3, "B"] = table.Rows[i].ItemArray[table.Columns["ProductName"].Ordinal].ToString();

worksheet.Cells[i + 3, "C"] = table.Rows[i].ItemArray[table.Columns["ProductQuantity"].Ordinal].ToString();

}

excelApp.Visible = true;

ReleaseObject(worksheet);

ReleaseObject(worksheets);

ReleaseObject(workbook);

ReleaseObject(workbooks);

excelApp.Quit();

ReleaseObject(excelApp);

}

public static void ProductRevenueReport(DataTable data, DateTime dateFrom, DateTime dateTo)

{

var excelApp = new Excel.Application();

Excel.Workbooks workbooks = excelApp.Workbooks;

Excel.Workbook workbook = workbooks.Add(Type.Missing);

Excel.Sheets worksheets = workbook.Worksheets;

Excel.Worksheet worksheet = worksheets.Add();

worksheet.Cells[1, "A"] = $"Выручка товаров за период {dateFrom.Date.ToString("dd.MM.yyyy")}-{dateTo.Date.ToString("dd.MM.yyyy")}";

worksheet.Cells[2, "A"] = "#";

worksheet.Cells[2, "B"] = "Наименование";

worksheet.Cells[2, "C"] = "Сумма";

var table = data;

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

{

var id = table.Rows[i].ItemArray[table.Columns["ProductId"].Ordinal].ToString();

var revenue = Connection.GetProductRevenue(id, dateFrom, dateTo);

worksheet.Cells[i + 3, "A"] = id;

worksheet.Cells[i + 3, "B"] = table.Rows[i].ItemArray[table.Columns["ProductName"].Ordinal].ToString();

worksheet.Cells[i + 3, "C"] = revenue;

}

excelApp.Visible = true;

ReleaseObject(worksheet);

ReleaseObject(worksheets);

ReleaseObject(workbook);

ReleaseObject(workbooks);

excelApp.Quit();

ReleaseObject(excelApp);

}

public static void OrderItemsReport(string orderId, DateTime orderDate, string orderWorker)

{

var excelApp = new Excel.Application();

Excel.Workbooks workbooks = excelApp.Workbooks;

Excel.Workbook workbook = workbooks.Add(Type.Missing);

Excel.Sheets worksheets = workbook.Worksheets;

Excel.Worksheet worksheet = worksheets.Add();

worksheet.Cells[1, "A"] = "ИП \"Пекарни Круглова\"";

worksheet.Cells[2, "A"] = "019193, г. Городец, ул. Ленина 52";

worksheet.Cells[3, "A"] = "РН ККТ 0893512481100103";

worksheet.Cells[3, "B"] = "ФН 1872827102002873";

worksheet.Cells[4, "A"] = "ККТ 1231541792055631";

worksheet.Cells[4, "B"] = "ИНН 1023715210";

worksheet.Cells[5, "A"] = "КАССОВЫЙ ЧЕК/ПРИХОД";

worksheet.Cells[5, "B"] = orderId;

worksheet.Cells[6, "A"] = "КАССИР";

worksheet.Cells[6, "B"] = orderWorker;

worksheet.Cells[7, "B"] = orderDate.ToString("dd.MM.yyyy");

var table = Connection.GetOrderItemsForReport(orderId);

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

{

var quantity = table.Rows[i].ItemArray[table.Columns["OrderItemQuantity"].Ordinal].ToString();

var cost = table.Rows[i].ItemArray[table.Columns["OrderItemCost"].Ordinal].ToString();

worksheet.Cells[i \* 2 + 8, "A"] = table.Rows[i].ItemArray[table.Columns["ProductName"].Ordinal].ToString();

worksheet.Cells[i \* 2 + 8, "B"] = $"{quantity}x{cost}р";

worksheet.Cells[i \* 2 + 9, "B"] = table.Rows[i].ItemArray[table.Columns["TotalCost"].Ordinal].ToString() + "р";

}

var row = table.Rows.Count \* 2 + 8;

worksheet.Cells[row, "A"] = "ИТОГО:";

worksheet.Cells[row, "B"] = Connection.GetOrderTotalCost(orderId) + "р";

row++;

worksheet.Cells[row, "A"] = "Сайт ФНС:";

worksheet.Cells[row, "B"] = "www.nalog.ru";

excelApp.Visible = true;

ReleaseObject(worksheet);

ReleaseObject(worksheets);

ReleaseObject(workbook);

ReleaseObject(workbooks);

excelApp.Quit();

ReleaseObject(excelApp);

}

}

}

**User.cs** – объект, содержащий данные о текущем пользователе

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace Course

{

static class User

{

static public string Id;

static public string Role;

static public string WorkerId;

}

}

**UserForm.cs** – форма 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;

namespace Course

{

public partial class UserForm : Form

{

Dictionary<string, string> user = null;

public UserForm()

{

InitializeComponent();

comboBox1.DisplayMember = "Value";

comboBox1.ValueMember = "Key";

FillCombos();

}

public UserForm(Dictionary<string, string> user)

{

InitializeComponent();

comboBox1.DisplayMember = "Value";

comboBox1.ValueMember = "Key";

FillCombos();

textBox2.Text = user["UserLogin"];

comboBox1.SelectedValue = user["UserWorkerId"];

comboBox2.SelectedItem = Connection.GetRoleById(user["UserRoleId"]);

textBoxUnderline2.BackColor = Color.DarkGoldenrod;

this.user = user;

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!(char.IsLetterOrDigit(e.KeyChar) ||

char.IsPunctuation(e.KeyChar) ||

char.IsSeparator(e.KeyChar) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

}

private void FillCombos()

{

comboBox1.DataSource = Connection.GetWorkers();

comboBox2.DataSource = Connection.GetRoles();

}

private void SwitchButton()

{

button1.Enabled = label3.BackColor == Color.DarkGreen &&

label2.BackColor == Color.DarkGreen &&

label11.BackColor == Color.DarkGreen &&

(textBoxUnderline2.BackColor == Color.DarkGreen ||

textBoxUnderline2.BackColor == Color.DarkGoldenrod);

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.Length > 0)

{

label3.BackColor = Color.DarkGreen;

}

else

{

label3.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void comboBox2\_SelectedIndexChanged(object sender, EventArgs e)

{

if (comboBox2.SelectedIndex >= 0)

{

label11.BackColor = Color.DarkGreen;

}

else

{

label11.BackColor = Color.DarkRed;

}

SwitchButton();

}

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

{

if (comboBox1.SelectedIndex >= 0)

{

label2.BackColor = Color.DarkGreen;

}

else

{

label2.BackColor = Color.DarkRed;

}

SwitchButton();

}

private void textBox3\_TextChanged(object sender, EventArgs e)

{

if (textBox3.Text.Length > 0)

{

textBoxUnderline2.BackColor = Color.DarkGreen;

}

else

{

if (user == null)

{

textBoxUnderline2.BackColor = Color.DarkRed;

}

else

{

textBoxUnderline2.BackColor = Color.DarkGoldenrod;

}

}

SwitchButton();

}

private void hidePasswordButton1\_Click(object sender, EventArgs e)

{

textBox3.UseSystemPasswordChar = !textBox3.UseSystemPasswordChar;

}

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

{

var entitySelected = user != null;

if (user == null)

{

user = new Dictionary<string, string>();

}

user["UserLogin"] = textBox2.Text;

if (textBoxUnderline2.BackColor != Color.DarkGoldenrod)

{

var a = MD5.Create().ComputeHash(Encoding.UTF8.GetBytes(textBox3.Text));

var password = BitConverter.ToString(a).Replace("-", string.Empty).ToLower();

user["UserPassword"] = password;

}

user["UserRoleId"] = Connection.GetRoleId(comboBox2.SelectedValue.ToString());

user["UserWorkerId"] = ((KeyValuePair<string, string>)comboBox1.SelectedItem).Key;

if (entitySelected)

{

//update

if (MessageBox.Show("Редактировать запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.UpdateObject("user", user))

{

MessageBox.Show("Запись редактирована", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось редактировать запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

else

{

//insert

if (MessageBox.Show("Добавить запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.InsertObject("user", user))

{

MessageBox.Show("Запись добавлена", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось добавить запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

private void comboBox2\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0)

return;

using (var brush = new SolidBrush(e.ForeColor))

{

e.DrawBackground();

e.Graphics.DrawString(comboBox2.Items[e.Index].ToString(), e.Font, brush, e.Bounds);

e.DrawFocusRectangle();

}

}

private void comboBox1\_DrawItem(object sender, DrawItemEventArgs e)

{

if (e.Index < 0)

return;

using (var brush = new SolidBrush(e.ForeColor))

{

e.DrawBackground();

e.Graphics.DrawString(((KeyValuePair<string, string>)comboBox1.Items[e.Index]).Value.ToString(), e.Font, brush, e.Bounds);

e.DrawFocusRectangle();

}

}

}

}

**WorkerForm.cs** – форма работника

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 Course

{

public partial class WorkerForm : Form

{

private Dictionary<string, string> worker = null;

public WorkerForm()

{

InitializeComponent();

}

public WorkerForm(Dictionary<string, string> worker)

{

InitializeComponent();

textBox1.Text = worker["WorkerSurname"];

textBox2.Text = worker["WorkerName"];

textBox3.Text = worker["WorkerPatronymic"];

maskedTextBox1.Text = worker["WorkerPhone"].Substring(1);

this.worker = worker;

}

private void textBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("ёйцукенгшщзхъфывапролджэячсмитьбю- ".Contains(e.KeyChar.ToString().ToLower()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

if ("ёйцукенгшщзхъфывапролджэячсмитьбю".Contains(e.KeyChar.ToString().ToLower()) && textBox1.Text.Length == 0)

{

e.KeyChar = e.KeyChar.ToString().ToUpper()[0];

}

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("ёйцукенгшщзхъфывапролджэячсмитьбю- ".Contains(e.KeyChar.ToString().ToLower()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

if ("ёйцукенгшщзхъфывапролджэячсмитьбю".Contains(e.KeyChar.ToString().ToLower()) && textBox1.Text.Length == 0)

{

e.KeyChar = e.KeyChar.ToString().ToUpper()[0];

}

}

private void textBox3\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("ёйцукенгшщзхъфывапролджэячсмитьбю- ".Contains(e.KeyChar.ToString().ToLower()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

if ("ёйцукенгшщзхъфывапролджэячсмитьбю".Contains(e.KeyChar.ToString().ToLower()) && textBox1.Text.Length == 0)

{

e.KeyChar = e.KeyChar.ToString().ToUpper()[0];

}

}

private void maskedTextBox1\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!("1234567890".Contains(e.KeyChar.ToString()) ||

e.KeyChar == (char)Keys.Back ||

e.KeyChar == (char)Keys.Delete))

{

e.Handled = true;

return;

}

}

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

{

if (textBox1.Text.Length > 0)

{

textBoxUnderline1.BackColor = Color.DarkGreen;

}

else

{

textBoxUnderline1.BackColor = Color.DarkRed;

}

button1.Enabled = textBoxUnderline1.BackColor == Color.DarkGreen && label3.BackColor == Color.DarkGreen && label7.BackColor == Color.DarkGreen;

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox2.Text.Length > 0)

{

label3.BackColor = Color.DarkGreen;

}

else

{

label3.BackColor = Color.DarkRed;

}

button1.Enabled = textBoxUnderline1.BackColor == Color.DarkGreen && label3.BackColor == Color.DarkGreen && label7.BackColor == Color.DarkGreen;

}

private void maskedTextBox1\_TextChanged(object sender, EventArgs e)

{

if (maskedTextBox1.MaskCompleted)

{

label7.BackColor = Color.DarkGreen;

}

else

{

label7.BackColor = Color.DarkRed;

}

button1.Enabled = textBoxUnderline1.BackColor == Color.DarkGreen && label3.BackColor == Color.DarkGreen && label7.BackColor == Color.DarkGreen;

}

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

{

var workerSelected = worker != null;

if (worker == null)

{

worker = new Dictionary<string, string>();

}

worker["WorkerSurname"] = textBox1.Text;

worker["WorkerName"] = textBox2.Text;

worker["WorkerPatronymic"] = textBox3.Text;

worker["WorkerPhone"] = maskedTextBox1.Text

.Replace("+7", "8")

.Replace("(", "")

.Replace(")", "")

.Replace("-", "");

if (workerSelected)

{

//update

if (MessageBox.Show("Редактировать запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.UpdateObject("worker", worker))

{

MessageBox.Show("Запись редактирована", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось редактировать запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

else

{

//insert

if (MessageBox.Show("Добавить запись?", "", MessageBoxButtons.OKCancel, MessageBoxIcon.Asterisk) != DialogResult.OK)

{

return;

}

if (Connection.InsertObject("worker", worker))

{

MessageBox.Show("Запись добавлена", "", MessageBoxButtons.OK, MessageBoxIcon.Asterisk);

this.DialogResult = DialogResult.OK;

this.Close();

}

else

{

MessageBox.Show("Не удалось добавить запись", "", MessageBoxButtons.OK, MessageBoxIcon.Exclamation);

}

}

}

}

}