**Приложение В**

**Министерство образования и науки Российской федерации**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Саратовский государственный технический университет имени Гагарина Ю.А.»**

**Энгельсский технологический институт (филиал)**

**ЛИСТИНГ ПРОГРАММНОГО КОДА**

Энгельс 2025 г.

using CinemaApplication.Model;

using CinemaApplication.Repository;

using CinemaApplication.StaticModel;

namespace CinemaApplication;

public partial class FormLogin : Form

{

private UserRepository userRepository { get; set; } = new();

private int failedAttempts = 0;

private DateTime lastFailedAttemptTime = DateTime.MinValue;

private System.Windows.Forms.Timer lockoutTimer = new System.Windows.Forms.Timer();

public FormLogin()

{

InitializeComponent();

lockoutTimer.Interval = 30000; // 30 секунд

lockoutTimer.Tick += LockoutTimer\_Tick;

}

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

{

guna2CheckBox1.Checked = true;

}

private void guna2CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

guna2TextBox5.UseSystemPasswordChar = guna2CheckBox1.Checked;

}

private void guna2Button8\_Click(object sender, EventArgs e)

{

if (string.IsNullOrEmpty(guna2TextBox5.Text) || string.IsNullOrEmpty(guna2TextBox6.Text))

{

MessageBox.Show("Заполните логин и пароль.", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (lockoutTimer.Enabled)

{

MessageBox.Show("Пожалуйста, подождите 30 секунд перед следующей попыткой.", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

UserModel user = new()

{

Login = guna2TextBox6.Text,

Password = guna2TextBox5.Text

};

if (userRepository.LoginUser(user))

{

MessageBox.Show($"Добро пожаловать {CurrentUser.Role}, {CurrentUser.FullName}!", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormMain formMain = new();

formMain.Show();

this.Hide();

}

else

{

failedAttempts++;

lastFailedAttemptTime = DateTime.Now;

if (failedAttempts >= 3 && (DateTime.Now - lastFailedAttemptTime).TotalSeconds <= 10)

{

lockoutTimer.Start();

MessageBox.Show("Слишком много неудачных попыток. Пожалуйста, подождите 30 секунд.", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

MessageBox.Show("Логин и/или пароль введены не верно.", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

private void LockoutTimer\_Tick(object sender, EventArgs e)

{

lockoutTimer.Stop();

failedAttempts = 0;

lastFailedAttemptTime = DateTime.MinValue;

}

private void label2\_Click(object sender, EventArgs e)

{

FormRegistration formRegistration = new();

formRegistration.Show();

this.Hide();

}

}

using CinemaApplication.Model;

using CinemaApplication.Repository;

using CinemaApplication.Validator;

namespace CinemaApplication

{

public partial class FormRegistration : Form

{

private ClientRepository ClientRepository { get; set; } = new();

private ClientValidator ClientValidator { get; set; } = new();

private UserRepository UserRepository { get; set; } = new();

private UserValidator UserValidator { get; set; } = new();

public FormRegistration()

{

InitializeComponent();

}

private void guna2Button8\_Click(object sender, EventArgs e)

{

ClientModel client = new()

{

FirstName = guna2TextBox2.Text,

LastName = guna2TextBox1.Text,

MiddleName = guna2TextBox4.Text,

DateBirthday = guna2DateTimePicker1.Value.Date,

PhoneNumber = guna2TextBox3.Text

};

if (!ClientValidator.Validate(client))

{

return;

}

UserModel user = new()

{

Login = guna2TextBox6.Text,

Password = guna2TextBox5.Text,

RoleId = 3

};

if (!UserValidator.Validate(user))

{

return;

}

if (guna2TextBox7.Text != guna2TextBox5.Text)

{

MessageBox.Show("Пароли не совпадают.", "Внимание", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (!ClientRepository.CreateClient(client))

{

return;

}

if (!UserRepository.CreateUser(user))

{

return;

}

var lastClient = ClientRepository.GetLastClientId();

var lastUser = UserRepository.GetLastUserId();

if (UserRepository.OwnerAccountSet(lastUser, lastClient, 1))

{

MessageBox.Show("Регистрация прошла успешно!", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);

FormLogin formLogin = new();

formLogin.Show();

this.Hide();

}

}

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

{

guna2CheckBox1.Checked = true;

}

private void guna2CheckBox1\_CheckedChanged(object sender, EventArgs e)

{

guna2TextBox5.UseSystemPasswordChar = guna2TextBox7.UseSystemPasswordChar = guna2CheckBox1.Checked;

}

private void label1\_Click(object sender, EventArgs e)

{

FormLogin formLogin = new();

formLogin.Show();

this.Hide();

}

}

}

using CinemaApplication.Model;

using CinemaApplication.Repository;

using CinemaApplication.StaticModel;

using System.Data;

namespace CinemaApplication;

public partial class FormPersonalAccount : Form

{

private ClientRepository ClientRepository { get; set; } = new();

private EmployeeRepository EmployeeRepository { get; set; } = new();

public FormPersonalAccount()

{

InitializeComponent();

}

private void guna2Button8\_Click(object sender, EventArgs e)

{

if (CurrentUser.Role.Equals("клиент", StringComparison.OrdinalIgnoreCase))

{

ClientModel client = new()

{

PhoneNumber = guna2TextBox3.Text,

LastName = guna2TextBox1.Text,

FirstName = guna2TextBox2.Text,

MiddleName = guna2TextBox4.Text,

DateBirthday = guna2DateTimePicker1.Value,

ClientId = (int)CurrentUser.PersonId

};

if (!ClientRepository.UpdateClient(client))

{

return;

}

}

else

{

EmployeeModel employee = new()

{

PhoneNumber = guna2TextBox3.Text,

LastName = guna2TextBox1.Text,

FirstName = guna2TextBox2.Text,

MiddleName = guna2TextBox4.Text,

DateBirthday = guna2DateTimePicker1.Value,

EmployeeId = (int)CurrentUser.PersonId

};

if (!EmployeeRepository.UpdateEmployee(employee))

{

return;

}

}

MessageBox.Show("Персональная информация успешно изменена.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

DataTable dataTable = new();

if (CurrentUser.Role.Equals("клиент", StringComparison.OrdinalIgnoreCase))

{

dataTable = ClientRepository.GetClients();

DataRow[] filteredRows = dataTable.Select($"[ID клиента] = {CurrentUser.PersonId}");

dataTable = filteredRows.CopyToDataTable();

}

else

{

dataTable = EmployeeRepository.GetEmployees();

DataRow[] filteredRows = dataTable.Select($"[ID сотрудника] = {CurrentUser.PersonId}");

dataTable = filteredRows.CopyToDataTable();

}

PopulateFields(dataTable);

}

private void PopulateFields(DataTable dataTable)

{

if (dataTable.Rows.Count > 0)

{

DataRow row = dataTable.Rows[0];

guna2TextBox1.Text = row["Фамилия"].ToString();

guna2TextBox2.Text = row["Имя"].ToString();

guna2TextBox3.Text = row["Номер телефона"].ToString();

guna2TextBox4.Text = row["Отчество"].ToString();

if (DateTime.TryParse(row["Дата рождения"].ToString(), out DateTime birthDate))

{

guna2DateTimePicker1.Value = birthDate;

}

}

else

{

MessageBox.Show("Данные о пользвателе не были получены.", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

}

using CinemaApplication.Helper;

using CinemaApplication.Repository;

namespace CinemaApplication;

public partial class FormStats : Form

{

private StatsRepository statsRepository { get; set; } = new();

private ChartManager chartManager { get; set; } = new();

public FormStats()

{

InitializeComponent();

}

private void guna2Button8\_Click(object sender, EventArgs e)

{

try

{

guna2DataGridView1.DataSource = statsRepository.GetReport(guna2ComboBox1.SelectedIndex);

MessageBox.Show("Информация успешно сформирована!", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch { }

}

private void guna2Button2\_Click(object sender, EventArgs e)

{

if (guna2DataGridView1.RowCount <= 0)

{

MessageBox.Show("Информация для построения статистики отсутствует.", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

cartesianChart1.Visible = true;

List<ChartModel> data = chartManager.GetChartDataFromGrid(guna2DataGridView1);

chartManager.UpdateBarChart(cartesianChart1, data, guna2ComboBox1.SelectedItem.ToString(), guna2DataGridView1.Columns[0].HeaderText.ToString());

}

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

{

cartesianChart1.Visible = false;

}

private void guna2Button1\_Click(object sender, EventArgs e)

{

cartesianChart1.Visible = false;

}

}

using LiveChartsCore;

using LiveChartsCore.Measure;

using LiveChartsCore.SkiaSharpView;

using LiveChartsCore.SkiaSharpView.Painting;

using LiveChartsCore.SkiaSharpView.VisualElements;

using LiveChartsCore.SkiaSharpView.WinForms;

using SkiaSharp;

using LiveChartsAxis = LiveChartsCore.SkiaSharpView.Axis;

namespace CinemaApplication.Helper;

public class ChartModel

{

public string NameCategory { get; set; }

public double ValueCategory { get; set; }

}

public class ChartManager

{

public List<ChartModel> GetChartDataFromGrid(DataGridView grid)

{

return grid.Rows

.Cast<DataGridViewRow>()

.Where(row => !row.IsNewRow && row.Cells[0].Value != null && row.Cells[1].Value != null)

.Select(row => new ChartModel

{

NameCategory = row.Cells[0].Value.ToString(),

ValueCategory = double.TryParse(row.Cells[1].Value.ToString(), out double profit) ? profit : 0

})

.ToList();

}

public void UpdateLineChart(CartesianChart chart, List<ChartModel> data, string nameTitle, string seriesName)

{

ClearChart(chart);

ISeries lineSeries = CreateLineSeries(data, seriesName);

chart.Series = new List<ISeries> { lineSeries };

chart.XAxes = new List<LiveChartsAxis>

{

new LiveChartsAxis

{

Labels = data.Select(d => d.NameCategory).ToArray(),

LabelsRotation = 0

}

};

chart.YAxes = new List<LiveChartsAxis>

{

new LiveChartsAxis

{

}

};

chart.Title = CreateChartTitle(nameTitle);

chart.ZoomMode = ZoomAndPanMode.X;

chart.Update();

MessageBox.Show("Линейный график сформирован.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

public void UpdateBarChart(CartesianChart chart, List<ChartModel> data, string nameTitle, string seriesName)

{

ClearChart(chart);

ISeries barSeries = CreateRowSeries(data, seriesName);

chart.Series = new List<ISeries> { barSeries };

chart.YAxes = new List<LiveChartsAxis>

{

new LiveChartsAxis

{

Labels = data.Select(d => d.NameCategory).ToArray(),

LabelsRotation = 0

}

};

chart.XAxes = new List<LiveChartsAxis>

{

new LiveChartsAxis

{

}

};

chart.Title = CreateChartTitle(nameTitle);

chart.ZoomMode = ZoomAndPanMode.X;

chart.Update();

MessageBox.Show("Столбчатый график сформирован.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

private ISeries CreateLineSeries(List<ChartModel> data, string seriesName)

{

return new LineSeries<double>

{

Values = data.Select(d => d.ValueCategory).ToArray(),

Name = seriesName,

LineSmoothness = 0

};

}

private ISeries CreateRowSeries(List<ChartModel> data, string seriesName)

{

return new RowSeries<double>

{

Values = data.Select(d => d.ValueCategory).ToArray(),

Name = seriesName

};

}

private void ClearChart(CartesianChart chart)

{

chart.Series = new List<ISeries>();

chart.XAxes = new List<LiveChartsAxis>();

chart.YAxes = new List<LiveChartsAxis>();

}

private LabelVisual CreateChartTitle(string nameTitle)

{

return new LabelVisual

{

Text = nameTitle,

TextSize = 25,

Padding = new LiveChartsCore.Drawing.Padding(15),

Paint = new SolidColorPaint(SKColors.DarkSlateGray)

};

}

}

using iTextSharp.text.pdf;

using iTextSharp.text;

using Font = iTextSharp.text.Font;

namespace CinemaApplication.Helper;

public class ExportPDF

{

public void Export(string inputTitle, Guna2DataGridView grid)

{

if (grid == null || grid.Rows.Count <= 0)

{

MessageBox.Show("Данные для экспорта не обнаружены.", "Уведомление", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

SaveFileDialog saveFileDialog = new SaveFileDialog

{

Filter = "PDF Files|\*.pdf",

Title = "Сохранить как PDF файл",

FileName = "Export.pdf"

};

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

{

using (FileStream stream = new(saveFileDialog.FileName, FileMode.Create))

{

Document pdfDoc = new(PageSize.A4, 10f, 10f, 10f, 10f);

PdfWriter writer = PdfWriter.GetInstance(pdfDoc, stream);

pdfDoc.Open();

string fontPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.Fonts), "arial.ttf");

BaseFont baseFont = BaseFont.CreateFont(fontPath, BaseFont.IDENTITY\_H, BaseFont.EMBEDDED);

var titleFont = new Font(baseFont, 18, Font.BOLD);

var font = new Font(baseFont, 12);

pdfDoc.Add(new Paragraph("\n"));

// Добавление заголовка

var title = new Paragraph(inputTitle, titleFont)

{

Alignment = Element.ALIGN\_CENTER

};

pdfDoc.Add(title);

pdfDoc.Add(new Paragraph("\n"));

PdfPTable pdfTable = new(grid.Columns.Count);

pdfTable.WidthPercentage = 100;

foreach (DataGridViewColumn column in grid.Columns)

{

PdfPCell cell = new(new Phrase(column.HeaderText, font));

cell.BackgroundColor = BaseColor.LIGHT\_GRAY;

cell.HorizontalAlignment = Element.ALIGN\_CENTER;

pdfTable.AddCell(cell);

}

foreach (DataGridViewRow row in grid.Rows)

{

foreach (DataGridViewCell cell in row.Cells)

{

pdfTable.AddCell(new Phrase(cell.Value?.ToString() ?? string.Empty, font));

}

}

pdfDoc.Add(pdfTable);

pdfDoc.Close();

writer.Close();

}

MessageBox.Show("Экспорт в PDF успешно завершен!", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}