Приложение А

Министерство образования и науки Нижегородской области

Государственное бюджетное профессиональное

образовательное учреждение

«Заволжский автомоторный техникум»

**Листинг**

Автоматизируемой информационной

системы кадровое агентво на базе платформы .NET

Разработал:

студент группы ИС-21А

Низов Е.А.

г. Заволжье

2024– 2025 уч. год

**Листинг программы**

**AddC.cs - форма создания компании**

public partial class AddC : Form

{

int componys;

string name;

string description;

string adress;

string link;

string phoneNumber;

int flag;

public AddC(int compony =0)

{

InitializeComponent();

componys = compony;

}

// фунция для проверки изменены ли поля

void checkEnableUpdate()

{

if (flag == 1)

{

var count = 0;

if (textBoxAdress.Text != adress)

count++;

if (textBoxDesc.Text != description)

count++;

if (textBoxLink.Text != link)

count++;

if (textBoxName.Text != name)

count++;

if (maskedTextBoxPhoneNumber.Text != phoneNumber)

count++;

if (count > 0)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

}

// фунция для проверки заполнены ли поля

void checkEnable()

{

var count = 0;

if (textBoxDesc.Text.Length > 0)

count++;

if (textBoxName.Text.Length > 0)

count++;

if (textBoxAdress.Text.Length > 0)

count++;

if (maskedTextBoxPhoneNumber.Text.Length >= 18)

count++;

if (count == 4)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

// фунция для проверки изменены ли поля

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

{

if (componys != 0)

{

buttonAddS.Text = "Изменить";

label1.Text = "Редактирование компании";

MySqlConnection con = new MySqlConnection(Connection.con);

con.Open();

string search = $"SELECT \* FROM company WHERE id = {componys};";

MySqlCommand comm = new MySqlCommand(search, con);

MySqlDataReader readerr = comm.ExecuteReader();

while (readerr.Read())

{

textBoxName.Text = readerr[1].ToString();

textBoxDesc.Text = readerr[2].ToString();

maskedTextBoxPhoneNumber.Text = readerr[3].ToString();

textBoxAdress.Text = readerr[4].ToString();

textBoxLink.Text = readerr[5].ToString();

}

name = textBoxName.Text;

adress = textBoxAdress.Text;

link = textBoxLink.Text;

phoneNumber = maskedTextBoxPhoneNumber.Text;

description = textBoxDesc.Text;

flag = 1;

}

}

// при нажатии на кнопку переходит переход на формочку-меню «Комапнании»

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

{

AdminC adminC = new AdminC();

adminC.Show();

this.Close();

}

// при нажатии на кнопку происходит или добавление новой компании или изменение компани

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

{

name = textBoxName.Text;

adress = textBoxAdress.Text;

link = textBoxLink.Text;

phoneNumber = maskedTextBoxPhoneNumber.Text;

description = textBoxDesc.Text;

if (componys == 0)

{

DialogResult result = MessageBox.Show(

"Добавить компанию?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

func.direction($@"INSERT INTO company (company\_name,company\_desceiption,company\_phone\_number,company\_address,companyc\_linq) VALUE ('{name}','{description}','{phoneNumber}','{adress}','{link}')");

textBoxDesc.Clear();

textBoxAdress.Clear();

textBoxLink.Clear();

textBoxName.Clear();

maskedTextBoxPhoneNumber.Clear();

MessageBox.Show("Запись успешно добавлена", "Уведомление");

DialogResult results = MessageBox.Show(

"Создать вакансии?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

int componyId = Convert.ToInt32(func.search($"SELECT id FROM company WHERE company\_name = '{name}' and company\_address = '{adress}' and company\_phone\_number = '{phoneNumber}'"));

AddV addV = new AddV(componyId);

addV.Show();

this.Close();

}

}

}

else

{

DialogResult result = MessageBox.Show(

"Вы действительно хотите изменить компанию?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

string searchIn = $@"UPDATE company SET company\_name = '{name}', company\_desceiption = '{description}', company\_phone\_number = '{phoneNumber}', company\_address = '{adress}', companyc\_linq = '{link}';";

func.direction(searchIn);

MessageBox.Show("Запись успешно изменена", "Уведомление");

SeeVacancy seeVacancy = new SeeVacancy();

seeVacancy.Show();

this.Close();

}

}

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (componys == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (componys == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

private void maskedTextBoxPhoneNumber\_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)

{

if (componys == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (componys == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (componys == 0)

checkEnable();

else

checkEnableUpdate();

}

}

**AddE.cs - форма создания сотрудника**

public partial class AddE : Form

{

string name;

string surname;

string patronomic;

string adress;

string phone;

string pwd;

string login;

int post;

int empoyIds;

int gender;

string photo;

int flag;

Random Random = new Random();

public AddE(int employId)

{

InitializeComponent();

empoyIds = employId;

}

// обработка изменения текстового поля на заполнение и изменение

void checkEnable()

{

var count = 0;

if (textBoxSurname.Text.Length > 0)

count++;

if (textBoxName.Text.Length > 0)

count++;

if (textBoxPatronomic.Text.Length > 0)

count++;

if (textBoxAdress.Text.Length > 0)

count++;

if (maskedTextBoxPhoneNumber.Text.Length >= 18)

count++;

if (textBoxLogin.Text.Length > 0)

count++;

if (textBoxPwd.Text.Length > 0)

count++;

if (comboBoxPost.SelectedIndex != -1)

count++;

if (count == 8)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

void checkEnableUpdate()

{

if (flag == 1)

{

var count = 0;

if (textBoxSurname.Text != surname)

count++;

if (textBoxName.Text != name)

count++;

if (textBoxPatronomic.Text != patronomic)

count++;

if (textBoxAdress.Text != adress)

count++;

if (maskedTextBoxPhoneNumber.Text != phone)

count++;

if (textBoxLogin.Text != login)

count++;

if (textBoxPwd.Text != pwd)

count++;

if (comboBoxPost.SelectedIndex + 1 != post)

count++;

if (count > 0)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

}

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

{

comboBoxPost.Items.Add("Администратор");

comboBoxPost.Items.Add("Менеджер");

if (empoyIds != 0)

{

buttonAddS.Text = "Изменить";

label1.Text = "Редактирование резюме";

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

string find = $"SELECT \* FROM employe WHERE id = {empoyIds};";

MySqlCommand com = new MySqlCommand(find, connection);

MySqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

textBoxSurname.Text = reader[1].ToString();

textBoxName.Text = reader[2].ToString();

textBoxPatronomic.Text = reader[3].ToString();

maskedTextBoxPhoneNumber.Text = reader[4].ToString();

textBoxAdress.Text = reader[5].ToString();

textBoxLogin.Text = reader[6].ToString();

pwd = reader[7].ToString();

post = Convert.ToInt32(reader[8].ToString());

if (post == 1)

{

comboBoxPost.SelectedIndex = 0;

}

else

{

comboBoxPost.SelectedIndex = 1;

}

}

name = textBoxName.Text;

surname = textBoxSurname.Text;

patronomic = textBoxPatronomic.Text;

adress = textBoxAdress.Text;

phone = maskedTextBoxPhoneNumber.Text.ToString();

login = textBoxLogin.Text;

post = comboBoxPost.SelectedIndex;

connection.Close();

flag = 1;

}

}

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

{

if (empoyIds == 0) {

name = textBoxName.Text;

surname = textBoxSurname.Text;

patronomic = textBoxPatronomic.Text;

adress = textBoxAdress.Text;

phone = maskedTextBoxPhoneNumber.Text;

login = textBoxLogin.Text;

pwd = BCrypt.Net.BCrypt.HashPassword(textBoxPwd.Text);

DialogResult result = MessageBox.Show(

"Добавить сотрудника?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

func.insertInto($"INSERT INTO employe (employe\_surname, employe\_name, employe\_partronymic, employe\_phone\_number, employe\_adress, employe\_login, employe\_pwd, employe\_post) VALUES ('{surname}','{name}','{patronomic}','{phone}','{adress}','{login}','{pwd}',{comboBoxPost.SelectedIndex + 1});");

textBoxName.Clear();

textBoxAdress.Clear();

textBoxPatronomic.Clear();

textBoxSurname.Clear();

maskedTextBoxPhoneNumber.Clear();

textBoxLogin.Clear();

textBoxPwd.Clear();

}

}

else

{

string update = $@"UPDATE employe

SET employe\_surname = '{textBoxSurname.Text}', employe\_name = '{textBoxName.Text}', employe\_partronymic = '{textBoxPatronomic.Text}', employe\_phone\_number = '{maskedTextBoxPhoneNumber.Text}', employe\_adress = '{textBoxAdress.Text}', employe\_login = '{textBoxLogin.Text}', employe\_post = {comboBoxPost.SelectedIndex + 1} ";

if (textBoxPwd.Text.Length > 0)

{

string addPwd = $", employe\_pwd='{BCrypt.Net.BCrypt.HashPassword(textBoxPwd.Text)}' ";

update += addPwd;

}

update += $"WHERE id = {empoyIds}";

DialogResult results = MessageBox.Show(

"Вы действительно хотите изменить сотрудника?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

func.direction(update);

MessageBox.Show("Запись успешно изменена", "Уведомление");

SeeEmp seeEmp = new SeeEmp();

seeEmp.Show();

this.Close();

}

}

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

private void maskedTextBoxPhoneNumber\_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (empoyIds == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка нажатия на кнопку при нажатии на которую происходит переход на другую форму

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

{

if (empoyIds == 0)

{

AdminE adminE = new AdminE();

adminE.Show();

this.Close();

}

else

{

SeeEmp seeEmp = new SeeEmp();

seeEmp.Show();

this.Close();

}

}

}

**AddR.cs - форма создания компании**

public partial class AddR : Form

{

int aplicantID;

int resumeId;

string educ;

string salary;

string pers;

string lang;

int prof;

string exp;

int flag;

public AddR(int applicant=0,int resume=0)

{

InitializeComponent();

aplicantID = applicant;

resumeId = resume;

if (resumeId != 0)

{

label1.Text = "Редактировать резюме";

}

}

// обработка заполнения текстового поля

void checkEnable()

{

var count = 0;

if (textBoxEducation.Text.Length > 0)

count++;

if (textBoxSalary.Text.Length > 0)

count++;

if (textBoxPers.Text.Length > 0)

count++;

if (textBoxLang.Text.Length > 0)

count++;

if (comboBoxProfessioin.SelectedIndex != -1)

count++;

if (textBoxExp.Text.Length > 0)

count++;

if (count == 6)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

// обработка изменения текстового поля на заполнение и изменение

void checkEnableUpdate()

{

if (flag == 1)

{

var count = 0;

if (textBoxEducation.Text != educ)

count++;

if (textBoxSalary.Text != salary)

count++;

if (textBoxPers.Text != pers)

count++;

if (textBoxLang.Text != lang)

count++;

if (comboBoxProfessioin.SelectedIndex != prof)

count++;

if (textBoxExp.Text != exp)

count++;

if (count > 0)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

}

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

{

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

string find = $"SELECT name FROM profession;";

MySqlCommand com = new MySqlCommand(find, connection);

MySqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

if (!comboBoxProfessioin.Items.Contains(reader[0].ToString()))

comboBoxProfessioin.Items.Add(reader[0].ToString());

}

connection.Close();

if (resumeId != 0)

{

buttonAddS.Text = "Изменить";

label1.Text = "Редактирование резюме";

MySqlConnection con = new MySqlConnection(Connection.con);

con.Open();

string search = $"SELECT \* FROM resume WHERE id = {resumeId};";

MySqlCommand comm = new MySqlCommand(search, con);

MySqlDataReader readerr = comm.ExecuteReader();

while (readerr.Read())

{

aplicantID = Convert.ToInt32(readerr[1].ToString());

comboBoxProfessioin.SelectedIndex = Convert.ToInt32(readerr[2].ToString()) - 1;

textBoxSalary.Text = readerr[3].ToString();

textBoxEducation.Text = readerr[4].ToString();

textBoxExp.Text = readerr[5].ToString();

textBoxLang.Text = readerr[6].ToString();

textBoxPers.Text = readerr[7].ToString();

}

prof = comboBoxProfessioin.SelectedIndex;

salary = textBoxSalary.Text;

educ = textBoxEducation.Text;

exp = textBoxExp.Text;

lang = textBoxLang.Text;

pers = textBoxPers.Text;

buttonAddS.Text = "Изменить";

flag = 1;

}

}

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

{

if (resumeId == 0)

{

DialogResult result = MessageBox.Show(

"Добавить резюме?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

string searchIn = $"INSERT INTO resume (resume\_applicant, resume\_profession, salary, resume\_education, resume\_work\_experience, resume\_knowledge\_of\_languages, resume\_personal\_qualities) VALUE ('{aplicantID}','{comboBoxProfessioin.SelectedIndex + 1}', '{Convert.ToInt32(textBoxSalary.Text)}', '{textBoxEducation.Text}', '{textBoxExp.Text}', '{textBoxLang.Text}', '{textBoxPers.Text}');";

func.direction(searchIn);

MessageBox.Show("Запись успешно добавлена", "Уведомление");

AdminS adminS = new AdminS();

adminS.Show();

this.Close();

}

}

else

{

DialogResult result = MessageBox.Show(

"Вы действительно хотите изменить резюме?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

string searchIn = $@"UPDATE resume SET resume\_profession = '{comboBoxProfessioin.SelectedIndex + 1}', salary = '{Convert.ToInt32(textBoxSalary.Text)}', resume\_education = '{textBoxEducation.Text}', resume\_work\_experience = '{textBoxExp.Text}', resume\_knowledge\_of\_languages = '{textBoxLang.Text}', resume\_personal\_qualities = '{textBoxPers.Text}';";

func.direction(searchIn);

MessageBox.Show("Запись успешно изменена", "Уведомление");

SeeResume seeResume = new SeeResume();

seeResume.Show();

this.Close();

}

}

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (resumeId == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (resumeId == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (resumeId == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (resumeId == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (resumeId == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка изменения текстового поля на заполнение и изменение

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

{

if (resumeId == 0)

checkEnable();

else

checkEnableUpdate();

}

// обработка нажатия на кнопку при нажатии на которую происходит переход на другую форму

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

{

if(resumeId == 0)

{

SeeS see = new SeeS(true);

see.Show();

this.Close();

}

else

{

SeeResume seeResume = new SeeResume();

seeResume.Show();

this.Close();

}

}

}

**AddS.cs – форма добавления соискателя**

public partial class AddS : Form

{

string name;

string surname;

string patronomic;

string adress;

string phone;

string image;

string date;

int gender;

string photo;

int empIds;

string bdPhoto;

int flag;

int aplicantIds;

Random Random = new Random();

public AddS(int aplicantId)

{

InitializeComponent();

aplicantIds = aplicantId;

}

void checkEnable()

{

var count = 0;

if (textBoxSurname.Text.Length > 0)

count++;

if (textBoxName.Text.Length > 0)

count++;

if (textBoxPatronomic.Text.Length > 0)

count++;

if (textBoxAdress.Text.Length > 0)

count++;

if (maskedTextBoxPhoneNumber.Text.Length >= 18)

count++;

if (count == 5)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

void checkEnableUpdate()

{

if (flag == 1)

{

var count = 0;

if (textBoxSurname.Text != surname)

count++;

if (textBoxName.Text != name)

count++;

if (textBoxPatronomic.Text != patronomic)

count++;

if (textBoxAdress.Text != adress)

count++;

if (maskedTextBoxPhoneNumber.Text != phone)

count++;

if (bdPhoto != pictureBox1.ToString())

count++;

if (count > 0)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

}

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

{

if (aplicantIds == 0)

{

name = textBoxName.Text;

surname = textBoxSurname.Text;

patronomic = textBoxPatronomic.Text;

adress = textBoxAdress.Text;

phone = maskedTextBoxPhoneNumber.Text;

date = dateTimePicker1.Value.ToString("yyyy-MM-dd");

if (comboBoxGender.SelectedIndex == 0)

gender = 1;

else

gender = 2;

DialogResult result = MessageBox.Show(

"Добавить соискателя?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

func.insertInto($"INSERT INTO applicant (applicant\_surname, applicant\_name, applicant\_patronymic, applicant\_phone\_number, applicant\_address, applicant\_date\_of\_birth, applicant\_image, applicant\_gender) VALUES ('{surname}','{name}','{patronomic}','{phone}','{adress}','{date}','{photo}',{gender});");

textBoxName.Clear();

textBoxAdress.Clear();

textBoxPatronomic.Clear();

textBoxSurname.Clear();

maskedTextBoxPhoneNumber.Clear();

dateTimePicker1.Value = new DateTime(2006, 01, 01);

comboBoxGender.SelectedIndex = 0;

pictureBox1.Image = Image.FromFile($@"..\..\photo\default\_user.png");

MessageBox.Show("Запись успешно добавлена", "Уведомление");

DialogResult results = MessageBox.Show(

"Создать резюме?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

int applicantId = Convert.ToInt32(func.search($"SELECT applicant\_id FROM applicant WHERE (applicant\_surname = '{surname}' and applicant\_name = '{name}' and applicant\_patronymic = '{patronomic}' and applicant\_address = '{adress}' and applicant\_phone\_number = '{phone}')"));

AddR addR = new AddR(applicantId);

addR.Show();

this.Close();

}

}

}

else

{

DialogResult results = MessageBox.Show(

"Вы действительно хотите изменить соискателя?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

func.direction($@"UPDATE applicant

SET applicant\_surname = '{textBoxSurname.Text}', applicant\_name = '{textBoxName.Text}', applicant\_patronymic = '{textBoxPatronomic.Text}', applicant\_phone\_number = '{maskedTextBoxPhoneNumber.Text}', applicant\_address = '{textBoxAdress.Text}', applicant\_date\_of\_birth = '{dateTimePicker1.Value.ToString("yyyy-MM-dd")}', applicant\_image = '{photo}', applicant\_gender = '{comboBoxGender.SelectedIndex + 1}'

WHERE applicant\_id = {aplicantIds}");

flag = 0;

MessageBox.Show("Запись успешно изменена", "Уведомление");

SeeS seeS = new SeeS();

seeS.Show();

this.Close();

}

}

}

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

{

openFileDialog1.InitialDirectory = "C:\\";

openFileDialog1.Filter = "Image files|\*.bmp;\*.jpg;\*.gif;\*.png;\*.tif|All files|\*.\*";

if (openFileDialog1.ShowDialog() == DialogResult.Cancel)

return;

string filePath = openFileDialog1.FileName;

FileInfo fileInfo = new FileInfo(filePath);

if (fileInfo.Length < (5 \* 8 \* 1024 \* 1024))

{

string[] path = filePath.Split('\\');

string[] namePhoto = path[path.Length - 1].Split('.');

DateTime dateTime = new DateTime();

string photoName = BCrypt.Net.BCrypt.EnhancedHashPassword(dateTime.ToLongTimeString());

photoName= photoName.Replace('.', 't');

photoName= photoName.Replace('\\', 't');

photoName= photoName.Replace('/', 't');

photo = photoName + "." + namePhoto[namePhoto.Length-1];

string newFilePath = $@"\photo\{photo}";

fileInfo.CopyTo($@"..\.." + $"{newFilePath}", true);

pictureBox1.Image = Image.FromFile($@"..\..\" + $"{newFilePath}");

pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

}

else

{

MessageBox.Show("Привышен размер фотографии, выберите другую фотографию");

photo = "";

}

}

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

{

comboBoxGender.Items.Add("Мужской");

comboBoxGender.Items.Add("Женский");

if (aplicantIds == 0)

{

comboBoxGender.SelectedIndex = 0;

pictureBox1.Image = Image.FromFile($@"..\..\photo\default\_user.png");

pictureBox1.SizeMode = PictureBoxSizeMode.Zoom;

}

else

{

buttonAddS.Text = "Изменить";

label1.Text = "Редактирование соискателя";

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

string find = $"SELECT \* FROM applicant WHERE applicant\_id = {aplicantIds};";

MySqlCommand com = new MySqlCommand(find, connection);

MySqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

textBoxSurname.Text = reader[1].ToString();

textBoxName.Text = reader[2].ToString();

textBoxPatronomic.Text = reader[3].ToString();

maskedTextBoxPhoneNumber.Text = reader[4].ToString();

textBoxAdress.Text = reader[5].ToString();

dateTimePicker1.Text = reader[6].ToString();

bdPhoto = reader[7].ToString();

if (reader[7].ToString().Length > 0)

pictureBox1.Image = Image.FromFile($@"..\..\photo\{reader[7].ToString()}.png");

else

pictureBox1.Image = Image.FromFile($@"..\..\photo\default\_user.png");

if (reader[8].ToString() == "1")

comboBoxGender.SelectedIndex = 0;

else

comboBoxGender.SelectedIndex = 1;

}

name = textBoxName.Text;

surname = textBoxSurname.Text;

patronomic = textBoxPatronomic.Text;

adress = textBoxAdress.Text;

date = dateTimePicker1.Value.ToString();

phone = maskedTextBoxPhoneNumber.Text.ToString();

connection.Close();

flag = 1;

}

}

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

{

func.keyPressRu(e);

}

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

{

func.keyPressRu(e);

}

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

{

func.keyPressRu(e);

}

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

{

if (aplicantIds == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (aplicantIds == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (aplicantIds == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (aplicantIds == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (aplicantIds == 0)

{

AdminS adminS = new AdminS();

adminS.Show();

this.Close();

}

else

{

SeeS seeS = new SeeS();

seeS.Show();

this.Close();

}

}

private void maskedTextBoxPhoneNumber\_MaskInputRejected(object sender, MaskInputRejectedEventArgs e)

{

if (aplicantIds == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (aplicantIds == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (aplicantIds == 0)

checkEnable();

else

checkEnableUpdate();

}

}

**AddV.cs - форма создания компании**

public partial class AddV : Form

{

int companyIDS;

int vacancyIDS;

string responsibilities;

string requerements;

string conditions;

string adress;

string salary\_by;

string salary\_before;

int profession;

int flag;

public AddV(int companyID=0, int vacancyID = 0)

{

InitializeComponent();

companyIDS = companyID;

vacancyIDS = vacancyID;

}

void checkEnableUpdate()

{

if (flag == 1)

{

var count = 0;

if (textBoxAdress.Text != adress)

count++;

if (textBoxCond.Text != conditions)

count++;

if (textBoxTreb.Text != requerements)

count++;

if (textBoxObz.Text != responsibilities)

count++;

if (comboBoxProfessioin.SelectedIndex != profession)

count++;

if (textBoxSalaryBy.Text != salary\_by)

count++;

if (textBoxSalaryBefore.Text != salary\_by)

count++;

if (count > 0)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

}

void checkEnable()

{

var count = 0;

if (textBoxAdress.Text.Length > 0)

count++;

if (textBoxCond.Text.Length > 0)

count++;

if (textBoxObz.Text.Length > 0)

count++;

if (textBoxSalaryBefore.Text.Length > 0)

count++;

if (textBoxSalaryBy.Text.Length > 0)

count++;

if (textBoxTreb.Text.Length > 0)

count++;

if (comboBoxProfessioin.SelectedIndex != -1)

count++;

if (count == 7)

{

buttonAddS.Enabled = true;

}

else

{

buttonAddS.Enabled = false;

}

}

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

{

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

string find = $"SELECT name FROM profession;";

MySqlCommand com = new MySqlCommand(find, connection);

MySqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

if (!comboBoxProfessioin.Items.Contains(reader[0].ToString()))

comboBoxProfessioin.Items.Add(reader[0].ToString());

}

connection.Close();

if (vacancyIDS != 0)

{

buttonAddS.Text = "Изменить";

label1.Text = "Редактирование вакансии";

MySqlConnection con = new MySqlConnection(Connection.con);

con.Open();

string search = $"SELECT \* FROM vacancy WHERE id = {vacancyIDS};";

MySqlCommand comm = new MySqlCommand(search, con);

MySqlDataReader readerr = comm.ExecuteReader();

while (readerr.Read())

{

comboBoxProfessioin.SelectedIndex = Convert.ToInt32(readerr[2].ToString()) - 1;

profession = Convert.ToInt32(readerr[2].ToString()) - 1;

textBoxTreb.Text = readerr[3].ToString();

textBoxObz.Text = readerr[4].ToString();

textBoxCond.Text = readerr[5].ToString();

textBoxAdress.Text = readerr[6].ToString();

textBoxSalaryBy.Text = readerr[7].ToString();

textBoxSalaryBefore.Text = readerr[8].ToString();

}

responsibilities = textBoxObz.Text.ToString();

requerements = textBoxTreb.Text.ToString();

conditions=textBoxCond.Text.ToString();

adress=textBoxAdress.Text.ToString();

salary\_before = textBoxSalaryBefore.Text.ToString();

salary\_by = textBoxSalaryBy.Text.ToString();

flag = 1;

}

}

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

{

if (vacancyIDS == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (vacancyIDS == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (vacancyIDS == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (vacancyIDS == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (vacancyIDS == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (vacancyIDS == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (vacancyIDS == 0)

checkEnable();

else

checkEnableUpdate();

}

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

{

if (vacancyIDS == 0)

{

AdminC adminC = new AdminC();

adminC.Show();

this.Close();

}

else

{

SeeVacancy seeVacancy = new SeeVacancy();

seeVacancy.Show();

this.Close();

}

}

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

{

if (vacancyIDS == 0)

{

DialogResult result = MessageBox.Show(

"Добавить вакансию?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

func.direction($@"INSERT INTO vacancy (vacancy\_company,vacancy\_profession,vacancy\_responsibilities,vacancy\_requirements,vacancy\_conditions,vacancy\_address,vacancy\_salary\_by,vacancy\_salary\_before ) VALUE ('{companyIDS}','{comboBoxProfessioin.SelectedIndex + 1}','{textBoxObz.Text}','{textBoxTreb.Text}','{textBoxCond.Text}', '{textBoxAdress.Text}', '{textBoxSalaryBy.Text}', '{textBoxSalaryBefore.Text}' )");

textBoxCond.Clear();

textBoxAdress.Clear();

textBoxObz.Clear();

textBoxSalaryBefore.Clear();

textBoxSalaryBy.Clear();

textBoxTreb.Clear();

comboBoxProfessioin.SelectedIndex = -1;

MessageBox.Show("Запись успешно добавлена", "Уведомление");

}

}

else

{

DialogResult result = MessageBox.Show(

"Вы действительно хотите изменить вакансию?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

responsibilities = textBoxObz.Text.ToString();

requerements = textBoxTreb.Text.ToString();

conditions = textBoxCond.Text.ToString();

adress = textBoxAdress.Text.ToString();

salary\_before = textBoxSalaryBefore.Text.ToString();

salary\_by = textBoxSalaryBy.Text.ToString();

string searchIn = $@"UPDATE vacancy SET vacancy\_profession = '{comboBoxProfessioin.SelectedIndex + 1}', vacancy\_responsibilities = '{responsibilities}', vacancy\_requirements = '{requerements}', vacancy\_conditions = '{conditions}', vacancy\_address = '{adress}', vacancy\_salary\_by = '{salary\_by}', vacancy\_salary\_before = '{salary\_before}' WHERE id = {vacancyIDS};";

func.direction(searchIn);

MessageBox.Show("Запись успешно изменена", "Уведомление");

SeeVacancy seeVacancy = new SeeVacancy();

seeVacancy.Show();

this.Close();

}

}

}

}

**AdminC.cs – форма-меню «Компании»**

public partial class AdminC : Form

{

public AdminC()

{

InitializeComponent();

}

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

{

MenuAdmin menuAdmin = new MenuAdmin();

menuAdmin.Show();

this.Close();

}

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

{

SeeCompany seeCompany = new SeeCompany();

seeCompany.Show();

this.Close();

}

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

{

SeeCompany seeCompany = new SeeCompany(1);

seeCompany.Show();

this.Close();

}

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

{

AddC addC = new AddC();

addC.Show();

this.Close();

}

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

{

}

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

{

SeeVacancy seeVacancy = new SeeVacancy();

seeVacancy.Show();

this.Close();

}

}

**AdminE.cs – форма-меню «Сотрудники»**

public partial class AdminE : Form

{

int empIds;

public AdminE()

{

InitializeComponent();

}

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

{

MenuAdmin menuAdmin = new MenuAdmin();

menuAdmin.Show();

this.Close();

}

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

{

AddE add = new AddE(0);

add.Show();

this.Close();

}

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

{

SeeEmp seeEmp = new SeeEmp();

seeEmp.Show();

this.Close();

}

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

{

}

}

**AdminS.cs – форма-меню «Соискатели»**

public partial class AdminS : Form

{

int empIds;

public AdminS()

{

InitializeComponent();

}

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

{

}

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

{

MenuAdmin menuAdmin = new MenuAdmin();

menuAdmin.Show();

this.Close();

}

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

{

AddS addS = new AddS(0);

addS.Show();

this.Close();

}

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

{

SeeS seeS = new SeeS();

seeS.Show();

this.Close();

}

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

{

}

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

{

SeeS seeS = new SeeS(true);

seeS.Show();

this.Close();

}

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

{

SeeResume see = new SeeResume();

see.Show();

this.Close();

}

}

**DirectionStatus.cs – изменения статуса «Направления»**

public partial class DirectionStatus : Form

{

int dirs;

int applicant;

int vacancy;

int emp;

string e = null;

public DirectionStatus(int dir)

{

dirs = dir;

InitializeComponent();

}

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

{

this.Close();

}

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

{

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

string find = $"SELECT \* FROM direction WHERE id = {dirs};";

MySqlCommand com = new MySqlCommand(find, connection);

MySqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

applicant = Convert.ToInt32(reader[1].ToString());

vacancy = Convert.ToInt32(reader[2].ToString());

emp = Convert.ToInt32(reader[3].ToString());

}

connection.Close();

port.status = comboBox1.SelectedItem.ToString();

func.direction($@"UPDATE direction

SET direction\_status = '{port.status}'

WHERE id = {dirs};");

if(port.status == "Принято")

{

func.direction($@"UPDATE applicant

SET applicant\_delete\_status = 3

WHERE applicant\_id = '{applicant}';");

func.direction($@"UPDATE vacancy

SET vacancy\_delete\_status = 2

WHERE id = '{vacancy}';");

}else if(port.status == "Ожидание")

{

func.direction($@"UPDATE applicant

SET applicant\_delete\_status = '4'

WHERE applicant\_id = '{applicant}';");

func.direction($@"UPDATE vacancy

SET vacancy\_delete\_status = '4'

WHERE id = '{vacancy}';");

}

else if(port.status == "Отклонено")

{

func.direction($@"UPDATE applicant

SET applicant\_delete\_status = '4'

WHERE applicant\_id = '{applicant}';");

func.direction($@"UPDATE vacancy

SET vacancy\_delete\_status = '4'

WHERE id = '{vacancy}';");

}

this.Close();

}

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

{

comboBox1.Items.Add("Ожидание");

comboBox1.Items.Add("Принято");

comboBox1.Items.Add("Отклонено");

if (port.haveStatus == "Ожидание")

comboBox1.SelectedIndex = 0;

else if (port.haveStatus == "Принято")

comboBox1.SelectedIndex = 1;

else if (port.haveStatus == "Отклонено")

comboBox1.SelectedIndex = 2;

}

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

{

enter.Enabled = true;

}

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

{

}

}

**Form1.cs – форма авторизации**

public partial class Auntification : Form

{

string login;

string password;

string loginBD;

string passwordBD;

int empId;

public Auntification()

{

InitializeComponent();

}

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

{

textBoxLogin.Text = "1";

textBoxPwd.Text = "1";

}

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

{

if(string.IsNullOrEmpty(textBoxLogin.Text) || string.IsNullOrEmpty(textBoxPwd.Text))

{

string message = string.IsNullOrEmpty(textBoxLogin.Text) ? "Введите логин" : "Введите пароль";

MessageBox.Show(message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

login = textBoxLogin.Text;

password = textBoxPwd.Text;

passwordBD = func.search($"SELECT employe\_pwd FROM employe WHERE employe\_login = '{login}'");

if (BCrypt.Net.BCrypt.Verify(password, passwordBD))

{

string post = func.search($"SELECT employe\_post FROM employe WHERE employe\_login = '{login}'");

empId = Convert.ToInt32(func.search($"SELECT id FROM employe WHERE employe\_login = '{login}'"));

port.empIds = empId;

if (post == "1")

{

MenuAdmin admin = new MenuAdmin();

admin.Show();

this.Hide();

}

else if (post == "2")

{

MenuManager manager = new MenuManager();

manager.Show();

this.Hide();

}

}

else

{

MessageBox.Show(

"Авторизация не пройдена. Ошибка в логине или пароле.",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Exclamation

);

textBoxLogin.Text = "";

textBoxPwd.Text = "";

}

}

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

{

DialogResult results = MessageBox.Show(

"Вы действительно хотите выйти?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information,

MessageBoxDefaultButton.Button1,

MessageBoxOptions.DefaultDesktopOnly

);

if (results == DialogResult.Yes)

{

Application.Exit();

}

}

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

{

}

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

{

}

private void checkBox1\_CheckedChanged(object sender, EventArgs e)

{

if (checkBox1.Checked)

textBoxPwd.PasswordChar = '\0';

else

textBoxPwd.PasswordChar = '\*';

}

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

{

func.keyPressEn(e);

}

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

{

func.keyPressEn(e);

}

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

{

}

}

**MenuAdmin.cs – главная форма-меню админитсратора**

public partial class MenuAdmin : Form

{

public MenuAdmin()

{

InitializeComponent();

}

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

{

}

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

{

AdminS adminS = new AdminS();

adminS.Show();

this.Close();

}

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

{

Auntification auntification = new Auntification();

auntification.Show();

this.Close();

}

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

{

AdminC adminC = new AdminC();

adminC.Show();

this.Close();

}

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

{

AdminE adminE = new AdminE();

adminE.Show();

this.Close();

}

}

**MenuManager.cs – главная форма-меню менеджера**

public partial class MenuManager : Form

{

public MenuManager()

{

InitializeComponent();

}

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

{

Auntification auntification = new Auntification();

auntification.Show();

this.Close();

}

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

{

SeeVacancy seeVacancy = new SeeVacancy();

seeVacancy.Show();

this.Close();

}

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

{

}

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

{

seeDirection seeDirection = new seeDirection();

seeDirection.Show();

this.Close();

}

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

{

SeeResume seeResume = new SeeResume();

seeResume.Show();

this.Close();

}

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

{

}

}

**res.cs – форма просмотра подробной информации резюме**

public partial class res : Form

{

int idVacancy;

int idResumeID;

int applicantID;

string path;

string searchIn;

string roleEmp;

public res(int vacancyID=0,int resumeID = 0)

{

idVacancy = vacancyID;

idResumeID = resumeID;

InitializeComponent();

roleEmp = func.search($"SELECT employe\_post FROM employe WHERE id = {port.empIds}");

searchIn = $@"SELECT CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель', profession.name as 'Профессия', resume\_education, resume\_work\_experience, resume.resume\_knowledge\_of\_languages as 'Знание языков', resume.resume\_personal\_qualities as 'Личностные качества', resume.salary as 'Зарплата', applicant.applicant\_image, CONCAT(applicant.applicant\_phone\_number, '^',applicant.applicant\_address, '^', applicant.applicant\_date\_of\_birth, '^'), resume.resume\_applicant

FROM resume

INNER JOIN applicant ON resume.resume\_applicant = applicant.applicant\_id

INNER JOIN profession ON resume.resume\_profession = profession.id

WHERE resume.id = {idResumeID}";

if (roleEmp == "2")

{

if (idVacancy > 0)

{

button1.Visible = true;

}

else

{

button2.Visible = true;

}

}

}

//SELECT direction.\*, profession.name FROM agent.direction

//inner join vacancy on direction\_vacancy = vacancy.id

//inner join profession on vacancy\_profession = profession.id;

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

{

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

MySqlCommand command = new MySqlCommand(searchIn, connection);

MySqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{

labelFIO.Text = reader[0].ToString();

labelProfession.Text = reader[1].ToString();

labelEducate.Text = reader[2].ToString();

labelWorkExp.Text = reader[3].ToString().Replace(": ","\n");

labelLang.Text = reader[4].ToString().Replace(", ", "\n");

labelQual.Text = reader[5].ToString().Replace(", ", "\n");

labelSalary.Text = reader[6].ToString()+" рублей";

labelPesonal.Text = reader[8].ToString().Replace("^", "\n");

path = reader[7].ToString();

if (path == "")

{

path = "default\_user.png";

}

pictureBox2.Image = Image.FromFile($@"..\..\photo\{path}");

applicantID = Convert.ToInt32(reader[9].ToString());

}

}

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

{

DialogResult result = MessageBox.Show(

"Создать направление с этим резюме?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

DateTime now = DateTime.Now;

func.direction($"INSERT INTO direction(direction\_aplicant,direction\_vacancy,direction\_employee,direction\_date,direction\_status) VALUES ('{applicantID}','{idVacancy}','{port.empIds}','{now.ToString("yyyy-MM-dd")}','Ожидание');");

MessageBox.Show(

"Направление успешно создано",

"Уведомление"

);

SeeVacancy seeVacancy = new SeeVacancy();

seeVacancy.Show();

this.Close();

}

}

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

{

if (roleEmp == "2")

{

if (idVacancy > 0)

{

SeeResume seeResume = new SeeResume( labelProfession.Text, idVacancy);

seeResume.Show();

this.Close();

}

else

{

SeeResume seeResume = new SeeResume();

seeResume.Show();

this.Close();

}

}

else

{

AdminS adminS = new AdminS();

adminS.Show();

this.Close();

}

}

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

{

SeeVacancy seeVacancy = new SeeVacancy(idResumeID,labelProfession.Text);

seeVacancy.Show();

this.Close();

}

}

**SeeCompany.cs – форма просмотра компаний**

public partial class SeeCompany : Form

{

int currentRowIndex;

int currentColumnIndex;

string roleEmp;

int pos;

public SeeCompany(int position=0)

{

InitializeComponent();

pos = position;

}

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

{

roleEmp = func.search($"SELECT employe\_post FROM employe WHERE id = {port.empIds}");

labelFIO.Text = func.search($"SELECT CONCAT(employe\_surname, ' ', employe\_name, ' ', employe\_partronymic) FROM employe WHERE id = '{port.empIds}'");

func.load(dataGridView1, "SELECT id, company\_name as 'Название', company\_desceiption as 'Описание', company\_phone\_number as 'Номер для связи', company\_address as 'Адресс расположения', companyc\_linq as 'Cсылка', company\_delete\_status as 'Status' FROM company");

dataGridView1.Columns["id"].Visible = false;

dataGridView1.Columns["Cсылка"].Visible = false;

dataGridView1.Columns["Status"].Visible = false;

dataGridView1.Columns["Название"].Width = 150;

dataGridView1.Columns["Описание"].Width = 400;

dataGridView1.Columns["Номер для связи"].Width = 200;

dataGridView1.Columns["Адресс расположения"].Width = 350;

foreach (DataGridViewRow r in dataGridView1.Rows)

{

if (System.Uri.IsWellFormedUriString(r.Cells["Cсылка"].Value.ToString(), UriKind.Absolute))

{

r.Cells["Название"] = new DataGridViewLinkCell();

DataGridViewLinkCell c = r.Cells["Название"] as DataGridViewLinkCell;

c.LinkColor = Color.Green;

}

}

dataGridView1.Columns[2].DefaultCellStyle.WrapMode = DataGridViewTriState.True;

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.SortMode = DataGridViewColumnSortMode.NotSortable;

}

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1")

row.Visible = false;

}

manager.ResumeBinding();

}

void menu(object sender, MouseEventArgs e)

{

ContextMenu contextMenu = new ContextMenu();

this.currentColumnIndex = dataGridView1.HitTest(e.X, e.Y).ColumnIndex;

this.currentRowIndex = dataGridView1.HitTest(e.X, e.Y).RowIndex;

if (e.Button == MouseButtons.Right)

{

if (pos == 0)

{

contextMenu.MenuItems.Add(new MenuItem("Редактировать компанию", update));

contextMenu.MenuItems.Add(new MenuItem("Удалить компанию", delete));

}

else

{

contextMenu.MenuItems.Add(new MenuItem("Создать вакансию", vaca));

}

if (currentRowIndex >= 0)

{

dataGridView1.Rows[currentRowIndex].Selected = true;

contextMenu.Show(dataGridView1, new Point(e.X, e.Y));

}

else

{

currentRowIndex = 0;

}

}

}

void update(object sender, EventArgs e)

{

int company = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value.ToString());

AddC addC = new AddC(company);

addC.Show();

this.Close();

}

void vaca(object sender, EventArgs e)

{

int company =Convert.ToInt32( dataGridView1.Rows[currentRowIndex].Cells["id"].Value.ToString());

AddV addV = new AddV(company);

addV.Show();

this.Close();

}

void delete(object sender, EventArgs e)

{

string company = dataGridView1.Rows[currentRowIndex].Cells["id"].Value.ToString();

DialogResult results = MessageBox.Show(

"Вы действительно хотите удалить компанию? При удаление компании все вакансии этой компании будут УДАЛЕНЫ!!!!",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

func.direction($@"UPDATE company

SET company\_delete\_status = 1

WHERE id = {company}");

func.direction($@"UPDATE vacancy

SET vacancy\_delete\_status = 1

WHERE vacancy\_company = {company}");

MessageBox.Show("Компания и ее вакансии успешно удалены", "Уведомление");

}

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 1)

{

string link = dataGridView1.Rows[e.RowIndex].Cells["Cсылка"].Value.ToString();

if (link != "")

{

try

{

System.Diagnostics.Process.Start(link);

}

catch (Exception ex)

{

MessageBox.Show("Ссылка указана не верно. Удалите или измените ссылку", "Ошибка");

}

}

}

}

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

{

AdminC adminC = new AdminC();

adminC.Show();

this.Close();

}

private void dataGridView1\_MouseDown(object sender, MouseEventArgs e)

{

menu(sender,e);

}

}

**SeeDirection.cs – форма просмотра направлений**

public partial class seeDirection : Form

{

int currentRowIndex;

int currentColumnIndex;

string stringSearch = $@"SELECT direction.id, CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель',profession.name as 'Профессия' , CONCAT(employe.employe\_surname, ' ', employe.employe\_name, ' ', employe.employe\_partronymic) as 'Работник', direction.direction\_date as 'Дата направления', direction.direction\_status as 'Статус', direction.direction\_delete\_status as 'Delete'

FROM direction

INNER JOIN applicant ON direction.direction\_aplicant = applicant.applicant\_id

INNER JOIN employe ON direction.direction\_employee = employe.id

INNER JOIN vacancy on direction\_vacancy = vacancy.id

INNER JOIN profession on vacancy\_profession = profession.id

ORDER BY direction.direction\_date DESC;";

public seeDirection()

{

InitializeComponent();

}

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

{

MenuManager menuManager = new MenuManager();

menuManager.Show();

this.Close();

}

void loadDate()

{

labelFIO.Text = func.search($"SELECT CONCAT(employe\_surname, ' ', employe\_name, ' ', employe\_partronymic) FROM employe WHERE id = '{port.empIds}'");

func.load(dataGridView1, stringSearch);

dataGridView1.Columns["id"].Visible = false;

dataGridView1.Columns["Соискатель"].Width = 270;

dataGridView1.Columns["Профессия"].Width = 250;

dataGridView1.Columns["Работник"].Width = 270;

dataGridView1.Columns["Дата направления"].Width = 100;

dataGridView1.Columns["Статус"].Width = 100;

dataGridView1.Columns["Delete"].Visible = false;

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status\_delete = row.Cells["Delete"].Value.ToString();

string status = row.Cells["Статус"].Value.ToString();

if (status == "Ожидание")

{

row.Cells["Статус"].Style.ForeColor = Color.FromArgb(218, 165, 32);

if (status\_delete == "1")

row.Visible = false;

}

}

}

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

{

loadDate();

}

void menu(object sender, MouseEventArgs e)

{

ContextMenu contextMenu = new ContextMenu();

this.currentColumnIndex = dataGridView1.HitTest(e.X, e.Y).ColumnIndex;

this.currentRowIndex = dataGridView1.HitTest(e.X, e.Y).RowIndex;

if (e.Button == MouseButtons.Right)

{

contextMenu.MenuItems.Add(new MenuItem("Изменить запись", dir));

if (currentRowIndex >= 0)

{

dataGridView1.Rows[currentRowIndex].Selected = true;

contextMenu.Show(dataGridView1, new Point(e.X, e.Y));

}

else

{

currentRowIndex = 0;

dataGridView1.Rows[currentRowIndex].Selected = false;

}

}

}

void dir(object sender, EventArgs e)

{

int dirId = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

string dirStatus = dataGridView1.Rows[currentRowIndex].Cells["Статус"].Value.ToString();

if (dirStatus == "Ожидание")

port.haveStatus = "Ожидание";

else if (dirStatus == "Принято")

port.haveStatus = "Принято";

else if (dirStatus == "Отклонено")

port.haveStatus = "Отклонено";

DirectionStatus status = new DirectionStatus(dirId);

status.ShowDialog();

port.status = "";

loadDate();

}

private void dataGridView1\_MouseDown(object sender, MouseEventArgs e)

{

menu(sender, e);

}

}

**SeeEmp.cs – форма просмотра работников**

public partial class SeeEmp : Form

{

int employId;

int currentRowIndex;

int currentColumnIndex;

public SeeEmp()

{

InitializeComponent();

}

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

{

labelFIO.Text = func.search($"SELECT CONCAT(employe\_surname, ' ', employe\_name, ' ', employe\_partronymic) FROM employe WHERE id = '{port.empIds}'");

load();

dataGridView1.Columns["ФИО"].Width = 270;

dataGridView1.Columns["Номер телефона"].Width = 170;

dataGridView1.Columns["Адрес"].Width = 310;

dataGridView1.Columns["Логин"].Width = 170;

dataGridView1.Columns["Пароль"].Width = 200;

dataGridView1.Columns["Роль"].Width = 140;

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.SortMode = DataGridViewColumnSortMode.NotSortable;

}

}

void load()

{

func.load(dataGridView1, $@"SELECT employe.id, CONCAT(employe.employe\_surname,' ', employe.employe\_name,' ', employe.employe\_partronymic) as 'ФИО', employe.employe\_phone\_number as 'Номер телефона', employe.employe\_adress as 'Адрес', employe.employe\_login as 'Логин', employe.employe\_pwd as 'Пароль', post.posts as 'Роль', employe.employe\_delete\_status as 'Status'

FROM employe

INNER JOIN post ON employe.employe\_post = post.id");

dataGridView1.Columns["id"].Visible = false;

dataGridView1.Columns["Status"].Visible = false;

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1")

row.Visible = false;

}

manager.ResumeBinding();

}

void menu(object sender, MouseEventArgs e)

{

ContextMenu contextMenu = new ContextMenu();

this.currentColumnIndex = dataGridView1.HitTest(e.X, e.Y).ColumnIndex;

this.currentRowIndex = dataGridView1.HitTest(e.X, e.Y).RowIndex;

if (e.Button == MouseButtons.Right)

{

contextMenu.MenuItems.Add(new MenuItem("Редактировать сотрудника", update));

contextMenu.MenuItems.Add(new MenuItem("Удалить сотрудника", delete));

if (currentRowIndex >= 0)

{

dataGridView1.Rows[currentRowIndex].Selected = true;

contextMenu.Show(dataGridView1, new Point(e.X, e.Y));

}

else

{

currentRowIndex = 0;

dataGridView1.Rows[currentRowIndex].Selected = false;

}

}

}

void update(object sender, EventArgs e)

{

employId = Convert.ToInt32( dataGridView1.Rows[currentRowIndex].Cells["id"].Value.ToString());

AddE addE = new AddE(employId);

addE.Show();

this.Close();

}

void delete(object sender, EventArgs e)

{

employId = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value.ToString());

DialogResult results = MessageBox.Show(

"Вы действительно хотите удалить сотрудника?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

func.direction($@"UPDATE employe

SET employe\_delete\_status = 1

WHERE id = {employId}");

load();

}

}

private void dataGridView1\_MouseDown(object sender, MouseEventArgs e)

{

menu(sender, e);

}

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

{

AdminE adminE = new AdminE();

adminE.Show();

this.Close();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

**SeeResume.cs – форма просмотра работников**

public partial class SeeResume : Form

{

int currentRowIndex;

string professions;

int idVacancy;

string searcIn;

string roleEmp;

public SeeResume(string profession = " ", int VacancyID = 0)

{

professions = profession;

InitializeComponent();

idVacancy = VacancyID;

roleEmp = func.search($"SELECT employe\_post FROM employe WHERE id = {port.empIds}");

searcIn = $@"SELECT resume.id, CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель', profession.name as 'Профессия', resume.resume\_knowledge\_of\_languages as 'Знание языков', resume.resume\_personal\_qualities as 'Личностные качества', resume.salary as 'Зарплата', applicant.applicant\_delete\_status as 'Status'

FROM resume

INNER JOIN applicant ON resume.resume\_applicant = applicant.applicant\_id

INNER JOIN profession ON resume.resume\_profession = profession.id ";

if (roleEmp == "2")

{

if (idVacancy >= 0 && professions != " ")

searcIn += $"WHERE profession.name = '{professions}'";

else

{

textBoxSearch.Visible = true;

comboBox1.Visible = true;

comboBox2.Visible = true;

}

}else if (roleEmp == "1")

{

}

}

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

{

labelFIO.Text = func.search($"SELECT CONCAT(employe\_surname, ' ', employe\_name, ' ', employe\_partronymic) FROM employe WHERE id = '{port.empIds}'");

if (idVacancy != 0)

{

exit.Location = new Point(1161, 870);

this.Size = new Size(1391, 1048);

string searcIn = $@"SELECT vacancy.id, company.company\_name as 'Компания', profession.name as 'Профессия', vacancy.vacancy\_responsibilities as 'Обязанности', vacancy.vacancy\_requirements as 'Требования', vacancy.vacancy\_conditions as 'Условия', CONCAT( vacancy.vacancy\_salary\_by, ' - ', vacancy.vacancy\_salary\_before) as 'Размер зарплаты', vacancy.vacancy\_address as 'Адресс работы', vacancy.vacancy\_delete\_status as 'Status' FROM vacancy INNER JOIN company ON vacancy.vacancy\_company = company.id INNER JOIN profession ON vacancy.vacancy\_profession = profession.id WHERE vacancy.id ='{idVacancy}'";

func.load(dataGridView2, searcIn);

dataGridView2.Columns["id"].Visible = false;

dataGridView2.Columns["Status"].Visible = false;

dataGridView2.Columns["Компания"].Width = 100;

dataGridView2.Columns["Профессия"].Width = 140;

dataGridView2.Columns["Обязанности"].Width = 260;

dataGridView2.Columns["Требования"].Width = 200;

dataGridView2.Columns["Условия"].Width = 200;

dataGridView2.Columns["Размер зарплаты"].Width = 110;

dataGridView2.Columns["Адресс работы"].Width = 300;

dataGridView2.Visible = true;

}

comboBox2.Items.Add("Без фильтра");

comboBox1.Items.Add("По возрастанию зарплаты");

comboBox1.Items.Add("По убыванию зарплаты");

List<string> list = new List<string>();

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

string find = $"SELECT name FROM profession;";

MySqlCommand com = new MySqlCommand(find, connection);

MySqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

if (!comboBox2.Items.Contains(reader[0].ToString()))

comboBox2.Items.Add(reader[0].ToString());

}

connection.Close();

func.load(dataGridView1, searcIn);

dataGridView1.Columns["id"].Visible = false;

dataGridView1.Columns["Status"].Visible = false;

dataGridView1.Columns["Соискатель"].Width = 270;

dataGridView1.Columns["Профессия"].Width = 250;

dataGridView1.Columns["Знание языков"].Width = 250;

dataGridView1.Columns["Личностные качества"].Width = 400;

dataGridView1.Columns["Зарплата"].Width = 140;

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.SortMode = DataGridViewColumnSortMode.NotSortable;

}

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1" || status == "3")

row.Visible = false;

}

manager.ResumeBinding();

}

void menu(object sender, MouseEventArgs e)

{

ContextMenu contextMenu = new ContextMenu();

this.currentRowIndex = dataGridView1.HitTest(e.X, e.Y).RowIndex;

if (e.Button == MouseButtons.Right)

{

if (roleEmp == "2")

{

contextMenu.MenuItems.Add(new MenuItem("Посмотреть подробную информацию", click\_to));

}else if (roleEmp == "1")

{

contextMenu.MenuItems.Add(new MenuItem("Редактировать резюме", update));

contextMenu.MenuItems.Add(new MenuItem("Удалить резюме", delete));

}

if (currentRowIndex >= 0)

{

dataGridView1.Rows[currentRowIndex].Selected = true;

contextMenu.Show(dataGridView1, new Point(e.X, e.Y));

}

else

{

currentRowIndex = 0;

}

}

}

string Search()

{

string basis = $@"SELECT resume.id, CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель', profession.name as 'Профессия', resume.resume\_knowledge\_of\_languages as 'Знание языков', resume.resume\_personal\_qualities as 'Личностные качества', resume.salary as 'Зарплата', applicant.applicant\_delete\_status as 'Status' FROM resume INNER JOIN applicant ON resume.resume\_applicant = applicant.applicant\_id INNER JOIN profession ON resume.resume\_profession = profession.id "; ;

if (comboBox2.SelectedIndex != -1 && comboBox2.SelectedIndex != 0 || textBoxSearch.Text.Length > 0)

{

basis += "WHERE ";

}

if (comboBox2.SelectedIndex != -1 && comboBox2.SelectedIndex != 0)

{

basis += $"(profession.name = '{comboBox2.SelectedItem}')";

}

if (textBoxSearch.Text.Length > 0)

{

if (comboBox2.SelectedIndex != -1 && comboBox2.SelectedIndex != 0)

basis += " AND ";

basis += $"(CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) LIKE '%{textBoxSearch.Text}%' OR resume.resume\_personal\_qualities LIKE '%{textBoxSearch.Text}%' OR resume.resume\_knowledge\_of\_languages LIKE '%{textBoxSearch.Text}%')";

}

if (comboBox1.SelectedIndex == 0)

{

basis += $"ORDER BY resume.salary";

}

else if (comboBox1.SelectedIndex == 1)

{

basis += $"ORDER BY resume.salary DESC";

}

return basis;

}

void click\_to(object sender, EventArgs e)

{

int resume = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

res res = new res(idVacancy, resume);

res.Show();

this.Hide();

}

void update(object sender, EventArgs e)

{

int resumes = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

AddR addR = new AddR(0, resumes);

addR.Show();

this.Close();

}

void delete(object sender, EventArgs e)

{

DialogResult results = MessageBox.Show(

"Вы действительно хотите удалить резюме?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

int resume = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

func.direction($"DELETE FROM resume WHERE id ={resume}");

func.load(dataGridView1, searcIn);

}

}

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

{

if (roleEmp == "2")

{

if (idVacancy >= 0 && professions != " ")

{

SeeVacancy seeVacancy = new SeeVacancy();

seeVacancy.Show();

this.Close();

}

else

{

MenuManager menuManager = new MenuManager();

menuManager.Show();

this.Close();

}

}

else

{

AdminS adminS = new AdminS();

adminS.Show();

this.Close();

}

}

private void dataGridView1\_MouseDown(object sender, MouseEventArgs e)

{

menu(sender, e);

}

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

{

func.load(dataGridView1, Search());

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1" || status == "3")

row.Visible = false;

}

manager.ResumeBinding();

}

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

{

func.load(dataGridView1, Search());

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1" || status == "3")

row.Visible = false;

}

manager.ResumeBinding();

}

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

{

func.load(dataGridView1, Search());

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1" || status == "3")

row.Visible = false;

}

manager.ResumeBinding();

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

**SeeS.cs – форма просмотра соискателей**

public partial class SeeS : Form

{

int currentRowIndex;

int currentColumnIndex;

int applicantID;

string searchStering;

bool resume;

public SeeS(bool res = false)

{

InitializeComponent();

resume = res;

if (!resume)

{

searchStering = "SELECT applicant\_id,CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель', applicant.applicant\_phone\_number as 'Номер телефона', applicant.applicant\_address as 'Адресс', applicant.applicant\_date\_of\_birth as 'Дата рождения', applicant.applicant\_image, gender.genders as 'Пол', applicant.applicant\_delete\_status as 'Status'" +

"FROM applicant " +

"INNER JOIN gender ON applicant.applicant\_gender = gender.id";

}

else

{

ladelHeader.Text = "Выберете соискателя для создания резюме";

searchStering = @"SELECT applicant\_id,CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель', applicant.applicant\_phone\_number as 'Номер телефона', applicant.applicant\_address as 'Адресс', applicant.applicant\_date\_of\_birth as 'Дата рождения', applicant.applicant\_image, gender.genders as 'Пол', applicant.applicant\_delete\_status as 'Status' FROM applicant INNER JOIN gender ON applicant.applicant\_gender = gender.id WHERE applicant\_id NOT IN (SELECT resume\_applicant FROM resume);";

}

}

void menu(object sender, MouseEventArgs e)

{

ContextMenu contextMenu = new ContextMenu();

this.currentColumnIndex = dataGridView1.HitTest(e.X, e.Y).ColumnIndex;

this.currentRowIndex = dataGridView1.HitTest(e.X, e.Y).RowIndex;

if (e.Button == MouseButtons.Right)

{

if (resume == false)

{

contextMenu.MenuItems.Add(new MenuItem("Редактировать соискателя", update));

contextMenu.MenuItems.Add(new MenuItem("Удалить соискателя", delete));

}

else

{

contextMenu.MenuItems.Add(new MenuItem("Создать резюме", resum));

}

if (currentRowIndex >= 0)

{

dataGridView1.Rows[currentRowIndex].Selected = true;

contextMenu.Show(dataGridView1, new Point(e.X, e.Y));

}

else

{

currentRowIndex = 0;

dataGridView1.Rows[currentRowIndex].Selected = false;

}

}

}

void update(object sender, EventArgs e)

{

applicantID = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["applicant\_id"].Value.ToString());

AddS addS = new AddS(applicantID);

addS.Show();

this.Close();

}

void resum(object sender, EventArgs e)

{

AddR addR = new AddR(Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["applicant\_id"].Value.ToString()));

addR.Show();

this.Close();

}

void delete(object sender, EventArgs e)

{

applicantID = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["applicant\_id"].Value.ToString());

DialogResult results = MessageBox.Show(

"Вы действительно хотите удалить соискателя?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

func.direction($@"UPDATE applicant

SET applicant\_delete\_status = 1

WHERE applicant\_id = {applicantID}");

func.direction($"DELETE FROM resume WHERE resume\_applicant ={applicantID}");

dataGridView1.Columns.Remove("Изображение");

load();

}

}

void load()

{

func.load(dataGridView1, searchStering);

dataGridView1.Columns["applicant\_id"].Visible = false;

dataGridView1.Columns["Status"].Visible = false;

dataGridView1.Columns["applicant\_image"].Visible = false;

labelFIO.Text = func.search($"SELECT CONCAT(employe\_surname, ' ', employe\_name, ' ', employe\_partronymic) FROM employe WHERE id = '{port.empIds}'");

DataGridViewImageColumn imageColumn = new DataGridViewImageColumn();

imageColumn.Name = "Изображение";

imageColumn.ImageLayout = DataGridViewImageCellLayout.Zoom;

dataGridView1.Columns.Add(imageColumn);

dataGridView1.AllowUserToAddRows = false;

dataGridView1.Columns["Адресс"].Width = 400;

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string path = row.Cells["applicant\_image"].Value.ToString();

if (path == "")

{

path = "default\_user.png";

}

try

{

row.Cells["Изображение"].Value = Image.FromFile($@"..\..\photo\{path}");

}

catch

{

row.Cells["Изображение"].Value = Image.FromFile($@"..\..\photo\default\_user.png");

}

}

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.SortMode = DataGridViewColumnSortMode.NotSortable;

}

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1" || status == "3")

row.Visible = false;

}

manager.ResumeBinding();

}

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

{

load();

dataGridView1.Columns["Соискатель"].Width = 340;

dataGridView1.Columns["Номер телефона"].Width = 200;

dataGridView1.Columns["Адресс"].Width = 400;

dataGridView1.Columns["Дата рождения"].Width = 140;

dataGridView1.Columns["Изображение"].Width = 250;

dataGridView1.Columns["Пол"].Width = 120;

}

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

{

AdminS adminS = new AdminS();

adminS.Show();

this.Close();

}

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

{

}

private void dataGridView1\_MouseDown(object sender, MouseEventArgs e)

{

menu(sender, e);

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

**SeeVacancy.cs – форма просмотра соискателей**

public partial class SeeVacancy : Form

{

int currentRowIndex;

int currentColumnIndex;

string roleEmp;

string searchIn;

int resume;

string profession;

int applicantId;

//Добавить возможность у администратора редактировать и удалять вакансии при нажатии на правую кнопку мыши

public SeeVacancy(int idResume=0, string profession = "")

{

InitializeComponent();

resume = idResume;

roleEmp = func.search($"SELECT employe\_post FROM employe WHERE id = {port.empIds}");

searchIn = $@"SELECT vacancy.id, company.company\_name as 'Комапния', profession.name as 'Профессия', vacancy.vacancy\_responsibilities as 'Обязанности', vacancy.vacancy\_requirements as 'Требования', vacancy.vacancy\_conditions as 'Условия', CONCAT( vacancy.vacancy\_salary\_by, ' - ', vacancy.vacancy\_salary\_before) as 'Размер зарплаты', vacancy.vacancy\_address as 'Адресс работы', vacancy.vacancy\_delete\_status as 'Status', companyc\_linq as 'Cсылка'

FROM vacancy

INNER JOIN company ON vacancy.vacancy\_company = company.id

INNER JOIN profession ON vacancy.vacancy\_profession = profession.id";

if (roleEmp == "2" )

{

if (resume == 0)

{

textBoxSearch.Visible = true;

comboBox1.Visible = true;

comboBox2.Visible = true;

}

else

{

applicantId = Convert.ToInt32(func.search($"SELECT resume\_applicant FROM resume WHERE id = {idResume}"));

searchIn += $" WHERE profession.name = '{profession}' ";

}

}

}

string Search()

{

string basis = "SELECT vacancy.id, company.company\_name as 'Комапния', profession.name as 'Профессия', vacancy.vacancy\_responsibilities as 'Обязанности', vacancy.vacancy\_requirements as 'Требования', vacancy.vacancy\_conditions as 'Условия', CONCAT( vacancy.vacancy\_salary\_by, ' - ', vacancy.vacancy\_salary\_before) as 'Размер зарплаты', vacancy.vacancy\_address as 'Адресс работы', vacancy.vacancy\_delete\_status as 'Status', companyc\_linq as 'Cсылка' " +

"FROM vacancy " +

"INNER JOIN company ON vacancy.vacancy\_company = company.id " +

"INNER JOIN profession ON vacancy.vacancy\_profession = profession.id ";

if ((comboBox2.SelectedIndex != -1 && comboBox2.SelectedIndex != 0 || textBoxSearch.Text.Length > 0)&& resume==0)

{

basis += "WHERE ";

}

if (comboBox2.SelectedIndex != -1 && comboBox2.SelectedIndex != 0)

{

basis += $"(profession.name = '{comboBox2.SelectedItem}')";

}

if (textBoxSearch.Text.Length > 0)

{

if (comboBox2.SelectedIndex != -1 && comboBox2.SelectedIndex != 0)

basis += " AND ";

basis += $"(company.company\_name LIKE '%{textBoxSearch.Text}%' OR vacancy.vacancy\_requirements LIKE '%{textBoxSearch.Text}%' OR vacancy.vacancy\_conditions LIKE '%{textBoxSearch.Text}%' OR vacancy.vacancy\_responsibilities LIKE '%{textBoxSearch.Text}%')";

}

if (comboBox1.SelectedIndex == 0)

{

basis += $"ORDER BY vacancy.vacancy\_salary\_by";

} else if (comboBox1.SelectedIndex == 1)

{

basis += $"ORDER BY vacancy.vacancy\_salary\_by DESC";

}

return basis;

}

void load\_load()

{

func.load(dataGridView1, searchIn);

dataGridView1.Columns["Комапния"].Width = 160;

dataGridView1.Columns["Профессия"].Width = 200;

dataGridView1.Columns["Обязанности"].Width = 340;

dataGridView1.Columns["Требования"].Width = 270;

dataGridView1.Columns["Условия"].Width = 270;

dataGridView1.Columns["Размер зарплаты"].Width = 150;

dataGridView1.Columns["Адресс работы"].Width = 310;

dataGridView1.Columns["id"].Visible = false;

dataGridView1.Columns["Status"].Visible = false;

dataGridView1.Columns["Cсылка"].Visible = false;

foreach (DataGridViewColumn column in dataGridView1.Columns)

{

column.SortMode = DataGridViewColumnSortMode.NotSortable;

}

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1" || status == "2")

row.Visible = false;

}

manager.ResumeBinding();

foreach (DataGridViewRow r in dataGridView1.Rows)

{

if (System.Uri.IsWellFormedUriString(r.Cells["Cсылка"].Value.ToString(), UriKind.Absolute))

{

r.Cells["Комапния"] = new DataGridViewLinkCell();

DataGridViewLinkCell c = r.Cells["Комапния"] as DataGridViewLinkCell;

c.LinkColor = Color.Green;

}

}

}

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

{

if (resume != 0)

{

string searcNow = $@"SELECT resume.id, CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель', profession.name as 'Профессия', resume.resume\_knowledge\_of\_languages as 'Знание языков', resume.resume\_personal\_qualities as 'Личностные качества', resume.salary as 'Зарплата', applicant.applicant\_delete\_status as 'Status'

FROM resume

INNER JOIN applicant ON resume.resume\_applicant = applicant.applicant\_id

INNER JOIN profession ON resume.resume\_profession = profession.id

WHERE resume.id = '{resume}'";

func.load(dataGridView2, searcNow);

dataGridView2.Columns["id"].Visible = false;

dataGridView2.Columns["Status"].Visible = false;

dataGridView2.Columns["Соискатель"].Width = 270;

dataGridView2.Columns["Профессия"].Width = 250;

dataGridView2.Columns["Знание языков"].Width = 208;

dataGridView2.Columns["Личностные качества"].Width = 300;

dataGridView2.Columns["Зарплата"].Width = 140;

dataGridView2.Visible = true;

}

labelFIO.Text = func.search($"SELECT CONCAT(employe\_surname, ' ', employe\_name, ' ', employe\_partronymic) FROM employe WHERE id = '{port.empIds}'");

comboBox2.Items.Add("Без фильтра");

List<string> list = new List<string>();

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

string find = $"SELECT name FROM profession;";

MySqlCommand com = new MySqlCommand(find, connection);

var pwd = "";

MySqlDataReader reader = com.ExecuteReader();

while (reader.Read())

{

if (!comboBox2.Items.Contains(reader[0].ToString()))

comboBox2.Items.Add(reader[0].ToString());

}

connection.Close();

comboBox1.Items.Add("По возрастанию зарплаты");

comboBox1.Items.Add("По убыванию зарплаты");

load\_load();

}

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

{

if (roleEmp == "2")

{

if (resume == 0)

{

MenuManager menuManager = new MenuManager();

menuManager.Show();

this.Close();

}

else

{

res res = new res(0, resume);

res.Show();

this.Close();

}

}

else

{

MenuAdmin menuA = new MenuAdmin();

menuA.Show();

this.Close();

}

}

void change()

{

CurrencyManager manager = (CurrencyManager)BindingContext[dataGridView1.DataSource];

manager.SuspendBinding();

foreach (DataGridViewRow row in dataGridView1.Rows)

{

string status = row.Cells["Status"].Value.ToString();

if (status == "1" || status == "2")

row.Visible = false;

}

manager.ResumeBinding();

foreach (DataGridViewRow r in dataGridView1.Rows)

{

if (System.Uri.IsWellFormedUriString(r.Cells["Cсылка"].Value.ToString(), UriKind.Absolute))

{

r.Cells["Комапния"] = new DataGridViewLinkCell();

DataGridViewLinkCell c = r.Cells["Комапния"] as DataGridViewLinkCell;

c.LinkColor = Color.Green;

}

}

dataGridView1.Columns["Комапния"].Width = 160;

dataGridView1.Columns["Профессия"].Width = 200;

dataGridView1.Columns["Обязанности"].Width = 340;

dataGridView1.Columns["Требования"].Width = 270;

dataGridView1.Columns["Условия"].Width = 270;

dataGridView1.Columns["Размер зарплаты"].Width = 150;

dataGridView1.Columns["Адресс работы"].Width = 310;

dataGridView1.Columns["id"].Visible = false;

dataGridView1.Columns["Status"].Visible = false;

dataGridView1.Columns["Cсылка"].Visible = false;

}

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

{

func.load(dataGridView1, Search());

dataGridView1.ClearSelection();

change();

}

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

{

func.load(dataGridView1, Search());

change();

}

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

{

func.load(dataGridView1, Search());

change();

}

void menu(object sender, MouseEventArgs e)

{

ContextMenu contextMenu = new ContextMenu();

this.currentColumnIndex = dataGridView1.HitTest(e.X, e.Y).ColumnIndex;

this.currentRowIndex = dataGridView1.HitTest(e.X, e.Y).RowIndex;

if(e.Button == MouseButtons.Right)

{

if (roleEmp == "2")

{

if (resume == 0)

contextMenu.MenuItems.Add(new MenuItem("Выбрать соискателя", click\_to));

else

contextMenu.MenuItems.Add(new MenuItem("Создать направление", dir));

}

else

{

contextMenu.MenuItems.Add(new MenuItem("Редактировать вакансию", update));

contextMenu.MenuItems.Add(new MenuItem("Удалить вакансию", delete));

}

if (currentRowIndex >= 0)

{

dataGridView1.Rows[currentRowIndex].Selected = true;

contextMenu.Show(dataGridView1, new Point(e.X, e.Y));

}

else

{

currentRowIndex = 0;

}

}

}

void dir(object sender, EventArgs e)

{

DialogResult result = MessageBox.Show(

"Создать направление с этим резюме?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (result == DialogResult.Yes)

{

string vacancyProfession = dataGridView1.Rows[currentRowIndex].Cells["Профессия"].Value.ToString();

int vacancyID = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

DateTime now = DateTime.Now;

func.direction($"INSERT INTO direction(direction\_aplicant,direction\_vacancy,direction\_employee,direction\_date,direction\_status) VALUES ('{applicantId}','{vacancyID}','{port.empIds}','{now.ToString("yyyy-MM-dd")}','Ожидание');");

MessageBox.Show(

"Направление успешно создано",

"Уведомление"

);

SeeResume resume = new SeeResume();

resume.Show();

this.Close();

}

}

void click\_to(object sender, EventArgs e)

{

string vacancyProfession = dataGridView1.Rows[currentRowIndex].Cells["Профессия"].Value.ToString();

int vacancyID = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

SeeResume seeResume = new SeeResume( vacancyProfession,vacancyID);

seeResume.Show();

this.Hide();

dataGridView1.Rows[currentRowIndex].Selected = false;

}

void update(object sender, EventArgs e)

{

int vacancyID = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

AddV addV = new AddV(0,vacancyID);

addV.Show();

this.Close();

}

void delete(object sender, EventArgs e)

{

int vacancyID = Convert.ToInt32(dataGridView1.Rows[currentRowIndex].Cells["id"].Value);

DialogResult results = MessageBox.Show(

"Вы действительно хотите удалить вакансию?",

"Подтверждение",

MessageBoxButtons.YesNo,

MessageBoxIcon.Information

);

if (results == DialogResult.Yes)

{

func.direction($@"UPDATE vacancy

SET vacancy\_delete\_status = 1

WHERE id = {vacancyID}");

MessageBox.Show("Ваканисия успешно удалена", "Уведомление");

load\_load();

}

}

private void dataGridView1\_MouseDown(object sender, MouseEventArgs e)

{

menu(sender, e);

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

if (e.ColumnIndex == 1)

{

string link = dataGridView1.Rows[e.RowIndex].Cells["Cсылка"].Value.ToString();

if (link != "")

{

try

{

System.Diagnostics.Process.Start(link);

}

catch (Exception ex)

{

MessageBox.Show("Ссылка указана не верно. Удалите или измените ссылку", "Ошибка");

}

}

}

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

}

**func.cs – форма просмотра соискателей**

static public void keyPressRu(KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[а-яА-Я]|[\b]|[-]").Success)

{

e.Handled = true;

}

}

static public void keyPressEn(KeyPressEventArgs e)

{

string Symbol = e.KeyChar.ToString();

if (!Regex.Match(Symbol, @"[a-zA-Z]|[\b]|[-]|[0-9]|[.]|[\_]|[,]|[$]|[@]|[#]").Success)

{

e.Handled = true;

}

}

static public void load(DataGridView gridView, string search)

{

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

MySqlCommand command = new MySqlCommand(search, connection);

command.ExecuteNonQuery();

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

DataTable dataTable = new DataTable();

adapter.Fill(dataTable);

gridView.DataSource = dataTable;

connection.Close();

}

public static string search(string com)

{

MySqlConnection conn = new MySqlConnection(Connection.con);

conn.Open();

MySqlCommand command = new MySqlCommand(com, conn);

command.ExecuteScalar();

var pwd = "$2a$12$FVR0VemGE//V5LB1LeHZsu0h6CmyCVD2px9YMTkH0gsSmY/3KDRfO";

if (command.ExecuteScalar() == null)

{

return pwd;

}

else

{

pwd = command.ExecuteScalar().ToString();

}

conn.Close();

return pwd;

}

public static void direction(string search)

{

MySqlConnection con = new MySqlConnection(Connection.con);

con.Open();

MySqlCommand com = new MySqlCommand(search, con);

com.ExecuteNonQuery();

con.Close();

}

public static void insertInto(string request)

{

MySqlConnection conn = new MySqlConnection(Connection.con);

conn.Open();

MySqlCommand mySqlCommand = new MySqlCommand(request, conn);

int number = mySqlCommand.ExecuteNonQuery();

conn.Close();

}

}  
**word.cs** **– создания отчета**

public partial class word : Form

{

string startDate;

string endDate;

int count;

int cool;

int bad;

int load;

double percent;

public word()

{

InitializeComponent();

}

public System.Data.DataTable returnDate(string query, System.Data.DataTable dataTable)

{

MySqlConnection connection = new MySqlConnection(Connection.con);

connection.Open();

MySqlCommand command = new MySqlCommand(query, connection);

MySqlDataAdapter adapter = new MySqlDataAdapter(command);

adapter.SelectCommand.ExecuteNonQuery();

adapter.Fill(dataTable);

connection.Close();

return dataTable;

}

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

{

dateTimePicker1.MaxDate = DateTime.Now;

dateTimePicker1.Value = DateTime.Now;

dateTimePicker3.Value = DateTime.Now;

dateTimePicker3.MaxDate = DateTime.Now;

count = 0;

cool = 0;

bad = 0;

load = 0;

percent = 0.0;

}

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

{

}

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

{

this.Close();

}

private void exportToWord(System.Data.DataTable dataTable)

{

var wordDoc = new WordApp.Application();

string path = Path.GetDirectoryName(Assembly.GetEntryAssembly().Location);

path += "\\doc.docx";

Document doc =wordDoc.Documents.Open(path, ReadOnly: false);

wordDoc.Visible = true;

doc.Content.End = doc.Content.End; //не обязательно, но для ясности кода

doc.Content.Select(); //Выделение курсора в конце

// Пример добавления текста в конец:

Range r = doc.Content;

if (dataTable.Rows.Count > 0)

{

Range rng = wordDoc.Selection.Range;

rng.Collapse(WdCollapseDirection.wdCollapseEnd); // или wdCollapseStart

r.Collapse(0);

WordApp.Table table = wordDoc.Application.ActiveDocument.Tables.Add(r, dataTable.Rows.Count + 1, dataTable.Columns.Count);

table.Borders.OutsideLineStyle = WordApp.WdLineStyle.wdLineStyleSingle;

table.Borders.InsideLineStyle = WordApp.WdLineStyle.wdLineStyleSingle;

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

{

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

}

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

{

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

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

}

}

doc.Content.InsertAfter("<nowDate>");

var Items = new Dictionary<string, string>

{

{ "<startDate>",startDate},

{ "<endDate>",endDate},

{ "<count>",count.ToString()},

{ "<cool>",cool.ToString()},

{ "<bad>",bad.ToString()},

{ "<percent>",percent.ToString()},

{ "<load>",load.ToString()},

{ "<nowDate>",DateTime.Now.ToString()}

};

wordDoc.Documents.Open(path, ReadOnly: false);

WordApp.Find find = wordDoc.Selection.Find;

foreach (var item in Items)

{

find.Text = item.Key;

find.Replacement.Text = item.Value;

object wrap = WdFindWrap.wdFindContinue;

object replace = WdReplace.wdReplaceAll;

find.Execute(FindText: find.Text, ReplaceWith: find.Replacement.Text, Replace: replace, Wrap: wrap, MatchCase: false, MatchWholeWord: false, MatchWildcards: false);

}

wordDoc.Visible = true;

//wordDoc.Documents.Save();

//wordDoc.Documents.Close();

//wordDoc.Quit();

}

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

{

startDate = dateTimePicker3.Value.ToString("d");

endDate = dateTimePicker1.Value.ToString("d");

string sss = $"SELECT COUNT(\*) FROM agent.direction WHERE direction\_date BETWEEN '{dateTimePicker3.Value.ToString("yyyy-MM-dd")}' AND '{dateTimePicker1.Value.ToString("yyyy-MM-dd")}'";

count = Convert.ToInt32(func.search(sss));

cool = Convert.ToInt32(func.search(sss+ " AND direction\_status = 'Принято'"));

bad = Convert.ToInt32(func.search(sss + " AND direction\_status = 'Отклонено'"));

load = Convert.ToInt32(func.search(sss + " AND direction\_status = 'Ожидание'"));

percent = cool \* 100 / count;

var dataTable = new System.Data.DataTable();

string search = $@"SELECT CONCAT(applicant.applicant\_surname, ' ',applicant.applicant\_name, ' ', applicant.applicant\_patronymic) as 'Соискатель',profession.name as 'Профессия' , direction.direction\_date as 'Дата направления', direction.direction\_status as 'Статус'

FROM direction

INNER JOIN applicant ON direction.direction\_aplicant = applicant.applicant\_id

INNER JOIN employe ON direction.direction\_employee = employe.id

INNER JOIN vacancy on direction\_vacancy = vacancy.id

INNER JOIN profession on vacancy\_profession = profession.id

WHERE direction.direction\_date BETWEEN '{dateTimePicker3.Value.ToString("yyyy-MM-dd")}' AND '{dateTimePicker1.Value.ToString("yyyy-MM-dd")}'

ORDER BY direction.direction\_date DESC

";

returnDate(search, dataTable);

exportToWord(dataTable);

this.Close();

}

}