# РЕАЛИЗАЦИЯ

## Разработка классов информационной системы

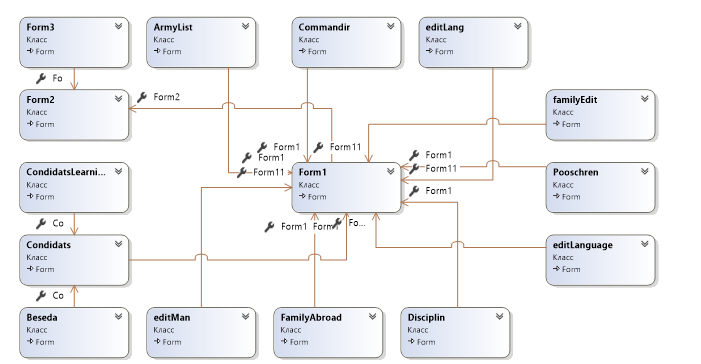


Рисунок 4.1 – диаграмма классов

Запускаемым классом является класс Programm.cs. При запуске он выводит на экран форму Form2. Она позволяет пользователю перейти в окно редактирования списков, Form 3, или на главное окно работы с программой Form1.

За работу с военнослужащими отвечает класс editMan. Он показывает форму, через которую можно добавить нового служащего или изменить его данные. За работу с послужным списком отвечает класс ArmyList. Работа с данными кандидата, а также с учебой кандидата и его беседы ведется через класс Condidats, CondidatsLearning и Beseda соответственно. Класс Commandir отвечает за командировки, editLang за учебу служащего. familyEdit и Pooschren за редактирование родственников военнослужащего и поощрений соответственно. За работу с иностранными языками отвечает класс editLanguage .Disciplin за дисциплинарные взыскания и FamilyAbroad за родственников, живущих за грницей.

## [Разработка интерфейса программного продукта](#_Toc474749001)

Стартовая форма представляет меню с вариантами перехода между редактированием информации списка и главным окном программысобой описание сайта. В полноразмерном варианте она выглядит как на рисунке ниже.

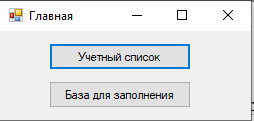


Рисунок 4.2 – Стартовое окно

База для заполнения выглядит следующим образом

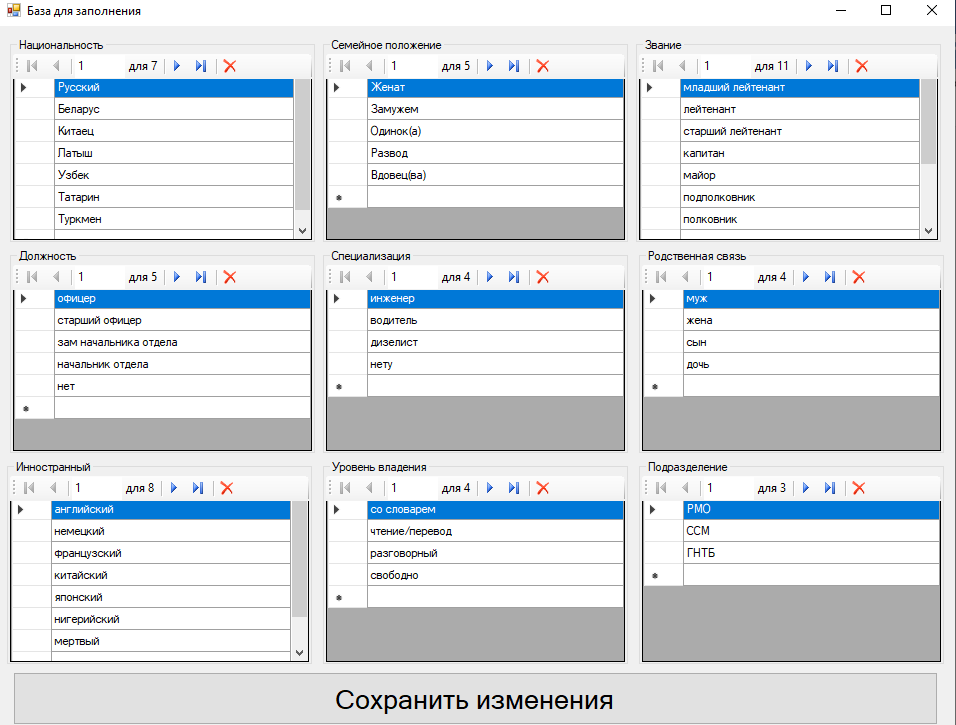


Рисунок 4.3 – База для заполнения

На ней можно изменить данные в базе, которые предлагаются пользователям в выпадающих списках (за исключением самих служащих и кондидатов)

При выборе учетного списка откроется следующее окно

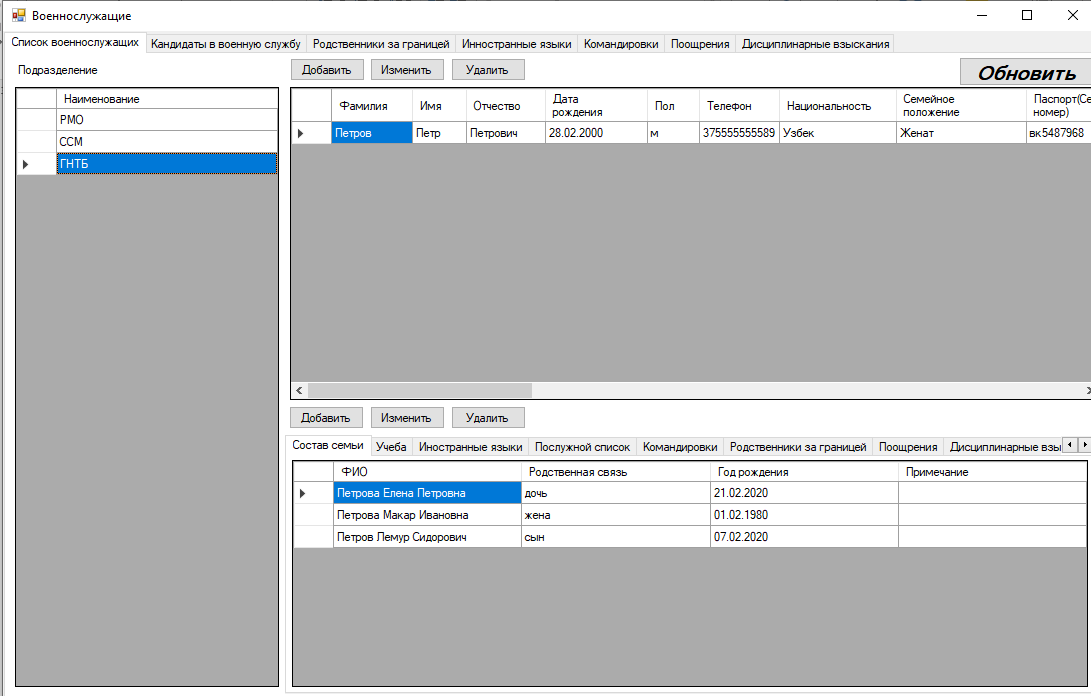


Рисунок 4.4 – Главное окно программы

На нем сразу же отображается подразделение. При щелчке на оном, в правом верхнем окне отобразятся все служащие данного подразделения.

Окно добавления служащего выглядит следующим образом

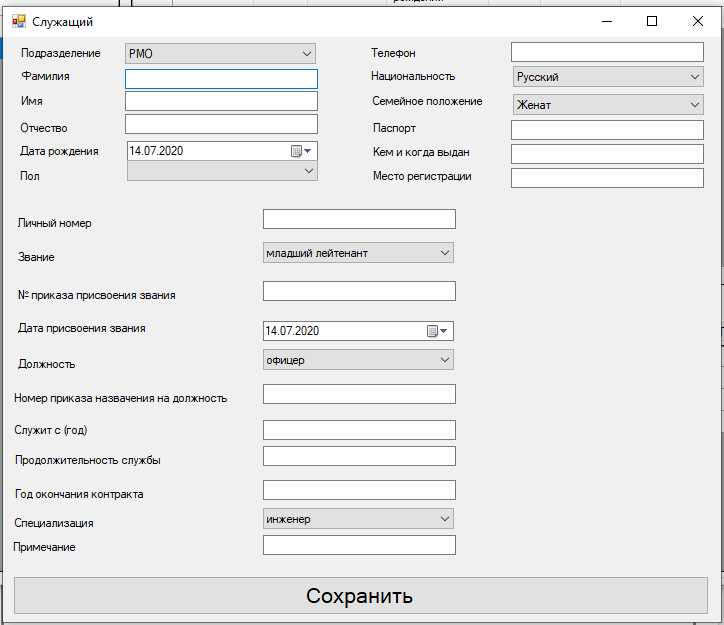


Рисунок 4.5 – Окно добавления служащего

Сразу под списком служащих указаны все их свойства: состав семьи, учеба, иностранные языки, послужной список, Командировки, родственники за границей, поощрения и дисциплинарные взыскания. Для перехода по ним нужно нажать на название категории для просмотра.

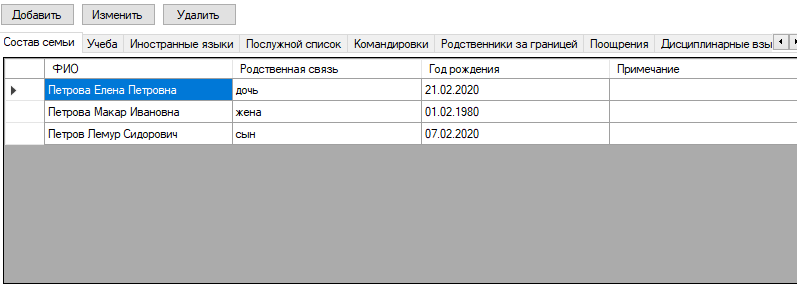


Рисунок 4.6 – Состав семьи

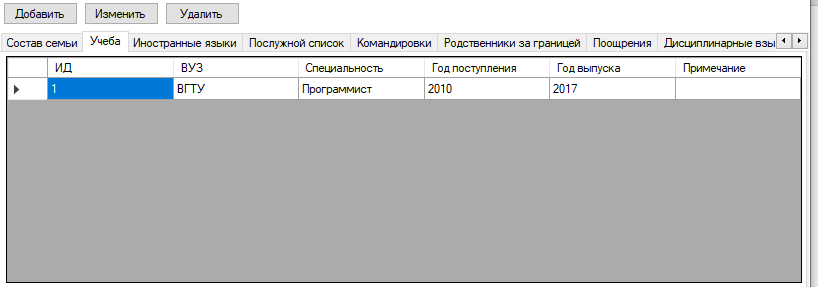


Рисунок 4.7 – Учеба

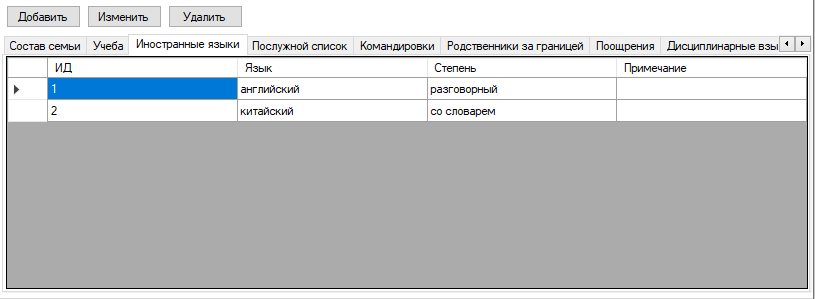


Рисунок 4.8 – иностранные языки

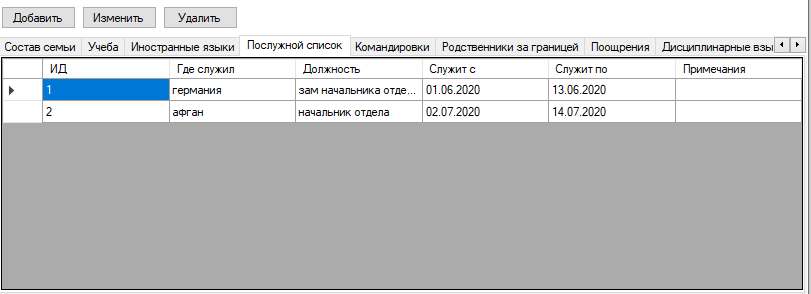


Рисунок 4.9 – послужной список

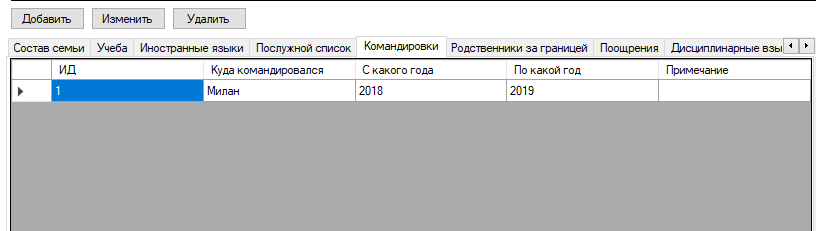


Рисунок 4.10 – Командировки

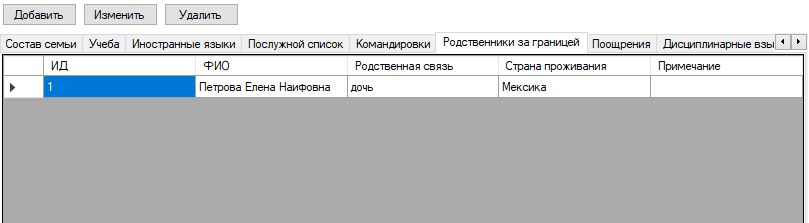


Рисунок 4.11 – родственники за границей

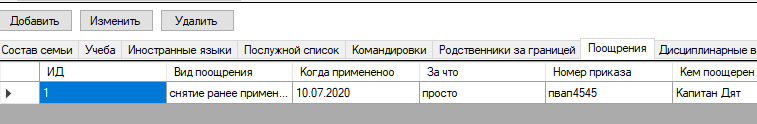


Рисунок 4.12 – поощрения

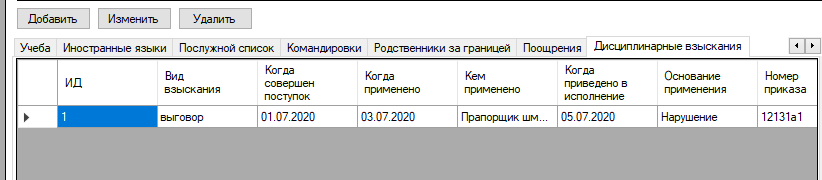


Рисунок 4.13 – дисциплинарные взыскания

Далее идут формы для работы с этими данными. Данные формы появятся при нажатии на кнопку добавить или изменить.

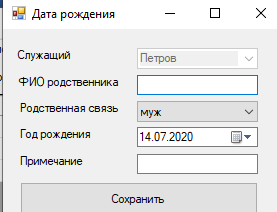


Рисунок 4.14 – состав семьи

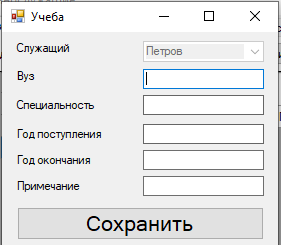


Рисунок 4.15 – учеба

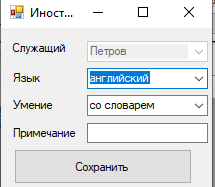


Рисунок 4.16 – иностранные языки

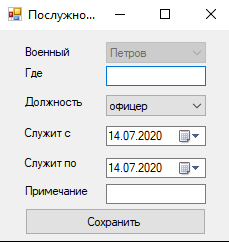


Рисунок 4.17 – послужной список

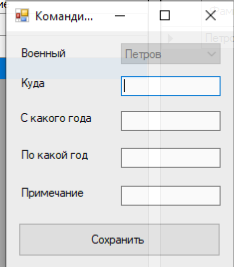


Рисунок 4.18 – командировки

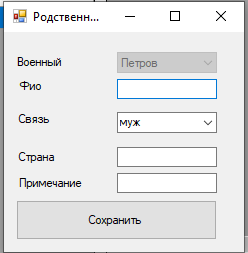


Рисунок 4.19 – родственники за границей

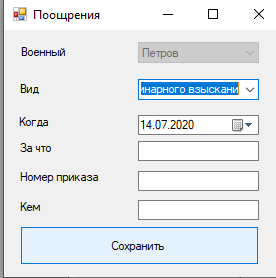


Рисунок 4.20 – поощрения

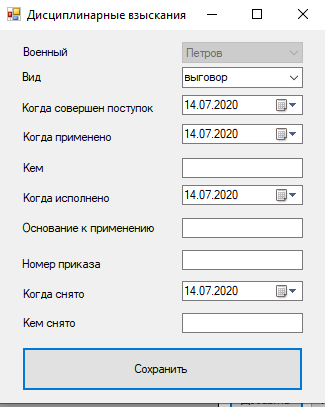


Рисунок 4.21 – дисциплинарные взыскания

При переходе на вторую вкладку на главной форме, мы попадаем в меню работы с кандидатами на военную сдужбу

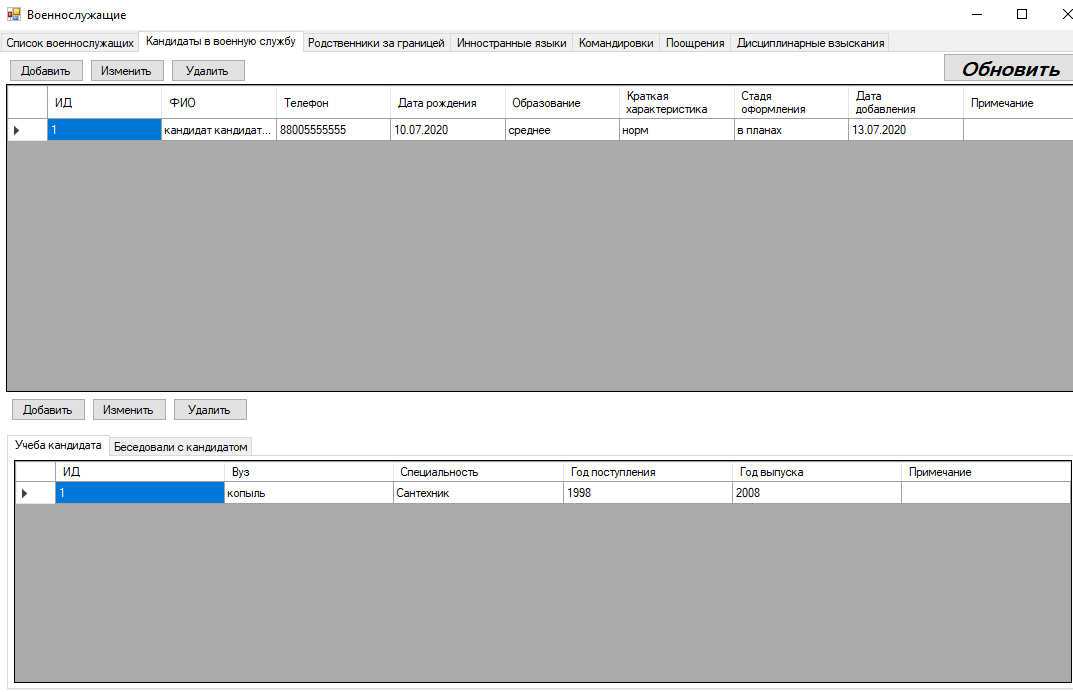


Рисунок 4.22 – кандидаты

У них также есть свойства – учеба канидата и беседа с кандидатами

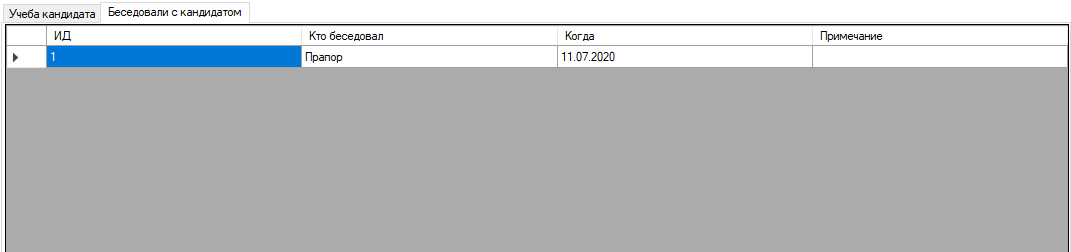


Рисунок 4.23 – беседа с кандидатом

Следующая вкладка показывает родственников служащих за границей

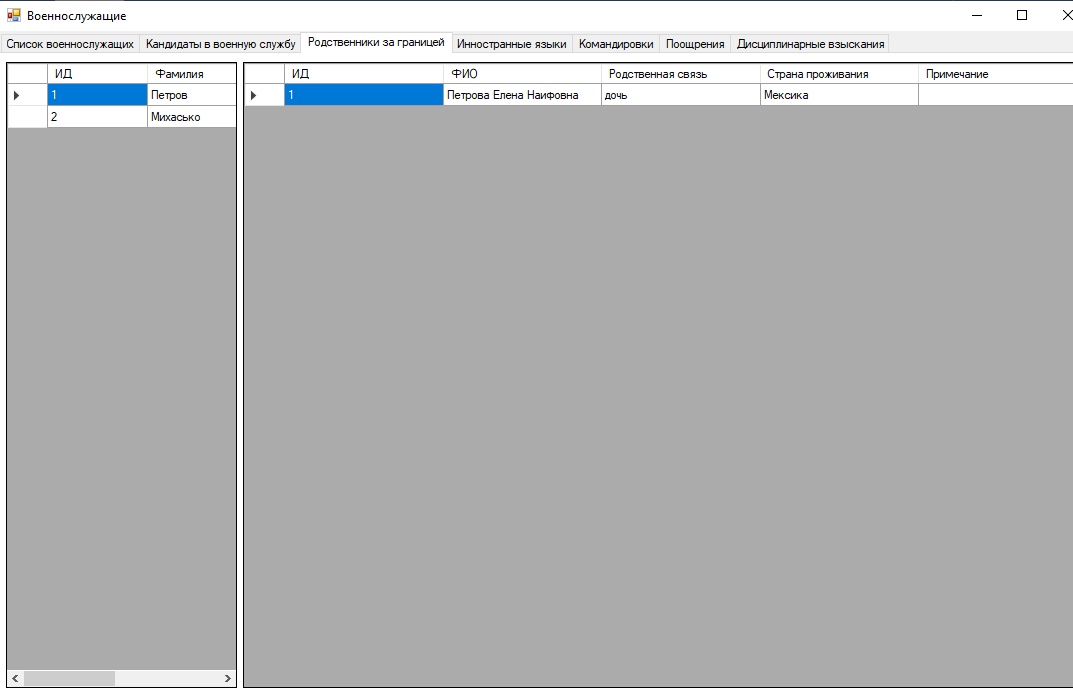


Рисунок 4.24 – родственники за границей

Дальнейшие окна показывают нам списки в общем виде.

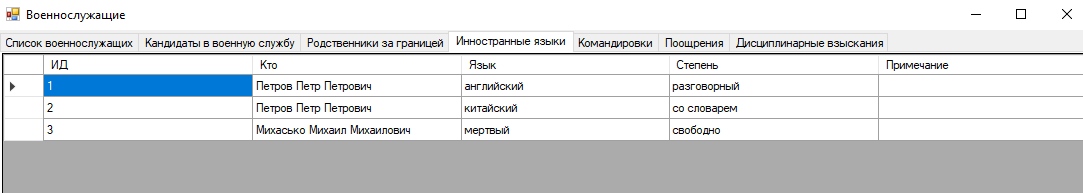


Рисунок 4.25 – иностранные языки

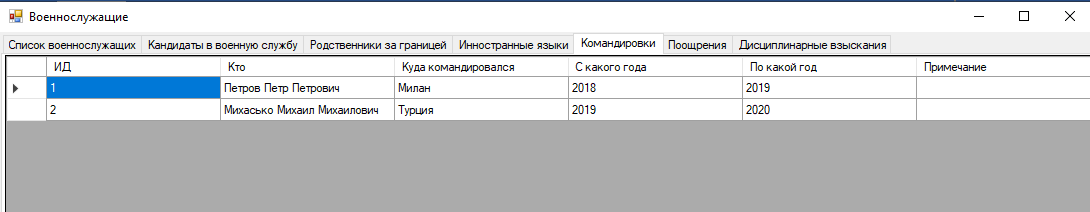


Рисунок 4.26 – командировки

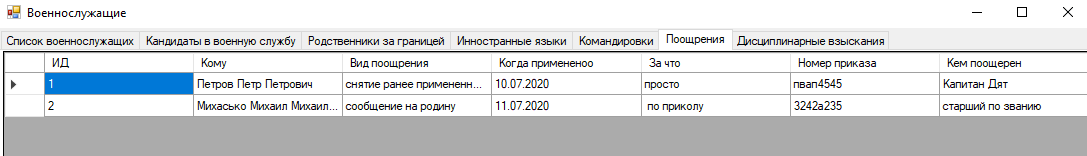


Рисунок 4.27 – поощрения

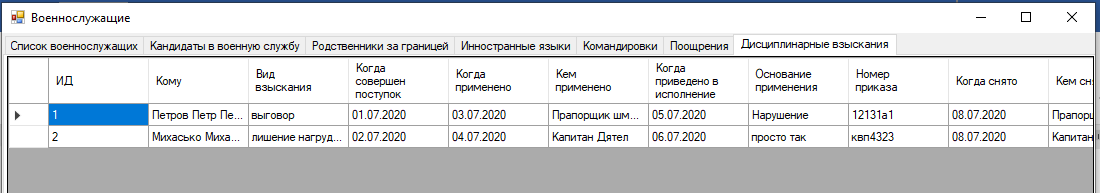


Рисунок 4.28 – дисциплинарные выскания

Рисунок 4.29 –

## [Разработка алгоритмов реализации вариантов использования](#_Toc474749002)

Реализация работы с данными служащего представлена на листинге 4.3.

Листинг 4.3 – Работа с данными служащего

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

{

if (id < 0) save();

else update();

}

private void save() {

try

{

Database1DataSet.НациональностьRow nation = (Database1DataSet.НациональностьRow)database1DataSet.Национальность.Select($" Название ='{comboBox8.SelectedValue}'")[0];

Database1DataSet.Семейное\_положениеRow family = (Database1DataSet.Семейное\_положениеRow)database1DataSet.Семейное\_положение.Select($" Название ='{comboBox3.SelectedValue}'")[0];

Database1DataSet.ЗваниеRow swan = (Database1DataSet.ЗваниеRow)database1DataSet.Звание.Select($" Название ='{comboBox4.SelectedValue}'")[0];

Database1DataSet.ДолжностьRow dolsh = (Database1DataSet.ДолжностьRow)database1DataSet.Должность.Select($" Название ='{comboBox5.SelectedValue}'")[0];

Database1DataSet.СпециализацияRow spec = (Database1DataSet.СпециализацияRow)database1DataSet.Специализация.Select($" Название ='{comboBox6.SelectedValue}'")[0];

Database1DataSet.ПодразделениеRow podr = (Database1DataSet.ПодразделениеRow)database1DataSet.Подразделение.Select($" Наименование ='{comboBox7.SelectedValue}'")[0];

database1DataSet.Главная.AddГлавнаяRow(

textBox2.Text, textBox3.Text, textBox4.Text,

dateTimePicker1.Value, comboBox1.Text,

textBox5.Text, nation, family,

textBox6.Text, textBox7.Text, textBox8.Text,

textBox9.Text, swan, textBox10.Text, dateTimePicker2.Value,

dolsh, textBox12.Text, textBox13.Text,

int.Parse(textBox14.Text), int.Parse(textBox15.Text),

spec, textBox16.Text,

podr).EndEdit();

this.главнаяBindingSource.EndEdit();

this.главнаяTableAdapter.Update(this.database1DataSet.Главная);

this.database1DataSet.Главная.AcceptChanges();

this.database1DataSet.AcceptChanges();

MessageBox.Show("Добавлено. Обновите форму для просмотра");

this.Close();

}

catch (Exception ex) {

MessageBox.Show("Ошибка добавления. Проверьте вводимые данные.");

}

}

private void update() {

try {

//DataRow row = database1DataSet.Главная.Select(" ИД = " + id)[0];

Database1DataSet.НациональностьRow nation = (Database1DataSet.НациональностьRow)database1DataSet.Национальность.Select($" Название ='{comboBox8.SelectedValue}'")[0];

Database1DataSet.Семейное\_положениеRow family = (Database1DataSet.Семейное\_положениеRow)database1DataSet.Семейное\_положение.Select($" Название ='{comboBox3.SelectedValue}'")[0];

Database1DataSet.ЗваниеRow swan = (Database1DataSet.ЗваниеRow)database1DataSet.Звание.Select($" Название ='{comboBox4.SelectedValue}'")[0];

Database1DataSet.ДолжностьRow dolsh = (Database1DataSet.ДолжностьRow)database1DataSet.Должность.Select($" Название ='{comboBox5.SelectedValue}'")[0];

Database1DataSet.СпециализацияRow spec = (Database1DataSet.СпециализацияRow)database1DataSet.Специализация.Select($" Название ='{comboBox6.SelectedValue}'")[0];

Database1DataSet.ПодразделениеRow podr = (Database1DataSet.ПодразделениеRow)database1DataSet.Подразделение.Select($" Наименование ='{comboBox7.SelectedValue}'")[0];

Database1DataSet.ГлавнаяRow row = database1DataSet.Главная.FindByИД(id);

//row.BeginEdit();

//row.Подразделение = comboBox7.SelectedValue.ToString();

row[1] = textBox2.Text;

row[2] = textBox3.Text;

row[3] = textBox4.Text;

row[4] = dateTimePicker1.Text;

row[5] = comboBox1.Text;

row[6] = textBox5.Text;

row[7] = comboBox8.SelectedValue;

row[8] = comboBox3.SelectedValue;

row[9] = textBox6.Text;

row[10] = textBox7.Text;

row[11] = textBox8.Text;

row[12] = textBox9.Text;

row[13] = comboBox4.SelectedValue;

row[14] = textBox10.Text;

row[15] = dateTimePicker2.Text;

row[16] = comboBox5.SelectedValue;

row[17] = textBox12.Text;

row[18] = textBox13.Text;

row[19] = int.Parse(textBox14.Text);

row[20] = int.Parse(textBox15.Text);

row[21] = comboBox6.SelectedValue;

row[22] = textBox16.Text;

row[23] = comboBox7.SelectedValue;

row.EndEdit();

//row.AcceptChanges();

this.главнаяBindingSource.EndEdit();

this.главнаяTableAdapter.Update(this.database1DataSet.Главная);

this.database1DataSet.Главная.AcceptChanges();

this.database1DataSet.AcceptChanges();

MessageBox.Show("Обнавлено. Обновите форму для просмотра");

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка Обновления. Проверьте вводимые данные.");

}

}

Реализация загрузки данных.

Листинг 4.4 - Загрузка информации о выбранном члене семьи

private void load() {

id = int.Parse(currentRow.Cells[0].Value.ToString());

comboBox1.SelectedText = currentRow.Cells[1].Value.ToString();

textBox1.Text = currentRow.Cells[2].Value.ToString();

comboBox2.SelectedValue = currentRow.Cells[3].Value.ToString();

dateTimePicker1.Text= currentRow.Cells[4].Value.ToString();

textBox3.Text = currentRow.Cells[5].Value.ToString();

}

Реализация вывода слов без повторений.

Листинг 4.5 – вывод слов из словарей без повторений

@Autowired

WordService wordService;

@GetMapping

String getIndexController(@RequestParam(name = "letter") String letter, Model model) {

ArrayList<Character> letters = **new** ArrayList<Character>();

**for**(**char** i = 'а';i<='я';i++)

letters.add(i);

model.addAttribute("letters", letters);

List<Word> words = wordService.repository.findAllByLetterIgnoreCase(letter);

Collections.*sort*(words);

List<Word> words2 = **new** ArrayList<Word>();

**if**(words.size()>0)

words2.add(words.get(0));

**for**(**int** i=1; i<words.size();i++) {

words2.add(words.get(i));

**if**(words2.get(words2.size()-1).getValue().equals(words2.get(words2.size()-2).getValue()))

words2.remove(words2.size()-1);

}

model.addAttribute("words", words2);

**return** "alphabet";

}

Класс для реализации работы с учебой кандидата.

Листинг 4.6 – Класс CondidatsLearning

public partial class CondidatsLearning : Form

{

int id = -1;

int idWho = -1;

DataGridViewRow currentRow = null;

public CondidatsLearning(int idWho)

{

InitializeComponent();

this.idWho = idWho;

}

public CondidatsLearning(int idWho, DataGridViewRow currentRow)

{

InitializeComponent();

this.currentRow = currentRow;

this.id = int.Parse(currentRow.Cells[0].Value.ToString());

this.idWho = idWho;

}

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

{

if (currentRow == null) save();

else update();

}

private void load()

{

textBox1.Text = currentRow.Cells[2].Value.ToString();

textBox2.Text = currentRow.Cells[3].Value.ToString();

textBox3.Text = currentRow.Cells[4].Value.ToString();

textBox4.Text = currentRow.Cells[5].Value.ToString();

textBox5.Text = currentRow.Cells[6].Value.ToString();

}

private void save()

{

try

{

Database1DataSet.Кандидаты\_на\_военную\_службуRow ground = (Database1DataSet.Кандидаты\_на\_военную\_службуRow)database1DataSet.Кандидаты\_на\_военную\_службу.Select($" ИД ='{idWho}'")[0];

database1DataSet.Учеба\_кандидата.AddУчеба\_кандидатаRow(

ground, textBox1.Text, textBox2.Text, int.Parse(textBox3.Text), int.Parse(textBox4.Text), textBox5.Text);

this.учебаКандидатаBindingSource.EndEdit();

this.учеба\_кандидатаTableAdapter.Update(this.database1DataSet.Учеба\_кандидата);

this.database1DataSet.Учеба\_кандидата.AcceptChanges();

this.database1DataSet.AcceptChanges();

MessageBox.Show("Добавлено. Обновите форму для просмотра");

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка добавления. Проверьте вводимые данные");

}

}

private void update()

{

try

{

Database1DataSet.Учеба\_кандидатаRow row = database1DataSet.Учеба\_кандидата.FindByИД(id);

row[2] = textBox1.Text;

row[3] = textBox2.Text;

row[4] = textBox3.Text;

row[5] = textBox4.Text;

row[6] = textBox5.Text;

row.EndEdit();

this.учебаКандидатаBindingSource.EndEdit();

this.учеба\_кандидатаTableAdapter.Update(this.database1DataSet.Учеба\_кандидата);

this.database1DataSet.Учеба\_кандидата.AcceptChanges();

this.database1DataSet.AcceptChanges();

MessageBox.Show("Обновлено. Обновите форму для просмотра");

this.Close();

}

catch (Exception ex)

{

MessageBox.Show("Ошибка обновления. Проверьте вводимые данные");

}

}

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

{

// TODO: данная строка кода позволяет загрузить данные в таблицу "database1DataSet.Учеба\_кандидата". При необходимости она может быть перемещена или удалена.

this.учеба\_кандидатаTableAdapter.Fill(this.database1DataSet.Учеба\_кандидата);

// TODO: данная строка кода позволяет загрузить данные в таблицу "database1DataSet.Кандидаты\_на\_военную\_службу". При необходимости она может быть перемещена или удалена.

this.кандидаты\_на\_военную\_службуTableAdapter.Fill(this.database1DataSet.Кандидаты\_на\_военную\_службу);

if (currentRow != null) load();

comboBox2.SelectedValue = idWho;

}

}

## Модульное тестирование алгоритмов реализации вариантов использования

Модульное тестирование, или юнит-тестирование - это процесс в программировании, который направлен на проверку отдельных небольших частей приложения, также называемых атомарными, которые можно исследовать изолированно от других подобных частей. При выполнении данного тестирования могут проверяться как отдельные функции или методы классов, так и сами классы, взаимодействие классов, небольшие библиотеки и отдельные части приложения. Довольно часто данный вид тестирования реализуется с использованием специальных технологий и инструментальных средств автоматизации тестирования, значительно упрощающих и ускоряющих разработку соответствующих тест-кейсов [5]. Таким образом, оценивая каждый элемент отдельно и подтверждая правильность его работы, установить проблему значительно проще чем, если бы элемент был частью системы.

Примеры тестирования алгоритмов реализации вариантов использования:

1. Тестирование вывода служащих при выборе подразделения (таблица 4.1).

Таблица 4.1 - Тестирование вывода служащих при выборе подразделения

|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Предварительные условия | Шаги для воспроизведения | Ожидаемый результат |
| Тестирование вывода служащих | 1. Запустить программу 2. Перейти в учет служащих | 1. Нажать на подразделение в левом списке | 1. После нажатия на подразделение, в правом верхнем списке отобразятся все служащие данного подразделения |

2. Тестирование добавления служащего(таблица 4.2).

Таблица 4.2 - Тестирование добавления служащего

|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Предварительные условия | Шаги для воспроизведения | Ожидаемый результат |
| Тестирование добавления служащего | 1. Запустить программу 2. Перейти в учет служащих | 1. Над таблицей служащих нажать кнопку добавить 2. Заполнить все поля 3. Нажать сохранить 4. Нажать обновить | При нажатии кнопки сохранить, пользователю будет показано информационна окно, о статусе сохранения. После подтверждения этого окна ему остается перезагрузить форму. При нажатии на кнопку обновить, добавленным им служащий отобразится в списке |

3. Тестирование изменение данных (таблица 4.3).

Таблица 4.3 - Тестирование изменения поощрения

|  |  |  |  |
| --- | --- | --- | --- |
| Краткое описание | Предварительные условия | Шаги для воспроизведения | Ожидаемый результат |
| Тестирование изменения поощрения | 1. Запустить программу 2. Перейти в учет служащих | 1. Справа внизу перейти во вкладку поощрения 2. Из списка поощрений выбрать одно и нажать на него 3. Нажать на кнопку изменить над списком 4. Изменить данные 5. Нажать кнопку сохранить 6. Нажать кнопку обновить | При нажатии кнопки сохранить, пользователю будет показано информационна окно, о статусе сохранения. После подтверждения этого окна ему остается перезагрузить форму. При нажатии на кнопку обновить, измененные данные отобразятся в таблице |