**Московский государственный технический**
**университет им. Н.Э. Баумана**

Факультет «Радиотехнический»
Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Базовые компоненты интернет-технологий»

# Отчет по лабораторной работе №5

Выполнил:
студент группы РТ5-31Б

Ходосов Михаил
Подпись и дата:

Проверил:
доцент каф. ИУ5

Гапанюк Ю.Е.
Подпись и дата:

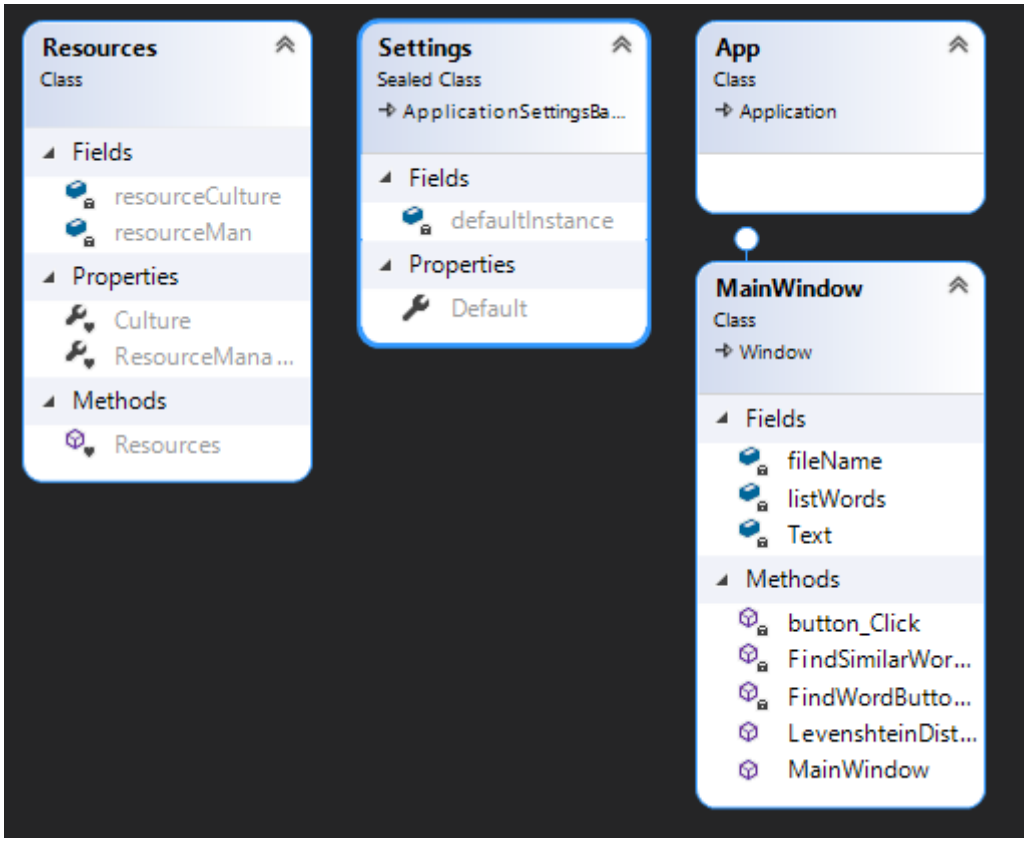
Москва, 2020 г.

# Описание задания

Разработать программу, реализующую вычисление расстояния Левенштейна с использованием алгоритма Вагнера-Фишера.

1. Программа должна быть разработана в виде библиотеки классов на языке C#.
2. Использовать самый простой вариант алгоритма без оптимизации.
3. Дополнительно возможно реализовать вычисление расстояния Дамерау-Левенштейна (с учетом перестановок соседних символов).
4. Модифицировать предыдущую лабораторную работу, вместо поиска подстроки используется вычисление расстояния Левенштейна.
5. Предусмотреть отдельное поле ввода для максимального расстояния. Если расстояние Левенштейна между двумя строками больше максимального, то строки считаются несовпадающими и не выводятся в список результатов.

# Диаграмма классов




# Текст программы

MainWindow.xaml

```
<Window x:Class="lab5.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
        xmlns:local="clr-namespace:lab5"
        mc:Ignorable="d"
        Title="Лабораторная работа №5" Height="450" Width="1200">
    <Grid Width="1200" Height="420">
        <StackPanel Orientation="Horizontal" Margin="0,0,0,0">
            <Button x:Name="OpenFileButton" Content="Чтение из файла"
HorizontalAlignment="Left" Margin="10,10,0,0" VerticalAlignment="Top" Width="100"
Height="40" Click="button_Click"/>
            <Label x:Name="ReadingTimeLabel" VerticalAlignment="Top" Content="Время
чтения:" Height="30" Width="120" Margin="10, 5, 0, 0"/>
            <TextBox x:Name="ElapsedTime" Height="20" Margin="-120,30,0,0"
VerticalAlignment="Top" Width="120"/>
```

```
            <Label x:Name="SearchingTimeLabel" VerticalAlignment="Top" Content="Время
поиска:" Height="30" Width="120" Margin="210, 10, 0, 0"/>
            <TextBox x:Name="SearchingTime" Height="20" Margin="-185,40,-70,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="120"/>
            <Button x:Name="FindWordButton" Content="Найти слово" Height="20"
Margin="10,10,0,0" VerticalAlignment="Top" Width="140" Click="FindWordButton_Click"/>
            <TextBox x:Name="FindWordField" TextWrapping="Wrap" Width="140" Height="20"
Margin="-140,40,10,2" VerticalAlignment="Top"/>
            <Label x:Name="SampleWordLabel" VerticalAlignment="Top" Content="Слово для
поиска:" Height="30" Width="180" Margin="60, 5, 0, 0"/>
            <TextBox x:Name="SampleWord" Height="20" Margin="-175,30,0,0"
VerticalAlignment="Top" Width="180"/>
            <Label x:Name="MaxDistLabel" VerticalAlignment="Top" Content="Максимальное
расстояние:" Height="30" Width="180" Margin="10, 5, 0, 0"/>
            <TextBox x:Name="MaxDistText" Height="20" Margin="-235,30,-70,0"
TextWrapping="Wrap" VerticalAlignment="Top" Width="185"/>
            <Button x:Name="FindSimilarWords" Content="Найти похожие слова" Width="380"
Margin="-370, 60, 0, 0" VerticalAlignment="Top" Click="FindSimilarWords_Click"/>

        </StackPanel>
        <Label x:Name="PathFileLabel" Content="Путь к файлу:" Height="30" Width="120"
Margin="-1065, -300, 0, 0"/>
        <ListBox x:Name="Content" HorizontalAlignment="Left" Height="308"
Margin="10,100,0,0" VerticalAlignment="Top" Width="390"/>
        <TextBox x:Name="PathFile" HorizontalAlignment="Left" Height="20"
Margin="10,70,0,0" TextWrapping="Wrap" VerticalAlignment="Top" Width="390"/>
        <ListBox x:Name="FoundWords" HorizontalAlignment="Left" Height="308"
Margin="455,100,10,0" VerticalAlignment="Top" Width="265"/>
        <ListBox x:Name="SimilarWordsBox" HorizontalAlignment="Left" Height="308"
Margin="795,100,0,0" VerticalAlignment="Top" Width="380"/>

    </Grid>
</Window>
```

## MainWindow.xaml.cs

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Data;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Imaging;
using System.Windows.Navigation;
using System.Windows.Shapes;
using System.Diagnostics;
using System.Threading;

namespace lab5
{
    /// <summary>
    /// Логика взаимодействия для MainWindow.xaml
    /// </summary>
    public partial class MainWindow : Window
    {
```

```
        List<string> listWords = new List<string>();
        private string fileName;
        private string Text;
        public MainWindow()
        {
            InitializeComponent();
        }
        //Чтение из файла
        private void button_Click(object sender, RoutedEventArgs e)
        {
            Content.Items.Clear();
            listWords.Clear();
            fileName = "";
            PathFile.Text = "";
            Text = "";

            Microsoft.Win32.OpenFileDialog fileDialog = new
Microsoft.Win32.OpenFileDialog();
            fileDialog.Filter = "Только текстовые файлы|*.txt";
            if (fileDialog.ShowDialog() == true)
            {
                Stopwatch timeLoading = new Stopwatch();
                timeLoading.Start();

                fileName = fileDialog.FileName;
                PathFile.Text = fileName;

                Text = File.ReadAllText(fileName);

                string[] words = Text.Split(' ', ',', '.', '?', '!', '/', '|', '"', '\n',
'\t', '_', '-', '(', ')', '*', '{', '}', '[', ']');

                foreach (string word in words)
                {
                    if (!listWords.Contains(word))
                    {
                        listWords.Add(word);
                    }
                }

                timeLoading.Stop();
                ElapsedTime.Text = timeLoading.Elapsed.ToString();

                foreach (string word in listWords)
                {
                    Content.Items.Add(word);
                }

            }
        }
        //Поиск слов, для которых заданная строка является подстрокой
        private void FindWordButton_Click(object sender, RoutedEventArgs e)
        {
            FoundWords.Items.Clear();

            if (FindWordField.Text == null)
                return;

            Stopwatch timeSearching = new Stopwatch();
            timeSearching.Start();

            foreach (string word in listWords)
            {
                if (word.Contains(FindWordField.Text))
                {
```

```
                    FoundWords.Items.Add(word);
                }
            }

            timeSearching.Stop();
            SearchingTime.Text = timeSearching.Elapsed.ToString();
            FindWordField.Text = "";
        }

        //Поиск похожих слов
        private void FindSimilarWords_Click(object sender, RoutedEventArgs e)
        {
            string sampleWord = SampleWord.Text.Trim();
            int maxDistance = Convert.ToInt32(MaxDistText.Text);

            if (!string.IsNullOrWhiteSpace(sampleWord) && listWords.Count > 0)
            {
                List<string> SimilarWords = new List<string>();
                foreach (string word in listWords)
                {
                    int dist = LevenshteinDistance(word, sampleWord);
                    if (dist <= maxDistance)
                    {
                        SimilarWords.Add(word);
                    }
                }

                SimilarWordsBox.Items.Clear();
                foreach (string word in SimilarWords)
                {
                    SimilarWordsBox.Items.Add(word);
                }

            }
        }

        //Поиск расстояния Левенштейна
        public static int LevenshteinDistance(string str1, string str2)
        {
            //Проверка на исключительные случаи

            if ((str1 == null && str2 == null) || (str1 == str2)) return 0;
            if (str1 == null || str2 == null) throw new ArgumentNullException("Одна из
строк пустая!\n");

            //Алгоритм Вагнера – Фишера

            int[,] matrix = new int[str1.Length + 1, str2.Length + 1];

            for (int i = 0; i <= str1.Length; i++)
            {
                matrix[i, 0] = i;
            }
            for (int j = 0; j <= str2.Length; j++)
            {
                matrix[0, j] = j;
            }

            for (int i = 1; i <= str1.Length; i++)
            {
                for (int j = 1; j <= str2.Length; j++)
                {
                    int d = 1;
                    if (str1[i - 1] == str2[j - 1]) d = 0;
```

```
                    matrix[i, j] = Math.Min(Math.Min(matrix[i - 1, j] + 1, matrix[i, j - 1] + 1), matrix[i - 1, j - 1] + d);

                }
            }
            return matrix[str1.Length, str2.Length];
        }
    }
}
```

# Результат работы программы