МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет ИТМО»

ФАКУЛЬТЕТ БЕЗОПАСНОСТИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Управление мобильными устройствами

Лабораторная работа №3 «Формирование счёта на оплату услуг» Вариант № 3

Работу выполнил:
Студент группы N3347
Волкова Е. А.

Проверено:
Таранов С. В.

Санкт-Петербург
2020

УНИВЕРСИТЕТ ИТМО

1. Задание

По полученным результатам тарификации услуг «Телефония» и «Интернет» в лабораторных работах 1, 2 сформировать счет на оплату в формате .pdf.

Все поля печатной формы должны заполняться разработанным программным модулем. Название банка, имена покупателей и прочие формальные поля можно заполнить какими угодно значениями, зависит от фантазии.
Стоимость услуг услуг берется из предыдущих двух работ.

В качестве результата работы необходимо представить программный модуль для генерации печатной формы счета на оплату и полученный файл .pdf.

2. Описание выбранных средств реализации и обоснование выбора

Для реализации программного модуля я выбрала C# так как владею им наиболее свободно из всех языков, и считаю самым удобным инструментом для реализации большинства задач.

3. Исходный код

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Word = Microsoft.Office.Interop.Word;

namespace WindowsFormsApp4
{
   public partial class Form1 : Form
   {
      private readonly string TemplateFileName = "eyx.docx";
      public Form1()
```

```
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            var name = textBox3.Text;
            var billtake = textBox4.Text;
            var bank = textBox5.Text;
            var inn = textBox7.Text;
            var kpp = textBox8.Text;
            var bik = textBox9.Text;
            var bill = textBox6.Text;
            var date = dateTimePicker2.Value.ToShortDateString();
            var results = Convert.ToString (Math.Round((Lab1.Calculate()),2));
            var result = Convert.ToString(Math.Round((Lab2.Calculate()),2));
            var all = Convert.ToString (Math.Round ((Lab1.Calculate() +
Lab2.Calculate()),2));
            var a = Lab1.Calculate() + Lab2.Calculate();
            var nds = Convert.ToString(Math.Round((a * 0.2), 2));
            var b = Convert.ToString(Math.Round((a * 1.2), 2));
            var num = textBox1.Text;



            var wordApp = new Word.Application();
            wordApp.Visible = false;

            var wordDoc = wordApp.Documents.Open(TemplateFileName);
            ReplaceWordStub("{name}", name, wordDoc);
            ReplaceWordStub("{billtake}", billtake, wordDoc);
            ReplaceWordStub("{bank}", bank, wordDoc);
            ReplaceWordStub("{inn}", inn, wordDoc);
            ReplaceWordStub("{kpp}", kpp, wordDoc);
            ReplaceWordStub("{bik}", bik, wordDoc);
            ReplaceWordStub("{bill}", bill, wordDoc);
            ReplaceWordStub("{date}", date, wordDoc);
            ReplaceWordStub("{name}", name, wordDoc);
            ReplaceWordStub("{results}", results, wordDoc);
```

```
            ReplaceWordStub("{result}", result, wordDoc);
            ReplaceWordStub("{results}", results, wordDoc);
            ReplaceWordStub("{result}", result, wordDoc);
            ReplaceWordStub("{all}", all, wordDoc);
            ReplaceWordStub("{result}", result, wordDoc);
            ReplaceWordStub("{nds}", nds, wordDoc);
            ReplaceWordStub("{b}", b, wordDoc);
            ReplaceWordStub("{num}", num, wordDoc);


            wordDoc.SaveAs("C:\result.docx");
            wordDoc.ExportAsFixedFormat("C:\result.pdf",
Word.WdExportFormat.wdExportFormatPDF);

            wordApp.Visible = true;
        }

        private void ReplaceWordStub(string stubToReplace, string text,
Word.Document wordDoc)
        {
            var range = wordDoc.Content;
            range.Find.ClearFormatting();
            range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);

        }
    }
    class Lab1
    {
        public static float Calculate()
        {
            var data = ProcessCSV("data.csv");
            float costt = 0;
            float costs = 0;
            foreach (var ndata in data)
            {
                int result = String.Compare(ndata.msisdn_origin, Cost.num);
                if (result == 0)
                {
                    string x = ndata.call_duration;
```

```
                x = x.Replace('.', ',');
                float d = Convert.ToSingle(x);
                costt = (d * Cost.origin) - Cost.free;
                string b = ndata.sms_number;
                costs = Convert.ToSingle(b) * Cost.sms;
            }
        }
        float results = costt + costs;
        return results;
    }
    private static List<Data> ProcessCSV(string path)
    {
        return File.ReadAllLines(path)
            .Skip(1)
            .Where(row => row.Length > 0)
            .Select(Data.ParseRow).ToList();
    }
}
public class Data
{
    public string timestamp { get; set; }
    public string msisdn_origin { get; set; }
    public string msisdn_dest { get; set; }
    public string call_duration { get; set; }
    public string sms_number { get; set; }

    internal static Data ParseRow(string row)
    {
        var columns = row.Split(',');

        return new Data()
        {
            timestamp = columns[0],
            msisdn_origin = columns[1],
            msisdn_dest = columns[2],
            call_duration = columns[3],
            sms_number = columns[4]
        };
    }
```

```
}
public class Cost
{
    internal const float origin = 2;
    internal const float sms = 2;
    internal const float free = 40;
    internal const string num = "915783624";
    internal const float internet = 1;
    internal const string ip = "192.168.250.27";
}
class Lab2
{
    public static float Calculate()
    {
        var data = ProcessCSV("data1.csv");
        float costtI = 0;
        float costtO = 0;
        foreach (var ndata in data)
        {
            int result1 = String.Compare(ndata.Sa, Cost.ip);
            int result2 = String.Compare(ndata.Da, Cost.ip);
            if (result1 == 0 || result2 == 0)
            {
                float Ibyt = Convert.ToSingle(ndata.Ibyt);
                float Obyt = Convert.ToSingle(ndata.Obyt);
                costtI = (costtI + Ibyt);
                costtO = (costtO + Obyt);
            }
        }
        float costt = (costtI + costtO) / (1024 * 1024) * Cost.internet;
        return costt;
    }
    private static List<Data2> ProcessCSV(string path)
    {
        return File.ReadAllLines(path)
            .Skip(1)
            .Where(row => row.Length > 0)
            .Select(Data2.ParseRow).ToList();
    }
```

```
}
public class Data2
{
    public string Ts { get; set; }
    public string Te { get; set; }
    public string Td { get; set; }
    public string Sa { get; set; }
    public string Da { get; set; }
    public string Sp { get; set; }
    public string Dp { get; set; }
    public string Pr { get; set; }
    public string Ra_flg { get; set; }
    public string Fwd { get; set; }
    public string Stos { get; set; }
    public string Ipkt { get; set; }
    public string Ibyt { get; set; }
    public string Opkt { get; set; }
    public string Obyt { get; set; }
    public string In_ { get; set; }
    public string Out_ { get; set; }
    public string Sas { get; set; }
    public string Das { get; set; }
    public string Smk { get; set; }
    public string Dmk { get; set; }
    public string Dtos { get; set; }
    public string Dir { get; set; }
    public string Nh { get; set; }
    public string Nhb { get; set; }
    public string Svln { get; set; }
    public string Dvln { get; set; }
    public string Ismc { get; set; }
    public string Odmc { get; set; }
    public string Idmc { get; set; }
    public string Osmc { get; set; }
    public string Mpls1 { get; set; }
    public string Mpls2 { get; set; }
    public string Mpls3 { get; set; }
    public string Mpls4 { get; set; }
    public string Mpls5 { get; set; }
```

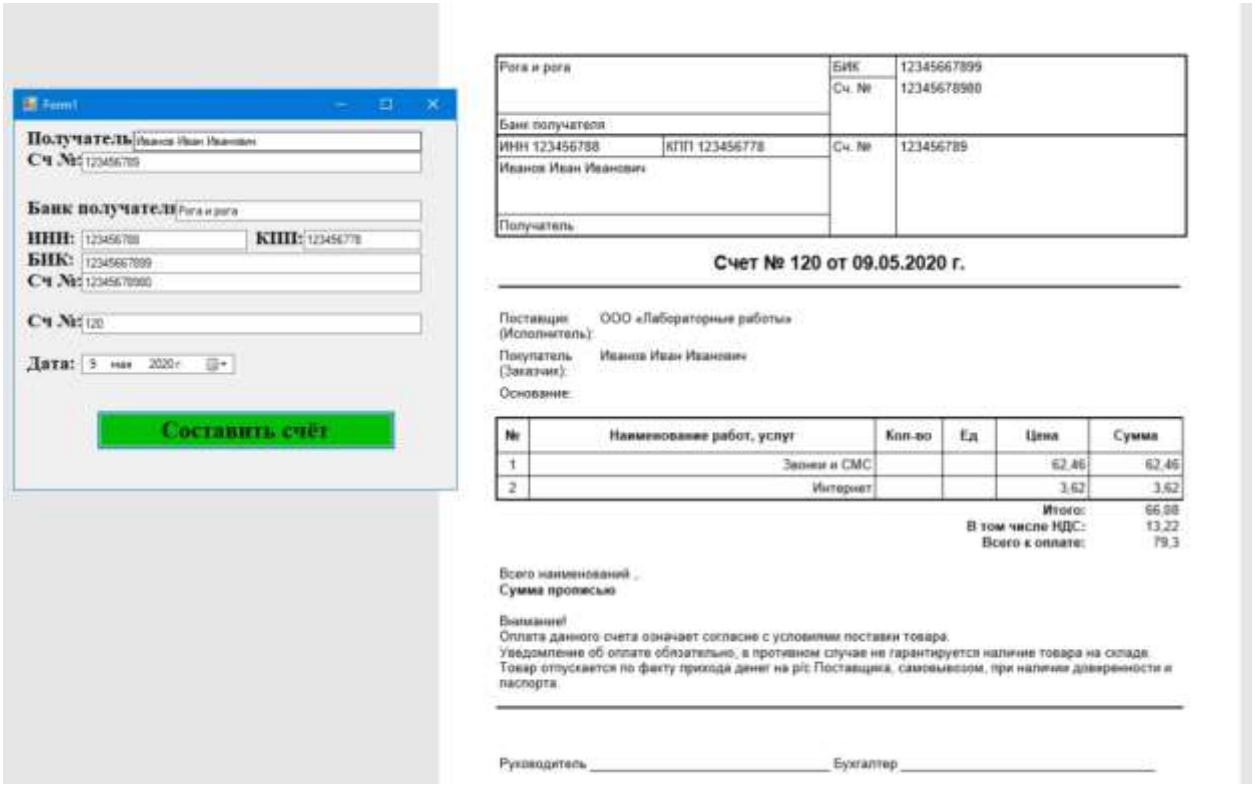
```
public string Mpls6 { get; set; }
public string Mpls7 { get; set; }
public string Mpls8 { get; set; }
public string Mpls9 { get; set; }
public string Mpls10 { get; set; }
public string Cl { get; set; }
public string Sl { get; set; }
public string Al { get; set; }
public string Ra { get; set; }
public string Eng { get; set; }
public string Exid { get; set; }
public string Tr { get; set; }

internal static Data2 ParseRow(string row)
{
   var columns = row.Split(';');

   return new Data2()
   {
      Ts = columns[0],
      Te = columns[1],
      Td = columns[2],
      Sa = columns[3],
      Da = columns[4],
      Sp = columns[5],
      Dp = columns[6],
      Pr = columns[7],
      Ra_flg = columns[8],
      Fwd = columns[9],
      Stos = columns[10],
      Ipkt = columns[11],
      Ibyt = columns[12],
      Opkt = columns[13],
      Obyt = columns[14],
      In_ = columns[15],
      Out_ = columns[16],
      Sas = columns[17],
      Das = columns[18],
      Smk = columns[19],
```

```
                Dmk = columns[20],
                Dtos = columns[21],
                Dir = columns[22],
                Nh = columns[24],
                Nhb = columns[24],
                Svln = columns[25],
                Dvln = columns[26],
                Ismc = columns[27],
                Odmc = columns[28],
                Idmc = columns[29],
                Osmc = columns[30],
                Mpls1 = columns[31],
                Mpls2 = columns[32],
                Mpls3 = columns[33],
                Mpls4 = columns[34],
                Mpls5 = columns[35],
                Mpls6 = columns[36],
                Mpls7 = columns[37],
                Mpls8 = columns[38],
                Mpls9 = columns[39],
                Mpls10 = columns[40],
                Cl = columns[41],
                Sl = columns[42],
                Al = columns[43],
                Ra = columns[44],
                Eng = columns[45],
                Exid = columns[46],
                Tr = columns[47]
            };
        }
    }
}
```

4. Результат работы



Form1

Получатель: Иванов Иван Иванович
Сч №: 123456789

Банк получателя: Рога и рога
ИНН: 123456788 КПП: 123456778
БИК: 12345667899
Сч №: 12345678980

Сч №: 120

Дата: 9 мая 2020 г.

Составить счёт

Рога и рога		БИК	12345667899
		Сч. №	12345678980
Банк получателя			
ИНН 123456788	КПП 123456778	Сч. №	123456789
Иванов Иван Иванович			
Получатель			

Счет № 120 от 09.05.2020 г.

Поставщик (Исполнитель): ООО «Лабораторные работы»
Покупатель (Заказчик): Иванов Иван Иванович
Основание:

№	Наименование работ, услуг	Кол-во	Ед	Цена	Сумма
1	Звонки и СМС			62,46	62,46
2	Интернет			3,62	3,62

Итого: 66,08
В том числе НДС: 13,22
Всего к оплате: 79,3

Всего наименований ,
Сумма прописью

Внимание!
Оплата данного счета означает согласие с условиями поставки товара.
Уведомление об оплате обязательно, в противном случае не гарантируется наличие товара на складе.
Товар отпускается по факту прихода денег на р/с Поставщика, самовывозом, при наличии доверенности и паспорта.

Руководитель _____ Бухгалтер _____