Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра «Информатика»

Лабораторная работа №2

по дисциплине

«Проектирование клиентских приложений»

Вариант №4

Выполнил:

студент гр.БЭИ2102

Борисов А. М.

Проверил:

Доцент к.т.н

Халабия Р. Ф.

Москва 2024

**1. Постановка задачи**

Доработать программу из ЛР№1, добавив функционал генерации docx-отчета и диаграмм в зависимости от существующих записей в таблицах.

**2. Разработка схемы алгоритмов**

Разработанные схемы алгоритмов представлены на рисунках 1-4.

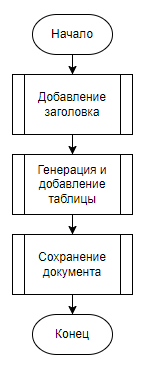


Рисунок 1 – Алгоритм создания отчета

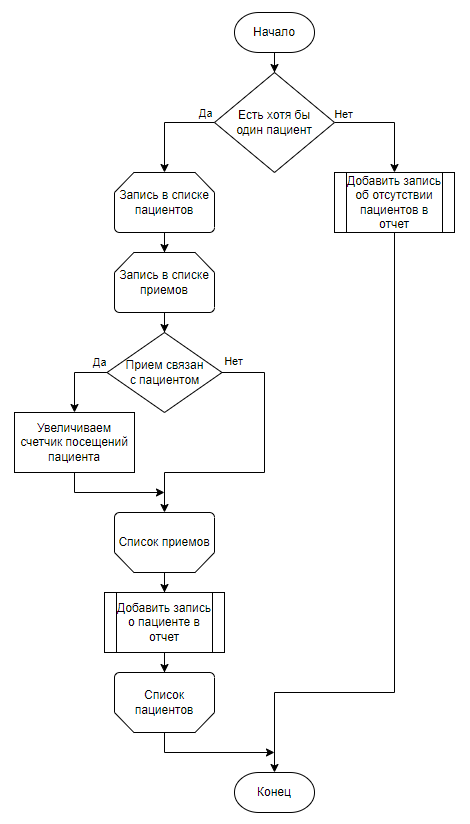


Рисунок 2 – Алгоритм генерации и добавления таблицы записей

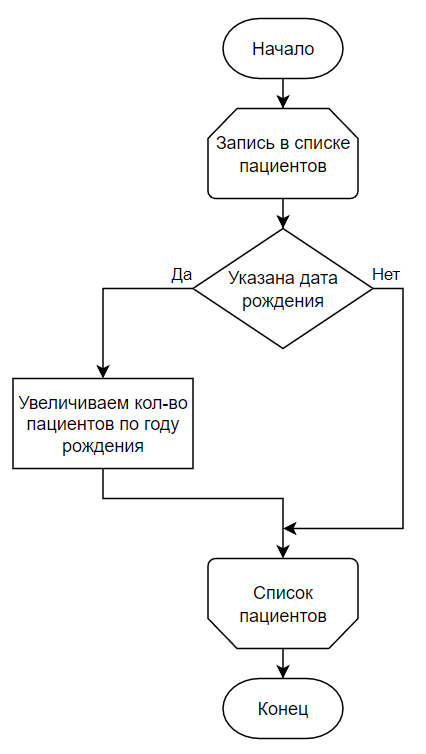


Рисунок 3 – Алгоритм создания диаграммы статистики пациентов по году рождения

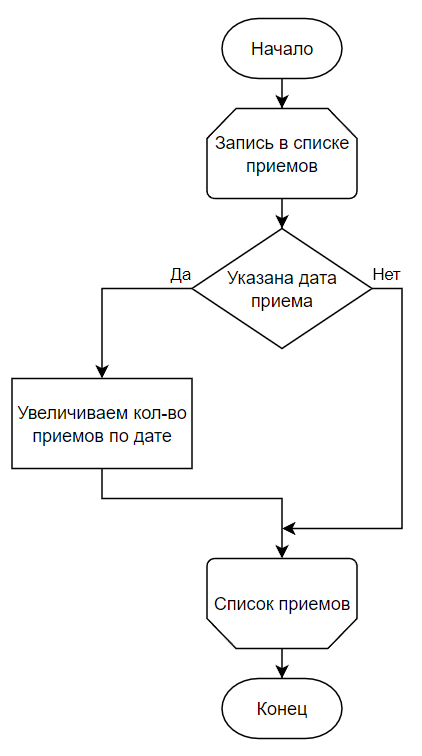


Рисунок 4 – Алгоритм создания диаграммы статистики приемов по датам

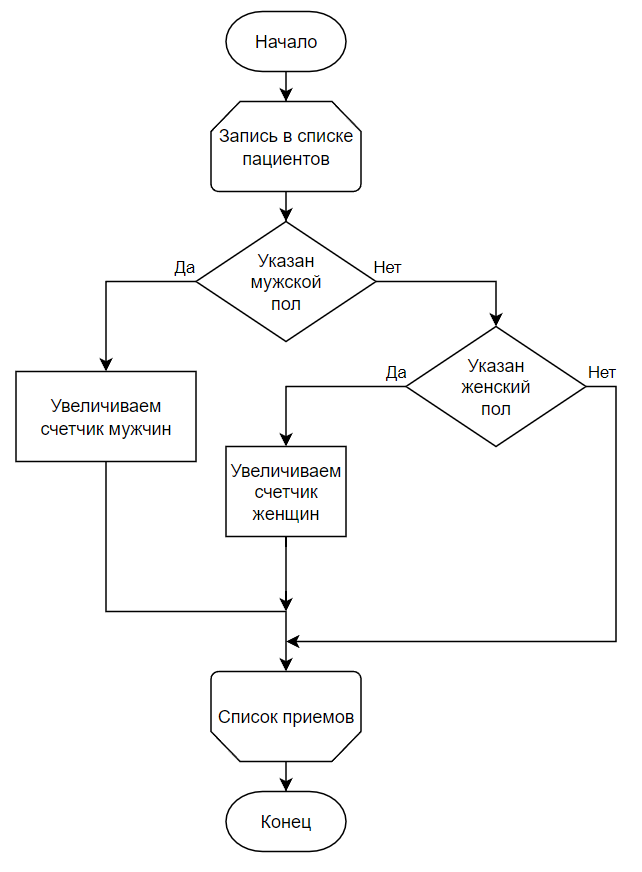


Рисунок 5 – Алгоритм создания диаграммы статистики пациентов по полу

**3. Разработка программы**

**3.1 Разработка графического интерфейса пользователя**

Разработанный интерфейс программы, представлен на рисунке 6.

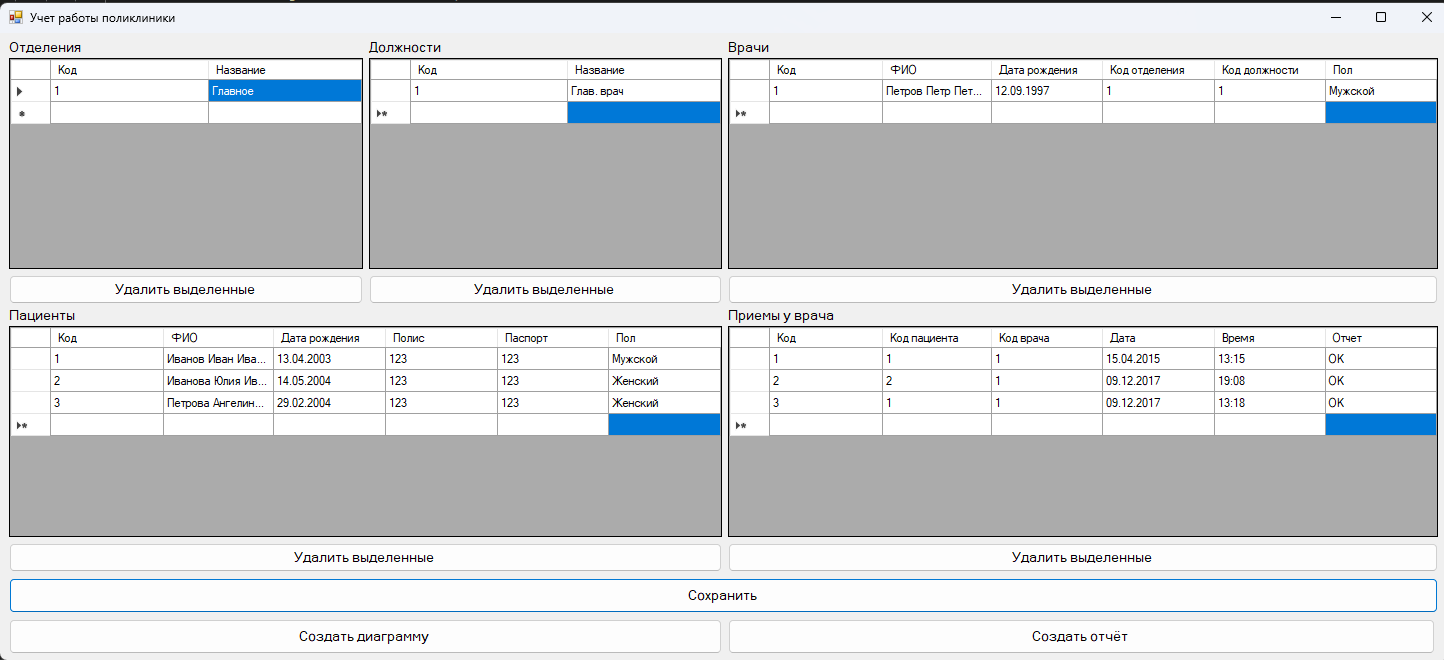


Рисунок 6 – Интерфейс программы

**3.2 Программный код**

Код класса создания диаграмм:

using LiveChartsCore;

using LiveChartsCore.SkiaSharpView.WinForms;

using System;

using System.Collections.Generic;

using LiveChartsCore.SkiaSharpView;

using HLP\_RKP\_LR1.Models;

using System.Linq;

namespace HLP\_RKP\_LR2.Models

{

internal class Charts

{

public static void FormatLinesChart(CartesianChart chart)

{

SortedDictionary<string, int> patients = new SortedDictionary<string, int>();

Patient.items.ForEach(patient =>

{

if (DateTime.TryParse(patient.Birth, out DateTime date))

{

string year = date.Year.ToString();

patients.TryGetValue(year, out int patientsByYear);

patients[year] = patientsByYear + 1;

}

});

int[] values = patients.Values.ToArray();

string[] keys = patients.Keys.ToArray();

chart.Series = new ISeries[]

{

new LineSeries<int>

{

Values = values,

}

};

chart.XAxes = new List<Axis>

{

new Axis

{

Labels = keys,

}

};

chart.ZoomMode = LiveChartsCore.Measure.ZoomAndPanMode.X;

}

public static void FormatHistChart(CartesianChart chart)

{

SortedDictionary<DateTime, int> appointments = new SortedDictionary<DateTime, int>();

Appointment.items.ForEach(appointment =>

{

if (DateTime.TryParse(appointment.Date, out DateTime date))

{

appointments.TryGetValue(date, out int currentAppointments);

appointments[date] = currentAppointments + 1;

}

});

int[] values = appointments.Values.ToArray();

string[] keys = appointments.Keys.ToArray().Select(key => key.ToShortDateString()).ToArray();

chart.Series = new ISeries[]

{

new ColumnSeries<int>

{

Values = values,

}

};

chart.XAxes = new List<Axis>

{

new Axis

{

Labels = keys

}

};

chart.ZoomMode = LiveChartsCore.Measure.ZoomAndPanMode.X;

}

public static void FormatPieChart(PieChart chart)

{

int male = 0;

int female = 0;

Patient.items.ForEach(patient =>

{

if (patient.Sex == "Мужской")

{

male++;

}

else if (patient.Sex == "Женский")

{

female++;

}

});

chart.Series = new ISeries[]

{

new PieSeries<double> { Values = new double[] { male }, Name = "Мужской" },

new PieSeries<double> { Values = new double[] { female }, Name = "Женский" },

};

}

}

}

Код класса создания отчета:

using System;

using DocumentFormat.OpenXml;

using DocumentFormat.OpenXml.Packaging;

using DocumentFormat.OpenXml.Wordprocessing;

using HLP\_RKP\_LR1.Models;

namespace HLP\_RKP\_LR2.Models

{

public class WordDocumentCreator

{

private const string DEFAULT\_FONT\_SIZE = "28";

public static void CreateDocument(string filePath, string fileName)

{

using (WordprocessingDocument wordDocument = WordprocessingDocument.Create(filePath + fileName, WordprocessingDocumentType.Document))

{

Body body = CreateBody(wordDocument);

AddBold(body, "Отчет о пациентах");

AddTable(body);

SaveDocument(wordDocument);

}

}

private static Body CreateBody(WordprocessingDocument doc)

{

MainDocumentPart mainPart = doc.AddMainDocumentPart();

mainPart.Document = new Document();

Body body = mainPart.Document.AppendChild(new Body());

return body;

}

private static void SaveDocument(WordprocessingDocument doc)

{

doc.MainDocumentPart.Document.Save();

}

private static Paragraph GetParagraph(string content, OpenXmlElement[] runParams, OpenXmlElement[] paragraphParams)

{

RunProperties runProps = new RunProperties();

Run run = new Run();

runProps.Append(runParams);

run.Append(runProps);

run.Append(new Text(content));

Paragraph p = new Paragraph();

ParagraphProperties paragraphProps = new ParagraphProperties();

paragraphProps.Append(paragraphParams);

p.Append(paragraphProps);

p.Append(run);

return p;

}

private static OpenXmlElement[] GetParams(params OpenXmlElement[] props)

{

return props;

}

private static FontSize GetFontSize(string fontSize)

{

return new FontSize() { Val = fontSize };

}

private static Justification GetCentered()

{

return new Justification() { Val = JustificationValues.Center };

}

private static void AddBold(Body body, string content, string fontSize = DEFAULT\_FONT\_SIZE)

{

Paragraph p = GetParagraph(content, GetParams(GetFontSize(fontSize), new Bold()), GetParams());

body.AppendChild(p);

}

private static void AddPlain(Body body, string content, string fontSize = DEFAULT\_FONT\_SIZE)

{

Paragraph p = GetParagraph(content, GetParams(GetFontSize(fontSize)), GetParams());

body.AppendChild(p);

}

private static void AddTable(Body body, string fontSize = DEFAULT\_FONT\_SIZE)

{

Table table = GetTable();

if (Patient.items.Count == 0)

{

AddPlain(body, "Пока что нет записей о пациентах...");

return;

}

AddRow(table, fontSize, "ФИО", "Дата рождения", "Пол", "Кол-во приемов");

Patient.items.ForEach(patient =>

{

int appointments = 0;

Appointment.items.ForEach(appointment =>

{

if (appointment.PatientID == patient.ID)

{

appointments++;

}

});

AddRow(table, fontSize, patient.Name, patient.Birth, patient.Sex, appointments.ToString());

});

body.AppendChild(table);

}

private static Table GetTable()

{

Table table = new Table();

TableProperties tblProp = new TableProperties(

new TableWidth() { Width = "100%", Type = TableWidthUnitValues.Pct },

new TableBorders(

new TopBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new BottomBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new LeftBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new RightBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new InsideHorizontalBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 },

new InsideVerticalBorder() { Val = new EnumValue<BorderValues>(BorderValues.Single), Size = 12 }

)

);

table.AppendChild(tblProp);

return table;

}

private static void AddRow(Table table, string fontSize = DEFAULT\_FONT\_SIZE, params string[] cellTexts)

{

TableRow row = new TableRow();

foreach (string cellText in cellTexts)

{

AddCell(row, cellText, fontSize);

}

table.Append(row);

}

private static void AddCell(TableRow row, string text, string fontSize = DEFAULT\_FONT\_SIZE)

{

TableCell tc = new TableCell();

Paragraph p = GetParagraph(text, GetParams(GetFontSize(fontSize)), GetParams(GetCentered()));

tc.Append(p);

row.Append(tc);

}

}

}

**4. Результат работы программы**

Результаты работы программы представлен на рисунках 7-8.

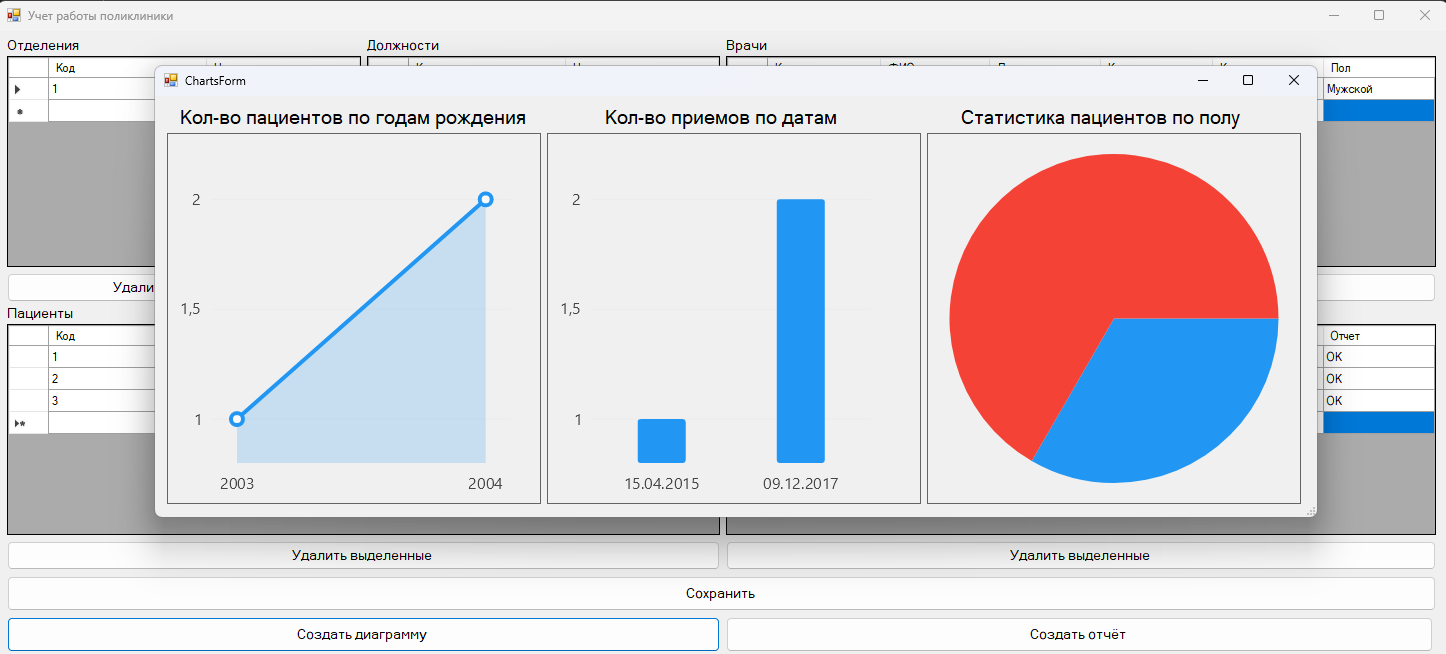


Рисунок 7 – Результат создания диаграмм

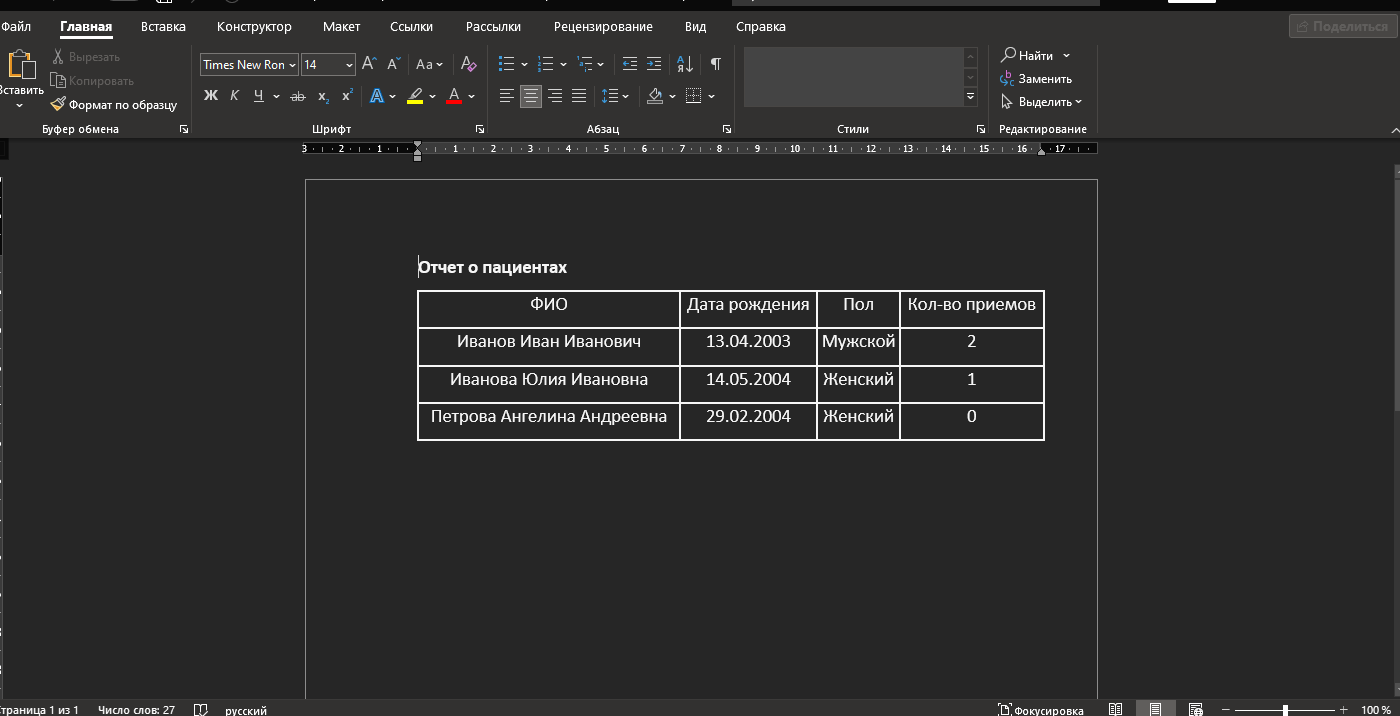


Рисунок 8 – Результат создания отчета