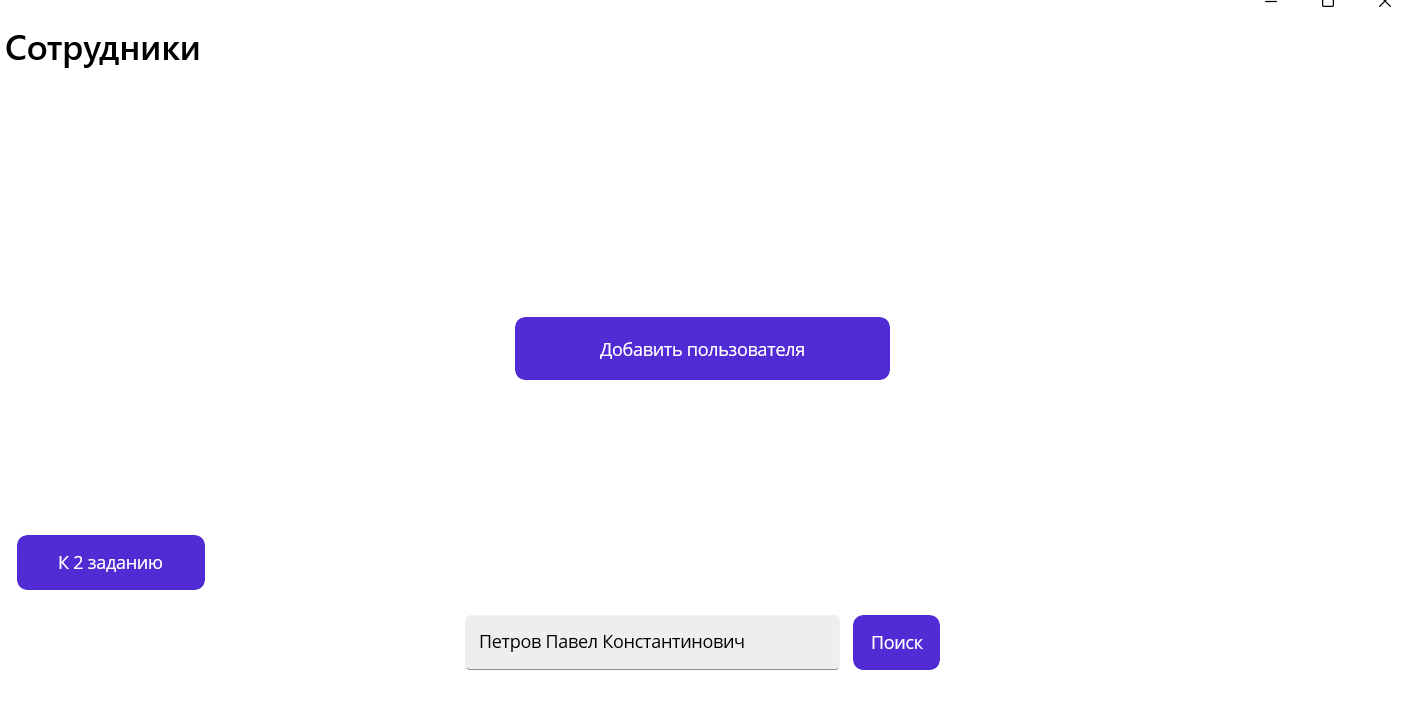
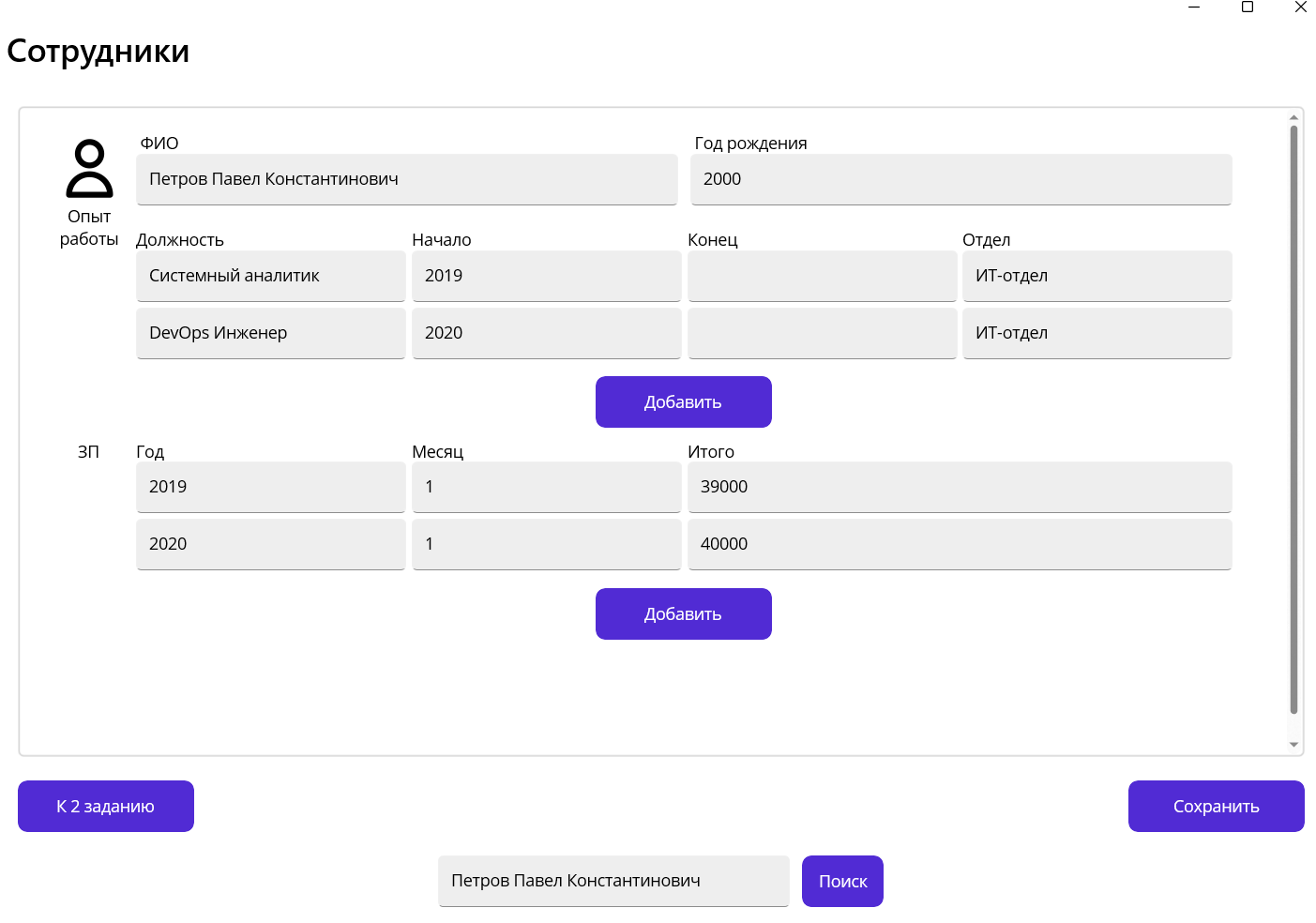
1) поиск сотрудника по фамилии, выводить историю о его трудовой деятельности (с возможностью сортировки по году начала работы) и начислениях заработной платы сотрудника (с возможностью указания периода) с расчетом максимального, минимального и среднего значения.





public void Find(string q)

{

try

{

ViewWorker = Workers.Where(worker => worker.Name == q).First();

}

catch(Exception ex)

{

Application.Current.MainPage.DisplayAlert("Ошибка", "Пользователь с таким ФИО не найден.", "Ладно");

return;

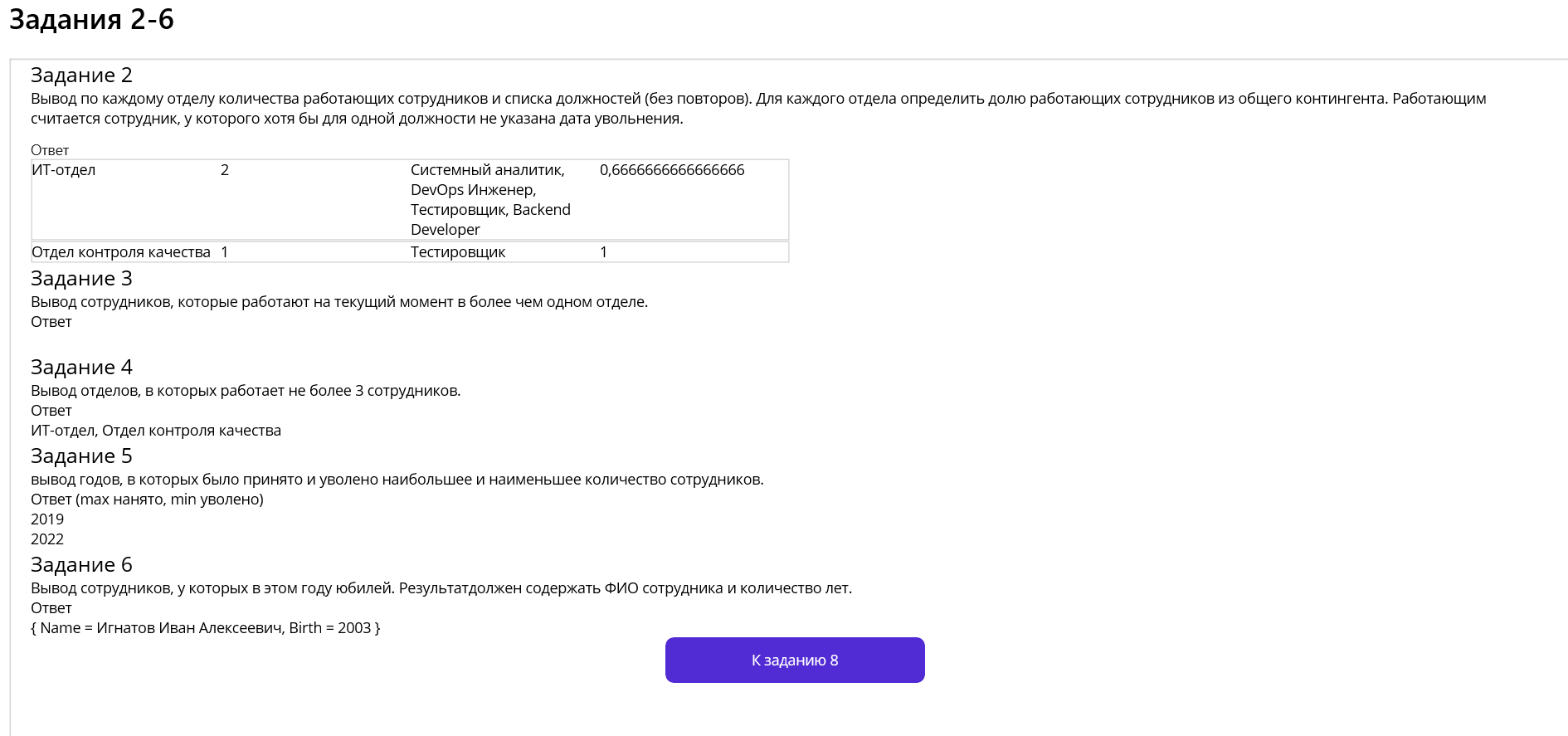
}

IsActive\_ScrollView = true;

IsActive\_AddWorkerButton = false;

}

2) вывод по каждому отделу количества работающих сотрудников и списка должностей (без повторов). Для каждого отдела определить долю работающих сотрудников из общего контингента. Работающим считается сотрудник, у которого хотя бы для одной должности не указана дата увольнения.



var workerInfo\_List =

from worker in Workers

from workerInfo in worker.WorkExperience

select new

{

workerInfo,

worker

};

var workerInfo\_byDepartment\_List =

from wil in workerInfo\_List

group wil by wil.workerInfo.Department;

var res1 =

workerInfo\_byDepartment\_List.Select(info =>

new CountOfWorkingWorkers

(

department : info.Key,

totalCount : info.DistinctBy(i => i.worker.Name).Count(),

count : info.Where(i => i.workerInfo.Finish == "").DistinctBy(i => i.worker).Count(),

listOfWorks : String.Join(", ", info.Select(i => i.workerInfo.Name).Distinct().ToArray())

));

foreach (var c in res1)

{

c.Activity = (double)c.Count / c.TotalCount;

CowList.Add(c);

}

3) вывод сотрудников, которые работают на текущий момент в более чем одном отделе.

// #3

WorkersThatWorkOnMultipleJobs = String.Join(", ", workerInfo\_List.GroupBy(wil => wil.worker.Name).Select(w => new

{

Name = w.Key,

DepsCount = w.Where(wentry => wentry.workerInfo.Finish == "").DistinctBy(wentry => wentry.workerInfo.Department).Count()

}).Where(r => r.DepsCount > 1).Select(r => r.Name).ToArray());

4) вывод отделов, в которых работает не более 3 сотрудников.

DepartmentsThatHaveMultipleWorkers = String.Join(", ", workerInfo\_List.GroupBy(wil => wil.workerInfo.Department).Select(w => new

{

Department = w.Key,

WorkersCount = w.Where(wentry => wentry.workerInfo.Finish == "").DistinctBy(wentry => wentry.worker.Name).Count()

}).Where(r => r.WorkersCount <= 3).Select(r => r.Department).ToArray());

5) вывод годов, в которых было принято и уволено наибольшее и наименьшие количество сотрудников.

YearSummonMax = workerInfo\_List.GroupBy(wil => wil.workerInfo.Start).Select(w => new

{

Year = w.Key,

WorkersCount = w.DistinctBy(wentry => wentry.worker.Name).Count()

}).OrderByDescending(r => r.WorkersCount).First().Year.ToString();

6) вывод сотрудников, у которых в этом году юбилей. Результат должен содержать ФИО сотрудника и количество лет.

YearFiredMin = workerInfo\_List.GroupBy(wil => wil.workerInfo.Finish).Select(w => new

{

Year = w.Key,

WorkersCount = w.Where(wentry => wentry.workerInfo.Finish != "").DistinctBy(wentry => wentry.worker.Name).Count()

}).OrderByDescending(r => r.WorkersCount).First().Year.ToString();

7) Экспорт данных в XML-файл:

var res2 = workerInfo\_byDepartment\_List.Select(info =>

new

{

department = info.Key,

totalCount = info.DistinctBy(i => i.worker.Name).Count(),

countYoung = info.DistinctBy(i => i.worker.Name).Where(i => (currentYear - int.Parse(i.worker.BirthString)) < 30).Count()

});

XmlDocument res = new XmlDocument();

XmlElement xDepartments = res.CreateElement("Отделы");

foreach (var item in res2)

{

XmlElement xDepartment = res.CreateElement("Отдел");

xDepartment.SetAttribute("Название", item.department);

XmlElement xCount = res.CreateElement("Количество\_работающих\_сотрудников");

xCount.InnerText = item.totalCount.ToString();

XmlElement xCountYoung = res.CreateElement("Количество\_работающих\_сотрудников\_молодёжь");

xCountYoung.InnerText = item.countYoung.ToString();

xDepartment.AppendChild(xCount);

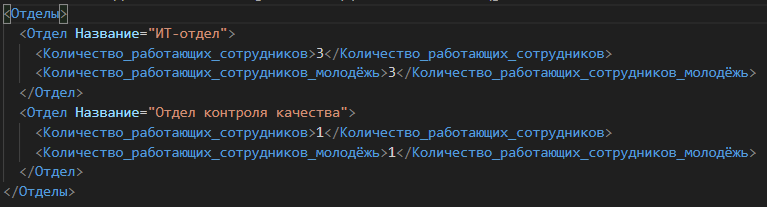
xDepartment.AppendChild(xCountYoung);

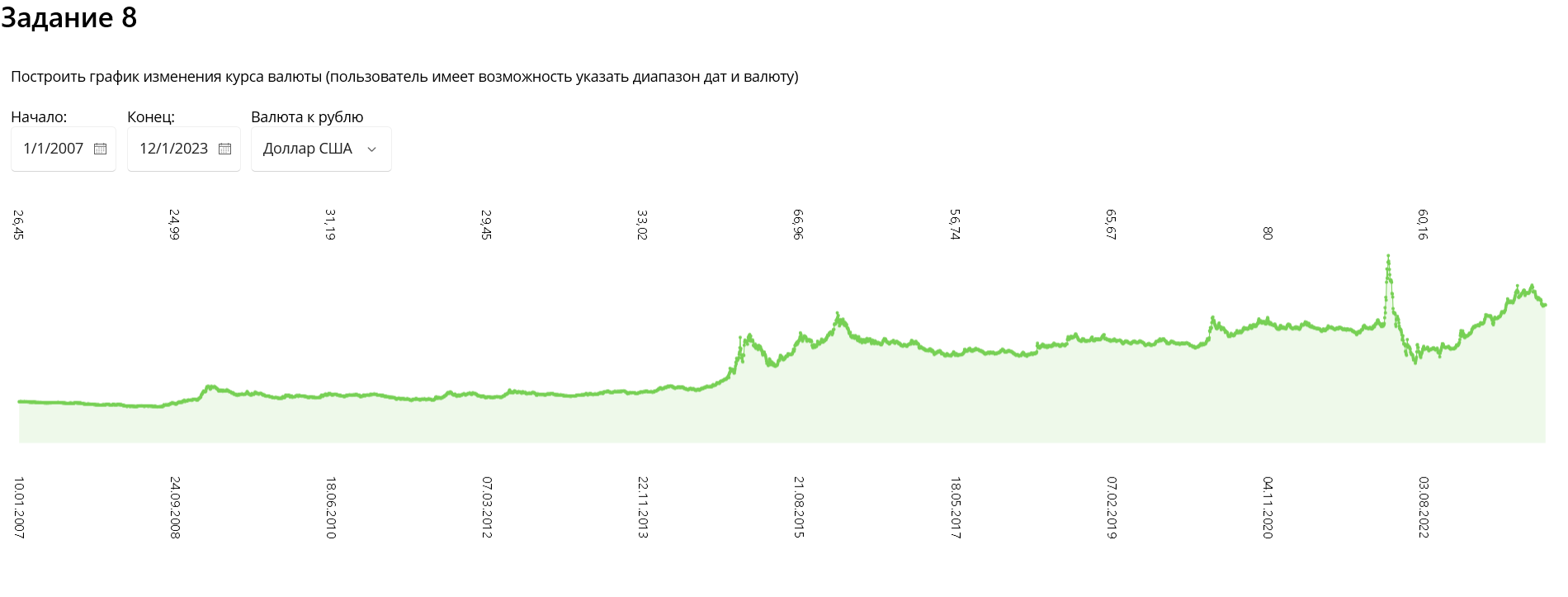
xDepartments.AppendChild(xDepartment);

}

res.AppendChild(xDepartments);

SaveToFile("res7.xml", res);





public void DrawChart(string startDate, string finishDate, string? moneyCode /\*доллар\*/)

{

if (moneyCode == null || moneyCode == "") moneyCode = "R01235";

XmlDocument Prices = new XmlDocument();

Encoding.RegisterProvider(CodePagesEncodingProvider.Instance); // добавляет кодировку windows-1251

Prices.Load($"https://www.cbr.ru/scripts/XML\_dynamic.asp?date\_req1={startDate}&date\_req2={finishDate}&VAL\_NM\_RQ={moneyCode}");

records = [];

XmlElement xValCurs = Prices.DocumentElement;

foreach (XmlElement xRecord in xValCurs)

{

try

{

Record record = new Record(xRecord.GetAttribute("Date"),

double.Parse(xRecord.SelectSingleNode("Nominal").InnerText),

double.Parse(xRecord.SelectSingleNode("Value").InnerText),

double.Parse(xRecord.SelectSingleNode("VunitRate").InnerText)

);

records.Add(record);

}

catch

{

}

}

if (records.Count == 0)

{

Application.Current.MainPage.DisplayAlert("Ошибка", "ЦБ не привёл актуальных данных по курсу валюты с кодом "

+ moneyCode, "Ладно");

return;

}

entries = [];

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

{

ChartEntry entry = new ChartEntry((float)records[i].value);

if (i % (records.Count / 10 + 10) == 0)

{

entry.ValueLabel = double.Round(records[i].value,2).ToString();

entry.Label = records[i].date;

}

entry.Color = SKColor.Parse("#77d055");

entries = entries.Append(entry).ToArray();

}

chartView.Chart = new LineChart()

{

Entries = entries,

LineSize = 1,

LabelColor = SKColor.Parse("#000000"),

PointSize = 4f,

LineMode = LineMode.Straight,

IsAnimated = false

};

}