Содержание

[Задание 1-4 2](#_Toc149385074)

[Структура проекта: 3](#_Toc149385075)

[Код сущности Note.cs 3](#_Toc149385076)

[Код сущности Contact.cs 3](#_Toc149385077)

[Код NoteController.cs 4](#_Toc149385078)

[Код ContactController.cs 5](#_Toc149385079)

[Код обработчика OutputHandler.cs 6](#_Toc149385080)

[Код обработчика SavingFileHandler.cs 7](#_Toc149385081)

[Код Program.cs 9](#_Toc149385082)

[Демонстрация работы. Консоль 12](#_Toc149385083)

[Демонстрация папки проекта 14](#_Toc149385084)

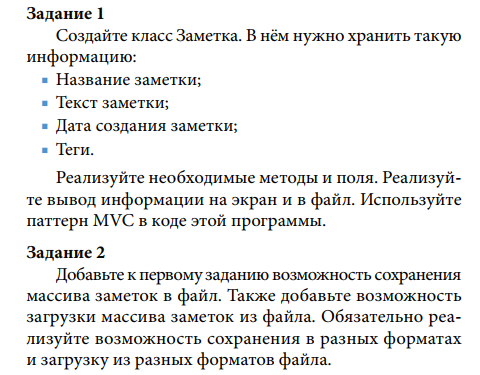
[Содержание файла Note.json 14](#_Toc149385085)

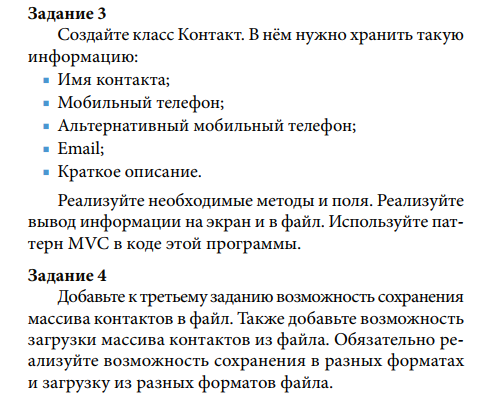
[Содержание файла Note.txt 15](#_Toc149385086)

[Содержание файла Note.xml 15](#_Toc149385087)

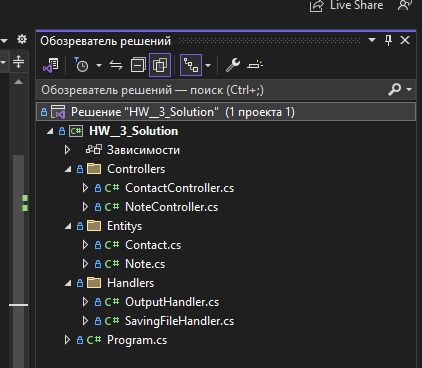
[Содержание файла NoteName.txt 15](#_Toc149385088)

Задание 1-4





Структура проекта:



Код сущности Note.cs

namespace HW\_\_3\_Solution.Entity

{

[Serializable]

public class Note

{

public string NoteName { get; set; }

public string NoteText { get; set; }

public DateTime DateCreation { get; set; }

public List<string>? Tags { get; set; }

public override string ToString()

{

return $"Название: {NoteName}\n" +

$"Текст заметки: {NoteText}\n" +

$"Дата: {DateCreation}\n" +

$"Теги: {(Tags is null ? string.Empty : string.Join(", ", Tags))}";

}

}

}

Код сущности Contact.cs

namespace HW\_\_3\_Solution.Entitys

{

[Serializable]

public class Contact

{

public string Name { get; set; }

public string MobilePhone { get; set; }

public string? AltMobilePhone { get; set; }

public string? Email { get; set; }

public string? Description { get; set; }

public override string ToString()

{

return $"Имя контакта: {Name}\n" +

$"Мобильный телефон: {MobilePhone}\n" +

$"Альтернативный мобильный телефон: {(AltMobilePhone is null ? string.Empty: AltMobilePhone)}\n" +

$"Email: {(Email is null ? string.Empty : Email)}\n" +

$"Описание: {(Description is null ? string.Empty : Description)}";

}

}

}

Код NoteController.cs

namespace HW\_\_3\_Solution.Controllers

{

public class NoteController

{

private List<Note> notes = new List<Note>();

public void AddNote(Note note)

{

notes.Add(note);

}

public void RemoveNote(Note note)

{

notes.Remove(note);

}

public void ClearNotes()

{

notes.Clear();

}

public void AddListNote(List<Note> list)

{

notes.AddRange(list);

}

public List<string> GetListNoteNames()

{

List<string> notesNamelist = new List<string>();

foreach (Note note in notes)

{

if(!notesNamelist.Contains(note.NoteName))

notesNamelist.Add(note.NoteName);

}

return notesNamelist;

}

public void ShowInformation(IOutputHandler handler)

{

handler.Show(notes);

}

public void ShowInformationNoteNames(IOutputHandler handler)

{

handler.Show(this.GetListNoteNames());

}

public void SaveInformation(ISavingFileHandler handler)

{

handler.Save(notes);

}

public void LoadInformation(ISavingFileHandler handler)

{

notes = handler.Load();

}

public int GetCount()

{

return notes.Count;

}

}

}

Код ContactController.cs

namespace HW\_\_3\_Solution.Controllers

{

public class ContactController

{

private List<Contact> contacts = new List<Contact>();

public void AddContact(Contact contact)

{

contacts.Add(contact);

}

public void RemoveContact(Contact contact)

{

contacts.Remove(contact);

}

public void ClearContacts()

{

contacts.Clear();

}

public void AddListContacts(List<Contact> contactsList)

{

contacts.AddRange(contactsList);

}

public List<string> GetListContactsNames()

{

List<string> Namelist = new List<string>();

foreach (Contact contact in contacts)

{

if (!Namelist.Contains(contact.Name))

Namelist.Add(contact.Name);

}

return Namelist;

}

public void ShowInformation(IOutputHandler handler)

{

handler.Show(contacts);

}

public void ShowInformationContactsNames(IOutputHandler handler)

{

handler.Show(this.GetListContactsNames());

}

public void SaveInformation(ISavingFileHandler handler)

{

handler.Save(contacts);

}

public void LoadInformation(ISavingFileHandler handler)

{

contacts = handler.LoadContacts();

}

public int GetCount()

{

return contacts.Count;

}

}

}

Код обработчика OutputHandler.cs

namespace HW\_\_3\_Solution.Handlers

{

public interface IOutputHandler

{

public void Show(List<Note> notes);

public void Show(List<string> strings);

public void Show(List<Contact> contactsList);

}

public class ConsoleOutputHandler : IOutputHandler

{

public void Show(List<Note> notes)

{

foreach (var item in notes)

{

Console.WriteLine(item);

}

}

public void Show(List<string> strings)

{

foreach(var item in strings)

{

Console.Write(item + "\n");

}

Console.WriteLine();

}

public void Show(List<Contact> contactsList)

{

foreach (var item in contactsList)

{

Console.WriteLine(item);

}

}

}

public class TxtFileOutputHandler : IOutputHandler

{

private string \_filePath;

private string \_defaultFilePath = "text.txt";

public TxtFileOutputHandler()

{

\_filePath = \_defaultFilePath;

}

public TxtFileOutputHandler(string filePath)

{

\_filePath = filePath;

}

public void Show(List<Note> notes)

{

using (StreamWriter sw = new StreamWriter(\_filePath, false, Encoding.UTF8))

{

foreach (Note n in notes)

{

sw.WriteLine(n);

sw.Write("\n");

}

}

}

public void Show(List<string> strings)

{

using (StreamWriter sw = new StreamWriter(\_filePath, false, Encoding.UTF8))

{

foreach (string s in strings)

{

sw.WriteLine(s);

}

}

}

public void Show(List<Contact> contactsList)

{

using (StreamWriter sw = new StreamWriter(\_filePath, false, Encoding.UTF8))

{

foreach (Contact c in contactsList)

{

sw.WriteLine(c);

sw.Write("\n");

}

}

}

}

}

Код обработчика SavingFileHandler.cs

namespace HW\_\_3\_Solution.Handlers

{

public interface ISavingFileHandler

{

public void Save(List<Note> notes);

public List<Note> Load();

public void Save(List<Contact> contactList);

public List<Contact> LoadContacts();

}

public class JsonSavingFileHandler : ISavingFileHandler

{

private string \_filePath;

public JsonSavingFileHandler(string filePath)

{

\_filePath = filePath;

}

public List<Note> Load()

{

string json = File.ReadAllText(\_filePath);

return JsonSerializer.Deserialize<List<Note>>(json);

}

public List<Contact> LoadContacts()

{

string json = File.ReadAllText(\_filePath);

return JsonSerializer.Deserialize<List<Contact>>(json);

}

public void Save(List<Note> objects)

{

var options = new JsonSerializerOptions { WriteIndented = true };

string json = JsonSerializer.Serialize(objects, options);

File.WriteAllText(\_filePath, json);

}

public void Save(List<Contact> contactList)

{

var options = new JsonSerializerOptions { WriteIndented = true };

string json = JsonSerializer.Serialize(contactList, options);

File.WriteAllText(\_filePath, json);

}

}

public class XmlSavingFileHandler : ISavingFileHandler

{

private string \_filePath;

public XmlSavingFileHandler(string filePath)

{

\_filePath = filePath;

}

public List<Note> Load()

{

List<Note> objects;

XmlSerializer xmlSerializer = new XmlSerializer(typeof(List<Note>));

using (FileStream fs = new FileStream(\_filePath, FileMode.Open))

{

objects = (List<Note>)xmlSerializer.Deserialize(fs);

}

return objects;

}

public List<Contact> LoadContacts()

{

List<Contact> objects;

XmlSerializer xmlSerializer = new XmlSerializer(typeof(List<Contact>));

using (FileStream fs = new FileStream(\_filePath, FileMode.Open))

{

objects = (List<Contact>)xmlSerializer.Deserialize(fs);

}

return objects;

}

public void Save(List<Note> objects)

{

XmlSerializer xmlSerializer = new XmlSerializer(typeof(List<Note>));

using (FileStream fs = new FileStream(\_filePath, FileMode.Create))

{

xmlSerializer.Serialize(fs, objects);

}

}

public void Save(List<Contact> contactList)

{

XmlSerializer xmlSerializer = new XmlSerializer(typeof(List<Contact>));

using (FileStream fs = new FileStream(\_filePath, FileMode.Create))

{

xmlSerializer.Serialize(fs, contactList);

}

}

}

}

Код Program.cs

using HW\_\_3\_Solution.Controllers;

using HW\_\_3\_Solution.Entity;

using HW\_\_3\_Solution.Entitys;

using HW\_\_3\_Solution.Handlers;

Console.WriteLine("Задание 1--2\n");

NoteController noteController = new NoteController();

noteController.AddNote(new Note()

{

NoteName = "Test",

DateCreation = DateTime.Now,

NoteText = "Test notes start this",

Tags = new List<string>()

{

"test", "notes for test"

}

});

noteController.AddNote(new Note()

{

NoteName = "Test2",

DateCreation = DateTime.Now,

NoteText = "Lorem Ipsum is simply dummy text of the printing and typesetting industry.",

Tags = new List<string>()

{

"test", "New"

}

});

noteController.AddNote(new Note()

{

NoteName = "Test21",

DateCreation = DateTime.Now.AddDays(-1),

NoteText = "Test21-----1daysy",

Tags = new List<string>()

{

"sadasd"

}

});

noteController.AddNote(new Note()

{

NoteName = "11Test21",

DateCreation = DateTime.Now.AddDays(-12),

NoteText = "Test12-----12daysy",

});

Console.WriteLine("\tВывод полной информации о всех заметках в консоль и файл Note.txt");

noteController.ShowInformation(new ConsoleOutputHandler());

noteController.ShowInformation(new TxtFileOutputHandler("Note.txt"));

Console.WriteLine("\n\tВывод информации о именах заметок в консоль и файл");

noteController.ShowInformationNoteNames(new ConsoleOutputHandler());

noteController.ShowInformationNoteNames(new TxtFileOutputHandler("NoteName.txt"));

Console.WriteLine("\tСохранение информации в разных форматах (Note.json)");

noteController.SaveInformation(new JsonSavingFileHandler("Note.json"));

noteController.ClearNotes();

Console.WriteLine($"Количество заметок в контроллере: {noteController.GetCount()}");

Console.WriteLine("\n\tЗагрузка информации из разных форматов (Note.json)");

noteController.LoadInformation(new JsonSavingFileHandler("Note.json"));

noteController.ShowInformation(new ConsoleOutputHandler());

Console.WriteLine("\n\tСохранение информации в разных форматах (Note.xml)");

noteController.SaveInformation(new XmlSavingFileHandler("Note.xml"));

noteController.ClearNotes();

Console.WriteLine($"Количество заметок в контроллере: {noteController.GetCount()}");

Console.WriteLine("\n\tЗагрузка информации из разных форматов (Note.xml)");

noteController.LoadInformation(new XmlSavingFileHandler("Note.xml"));

noteController.ShowInformation(new ConsoleOutputHandler());

////-------------------------------------------------------------------------------------------------------

///----------------------------------------Задание 3---4---------------------------------------------------

Console.WriteLine("\n\nДля отображение задание 3-4 нажмите любую клавишу");

Console.ReadKey();

Console.Clear();

Console.WriteLine("Задание 3--4\n");

ContactController contactController = new ContactController();

contactController.AddContact(new Contact() { Name = "Василий Петрович", MobilePhone = "+380953337800" });

contactController.AddContact(new Contact() { Name = "George Kush", MobilePhone = "+182953337800", AltMobilePhone = "+1233394213", Email = "Gkush@mg.com", Description = "PR Director MFG" });

contactController.AddContact(new Contact() { Name = "Сергей Иванович", MobilePhone = "+380953337800" });

Console.WriteLine("\tВывод полной информации о всех контактах в консоль и файл Contact.txt");

contactController.ShowInformation(new ConsoleOutputHandler());

contactController.ShowInformation(new TxtFileOutputHandler("Contact.txt"));

Console.WriteLine("\n\tВывод информации о именах контактов в консоль и файл");

contactController.ShowInformationContactsNames(new ConsoleOutputHandler());

contactController.ShowInformationContactsNames(new TxtFileOutputHandler("ContactName.txt"));

Console.WriteLine("\tСохранение информации в разных форматах (Contact.json)");

contactController.SaveInformation(new JsonSavingFileHandler("Contact.json"));

contactController.ClearContacts();

Console.WriteLine($"Количество контактов в контроллере: {contactController.GetCount()}");

Console.WriteLine("\n\tЗагрузка информации из разных форматов (Contact.json)");

contactController.LoadInformation(new JsonSavingFileHandler("Contact.json"));

contactController.ShowInformation(new ConsoleOutputHandler());

Console.WriteLine("\n\tСохранение информации в разных форматах (Contact.xml)");

contactController.SaveInformation(new XmlSavingFileHandler("Contact.xml"));

contactController.ClearContacts();

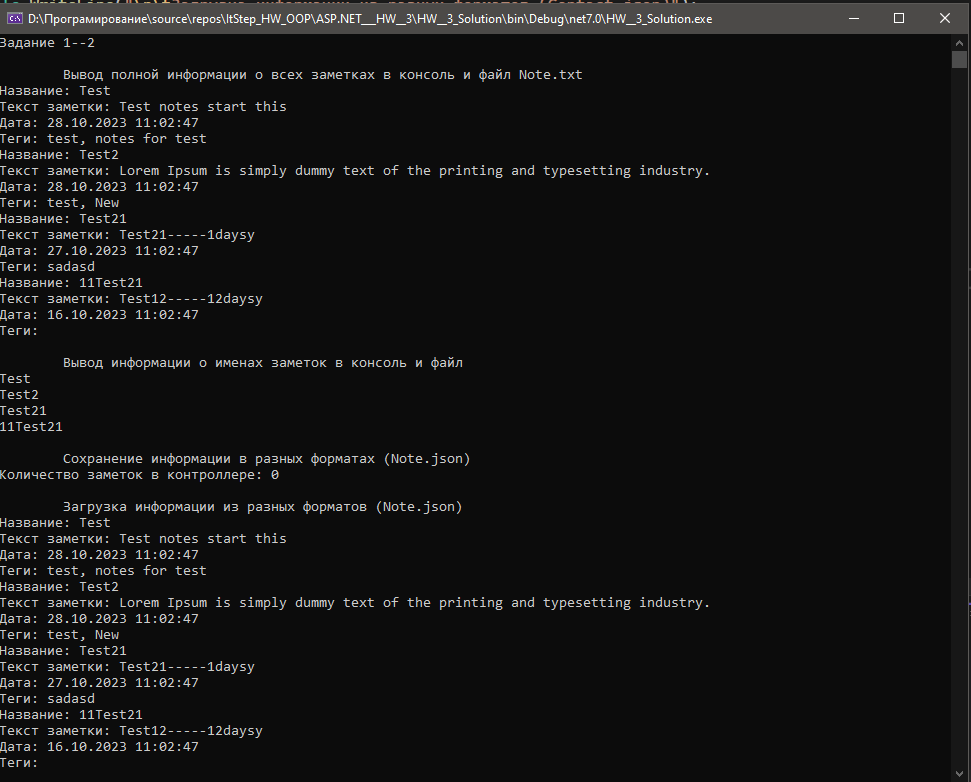
Console.WriteLine($"Количество контактов в контроллере: {contactController.GetCount()}");

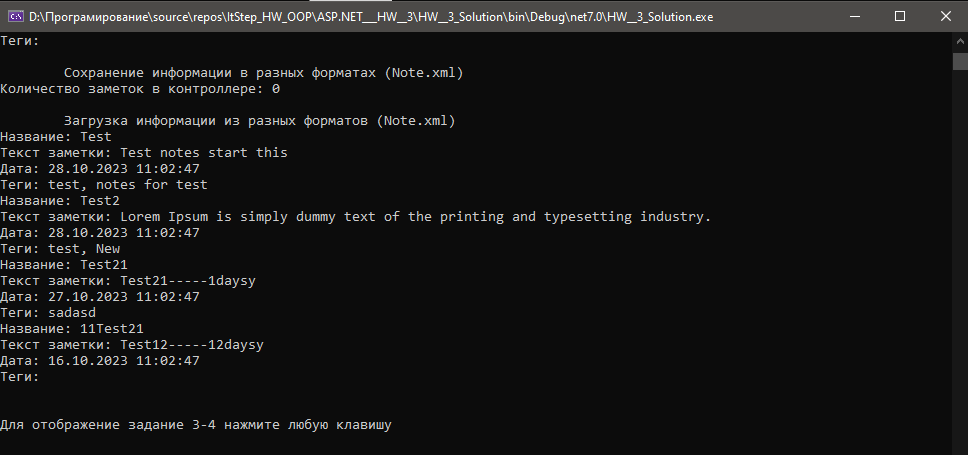
Console.WriteLine("\n\tЗагрузка информации из разных форматов (Contact.xml)");

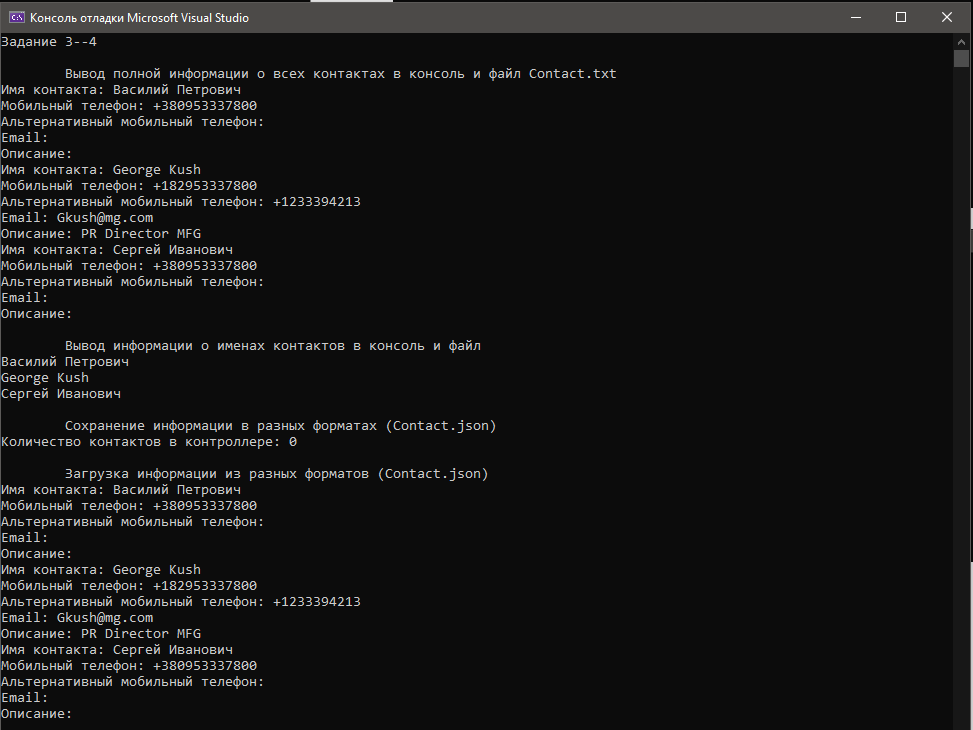
contactController.LoadInformation(new XmlSavingFileHandler("Contact.xml"));

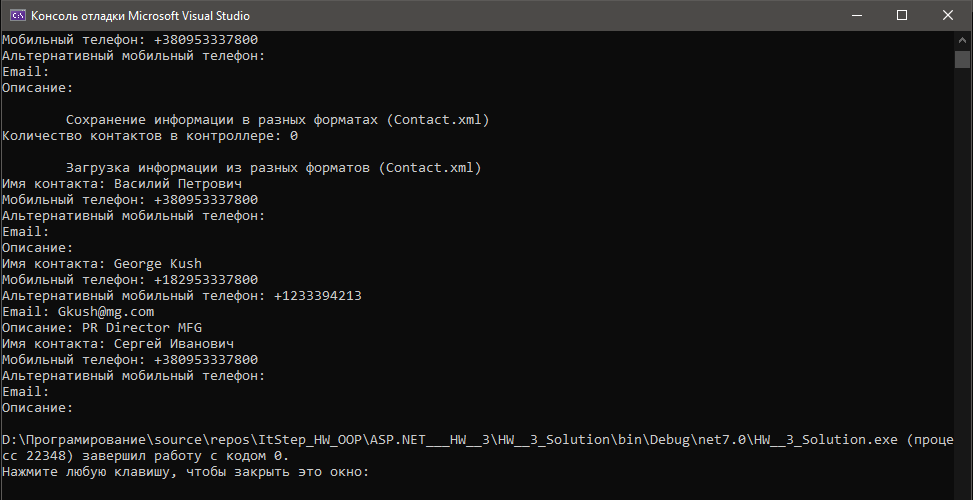
contactController.ShowInformation(new ConsoleOutputHandler());

Демонстрация работы. Консоль

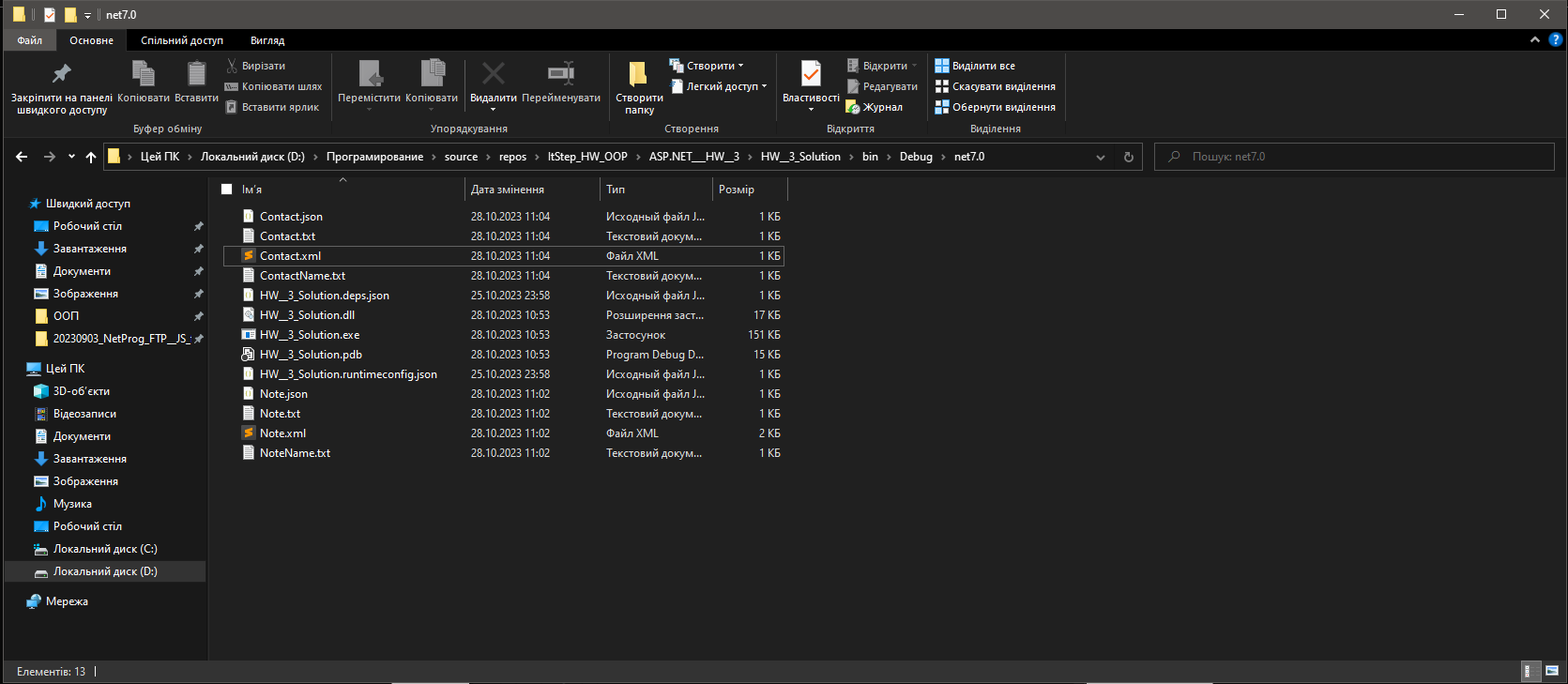




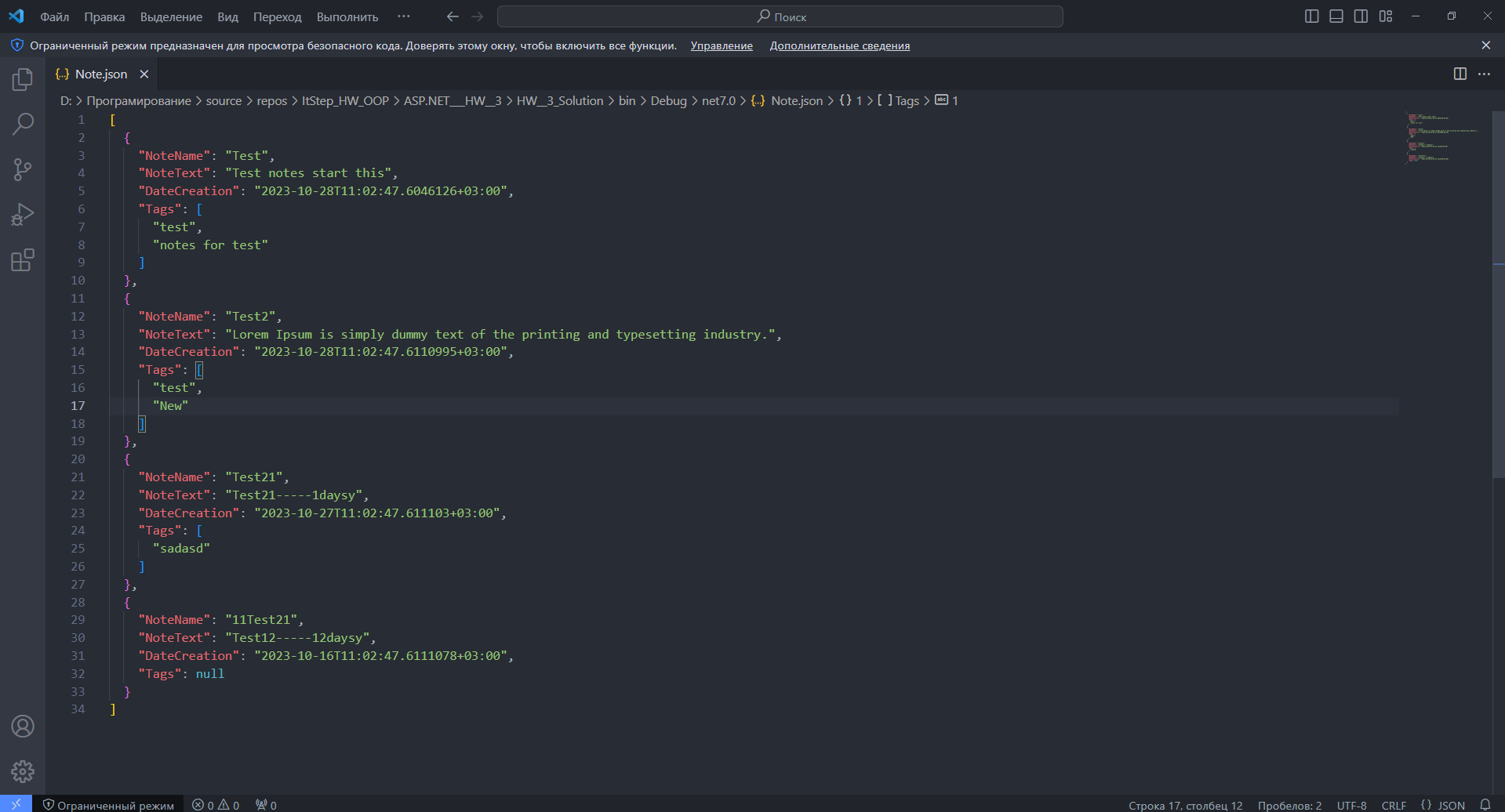




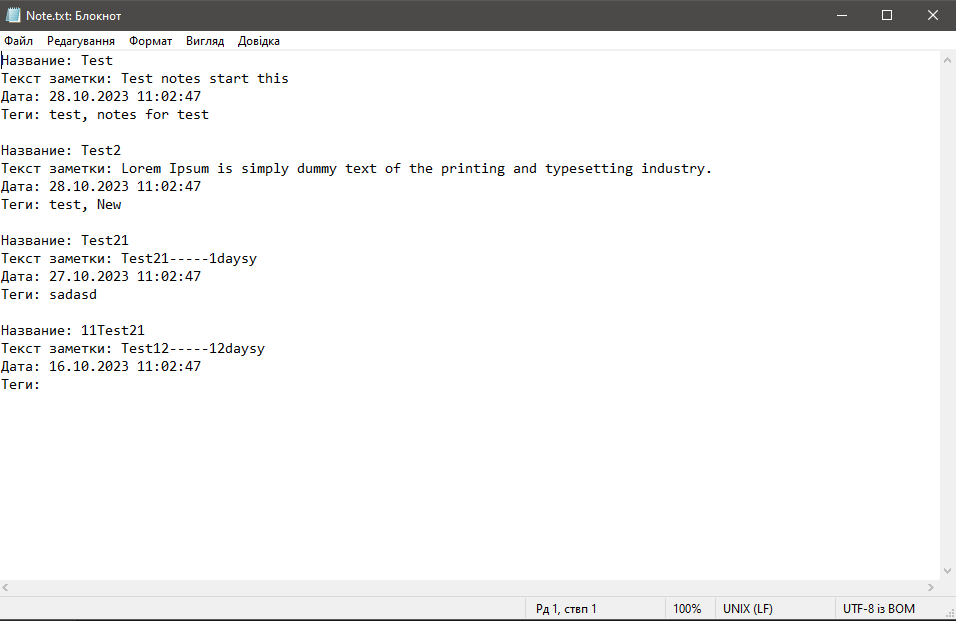
Демонстрация папки проекта



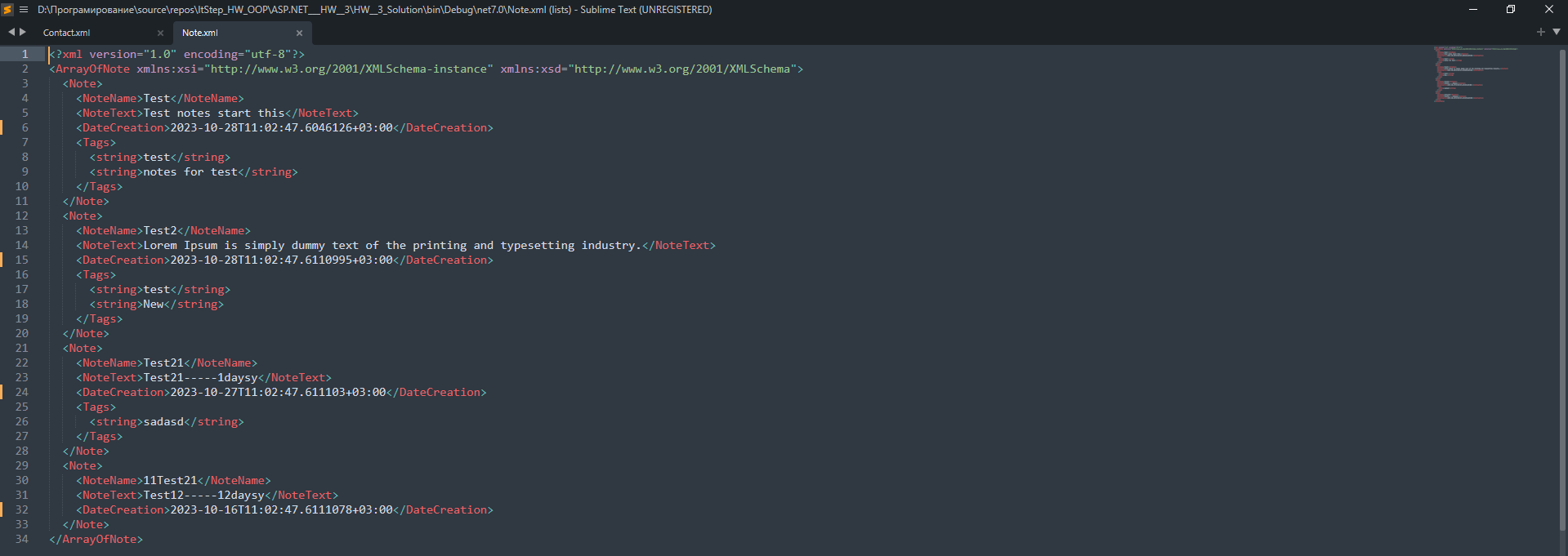
Содержание файла Note.json



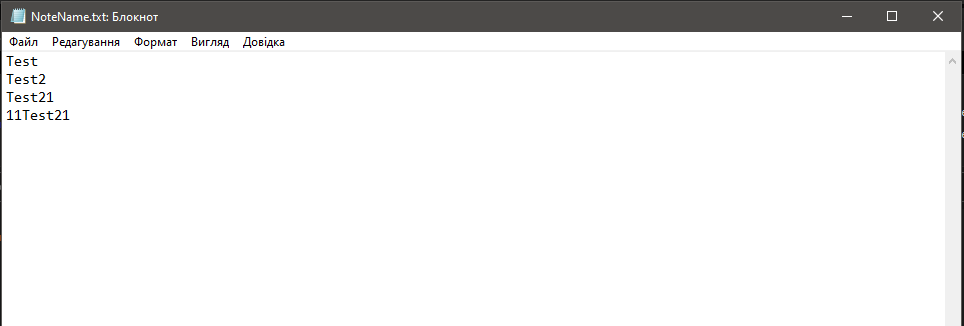
Содержание файла Note.txt



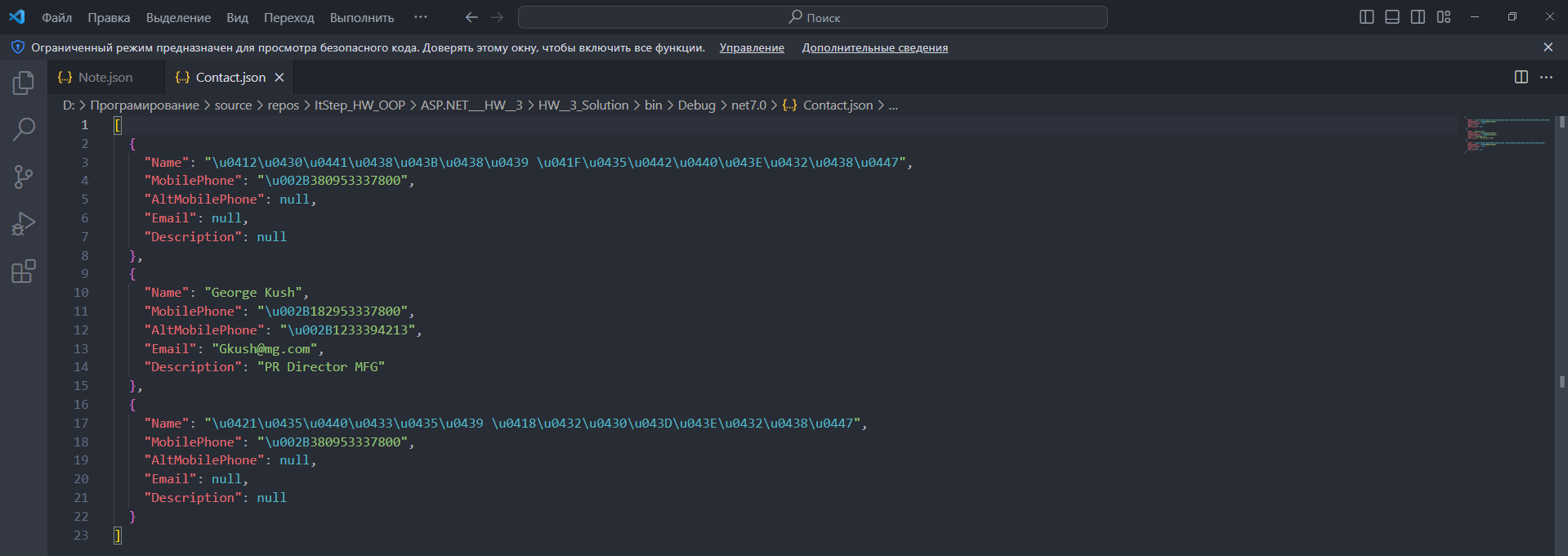
Содержание файла Note.xml



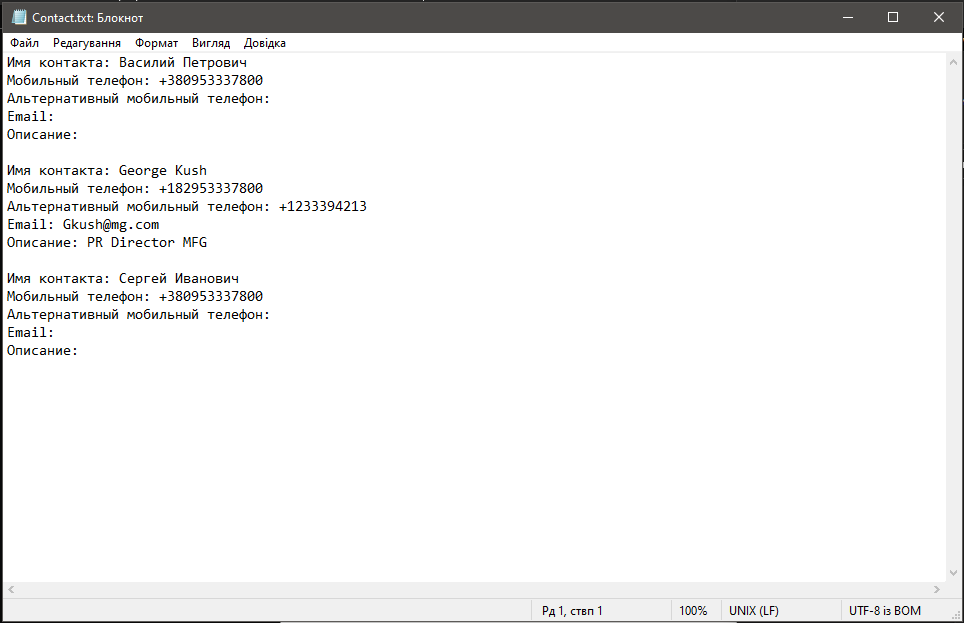
Содержание файла NoteName.txt



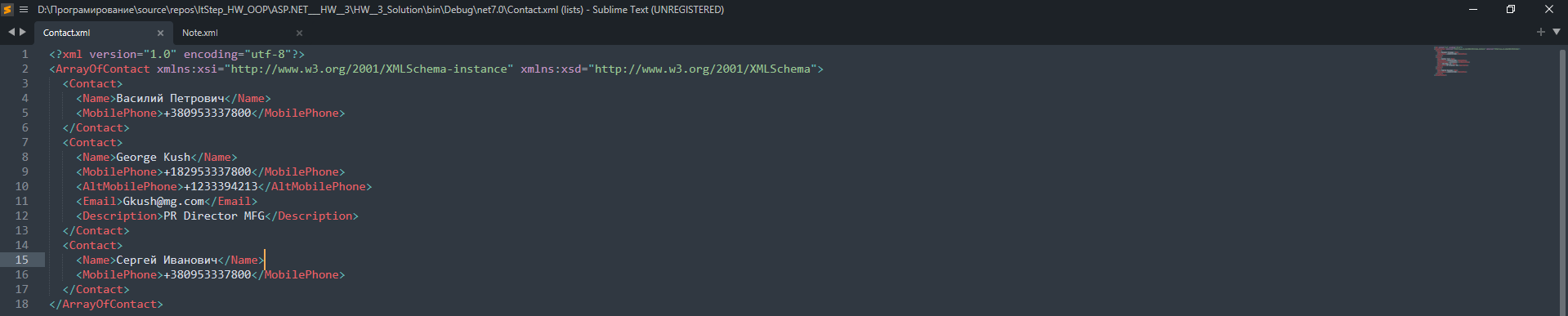
Содержание файла Contact.json



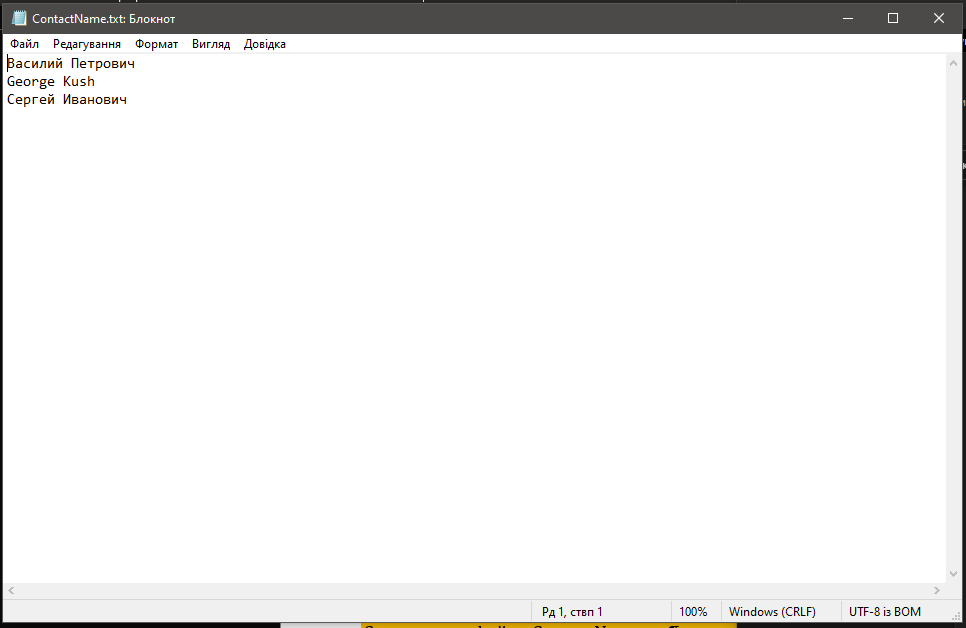
Содержание файла Contact.txt



Содержание файла Contact.xml



Содержание файла ContactName.txt



Ссылка на репозиторий GitHub проекта

<https://github.com/CeznVi/ASP.NET___HW__3/>