**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

# Дисциплина «Разработка приложений баз данных для информационных систем»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Обработка *HTTP* запросов средствами *ASP.NET Core*. Сохранение состояния. Кэширование»

ВАРИАНТ №23

«Электронная карточка пациента»

Выполнил:

студент гр. ИТИ-31 Пинчукова В.А.

Принял:

доцент Асенчик О.Д.

Гомель 2023

**Цель работы:** Ознакомиться c методами обработкой *HTTP* средствами *ASP.NET Core*, методами сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.

**Ход работы и результаты:**

1. Перед началом выполнения работы был создан репозиторий на *GitHub*, в процессе работы в нем разместится решение. Ссылка на публичный репозиторий:

*«*[*PValeriyanka/RPBDIS\_Lab3*](https://github.com/PValeriyanka/RPBDIS_Lab3)*».*

2. Модель БД была экспортирована из предыдущей работы и немного доработана (добавлены *Infrastructure* и *Services*). Основные компоненты представлены на Рисунке 1.

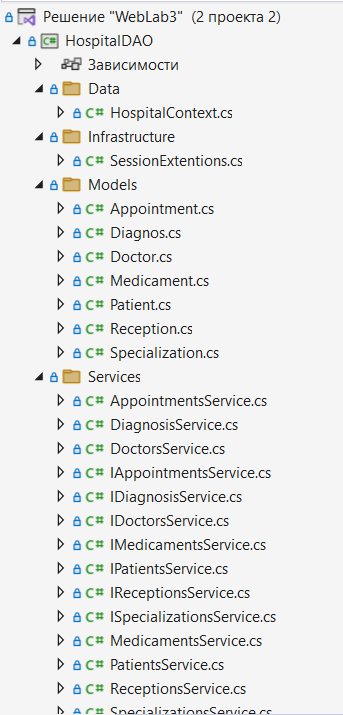


Рисунок 1 – Созданная модель БД

3. Кэширование таблиц будет осуществляться по заданному количеству записей заданной предметной области с помощью встроенного инструмента кэширования - объекта *IMemoryCache*. Данные в кэше будут храниться неизменными в течение 2\*23+240 = 286 секунд. Пример такого класса приведен в Приложениях А-Б.

Остальные классы модели сделаны по его подобию.

4. В класс *Program* (подробный листинг приведен в Приложении В) внедрена зависимость для доступа к удаленной БД, с помощью *json*-файла и секрета пользователя выполняется подключение, добавлены поддержки *Cookies* и *Session*, а также внедрены зависимости разработанных сервисов.

5. Разработана собственная система маршрутизации:

5.1. Если *URL* адрес входящего запроса содержит /*info* – программа выводит в выходной поток для отображения браузером информации о клиенте и выходит из конвейера обработки запроса;

Результат выполнения запроса приведен на Рисунке 2.

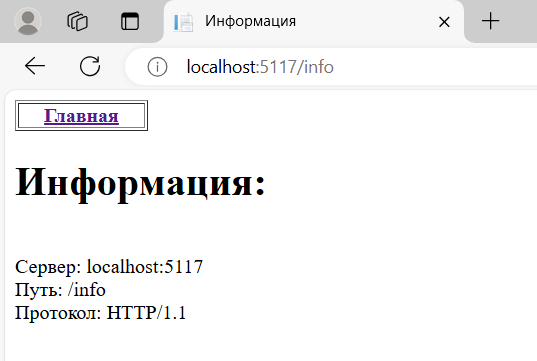


Рисунок 2 – Результат выполнения запроса /*info*

5.2. Если *URL* адрес входящего запроса содержит /*table* (где *table* – имя таблицы из базы данных) – программа выводить в выходной поток для отображения браузером с использованием метода *Response.WriteAsync* кэшированную информацию из соответствующей таблицы базы данных и выходит из конвейера обработки запроса;

Результат выполнения запроса приведен на Рисунке 3.

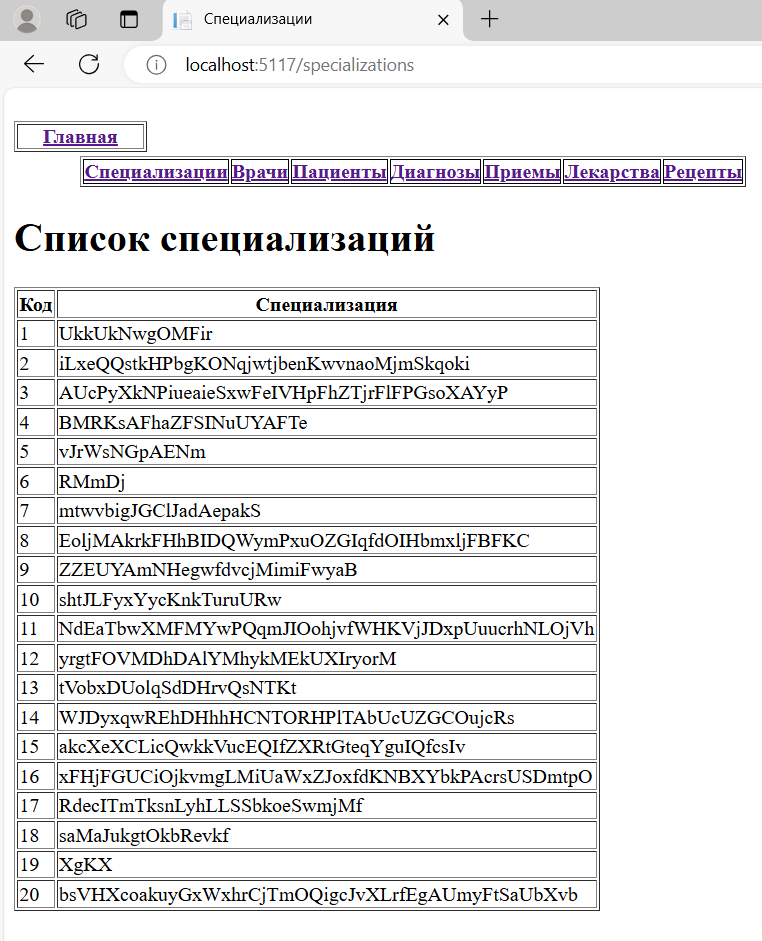


Рисунок 3 – Результат выполнения запроса /*specializations*

5.3. Если *URL* адрес входящего запроса содержит /*searchform*1 или /*searchform*2 – программа выводит в выходной поток для отображения браузером с использованием метода *Response.WriteAsync* формы для поиска информации из базы данных и выходить из конвейера обработки запроса (для того, чтобы не дублировать код, форма разработана отдельно в статическом методе *GenerateForm*. Листинг данного метода приведен в Приложении Г);

5.4. В противном случае (*URL* адрес входящего запроса не содержит перечисленных выше элементов) – программа продолжает обрабатывать другие компоненты конвейера обработки запросов.

Изображение главной страницы, на которой более точно описана возможная маршрутизация, приведено на Рисунке 4.

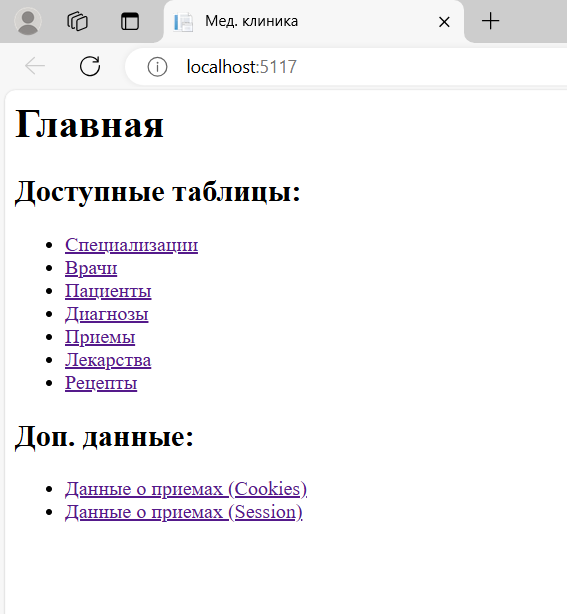


Рисунок 4 – Примерная маршрутизация сайта

1. Сохранение состояния элементов одной формы (/*searchform*1) одной страницы осуществляется с использованием *Cookies*.

Результат выполнения запроса приведен на Рисунке 5.

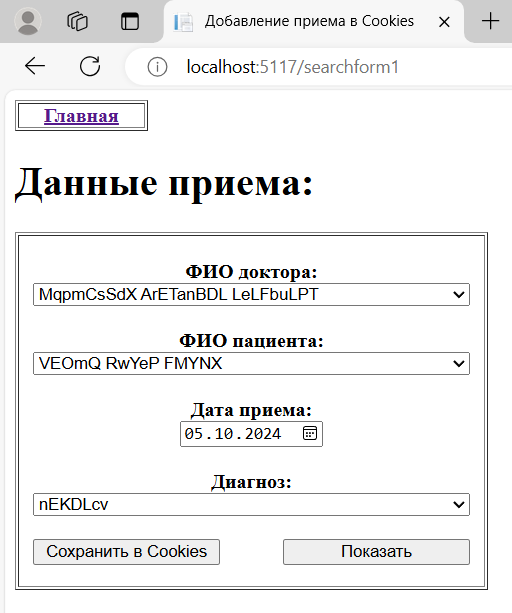


Рисунок 5 – Результат выполнения запроса /*searchform*1

1. Сохранение состояния элементов одной формы (/*searchform*2) одной страницы в виде одного объекта специальной структуры осуществляется с использованием объекта *Session*.

Результат выполнения запроса приведен на Рисунке 6.

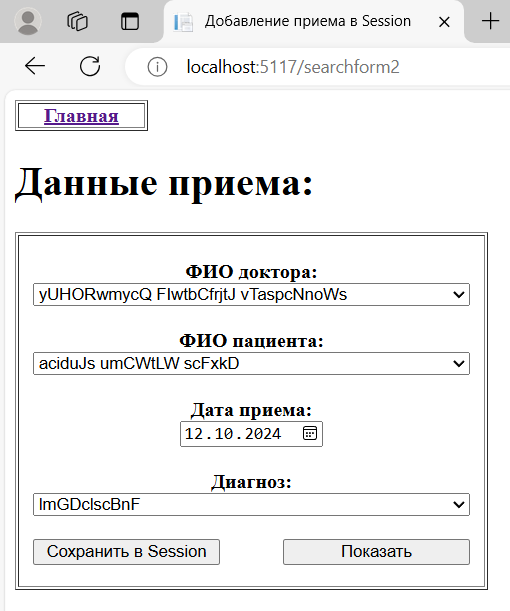


Рисунок 6 – Результат выполнения запроса /*searchform*2

1. Элементы формы при их загрузке заполняются данными ранее сохранненными в объекте *Cookies* и *Session* (/*searchform*1, /*searchform*2).

Листинг методов для работы с *Cookies* и *Session* продемонстрирован в Приложениях Д-Ж.

1. Выполненный проект размещен в удаленном репозитории на *GitHub*, к нему создан *README.md* файл.
2. Используя средство *GitHub Actions*, написан рабочий процесс, который осуществляет компиляцию проекта под две разные платформы при любом изменении в репозитории.

Листинг .*yml* приведен в Приложении Ж.

1. В *README.md* вставлен код для создания эмблемы состояния рабочего процесса (*status badge*), показывающей, чем в данный момент завершился рабочий процесс.

Демонстрация работоспособности находится на Рисунке 7.

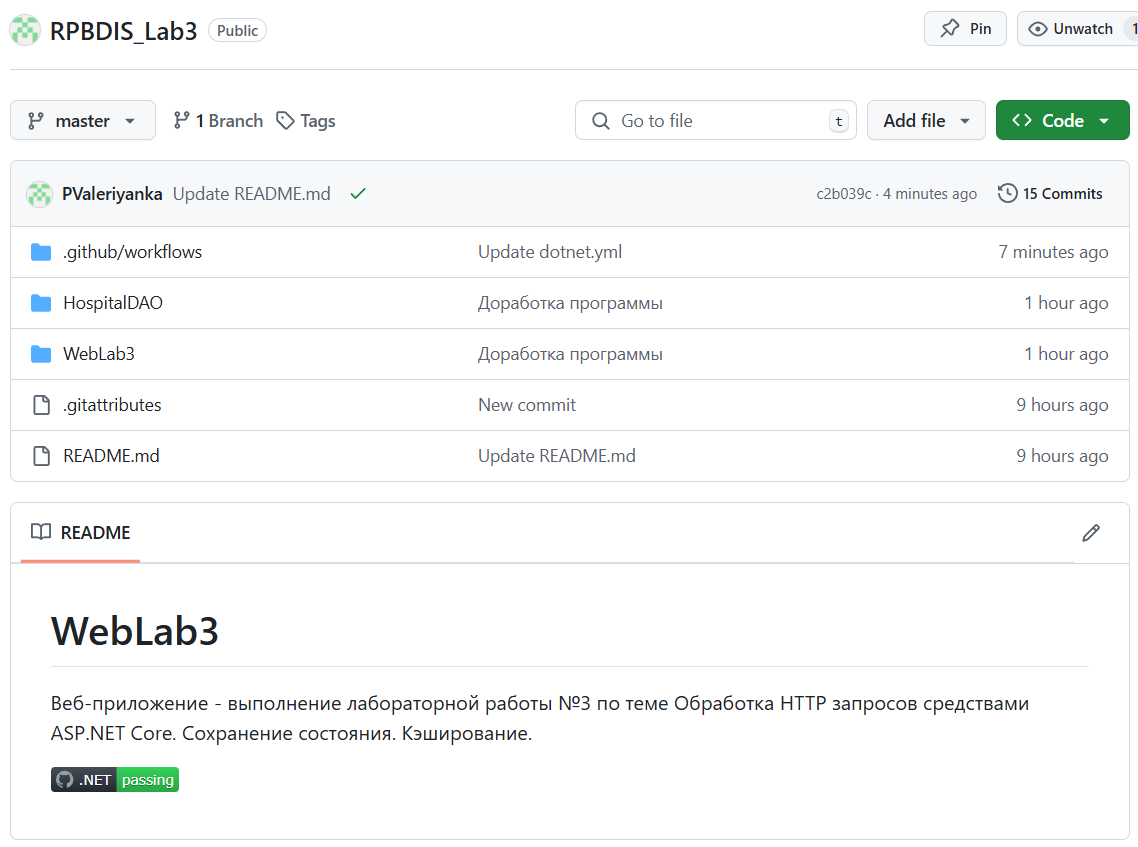


Рисунок 7 – Демонстрация работоспособности приложения

**Вывод:** В ходе выполнения лабораторной работы были получены навыки работы c методами обработки *HTTP* средствами *ASP.NET Core*, методами сохранения состояния приложения и повышение производительности приложений путем использования разных видов кэширования.

# ПРИЛОЖЕНИЕ А

# Листинг класса *AppointmentsService*

using HospitalDAO.Data;

using HospitalDAO.Models;

using Microsoft.Extensions.Caching.Memory;

namespace HospitalDAO.Services

{

public class AppointmentsService(HospitalContext dbContext, IMemoryCache memoryCache) : IAppointmentsService

{

private readonly HospitalContext \_dbContext = dbContext;

private readonly IMemoryCache \_memoryCache = memoryCache;

// Получение списка приемов из базы

public IEnumerable<Appointment> GetAppointments(int rowsNumber)

{

return \_dbContext.Appointments.Take(rowsNumber).ToList();

}

// Добавление списка приемов в кэш

public void AddAppointments(string cacheKey, int rowsNumber)

{

IEnumerable<Appointment> appointments = \_dbContext.Appointments.Take(rowsNumber).ToList();

if (appointments != null)

{

\_memoryCache.Set(cacheKey, appointments, new MemoryCacheEntryOptions

{

AbsoluteExpirationRelativeToNow = TimeSpan.FromSeconds(286)

});

}

}

// Получение списка приемов из кэша или из базы, если нет в кэше

public IEnumerable<Appointment> GetAppointments(string cacheKey, int rowsNumber)

{

if (!\_memoryCache.TryGetValue(cacheKey, out IEnumerable<Appointment> appointments))

{

appointments = \_dbContext.Appointments.Take(rowsNumber).ToList();

if (appointments != null)

{

\_memoryCache.Set(cacheKey, appointments,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(286)));

}

}

return appointments.Take(rowsNumber).ToList();

}

// Получение количества записей

public int GetAppointmentsCount()

{

return \_dbContext.Appointments.Count();

}

}

}

# ПРИЛОЖЕНИЕ Б

# Листинг интерфейса *IAppointmentsService*

using HospitalDAO.Models;

namespace HospitalDAO.Services

{

public interface IAppointmentsService

{

public IEnumerable<Appointment> GetAppointments(int rowsNumber);

public void AddAppointments(string cacheKey, int rowsNumber);

public IEnumerable<Appointment> GetAppointments(string cacheKey, int rowsNumber);

public int GetAppointmentsCount();

}

}

# 

# ПРИЛОЖЕНИЕ В

# Листинг класса *Program*

using HospitalDAO.Data;

using HospitalDAO.Infrastructure;

using HospitalDAO.Models;

using HospitalDAO.Services;

using Microsoft.Data.SqlClient;

using Microsoft.EntityFrameworkCore;

namespace WebLab3

{

public class Program

{

public static void Main(string[] args)

{

// Создание хоста

var builder = WebApplication.CreateBuilder(args);

var services = builder.Services;

// Внедрение зависимости для доступа к БД с использованием EF

IConfigurationRoot configuration = builder.Configuration.AddUserSecrets<Program>().Build();

string connectionString = configuration.GetConnectionString("RemoteSQLConnection");

// Получение пароя и имени пользователя из secrets.json

string secretPass = configuration["Database:password"];

string secretUser = configuration["Database:login"];

SqlConnectionStringBuilder sqlConnectionStringBuilder = new(connectionString)

{

Password = secretPass,

UserID = secretUser

};

connectionString = sqlConnectionStringBuilder.ConnectionString;

services.AddDbContext<HospitalContext>(options => options.UseSqlServer(connectionString));

// Добавление кэширования

services.AddMemoryCache();

// Добавление поддержки Cookies

services.AddHttpContextAccessor();

services.AddSingleton<IHttpContextAccessor, HttpContextAccessor>();

// Добавление поддержки Session

services.AddDistributedMemoryCache();

services.AddSession();

// Внедрение зависимости AppointmentsService

services.AddScoped<IAppointmentsService, AppointmentsService>();

// Внедрение зависимости DiagnosisService

services.AddScoped<IDiagnosisService, DiagnosisService>();

// Внедрение зависимости DoctorsService

services.AddScoped<IDoctorsService, DoctorsService>();

// Внедрение зависимости MedicamentsService

services.AddScoped<IMedicamentsService, MedicamentsService>();

// Внедрение зависимости PatientsService

services.AddScoped<IPatientsService, PatientsService>();

// Внедрение зависимости ReceptionsService

services.AddScoped<IReceptionsService, ReceptionsService>();

// Внедрение зависимости SpecializationsService

services.AddScoped<ISpecializationsService, SpecializationsService>();

var app = builder.Build();

// Добавление поддержки статических файлов

app.UseStaticFiles();

// Добавление поддержки Session

app.UseSession();

// Вывод информации о клиенте

app.Map("/info", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Формирование строки для вывода

string strResponse = "<HTML><HEAD><TITLE>Информация</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><TABLE BORDER><TR><TH style='padding: 0; width: 100px;'><A href='/'>Главная</A></TH></TR></TABLE><H1>Информация:</H1>";

strResponse += "<BR> Сервер: " + context.Request.Host;

strResponse += "<BR> Путь: " + context.Request.PathBase;

strResponse += "<BR> Протокол: " + context.Request.Protocol;

strResponse += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(strResponse);

});

});

// Вывод кэшированной информации из таблиц базы данных

// Вывод Specializations

app.Map("/specializations", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Обращение к сервису

ISpecializationsService iSpecializationsService = context.RequestServices.GetService<ISpecializationsService>();

IEnumerable<Specialization> specializations = iSpecializationsService.GetSpecializations("Specializations", 20);

string HtmlString = "<HTML><HEAD><TITLE>Специализации</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY>" + PrintContentInTables() +

"<H1>Список специализаций</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Код</TH>";

HtmlString += "<TH>Специализация</TH>";

HtmlString += "</TR>";

foreach (var specialization in specializations)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + specialization.SpecializationId + "</TD>";

HtmlString += "<TD>" + specialization.SpecializationName + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

// Вывод Doctors

app.Map("/doctors", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Обращение к сервису

IDoctorsService iDoctorsService = context.RequestServices.GetService<IDoctorsService>();

IEnumerable<Doctor> doctors = iDoctorsService.GetDoctors("Doctors", 20);

ISpecializationsService iSpecializationsService = context.RequestServices.GetService<ISpecializationsService>();

IEnumerable<Specialization> specializations = iSpecializationsService.GetSpecializations("Specializations", iSpecializationsService.GetSpecializationsCount());

Dictionary<int, string> specializationsDict = specializations.ToDictionary(s => s.SpecializationId, s => s.SpecializationName);

string HtmlString = "<HTML><HEAD><TITLE>Врачи</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY>" + PrintContentInTables() +

"<H1>Список докторов</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Код</TH>";

HtmlString += "<TH>ФИО</TH>";

HtmlString += "<TH>Специализация</TH>";

HtmlString += "<TH>Контактная информация</TH>";

HtmlString += "<TH>Пароль</TH>";

HtmlString += "</TR>";

foreach (var doctor in doctors)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + doctor.DoctorId + "</TD>";

HtmlString += "<TD>" + doctor.DoctorLastName + " " + doctor.DoctorFirstName + " " + doctor.DoctorSurname + "</TD>";

HtmlString += "<TD>" + specializationsDict.GetValueOrDefault(doctor.SpecializationId, "Неизвестно") + "</TD>";

HtmlString += "<TD>" + doctor.ContactData + "</TD>";

HtmlString += "<TD>" + doctor.Password + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

// Вывод Patients

app.Map("/patients", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Обращение к сервису

IPatientsService iPatientsService = context.RequestServices.GetService<IPatientsService>();

IEnumerable<Patient> patients = iPatientsService.GetPatients("Patients", 20);

string HtmlString = "<HTML><HEAD><TITLE>Пациенты</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY>" + PrintContentInTables() +

"<H1>Список пациентов</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Код</TH>";

HtmlString += "<TH>ФИО</TH>";

HtmlString += "<TH>Дата рождения</TH>";

HtmlString += "<TH>Контактная информация</TH>";

HtmlString += "</TR>";

foreach (var patient in patients)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + patient.PatientId + "</TD>";

HtmlString += "<TD>" + patient.PatientLastName + " " + patient.PatientFirstName + " " + patient.PatientSurname + "</TD>";

HtmlString += "<TD>" + patient.BirthDate + "</TD>";

HtmlString += "<TD>" + patient.ContactData + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

// Вывод Diagnosis

app.Map("/diagnosis", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Обращение к сервису

IDiagnosisService iDiagnosisService = context.RequestServices.GetService<IDiagnosisService>();

IEnumerable<Diagnos> diagnosis = iDiagnosisService.GetDiagnosis("Diagnosis", 20);

string HtmlString = "<HTML><HEAD><TITLE>Диагнозы</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY>" + PrintContentInTables() +

"<H1>Список диагнозов</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Код</TH>";

HtmlString += "<TH>Диагноз</TH>";

HtmlString += "<TH>Описание</TH>";

HtmlString += "</TR>";

foreach (var diagnos in diagnosis)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + diagnos.DiagnosId + "</TD>";

HtmlString += "<TD>" + diagnos.DiagnosName + "</TD>";

HtmlString += "<TD>" + diagnos.DiagnosDescription + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

// Вывод Appointments

app.Map("/appointments", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Обращение к сервису

IAppointmentsService iAppointmentsService = context.RequestServices.GetService<IAppointmentsService>();

IEnumerable<Appointment> appointments = iAppointmentsService.GetAppointments("Appointments", 20);

IDoctorsService iDoctorsService = context.RequestServices.GetService<IDoctorsService>();

IEnumerable<Doctor> doctors = iDoctorsService.GetDoctors("Doctors", iDoctorsService.GetDoctorsCount());

Dictionary<int, string> doctorsDict = doctors.ToDictionary(d => d.DoctorId, d => d.DoctorLastName + " " + d.DoctorFirstName + " " + d.DoctorSurname);

IPatientsService iPatientsService = context.RequestServices.GetService<IPatientsService>();

IEnumerable<Patient> patients = iPatientsService.GetPatients("Patients", iPatientsService.GetPatientsCount());

Dictionary<int, string> patientsDict = patients.ToDictionary(p => p.PatientId, p => p.PatientLastName + " " + p.PatientFirstName + " " + p.PatientSurname);

IDiagnosisService iDiagnosisService = context.RequestServices.GetService<IDiagnosisService>();

IEnumerable<Diagnos> diagnosis = iDiagnosisService.GetDiagnosis("Diagnosis", iDiagnosisService.GetDiagnosisCount());

Dictionary<int, string> diagnosisDict = diagnosis.ToDictionary(d => d.DiagnosId, d => d.DiagnosName);

string HtmlString = "<HTML><HEAD><TITLE>Приемы</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY>" + PrintContentInTables() +

"<H1>Список приемов</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Код</TH>";

HtmlString += "<TH>ФИО доктора</TH>";

HtmlString += "<TH>ФИО пациента</TH>";

HtmlString += "<TH>Дата приема</TH>";

HtmlString += "<TH>Диагноз</TH>";

HtmlString += "</TR>";

foreach (var appointment in appointments)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + appointment.AppointmentId + "</TD>";

HtmlString += "<TD>" + doctorsDict.GetValueOrDefault(appointment.DoctorId, "Неизвестно") + "</TD>";

HtmlString += "<TD>" + patientsDict.GetValueOrDefault(appointment.PatientId, "Неизвестно") + "</TD>";

HtmlString += "<TD>" + appointment.AppointmentDate + "</TD>";

HtmlString += "<TD>" + diagnosisDict.GetValueOrDefault(appointment.DiagnosId, "Неизвестно") + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

// Вывод Medicaments

app.Map("/medicaments", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Обращение к сервису

IMedicamentsService iMedicamentsService = context.RequestServices.GetService<IMedicamentsService>();

IEnumerable<Medicament> medicaments = iMedicamentsService.GetMedicaments("Medicaments", 20);

string HtmlString = "<HTML><HEAD><TITLE>Лекарства</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY>" + PrintContentInTables() +

"<H1>Список лекарств</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Код</TH>";

HtmlString += "<TH>Название</TH>";

HtmlString += "<TH>Дозировка</TH>";

HtmlString += "<TH>Стоимость</TH>";

HtmlString += "</TR>";

foreach (var medicament in medicaments)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + medicament.MedicamentId + "</TD>";

HtmlString += "<TD>" + medicament.MedicamentName + "</TD>";

HtmlString += "<TD>" + medicament.MedicamentDose + "</TD>";

HtmlString += "<TD>" + Math.Round(medicament.MedicamentPrice, 2) + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

// Вывод Receptions

app.Map("/receptions", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Обращение к сервису

IReceptionsService iReceptionsService = context.RequestServices.GetService<IReceptionsService>();

IEnumerable<Reception> receptions = iReceptionsService.GetReceptions("Receptions", 20);

IMedicamentsService iMedicamentsService = context.RequestServices.GetService<IMedicamentsService>();

IEnumerable<Medicament> medicaments = iMedicamentsService.GetMedicaments("Medicaments", iMedicamentsService.GetMedicamentsCount());

Dictionary<int, string> medicamentsDict = medicaments.ToDictionary(m => m.MedicamentId, m => m.MedicamentName);

string HtmlString = "<HTML><HEAD><TITLE>Выписки</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY>" + PrintContentInTables() +

"<H1>Список рецептов</H1>" +

"<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH>Код</TH>";

HtmlString += "<TH>Код приема</TH>";

HtmlString += "<TH>Лекарство</TH>";

HtmlString += "<TH>Дозировка</TH>";

HtmlString += "</TR>";

foreach (var reception in receptions)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + reception.ReceptionId + "</TD>";

HtmlString += "<TD>" + reception.AppointmentId + "</TD>";

HtmlString += "<TD>" + medicamentsDict.GetValueOrDefault(reception.MedicamentId, "Неизвестно") + "</TD>";

HtmlString += "<TD>" + reception.ReceptionDose + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "</BODY></HTML>";

// Вывод данных

await context.Response.WriteAsync(HtmlString);

});

});

// Работа с Cookies

app.Map("/searchform1", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Считывание из Cookies

string Doctor = context.Request.Cookies["DoctorId"];

int DoctorId = !string.IsNullOrEmpty(Doctor) ? int.Parse(Doctor) : 0;

string Patient = context.Request.Cookies["PatientId"];

int PatientId = !string.IsNullOrEmpty(Patient) ? int.Parse(Patient) : 0;

string Date = context.Request.Cookies["AppointmentDate"];

DateOnly AppointmentDate = string.IsNullOrEmpty(Date) ? DateOnly.FromDateTime(DateTime.Now) : DateOnly.Parse(Date);

string Diagnos = context.Request.Cookies["DiagnosId"];

int DiagnosId = !string.IsNullOrEmpty(Diagnos) ? int.Parse(Diagnos) : 0;

string strResponse = GenerateForm(context, DoctorId, PatientId, AppointmentDate, DiagnosId, "Cookies");

// Загрузка данных в Cookies

if (context.Request.Method == "POST")

{

context.Response.Cookies.Append("DoctorId", context.Request.Form["DoctorId"]);

context.Response.Cookies.Append("PatientId", context.Request.Form["PatientId"]);

context.Response.Cookies.Append("AppointmentDate", context.Request.Form["AppointmentDate"]);

context.Response.Cookies.Append("DiagnosId", context.Request.Form["DiagnosId"]);

}

await context.Response.WriteAsync(strResponse);

});

});

// Работа с Session

app.Map("/searchform2", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Считывание из Session объекта Appointmment

Appointment appointment = context.Session.Get<Appointment>("appointment") ?? new Appointment();

string strResponse = GenerateForm(context, appointment.DoctorId, appointment.PatientId, appointment.AppointmentDate, appointment.DiagnosId, "Session");

// Загрузка данных в Session

if (context.Request.Method == "POST")

{

string Doctor = context.Request.Form["DoctorId"];

appointment.DoctorId = !string.IsNullOrEmpty(Doctor) ? int.Parse(Doctor) : 0;

string Patient = context.Request.Form["PatientId"];

appointment.PatientId = !string.IsNullOrEmpty(Patient) ? int.Parse(Patient) : 0;

string Date = context.Request.Form["AppointmentDate"];

appointment.AppointmentDate = string.IsNullOrEmpty(Date) ? DateOnly.FromDateTime(DateTime.Now) : DateOnly.Parse(Date);

string Diagnos = context.Request.Form["DiagnosId"];

appointment.DiagnosId = !string.IsNullOrEmpty(Diagnos) ? int.Parse(Diagnos) : 0;

context.Session.Set<Appointment>("appointment", appointment);

}

await context.Response.WriteAsync(strResponse);

});

});

// Стартовая страница и кэширование данных таблицы на web-сервере

app.Run((context) =>

{

// Обращение к сервису

ISpecializationsService iSpecializationsService = context.RequestServices.GetService<ISpecializationsService>();

iSpecializationsService.AddSpecializations("Specializations", iSpecializationsService.GetSpecializationsCount());

IDoctorsService iDoctorsService = context.RequestServices.GetService<IDoctorsService>();

iDoctorsService.AddDoctors("Doctors", iDoctorsService.GetDoctorsCount());

IPatientsService iPatientsService = context.RequestServices.GetService<IPatientsService>();

iPatientsService.AddPatients("Patients", iPatientsService.GetPatientsCount());

IDiagnosisService iDiagnosisService = context.RequestServices.GetService<IDiagnosisService>();

iDiagnosisService.AddDiagnosis("Diagnosis", iDiagnosisService.GetDiagnosisCount());

IAppointmentsService iAppointmentsService = context.RequestServices.GetService<IAppointmentsService>();

iAppointmentsService.AddAppointments("Appointments", iAppointmentsService.GetAppointmentsCount());

IMedicamentsService iMedicamentsService = context.RequestServices.GetService<IMedicamentsService>();

iMedicamentsService.AddMedicaments("Medicaments", iMedicamentsService.GetMedicamentsCount());

IReceptionsService iReceptionsService = context.RequestServices.GetService<IReceptionsService>();

iReceptionsService.AddReceptions("Reception", iReceptionsService.GetReceptionsCount());

string HtmlString = "<HTML><HEAD><TITLE>Мед. клиника</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><H1>Главная</H1>";

HtmlString += "<H2>Доступные таблицы:</H2>";

HtmlString += "<UL>";

HtmlString += "<LI><A href='/specializations'>Специализации</A></LI>";

HtmlString += "<LI><A href='/doctors'>Врачи</A></LI>";

HtmlString += "<LI><A href='/patients'>Пациенты</A></LI>";

HtmlString += "<LI><A href='/diagnosis'>Диагнозы</A></LI>";

HtmlString += "<LI><A href='/appointments'>Приемы</A></LI>";

HtmlString += "<LI><A href='/medicaments'>Лекарства</A></LI>";

HtmlString += "<LI><A href='/receptions'>Рецепты</A></LI>";

HtmlString += "</UL>";

HtmlString += "<H2>Доп. данные:</H2>";

HtmlString += "<UL>";

HtmlString += "<LI><A href='/searchform1'>Данные о приемах (Cookies)</A></LI>";

HtmlString += "<LI><A href='/searchform2'>Данные о приемах (Session)</A></LI>";

HtmlString += "</UL>";

HtmlString += "</BODY></HTML>";

return context.Response.WriteAsync(HtmlString);

});

app.Run();

}

// Вывод содержания

static string PrintContentInTables()

{

string HtmlString = "<BR>";

HtmlString += "<TABLE BORDER=1>";

HtmlString += "<TR>";

HtmlString += "<TH style='padding: 0; width: 100px;'><A href='/'>Главная</A></TH>";

HtmlString += "</TR>";

HtmlString += "</TABLE>";

HtmlString += "<TABLE BORDER=1 style='margin-left: 53px; margin-top: 3px'>";

HtmlString += "<TR>";

HtmlString += "<TH style='border: 1px solid black; padding: 0;'><A href='/specializations'>Специализации</A></TH>";

HtmlString += "<TH style='border: 1px solid black; padding: 0;'><A href='/doctors'>Врачи</A></TH>";

HtmlString += "<TH style='border: 1px solid black; padding: 0;'><A href='/patients'>Пациенты</A></TH>";

HtmlString += "<TH style='border: 1px solid black; padding: 0;'><A href='/diagnosis'>Диагнозы</A></TH>";

HtmlString += "<TH style='border: 1px solid black; padding: 0;'><A href='/appointments'>Приемы</A></TH>";

HtmlString += "<TH style='border: 1px solid black; padding: 0;'><A href='/medicaments'>Лекарства</A></TH>";

HtmlString += "<TH style='border: 1px solid black; padding: 0;'><A href='/receptions'>Рецепты</A></TH>";

HtmlString += "</TR>";

HtmlString += "</TABLE>";

return HtmlString;

}

// Форма для просмотра информации

public static string GenerateForm(HttpContext context, int doctorId, int patientId, DateOnly appointmentDate, int diagnosId, string str)

{

var db = context.RequestServices.GetService<HospitalContext>();

List<Diagnos> diagnosis = db.Diagnosis.ToList();

diagnosis.Sort();

List<Doctor> doctors = db.Doctors.ToList();

doctors.Sort();

List<Patient> patients = db.Patients.ToList();

patients.Sort();

string strResponse = $"<HTML><HEAD><TITLE>Добавление приема в {str}</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><TABLE BORDER><TR><TH style='padding: 0; width: 100px;'><A href='/'>Главная</A></TH></TR></TABLE>" +

"<H1>Данные приема:</H1><TABLE BORDER=1><TR><TH><BR><FORM style='margin-left:10px; margin-right:10px' method='POST'>" +

"ФИО доктора:<BR><SELECT style='width: 350px;' padding: 0;' name='DoctorId'>";

foreach (var doctor in doctors)

{

strResponse += $"<option value='{doctor.DoctorId}'" +

$"{(doctorId == doctor.DoctorId ? " selected" : "")}>" +

$"{doctor.DoctorLastName + " " + doctor.DoctorFirstName + " " + doctor.DoctorSurname}</option>";

}

strResponse += "</SELECT><BR><BR>" +

"ФИО пациента:<BR><SELECT style='width: 350px;' name='PatientId'>";

foreach (var patient in patients)

{

strResponse += $"<option value='{patient.PatientId}'" +

$"{(patientId == patient.PatientId ? " selected" : "")}>" +

$"{patient.PatientLastName + " " + patient.PatientFirstName + " " + patient.PatientSurname}</option>";

}

strResponse += "</SELECT><BR><BR>" +

"Дата приема:<BR><INPUT type='date' name='AppointmentDate' value='" + appointmentDate.ToString("yyyy-MM-dd") + "'><BR><BR>" +

"Диагноз:<BR><SELECT style='width: 350px;' name='DiagnosId'>";

foreach (var diagnos in diagnosis)

{

strResponse += $"<option value='{diagnos.DiagnosId}'" +

$"{(diagnosId == diagnos.DiagnosId ? " selected" : "")}>" +

$"{diagnos.DiagnosName}</option>";

}

strResponse += "</SELECT><BR><BR>" +

$"<INPUT style='width: 150px;' type='submit' value='Сохранить в {str}'>" +

"<INPUT style='width: 150px; margin-left:50px;' type='submit' value='Показать'></FORM></TH></TR></TABLE></BODY></HTML>";

return strResponse;

}

}

}

**ПРИЛОЖЕНИЕ Г**

# Листинг метода для создания формы *GenerateForm*

public static string GenerateForm(HttpContext context, int doctorId, int patientId, DateOnly appointmentDate, int diagnosId, string str)

{

var db = context.RequestServices.GetService<HospitalContext>();

List<Diagnos> diagnosis = db.Diagnosis.ToList();

diagnosis.Sort();

List<Doctor> doctors = db.Doctors.ToList();

doctors.Sort();

List<Patient> patients = db.Patients.ToList();

patients.Sort();

string strResponse = $"<HTML><HEAD><TITLE>Добавление приема в {str}</TITLE></HEAD>" +

"<META http-equiv='Content-Type' content='text/html; charset=utf-8'/>" +

"<BODY><TABLE BORDER><TR><TH style='padding: 0; width: 100px;'><A href='/'>Главная</A></TH></TR></TABLE>" +

"<H1>Данные приема:</H1><TABLE BORDER=1><TR><TH><BR><FORM style='margin-left:10px; margin-right:10px' method='POST'>" +

"ФИО доктора:<BR><SELECT style='width: 350px;' padding: 0;' name='DoctorId'>";

foreach (var doctor in doctors)

{

strResponse += $"<option value='{doctor.DoctorId}'" +

$"{(doctorId == doctor.DoctorId ? " selected" : "")}>" +

$"{doctor.DoctorLastName + " " + doctor.DoctorFirstName + " " + doctor.DoctorSurname}</option>";

}

strResponse += "</SELECT><BR><BR>" +

"ФИО пациента:<BR><SELECT style='width: 350px;' name='PatientId'>";

foreach (var patient in patients)

{

strResponse += $"<option value='{patient.PatientId}'" +

$"{(patientId == patient.PatientId ? " selected" : "")}>" +

$"{patient.PatientLastName + " " + patient.PatientFirstName + " " + patient.PatientSurname}</option>";

}

strResponse += "</SELECT><BR><BR>" +

"Дата приема:<BR><INPUT type='date' name='AppointmentDate' value='" + appointmentDate.ToString("yyyy-MM-dd") + "'><BR><BR>" +

"Диагноз:<BR><SELECT style='width: 350px;' name='DiagnosId'>";

foreach (var diagnos in diagnosis)

{

strResponse += $"<option value='{diagnos.DiagnosId}'" +

$"{(diagnosId == diagnos.DiagnosId ? " selected" : "")}>" +

$"{diagnos.DiagnosName}</option>";

}

strResponse += "</SELECT><BR><BR>" +

$"<INPUT style='width: 150px;' type='submit' value='Сохранить в {str}'>" +

"<INPUT style='width: 150px; margin-left:50px;' type='submit' value='Показать'></FORM></TH></TR></TABLE></BODY></HTML>";

return strResponse;

# } ПРИЛОЖЕНИЕ Д Листинг работы с *Cookies*

app.Map("/searchform1", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Считывание из Cookies

string Doctor = context.Request.Cookies["DoctorId"];

int DoctorId = !string.IsNullOrEmpty(Doctor) ? int.Parse(Doctor) : 0;

string Patient = context.Request.Cookies["PatientId"];

int PatientId = !string.IsNullOrEmpty(Patient) ? int.Parse(Patient) : 0;

string Date = context.Request.Cookies["AppointmentDate"];

DateOnly AppointmentDate = string.IsNullOrEmpty(Date) ? DateOnly.FromDateTime(DateTime.Now) : DateOnly.Parse(Date);

string Diagnos = context.Request.Cookies["DiagnosId"];

int DiagnosId = !string.IsNullOrEmpty(Diagnos) ? int.Parse(Diagnos) : 0;

string strResponse = GenerateForm(context, DoctorId, PatientId, AppointmentDate, DiagnosId, "Cookies");

// Загрузка данных в Cookies

if (context.Request.Method == "POST")

{

context.Response.Cookies.Append("DoctorId", context.Request.Form["DoctorId"]);

context.Response.Cookies.Append("PatientId", context.Request.Form["PatientId"]);

context.Response.Cookies.Append("AppointmentDate", context.Request.Form["AppointmentDate"]);

context.Response.Cookies.Append("DiagnosId", context.Request.Form["DiagnosId"]);

}

await context.Response.WriteAsync(strResponse);

});

# });

# ПРИЛОЖЕНИЕ Е Листинг работы с *Session*

app.Map("/searchform2", (appBuilder) =>

{

appBuilder.Run(async (context) =>

{

// Считывание из Session объекта Appointmment

Appointment appointment = context.Session.Get<Appointment>("appointment") ?? new Appointment();

string strResponse = GenerateForm(context, appointment.DoctorId, appointment.PatientId, appointment.AppointmentDate, appointment.DiagnosId, "Session");

// Загрузка данных в Session

if (context.Request.Method == "POST")

{

string Doctor = context.Request.Form["DoctorId"];

appointment.DoctorId = !string.IsNullOrEmpty(Doctor) ? int.Parse(Doctor) : 0;

string Patient = context.Request.Form["PatientId"];

appointment.PatientId = !string.IsNullOrEmpty(Patient) ? int.Parse(Patient) : 0;

string Date = context.Request.Form["AppointmentDate"];

appointment.AppointmentDate = string.IsNullOrEmpty(Date) ? DateOnly.FromDateTime(DateTime.Now) : DateOnly.Parse(Date);

string Diagnos = context.Request.Form["DiagnosId"];

appointment.DiagnosId = !string.IsNullOrEmpty(Diagnos) ? int.Parse(Diagnos) : 0;

context.Session.Set<Appointment>("appointment", appointment);

}

await context.Response.WriteAsync(strResponse);

});

});

# ПРИЛОЖЕНИЕ Ж Листинг *dotnet.yml*

# This workflow will build a .NET project

# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-net

name: .NET

on:

push:

pull\_request:

branches: [ RequestProcessing ]

paths:

- '\*\*.cs'

- '\*\*.csproj'

env:

DOTNET\_VERSION: '8.0' # The .NET SDK version to use

jobs:

build-and-test:

name: build-and-test-${{matrix.os}}

runs-on: ${{ matrix.os }}

strategy:

matrix:

os: [ubuntu-latest, windows-latest]

steps:

- uses: actions/checkout@v4

- name: Setup .NET Core

uses: actions/setup-dotnet@v4

with:

dotnet-version: ${{ env.DOTNET\_VERSION }}

- name: Restore dependencies

run: dotnet restore WebLab3/WebLab3.sln

- name: Build

run: dotnet build WebLab3/WebLab3.sln --no-restore

- name: Test

run: dotnet test WebLab3/WebLab3.sln --no-build --verbosity normal