**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 3**

по дисциплине «Разработка приложений баз данных для информационных систем»

на тему: «Обработка HTTP запросов средствами ASP.NET Core. Сохранение состояния. Кэширование.»

Вариант 9

Выполнил: студент гр. ИТП-31

Коршиков Н.В.

Принял: доцент

Асенчик О. Д.

Гомель 2020

**Цель работы:**

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

**Задание:**

Используя ранее разработанные объектную модель для доступа к данным в заданной предметной области разработать простое ASP.NET Core приложение.

1. С использование методов Run, Map и Use разработать:
   1. компоненты промежуточного уровня (middleware) и встроить их в конвейер обработки HTTP запроса с целью кэширования 20 записей из каждой таблицы базы данных заданной предметной области с помощью встроенного инструмента кэширования - объекта IMemoryCache. Данные в кэше хранить неизменными в течение 2\*N+240 секунд, где N- номер вашего варианта.
   2. собственную систему маршрутизации входящих запросов:

* если URL адрес входящего запроса содержит \info – выводить в выходной поток для отображения браузером информацию о клиенте и выходить из конвейера обработки запроса;
* если URL адрес входящего запроса содержит \table (где table – имя таблицы из базы данных) – выводить в выходной поток для отображения браузером с использование метода Response.WriteAsync кэшированную информацию из соответствующей таблицы базы данных и выходить из конвейера обработки запроса;
* если URL адрес входящего запроса содержит \searchform1 или \searchform2 – выводить в выходной поток для отображения браузером с использование метода Response.WriteAsync формы для поиска информации из базы данных и выходить из конвейера обработки запроса;

форма должна содержать, как минимум: одно поле, одного поле со списком, один список, одну кнопку;

* в противном случае (URL адрес входящего запроса не содержит перечисленных выше элементов) - продолжать обрабатывать другие компоненты конвейера обработки запросов и передавать управление системе маршрутизации MVC фреймворка;

1. Реализовать сохранение состояния элементов одной формы одной страницы с использованием куки.
2. Реализовать сохранение состояния элементов одной формы одной страницы в виде одного объекта специальной структуры с использованием объекта Session.
3. Осуществить заполнение элементов формы при их загрузке данными ранее сохранненными в объекте Session и куки.
4. С использованием средств разработчика браузера (Chrome, Firefox) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием MemoryCache.

**Вариант задания:**

Предметная область задания изображена на рисунке 1.

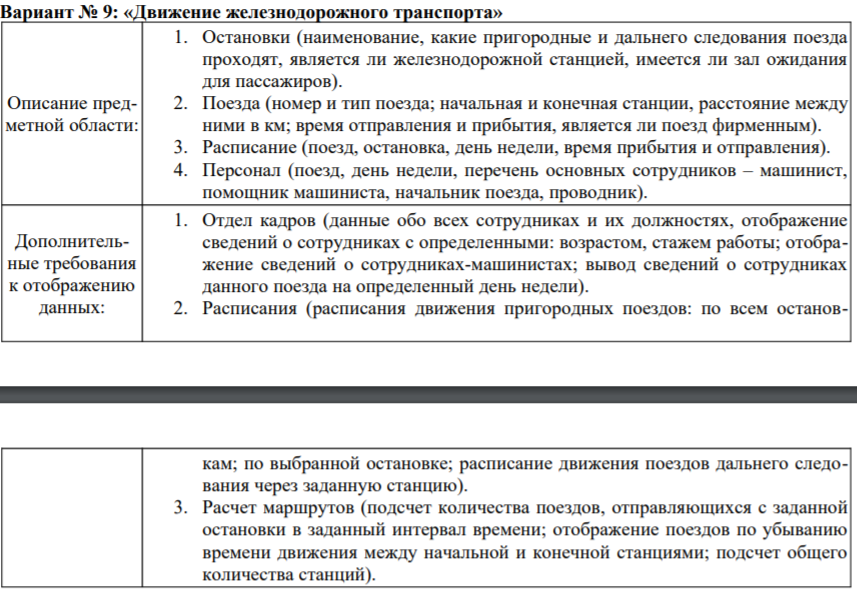


Рисунок 1 – Предметная область

**Ход работы:**

Результаты выполнения запросов приведены на рисунках 2 - .

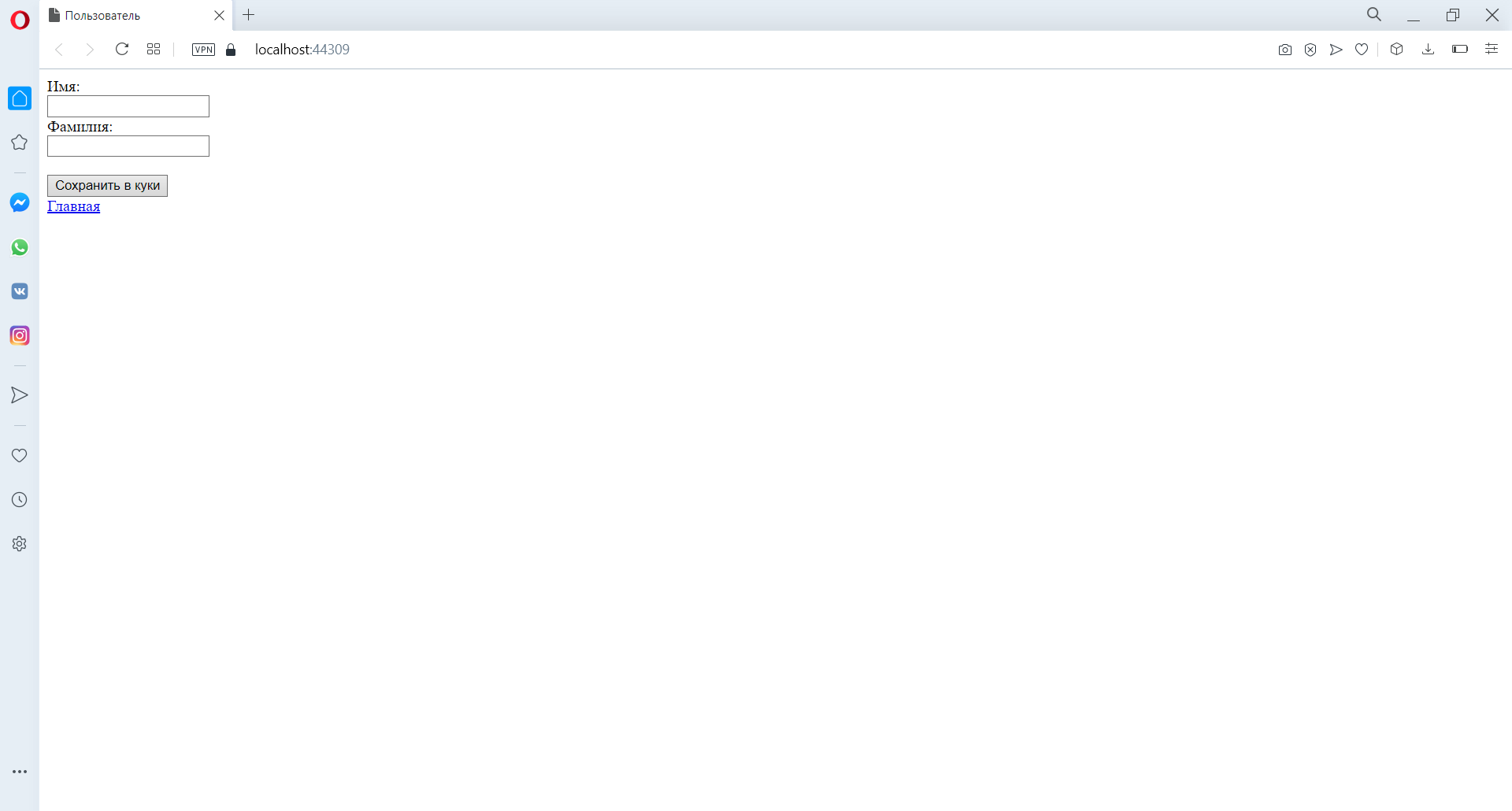


Рисунок 2 – форма для сохранения данных пользователя в куки

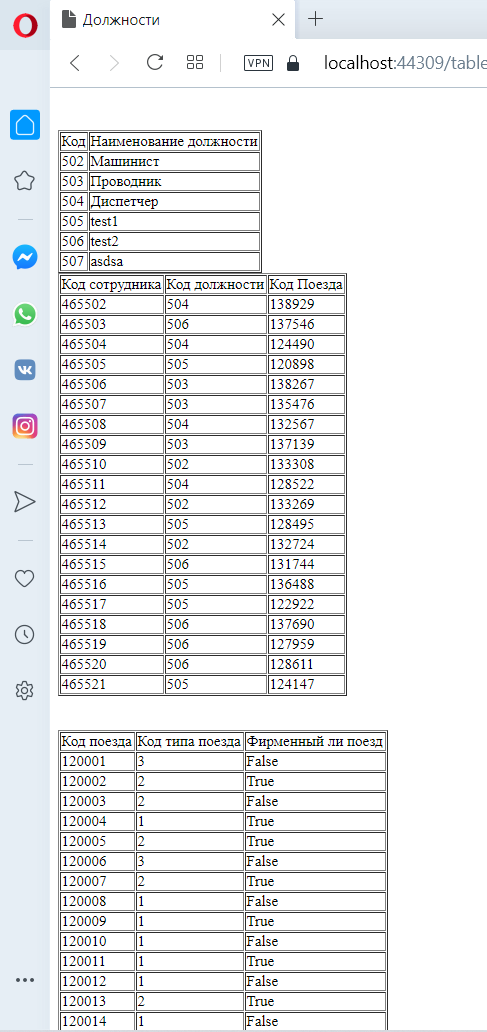


Рисунок 3 – форма для вывода кэшированных данных из таблиц базы данных

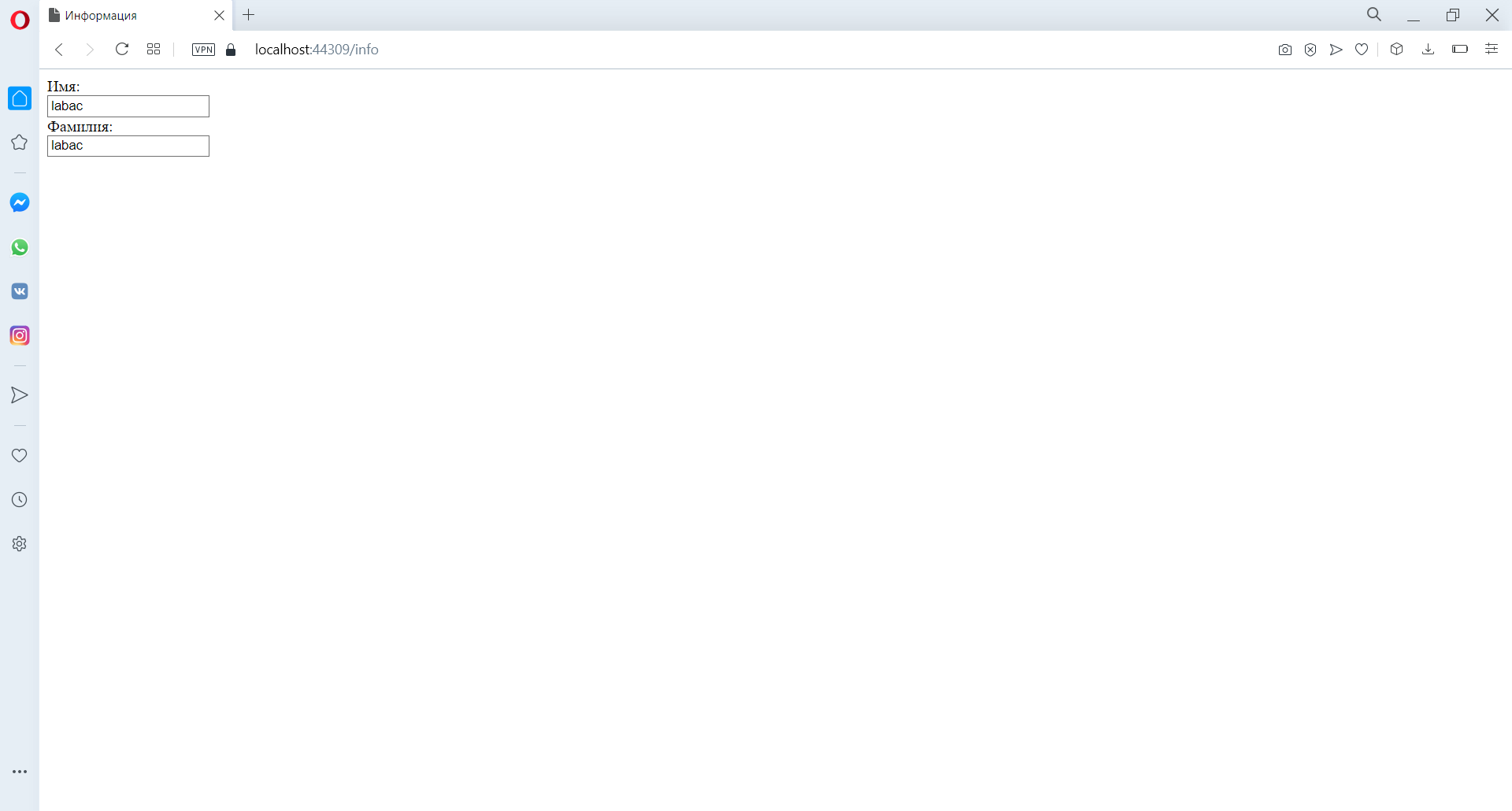


Рисунок 4 – форма для вывода данных клиента из куки

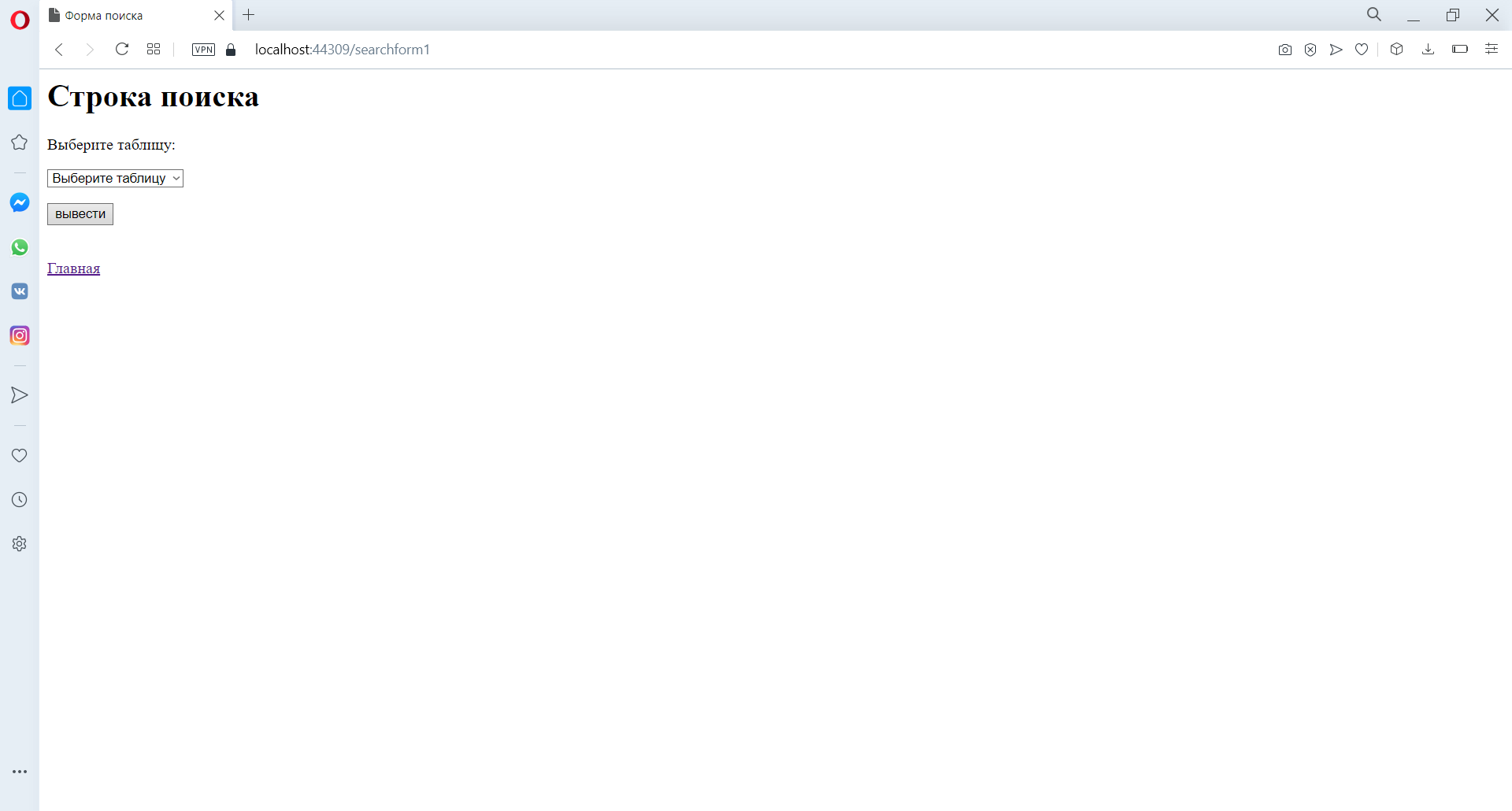


Рисунок 5 – форма для взаимодействия с таблицами базы данных

**Вывод:**

Были созданы различные формы для взаимодействия с таблицами базы данных, а также для изучения различных видов кэширования.

**ПРИЛОЖЕНИЕ А**

Листинг класса с запросами

*Startup.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using lab2;

using Microsoft.EntityFrameworkCore;

using lab3.Services;

using System.IO;

using lab3.Infrastracture;

using Microsoft.CodeAnalysis.CSharp.Syntax;

using lab3.Middleware;

using Microsoft.AspNetCore.Html;

namespace lab3

{

public class Startup

{

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddMemoryCache();

services.AddDistributedMemoryCache();

services.AddSession();

services.AddMemoryCache();

services.AddTransient<ApplicationContext>();

services.AddTransient<CachedInfos>();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

app.UseExceptionHandler("/Home/Error");

}

app.UseCacheMiddleware();

app.UseStaticFiles();

app.UseSession();

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

{

appBuilder.Run(async (context) => {

string FirstName = context.Request.Cookies["FirstName"];

string LastName = context.Request.Cookies["LastName"];

string HtmlString = "<HTML><HEAD>" +

"<TITLE>Информация</TITLE></HEAD>"+

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

if(FirstName != null && LastName != null)

{

HtmlString += "Имя:<BR><INPUT type = 'text' name = 'FirstName' value = " + FirstName + ">" +

"<BR>Фамилия:<BR><INPUT type = 'text' name = 'LastName' value = " + LastName + " >";

}

else

HtmlString += "<BODY><H1>Нет информации о клиенте</H1></BODY></HTML>";

await context.Response.WriteAsync(HtmlString);

});

});

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

{

appBuilder.Run(async (context) => {

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

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

CachedInfos cachedPosts = context.RequestServices.GetService<CachedInfos>();

IEnumerable<Post> posts = cachedPosts.GetPosts("Posts20");

HtmlString += "<BODY>" +

"<TABLE BORDER=1>";

HtmlString += "<TD>Код</TD>";

HtmlString += "<TD>Наименование должности</TD>";

foreach (var post in posts)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + post.postId + "</TD>";

HtmlString += "<TD>" + post.nameOfPost + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "<BR><BR>";

CachedInfos cachedStaff = context.RequestServices.GetService<CachedInfos>();

IEnumerable<Staff> staffs = cachedStaff.GetStaff("Staff20");

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

HtmlString += "<TD>Код сотрудника</TD>";

HtmlString += "<TD>Код должности</TD>";

HtmlString += "<TD>Код Поезда</TD>";

foreach (var staff in staffs)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + staff.staffId + "</TD>";

HtmlString += "<TD>" + staff.postId + "</TD>";

HtmlString += "<TD>" + staff.trainId + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "<BR><BR>";

CachedInfos cachedTrain = context.RequestServices.GetService<CachedInfos>();

IEnumerable<Train> trains = cachedTrain.GetTrain("Train20");

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

HtmlString += "<TD>Код поезда</TD>";

HtmlString += "<TD>Код типа поезда</TD>";

HtmlString += "<TD>Фирменный ли поезд</TD>";

foreach (var train in trains)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + train.trainId + "</TD>";

HtmlString += "<TD>" + train.typeId + "</TD>";

HtmlString += "<TD>" + train.isFirm + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "<BR><A href='/searchform1'>Форма поиска</A></BR>";

HtmlString += "<BR><A href='/info'>Информация о клиенте</A></BR>";

HtmlString += "<BR><A href='/'>Ввести информацию о клиенте</A></BR>";

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

await context.Response.WriteAsync(HtmlString);

});

});

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

appBuilder.Run(async (context) =>

{

string HtmlString = "<HTML><HEAD><TITLE>Форма поиска</TITLE></HEAD>" +

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

HtmlString += "<BODY><H1>Строка поиска</H1>";

HtmlString += "<FORM action = '/searchform1'> " +

"Выберите таблицу: <p><select name='table'>" +

"<option> Выберите таблицу </option>" +

"<option value = 'Post'> Post </option>" +

"<option value = 'Staff' > Staff </option>" +

"<option value = 'Train' > Train </option> " +

"</select>" +

"<p><input type = 'submit' value = 'вывести'></p>" +

"</p>";

string s = context.Session.Get<string>("tables") ?? context.Request.Query["table"];

if (context.Session.Get<string>("tables") != context.Request.Query["table"] && context.Request.Query["table"] != "Выберите таблицу")

s = context.Request.Query["table"];

context.Session.Set("tables", s);

string get = "";

switch (s)

{

case "Post":

CachedInfos cachedPosts = context.RequestServices.GetService<CachedInfos>();

IEnumerable<Post> posts = cachedPosts.GetPosts("Posts20");

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

HtmlString += "<TD>Код</TD>";

HtmlString += "<TD>Наименование должности</TD>";

foreach (var post in posts)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + post.postId + "</TD>";

HtmlString += "<TD>" + post.nameOfPost + "</TD>";

HtmlString += "</TR>";

}

HtmlString += "</TABLE>";

HtmlString += "Введите номер должности: <BR><input type = 'text' name = 'Choose'>";

HtmlString += "<input type = 'submit' value = 'Вывести'>";

HtmlString += "</p>";

get = context.Request.Query["Choose"];

if (get != "" && get != null)

{

int id = int.Parse(get);

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

HtmlString += "<TD>Код</TD>";

HtmlString += "<TD>Наименование должности</TD>";

HtmlString += "<TD>Номер сотрудника</TD>";

using (ApplicationContext db = new ApplicationContext())

{

var result = (from post in db.Posts

from staff in db.Staffs

where staff.postId == post.postId && post.postId == id

select new { post, staff }).Take(20).ToList();

foreach (var p in result)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + p.post.postId + "</TD>";

HtmlString += "<TD>" + p.post.nameOfPost + "</TD>";

HtmlString += "<TD>" + p.staff.staffId + "</TD>";

HtmlString += "</TR>";

}

}

HtmlString += "</table></Form>";

}

break;

case "Train":

HtmlString += "Желаете увидеть фирменные поезда? <input type = 'text' name = 'Choose'>";

HtmlString += "<input type = 'submit' value = 'Вывести'>";

HtmlString += "</p>";

get = context.Request.Query["Choose"];

if (get != null && get != "")

{

bool isFirm;

if (get == "Да")

isFirm = true;

else

isFirm = false;

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

HtmlString += "<TD>Номер поезда</TD>";

HtmlString += "<TD>Тип поезда</TD>";

HtmlString += "<TD>Фирменный ли</TD>";

using (ApplicationContext db = new ApplicationContext())

{

var result = (from train in db.Trains

where train.isFirm == isFirm

select train).Take(20).ToList();

foreach (Train t in result)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + t.trainId + "</TD>";

HtmlString += "<TD>" + t.typeId + "</TD>";

HtmlString += "<TD>" + t.isFirm + "</TD>";

HtmlString += "</TR>";

}

}

}

HtmlString += "</table>";

break;

case "Staff":

CachedInfos cachedStaff = context.RequestServices.GetService<CachedInfos>();

IEnumerable<Staff> staffs = cachedStaff.GetStaff("Staff20");

HtmlString += "Введите номер сотрудника <input type = 'text' name = 'Choose'>";

HtmlString += "<input type = 'submit' value = 'Вывести'>";

HtmlString += "</p>";

get = context.Request.Query["Choose"];

if(get != "" && get != null)

{

using (ApplicationContext db = new ApplicationContext())

{

var result = (from staff in db.Staffs

from post in db.Posts

where staff.postId == post.postId && staff.staffId == int.Parse(get)

select new { post, staff }).Take(20).ToList();

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

HtmlString += "<TD>Номер сотрудника</TD>";

HtmlString += "<TD>Название должности</TD>";

HtmlString += "<TD>Номер поезда</TD>";

foreach (var r in result)

{

HtmlString += "<TR>";

HtmlString += "<TD>" + r.staff.staffId + "</TD>";

HtmlString += "<TD>" + r.post.nameOfPost + "</TD>";

HtmlString += "<TD>" + r.staff.trainId + "</TD>";

HtmlString += "</TR>";

}

}

HtmlString += "</TABLE>";

}

break;

}

HtmlString += "<BR><A href='/table'>Главная</A></BR>";

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

await context.Response.WriteAsync(HtmlString);

});

});

app.Run((context) =>

{

string FirstName = "";

string LastName = "";

string HtmlString = "<HTML><HEAD>" +

"<TITLE>Пользователь</TITLE></HEAD>" +

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

"<BODY><FORM action ='/takeinfo'>" +

"Имя:<BR><INPUT type = 'text' name = 'FirstName' value = " + FirstName + ">" +

"<BR>Фамилия:<BR><INPUT type = 'text' name = 'LastName' value = " + LastName + " >" +

"<BR><BR><INPUT type ='submit' value='Сохранить в куки'>";

FirstName = context.Request.Query["FirstName"];

LastName = context.Request.Query["LastName"];

if (FirstName != null && LastName != null)

{

context.Response.Cookies.Append("Firstname", FirstName);

context.Response.Cookies.Append("LastName", LastName);

}

HtmlString += "<BR><A href='/table'>Главная</A></BR>";

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

return context.Response.WriteAsync(HtmlString);

});

}

}

}

*CacheMiddleware.cs*

using lab2;

using lab3.Infrastracture;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

namespace lab3.Middleware

{

public class CacheMiddlware

{

private readonly RequestDelegate \_next;

private int \_rowsNumber = 20;

string[] tables = { "Posts20", "Train20", "Staff20" };

ApplicationContext ac;

IMemoryCache cache;

public CacheMiddlware(RequestDelegate next, IMemoryCache cache)

{

this.\_next = next;

this.cache = cache;

}

public async Task Invoke(HttpContext context)

{

ac = new ApplicationContext();

int step = 0;

foreach(string table in tables)

{

if(step == 0)

{

var temptable = ac.Posts.Take(\_rowsNumber).ToList();

if (temptable != null)

{

cache.Set(table, temptable,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(258)));

step += 1;

}

}

else if(step == 1)

{

var temptable = ac.Trains.Take(\_rowsNumber).ToList();

if (temptable != null)

{

cache.Set(table, temptable,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(258)));

step += 1;

}

}

else

{

var temptable = ac.Staffs.Take(\_rowsNumber).ToList();

if (temptable != null)

{

cache.Set(table, temptable,

new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromSeconds(258)));

}

}

}

await \_next.Invoke(context);

}

}

public static class CacheExtensions

{

public static IApplicationBuilder UseCacheMiddleware(this IApplicationBuilder builder)

{

return builder.UseMiddleware<CacheMiddlware>();

}

}

}

*SessionExtension.cs*

using Microsoft.AspNetCore.Http;

using Newtonsoft.Json;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace lab3.Infrastracture

{

public static class SessionExtension

{

public static void Set<T>(this ISession session, string key, T value)

{

session.SetString(key, JsonConvert.SerializeObject(value));

}

public static T Get<T>(this ISession session, string key)

{

var value = session.GetString(key);

return value == null ? default(T) : JsonConvert.DeserializeObject<T>(value);

}

}

}