**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 4**

по дисциплине «Разработка приложений баз данных для информационных систем»

на тему: «Разработка моделей и контроллеров ASP.NET MVC приложения баз данных»

Вариант 9

Выполнил: студент гр. ИТП-31

Коршиков Н.В.

Принял: доцент

Асенчик О. Д.

Гомель 2020

**Цель работы:**

Ознакомиться с возможностями ASP.NET Core MVC и Entity Framework Core для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

**Задание:**

Создать с использованием ASP.NET Core MVC Web-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных. Разработать один компонент middleware, контроллеры и представления для выборки и отображения информации из не менее чем 3- таблиц базы данных с использованием механизма внедрение зависимостей.

Для этого необходимо создать:

1. Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Перечень таблиц предварительно согласовывается с преподавателем. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.
2. Класс контекста данных.
3. Другие классы, например, классы View Model и т.п. (при необходимости).
4. Компонент middleware, вызываемый в классе Startup, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.
5. Классы контроллеров (по одному на каждую таблицу базы данных) для обработки обращений пользователя, выборки данных из таблиц и вызова соответствующих представлений для отображения выбранных данных.
6. Разработать представления для отображения данных из таблиц, выбранных контроллерами. Представления, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».
7. Используя предварительно созданный и сконфигурированный в классе Startup профиль кэширования, подключить кэширование вывода для страниц с использованием атрибута ResponseCache для соответствующих методов контроллера. Данные в кэше хранить неизменными в течение 2\*N+240 секунд, где N- номер вашего варианта.
8. С использованием средств разработчика браузера (Chrome, Firefox) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием атрибута ResponseCache.

**Вариант задания:**

Предметная область задания изображена на рисунке 1.

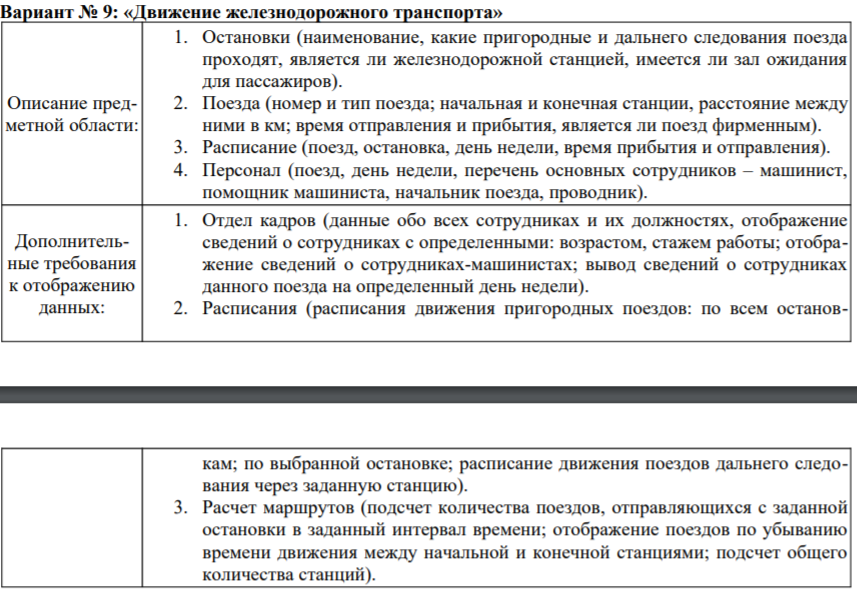


Рисунок 1 – Предметная область

**Ход работы:**

Результаты работы контроллеров приведены на рисунках 2 - 4.

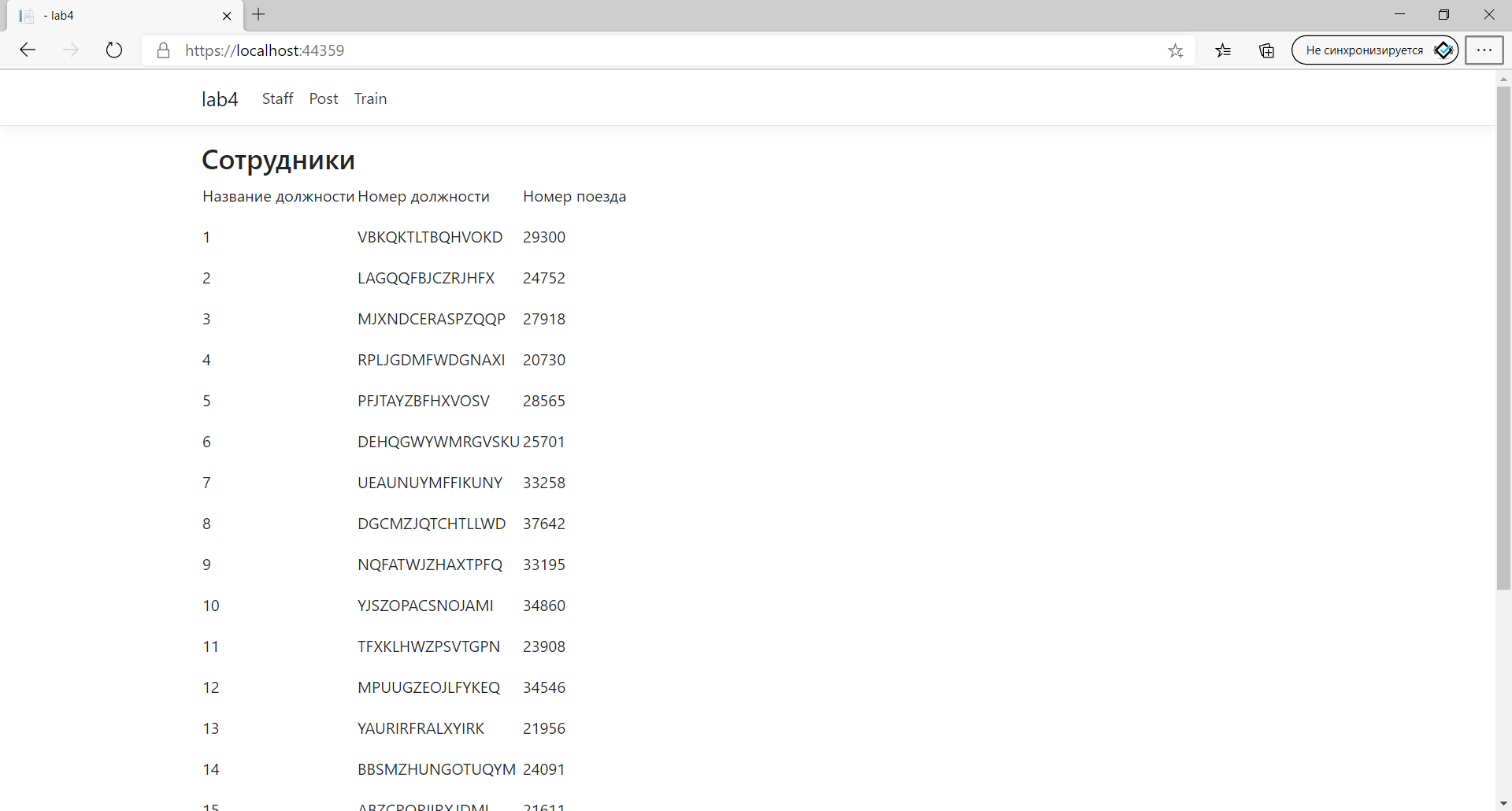


Рисунок 2 – результат работы контроллера StaffController

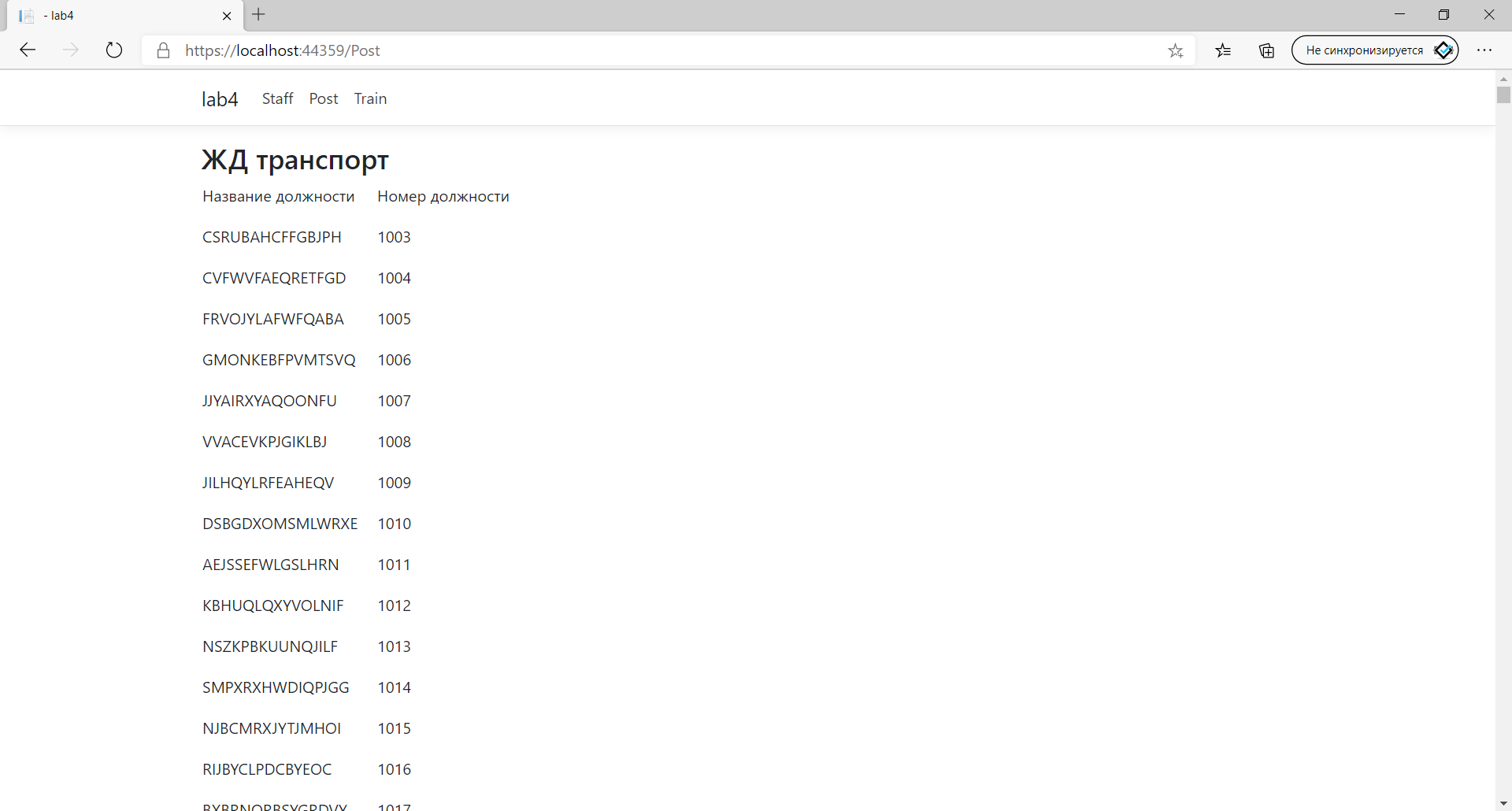


Рисунок 3 – результат работы контроллера PostController

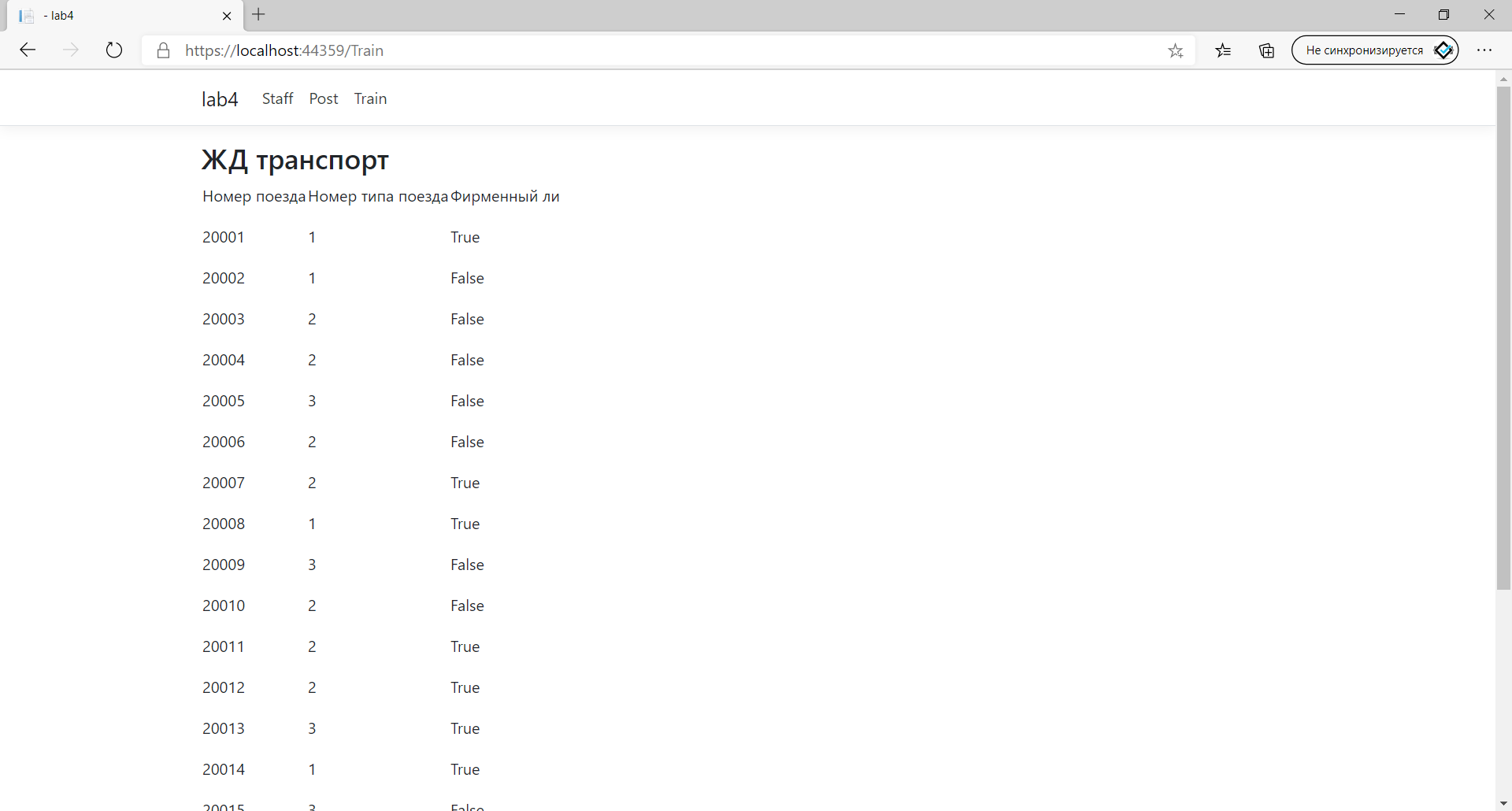


Рисунок 4 – результат работы контроллера TrainController

**Вывод:**

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

**ПРИЛОЖЕНИЕ А**

Листинг класса с запросами

Строка подключения из конфигурационного файла:

"ConnectionStrings": {

"SqlServerConnection": "Server=DESKTOP-OI4GN8K\\SQLEXPRESS;Database=Course;Trusted\_Connection=True;MultipleActiveResultSets=true"

}

*Startup.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab2;

using lab3.Middleware;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace lab4

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddControllersWithViews();

string connection = Configuration.GetConnectionString("SqlServerConnection");

services.AddDbContext<ApplicationContext>(options => options.UseSqlServer(connection));

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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

app.UseHsts();

}

app.UseDbInitializerMiddleware();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Staff}/{action=ShowTable}");

});

}

}

}

*DbInitializeMiddleware.cs*

using lab2;

using lab3.Infrastracture;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Connections;

using Microsoft.AspNetCore.Http;

using Microsoft.Data.SqlClient;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.SqlServer.Storage.Internal;

using Microsoft.Extensions.Caching.Memory;

using Microsoft.Extensions.Configuration;

using Microsoft.IdentityModel.Protocols;

using System;

using System.Collections;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Threading.Tasks;

namespace lab3.Middleware

{

public class DbInitializeMiddlware

{

private readonly RequestDelegate \_next;

public DbInitializeMiddlware(RequestDelegate next, IConfiguration configuration)

{

this.\_next = next;

}

public async Task Invoke(HttpContext context, ApplicationContext ac)

{

var Staff = ac.Staffs;

if (Staff.Count() == 0)

{

ac.Database.ExecuteSqlCommand("FillPosts");

ac.Database.ExecuteSqlCommand("FillTrains");

ac.Database.ExecuteSqlCommand("FillStaff");

}

await \_next.Invoke(context);

}

}

public static class DbInitializerExtensions

{

public static IApplicationBuilder UseDbInitializerMiddleware(this IApplicationBuilder builder)

{

return builder.UseMiddleware<DbInitializeMiddlware>();

}

}

}

*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);

}

}

}

*PostController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using lab2;

using Microsoft.AspNetCore.Mvc;

namespace lab4.Controllers

{

public class PostController : Controller

{

ApplicationContext ac;

public PostController (ApplicationContext ac)

{

this.ac = ac;

}

[ResponseCache(Duration = 258)]

public IActionResult ShowTable()

{

return View(ac.Posts);

}

}

}

*StaffController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab2;

using lab4.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Query;

namespace lab4.Controllers

{

public class StaffController : Controller

{

ApplicationContext ac;

public StaffController(ApplicationContext ac)

{

this.ac = ac;

}

[ResponseCache(Duration = 258)]

public IActionResult ShowTable()

{

var result = ac.Staffs.Include(p => p.Post).Take(20);

return View(result.ToList());

}

}

}

*TrainController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab2;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Mvc;

namespace lab4.Controllers

{

public class TrainController : Controller

{

ApplicationContext ac;

public TrainController(ApplicationContext ac)

{

this.ac = ac;

}

[ResponseCache(Duration = 258)]

public IActionResult ShowTable()

{

var t = ac.Trains.Take(20);

return View(t);

}

}

}

*ShowTable.cshtml(post)*

@model IEnumerable<lab2.Post>

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div>

<h3>ЖД транспорт</h3>

<table>

<tr class="header">

<td><p>Название должности</p></td>

<td><p>Номер должности</p></td>

<td></td>

</tr>

@foreach (lab2.Post p in Model)

{

<tr>

<td><p>@p.nameOfPost</p></td>

<td><p>@p.postId</p></td>

</tr>

}

</table>

</div>

*ShowTable.cshtml(staff)*

@model IEnumerable<lab2.Staff>

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div>

<h3>Сотрудники</h3>

<table>

<tr class="header">

<td><p>Название должности</p></td>

<td><p>Номер должности</p></td>

<td><p>Номер поезда</p></td>

</tr>

@foreach (var item in Model)

{

<tr>

<td><p>@Html.DisplayFor(modelItem => item.staffId)</p></td>

<td><p>@Html.DisplayFor(modelItem => item.Post.nameOfPost)</p></td>

<td><p>@Html.DisplayFor(modelItem => item.trainId)</p></td>

</tr>

}

</table>a

</div>

*ShowTable.cshtml(train)*

@model IEnumerable<lab2.Train>

@{

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<div>

<h3>ЖД транспорт</h3>

<table>

<tr class="header">

<td><p>Номер поезда</p></td>

<td><p>Номер типа поезда</p></td>

<td><p>Фирменный ли</p></td>

<td></td>

</tr>

@foreach (lab2.Train t in Model)

{

<tr>

<td><p>@t.trainId</p></td>

<td><p>@t.typeId</p></td>

<td><p>@t.isFirm</p></td>

</tr>

}

</table>

</div>