**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 2

Выполнила: студентка гр. ИТП-31

Гайкевич Т. Г.

Принял: доцентАсенчик 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 внедрения зависимостей.

Для выполнения задания необходимо создать:

* 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.
* Класс контекста данных.
* Другие классы, например, классы *View Model* и т.п. (при необходимости).
* Компонент *middleware*, вызываемый в классе *Startup*, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.
* 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.
* 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».
* 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.
* С использованием средств разработчика браузера (*Chrome, Firefox*) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием атрибута *ResponseCach*e.
* Для проверки преподавателем следует разместить разработанный проект на *GitHub*.

**Ход выполнения:**

Сначала были созданы модели базы данных. Они были помещены в папку *Models*.

Далее был создан класс контекста базы данных. Он был помещён в папку *Data*.

После этого был написан класс, роль которого заключается в заполнении базы данных начальными записями. Он заполняет таблицу базы данных только в том случае, если в ней нет записей.

Потом был написан компонент промежуточного слоя, который создаёт класс для инициализации базы данных и инициализирует её. Он был добавлен в метод *Configure* код класса *Startup*.

Далее были созданы классы контроллеров. Они все были помещены в папку Controllers.

Для всех контроллеров были созданы папки в папке *Views*. В каждом контроллере есть метод *Index*. И поэтому в каждой папке с представлениями были созданы соответствующие представления для этих методов.

Эти представления просто выполняют вывод таблиц базы данных в *html* страницу.

Для каждого метода *Index* был прописан атрибут *ResponseCache* с длительностью кэширования 244 секунды и хранением в любой области кеша.

На рисунках 1 – 3 изображены все доступные *html* страницы.

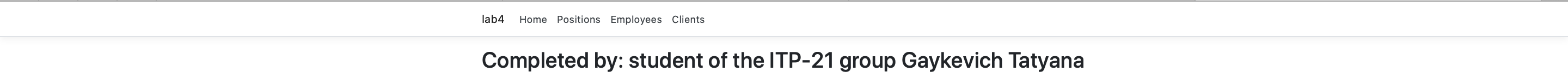


Рисунок 1 – Корневой каталог веб-приложения

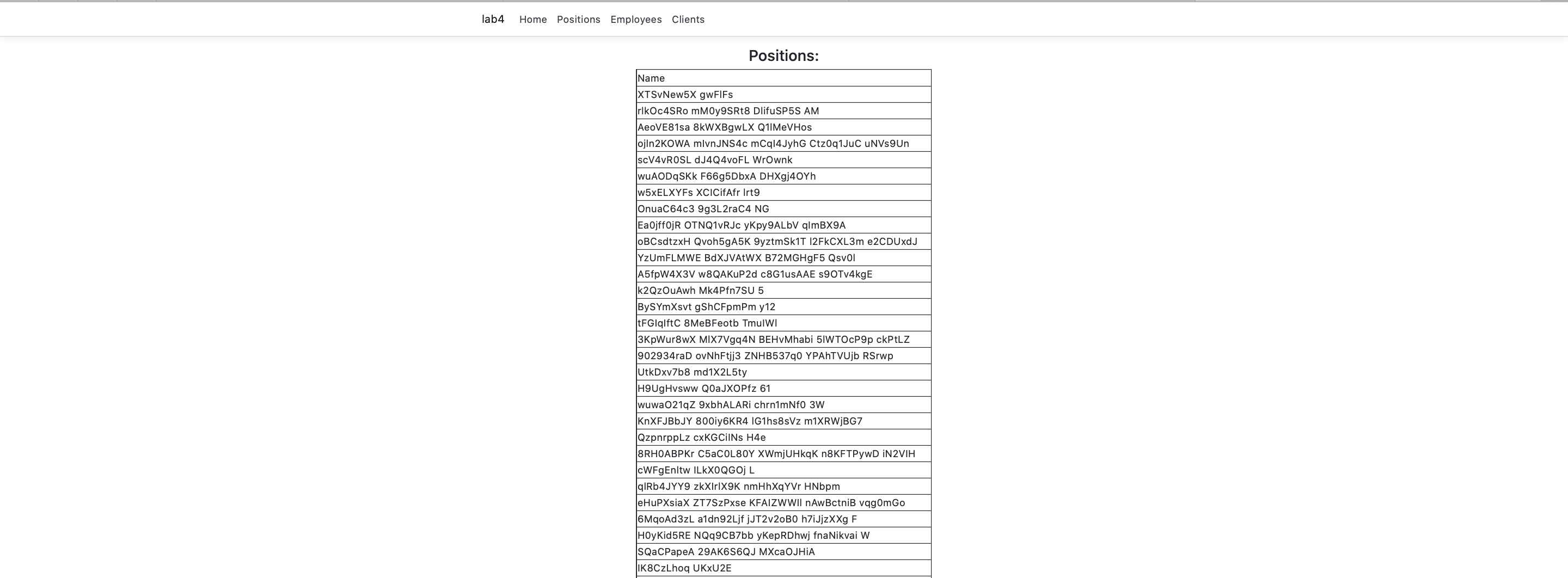


Рисунок 2 – Вывод приложения при нажатии на вкладку *Positions*



Рисунок 3 – Вывод приложения при нажатии на вкладку *Employees*

Можно заметить, что все идентификационные поля при выводе сотрудников заменены на наименования этих полей.

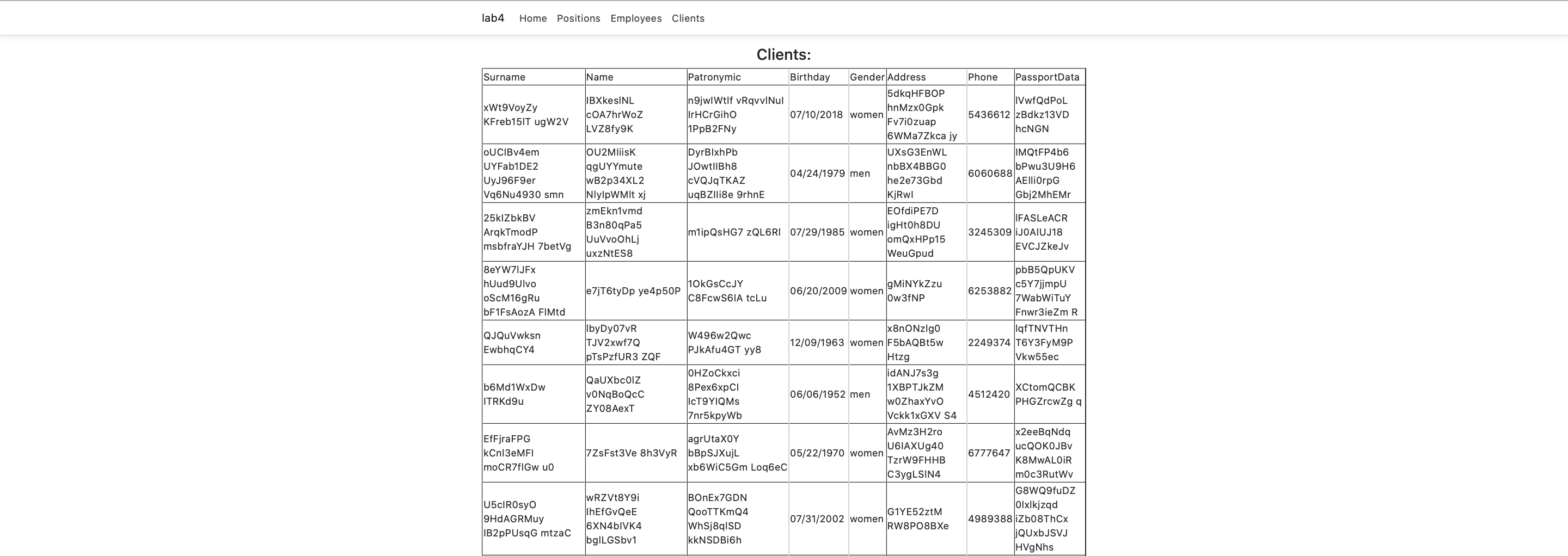


Рисунок 4 – Вывод приложения при нажатии на вкладку *Clients*

Структура созданного приложения изображена на рисунке 5.

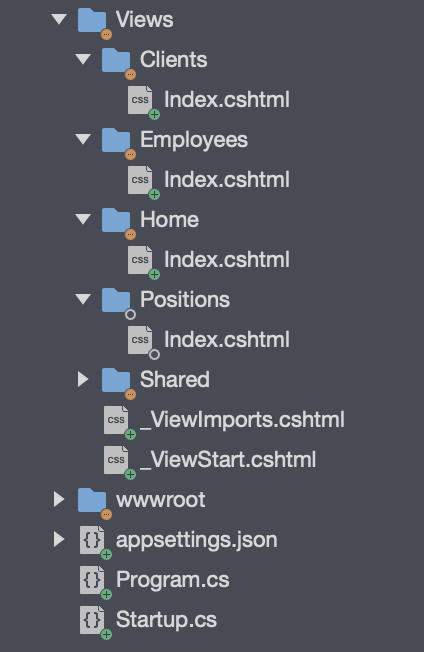
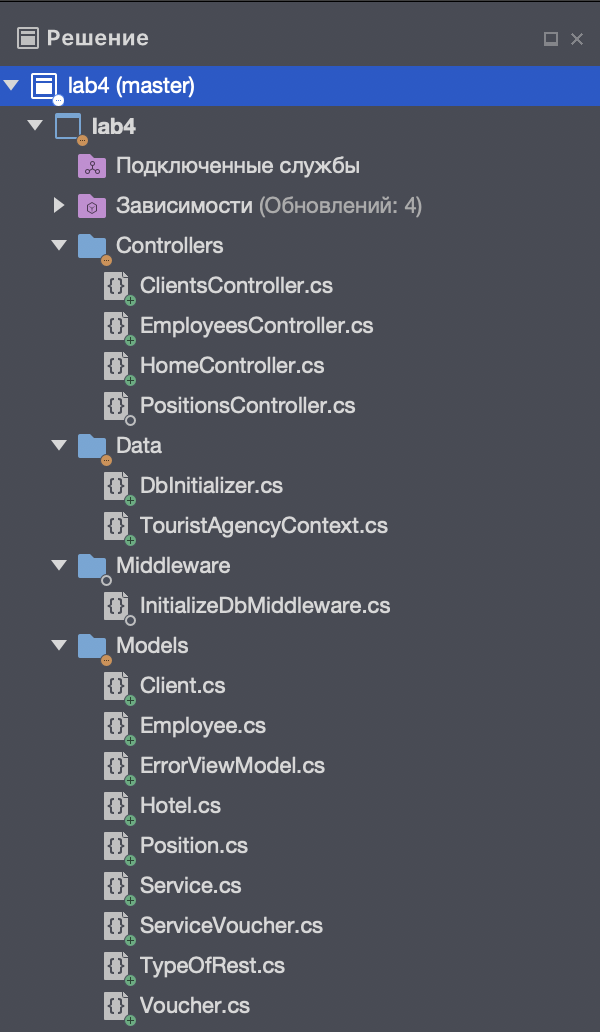


Рисунок 5 – Структура созданного приложения

На гитхабе был создан репозиторий для проекта: https://github.com/Tanya-Gaykevich/DB/tree/main/lab4

**Вывод:** в ходе выполнения был создан проект *ASP.NET CORE MVC*. Были созданы модели, контроллеры и представления. Был создан компонент промежуточного слоя для заполнения базы данных.

**ПРИЛОЖЕНИЕ А**

Листинг созданных классов

Листинг класса *Startup.cs*

using lab4.Middleware;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

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)

{

string connectionString = Configuration.GetConnectionString("DefaultConnection");

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

services.AddControllersWithViews();

}

// 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");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseDbInitializer();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

});

}

}

}

Листинг класса *D**bInitializer.cs*

using Microsoft.AspNetCore.Authentication;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace lab4.Data

{

public class DbInitializer

{

int \_refferenceTableSize;

int \_operationalTableSize;

public DbInitializer(int refferenceTableSize, int operationalTableSize)

{

\_refferenceTableSize = refferenceTableSize;

\_operationalTableSize = operationalTableSize;

}

public void Initialize(TouristAgencyContext dbContext)

{

Random rand = new Random();

dbContext.SaveChanges();

if (!dbContext.Positions.Any())

{

for (int i = 0; i < \_refferenceTableSize; i++)

{

dbContext.Positions.Add(new Position

{

Name = GetRandomString(50),

});

}

}

dbContext.SaveChanges();

if (!dbContext.Clients.Any())

{

for (int i = 0; i < \_refferenceTableSize; i++)

{

dbContext.Clients.Add(new Client

{

Surname = GetRandomString(50),

Name = GetRandomString(50),

Patronymic = GetRandomString(50),

Birthday = GetRandomDate(new DateTime(1950, 1, 1), DateTime.Now),

Gender = rand.Next(0, 1) == 1 ? true : false,

Address = GetRandomString(50),

Phone = rand.Next(1000000, 9999999),

PassportData = GetRandomString(50)

});

}

}

dbContext.SaveChanges();

if (!dbContext.Employees.Any())

{

var positions = dbContext.Positions.ToList();

for (int i = 0; i < \_operationalTableSize; i++)

{

var position = positions.ElementAt(rand.Next(dbContext.Positions.Count() - 1));

dbContext.Employees.Add(new Employee

{

Surname = GetRandomString(50),

Name = GetRandomString(50),

Patronymic = GetRandomString(50),

Birthday = GetRandomDate(new DateTime(1950, 1, 1), DateTime.Now),

PositionId = position.Id,

Position = position

});

}

}

dbContext.SaveChanges();

}

public string GetRandomString(int maxLength)

{

Random rand = new Random();

int length = rand.Next(maxLength / 3, maxLength);

var chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789";

var str = new char[length];

for (int i = 0; i < length; i++)

{

str[i] = chars[rand.Next(chars.Length)];

}

return new string(str);

}

public DateTime GetRandomDate(DateTime minDate, DateTime maxDate)

{

Random rand = new Random();

int range = (maxDate - minDate).Days;

return minDate.AddDays(rand.Next(range));

}

}

}

Листинг класса *EmployeesController.cs*

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

// For more information on enabling MVC for empty projects, visit https://go.microsoft.com/fwlink/?LinkID=397860

namespace lab4.Controllers

{

public class EmployeesController : Controller

{

TouristAgencyContext \_dbContext;

public EmployeesController(TouristAgencyContext dbContext)

{

\_dbContext = dbContext;

}

// GET: /<controller>/

[ResponseCache(Location = ResponseCacheLocation.Any, Duration = 244)]

public IActionResult Index()

{

return View(\_dbContext.Employees

.Include(emp=>emp.Position)

.Take(500));

}

}

}

Листинг класса *InitializeDbMiddleware*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab4.Data;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

namespace lab4.Middleware

{

// You may need to install the Microsoft.AspNetCore.Http.Abstractions package into your project

public class InitializeDbMiddleware

{

private readonly RequestDelegate \_next;

public InitializeDbMiddleware(RequestDelegate next)

{

\_next = next;

}

public Task Invoke(HttpContext httpContext, TouristAgencyContext dbContext)

{

DbInitializer initializer = new DbInitializer(500, 5000);

initializer.Initialize(dbContext);

return \_next(httpContext);

}

}

// Extension method used to add the middleware to the HTTP request pipeline.

public static class InitializeDbMiddlewareExtensions

{

public static IApplicationBuilder UseDbInitializer(this IApplicationBuilder builder)

{

return builder.UseMiddleware<InitializeDbMiddleware>();

}

}

}

Листинг класса *Index.cshtml* из папки *Views/Employees*

@model IEnumerable<lab4.Employee>

@{

ViewData["title"] = "Employee";

}

<!DOCTYPE html>

<html>

<head>

<title>A LIST OF EMPLOYEES</title>

</head>

<body>

<h3 align="center">Employees:</h3>

<table align="left" border="1">

<tr>

<td>Surmane</td>

<td>Name</td>

<td>Patronymic</td>

<td>Birthday</td>

<td>Position name</td>

</tr>

@foreach (var employee in Model)

{

<tr>

<td>@employee.Surname</td>

<td>@employee.Name</td>

<td>@employee.Patronymic</td>

<td>@employee.Birthday.ToShortDateString()</td>

<td>@employee.Position.Name</td>

</tr>

}

</table>

</body>

</html>