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»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ

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

Вариант №6

Выполнил:

студент группы ИТП-31, Зайцев А.В.

Принял:

доцент Асенчик О.Д.

Гомель 2020

**1. 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.

**2. Ход работы и результаты:**

Исходные коды класса *Startup*, классов моделей и класса контекста данных:

*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.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using Microsoft.EntityFrameworkCore;

using Lab4\_WebApp.Data;

using Lab4\_WebApp.Models;

using Lab4\_WebApp.Middleware;

namespace Lab4\_WebApp

{

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<Lab4\_CarSharingContext>(options => options.UseSqlServer(connectionString));

services.AddDistributedMemoryCache();

services.AddSession();

services.AddControllersWithViews(options =>

{

options.CacheProfiles.Add("CacheProfile",

new CacheProfile()

{

Duration = 262

});

});

}

// 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.UseSession();

app.UseDbInitializer();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

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

});

}

}

}

*ErrorViewModel.cs*

using System;

namespace Lab4\_WebApp.Models

{

public class ErrorViewModel

{

public string RequestId { get; set; }

public bool ShowRequestId => !string.IsNullOrEmpty(RequestId);

}

}

*Car.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.ComponentModel.DataAnnotations.Schema;

namespace Lab4\_WebApp.Models

{

public class Car

{

public int CarId { get; set; }

public int RegNum { get; set; }

public string VINcode { get; set; }

public int EngineNum { get; set; }

[Column(TypeName = "money")]

public decimal Price { get; set; }

[Column(TypeName = "money")]

public decimal RentalPrice { get; set; }

public DateTime IssueDate { get; set; }

public string Specs { get; set; }

public DateTime TechnicalMaintenanceDate { get; set; }

public bool SpecMark { get; set; }

public bool ReturnMark { get; set; }

public int EmployeeId { get; set; }

public int CarModelId { get; set; }

public virtual Employee Employee { get; set; }

public virtual CarModel CarModel { get; set; }

}

}

*CarModel.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Lab4\_WebApp.Models

{

public class CarModel

{

public int CarModelId { get; set; }

/// <summary>

/// Марка авто-производителя

/// </summary>

public string Brand { get; set; }

/// <summary>

/// Название модели авто

/// </summary>

public string Name { get; set; }

public string Description { get; set; }

public virtual ICollection<Car> Cars { get; set; }

public CarModel()

{

Cars = new List<Car>();

}

}

}

*Employee.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Lab4\_WebApp.Models

{

public class Employee

{

public int EmployeeId { get; set; }

public string Post { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Patronymic { get; set; }

public DateTime EmploymentDate { get; set; }

public virtual ICollection<Car> Cars { get; set; }

public Employee()

{

Cars = new List<Car>();

}

}

}

*Lab4\_CarSharingContext.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.EntityFrameworkCore;

using Lab4\_WebApp.Models;

namespace Lab4\_WebApp.Data

{

public class Lab4\_CarSharingContext : DbContext

{

public Lab4\_CarSharingContext(DbContextOptions<Lab4\_CarSharingContext> options) : base(options)

{

}

public DbSet<Service> Services { get; set; }

public DbSet<Employee> Employees { get; set; }

public DbSet<CarModel> CarModels { get; set; }

}

}

Строка подключения к базе данных из конфигурационного файла:

*appsettings.json*

{

"ConnectionStrings": {

"DefaultConnection": "Server=DESKTOP-J4QINE6\\SQLEXPRESS;Database=Lab4\_CarSharing;Trusted\_Connection=True;"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

Исходный код компонента *middleware* для инициализации базы данных путем заполнения ее таблиц тестовым набором записей:

*DbInitializeMiddlware.cs*

using Lab4\_WebApp.Data;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Lab4\_WebApp.Middleware

{

public class DbInitializerMiddleware

{

private readonly RequestDelegate \_next;

public DbInitializerMiddleware(RequestDelegate next)

{

\_next = next;

}

public Task Invoke(HttpContext context, IServiceProvider serviceProvider, Lab4\_CarSharingContext db)

{

if (!(context.Session.Keys.Contains("starting")))

{

DbInitializer.Initialize(db);

context.Session.SetString("starting", "Yes");

}

return \_next.Invoke(context);

}

}

public static class DbInitializerExtensions

{

public static IApplicationBuilder UseDbInitializer(this IApplicationBuilder builder)

{

return builder.UseMiddleware<DbInitializerMiddleware>();

}

}

}

*DbInitializer.cs*

using Lab4\_WebApp.Data;

using Microsoft.EntityFrameworkCore.Internal;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography.X509Certificates;

using System.Threading.Tasks;

namespace Lab4\_WebApp.Data

{

public static class DbInitializer

{

private static char[] letters = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz".ToCharArray();

private static Random random = new Random();

public static void Initialize(Lab4\_CarSharingContext db)

{

int rowCount;

int rowIndex;

int minStringLength;

int maxStringLength;

db.Database.EnsureCreated();

if (!db.CarModels.Any())

{

rowCount = 500;

string carModelName;

string carModelDescription;

string carBrand;

for(rowIndex = 0; rowIndex < rowCount; rowIndex++)

{

minStringLength = 4;

maxStringLength = 20;

carModelName = GetString(minStringLength, maxStringLength);

carBrand = GetString(minStringLength, maxStringLength);

minStringLength = 50;

maxStringLength = 1000;

carModelDescription = GetString(minStringLength, maxStringLength);

db.CarModels.Add(new Models.CarModel { Name = carModelName, Description = carModelDescription, Brand = carBrand });

}

db.SaveChanges();

}

if (!db.Employees.Any())

{

rowCount = 500;

string employeePost;

string employeeName;

string employeeSurname;

string employeePatronymic;

DateTime employeeEmploymentDate;

for (rowIndex = 0; rowIndex < rowCount; rowIndex++)

{

minStringLength = 2;

maxStringLength = 20;

employeePost = GetString(minStringLength, maxStringLength);

employeeName = GetString(minStringLength, maxStringLength);

employeeSurname = GetString(minStringLength, maxStringLength);

employeePatronymic = GetString(minStringLength, maxStringLength);

employeeEmploymentDate = GetDateTime();

db.Employees.Add(new Models.Employee { Name = employeeName, Surname = employeeSurname, Patronymic = employeePatronymic, Post = employeePost, EmploymentDate = employeeEmploymentDate });

}

db.SaveChanges();

}

if (!db.Cars.Any())

{

rowCount = 20000;

int carRegNum;

string carVINcode;

int carEngineNum;

decimal carPrice;

decimal carRentalPrice;

DateTime carIssueDate;

string carSpecs;

DateTime carTechnicalMaintainceDate;

bool carSpecMark;

bool carReturnMark;

int carEmployeeId;

int carCarModelId;

for (rowIndex = 0; rowIndex < rowCount; rowIndex++)

{

carRegNum = random.Next(1000000000, 2000000000);

carEngineNum = random.Next(10000000, 100000000);

carPrice = Convert.ToDecimal(random.Next(150000, 2000000));

carRentalPrice = Convert.ToDecimal(random.Next(1500, 10000));

carSpecMark = random.Next(0, 1) == 1 ? true : false;

carReturnMark = random.Next(0, 1) == 1 ? true : false;

carIssueDate = GetDateTime(2000);

carTechnicalMaintainceDate = GetDateTime(2020);

carEmployeeId = random.Next(1, 499);

carCarModelId = random.Next(1, 499);

minStringLength = 21;

maxStringLength = 200;

carSpecs = GetString(minStringLength, maxStringLength);

minStringLength = 4;

maxStringLength = 20;

carVINcode = GetString(minStringLength, maxStringLength);

db.Cars.Add(new Models.Car

{

CarModelId = carCarModelId,

EmployeeId = carEmployeeId,

EngineNum = carEngineNum,

RegNum = carRegNum,

Price = carPrice,

RentalPrice = carRentalPrice,

SpecMark = carSpecMark,

ReturnMark = carReturnMark,

IssueDate = carIssueDate,

TechnicalMaintenanceDate = carTechnicalMaintainceDate,

Specs = carSpecs,

VINcode = carVINcode

});

}

db.SaveChanges();

}

}

private static string GetString(int minStringLength, int maxStringLength)

{

string result = "";

int stringLimit = minStringLength + random.Next(maxStringLength - minStringLength);

int stringPosition;

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

{

stringPosition = random.Next(letters.Length);

result += letters[stringPosition];

}

return result;

}

private static DateTime GetDateTime()

{

DateTime start = new DateTime(1995, 1, 1);

int range = (DateTime.Today - start).Days;

return start.AddDays(random.Next(range));

}

private static DateTime GetDateTime(int begin)

{

DateTime start = new DateTime(begin, 1, 1);

int range = (DateTime.Today - start).Days;

return start.AddDays(random.Next(range));

}

}

}

Исходные коды контроллеров и представлений:

Контроллеры:

*HomeController.cs*

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using lab4.Models;

namespace lab4.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

public IActionResult Privacy()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

*CarModelsController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab4\_WebApp.Data;

using Microsoft.AspNetCore.Mvc;

namespace Lab4\_WebApp.Controllers

{

public class CarModelsController : Controller

{

private Lab4\_CarSharingContext db;

public CarModelsController(Lab4\_CarSharingContext context)

{

db = context;

}

[ResponseCache(CacheProfileName = "CacheProfile")]

public IActionResult Index()

{

return View(db.CarModels.Take(20).ToList());

}

}

}

*CarsController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab4\_WebApp.Data;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace Lab4\_WebApp.Controllers

{

public class CarsController : Controller

{

private Lab4\_CarSharingContext db;

public CarsController(Lab4\_CarSharingContext context)

{

db = context;

}

[ResponseCache(CacheProfileName = "CacheProfile")]

public IActionResult Index()

{

return View(db.Cars.Include(s => s.CarModel).Take(20).ToList());

}

}

}

*EmployeesController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Lab4\_WebApp.Data;

using Microsoft.AspNetCore.Mvc;

namespace Lab4\_WebApp.Controllers

{

public class EmployeesController : Controller

{

private Lab4\_CarSharingContext db;

public EmployeesController(Lab4\_CarSharingContext context)

{

db = context;

}

[ResponseCache(CacheProfileName = "CacheProfile")]

public IActionResult Index()

{

return View(db.Employees.Take(20).ToList());

}

}

}

Представления:

*Home\Index.cshtml*

@{ ViewData["Title"] = "Home Page"; }

<div class="text-center">

<h1 class="display-4">Welcome!</h1>

<h1>Choose table for showing.</h1>

</div>

Представление *Cars\Index.cshtml*

@model IEnumerable<Car>

@{

ViewData["Title"] = "Table 'Cars'";

}

<h1>@ViewData["Title"]</h1>

<p align="center">

<table border="1">

<tr>

<td>VINcode</td>

<td>RegNum</td>

<td>EngineNum</td>

<td>IssueDate</td>

<td>TechnicalMaintenanceDate</td>

<td>Specs</td>

<td>ReturnMark</td>

<td>SpecMark</td>

<td>Price</td>

<td>RentalPrice</td>

</tr>

@foreach (var genre in Model)

{

<tr>

<td>@genre.VINcode</td>

<td>@genre.RegNum</td>

<td>@genre.EngineNum</td>

<td>@genre.IssueDate</td>

<td>@genre.TechnicalMaintenanceDate</td>

<td>@genre.Specs</td>

<td>@genre.ReturnMark</td>

<td>@genre.SpecMark</td>

<td>@genre.Price</td>

<td>@genre.RentalPrice</td>

</tr>

}

</table>

</p>

Представление *CarModels\Index.cshtml*

@model IEnumerable<CarModel>

@{

ViewData["Title"] = "Table 'CarModels'";

}

<h1>@ViewData["Title"]</h1>

<p align="center">

<table border="1">

<tr>

<td>Brand</td>

<td>Name</td>

<td>Description</td>

</tr>

@foreach (var genre in Model)

{

<tr>

<td>@genre.Brand</td>

<td>@genre.Name</td>

<td>@genre.Description</td>

</tr>

}

</table>

</p>

Представление *Employees\Index.cshtml*

@model IEnumerable<Employee>

@{

ViewData["Title"] = "Table 'Employees'";

}

<h1>@ViewData["Title"]</h1>

<p align="center">

<table border="1">

<tr>

<td>Post</td>

<td>Name</td>

<td>Surname</td>

<td>Patronymic</td>

<td>EmploymentDate</td>

</tr>

@foreach (var genre in Model)

{

<tr>

<td>@genre.Post</td>

<td>@genre.Name</td>

<td>@genre.Surname</td>

<td>@genre.Patronymic</td>

<td>@genre.EmploymentDate</td>

</tr>

}

</table>

</p>

Графические представления отображения представлений в браузере:

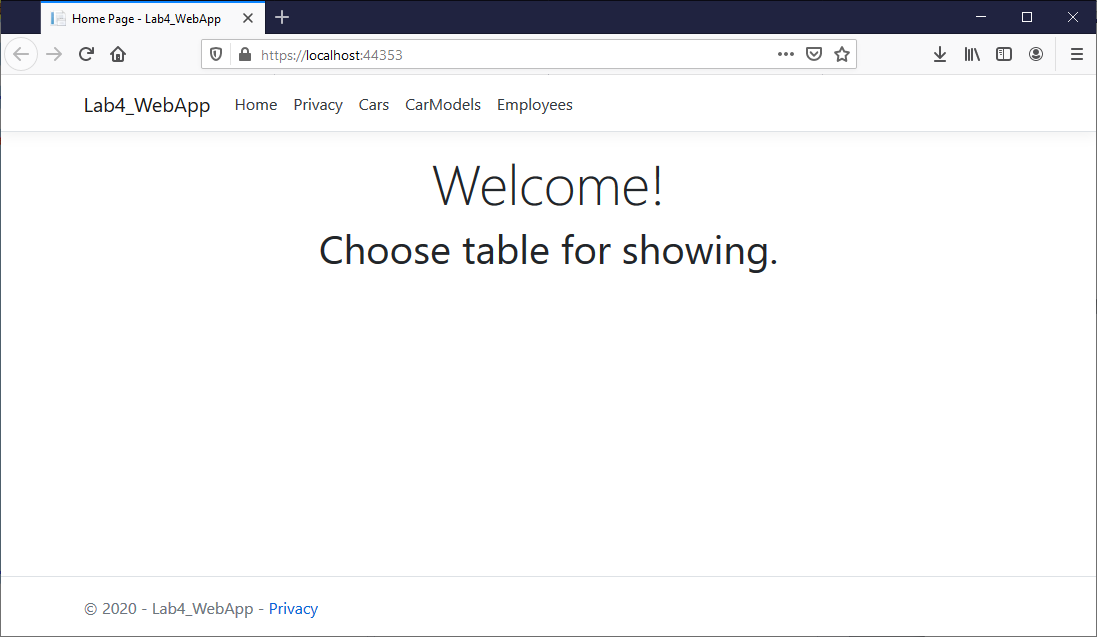


Рисунок 1 – Отображения представления *Home\Index.cshtml* в браузере

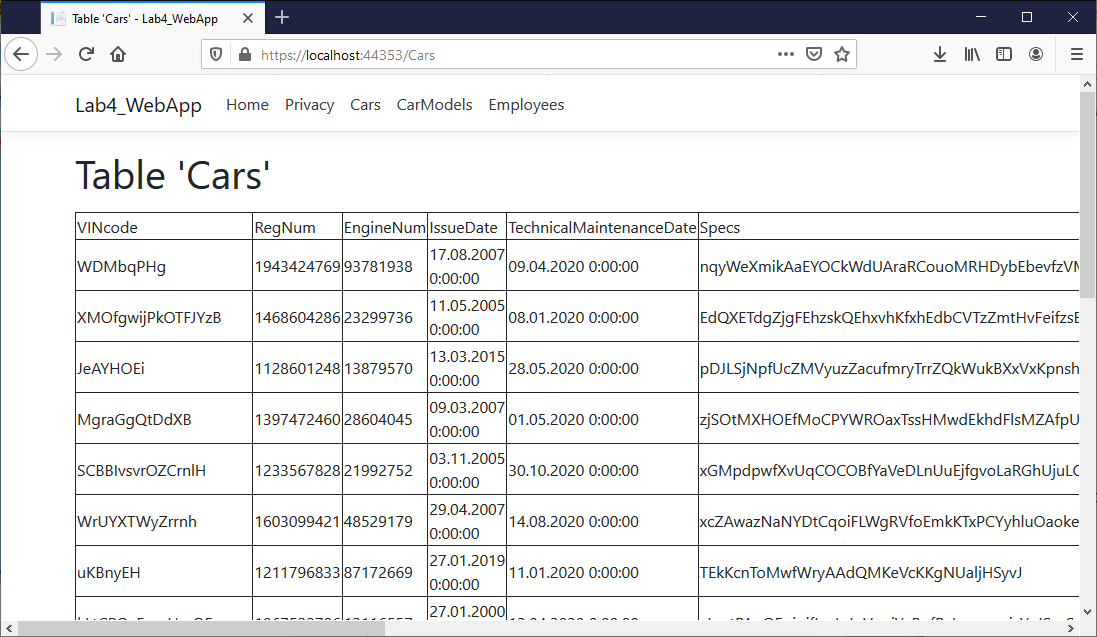


Рисунок 2 – Отображения представления *Cars\Index.cshtml* в браузере

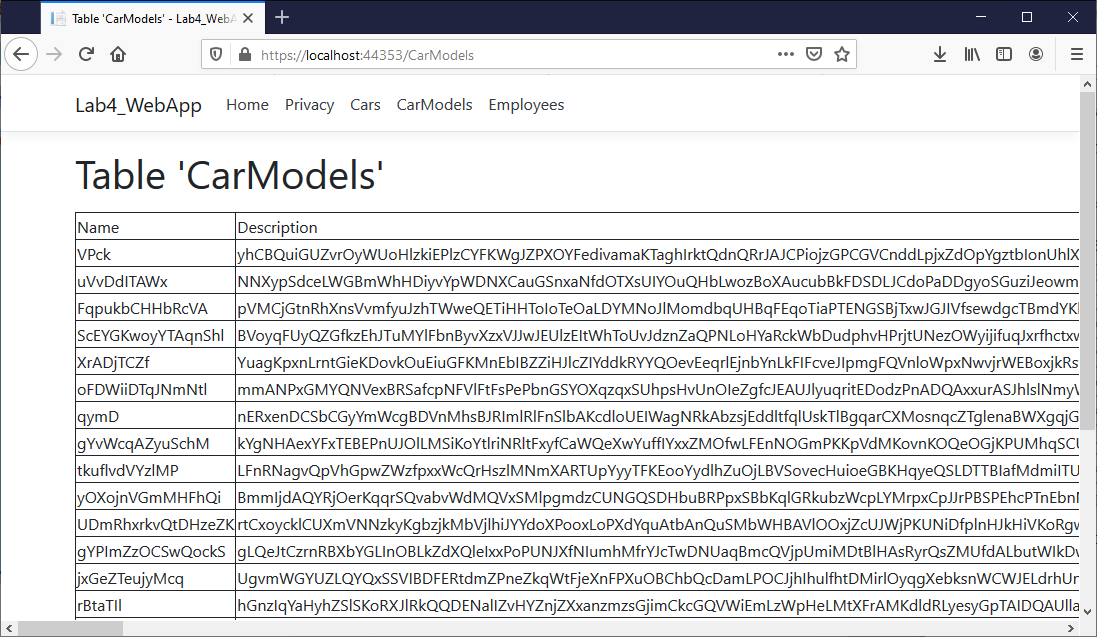


Рисунок 3 – Отображения представления *CarModels/Index.cshtml* в браузере

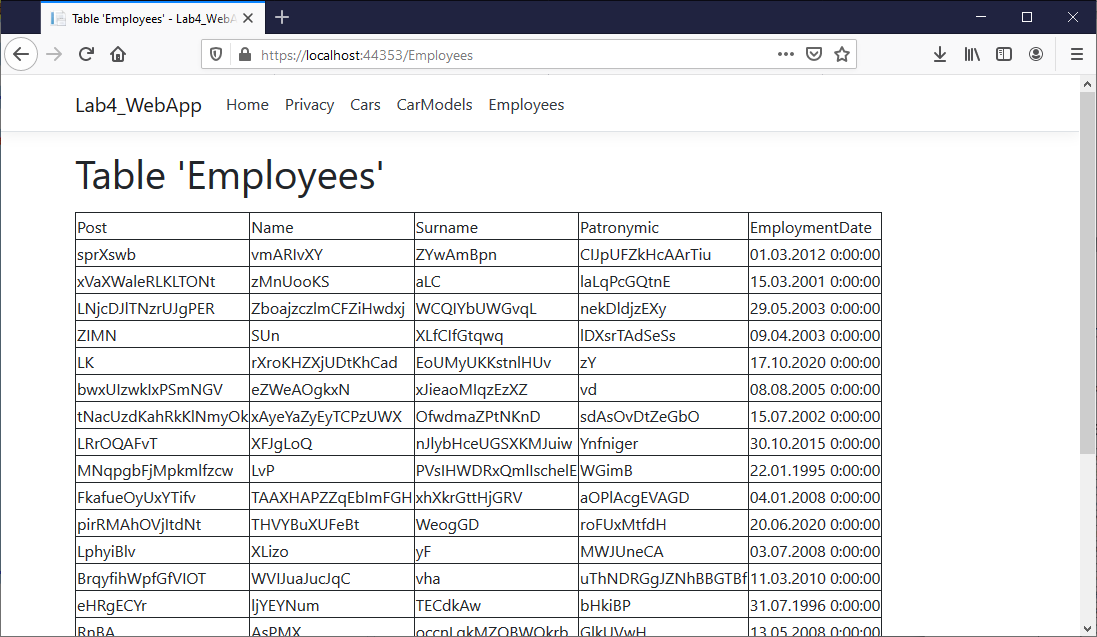


Рисунок 4 – Отображения представления *Employees\Index.cshtml* в браузере

**3. Вывод:** в результате выполнения данной лабораторной работы были изучены возможности *ASP.NET Core MVC* *и Entity Framework Core* для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.