**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ**

**УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ЛАБОРАТОРНАЯ РАБОТА №5

**по дисциплине: «Разработка интерфейса приложения баз данных с использованием с использованием аутентификации и авторизации»**

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

Выполнил: студент гр. ИТП-31

Солодков М.А

Принял: доцент

Асенчик О.Д.

Гомель 2019

**Цель работы:** научиться использовать ASP.NET MVC Сore для создания типовых web-приложений для работы с информацией из реляционных баз данных.

**Задание**

Используя разработанный ранее слой доступа к базе данным согласно своего варианта, спроектировать и создать интерфейс Web-приложения на основе ASP.NET Core MVC Framework и Entity Framework Core.

Web- приложение должно удовлетворять следующим требованиям:

1. Осуществлять ввод, редактирование, добавление и просмотр данных не менее чем из трех таблиц реляционной базы согласно варианту. Не менее, чем одна из таблиц должна находиться на стороне отношения «многие» в схеме базы данных.
2. Иметь единое стилевое оформление, основанное на использовании мастер-страниц.
3. Иметь удобную систему навигации (строка меню, гиперссылки, кнопки), которая обеспечивает оптимальный путь перехода между двумя произвольно выбранными страницами в соответствии с логикой приложения.
4. Пользователь для работы с приложением должен пройти аутентификацию.
5. Должно поддерживать реализацию не менее двух ролевых политик.
6. Администратор должен иметь возможность управлять пользователями: просмативать, создавать, удалять и редактировать данные учетных записей.
7. Представления для просмотра данных из таблиц должны предусматривать разбиение данных на страницы, фильтрацию по одному или нескольким полям.

***На оценку 9 или 10:***

1. *Осуществить кэширования данных для отображения с помощью встроенного инструмента кэширования - объекта ImemoryCache. Выводить кэшированные данные таблиц MemoryCache на соответствующие страницы на сайта, генрируемые с использованием представлений (Views). Данные в кэше хранить неизменными до проведения операций вставки, изменения или удаления данных. После проведения этих операций кэш должен формироваться заново.*
2. *Реализовать сохранение состояния (значений) элементов представлений, предназначенных для осуществления фильтрации, с использованием куки и (или) с объекта Session. Осуществить заполнение элементов представлений, предназначенных для осуществления фильтрации, при их загрузке данными, ранее сохранненными в объекте куки и (или) Session.*

**Ход работы**

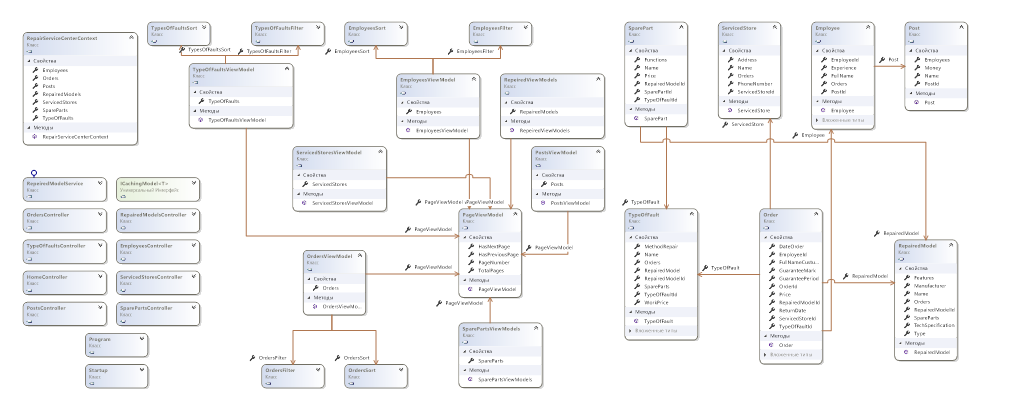


Рисунок 1 – Диаграмма базы данных

Исходный код классов представлен в приложении А.

**Результат выполнения:**

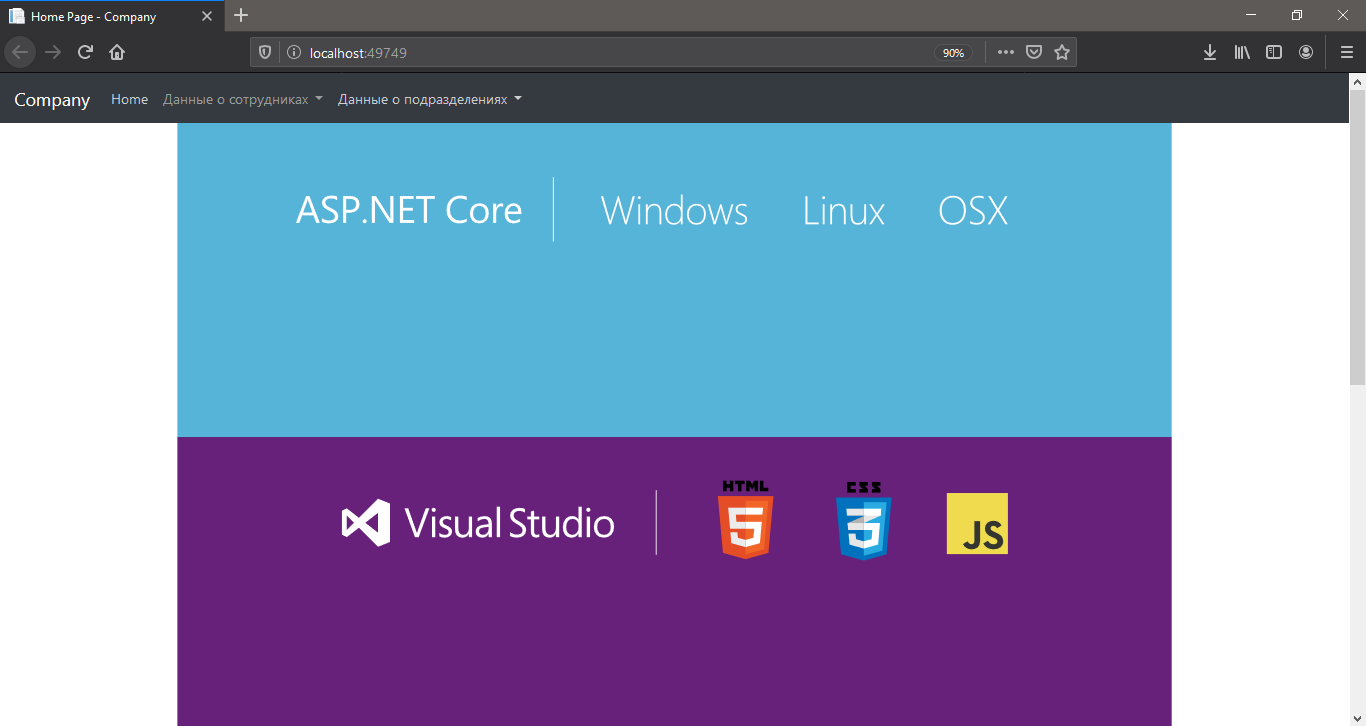


Рисунок 2 – Главная страница сайта

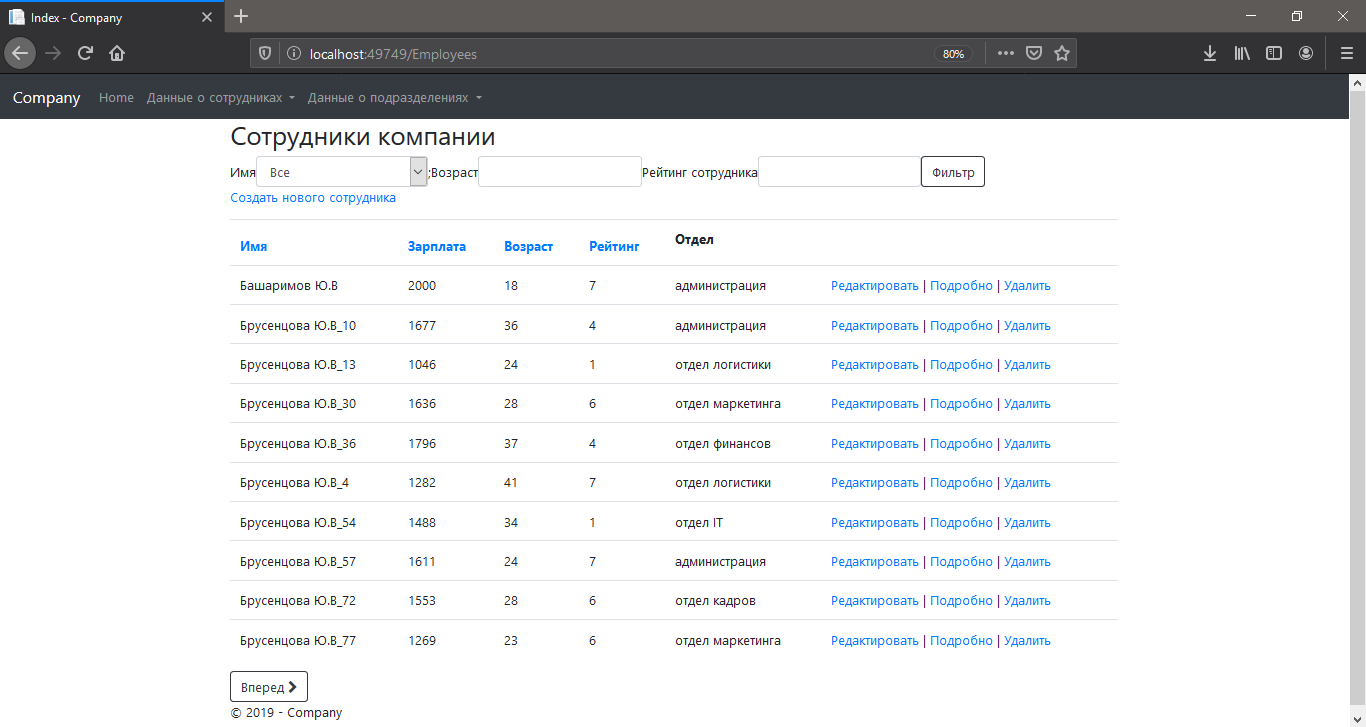


Рисунок 3 – Выборка из 20 записей таблицы Employees

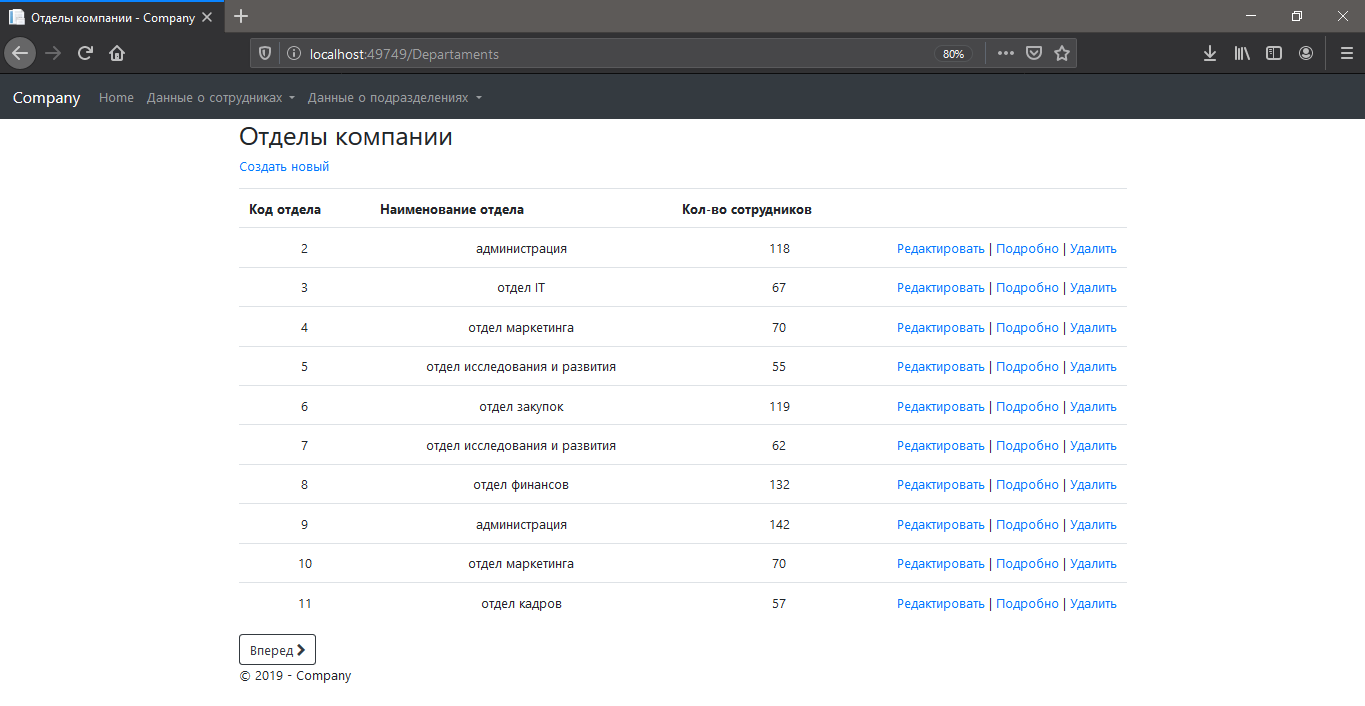


Рисунок 4 – Выборка из таблицы Units

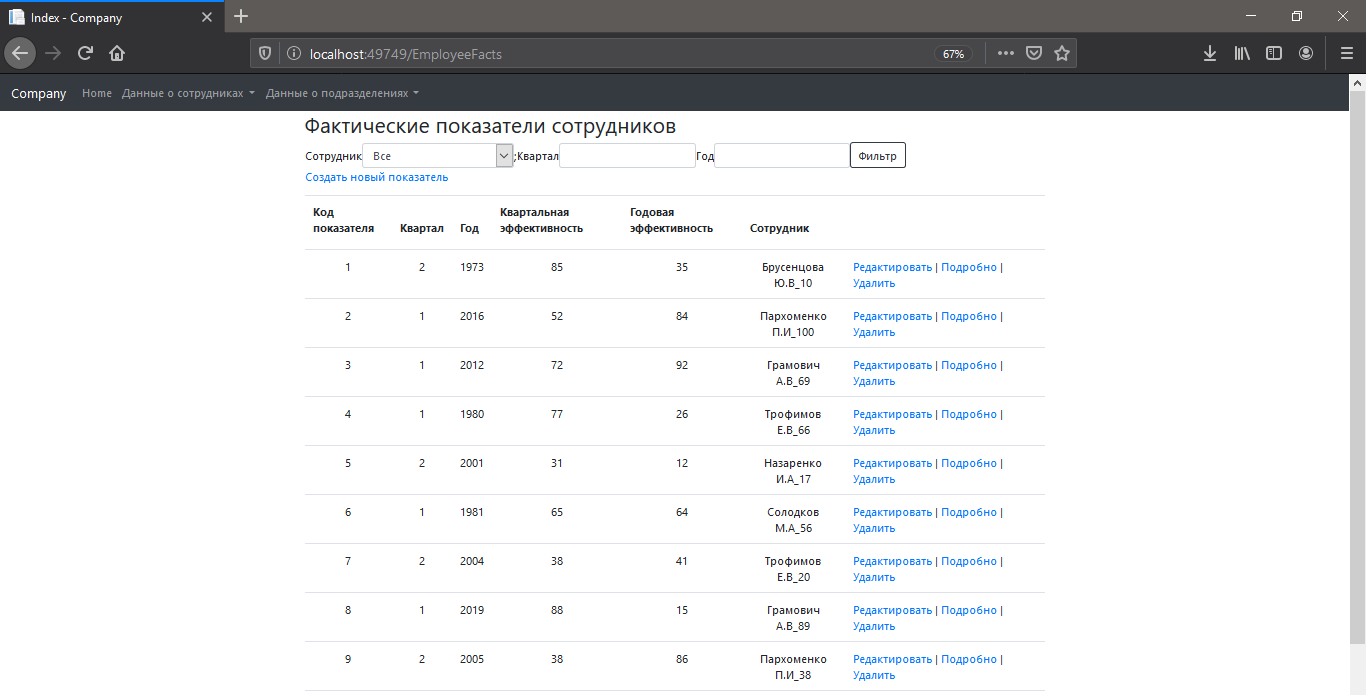
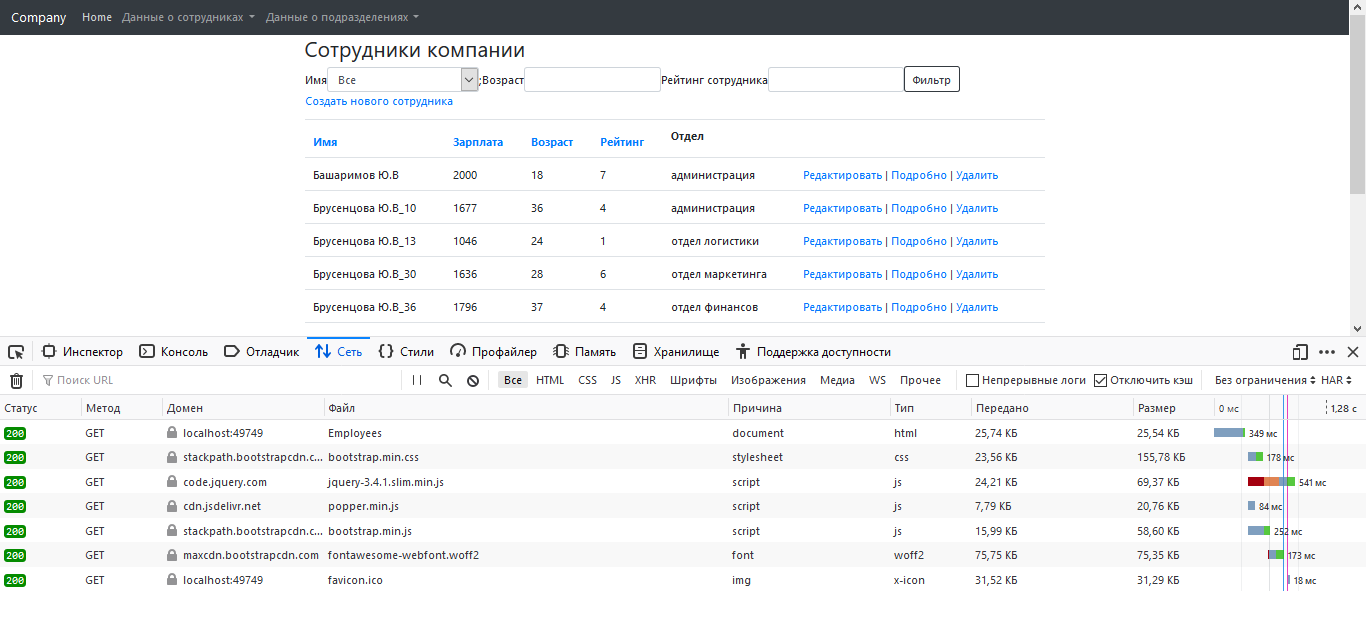


Рисунок 5 – Выборка из таблицы FactEmployee



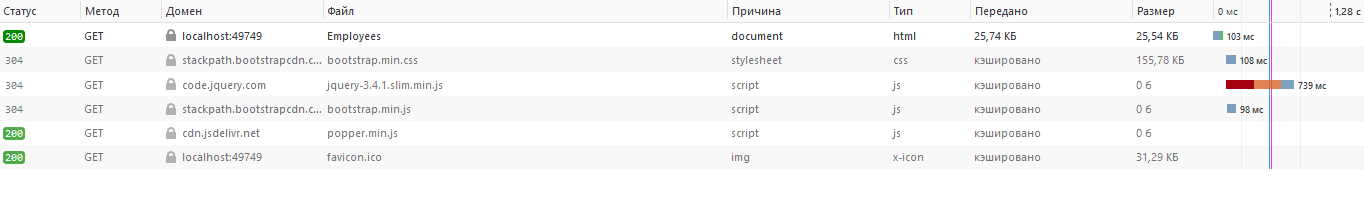


Рисунок 6 – Демонстрировать ускоренной обработки запроса при наличии кэширования с использованием ResponseCache

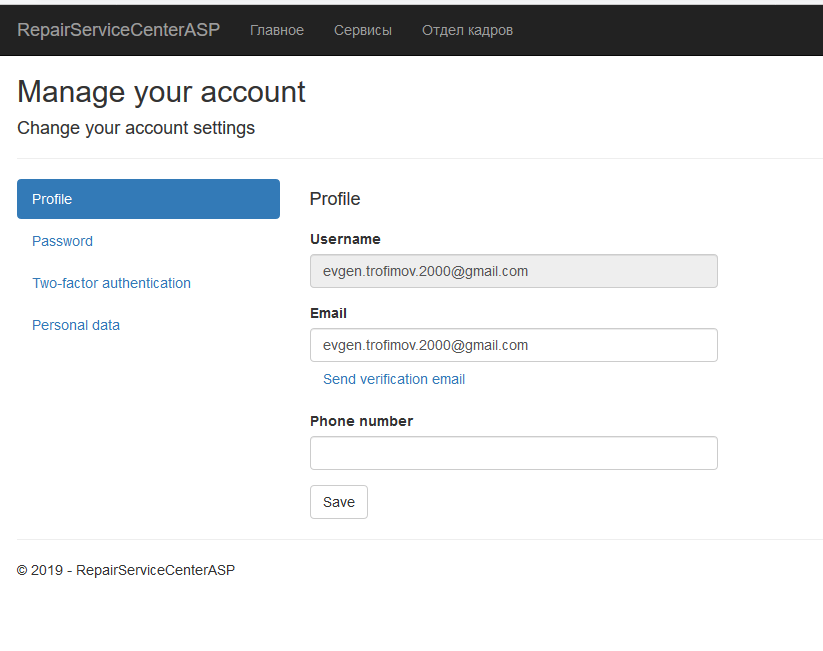


Рисунок 7 – Страница для управление аккаунтом

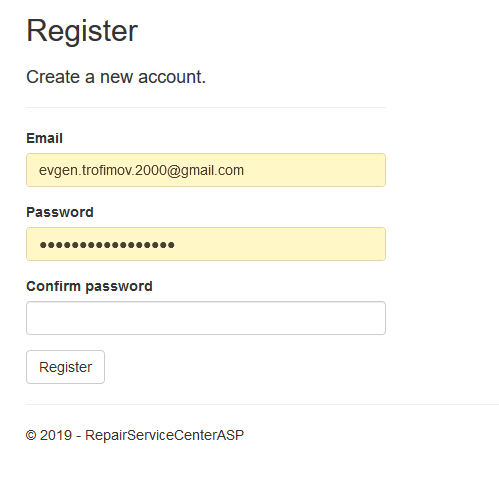


Рисунок 8 – Страница регистрации

**Вывод:** научились использовать ASP.NET MVC Сore для создания типовых web-приложений для работы с информацией из реляционных баз данных.

**ПРИЛОЖЕНИЕ А**

**Код программы**

**Листинг программы 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.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.SqlServer;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using RepairServiceCenterASP.Data;

using RepairServiceCenterASP.Middleware;

using RepairServiceCenterASP.Models;

using RepairServiceCenterASP.Services;

namespace RepairServiceCenterASP

{

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 connection = Configuration.GetConnectionString("DefaultConnection");

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

services.AddMemoryCache();

services.AddSession();

services.AddTransient<ICachingModel<RepairedModel>, RepeiredModelService>();

services.AddMvc();

string connectionUsers = Configuration.GetConnectionString("AccountContextConnection");

services.AddDbContext<AccountContext>(options => options.UseSqlServer(connectionUsers));

services.AddIdentity<UserAccount, IdentityRole>()

.AddDefaultTokenProviders()

.AddDefaultUI()

.AddEntityFrameworkStores<AccountContext>();

}

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

public void Configure(IApplicationBuilder app, IHostingEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

//app.UseBrowserLink();

}

else

{

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

}

app.UseStaticFiles();

app.UseCookiePolicy();

app.UseSession();

app.UseAuthentication();

app.UseDbInitializer();

app.UseMvc(routes =>

{

routes.MapRoute(

name: "default",

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

});

}

}

}

**Листинг программы EmployeesController.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using CompanyASP.Data;

using CompanyASP.Models;

using CompanyASP.ViewModel;

using CompanyASP.ViewModel.SortedEmployee;

namespace CompanyASP.Controllers

{

public class EmployeesController : Controller

{

private readonly CompanyContext \_context;

private int pageSize = 10;

public EmployeesController(CompanyContext context)

{

\_context = context;

}

// GET: Employees

[ResponseCache(Location = ResponseCacheLocation.Any, Duration = 270)]

public async Task<IActionResult> Index(int? fullName, int? age, double? salary, double? raiting,

int page = 1, SortStateEmployee employeeSort = SortStateEmployee.FullNameAsc)

{

//фильтрация

IQueryable<Employee> companyContext = \_context.Employees

.Include(d => d.Departament);

if (fullName != null && fullName != 0)

{

companyContext = companyContext.Where(d => d.Id == fullName);

}

if (age != null && age.Value != 0)

{

companyContext = companyContext.Where(d => d.Age == age);

}

if (salary != null && salary.Value != 0)

{

companyContext = companyContext.Where(d => d.Salary == salary);

}

if (raiting != null && raiting.Value != 0)

{

companyContext = companyContext.Where(d => d.Raiting == raiting);

}

//сортировка

switch (employeeSort)

{

case SortStateEmployee.AgeAsc:

companyContext = companyContext.OrderBy(d => d.Age);

break;

case SortStateEmployee.AgeDesc:

companyContext = companyContext.OrderByDescending(d => d.Age);

break;

case SortStateEmployee.FullNameAsc:

companyContext = companyContext.OrderBy(d => d.FullName);

break;

case SortStateEmployee.FullNameDesc:

companyContext = companyContext.OrderByDescending(d => d.FullName);

break;

case SortStateEmployee.SalaryAsc:

companyContext = companyContext.OrderBy(d => d.Salary);

break;

case SortStateEmployee.SalaryDesc:

companyContext = companyContext.OrderByDescending(d => d.Salary);

break;

case SortStateEmployee.RaitingAsc:

companyContext = companyContext.OrderBy(d => d.Raiting);

break;

case SortStateEmployee.RaitingDesc:

companyContext = companyContext.OrderByDescending(d => d.Raiting);

break;

case SortStateEmployee.DepartamentAsc:

companyContext = companyContext.OrderBy(d => d.Departament.FullName);

break;

case SortStateEmployee.DepartamentDesc:

companyContext = companyContext.OrderByDescending(d => d.Departament.FullName);

break;

default:

companyContext = companyContext.OrderBy(d => d.FullName);

break;

}

//разбиение на страницы

var count = await companyContext.CountAsync();

var companys = await companyContext.Skip((page - 1) \* pageSize).Take(pageSize).ToListAsync();

//формирование представления

IndexEmployeeViewModel employeesView = new IndexEmployeeViewModel

{

Employees = companys,

PageViewModel = new PageViewModel(count, page, pageSize),

SortEmployeeViewModel = new SortEmployeeViewModel(employeeSort),

FilterEmployeeViewModel = new FilterEmployeeViewModel(companyContext.ToList(), fullName, age, salary, raiting)

};

return View(employeesView);

}

// GET: Employees/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return NotFound();

}

var employee = await \_context.Employees

.Include(e => e.Departament)

.FirstOrDefaultAsync(m => m.Id == id);

if (employee == null)

{

return NotFound();

}

return View(employee);

}

// GET: Employees/Create

public IActionResult Create()

{

ViewData["DepartamentId"] = new SelectList(\_context.Departaments, "Id", "FullName");

return View();

}

// POST: Employees/Create

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,FullName,Salary,Age,Raiting,DepartamentId")] Employee employee)

{

if (ModelState.IsValid)

{

\_context.Add(employee);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

ViewData["DepartamentId"] = new SelectList(\_context.Departaments, "Id", "FullName", employee.DepartamentId);

return View(employee);

}

// GET: Employees/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return NotFound();

}

var employee = await \_context.Employees.FindAsync(id);

if (employee == null)

{

return NotFound();

}

ViewData["DepartamentId"] = new SelectList(\_context.Departaments, "Id", "FullName", employee.DepartamentId);

return View(employee);

}

// POST: Employees/Edit/5

// To protect from overposting attacks, please enable the specific properties you want to bind to, for

// more details see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,FullName,Salary,Age,Raiting,DepartamentId")] Employee employee)

{

if (id != employee.Id)

{

return NotFound();

}

if (ModelState.IsValid)

{

try

{

\_context.Update(employee);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!EmployeeExists(employee.Id))

{

return NotFound();

}

else

{

throw;

}

}

return RedirectToAction(nameof(Index));

}

ViewData["DepartamentId"] = new SelectList(\_context.Departaments, "Id", "FullName", employee.DepartamentId);

return View(employee);

}

// GET: Employees/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return NotFound();

}

var employee = await \_context.Employees

.Include(e => e.Departament)

.FirstOrDefaultAsync(m => m.Id == id);

if (employee == null)

{

return NotFound();

}

return View(employee);

}

// POST: Employees/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var employee = await \_context.Employees.FindAsync(id);

\_context.Employees.Remove(employee);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool EmployeeExists(int id)

{

return \_context.Employees.Any(e => e.Id == id);

}

}

}

**Листинг программы EmployeesViewModel.cs:**

using CompanyASP.Models;

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

using System.Linq;

using System.Threading.Tasks;

namespace CompanyASP.ViewModel

{

public class EmployeeViewModel

{

public IEnumerable<Employee> Employees { get; set; } //свойство для фильтрации

[Display(Name = "Код сотрудника")]

public int Id { get; set; }

[Display(Name = "ФИО")]

public string FullName { get; set; }

[Display(Name = "Зарплата")]

public double Salary { get; set; }

[Display(Name = "Возраст")]

public int Age { get; set; }

[Display(Name ="Отдел")]

public int DepartamentId { get; set; }

[Display(Name ="Рейтинг")]

public int Raiting { get; set; }

public Departament Departament { get; set; }

}

}

**Листинг программы SortEmployeeViewModel.cs:**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace CompanyASP.ViewModel.SortedEmployee

{

public enum SortStateEmployee

{

FullNameAsc,

FullNameDesc,

SalaryAsc,

SalaryDesc,

AgeAsc,

AgeDesc,

RaitingAsc,

RaitingDesc,

DepartamentAsc,

DepartamentDesc

}

public class SortEmployeeViewModel

{

public SortStateEmployee FullNameSort { get; set; }

public SortStateEmployee SalarySort { get; set; }

public SortStateEmployee AgeSort { get; set; }

public SortStateEmployee RaitingSort { get; set; }

public SortStateEmployee DepartamentSort { get; set; }

public SortStateEmployee CurrentState { get; set; }

public SortEmployeeViewModel(SortStateEmployee SortStateEmployee)

{

FullNameSort = SortStateEmployee == SortStateEmployee.FullNameAsc ? SortStateEmployee.FullNameDesc : SortStateEmployee.FullNameAsc;

SalarySort = SortStateEmployee == SortStateEmployee.SalaryAsc ? SortStateEmployee.SalaryDesc : SortStateEmployee.SalaryAsc;

AgeSort = SortStateEmployee == SortStateEmployee.AgeAsc ? SortStateEmployee.AgeDesc : SortStateEmployee.AgeAsc;

RaitingSort = SortStateEmployee == SortStateEmployee.RaitingAsc ? SortStateEmployee.RaitingDesc : SortStateEmployee.RaitingAsc;

DepartamentSort = SortStateEmployee == SortStateEmployee.DepartamentAsc ? SortStateEmployee.DepartamentDesc : SortStateEmployee.DepartamentAsc;

CurrentState = SortStateEmployee;

}

}

}

**Листинг программы FilterEmployeeViewModel.cs:**

using CompanyASP.Models;

using Microsoft.AspNetCore.Mvc.Rendering;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace CompanyASP.ViewModel.SortedEmployee

{

public class FilterEmployeeViewModel

{

public SelectList Employees { get; private set; }

public int? SelectFullName { get; private set; }

public int? SelectAge { get; private set; }

public double? SelectSalary { get; private set; }

public double? SelectRaiting { get; private set; }

public FilterEmployeeViewModel(List<Employee> employees, int? fullName, int? age, double? salary,

double? raiting)

{

employees.Insert(0, new Employee { FullName = "Все", Id = 0 });

Employees = new SelectList(employees, "Id", "FullName", fullName);

SelectFullName = fullName;

SelectAge = age;

SelectSalary = salary;

SelectRaiting = raiting;

}

}

}

**Листинг программы IndexEmployeeViewModel.cs:**

using CompanyASP.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace CompanyASP.ViewModel.SortedEmployee

{

public class IndexEmployeeViewModel

{

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

public PageViewModel PageViewModel { get; set; }

public SortEmployeeViewModel SortEmployeeViewModel { get; set; }

public FilterEmployeeViewModel FilterEmployeeViewModel { get; set; }

}

}