**9МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные системы и технологии

(в проектировании и производстве)»

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

 «Разработка интерфейса приложения баз данных с использованием с использованием аутентификации и авторизации»

Выполнила: студент гр. ИТП-21

Шеляхин М. С.

         Принял: преподаватель

Асенчик О. Д.

Гомель 2020

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

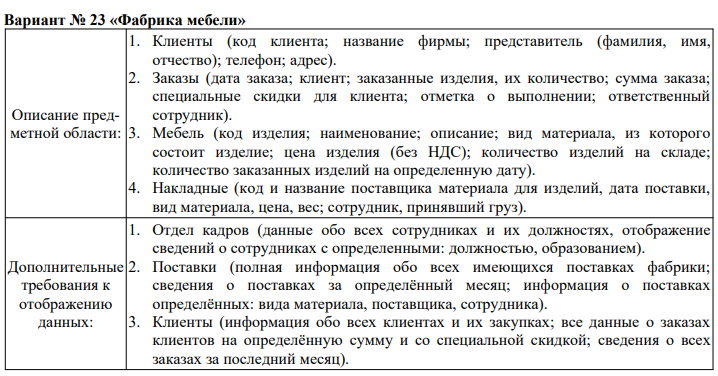
**

Рисунок 1 – вариант задания

**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.*

Листинг кодов классов *startup*, классов моделей, классов *View Model* и контекстов данных:

*Startup.cs*

using CourseProject.MiddleWares;

using CourseProject.Models;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace CourseProject

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

// Добавление профилей кэширования для таблиц и остальных страниц

services.AddControllersWithViews(options =>

{

options.CacheProfiles.Add("TablesCaching",

new CacheProfile()

{

Duration = 286

});

options.CacheProfiles.Add("NoCaching",

new CacheProfile()

{

Location = ResponseCacheLocation.None,

NoStore = true

});

});

// Добавление контекста данных со строкой подключения, хранящейся в файле appsettings.json

services.AddDbContext<ApplicationContext>(options => options.UseSqlServer(Configuration.GetConnectionString("SqlServerConnection")));

services.AddDbContext<IdentityContext>(options => options.UseSqlServer(Configuration.GetConnectionString("SqlServerConnection")));

services.AddIdentity<User, IdentityRole>(options => { options.User.RequireUniqueEmail = true; }).AddEntityFrameworkStores<IdentityContext>();

services.AddMemoryCache();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseInitializeMiddleware();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

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

});

}

}

}

*Client.cs*

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

// Модель, представляющая запись в таблице Клиенты

public class Client

{

[Key]

public int Id { get; set; }

public string Name { get; set; }

public string RepresentativeFIO { get; set; }

public int Number { get; set; }

public string Address { get; set; }

public Client() { }

}

}

*Employee.cs*

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

// Модель, представляющая запись в таблице Работники

public class Employee

{

[Key]

public int Id { get; set; }

public string FIO { get; set; }

public string Position { get; set; }

public string Education { get; set; }

public Employee() { }

}

}

*Furniture.cs*

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

// Модель, представляющая запись в таблице Мебель

public class Furniture

{

[Key]

public int Id { get; set; }

public string Name { get; set; }

public string Description { get; set; }

public string Material { get; set; }

public decimal Price { get; set; }

public int Count { get; set; }

public Furniture() { }

}

}

*Order.cs*

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

// Модель, представляющая запись в таблице Заказы

public class Order

{

[Key]

public int Id { get; set; }

public int ClientId { get; set; }

public int FurnitureId { get; set; }

public int FurnitureCount { get; set; }

public decimal Price { get; set; }

public int DiscountPercent { get; set; }

public int IsCompleted { get; set; }

public int EmployeeId { get; set; }

public Order() { }

}

}

*Waybill.cs*

using System;

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

// Модель, представляющая запись в таблице Накладные

public class Waybill

{

[Key]

public int Id { get; set; }

public int ProviderId { get; set; }

public string ProviderName { get; set; }

public DateTime DateOfSupply { get; set; }

public string Material { get; set; }

public decimal Price { get; set; }

public double Weight { get; set; }

public int FurnitureId { get; set; }

public int EmployeeId { get; set; }

public Waybill() { }

}

}

*CreateUserViewModel.cs*

namespace CourseProject.Models

{

public class CreateUserViewModel

{

public string Email { get; set; }

public string Password { get; set; }

public string FIO { get; set; }

public int Age { get; set; }

}

}

*EditUserViewModel.cs*

namespace CourseProject.Models

{

public class EditUserViewModel

{

public string Id { get; set; }

public string Email { get; set; }

public int Age { get; set; }

public string FIO { get; set; }

}

}

*IdentityContext.cs*

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace CourseProject.Models

{

public class IdentityContext : IdentityDbContext<User>

{

public IdentityContext(DbContextOptions<IdentityContext> options) : base(options)

{

Database.EnsureCreated();

}

}

}

*LoginViewModel.cs*

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

public class LoginViewModel

{

[Required]

[Display(Name = "Email")]

public string Email { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[Display(Name = "Запомнить?")]

public bool RememberMe { get; set; }

public string ReturnUrl { get; set; }

}

}

*RegisterViewModel.cs*

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

public class RegisterViewModel

{

[Required]

[Display(Name = "Роль пользователя")]

public string UserRole { get; set; }

[Required]

[DataType(DataType.EmailAddress)]

[Display(Name = "Email")]

public string Email { get; set; }

[Required]

[Display(Name = "ФИО пользователя")]

public string FIO { get; set; }

[Required]

[Display(Name = "Возраст")]

public int Age { get; set; }

[Required]

[DataType(DataType.Password)]

[Display(Name = "Пароль")]

public string Password { get; set; }

[Required]

[Compare("Password", ErrorMessage = "Пароли не совпадают")]

[DataType(DataType.Password)]

[Display(Name = "Подтвердить пароль")]

public string PasswordConfirm { get; set; }

}

}

*User.cs*

using Microsoft.AspNetCore.Identity;

namespace CourseProject.Models

{

public class User : IdentityUser

{

public string FIO { get; set; }

public int Age { get; set; }

}

}

*ClientIndexViewModel.cs*

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

public class ClientIndexViewModel

{

public List<Client> Clients { get; set; }

public List<int> Ids { get; set; }

public PageViewModel PageViewModel { get; set; }

public List<string> FilterRepresFIOs { get; set; }

public string FilterRepresFIO { get; set; }

public string FilterAddress { get; set; }

[Display(Name = "Id")]

public int Id { get; set; }

[Display(Name = "Название")]

public string Name { get; set; }

[Display(Name = "ФИО представителя")]

public string RepresentativeFIO { get; set; }

[Display(Name = "Номер")]

public int Number { get; set; }

[Display(Name = "Адрес")]

public string Address { get; set; }

}

}

*EmployeeIndexViewModel.cs*

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

public class EmployeeIndexViewModel

{

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

public PageViewModel PageViewModel { get; set; }

public List<int> Ids { get; set; }

public List<string> FilterPositions { get; set; }

public string FilterPosition { get; set; }

public string FilterEducation { get; set; }

[Display(Name = "Id")]

public int Id { get; set; }

[Display(Name = "ФИО")]

public string FIO { get; set; }

[Display(Name = "Должность")]

public string Position { get; set; }

[Display(Name = "Образование")]

public string Education { get; set; }

}

}

*FurnitureIndexViewModel.cs*

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

public class FurnitureIndexViewModel

{

public List<Furniture> Furniture { get; set; }

public PageViewModel PageViewModel { get; set; }

public List<string> FilterMaterials { get; set; }

public string FilterMaterial { get; set; }

public decimal FilterPrice { get; set; }

public List<int> Ids { get; set; }

[Display(Name = "Id")]

public int Id { get; set; }

[Display(Name = "Название")]

public string Name { get; set; }

[Display(Name = "Описание")]

public string Description { get; set; }

[Display(Name = "Материал")]

public string Material { get; set; }

[Display(Name = "Цена")]

public decimal Price { get; set; }

[Display(Name = "Количество")]

public int Count { get; set; }

}

}

*OrderViewModel.cs*

namespace CourseProject.Models

{

// Модель отображения записей из таблицы Заказы с заменой внешних ключей смысловыми данными

public class OrderViewModel

{

public int Id { get; set; }

public string ClientName { get; set; }

public string FurnitureName { get; set; }

public int FurnitureCount { get; set; }

public decimal Price { get; set; }

public int DiscountPercent { get; set; }

public bool IsCompleted { get; set; }

public string EmployeeFIO { get; set; }

}

}

*OrderIndexViewModel.cs*

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

public class OrderIndexViewModel

{

public List<OrderViewModel> OrderViewModels { get; set; }

public List<string> EmployeesFIOs { get; set; }

public List<string> ClientNames { get; set; }

public List<string> FurnitureNames { get; set; }

public PageViewModel PageViewModel { get; set; }

public List<string> FilterClients { get; set; }

public string FilterClient { get; set; }

public List<string> FilterFurnitures { get; set; }

public string FilterFurniture { get; set; }

public List<int> Ids { get; set; }

[Display(Name = "Id")]

public int Id { get; set; }

[Display(Name = "Название клиента")]

public string ClientName { get; set; }

[Display(Name = "Название мебели")]

public string FurnitureName { get; set; }

[Display(Name = "Количество мебели")]

public int FurnitureCount { get; set; }

[Display(Name = "Стоимость")]

public decimal Price { get; set; }

[Display(Name = "Процент скидки")]

public int DiscountPercent { get; set; }

[Display(Name = "Выполнено?")]

public bool IsCompleted { get; set; }

[Display(Name = "ФИО работника")]

public string EmployeeFIO { get; set; }

}

}

*WaybillViewModel.cs*

using System;

namespace CourseProject.Models

{

// Модель отображения записей из таблицы Накладные с заменой внешних ключей смысловыми данными

public class WaybillViewModel

{

public int Id { get; set; }

public int ProviderId { get; set; }

public string ProviderName { get; set; }

public DateTime DateOfSupply { get; set; }

public string Material { get; set; }

public decimal Price { get; set; }

public double Weight { get; set; }

public string FurnitureName { get; set; }

public string EmployeeFIO { get; set; }

}

}

*WaybillIndexViewModel.cs*

using System;

using System.Collections.Generic;

using System.ComponentModel.DataAnnotations;

namespace CourseProject.Models

{

public class WaybillIndexViewModel

{

public List<WaybillViewModel> WaybillViewModels { get; set; }

public List<string> EmployeesFIOs { get; set; }

public List<string> FurnitureNames { get; set; }

public PageViewModel PageViewModel { get; set; }

public List<string> FilterFurnitureNames { get; set; }

public string FilterFurnitName { get; set; }

public string FilterProviderName { get; set; }

public List<int> Ids { get; set; }

[Display(Name = "Id")]

public int Id { get; set; }

[Display(Name = "Номер поставщика")]

public int ProviderId { get; set; }

[Display(Name = "Название мебели")]

public string FurnitureName { get; set; }

[Display(Name = "Название поставщика")]

public string ProviderName { get; set; }

[Display(Name = "Стоимость")]

public decimal Price { get; set; }

[Display(Name = "Дата поставок")]

public DateTime DateOfSupply { get; set; }

[Display(Name = "Материал")]

public string Material { get; set; }

[Display(Name = "ФИО работника")]

public string EmployeeFIO { get; set; }

[Display(Name = "Вес")]

public double Weight { get; set; }

}

}

*ApplicationContext.cs*

using Microsoft.EntityFrameworkCore;

namespace CourseProject.Models

{

// Класс контекста данных

public class ApplicationContext : DbContext

{

public DbSet<Client> Clients { get; set; }

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

public DbSet<Furniture> Furniture { get; set; }

public DbSet<Order> Orders { get; set; }

public DbSet<Waybill> Waybills { get; set; }

public ApplicationContext(DbContextOptions<ApplicationContext> options) : base(options)

{

}

}

}

*PageViewModel.cs*

using System;

namespace CourseProject.Models

{

public class PageViewModel

{

public int PageNumber { get; private set; }

public int TotalPages { get; private set; }

public PageViewModel(int count, int pageNumber, int pageSize)

{

PageNumber = pageNumber;

TotalPages = (int)Math.Ceiling(count / (double)pageSize);

}

public bool HasPreviousPage

{

get

{

return (PageNumber > 1);

}

}

public bool HasNextPage

{

get

{

return (PageNumber < TotalPages);

}

}

}

}

Листинг исходного кода контроллеров и представлений:

*HomeController.cs*

using System.Diagnostics;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using CourseProject.Models;

namespace CourseProject.Controllers

{

// Контроллер начальной страницы

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

// Метод получения начальной страницы.

// Начальная страница не кэшируется.

[ResponseCache(CacheProfileName = "NoCaching")]

public IActionResult Index()

{

return View();

}

// Метод получения страницы ошибки.

// Страница ошибки не кэшируется.

[ResponseCache(CacheProfileName = "NoCaching")]

public IActionResult Error()

{

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

}

}

}

*AccountController.cs*

using CourseProject.Models;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using System.Threading.Tasks;

namespace CourseProject.Controllers

{

public class AccountController : Controller

{

private readonly UserManager<User> \_userManager;

private readonly SignInManager<User> \_signInManager;

private readonly RoleManager<IdentityRole> \_roleManager;

public AccountController(UserManager<User> userManager, SignInManager<User> signInManager, RoleManager<IdentityRole> roleManager)

{

\_userManager = userManager;

\_signInManager = signInManager;

\_roleManager = roleManager;

}

[HttpGet]

public IActionResult Register()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Register(RegisterViewModel model)

{

if (ModelState.IsValid)

{

User user = new User { Email = model.Email, UserName = model.Email, FIO = model.FIO, Age = model.Age };

// добавляем пользователя

var result = await \_userManager.CreateAsync(user, model.Password);

if (!await \_roleManager.RoleExistsAsync(model.UserRole))

{

await \_roleManager.CreateAsync(new IdentityRole(model.UserRole));

}

await \_userManager.AddToRoleAsync(user, model.UserRole);

if (result.Succeeded)

{

// установка куки

await \_signInManager.SignInAsync(user, false);

return RedirectToAction("Index", "Home");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

[HttpGet]

public IActionResult Login(string returnUrl = null)

{

return View(new LoginViewModel { ReturnUrl = returnUrl });

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Login(LoginViewModel model)

{

if (ModelState.IsValid)

{

var result = await \_signInManager.PasswordSignInAsync(model.Email, model.Password, model.RememberMe, false);

if (result.Succeeded)

{

// проверяем, принадлежит ли URL приложению

if (!string.IsNullOrEmpty(model.ReturnUrl) && Url.IsLocalUrl(model.ReturnUrl))

{

return Redirect(model.ReturnUrl);

}

else

{

return RedirectToAction("Index", "Home");

}

}

else

{

ModelState.AddModelError("", "Неправильный логин и (или) пароль");

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout()

{

// удаляем аутентификационные куки

await \_signInManager.SignOutAsync();

return RedirectToAction("Index", "Home");

}

}

}

*ClientController.cs*

using CourseProject.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

namespace CourseProject.Controllers

{

public class ClientController : Controller

{

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public ClientController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

// Метод получения страницы клиентов.

// Данная страница кэшируется на 286 секунд.

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string filterRepresFIO = "Все", string address = null)

{

int pageSize = 20;

List<Client> clients;

if (!cache.TryGetValue("Clients", out clients))

{

clients = db.Clients.ToList();

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

List<int> Ids = clients.Select(item => item.Id).ToList();

var repFios = clients.Select(item => item.RepresentativeFIO).ToList();

repFios.Add("Все");

if (filterRepresFIO != "Все")

{

clients = clients.Where(item => item.RepresentativeFIO == filterRepresFIO).ToList();

}

if (address != null)

{

clients = clients.Where(item => item.Address.Contains(address)).ToList();

}

ClientIndexViewModel clientIndexViewModel = new ClientIndexViewModel()

{

Clients = clients.Skip((page - 1) \* pageSize).Take(pageSize).ToList(),

Ids = Ids,

PageViewModel = new PageViewModel(clients.Count, page, pageSize),

FilterRepresFIOs = repFios

};

return View(clientIndexViewModel);

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult AddClient(ClientIndexViewModel model)

{

var names = db.Clients.Select(item => item.Name);

ViewData["Message"] = "";

model.Clients = db.Clients.ToList();

model.Ids = db.Employees.Select(item => item.Id).ToList();

if (model.Name == null || model.RepresentativeFIO == null || model.Address == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Client/Index.cshtml", model);

}

if (names.Contains(model.Name) || model.Name.Length == 0 || model.Name.Length > 25)

{

ViewData["Message"] += "Неправильный ввод названия";

return View("~/Views/Client/Index.cshtml", model);

}

else if (model.RepresentativeFIO.Length == 0 || model.RepresentativeFIO.Length > 100)

{

ViewData["Message"] += "Неправильный ввод ФИО представителя";

return View("~/Views/Client/Index.cshtml", model);

}

else if (model.Number.ToString().Length < 9 || model.Number.ToString().Length > 13)

{

ViewData["Message"] += "Неправильный ввод номера";

return View("~/Views/Client/Index.cshtml", model);

}

else if (model.Address.Length == 0 || model.Address.Length > 40)

{

ViewData["Message"] += "Неправильный ввод адреса";

return View("~/Views/Client/Index.cshtml", model);

}

else

{

var id = 0;

if (db.Clients.Count() != 0)

{

id = db.Clients.Select(item => item.Id).Max();

}

id++;

db.Clients.Add(new Client() { Id = id, Name = model.Name, Address = model.Address, Number = model.Number, RepresentativeFIO = model.RepresentativeFIO });

db.SaveChanges();

cache.Remove("Clients");

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Client");

}

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult DeleteClient(ClientIndexViewModel model)

{

ViewData["Message"] = "";

var client = db.Clients.Where(item => item.Id == model.Id).FirstOrDefault();

db.Clients.Remove(client);

db.SaveChanges();

cache.Remove("Clients");

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Client");

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult UpdateClient(ClientIndexViewModel model)

{

var names = db.Clients.Select(item => item.Name);

ViewData["Message"] = "";

model.Clients = db.Clients.ToList();

model.Ids = db.Clients.Select(item => item.Id).ToList();

if (model.Name == null || model.RepresentativeFIO == null || model.Address == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Client/Index.cshtml", model);

}

if (names.Contains(model.Name) || model.Name.Length == 0 || model.Name.Length > 25)

{

ViewData["Message"] += "Неправильный ввод названия";

return View("~/Views/Client/Index.cshtml", model);

}

else if (model.RepresentativeFIO.Length == 0 || model.RepresentativeFIO.Length > 100)

{

ViewData["Message"] += "Неправильный ввод ФИО представителя";

return View("~/Views/Client/Index.cshtml", model);

}

else if (model.Number.ToString().Length < 9 || model.Number.ToString().Length > 13)

{

ViewData["Message"] += "Неправильный ввод номера";

return View("~/Views/Client/Index.cshtml", model);

}

else if (model.Address.Length == 0 || model.Address.Length > 40)

{

ViewData["Message"] += "Неправильный ввод адреса";

return View("~/Views/Client/Index.cshtml", model);

}

else

{

var client = db.Clients.Where(item => item.Id == model.Id).FirstOrDefault();

client.Name = model.Name;

client.RepresentativeFIO = model.RepresentativeFIO;

client.Number = model.Number;

client.Address = model.Address;

db.SaveChanges();

cache.Remove("Clients");

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Client");

}

}

}

}

*EmployeeController.cs*

using CourseProject.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

namespace CourseProject.Controllers

{

public class EmployeeController : Controller

{

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public EmployeeController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

// Метод получения страницы работников.

// Данная страница кэшируется на 286 секунд.

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string position = "Все", string education = null)

{

int pageSize = 20;

List<Employee> employees;

if (!cache.TryGetValue("Employees", out employees))

{

employees = db.Employees.ToList();

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

List<int> Ids = employees.Select(item => item.Id).ToList();

List<string> positions = employees.Select(item => item.Position).ToList();

positions.Add("Все");

if (position != "Все")

{

employees = employees.Where(item => item.Position == position).ToList();

}

if (education != null)

{

employees = employees.Where(item => item.Education.Contains(education)).ToList();

}

EmployeeIndexViewModel employeeIndexViewModel = new EmployeeIndexViewModel()

{

Employees = employees.Skip((page - 1) \* pageSize).Take(pageSize).ToList(),

Ids = Ids,

PageViewModel = new PageViewModel(employees.Count, page, pageSize),

FilterPositions = positions

};

return View(employeeIndexViewModel);

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult AddEmployee(EmployeeIndexViewModel model)

{

var fios = db.Employees.Select(item => item.FIO);

ViewData["Message"] = "";

model.Employees = db.Employees.ToList();

model.Ids = db.Employees.Select(item => item.Id).ToList();

if (model.FIO == null || model.Education == null || model.Position == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Employee/Index.cshtml", model);

}

if (fios.Contains(model.FIO) || model.FIO.Length == 0 || model.FIO.Length > 100)

{

ViewData["Message"] += "Неправильный ввод ФИО";

return View("~/Views/Employee/Index.cshtml", model);

}

else if (model.Position.Length == 0 || model.Position.Length > 50)

{

ViewData["Message"] += "Неправильный ввод должности";

return View("~/Views/Employee/Index.cshtml", model);

}

else if (model.Education.Length < 9 || model.Education.Length > 200)

{

ViewData["Message"] += "Неправильный ввод образования";

return View("~/Views/Employee/Index.cshtml", model);

}

else

{

var id = 0;

if (db.Employees.Count() != 0)

{

id = db.Employees.Select(item => item.Id).Max();

}

id++;

db.Employees.Add(new Employee() { Id = id, FIO = model.FIO, Position = model.Position, Education = model.Education });

db.SaveChanges();

cache.Remove("Employees");

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Employee");

}

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult DeleteEmployee(EmployeeIndexViewModel model)

{

ViewData["Message"] = "";

var employee = db.Employees.Where(item => item.Id == model.Id).FirstOrDefault();

db.Employees.Remove(employee);

db.SaveChanges();

cache.Remove("Employees");

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Employee");

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult UpdateEmployee(EmployeeIndexViewModel model)

{

var fios = db.Employees.Select(item => item.FIO);

ViewData["Message"] = "";

model.Employees = db.Employees.ToList();

model.Ids = db.Employees.Select(item => item.Id).ToList();

if (model.FIO == null || model.Education == null || model.Position == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Employee/Index.cshtml", model);

}

if (fios.Contains(model.FIO) || model.FIO.Length == 0 || model.FIO.Length > 100)

{

ViewData["Message"] += "Неправильный ввод ФИО";

return View("~/Views/Employee/Index.cshtml", model);

}

else if (model.Position.Length == 0 || model.Position.Length > 50)

{

ViewData["Message"] += "Неправильный ввод должности";

return View("~/Views/Employee/Index.cshtml", model);

}

else if (model.Education.Length < 0 || model.Education.Length > 200)

{

ViewData["Message"] += "Неправильный ввод образования";

return View("~/Views/Employee/Index.cshtml", model);

}

else

{

var employee = db.Employees.Where(item => item.Id == model.Id).FirstOrDefault();

employee.FIO = model.FIO;

employee.Position = model.Position;

employee.Education = model.Education;

db.SaveChanges();

cache.Remove("Employees");

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Employee");

}

}

}

}

*FurnitureController.cs*

using CourseProject.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

namespace CourseProject.Controllers

{

public class FurnitureController : Controller

{

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public FurnitureController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

// Метод получения страницы клиентов.

// Данная страница кэшируется на 286 секунд.

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string material = "Все", decimal price = 0)

{

int pageSize = 20;

List<Furniture> furniture;

if (!cache.TryGetValue("Furniture", out furniture))

{

furniture = db.Furniture.ToList();

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

List<int> Ids = furniture.Select(item => item.Id).ToList();

List<string> materials = furniture.Select(item => item.Material).ToList();

materials.Add("Все");

if (material != "Все")

{

furniture = furniture.Where(item => item.Material == material).ToList();

}

if (price != 0)

{

furniture = furniture.Where(item => item.Price == price).ToList();

}

FurnitureIndexViewModel furnitureIndexViewModel = new FurnitureIndexViewModel()

{

Furniture = furniture.Skip((page - 1) \* pageSize).Take(pageSize).ToList(),

Ids = Ids,

PageViewModel = new PageViewModel(furniture.Count, page, pageSize),

FilterMaterials = materials

};

return View(furnitureIndexViewModel);

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult AddFurniture(FurnitureIndexViewModel model)

{

var names = db.Furniture.Select(item => item.Name);

ViewData["Message"] = "";

model.Furniture = db.Furniture.ToList();

model.Ids = db.Furniture.Select(item => item.Id).ToList();

if (model.Name == null || model.Description == null || model.Material == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Furniture/Index.cshtml", model);

}

if (names.Contains(model.Name) || model.Name.Length == 0 || model.Name.Length > 50)

{

ViewData["Message"] += "Неправильный ввод названия";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Description.Length == 0 || model.Description.Length > 200)

{

ViewData["Message"] += "Неправильный ввод описания";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Material.Length == 0 || model.Material.Length > 60)

{

ViewData["Message"] += "Неправильный ввод материала";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Price <= 0)

{

ViewData["Message"] += "Неправильное значение стоимости";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Count <= 0)

{

ViewData["Message"] += "Неправильное значение количества";

return View("~/Views/Furniture/Index.cshtml", model);

}

else

{

var id = 0;

if (db.Furniture.Count() != 0)

{

id = db.Furniture.Select(item => item.Id).Max();

}

id++;

db.Furniture.Add(new Furniture() { Id = id, Name = model.Name, Description = model.Description, Material = model.Material, Price = model.Price, Count = model.Count });

db.SaveChanges();

cache.Remove("Furniture");

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Furniture");

}

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult DeleteFurniture(FurnitureIndexViewModel model)

{

ViewData["Message"] = "";

var furniture = db.Furniture.Where(item => item.Id == model.Id).FirstOrDefault();

db.Furniture.Remove(furniture);

db.SaveChanges();

cache.Remove("Furniture");

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Furniture");

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult UpdateFurniture(FurnitureIndexViewModel model)

{

var names = db.Furniture.Select(item => item.Name);

ViewData["Message"] = "";

model.Furniture = db.Furniture.ToList();

model.Ids = db.Furniture.Select(item => item.Id).ToList();

if (model.Name == null || model.Description == null || model.Material == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Furniture/Index.cshtml", model);

}

if (names.Contains(model.Name) || model.Name.Length == 0 || model.Name.Length > 50)

{

ViewData["Message"] += "Неправильный ввод названия";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Description.Length == 0 || model.Description.Length > 200)

{

ViewData["Message"] += "Неправильный ввод описания";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Material.Length == 0 || model.Material.Length > 60)

{

ViewData["Message"] += "Неправильный ввод материала";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Price <= 0)

{

ViewData["Message"] += "Неправильное значение стоимости";

return View("~/Views/Furniture/Index.cshtml", model);

}

else if (model.Count <= 0)

{

ViewData["Message"] += "Неправильное значение количества";

return View("~/Views/Furniture/Index.cshtml", model);

}

else

{

var furniture = db.Furniture.Where(item => item.Id == model.Id).FirstOrDefault();

furniture.Name = model.Name;

furniture.Description = model.Description;

furniture.Material = model.Material;

furniture.Price = model.Price;

furniture.Count = model.Count;

db.SaveChanges();

cache.Remove("Furniture");

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Furniture");

}

}

}

}

*OrderController.cs*

using CourseProject.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

namespace CourseProject.Controllers

{

public class OrderController : Controller

{

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public OrderController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

// Метод получения страницы клиентов.

// Данная страница кэшируется на 286 секунд.

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string furniture = "Все", string clientName = "Все")

{

return View(GetViewModel(page, furniture, clientName));

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult AddOrder(OrderIndexViewModel model)

{

ViewData["Message"] = "";

if (model.FurnitureCount <= 0)

{

ViewData["Message"] += "Неправильное значение количества мебели";

return View("~/Views/Order/Index.cshtml", GetViewModel());

}

else if (model.DiscountPercent < 0 || model.DiscountPercent > 99)

{

ViewData["Message"] += "Неправильное значение скидки";

return View("~/Views/Order/Index.cshtml", GetViewModel());

}

else if (model.Price <= 0)

{

ViewData["Message"] += "Неправильное значение стоимости";

return View("~/Views/Order/Index.cshtml", GetViewModel());

}

else

{

var id = 0;

if (db.Orders.Count() != 0)

{

id = db.Orders.Select(item => item.Id).Max();

}

id++;

db.Orders.Add(new Order()

{

Id = id,

ClientId = db.Clients.Where(item => item.Name == model.ClientName).First().Id,

DiscountPercent = model.DiscountPercent,

FurnitureCount = model.FurnitureCount,

Price = model.Price,

IsCompleted = model.IsCompleted ? 1 : 0,

EmployeeId = db.Employees.Where(item => item.FIO == model.EmployeeFIO).First().Id,

FurnitureId = db.Furniture.Where(item => item.Name == model.FurnitureName).First().Id

});

db.SaveChanges();

cache.Remove("Orders");

return RedirectToAction("Index", "Order");

}

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult DeleteOrder(OrderIndexViewModel model)

{

ViewData["Message"] = "";

var order = db.Orders.Where(item => item.Id == model.Id).FirstOrDefault();

db.Orders.Remove(order);

db.SaveChanges();

cache.Remove("Orders");

return RedirectToAction("Index", "Order");

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult UpdateOrder(OrderIndexViewModel model)

{

ViewData["Message"] = "";

if (model.FurnitureCount <= 0)

{

ViewData["Message"] += "Неправильное значение количества мебели";

return View("~/Views/Order/Index.cshtml", GetViewModel());

}

else if (model.DiscountPercent < 0 || model.DiscountPercent > 99)

{

ViewData["Message"] += "Неправильное значение скидки";

return View("~/Views/Order/Index.cshtml", GetViewModel());

}

else if (model.Price <= 0)

{

ViewData["Message"] += "Неправильное значение стоимости";

return View("~/Views/Order/Index.cshtml", GetViewModel());

}

else

{

var order = db.Orders.Where(item => item.Id == model.Id).FirstOrDefault();

order.DiscountPercent = model.DiscountPercent;

order.ClientId = db.Clients.Where(item => item.Name == model.ClientName).First().Id;

order.EmployeeId = db.Employees.Where(item => item.FIO == model.EmployeeFIO).First().Id;

order.Price = model.Price;

order.FurnitureCount = model.FurnitureCount;

order.FurnitureId = db.Furniture.Where(item => item.Name == model.FurnitureName).First().Id;

order.IsCompleted = model.IsCompleted ? 1 : 0;

db.SaveChanges();

cache.Remove("Orders");

return RedirectToAction("Index", "Order");

}

}

private OrderIndexViewModel GetViewModel(int page = 1, string furnitureName = "Все", string clientName = "Все")

{

int pageSize = 20;

List<Employee> employees = db.Employees.ToList();

List<Furniture> furniture = db.Furniture.ToList();

List<Client> clients = db.Clients.ToList();

List<OrderViewModel> orderViewModels;

if (!cache.TryGetValue("Orders", out orderViewModels))

{

List<Order> orders = db.Orders.ToList();

orderViewModels = new List<OrderViewModel>();

foreach (var order in orders)

{

orderViewModels.Add(new OrderViewModel()

{

Id = order.Id,

ClientName = clients.Where(item => item.Id == order.ClientId).First().Name,

DiscountPercent = order.DiscountPercent,

EmployeeFIO = employees.Where(item => item.Id == order.EmployeeId).First().FIO,

FurnitureCount = order.FurnitureCount,

FurnitureName = furniture.Where(item => item.Id == order.FurnitureId).First().Name,

IsCompleted = order.IsCompleted != 0,

Price = order.Price

});

}

cache.Set("Orders", db.Orders.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

List<int> Ids = orderViewModels.Select(item => item.Id).ToList();

List<string> furnituresNames = furniture.Select(item => item.Name).ToList();

List<string> clientNames = clients.Select(item => item.Name).ToList();

furnituresNames.Add("Все");

clientNames.Add("Все");

if (furnitureName != "Все")

{

orderViewModels = orderViewModels.Where(item => item.FurnitureName == furnitureName).ToList();

}

if (clientName != "Все")

{

orderViewModels = orderViewModels.Where(item => item.ClientName == clientName).ToList();

}

OrderIndexViewModel orderIndexViewModel = new OrderIndexViewModel()

{

OrderViewModels = orderViewModels.Skip((page - 1) \* pageSize).Take(pageSize).ToList(),

Ids = Ids,

ClientNames = clients.Select(item => item.Name).ToList(),

EmployeesFIOs = employees.Select(item => item.FIO).ToList(),

FurnitureNames = furniture.Select(item => item.Name).ToList(),

PageViewModel = new PageViewModel(orderViewModels.Count, page, pageSize),

FilterClients = clientNames,

FilterFurnitures = furnituresNames

};

return orderIndexViewModel;

}

}

}

*WaybillController.cs*

using CourseProject.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Caching.Memory;

using System;

using System.Collections.Generic;

using System.Linq;

namespace CourseProject.Controllers

{

public class WaybillController : Controller

{

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public WaybillController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

// Метод получения страницы клиентов.

// Данная страница кэшируется на 286 секунд.

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string providerName = null, string furnitName = "Все")

{

return View(GetViewModel(page, providerName, furnitName));

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult AddWaybill(WaybillIndexViewModel model)

{

ViewData["Message"] = "";

if (model.ProviderName == null || model.Material == null || model.DateOfSupply == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.ProviderId <= 0)

{

ViewData["Message"] += "Неправильное значение номера поставщика";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.Weight <= 0)

{

ViewData["Message"] += "Неправильное значение веса";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.Price <= 0)

{

ViewData["Message"] += "Неправильное значение стоимости";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.ProviderName.Length > 100)

{

ViewData["Message"] += "Неправильный ввод названия поставщика";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.Material.Length > 60)

{

ViewData["Message"] += "Неправильный ввод материала";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else

{

var id = 0;

if (db.Waybills.Count() != 0)

{

id = db.Waybills.Select(item => item.Id).Max();

}

id++;

db.Waybills.Add(new Waybill()

{

Id = id,

ProviderId = model.ProviderId,

ProviderName = model.ProviderName,

Price = model.Price,

DateOfSupply = model.DateOfSupply,

EmployeeId = db.Employees.Where(item => item.FIO == model.EmployeeFIO).First().Id,

FurnitureId = db.Furniture.Where(item => item.Name == model.FurnitureName).First().Id,

Material = model.Material,

Weight = model.Weight

});

db.SaveChanges();

cache.Remove("Waybills");

return RedirectToAction("Index", "Waybill");

}

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult DeleteWaybill(WaybillIndexViewModel model)

{

ViewData["Message"] = "";

var waybill = db.Waybills.Where(item => item.Id == model.Id).FirstOrDefault();

db.Waybills.Remove(waybill);

db.SaveChanges();

cache.Remove("Waybills");

return RedirectToAction("Index", "Waybill");

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult UpdateWaybill(WaybillIndexViewModel model)

{

ViewData["Message"] = "";

if (model.ProviderName == null || model.Material == null || model.DateOfSupply == null)

{

ViewData["Message"] += "Отсутствие значений в строках";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.ProviderId <= 0)

{

ViewData["Message"] += "Неправильное значение номера поставщика";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.Weight <= 0)

{

ViewData["Message"] += "Неправильное значение веса";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.Price <= 0)

{

ViewData["Message"] += "Неправильное значение стоимости";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.ProviderName.Length > 100)

{

ViewData["Message"] += "Неправильный ввод названия поставщика";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else if (model.Material.Length > 60)

{

ViewData["Message"] += "Неправильный ввод материала";

return View("~/Views/Waybill/Index.cshtml", GetViewModel());

}

else

{

var waybill = db.Waybills.Where(item => item.Id == model.Id).FirstOrDefault();

waybill.ProviderId = model.ProviderId;

waybill.ProviderName = model.ProviderName;

waybill.Price = model.Price;

waybill.DateOfSupply = model.DateOfSupply;

waybill.EmployeeId = db.Employees.Where(item => item.FIO == model.EmployeeFIO).First().Id;

waybill.FurnitureId = db.Furniture.Where(item => item.Name == model.FurnitureName).First().Id;

waybill.Material = model.Material;

waybill.Weight = model.Weight;

db.SaveChanges();

cache.Remove("Waybills");

return RedirectToAction("Index", "Waybill");

}

}

private WaybillIndexViewModel GetViewModel(int page = 1, string providerName = null, string furnitName = "Все")

{

int pageSize = 20;

List<Employee> employees = db.Employees.ToList();

List<Furniture> furniture = db.Furniture.ToList();

List<WaybillViewModel> waybillViewModels;

if (!cache.TryGetValue("Waybills", out waybillViewModels))

{

List<Waybill> waybills = db.Waybills.ToList();

waybillViewModels = new List<WaybillViewModel>();

foreach (var waybill in waybills)

{

waybillViewModels.Add(new WaybillViewModel()

{

Id = waybill.Id,

ProviderId = waybill.ProviderId,

ProviderName = waybill.ProviderName,

EmployeeFIO = employees.Where(item => item.Id == waybill.EmployeeId).First().FIO,

DateOfSupply = waybill.DateOfSupply,

FurnitureName = furniture.Where(item => item.Id == waybill.FurnitureId).First().Name,

Material = waybill.Material,

Price = waybill.Price,

Weight = waybill.Weight

});

}

cache.Set("Waybills", db.Waybills.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

List<int> Ids = waybillViewModels.Select(item => item.Id).ToList();

List<string> furnitNames = furniture.Select(item => item.Name).ToList();

furnitNames.Add("Все");

if (providerName != null)

{

waybillViewModels = waybillViewModels.Where(item => item.ProviderName.Contains(providerName)).ToList();

}

if (furnitName != "Все")

{

waybillViewModels = waybillViewModels.Where(item => item.FurnitureName == furnitName).ToList();

}

WaybillIndexViewModel waybillIndexViewModel = new WaybillIndexViewModel()

{

WaybillViewModels = waybillViewModels.Skip((page - 1) \* pageSize).Take(pageSize).ToList(),

Ids = Ids,

EmployeesFIOs = employees.Select(item => item.FIO).ToList(),

FurnitureNames = furniture.Select(item => item.Name).ToList(),

PageViewModel = new PageViewModel(waybillViewModels.Count, page, pageSize),

FilterFurnitureNames = furnitNames

};

return waybillIndexViewModel;

}

}

}

*UserManageController.cs*

using CourseProject.Models;

using Microsoft.AspNetCore.Authorization;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using System.Linq;

using System.Threading.Tasks;

namespace CourseProject.Controllers

{

[Authorize(Roles = "Администратор")]

public class UserManageController : Controller

{

UserManager<User> \_userManager;

public UserManageController(UserManager<User> userManager)

{

\_userManager = userManager;

}

public IActionResult Index()

{

return View(\_userManager.Users.ToList());

}

public IActionResult Create()

{

return View();

}

[HttpPost]

public async Task<IActionResult> Create(CreateUserViewModel model)

{

if (ModelState.IsValid)

{

User user = new User { Email = model.Email, UserName = model.Email, Age = model.Age, FIO = model.FIO };

var result = await \_userManager.CreateAsync(user, model.Password);

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

return View(model);

}

public async Task<IActionResult> Edit(string id)

{

User user = await \_userManager.FindByIdAsync(id);

if (user == null)

{

return NotFound();

}

EditUserViewModel model = new EditUserViewModel { Id = user.Id, Email = user.Email, Age = user.Age, FIO = user.FIO };

return View(model);

}

[HttpPost]

public async Task<IActionResult> Edit(EditUserViewModel model)

{

if (ModelState.IsValid)

{

User user = await \_userManager.FindByIdAsync(model.Id);

if (user != null)

{

user.Email = model.Email;

user.UserName = model.Email;

user.Age = model.Age;

user.FIO = model.FIO;

var result = await \_userManager.UpdateAsync(user);

if (result.Succeeded)

{

return RedirectToAction("Index");

}

else

{

foreach (var error in result.Errors)

{

ModelState.AddModelError(string.Empty, error.Description);

}

}

}

}

return View(model);

}

[HttpPost]

public async Task<ActionResult> Delete(string id)

{

User user = await \_userManager.FindByIdAsync(id);

if (user != null)

{

await \_userManager.DeleteAsync(user);

}

return RedirectToAction("Index");

}

}

}

*Home/Index.cshtml*

@{

ViewData["Title"] = "Home Page";

}

<div class="text-center">

<h1 class="display-4">Welcome to the fourth task</h1>

<h2>Список таблиц в базе данных:</h2>

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Client" **asp-action**="Index">Клиенты</a>

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Employee" **asp-action**="Index">Работники</a>

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Furniture" **asp-action**="Index">Мебель</a>

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Order" **asp-action**="Index">Заказы</a>

<a class="nav-link text-dark" **asp-area**="" **asp-controller**="Waybill" **asp-action**="Index">Накладные</a>

</div>

*Account/Register.cshtml*

@model CourseProject.Models.RegisterViewModel

<h2>Регистрация нового пользователя</h2>

<form method="post" **asp-controller**="Account" **asp-action**="Register">

<div **asp-validation-summary**="ModelOnly"></div>

<div>

<label **asp-for**="UserRole"></label><br />

<select **asp-for**="UserRole">

<option>Базовый пользователь</option>

<option>Работник фабрики</option>

<option>Администратор</option>

</select>

</div>

<div>

<label **asp-for**="Email"></label><br />

<input **asp-for**="Email" />

<span **asp-validation-for**="Email"></span>

</div>

<div>

<label **asp-for**="FIO"></label><br />

<input **asp-for**="FIO" />

<span **asp-validation-for**="FIO"></span>

</div>

<div>

<label **asp-for**="Age"></label><br />

<input **asp-for**="Age" />

<span **asp-validation-for**="Age"></span>

</div>

<div>

<label **asp-for**="Password"></label><br />

<input **asp-for**="Password" />

<span **asp-validation-for**="Password"></span>

</div>

<div>

<label **asp-for**="PasswordConfirm"></label><br />

<input **asp-for**="PasswordConfirm" />

<span **asp-validation-for**="PasswordConfirm"></span>

</div>

<hr />

<div>

<input type="submit" value="Регистрация" />

</div>

</form>

*Account/Login.cshtml*

@model CourseProject.Models.LoginViewModel

<h2>Вход в приложение</h2>

<form method="post" **asp-controller**="Account" **asp-action**="Login"

**asp-route-returnUrl**="@Model.ReturnUrl">

<div **asp-validation-summary**="ModelOnly"></div>

<div>

<label **asp-for**="Email"></label><br />

<input **asp-for**="Email" />

<span **asp-validation-for**="Email"></span>

</div>

<div>

<label **asp-for**="Password"></label><br />

<input **asp-for**="Password" />

<span **asp-validation-for**="Password"></span>

</div>

<div>

<label **asp-for**="RememberMe"></label><br />

<input **asp-for**="RememberMe" />

</div>

<div>

<input type="submit" value="Войти" />

</div>

</form>

*Client/Index.cshtml*

@model ClientIndexViewModel

@{

ViewBag.Title = "Таблица клиентов";

}

<script>

function changeAddVisibility() {

document.getElementById('DeleteClient').style.display = "none";

document.getElementById('UpdateClient').style.display = "none";

document.getElementById('AddClient').style.display = "block";

}

function changeUpdateVisibility() {

document.getElementById('DeleteClient').style.display = "none";

document.getElementById('UpdateClient').style.display = "block";

document.getElementById('AddClient').style.display = "none";

}

function changeDeleteVisibility() {

document.getElementById('DeleteClient').style.display = "block";

document.getElementById('UpdateClient').style.display = "none";

document.getElementById('AddClient').style.display = "none";

}

function SetSession() {

sessionStorage.setItem("address", $("#value1").val());

sessionStorage.setItem("filterRepresFIO", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("address"));

$("#value2").val(sessionStorage.getItem("filterRepresFIO"));

}

</script>

<H1>@ViewBag.Title:</H1>

<body onbeforeunload="SetSession()" onload="GetSession()">

<form method="get">

<div>

<label class="control-label">Адрес:&nbsp;</label>

<input name="address" id="value1" class="form-control" />

</div>

<div>

<label class="control-label">ФИО представителя:&nbsp;</label>

<select name="filterRepresFIO" id="value2" class="form-control">

@foreach (var elem in Model.FilterRepresFIOs)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

<br />

<table border="1">

<thead>

<tr>

<td>

Id

</td>

<td>

Название

</td>

<td>

ФИО представителя

</td>

<td>

Номер

</td>

<td>

Адрес

</td>

</tr>

</thead>

@foreach (var elem in Model.Clients)

{

<tr>

<td>

@elem.Id

</td>

<td>

@elem.Name

</td>

<td>

@elem.RepresentativeFIO

</td>

<td>

@elem.Number

</td>

<td>

@elem.Address

</td>

</tr>

}

</table>

<br />

@if (Model.PageViewModel.HasPreviousPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-outline-dark" style="border-color:black;">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</a>

}

@if (Model.PageViewModel.HasNextPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-outline-dark" style="border-color:black;">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</a>

}

<br />

@if (User.IsInRole("Администратор"))

{

<div>

<input type="submit" style="border-color:black;" value="Добавить запись" onclick="changeAddVisibility()" />

<input type="submit" style="border-color:black;" value="Удалить запись" onclick="changeDeleteVisibility()" />

<input type="submit" style="border-color:black;" value="Изменить запись" onclick="changeUpdateVisibility()" />

</div>

<hr />

@ViewData["Message"]

<div>

<div id="AddClient" style="display:none;">

@{await Html.RenderPartialAsync("\_AddClient");}

</div>

<div id="DeleteClient" style="display:none;">

@{await Html.RenderPartialAsync("\_DeleteClient");}

</div>

<div id="UpdateClient" style="display:none;">

@{await Html.RenderPartialAsync("\_UpdateClient");}

</div>

</div>

}

</body>

*Client/\_AddClient.cshtml*

@model ClientIndexViewModel

<h3>

Добавление записи

</h3>

<hr />

@using (Html.BeginForm("AddClient", "Client", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Name)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Name)

</div>

</div>

<div>

@Html.LabelFor(m => m.RepresentativeFIO)

<div class="col-md-10">

@Html.TextBoxFor(m => m.RepresentativeFIO)

</div>

</div>

<div>

@Html.LabelFor(m => m.Number)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Number)

</div>

</div>

<div>

@Html.LabelFor(m => m.Address)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Address)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Добавить" />

</div>

</div>

}

*Client/\_DeleteClient.cshtml*

@model ClientIndexViewModel

<h3>

Удаление записи

</h3>

<hr />

@using (Html.BeginForm("DeleteClient", "Client", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Удалить" />

</div>

</div>

}

*Client/\_UpdateClient.cshtml*

@model ClientIndexViewModel

<h3>

Изменение записи

</h3>

@using (Html.BeginForm("UpdateClient", "Client", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.Name)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Name)

</div>

</div>

<div>

@Html.LabelFor(m => m.RepresentativeFIO)

<div class="col-md-10">

@Html.TextBoxFor(m => m.RepresentativeFIO)

</div>

</div>

<div>

@Html.LabelFor(m => m.Number)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Number)

</div>

</div>

<div>

@Html.LabelFor(m => m.Address)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Address)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Изменить" />

</div>

</div>

}

*Employee/Index.cshtml*

@model EmployeeIndexViewModel

@{

ViewBag.Title = "Таблица работников";

}

<script>

function changeAddVisibility() {

document.getElementById('DeleteEmployee').style.display = "none";

document.getElementById('UpdateEmployee').style.display = "none";

document.getElementById('AddEmployee').style.display = "block";

}

function changeDeleteVisibility() {

document.getElementById('DeleteEmployee').style.display = "block";

document.getElementById('UpdateEmployee').style.display = "none";

document.getElementById('AddEmployee').style.display = "none";

}

function changeUpdateVisibility() {

document.getElementById('DeleteEmployee').style.display = "none";

document.getElementById('UpdateEmployee').style.display = "block";

document.getElementById('AddEmployee').style.display = "none";

}

function SetSession() {

sessionStorage.setItem("education", $("#value1").val());

sessionStorage.setItem("position", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("education"));

$("#value2").val(sessionStorage.getItem("position"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Образование:&nbsp;</label>

<input name="education" id="value1" class="form-control" />

</div>

<div>

<label class="control-label">Должностm:&nbsp;</label>

<select name="position" id="value2" class="form-control">

@foreach (var elem in Model.FilterPositions)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

<br />

<div>

<table border="1">

<thead>

<tr>

<td>

Id

</td>

<td>

ФИО

</td>

<td>

Должность

</td>

<td>

Образование

</td>

</tr>

</thead>

@foreach (var elem in Model.Employees)

{

<tr>

<td>

@elem.Id

</td>

<td>

@elem.FIO

</td>

<td>

@elem.Position

</td>

<td>

@elem.Education

</td>

</tr>

}

</table>

</div>

<br />

@if (Model.PageViewModel.HasPreviousPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-outline-dark">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</a>

}

@if (Model.PageViewModel.HasNextPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-outline-dark">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</a>

}

<br />

@if (User.IsInRole("Администратор"))

{

<div>

<input type="submit" value="Добавить запись" onclick="changeAddVisibility()" />

<input type="submit" value="Удалить запись" onclick="changeDeleteVisibility()" />

<input type="submit" value="Изменить запись" onclick="changeUpdateVisibility()" />

</div>

<hr />

@ViewData["Message"]

<div>

<div id="AddEmployee" style="display:none;">

@{await Html.RenderPartialAsync("\_AddEmployee");}

</div>

<div id="DeleteEmployee" style="display:none;">

@{await Html.RenderPartialAsync("\_DeleteEmployee");}

</div>

<div id="UpdateEmployee" style="display:none;">

@{await Html.RenderPartialAsync("\_UpdateEmployee");}

</div>

</div>

}

</body>

*Employee/\_AddEmployee.cshtml*

@model EmployeeIndexViewModel

<h3>

Добавление записи

</h3>

<hr />

@using (Html.BeginForm("AddEmployee", "Employee", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.FIO)

<div class="col-md-10">

@Html.TextBoxFor(m => m.FIO)

</div>

</div>

<div>

@Html.LabelFor(m => m.Position)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Position)

</div>

</div>

<div>

@Html.LabelFor(m => m.Education)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Education)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Добавить" />

</div>

</div>

}

*Employee/\_DeleteEmployee.cshtml*

@model EmployeeIndexViewModel

<h3>

Удаление записи

</h3>

<hr />

@using (Html.BeginForm("DeleteEmployee", "Employee", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Удалить" />

</div>

</div>

}

*Employee/\_UpdateEmployee.cshtml*

@model EmployeeIndexViewModel

<h3>

Изменение записи

</h3>

@using (Html.BeginForm("UpdateEmployee", "Employee", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.FIO)

<div class="col-md-10">

@Html.TextBoxFor(m => m.FIO)

</div>

</div>

<div>

@Html.LabelFor(m => m.Position)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Position)

</div>

</div>

<div>

@Html.LabelFor(m => m.Education)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Education)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Изменить" />

</div>

</div>

}

*Furniture/Index.cshtml*

@model FurnitureIndexViewModel

@{

ViewBag.Title = "Таблица мебели";

}

<script>

function changeAddVisibility() {

document.getElementById('DeleteFurniture').style.display = "none";

document.getElementById('UpdateFurniture').style.display = "none";

document.getElementById('AddFurniture').style.display = "block";

}

function changeDeleteVisibility() {

document.getElementById('DeleteFurniture').style.display = "block";

document.getElementById('UpdateFurniture').style.display = "none";

document.getElementById('AddFurniture').style.display = "none";

}

function changeUpdateVisibility() {

document.getElementById('DeleteFurniture').style.display = "none";

document.getElementById('UpdateFurniture').style.display = "block";

document.getElementById('AddFurniture').style.display = "none";

}

function SetSession() {

sessionStorage.setItem("price", $("#value1").val());

sessionStorage.setItem("material", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("price"));

$("#value2").val(sessionStorage.getItem("material"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Стоимость:&nbsp;</label>

<input name="price" type="number" min="0" id="value1" class="form-control" />

</div>

<div>

<label class="control-label">Материалы:&nbsp;</label>

<select name="material" id="value2" class="form-control">

@foreach (var elem in Model.FilterMaterials)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

<br />

<div>

<table border="1">

<thead>

<tr>

<td>

Id

</td>

<td>

Название

</td>

<td>

Описание

</td>

<td>

Материал

</td>

<td>

Цена

</td>

<td>

Количество

</td>

</tr>

</thead>

@foreach (var elem in Model.Furniture)

{

<tr>

<td>

@elem.Id

</td>

<td>

@elem.Name

</td>

<td>

@elem.Description

</td>

<td>

@elem.Material

</td>

<td>

@elem.Price

</td>

<td>

@elem.Count

</td>

</tr>

}

</table>

</div>

<br />

@if (Model.PageViewModel.HasPreviousPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-outline-dark">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</a>

}

@if (Model.PageViewModel.HasNextPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-outline-dark">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</a>

}

<br />

@if (User.IsInRole("Администратор") || User.IsInRole("Работник фабрики"))

{

<div>

<input type="submit" value="Добавить запись" onclick="changeAddVisibility()" />

<input type="submit" value="Удалить запись" onclick="changeDeleteVisibility()" />

<input type="submit" value="Изменить запись" onclick="changeUpdateVisibility()" />

</div>

<hr />

@ViewData["Message"]

<div>

<div id="AddFurniture" style="display:none;">

@{await Html.RenderPartialAsync("\_AddFurniture");}

</div>

<div id="DeleteFurniture" style="display:none;">

@{await Html.RenderPartialAsync("\_DeleteFurniture");}

</div>

<div id="UpdateFurniture" style="display:none;">

@{await Html.RenderPartialAsync("\_UpdateFurniture");}

</div>

</div>

}

</body>

*Furniture/\_AddFurniture.cshtml*

@model FurnitureIndexViewModel

<h3>

Добавление записи

</h3>

<hr />

@using (Html.BeginForm("AddFurniture", "Furniture", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Name)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Name)

</div>

</div>

<div>

@Html.LabelFor(m => m.Description)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Description)

</div>

</div>

<div>

@Html.LabelFor(m => m.Material)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Material)

</div>

</div>

<div>

@Html.LabelFor(m => m.Price)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Price)

</div>

</div>

<div>

@Html.LabelFor(m => m.Count)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Count)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Добавить" />

</div>

</div>

}

*Furniture/\_DeleteFurniture.cshtml*

@model FurnitureIndexViewModel

<h3>

Удаление записи

</h3>

<hr />

@using (Html.BeginForm("DeleteFurniture", "Furniture", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Удалить" />

</div>

</div>

}

*Furniture/\_UpdateFurniture.cshtml*

@model FurnitureIndexViewModel

<h3>

Изменение записи

</h3>

@using (Html.BeginForm("UpdateFurniture", "Furniture", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.Name)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Name)

</div>

</div>

<div>

@Html.LabelFor(m => m.Description)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Description)

</div>

</div>

<div>

@Html.LabelFor(m => m.Material)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Material)

</div>

</div>

<div>

@Html.LabelFor(m => m.Price)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Price)

</div>

</div>

<div>

@Html.LabelFor(m => m.Count)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Count)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Изменить" />

</div>

</div>

}

*Order/Index.cshtml*

@model OrderIndexViewModel

@{

ViewBag.Title = "Таблица заказов";

}

<script>

function changeAddVisibility() {

document.getElementById('DeleteOrder').style.display = "none";

document.getElementById('UpdateOrder').style.display = "none";

document.getElementById('AddOrder').style.display = "block";

}

function changeDeleteVisibility() {

document.getElementById('DeleteOrder').style.display = "block";

document.getElementById('UpdateOrder').style.display = "none";

document.getElementById('AddOrder').style.display = "none";

}

function changeUpdateVisibility() {

document.getElementById('DeleteOrder').style.display = "none";

document.getElementById('UpdateOrder').style.display = "block";

document.getElementById('AddOrder').style.display = "none";

}

function SetSession() {

sessionStorage.setItem("clientName", $("#value1 option:selected").val());

sessionStorage.setItem("furniture", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("clientName"));

$("#value2").val(sessionStorage.getItem("furniture"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Название клиента:&nbsp;</label>

<select name="clientName" id="value1" class="form-control">

@foreach (var elem in Model.FilterClients)

{

<option>@elem</option>

}

</select>

</div>

<div>

<label class="control-label">Название мебели:&nbsp;</label>

<select name="furniture" id="value2" class="form-control">

@foreach (var elem in Model.FilterFurnitures)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

<br />

<div>

<table border="1">

<thead>

<tr>

<td>

Id

</td>

<td>

Название клиента

</td>

<td>

Название мебели

</td>

<td>

Кол-во мебели

</td>

<td>

Цена

</td>

<td>

Скидка

</td>

<td>

Выполнен?

</td>

<td>

ФИО сотрудника

</td>

</tr>

</thead>

@foreach (var elem in Model.OrderViewModels)

{

<tr>

<td>

@elem.Id

</td>

<td>

@elem.ClientName

</td>

<td>

@elem.FurnitureName

</td>

<td>

@elem.FurnitureCount

</td>

<td>

@elem.Price

</td>

<td>

@elem.DiscountPercent

%

</td>

@if (elem.IsCompleted)

{

<td>

Да

</td>

}

else

{

<td>

Нет

</td>

}

<td>

@elem.EmployeeFIO

</td>

</tr>

}

</table>

</div>

<br />

@if (Model.PageViewModel.HasPreviousPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-outline-dark">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</a>

}

@if (Model.PageViewModel.HasNextPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-outline-dark">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</a>

}

<br />

@if (User.IsInRole("Администратор") || User.IsInRole("Работник фабрики"))

{

<div>

<input type="submit" value="Добавить запись" onclick="changeAddVisibility()" />

<input type="submit" value="Удалить запись" onclick="changeDeleteVisibility()" />

<input type="submit" value="Изменить запись" onclick="changeUpdateVisibility()" />

</div>

<hr />

@ViewData["Message"]

<div>

<div id="AddOrder" style="display:none;">

@{await Html.RenderPartialAsync("\_AddOrder");}

</div>

<div id="DeleteOrder" style="display:none;">

@{await Html.RenderPartialAsync("\_DeleteOrder");}

</div>

<div id="UpdateOrder" style="display:none;">

@{await Html.RenderPartialAsync("\_UpdateOrder");}

</div>

</div>

}

</body>

*Order/\_AddOrder.cshtml*

@model OrderIndexViewModel

<h3>

Добавление записи

</h3>

<hr />

@using (Html.BeginForm("AddOrder", "Order", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.ClientName)

<div class="col-md-10">

<select name="ClientName">

@foreach (var name in Model.ClientNames)

{

<option>@name</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.EmployeeFIO)

<div class="col-md-10">

<select name="EmployeeFIO">

@foreach (var fio in Model.EmployeesFIOs)

{

<option>@fio</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.FurnitureName)

<div class="col-md-10">

<select name="FurnitureName">

@foreach (var name in Model.FurnitureNames)

{

<option>@name</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.FurnitureCount)

<div class="col-md-10">

@Html.TextBoxFor(m => m.FurnitureCount)

</div>

</div>

<div>

@Html.LabelFor(m => m.DiscountPercent)

<div class="col-md-10">

@Html.TextBoxFor(m => m.DiscountPercent)

</div>

</div>

<div>

@Html.LabelFor(m => m.Price)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Price)

</div>

</div>

<div>

@Html.LabelFor(m => m.IsCompleted)

<div class="col-md-10">

@Html.CheckBoxFor(m => m.IsCompleted)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Добавить" />

</div>

</div>

}

*Order/\_DeleteOrder.cshtml*

@model OrderIndexViewModel

<h3>

Удаление записи

</h3>

<hr />

@using (Html.BeginForm("DeleteOrder", "Order", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Удалить" />

</div>

</div>

}

*Order/\_UpdateOrder.cshtml*

@model OrderIndexViewModel

<h3>

Изменение записи

</h3>

@using (Html.BeginForm("UpdateOrder", "Order", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.ClientName)

<div class="col-md-10">

<select name="ClientName">

@foreach (var name in Model.ClientNames)

{

<option>@name</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.EmployeeFIO)

<div class="col-md-10">

<select name="EmployeeFIO">

@foreach (var fio in Model.EmployeesFIOs)

{

<option>@fio</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.FurnitureName)

<div class="col-md-10">

<select name="FurnitureName">

@foreach (var name in Model.FurnitureNames)

{

<option>@name</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.FurnitureCount)

<div class="col-md-10">

@Html.TextBoxFor(m => m.FurnitureCount)

</div>

</div>

<div>

@Html.LabelFor(m => m.DiscountPercent)

<div class="col-md-10">

@Html.TextBoxFor(m => m.DiscountPercent)

</div>

</div>

<div>

@Html.LabelFor(m => m.Price)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Price)

</div>

</div>

<div>

@Html.LabelFor(m => m.IsCompleted)

<div class="col-md-10">

@Html.CheckBoxFor(m => m.IsCompleted)

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Изменить" />

</div>

</div>

}

*Shared/\_Layout.cshtml*

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - CourseProject</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-light bg-white border-bottom box-shadow mb-3">

<div class="container">

<a class="navbar-brand" **asp-area**="" **asp-controller**="Home" **asp-action**="Index">Домашняя страница</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

</div>

<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">

<**partial** **name**="\_LoginPartial" />

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2020 - CourseProject - <a **asp-area**="" **asp-controller**="Home" **asp-action**="Privacy">Privacy</a>

</div>

</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script **src**="~/js/site.js" **asp-append-version**="true"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

*Shared/\_LoginPartial.cshtml*

@using Microsoft.AspNetCore.Identity

@inject SignInManager<User> SignInManager

@inject UserManager<User> UserManager

<ul class="navbar-nav">

@if (SignInManager.IsSignedIn(User))

{

if (User.IsInRole("Администратор"))

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-controller**="UserManage" **asp-action**="Index" title="Manage">Пользователи</a>

</li>

}

else

{

<li class="nav-item">

<label class="col-form-label">Здравствуйте, @User.Identity.Name</label>

</li>

}

<li class="nav-item">

<form class="form-inline" **asp-controller**="Account" **asp-action**="Logout">

<button type="submit" class="nav-link btn btn-link text-dark">Выход</button>

</form>

</li>

}

else

{

<li class="nav-item">

<a class="nav-link text-dark" **asp-controller**="Account" **asp-action**="Register">Регистрация</a>

</li>

<li class="nav-item">

<a class="nav-link text-dark" **asp-controller**="Account" **asp-action**="Login">Вход</a>

</li>

}

</ul>

*UserManage/Create.cshtml*

@model CourseProject.Models.CreateUserViewModel

@{

ViewBag.Title = "Добавление пользователя";

}

<h1>@ViewBag.Title</h1>

<form **asp-action**="Create" **asp-controller**="UserManage">

<div **asp-validation-summary**="All" class="text-danger"></div>

<div class="form-group">

<label **asp-for**="Email" class="control-label">Email</label>

<input **type**="text" **asp-for**="Email" class="form-control" />

</div>

<div class="form-group">

<label **asp-for**="Password" class="control-label">Пароль</label>

<input **type**="password" **asp-for**="Password" class="form-control" />

</div>

<div class="form-group">

<label **asp-for**="FIO" class="control-label">ФИО</label>

<input **type**="text" **asp-for**="FIO" class="form-control" />

</div>

<div class="form-group">

<label **asp-for**="Age" class="control-label">Возраст</label>

<input **type**="number" **asp-for**="Age" class="form-control" min="16" max="67" **value**="16"/>

</div>

<div class="form-group">

<input type="submit" value="Добавить" class="btn btn-outline-secondary" />

</div>

</form>

*UserManage/Edit.cshtml*

@model CourseProject.Models.EditUserViewModel

@{

ViewBag.Title = "Редактирование пользователя";

}

<h1>@ViewBag.Title</h1>

<form **asp-action**="Edit" **asp-controller**="UserManage">

<div **asp-validation-summary**="All" class="text-danger"></div>

<div class="form-group">

<input **type**="hidden" **asp-for**="Id" />

</div>

<div class="form-group">

<label **asp-for**="Email" class="control-label">Email</label>

<input **type**="text" **asp-for**="Email" class="form-control" />

</div>

<div class="form-group">

<label **asp-for**="FIO" class="control-label">ФИО</label>

<input **type**="text" **asp-for**="FIO" class="form-control" />

</div>

<div class="form-group">

<label **asp-for**="Age" class="control-label">Возраст</label>

<input **type**="number" **asp-for**="Age" class="form-control" min="16" max="67" **value**="16"/>

</div>

<div class="form-group">

<input type="submit" value="Сохранить" class="btn btn-outline-secondary" />

</div>

</form>

*UserManage/Index.cshtml*

@model IEnumerable<CourseProject.Models.User>

@{

ViewBag.Title = "Список пользователей";

}

<a **asp-action**="Create">Добавить пользователя</a>

<table class="table">

<tr><th>Email</th><th>ФИО</th><th>Возраст</th></tr>

@foreach (var user in Model)

{

<tr>

<td>@user.Email</td>

<td>@user.FIO</td>

<td>@user.Age</td>

<td>

<form **asp-action**="Delete" **asp-route-id**="@user.Id" method="post">

<button type="submit" class="btn btn-sm btn-danger">

Удалить

</button>

</form>

</td>

<td>

<form **asp-action**="Edit" **asp-route-id**="@user.Id" method="post">

<button type="submit" class="btn btn-sm btn-secondary">

Изменить

</button>

</form>

</td>

</tr>

}

</table>

*Waybill/Index.cshtml*

@model WaybillIndexViewModel

@{

ViewBag.Title = "Таблица накладных";

}

<script>

function changeAddVisibility() {

document.getElementById('DeleteWaybill').style.display = "none";

document.getElementById('UpdateWaybill').style.display = "none";

document.getElementById('AddWaybill').style.display = "block";

}

function changeUpdateVisibility() {

document.getElementById('DeleteWaybill').style.display = "none";

document.getElementById('UpdateWaybill').style.display = "block";

document.getElementById('AddWaybill').style.display = "none";

}

function changeDeleteVisibility() {

document.getElementById('DeleteWaybill').style.display = "block";

document.getElementById('UpdateWaybill').style.display = "none";

document.getElementById('AddWaybill').style.display = "none";

}

function SetSession() {

sessionStorage.setItem("providerName", $("#value1").val());

sessionStorage.setItem("furnitName", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("providerName"));

$("#value2").val(sessionStorage.getItem("furnitName"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Название поставщика:&nbsp;</label>

<input type="text" name="providerName" id="value1" class="form-control"/>

</div>

<div>

<label class="control-label">Название мебели:&nbsp;</label>

<select name="furnitName" id="value2" class="form-control">

@foreach (var elem in Model.FilterFurnitureNames)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

<br />

<div>

<table border="1">

<thead>

<tr>

<td>

Id

</td>

<td>

Номер поставщика

</td>

<td>

Название поставщика

</td>

<td>

Дата поставок

</td>

<td>

Материал

</td>

<td>

Стоимость

</td>

<td>

Вес

</td>

<td>

Название мебели

</td>

<td>

ФИО сотрудника

</td>

</tr>

</thead>

@foreach (var elem in Model.WaybillViewModels)

{

<tr>

<td>

@elem.Id

</td>

<td>

@elem.ProviderId

</td>

<td>

@elem.ProviderName

</td>

<td>

@elem.DateOfSupply

</td>

<td>

@elem.Material

</td>

<td>

@elem.Price

</td>

<td>

@elem.Weight

</td>

<td>

@elem.FurnitureName

</td>

<td>

@elem.EmployeeFIO

</td>

</tr>

}

</table>

</div>

@if (Model.PageViewModel.HasPreviousPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber - 1)"

class="btn btn-outline-dark">

<i class="glyphicon glyphicon-chevron-left"></i>

Назад

</a>

}

@if (Model.PageViewModel.HasNextPage)

{

<a **asp-action**="Index"

**asp-route-page**="@(Model.PageViewModel.PageNumber + 1)"

class="btn btn-outline-dark">

Вперед

<i class="glyphicon glyphicon-chevron-right"></i>

</a>

}

@if (User.IsInRole("Администратор"))

{

<div>

<input type="submit" value="Добавить запись" onclick="changeAddVisibility()" />

<input type="submit" value="Удалить запись" onclick="changeDeleteVisibility()" />

<input type="submit" value="Изменить запись" onclick="changeUpdateVisibility()" />

</div>

<hr />

@ViewData["Message"]

<div>

<div id="AddWaybill" style="display:none;">

@{await Html.RenderPartialAsync("\_AddWaybill");}

</div>

<div id="DeleteWaybill" style="display:none;">

@{await Html.RenderPartialAsync("\_DeleteWaybill");}

</div>

<div id="UpdateWaybill" style="display:none;">

@{await Html.RenderPartialAsync("\_UpdateWaybill");}

</div>

</div>

}

</body>

*Waybill/\_AddWaybill.cshtml*

@model WaybillIndexViewModel

<h3>

Добавление записи

</h3>

<hr />

@using (Html.BeginForm("AddWaybill", "Waybill", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.EmployeeFIO)

<div class="col-md-10">

<select name="EmployeeFIO">

@foreach (var fio in Model.EmployeesFIOs)

{

<option>@fio</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.FurnitureName)

<div class="col-md-10">

<select name="FurnitureName">

@foreach (var name in Model.FurnitureNames)

{

<option>@name</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.ProviderId)

<div class="col-md-10">

@Html.TextBoxFor(m => m.ProviderId)

</div>

</div>

<div>

@Html.LabelFor(m => m.ProviderName)

<div class="col-md-10">

@Html.TextBoxFor(m => m.ProviderName)

</div>

</div>

<div>

@Html.LabelFor(m => m.Material)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Material)

</div>

</div>

<div>

@Html.LabelFor(m => m.Weight)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Weight)

</div>

</div>

<div>

@Html.LabelFor(m => m.Price)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Price)

</div>

</div>

<div>

@Html.LabelFor(m => m.DateOfSupply)

<div class="col-md-10">

<input type="date" id="datepicker" name="DateOfSupply" min="2000-01-01" max="2022-01-01" />

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Добавить" />

</div>

</div>

}

*Waybill/\_DeleteWaybill.cshtml*

@model WaybillIndexViewModel

<h3>

Удаление записи

</h3>

<hr />

@using (Html.BeginForm("DeleteWaybill", "Waybill", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Удалить" />

</div>

</div>

}

*Waybill/\_UpdateWaybill.cshtml*

@model WaybillIndexViewModel

<h3>

Изменение записи

</h3>

@using (Html.BeginForm("UpdateWaybill", "Waybill", FormMethod.Post))

{

<div>

@Html.LabelFor(m => m.Id)

<div class="col-md-10">

<select name="Id">

@foreach (var id in Model.Ids)

{

<option>@id</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.EmployeeFIO)

<div class="col-md-10">

<select name="EmployeeFIO">

@foreach (var fio in Model.EmployeesFIOs)

{

<option>@fio</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.FurnitureName)

<div class="col-md-10">

<select name="FurnitureName">

@foreach (var name in Model.FurnitureNames)

{

<option>@name</option>

}

</select>

</div>

</div>

<div>

@Html.LabelFor(m => m.ProviderId)

<div class="col-md-10">

@Html.TextBoxFor(m => m.ProviderId)

</div>

</div>

<div>

@Html.LabelFor(m => m.ProviderName)

<div class="col-md-10">

@Html.TextBoxFor(m => m.ProviderName)

</div>

</div>

<div>

@Html.LabelFor(m => m.Material)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Material)

</div>

</div>

<div>

@Html.LabelFor(m => m.Weight)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Weight)

</div>

</div>

<div>

@Html.LabelFor(m => m.Price)

<div class="col-md-10">

@Html.TextBoxFor(m => m.Price)

</div>

</div>

<div>

@Html.LabelFor(m => m.DateOfSupply)

<div class="col-md-10">

<input type="date" id="datepicker" name="DateOfSupply" min="2000-01-01" max="2022-01-01" />

</div>

</div>

<div>

<div>

<input type="submit" class="btn btn-default" value="Изменить" />

</div>

</div>

}

Результаты:

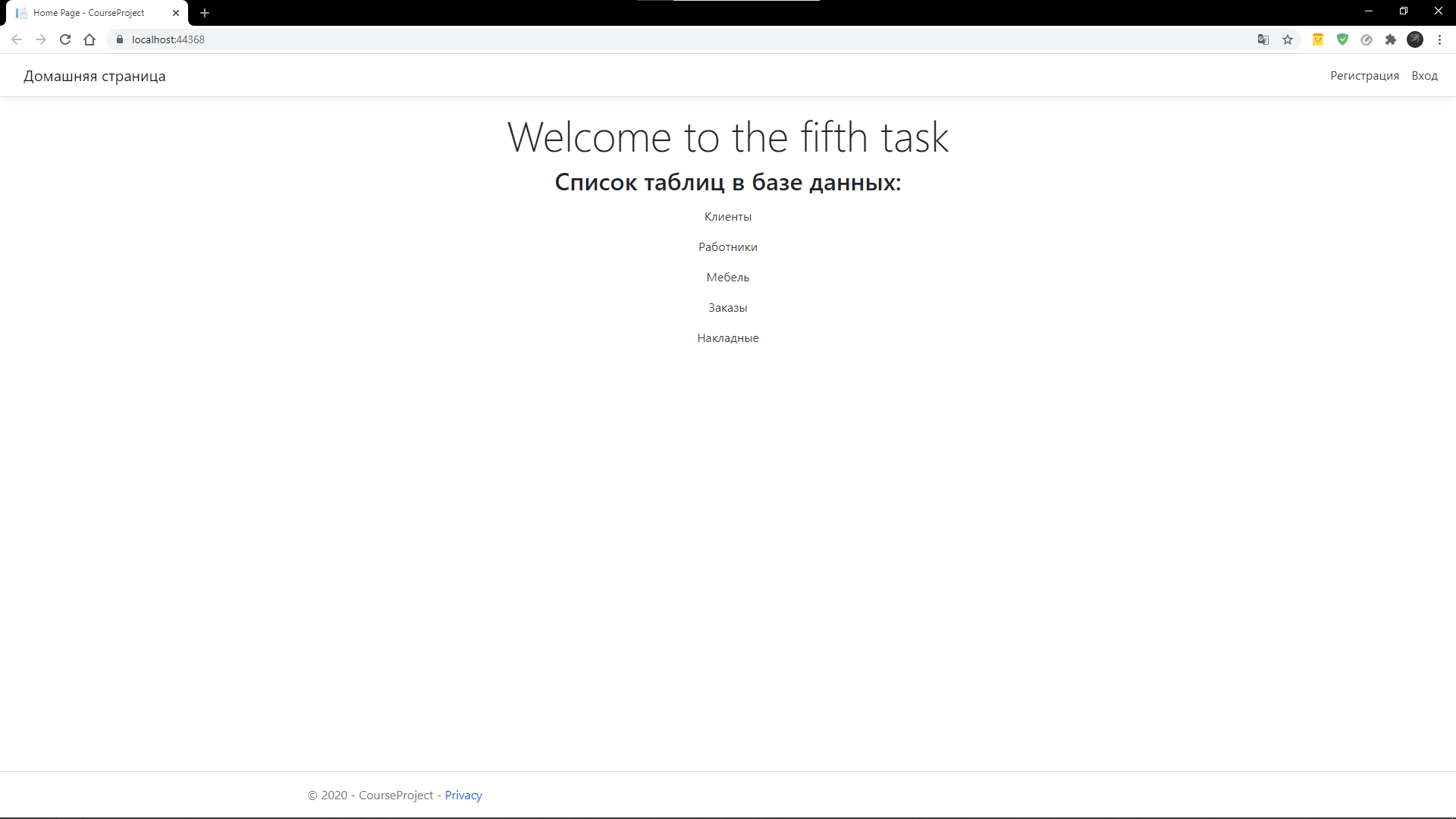


Рисунок 2 – окно меню

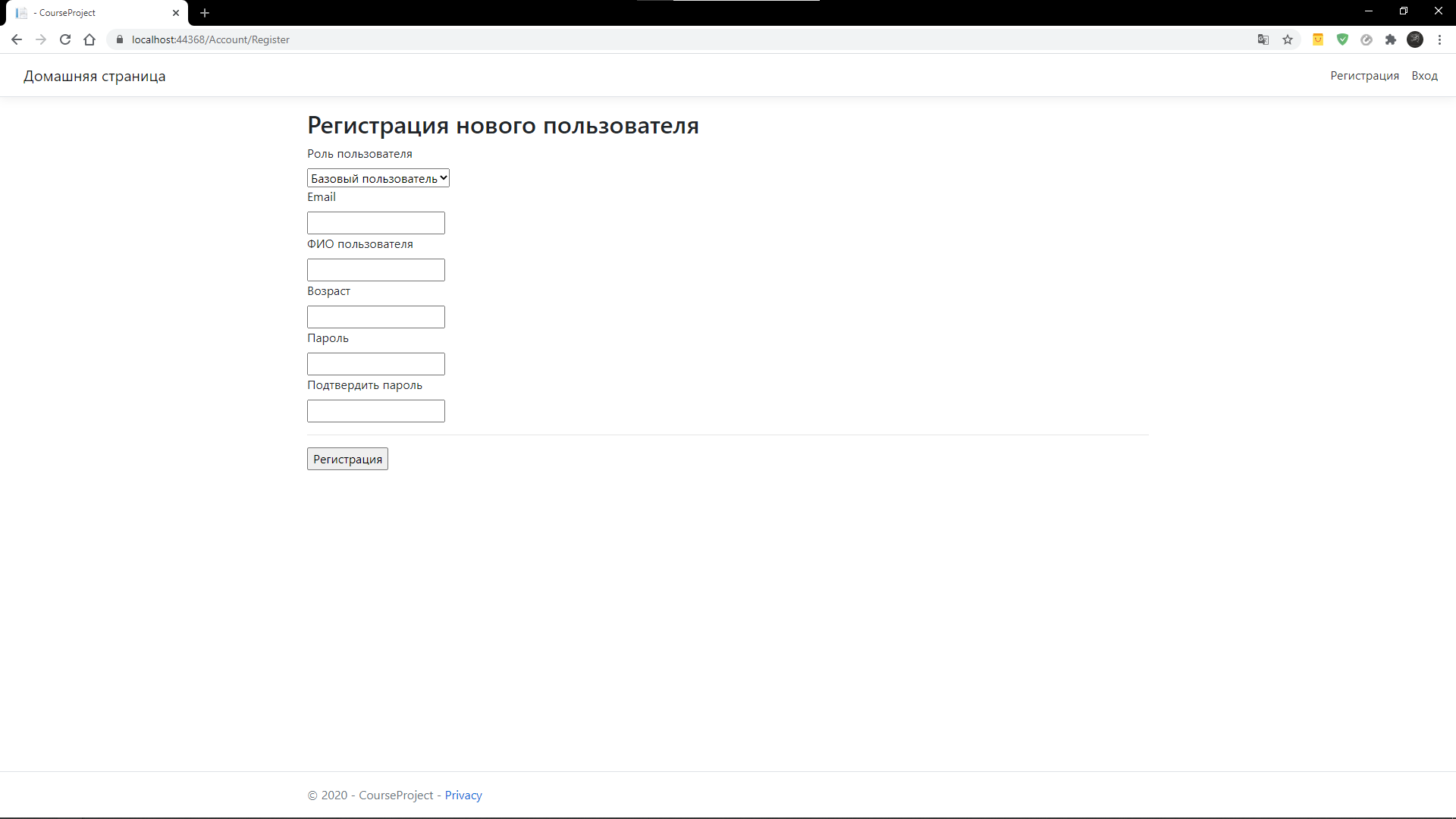


Рисунок 3 – страница регистрации

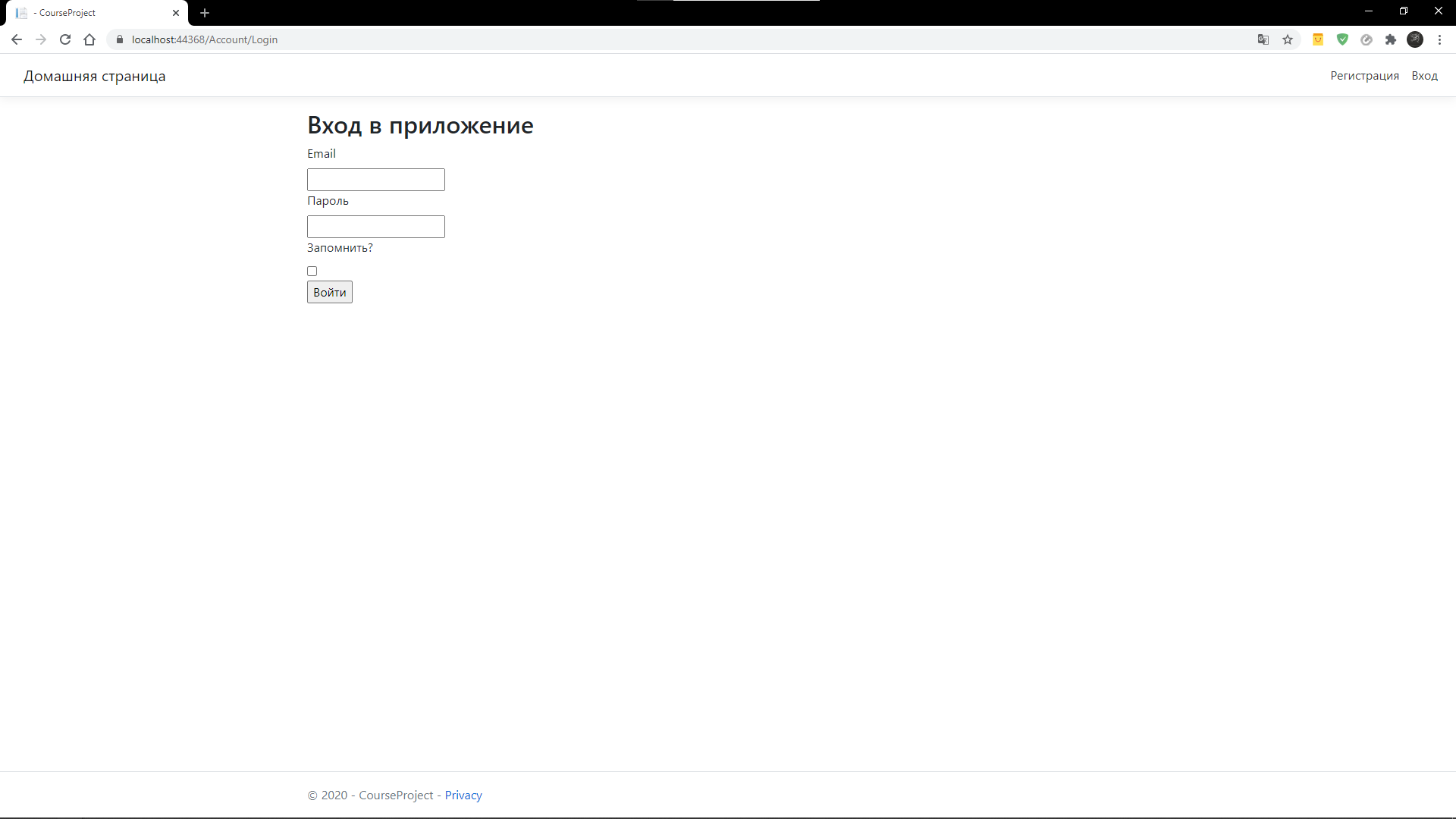


Рисунок 4 – страница входа

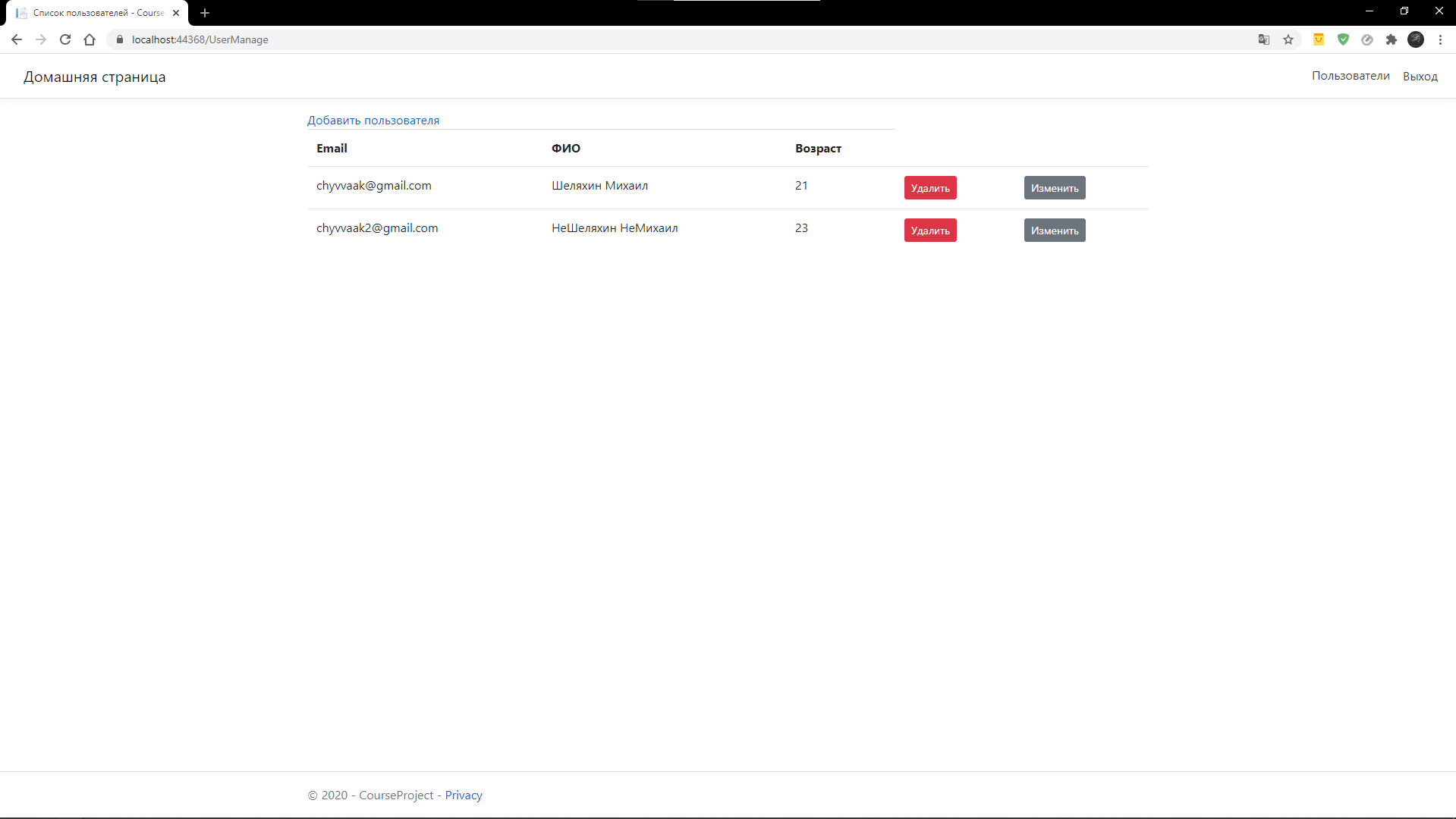


Рисунок 5 – страница пользователей

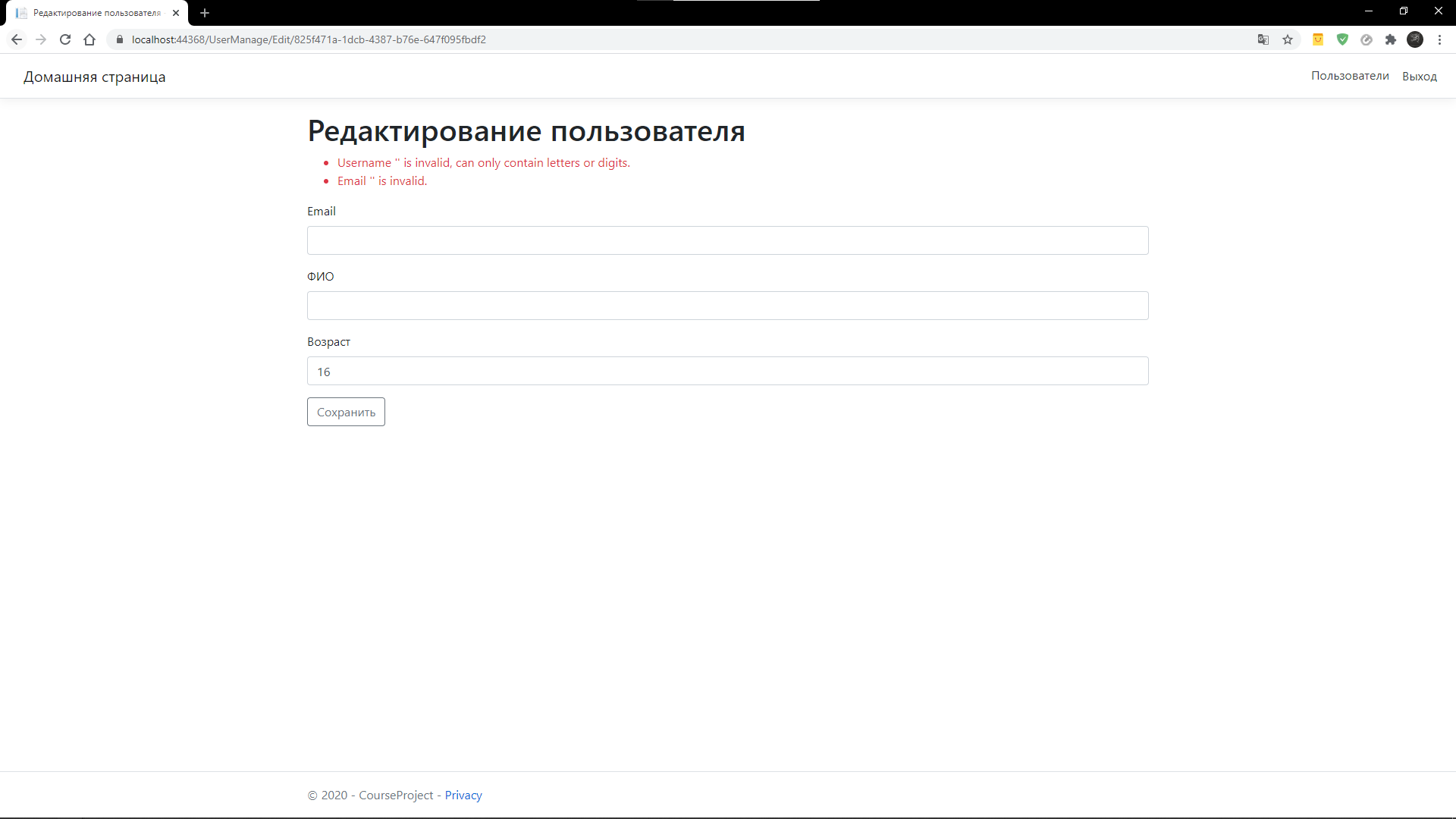
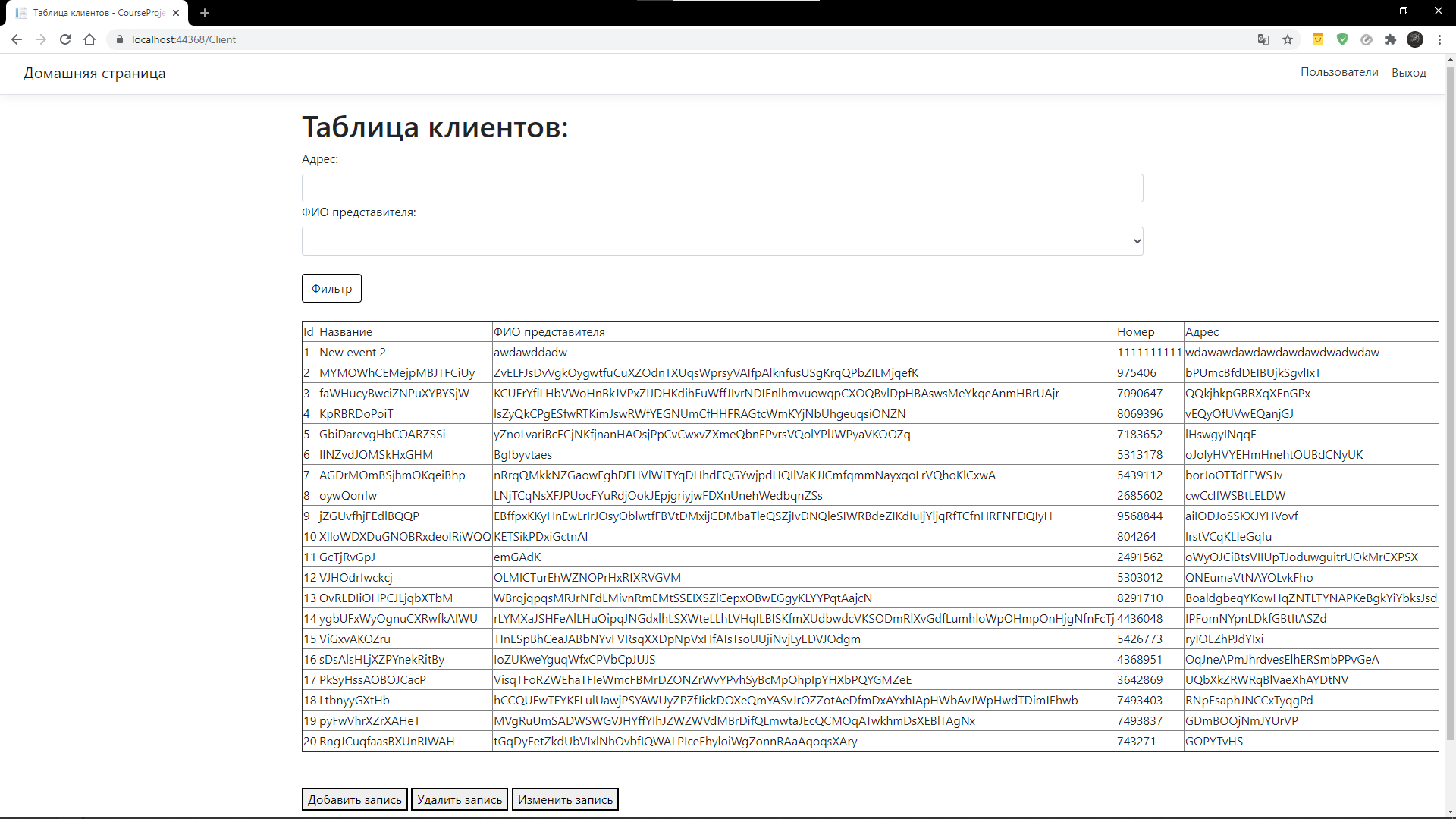
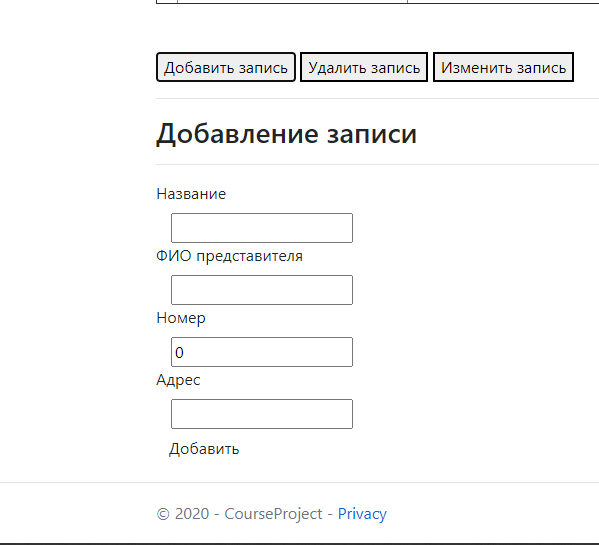
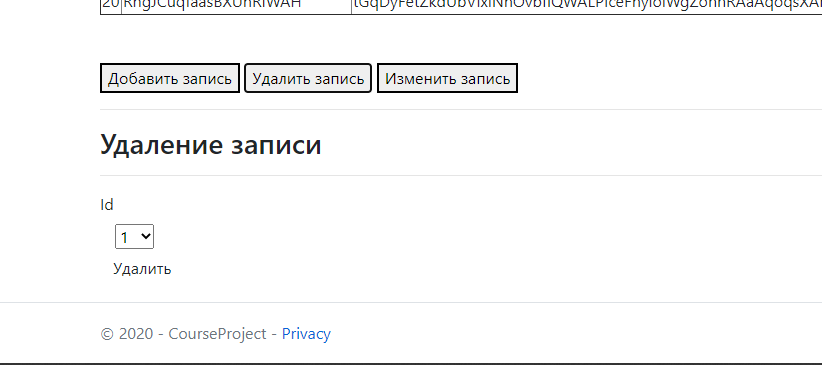


Рисунок 6 – страница редактирования пользователя







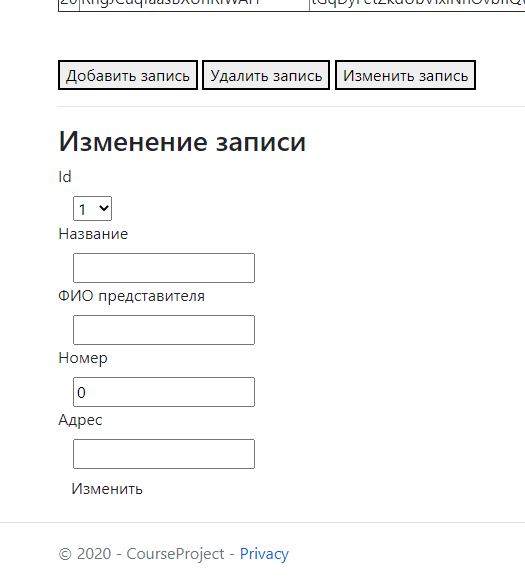
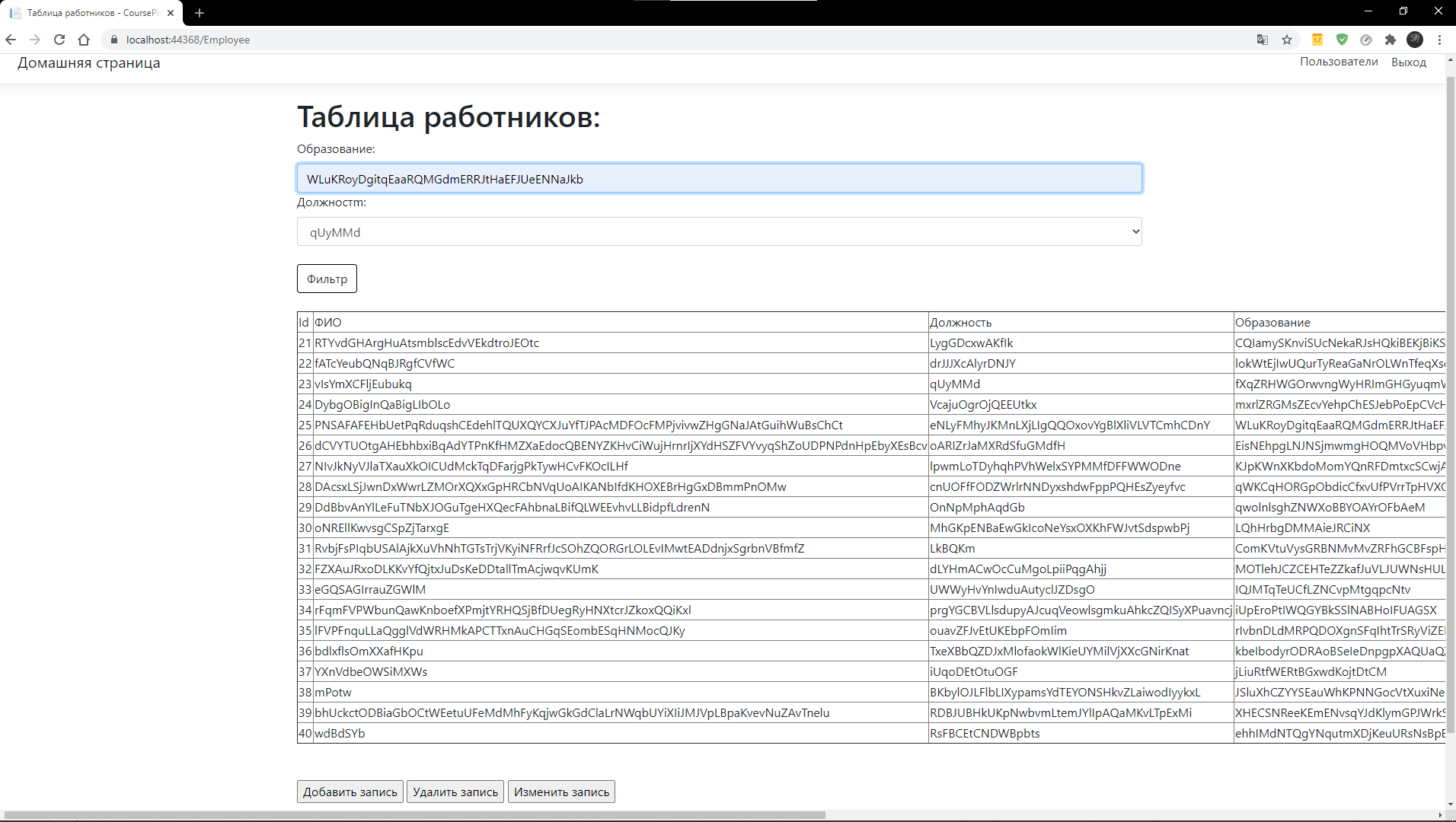
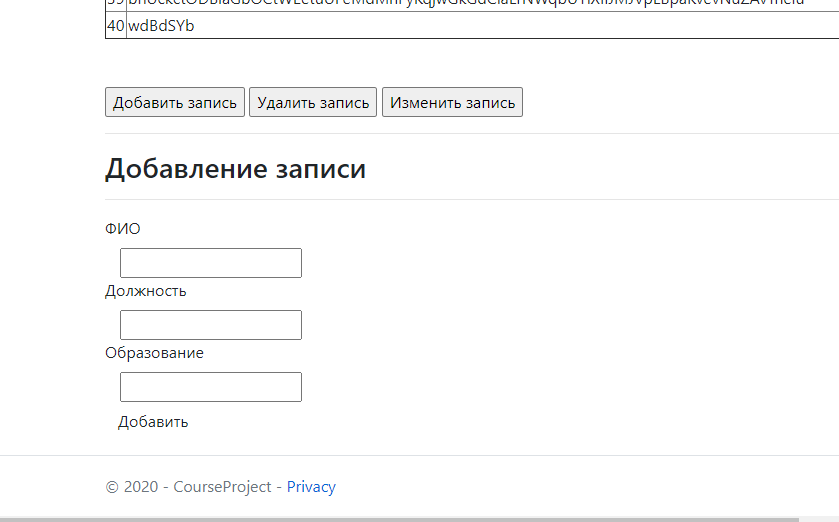
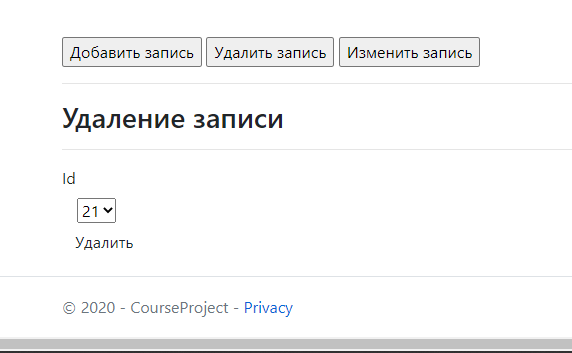


Рисунок 7-10 – страница клиентов с функционалом управления записями для роли «Администратор»







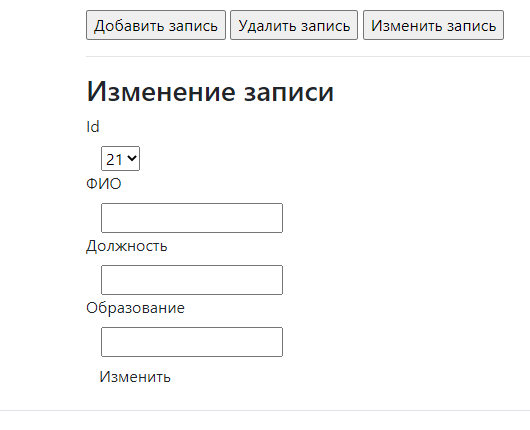
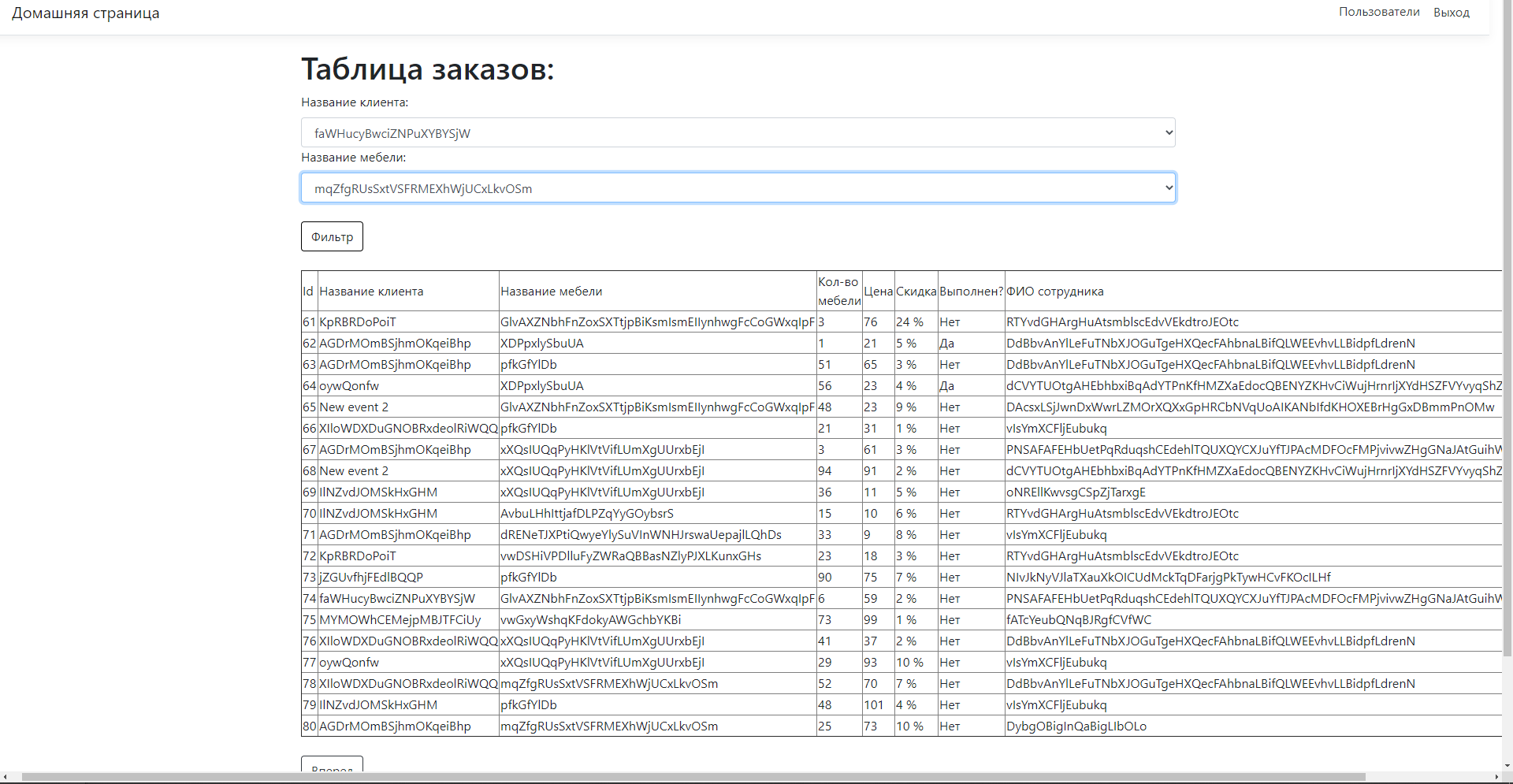
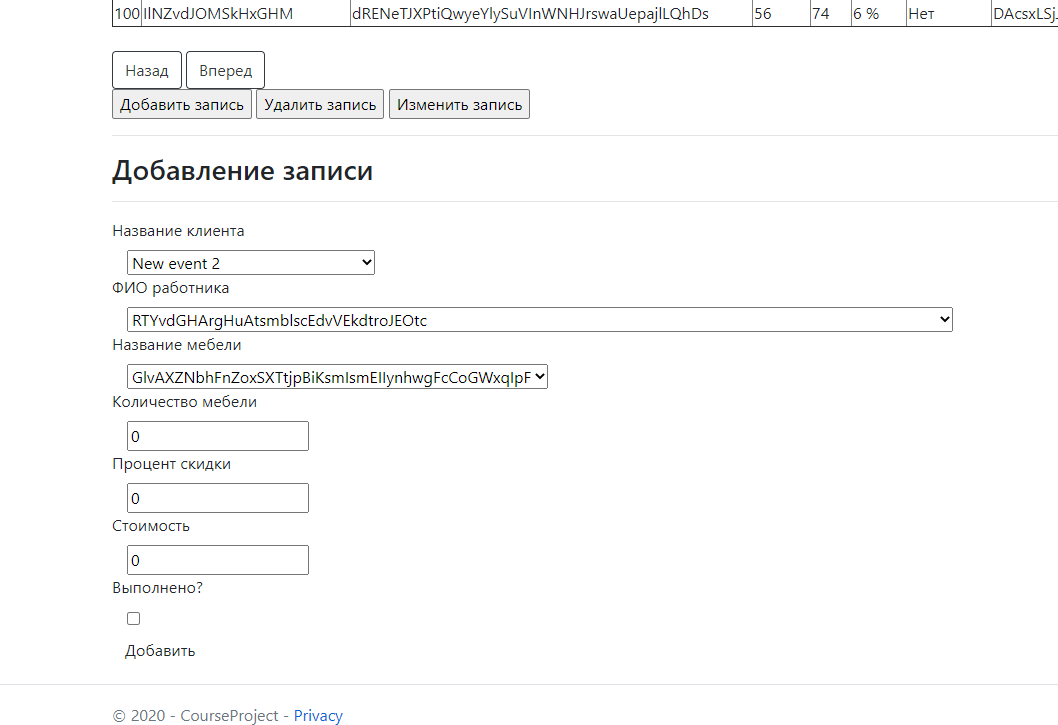
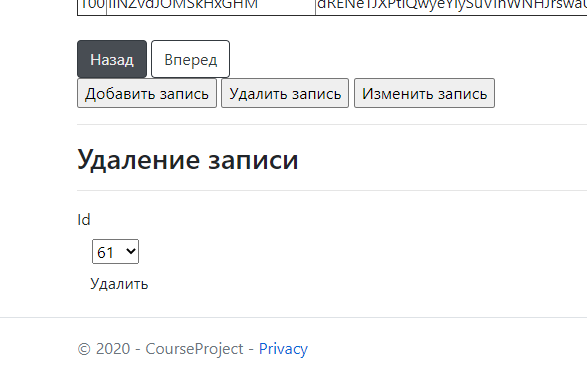


Рисунок 11-14 – страница работников с функционалом управления записями для роли «Администратор»







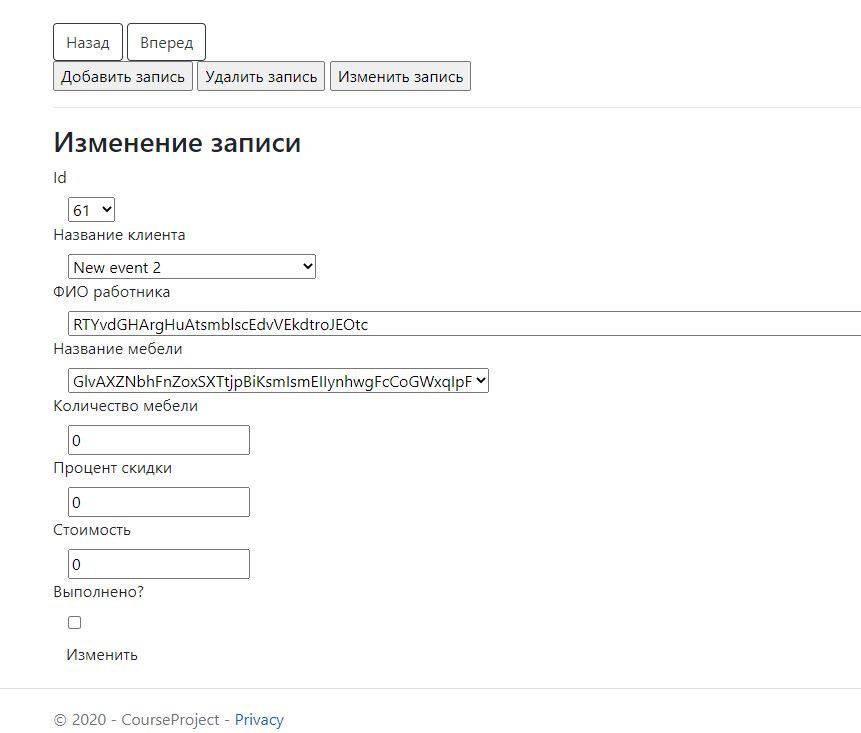


Рисунок 15-18 – страница заказов с функционалом управления записями для ролей «Администратор» и «Работник фабрики». Также отображены кнопки перехода по страницам таблицы

Оставшиеся две страницы имеют такой же функционал, поэтому они здесь не отображены.

Листинг кода для сохранения состояний форм фильтрации в *Session* и для сохранения данных таблиц в кэше:

В методе ConfigureServices добавляем поддержку интерфейса IMemoryCache():

public void ConfigureServices(IServiceCollection services)

{

// ...

services.AddMemoryCache();

}

В каждом контроллере в метод *Index()* добавлен код получения записей из *Session* для их использования в представлении. Если такие данные отсутствуют, то данные берутся из контекста, а затем они добавляются в *Session*. В остальных методах после успешного изменения контекста данных записи в *Session* перезаписываютсяются. Код для каждого контроллера:

*ClientController.cs:*

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public ClientController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string filterRepresFIO = "Все", string address = null)

{

// ...

List<Client> clients;

if (!cache.TryGetValue("Clients", out clients))

{

clients = db.Clients.ToList();

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

// ...

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult AddClient(ClientIndexViewModel model)

{

// ...

cache.Remove("Clients");

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Client");

// ...

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult DeleteClient(ClientIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Clients");

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Client");

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult UpdateClient(ClientIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Clients");

cache.Set("Clients", db.Clients.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Client");

}

}

*EmployeeController.cs:*

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public EmployeeController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string position = "Все", string education = null)

{

// ...

List<Employee> employees;

if (!cache.TryGetValue("Employees", out employees))

{

employees = db.Employees.ToList();

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

// ...

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult AddEmployee(EmployeeIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Employees");

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Employee");

}

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult DeleteEmployee(EmployeeIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Employees");

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Employee");

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult UpdateEmployee(EmployeeIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Employees");

cache.Set("Employees", db.Employees.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Employee");

}

}

*FurnitureController.cs:*

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public FurnitureController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

[ResponseCache(CacheProfileName = "TablesCaching")]

public IActionResult Index(int page = 1, string material = "Все", decimal price = 0)

{

// ...

List<Furniture> furniture;

if (!cache.TryGetValue("Furniture", out furniture))

{

furniture = db.Furniture.ToList();

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

// ...

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult AddFurniture(FurnitureIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Furniture");

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Furniture");

}

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult DeleteFurniture(FurnitureIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Furniture");

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Furniture");

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult UpdateFurniture(FurnitureIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Furniture");

cache.Set("Furniture", db.Furniture.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

return RedirectToAction("Index", "Furniture");

}

}

*OrderController.cs:*

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public OrderController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult AddOrder(OrderIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Orders");

return RedirectToAction("Index", "Order");

}

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult DeleteOrder(OrderIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Orders");

return RedirectToAction("Index", "Order");

}

[Authorize(Roles = "Администратор, Работник фабрики")]

[HttpPost]

public IActionResult UpdateOrder(OrderIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Orders");

return RedirectToAction("Index", "Order");

}

}

private OrderIndexViewModel GetViewModel(int page = 1, string furnitureName = "Все", string clientName = "Все")

{

// ...

List<OrderViewModel> orderViewModels;

if (!cache.TryGetValue("Orders", out orderViewModels))

{

List<Order> orders = db.Orders.ToList();

orderViewModels = new List<OrderViewModel>();

foreach (var order in orders)

{

orderViewModels.Add(new OrderViewModel()

{

Id = order.Id,

ClientName = clients.Where(item => item.Id == order.ClientId).First().Name,

DiscountPercent = order.DiscountPercent,

EmployeeFIO = employees.Where(item => item.Id == order.EmployeeId).First().FIO,

FurnitureCount = order.FurnitureCount,

FurnitureName = furniture.Where(item => item.Id == order.FurnitureId).First().Name,

IsCompleted = order.IsCompleted != 0,

Price = order.Price

});

}

cache.Set("Orders", db.Orders.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

// ...

}

*WaybillController.cs:*

// Объект контекста данных

private readonly ApplicationContext db;

private IMemoryCache cache;

public WaybillController(ApplicationContext applicationContext, IMemoryCache cache)

{

db = applicationContext;

this.cache = cache;

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult AddWaybill(WaybillIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Waybills");

return RedirectToAction("Index", "Waybill");

}

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult DeleteWaybill(WaybillIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Waybills");

return RedirectToAction("Index", "Waybill");

}

[Authorize(Roles = "Администратор")]

[HttpPost]

public IActionResult UpdateWaybill(WaybillIndexViewModel model)

{

// ...

db.SaveChanges();

cache.Remove("Waybills");

return RedirectToAction("Index", "Waybill");

}

}

private WaybillIndexViewModel GetViewModel(int page = 1, string providerName = null, string furnitName = "Все")

{

// ...

List<WaybillViewModel> waybillViewModels;

if (!cache.TryGetValue("Waybills", out waybillViewModels))

{

List<Waybill> waybills = db.Waybills.ToList();

waybillViewModels = new List<WaybillViewModel>();

foreach (var waybill in waybills)

{

waybillViewModels.Add(new WaybillViewModel()

{

Id = waybill.Id,

ProviderId = waybill.ProviderId,

ProviderName = waybill.ProviderName,

EmployeeFIO = employees.Where(item => item.Id == waybill.EmployeeId).First().FIO,

DateOfSupply = waybill.DateOfSupply,

FurnitureName = furniture.Where(item => item.Id == waybill.FurnitureId).First().Name,

Material = waybill.Material,

Price = waybill.Price,

Weight = waybill.Weight

});

}

cache.Set("Waybills", db.Waybills.ToList(), new MemoryCacheEntryOptions().SetAbsoluteExpiration(TimeSpan.FromMinutes(10)));

}

// ...

}

Так как накладные и заказы являются таблицами на стороне отношений «многие», то создание данных для занесения в кэш является медлительным процессом. Поэтому производится только удаление, а занесение данных ложится на метод создания модели.

Для сохранения состояния форм фильтрации используется сессия. Сохранение и вставка производится с помощью скриптов JavaScript, так как они позволяют сократить объем кода, а также предоставляют возможность динамического сохранения при обновлении или закрытии страницы:

*Client/Index.cshtml:*

<script>

function SetSession() {

sessionStorage.setItem("address", $("#value1").val());

sessionStorage.setItem("filterRepresFIO", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("address"));

$("#value2").val(sessionStorage.getItem("filterRepresFIO"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<form method="get">

<div>

<label class="control-label">Адрес:&nbsp;</label>

<input name="address" id="value1" class="form-control" />

</div>

<div>

<label class="control-label">ФИО представителя:&nbsp;</label>

<select name="filterRepresFIO" id="value2" class="form-control">

@foreach (var elem in Model.FilterRepresFIOs)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

/...

*Employee/Index.cshtml:*

// ...

function SetSession() {

sessionStorage.setItem("education", $("#value1").val());

sessionStorage.setItem("position", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("education"));

$("#value2").val(sessionStorage.getItem("position"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Образование:&nbsp;</label>

<input name="education" id="value1" class="form-control" />

</div>

<div>

<label class="control-label">Должностm:&nbsp;</label>

<select name="position" id="value2" class="form-control">

@foreach (var elem in Model.FilterPositions)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

// ...

*Furniture/Index.cshtml:*

*// …*

function SetSession() {

sessionStorage.setItem("price", $("#value1").val());

sessionStorage.setItem("material", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("price"));

$("#value2").val(sessionStorage.getItem("material"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Стоимость:&nbsp;</label>

<input name="price" type="number" min="0" id="value1" class="form-control" />

</div>

<div>

<label class="control-label">Материалы:&nbsp;</label>

<select name="material" id="value2" class="form-control">

@foreach (var elem in Model.FilterMaterials)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

*// ...*

*Order/Index.cshtml:*

*// …*

function SetSession() {

sessionStorage.setItem("clientName", $("#value1 option:selected").val());

sessionStorage.setItem("furniture", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("clientName"));

$("#value2").val(sessionStorage.getItem("furniture"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Название клиента:&nbsp;</label>

<select name="clientName" id="value1" class="form-control">

@foreach (var elem in Model.FilterClients)

{

<option>@elem</option>

}

</select>

</div>

<div>

<label class="control-label">Название мебели:&nbsp;</label>

<select name="furniture" id="value2" class="form-control">

@foreach (var elem in Model.FilterFurnitures)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

// ...

*Waybill/Index.cshtml:*

*// …*

function SetSession() {

sessionStorage.setItem("providerName", $("#value1").val());

sessionStorage.setItem("furnitName", $("#value2 option:selected").val());

}

function GetSession() {

$("#value1").val(sessionStorage.getItem("providerName"));

$("#value2").val(sessionStorage.getItem("furnitName"));

}

</script>

<body onbeforeunload="SetSession()" onload="GetSession()">

<H1>@ViewBag.Title:</H1>

<form method="get">

<div>

<label class="control-label">Название поставщика:&nbsp;</label>

<input type="text" name="providerName" id="value1" class="form-control"/>

</div>

<div>

<label class="control-label">Название мебели:&nbsp;</label>

<select name="furnitName" id="value2" class="form-control">

@foreach (var elem in Model.FilterFurnitureNames)

{

<option>@elem</option>

}

</select>

</div>

<br />

<div>

<input type="submit" style="border-color:black;" value="Фильтр" class="btn btn-default" />

</div>

</form>

*// …*

**Вывод:** в результате выполнения работы было изучено создание типовых web-приложений для работы с информацией из реляционных баз данных.