**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ 5**

по дисциплине «Разработка приложений баз данных для информационных систем»

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

Вариант 4

Выполнила: студентка гр. ИТП-31

Довгая Е. К.

Принял: доцент

Асенчик О. Д.

Гомель 2020

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

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

Web- приложение должно удовлетворять следующим требованиям:

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

***На оценку 9 или 10:***

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

**Ход выполнения:**

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

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

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

Далее в проект был добавлен *Identity.* Он необходим для авторизации и аутентификации пользователей.

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

Для всех контроллеров были созданы папки в папке *Views*. В каждом контроллере есть методы *Index, Create, Edit, Delete, Details*. И поэтому в каждой папке с представлениями были созданы соответствующие представления для этих методов.

На гитхабе был создан репозиторий для проекта. Ссылка https://github.com/AlyonaDovgaya/DDBAIS\_Lab5

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

**ПРИЛОЖЕНИЕ А**

Листинг созданных классов

Листинг класса *Startup.cs*

using Lab5.Data;

using Lab5.Middleware;

using Lab5.Models;

using Lab5.Services;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Lab5

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

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

public void ConfigureServices(IServiceCollection services)

{

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

IServiceCollection serviceCollections = services.AddDbContext<Lab5Context>(options => options.UseSqlServer(connectionString));

string sqlConnectionIdentityString = Configuration.GetConnectionString("SqlServerIdentity");

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

services.AddIdentity<User, IdentityRole>()

.AddEntityFrameworkStores<ApplicationContext>();

services.AddTransient<CacheProvider>();

services.AddMemoryCache();

services.AddDistributedMemoryCache();

services.AddSession();

services.AddControllersWithViews(options =>

{

options.CacheProfiles.Add("CacheProfile",

new CacheProfile()

{

Duration = 262

});

});

}

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

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

}

else

{

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

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

app.UseHsts();

}

app.UseSession();

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseRoleInitializer();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

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

});

}

}

}

Листинг класса *AccountController.cs*

using Lab5.Models;

using Lab5.ViewModels.Account;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Mvc;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Lab5.Controllers

{

public class AccountController : Controller

{

private readonly UserManager<User> userManager;

private readonly SignInManager<User> signInManager;

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

{

this.userManager = userManager;

this.signInManager = signInManager;

}

[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

};

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

if (result.Succeeded)

{

//await userManager.AddToRoleAsync(user, "user");

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)

{

LoginViewModel model = new LoginViewModel

{

ReturnUrl = returnUrl

};

return View(model);

}

[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)

{

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

{

return Redirect(model.ReturnUrl);

}

else

{

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

}

}

else

{

ModelState.AddModelError(string.Empty, "Inccorect login or password.");

}

}

return View(model);

}

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Logout()

{

await signInManager.SignOutAsync();

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

}

}

}

Листинг класса *RoleInitializerMiddleware*

using Lab5.Data;

using Lab5.Models;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

using Microsoft.AspNetCore.Identity;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace Lab5.Middleware

{

public class RoleInitializerMiddleware

{

private readonly RequestDelegate \_next;

public RoleInitializerMiddleware(RequestDelegate next)

{

\_next = next;

}

public async Task InvokeAsync(HttpContext context, IServiceProvider serviceProvider, UserManager<User> userManager, RoleManager<IdentityRole> roleManager)

{

await RoleInitializer.InitializeAsync(userManager, roleManager);

await \_next(context);

}

}

public static class DbInitializerExtensions

{

public static IApplicationBuilder UseRoleInitializer(this IApplicationBuilder builder)

{

return builder.UseMiddleware<RoleInitializerMiddleware>();

}

}

}