**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

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

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

**ЛАБОРАТОРНАЯ РАБОТА №4**

по дисциплине: «Разработка приложений баз данных для информационных систем»

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

Выполнил: студент гр. ИТП-31

Бондарев Е.Ю.

Принял: ректор

Асенчик О.Д.

Гомель 2023

**Цель работы:** ознакомиться с возможностями *ASP.NET Core MVC* и *Entity Framework Core* для разработки слоя доступа к данным, хранящимся в базе данных, и обработки запросов пользователя посредством контроллеров.

**Задание:**

Создать с использованием *ASP.NET Core MVC Web*-приложение, содержащее набор классов, моделирующих предметную область, и осуществляющих генерацию и заполнение тестовыми наборами записей базу данных. Разработать один компонент *middleware*, контроллеры и представления для выборки и отображения информации из не менее чем 3- таблиц базы данных с использованием механизма внедрение зависимостей.

Для выполнения задания необходимо создать:

• Классы, моделирующие не менее чем три таблицы базы данных согласно вашему варианту. Перечень таблиц предварительно согласовывается с преподавателем. Одна из таблиц обязательно должна находиться на стороне отношения «многие» связи с другой таблицей в схеме базы данных.

• Класс контекста данных.

• Другие классы, например, классы *View Model* и т.п. (при необходимости).

• Компонент *middleware*, вызываемый в классе *Startup*, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.

• Классы контроллеров (по одному на каждую таблицу базы данных) для обработки обращений пользователя, выборки данных из таблиц и вызова соответствующих представлений для отображения выбранных данных.

• Разработать представления для отображения данных из таблиц, выбранных контроллерами. Представления, работающими с таблицами, стоящими на стороне отношения «многие» в схеме базы данных, должны выводить вместо кодов внешних ключей смысловые значения из связанных таблиц, стоящих на стороне отношения «один».

• Используя предварительно созданный и сконфигурированный в классе *Startup* профиль кэширования, подключить кэширование вывода для страниц с использованием атрибута *ResponseCache* для соответствующих методов контроллера. Данные в кэше хранить неизменными в течение 2\**N*+240 секунд, где *N* – номер вашего варианта.

• С использованием средств разработчика браузера (*Chrome, Firefox*) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием атрибута *ResponseCache*.

Для проверки преподавателем следует разместить разработанный проект на *GitHub*.

**Ход работы**

В процессе выполнения лабораторной работы с использованием *Entity Framework* были перенесены три взаимозвязанные модели. Информация о подключении к базе данных хранится в файле *appsettings.json*, а класс контекста был интегрирован в приложение с применением внедрения зависимостей (*DI*).

Далее были разработаны контроллеры, которые используются для передачи данных, которые хранятся в моделях в представления. Были разработаны три контроллера *EmployeeController, EmployeePositionController, OfficeController.* Далее были разработаны представления для отображения данных из котроллеров.

Пример представления с информаций о должностях работников указана на рисунке 1.

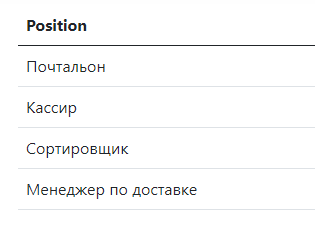


Рисунок 1 – Пример страницы с информаций о должностях работников

Пример представления с информаций о работниках указан на рисунке 1.



Рисунок 2 – Пример страницы с информацией о работниках

Пример представления с информаций о офисах указан на рисунке 3.

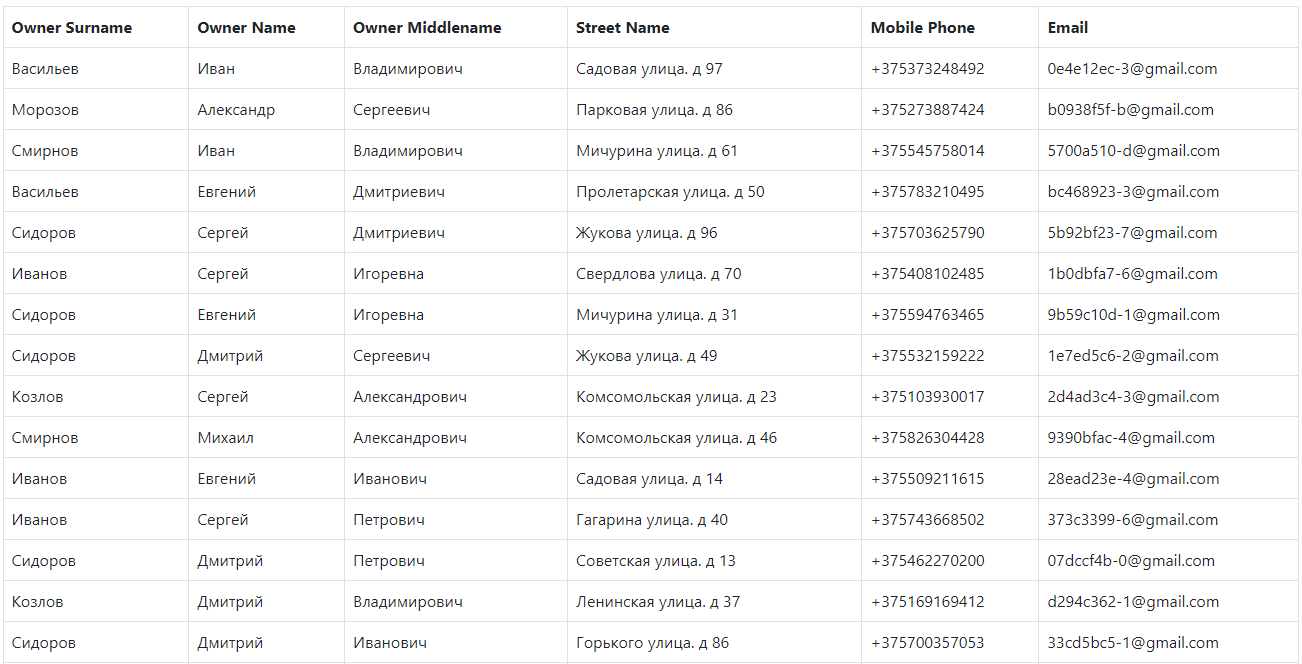
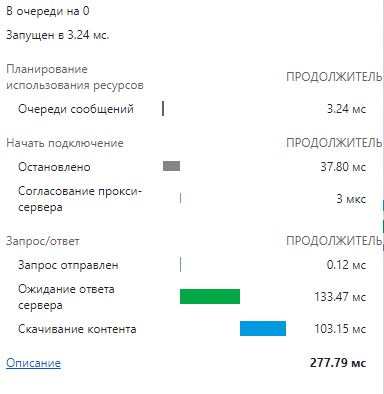


Рисунок 3 – Пример страницы с информацией о офисах

Далее для более быстрого доступа ко вкладкам сайта при помощи класса *ResponseCache* было реализовано кэширование запросов. В классе *Program* был добавлен новый профиль кэш. А в контролерах был указан атрибут, в котором указывается имя этого профиля. Пример запроса до кэширования указан на рисунке 4.



Пример до кэширования указан на рисунке 4.

Пример запроса после кэширования указан на рисунке 5.

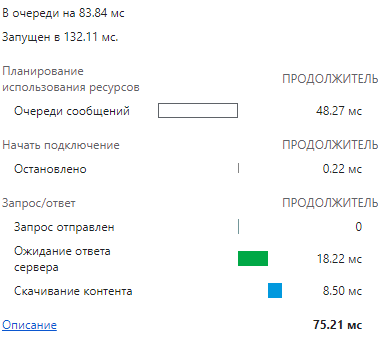


Рисунок 5 – Пример кэшированного запроса

После выполнения лабораторной работы созданные проект был добавлен в локальный *git* репозиторий а потом перенесен в *GitHub* репозиторий своего аккаунта. Чтобы ознакомится с созданным проектом можно по ссылке https://github.com/EvgeniBondarev/DDBAISE/tree/main/Laba4*.*

**Вывод**: в ходе выполнения лабораторной работы была освоена технология *ASP .NET Core MVC*. Были созданы модельные и контекстные классы для обработки данных, контроллеры для связи моделей с представлениями, и представления для отображения результата. Кроме того, мы изучили класс *ResponseCache*, который позволяет кэшировать методы контроллеров для более быстрого доступа к данным.

**ПРИЛОЖНИЕ А**

Листинг класса *Program*

using Laba4.Middleware;

using Laba4.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);

IServiceCollection services = builder.Services;

string? connection = builder.Configuration.GetConnectionString("SqlServerConnection");

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

builder.Services.AddControllersWithViews();

builder.Services.AddDistributedMemoryCache();

builder.Services.AddSession();

builder.Services.AddControllersWithViews(options => {

options.CacheProfiles.Add("DbTableCache",

new CacheProfile()

{

Location = ResponseCacheLocation.Any,

Duration = 2 \* 2 + 240

});

});

var app = builder.Build();

app.UseSession();

app.UseDbInitializerMiddleware();

if (!app.Environment.IsDevelopment())

{

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

}

app.UseStaticFiles();

app.UseRouting();

app.UseAuthorization();

app.MapControllerRoute(

name: "default",

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

app.MapControllerRoute(

name: "Employee Position",

pattern: "{controller=EmployeePosition}/{action=ShowEmployeePosition}");

app.MapControllerRoute(

name: "Employee",

pattern: "{controller=Employee}/{action=ShowEmployee}");

app.MapControllerRoute(

name: "Office",

pattern: "{controller=Office}/{action=ShowOffice}");

app.Run();

Листинг класса *EmployeeController*

using Laba4.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace Laba4.Controllers

{

public class EmployeeController : Controller

{

private readonly SubsCityContext \_context;

public EmployeeController(SubsCityContext subsCityContext)

{

\_context = subsCityContext;

}

[ResponseCache(CacheProfileName = "DbTableCache")]

public IActionResult ShowEmployee()

{

return View(\_context.Employees.Include(p => p.Position).Include(o => o.Office));

}

}

}

Листинг класса *EmployeePositionController*

using Laba4.Models;

using Microsoft.AspNetCore.Mvc;

namespace Laba4.Controllers

{

public class EmployeePositionController : Controller

{

private readonly SubsCityContext \_context;

public EmployeePositionController(SubsCityContext context)

{

\_context = context;

}

[ResponseCache(CacheProfileName = "DbTableCache")]

public IActionResult ShowEmployeePosition()

{

return View(\_context.EmployeePositions);

}

}

}

Листинг класса *OfficeController*

using Laba4.Models;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace Laba4.Controllers

{

public class OfficeController : Controller

{

private readonly SubsCityContext \_context;

public OfficeController(SubsCityContext context)

{

\_context = context;

}

[ResponseCache(CacheProfileName = "DbTableCache")]

public IActionResult ShowOffice()

{

return View(\_context.Offices);

}

}

}

Листинг представления *ShowEmployee*

@{

ViewData["Title"] = "Employee";

}

@model IEnumerable<Employee>

<table class="table table-bordered">

<thead>

<tr>

<th>Surname</th>

<th>Name</th>

<th>Middlename</th>

<th>Position</th>

<th>Office</th>

</tr>

</thead>

<tbody>

@foreach (var employee in Model)

{

<tr>

<td>@employee.Surname</td>

<td>@employee.Name</td>

<td>@employee.Middlename</td>

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

<td>@employee.Office.StreetName</td>

</tr>

}

</tbody>

</table>

Листинг представления *ShowEmployeePosition*

@{

ViewData["Title"] = "Employee";

}

@model IEnumerable<Employee>

<table class="table table-bordered">

<thead>

<tr>

<th>Surname</th>

<th>Name</th>

<th>Middlename</th>

<th>Position</th>

<th>Office</th>

</tr>

</thead>

<tbody>

@foreach (var employee in Model)

{

<tr>

<td>@employee.Surname</td>

<td>@employee.Name</td>

<td>@employee.Middlename</td>

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

<td>@employee.Office.StreetName</td>

</tr>

}

</tbody>

</table>

Листинг представления *ShowOffice*

@{

ViewData["Title"] = "Employee";

}

@model IEnumerable<Employee>

<table class="table table-bordered">

<thead>

<tr>

<th>Surname</th>

<th>Name</th>

<th>Middlename</th>

<th>Position</th>

<th>Office</th>

</tr>

</thead>

<tbody>

@foreach (var employee in Model)

{

<tr>

<td>@employee.Surname</td>

<td>@employee.Name</td>

<td>@employee.Middlename</td>

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

<td>@employee.Office.StreetName</td>

</tr>

}

</tbody>

</table>