**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 4**

по дисциплине «Разработка приложений баз данных для информационных систем»

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

Вариант 1

Выполнил: студент гр. ИТП-31

Аскерко Ю.С.

Принял: доцент

Асенчик О. Д.

Гомель 2020

**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 этого необходимо создать:

1. 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.
2. Класс контекста данных.
3. Другие классы, например, классы View Model и т.п. (при необходимости).
4. Компонент middleware, вызываемый в классе Startup, для инициализации базы данных путем заполнения ее таблиц тестовым набором записей.
5. 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.
6. 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».
7. 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.
8. С использованием средств разработчика браузера (Chrome, Firefox) продемонстрировать ускорение обработки запроса при наличии кэширования с использованием атрибута ResponseCache.

**Вариант задания:**

Предметная область задания изображена на рисунке 1.

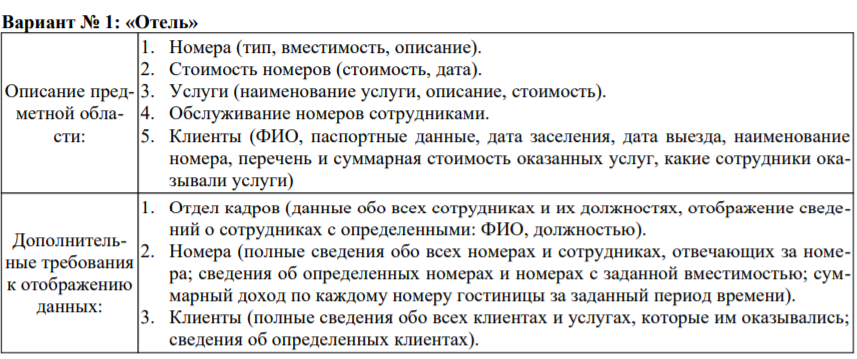


Рисунок 1 – Предметная область

**Ход работы:**

Результаты работы контроллеров приведены на рисунках 2 - 4.

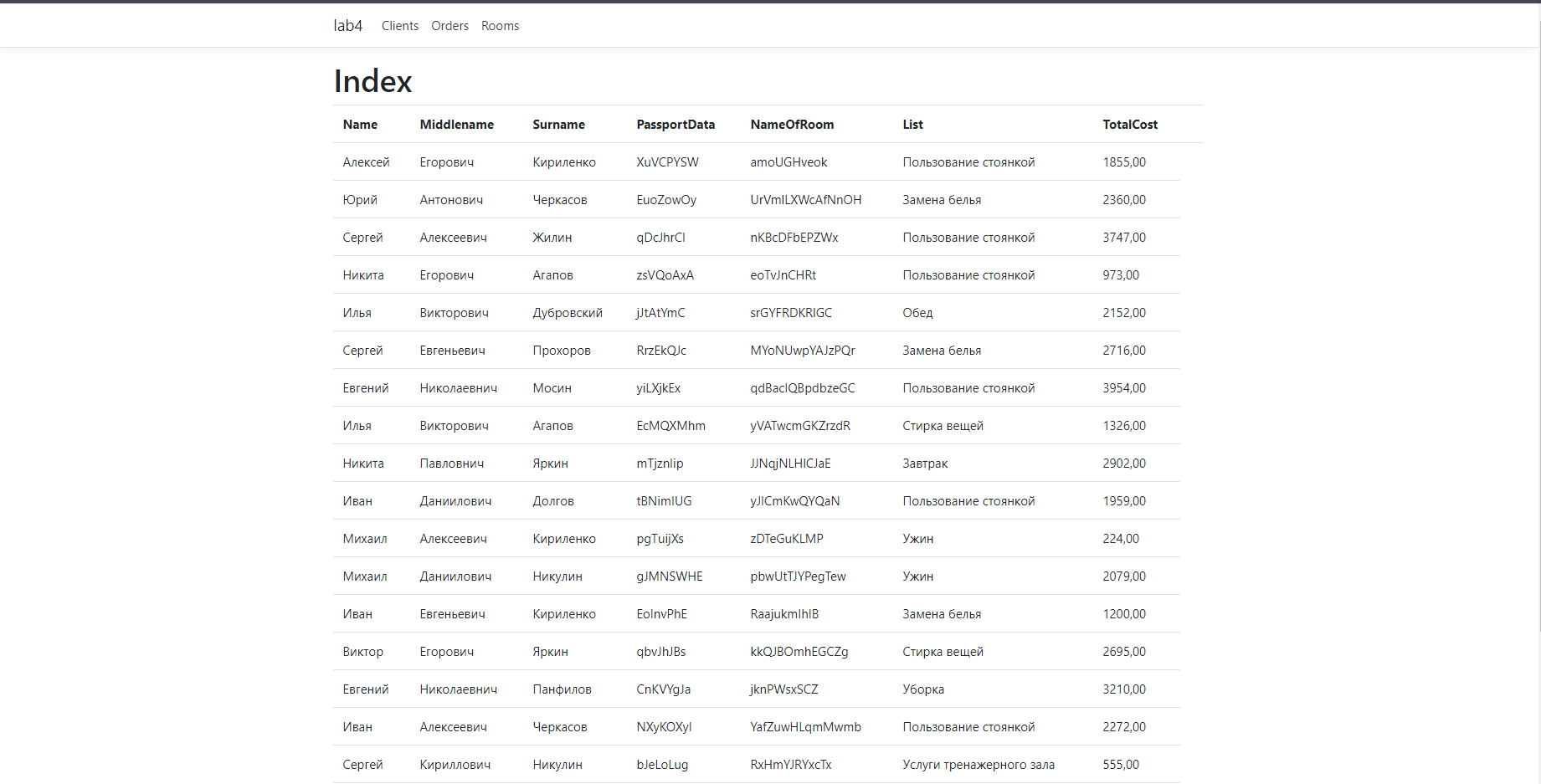


Рисунок 2 – результат работы контроллера ClientsController

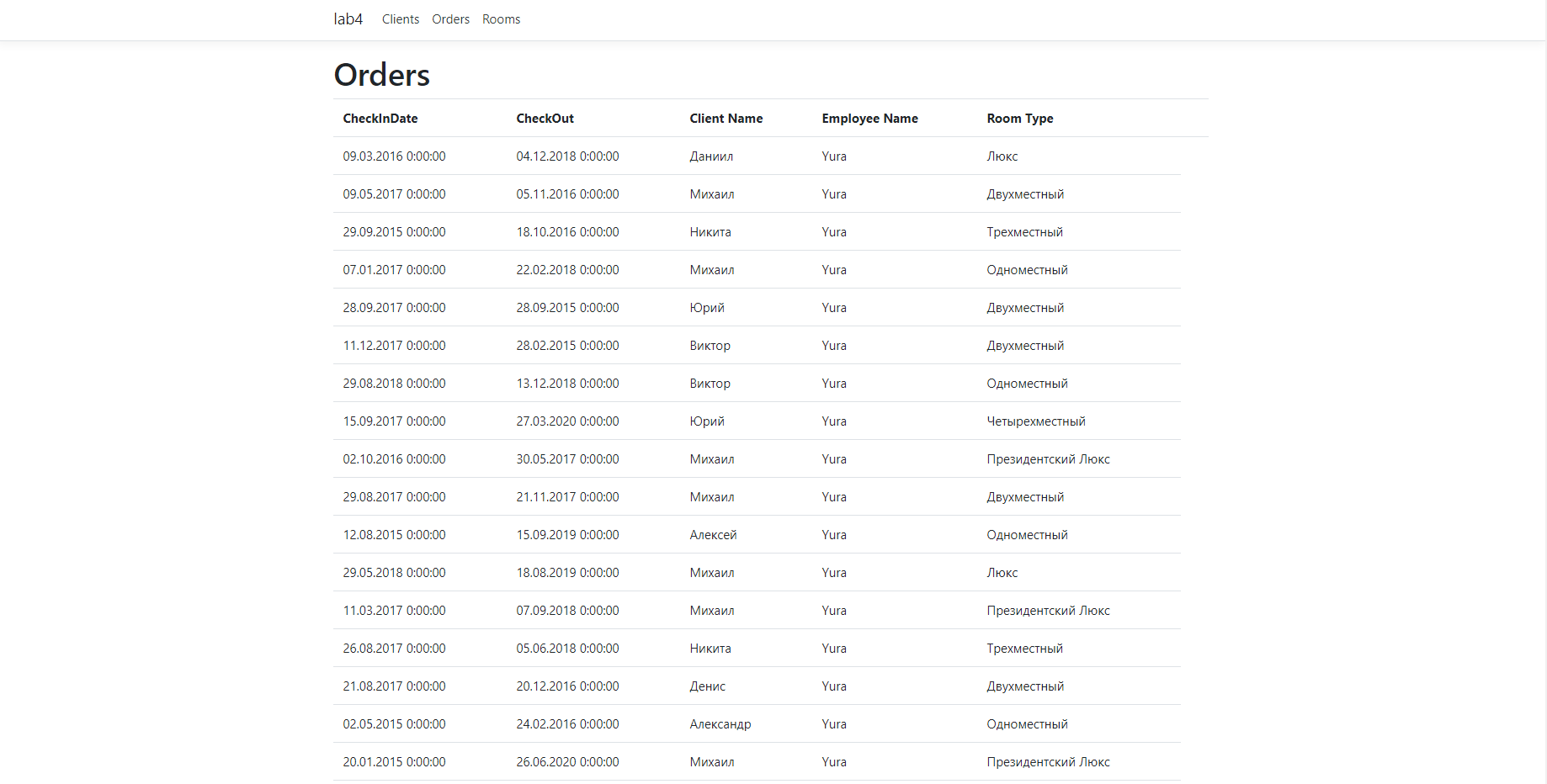


Рисунок 3 – результат работы контроллера OrdersController

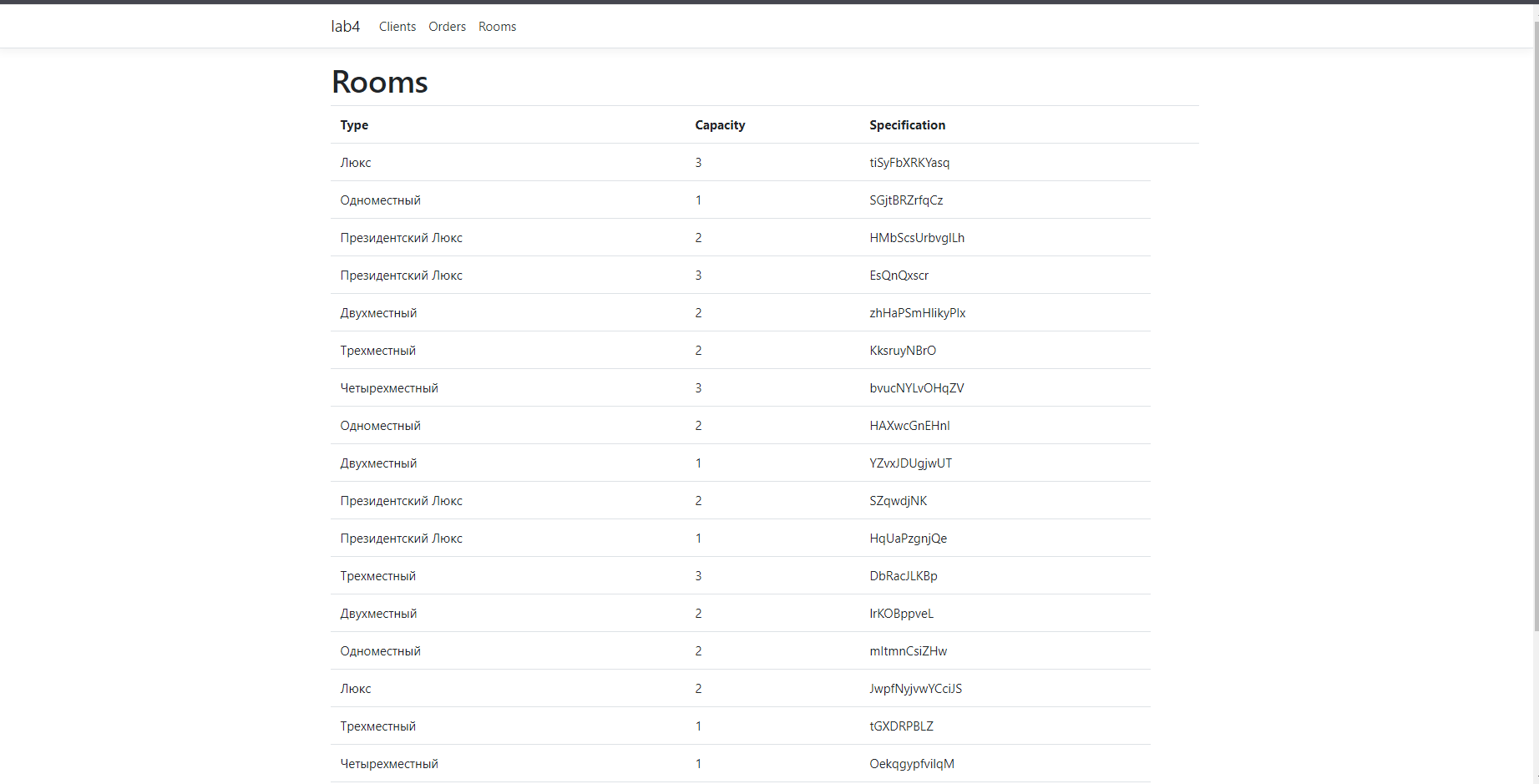


Рисунок 4 – результат работы контроллера RoomsController

**Вывод:**

Были созданы различные формы для взаимодействия с таблицами базы данных, а также для изучения различных видов кэширования.

**ПРИЛОЖЕНИЕ А**

Листинг класса с запросами

Строка подключения из конфигурационного файла:

"ConnectionStrings": {

"SqlServer": "Server=DESKTOP-9QN6I91\\SQLEXPRESS;Database=Hotel;Trusted\_Connection=True;"

*Startup.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab4.Data;

using lab4.Middleware;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Hosting;

using Microsoft.AspNetCore.HttpsPolicy;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

namespace lab4

{

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("SqlServer");

services.AddDbContext<HotelContext>(options => options.UseSqlServer(connectionString));

services.AddDistributedMemoryCache();

services.AddSession();

services.AddControllersWithViews(options =>

{

options.CacheProfiles.Add("CacheProfile",

new CacheProfile()

{

Duration = 242

});

});

}

// 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.UseHttpsRedirection();

app.UseStaticFiles();

app.UseSession();

app.UseDbInitializerMiddleware();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

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

});

}

}

}

*InitializeDb.cs*

using lab4.Models;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

namespace lab4.Data

{

public static class InitializeDB

{

private static Random random = new Random();

private static string[] names =

{

"Никита", "Евгений", "Михаил", "Даниил", "Юрий", "Илья", "Алексей", "Сергей", "Виталий",

"Денис", "Михаил", "Артем", "Иван", "Виктор", "Александр"

};

private static string[] surnames =

{

"Кириленко", "Панфилов", "Дубровский", "Долгов", "Яркин", "Черкасов", "Рябцев", "Жилин", "Фокин",

"Мосин", "Агапов", "Якубовский", "Никулин", "Котов", "Прохоров"

};

private static string[] middleNames =

{

"Алексеевич", "Дмитриевич", "Владимирович", "Евгеньевич", "Артёмович", "Павловнич", "Николаевнич",

"Антонович", "Викторович", "Кириллович", "Андреевич", "Геннадьевич", "Даниилович", "Олегович", "Егорович"

};

private static string[] lists =

{

"Уборка", "Замена белья", "Стирка вещей", "Доставка", "Обед", "Ужин", "Завтрак", "Пользование стоянкой", "Услуги тренажерного зала"

};

private static string[] types =

{

"Одноместный", "Двухместный", "Трехместный", "Четырехместный", "Люкс", "Президентский Люкс"

};

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

private static string GetRandomEl(string[] arr)

{

return arr[random.Next(0, arr.Length)];

}

private static DateTime NextDateTime()

{

DateTime start = new DateTime(2015, 1, 1);

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

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

}

private static string GetString(int minStringLength, int maxStringLength)

{

string result = "";

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

int stringPosition;

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

{

stringPosition = random.Next(letters.Length);

result += letters[stringPosition];

}

return result;

}

public static void Initialize(HotelContext db)

{

db.Database.EnsureCreated();

int rowCount;

int rowIndex;

if (!db.Clients.Any())

{

rowCount = 2000;

rowIndex = 0;

while (rowIndex < rowCount)

{

Client client = new Client

{

Name = GetRandomEl(names),

Middlename = GetRandomEl(middleNames),

Surname = GetRandomEl(surnames),

PassportData = GetString(8, 8),

NameOfRoom = GetString(10, 16),

List = GetRandomEl(lists),

TotalCost = (decimal)random.Next(90, 4000)

};

db.Clients.Add(client);

rowIndex++;

}

db.SaveChanges();

}

if (!db.Rooms.Any())

{

rowCount = 2000;

rowIndex = 0;

while (rowIndex < rowCount)

{

Room room = new Room

{

Type = GetRandomEl(types),

Capacity = random.Next(1, 4),

Specification = GetString(8, 16)

};

db.Rooms.Add(room);

rowIndex++;

}

db.SaveChanges();

}

if (!db.Orders.Any())

{

rowIndex = 0;

rowCount = 2000;

int minClientId = db.Clients.Min(x => x.Id);

int maxClientId = db.Clients.Max(x => x.Id);

int minEmployeeId = db.Employees.Min(x => x.Id);

int maxEmployeeId = db.Employees.Max(x => x.Id);

int minRoomId = db.Rooms.Min(x => x.Id);

int maxRoomId = db.Rooms.Max(x => x.Id);

while (rowIndex < rowCount)

{

Order order = new Order

{

CheckInDate = NextDateTime(),

CheckOut = NextDateTime(),

EmployeeId = random.Next(minEmployeeId, maxEmployeeId + 1),

ClientId = random.Next(minClientId, maxClientId + 1),

RoomId = random.Next(minRoomId, maxRoomId + 1)

};

db.Orders.Add(order);

rowIndex++;

}

db.SaveChanges();

}

}

}

}

*DbInitializeMiddleware.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab4.Data;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Http;

namespace lab4.Middleware

{

// You may need to install the Microsoft.AspNetCore.Http.Abstractions package into your project

public class DbInitializerMiddleware

{

private readonly RequestDelegate \_next;

public DbInitializerMiddleware(RequestDelegate next)

{

\_next = next;

}

public Task Invoke(HttpContext httpContext, HotelContext db)

{

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

{

InitializeDB.Initialize(db);

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

}

return \_next.Invoke(httpContext);

}

}

// Extension method used to add the middleware to the HTTP request pipeline.

public static class DbInitializerMiddlewareExtensions

{

public static IApplicationBuilder UseDbInitializerMiddleware(this IApplicationBuilder builder)

{

return builder.UseMiddleware<DbInitializerMiddleware>();

}

}

}

*ClientsController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab4.Data;

using Microsoft.AspNetCore.Mvc;

namespace lab4.Controllers

{

public class ClientsController : Controller

{

HotelContext \_context = new HotelContext();

public ClientsController(HotelContext context)

{

\_context = context;

}

public IActionResult Index()

{

return View();

}

public IActionResult Clients()

{

return View(\_context.Clients.Take(20).ToList());

}

}

}

*OrdersController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab4.Data;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

namespace lab4.Controllers

{

public class OrdersController : Controller

{

HotelContext \_context = new HotelContext();

public OrdersController(HotelContext context)

{

\_context = context;

}

public IActionResult Index()

{

return View();

}

public IActionResult Orders()

{

var orders = \_context.Orders.Include(x => x.Client).Include(x => x.Employee).Include(x => x.Room).Take(20);

return View(orders.ToList());

}

}

}

*RoomsController.cs*

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using lab4.Data;

using Microsoft.AspNetCore.Mvc;

namespace lab4.Controllers

{

public class RoomsController : Controller

{

HotelContext \_context = new HotelContext();

public RoomsController(HotelContext context)

{

\_context = context;

}

public IActionResult Index()

{

return View();

}

public IActionResult Rooms()

{

return View(\_context.Rooms.Take(20).ToList());

}

}

}

*Clients.cshtml*

@model IEnumerable<lab4.Models.Client>

@{

ViewData["Title"] = "Clients";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Index</h1>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Name)

</th>

<th>

@Html.DisplayNameFor(model => model.Middlename)

</th>

<th>

@Html.DisplayNameFor(model => model.Surname)

</th>

<th>

@Html.DisplayNameFor(model => model.PassportData)

</th>

<th>

@Html.DisplayNameFor(model => model.NameOfRoom)

</th>

<th>

@Html.DisplayNameFor(model => model.List)

</th>

<th>

@Html.DisplayNameFor(model => model.TotalCost)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Middlename)

</td>

<td>

@Html.DisplayFor(modelItem => item.Surname)

</td>

<td>

@Html.DisplayFor(modelItem => item.PassportData)

</td>

<td>

@Html.DisplayFor(modelItem => item.NameOfRoom)

</td>

<td>

@Html.DisplayFor(modelItem => item.List)

</td>

<td>

@Html.DisplayFor(modelItem => item.TotalCost)

</td>

</tr>

}

</tbody>

</table>

*Orders.cshtml*

@model IEnumerable<lab4.Models.Order>

@{

ViewData["Title"] = "Orders";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Orders</h1>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.CheckInDate)

</th>

<th>

@Html.DisplayNameFor(model => model.CheckOut)

</th>

<th>

Client @Html.DisplayNameFor(model => model.Client.Name)

</th>

<th>

Employee @Html.DisplayNameFor(model => model.Employee.Name)

</th>

<th>

Room @Html.DisplayNameFor(model => model.Room.Type)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.CheckInDate)

</td>

<td>

@Html.DisplayFor(modelItem => item.CheckOut)

</td>

<td>

@Html.DisplayFor(modelItem => item.Client.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Employee.Name)

</td>

<td>

@Html.DisplayFor(modelItem => item.Room.Type)

</td>

</tr>

}

</tbody>

</table>

*Rooms.cshtml*

@model IEnumerable<lab4.Models.Room>

@{

ViewData["Title"] = "Rooms";

Layout = "~/Views/Shared/\_Layout.cshtml";

}

<h1>Rooms</h1>

<table class="table">

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Type)

</th>

<th>

@Html.DisplayNameFor(model => model.Capacity)

</th>

<th>

@Html.DisplayNameFor(model => model.Specification)

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model) {

<tr>

<td>

@Html.DisplayFor(modelItem => item.Type)

</td>

<td>

@Html.DisplayFor(modelItem => item.Capacity)

</td>

<td>

@Html.DisplayFor(modelItem => item.Specification)

</td>

</tr>

}

</tbody>

</table>