Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Российский университет транспорта»

РУТ (МИИТ)

Институт управления и цифровых технологий

Кафедра «Цифровые технологии управления транспортными процессами»

ЛАБОРАТОРНАЯ РАБОТА ПО ДИСЦИПЛИНЕ:

**“** Проектирование хранилищ данных**”**

Выполнил: Веселков К.О.

Группа: УВП-312

Проверил: Заманов Д. А.

Москва 2023 г.

**СОДЕРЖАНИЕ**

[**ЗАДАНИЕ** 3](#_Toc150392242)

[**РЕАЛИЗАЦИЯ** 4](#_Toc150392243)

[**1.1Реализация базы данных MongoDB** 4](#_Toc150392244)

[**1.2 Каскадное удаление в MongoDB** 7](#_Toc150392245)

[**ВЫВОД** 8](#_Toc150392246)

[**ПРИЛОЖЕНИЯ** 9](#_Toc150392247)

[**Приложение А** 9](#_Toc150392248)

[**Приложение В** 11](#_Toc150392249)

# **ЗАДАНИЕ**

Развернуть базу данных MongoDB на своем компьютере или в контейнере.

Написать примитивный API-сервис для работы с базой данных

MongoDB. Сервис должен реализовывать CRUD-операции (Create, Read, Update, Delete) для одной коллекции в базе данных.

Продемонстрировать работу агрегатов в MongoDB. Написать запрос с использованием агрегатов и показать результат работы.

Реализовать каскадное удаление.

**РЕАЛИЗАЦИЯ**

## **1.1Реализация базы данных MongoDB**

Работа была проделана на языке высокого уровня С#

Выбранная тема Магазин котов. Коллекции: Коты, комментарии.

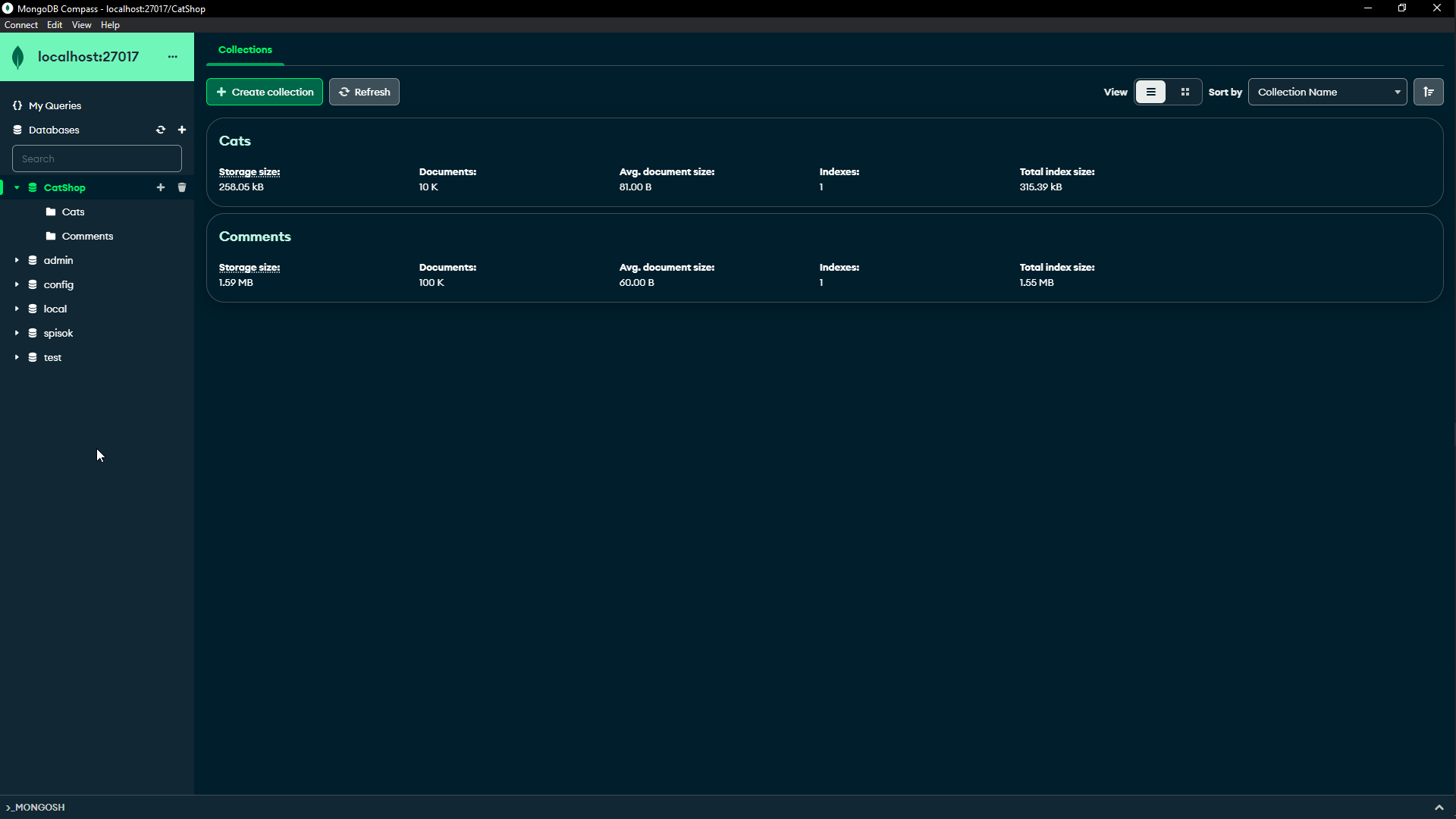


Рисунок 1 – создание базы данных

В коллекцию Cat было добавлено 10000 документов с атрибутами имён, цены, породы и возраста. Для каждой записи из коллекции котов были созданы документы содержащие ObjectID документов из коллекции котов.

for (int i = 0; i < 10000; i++)

{

Cat newCat = new Cat

{

Id = ObjectId.GenerateNewId().ToString(),

name = "Cat" + i,

price = random.Next(50, 200),

breed = "Breed" + random.Next(1, 5),

age = random.Next(1, 10)

};

await catsCollection.InsertOneAsync(newCat);

// Генерация и вставка 1 комментария для каждого кота

Comment newComment = new Comment

{

Id = ObjectId.GenerateNewId().ToString(),

Text = "Comment for " + newCat.name,

CatId = newCat.Id // Устанавливаем CatId комментария равным Id кота

};

await commentsCollection.InsertOneAsync(newComment);

}

Console.WriteLine("Документы успешно добавлены в коллекции Cats и Comments.");

Для удобного отображения и наглядной работы CRUD-операций был использован такой инструмент как Swagger UI.

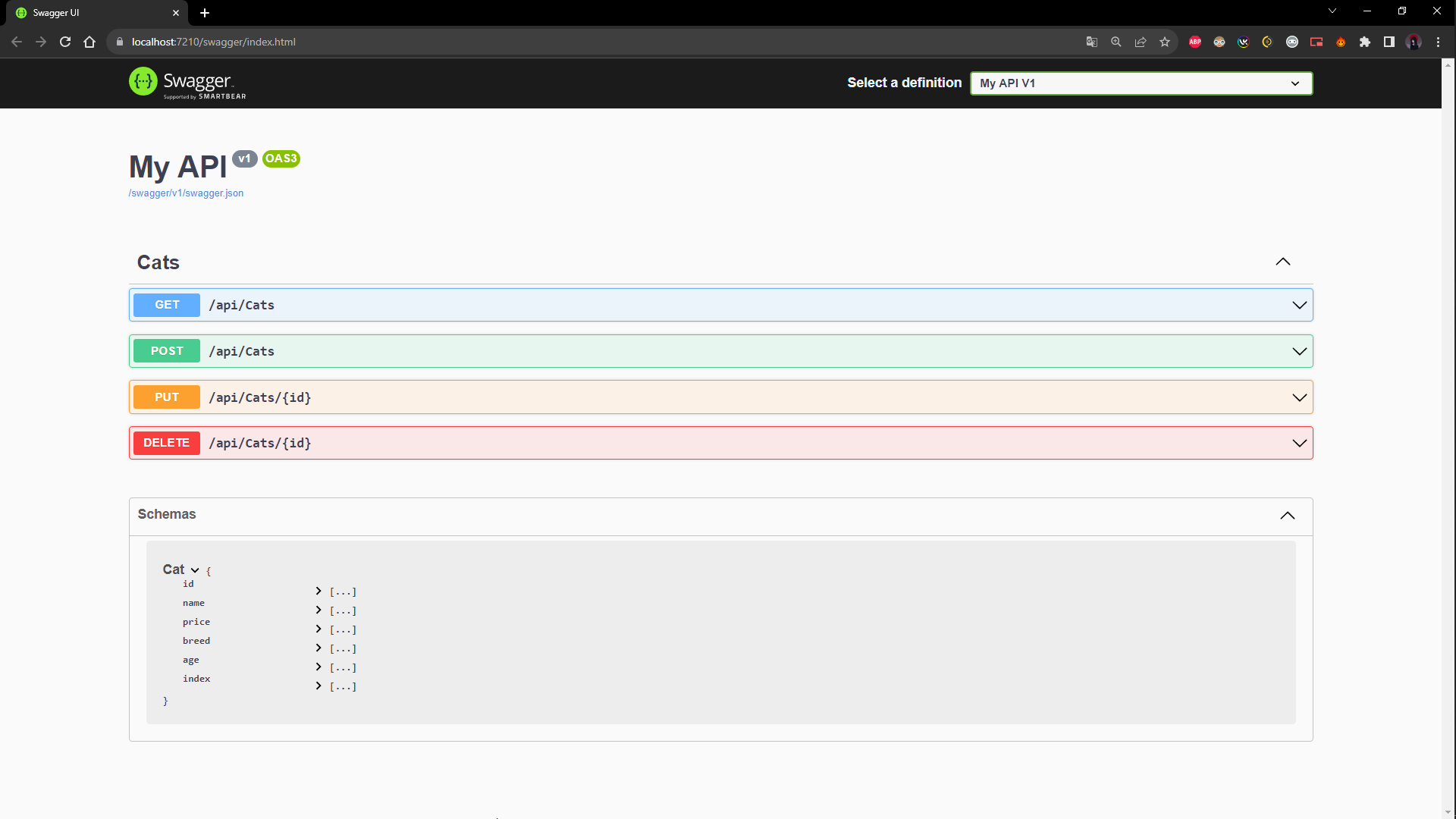


Рисунок 2 – Вид API – сервиса

Для использования Swagger UI был создан класс Starter в котором указаны необходимые параметры

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using MongoDB.Driver;

using CatShopService;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Options;

using Microsoft.AspNetCore.Builder;

using Microsoft.OpenApi.Models;

namespace Startup;

public class Startup

{

public IConfiguration Configuration { get; }

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.Configure<MongoDbSettings>(

Configuration.GetSection(nameof(MongoDbSettings)));

services.AddSingleton<IMongoClient>(ServiceProvider =>

{

var settings = ServiceProvider.GetRequiredService<IOptions<MongoDbSettings>>().Value;

return new MongoClient(settings.ConnectionString);

});

services.AddSingleton<IMongoDatabase>(ServiceProvider =>

{

var client = ServiceProvider.GetRequiredService<IMongoClient>();

var settings = ServiceProvider.GetRequiredService<IOptions<MongoDbSettings>>().Value;

return client.GetDatabase(settings.DatabaseName);

});

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });

});

// Регистрируйте ICatService и CatService в DI контейнере

services.AddScoped<ICatService, CatService>();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");

});

app.UseRouting();

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage(); // Middleware для обработки исключений в режиме разработки

app.UseSwagger(c =>

{

c.RouteTemplate = "swagger/{documentName}/swagger.json";

}); // Middleware для Swagger

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1"); // Настройка Swagger UI

});

}

else

{

app.UseHsts(); // Middleware для настройки HTTP Strict Transport Security (HSTS) в продакшн среде

}

app.UseHttpsRedirection(); // Middleware для перенаправления на HTTPS

app.UseStaticFiles(); // Middleware для обслуживания статических файлов (CSS, JS и др.)

app.UseRouting(); // Middleware для маршрутизации запросов

app.UseAuthentication(); // Middleware для аутентификации

app.UseAuthorization(); // Middleware для авторизации

IApplicationBuilder applicationBuilder = app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "CatController",

pattern: "Classes/CatController/{action=Index}/{id?}",

defaults: new { controller = "CatController" }

);

})

;

}

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

public class MongoDbSettings

{

public required string ConnectionString { get; set; }

public required string DatabaseName { get; set; }

}

Данный API – сервис реализует простейшие операции с базой данных

## **1.2 Каскадное удаление в MongoDB**

Каскадное удаление было реализовано в отдельном консольном приложении работающего с той же базой данных. При удалении документа по его id из коллекции Cats вместе с ним также удалялся документ из коллекции Comments содержащий в одном из своих полей Id документа с котом.

using System;

using System.Collections.Generic;

using System.Threading.Tasks;

using MongoDB.Bson;

using MongoDB.Bson.Serialization.Attributes;

using MongoDB.Driver;

class Program

{

static async Task Main()

{

string connectionString = "mongodb://localhost:27017";

MongoClient client = new MongoClient(connectionString);

IMongoDatabase database = client.GetDatabase("CatShop");

IMongoCollection<Cat> catsCollection = database.GetCollection<Cat>("Cats");

IMongoCollection<Comment> commentsCollection = database.GetCollection<Comment>("Comments");

var random = new Random();

string catIdToDelete = "id кота, который нужно удалить"; // Замените этот Id на реальный Id кота

// Шаг 1: Удаление комментариев, связанных с котом

await commentsCollection.DeleteManyAsync(comment => comment.CatId == catIdToDelete);

// Шаг 2: Удаление самого кота

await catsCollection.DeleteOneAsync(cat => cat.Id == catIdToDelete);

}

}

public class Cat

{

[BsonId]

[BsonRepresentation(BsonType.ObjectId)]

public string Id { get; set; }

[BsonElement("name")]

public string name { get; set; }

public decimal price { get; set; }

public string breed { get; set; }

public int age { get; set; }

}

public class Comment

{

[BsonId]

[BsonRepresentation(BsonType.ObjectId)]

public string Id { get; set; }

public string Text { get; set; }

[BsonRepresentation(BsonType.ObjectId)]

public string CatId { get; set; }

}

# **ВЫВОД**

В качестве вывода, можно отметить, что были изучены базовые принципы работы с API – сервисами, создание базы данных и внесения в нее изменений на программном уровне, изучен принцип каскадного удаления.

# **ПРИЛОЖЕНИЯ**

## **Приложение А**

Исходный код с созданием базы данных и каскадным удалением:  
using MongoDB.Bson;

using MongoDB.Bson.Serialization.Attributes;

using MongoDB.Driver;

class Program

{

static async Task Main()

{

string connectionString = "mongodb://localhost:27017";

MongoClient client = new MongoClient(connectionString);

IMongoDatabase database = client.GetDatabase("CatShop");

IMongoCollection<Cat> catsCollection = database.GetCollection<Cat>("Cats");

IMongoCollection<Comment> commentsCollection = database.GetCollection<Comment>("Comments");

var random = new Random();

// Генерация и вставка 10,000 документов для котов

for (int i = 0; i < 10000; i++)

{

Cat newCat = new Cat

{

Id = ObjectId.GenerateNewId().ToString(),

name = "Cat" + i,

price = random.Next(50, 200),

breed = "Breed" + random.Next(1, 5),

age = random.Next(1, 10)

};

await catsCollection.InsertOneAsync(newCat);

// Генерация и вставка 1 комментария для каждого кота

Comment newComment = new Comment

{

Id = ObjectId.GenerateNewId().ToString(),

Text = "Comment for " + newCat.name,

CatId = newCat.Id // Устанавливаем CatId комментария равным Id кота

};

await commentsCollection.InsertOneAsync(newComment);

}

Console.WriteLine("Документы успешно добавлены в коллекции Cats и Comments.");

string catIdToDelete = "id кота, который нужно удалить"; // Замените этот Id на реальный Id кота

// Шаг 1: Удаление комментариев, связанных с котом

await commentsCollection.DeleteManyAsync(comment => comment.CatId == catIdToDelete);

// Шаг 2: Удаление самого кота

await catsCollection.DeleteOneAsync(cat => cat.Id == catIdToDelete);

}

}

public class Cat

{

[BsonId]

[BsonRepresentation(BsonType.ObjectId)]

public string Id { get; set; }

[BsonElement("name")]

public string name { get; set; }

public decimal price { get; set; }

public string breed { get; set; }

public int age { get; set; }

}

public class Comment

{

[BsonId]

[BsonRepresentation(BsonType.ObjectId)]

public string Id { get; set; }

public string Text { get; set; }

[BsonRepresentation(BsonType.ObjectId)]

public string CatId { get; set; }

}

**Приложение В**

Исходный код для API – Сервиса

Класс Main:

using Microsoft.Extensions.Options;

using MongoDB.Bson;

using MongoDB.Bson.Serialization.Attributes;

using MongoDB.Driver;

namespace CatShopApi.Models

{

public class Cat

{

[BsonId]

[BsonRepresentation(BsonType.ObjectId)]

public required string Id { get; set; }

[BsonElement("name")]

public string name { get; set; }

public decimal price { get; set; }

public string breed { get; set; }

public int age { get; set; }

public int index { get; set; }

}

public class Comment

{

[BsonId]

[BsonRepresentation(BsonType.ObjectId)]

public string Id { get; set; }

public string Text { get; set; }

[BsonRepresentation(BsonType.ObjectId)]

public string CatId { get; set; } // Ссылка на кота

}

}

public class CatShopDatabaseSettings

{

public CatShopDatabaseSettings(string connectionString, string databaseName, string catCollectionName)

{

ConnectionString = connectionString;

DatabaseName = databaseName;

CatCollectionName = catCollectionName;

}

public string ConnectionString { get; set; } = null!;

public string DatabaseName { get; set; } = null!;

public string CatCollectionName { get; set; } = null!;

}

Класс CatService:

using CatShopApi.Models;

using Microsoft.Extensions.Options;

using MongoDB.Bson;

using MongoDB.Bson.Serialization.Attributes;

using MongoDB.Driver;

namespace CatShopService

{

public interface ICatService

{

Task<List<CatShopApi.Models.Cat>> GetAsync();

Task<CatShopApi.Models.Cat> GetAsync(string id);

Task CreateAsync(CatShopApi.Models.Cat newCat);

Task UpdateAsync(string id, CatShopApi.Models.Cat updatedCat);

Task<bool> DeleteCatAndAssociatedComments(string catId);

Task RemoveAsync(string id);

}

public class CatService : ICatService

{

private readonly IMongoCollection<CatShopApi.Models.Cat> \_catsCollection;

private readonly IMongoCollection<Comment> \_commentsCollection;

public CatService(IMongoDatabase database)

{

\_catsCollection = database.GetCollection<CatShopApi.Models.Cat>("Cats");

\_commentsCollection = database.GetCollection<Comment>("comments");

}

public async Task<List<CatShopApi.Models.Cat>> GetAsync()

{

return await \_catsCollection.Find(cat => true).ToListAsync();

}

public async Task<CatShopApi.Models.Cat> GetAsync(string id)

{

return await \_catsCollection.Find(x => x.Id == id).FirstOrDefaultAsync();

}

public async Task CreateAsync(CatShopApi.Models.Cat newCat)

{

await \_catsCollection.InsertOneAsync(newCat);

}

public async Task UpdateAsync(string id, CatShopApi.Models.Cat updatedCat)

{

var filter = Builders<CatShopApi.Models.Cat>.Filter.Eq(x => x.Id, id);

var update = Builders<CatShopApi.Models.Cat>.Update

.Set(x => x.name, updatedCat.name)

.Set(x => x.breed, updatedCat.breed);

await \_catsCollection.UpdateOneAsync(filter, update);

}

public async Task<bool> DeleteCatAndAssociatedComments(string catId)

{

// Шаг 1: Удаление комментариев, связанных с котом

await \_commentsCollection.DeleteManyAsync(comment => comment.CatId == catId);

// Шаг 2: Удаление самого кота

var deleteResult = await \_catsCollection.DeleteOneAsync(cat => cat.Id == catId);

// Возвращаем результат удаления (true, если удалено успешно)

return deleteResult.DeletedCount > 0;

}

public async Task RemoveAsync(string id)

{

var filter = Builders<CatShopApi.Models.Cat>.Filter.Eq("Id", id);

await \_catsCollection.DeleteOneAsync(filter);

}

}

}

Класс CatController:

using CatShopApi.Models;

using CatShopService;

using Microsoft.AspNetCore.Mvc;

namespace CatShopApi.Controllers

{

[ApiController]

[Route("api/[controller]")]

public class CatsController : ControllerBase

{

private readonly ICatService \_catsService;

public CatsController(ICatService catsService)

{

\_catsService = catsService;

}

[HttpGet]

public async Task<List<Cat>> Get() =>

await \_catsService.GetAsync();

[HttpPost]

public async Task<IActionResult> Post(Cat newCat)

{

await \_catsService.CreateAsync(newCat);

return CreatedAtAction(nameof(Get), new { id = newCat.Id }, newCat);

}

[HttpPut("{id:length(24)}")]

public async Task<IActionResult> Update(string id, Cat updatedCat)

{

var cat = await \_catsService.GetAsync(id);

if (cat is null)

{

return NotFound();

}

updatedCat.Id = id;

await \_catsService.UpdateAsync(id, updatedCat);

return NoContent();

}

[HttpDelete("{id:length(24)}")]

public async Task<IActionResult> Delete(string id)

{

var deleteResult = await \_catsService.DeleteCatAndAssociatedComments(id);

if (deleteResult)

{

return Ok("Cat and associated comments deleted successfully.");

}

else

{

return NotFound("Cat not found or error deleting associated comments.");

}

}

}

}

Класс Startup:

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using MongoDB.Driver;

using CatShopService;

using Microsoft.AspNetCore.Hosting;

using Microsoft.Extensions.Hosting;

using Microsoft.Extensions.Options;

using Microsoft.AspNetCore.Builder;

using Microsoft.OpenApi.Models;

namespace Startup;

public class Startup

{

public IConfiguration Configuration { get; }

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public void ConfigureServices(IServiceCollection services)

{

services.AddControllers();

services.Configure<MongoDbSettings>(

Configuration.GetSection(nameof(MongoDbSettings)));

services.AddSingleton<IMongoClient>(ServiceProvider =>

{

var settings = ServiceProvider.GetRequiredService<IOptions<MongoDbSettings>>().Value;

return new MongoClient(settings.ConnectionString);

});

services.AddSingleton<IMongoDatabase>(ServiceProvider =>

{

var client = ServiceProvider.GetRequiredService<IMongoClient>();

var settings = ServiceProvider.GetRequiredService<IOptions<MongoDbSettings>>().Value;

return client.GetDatabase(settings.DatabaseName);

});

services.AddSwaggerGen(c =>

{

c.SwaggerDoc("v1", new OpenApiInfo { Title = "My API", Version = "v1" });

});

// Регистрируйте ICatService и CatService в DI контейнере

services.AddScoped<ICatService, CatService>();

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

app.UseSwagger();

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1");

});

app.UseRouting();

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage(); // Middleware для обработки исключений в режиме разработки

app.UseSwagger(c =>

{

c.RouteTemplate = "swagger/{documentName}/swagger.json";

}); // Middleware для Swagger

app.UseSwaggerUI(c =>

{

c.SwaggerEndpoint("/swagger/v1/swagger.json", "My API V1"); // Настройка Swagger UI

});

}

else

{

app.UseHsts(); // Middleware для настройки HTTP Strict Transport Security (HSTS) в продакшн среде

}

app.UseHttpsRedirection(); // Middleware для перенаправления на HTTPS

app.UseStaticFiles(); // Middleware для обслуживания статических файлов (CSS, JS и др.)

app.UseRouting(); // Middleware для маршрутизации запросов

app.UseAuthentication(); // Middleware для аутентификации

app.UseAuthorization(); // Middleware для авторизации

IApplicationBuilder applicationBuilder = app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "CatController",

pattern: "Classes/CatController/{action=Index}/{id?}",

defaults: new { controller = "CatController" }

);

})

;

}

public class Program

{

public static void Main(string[] args)

{

CreateHostBuilder(args).Build().Run();

}

public static IHostBuilder CreateHostBuilder(string[] args) =>

Host.CreateDefaultBuilder(args)

.ConfigureWebHostDefaults(webBuilder =>

{

webBuilder.UseStartup<Startup>();

});

}

}

public class MongoDbSettings

{

public required string ConnectionString { get; set; }

public required string DatabaseName { get; set; }

}