

*Cài Đặt Gói NuGet Cần Thiết*

Cài đặt các gói sau bằng NuGet:

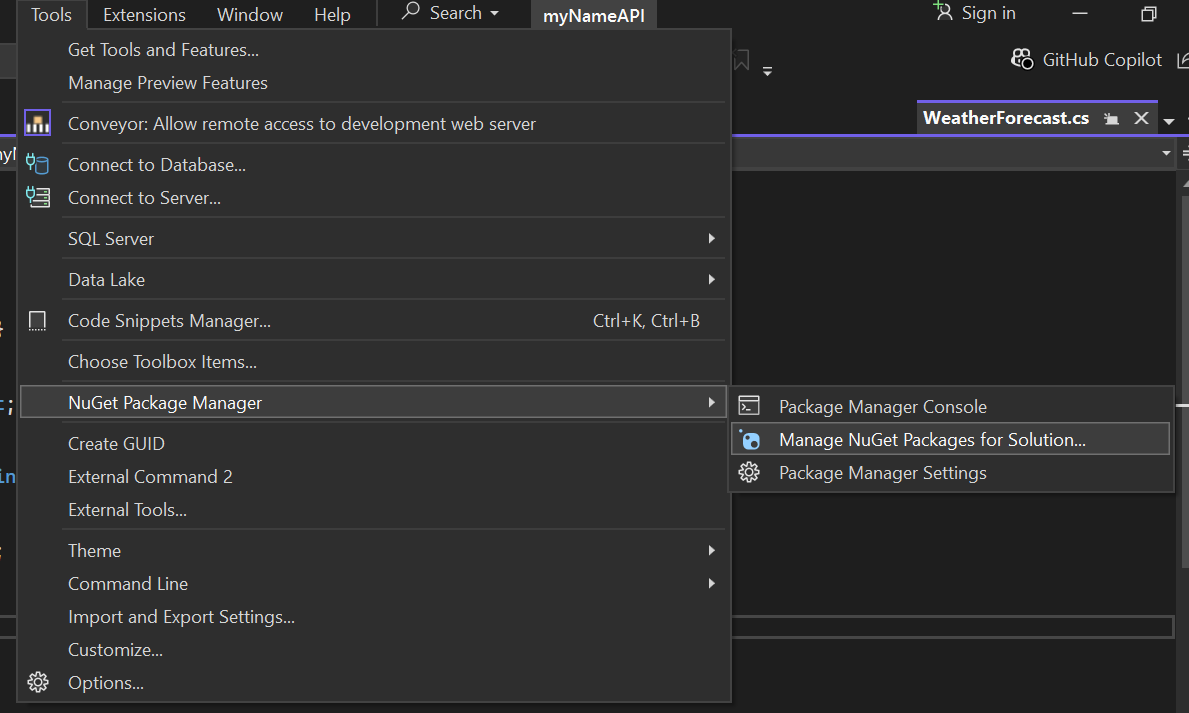
- ‘ **Microsoft.EntityFrameworkCore**`🡺phiên bản 8.0.3

- ‘ **Microsoft.EntityFrameworkCore.Design**`🡺phiên bản 8.0.3

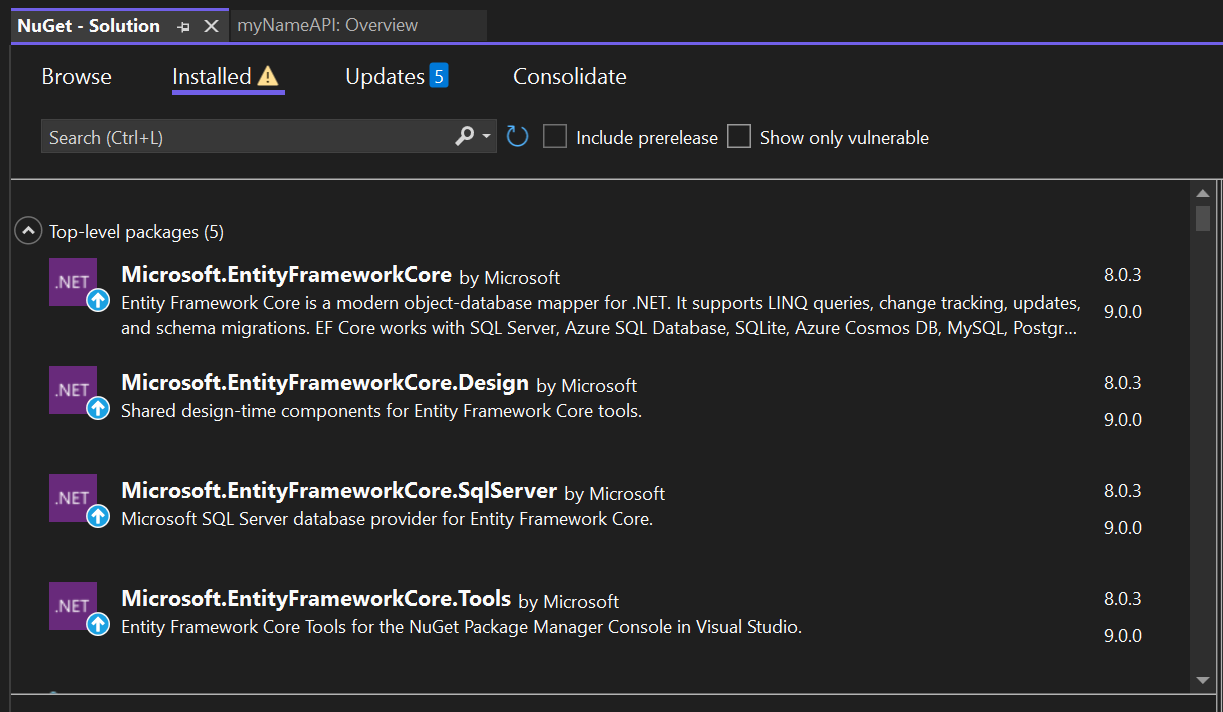
- `**Microsoft.EntityFrameworkCore.SqlServer**`🡺phiên bản 8.0.3 (hoặc

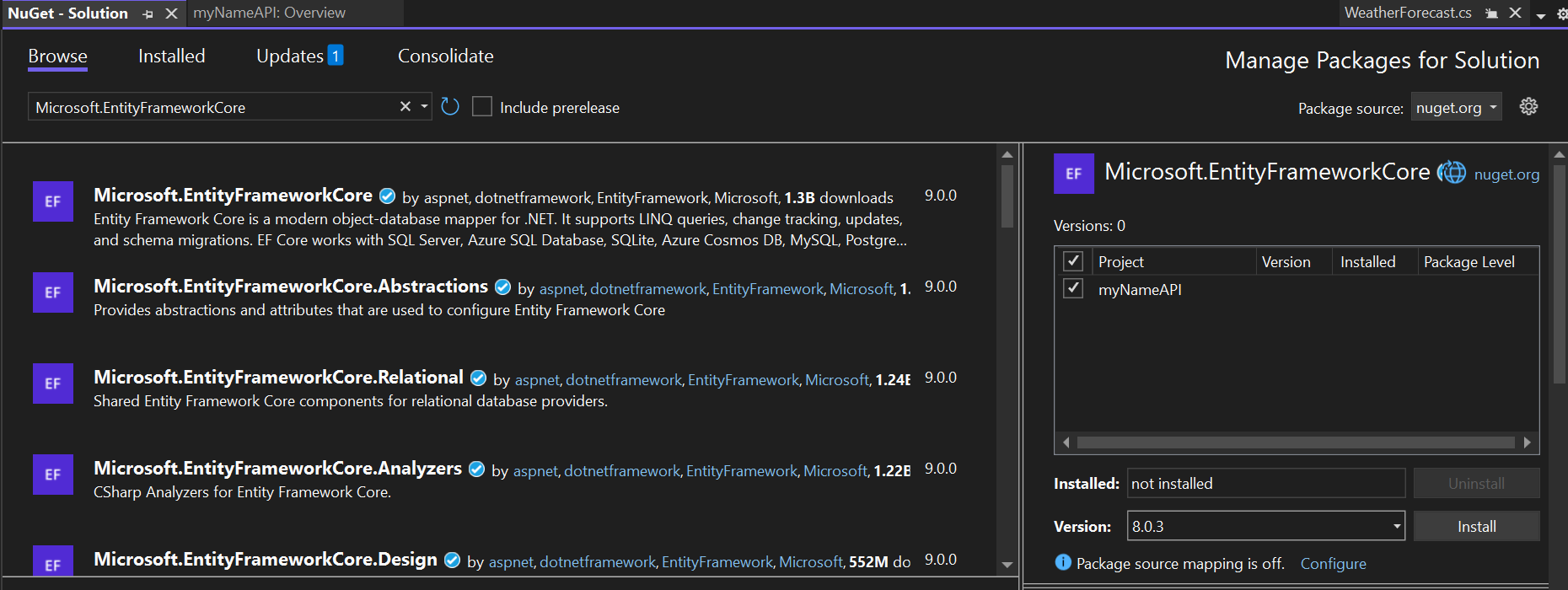
provider cơ sở dữ liệu khác)

- `**Microsoft.EntityFrameworkCore.Tools**`🡺phiên bản 8.0.3



Kiểm tra các gói đã cài đặt





*Dưới đây là một ví dụ đơn giản về cách xây dựng một* RESTful API *cho* Product *trong* ASP.NET Core*, sử dụng* Controller và Repository Pattern.

Tạo thư mục models trong Project

Tạo Class Product trong thư mục Models

public class Product

{

public int Id { get; set; }

public string? Name { get; set; }

public decimal Price { get; set; }

public string? Image { get; set; }

public string? Description { get; set; }

// Other propertie

}

Tạo Class IproductRepository trong thư mục Repositories

public interface IProductRepository

{

Task<IEnumerable<Product>> GetProductsAsync();

Task<Product> GetProductByIdAsync(int id);

Task AddProductAsync(Product product);

Task UpdateProductAsync(Product product);

Task DeleteProductAsync(int id);

}

Tạo Class ProductRepository trong thư mục Repositories

public class ProductRepository : IProductRepository

{

private readonly ApplicationDbContext \_context;

public ProductRepository(ApplicationDbContext context)

{

\_context = context;

}

public async Task<IEnumerable<Product>> GetProductsAsync()

{

return await \_context.Products.ToListAsync();

}

public async Task<Product> GetProductByIdAsync(int id)

{

return await \_context.Products.FindAsync(id);

}

public async Task AddProductAsync(Product product)

{

\_context.Products.Add(product);

await \_context.SaveChangesAsync();

}

public async Task UpdateProductAsync(Product product)

{

\_context.Entry(product).State = EntityState.Modified;

await \_context.SaveChangesAsync();

}

public async Task DeleteProductAsync(int id)

{

var product = await \_context.Products.FindAsync(id);

if (product != null)

{

\_context.Products.Remove(product);

await \_context.SaveChangesAsync();

}

}

}

Tạo Class ApplicationDbContext trong thư mục Models và cấu hình

public class ApplicationDbContext : DbContext

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options) : base(options) { }

public DbSet<Product> Products { get; set; }

}

***Cấu hình EF Core trong file `*Program.cs*`:***

var builder = WebApplication.CreateBuilder(args);

//Cấu hình: Dùng SQL Server làm hệ quản trị csdl cho ứng dụng

builder.Services.AddDbContext<ApplicationDbContext>(options =>options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

//Cấu hình khác…

***Cấu hình connection string trong `appsettings.json`.***

{

//Chỉ cho chương trình biết tên server và tên DB muốn đặt

"ConnectionStrings": {

"DefaultConnection": "Server=MY-PC\\MRHUNG189;Database=myNameAPI;Trusted\_Connection=True;TrustServerCertificate=True"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft.AspNetCore": "Warning"

}

},

"AllowedHosts": "\*"

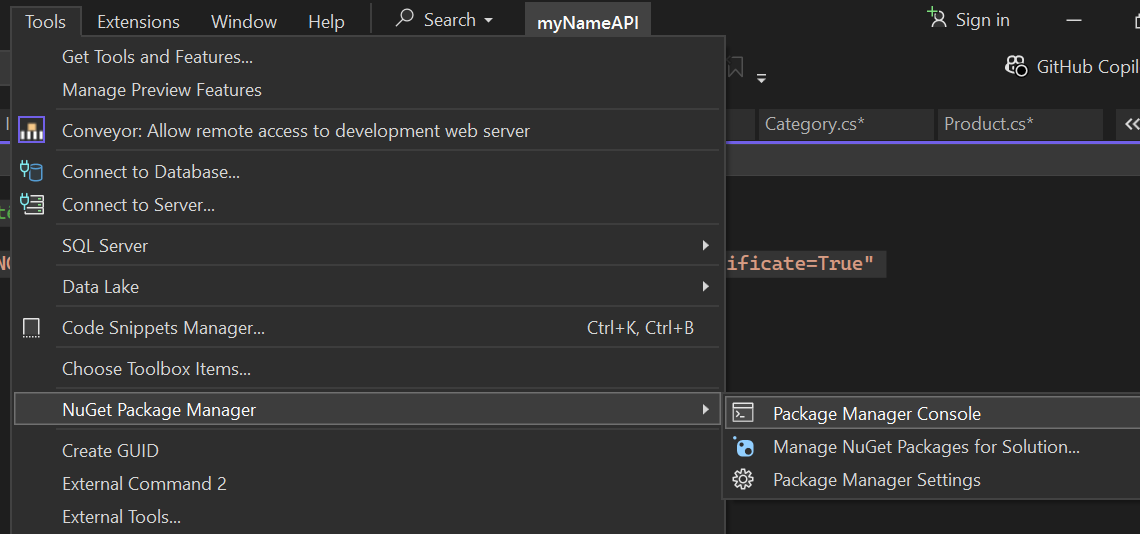
}

**Sử dụng migrations để ánh xạ và cập nhật các bảng vào trong CSDL**

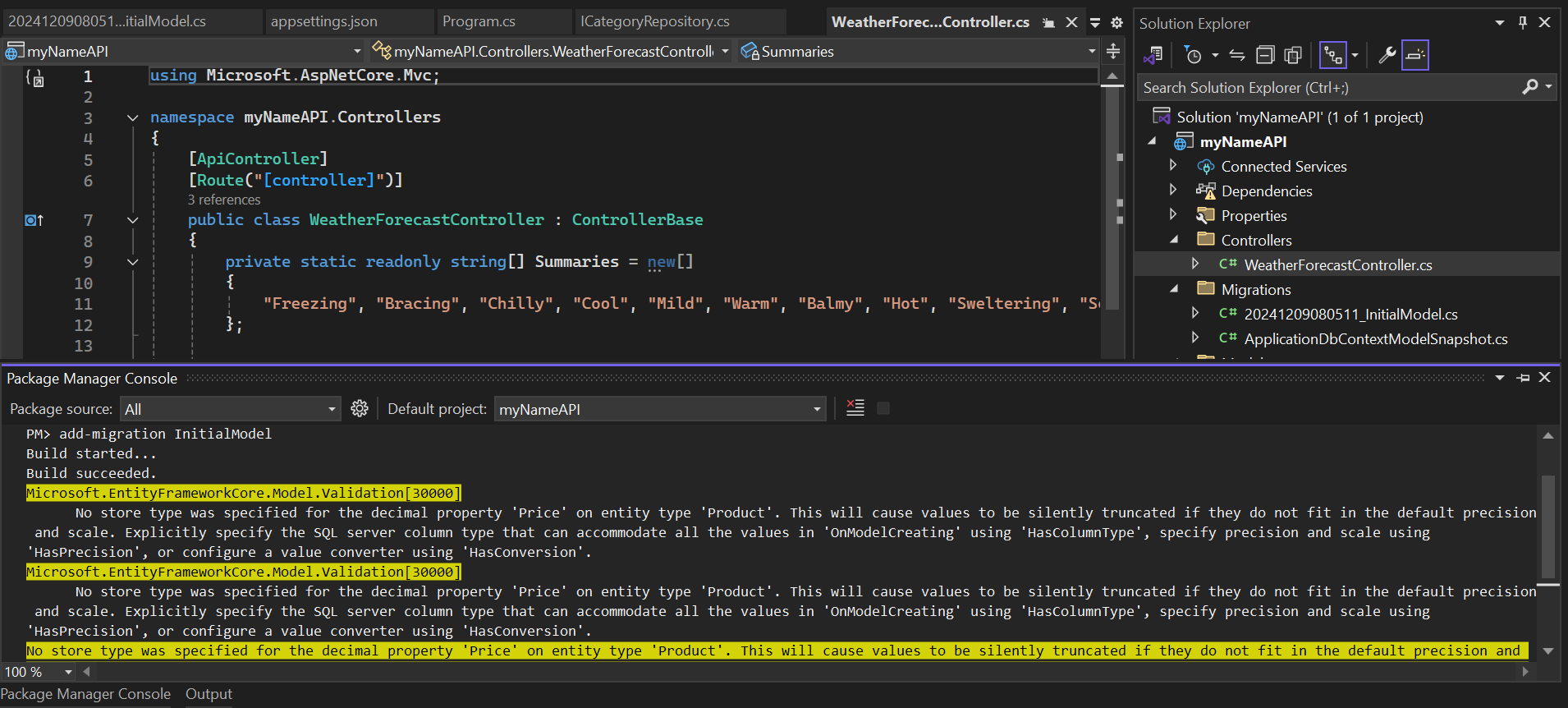
**Entity Framework Code First**

*Add-Migration Initial*

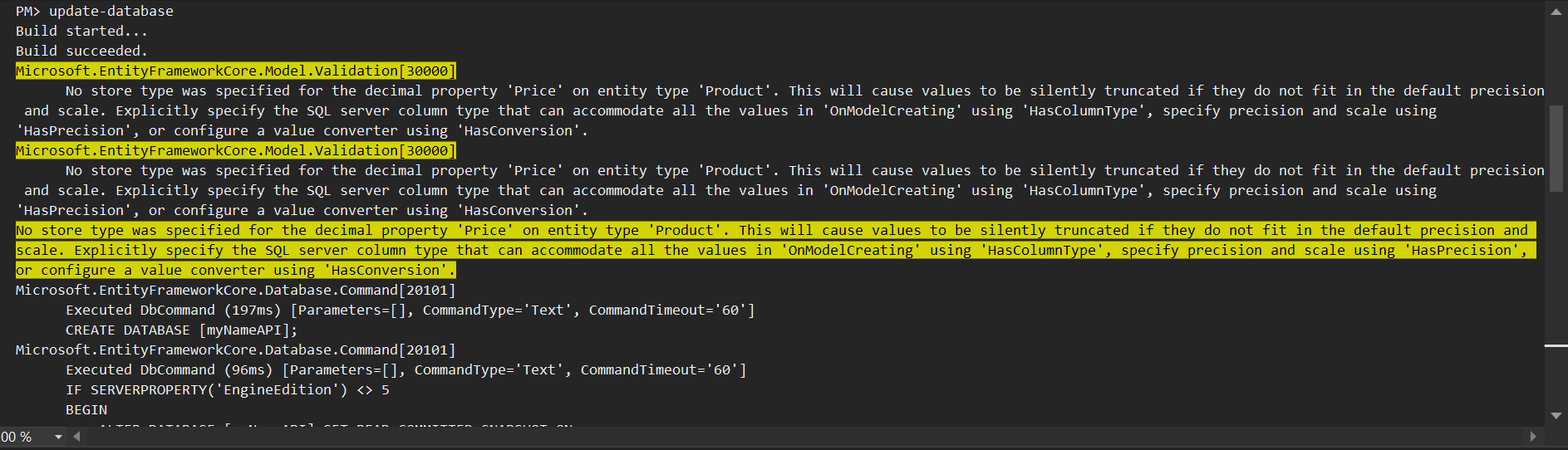
*Update-Database*

****

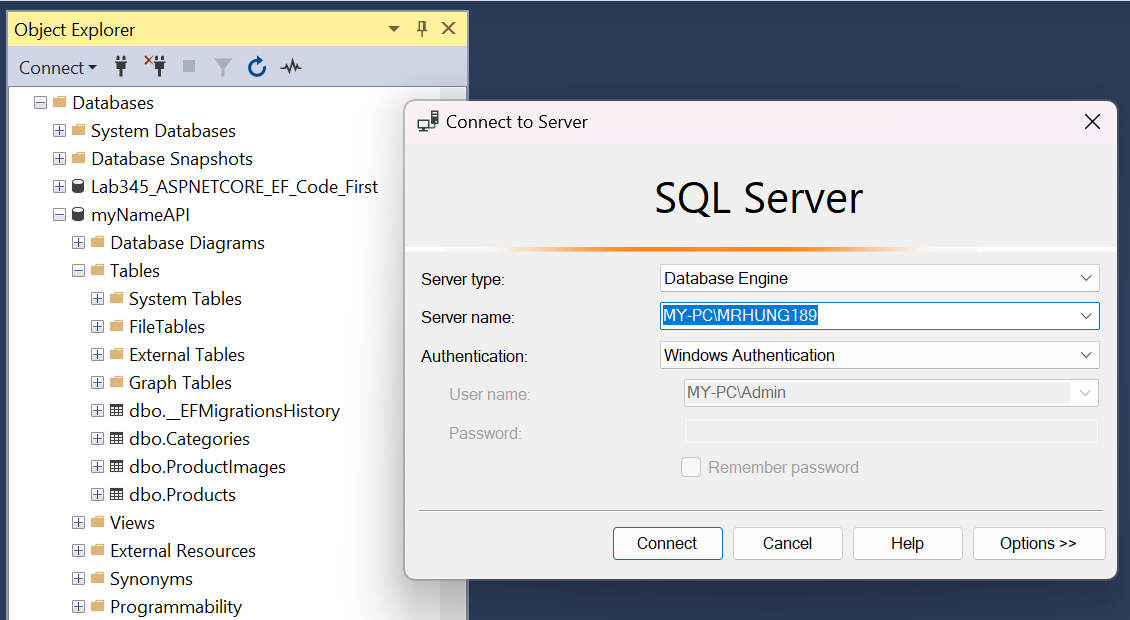
**Lệnh: add-migration InitialModel**

****

**Lệnh: update-database**

****

**Kiểm tra CSDL đã được cập nhật trong SQL Server chưa?**

****

Tạo Class ProductApiController trong thư mục Controllers

public class ProductApiController : ControllerBase

{

private readonly IProductRepository \_productRepository;

public ProductApiController(IProductRepository productRepository)

{

\_productRepository = productRepository;

}

[

HttpGet]

public async Task<IActionResult> GetProducts()

{

try

{

var products = await \_productRepository.GetProductsAsync();

return Ok(products);

}

catch (Exception ex)

{

// Handle exception

return StatusCode(500, "Internal server error");

}

}

[

HttpGet("{id}")]

public async Task<IActionResult> GetProductById(int id)

{

try

{

var product = await \_productRepository.GetProductByIdAsync(id);

if (product == null)

return NotFound();

return Ok(product);

}

catch (Exception ex)

{

// Handle exception

return StatusCode(500, "Internal server error");

}

}

[

HttpPost]

public async Task<IActionResult> AddProduct([FromBody] Product product)

{

try

{

await \_productRepository.AddProductAsync(product);

return CreatedAtAction(nameof(GetProductById), new

{

id =

product.Id

}, product);

}

catch (Exception ex)

{

// Handle exception

return StatusCode(500, "Internal server error");

}

}

[

HttpPut("{id}")]

public async Task<IActionResult> UpdateProduct(int id, [FromBody] Product product)

{

try

{

if (id != product.Id)

return BadRequest();

await \_productRepository.UpdateProductAsync(product);

return NoContent();

}

catch (Exception ex)

{

// Handle exception

return StatusCode(500, "Internal server error");

}

}

[

HttpDelete("{id}")]

public async Task<IActionResult> DeleteProduct(int id)

{

try

{

await \_productRepository.DeleteProductAsync(id);

return NoContent();

}

catch (Exception ex)

{

// Handle exception

return StatusCode(500, "Internal server error");

}

}

}

Mở file Program.cs và bổ sung cấu hình

using Microsoft.EntityFrameworkCore;

using testAPI.models;

using testAPI.Repositories;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(builder.Configuration.GetConnectionString("DefaultConnection")));

// Add services to the container.

builder.Services.AddControllers();

builder.Services.AddScoped<IProductRepository, ProductRepository>();

// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddSwaggerGen();

// Configure CORS: Cross-Origin Resource Sharing, được dịch là “Chia sẻ tài nguyên giữa các nguồn gốc khác nhau

builder.Services.AddCors(options =>

{

options.AddPolicy(name: "MyAllowOrigins", policy =>

{

//Thay bằng địa chỉ localhost khi khởi chạy bên frontend (VSCode)

policy.WithOrigins("http://127.0.0.1:5500", "http://localhost:5500")

.AllowAnyHeader()

.AllowAnyMethod();

});

});

var app = builder.Build();

// Middleware for production error handling

if (!app.Environment.IsDevelopment())

{

app.UseExceptionHandler("/error"); // Custom error handling endpoint

app.UseHsts(); // Enforce HTTPS in production

}

else

{

app.UseSwagger();

app.UseSwaggerUI();

}

app.UseHttpsRedirection();

// Áp dụng CORS cho các yêu cầu đến API

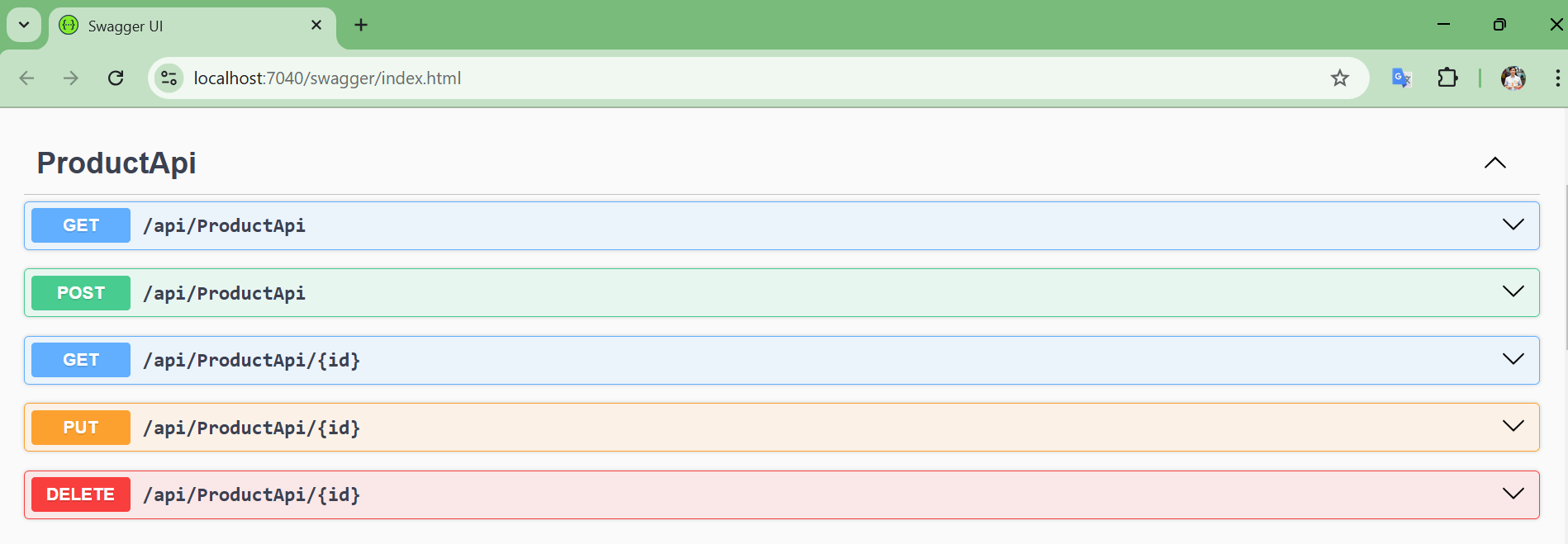
app.UseCors("MyAllowOrigins");

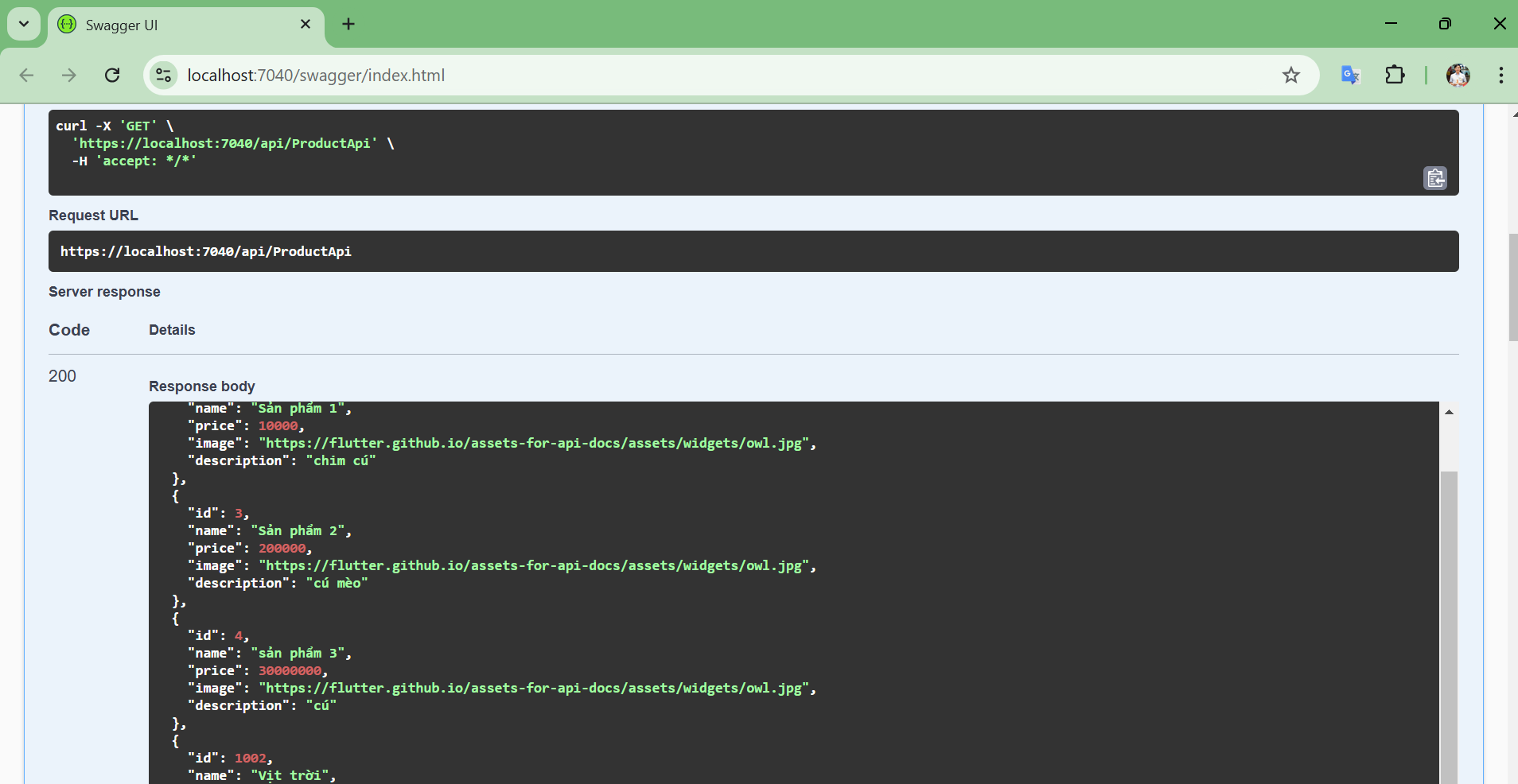
app.UseAuthorization();

app.MapControllers();

app.Run();

chạy thử chương trình và xem kết quả API (nhớ thêm dữ liệu vào DB trước)





Json nhận được từ API trả về

**{**

**"id": 2,**

**"name": "Sản phẩm 1",**

**"price": 10000,**

**"image": "https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg",**

**"description": "chim cú"**

**},**

**{**

**"id": 3,**

**"name": "Sản phẩm 2",**

**"price": 200000,**

**"image": "https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg",**

**"description": "cú mèo"**

**},**

**{**

**"id": 4,**

**"name": "sản phẩm 3",**

**"price": 30000000,**

**"image": "https://flutter.github.io/assets-for-api-docs/assets/widgets/owl.jpg",**

**"description": "cú"**

**},**

**{**

**"id": 1002,**

**"name": "Vịt trời",**

**"price": 10000,**

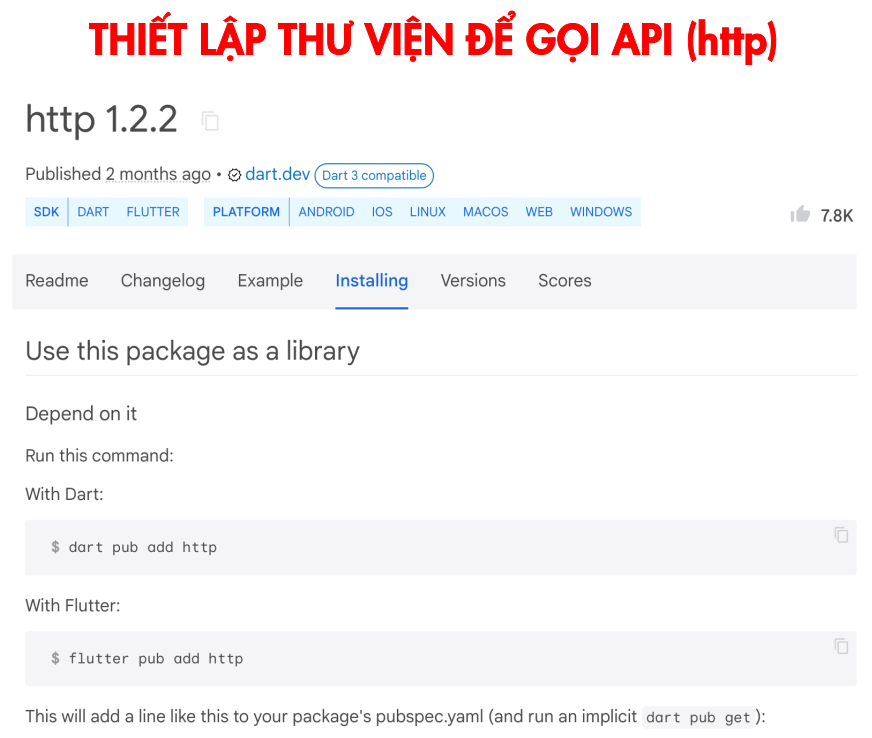
**"image": "Cập nhật sau",**

**"description": "Con này làm món gì ta?"**

**}**

Có thể dùng Live Server trên Visual Code để chạy server hoặc dùng ứng dụng bên thứ 3 (Conveyor)

Thiết lập thư viện gọi API (HTTP) từ pub.dev



Tạo model Post cho bài viết

Copy dữ liệu Json nhận được về từ API vào trang web: <https://dart-quicktype.netlify.app/> để tạo model tương ứng với dữ liệu Json

class productPost {  
 productPost({  
 required this.id,  
 required this.name,  
 required this.price,  
 required this.image,  
 required this.description,  
 });  
  
 final int? id;  
 final String? name;  
 final double? price;  
 final String? image;  
 final String? description;  
  
 factory productPost.fromJson(Map<String, dynamic> json){  
 return productPost(  
 id: json["id"],  
 name: json["name"],  
 price: json["price"],  
 image: json["image"],  
 description: json["description"],  
 );  
 }  
  
 Map<String, dynamic> toJson() => {  
 "id": id,  
 "name": name,  
 "price": price,  
 "image": image,  
 "description": description,  
 };  
  
}

//đổ dữ liệu vào trang marketPlace đã làm từ bài hôm trước

import 'dart:convert';  
  
import 'package:facebook\_interface/models/productPost.dart';  
import 'package:flutter/material.dart';  
import 'package:http/http.dart' as http;  
  
class MarketScreen extends StatelessWidget {  
 const MarketScreen({super.key});  
  
 Future<List<productPost>> fetchPosts() async {  
 final String baseUrl= 'https://greatshinypen67.conveyor.cloud/api/ProductApi';  
 final response= await http.get(Uri.*parse*(baseUrl));  
 if (response.statusCode == 200) {  
 List<dynamic> data = json.decode(response.body);  
 return data.map((item) => productPost.fromJson(item)).toList();  
 } else {  
 throw Exception('Failed to load products');  
 }  
 }  
  
 @override  
 Widget build(BuildContext context) {  
 return Scaffold(  
 appBar: AppBar(  
 title: const Text('Market Screen'),  
 ),  
 body: FutureBuilder<List<productPost>>(  
 future: fetchPosts(), // Fetch data from the API  
 builder: (context, snapshot) {  
 if (snapshot.connectionState == ConnectionState.waiting) {  
 return const Center(child: CircularProgressIndicator());  
 } else if (snapshot.hasError) {  
 return Center(child: Text('Error: ${snapshot.error}'));  
 } else if (!snapshot.hasData || snapshot.data!.isEmpty) {  
 return const Center(child: Text('No products available.'));  
 } else {  
 List<productPost> posts = snapshot.data!;  
  
 return ListView.builder(  
 itemCount: posts.length,  
 itemBuilder: (context, index) {  
 productPost post = posts[index];  
 return ListTile(  
 leading: Image.network(post.image ?? ''),  
 title: Text(post.name ?? 'No name'),  
 subtitle: Text(post.description ?? 'No description'),  
 trailing: Text('${post.price} VND'),  
 );  
 },  
 );  
 }  
 },  
 ),  
 );  
 }  
}

Bài tập làm thêm:

1. Thực hiện gọi các API: Thêm/xóa/ sửa/ Tìm theo ID và đổ vào giao diện
2. Thực hiện viết API tìm kiếm và đổ vào giao diện

Bài tập về nhà:

1. Tạo API bài đăng (tương tự như bài đăng fb)
2. Thực hiện việc đổ dữ liệu bài đăng, thêm/ xóa/ sửa
3. Xây dựng giao diện cho đồ án môn học + xây dựng API và đổ dữ liệu lên giao diện