<https://github.com/occbuu/LTCSDL-192>

\* Môi trường:

1) Cài đặt Visual Studio 2019

https://visualstudio.microsoft.com/

2) Cài đặt Dotnet Core 3.1.3 hoặc cao hơn

https://dotnet.microsoft.com/download...0

3) Cài đặt NodeJS

https://nodejs.org/en/download/

4) Cài đặt Angular CLI (9.0 trở lên)

https://cli.angular.io/

5) Cài đặt Visual Studio Code

https://visualstudio.microsoft.com/

\* Các bước thực hiện

6) Tạo Solution & project Web MVC (Presentation layer)

7) Tạo project DAL (Data Layer)

8) Tạo Entity Framework trong project DAL

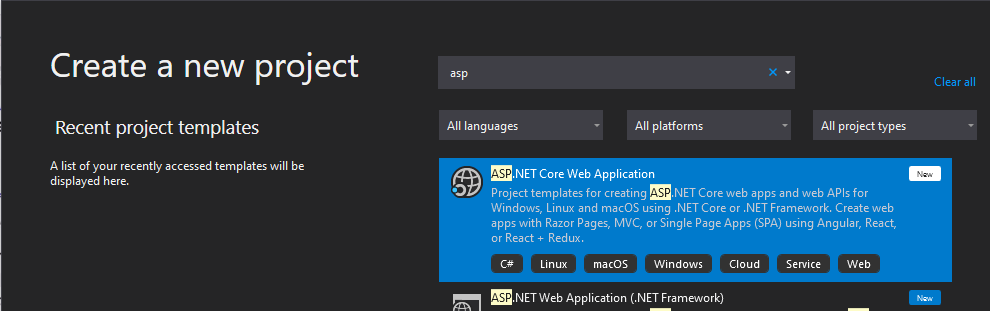
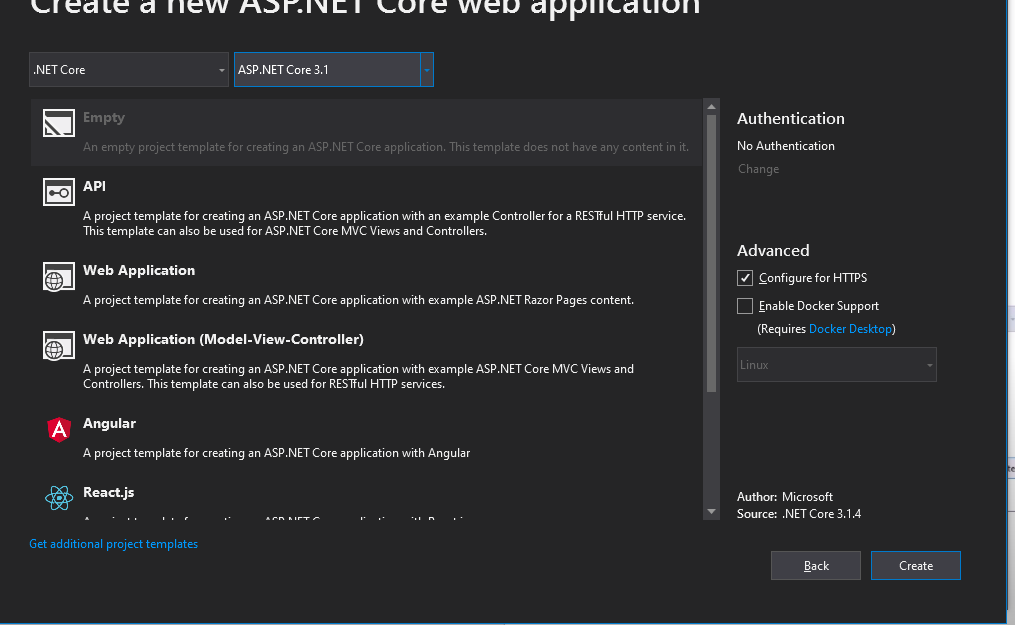
9) Tạo project BLL (Business Layer)

10) Tạo project Common (Các đối tượng sử dụng chung)

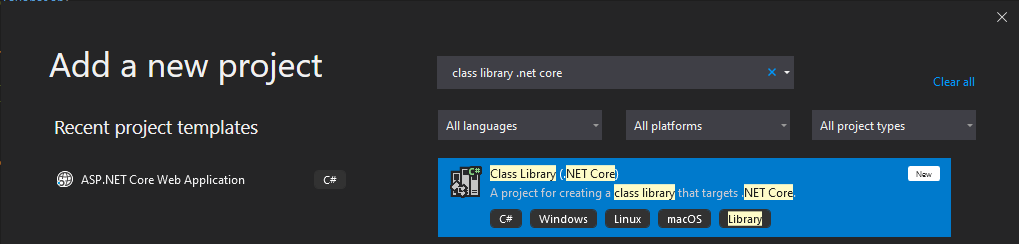
11) Tạo mẫu CategoryController

12) Nhúng Swagger Dependencies vào project Web

**6) Tạo Solution & project Web MVC (Presentation layer) :** Tạo dự án angular trong project luôn.

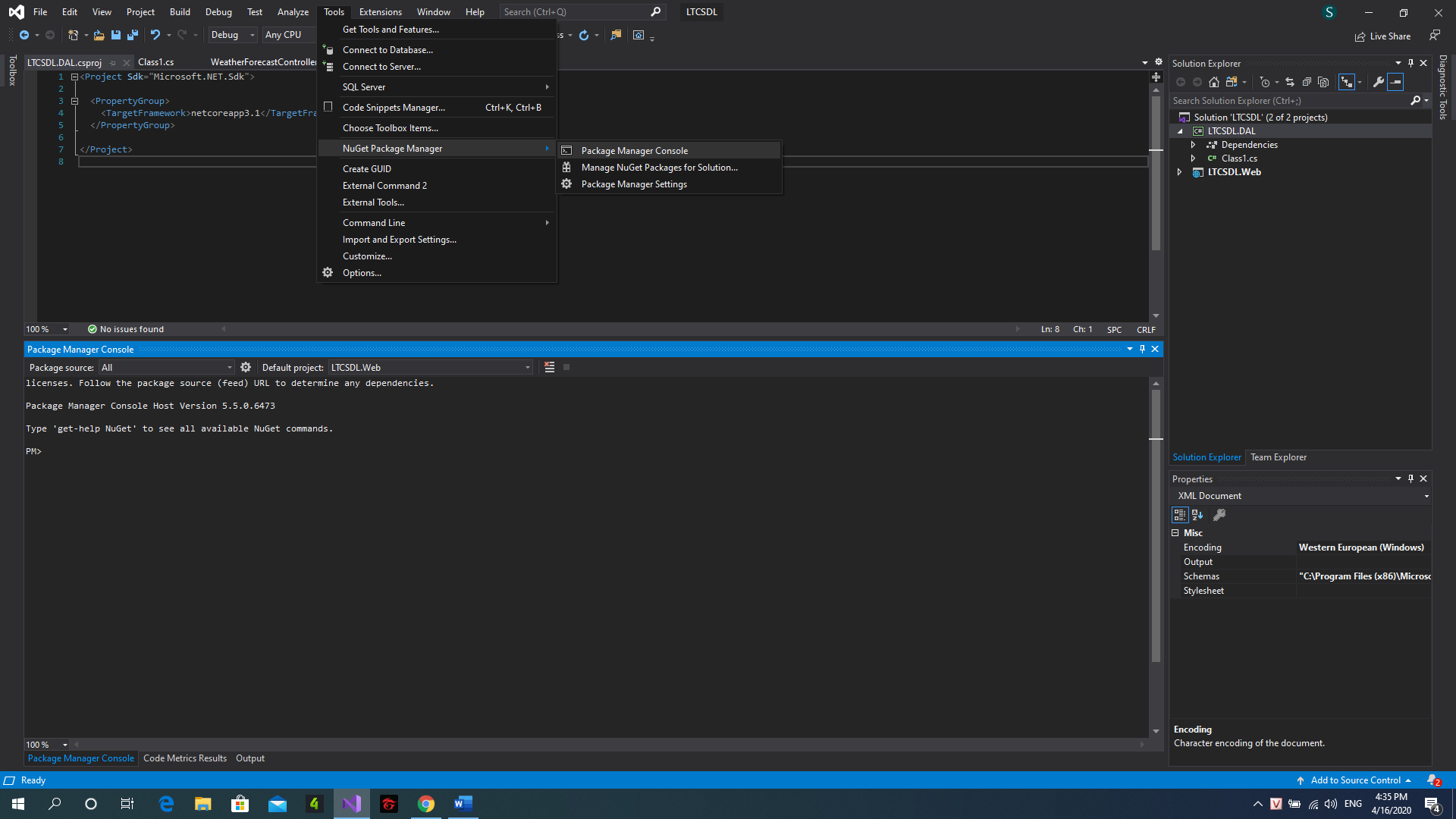
1.  => chon ASP.NET core web Application.
2. 1 Solution thì có nhiều Project . Đặt tên solution và project theo ý mình.
3.  => Dùng core 3.1 và angular.
4.  => Nhấn IIS Express để Run project. Nó sẽ tự động chạy dự án trên Browser cho mình.

**7) Tạo project DAL (Data Layer):**

1. Tạo mới 1 project mới tượng trưng cho Data Layer trong Solution: Class library .NET core
   1.  => đặt tên project là LTCSDL.DAL

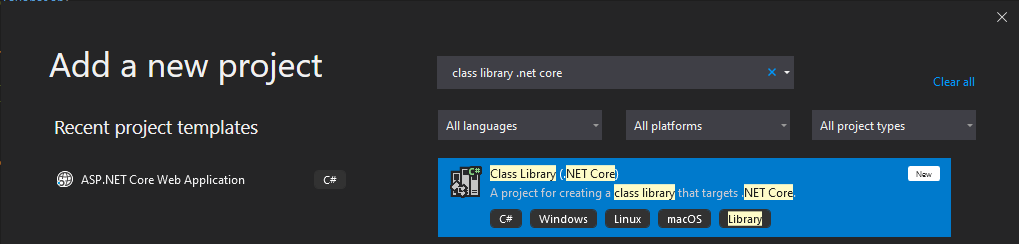
**8) Tạo Entity Framework trong project DAL:**

1.Cần kết nối với Enity Framework để kết nối Project với database:

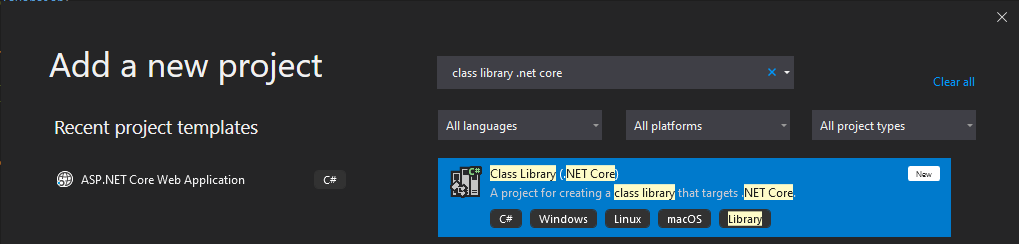
* 1.  => Mở cửa sổ console, chuyển qua class DAL
  2. *‘Install-Package Microsoft.EntityFrameworkCore’*
  3. *‘Install-Package Microsoft.EntityFrameworkCore.SqlServer’*
  4. *‘Install-Package Microsoft.EntityFrameworkCore.Tools’*
  5. *‘*Scaffold-DbContext "Data Source=.;Initial Catalog=LibraryManagement;Persist Security Info=True;User ID=sa;Password=Password123;Pooling=False;MultipleActiveResultSets=False;Encrypt=False;TrustServerCertificate=True;" Microsoft.EntityFrameworkCore.SqlServer -OutputDir Models -Force*’.*
  6. Lưu ý: mỗi lớp đều kết nối vs database thì đều chạy các câu lệnh trên, phải giữ dụng SQL Authentication, phải có username và password.

1. Sau khi kết nối Entity với database Northwind, sẽ tạo ra folder Models, là ánh xạ từ database lên project, copy file NorthwindContext.cs ra ngoài thư mục LTCSDL.web, file này là đùng để kết nối database với project.

**9) Tạo project BLL (Business Layer):**

1. Tạo mới 1 project mới tượng trưng cho Business Layer trong Solution: Class library .NET core
   1.  => đặt tên project là LTCSDL.BLL

**10) Tạo project Common (Các đối tượng sử dụng chung):**

1. Tạo mới 1 project mới tượng trưng cho Common Layer trong Solution: Class library .NET core
   1.  => đặt tên project là LTCSDL.Common
2. Cài đặt Entity Framework cho 2 lớp Business và Common bằng cách cài đặt 3 package, 3 line trên cho cả 2 lớp:
   1. *‘Install-Package Microsoft.EntityFrameworkCore’*
   2. *‘Install-Package Microsoft.EntityFrameworkCore.SqlServer’*
   3. *‘Install-Package Microsoft.EntityFrameworkCore.Tools’*
   4. *Khỏi cần chuỗi kết nối map với database.*

-Trong lớp LTCSDL.Common, lớp này dùng để chứa những đối tượng xài chung cho từng lớp DAL, BLL và lớp WEB của Solution:

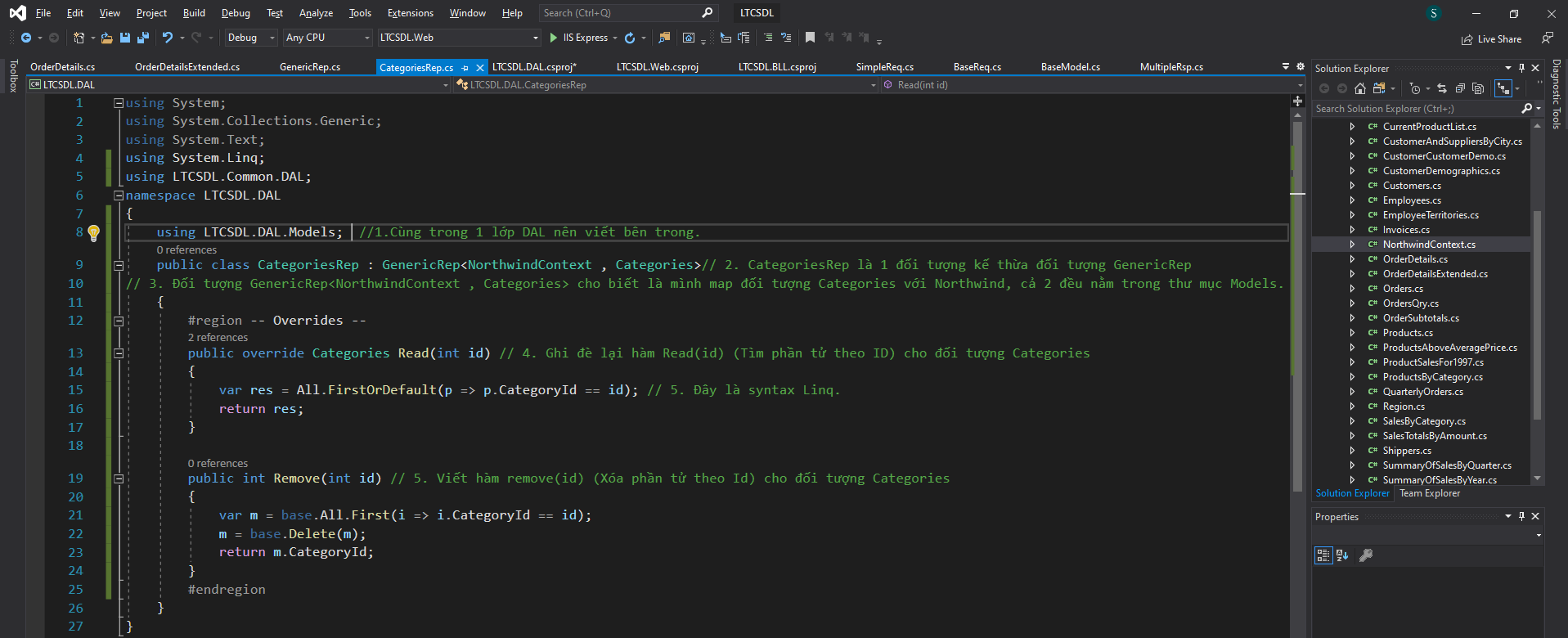
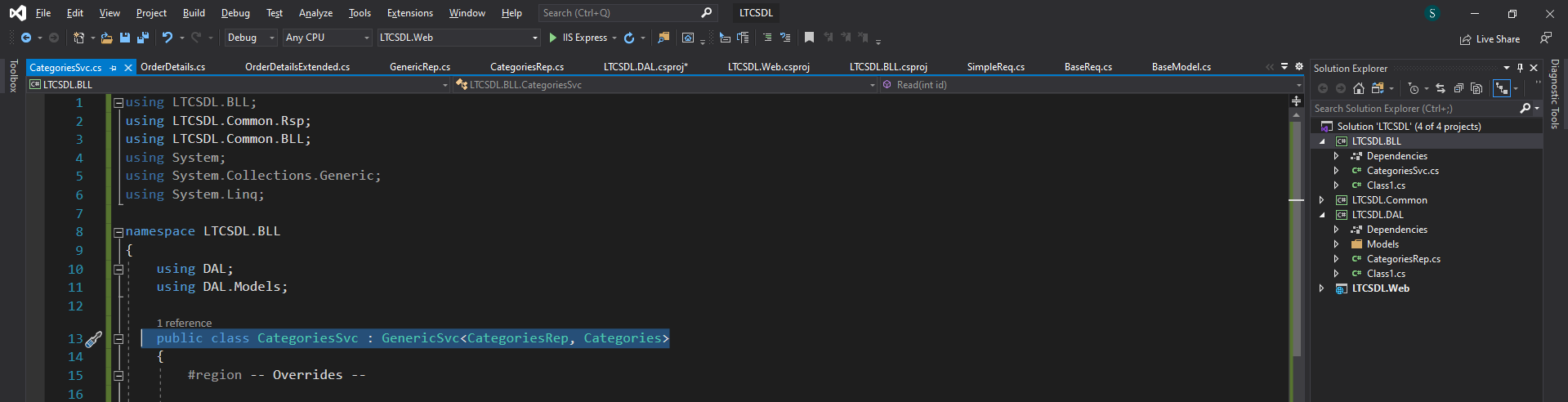
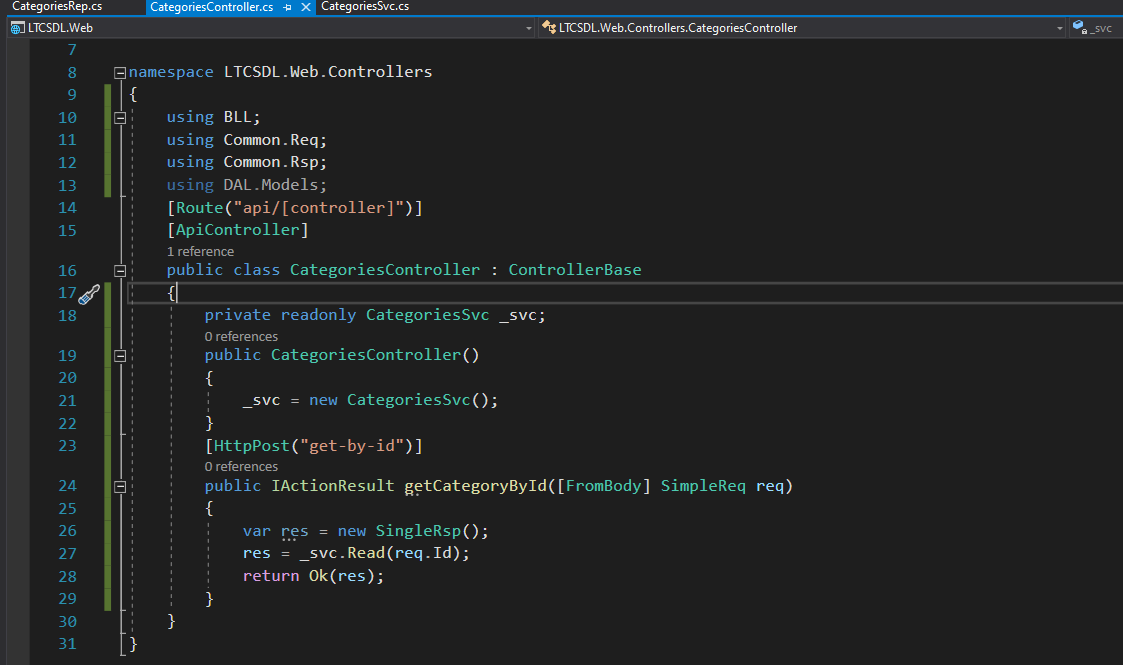
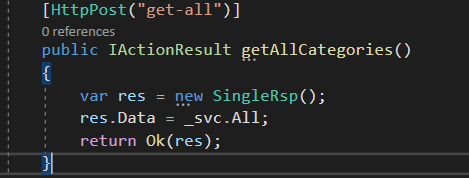
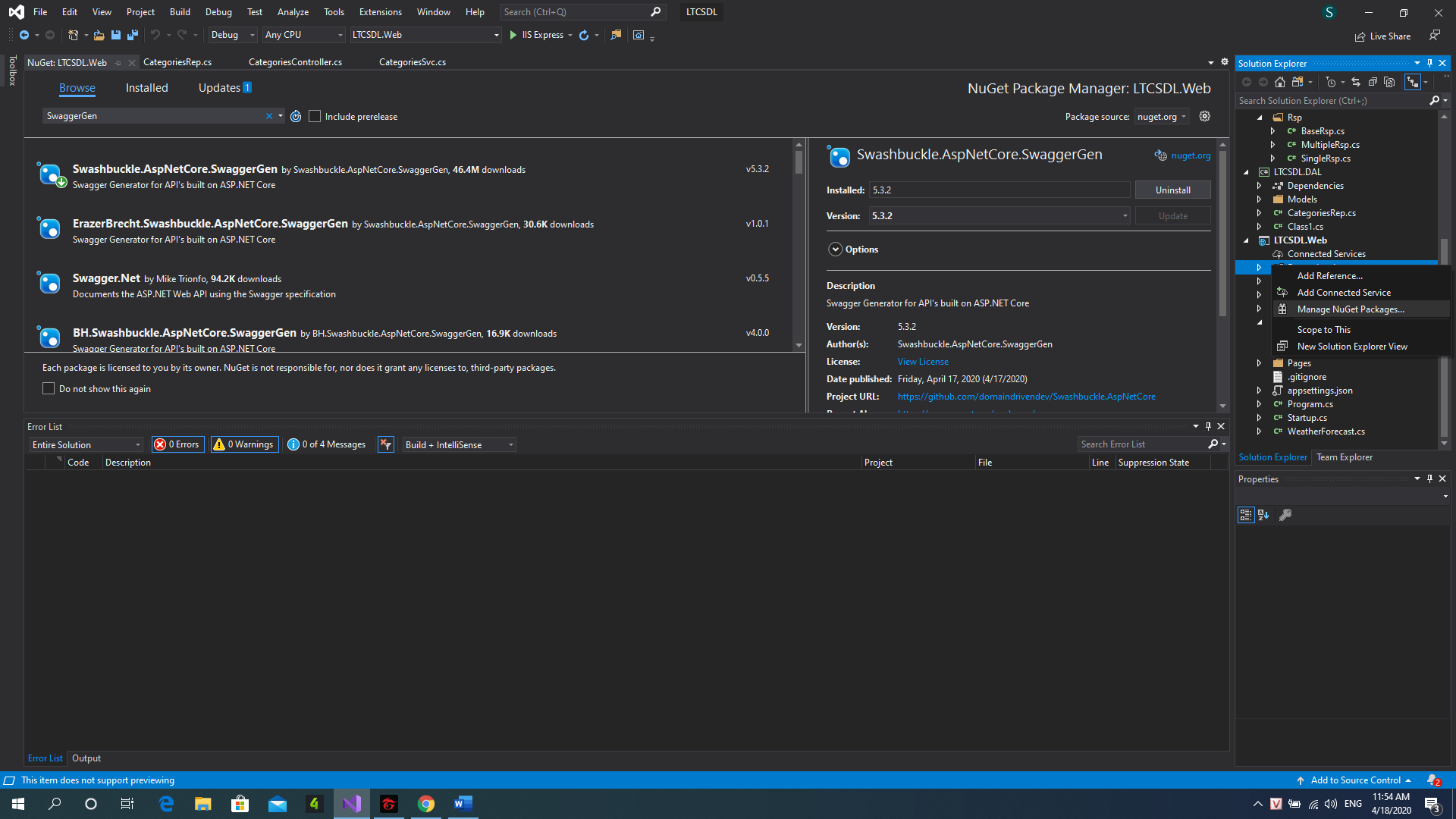
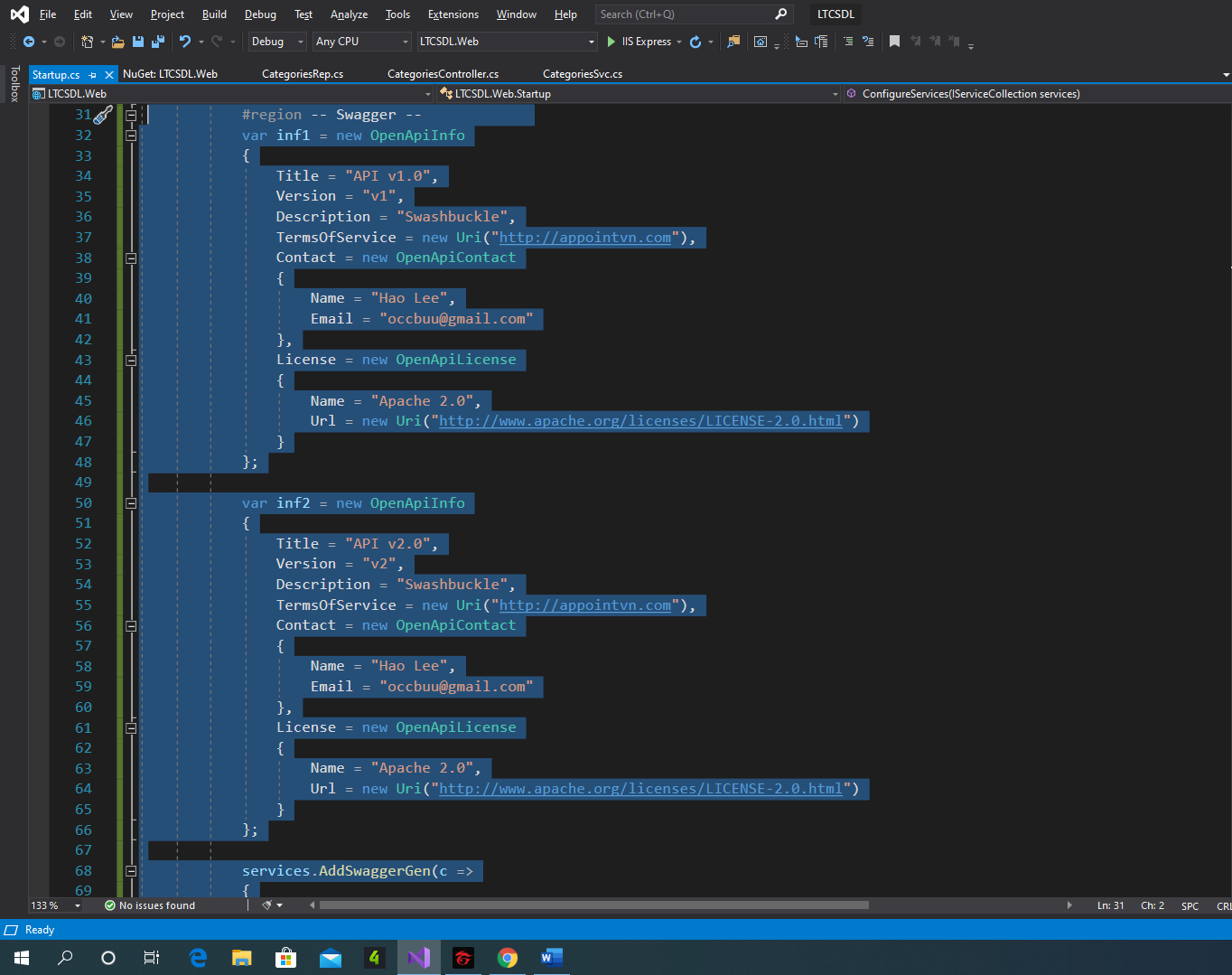
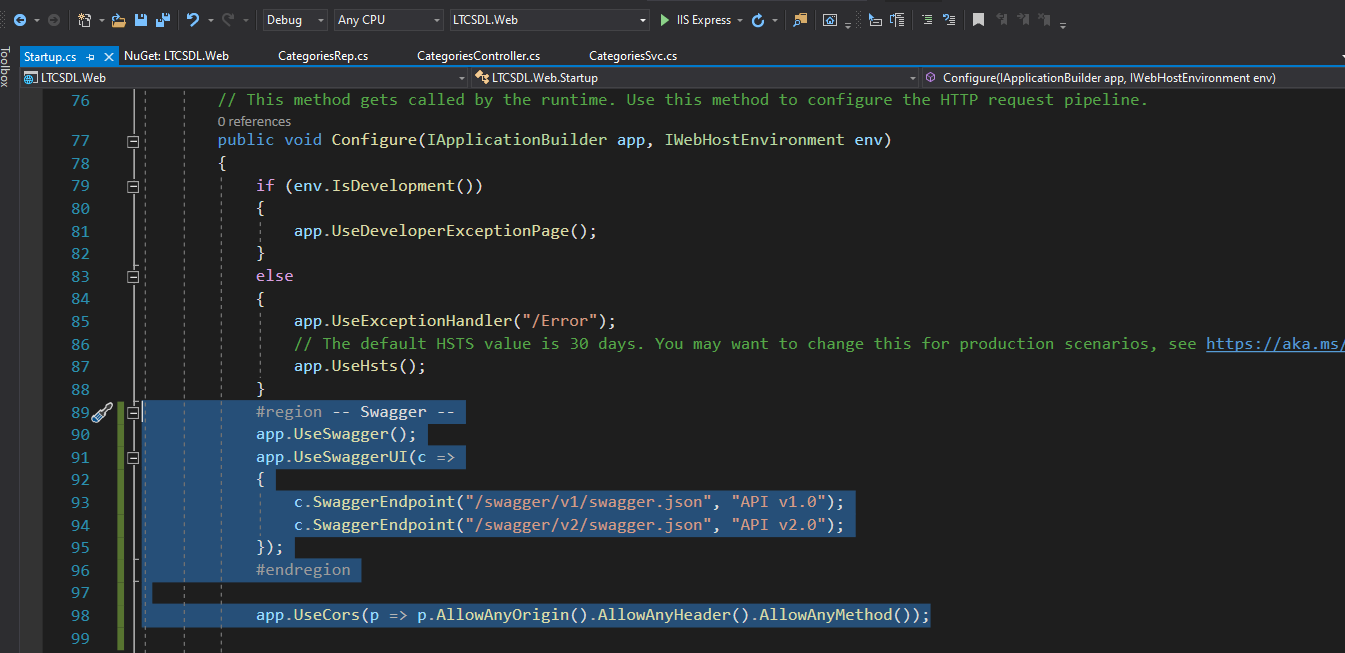
1. Tạo 4 thư mục trong lớp Common: BLL, DAL, RES,REQ
2. Trong thư mục /DAL, tạo đối tượng interface Genergic “IGenergicRep.cs”, tạo lớp Class “GenericRep.cs”.Sau đó, copy nội dung từ file tương ứng của thầy đăng trên Github để paste vào.
3. Trong folder /BLL, tạo tương tự đối tượng interface “IGenergicSvc.cs”, tạo lớp Class “GenericSvc.cs”. Sau đó, copy nội dung từ file tương ứng của thầy đăng trên Github để paste vào.
4. Trong folder /Rsp (response), tạo 3 class “BaseRsp.cs”, “SingleRsp.cs”, “MultipleRsp.cs”. Sau đó copy nội dung từ file tương ứng của thầy đăng trên Github để paste vào.
5. Trong folder /Req (Request), tạo 3 class “SimpleReq.cs”, “BaseReq.cs”, “BaseModel.cs”. Sau đó copy nội dung từ file tương ứng của thầy đăng trên Github để paste vào.

-**Qui tắc** của Project là lớp WEB gọi BLL, BLL gọi DAL (WEB => BLL => DAL).

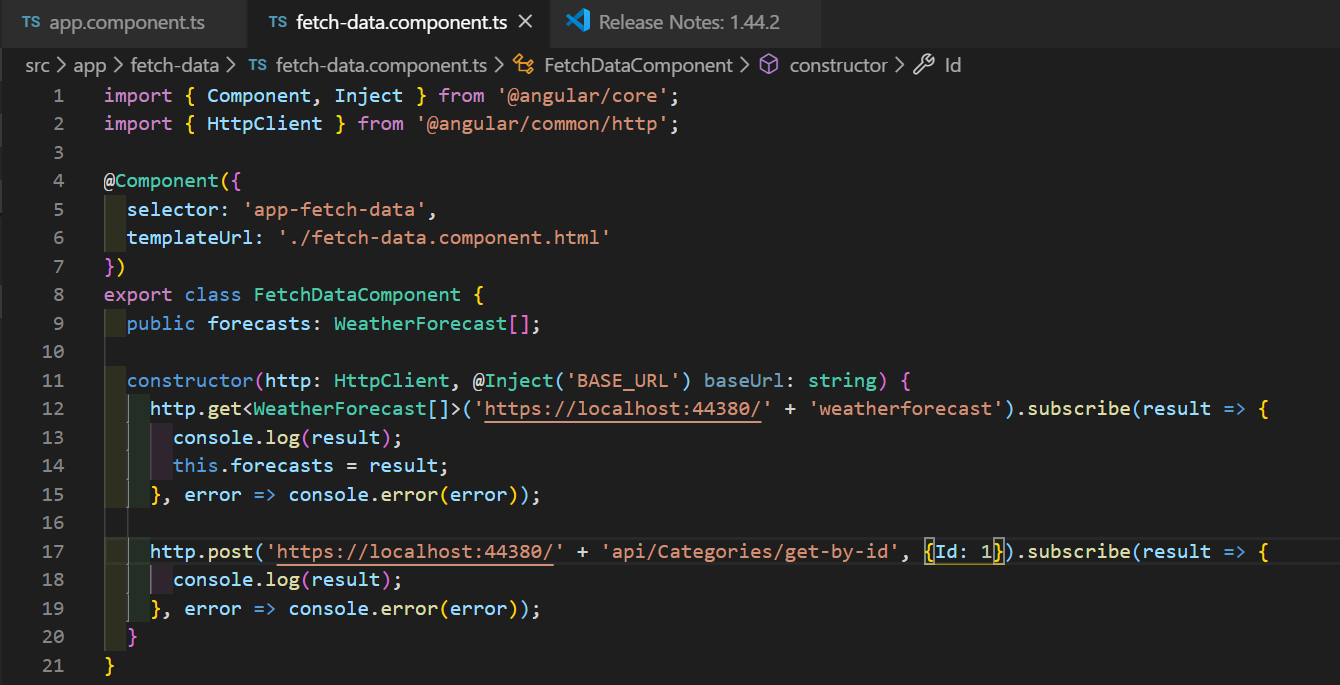
-**Tình hình:** Hiện tại các project trong solution là độc lập với nhau, không có liên kết gì với nhau. Để match với nhau thì vào project ấy -> Dependencies => Add Reference => Chọn các dự án có sử dụng trong Project ấy theo quy tắc (WEB => BLL => DAL):

1. Project LTCSDL.WEB: add cả 3 LTCSDL.Common, LTCSDL.BLL, LTCSDL.DAL.
2. Project LTCSDL.BLL: add LTCSDL.DAL, LTCSDL.Common.
3. Project LTCSDL.DAL: add LTCSDL.Common.

**11) Tạo mẫu CategoryController:**

1. **Trong lớp LTCSDL.DAL,** tạo đối tượng Category “CategoriesRep.cs”(RepsentationS), đại diện cho đối tượng Categories, nó đang trực tiếp kết nối với database.
2. Trong class CategoriesRep.cs:
   1. Import các hàm thư viện cần thiết vào như System.Linq, LTCSDL.Common.DAL
   2. 
      1. Line 8: Cùng trong 1 lớp DAL nên viết bên trong.
      2. Line 9: CategoriesRep là 1 đối tượng kế thừa đối tượng GenericRep.
      3. Line 9: Đối tượng GenericRep<NorthwindContext , Categories> cho biết là mình map đối tượng Categories với Northwind, cả 2 đều nằm trong thư mục Models.
      4. Line 13: Ghi đè lại hàm Read(id) (Tìm phần tử theo ID) cho đối tượng Categories.
      5. Line 15: Đây là syntax Linq.
      6. Line 19: Viết hàm remove(id) (Xóa phần tử theo Id) cho đối tượng Categories.
3. **Trong lớp LTCSDL.BLL**, tạo 1 lớp class service “CategorySvc.cs”
4. Trong class “CategorySvc.cs”:
   1. 
      1. Line 13: CategoriesSvc kế thừa lớp GenericSvc<>, lớp GenericSvc thì map đối tượng ‘Categories’ trong Models DAL với đối tượng ‘CategoriesRep’ trong lớp DAL.
      2. Copy nội dung từ thầy đỡ.
      3. Sẽ Viết cái hàm ở service này để cho controller gọi sử dụng.
5. **Trong lớp LTCSDL.WEB,** tạo mới controller “CategoriesController” (API controller- empty) trong folder controller.
   1. Phải import các đối tượng mình sử dụng vào controler.
   2. Tạo 1 đối tượng CategoriesSvc ‘\_svc’ (private readonly). Tạo 1 hàm khởi tạo (constructor) để khởi tạo giá trị ‘\_svc’ ấy. Vậy là ta map được service với controller
   3. Viết Api ‘get-by-id’:
      1. Viết hàm sử dụng phương thức Post thì viết [HttpPost(“**get-by-id**”)]
      2. Viết 1 hàm public trả về đối tượng IactionResult ‘getCategoryById’. Nhận vào đối tượng SimpleReq “req”.
      3. Dùng hàm Read của service để thực thi, truyền vào Id và trả về result . Dùng phương thức Ok(res) nghĩa là tốt đã thành công.
   4. Viết Api “get-all” :
6. Nhúng Swagger Dependencies vào project Web để hiển thị Api ra bên ngoài nhưng không phải thuộc kiểu giống Front-end.
   1.  => Vào Nuget Add Dependencies cuả WEB: install SwaggerGen và SwaggerUI.
   2. Add Swagger vào project:
      1. Vào Startup.cs, vào phương thức ConfigureServices() (nơi để add các service kèm theo): Vào github của thầy, vào file tương ứng để copy nội dung rồi pass qua. Đây là cách setup. Nhớ using setup các thư viện kèm theo. Như Uri, OpenApi.
      2. 
      3. 
   3. Vào url /Swagger, để gọi mô phỏng Api database.

* OKEY, vậy là đã setup Back-end thành công, có 2 Api “get-by-id”, “get-all-categories”. Giờ qua chạy độc lập Angular. Nhờ thay đổi domain localhost4200 thành localhostXXXXX phù hợp với server.
* Bây giở mình sẽ qua Angular độc lập và gọi API đã viết.

1. Gọi Api bên Angular Project với Http Service:
   1. 

\*HƯỚNG DẪN VIẾT API SEARCH PRODUCT:

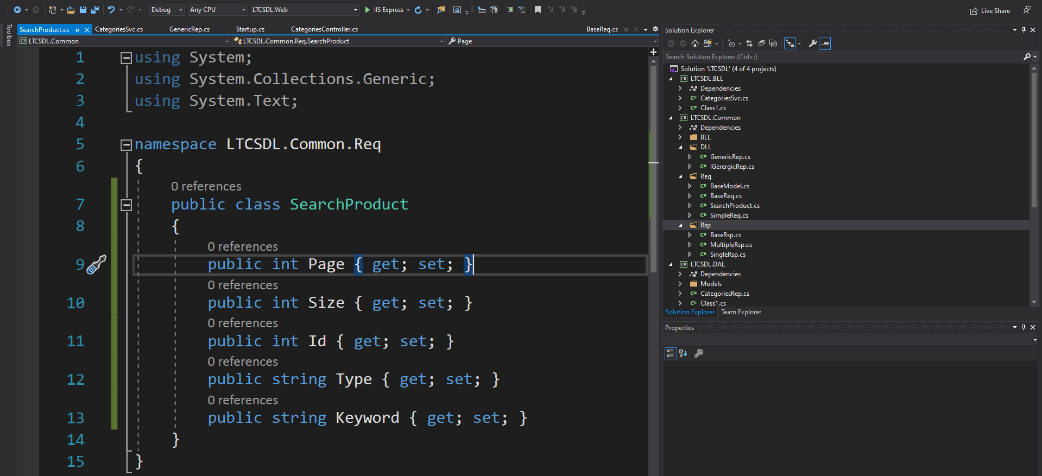
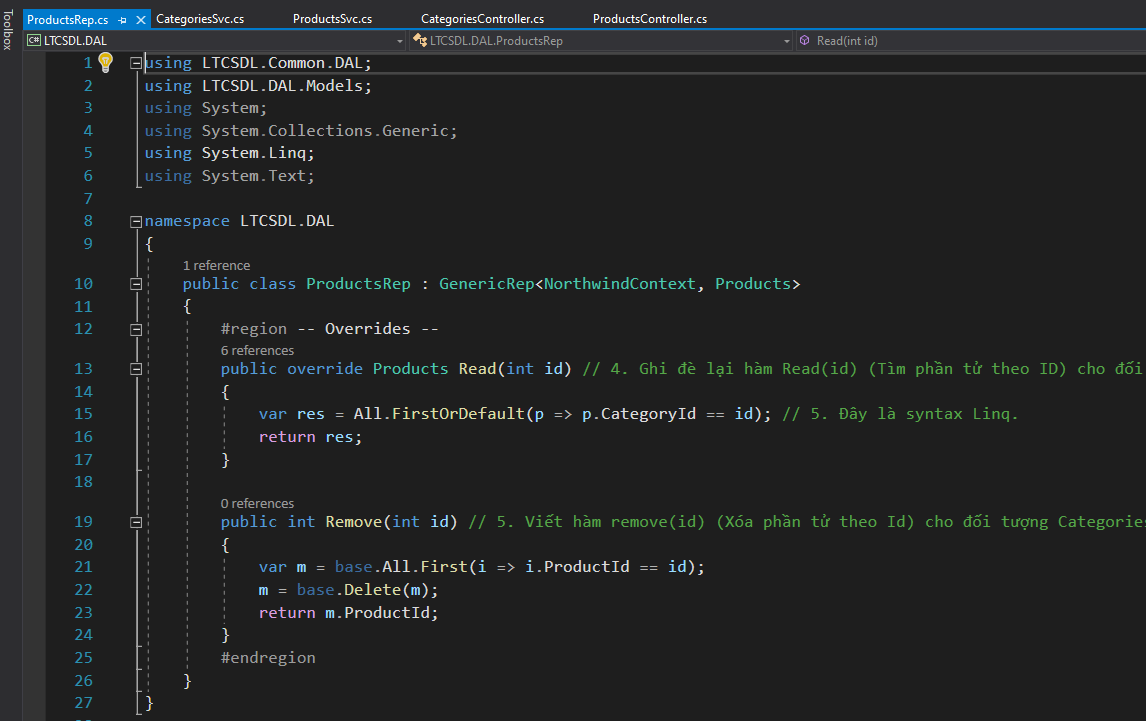
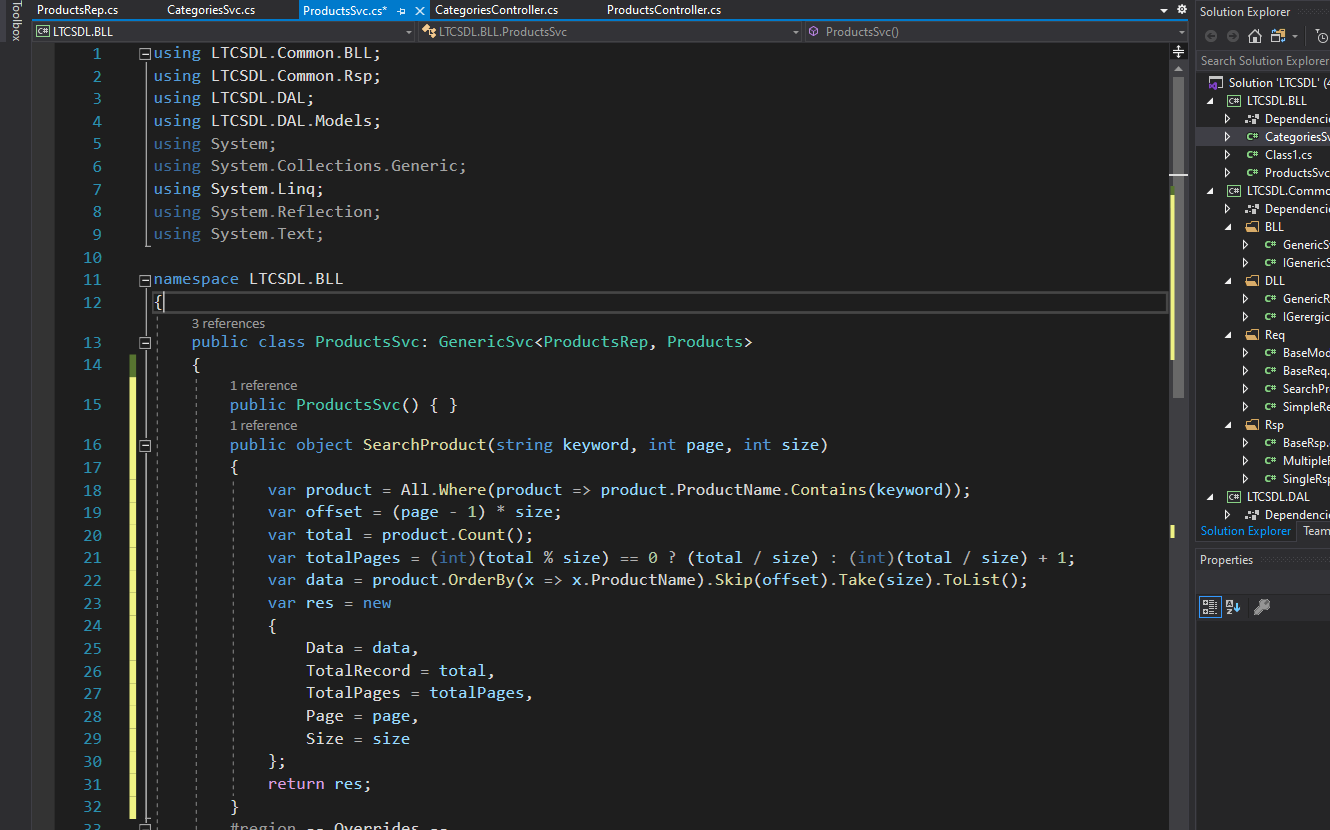
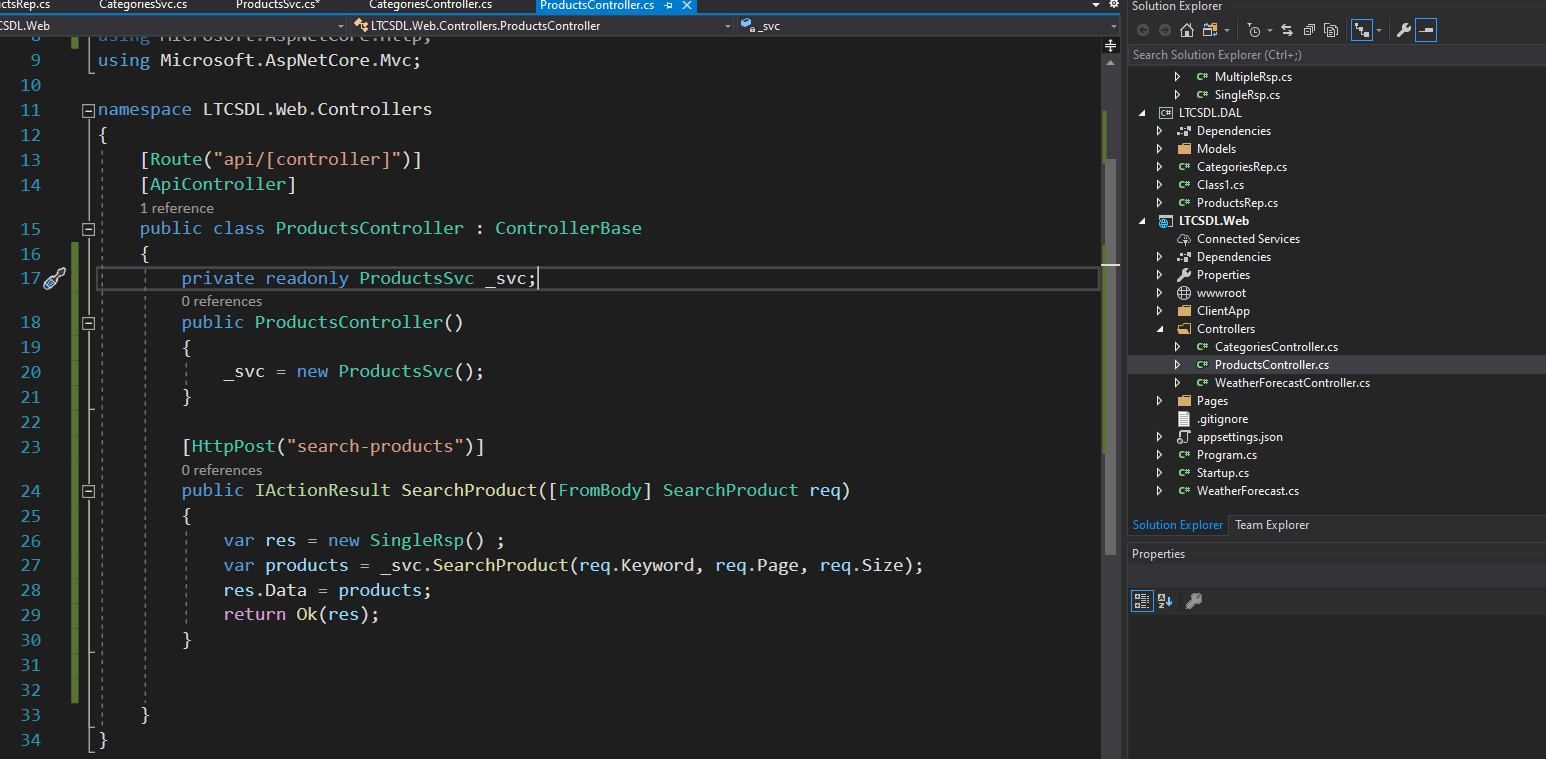
**1) Tạo Class SearchProductReq (Common layer)**

**2) Viết class ProductsRep (Data Access Layer)**

**3) Viết Class ProductsSvc BLL (Business Layer)**

**4) Viết Class ProductsController (Web)**

**5) Test Swagger**

* **Bước 1:** Tạo SearchProductReq trong Common Layer/ Req, đây là kiểu dữ liệu đầu vào của Req được gửi từ FE:
  +  => đây là kiểu dữ liệu đầu vào của 1 Req được gửi lên từ client. Sử dụng ở Client Controller.
* **Bước 2:** Tạo ProductsRep, để xử lý các việc như thêm xóa sửa database đối với bảng Products:
  +  => Trong ví dụ này thì ko viết gì ở DAL chỉ copy các phương thức mẫu thui, do thầy viết vậy chứ đúng là phải thao tác với database ở lớp này.
* **Bước 3:** Tạo ProductSvc.cs ở lớp BLL,
  + 
    - Line 15: Phải có hàm constructor() để khởi tạo giá trị cho Svc. Ta sẽ khởi tạo giá trị cho ProductsSvc ở controller để liên kết ProductsSvc và ProductsController.
    - Line 16: Tạo phương thức “**SearchProduct”,** phương thức này có phân trang line 22.
* **Bước 4:** Tạo ProductsController.cs ở lớp **WEB:**
  + 
    - Đầu tiên, phải khai báo và khởi tạo ProductsSvc để kết nối Controler với Service (line 17, 18);
    - Sau đó, tạo các API như trên để trả cho Client sử dụng Http service của angular để gọi và sử dụng.
* **Bước 5:** Test Swagger bình thường .

\*HƯỚNG DẪN VIẾT API CREATE/ UPDATE PRODUCT:

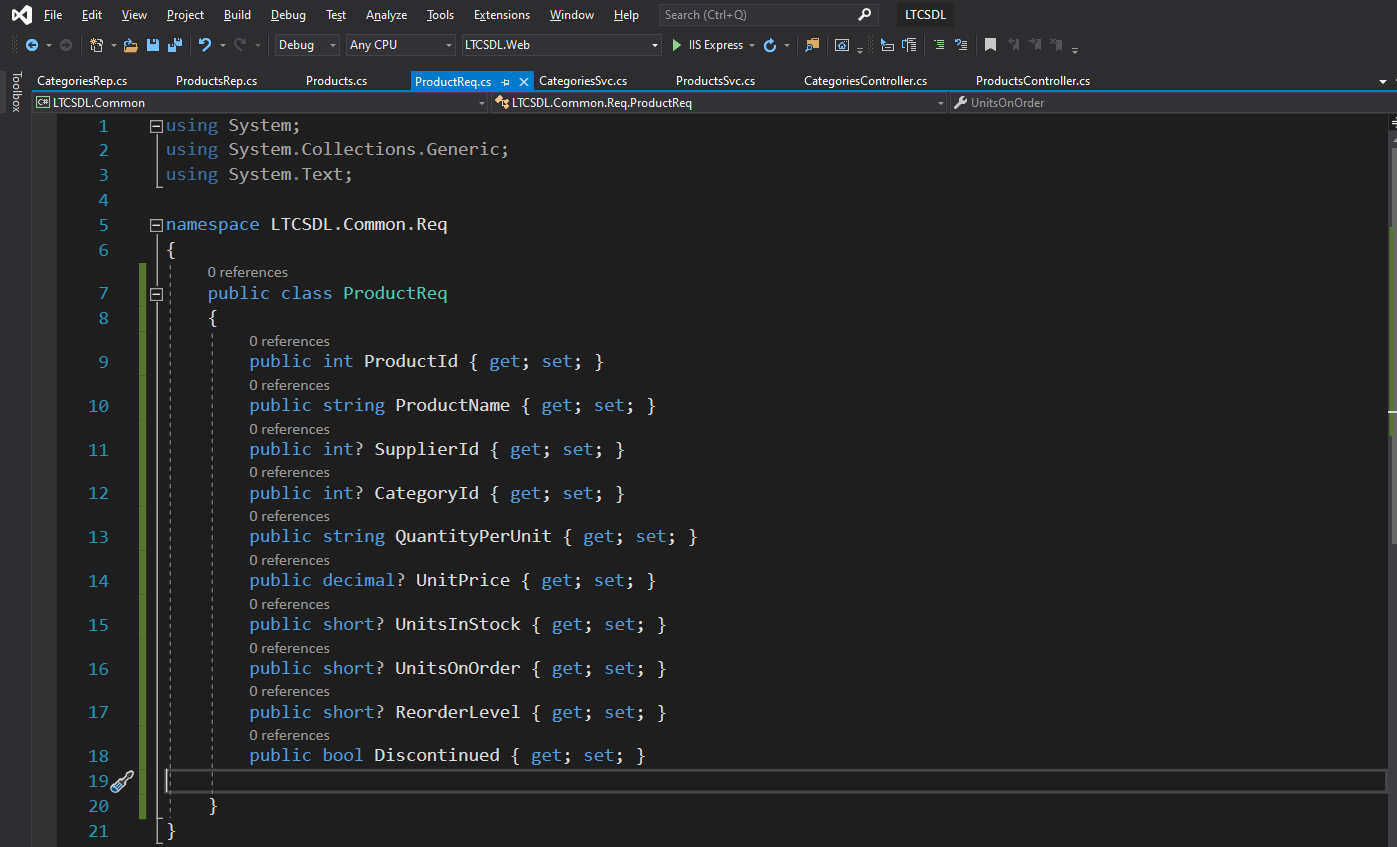
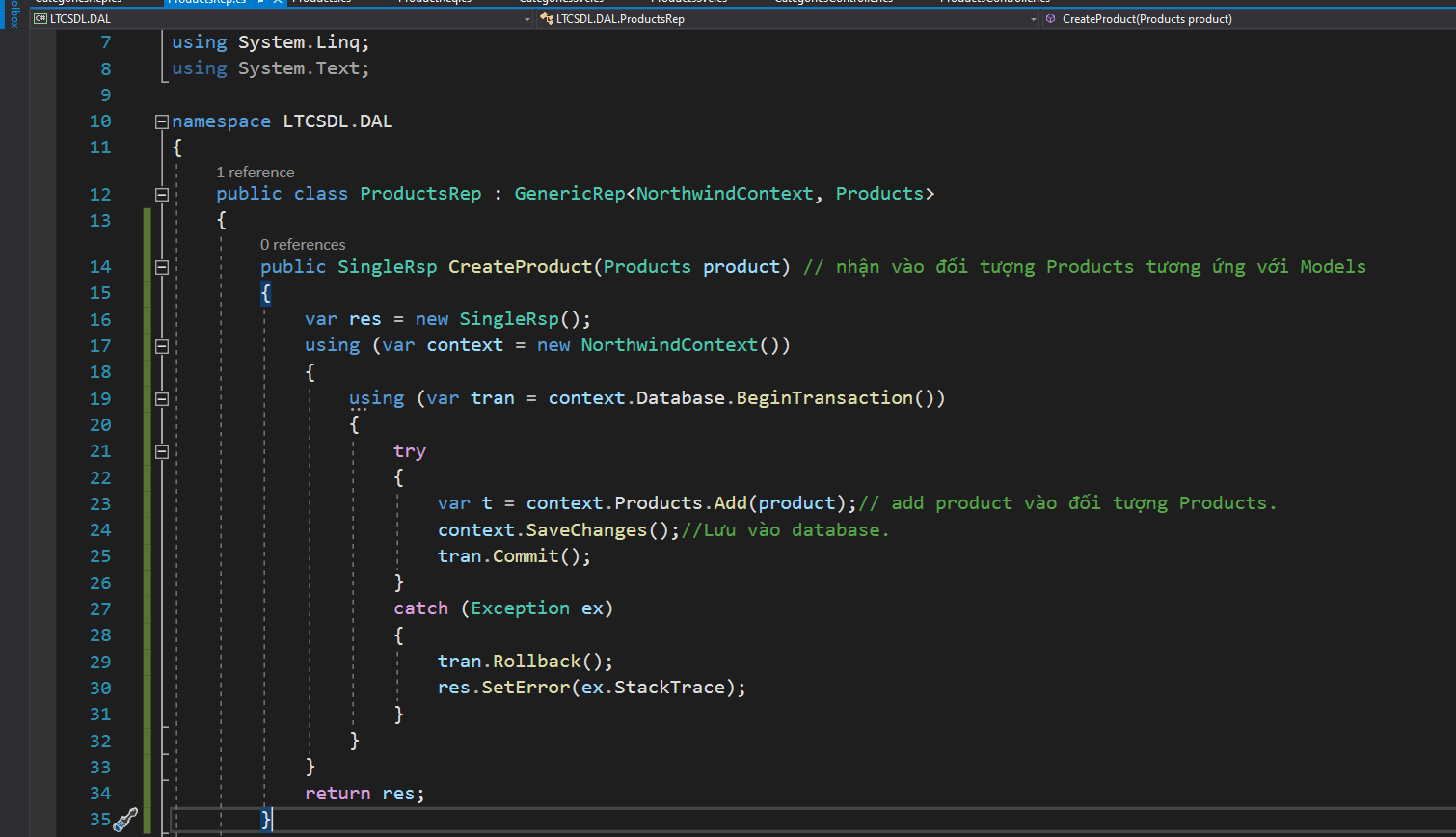
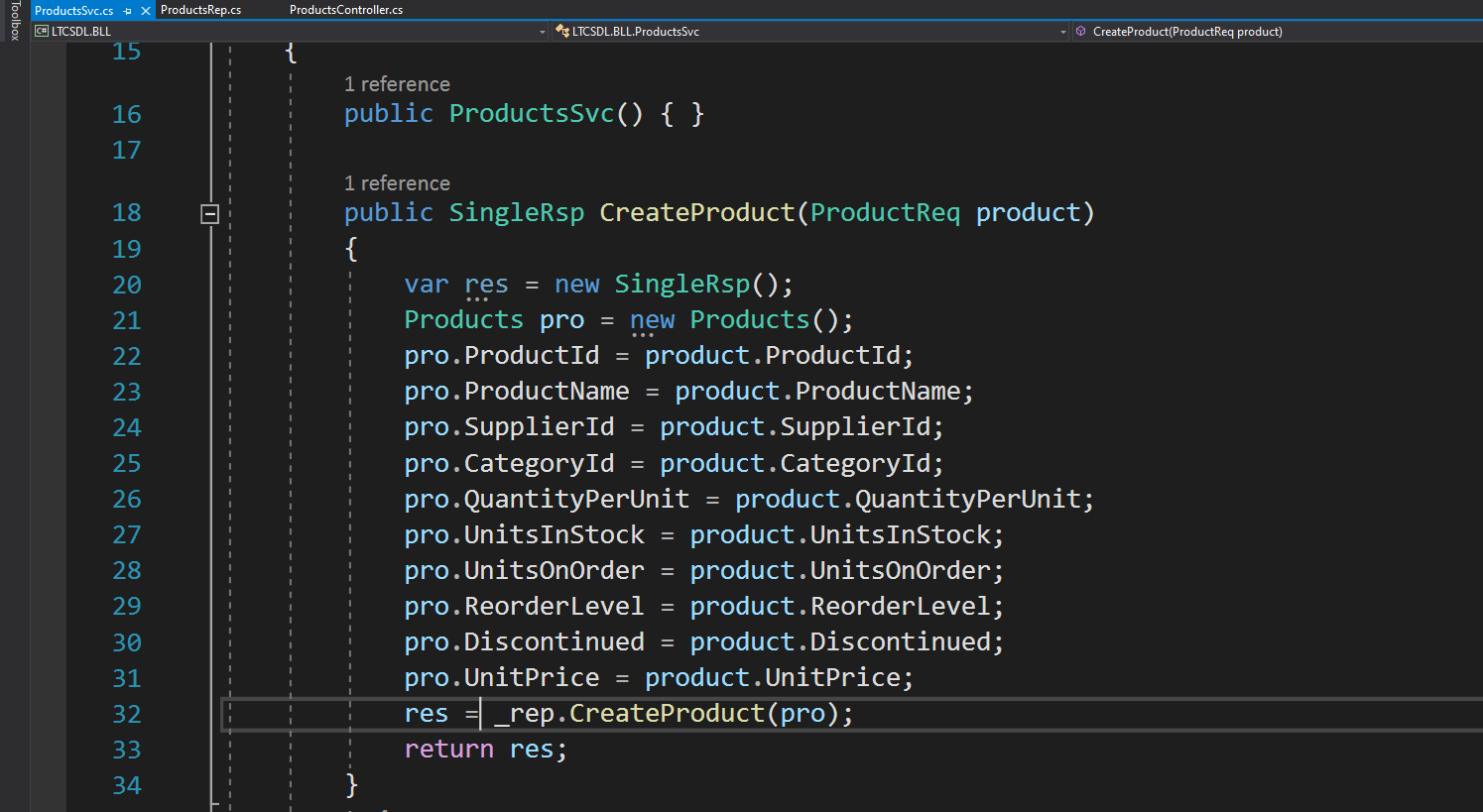
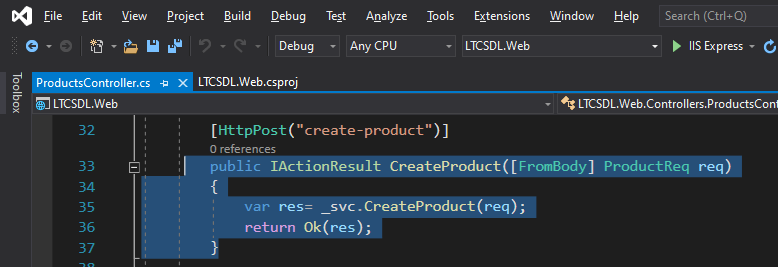
**1) Tạo Class ProductsReq (Common layer)**

**2) Viết function CreateProduct/UpdateProduct class ProductsRep (Data Access Layer)**

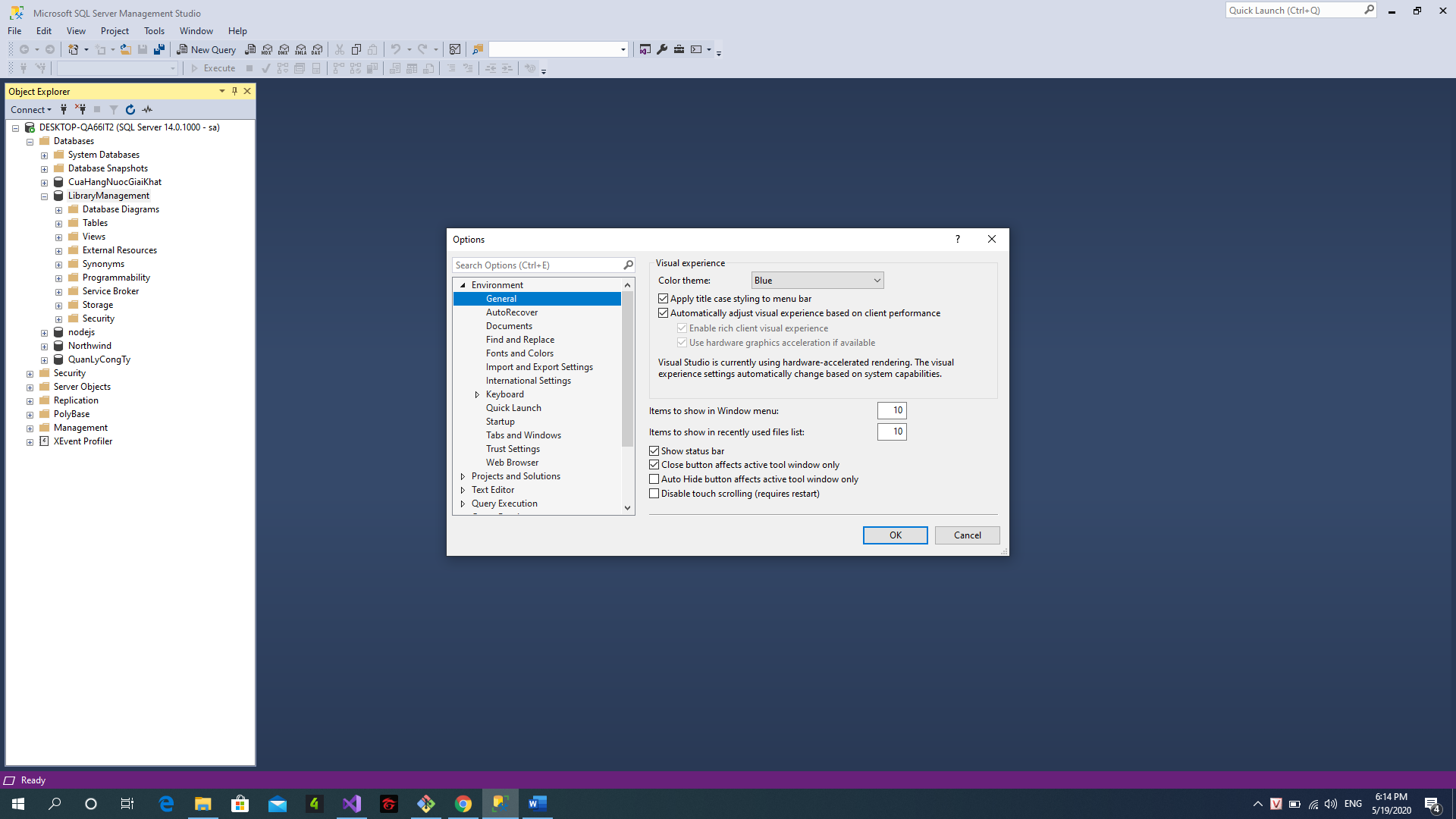
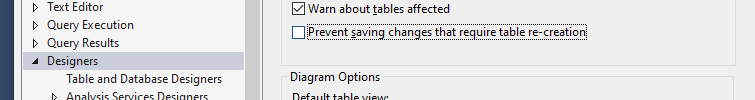
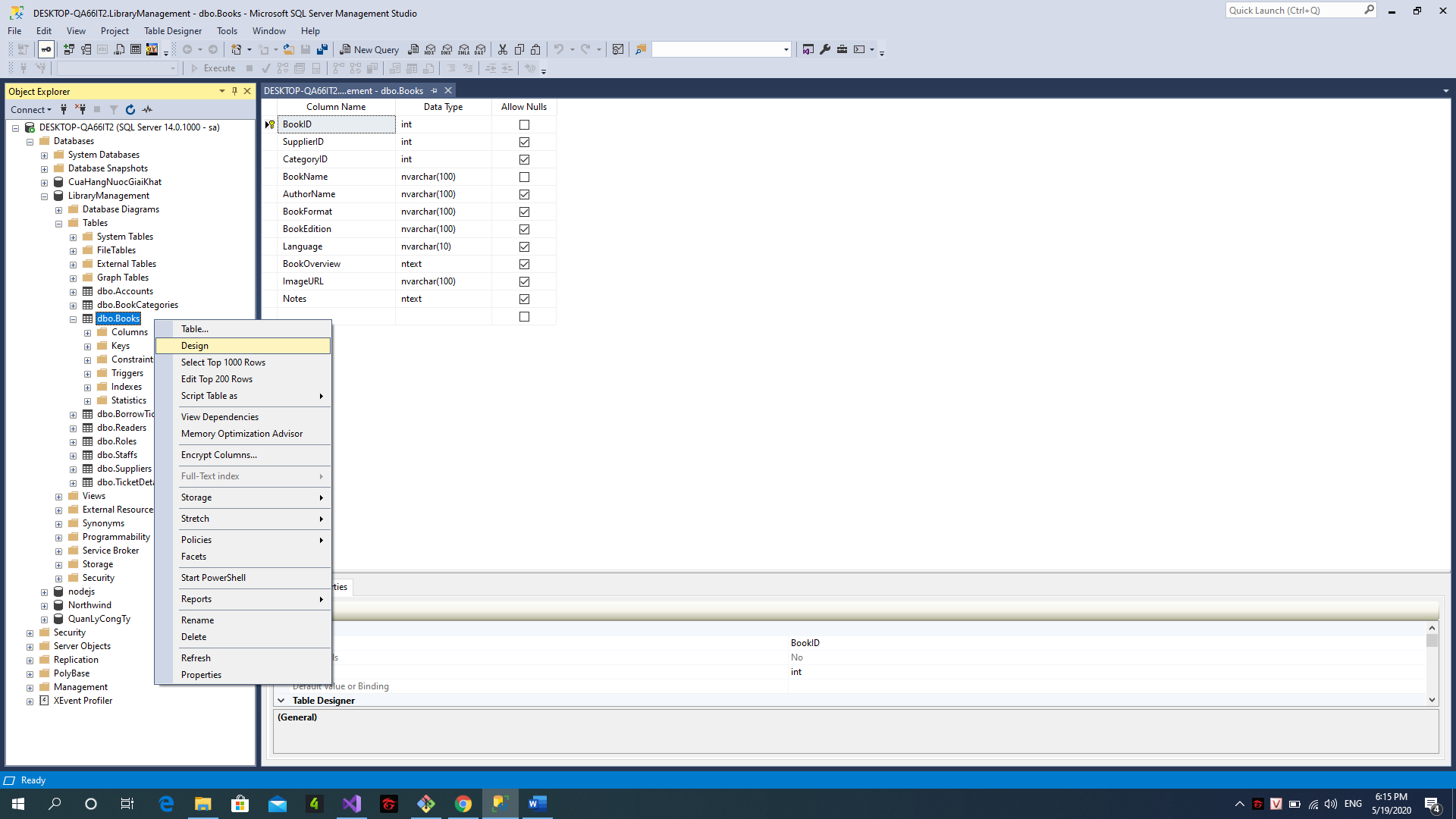
**3) Viết function CreateProduct/ UpdateProduct Class ProductsSvc BLL (Business Layer)**

**4) Viết function CreateProduct/ UpdateProduct Class ProductsController (Web)**

**5) Test Swagger**

* **Bước 1:** trong lớp Common Layer, tạo lớp ProductReq.cs là kiểu dữ liệu đầu vào của 2 API trên. Ở ví dụ này, nó giống với kiểu dữ liệu được lưu trong đối tượng Products :
  + 
* **Bước 2:** Tạo ProductRep.cs ở lớp DAL:
  + 
  + Kết nối NorthwindContext với model Products line 12.
  + Viết hàm CreateProduct().
* Bước 3: Tạo phương thức CreateProduct() trong ProductsService:
  + 
* **Bước 4:** Tại layer WEB/Controller/ProductsController.cs, viết Api CreateProduct():
  + 

-**Thay đổi databse và ánh xạ lên folder Models:**

1. Tools => Options: 
2. Designers => click bỏ “Prevent saving changes …..”
3. Vào bảng cần thay đổi (Book) => Designs => Cập nhật column => Save => Yes
4. Tạo mới 1 column ở cuối rồi mới kéo lên trên vị trí cần thay đổi => xóa column cũ => save.
5. Sau đó vào Project chỉnh sửa lại cho hợp lý dựa theo lỗi do VS báo.

**-GOOGLE CHART:**