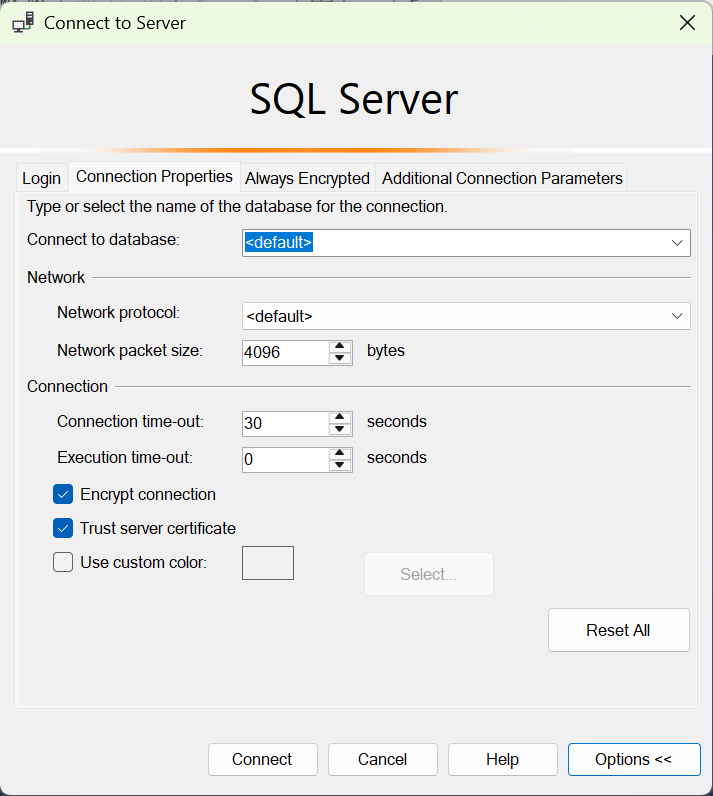
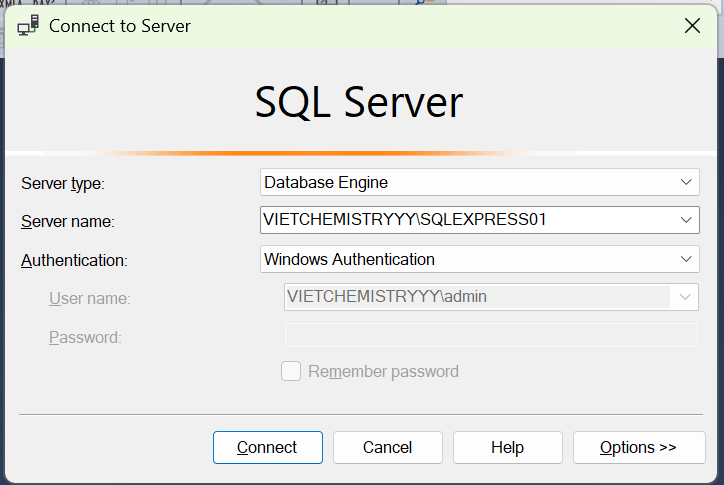
**I. Thực hiện cấu hình**

**1. Cấu hình SQL Server theo nguyên tắc bottom up**

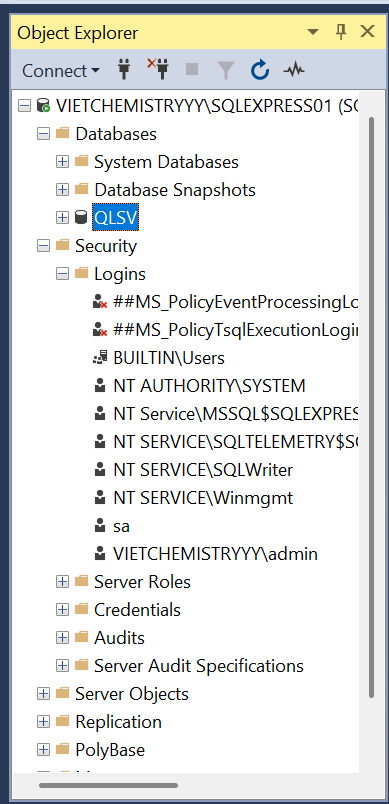
**-** Đầu tiên là browse tới máy của mình trước sau đó vào option tick chọn 2 Encrypt connection và Trust server certificate, nhớ tick cả lúc sau đăng nhập bằng tài khoản sa nữa nếu nó chưa tick

****

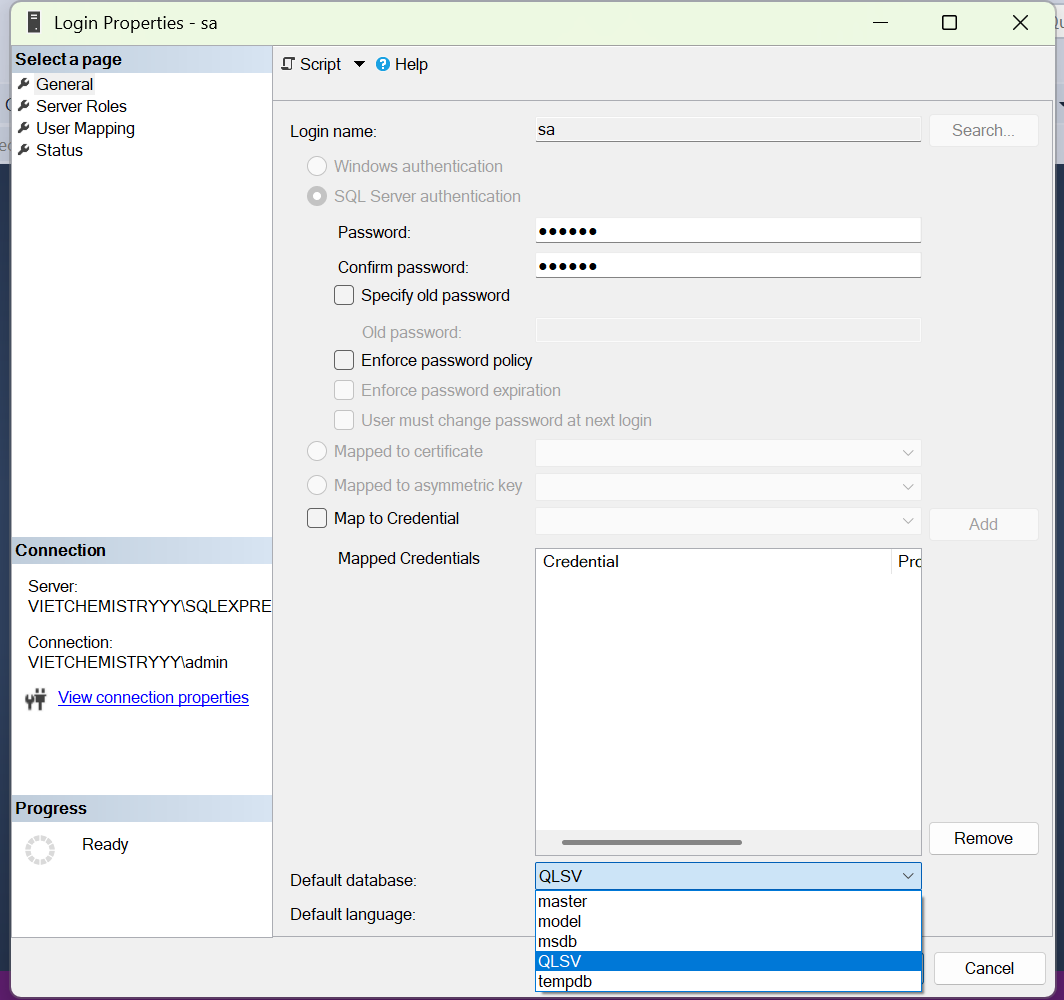
**-** Lần đầu đăng nhập thì cứ dùng tạm Windows Authentication để tạo database và để thay đổi setting đã rồi logout sau.

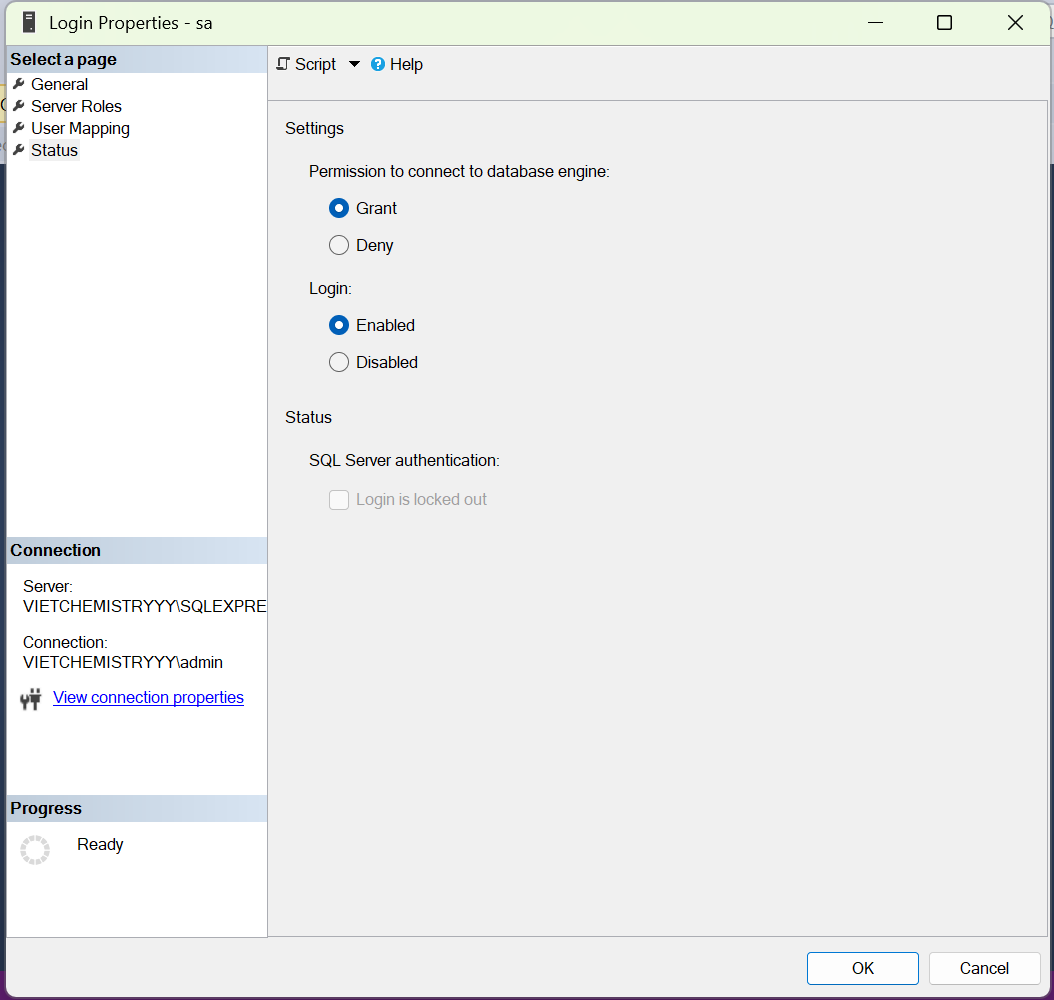
****

**-** Tạo database thủ công hoặc nếu đề cho trước thì cứ chạy query để nó tự gen. Ở đây gọi là bottom-up vì ta sẽ setting từ dưới lên, bắt đầu Login của Security, nếu như mới login lần đầu thì tài khoản sa sẽ hiện dấu “x” màu đỏ vì chưa được active, do ở đây đã active rồi nên nó hiện như này là chính xác

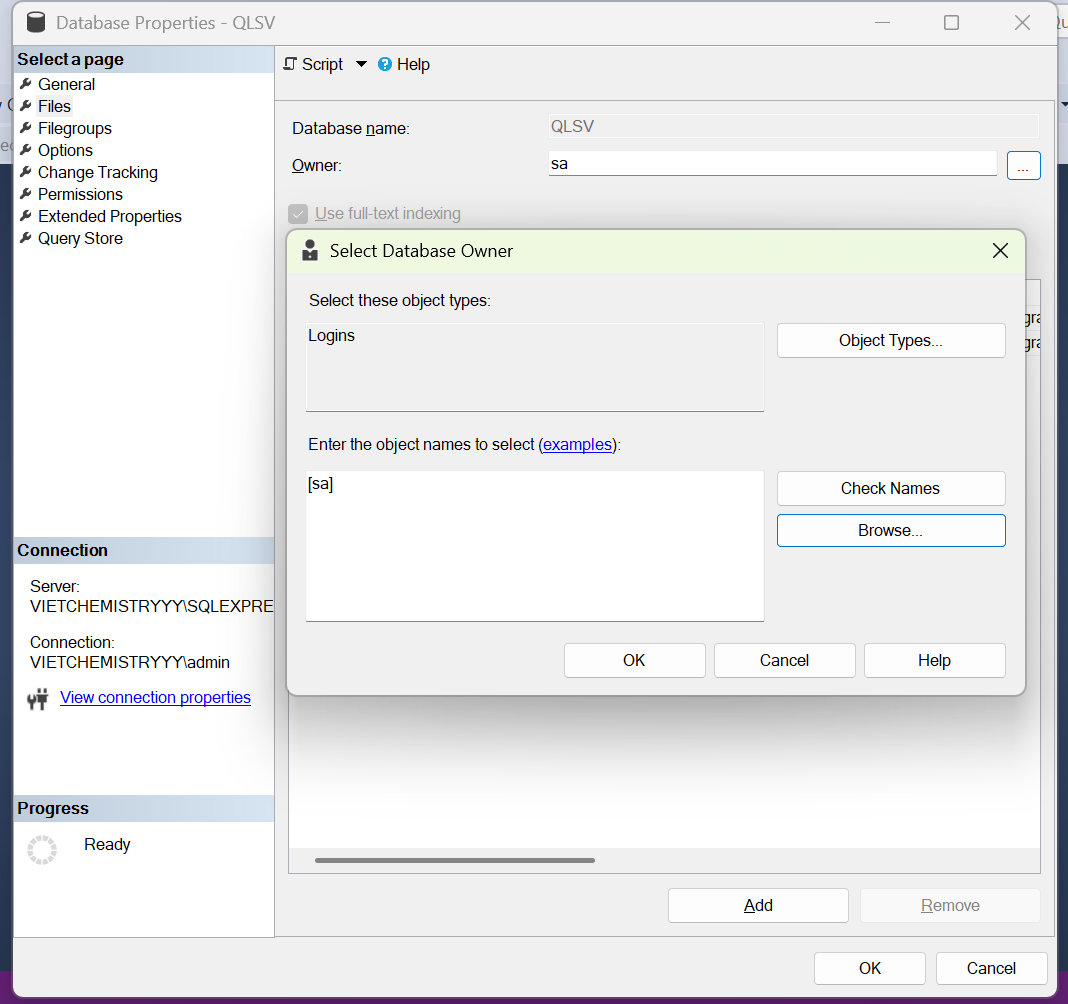
****

**-** Thực hiện active tài khoản sa, đầu tiên kích chuột phải vào sa và chọn properties. Thực hiện set lại mật khẩu cho tài khoản sa (123456) và bỏ hết tick ở trong các ô nếu có, chỉnh default DB link tới DB đã tạo trước đó, sau đó vào mục Status ở góc trái tick chọn Grant và Enabled 🡺 Ok để là xong việc active tài khoản sa (admin)

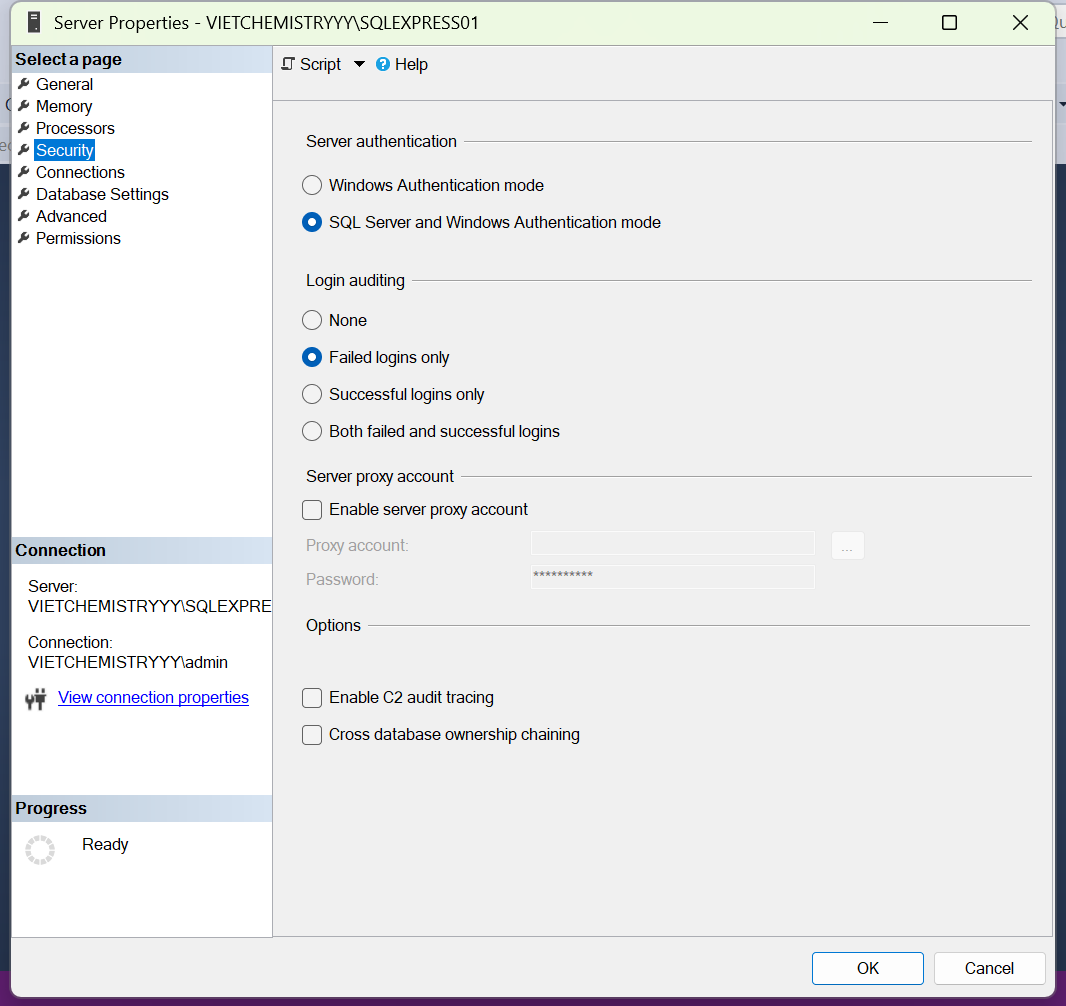




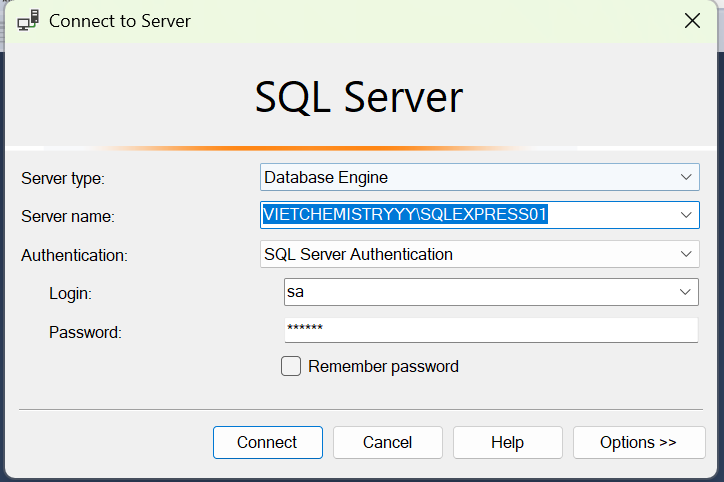
- Tiếp đến là setup cho tài khoản sa thành owner của DB vừa tạo, kích chuột phải vào DB chọn Properties 🡪 Chọn vào mục Files 🡪 Bấm vào nút 3 chấm để browse tài khoản sa đến DB 🡪 Ok done



- Sau khi chỉnh xong owner cho DB, cuối cùng là chỉnh kiểu đăng nhập cho server. Chuột phải vào server 🡪 Chọn Properties 🡪 Chọn Security 🡪 Tick chọn như trong hình 🡪 Ok done.



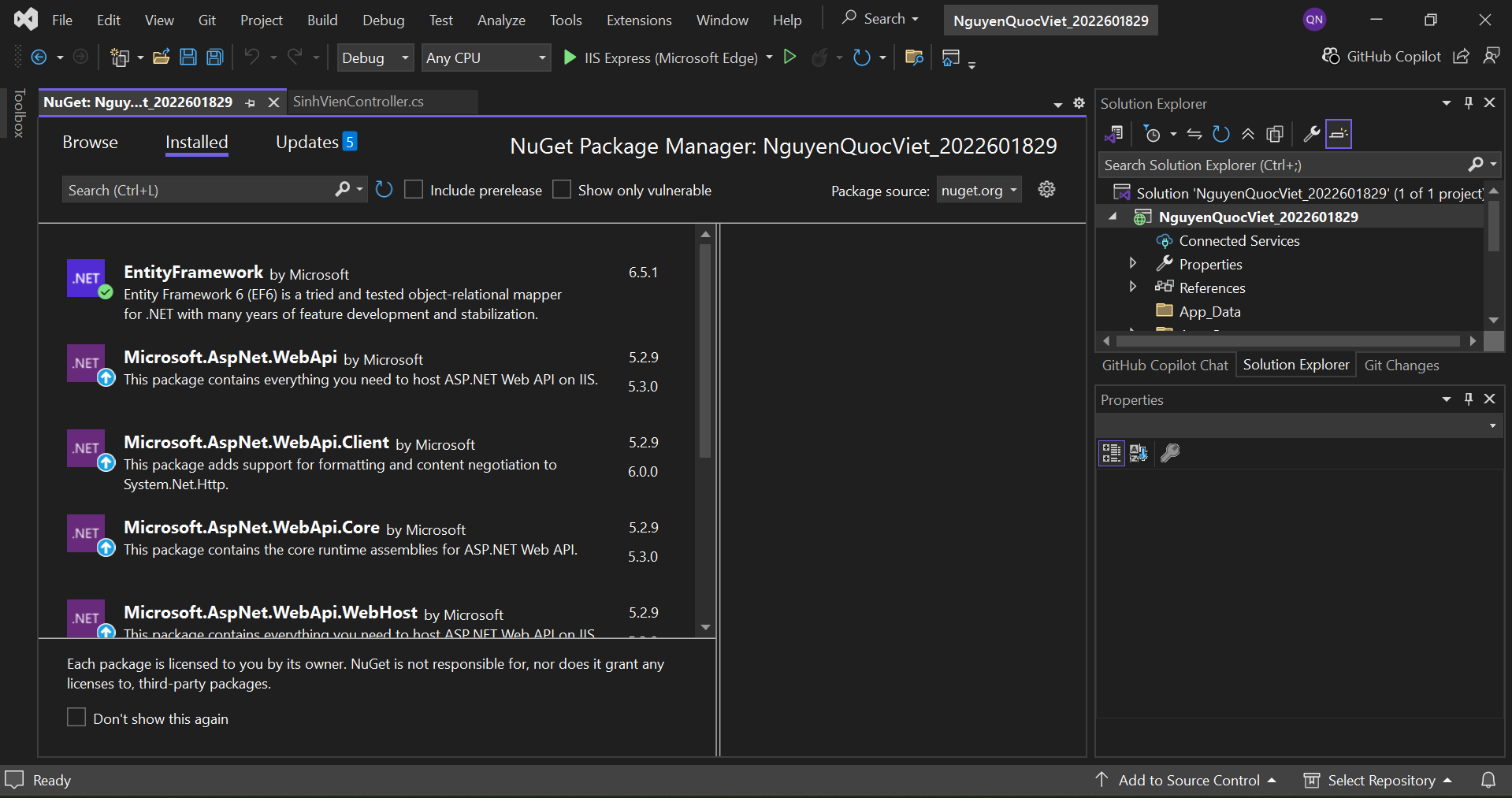
- Đăng nhập lại bằng tài khoản sa xem có lỗi DB không, trước hết phải restart lại server, disconnect, sau đó connect lại dưới tài khoản sa, ở phần này nhớ check lại phần Options xem đã tick đủ 2 cái lúc đầu chưa, nếu chưa có sẽ xảy ra lỗi về sau. Sau khi đăng nhập bằng tài khoản sa nhớ check lại xem DB có hoạt động chưa



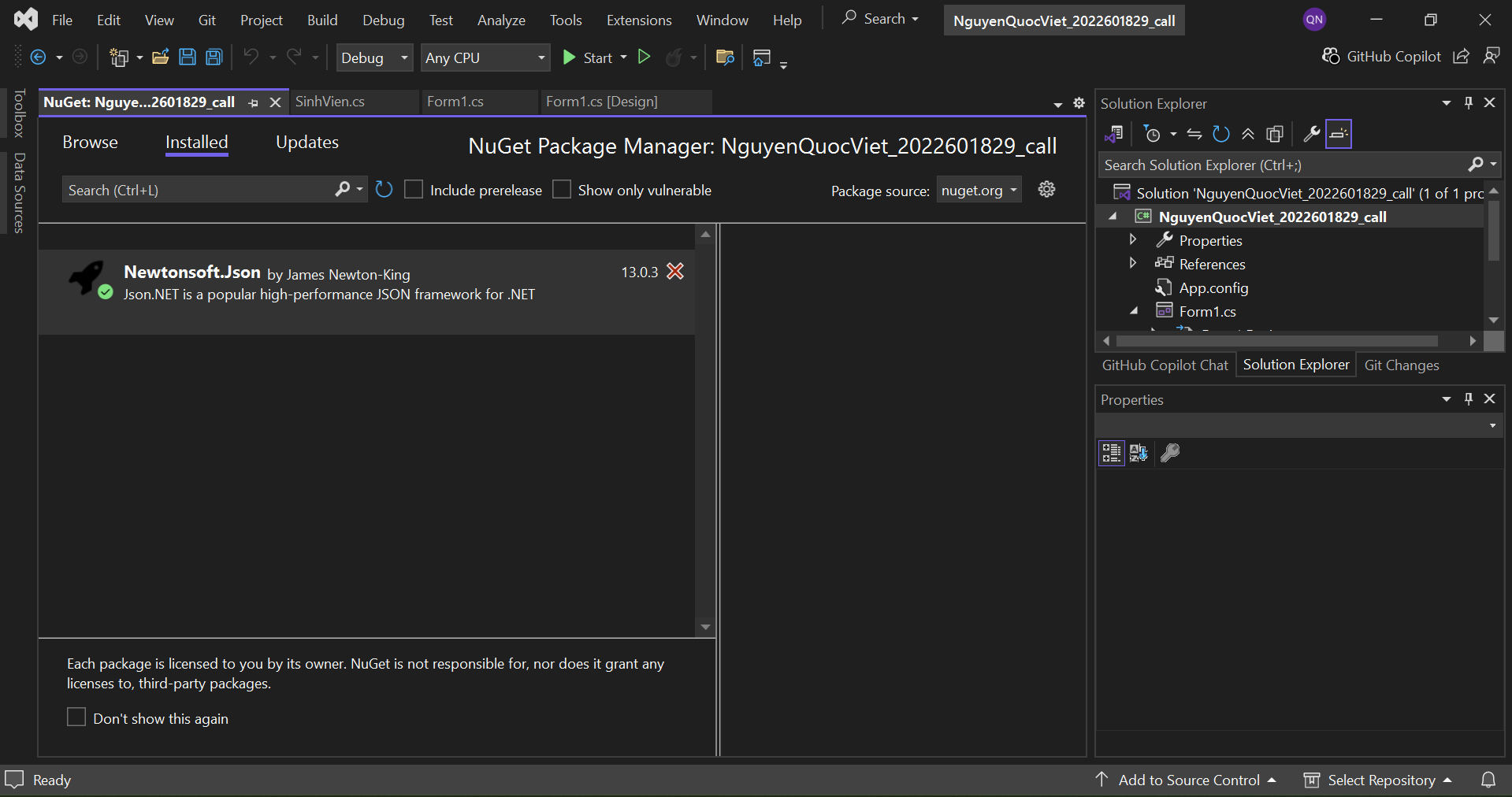
**2. Cài đặt package bằng NuGet trong Visual Studio**

**-** Ở đây ta cần tạo 2 proeject, một là project ASP.NET để viết API và 1 project .NET Winforms để thực hiện call API trước đó ta cần tải 1 số tài nguyên cần thiết.

+ Với project ASP.NET ta cần tải Entity Framework để có thể thực hiện thao tác với các dữ liệu trong database (mapping, LinQ,…). Cách tải: Chuột phải vào project vừa tạo 🡪 Chọn Manage NuGet Packages 🡪 Vào tab Browse tìm Entity Framework và Install 🡪 Check bên tab Installed xem đã tải thành công chưa.

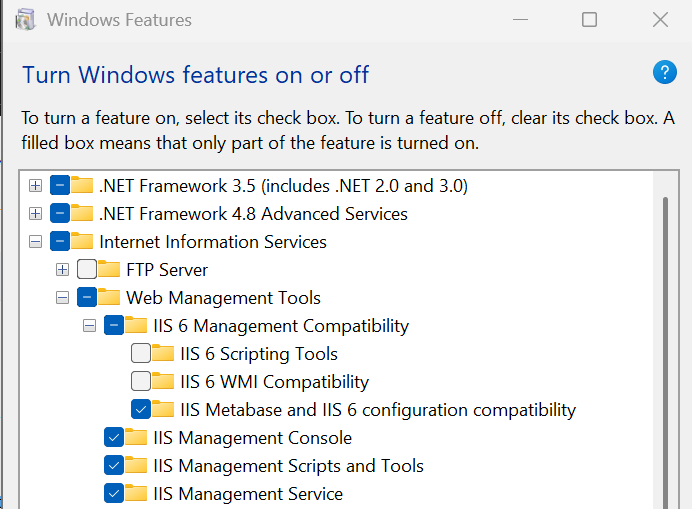


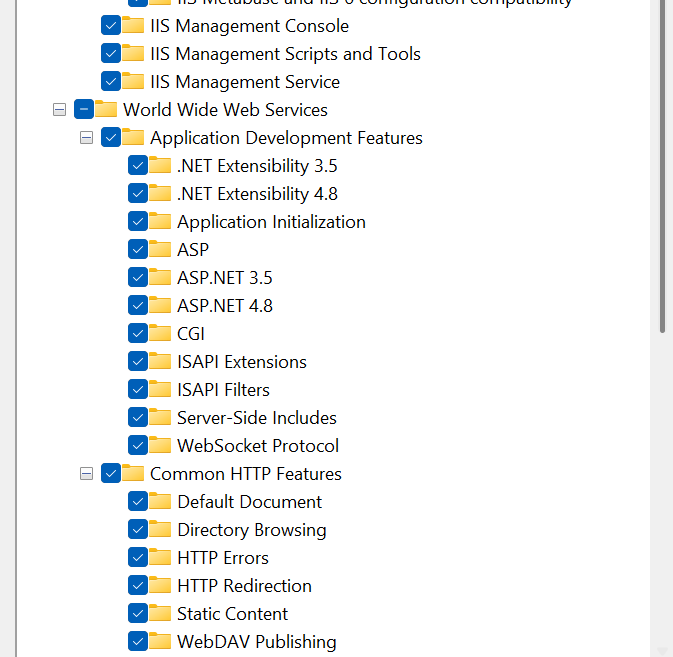
+ Với project .NET Winforms ta cần tải Newtonsoft để có thể thực hiện thao tác với JSON. Cách tải: Chuột phải vào project 🡪 Chọn Manage NuGet Packages 🡪 Vào tab Browse tìm Newtonsoft và kích chọn Install 🡪 Check bên tab Installed xem đã tải thành công chưa.

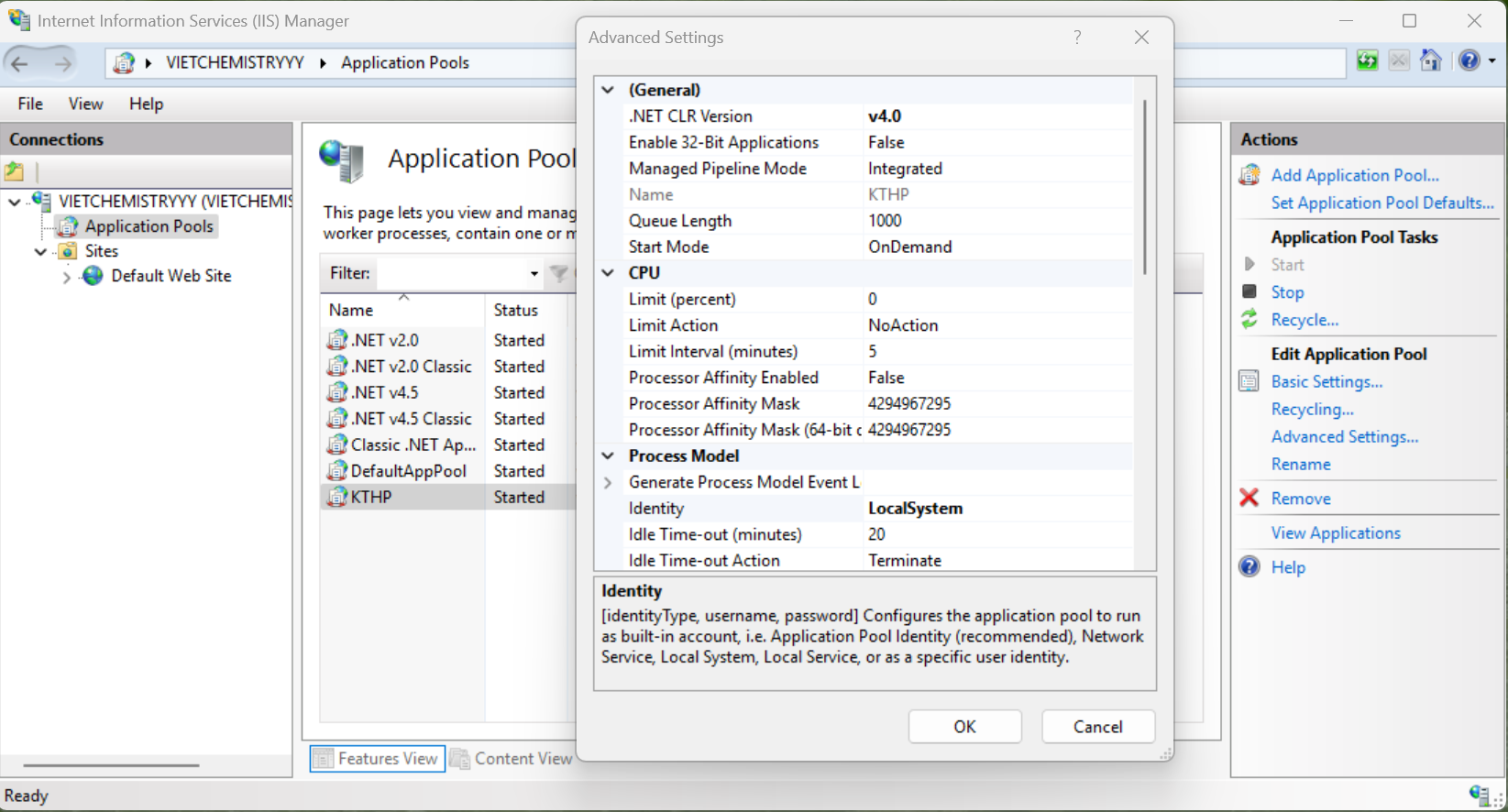


**3. Cấu hình IIS**

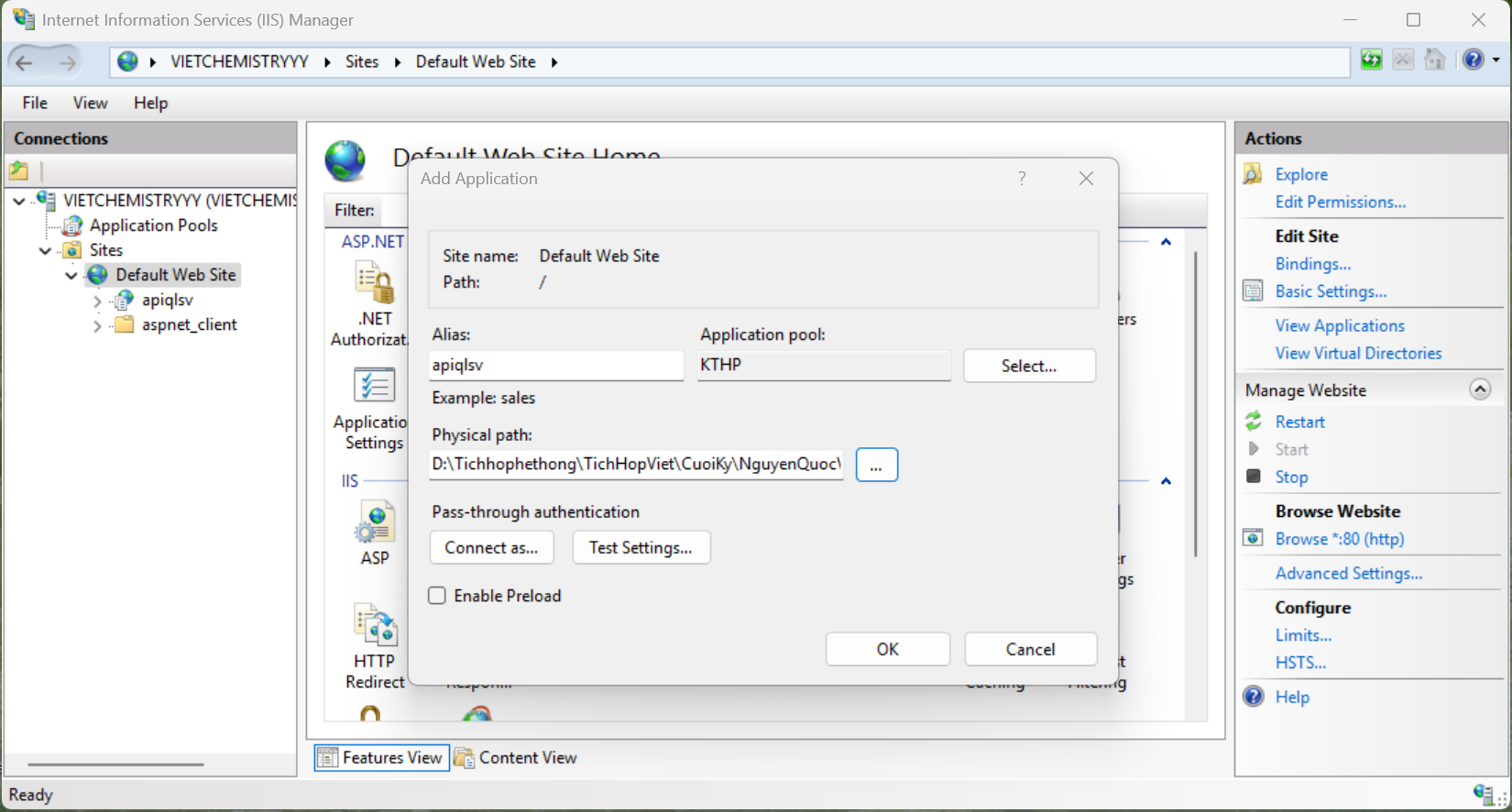
- Một setting cực kỳ quan trọng trong control panel. Mở control panel 🡪 Programs 🡪 Programs and Features 🡪 Turn Windows Features on or of (góc trên bên trái) 🡪 Check mục IIS (Internet Information Services) và tick đủ các mục như trong hình

****

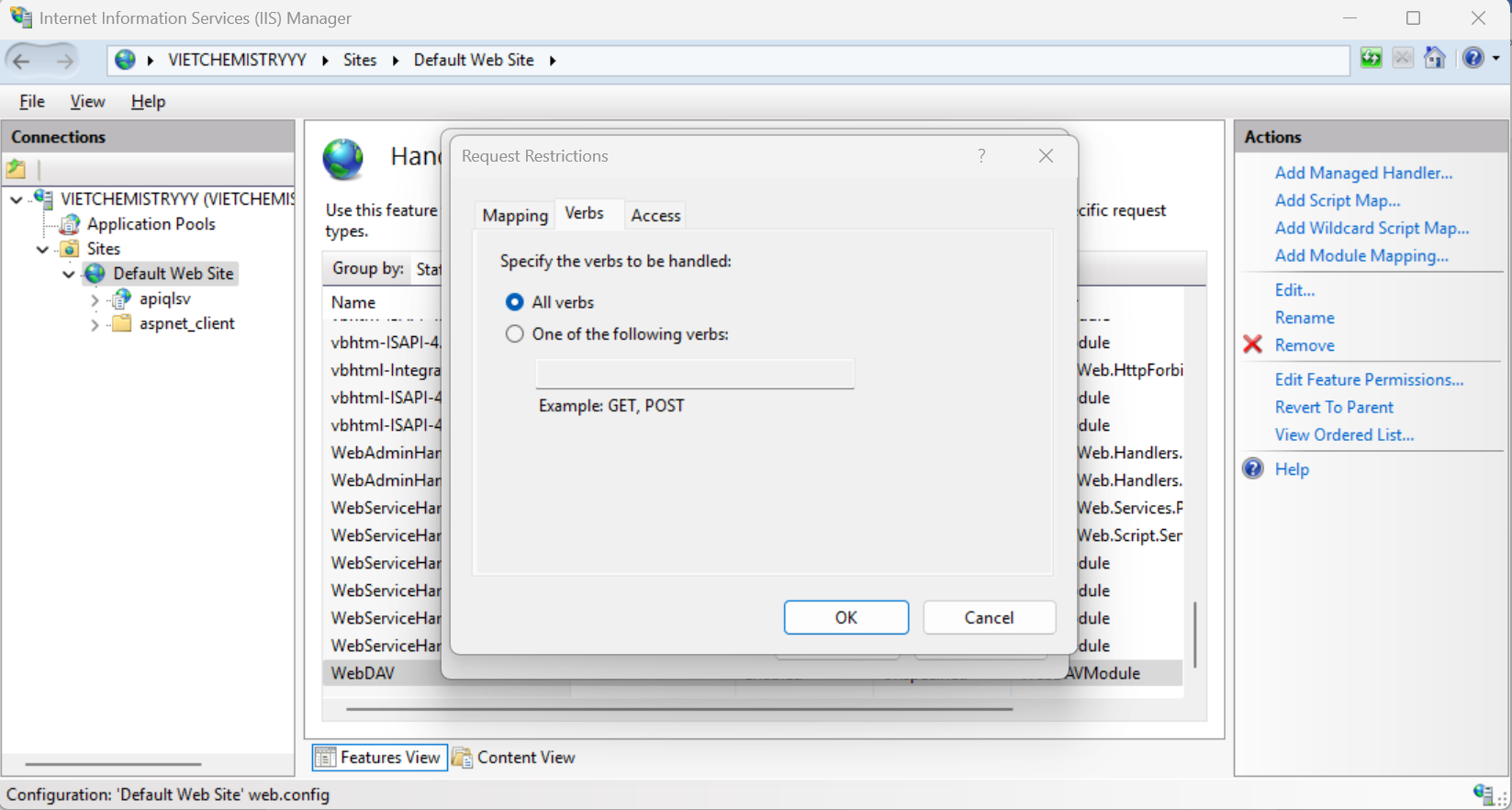
****

**-** Tiếp đến là cấu hình IIS, search trên IIS trên thanh công cụ. Các bước thực hiện tạo API. Chuột phải Application Pools 🡪 Add Application Pool 🡪 Tạo 1 Pool mới bất kỳ (Như trong hình là KTHP) 🡪 Chọn vào pool vừa tạo 🡪 Kích chọn Advanced Settings ở góc phải 🡪 Sửa Identity thành LocalSystem 🡪 Ok 

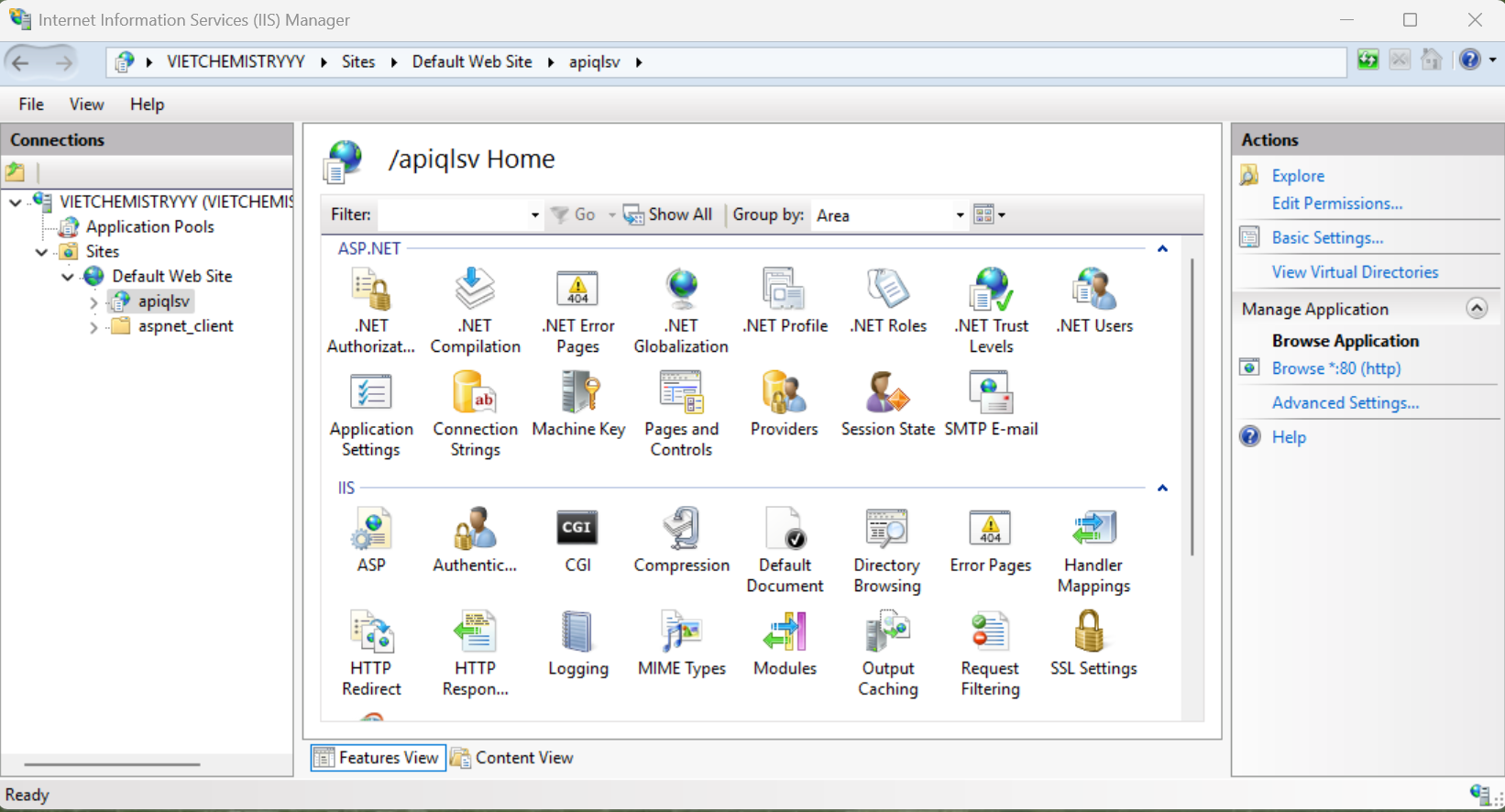
- Tạo web api. Chuột phải vào Default Web Site 🡪 Add Application 🡪 Đặt tên cho site, link tới pool vừa tạo ở trên và link tới đường dẫn vật lý chứa project API (Project ASP.NET) 🡪 Ok

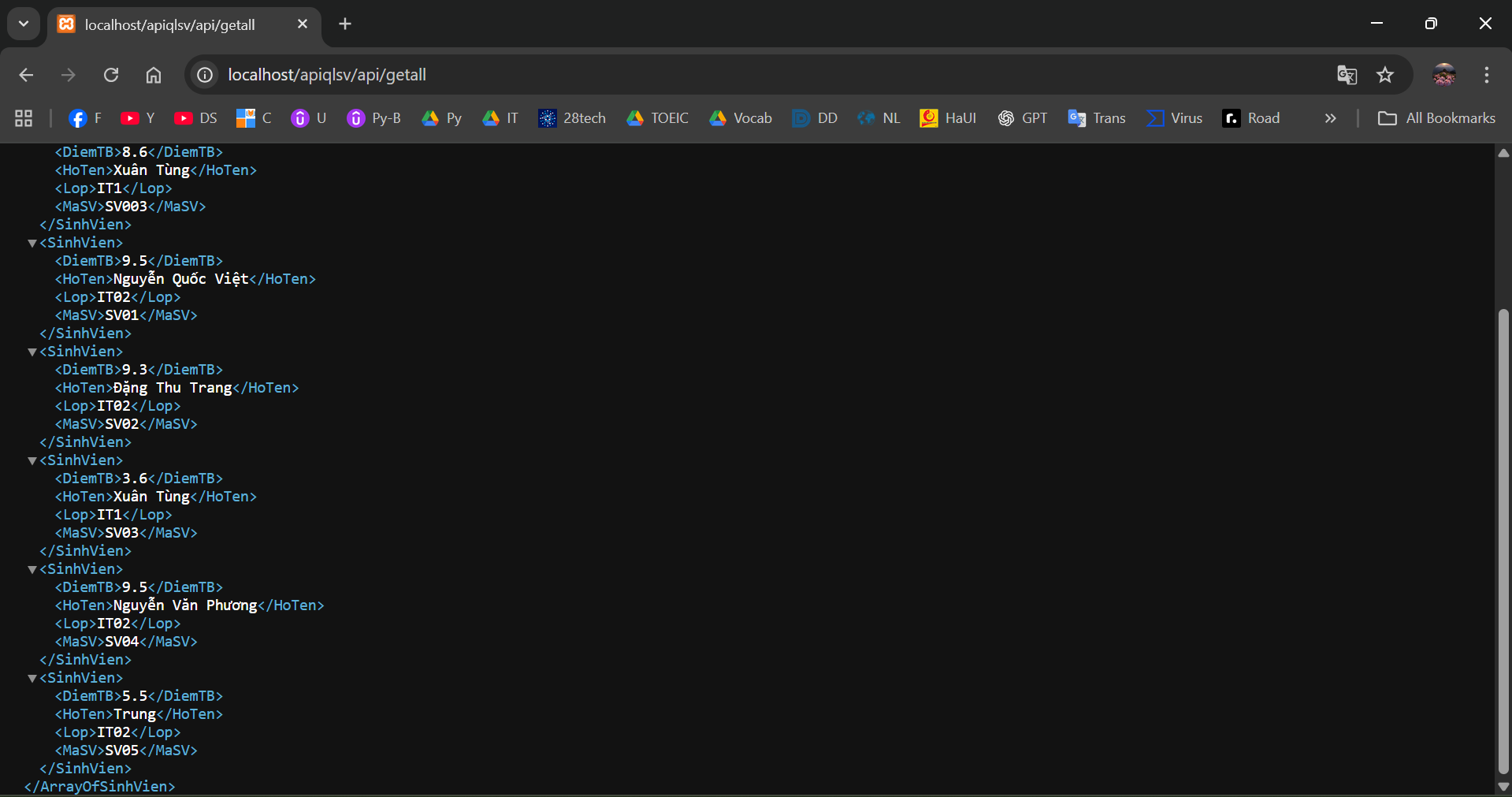


- Setting để có thể chạy được tất cả các API (Thêm, sửa, xóa,…). Vào Handler Mappings 🡪 Tìm chọn Web DAV 🡪 Request Restrictions 🡪 Verbs 🡪 Tick chọn All Verbs

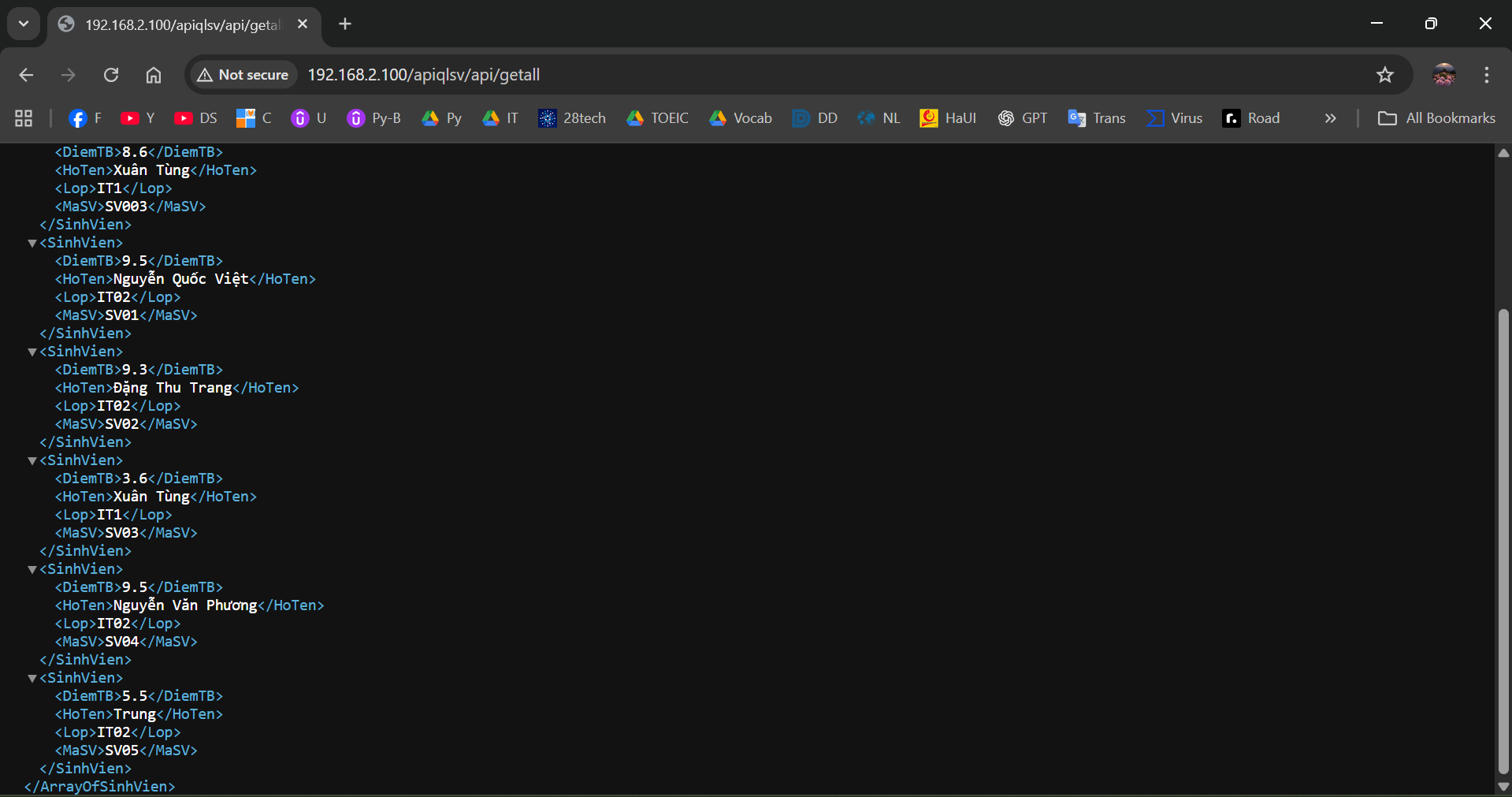


- Chạy thử site xem đã được chưa. Chọn Browse \*.80(http) góc phải. Chạy thử vài API xem ổn chưa, nhớ cú pháp <http://localhost/ten_site_da_tao/api/ten_api_can_goi>





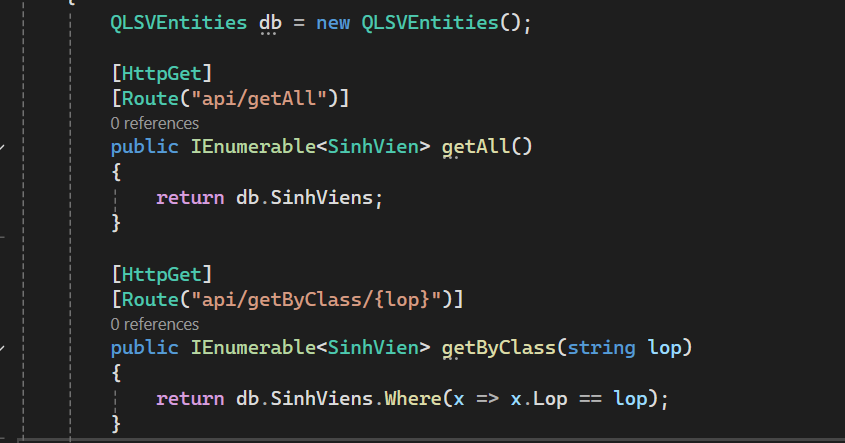
- Ở đây cũng có thể dùng ip của máy để thay cho localhost chạy cho nó flex 😊



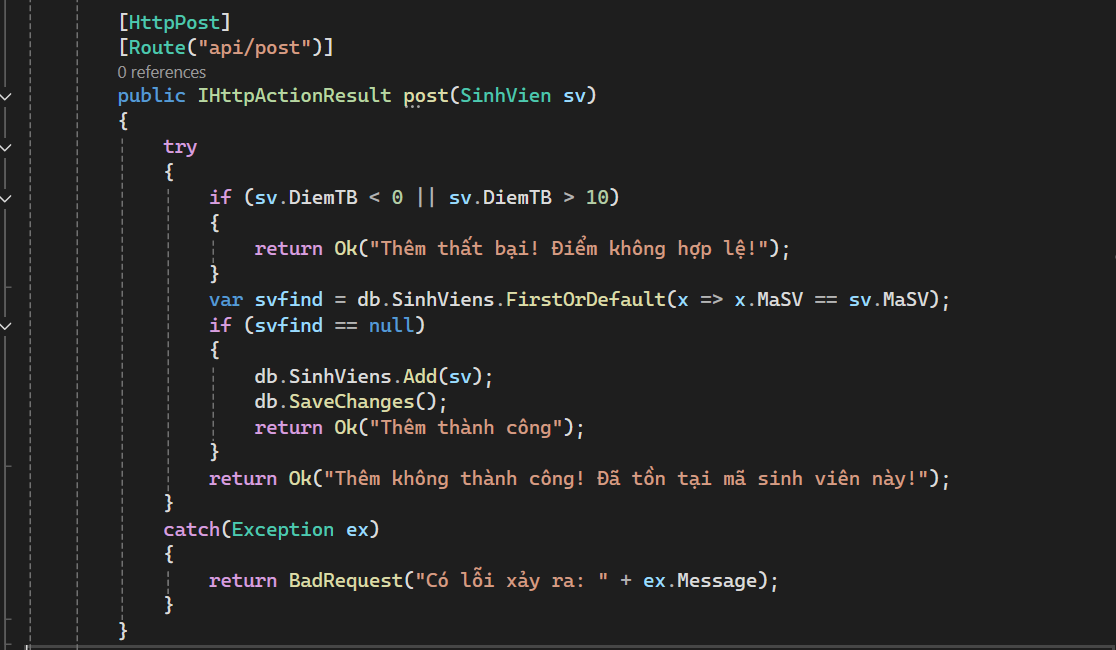
**Note**: Nên dùng Postman để test các API trước khi bước sang code Winforms để kiểm tra xem nó có hoạt động đúng hay chưa.

**II. Code API và Winform**

**1. API Get trả về toàn bộ danh sách & lọc theo 1 điều kiện bằng LinQ**

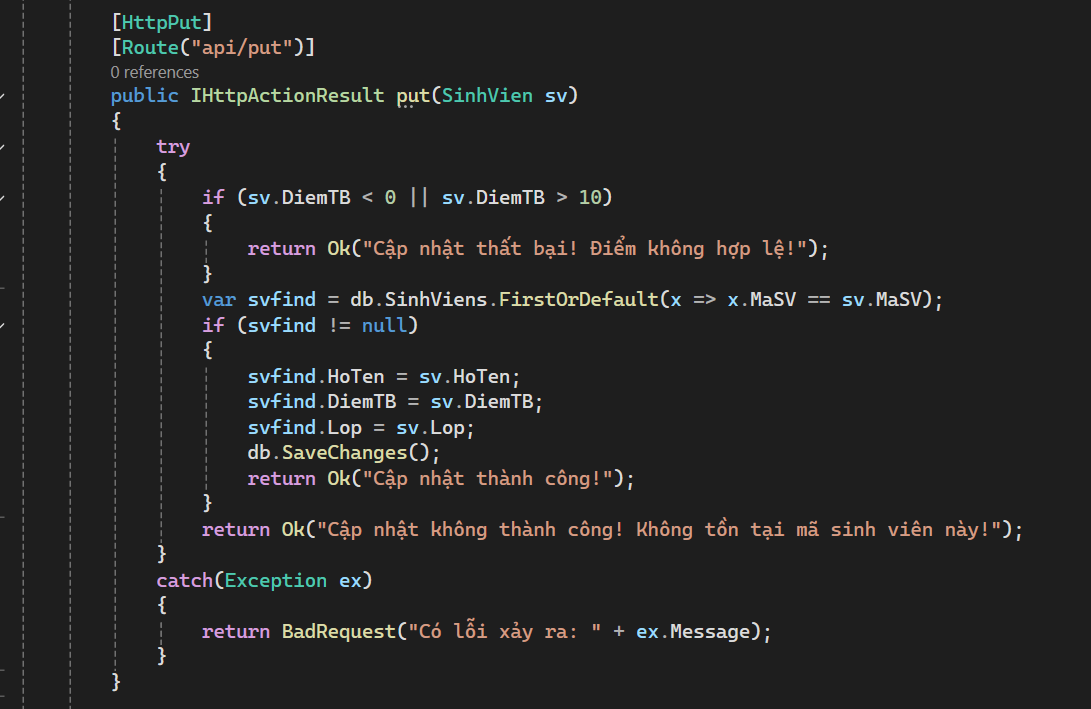


**2. API Post để thêm 1 phần tử**

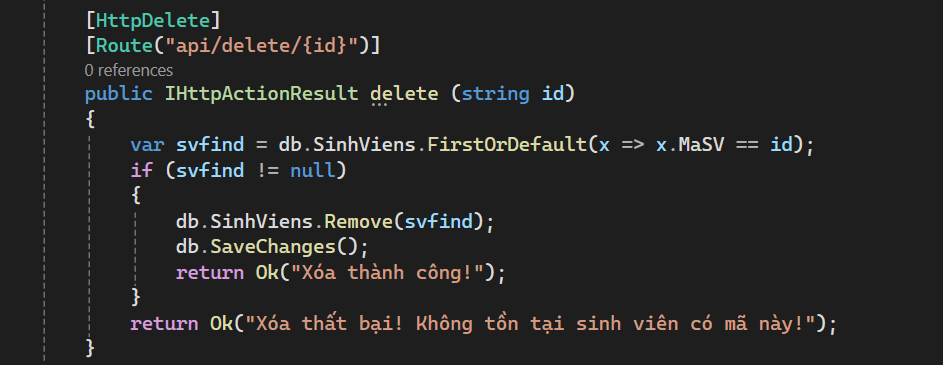
- FirstOrDefault trả về phần tử đầu tiên thỏa mãn điều kiện, nếu không thỏa mãn trả về Null 🡺 Nếu chưa tồn tại (tức là Null) thì thực hiện thêm, ngược lại thông báo đã tồn tại. 

**3. API Put để sửa 1 phần tử**

**-** Tương tự như với API Post, về mặt ý tưởng thì thêm và sửa chỉ khác nhau ở việc là có tìm thấy hay không tìm thấy. Nếu tìm thấy 🡺 Cho phép sửa, ngược lại đưa ra thông báo lỗi.



**4. API Delete để xóa 1 phần tử**

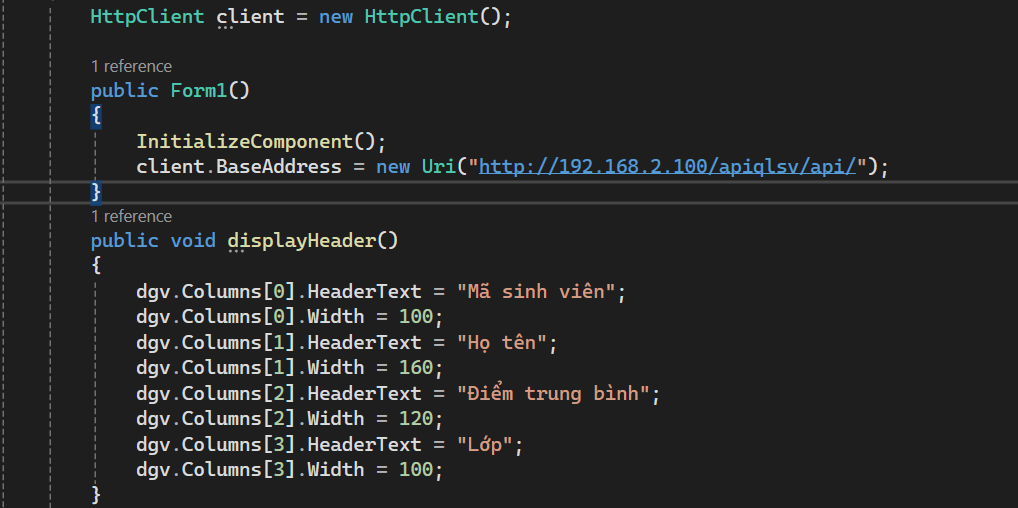
****

**5. Thao tác đầu tiên trên Winform**

- Tạo 1 client mới

- Link tới Uri của API sau khi đã set up xong ở phần cấu hình IIS đã đề cập ở trên. Ở đây có thể để IP của máy (dùng ipconfig trong cmd để check) hoặc để mặc định localhost cũng được.

- DisplayHeader để hiển thị header của các trường trong data grid view.

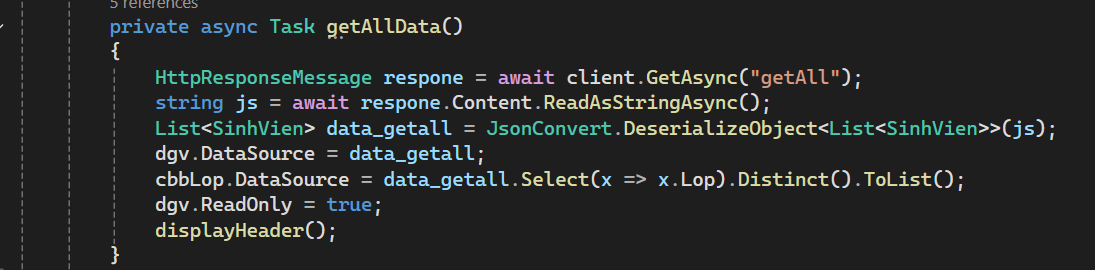
****

**6. Hàm lấy ra tất cả dữ liệu trong DB**

- Đây là hàm quan trọng nhất vì sai từ đây thì sẽ không lấy ra data trong DB được

- Một điều nữa là khi thực hiện các thao tác thêm sửa xóa thì cần gọi lại hàm này để load lại data nên mới nói nó cực kỳ quan trọng.

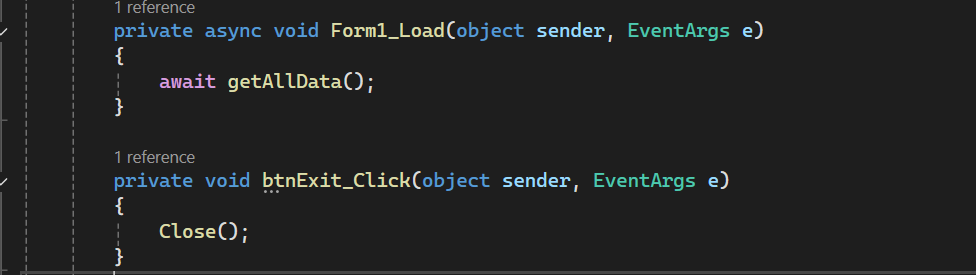
- Vì đề không yêu cầu cố định combobox hay textbox nên cứ để hết là textbox cho đỡ phải xử lý thêm 1 bước load data cho combobox



**7. Xử lý 2 thứ thiết yếu là load form và nút thoát**

**-** Load data lên data grid view khi chạy chương trình.

- Nút thoát để thoát khỏi form, điều này để tránh việc dừng chương trình thủ công.

****

**8. Nút thêm 1 phần tử mới**

- Xử lý trường hợp bỏ trống các trường thông tin bắt buộc  
- Xử lý kiểu dữ liệu (nếu cần)

- Ở đây vì dữ liệu được ta nhập từ bàn phím để load xuống data grid view nên các thuộc tính của đối tượng sẽ được gán bằng giá trị đã nhập, nó sẽ đối lập với sự kiện cell click (Get a record: Lấy ra bản ghi) vì khi đó ta sẽ click và 1 hàng trong data grid view để nó load lên textbox hoặc combobox tương ứng với các thuộc tính của đối tượng.

- Phần này sẽ dùng đến Newtonsoft.Json để chuyển đối tượng thành chuỗi JSON, sau khi thực hiện encode thì sẽ gửi request POST bất đồng bộ đến endpoint là đường dẫn của API ta đã viết ở trên có Route là “post”, send là body của request (JSON dữ liệu của đối tượng) 🡺 Kết quả từ server sẽ lưu vào biến respone và được thông báo lên màn hình dưới dạng string.

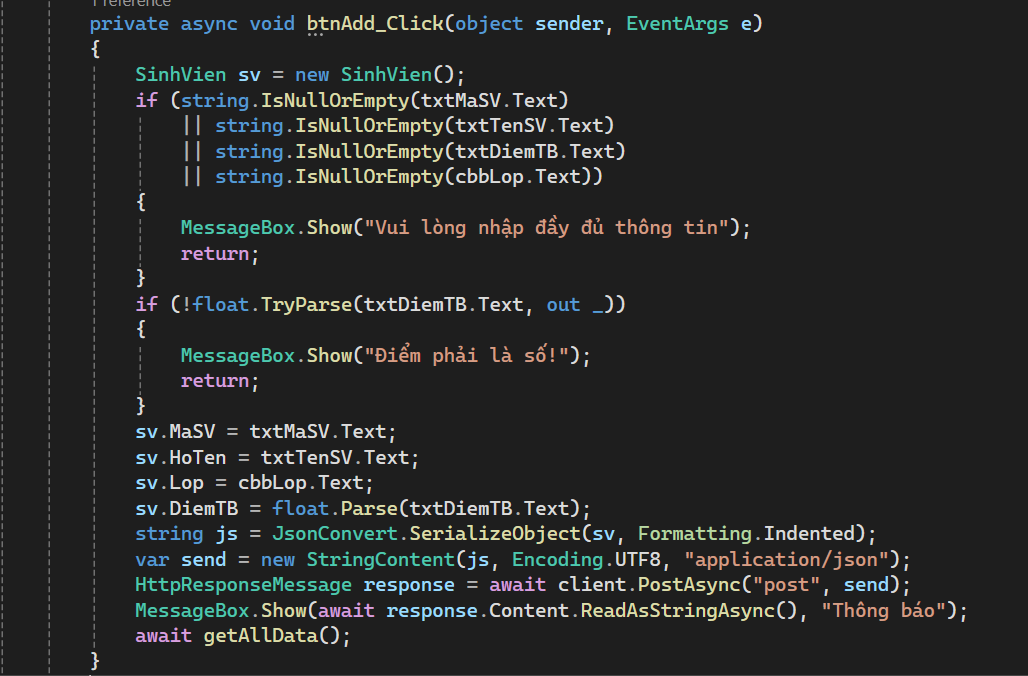
- Sau khi thêm thì gọi lại hàm getAllData() để load lại dữ liệu ta đã thay đổi, vì nó làm hàm bất đồng bộ nên cần await.

\* *Giải đắp thắc mắc*:

+ Nếu không dùng async thì code sẽ chạy kiểu: Người dùng bấm nút ➝ Gửi request ➝ Đợi server trả về ➝ UI treo cứng ➝ Hiện kết quả

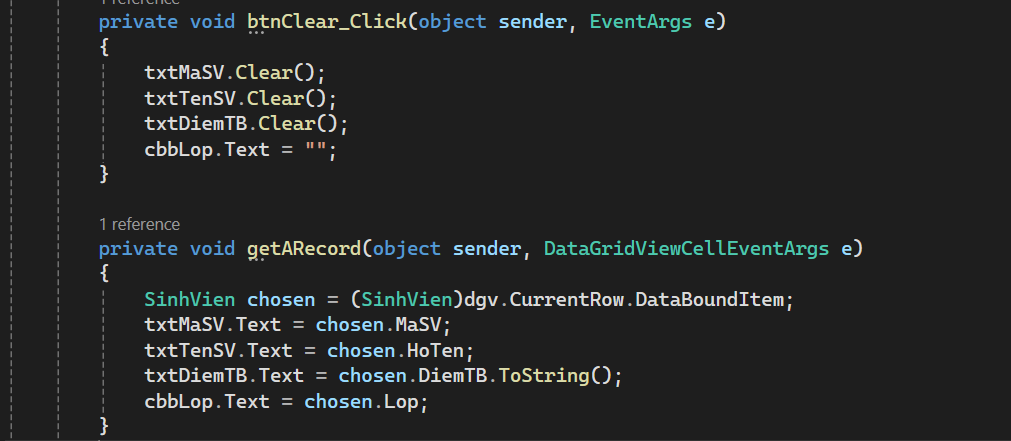
+ Nếu dùng async thì code sẽ chạy: Người dùng bấm nút ➝ Gửi request ➝ UI vẫn chạy bình thường ➝ Khi có kết quả thì tự động nhảy vào đoạn code tiếp

* Tóm lại async giúp cho Winforms không bị treo khi call API.

****

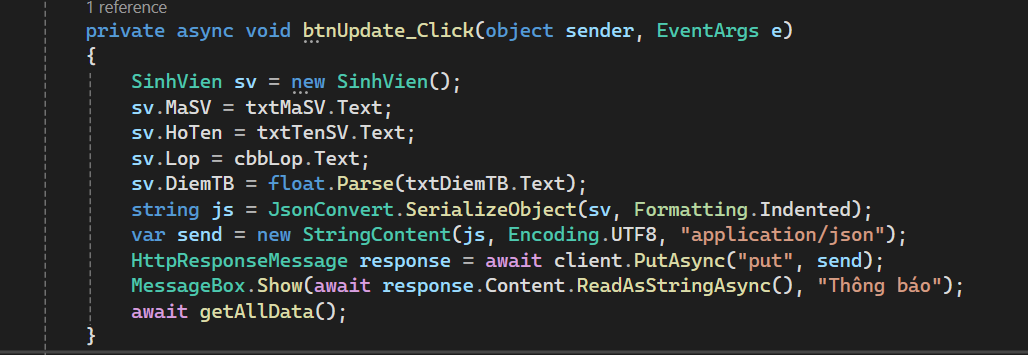
**9. Nút clear thông tin trên box và sự kiện cell click**

**-** Như đã đề cập ở trên sự kiện cell click này trái ngược với việc thêm 1 phần tử vì ta cần đưa data từ data grid view lên trên textbox hoặc là combobox nên cần phải gán giá trị từ cell được chọn cho textbox hoặc combobox.

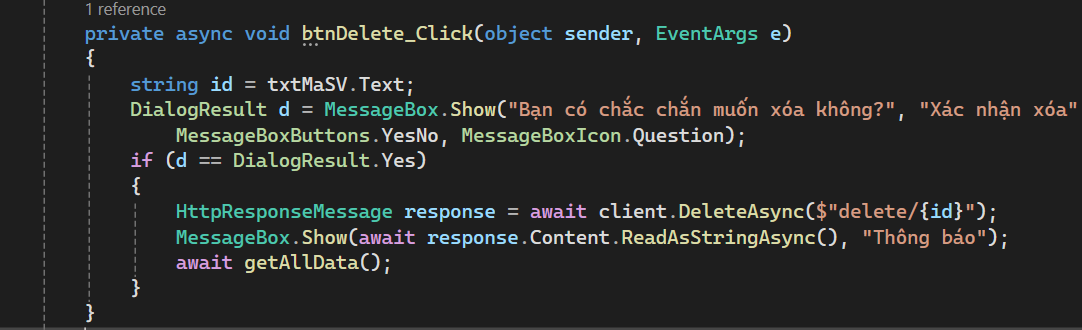
****

**10. Nút sửa thông tin của 1 phần tử**

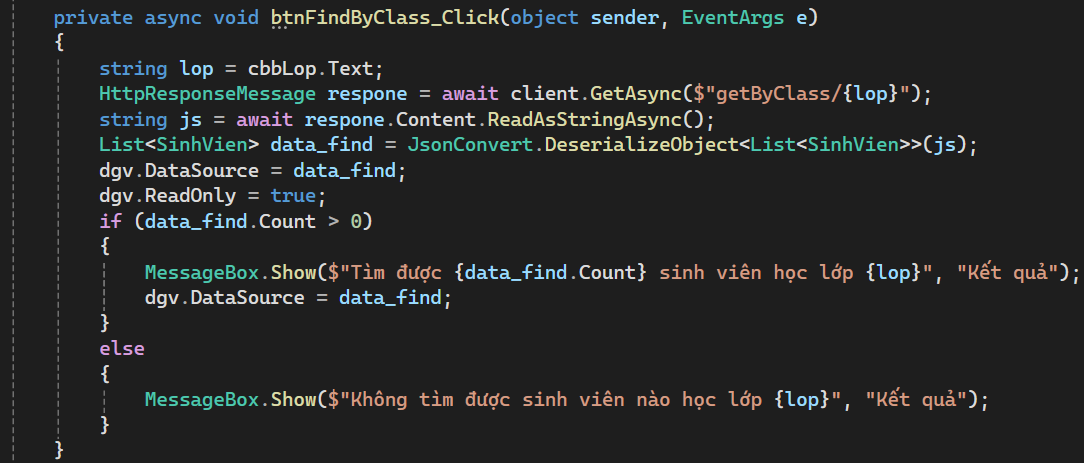
- Như ta thấy dù viết API hay thực hiện viết logic cho nút thì việc thêm hay sửa đều giống nhau ở cách xử lý, khác một chút là ta cần call API và hàm bất đồng bộ PUT chứ không phải POST.

****

**11. Nút xóa 1 phần tử**

**-** Thường sẽ xóa theo thuộc tính định danh, ở đây ta set là id. ****

**12. Tìm kiếm theo điều kiện bất kì**

****