Triển khai API cho website Caro online

Chia các bước:

* Phân tích yêu cầu bài toán
* Phân tích cơ sở dữ liệu
* Tích hợp Entity framework Core 5.x
* Swagger
* Yêu cầu: C#, OOP, SQL Server

\*Phân tích bài toán:

- Đăng nhập, đăng xuất – cần bảng Tài khoản người dùng

- Phòng chat nhắn tin, (bảng phòng) (bảng tin nhắn)

- 2 người đánh với nhau (X. O) => Bảng trận đấu(id, id phòng, id 2 người, id người tạo, thòi gian tạo phòng, điểm tháng của người 1, điểm thắng của người 2, người thắng chung cuộc)

- Bảng chi tiết trận đấu (id, id trận đấu, ván thú mấy, tọa độ X, Tọa độ y, nước đi thú bao nhiêu, nước này do ai đánh)

- Bảng lịch sử đánh,

- Bảng tin nhắn (id, id người chơi, nội dung tin nhắn, thời gian nhắn)

\*Phân tích cơ sở dữ liệu

User(Id, Name, Icon, Username, Password, IsAdmin, Score)

Room(Id, Name, Status)

Match(Id, RoomId, FirstUserId, SecondUserId, CreatedRoomUser, CreatedDate, FirstScore, SecondScore, Winner)

MatchDetail(Id, MatchId, Order, X, Y, Sort, UserId)

Message(Id, UserId, Content, CreatedDate, RoomId)

\*Thực hiện viết code xử lý cho chương trình

Công nghệ ASP.NET Core 5.x

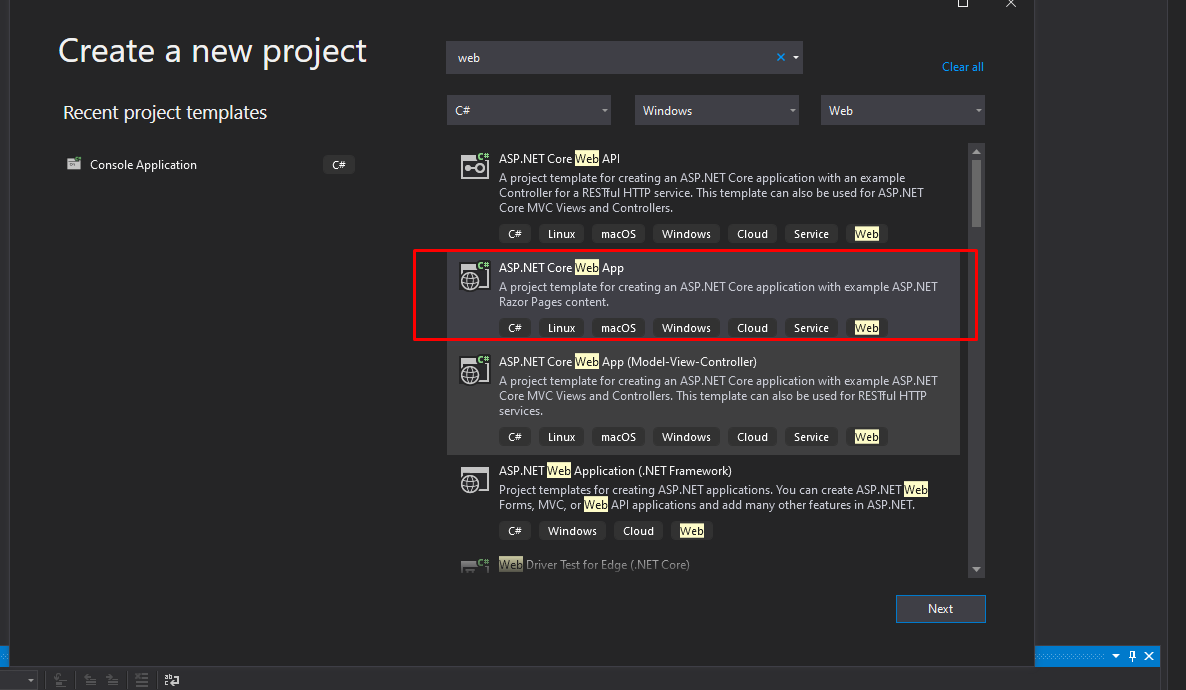
RestfullAPI, SQL, linq, Entity framework, SignalR, C#, kiểu dữ liệu JSON

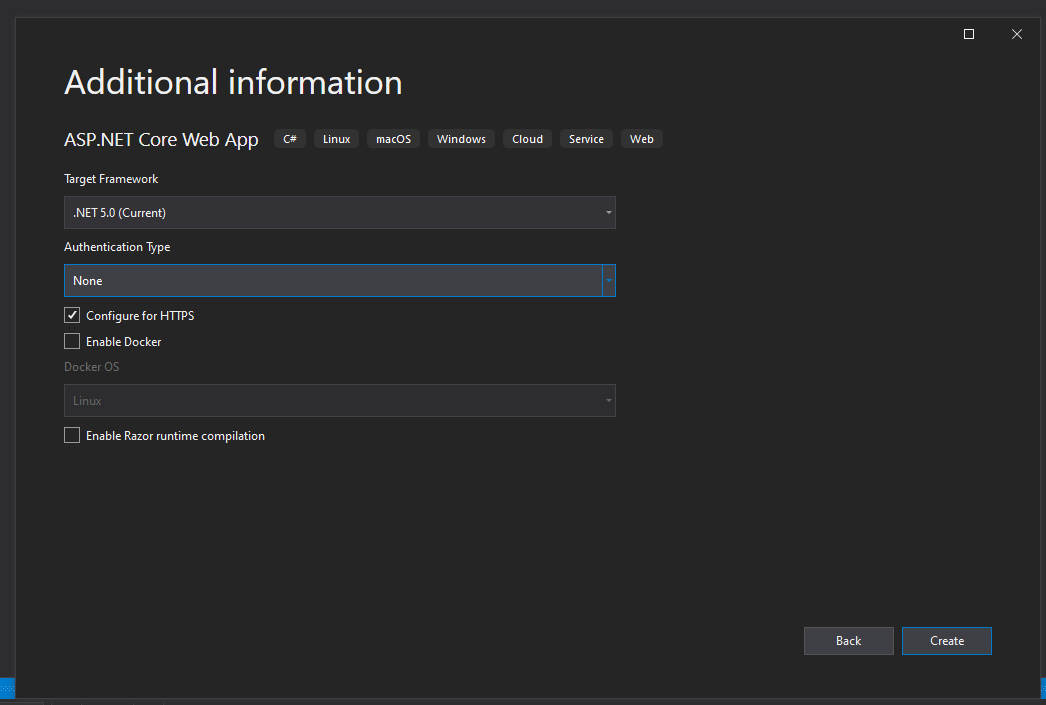
Bắt đầu viết….

* Cài đặt Visual studio 2019
* SQL Server 2014 đổ lên

Cách viết

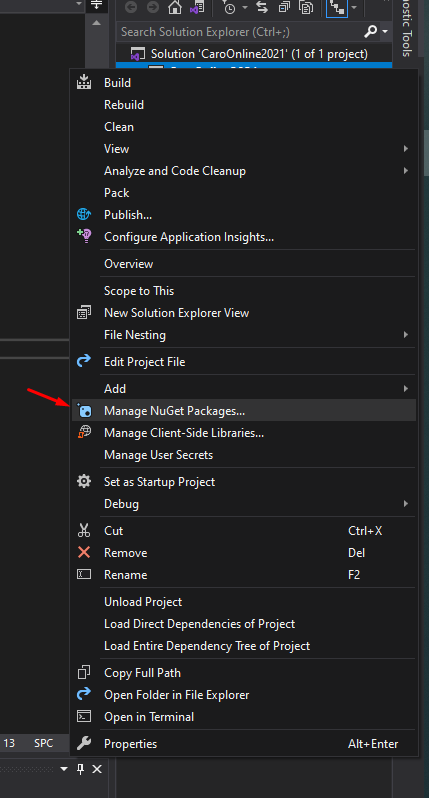
* Tạo project với công nghệ là .Net core Web application

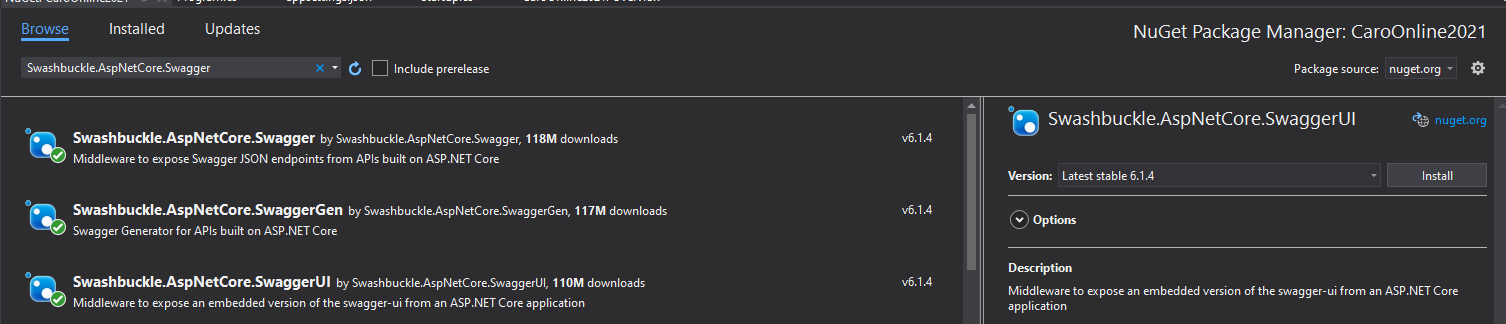




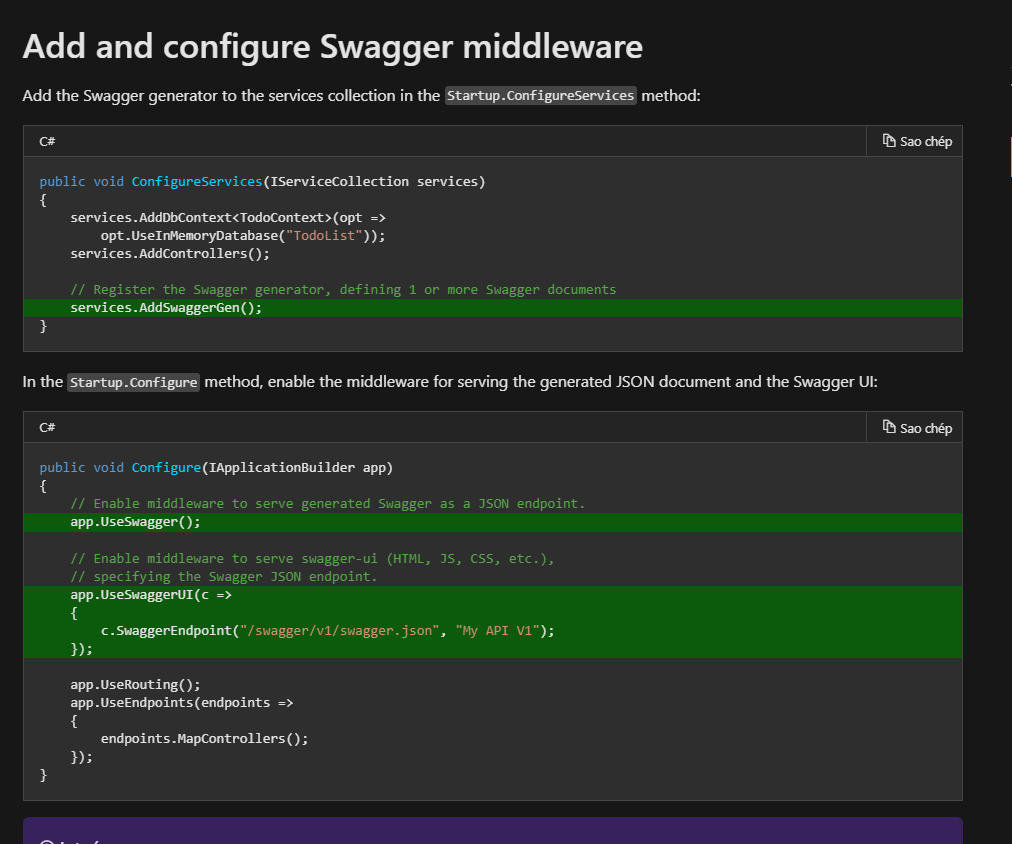
Nếu chưa Có swagger thì add theo cách sau

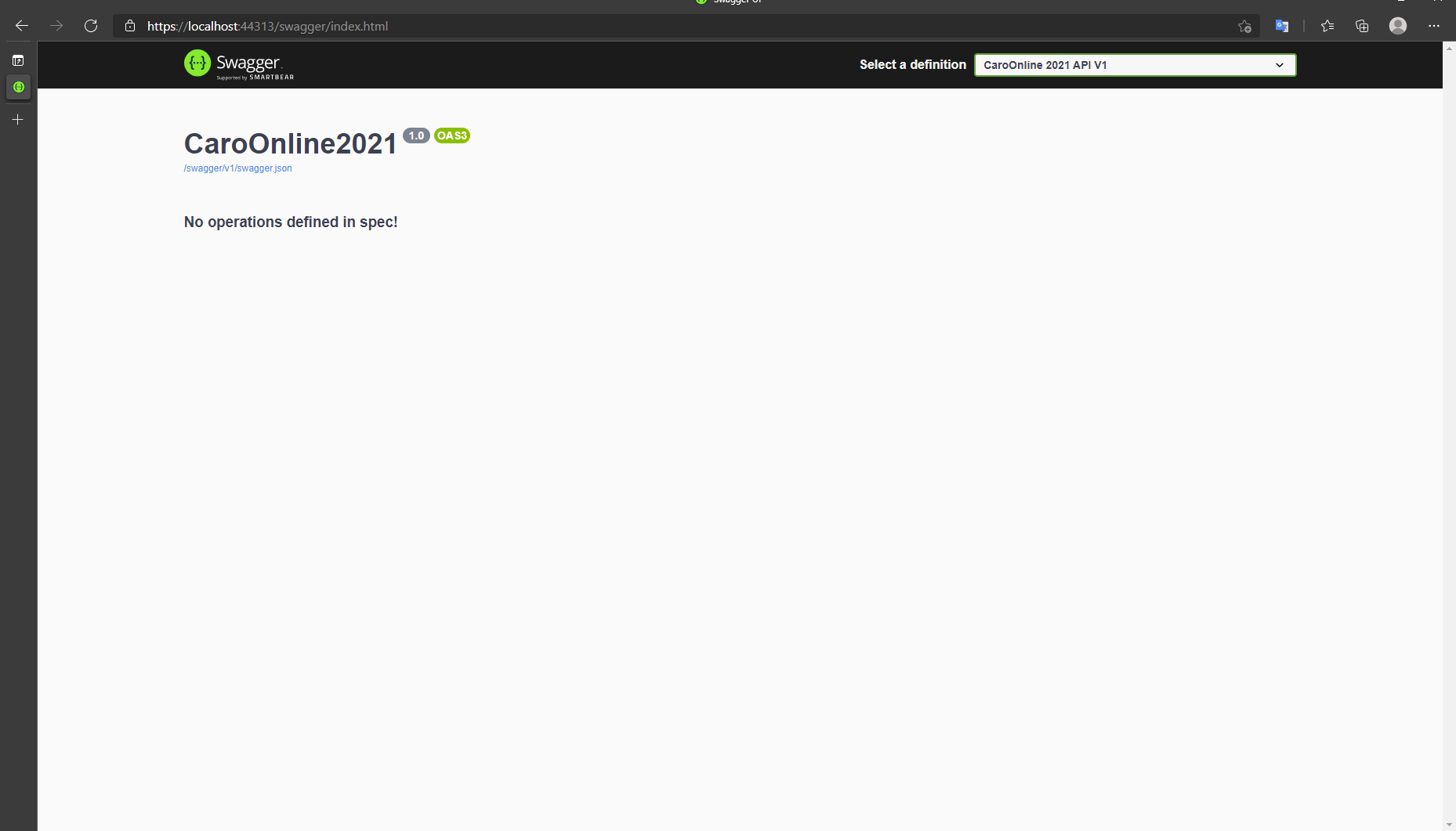
<https://docs.microsoft.com/vi-vn/aspnet/core/tutorials/getting-started-with-swashbuckle?view=aspnetcore-5.0&tabs=visual-studio>





Thêm cấu hình trong file Startup.cs





Đã tích hợp đc swagger

……………………………………………………

Add swagger là Api document : tài liệu về api mình viết trong project của mình.

Dùng để test api do mình viết ra

Bước tiếp theo là tạo Model – tương ứng với các bảng trong SQL bằng cách sử dụng Code first

Tức là viết code trước thông qua các class => tạo ra database trong SQL bằng Entity framework

Ưu điểm là chúng ta thay đổi cấu trúc dữ liệu của các bảng là đơn giản và nhanh chóng. Giúp cho việc truy vấn rành mạch và có cú pháp là ngắn gọn.

Nhận thấy csdl mà ta phân tích có dạng

User(Id, Name, Icon, Username, Password, IsAdmin, Score)

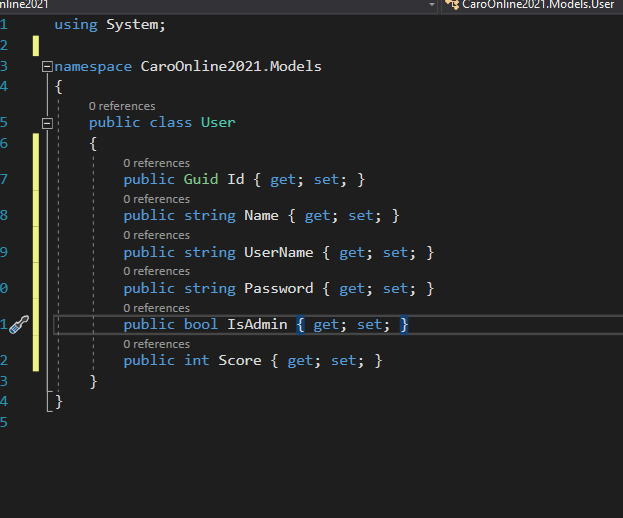
Room(Id, Name, Status)

Match(Id, RoomId, FirstUserId, SecondUserId, CreatedRoomUser, CreatedDate, FirstScore, SecondScore, Winner)

MatchDetail(Id, MatchId, Order, X, Y, Sort, UserId)

Message(Id, UserId, Content, CreatedDate, RoomId)

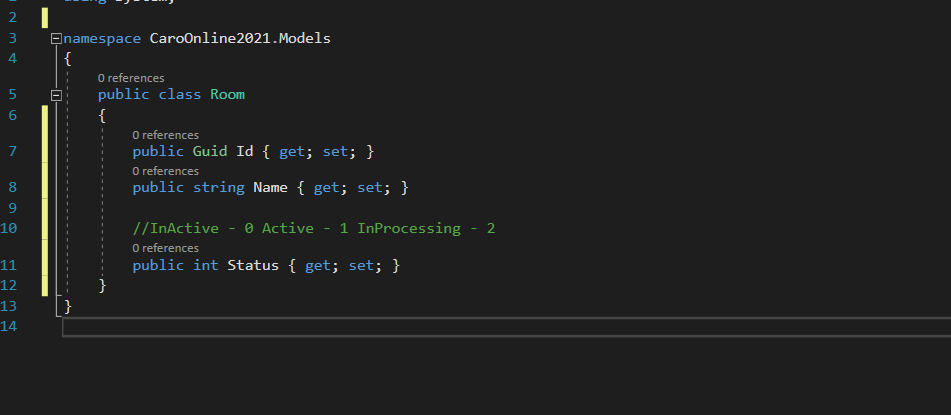
* Cần tạo các class tương ứng với mỗi bảng

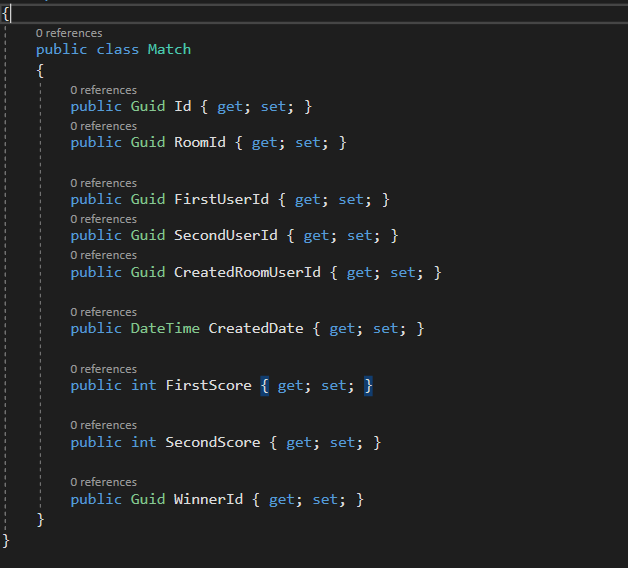


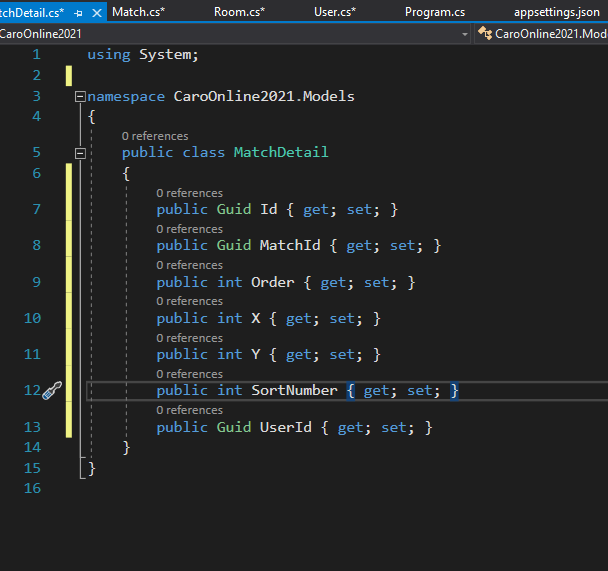
Tạo thành công lớp User tương ứng với bảng User

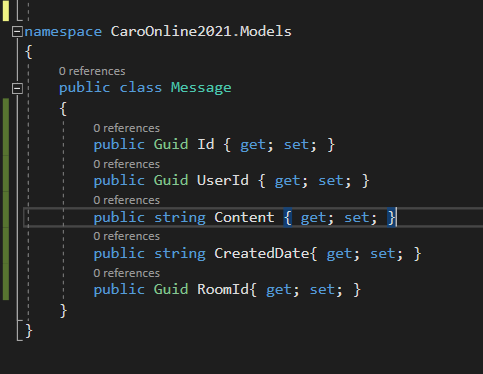
Class Phòng( nhận thấy là có 3 trạng thái: (0)InActive – không hoạt động, (1)Active – có thể hoạt động(sử dụng), (2)InProcessing – Đang có người bên trong phòng này)

* Sử dụng kiểu int hoặc Enum ( để đơn giản chúng chọn kiểu int)









Sự khác nhau giữa class và các bảng trong SQL là gì ?

Class nó được dùng để đại diện 1 cái bảng nào đó trong SQL, và các đối tượng được tạo bởi class thì trong ngôn ngữ C# thì được lưu ở thanh RAM

Còn bảng SQL thì nó lưu ở ổ cứng

* Tốc độ truy xuất ở 1 thời điểm là RAM thì luôn nhanh hơn.
* Nhưng sẽ chậm khi dữ liệu là quá lớn.

Class trong entity framework (linq ) sẽ đọc chậm so với sql thuần túy ( nó phải mapping từ bảng sql sang dạng class sau đó mới thực truy vấn dưới dạng Iqueryable lên tốc độ là chậm khoảng 5 lần)

Nhưng vì chúng ta làm chương trình demo lên việc chậm 5 lần k nhìn rõ đc ( dữ liệu ít) – ngoài ra còn phụ thuộc vào phần cứng

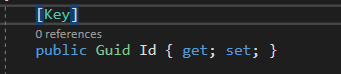
Làm nào tạo các mối quan hệ dữ liệu giữa các class này ( đóng vai trò như các bảng SQL)

* Tạo các khóa chính và khóa phụ
* Cho các class tương ứng

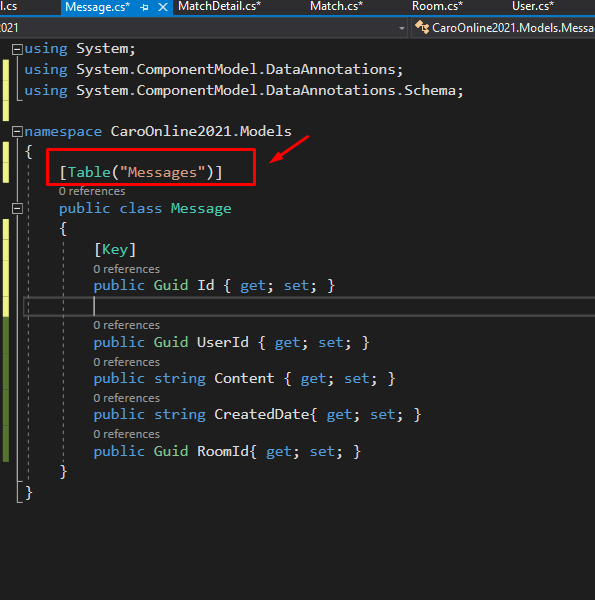
Cung cấp keywork annotation trong C#

using System.ComponentModel.DataAnnotations;

chỉ định khóa chính (primary key) cho bảng- class



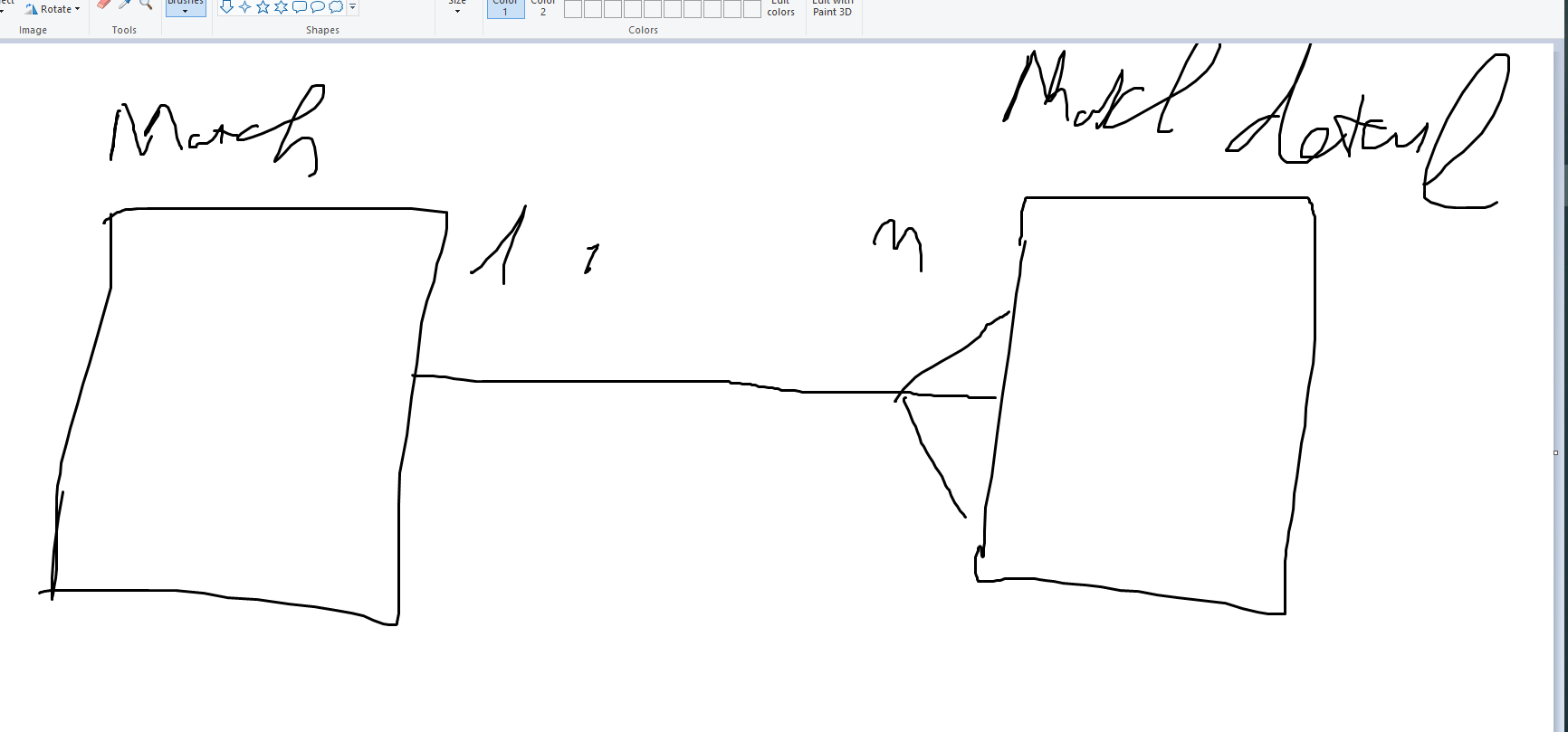
Chỉ định tên bảng bằng cách



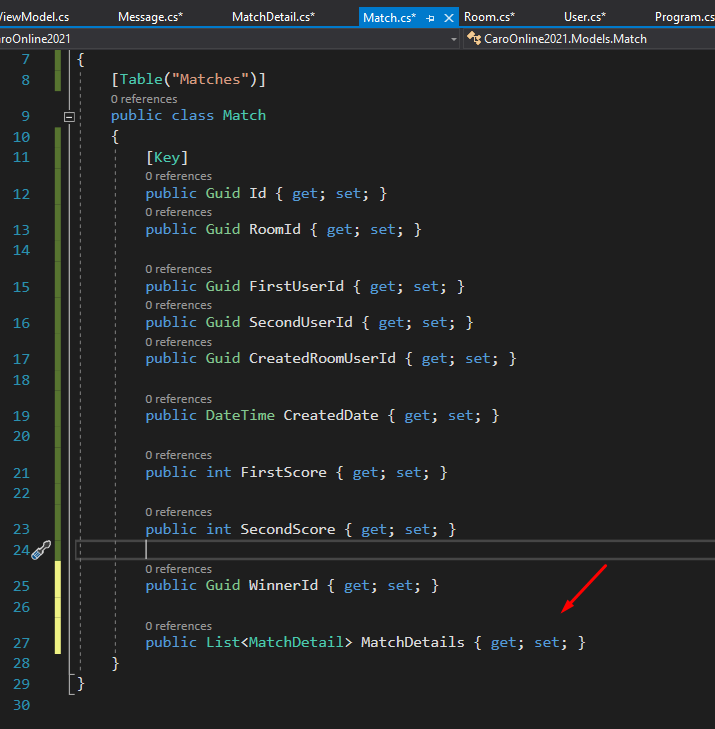
Tương tự với các bảng – class khác trong csdl mà chúng ta đã phân tích

…

Tiếp tục thiết lập khóa phụ với các bảng có mối quan hệ là 1-n hoặc n-n (thì phải tạo thêm class đóng vai trò là bảng khử mối quan hệ n-n)

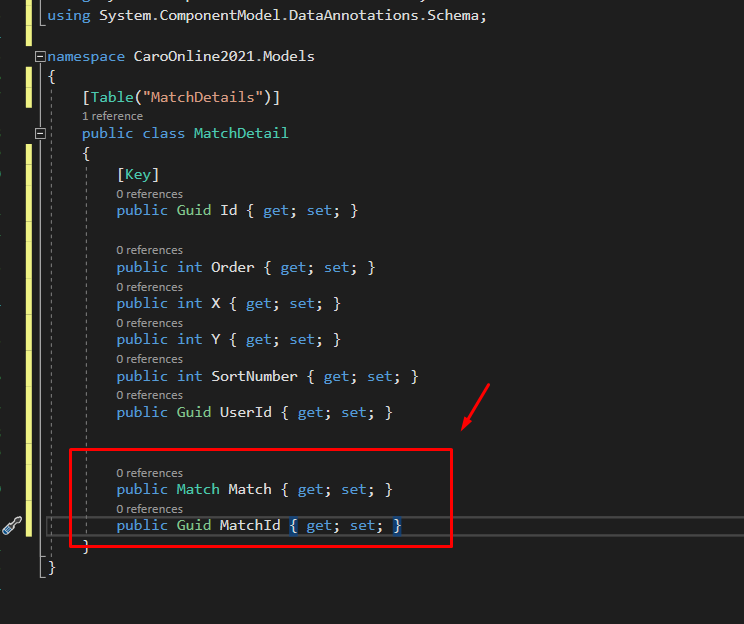


Như vậy Match thì cần lưu 1 danh sách MatchDetal



Và trong MatchDetal chúng ta cần lưu 1 đối tượng Match và 1 Id của Match đó.

Cụ thể như sau:



* Ta đã tạo khóa chính và khóa phụ cho mối quan hệ giữa 2 bảng Match- MatchDetail
* Tương tự với các bảng có mối hệ 1-n

Bước tiếp theo:

Tích hợp Entity framework vào trong project của mình để có thể tạo được database từ các class có chỉ định tên bảng và khóa chính, khóa phụ tương ứng mối quan hệ của chúng.

<https://docs.microsoft.com/en-us/aspnet/core/data/ef-mvc/intro?view=aspnetcore-5.0>

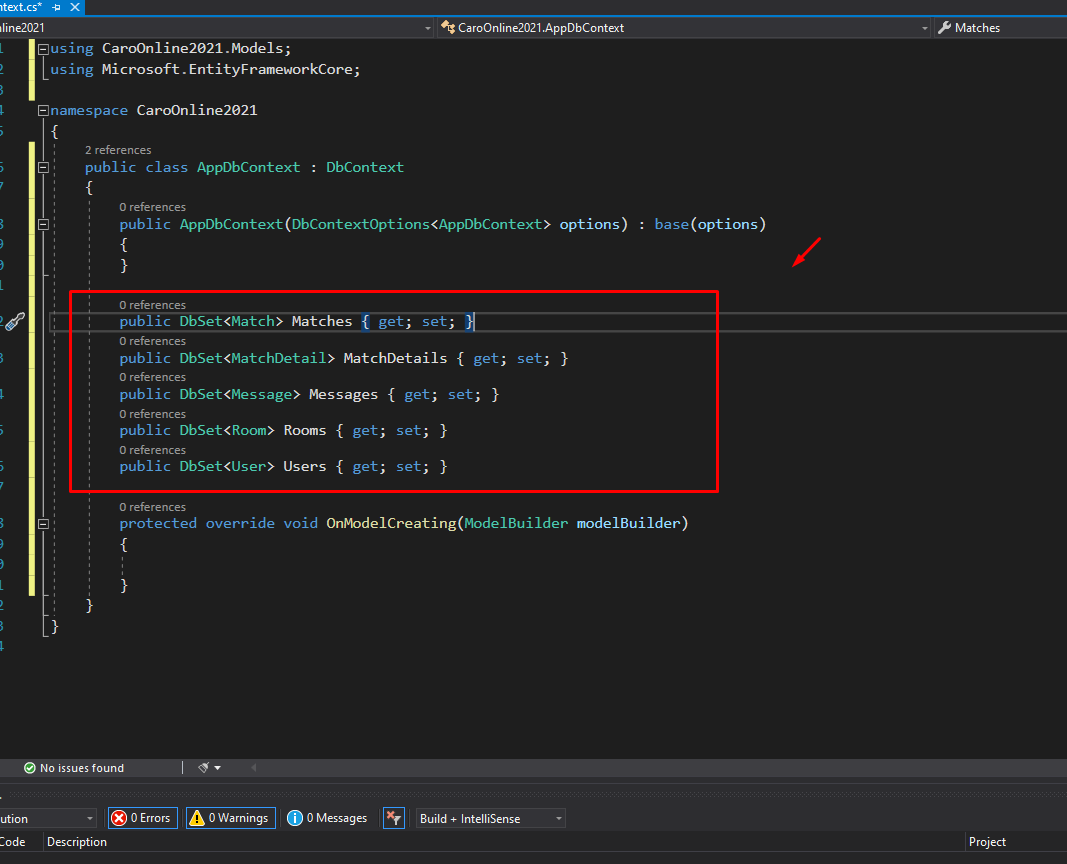
Cài

Microsoft.AspNetCore.Diagnostics.EntityFrameworkCore

[Microsoft.EntityFrameworkCore.SqlServer](https://www.nuget.org/packages/Microsoft.EntityFrameworkCore.SqlServer/)

Và cài thêm Microsoft.EntityFrameworkCore.Tools : giúp chúng ta sử dụng được câu lệnh để tạo ra database từ các class mà chúng ta viết trước đó.

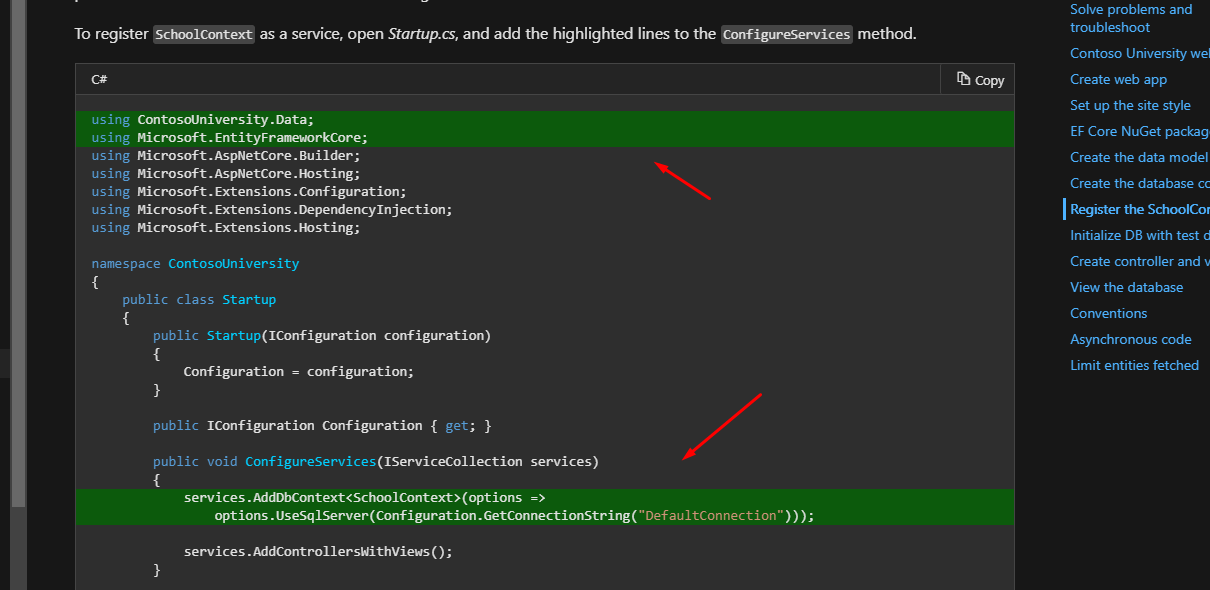
Tiếp theo cần tạo ra 1 class là kế thừa từ lớp dbContext dùng cho việc tương tác và truy vấn dữ liệu cũng như tạo ra cái db từ câu lệnh add-migration <tên migration>



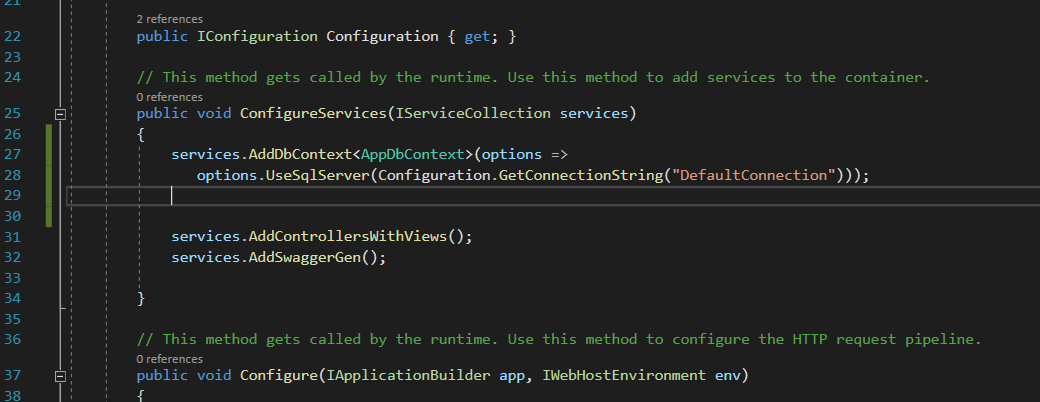
Vùng chỉ ra các bảng mà chúng ta tương tác và khởi ra trong SQL

Nó chính là toàn bộ các class hay các bảng mà chúng ta cần khởi tạo.

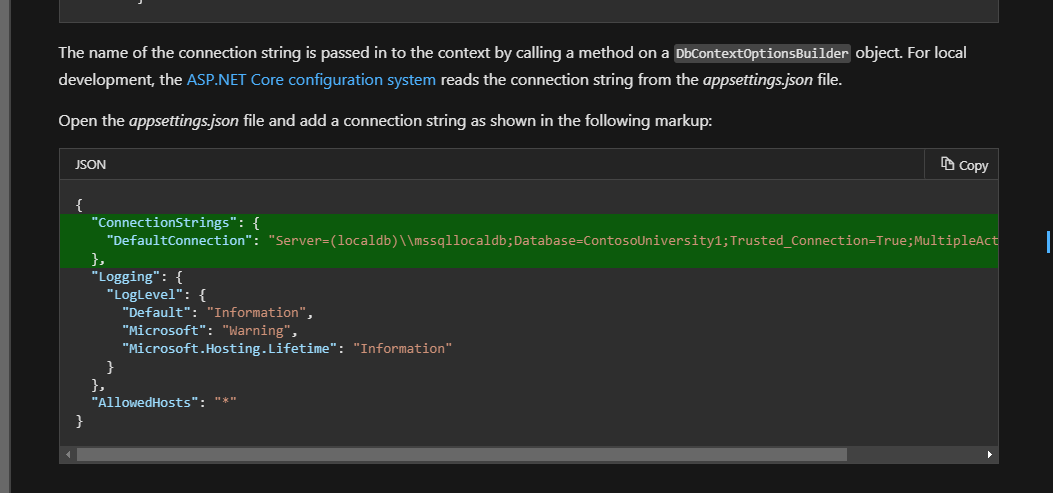
Chúng cần có cấu hình để chỉ ra rằng cái chuỗi kết nối với SQL ở đâu

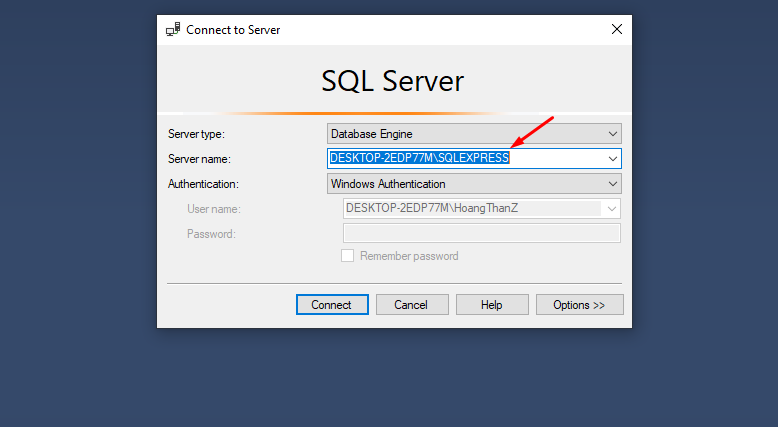
* File startup.cs
* 

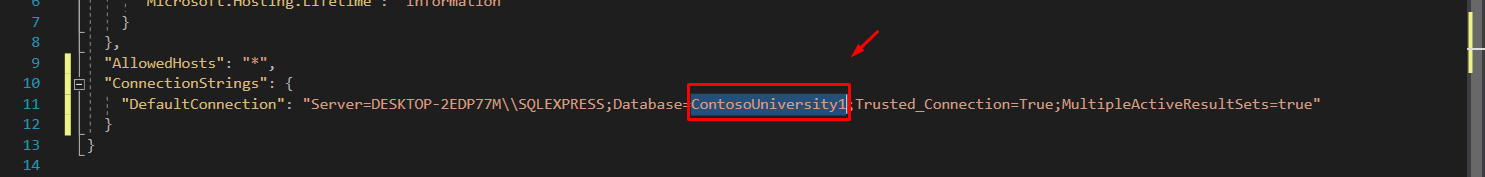
Áp dụng tương tự với bài của mình



Trong file appsetting.json

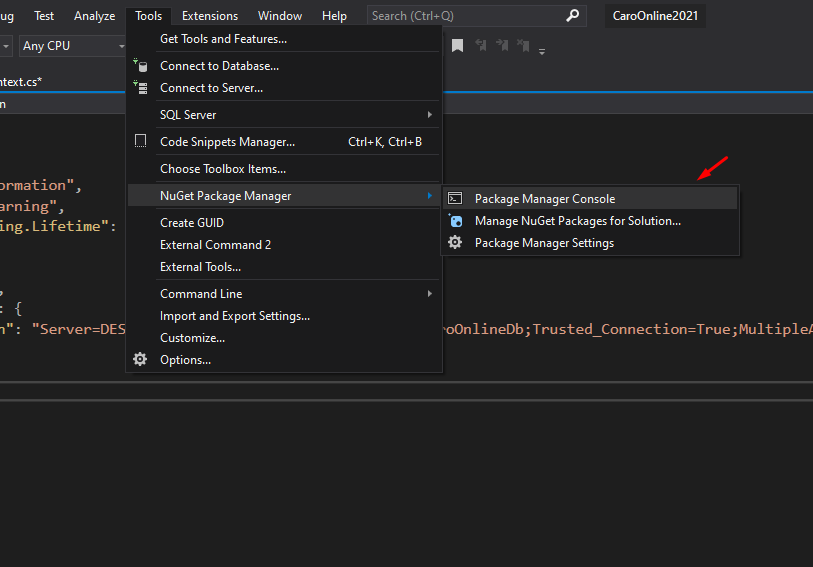


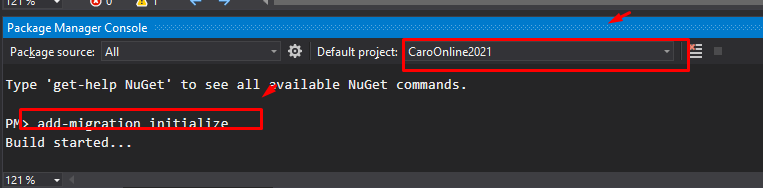


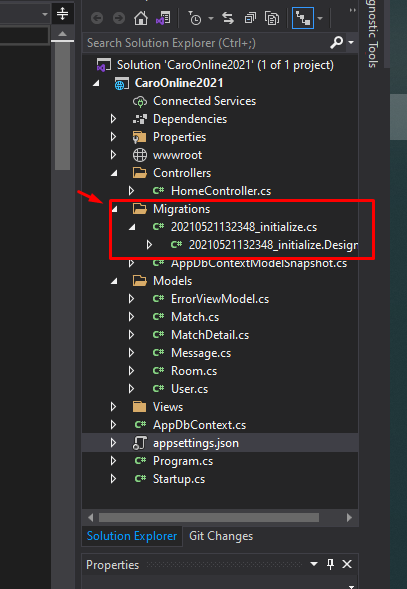


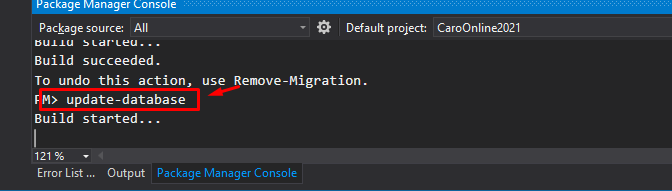
Tên của db mà mình muốn đặt.

Sử dụng console để generate ra db









Thực hiện update-database để nó tạo ra cái db mà mình muốn có.

