Setup

1. If you don’t have visual studio you can go to the google and download it from the Microsoft and you can choose the visual studio community or replace it by the visual studio pro more open sources and have payment.
2. Go to visual studio installer. Install ASP.NET and web development (Web & cloud) for the API crud operation in the web and .Net desktop development for the basic C#.
3. Open create new project when entering visual studio and choose Asp.net Core Web API for the project click next then name the project, choose location and solution name for the project. Click next and choose the framework and it is better to choose the framework that have long standard support.
4. The project will be created with controller and properties folder which the properties has launchsettings.json to choose which URL you want it to run, programs.cs that run the operation of the project and appsettings.json that can be use for the debug and connection of database.
   1. First create a model folder inside a class name Study.
   2. Second create a data folder inside a class name StudentContext.
      1. The StudentContext class is a part of Entity Framework Core, and it represents the database context. It acts as a bridge between your application code and the underlying database. the StudentContext class is an essential component in Entity Framework Core that simplifies working with databases in your application.

Code:

using Microsoft.EntityFrameworkCore;

using Student.Model;

namespace Student.Data

{

public class StudentContext : DbContext

{

public StudentContext(DbContextOptions<StudentContext> options) : base(options)

{

}

public DbSet<Study> study { get; set; }

}

}

* 1. Create a connection of your database in visual studio inside appsettings.json and add the debug where it will show the debug operation in the output which can make for the person were to find his error.

Connection:

"ConnectionStrings":{

"MyConnectionString": "Server=host/address;Port=number;User Id=name of user; Password=password;Database=name; SearchPath= schema name "

}

Debug:

"Logging": {

"LogLevel": {

"Default": "Debug",

"Microsoft": "Debug",

"Microsoft.Hosting": "Debug",

"Microsoft.EntityFrameworkCore.Database.Command": "Debug"

}

"AllowedHosts": "\*" == The "AllowedHosts": "\*" configuration setting is used in ASP.NET Core applications to specify which hosts are allowed to access the application.

* 1. Create a class startup that has method configuration of the database connection path and UseNpgsql for the PostgreSQL to work.
     1. Add another services just like swagger that will be use launcherSetting.json when it runs by the operation.
     2. Code:

// Startup.cs

using Microsoft.EntityFrameworkCore;

using Microsoft.OpenApi.Models;

using Student.Data;

namespace Student

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<StudentContext>(options =>

options.UseNpgsql(Configuration.GetConnectionString("MyConnectionString")));

services.AddControllers();

services.AddSwaggerGen(c =>

{

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

});

}

public void Configure(IApplicationBuilder app, IWebHostEnvironment env)

{

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseSwagger();

// Specify the Swagger JSON endpoint

app.UseSwaggerUI(c =>

{

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

// If you want to serve Swagger UI at the root URL, use the following line instead:

// c.RoutePrefix = string.Empty;

});

}

else

{

app.UseExceptionHandler("/Error");

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseRouting();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllers();

});

}

}

}

* 1. Change the writing of program.cs that id link to startup class then run it

namespace Student

{

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>();

});

}

}

1. Error can be found in program if the connection doesn’t work right.
   1. If it is because of connection String put the right information of the database.
   2. Class is not match to the table entity attribute.
   3. Problem is that the database couldn’t be found in the PostgreSQL
      1. Delete the database.
      2. Open the command prompt or terminal.
      3. Navigate to the directory that contains your project file (.csproj).
      4. Run the following command to create the database:[ dotnet ef database update ]
      5. This command applies any pending migrations to the database and creates it if it doesn't already exist.
      6. After the dotnet ef database update command successfully executes, the tables defined in the pending migrations will be created in the database, and any other schema modifications will be applied as well.
2. To add dependencies
   1. Go to Tools on the header.
   2. Click on it and go to NuGet Package Manager
   3. Then click on the NuGet Packages for solution…
   4. It will open a window of the dependencies and can be seen.
      1. It can be updated or installed the dependencies from t.
3. If the UseSqlServer of: options.UseSqlServer(Configuration.GetConnectionString("DefaultConnection")));
   1. Please install in the terminal or Package Manager console
      1. Install-Package Microsoft.EntityFrameworkCore.SqlServer