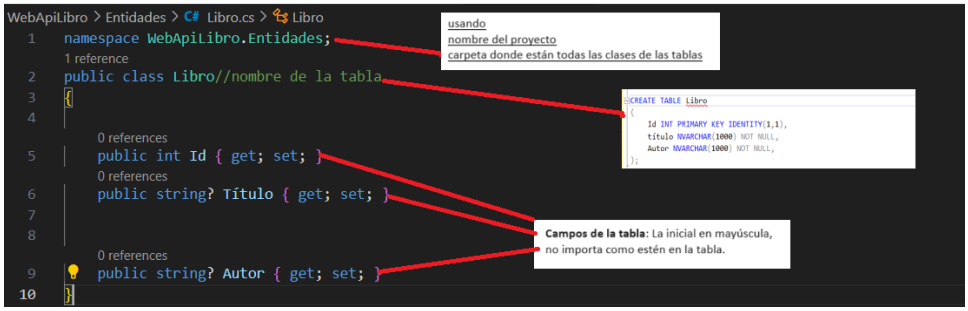
CRUD REACT-NET

Creación del proyecto: => dotnet new weapi –o NombreProyecto

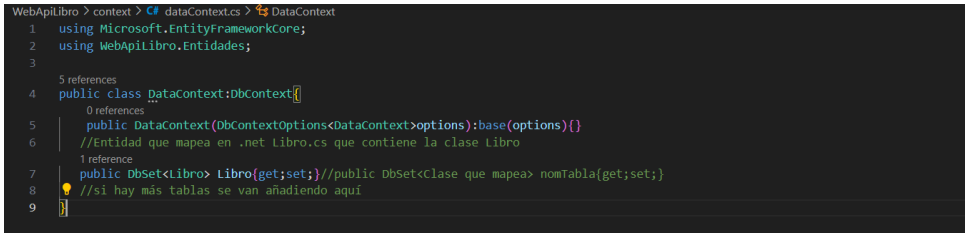
Navegar hacia la carpeta: => cd nombreProyecto

Crear los archivos de configuración => run -> genera la carpeta .vscode

1- En la carpeta principal crear la carpeta Entidades, esta carpeta guardará todas las clases para cada tabla de la base de datos. **Entidades=>Libro.cs**



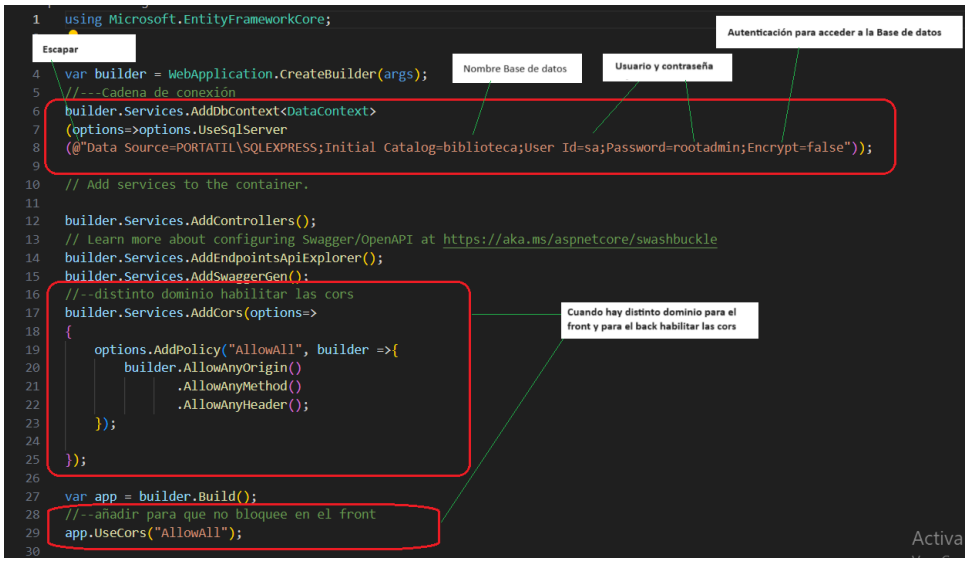
2- En la carpeta principal crear la carpeta **Context=>dataContext.cs**



3- En la carpeta **progam.cs**

WebApiLibro > C# Program.cs

```
using Microsoft.EntityFrameworkCore;

var builder = WebApplication.CreateBuilder(args);
//---Cadena de conexión
builder.Services.AddDbContext<DataContext>
(options=>options.UseSqlServer
(@"Data Source=PORTATIL\SQLEXPRESS;Initial Catalog=biblioteca;User Id=sa;Password=rootadmin;Encrypt=false"));

// Add services to the container.

builder.Services.AddControllers();
// Learn more about configuring Swagger/OpenAPI at https://aka.ms/aspnetcore/swashbuckle
builder.Services.AddEndpointsApiExplorer();
builder.Services.AddSwaggerGen();
//--distinto dominio habilitar las cors
builder.Services.AddCors(options=>
{
    options.AddPolicy("AllowAll", builder =>{
        builder.AllowAnyOrigin()
               .AllowAnyMethod()
               .AllowAnyHeader();
    });

});

var app = builder.Build();
//--añadir para que no bloquee en el front
app.UseCors("AllowAll");

```



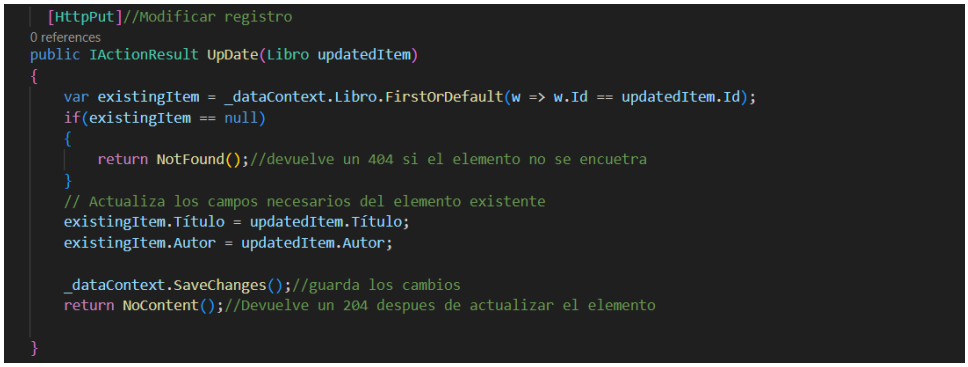
Lo demás en el archivo se queda tal cual.

```
// Configure the HTTP request pipeline.
if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseHttpsRedirection();

app.UseAuthorization();

app.MapControllers();

app.Run();
```

4- Se crea el controlador para la tabla, en la carpeta **‘controllers’** se crea el archivo, ya viene uno de ejemplo por lo que se copia y se le cambia el nombre.


```
[HttpGet]//obtener los datos de la tabla libro
0 references
public ActionResult Get()
{
    return Ok(_dataContext.Libro);

}
[HttpPost]//añadir registro
0 references
public ActionResult Add(Libro element)
{
    _dataContext.Libro.Add(element);//añade una fila nueva con el nuevo registro en la tabla Libro
    _dataContext.SaveChanges();//guarda los cambios
    return Ok(element);//devuelve un ok del elemento

}
```

```
[HttpDelete("{id}")]//Eliminar registro
0 references
public IActionResult Delete(int id)
{
    var itemToDelete = _dataContext.Libro.FirstOrDefault(w => w.Id == id);
    if(itemToDelete == null)
    {
        return NotFound();
    }
    _dataContext.Libro.Remove(itemToDelete);
    _dataContext.SaveChanges();//guarda los cambios
    return NoContent();//devuelve un ok del elemento

}
```



```
[HttpPut]//Modificar registro
0 references
public IActionResult UpDate(Libro updatedItem)
{
    var existingItem = _dataContext.Libro.FirstOrDefault(w => w.Id == updatedItem.Id);
    if(existingItem == null)
    {
        return NotFound();//devuelve un 404 si el elemento no se encuetra
    }
    // Actualiza los campos necesarios del elemento existente
    existingItem.Título = updatedItem.Título;
    existingItem.Autor = updatedItem.Autor;

    _dataContext.SaveChanges();//guarda los cambios
    return NoContent();//Devuelve un 204 despues de actualizar el elemento

}
```

```
[HttpGet]//obtener los datos de la tabla libro
   public ActionResult Get()
   {
      return Ok(_dataContext.Libro);


   }
   [HttpPost]//añadir registro
   public ActionResult Add(Libro element)
   {
      _dataContext.Libro.Add(element);//añade una fila nueva con el nuevo registro en la tabla Libro
      _dataContext.SaveChanges(); //guarda los cambios
      return Ok(element);//devuelve un ok del elemento


   }
    [HttpDelete("{id}")]//Eliminar registro
   public IActionResult Delete(int id)
   {
      var itemToDelete = _dataContext.Libro.FirstOrDefault(w => w.Id == id);
      if(itemToDelete == null)
      {
         return NotFound();
      }
      _dataContext.Libro.Remove(itemToDelete);
      _dataContext.SaveChanges();//guarda los cambios
      return NoContent();//devuelve un ok del elemento
```

```
}
  [HttpPut]//Modificar registro
public IActionResult UpDate(Libro updatedItem)
{
    var existingItem = _dataContext.Libro.FirstOrDefault(w => w.Id == updatedItem.Id);
    if(existingItem == null)
    {
        return NotFound();//devuelve un 404 si el elemento no se encuetra
    }
    // Actualiza los campos necesarios del elemento existente
    existingItem.Título = updatedItem.Título;
    existingItem.Autor = updatedItem.Autor;

    _dataContext.SaveChanges();//guarda los cambios
    return NoContent();//Devuelve un 204 despues de actualizar el elemento

}
```

Añadir los entityFramewor, en el archivo con extension => **csproj**

```
<ItemGroup>
  <PackageReference Include="Microsoft.AspNetCore.OpenApi" Version="7.0.12" />
  <PackageReference Include="Microsoft.EntityFrameworkCore" Version="7.0.12" />
  <PackageReference Include="Microsoft.EntityFrameworkCore.Design" Version="7.0.12">
    <IncludeAssets>runtime; build; native; contentfiles; analyzers; buildtransitive</IncludeAssets>
    <PrivateAssets>all</PrivateAssets>
  </PackageReference>
  <PackageReference Include="Microsoft.EntityFrameworkCore.SqlServer" Version="7.0.12" />
  <PackageReference Include="Swashbuckle.AspNetCore" Version="6.5.0" />
</ItemGroup>
```