Using Visual Studio 2022 Community

Create a new project “ASP.NET Core Web API”.

Name it “UsersLogInAPI”, and in location, select the folder “DotNet6” that we created earlier. Click on next. Select the Framework “.NET 6.0”. Check “Use controllers” and “Enable OpenAPI support”. Click on the button create.

You can eliminate the example class “WeatherForecast.cs” and the controller “WeatherForecastController.cs”

In the project “UserLogInAPI”, create a new class “User”

|  |
| --- |
| using System.ComponentModel.DataAnnotations.Schema;  namespace UsersLogInAPI  {  [Table("test\_users")]  public class User  {  [Column("id")]  public int Id { get; set; }  [Column("name")]  public string Name { get; set; } = string.Empty;  [Column("password")]  public string Password { get; set; } = string.Empty;  [Column("small\_description")]  public string Small\_Description { get; set; } = string.Empty;  }  } |

In the folder “Controllers” create a new controller of type “API Controller – Empty”. Name it “UserController.cs”

|  |
| --- |
| using Microsoft.AspNetCore.Mvc;  using Microsoft.EntityFrameworkCore;  using UsersLogInAPI.Data;  namespace UsersLogInAPI.Controllers  {  [Route("api/[controller]")]  [ApiController]  public class UserController : ControllerBase  {  private readonly DataContext \_context;  public UserController(DataContext context)  {  \_context = context;  }  [HttpGet]  public async Task<ActionResult<List<User>>> GetUsers()  {  return Ok(await \_context.Users.ToListAsync());    }  }  } |

Run the solution, so Swagger will show the API methods work. In addition, run the GET method and retrieve the API address, since we need to copy it into the FrontEnd application. For this example, in my case, it is : “https://localhost:7268/api” (NOTE: You can also find this string in the launchSettings.json/profiles/applicationUrl )

Using NuGet, install the following packages:

* Microsoft.EntityFrameworkCore
* Microsoft.EntityFrameworkCore.Design
* Npgsql.EntityFrameworkCore.PostgreSQL

In the project, create a folder “Data”. Inside that folder, create a class “DataContext.cs”

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  namespace UsersLogInAPI.Data  {  public class DataContext : DbContext  {  public DataContext(DbContextOptions<DataContext> options) : base(options) { }  public DbSet<User> Users => Set<User>();  }  } |

Now we add the connection string to appsettings.json

|  |
| --- |
| {  "ConnectionStrings": {  "DefaultConnection": "User ID =postgres;Password=root;Server=localhost;Port=5432;Database=Basic\_CRUD\_AngularDotNet; Integrated Security=true;Pooling=true;"  },  "Logging": {  "LogLevel": {  "Default": "Information",  "Microsoft.AspNetCore": "Warning"  }  },  "AllowedHosts": "\*"  } |

Add DataContext and CORS exception to “Program.cs”. The port can be different, but you can find it by doing “ng serve -o“ in the frontEnd.

|  |
| --- |
| using Microsoft.EntityFrameworkCore;  using UsersLogInAPI.Data;  var builder = WebApplication.CreateBuilder(args);  // 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();  builder.Services.AddDbContext<DataContext>(options =>  {  options.UseNpgsql(builder.Configuration.GetConnectionString("DefaultConnection"));  }  );  builder.Services.AddCors(options => options.AddPolicy(  name: "UsersLogInOrigins",  policy =>  {  policy.WithOrigins("http://localhost:4200").AllowAnyMethod().AllowAnyHeader();  }  ));  var app = builder.Build();  // Configure the HTTP request pipeline.  if (app.Environment.IsDevelopment())  {  app.UseSwagger();  app.UseSwaggerUI();  }  app.UseCors("UsersLogInOrigins");  app.UseHttpsRedirection();  app.UseAuthorization();  app.MapControllers();  app.Run(); |

Now open Tools/NuGet Package Manager/Package Manager Console

Go to the root of the project ( cd .\UsersLogInAPI\UserLogInAPI). If you do a ls, you should have several files and folders, between them the appsettings.json. If you see it, you are in the correct place.

Now execute the command:

* dotnet tool install --global dotnet-ef
* dotnet ef (If you see the unicorn, everything is ok :)
* dotnet ef migrations add Initial

Now Run it at the same time as the FrontEnd!!