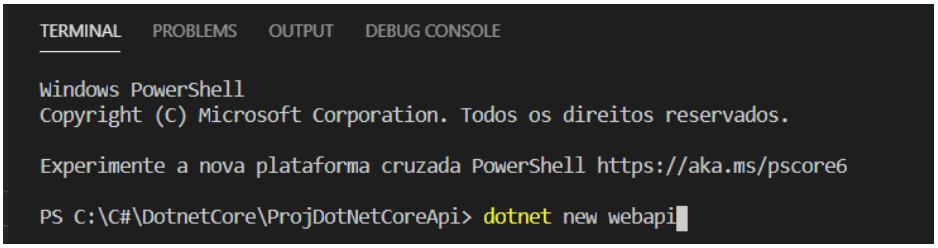
DotNet Core + EntityFramework

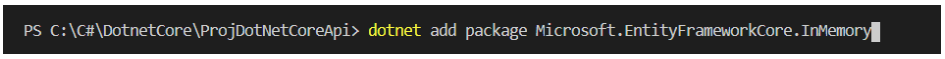
Sumário

Criar uma pasta “ProjDotNetCoreApi”

Nesta pasta criar o projeto via Terminal:

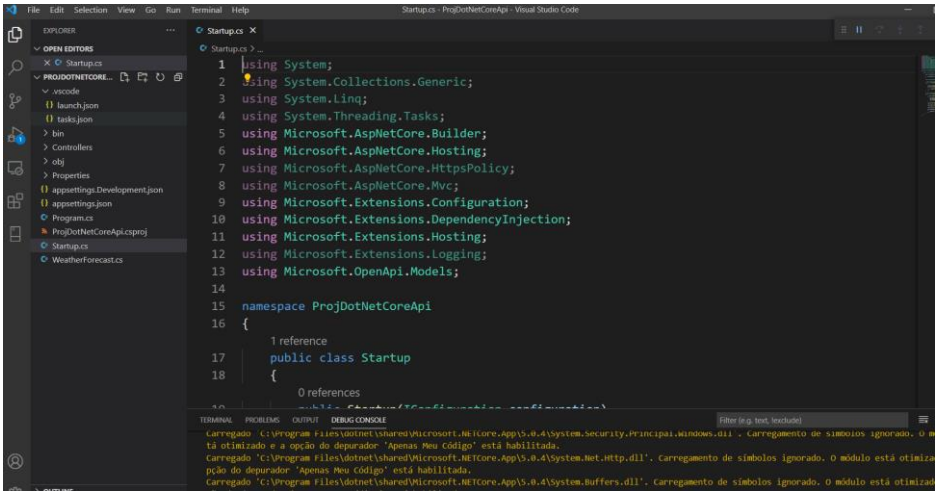


Incluir o EntityFramework no projeto:

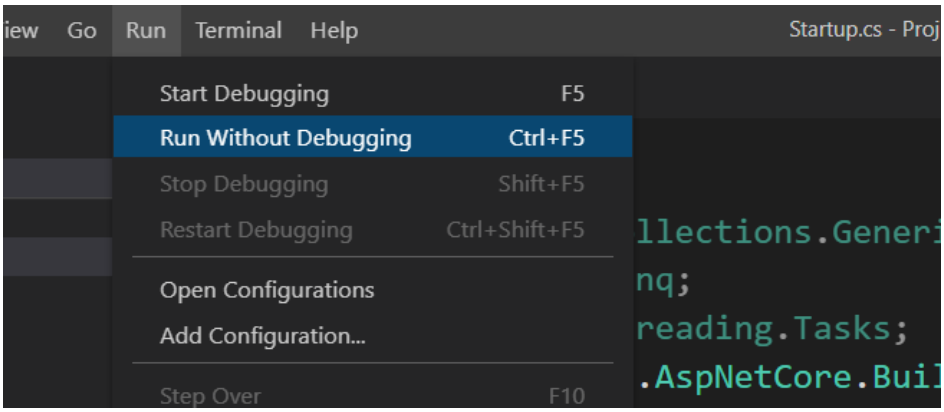


dotnet add package Microsoft.EntityFrameworkCore.InMemory

A seguinte estrutura será criada automaticamente:



Executar o projeto:



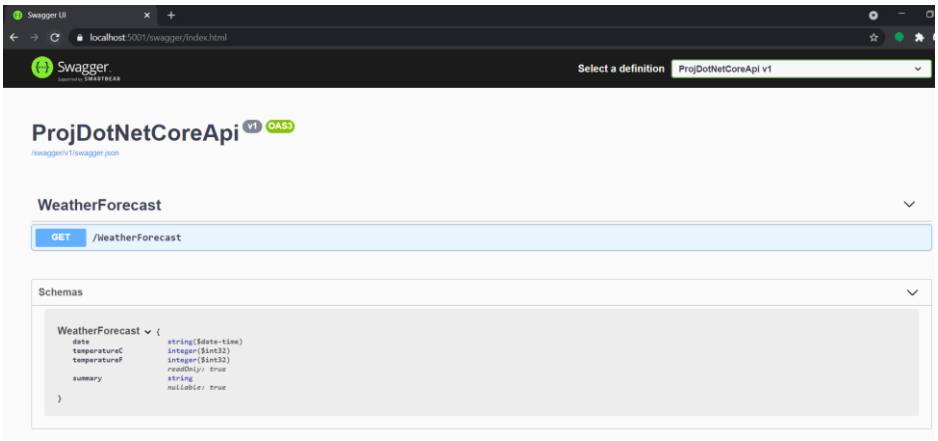
Neste momento os seguintes arquivos serão criados automaticamente:

- .vscode/launch.json
- .vscode/tasks.json

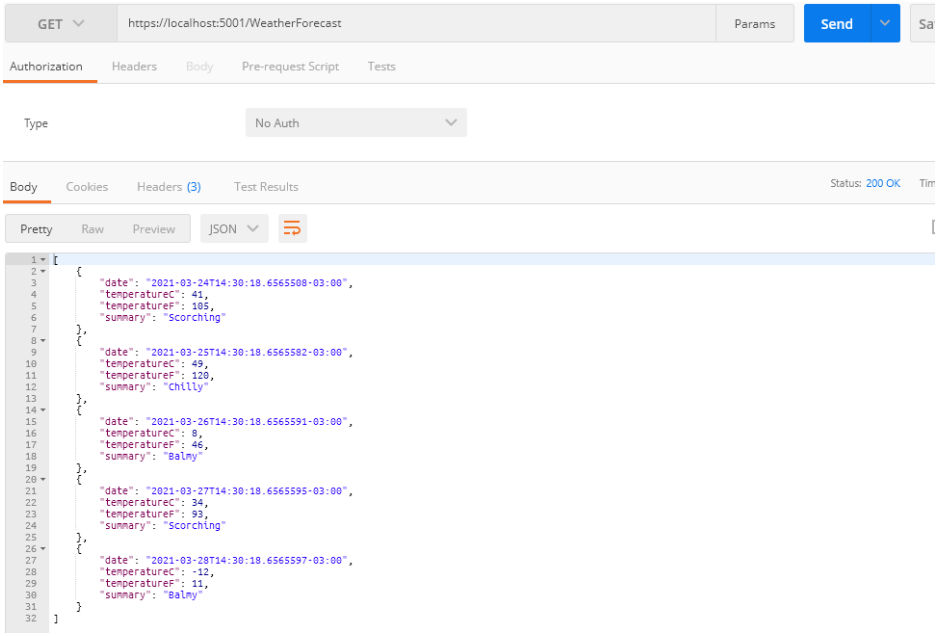
Após executar o projeto podemos utilizar o Swagger (documentação) para validar os endpoints existentes em nossa aplicação.

https://localhost:5001/swagger/index.html

Swagger:



E podemos chamar API através do Postman:



Após criar as classes: Pet, Dog e Breed

Criar a Classe (Dal) PetContext e fazer uma alteração na classe Startup.cs (Código nas próximas páginas)

Vamos instalar os seguintes pacotes com o objetivo de criar o Controller de forma automática:

- dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design
- dotnet add package Microsoft.EntityFrameworkCore.Design
- dotnet add package Microsoft.EntityFrameworkCore.SqlServer
- dotnet tool install -g dotnet-aspnet-codegenerator

Criar os Controllers:

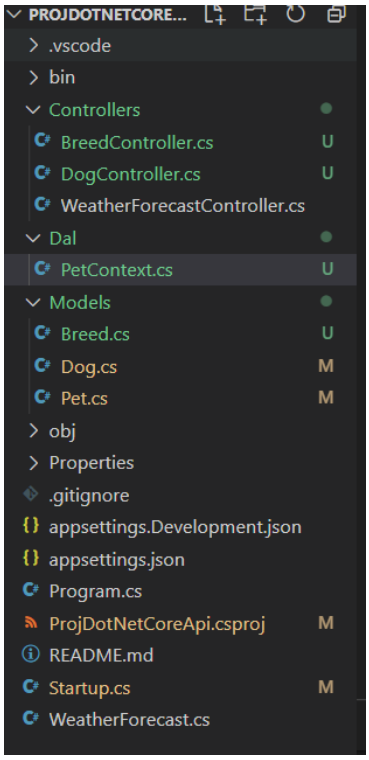
- dotnet aspnet-codegenerator controller -name **DogController** -async -api -m Dog -dc PetContext -outDir Controllers
- dotnet aspnet-codegenerator controller -name **BreedController** -async -api -m Breed -dc PetContext -outDir Controllers

Após a Implementação dos controllers temos os seguintes endpoints criados:

https://localhost:5001/swagger/index.html

Implementação do Projeto

Menu



Dog

```
public class Dog
    {
        #region Propriedades
            public int Id { get; set; }
            public string Name { get; set; }
            public virtual Breed Breed { get; set; }
        #endregion
    }
```

Breed

```
public class Breed
    {
        #region Propriedades
            public int Id { get; set; }
            public string Description { get; set; }
        #endregion
    }
```

PetContext

```
public class PetContext : DbContext
    {
        public PetContext(DbContextOptions<PetContext> options) : base(options)
        {  }
        public DbSet<Dog> Dogs {set; get;}
        public DbSet<Breed> Breeds {set; get;}

        protected override void OnModelCreating(ModelBuilder modelBuilder)
        {
            modelBuilder.Entity<Dog>()
            .HasOne(b => b.Breed);
            base.OnModelCreating(modelBuilder);
        }
    }
```

Startup

```
public IConfiguration Configuration { get; }

        // This method gets called by the runtime. Use this method to add services to
 the container.
        public void ConfigureServices(IServiceCollection services)
        {
            services.AddControllers();
            services.AddSwaggerGen(c =>
            {
 c.SwaggerDoc("v1", new OpenApiInfo { Title = "ProjDotNetCoreApi", Version = "v1" });
            });
            services.AddDbContext<PetContext>(options =>
         options.UseSqlServer(Configuration.GetConnectionString("SqlDbConnection")));
        }
```

appsettings.json

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Information",
      "Microsoft": "Warning",
      "Microsoft.Hosting.Lifetime": "Information"
    }
  },
  "ConnectionStrings": {
    "SqlDbConnection": "Data Source=(LocalDB)\\MSSQLLocalDB;Initial Catalog=dbdognew;
Integrated Security=True;"
  },
    "AllowedHosts": "*"
}
```

Ajustes no DogController

```
// GET: api/Dog
        [HttpGet]
        public async Task<ActionResult<IEnumerable<Dog>>> GetDogs()
        {
            //var dogs = await _context.Dogs.ToListAsync();
            return await _context.Dogs.Include(b => b.Breed).ToListAsync();;
        }
```

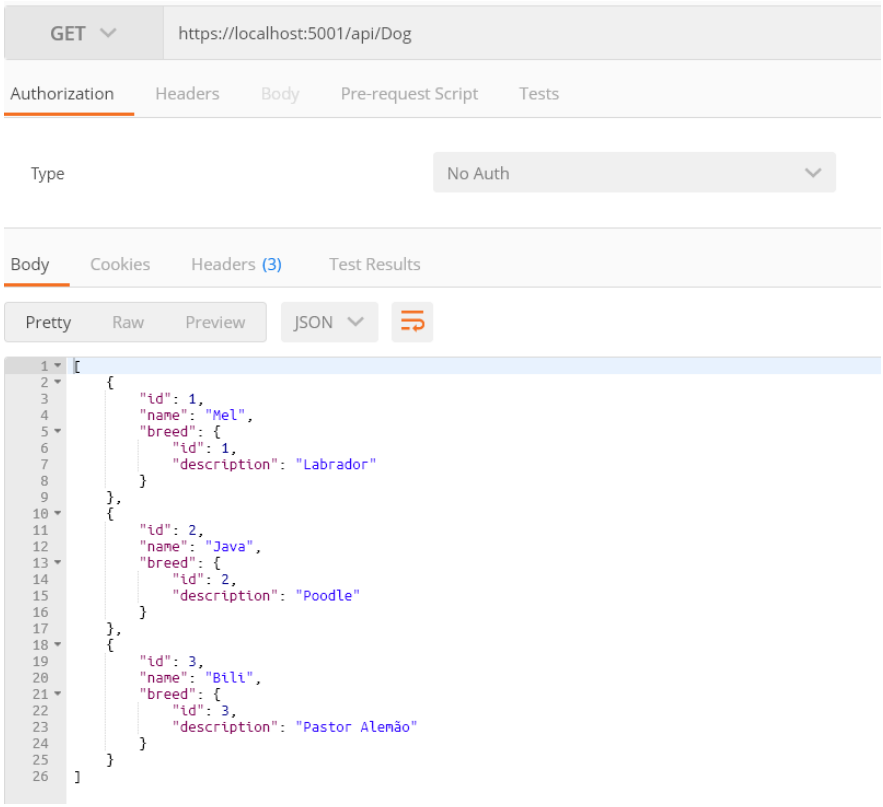
```
// GET: api/Dog/5
        [HttpGet("{id}")]
        public async Task<ActionResult<Dog>> GetDog(int id)
        {
           var dog = await _context.Dogs.Include(b => b.Breed).FirstOrDefaultAsync(d
=> d.Id == id);
            if (dog == null)
            {
                return NotFound();
            }
            return dog;
        }
```

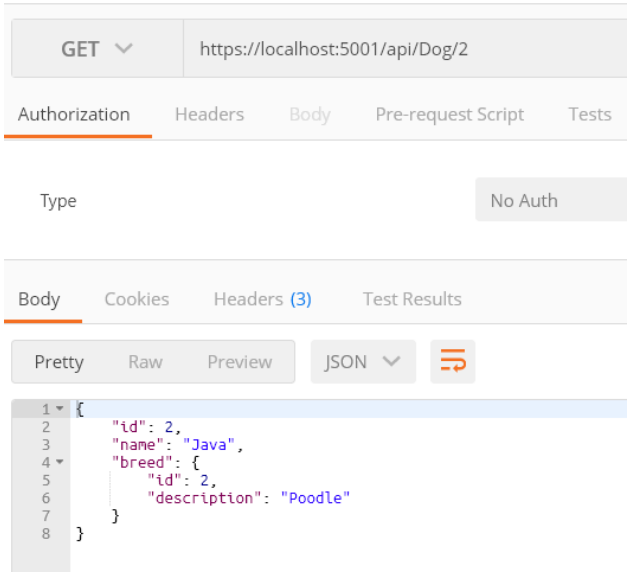
```
[HttpPost]
        public async Task<ActionResult<Dog>> PostDog(Dog dog)
        {

            dog.Breed = _context.Breeds.First(b => b.Id == dog.Breed.Id);
            _context.Dogs.Add(dog);
            await _context.SaveChangesAsync();

            return CreatedAtAction("GetDog", new { id = dog.Id }, dog);
        }
```

Gerar banco

- dotnet tool install --global dotnet-ef
- dotnet ef migrations add v1
- dotnet ef database update

Teste API





Referência

https://docs.microsoft.com/pt-br/aspnet/core/tutorials/first-web-api?view=aspnetcore-5.0&tabs=visual-studio