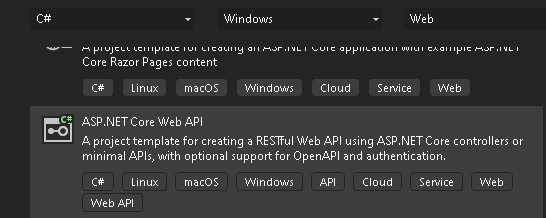
# Backend programozás

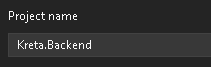
Ebben a feladatban egy C# backend alkalmazás beállítása történik.

## API projekt léterehozása

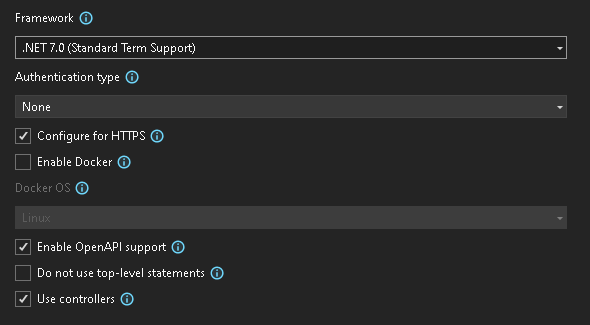
Válasszuk a Web API projektet:



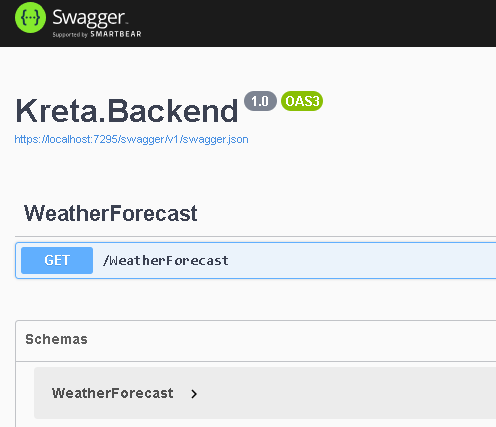
Megadjuk a projekt nevet:



Állítsuk be a projektet:



A projekt elindul:



## IP címek és cors beállítások

Állítsuk be a backend ip címeit:

launchSettings.json

"https": {

"commandName": "Project",

"dotnetRunMessages": true,

"launchBrowser": true,

"launchUrl": "swagger",

**"applicationUrl": "https://localhost:7120;http://localhost:7121",**

"environmentVariables": {

"ASPNETCORE\_ENVIRONMENT": "Development"

}

},

és a Cors beállításokat adjuk meg:

* [.NET CORS Guide: What It Is and How to Enable It (stackhawk.com)](https://www.stackhawk.com/blog/net-cors-guide-what-it-is-and-how-to-enable-it/)
* [Enabling CORS in ASP.NET Core By Example - Code Maze Blog (code-maze.com)](https://code-maze.com/enabling-cors-in-asp-net-core/)

public **static** void ConfigureCors(**this** IServiceCollection services)

{

services.AddCors(option =>

option.AddPolicy(name: "KretaCors",

policy =>

{

policy.WithOrigins("https://localhost:7020/")

.AllowAnyHeader()

.AllowAnyMethod();

}

)

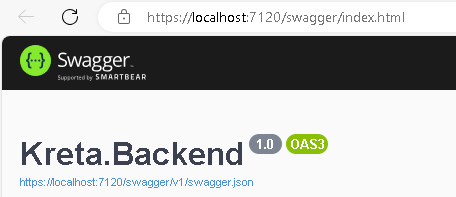
);

}

A Cors beállításokat a Program.cs állományban állítsuk is be:

builder.Services.ConfigureCors();

A program a megadott IP címen el is indul:



## DbContext

* [DbContext in Entity Framework Core (entityframeworktutorial.net)](https://www.entityframeworktutorial.net/efcore/entity-framework-core-dbcontext.aspx)
* [Entity Framework - DbContext (tutorialspoint.com)](https://www.tutorialspoint.com/entity_framework/entity_framework_dbcontext.htm)
* [Entity Framework Core - Getting Started (code-maze.com)](https://code-maze.com/entity-framework-core-getting-started/)
* [How to use EF Core as an in-memory database in ASP.NET Core 6 | InfoWorld](https://www.infoworld.com/article/3672154/how-to-use-ef-core-as-an-in-memory-database-in-asp-net-core-6.html)

KretaContext

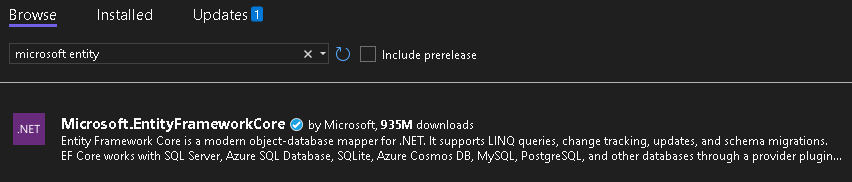
Készítsük el a Context mappába a KretaContext osztályt, amely a DbContext osztálytól öröklődik.

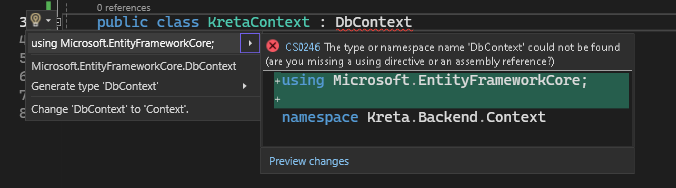
public class KretaContext : DbContext

{

}

A DbContext osztály akkor tudjuk használni ha telepítjük a „Microsoft.EntityFremework” csomagot:





A projektünket tesztadatokkal szeretnénk működtetni, ezért készítsük el a KretaInMemoryContext-et. Később készíthetünk az adatbázisunknak megfelelő Context-et is (pl. Mysql,…)

public class KretaInMemoryContext : KretaContext

{

protected override void OnModelCreating(ModelBuilder modelBuilder)

{

base.OnModelCreating(modelBuilder);

modelBuilder.Seed();

}

}

ModelBuilderExtension tesztadat felvitel lehetőséggel:

public static class ModelBuilderExtension

{

public static void Seed(this ModelBuilder modelBuilder)

{

// Ide kerülnek a tesztadatok

}

}

Ezután a service-k között is felverhetjük a KretaInMemoryContext-et

public static void ConfigureInMemoryContext(this IServiceCollection services)

{

string dbNameInMemoryContext = "Kreta" + Guid.NewGuid();

services.AddDbContext<KretaInMemoryContext>

(

options => options.UseInMemoryDatabase(databaseName: dbNameInMemoryContext),

ServiceLifetime.Scoped,

ServiceLifetime.Scoped

);

}

A megfelelő működéshez a KretaContex-be és a KretaInMemoryContextbe is el kell készíteni a nem alapértelemezett konstruktort:

public class KretaInMemoryContext : KretaContext

{

public KretaInMemoryContext(DbContextOptions options) : base(options)

{

}

…

public class KretaContext : DbContext

{

public KretaContext(DbContextOptions options) : base(options)

{

}

…

## Konfigurációs feladatok

### Tesztadatok betöltése a program indításakor

var app = builder.Build();

// InMemory database data

using (var scope = app.Services.CreateAsyncScope())

{

var dbContext = scope.ServiceProvider.GetRequiredService<KretaInMemoryContext>();

// InMemory test data

dbContext.Database.EnsureCreated();

}

### A használt CORS beállítása

// Cors

app.UseCors("KretaCors");