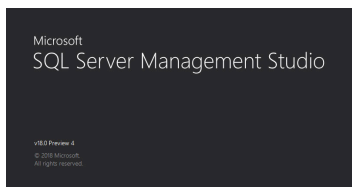


Creación de un API con arquitectura Hexagonal



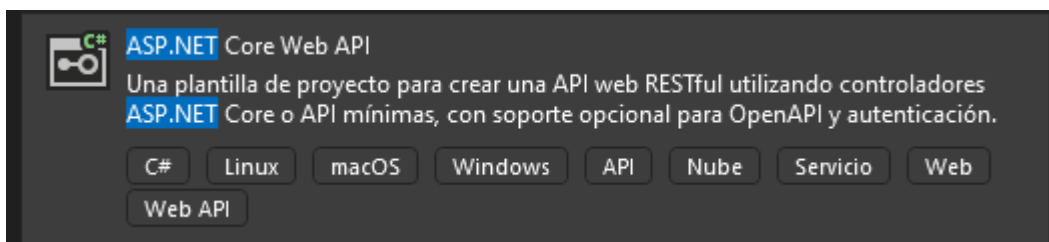
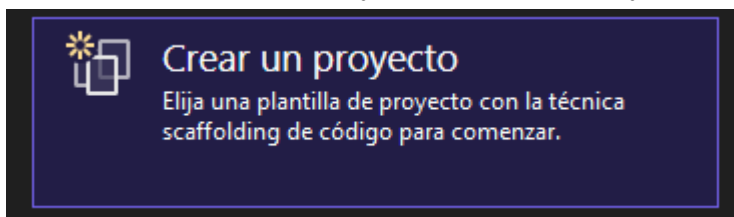
<https://visualstudio.microsoft.com/es/>



<https://learn.microsoft.com/es-es/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver16>

Como crear una API

Vamos a abrir visual studio y crear un nuevo proyecto “ASP.NET core Web API”



Configure su nuevo proyecto

ASP.NET Core Web API

C#

Linux

macOS

Windows

API

Nube

Servicio

Web

Web API

Nombre del proyecto

aspDotNetBlankProject

Ubicación

C:\Users\docto\source\repos

Nombre de la solución ⓘ

aspDotNetBlankProject

☐

Colocar la solución y el proyecto en el mismo directorio

Proyecto se creará en "C:\Users\docto\source\repos\aspDotNetBlankProject\aspDotNetBlankProject\"

Información adicional

ASP.NET Core Web API

C#

Linux

macOS

Windows

API

Nube

Servicio

Web

Web API

Framework ⓘ

.NET 8.0 (Compatibilidad a largo plazo)

Authentication de campo ⓘ

Ninguno

☒

Configurar para HTTPS ⓘ

☐

Habilitar compatibilidad con el contenedor ⓘ

SO del contenedor ⓘ

Linux

Tipo de compilación de contenedor ⓘ

Dockerfile

☒

Habilitar compatibilidad con OpenAPI ⓘ

☐


No usar instrucciones de nivel superior ⓘ

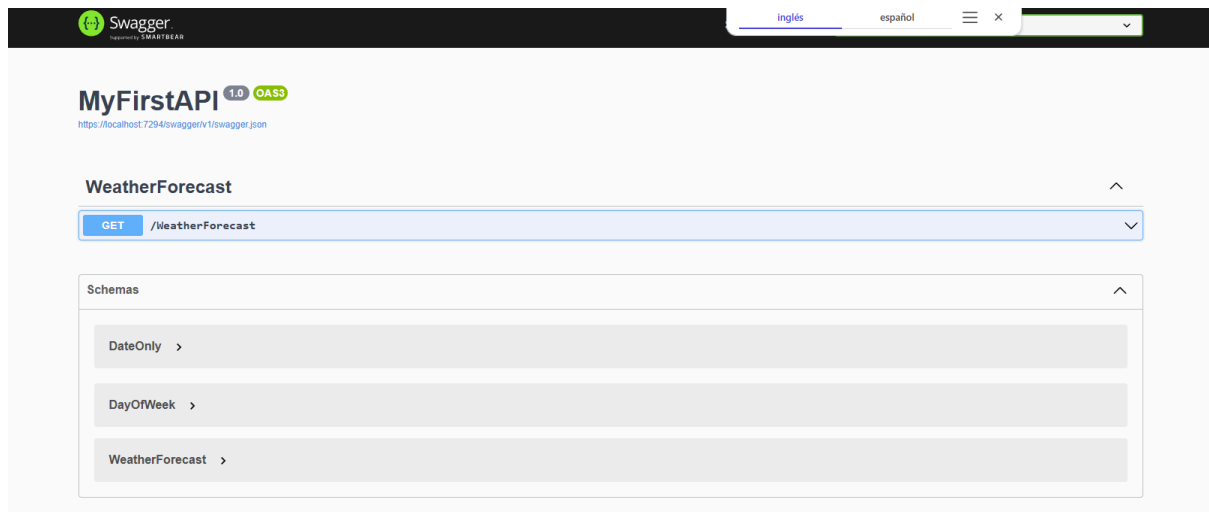
☒

Utilizar controladores ⓘ

☐

Inscribirse en la orquestación de .NET Aspire ⓘ

Una vez creado lo podemos ejecutar al darle play  [https](https://) y nos enviará a un swagger con el código de ejemplo del clima, este código lo vamos a borrar.

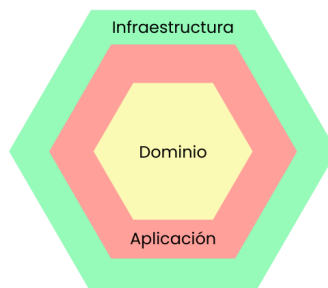


Borrar los siguientes archivos:

aspDotNetBlankProject\aspDotNetBlankProject\WeatherForecast.cs

aspDotNetBlankProject\aspDotNetBlankProject\Controllers\WeatherForecastController.cs

La arquitectura hexagonal



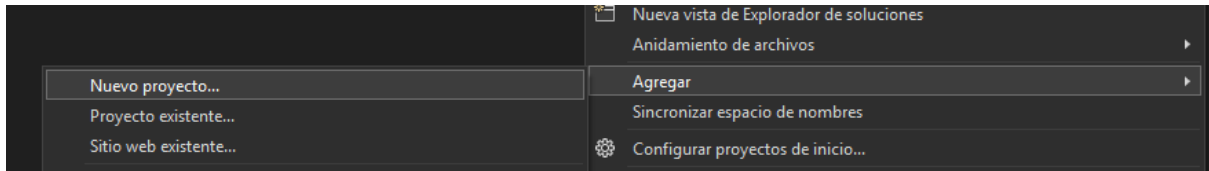
Esta arquitectura consta de 3 capas:

- Dominio: Resolver la lógica de negocio.
- Aplicación: Intermediar entre el dominio y la capa de infraestructura.
- Infraestructura: Conectarse a base de datos.

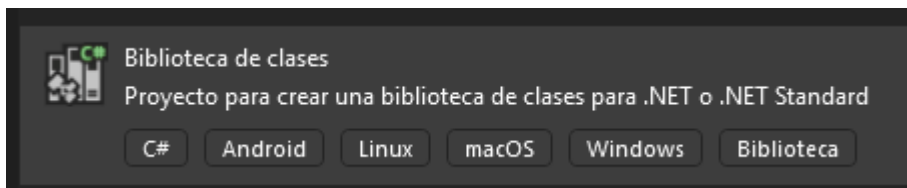
Como crear una capa

Una capa es un proyecto que tiene una única responsabilidad, vamos a crear nuestras 3 capas

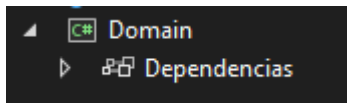
Basta solo con darle click derecho a la solución y agregar un nuevo proyecto:



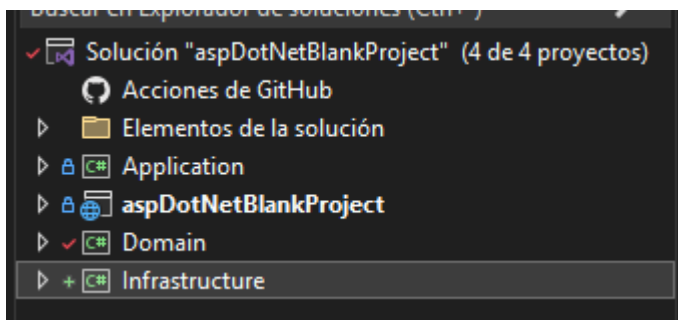
Y Agregamos una biblioteca de clases:



Por ejemplo creamos la capa de dominio el cual vendrá con un archivo de ejemplo el cual procederemos a borrar.



Y luego creamos nuestras capas de Infraestructure y Application



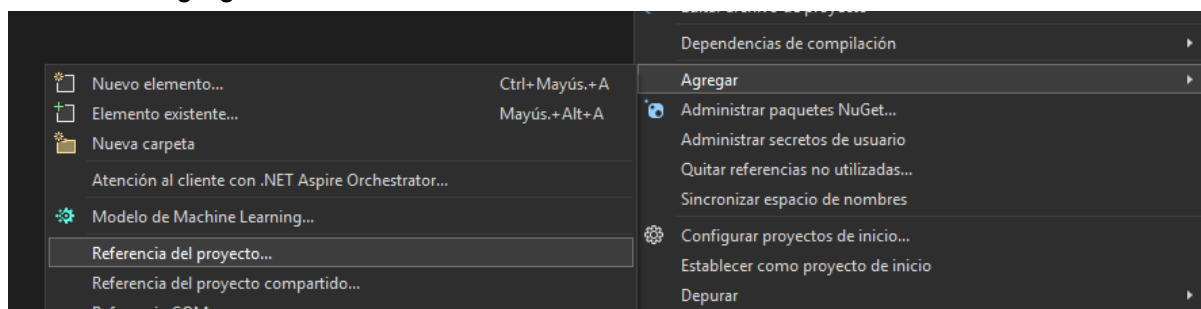
Ahora vamos a definir las capas internas y su responsabilidad:

Nr	Tipo	Ubicación	Descripción y responsabilidad.
01	Entidad de Dominio	Domain	<p>Es la encargada de contener todas las entidades del dominio de negocio como por ejemplo: Usuarios, Clientes, ventas, Productos.</p> <p>Su responsabilidad es modelar las entidades de bases de datos.</p>
02	Puertos	Domain	<p>Serán los encargados de conectar las capas e inyectar dependencias.</p> <p>Su responsabilidad es asegurar que los repositorios tengan una consistencia de métodos.</p>
03	Servicio	Domain	<p>Será el encargado de usar los repositorios para manipular la información.</p> <p>Su responsabilidad es ejecutar la lógica de los casos de uso.</p>
04	Entidades de aplicación	Application	<p>Se utilizará el patrón CQRS para modelar los récords y los handler. Cada 01 tendrá su propia carpeta que a su vez va a contener 2 carpetas: "Commands" y "Querys".</p> <p>Su responsabilidad será inyectar servicios y resolver lógica de operaciones con entidades.</p>
05	Adaptador	Infrastructure	<p>Son los repositorios e implementan el contrato pactado en 02.</p> <p>Su responsabilidad será hacer valer el contrato y ser inyectados.</p>
06	Conector a base de datos	Infrastructure	<p>Se encargará de usar la librería MSEntityFrameworkCore para:</p> <ul style="list-style-type: none"> ● Conectarse a la base de datos. ● Registrar por cada entidad su respectiva tabla en base de datos. ● <p>Su responsabilidad será conectar y modelar la base de datos.</p>

Cómo conectar las capas

Las capas no trabajan independientemente, cada capa necesita conectarse a otra para traer por ejemplo: un modelo, servicio o inyectar.

Para conectar una capa con otra basta con darle click derecho a nuestra capa y luego seleccionar agregar.

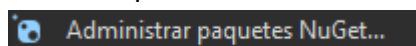


Las conexiones que realizaremos son:

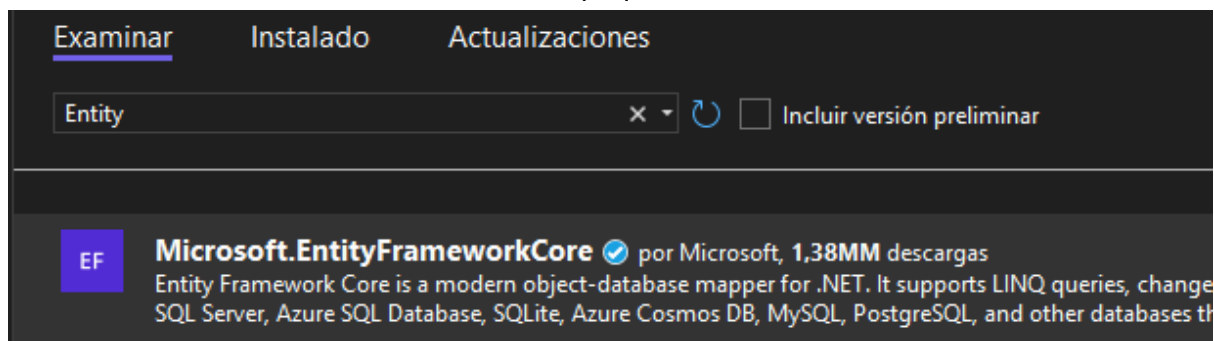
- aspDotNetBliankProject: infraestructura.
- Infraestructura: dominio

Instalación de paquetes NuGet

A cada capa vamos a darle click derecho y luego administración de paquetes “NuGet”.



Y se nos abrirá la ventana de instalación de paquetes, vamos a darle click a examinar:



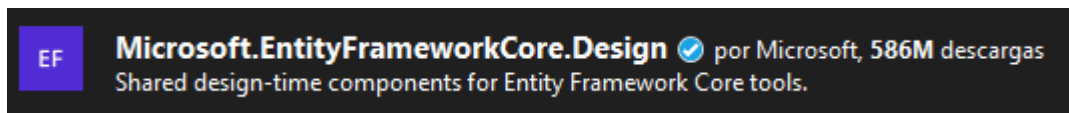
Y vamos a instalar los siguientes paquetes NuGet en las capas correspondientes:



Advertencia: en caso de fallar en el número de versión de instalación de NuGet basta con desinstalar y volver a instalar.

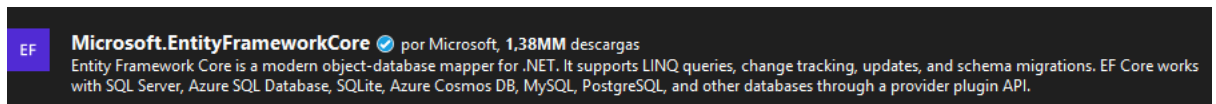
- **aspDotNetBlankProject:**

Microsoft.EntityFrameworkCore.Design
8.0.0

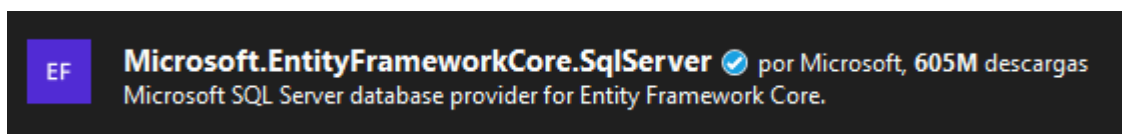


- **Infrastructure:**

Microsoft.EntityFrameworkCore
8.0.0



Microsoft.EntityFrameworkCore.SqlServer
8.0.0



Creación de nuestra ENTIDAD de Ejemplo

Vamos a crear una entidad de ejemplo:

Example(Id, Title, Description, Information, IsDelete)

La cual nos servirá para practicar de ahora en adelante como realizar la escritura en base de datos usando solo los modelos.

aspDotNetBlankProject\Domain\Entities\Example.cs

```
namespace Domain.Entities
{
    2 referencias
    public class Example
    {
        0 referencias
        public int Id { get; set; }
        0 referencias
        public string Title { get; set; } = string.Empty;
        0 referencias
        public string Description { get; set; } = string.Empty;
        0 referencias
        public string Information { get; set; } = string.Empty;
        0 referencias
        public bool IsDeleted { get; set; }
    }
}
```

Creación del contexto “mapeador a Base de Datos”

En la actualidad existe un paquete NuGet llamado: Microsoft.EntityFrameworkCore el cual se encarga de:

- Dados los modelos convertirlos a tablas de bases de datos sin escribir código SQL.
- Conectarse a la base de datos haciendo la configuración e inyección en la main del proyecto.

aspDotNetBlankProject\Infraestructure\Context\DbContext.cs


```

using Microsoft.EntityFrameworkCore;
using Domain.Entities;

namespace MyFirstAPI.Infraestructure
{
    2 referencias
    public class PersitenceContext : DbContext
    {
        0 referencias
        public PersitenceContext(DbContextOptions<PersitenceContext> options) : base(options)
        {
        }

        //Mapers
        0 referencias
        public DbSet<Example> examples { get; set; }

        //Create entity in DB
        0 referencias
        protected override void OnModelCreating(ModelBuilder builder)
        {
            builder.Entity<Example>().ToTable("examples");
        }
    }
}

```

El código que observamos se encarga de inyectar un conector de base de datos y mapear nuestras entidades a tablas de base de datos.

Registrar e inyectar nuestro contexto de base de datos

Nuestro archivo main es aspDotNetBlankProject\aspDotNetBlankProject\Program.cs allí nosotros tenemos que gestionar la inyección de dependencias, tenemos que hacerlo justo antes de la línea de código en donde se construye la APP

```
var app = builder.Build();
```

Nosotros procederemos a realizar la siguiente operación para poder dejar nuestro contexto inyectable.

Paso 0 para registrar la conexión a base de datos importar:

```

using Microsoft.EntityFrameworkCore;
using MyFirstAPI.Infraestructure;

```

Paso 1 declarar la conexión con el string de conexión a SQLserver:

```

builder.Services.AddSwaggerGen();
var connectionString = "Server=localhost\\MSSQLSERVER01;Database=master;Trusted_Connection=True;TrustServerCertificate=True;";

builder.Services.AddDbContext<PersitenceContext>(options =>
    options.UseSqlServer(connectionString));

var app = builder.Build();

```

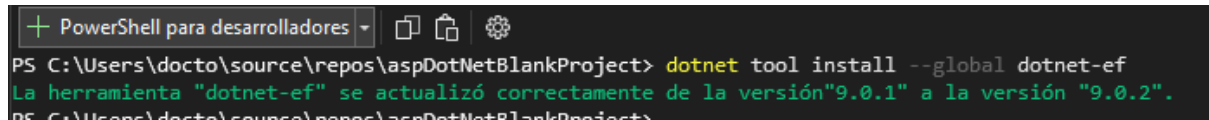
Ejecutar una “MIGRACIÓN”

Convertir entidades en tablas de bases de datos e insertar

Lo que vamos a hacer a continuación es a través de la consola ejecutar comandos que harán que nuestras entidades se conviertan en tablas de bases de datos y luego esas tablas procederemos a meterlas dentro de nuestra base de datos.

Paso 1 instalar las herramientas de ejecución de Entity Framework:

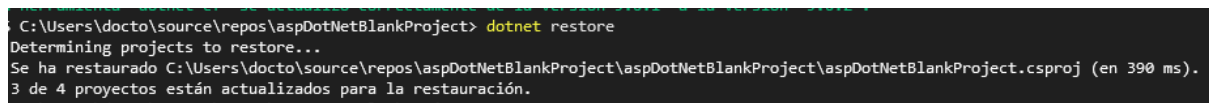
dotnet tool install --global dotnet-ef



```
+ PowerShell para desarrolladores
PS C:\Users\docto\source\repos\aspDotNetBlankProject> dotnet tool install --global dotnet-ef
La herramienta "dotnet-ef" se actualizó correctamente de la versión "9.0.1" a la versión "9.0.2".
PS C:\Users\docto\source\repos\aspDotNetBlankProject>
```

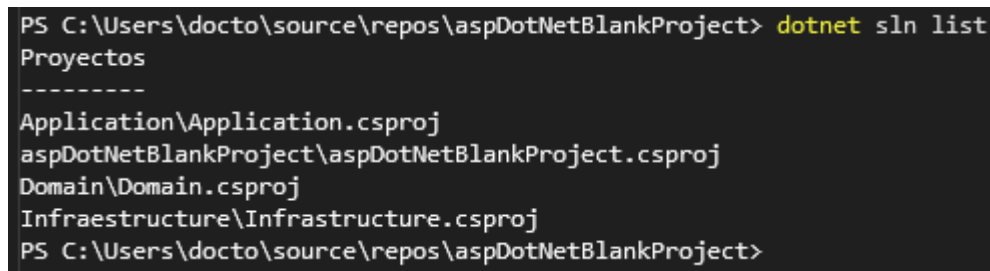
paso 2 restaurar dependencias:

dotnet restore



```
C:\Users\docto\source\repos\aspDotNetBlankProject> dotnet restore
Determining projects to restore...
Se ha restaurado C:\Users\docto\source\repos\aspDotNetBlankProject\aspDotNetBlankProject\aspDotNetBlankProject.csproj (en 390 ms).
3 de 4 proyectos están actualizados para la restauración.
```

Paso 4 listar los proyectos:



```
PS C:\Users\docto\source\repos\aspDotNetBlankProject> dotnet sln list
Proyectos
-----
Application\Application.csproj
aspDotNetBlankProject\aspDotNetBlankProject.csproj
Domain\Domain.csproj
Infraestructure\Infraestructure.csproj
PS C:\Users\docto\source\repos\aspDotNetBlankProject>
```

Paso 5: limpiar y construir el proyecto:

dotnet clean

dotnet restore

dotnet build

Paso 3 ejecutar la migración (leer modelos construir tablas sql):

dotnet ef migrations add InitialCreate --project Infraestructure/Infraestructure.csproj
--startup-project aspDotNetBlankProject/aspDotNetBlankProject.csproj

```
C:\Users\docto\source\repos\aspDotNetBlankProject> dotnet ef migrations add InitialCreate --project Infrastructure/Infrastructure.csproj --startup-project aspDotNetBlankProject/aspDotNetBlankProject.csproj
ld started...
ld succeeded.
e. To undo this action, use 'ef migrations remove'
C:\Users\docto\source\repos\aspDotNetBlankProject>
```

paso 4 con las tablas construidas en la migración actualizar los valores de la base de datos:

dotnet ef database update --project Infrastructure/Infrastructure.csproj --startup-project
aspDotNetBlankProject/aspDotNetBlankProject.csproj

```
C:\Users\docto\source\repos\aspDotNetBlankProject> dotnet ef database update --project Infrastructure/Infrastructure.csproj --startup-project aspDotNetBlankProject/aspDotNetBlankProject.csproj
ld started...
ld succeeded.
: Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (11ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
  SELECT 1
: Microsoft.EntityFrameworkCore.Database.Command[20101]
  Executed DbCommand (15ms) [Parameters=[], CommandType='Text', CommandTimeout='30']
```