.Net Clean Architecture

Creando Proyecto

dotnet new sln -o Proyecto // crea el proyecto solución

Creación de las tres capas básicas para una arquitectura limpia

NOTA: elimina en un principio la clase que aparece por defecto y la carpeta obj

dotnet new classlib -o Domain -f net7.0

dotnet new classlib -o Application -f net7.0

dotnet new classlib -o Infrastructure-f net7.0

Vista:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Crea la API (no borres nada de ella)

dotnet new webapi -o Web.API -f net7.0

NOTA: comprobar que todo esta bien hasta el momento 🡪 dotnet build

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Establecer relaciones (como, por ejemplo):

dotnet add Application/Application.csproj reference .\Domain\Domain.csproj

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

Comprobamos que todo está bien

dotnet build

Arrancar la API

dotnet run -p .\Web.API\

Vista:

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

**NOTA**: abre el navegador en el localhost, es verdad que no aparecerá nada, debes de añadir a la url el swagger:

<http://localhost:5229/swagger/index.html>

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

# Definir el dominio

Añadir primitivos

* Creas una carpeta primitivos en dominio
* Creas una clase AggregateRoot.cs
* Creas una clase DomainEvents.cs
  + Instala Nuget Gallery – MediatR- instalar en Domain y en Application
* Creas una Interfaz UnitOfWork.cs

Captura de pantalla de computadora

El contenido generado por IA puede ser incorrecto.

**AggregateRoot.cs**

namespace Domain.Primitivos;

public abstract class AggregateRoot

{

    // lista de eventos de dominio

    private readonly List<DomainEvent> \_domainEvents = new();

    // propiedad de solo lectura para acceder a la lista de eventos

    public ICollection<DomainEvent> DomainEvents => \_domainEvents;

    // método para levantar eventos de dominio

    protected void Raise(DomainEvent domainEvent) => \_domainEvents.Add(domainEvent);

}

**DomainEvent.cs**

using MediatR;

namespace Domain.Primitivos;

public record DomainEvent(Guid Id) : INotification;

**IUnitOfWork.cs**

namespace Domain.Primitivos;

public interface IUnitOfWork

{

    // método para guardar los cambios en la base de datos

    Task<bool> SaveChangesAsync(CancellationToken cancellationToken = default);

}