DESARROLLO PROFESIONAL CON ASP.NET CORE

.NET

Microsoft® SQL Server®

Modelo en Base Datos

CleanArchitecture.**Application**

- MediatR.Extensions.Microsoft.DependencyInjection
- FluentValidation
- FluenteValidation.DependencyInjectionExtensions
- AutoMapper
- AutoMapper.Extensions.Microsoft.Depedency
- Microsoft.Extensions.Logging.Abstractions

- Referencia de proyecto desde CleanArchitecture.Domain

CleanArchitecture.**Infrastructure**

- Microsoft.EntityFrameworkCore.SqlServer
- SendGrid

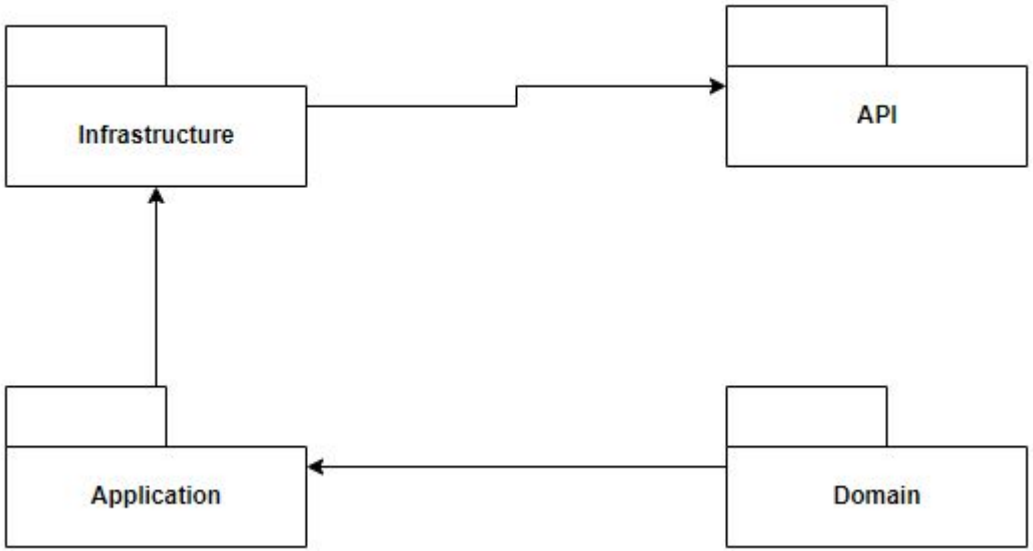
- Referencia de proyecto desde CleanArchitecture.Application

CleanArchitecture.**API**

- Microsoft.EntityFrameworkCore.Tools

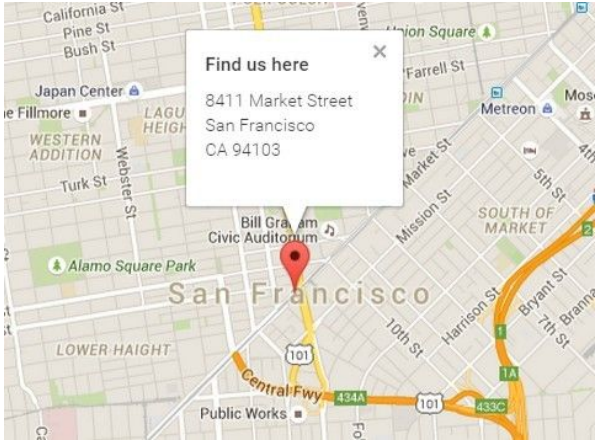
- Referencia de proyecto desde CleanArchitecture.Infrastructure
- Referencia de proyecto desde CleanArchitecture.Application

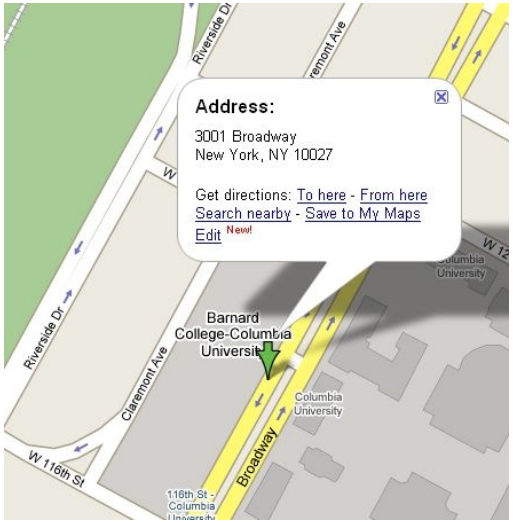
Solucion Referencias



Value Objects en Domain Driven Design

Los valores o estados de sus propiedades lo hacen unico





Func<T> vs Expression<Func<T>> y Linq

IEnumerable

in-memory

Func<T>

Where(Func<T, bool> predicate)

Where(x => x.property == "value")

IQueryable

SQL Server

Expression<Func<T>>

Where(Expression<Func<T, bool>> predicate)

Where(x => x.property == "value")

Func<T> vs Expression<Func<T>> y Linq

Func<Video, string>

Delegate

```
class Video {
   public string? Nombre { get; set; }
}
```

```
var videos = new List<Video>
{
        new Video{ Nombre = "matrix"},
        new Video{ Nombre = "mad max"},
        new Video{ Nombre = "avatar"}
}
```

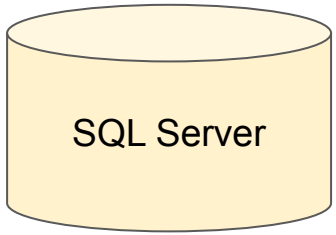
```
Func<Video, string> selector = video => "Pelicula :" + video.Nombre;
```

```
IEnumerable<string>  videoTitulos = videos.Select(selector);

foreach(string titulos in videoTitulos ){
        Console.WriteLine(titulos);
}
```

Func<T> vs Expression<Func<T>> y Linq

Expression Func<Video, bool>

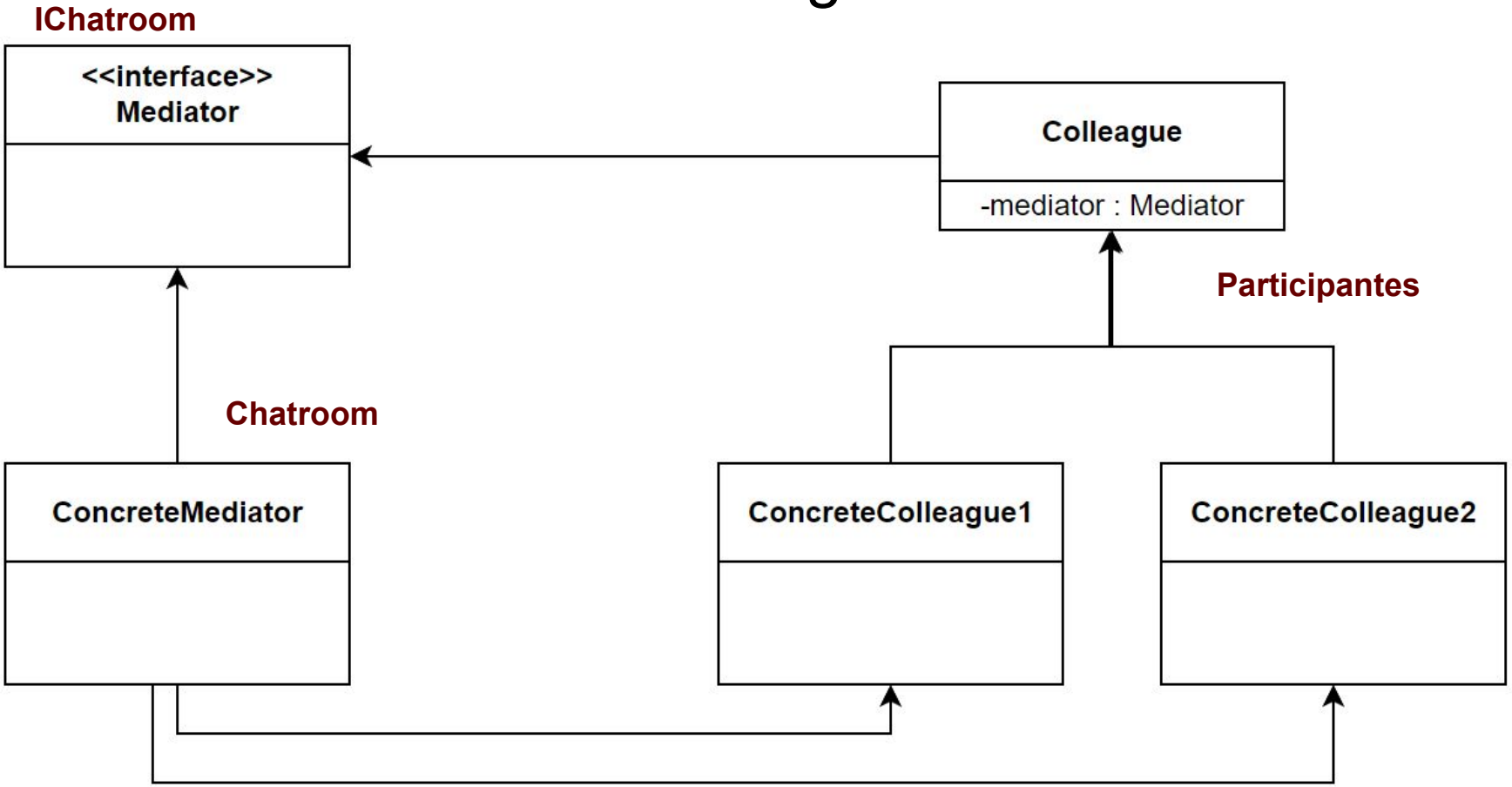
```
class Video {
   public string? Nombre { get; set; }
}
```



Expresion<Func<Video, bool>> expression = u => u.Nombre == "matrix"

```
var videos = dbContext.Videos.Where(expression).Select(u=>u).ToList();

foreach(string titulos in videoTitulos ){
        Console.WriteLine(titulos);
}
```

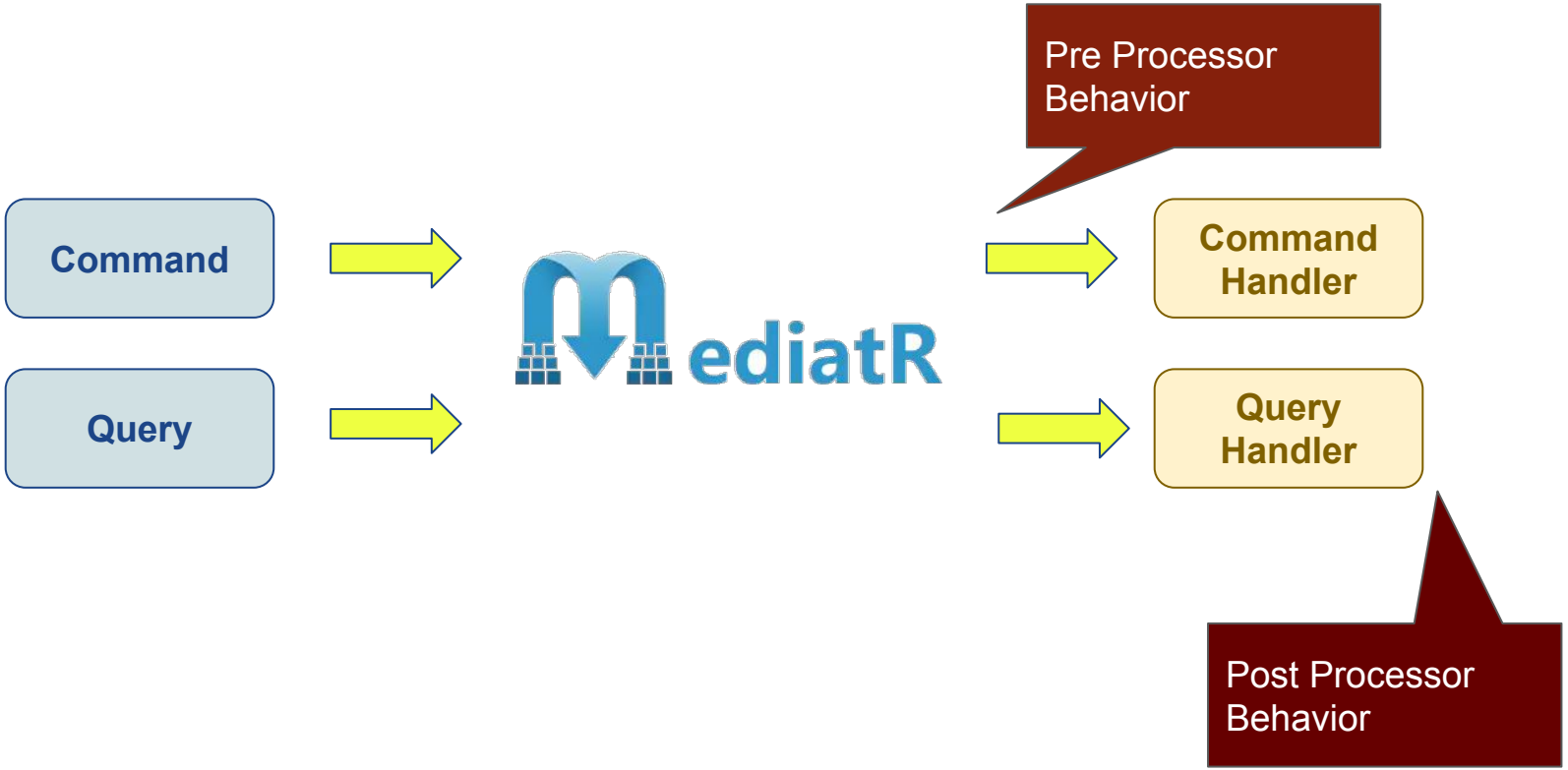
Mediator Design Pattern



Mediator Design Pattern

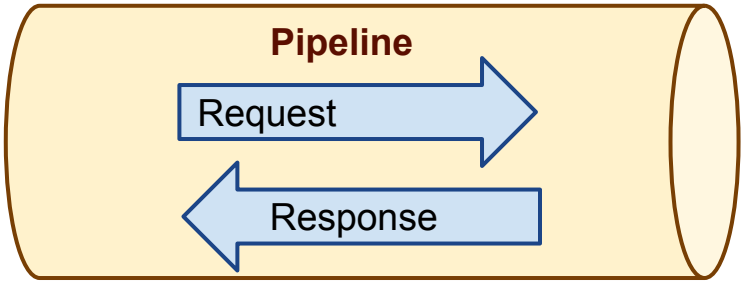
Fluent*
Validation

Mediator Design Pattern



Behaviours

Cliente



Command

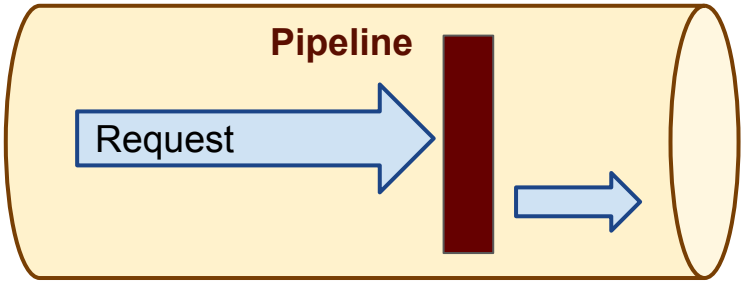
Query

MediatR

MediatR Behaviours

Mediator Design Pattern

Cliente



Command

Query

MediatR Behaviours

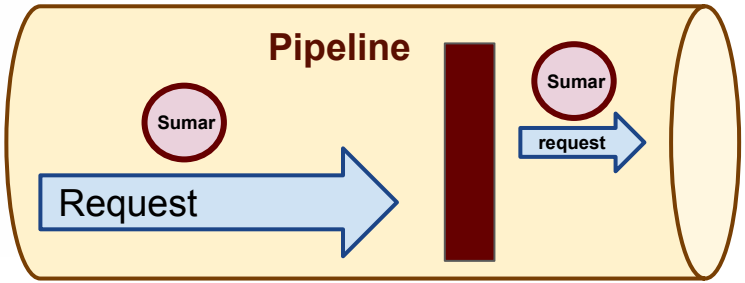
```
3 references
public class CreateStreamerCommandHandler : IRequestHandler<CreateStreamerCommand, int>
{
    private readonly IStreamerRepository _streamerRepository;
    private readonly IMapper _mapper;
    private readonly IEmailService _emailservice;
    private readonly ILogger<CreateStreamerCommandHandler> _logger;

    0 references
    public CreateStreamerCommandHandler(IStreamerRepository streamerRepository, IMapper mapper, IEmailService emailservice, ILogger<CreateStreamerCommandHandler> logger)
    {
        _streamerRepository = streamerRepository;
        _mapper = mapper;
        _emailservice = emailservice;
        _logger = logger;
    }

    0 references
    public async Task<int> Handle(CreateStreamerCommand request, CancellationToken cancellationToken)
    {
        var streamerEntity = _mapper.Map<Streamer>(request);
        var newStreamer = await _streamerRepository.AddAsync(streamerEntity);

        _logger.LogInformation($"Streamer {newStreamer.Id} fue creado existosamente");

        await SendEmail(newStreamer);

        return newStreamer.Id;
    }

    1 reference
    private async Task SendEmail(Streamer streamer)
    {
        var email = new Email
        {
            To = "vaxi.drez.social@gmail.com",
            Body = "La compania de streamer se creo correctamente",
            Subject = "Mensaje de alerta"
        };

        try
        {
            await _emailservice.SendEmail(email);
        }
        catch (Exception ex) {
            _logger.LogError($"Errores enviando el email de {streamer.Id}");
        }

    }

}
```

AddTransient

AddScoped

AddSingleton

Cliente



Command

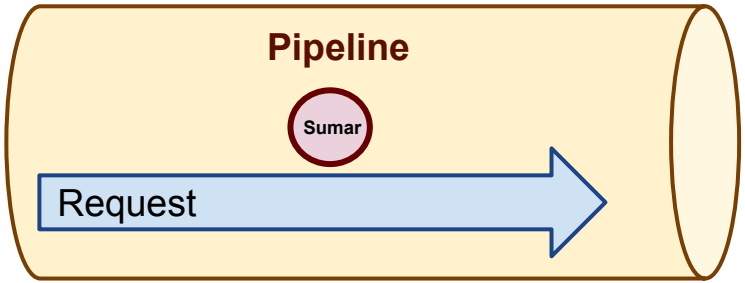
Query

AddTransient

AddScoped

AddSingleton

Cliente



Command

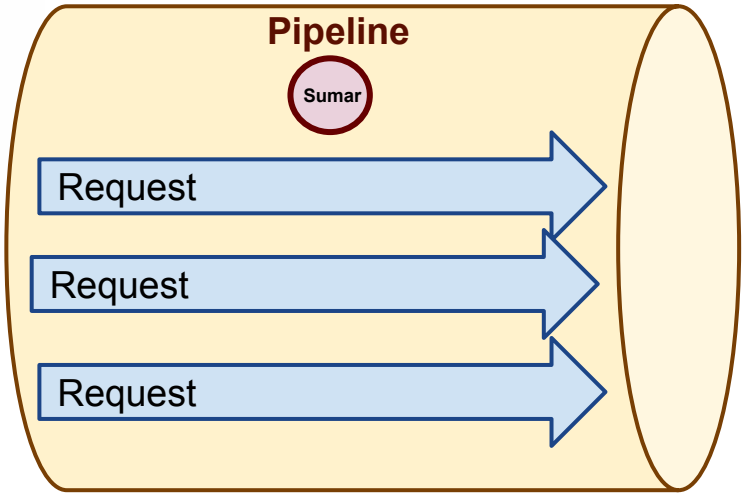
Query

AddTransient

AddScoped

AddSingleton

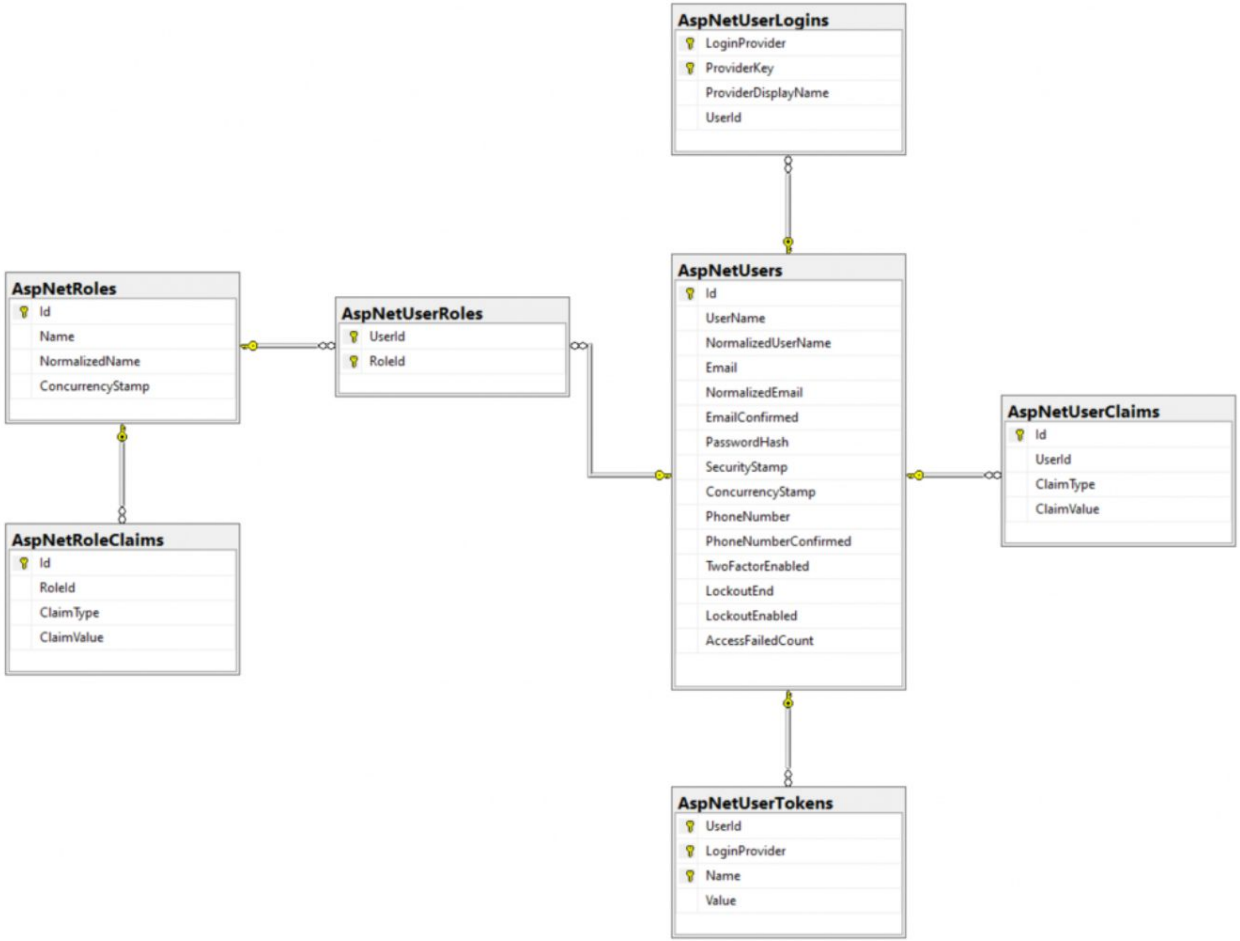
Cliente



Command

Query

Tipo Servicio	En el ambiente pipeline de un mismo http request	En diferentes http requests
Transient	New Instance	New Instance
Scoped	Same Instance	New Instance
Singleton	Same Instance	Same Instance


AspNetUserLogins
LoginProvider
ProviderKey
ProviderDisplayName
UserId
AspNetRoles
Id
Name
NormalizedName
ConcurrencyStamp
AspNetUserRoles
UserId
RoleId
AspNetUsers
Id
UserName
NormalizedUserName
Email
NormalizedEmail
EmailConfirmed
PasswordHash
SecurityStamp
ConcurrencyStamp
PhoneNumber
PhoneNumberConfirmed
TwoFactorEnabled
LockoutEnd
LockoutEnabled
AccessFailedCount
AspNetUserClaims
Id
UserId
ClaimType
ClaimValue
AspNetRoleClaims
Id
RoleId
ClaimType
ClaimValue
AspNetUserTokens
UserId
LoginProvider
Name
Value

Messages en CQRS y Event Sourcing

👉 **Commands**

Events

Queries

Messages en CQRS y Event Sourcing

👉 **Commands**

Events

Queries

AbrirCuentaAhorrosCommand

DepositarDineroCommand

Messages en CQRS y Event Sourcing

Commands

👉 **Events**

Queries

Messages en CQRS y Event Sourcing

Commands

👉 **Events**

Queries

El origen de los events son los Aggregate

Los aggregate disparan eventos

Messages en CQRS y Event Sourcing

Commands

👉 **Events**

Queries

CuentaCorrienteCreadaEvent

DineroDepositadoEvent