Capacitación .NET

Sesión 02

AVANTICA
an encora company

Temas

1. Migrations
2. Dependency Injection & IoC
3. CQRS & Mediator Patterns

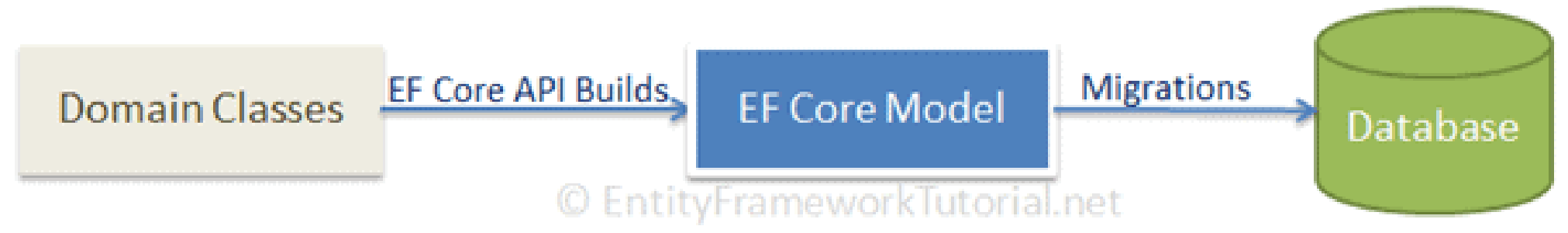
Expone: Juan Alberto Carlos Vera

AVANTICA
an encora company

Migrations

Migraciones

Al desarrollar aplicaciones, es probable que el modelo de datos cambie a medida que surgen nuevos requisitos.
La función de migraciones le permite realizar cambios en su modelo y luego propagar esos cambios al esquema de su base de datos.
Las migraciones están habilitadas de forma predeterminada en EF Core.
Se gestionan mediante la ejecución de comandos.



AVANTICA
an encora company

Migrations

Migraciones:

- Crear una migración.
- Eliminar una migración.
- Aplicar una migración.
- Revertir una migración.
- Aplicar una migración a una base de datos remota.
- Ejecución de SQL personalizado.
- Dirigirse a varios proveedores.
- Configuración del tiempo de espera de la migración.

Dependency Injection & IoC

Inyección de dependencia - DI

Es un patrón de diseño que se utiliza para evitar el acoplamiento dentro de su código. De acuerdo con las mejores prácticas en el desarrollo de software y parte de uno de los principios SOLID, una clase debe depender de abstracciones y no de clases concretas.

```
2 references
public class LeaderboardServiceManager : ILeaderboardService
{
    private readonly IMapper _mapper;

    0 references
    public LeaderboardServiceManager(IMapper mapper) =>
        _mapper = mapper;

    1 reference
    public Task<IEnumerable<LeaderboardDto>> Recover()
    {
        var leaderboard = new List<Leaderboard>();

        var leaderboardDto = _mapper.Map<IEnumerable<LeaderboardDto>>(leaderboard);

        return Task.FromResult(leaderboardDto);
    }
}
```

AVANTICA
an encora company

Dependency Injection & IoC

Beneficios:

- Acoplamiento bajo
- Sencillez
- Facilidad de mantenimiento
- Facilidad para aprender sobre el código / proyecto
- Fácil de probar su código

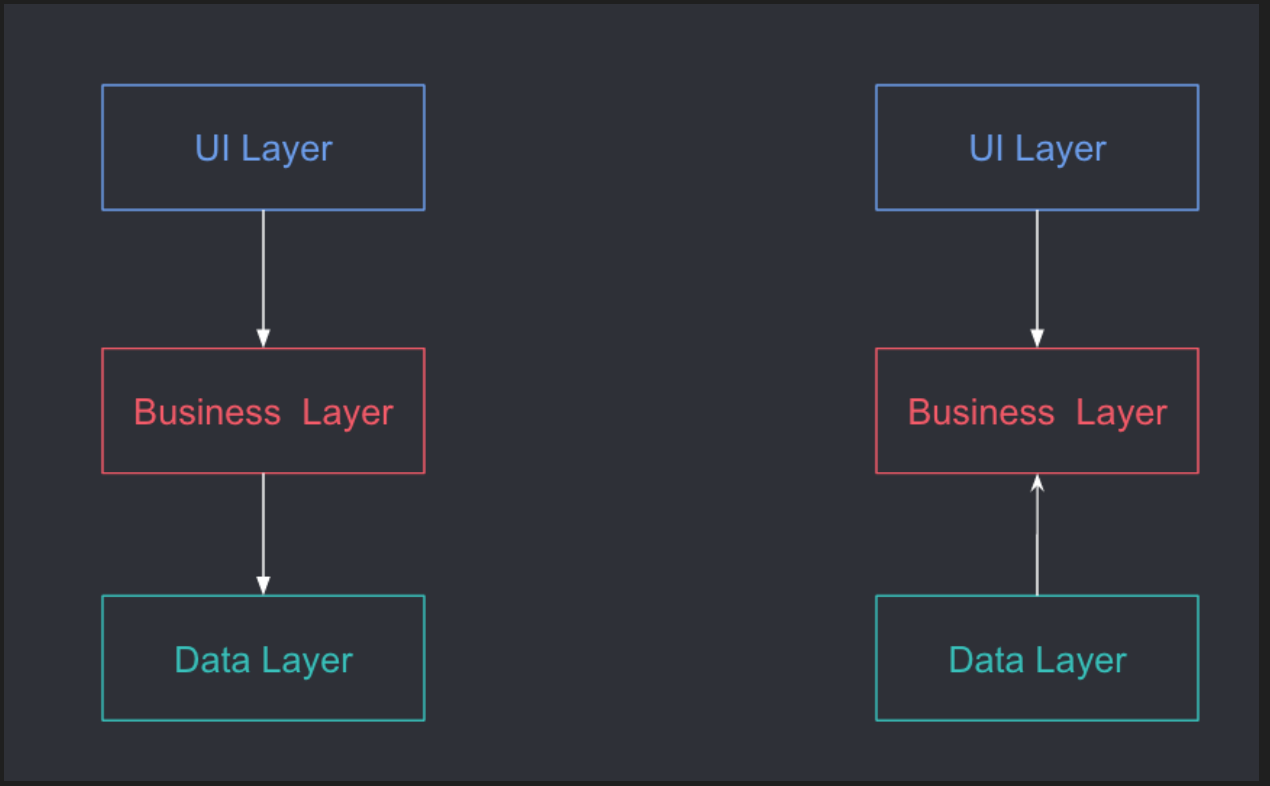
Dependency Injection & IoC

Inversion of Control - IoC

Es un patrón de diseño.

Es una forma diferente de manipular el control de los objetos. Por lo general, depende de la inyección de dependencia, porque la creación de instancias de un objeto se convierte en una responsabilidad de la arquitectura, no del desarrollador.

IoC asiste a uno de los principios SOLID, la Inversión de Dependencias.



AVANTICA
an encora company

CQRS & Mediator Patterns

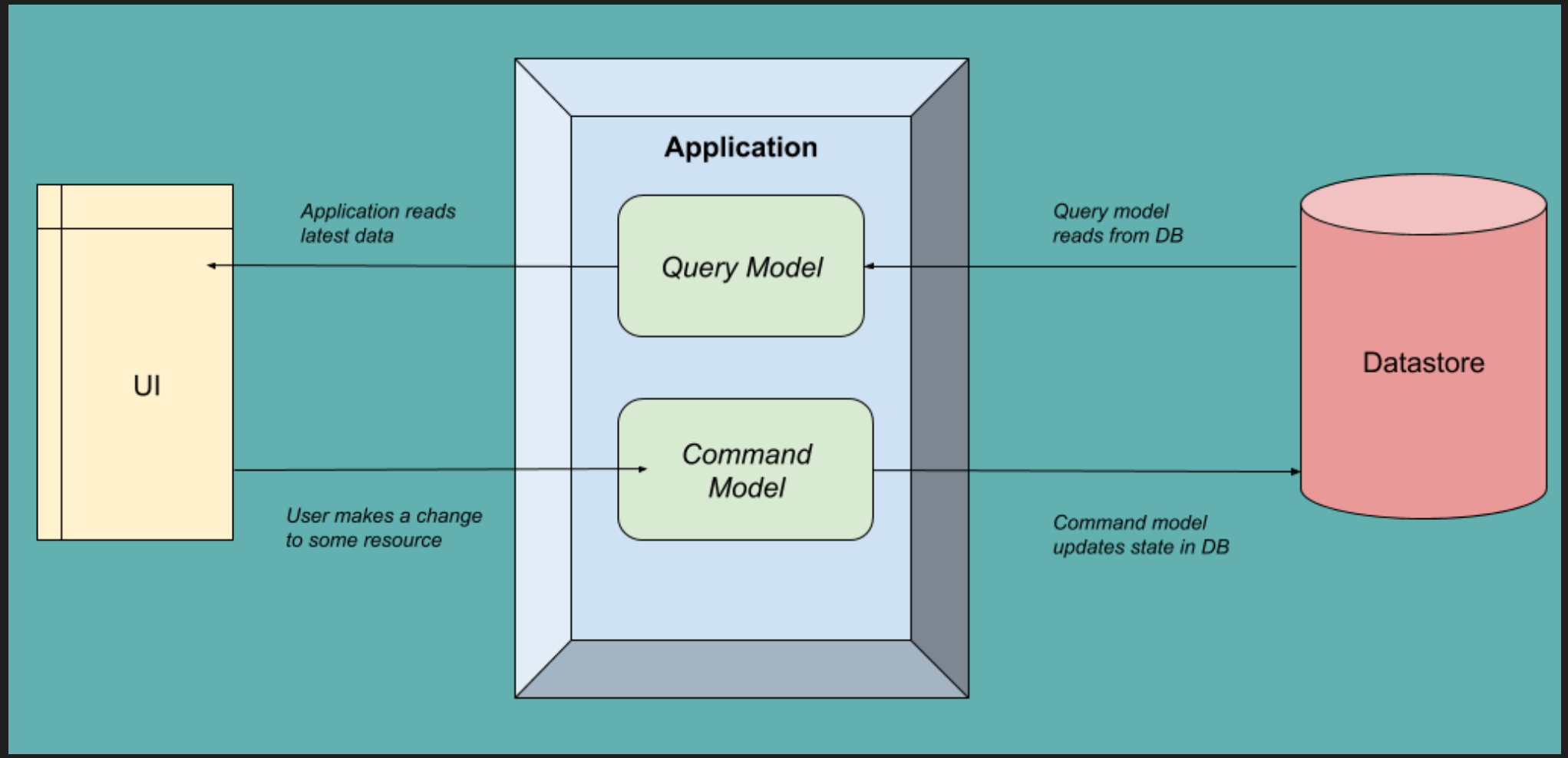
CQRS - Command Query Responsibility Segregation
(Segregación de responsabilidad de consultas de comando)

Divide la responsabilidad de los comandos (guardar) y las consultas (leer) en diferentes modelos.

Si pensamos en el patrón CRUD de uso común (Create-Read-Update-Delete), generalmente tenemos la interfaz de usuario interactuando con un almacén de datos responsable de las cuatro operaciones.

En cambio, CQRS nos haría dividir estas operaciones en dos modelos, uno para las consultas (también conocido como "R") y otro para los comandos (también conocido como "CUD").

CQRS & Mediator Patterns



CQRS & Mediator Patterns

Mediator (Mediador)

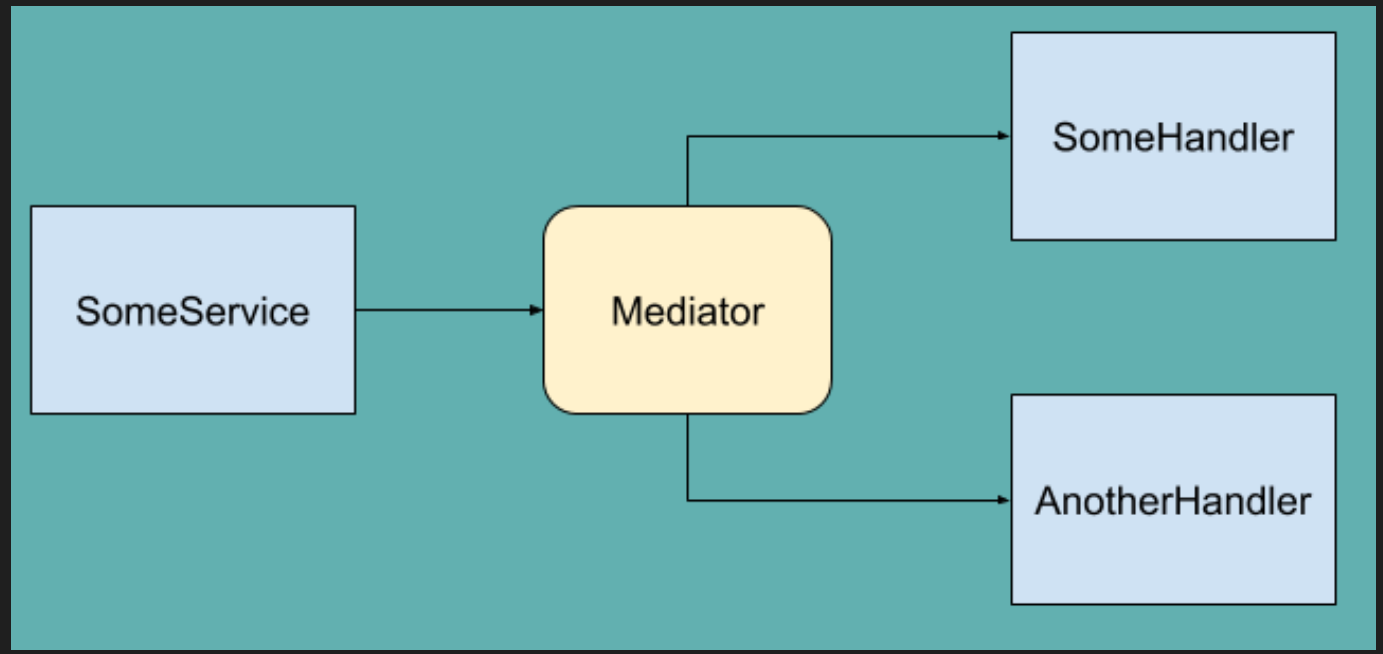
Define un objeto que encapsula cómo otros objetos interactúan entre sí.

En lugar de que dos o más objetos tengan una dependencia directa entre sí, interactúan con un "mediador", que está a cargo de enviar esas interacciones a la otra parte.

CQRS & Mediator Patterns

Mediator (Mediador)

Se envía un mensaje al Mediador, y el Mediador luego invoca múltiples servicios para manejar el mensaje. No hay dependencia directa entre ninguno de los componentes azules.



AVANTICA
an encora company

Demo

AVANTICA
an encora company

¡Iniciemos un proyecto juntos!

Contáctenos:

info@avantica.com

www.avantica.com

USA: *+1 (650) 641 3134*

Costa Rica: *+506 4040 0700*

Perú: *+501 616 7676*

Bolivia: *+591 4 4067250*

Colombia: *+57 (2) 321 7000*

AVANTICA
an encora company


AVANTICA
an encora company