DISEÑO DE APLICACIONES 2 - TECNOLOGÍA

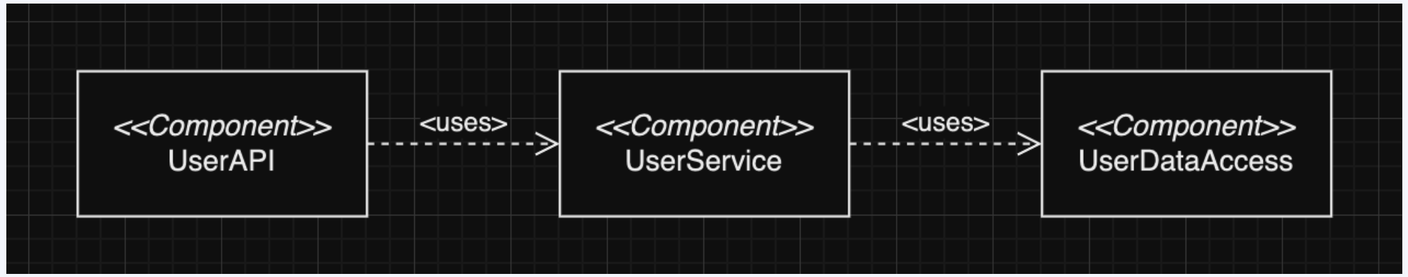
Inyección de Dependencias

Fundamentos

¿QUÉ ES UNA DEPENDENCIA?

- Relación entre dos componentes. Un componente “confía” en el trabajo del otro y lo utiliza con algún propósito.

- Si lo llevamos a una relación entre objetos, podemos decir que una clase depende de otra cuando:
  - Crea un objeto de la otra clase
  - Una función recibe como argumento una instancia de la otra clase
  - Retorna una instancia de la otra clase



buscamos otra cosa...

DIP

DEPENDENCY INVERSION PRINCIPLE (SOLID)

- **¿Cómo manejamos las dependencias?**

- Según *Uncle Bob…*
  - Los módulos de **alto nivel** no deben depender de módulos de **bajo nivel**: ambos deben depender de abstracciones.
  - Las abstracciones no deben depender de los detalles. Los detalles deben depender de abstracciones.

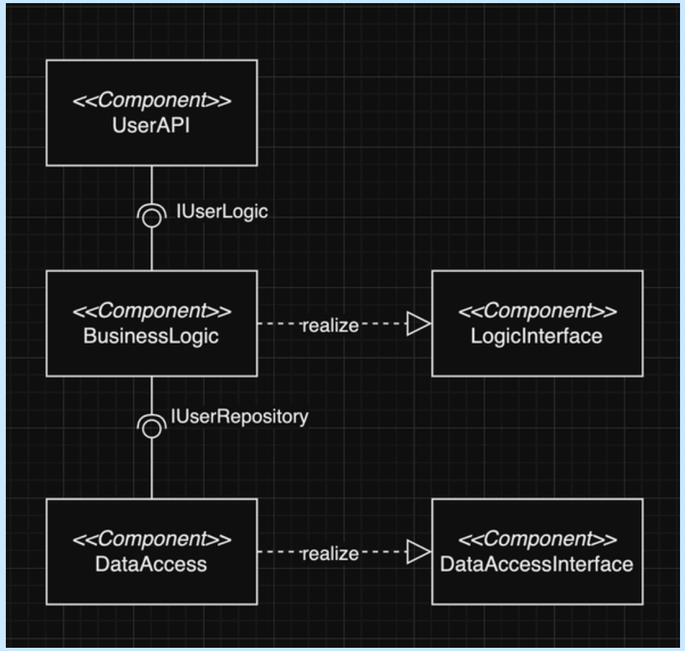
- ¿Alto nivel? *Core del negocio*
- ¿Bajo Nivel? *Propósitos más específicos, fuera del core*

- ¿Abstracciones? *Interfaces*

ABSTRACCIONES

Interfaces (o clases abstractas)

- Son parte necesaria de la solución. Nos van a permitir depender únicamente de contratos y no implementaciones específicas.
- Es decir…




```
// Ejemplo de CONTRATO
public interface IUserLogic
{
  void CreateUser(string name); // Declaro pero no implemento
  void GetUser(int id);
}

// Ejemplo de IMPLEMENTACIÓN (específica) de un contrato
public class UserLogic : IUserLogic
{
  public void CreateUser(string name) // Ahora sí implemento
  {
    // Creo un usuario
  }

  public void GetUser(int id)
  {
    // Obtengo un usuario según el ID
  }
}
```

INYECCIÓN DE DEPENDENCIAS

- Este patrón o técnica, separa la creación y el consumo de dependencias.
- Nos ayudará a cumplir con *DIP* y, por ende, con un correcto manejo de las dependencias en nuestros sistemas.
- Técnicamente, estamos implementando una inversión del control (IoC).

- Tipos:
  - ***Por constructor***
  - Por propiedad *(Setter)*
  - Por método

VENTAJAS Y DESVENTAJAS

- Mayor mantenibilidad
- Facilita las pruebas unitarias
- Mayor modularidad
- Ayuda a cumplir con DIP...

- Mayor complejidad
- Acoplamiento con el framework
- Posible afectación en el rendimiento