Inyección de dependencias

Un **patrón de diseño** orientado a objetos en el que **se suministran objetos a una clase** en lugar de ser la propia clase la que cree dichos objetos.

Formas :

- Constructor
- Método Setter
- Variable de instancia / Propiedad

Solid

Single-responsiblity	La clase, componente o microservicios deben hacer **una cosa** para disminuir futuras modificaciones y/o cambios. **Principio de responsabilidad única**
Open-closed	Entidades deben de estar **ABIERTAS** para su extensión pero **CERRADAS** para su modificación. **Principio de abierto/cerrado**
Liskov Substitution	Una subclase debe poder ser sustituible por su superclase. **Principio de sustitución de Liskov**
Interface Segregation	Segregar las clases tanto como sea posible. **Principio de segregación de la interfaz**
Dependency Inversion	Permite inyectar otras clases y añadir funcionalidad transversal

Arquitectura - Capas de Aplicación

Presentation (UI)	Handles HTTP requests and responds with HTML OR JSON/XML
Service	Business logic
Data	Database access

Despite having a logically modular architecture, the application is packaged and deployed as a **monolith**

Monolithic Architecture

Benefits	- Simple to develop - Simple to test (launch the application) - Simple to deploy (Copy packaged app) - Simple to scale horizontally (Copy apps)
Drawbacks	- Limitations in size and complexity - Application becomes to large and complex to fully understand and make changes fast. - Size of application can slow down start-up time. - Each update requieres entire application to be redeployed. - Changes require extensive manual testing. - Continuous deployment becomes difficult. - Adopting new technologies must be applied to entire application thus becomes expensive in both time and cost

In the early stages of the project it works well and basically most of the big and successful applications which exist today were started as a monolith.

Microservices Architecture

Benefits	Manageable services (Faster to develop) - Can be managed independently - New technologies are easier to apply - Deployed Independently - Scaled Independently
Drawbacks	Since it uses partitioned database architecture each database must be handled independently - Testing is more complex (Must deploy the service and all of its dependencies) - Deploying becomes more complex (Requires planning and coordination) - Every service must be configured, deployed, scaled and monitored independently

The ides is to split the application into a set of smaller interconnected services(**Microservice**) instead of building a single monolithic application.
Each microservice :
- Has its own hexagonal architecture (UI, BL,DA)
- Has its own database schema thus **ensures loose coupling**
- Can use the type of database that best suits its needs (Poliglot persistence architecture)

Que es OOP

Un modelo que organiza el diseño de software en torno a datos u objetos.
Beneficios :
- Reutilización
- Escalabilidad
- Eficiencia

Principios de OOP

Encapsulacion	Toda la información importante está contenida dentro de un objeto y solo se expone la información seleccionada La implementación y el estado de cada objeto se mantienen de forma privada dentro de una clase definida El objeto no tiene acceso a esta clase ni a la autorización para realizar cambios Llamar solo a una lista de métodos o funciones públicos
Abstraccion	Los objetos solo revelan mecanismos internos que son relevantes para el uso de otros objetos, ocultando cualquier código de implementación innecesario. La clase derivada puede tener su funcionalidad extendida
Herencia / Inheritance	Las clases pueden reutilizar código de otras clases Se pueden asignar relaciones y subclases entre objetos, lo que permite la reutilización de la lógica común mientras se mantiene una jerarquía única.
Polimorpphism	Los objetos están diseñados para compartir comportamientos y tomar más de una forma. Permite el paso de diferentes tipos de objetos a través de la misma interfaz.

Access Modifiers	
public	acceso no está restringido
protected	acceso está limitado a la clase contenedora
internal	acceso está limitado al ensamblado actual
protected internal	El acceso está limitado al ensamblado actual o a los tipos derivados de la clase contenedora.
private	El acceso está limitado al **tipo contenedor**

By **FreddyAlmeida**

cheatography.com/freddyalmeida/

Not published yet.
Last updated 8th June, 2022.