Principio SOLID - Open Closed

Introducción

El Principio Abierto–Cerrado (OCP) es uno de los cinco principios de diseño conocidos como SOLID. Fue propuesto por Bertrand Meyer en 1988 en su libro Object-Oriented Software Construction y más tarde fue popularizado dentro de SOLID por Robert C. Martin (Uncle Bob) a principios de los años 2000. Es considerado uno de los principios más importantes dentro de la arquitectura de software porque ayuda a que los sistemas sean más fáciles de mantener y de hacer crecer con el tiempo.

Definición del Principio

El principio dice:

> "Las entidades de software (clases, módulos, funciones, etc.) deben estar abiertas a la extensión, pero cerradas a la modificación."

Esto puede sonar contradictorio al principio… ¿Cómo algo puede estar abierto y cerrado al mismo tiempo? Vamos a explicarlo de forma sencilla.

¿Qué significa realmente?

Imagina que tienes una casa.

- **Cerrada a modificación** significa que no tienes que romper paredes cada vez que quieres hacer algo nuevo.
- **Abierta a extensión** significa que puedes agregar una nueva habitación sin destruir lo que ya está construido.

En software es igual:

- *Cerrado a modificación* → No deberías cambiar el código que ya funciona.
- *Abierto a extensión* → Sí deberías poder agregar nuevas funcionalidades.

En otras palabras:

El sistema debe poder crecer sin tener que editar constantemente lo que ya estaba hecho y funcionando.

¿Por qué es importante?

Cuando un sistema crece, empiezan a aparecer nuevos requisitos:

- Nuevo tipo de usuario
- Nuevo método de pago
- Nuevo idioma
- Nuevo tipo de producto

Si cada vez que aparece algo nuevo tenemos que modificar las mismas clases antiguas, pasan cosas peligrosas:

- Se pueden dañar funcionalidades que ya funcionaban.
- Aumentan los errores (bugs).
- El sistema se vuelve difícil de mantener.
- Los cambios se propagan en cascada a otras partes del código.

En cambio, si aplicamos el OCP:

- El código viejo se mantiene estable.
- Las nuevas funcionalidades se agregan sin romper lo anterior.
- El sistema es más flexible.
- Es más fácil probar y mantener el software.

¿Cómo se logra el OCP?

El principio no se cumple “por arte de magia”. Se apoya en conceptos de la Programación Orientada a Objetos.

Principalmente se usan:

Herencia

Permite crear nuevas clases que agregan comportamiento sin modificar la clase original.

Polimorfismo

Permite tratar diferentes objetos como si fueran del mismo tipo, aunque se comporten distinto internamente.

Interfaces o clases abstractas

Definen una estructura base común. Las clases nuevas solo deben seguir esa estructura, sin alterar el sistema principal.

La idea central es esta:

En lugar de preguntar constantemente “¿qué tipo eres?” y modificar el código cada vez que aparece un nuevo tipo, el sistema debe estar preparado para que simplemente se agreguen nuevas implementaciones.

Ventajas del Principio Abierto–Cerrado

Aplicar correctamente el OCP hace que:

- El sistema sea más mantenible.
- Se reduzcan errores al agregar nuevas funciones.
- El software sea más flexible.
- El código sea más fácil de probar.
- El crecimiento del sistema sea más ordenado.