Desarrollo Guía 5

Jullians Mauricio Amado Gutierrez

Tomas Alejandro Santiago Reyes

Juan Esteban Cabal Bautista

Universidad Manuela Beltrán

Taller de programación

Olga Lucia Roa Bohórquez

31 de marzo de 2025

**Características de un buen componente reusable:** Un buen componente reusable debe cumplir con las siguientes características:

* **Modularidad:** Debe ser independiente y fácilmente integrable en diferentes sistemas.
* **Bajo acoplamiento:** Cada módulo debe ser lo más independiente posible, para facilitar la modificación o sustitución de uno sin que afecte al resto del sistema.
* **Alta cohesión:** cada módulo cumple una función específica y bien definida.
* **Cumplimiento de los principios SOLID:** Debe seguir buenas prácticas de diseño para garantizar su calidad y reutilización.

**Definición de patrón de software:**Se refiere a una solución estandarizada que permite resolver problemas recurrentes en el diseño y desarrollo de software.Estos patrones mejoran la calidad, mantenibilidad y escalabilidad de los sistemas.

**Principio de alta cohesión y bajo acoplamiento:**

* **Alta cohesión:**Los elementos dentro de un módulo están fuertemente relacionados y trabajan juntos.
* **Bajo acoplamiento:**Se busca que los diferentes módulos sean lo mas independientes entre sí que sea posible, lo que facilita la modificación, actualización o eliminación de funciones.
* Este principio ayuda a mejorar la reutilización del código, facilita el mantenimiento y reduce la complejidad del sistema.

**Cinco patrones de concurrencia:**

* **Thread Pool:** grupo de hilos que ejecutan tareas sin crear nuevos.
* **Future:** obtiene resultados de tareas asincrónicas.
* **Producer-Consumer:** productores generan datos y consumidores los usan.
* **Read-Write Lock:** varios leen o uno escribe, nunca ambos.
* **Monitor Object:** encapsula la sincronización en un objeto.

**¿Por qué Thread Pool es un patrón de concurrencia?**

Porque maneja múltiples tareas al mismo tiempo usando hilos reutilizables, lo que mejora el rendimiento y evita sobrecarga.

| Patrón de Software | Breve descripción | Ejemplo donde se aplica |
| --- | --- | --- |
| Singleton | Una única instancia de una clase en todo el programa | Gestión de configuraciones globales |
| Factory Method | Genera una interfaz para crear objetos sin especificar su clase específicamente. | Creación de objetos en frameworks GUI. |
| Observer | Permite que un objeto notifique cambios a múltiples suscriptores. | Sistemas de eventos en interfaces gráficas. |
| Strategy | Permite cambiar el comportamiento de un objeto cuando se ejecuta. | Algoritmos de encriptación intercambiables. |
| Decorator | Añade funcionalidad a objetos de forma dinámica sin modificar su estructura. | Extensión de funciones en editores de texto. |
| Adapter | Permite que clases con interfaces en principio incompatibles trabajen juntas. | Integración de APIs de terceros |
| MVC (Modelo Vista Controlador) | Separa la lógica de negocio, la UI y el control de flujo. | Desarrollo de aplicaciones web. |
| Prototype | Crea nuevos objetos clonando una instancia existente. | Creación de documentos en software de diseño. |
| Command | Encapsula una acción en un objeto específico, lo que permite su ejecución de forma diferida. | Funciones de undo/redo en editores. |
| Thread pool | Gestiona hilos reutilizables para mejorar el rendimiento | Servidores web que manejen múltiples solicitudes. |