PRESENTADO POR

Gael Breton Rendon

GRUPO

10° B

MATERIA

Desarrollo móvil integral

PROFESOR

Ray Parra

Tijuana, Baja California, 07 de enero del 2025

Los **patrones de diseño** son soluciones reutilizables a problemas comunes que surgen durante el desarrollo de software. Estos patrones se basan en las mejores prácticas de diseño y ayudan a los desarrolladores a abordar problemas arquitectónicos y de diseño de manera más eficiente y estructurada. La **selección de patrones de diseño** es un paso crucial en la ingeniería de software, ya que una correcta elección puede mejorar la calidad, escalabilidad y mantenibilidad del sistema, mientras que una mala elección puede hacer que el software sea difícil de modificar, escalar o mantener.

Conceptos Clave de los Patrones de Diseño

1. **Definición de Patrones de Diseño:** Un patrón de diseño es una solución general y reutilizable a un problema común que se presenta en un contexto específico dentro de un sistema de software. Los patrones de diseño no son fragmentos de código concretos, sino más bien descripciones abstractas de cómo se pueden organizar las clases y objetos para resolver problemas comunes de diseño de manera eficiente.
2. **Categorías de Patrones de Diseño:** Los patrones de diseño se agrupan generalmente en tres categorías principales:
    - **Patrones Creacionales:** Se encargan de la creación de objetos de una manera adecuada y flexible. Ejemplos incluyen el **Patrón Singleton**, **Factory Method**, **Abstract Factory**, **Builder**, y **Prototype**.
    - **Patrones Estructurales:** Se centran en la organización y composición de clases y objetos, permitiendo la creación de estructuras complejas sin modificar el código existente. Ejemplos incluyen el **Patrón Adapter**, **Composite**, **Decorator**, **Facade**, y **Proxy**.
    - **Patrones de Comportamiento:** Tratan sobre la comunicación entre objetos, cómo interactúan y se coordinan entre sí. Ejemplos incluyen el **Patrón Observer**, **Strategy**, **Command**, **Iterator**, y **State**.
3. **Propósito de los Patrones de Diseño:**
    - **Reutilización de Soluciones Probadas:** Los patrones de diseño permiten aplicar soluciones que han sido probadas y perfeccionadas a lo largo del tiempo, evitando reinventar la rueda.
    - **Facilitar la Comprensión del Diseño:** Proporcionan un vocabulario común entre los desarrolladores para describir y discutir soluciones de diseño.
    - **Mejora de la Flexibilidad y Escalabilidad:** Ayudan a crear sistemas más flexibles y fáciles de escalar, adaptándose a cambios sin modificar en exceso la estructura existente del software.

- **Reducción de la Complejidad:** Los patrones estructuran el diseño de manera que se reduce la complejidad al abstraer detalles específicos de implementación.

Criterios para la Selección de Patrones de Diseño

La elección de un patrón de diseño debe basarse en una comprensión profunda del contexto y las necesidades del proyecto. No existe un "patrón único" que sirva para todos los casos, por lo que se debe evaluar cuidadosamente qué patrón se adapta mejor a la situación. Algunos de los factores a considerar incluyen:

1. **El Tipo de Problema que se Está Abordando:**
   - **Creación de Objetos:** Si el problema está relacionado con la creación de instancias de objetos, se podría usar un patrón creacional como **Factory Method** o **Abstract Factory**.
   - **Composición de Clases y Objetos:** Si se requiere estructurar o organizar objetos de manera eficiente, patrones estructurales como **Composite** o **Decorator** son útiles.
   - **Comunicación y Coordinación:** Si el desafío está relacionado con cómo los objetos se comunican entre sí, los patrones de comportamiento como **Observer** o **Strategy** serían apropiados.
2. **Flexibilidad y Escalabilidad:** Algunos patrones, como el **Strategy** o **Command**, permiten cambiar el comportamiento de los objetos en tiempo de ejecución, lo cual es útil para sistemas que deben adaptarse a cambios dinámicos.
3. **Facilidad de Mantenimiento:** Algunos patrones están diseñados para hacer que el código sea más fácil de mantener y extender. Patrones como **Facade** y **Adapter** ayudan a reducir el acoplamiento entre clases, facilitando la integración con otras partes del sistema.
4. **Consistencia del Diseño:** La elección de patrones debe mantener la coherencia en el diseño del sistema. Es importante no usar patrones innecesarios que compliquen el diseño sin aportar beneficios claros.
5. **Requisitos del Sistema:** Los patrones de diseño deben seleccionarse también en función de los requisitos específicos del sistema, como el rendimiento, la escalabilidad, la seguridad, y la facilidad de uso.