PATRONES DE DISEÑO DE SOFTWARE

Autor:
Johan Calderón Perdomo
Colaboradores:
Jesús Ariel González Bonilla
Jhon William Corredor Araujo

Servicio Nacional de Aprendizaje SENA Sede Industria
Neiva – Huila
11/12/2024

Resumen

Este artículo introduce los patrones de diseño de software, explicando qué son y por qué son importantes en el desarrollo de aplicaciones. Los patrones de diseño permiten a los desarrolladores utilizar soluciones probadas y reutilizables para problemas comunes en el desarrollo de software, facilitando la organización del código, la colaboración en equipo y la escalabilidad de los proyectos. A través de ejemplos sencillos, se demostrará cómo estos patrones mejoran la calidad del software y su mantenimiento.

Palabras clave: Patrones de diseño, Desarrollo de software, Soluciones reutilizables, Escalabilidad, Mantenimiento de software.

Introducción

En el mundo del desarrollo de software, enfrentarse a problemas recurrentes es parte del día a día. Sin embargo, reinventar la rueda cada vez que surge un problema es ineficiente y poco práctico. Es aquí donde los patrones de diseño de software juegan un papel fundamental. Estos patrones proporcionan soluciones generales y reutilizables para resolver problemas que los desarrolladores encuentran frecuentemente durante el proceso de diseño de software.

Este artículo explica qué son los patrones de diseño, sus principales categorías, y su relevancia en proyectos modernos, desde el mantenimiento hasta la escalabilidad y la colaboración en equipo.

¿Qué son los Patrones de Diseño de Software?

Los patrones de diseño de software son enfoques probados y documentados para resolver problemas comunes de diseño en el desarrollo de aplicaciones. Son como recetas"que

los desarrolladores pueden seguir para evitar errores conocidos y crear software más eficiente. Cada patrón está diseñado para abordar un problema específico que aparece repetidamente en el desarrollo, proporcionando una solución estándar que ha sido utilizada con éxito en múltiples proyectos.

Los patrones de diseño no son soluciones completas por sí mismas, sino que actúan como plantillas que se pueden aplicar en diferentes contextos para lograr una mejor organización y funcionamiento del código.

Tipos de Patrones de Diseño

Existen tres categorías principales de patrones de diseño:

1. Patrones Creacionales

Estos patrones se centran en la forma en que se crean los objetos. Ayudan a que el proceso de creación de objetos sea más flexible y eficiente.

Ejemplos:

- **Singleton:** Garantiza que una clase tenga solo una instancia en todo el programa.
- **Factory Method:** Proporciona una interfaz para crear objetos en una clase base, pero permite a las subclases modificar el tipo de objetos que se crean.

2. Patrones Estructurales

Estos patrones se enfocan en la composición de clases y objetos para formar estructuras más grandes.

Ejemplos:

- **Adapter:** Permite que dos clases incompatibles trabajen juntas al crear una interfaz común.
- **Decorator:** Añade funcionalidad a un objeto de manera dinámica sin alterar su estructura.

3. Patrones Comportamentales

Estos patrones se ocupan de la comunicación y la interacción entre objetos.

Ejemplos:

- **Observer:** Define una relación de uno a muchos entre objetos para que, cuando un objeto cambie su estado, los dependientes sean notificados automáticamente.
- **Strategy:** Permite a un objeto seleccionar un algoritmo de entre un conjunto de algoritmos, y cambiarlo en tiempo de ejecución.

Grafica entre patrones de diseño:

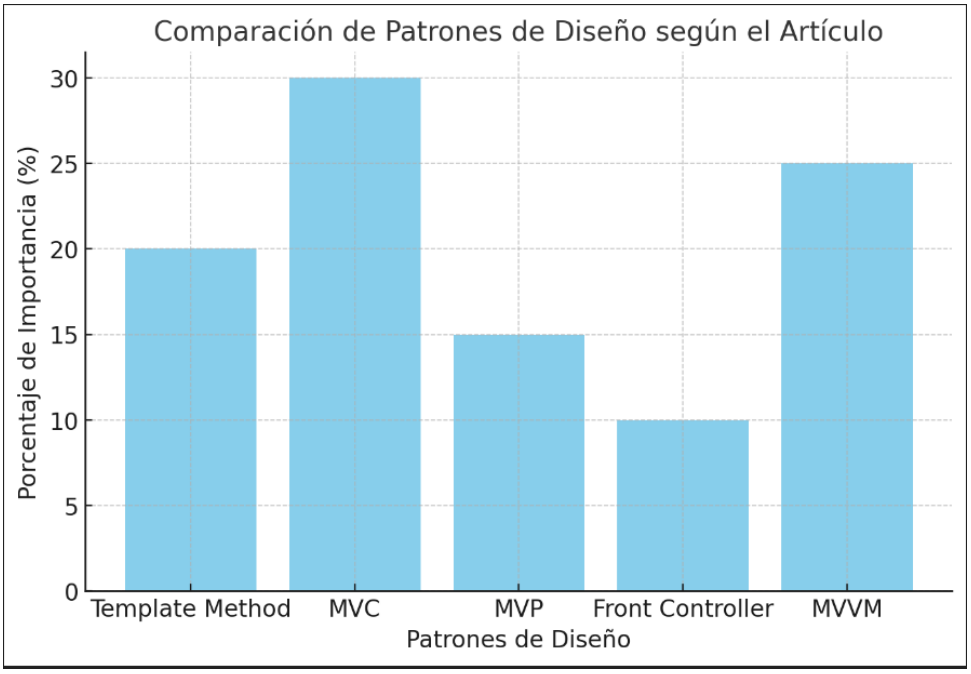

Figura 1: Gráfica que representa una comparación entre patrones de diseño.

Parámetros	**Template Method**	**Model View-Presenter (MVP)**	**Model Front Controller**	**Model View-ViewModel (MVVM)**
Lenguajes de Programación	PHP, C++, Java	.Net, Java, PHP	Java, PHP, Python, Ruby, Raku	.Net, JavaScript, C++
Complejidad	Baja	Baja	Alta	Alta
Seguridad	-	Baja	Alta	Baja
Uso	Frameworks	Aplicaciones Android	Frameworks Web	Windows y Gráficos Web

¿Por qué son importantes los Patrones de Diseño de Software?

1. **Organización del Código:** Los patrones de diseño proporcionan una estructura clara al código, lo que hace que sea más legible y fácil de entender. Cuando varios desarrolladores trabajan en un proyecto, seguir patrones comunes asegura que todos entiendan cómo está organizado el código.

2. **Facilitan la Reutilización:** Los patrones permiten la reutilización de soluciones probadas. En lugar de escribir nuevo código para cada problema, los desarrolladores pueden aplicar estos patrones y confiar en que ya han sido probados y optimizados.

3. **Mejora de la Escalabilidad:** A medida que una aplicación crece, es crucial que el diseño del software sea lo suficientemente flexible para adaptarse a nuevas fun-

cionalidades sin necesidad de reescribir grandes secciones del código. Los patrones de diseño ayudan a mantener esta flexibilidad y escalabilidad.

4. **Facilitan el Mantenimiento:** Con el uso de patrones, el código es más fácil de modificar y mantener. Como se siguen estructuras estándar, los errores son más fáciles de localizar y corregir.

5. **Colaboración Eficiente:** Cuando un equipo de desarrolladores utiliza patrones de diseño, todos hablan el mismo ïdioma". Esto facilita la comunicación y la colaboración en proyectos grandes y complejos.

¿Cómo funcionan?

Los patrones de diseño no son soluciones exactas para cada problema, sino más bien plantillas flexibles que se adaptan a diferentes contextos. Proporcionan una estructura y una terminología común para abordar problemas recurrentes, como:

- ¿Cómo crear objetos de manera flexible? (**Patrones creacionales:** Factory Method, Singleton, etc.)
- ¿Cómo componer objetos para formar estructuras más grandes? (**Patrones estructurales:** Adapter, Composite, etc.)
- ¿Cómo hacer que los objetos se comuniquen entre sí de manera efectiva? (**Patrones comportamentales:** Observer, Strategy, etc.)

Ejemplos Prácticos de Patrones de Diseño

- **Singleton:** Imagina una aplicación de banca en línea. Necesitamos un único objeto que maneje la conexión con la base de datos para evitar conflictos. Aquí, el patrón Singleton garantiza que solo haya una instancia de la conexión a la base de datos.
- **Observer:** Un ejemplo común es el sistema de notificaciones en redes sociales. Cuando un usuario publica una actualización, el patrón Observer permite que todos los seguidores reciban una notificación de forma automática.

Conclusión

Los patrones de diseño de software son herramientas esenciales que permiten a los desarrolladores abordar problemas comunes de manera eficiente y probada. Al usar estos patrones, el código se vuelve más organizado, escalable y fácil de mantener, lo que reduce costos y tiempo de desarrollo a largo plazo. Además, promueven la colaboración y la reutilización de soluciones, lo que mejora significativamente la calidad del software y su capacidad para crecer con el tiempo.

Dominar los patrones de diseño es una habilidad clave para cualquier desarrollador que busque crear software de alta calidad y preparado para el futuro.

Referencias

[1] Palmero, M. A. S., Martínez, N. S., & Grass, O. Y. R. (2019). Revisión de elementos conceptuales para la representación de las arquitecturas de referencias de software. *Revista cubana de ciencias informáticas*, 13(1), 143–157. http://scielo.sld.cu/scielo.php?pid=S2227-18992019000100143script=sci$_a rttext$

[2] (S/f). Unirioja.es. Recuperado el 12 de diciembre de 2024, de https://dialnet.unirioja.es/servlet/articulo?codigo=9042927

[3] Logroño, J. F. S., Berdún, L., Armentano, M., & Amandi, A. (s/f). Análisis de secuencias discretas para la detección de Patrones de Diseño de Software. Edu.ar. Recuperado el 12 de diciembre de 2024, de https://sedici.unlp.edu.ar/bitstream/handle/10915/152663/Documento$_c ompleto.pdf - PDFA.pdf?sequence = 1isAllowed = y$

[4] Español, M. (2022, febrero 15). Patrones de Diseño: Mejores experiencias y productos fácilmente escalables. *Modyo*. https://es.modyo.com/blog/patrones-de-diseno-mejores-experiencias-y-productos-facilmente-escalables

[5] (S/f-b). https://www.proquest.com/docview/2343015882/90C25C73A9E343F0PQ/6?accountid=31491sourcetype=Scholarly

[6] Mesías-Valencia, J. J. (2024). Incidencia de los patrones de diseño de software en la seguridad de aplicaciones web. *MQRInvestigar*, 8(1), 236–259. https://doi.org/10.56048/mqr20225.8.1.2024.236-259