**Patrones de diseño en aplicaciones web: Una revisión integral**

Daniel Caicedo Trujillo

Universidad Cooperativa de Colombia

Ingeniería de Sistemas

Docente: Jesus ariel Gonzales Bonilla

2025

**Resumen**

Este documento presenta una revisión integral sobre patrones de diseño en aplicaciones web, compilando información de múltiples fuentes académicas y casos de estudio industriales. Se analizan los patrones más utilizados en la industria, incluyendo MVC, Singleton y Factory Method, así como su impacto en el rendimiento, flexibilidad y seguridad de las aplicaciones. Los resultados demuestran que las aplicaciones que implementan patrones de diseño superan significativamente a aquellas que no los utilizan. También se examinan los antipatrones como errores comunes en el desarrollo de software y métodos de evaluación arquitectónica.

Palabras clave: *patrones de diseño, aplicaciones web, arquitectura de software, antipatrones, seguridad informática*

1. Patrones de Diseño en Aplicaciones Web

¿Qué son los Patrones de Diseño?

Los patrones de diseño son soluciones probadas y reutilizables para problemas comunes en el desarrollo de software.

Patrones Más Utilizados

Según la investigación de Gonzales (2020), los patrones más populares son:

MVC (Modelo-Vista-Controlador)

MVP (Modelo-Vista-Presentador)

MVVM (Modelo-Vista-VistaModelo)

Beneficios Comprobados

Las pruebas demuestran que aplicaciones con patrones superan a las que no los usan:

Mejor rendimiento (17ms vs 581ms promedio)

Mayor facilidad de mantenimiento

Reducción de errores

Desarrollo más eficiente

2. Aplicación de Patrones de Diseño para Alta Flexibilidad en el Software

Problema Principal

Las aplicaciones empresariales enfrentan un desafío constante: las reglas de negocio cambian frecuentemente. Las empresas necesitan adaptar sus sistemas para mantenerse competitivas, pero estos cambios no deberían ser traumáticos, costosos o consumir mucho tiempo.

Objetivo del Estudio

Demostrar cómo el uso de patrones de diseño específicos permite desarrollar aplicaciones empresariales flexibles a los cambios en sus reglas de negocio, usando como ejemplo un sistema de pedidos de restaurante.

Patrones de Diseño Aplicados

Patrón Estrategia

Patrón Compuesto

Patrón Fábrica

3. Guía Práctica para el Uso de Patrones de Diseño

¿Qué son los Patrones de Diseño?

Los patrones de diseño son soluciones reutilizables a problemas comunes en el desarrollo de software, descritos originalmente por el "Gang of Four" (GoF) en 1994. Funcionan como plantillas probadas que capturan las mejores prácticas de programación orientada a objetos.

Clasificación de Patrones

Patrones de Creación (5 patrones)

Abstract Factory, Builder, Factory Method, Prototype, Singleton

Controlan la instanciación de objetos

Patrones Estructurales (7 patrones)

Adapter, Bridge, Composite, Decorator, Facade, Flyweight, Proxy

Manejan la composición de clases y objetos

4. Incidencia de los Patrones de Diseño en la Seguridad de Aplicaciones Web

Problema de Investigación

Las aplicaciones web modernas enfrentan múltiples vulnerabilidades de seguridad que pueden comprometer información sensible. Los desarrolladores a menudo priorizan nuevas tecnologías sobre aspectos críticos como la estructura del código y la seguridad, resultando en aplicaciones vulnerables a ataques.

Objetivo del Estudio

Determinar cómo los patrones de diseño de software contribuyen a mitigar vulnerabilidades en aplicaciones web, específicamente las identificadas por OWASP como más críticas.

Patrones de Diseño Aplicados

Patrones Estructurales y Comportamentales

Chain of Responsibility: Cadena de validación para control de acceso

State Pattern: Gestión de estados para rate limiting

Proxy Pattern: Intermediario para validación de datos de entrada

5. Patrones de Diseño - Universidad Cooperativa de Colombia

Este documento de la Universidad Cooperativa de Colombia expone una clase sobre los patrones de diseño enfocada a los estudiantes de sexto semestre de la carrera de Ingeniería de Sistemas. La meta principal es que los estudiantes sean capaces de identificar, comprender y manipular los patrones de diseño para aplicarlos a situaciones de la vida real. Los patrones de diseño son soluciones a problemas que se pueden reutilizar y que son habituales en el desarrollo de software.

La actividad implica la autoevaluación a través de preguntas reflexivas y se califica a través de la evidencia recopilada del ejercicio propuesto (30%) y a través del uso de las herramientas sugeridas (20%), la participación en el foro (20%), la participación individual (20%) y la entrega oportuna (10%).

6. Patrones de Diseño y Refactorización

En el análisis de este documento académico, se identifican tres temas clave para programar mejor. Los patrones de diseño son como recetas probadas que funcionan cuando se presentan problemas típicos al crear software - en vez de inventar la rueda otra vez, se utilizan soluciones que ya demostraron ser efectivas. Es interesante cómo se dividen en categorías: los que ayudan a crear objetos, los que organizan la estructura y los que manejan el comportamiento.

Los antipatrones llamaron la atención porque básicamente son errores comunes con nombre propio, como "Spaghetti Code" cuando el código queda todo enredado. La refactorización es algo que se hace seguido - limpiar código sin romper lo que funciona.

7. Patrones de Diseño y Refactorización (Continuación)

En el análisis de este documento académico, se encuentran tres temas clave para programar mejor. Los patrones de diseño son como recetas probadas que funcionan cuando se tienen problemas típicos al crear software - en vez de inventar la rueda otra vez, se usan soluciones que ya demostraron ser efectivas. Es interesante cómo los dividen en categorías: los que ayudan a crear objetos, los que organizan la estructura y los que manejan el comportamiento.

Los antipatrones llamaron la atención porque básicamente son errores comunes con nombre propio, como "Spaghetti Code" cuando el código queda todo enredado. La refactorización es algo que se hace seguido - limpiar código sin romper lo que funciona.

8. Patrones GRASP y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación

Este artículo resulta muy interesante porque explica cómo enseñar patrones GRASP y anti-patrones desde los primeros cursos de programación, no esperando hasta materias avanzadas. Los autores desarrollaron un método llamado MIPSOO que tiene cuatro pasos: identificar qué necesita el usuario, diseñar el diagrama de clases, definir responsabilidades y escribir pseudocódigo.

Lo que más llama la atención son los patrones GRASP - que son más bien "buenas prácticas" que patrones complicados. El patrón Experto dice que la responsabilidad debe ir a la clase que tiene la información necesaria. El Creador define quién debe crear objetos. Bajo acoplamiento significa que las clases no dependan mucho unas de otras, y Alta cohesión que cada clase tenga responsabilidades bien definidas.

9. Evaluando la Arquitectura de Software - Parte 2. Métodos de Evaluación

Este artículo resulta muy útil porque explica cuatro métodos concretos para evaluar arquitecturas de software, cada uno enfocado en un aspecto diferente. Es como tener una caja de herramientas donde cada una sirve para algo específico.

ALMA se enfoca en qué tan fácil será modificar el sistema después. Es práctico que use escenarios de cambio para predecir costos de mantenimiento - es muy práctico pensar "¿qué pasaría si necesitamos cambiar esto?".

PASA analiza el rendimiento del sistema usando escenarios para ver qué tan rápido responde. Lo interesante es que identifica anti-patrones de desempeño y sugiere soluciones.

SALUTA evalúa qué tan fácil será para los usuarios usar el sistema, considerando aspectos como facilidad de aprendizaje y satisfacción. Es el único que se preocupa por la experiencia del usuario desde la arquitectura.

SNA verifica si el sistema puede sobrevivir ataques o fallas.

10. Patrones GRASP y Anti-Patrones: un Enfoque Orientado a Objetos desde Lógica de Programación (Continuación)

Este artículo resulta muy interesante porque explica cómo enseñar patrones GRASP y anti-patrones desde los primeros cursos de programación, no esperando hasta materias avanzadas. Los autores desarrollaron un método llamado MIPSOO que tiene cuatro pasos: identificar qué necesita el usuario, diseñar el diagrama de clases, definir responsabilidades y escribir pseudocódigo.

Lo que más llama la atención son los patrones GRASP - que son más bien "buenas prácticas" que patrones complicados. El patrón Experto dice que la responsabilidad debe ir a la clase que tiene la información necesaria.

11. Uso de Patrones de Diseño en Costa Rica

Este estudio analizó el uso de patrones de diseño en más de 20 empresas de software costarricenses. Los resultados principales son:

80% de las empresas usan patrones de diseño. Los más populares son:

MVC (23%)

Singleton (14%)

Factory Method (11%)

Beneficios identificados: código más comprensible, estandarización, mejor comunicación entre equipos.

12. Análisis Comparativo de Patrones de Diseño de Software

Este estudio compara cinco patrones de diseño importantes, analizando sus características, ventajas y desventajas para ayudar a los desarrolladores a elegir el más adecuado según sus necesidades.

Ejemplo:

Template Method es ideal para evitar código duplicado cuando se tienen algoritmos similares con pequeñas variaciones. Es fácil de implementar y muy útil en frameworks, pero puede ser confuso de depurar y limita la flexibilidad del programador.

13. Sistemas Empresariales con Herramientas Gratuitas

Las empresas tradicionalmente usan software caro porque desconocen las alternativas gratuitas disponibles. El documento propone usar J2EE con herramientas de código abierto para crear sistemas empresariales completos. También se mencionan patrones de diseño como MVC que separan la presentación de la lógica empresarial.

14. Antipatrones de Diseño de Software

El ser humano ha tratado de resolver los problemas cotidianos de una manera más óptima, teniendo en cuenta costos en tiempo, dinero, mano de obra, etc., de ahí se adopta la idea de estudiar los algoritmos que son procesos secuenciales en la solución de un problema concreto, y teniendo en cuenta que esta clase de soluciones han tenido un desarrollo bastante importante en la historia de la humanidad, tales estudios han llegado al punto de fundamentarse en cierta clase de patrones y antipatrones para el desarrollo de software que facilitan el análisis.

En 1979 el arquitecto Christopher Alexander implementa patrones en el mundo del diseño de construcción de edificios y se implementa y a mediados del tiempo se fue implementando ya la idea generalizada del uso de software y para 1990 nace Design Patterns de Gang of Four donde se recopilan 23 patrones de diseño comunes y de una manera contraria nace el libro Anti-patterns que recopila la contrapartida en cuanto a los patrones de diseño, mostrando así las pautas que no se deben seguir en desarrollo y diseño de software.

15. Errores Comunes en Programación

Los anti-patrones son formas equivocadas pero frecuentes de hacer software. Aunque parecen soluciones rápidas, crean problemas mayores después.

Errores al Escribir Código:

Código volcán: Programar sin orden ni explicaciones, creando un desastre que nadie entiende

Programa todopoderoso: Meter toda la funcionalidad en un solo archivo gigante

Martillo mágico: Querer usar siempre la misma tecnología para todo

Código espagueti: Escribir tan desordenadamente que tocar una línea rompe todo

Errores de Diseño:

Inventar lo ya inventado: Hacer desde cero cosas que ya existen y funcionan

Depender del proveedor: Quedarse atrapado pagando licencias de una sola empresa

Errores de Administración:

Súper equipo: Pensar que añadir más programadores hace el trabajo más rápido

Jefe ausente: No supervisar bien el proyecto hasta que es demasiado tarde

16. Patrones y Antipatrones - Parte II

Lo que se Entiende sobre Patrones de Arquitectura

Los patrones de arquitectura son como planos para construir la estructura general de un programa. Se identifica que hay diferentes tipos según qué tan grande sea lo que se esté construyendo.

17. Integración de Patrones de Seguridad y Diseño J2EE

¿De qué va todo esto?

Este trabajo analiza cómo combinar dos tipos de patrones de software: los patrones clásicos de arquitectura (CJP - Core J2EE Patterns) y los patrones de seguridad (CSP - Core Security Patterns). Básicamente, el problema era que aunque ambos catálogos están relacionados, en la práctica es muy difícil usarlos juntos porque no están bien conectados.

¿Qué son los patrones de diseño que se mencionan aquí?

Este trabajo habla de patrones de diseño pero a un nivel más alto que los típicos patrones del GoF (como Singleton o Observer). Se enfoca en dos tipos específicos:

Patrones de Diseño Arquitectónico (CJP)

Son patrones que ayudan a organizar toda la estructura de la aplicación web.

Patrones de Diseño de Seguridad (CSP)

Son patrones especializados en resolver problemas de seguridad.

18. Uso de Patrones de Diseño: Un Caso Práctico

Los patrones más útiles que menciona el documento:

1. Polimorfismo

Es como tener un control remoto universal

Diferentes dispositivos (TV, equipo de sonido) responden al mismo botón pero cada uno hace algo distinto

En programación: el mismo comando puede hacer cosas diferentes según el objeto

2. Fábrica

Como una máquina que produce objetos según lo que se necesite

Se le dice "necesito un procesador MIPS" y lo crea con todas sus partes

Separa la lógica de crear cosas del resto del programa

Ventajas principales:

Código más fácil de cambiar: Si se necesita modificar algo, no se tiene que tocar todo el programa

Menos errores: Se usan soluciones probadas

Trabajo en equipo: Otros programadores entienden el código más fácil

Reutilización: Se pueden usar partes del código en otros proyectos

19. Enseñanza de la Programación Orientada a Objetos Mediante Patrones de Diseño

¿De qué trata?

Este artículo habla sobre cómo enseñar programación orientada a objetos de una manera más práctica y efectiva, usando patrones de diseño como herramienta educativa.

El problema identificado

Los estudiantes tienen dificultades para aprender conceptos como:

Polimorfismo: Objetos diferentes que responden de manera distinta al mismo comando

Clases abstractas: Plantillas que no se pueden usar directamente

Funciones virtuales: Métodos que cambian según el tipo de objeto

La solución propuesta

En lugar de solo dar teoría, el profesor diseñó un ejercicio práctico.

20. Patrones: De Alexander a la Tecnología de Objetos

Este artículo explica la historia completa de los patrones de diseño, desde sus orígenes en arquitectura hasta su aplicación en programación. Todo comenzó con Christopher Alexander, un arquitecto austriaco que en los años 60 se dio cuenta de algo revolucionario: los espacios que realmente funcionan y hacen feliz a la gente siguen patrones similares, sin importar la cultura o el lugar. Alexander notó que las mejores ciudades, edificios y habitaciones no se diseñan desde cero, sino que siguen "recetas" probadas durante siglos que se pueden documentar y reutilizar.

Su trabajo se basaba en tres conceptos fundamentales: la calidad sin nombre (esa sensación de que un lugar "se siente bien"), el lenguaje de patrones (las "recetas" para crear esa calidad), y el proceso vivo (cómo aplicar estas recetas de forma creativa). Lo que Alexander propuso era revolucionario para su época: en lugar de que cada arquitecto inventara soluciones desde cero, deberían usar y adaptar soluciones que ya habían demostrado funcionar.

Los programadores de los años 80 y 90 se dieron cuenta de algo sorprendente: los problemas de software se repetían igual que los de arquitectura. Las mismas situaciones aparecían una y otra vez en diferentes proyectos. Ward Cunningham y Kent Beck fueron los primeros en adaptar las ideas de Alexander para enseñar programación en Smalltalk en 1987. Jim Coplien siguió documentando

**Referencias**

Alexander, C., Ishikawa, S., & Silverstein, M. (1977). \*A pattern language: Towns, buildings, construction\*. Oxford University Press.

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). \*Design patterns: Elements of reusable object-oriented software\*. Addison-Wesley.

Gonzales, M. (2020). Patrones de diseño en aplicaciones web: Análisis de adopción y beneficios. \*Revista de Ingeniería de Software, 15\*(3), 45-62.

Universidad Cooperativa de Colombia. (2023). \*Patrones de diseño: Guía para estudiantes de sexto semestre\*. Departamento de Ingeniería de Sistemas.