18 DE SEPTIEMBRE DE 2025

INVESTIGACION STRATEGY

COMPLEMENTO

CLAUDIA FERNANDA PEDRAZA PIZANO
TAP
B21140738

Patrón Strategy: Complemento teórico

El patrón Strategy pertenece al grupo de patrones de comportamiento definidos por el "Gang of Four" (GoF). Su propósito es encapsular algoritmos en clases independientes que pueden intercambiarse sin modificar el código cliente. Esto permite que el comportamiento de un objeto varíe en tiempo de ejecución según la estrategia seleccionada.

Componentes clave

1. **Contexto**: Clase que utiliza una estrategia sin conocer su implementación.
2. **Strategy (interfaz)**: Define el método común que todas las estrategias deben implementar.
3. **Estrategias concretas**: Implementan la interfaz y contienen la lógica específica.

Beneficios

- Desacoplamiento entre cliente y algoritmo.
- Extensión sin modificar clases existentes (principio abierto/cerrado).
- Cambio dinámico de comportamiento.
- Eliminación de condicionales.
- Reutilización de estrategias en distintos contextos.

Aplicaciones modernas

- Procesamiento de pagos (tarjeta, PayPal, criptomonedas).
- Validación de formularios.
- Comportamientos en videojuegos.
- Algoritmos de compresión.
- Sistemas de recomendación.

¿Cuándo usar Strategy?

- Cuando tienes **múltiples algoritmos** que realizan una tarea similar (por ejemplo, ordenamientos, validaciones, cálculos).

- Cuando deseas **evitar condicionales** extensos en el código cliente.
- Cuando necesitas **cambiar el comportamiento en tiempo de ejecución** sin alterar la estructura del objeto.

Ejemplos reales de uso

1. **Sistemas de pago**
   - Aplicaciones como Amazon o Mercado Libre usan Strategy para procesar pagos con tarjeta, PayPal, o criptomonedas.
   - Cada método de pago es una estrategia que implementa una interfaz común como ProcesarPago.
2. **Compresión de archivos**
   - Programas como WinRAR o 7-Zip permiten elegir entre ZIP, RAR, TAR, etc.
   - Cada algoritmo de compresión es una estrategia intercambiable.
3. **Validación de formularios**
   - En sistemas web, se pueden aplicar diferentes estrategias de validación: email, contraseña, teléfono.
   - Cada validación se encapsula como una estrategia que puede cambiar según el tipo de formulario.

Referencias

- Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.
- Ojeda, I. (2024, octubre 26). Flexibilidad y escalabilidad: Usando Strategy y Factory Patterns. *DEV Community*. https://dev.to/isaacojeda/flexibilidad-y-escalabilidad-usando-strategy-y-factory-patterns-5fe9
- IONOS. (2020, octubre 19). Strategy pattern: un patrón de diseño de software para estrategias variables de comportamiento. *IONOS Digital Guide*. https://www.ionos.es/digitalguide/paginas-web/desarrollo-web/strategy-pattern/

- Padua, M. (2024, noviembre 29). Patrones de diseño: ¿qué son?, usos, tipos y ventajas. *IT Masters Mag*. https://www.itmastersmag.com/transformacion-digital/patrones-de-diseno-descripciones-estandarizadas-para-problemas-repetitivos/