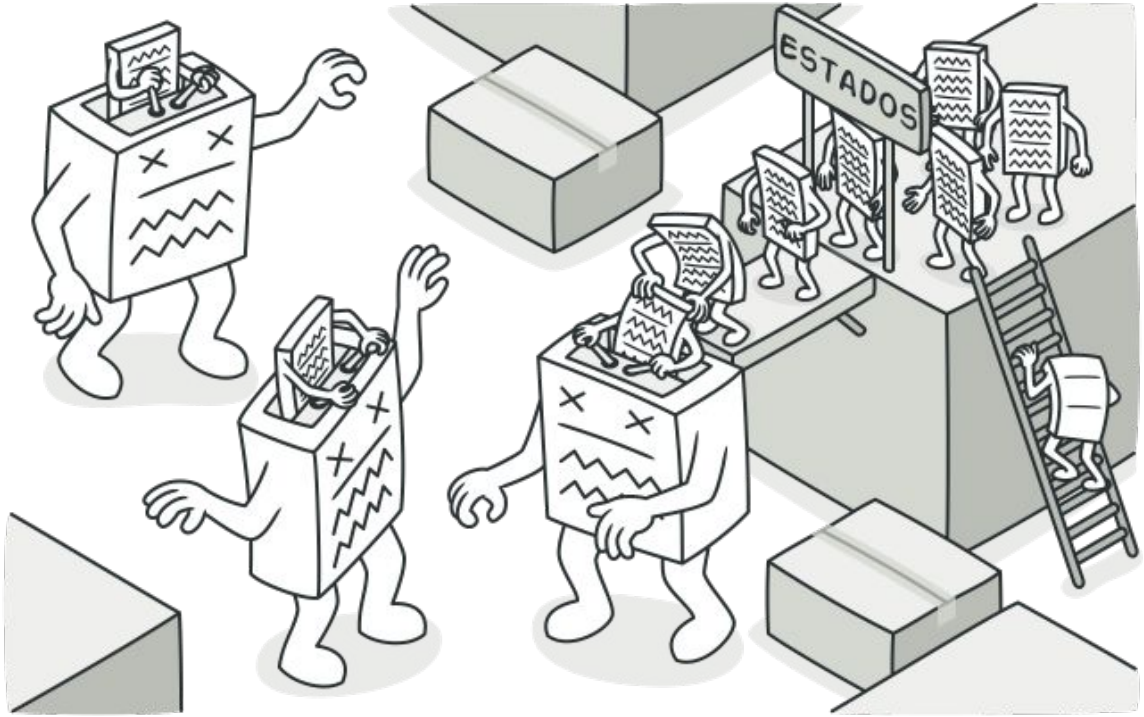
Universidad Popular del Cesar

Patrón de Diseño State (Estado)

Marryy Selena Garay Larios

Abraham Sarabia Sereno

¿Qué es el Patrón State?



Problema que resuelve

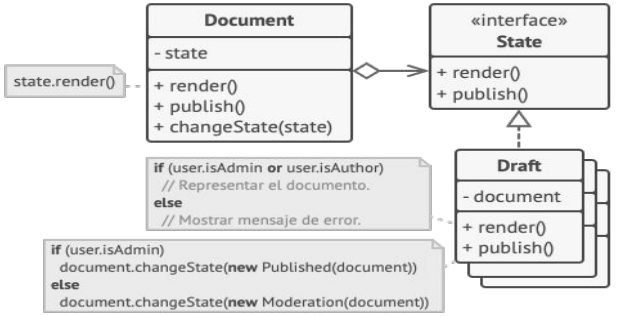
```
switch (expresion) {

      case valor1:
           sentencias B1;
      break;

      case valor2:
           sentencias B2;
      break;

      case valor3:
           sentencias B3;
      break;

      ...

      [default:
           sentencias B4;]

}
```

```
public class Principal {
    public static void main(String[] args) {
        int calificacion = 7;
        boolean avisarFamilia = false;
        if(calificacion >= 0 && calificacion < 5){
            System.out.println("Insuficiente");
            avisarFamilia = true;
        }else if(calificacion < 7){
            System.out.println("Suficiente");
        }else if(calificacion < 9){
            System.out.println("Notable");
        }else if(calificacion < 10){
            System.out.println("Sobresaliente");
        }else if(calificacion == 10){
            System.out.println("Matrícula");
        }else{
            System.out.println("Calificacion no valida");
        }
        if(avisarFamilia){
            System.out.println("Avisar a la familia");
        }
    }
}
```

Estructura del Patrón State

Componentes clave:

- **Contexto**: contiene referencia al estado actual
- **Interfaz State**: define métodos comunes
- **Estados concretos**: clases que implementan la lógica específica



Ejemplo

Reproductor de Música:

- Estado: Reproduciendo, Pausado, Detenido

Botones:

- Play: cambia entre Pausado → Reproduciendo
- Pause: Reproduciendo → Pausado
- Stop: Cualquier estado → Detenido

Cada estado es una clase con su propio comportamiento.

Ventajas del Patrón State

- Elimina condicionales
- Fácil de extender (principio OCP)
- Cada estado tiene su propia clase (SRP)
- Comportamiento más limpio y organizado
- Código más intuitivo y reusable
- Facilita pruebas unitarias por estado aislado

Desventajas del Patrón State

1. **Mayor número de clases**
   - Cada estado se implementa como una clase distinta.
   - Esto puede generar una explosión de clases si hay muchos estados.
2. **Complejidad innecesaria para casos simples**
   - Si solo hay 2 o 3 estados y poca lógica asociada, el patrón puede ser “overkill”.
   - En estos casos, un simple `if`/`switch` es más fácil y directo.
3. **Dificultad para compartir estado interno**
   - Si los estados necesitan acceder o modificar datos comunes del contexto, puede ser más difícil manejar ese acoplamiento sin violar encapsulamiento.
4. **Mayor esfuerzo de diseño inicial**
   - Se requiere una planificación más detallada de las transiciones y del comportamiento de cada estado.
   - No es tan flexible para cambios rápidos si el dominio aún no está bien definido.

Ejemplo Aplicado

Mostrar codigo

Casos de uso en la vida real

- Cajeros automáticos (esperando PIN, seleccionando operación, entregando dinero)
- Procesos de aprobación (borrador → en revisión → aprobado → publicado)
- Tráfico vehicular (semáforo verde, amarillo, rojo)

Cada uno con comportamientos distintos según su estado.

Conclusión

¿Cuándo usar el Patrón State?

- Cuando el objeto cambia su comportamiento según su estado
- Si tienes muchos `if`/`switch` para controlar estados

Beneficios clave:

- Escalabilidad
- Mantenibilidad
- Claridad en el diseño
- Separación de responsabilidades
- Permite que el contexto delegue comportamiento

Universidad Popular del Cesar

¡Gracias por su atención!