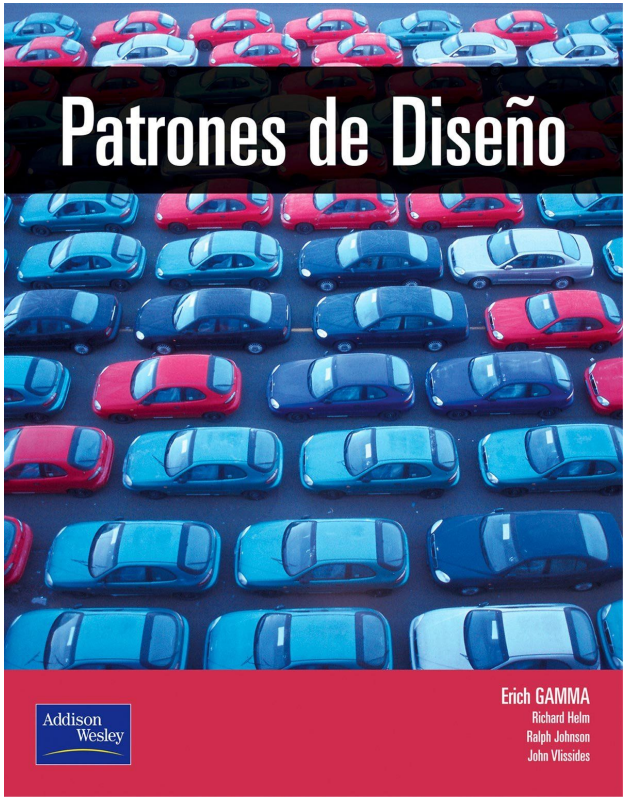
Diseño

Patrones de Diseño

Solución genérica a un problema recurrente de diseño.

Libro recomendado



Estructura de un patrón

- Nombre
- Clasificación: creación, estructural o comportamiento
- Intención:
- Motivación
- Estructura: diagrama de clases
- Colaboraciones: diag. secuencia y colaborac.
- Consecuencias
- Implementación

Clasificación

Estructurales: componer clases o ensamblar objetos. Ej: Adapter, Facade, Composite

Comportamiento: describen algoritmos y flujos de control. Ej: Observer, State

Creación: delegan alguna parte de la creación de objetos. Ej: Singleton, Factory

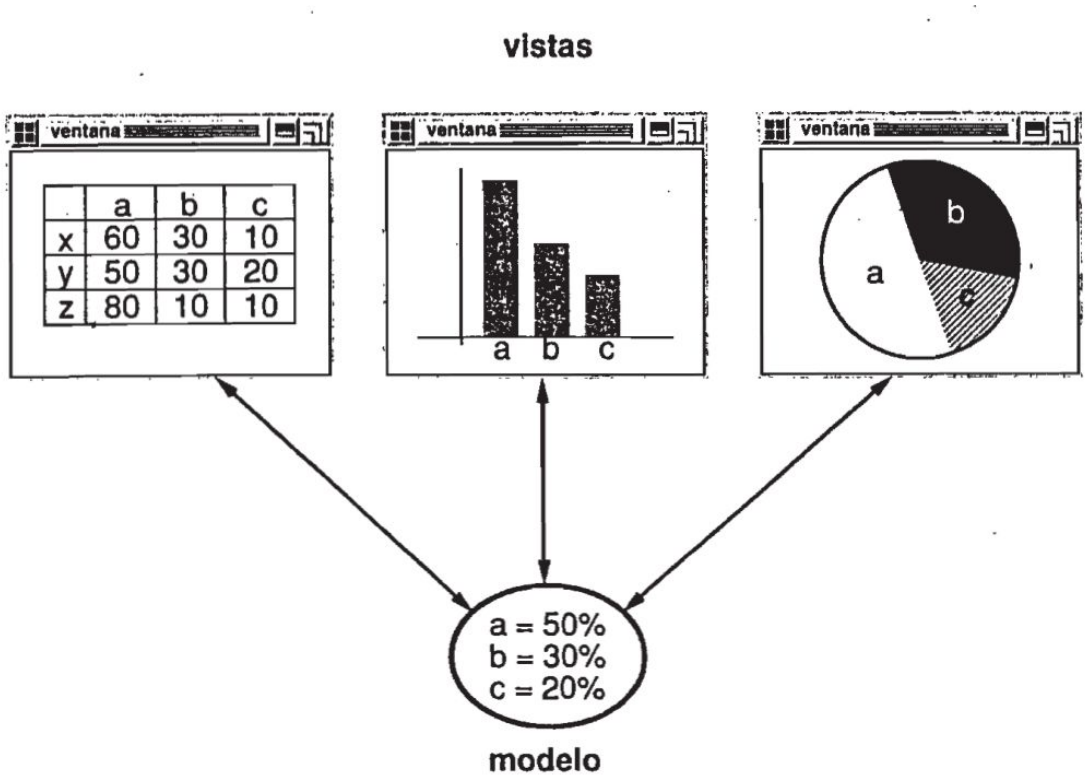
Tabla Completa GoF

Tabla 1.1: Patrones de diseño				
		Propósito		
		De Creación	**Estructurales**	**De comportamiento**
Ámbito	**Clase**	Factory Method (99)	Adapter (de clases) (131)	Interpreter (225) Template Method (299)
	Objeto	Abstract Factory (79) Builder (89) Prototype (109) Singleton (119)	Adapter (de objetos) (131) Bridge (141) Composite (151) Decorator (161) Facade (171) Flyweight (179) Proxy (191)	Chain of Responsibility (205) Command (215) Iterator (237) Mediator (251) Memento (261) Observer (269) State (279) Strategy (289) Visitor (305)

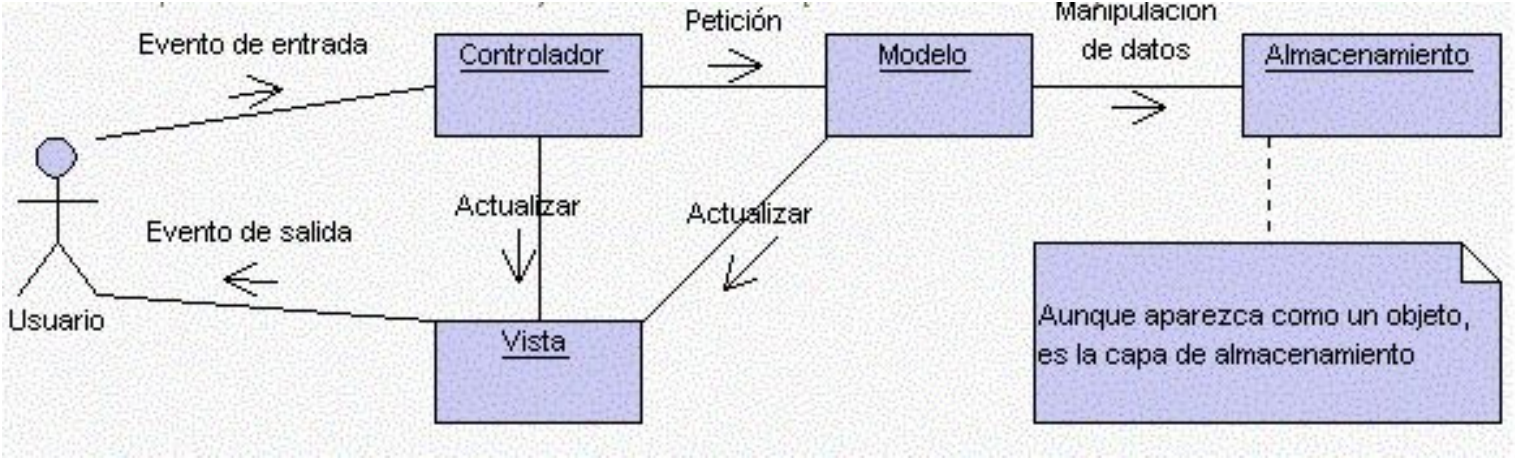
Modelo Vista Controlador

- **Modelo**: Representación de la información con la que el sistema opera.
- **Vista**: representación de la interfaz de usuario.
- **Controlador**: modo en que la interfaz reacciona a la entrada del usuario.

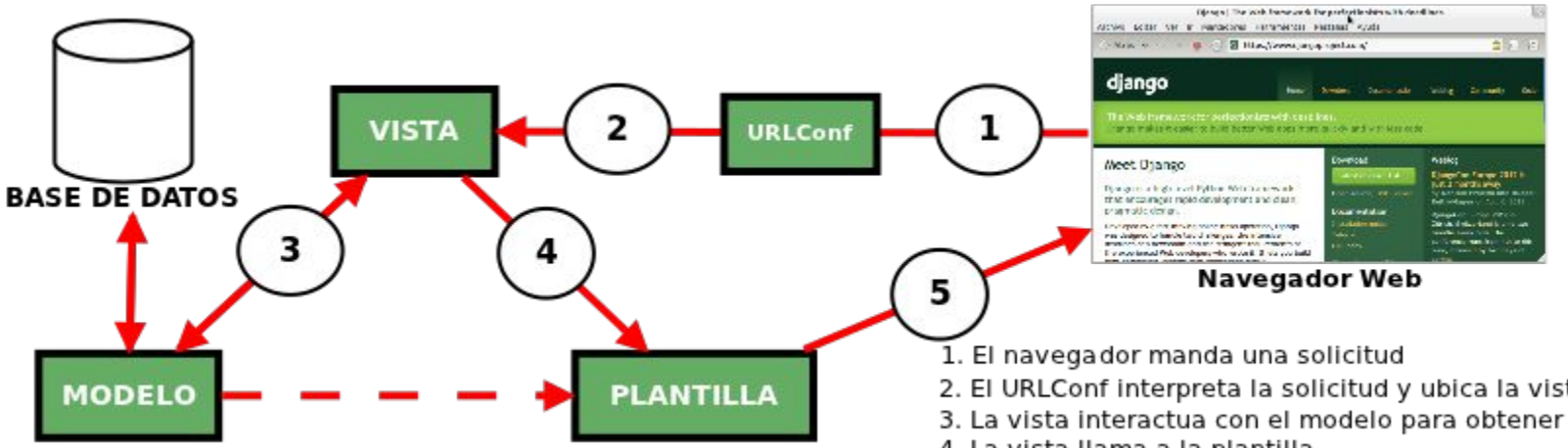
Patrón MVC SmallTalk



MVC genérico



MVP Django



1. El navegador manda una solicitud
2. El URLConf interpreta la solicitud y ubica la vista apropiada
3. La vista interactua con el modelo para obtener datos
4. La vista llama a la plantilla
5. La plantilla renderiza la respuesta a la solicitud del navegadc

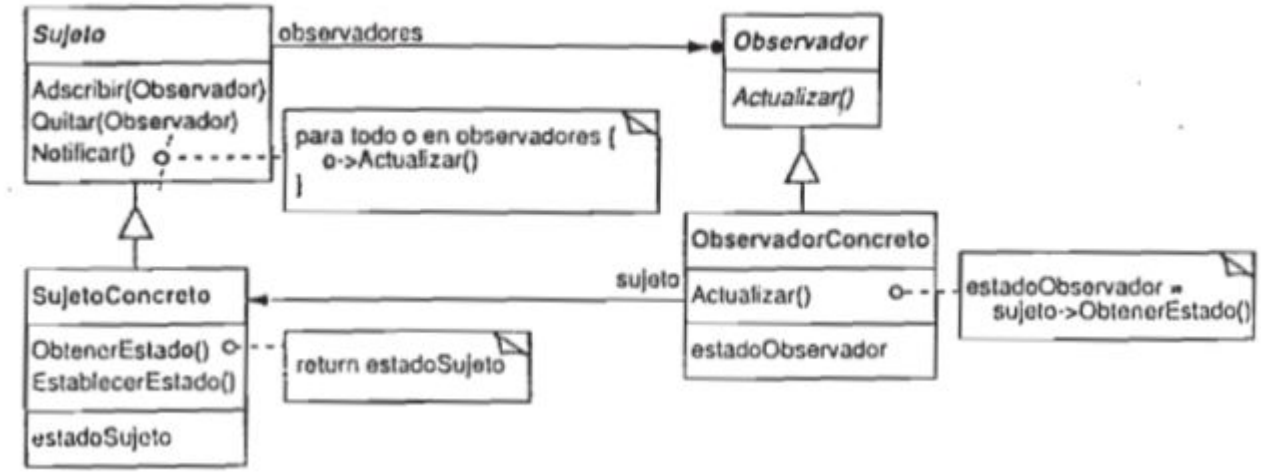
Observer

Propósito

Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado se notique y se actualicen automáticamente todos los objetos que dependen de él.

Observer

ESTRUCTURA

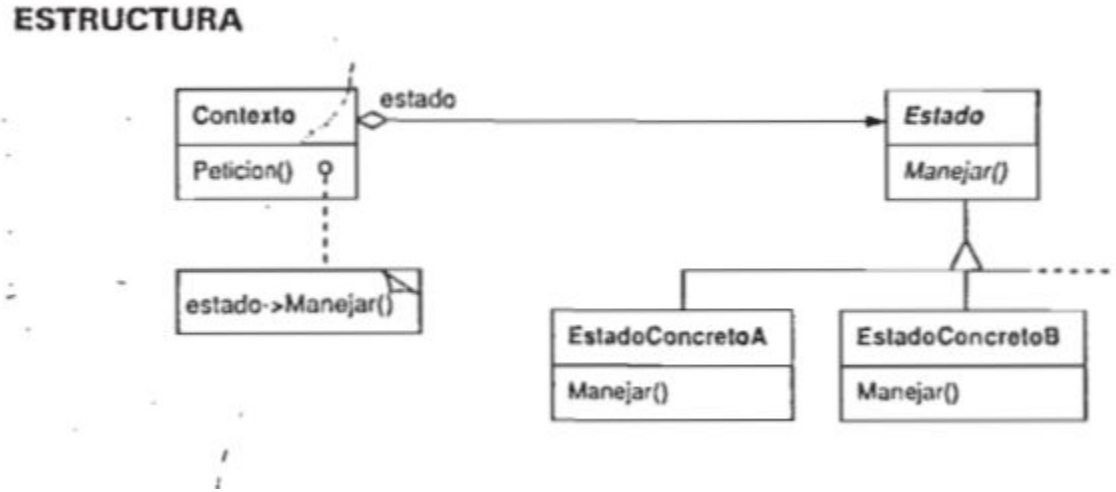


State

Propósito

Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Parecerá que cambia la clase del objeto.

State


ESTRUCTURA
Contexto
Peticion()
estado
Estado
Manejar()
estado->Manejar()
EstadoConcretoA
Manejar()
EstadoConcretoB
Manejar()

Ejemplo Patrón State

```
Class Impresora{
  Estado estado
    imprimir(){
          if(estado = bloquedo) mostrar mensaje
             Else if (estado = lista) imprimr
    }

}

Class EstadoLista extend Estado{
            imprimir(){
               Imprimir
            }
}
Class EstadoPapelBloqueado extend Estado{
            imprimir(){
               Devolver mensaje papel bloqueado
            }
}

Class Estado (){
            Id
            Nombre
}
```

impresora.estado.imprimir()