**Propuesta**

Nota: Especificar acá un diseño a alto nivel con diagramas explicando la implementación realizada.

MVC

En este patrón se separa el modelo, la vista y el controlador para incrementar la flexibilidad y la usabilidad.

El modelo contiene los datos, cuando estos datos cambian el modelo se comunica con sus vistas informándole del cambio y las vistas se comunican con el modelo para actualizar los cambios.

De esta forma se desacoplan las vistas y el modelo haciendo que, los cambios en el modelo afecten a sus vistas sin necesidad de que el modelo conozca los detalles de la representación de las mismas (Patrón Observer).

El MVC se compone de los siguientes 3 patrones: Observer, Composite y Strategy

Observer (Observador)

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

Composite (Compuesto)

Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de maneara uniforme a los objetos individuales y a los compuestos.

Strategy

Define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

Una estrategia es un objeto que representa un algoritmo. Este es útil cuando queremos reemplazar un algoritmo ya sea dinámica o estáticamente, cuando existen muchas variantes del mismo o cuando tiene estructuras de datos complejas que queremos encapsular.

Decorator

Para añadir capacidades a un objeto

Abstract Factory

Fabrica abstracta: Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.

Factory Method

Define una interface para crear un objeto, pero deja que sean las subclases quienes decidan qué clase se instancia. Permite que una clase delegue en sus subclases la creación de objetos.

Command (Orden, Comando)

Encapsula una petición dentro de un objeto, permitiendo asi parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer las operaciones.

Decorator (Decorador)

Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

State (Estado)

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

Interpreter

Dado un lenguaje, define una representación de su gramatica junto con un intérprete que usa dicha representación para interpretar sentencias del lenguaje.

Prototype (Prototipo)

Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crea nuevos objetos copiando de este prototipo.

Singleton (Único)

Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.