Catalogo patrones de diseño de software

Los grupos de patrones a implementar en el sistema de información, Conjunto Web son:

	Creación	**Estructurales**	**Comportamiento**
Clases	Factory Method		Template Method
Objetos	Abstract Factory Builder Prototype Singleton	Adapter Composite Decorator Proxy Flyweight	Observer Strategy Iterator Chain of responsibility

Grupo de patrones de creación:
Estos patrones buscan en cierta forma «despreocupar» al sistema de cómo sus objetos son creados o compuestos, en pocas palabras busca crear y configurar clases y objetos.

- **Factory method:** Se utilizara cuando definamos una clase a partir de la que se crearán objetos pero sin saber de qué tipo son, siendo otras subclases las encargadas de decidirlo.

- **Abstract factory:** resulta útil en casos en los que necesitemos crear familias de objetos relacionados o dependientes entre sí, sin especificar sus clases concretas.

- **Builder:** Puede ser utilizado cuando necesitemos crear objetos complejos compuestos de varias partes independientes.

- **Prototype:** Este patrón nos será útil si necesitamos crear y manipular copias de otros objetos.

- **Singleton:** Asegurar que una clase tenga una única instancia y provee acceso global a esta.

Grupo de patrones estructurales:
Los patrones estructurales nos ayudan a definir la forma en la que los objetos se componen.

- **Adapter:** Definir una clase intermedia que sirva para que dos clases con diferentes interfaces puedan comunicarse.
- **Composite:** Se creará y manejará estructuras de objetos en forma de árbol, en las que un objeto puede contener a otro(s). de tal forma que ambos objetos comparten una misma Interface que define métodos que deben implementar.
- **Decorator:** Se encargará de añadir dinámicamente funcionalidad a un Objeto.
- **Proxy:** Se utilizará como intermediario para acceder a un objeto, permitiendo controlar el acceso a él.
- **Flyweight:** El objetivo es evitar crear un gran número de objetos similares, mejorando con ello el rendimiento del sistema de información.

Grupo de patrones de comportamiento:
Este grupo de patrones se centran en la interacción entre asociaciones de clases y objetos definiendo cómo se comunican entre sí.

- **Template method:** Sera útil en casos en los que podamos implementar en una clase abstracta el código común que será usado por las clases que heredan de ella.
- **Observer:** Definiremos una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes.
- **Strategy:** Consistirá en crear un objeto que pueda comportarse de formas diferentes
- **Iterator:** Se utilizará cuando necesitemos recorrer secuencialmente los objetos de un elemento agregado sin exponer su representación interna. Así pues, este patrón de diseño nos resultará útil para acceder a los elementos de un array o colección de objetos contenida en otro objeto**.**
- **Chain of responsability:** Este patrón puede resultarnos útil en casos en los que un objeto emisor de una petición desconozca qué objeto(s) podrá(n) atender a la misma.