## Patrón: Modelo vista controlador

### Tipo: Arquitectónico

### Contexto de Aplicación

Nuestra aplicación maneja datos, los procesa y los muestra al usuario. Con este patrón, nuestra intención es distinguir estas actividades, y dividir el trabajo a realizar entre ellas (las actividades se dividen en 3 capas, en el modelo (procesamiento y almacenamiento de datos), en la vista (consumo de datos) y en el controlador (interacción entre el consumo de datos y las operaciones con esos datos)).

### Clases o paquetes creados

Paquetes model, repository, service y web. En el paquete de model se incluyen todas las clases que describen a los distintos tipos de objetos que almacenaremos en nuestra base de datos. En repository se incluyen las interfaces que luego implementarán las clases de service; en estas clases se definen las funciones que trabajan el acceso a la base de datos, es decir, donde se aplican las funciones CRUD. En web tenemos todas las clases controladoras, redirigen los datos a los distintos archivos jsp que conforman la vista de nuestra página.

### Ventajas alcanzadas al aplicar el patrón

Permite manejar fácilmente una gran cantidad de datos, operar con ellos, y mostrarlos al usuario. También es fácil cambiar aspectos de las distintas clases sin que afecten a otras partes de la aplicación (principio de modularidad), lo que facilita la mantenibilidad del sistema.

## Patrón: Front controller

### Tipo: Arquitectónico

### Contexto de Aplicación

Los datos a procesar por nuestra aplicación no reciben un tratamiento trivial, hay multitud de operaciones que son ejecutadas. Con este patrón pretendemos que las operaciones que se encuentran entre las operaciones del usuario y los datos almacenados en la base de datos estén mejor organizados y se simplifiquen en un menor número de clases, que contienen todos los métodos relacionados con cierto tipo de objetos.

### Clases o paquetes creados

Creamos el paquete web, que contiene las clases controladoras de la aplicación. Estas clases hacen uso de ciertas anotaciones de Spring que permiten hacer uso de este patrón con facilidad, simplificando la especificación de las urls que harán las llamadas a las funciones que se ejecutarán en nuestra aplicación.

### Ventajas alcanzadas al aplicar el patrón

Terminamos creando muchas menos clases y la complejidad de la definición de las funciones de los controladores se reduce, ya que se reutiliza gran parte del código de los distintos métodos que conforman el controlador de cierto tipo de objeto de la base de datos.