Modelo MVC, Arquitectura en Capas

¡Comencemos!

¿Qué aprenderás?

- Conocer el patrón de diseño MVC.
- Diferenciar el modelo de capas vs MVC.

Introducción

Previamente se ha estudiado el servlet y su funcionamiento, las páginas jsp con técnicas de scriptlets y jstl, el envío de valores entre formularios y su forma de comunicación. Con estos ingredientes y un poco de creatividad ya es posible generar pequeños sistemas que comparten datos entre sí, y con el lenguaje java es posible dar lógica a la información para generar un resultado que de valor al usuario, o en su defecto para crear prototipos.

En este capítulo utilizaremos todo lo aprendido hasta ahora para comprender lo que es una verdadera arquitectura empresarial bajo un estilo arquitectónico conocido como MVC.

Conocer el patrón de diseño MVC

Hasta el momento, hemos trabajado en lo que se conoce como arquitectura de 1 capa. En este tipo de arquitectura los request son capturados por páginas jsp o servlets que reciben, procesan y devuelven la información al cliente. En este modelo, un componente web perfectamente combina lógica de negocio y lógica de presentación.

Existe un segundo estilo de arquitectura de sistemas empresariales que es conocido como arquitectura de MVC (modelo-vista-controlador). En una arquitectura MVC se apuesta por la división de responsabilidades y funcionalidades la cual consta de separar la lógica de negocio de la lógica de presentación.

Para una definición más formal, pero no muy alejada a la descripción real dada por su creador Trygve Reenskaug en la década del 70, podemos decir que este modelo de arquitectura considera tres roles. El modelo es un objeto que representa información sobre el dominio. En su forma más esencial orientada a objetos el modelo es un objeto dentro de un modelo de dominio (las entidades de negocio). La vista representa la visualización del modelo en la interfaz de usuario, en este caso si el modelo es un objeto persona, tal objeto se visualiza en la pantalla en forma de lista, tabla, o entidad. Por último existe el controlador, que se encarga de tomar la entrada del usuario, manipular el modelo y hacer que la vista se actualice.

Cuando piensen en MVC, imaginen dos separaciones principales: separar la presentación del modelo y separar el controlador de la vista. La separación de presentación y modelo es una forma de estructurar un sistema basado en un buen diseño de software y es importante porque la vista y el modelo son de naturalezas distintas y tienen responsabilidades muy distintas. Por un lado si pensamos en la vista, pensamos en la interfaz de usuario y en todos los mecanismos para que el sistema tenga una interfaz amigable, intuitiva, rápida, etc. Por otro lado si pensamos en el modelo estamos enfocados a preocuparnos de la lógica de nuestro negocio, de los componentes que lo integran, en políticas empresariales y en base de datos por dar unos ejemplos.

Ahora, según el contexto de la aplicación los usuarios siempre desean ver la información básica del modelo de maneras diferentes. La separación de la vista permite desarrollar múltiples presentaciones y sin embargo utilizar el mismo modelo. Es tan importante esta separación que incluso permite cambiar de tecnología de vista sin mucho problema.

Diferencias entre el modelo de capas y MVC

Si bien estas dos arquitecturas son muy similares tanto en forma como en fondo no son lo mismo. A continuación examinaremos cada una para dejar claras sus diferencias:

- **Arquitectura en capas**: Es un estilo arquitectónico que desacopla las responsabilidades de un sistema en capas lógicas, identificando 1 capa o N dependiendo de la necesidad de la aplicación. En una arquitectura de 3 capas por ejemplo se consideran la capa de presentación, la capa de negocio y la capa de datos. Esta arquitectura ayuda a desacoplar un sistema y es una de las formas de estructurar software más común. Su flujo es unidireccional y debe proveer de buenas interfaces de comunicación.

- **Arquitectura MVC**: Aquí podemos analizar un punto en común entre la arquitectura en capas, y ese punto es la separación de responsabilidades, pero la gran diferencia radica en que la arquitectura modelo-vista-controlador, además de separar, asigna una responsabilidad única a cada capa permitiendo que la comunicación entre ellas no sea unidireccional sino que sea a petición del flujo de usuario. Esto quiere decir que existe una entidad de orquestación (el controlador) que es capaz de comunicarse con la capa de vista y modelo y decidir mediante mensajes a qué parte del modelo se dirigen las peticiones de la capa de vista.

El modelo de una capa se basa en contener la lógica de negocio y la lógica de presentación en un solo proyecto. Esto puede ser provechoso y no tener problemas para aplicaciones pequeñas, pero para sistemas de mayor complejidad no es para nada recomendado. Pensemos en los jsp que tienen en un mismo archivo mucha lógica java, o por otro lado imaginemos un servlet que está lleno de scriptlet que genera código html. Con MVC se separa en gran parte la presentación (html) de la lógica de negocio (java) en componentes aislados pero con la capacidad de comunicarse entre sí (responsabilidad única).

Componentes del modelo MVC

En una aplicación MVC, los componentes de una aplicación son divididos en tres categorías:

- **Modelo**: El modelo contiene todas las clases e instancias de clases que representan el dominio del sistema u aplicación. Pensemos por ejemplo en un sistema de facturación en el cual existen objetos que representan boletas, clientes, facturas, etc. En esta capa también estarán las clases de apoyo al sistema, no preocuparse por estos conceptos, ya que se verán más adelante, pero los DTO, los DAO y otras estructuras también están en esta capa.

- **Vista**: En esta capa es donde se ubica todo lo que se mostrará al usuario, como por ejemplo las páginas html, los archivos javascript y los elementos de estilos. Esta capa es la denominada front-end.

- **Controlador**: Si bien tenemos por un lado una capa de modelo en donde se programa la lógica de negocio del sistema y la representación de las entidades de negocio y por el otro la capa de vista que contiene todo lo que el usuario verá en su pantalla, hace falta un intermediario que sepa comunicarse con la capa de vista y también con la capa de modelo. Esta capa es el controlador, el cual conoce muy bien cómo conversar con el front end y también conoce todos los métodos del modelo.

Para graficar este modelo veremos un diagrama que lo explica de forma teórica:

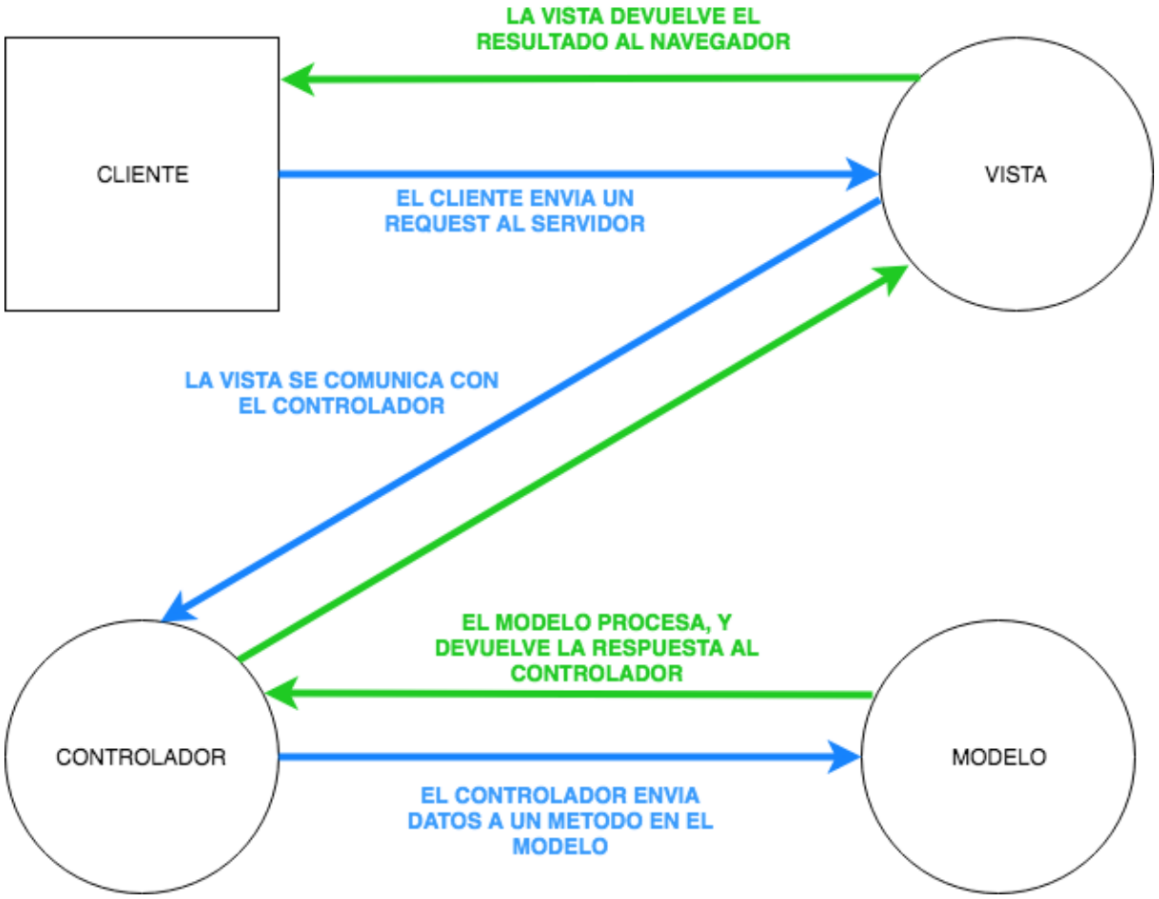

Imagen 1. Diagrama de alto nivel arquitectura MVC.
Fuente: Desafío Latam

El diagrama es de alto nivel por lo que desde aquí se va a profundizar en el diseño de un sistema web en capas. Cuando se diseña un sistema web sea en el lenguaje que sea, se dividen en capas separadas, tal y como se muestra en la imagen.

Arquitectura en capas

En estas arquitecturas cada elemento del sistema se describe y se contiene de acuerdo a su responsabilidad en capas muy bien definidas. En los proyectos empresariales con java que se pueden encontrar en distintas entidades como por ejemplo grandes bancos, tiendas de retail y minería por ejemplo se utilizan estas capas para estructurar los sistemas. A continuación describiremos estas capas.

- **Capa de datos**: Encargada de trabajar con los datos del sistema, comunicando directamente con algún sistema de base de datos.

- **Capa de acceso a datos**: Esta capa es en donde se representan las entidades de negocios en objetos perfectamente reproducibles en lenguaje java. En capítulos posteriores se profundizará en esta capa, la cual es conocida como DAO (Data Access Object).

- **Capa de aplicación**: Corresponde al modelo, pero en términos prácticos en esta capa se programa la lógica de negocio de la aplicación.

- **Capa web**: Esta es la capa de presentación, la cual contiene todos los componentes que presentan los datos al usuario.

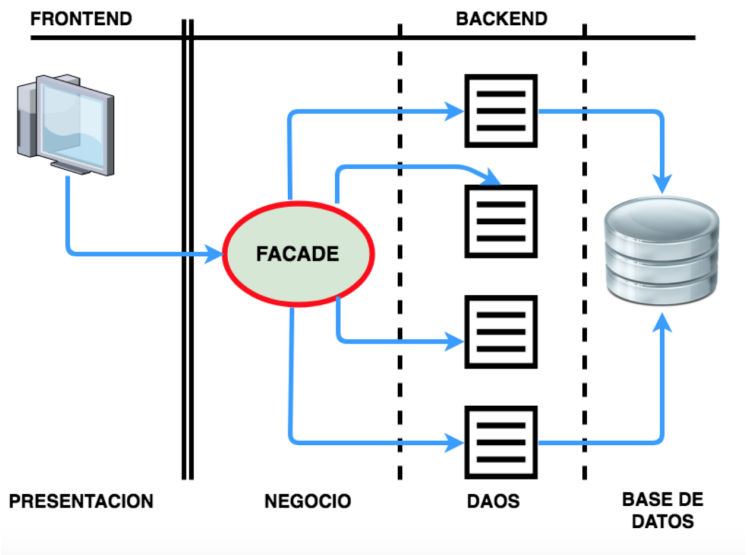

Imagen 2. Arquitectura en capas.
Fuente: Desafío Latam

Cada una de las capas descritas anteriormente se definen de acuerdo a su responsabilidad.

- **Capa de datos**: Comenzaremos describiendo la última capa, la encargada de los datos. En esta capa se proveen los mecanismos de persistencia que permiten que se almacenen, se consulten, modifiquen y eliminen los datos. Esta capa se implementa junto a algún motor de base de datos como oracle, mysql, sqlserver, etc.

- **La capa de acceso a datos DAO**: Este sector de la arquitectura interactúa directamente con la capa de datos y se encarga de acceder a ella para poder crear, eliminar, modificar y buscar información. Esta capa si bien se comunica con la base de datos, también cuenta con otro tipo de entidad de nombre DTO (Data Transfer Object).

- **DTO**: Si bien esta clase no aparece en el diagrama, ya que trabaja codo a codo con el dao, es necesario dar su descripción. Esta clase está encargada de proveer los objetos que son un fiel reflejo de las entidades de base de datos (tablas) que residen en la base de datos.

  En términos simples imaginemos que tenemos la tabla usuarios que tienen los atributos nombre, edad y apellidos. Para que el sistema pueda interactuar con la base de datos y con esta tabla, el DTO debe proveer una clase que tenga exactamente la misma estructura que la tabla de usuarios. Esto se traduce en una clase java de nombre usuarios con los atributos de nombre edad y apellidos.

- **La capa de negocios**: Esta capa como se describió anteriormente es la encargada de procesar los datos y de otorgar la lógica de negocios al sistema. En el diagrama se ve el nombre facade, el cual es la denominación de un patrón de diseño que al igual que el conserje de un edificio, conoce a todos los residentes y sabe dónde viven de acuerdo al número de departamento. La capa de negocios es parte importante de una arquitectura en capas.

Pensemos que nuestra aplicación provee la información de las ventas del mes y en la capa de datos existen multitud de clases. El facade es quien sabe dónde encontrar, por ejemplo las ventas con boletas, y se comunica con ella sin problemas. Este patrón facade provee un acceso ordenado a las clases DAOS o también a más clases de negocios. Esta capa es el controlador si lo extrapolamos a la arquitectura MVC.

- **La capa web**: Aquí se generan las páginas jsp que se mostrarán al usuario final después de que el request pase al controlador y su clase facade, a la capa de datos dao, consulte en la base de datos para luego retornar la salida.

- **Facade**: Este concepto no es una capa, es más bien un patrón de diseño que proporciona un acceso simple y unificado a una compleja estructura de sistemas. En los sistemas grandes siempre hay muchos módulos de software que deben interactuar entre sí, y el facade permite que el sistema tenga solo un punto de entrada, simplificando en gran medida el desarrollo y el mantenimiento del sistema. Si se observa nuevamente el diagrama anterior, el facade se encuentra marcado en el centro de la arquitectura y es el punto de entrada a los distintos integrantes del backend.

El backend y el frontend de una aplicación

Se ha hablado del backend y el front end. Vamos a desglosar los términos: Las capas que tienen interacción directa con el usuario final y aquellas capas que le dan soporte a las primeras conforman lo que se denomina el front-end de un sistema. Se denomina front end simplemente porque es la capa frontal y en él podemos encontrar distintas tecnologías como por ejemplo servlets y jsp por el lado java, la combinación html-css-js o algún framework de javascript como nodejs.

Por otro lado, todas las capas que están entre la capa de negocio y la capa de datos constituyen lo conocido como backend, o en mejores términos la parte trasera de la aplicación. Aquí podemos encontrar toda la lógica de negocio en distintos lenguajes de programación que trabajen en el lado del servidor. En las estructuras JEE se mantienen los controladores, las entidades de negocio, los java beans y todas las clases que componen el sistema.

Por el lado de la persistencia, se puede encontrar la tecnología JPA que se encarga de trabajar con la base de datos. Es importante señalar que estas capas y su disposición no son una regla y la implementación de las mismas puede variar de acuerdo al tipo de sistema o de los requerimientos. El siguiente diagrama grafica esta arquitectura:

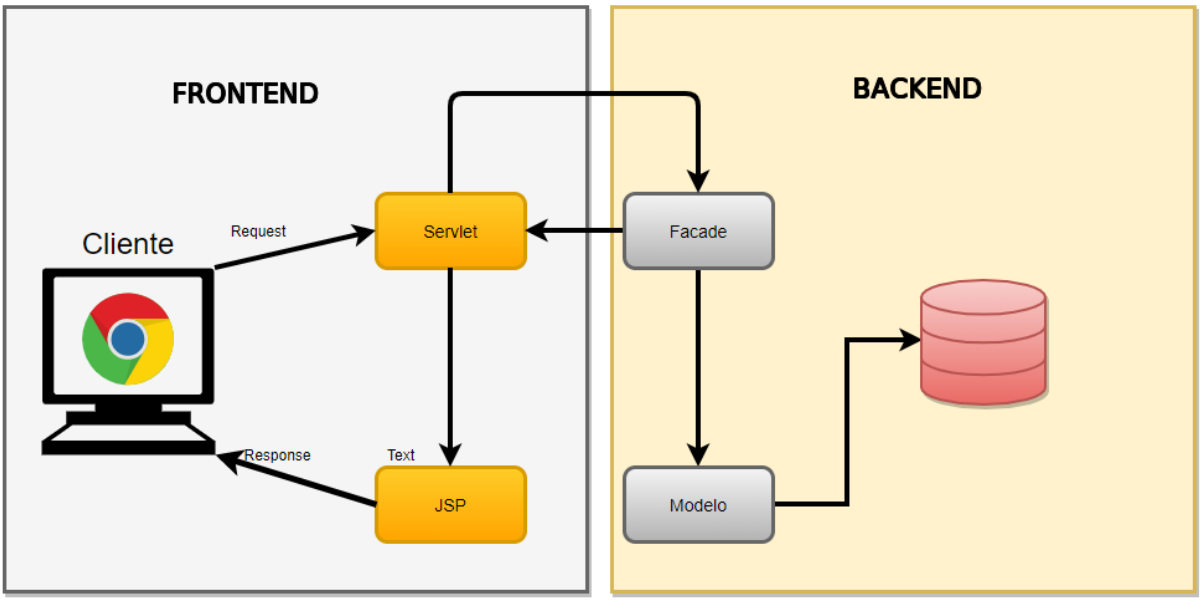

Imagen 3. Arquitectura backend y frontend.
Fuente: Desafío Latam