**Modelo Vista Controlador (MVC)**

Estructura basada en capas:

1. **Modelo**

Clase producto creo Datos

Atributos. Creo los Getters (si quiero que se pueda ver y no modificar solo dejo el get y elimino el Set, una vez creado no se puede cambiar.) y setters. Y los constructores que permiten crear el objeto, que es un método especial publico que se llama igual que la clase

Los modelos para acceder a los datos.

Las reglas de negocio

ToString()

No obstante, cabe mencionar que cuando se trabaja con MCV lo habitual también es utilizar otras librerías como PDO o algún ORM como Doctrine, que nos permiten trabajar con abstracción de bases de datos y persistencia en objetos. Por ello, en vez de usar directamente sentencias SQL, que suelen depender del motor de base de datos con el que se esté trabajando, se utiliza un dialecto de acceso a datos basado en clases y objetos

<https://desarrolloweb.com/articulos/que-es-mvc.html>

Ellos son los que deben saber cómo operar en diversas situaciones y las cosas que pueden permitir que ocurran en el proceso de ejecución de una aplicación.

Por ejemplo, pensemos en un sistema que implementa usuarios. Los usuarios pueden realizar comentarios. Pues si en un modelo nos piden eliminar un usuario nosotros debemos borrar todos los comentarios que ha realizado ese usuario también. Eso es una responsabilidad del modelo y forma parte de lo que se llama la lógica del negocio.

1. **Vista**

Main el que se comunica con el cliente, métodos seleccionar producto

Ingresar cantidad producto, seleccionar tipo de pago, seleccionar tipo de Entrega.

Sysout””

Métodos “ReservarStock()

Comprar()

New Clasecontroller().metodo(parámetros)

Validaciones de reglas, filtraciones.

Contiene el código de nuestro programa que nos permitirá interactuar a través de interfaces con el usuario.

Validaciones, filtros.

En la vista generalmente trabajamos con los datos, sin embargo, no se realiza un acceso directo a éstos. Las vistas requerirán los datos a los modelos y ellas se generará la salida, tal como nuestra aplicación requiera.

1. **Controlador cada clase debe tener su controlador**.

ClaseController

Para reservarStock necesita recibir fecha, cantidad Stock, tipo producto del vendedor, y del Cliente tipo producto, cantidad del producto,

**HashMap trabaja con parejas, índice o llave y el segundo parámetro el valor)**

**Id cliente me entrega los pedidos o reservas del cliente.**

Contiene el código necesario para responder a las acciones que se solicitan en la aplicación, como visualizar un elemento, realizar una compra, una búsqueda de información, etc.

En realidad es una capa que sirve de **enlace entre las vistas y los modelos, respondiendo a los mecanismos que puedan requerirse para implementar las necesidades de nuestra aplicación.** Sin embargo, su responsabilidad no es manipular directamente datos, ni mostrar ningún tipo de salida, sino servir de enlace entre los modelos y las vistas para implementar las diversas necesidades del desarrollo

Sin embargo, existe otro concepto que se usa en la terminología del MVC que es la "lógica de aplicación", que es algo que pertenece a los controladores. Por ejemplo, cuando me piden ver el resumen de datos de un usuario. Esa acción le llega al controlador, que tendrá que acceder al modelo del usuario para pedir sus datos. Luego llamará a la vista apropiada para poder mostrar esos datos del usuario. Si en el resumen del usuario queremos mostrar los artículos que ha publicado dentro de la aplicación, quizás el controlador tendrá que llamar al modelo de artículos, pedirle todos los publicados por ese usuario y con ese listado de artículos invocar a la vista correspondiente para mostrarlos. Todo ese conjunto de acciones que se realizan invocando métodos de los modelos y mandando datos a las vistas forman parte de la lógica de la aplicación.