**SPEECH DE PATRONES**

**PATRON MVC**

**POR**

**ASTRID CAROLINA**

**LUIS CARLOS MARIN CAMPOS**

**ANDRES DARIO HIGUITA PEREZ**

**MARIO DE JESUS PUENTES**

**ARQUITECTURA DE SOFTWARE**

**UNIVERSIDAD DE ANTIOQUIA**

**2020-1**



**PATRON MVC**

**¿Qué es?**

Es una propuesta de arquitectura del software utilizada para separar el código por sus distintas responsabilidades, manteniendo distintas capas que se encargan de hacer una tarea muy concreta, lo que ofrece beneficios diversos.

**¿Cuándo es necesario usarlo?**

Se usa inicialmente en sistemas donde se requiere el uso de interfaces de usuario, aunque en la práctica el mismo patrón de arquitectura se puede utilizar para distintos tipos de aplicaciones. Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Su fundamento es la separación del código en tres capas diferentes, acotadas por su responsabilidad, en lo que se llaman Modelos, Vistas y Controladores.

**¿Cuáles son sus capas y funcionamiento?**

**Modelo**

Esta es la capa donde se trabaja con los datos, entonces esta misma es la que contiene los diferentes mecanismos con los cuales se puede acceder a la información y actualizar sus estados. Si se trabaja con bases de datos, en esta es donde se realizan las funciones con las cuales se hacen consultas, búsquedas, filtros y actualizaciones hacia las tablas.

**Vista**

Esta capa se encarga de presentarle al usuario final “las pantallas” o diferentes ventanas, páginas y formularios, como resultado de una solicitud realizada por el usuario. En esta se encuentra todo los que tenga que ver con la interfaz gráfica, por lo tanto, ni la capa de modelo, ni el controlador se preocupan de cómo se ven los datos, siendo esta una responsabilidad exclusiva de la capa de vista.

Para nosotros los desarrolladores y programadores esta capa se traduciría como en el componente que se encarga del frontend de una aplicación, puede ser para una aplicación de escritorio, como también lo puede ser para páginas web (CSS, HTML, HTML5 y JavaScript).

**Controlador**

Esta capa la podemos imaginar que se encuentre posicionada entre la capa de modelo y la capa de vista, entre ellas esta capa se encuentra gestionando instrucciones que se reciben, atenderlas y procesarlas. Entonces en esta inmediación esta capa se encuentra como medio de comunicación entre estas capas solicitando los datos necesarios, para manipularlos y así obtener resultados para que por ultimo sean entregados a la vista para que puedan ser mostrados.

**¿Por qué es importante usar MVC?**

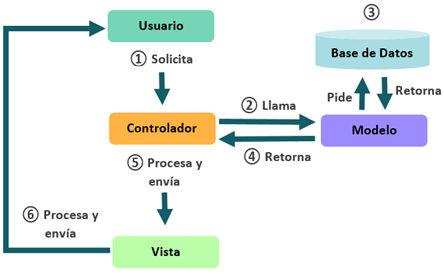
**Proceso de desarrollo más rápido:** MVC apoya el desarrollo rápido y paralelo, ya que, al utilizar el patrón, se desarrolla de una forma más eficiente debido a que una persona puede trabajar en la vista, mientras que otra puede trabajar en el controlador y así crear la lógica empresarial de la aplicación. Básicamente, el uso de este patrón nos ayuda a que nuestro proceso de desarrollo se vea beneficiado.

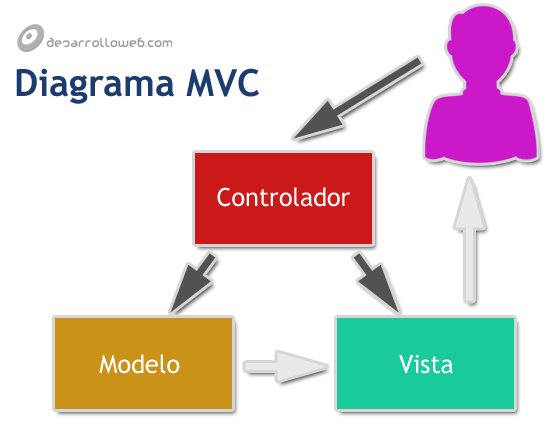
**Modularización:** Nos permite separar los componentes de nuestra aplicación dependiendo de la responsabilidad que tienen, esto significa que cuando hacemos un cambio en alguna parte de nuestro código, esto no afecte otra parte del mismo. Por ejemplo, si modificamos nuestra Base de Datos, sólo deberíamos modificar el modelo que es quién se encarga de los datos y el resto de la aplicación debería permanecer intacta. Esto respeta el principio de la responsabilidad única. Es decir, una parte de tu código no debe de saber qué es lo que hace toda la aplicación, sólo debe de tener una responsabilidad.

**Soporte para la técnica asincrónica:** es compatible con la técnica asíncrona, la cual ayuda al programador a desarrollar, y permite que la aplicación pueda tener un rendimiento superior al cargar su contenido.

**Ciclo de vida del MVC.**

* El usuario realiza una petición.
* El controlador captura la petición del usuario.
* El controlador llama al modelo.
* El modelo interactúa con la base de datos, y retorna la información al controlador.
* El controlador recibe la información y la envía a la vista.
* La vista procesa la información recibida y la entrega de una manera visualmente entendible al usuario.

****

****

**Veamos paso a paso cómo sería el flujo de trabajo característico en un esquema MVC.**

1. El usuario realiza una solicitud a nuestro sitio web. Generalmente estará desencadenada por acceder a una página de nuestro sitio. Esa solicitud le llega al controlador.
2. El controlador comunica tanto con modelos como con vistas. A los modelos les solicita datos o les manda realizar actualizaciones de los datos. A las vistas les solicita la salida correspondiente, una vez se hayan realizado las operaciones pertinentes según la lógica del negocio.
3. Para producir la salida, en ocasiones las vistas pueden solicitar más información a los modelos. En ocasiones, el controlador será el responsable de solicitar todos los datos a los modelos y de enviarlos a las vistas, haciendo de puente entre unos y otros. Sería corriente tanto una cosa como la otra, todo depende de nuestra implementación; por eso esa flecha la hemos coloreado de otro color.
4. Las vistas envían al usuario la salida. Aunque en ocasiones esa salida puede ir de vuelta al controlador y sería éste el que hace el envío al cliente, por eso he puesto la flecha en otro color.

**Ventajas de MVC**

* Las principales ventajas del uso del patrón MVC son (4):
* La separación del Modelo y la Vista, lo cual logra separar los datos, de su representación visual.
* Facilita el manejo de errores.
* Permite que el sistema sea escalable si es requerido.
* Es posible agregar múltiples representaciones de los datos.

**Desventajas de MVC**

* Las principales desventajas del uso del patrón MVC son (4):
* La cantidad de archivos que se deben mantener incrementa considerablemente.
* La curva de aprendizaje es más alta que utilizando otros modelos.
* Su separación en capas, aumenta la complejidad del sistema.

**Implementación del patrón de diseño MVC.**

Para comprender mejor cómo funcionan los componentes del patrón de diseño MVC, se realizará el siguiente ejercicio.

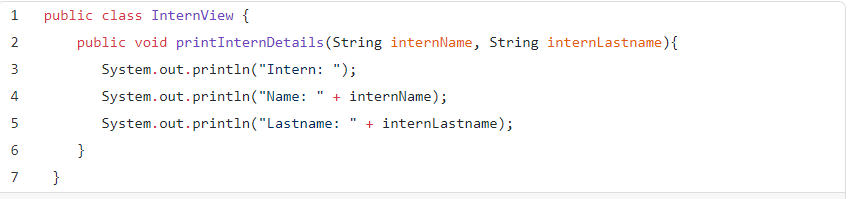
**1. Crear modelo**

Primeramente, se hará la creación del modelo, el cual se llamará “Intern” y nos ayudará a describir las características propias del objeto.



**2. Crear vista**

Se creará la vista para mostrar los resultados del ejemplo, en este caso, se utilizará imprimir en consola.



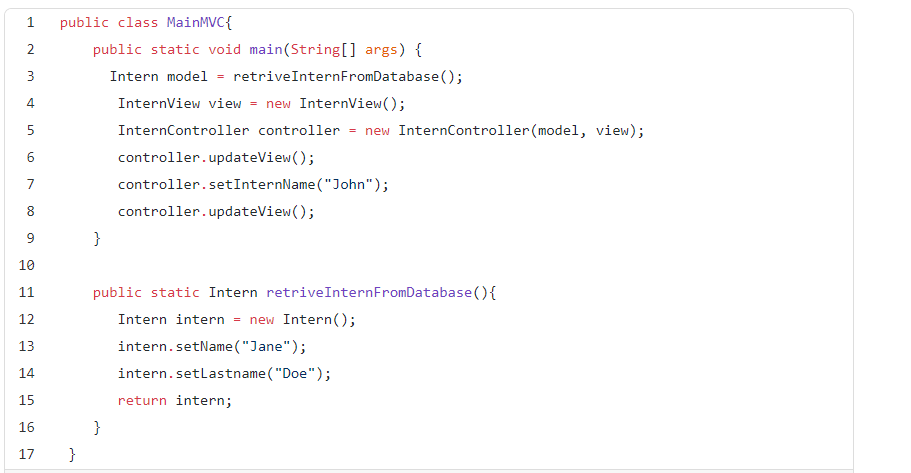
**3. Crear controlador**

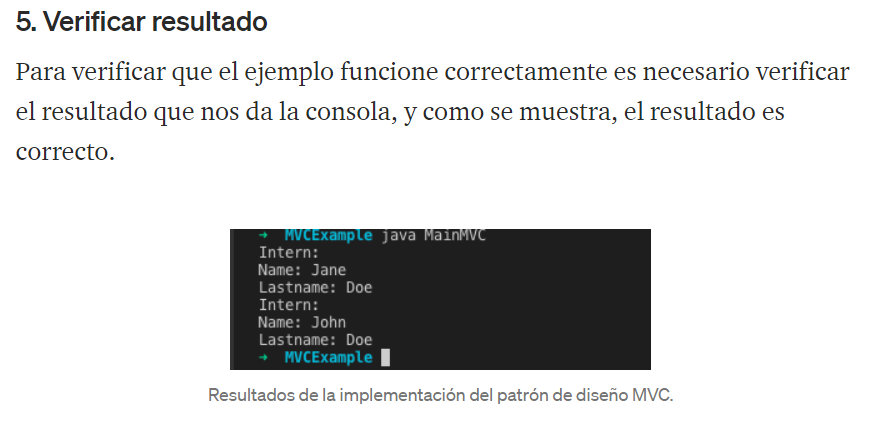
Para tener una ejecución de los eventos correctamente, se creará el controlador y será el intermediario entre el modelo y la vista.



**4. Crear clase principal**

Finalmente, se necesitará crear una clase principal para utilizar los métodos del controlador y demostrar el uso del patrón de diseño MVC.





**Video explicativo**

<https://www.youtube.com/watch?v=rRS9TQuSN0Q>

**Bibliografía**

**Enlaces informativos**

<https://blog.nearsoftjobs.com/patr%C3%B3n-de-dise%C3%B1o-mvc-2366948b5fc7>

<https://codigofacilito.com/articulos/mvc-model-view-controller-explicado>

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

<https://si.ua.es/es/documentacion/asp-net-mvc-3/1-dia/modelo-vista-controlador-mvc.html>