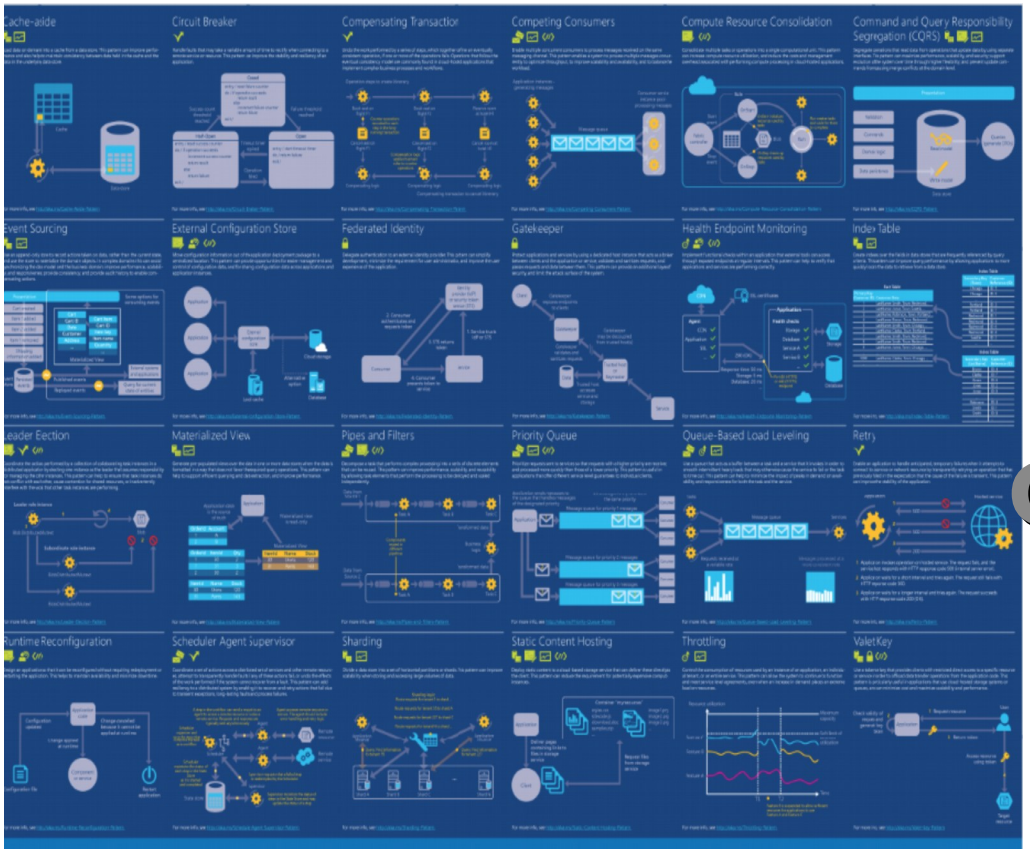
Dirección de Innovación y Desarrollo Docente
Universidad Andrés Bello

Diseño de Software



Claudio Flores Sapiain

Ingeniero Informático

Scrum Master

Ingeniero DevOps

2020

FORMAR

TRANSFORMAR

UNIVERSIDAD ANDRES BELLO

Introducción

Si la forma de solucionar ese problema se puede extraer, explicar y reutilizar en múltiples ámbitos, entonces nos encontramos ante un patrón de diseño de software.

Un patrón de diseño es una forma reutilizable de resolver un problema común.

Patrones de Diseño

1. Te ahorran tiempo:

Buscar siempre una nueva solución a los mismos problemas reduce tu eficacia como desarrollador, porque estás perdiendo mucho tiempo en el proceso. No hay que olvidar que el desarrollo de software también es una ingeniería, y que por tanto en muchas ocasiones habrá reglas comunes para solucionar problemas comunes.

Patrones de Diseño

2. Te ayudan a estar seguro de la validez de tu código:

Siempre que creamos algo nuevo nos surge la duda de si realmente estamos dando con la solución correcta, o si realmente habrá una respuesta mejor.

Los patrones de diseño son estructuras probadas por millones de desarrolladores a lo largo de muchos años, por lo que si se elige el patrón adecuado para modelar el problema adecuado, se puede estar seguro de que va a ser una de las soluciones más válidas que se pueda encontrar.

Patrones de Diseño

3. Establecen un lenguaje común:

Modelar tu código mediante patrones ayudará a explicar a otras personas, conozcan el código o no, a entender cómo se ha solucionado un problema.

Además ayudan a otros desarrolladores a comprender lo que está implementado, cómo y por qué.

Los patrones de diseño establecen un lenguaje común entre todos los miembros de un equipo.

Patrones de Diseño

¿Qué son los patrones de diseño de software?

Son formas "estandarizadas" de resolver problemas comunes de diseño en el desarrollo de software.

Las ventajas del uso de patrones son:

- Conforman un amplio catálogo de problemas y soluciones.
- Estandarizan la resolución de determinados problemas.
- Reúnen y simplifican el aprendizaje de las buenas prácticas.
- Proporcionan un vocabulario común entre desarrolladores.
- Evitan "reinventar la rueda".

MVC

Base de Datos (Modelo)

Métodos (Controlador)

Vista

MVC

Es un patrón de arquitectura de software que separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones.

Este patrón plantea la separación del problema en tres capas: la capa model, que representa la realidad; la capa controller , que conoce los métodos y atributos del modelo, recibe y realiza lo que el usuario quiere hacer; y la capa vista, que muestra un aspecto del modelo y es utilizada por la capa vista para interactuar con el usuario.

MVC

MVC es una propuesta de diseño de software que surge de la necesidad de crear soluciones más robustas, donde se potencie la facilidad de mantenimiento, reutilización del código y la separación de conceptos.

Tiene varias décadas y fue presentado incluso antes de la aparición de la Web.

No obstante, en los últimos años ha ganado mucha fuerza y seguidores gracias a la aparición de numerosos frameworks de desarrollo web que utilizan el patrón MVC como modelo para la arquitectura de las aplicaciones web.

MVC

Al escribir programas en lenguajes como PHP, cualquiera comienza mezclando tanto el código PHP con el código HTML (e incluso el Javascript) en el mismo archivo.

Esto produce lo que se denomina el "Código Espagueti". Si algún día pretendemos cambiar el modo en cómo queremos que se muestre el contenido, estamos obligados a repasar todas y cada una de las páginas que tiene nuestro proyecto.

Sería mucho más útil que el HTML estuviera separado del PHP.

MVC

Si queremos que en un equipo intervengan perfiles distintos de profesionales y trabajen de manera autónoma, como diseñadores o programadores.

Ambos tienen que tocar los mismos archivos y el diseñador se tiene necesariamente que relacionar con mucho código en un lenguaje de programación que puede no serle familiar, siendo que a éste quizás solo le interesan los bloques donde hay HTML.

De nuevo, sería mucho más fácil la separación del código.

MVC

Otro ejemplo, podemos acceder a los datos de un artículo desde la página donde se muestra éste, la página donde se listan los artículos de un manual o la página de backend donde se administran los artículos de un sitio web.

Si un día cambiamos los datos de los artículos (alteramos la tabla para añadir nuevos campos o cambiar los existentes porque las necesidades de nuestros artículos varían), estamos obligados a cambiar, página a página, todos los lugares donde se consumían datos de los artículos.

Además, si tenemos el código de acceso a BD disperso por varios lugares, es posible que estemos repitiendo las mismas funciones de acceso a esos datos y por lo tanto no estamos reutilizando código.

MVC

Modelo:

Es la capa donde se trabaja con los datos, por tanto contendrá mecanismos para acceder a la información y también para actualizar su estado.

Los datos los tendremos habitualmente en una base de datos, por lo que en los modelos tendremos todas las funciones que accederán a las tablas y harán los correspondientes selects, updates, inserts, etc.

MVC

Vistas:

Las vistas, como su nombre nos hace entender, contienen el código de nuestra aplicación que va a producir la visualización de las interfaces de usuario en nuestra aplicación en HTML.

En las vistas nada más tenemos los códigos HTML que nos permite mostrar la salida.

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.

MVC

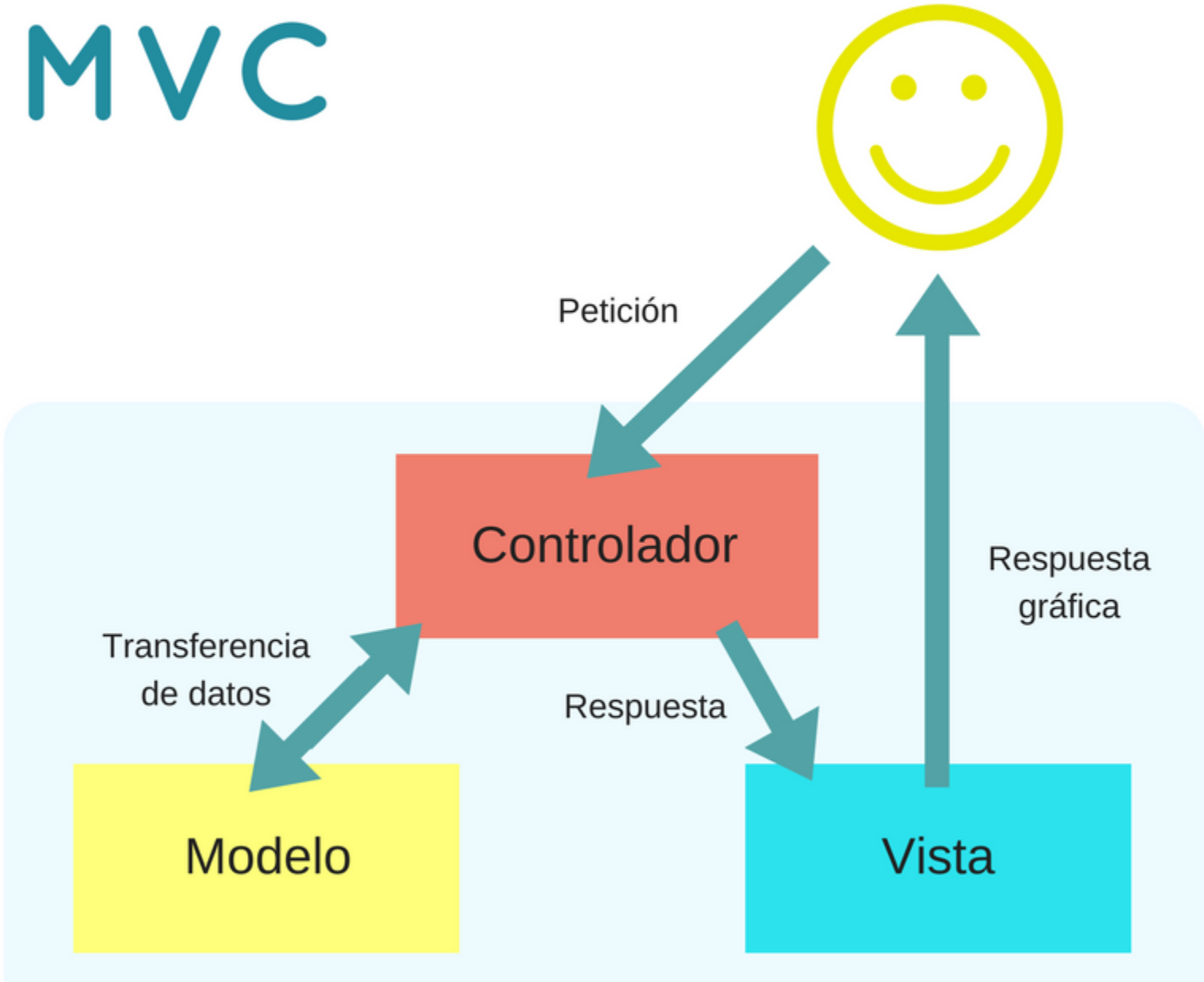
Controladores:

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.

Este componente se encarga de gestionar las instrucciones que se reciben, atenderlas y procesarlas.

Por medio de él se comunican el modelo y la vista: solicitando los datos necesarios; manipulándolos para obtener los resultados; y entregándolos a la vista para que pueda mostrarlos.

MVC


Petición
Controlador
Respuesta gráfica
Transferencia de datos
Respuesta
Modelo
Vista

Actividad Lab nro 1

Investigar IDE para desarrollo Híbrido:

Investigar qué IDEs existen en el mercado y que ayuden en el desarrollo móvil a través de framework Apache Cordova. Se pide:

- Historia.
- Características y Propiedades
- Ventajas.
- Desventajas.
- Ejemplo con un archivo HTML.
- Grupos de 3 personas y un grupo de 4.
- Presentación de mínimo 10 minutos y máximo 15 minutos.

Archivo en formato .PDF con el nombre de todos los integrantes. Formato Universidad.

- Entrega a mi correo cflores sapiain@gmail.com hasta el día 07-04-2020.

Preguntas...