



Estructura de las Aplicaciones con MVC

Introducción

- ❑ El **modelo–vista–controlador (MVC)** fue desarrollado en el Centro de Investigaciones Xerox Corporation a finales de los años setenta en California por Trygve Reenskaug.
- ❑ La arquitectura del patrón Modelo-Vista-Controlador es un paradigma de programación bien conocido para el desarrollo de aplicaciones con interfaz gráfica de usuario (GUI).
- ❑ El MVC es un patrón de software que separa los componentes de aplicación en tres niveles por sus diferentes responsabilidades.

Relación del MVC

Modelo

- Lógica de Negocio

Vista

- Interfaz de usuario

Controlador

- Lógica de Control

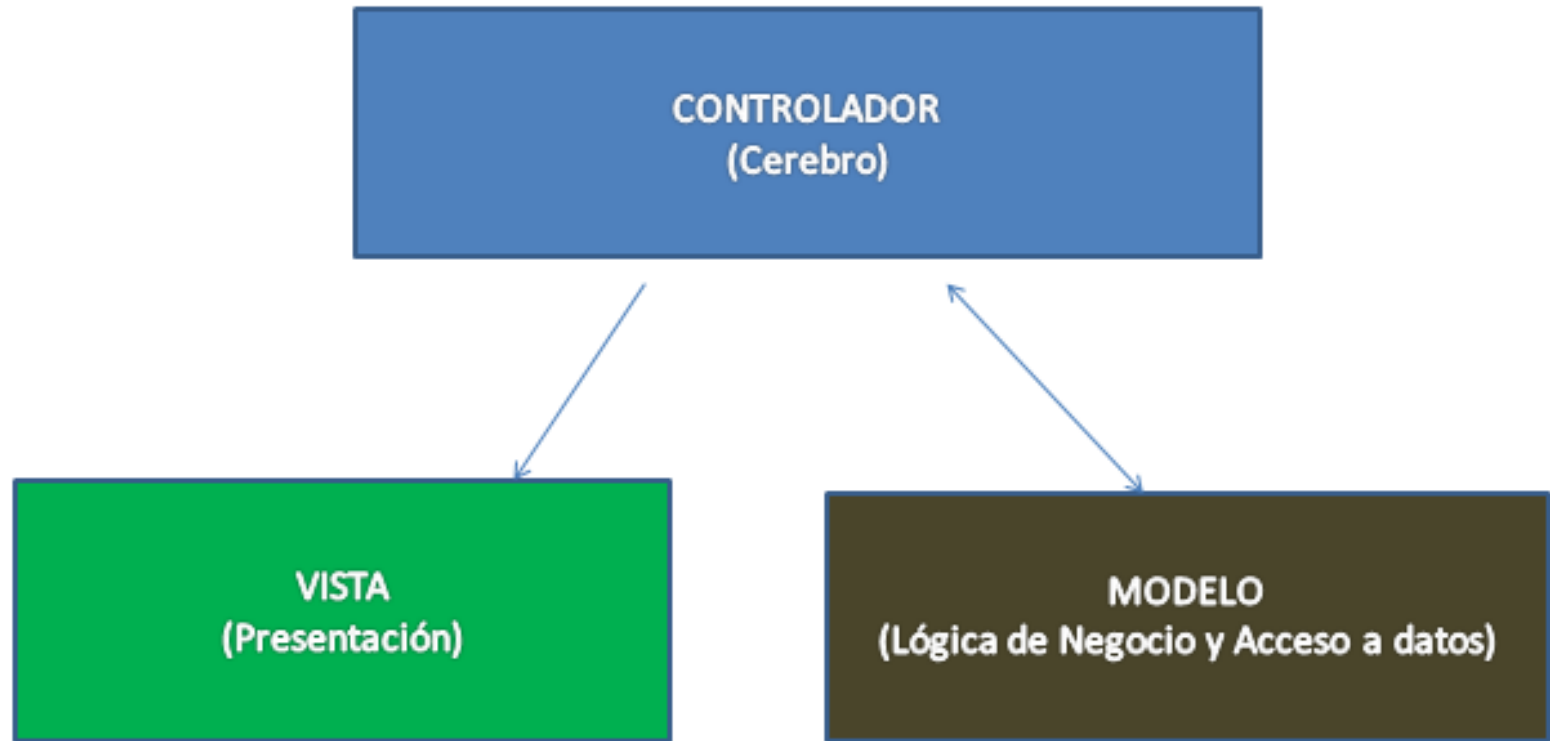
¿Por qué utilizar MVC?

- ❖ La razón es que 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.
- ❖ Surge de la necesidad de crear software más robusto con un ciclo de vida más adecuado, donde se potencie la reutilización del código y la separación de conceptos, características que buscan facilitar la tarea de desarrollo de aplicaciones y su posterior mantenimiento.

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.

Diagrama de MVC



El Modelo

- El modelo es la porción que implementa la “**Lógica del Negocio**”.
- Se le suele llamar a la parte del sistema que representa objetos y sus interacciones del mundo real.
- Son rutinas que realizan entradas de datos, consultas, generación de informes y más específicamente todo el procesamiento que se realiza detrás de la aplicación.
- Las peticiones de acceso o manipulación de información llegan al 'Modelo' a través del 'controlador', y este envía a la 'vista' aquella información que en cada momento se le solicita para que sea mostrada (típicamente a un usuario).

El Modelo (cont.)

- 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.

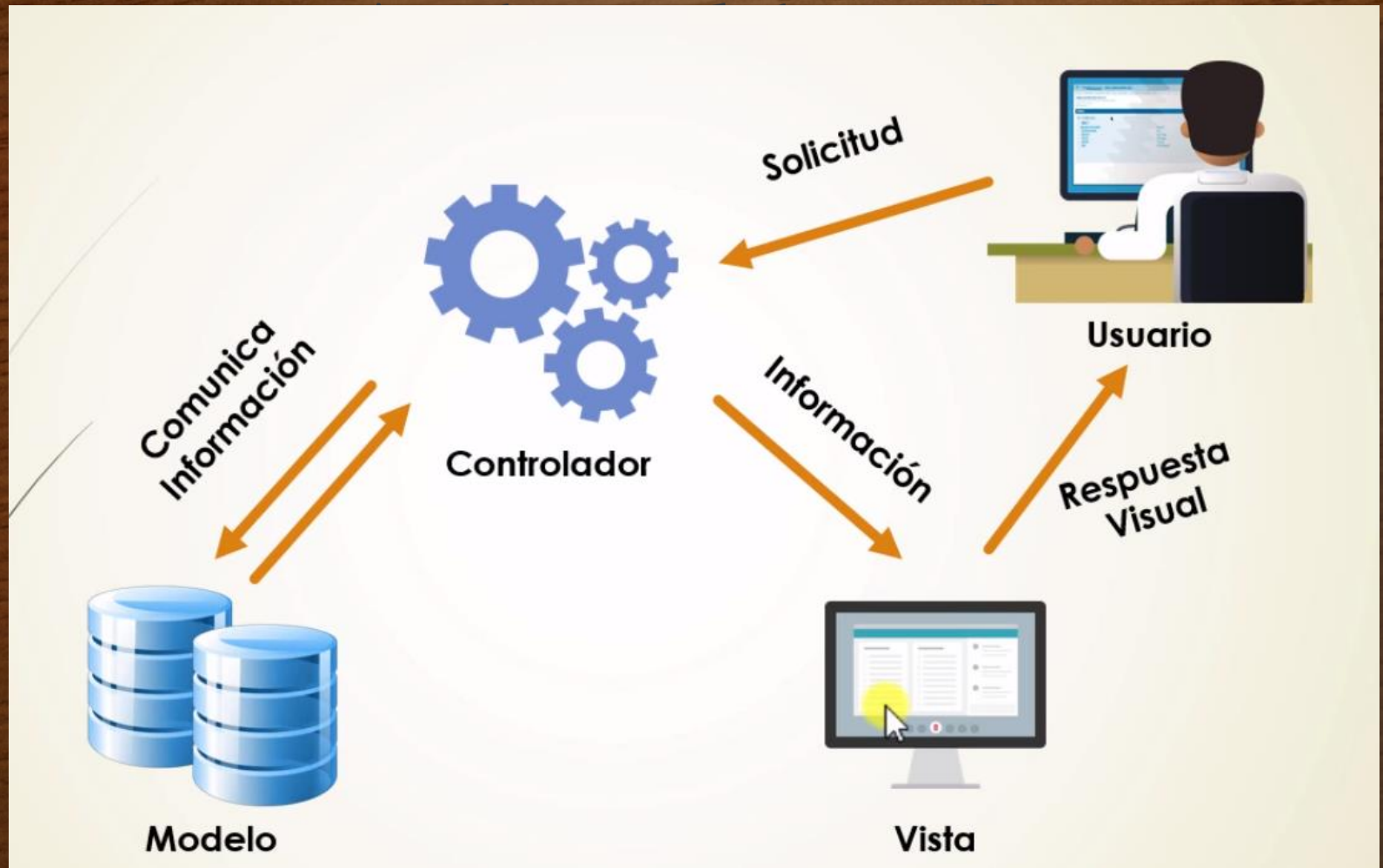


El controlador

- El controlador es el cerebro de la aplicación MVC.
- Responde a eventos (acciones del usuario) e invoca peticiones al “modelo” cuando se hace alguna solicitud sobre la información. También puede enviar comandos a su “vista” asociada si se solicita.
- Empleado como un mediador entre el medio gráfico ("View") y el modelo ("Model"), coordina las acciones que son llevadas a cabo entre ambos.
- El controlador generalmente crea instancias y utiliza métodos de esos modelos para conseguir los datos que se presentan a los usuarios, enviándolos a la vista correspondiente.

La vista

- Las vistas son las porciones de la aplicación MVC que presentan salida al usuario.
- Presenta el “modelo” (información y lógica de negocio) en un formato adecuado para interactuar (interfaz de usuario).
- Ni el modelo ni el controlador se preocupan de cómo se verán los datos, esa responsabilidad es únicamente de la vista.
- Por ejemplo: La salida más común para aplicaciones web es el HTML. Podrían ser otras como un formulario, gráficos, etc.



Flujo de control

1. El usuario realiza una acción en la interfaz.
2. El controlador trata el evento de entrada.
3. El controlador notifica al modelo la acción del usuario, lo que puede implicar un cambio del estado del modelo (si no es una mera consulta).
4. Se genera una nueva vista. La vista toma los datos del modelo.
 - El modelo no tiene conocimiento directo de la vista
5. La interfaz de usuario espera otra interacción del usuario, que comenzará otro nuevo ciclo.

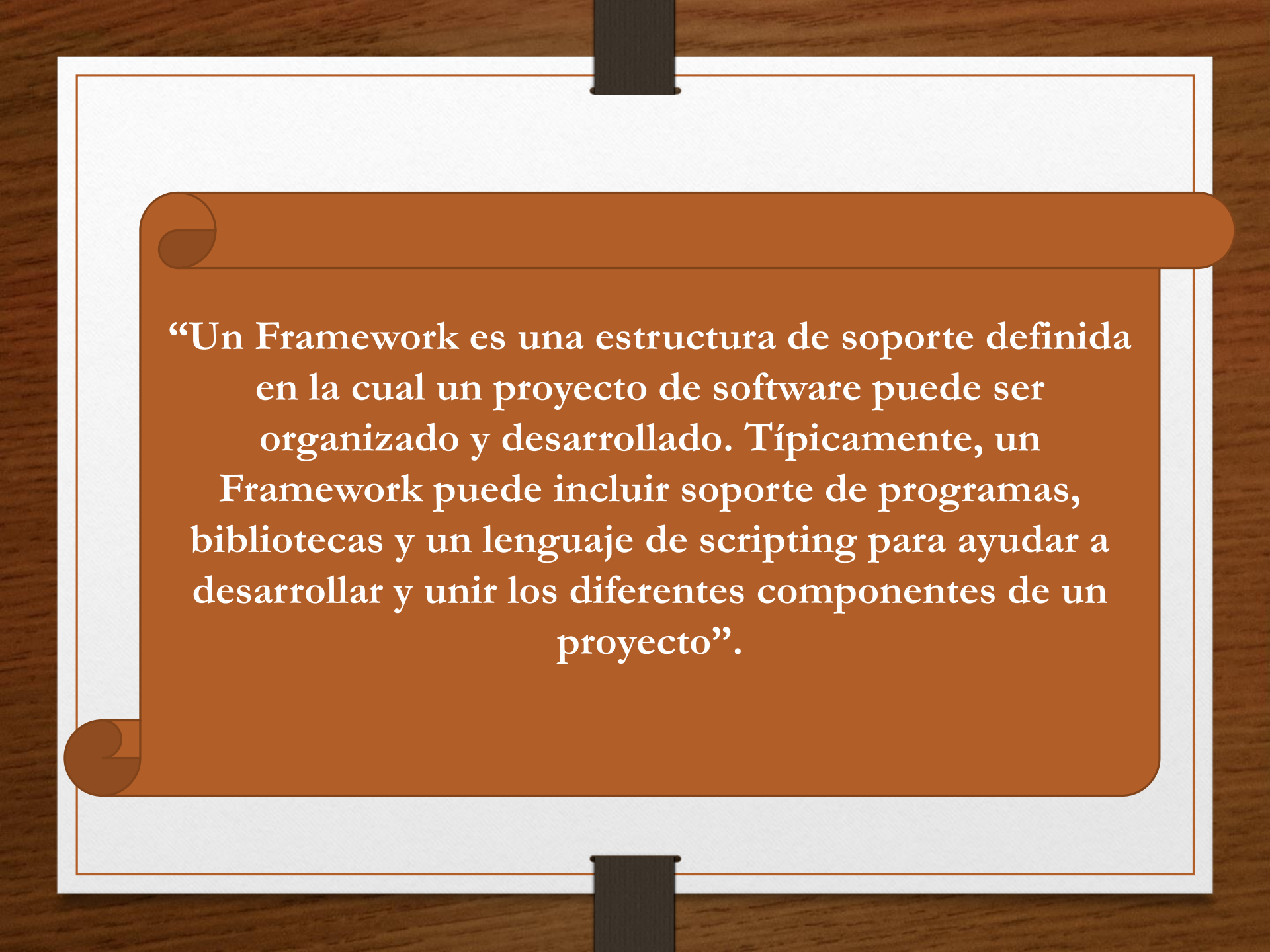
Funcionamiento en la Web

En la web, el MVC funcionaría así.

- ❑ Cuando el usuario manda una petición al navegador, digamos quiere ver un sitio específico el **controlador** responde a la solicitud, porque él es el que controla la lógica de la app, luego le pide al modelo la información del curso.
- ❑ **El modelo**, que se encarga de los datos de la app, consulta la base de datos y obtiene toda la información.

Funcionamiento en la Web

- ☐ Luego, el modelo responde al controlador con los datos que pidió.
- ☐ Finalmente, el controlador tiene los datos solicitados, se los manda a la vista, pudiendo aplicar los estilos (Hojas de estilos CSS), organizar la información y construir la página que se observa en el navegador.



“Un Framework es una estructura de soporte definida en la cual un proyecto de software puede ser organizado y desarrollado. Típicamente, un Framework puede incluir soporte de programas, bibliotecas y un lenguaje de scripting para ayudar a desarrollar y unir los diferentes componentes de un proyecto”.

Frameworks

Un framework es un diseño **re-usable** de un sistema, se puede considerar como una aplicación genérica incompleta y configurable a la que podemos añadirle las últimas piezas para construir una aplicación concreta, refiriendo a una estructura de software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación.

Frameworks

Actualmente existen muchos frameworks disponibles para el desarrollo del MVC.

Algunos ejemplos:

- ☐ Java Enterprise Edition (Java EE)
- ☐ Java Swing
- ☐ Spring
- ☐ AngularJS,
- ☐ Struts
- ☐ ASP.NET MVC Framework (Microsoft)
- ☐ Etc., etc., etc.



Ventajas del MVC

- ☐ Fácil organización del código en tres componentes diferentes.
- ☐ Crea independencia del funcionamiento.
- ☐ Facilita agregar nuevos tipos de datos según sea requerido por la aplicación ya que son independientes del funcionamiento de otras capas.
- ☐ Si trabaja con un equipo de programadores entonces les da una mayor facilidad para poder seguir el trabajo entre varios integrantes.
- ☐ Facilita el mantenimiento en caso de errores.
- ☐ Hacen que las aplicaciones sean fácilmente extensibles.
- ☐ Poder adaptarse a los frameworks de hoy en día.

Desventajas del MVC

- ❖ La separación de conceptos en capas agrega complejidad al sistema.
- ❖ La cantidad de archivos a mantener y desarrollar se incrementa considerablemente.
- ❖ La curva de aprendizaje del patrón de diseño es más alta que usando otros modelos sencillos.

EJEMPLOS DEL MVC