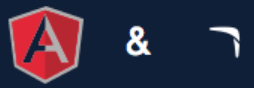
TEMA 0

Daniel Estévez  @  & 

 & 

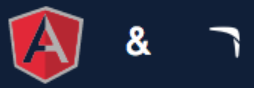
Modelo Vista Controlador (MVC)

- Patrón de diseño software para implementar interfaces de usuario.
- Divide la aplicación software en 3 partes interconectadas: El modelo, la vista y el controlador.
- La idea es separar la lógica de negocio de las forma en que se presenta la información al usuario.



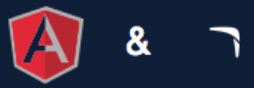
Modelo Vista Controlador (MVC)

- Controlador:
  - El Controlador envía comandos al modelo para actualizar su estado.
  - Por ejemplo al editar un documento
  - Puede también enviar comandos a la vista
  - Por ejemplo al hacer scroll en un documento



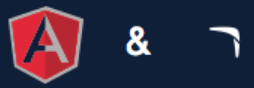
Modelo Vista Controlador (MVC)

- Modelo:
  - Representa la información del sistema, el acceso a los datos, y la lógica de negocio (Business Logic)
  - Notifica a las vistas cuando ha habido un cambio en su estado.
  - Por ejemplo si un una calculadora tras calcular el resultado de una operación

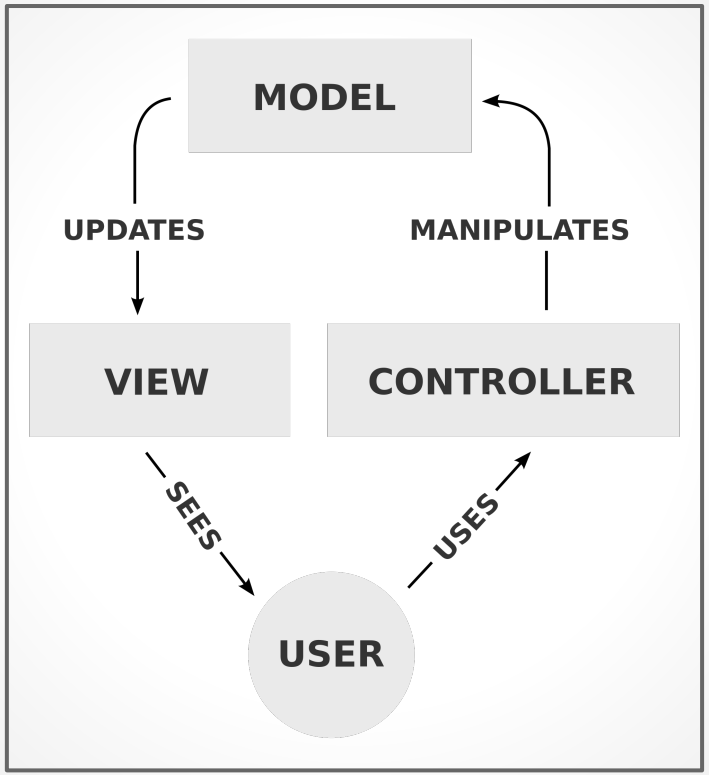


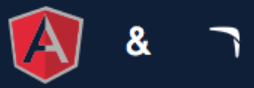
Modelo Vista Controlador (MVC)

- Vista:
  - Se encarga de presentar la información (el modelo) al usuario (UI)
  - Por ejemplo actualizar un widget del tiempo con una nube cuando el modelo cambia su estado a nublado



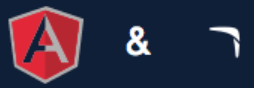
Modelo Vista Controlador (MVC)



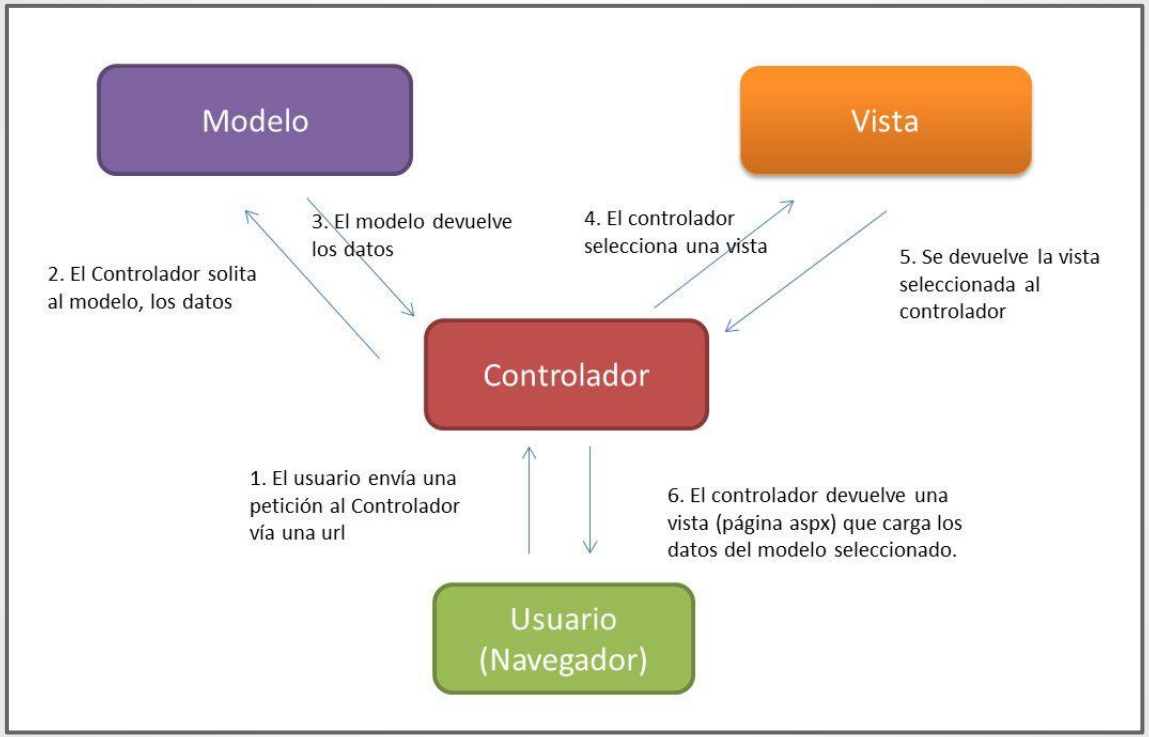


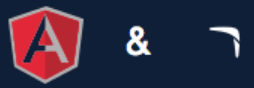
MVC en la WEB

- El concepto es el mismo, pero ahora tenemos una arquitectura físicamente distribuida. Y las interacciones cambian un poco
- La UI se ejecuta en el navegador de un usuario
- Otra máquina se encarga de servir y actualizar esta UI
- ¿Cómo dividir las funciones MVC entre cliente y servidor?



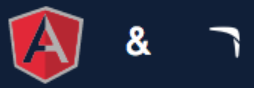
MVC en la WEB


Modelo
Vista
3. El modelo devuelve los datos
4. El controlador selecciona una vista
2. El Controlador solita al modelo, los datos
5. Se devuelve la vista seleccionada al controlador
Controlador
1. El usuario envía una petición al Controlador vía una url
6. El controlador devuelve una vista (página aspx) que carga los datos del modelo seleccionado.
Usuario (Navegador)



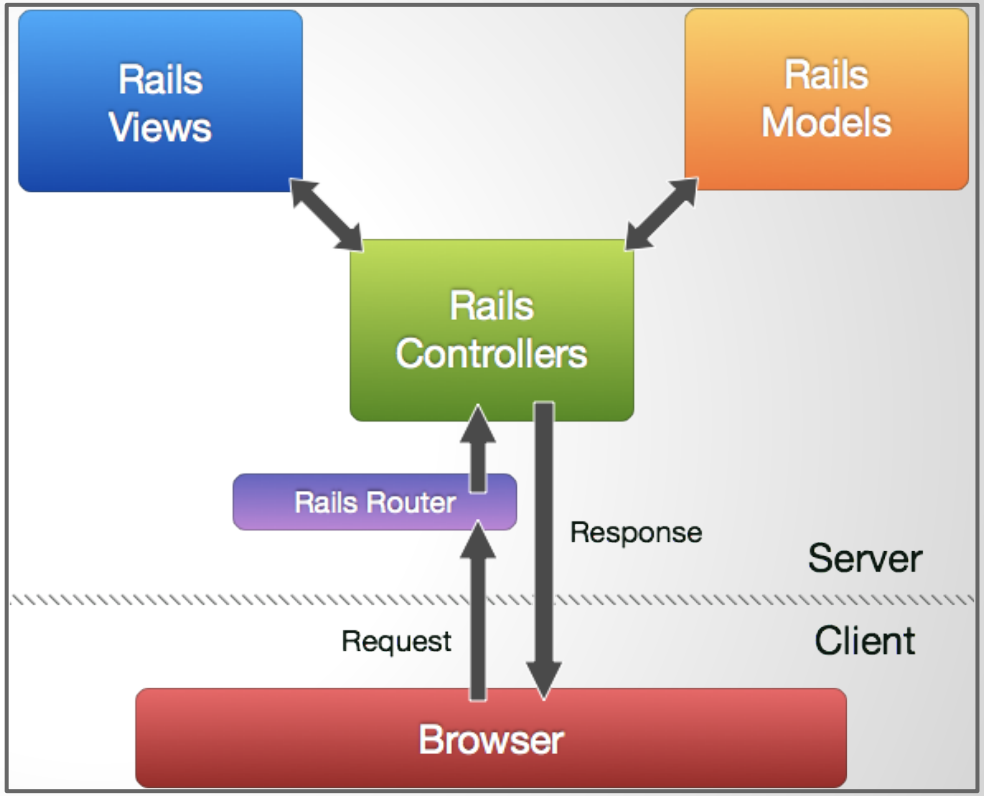
MVC en la WEB

- Los primeros FRW adquirían un enfoque de cliente super ligero con prácticamente toda la lógica del modelo, la vista y el controlador en el servidor
- El cliente envía peticiones de hipervínculos o formularios de entrada al controlador y luego recibe una página completa desde el servidor



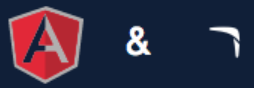
MVC en la WEB

- Arquitectura MVC en Ruby on Rails



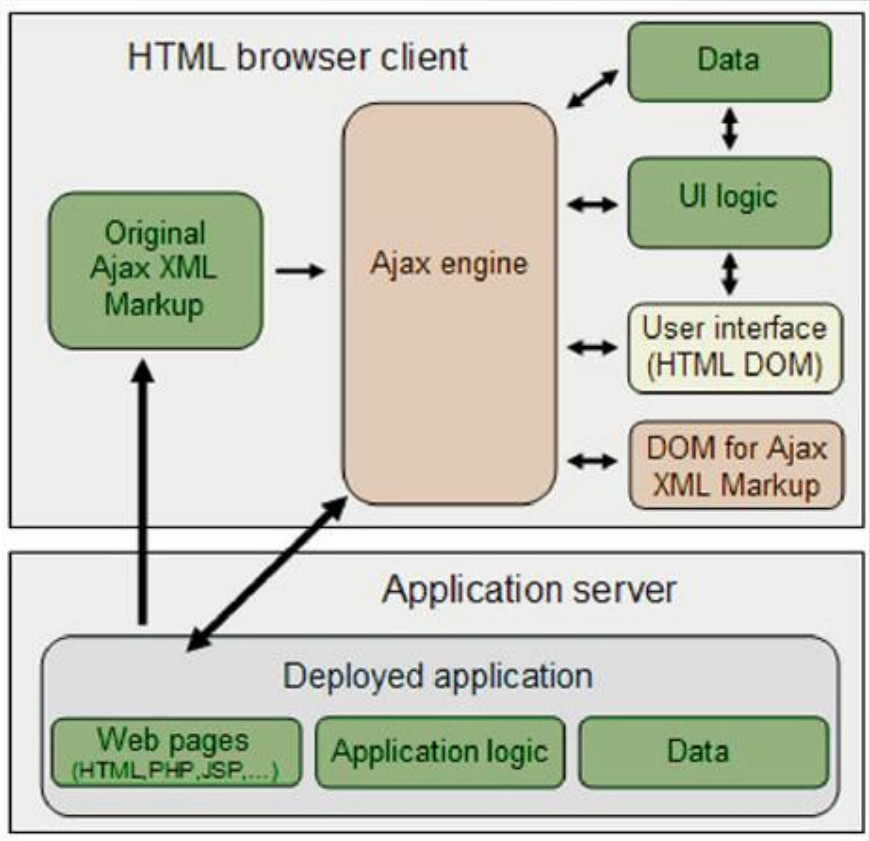
MVC en la WEB

- Pero poco a poco se ha ido delegando cada vez más responsabilidad en el cliente
- Con librerias como JQuery, podemos interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web.
- Menos llamadas al servidor. Petición de datos sin renderizar la página entera. Lógica de la vista ejecuta en el cliente.



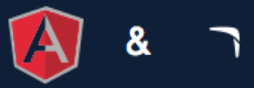
MVC en la WEB

- Arquitectura MVC con AJAX.



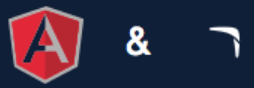
Single-page application (SPA)

- Aplicación que se ejecuta en una única página web.
- Experiencia de usuario similar a las aplicaciones de escritorio.
- Todo el HTML, CSS y JS se carga de una vez al principio cuando se carga la página



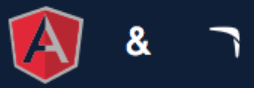
Single-page application (SPA)

- SPAs son capaces de manejar cualquier parte de la UI sin recurrir al servidor para obtener HTMLs
- Esto se consigue separando los datos (servidor) de la presentación de los datos (cliente)
- Teniendo en el cliente una capa de modelo y la vista que lee de ese modelo.



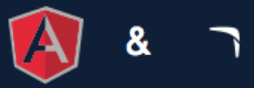
Single-page application (SPA)

- La lógica de manipulación del DOM, la lógica de negocio y las llamadas AJAX se realizan en javascript
- Convirtiendo el rol del servidor en una API pura de datos o de servicios web

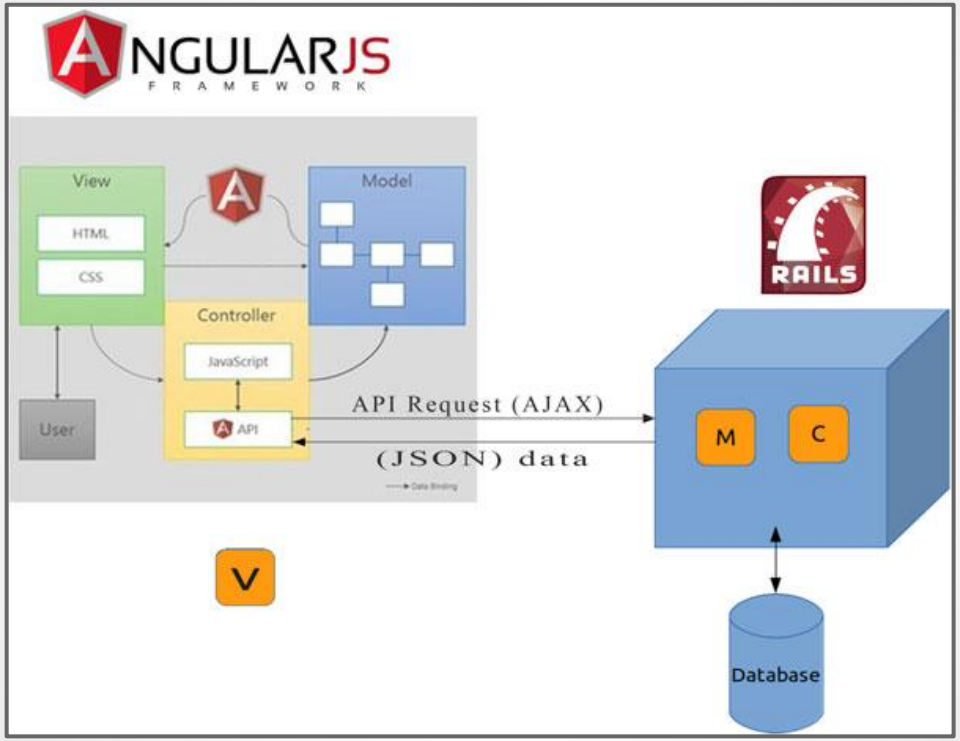


Single-page application (SPA)

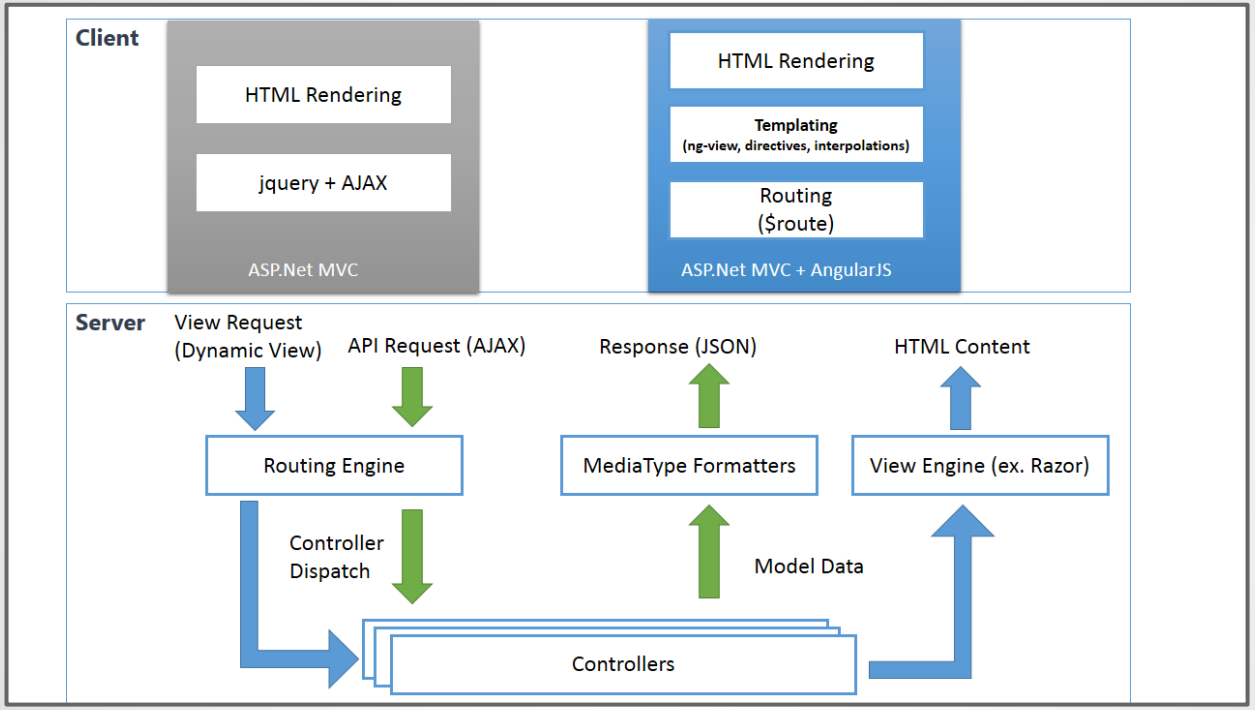
- Básicamente hemos llevado el patrón de diseño MVC al cliente.
- Incluso el enrutamiento
- FRWs como AngularJS, Ember.js, y ReactJS han adoptado los principios SPA.

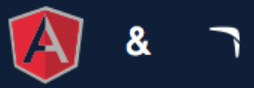


Single-page application (SPA)



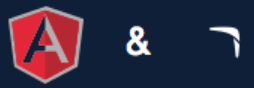
Single-page application (SPA)





Qué es AngularJS

- Angular es un FRW estructural para la creación de páginas web dinámicas.
- Permite extender la sintaxis HTML (Directivas)



Qué es AngularJS

- Es una solución completa para el cliente:
  - Se encarga del de todo el código de manipulación del DOM y AJAX
  - Aporta todo lo necesario para crear apps CRUD: Data-binding, templating, validación de formularios, routing, reutilización de componentes, inyección de dependencia etc
  - Lo necesario para testeo: Unit tests, e2e tests, mocking, etc