

Ingeniería Web

Arquitecturas de una Aplicacion Web

Visión General

- **Arquitecturas de una aplicación web**
 - Introducción
 - Arquitecturas de Software
 - Desarrollo de arquitecturas
 - Patrones y frameworks
 - Especificaciones de las Aplicaciones web
 - Arquitecturas de una Aplicación Web
 - Tipos de Arquitecturas
 - MVC, Client-Server, N-Layer, JSP Model 1 and 2, Apache Struts, JSF
- **Resumen**

INTRODUCCION

Que es entiendes por arquitectura?

Arquitecturas de Software

“Una Arquitectura se define [...] como la organización fundamental de un sistema, embebida en sus componentes, sus relaciones entre si y con el entorno, y los principios que gobiernan su diseño y evolucion.”(IEEE Architecture Working Group, P1471, 1999)

- **Las Arquitecturas describen la estructura:**
 - Componentes de software de sistemas, sus interfaces y relaciones
 - Aspectos dinamicos y estáticos
 - Diseño de sistemas software
- **Las Arquitecturas conectan fases del desarrollo de software**
 - Requerimientos mapeados iterativamente a componentes y sus relaciones

Arquitecturas de Software

“La Arquitectura es el conjunto de decisiones de diseño [...] que mantienen sus implementadores y encargados de mantenimiento, donde el ejercicio de la creatividad no es necesaria.”(Desmond F. D’Souza and Alan C. Wills, 1999)

- **Las arquitecturas describen diferentes puntos de vista**
 - Vista conceptual: entidades del dominio de la aplicacion y sus relaciones
 - Vista de procesos: ejecucion de sistemas, concurrencia, sincronizacion
 - Vista de implementación: artefactos de software (subsistemas, componentes, codigo fuente)
 - Vista en tiempo de ejecución: componentes en tiempo de ejecucion y su comunicacion
- **Las arquitecturas hacen los sistemas comprensibles y controlables**
 - Estructurandolos acorde a diferentes puntos de vista
 - Permite la comunicacion entre diferentes stakeholders

Desarrollando Arquitecturas

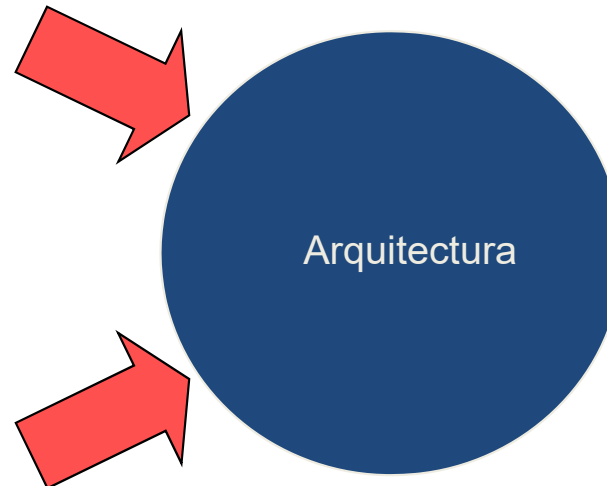
Influencias sobre las Arquitecturas

Requerimientos Funcionales

- Clientes
- Usuarios
- Otros Stakeholders

Requerimientos No Funcionales

- Rendimiento
- Escalabilidad
- Reusabilidad
- Otros?



Desarrollando Arquitecturas

Influencias sobre las Arquitecturas



Desarrollando Arquitecturas

- **Recordar, los requerimientos estan sujetos a cambios**
 - Cambio organizacional y entorno
 - Requerimientos iniciales ambiguos
- **Por lo tanto, los enfoques iterativos son los medios sugeridos para el desarrollo**
 - Pro: Ayuda a mitigar los riesgos de diseño
 - Precaución: No garantiza una buena arquitectura

Patrones y Frameworks

- **Patrones describen problemas de diseño recurrentes**
- **Hay 3 tipos de patrones**
 - Patrones arquitecturales (e.g. MVC)
 - Patrones de diseño (e.g. Publisher-Subscriber)
 - Lenguaje (e.g. Counted-Pointer in C++)
- **Son una guía, la implementación debe estar conectada con el problema específico**
- **Los Patrones deben estar “integrados” entre sí!**

Patrones y Frameworks

- **Frameworks: otra opcion para reutilizar arquitecturas existentes**
 - Algo que proporciona un marco para ser completado!
- **Reutilizacion de objetos de software existente que solo necesitan configurarse apropiadamente**
- **Atado a una tecnologia específica**
 - Requiere entrenamiento
 - Alto costo de interrupcion
 - Nivel de personalizacion no siempre aceptable

Especificaciones de la Web apps

- **Altas demandas de calidad**
 - Seguridad
 - Extensibilidad
 - Adaptabilidad
 - Estabilidad
 - Rendimiento
- **Amplio rango de soluciones tecnicas que pueden ser integradas.**
 - Dificil evaluar demandas de calidad (varios componentes)
 - Dificil de resolver (donde está el problema?)

Especificaciones de la Web apps

- **Falta de homogeneidad e inmadurez de las soluciones tecnicas**
 - Rápidos ciclos de vida de productos
 - Muchos componentes: open source – calidad?
 - Falta de estándares
- **Requerimientos Globales**
 - Multi linguas
 - Adaptacion Cultural?
 - E.g. Google for Korea

ARQUITECTURAS DE UNA APLICACION WEB

Tipos de Arquitecturas

- **Aspecto de Capas**

- “Separación de intereses”
- A cuantos usuarios concurrentes serviremos?
- Compartir necesidades entre multiples aplicaciones? (e.g., seguridad)

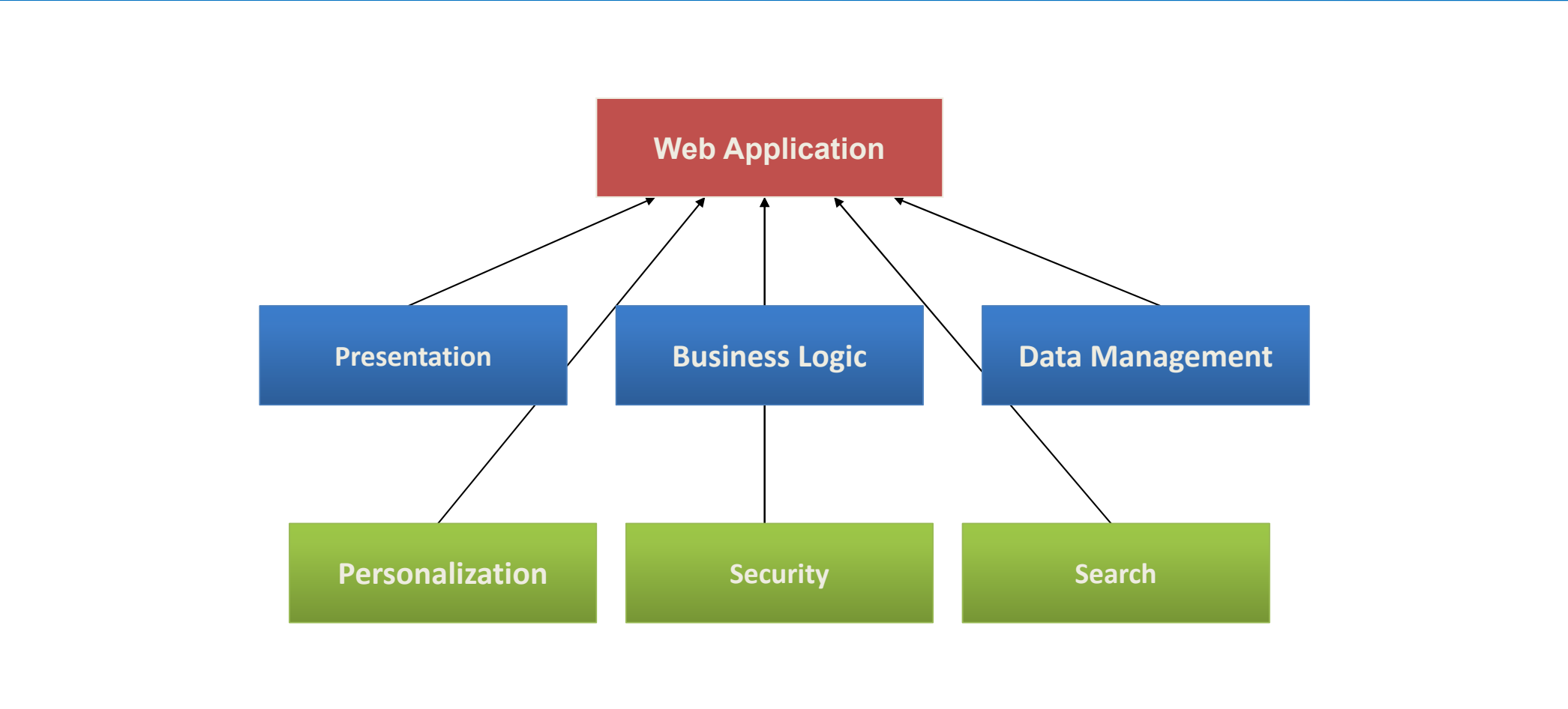
- **Aspecto de datos**

- Que tipo(s) de datos seran entregados?
 - Structurados vs. no estructurados
 - Bajo demanda vs. tiempo real
- Cuales son los requerimientos de ancho de banda?
 - Tamaño y naturaleza de los datos
 - Una vez mas, intereses del público

Tipos de Arquitecturas

- **Arquitectura de una Plataforma Web (WPA)**
 - Plataforma = Infraestructura
 - Hardware
 - Modulos y configuraciones de Software
 - Seleccin de una plataforma software (e.g., J2EE, python, javascript, .NET)
- **Arquitectura de una Aplicación Web (WAA)**
 - Vista conceptual de como los procesos clave del negocio y necesidades son separadas e implementadas
 - Muchas veces de dominio específico
 - Mayor complejidad requiere mayor modularidad

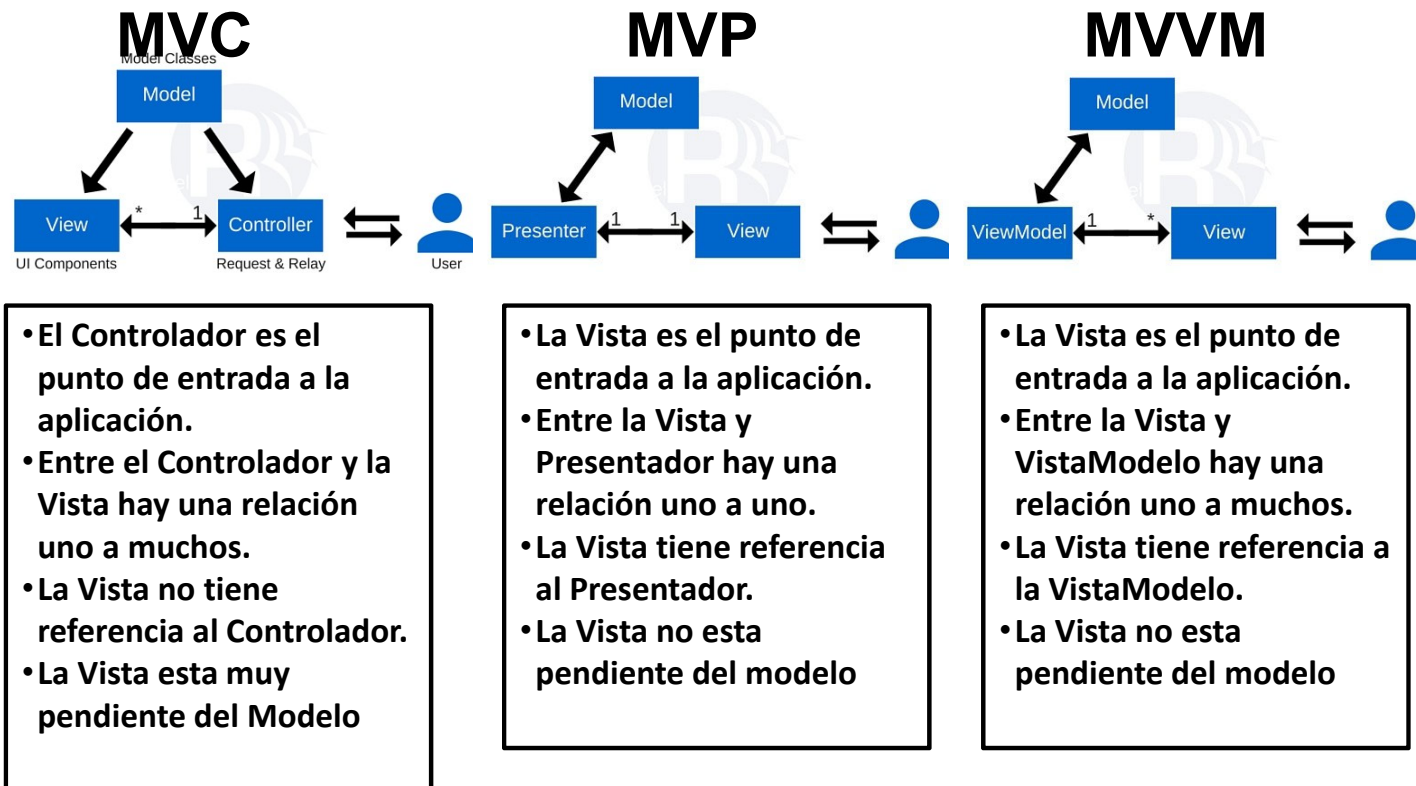
Ejemplo de una WAA



Arquitectura (Plataforma) web generica

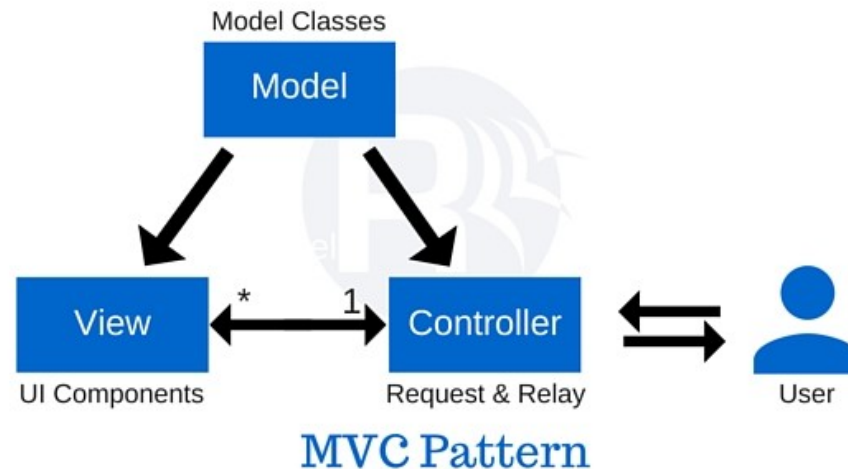
- **La plataforma esta basado en**
 - TCP/IP
 - HTTP
 - HTML
- **Es esencialmente una arquitectura Client/Server**
 - En terminos de patrones una de las mas sencillas
- **Pero aun así las cosas pueden ser complejas...**
 - Componentes en la red (firewall, proxy, balanceo de carga)
 - Componentes en la intranet (Web server, application server, data base, legacy systems, web services)

Patrones de diseño



Patrón de diseño MVC

- Patrón de diseño de software, de la década de 1970.
- Separa el modelo de dominio y la lógica del controlador de la interfaz de usuario (vista).
- El mantenimiento y las pruebas de la aplicación se vuelven simples y fáciles. Facilita el desarrollo.
- Divide una aplicación en tres aspectos principales:

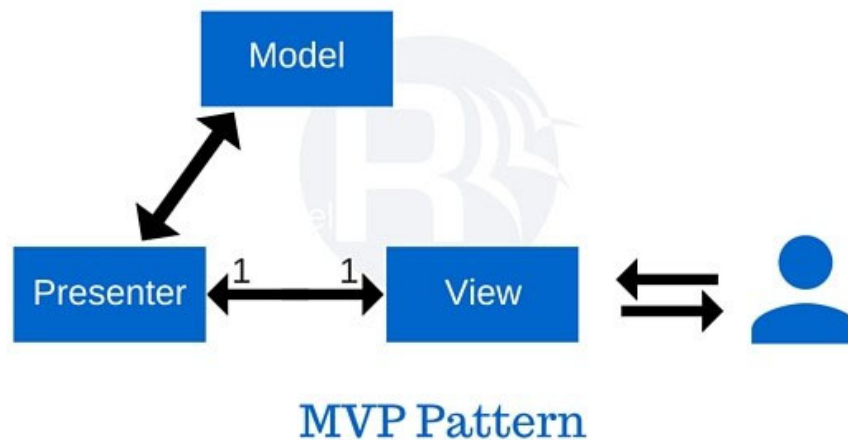


Patrón de diseño MVC

- **Modelo:** Representa una colección de clases que explica la lógica del negocio, es decir, el modelo del negocio y modelo de datos (operaciones de acceso a datos). Define las reglas del negocio para los datos, es decir como los datos pueden ser alterados y manipulados
- **Vista:** Representa los componentes de la interfaz de usuario como CSS, JavaScript, HTML, etc. La vista muestra los datos que se reciben del controlador como resultado. Esto también cambia el modelo (s) en la interfaz de usuario.
- **Controlador:** La responsabilidad del controlador es procesar las solicitudes entrantes. Obtiene la entrada de los usuarios a través de la Vista, luego procesa los datos del usuario a través del Modelo y devuelve los resultados a la Vista. Normalmente actúa como un mediador entre la Vista y el Modelo.

Patrón de diseño MVP

- El patrón MVP es similar al patrón MVC, en donde el controlador es reemplazado por el presentador.
- Este patron divide una aplicacion en tres principales aspectos: Model, View, and Presenter.

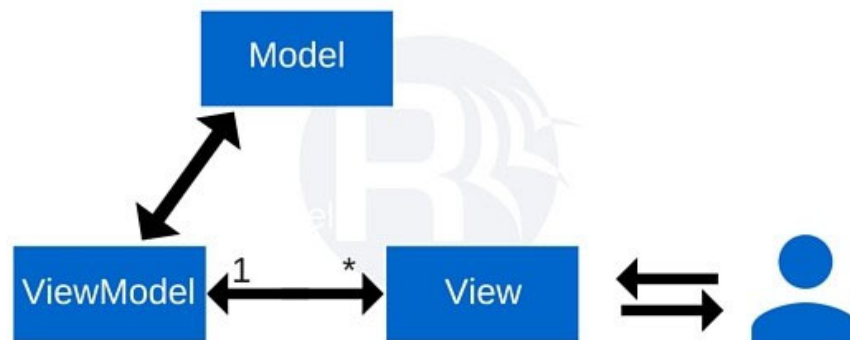


Patrón de diseño MVP

- **Modelo:** Representa una colección de clases que explica el modelo de negocio y el modelo de datos. Define las reglas del negocio para los datos, es decir como los datos pueden ser alterados y manipulados.
- **Vista:** Representa los componentes de la interfaz de usuario como CSS, JavaScript, HTML, etc. La vista muestra los datos que se reciben del controlador como resultado. Esto también cambia el modelo (s) en la interfaz de usuario.
- **Presentador:** Es responsable de direccionar todos los eventos de la interfaz de usuario en nombre de la vista. Recibe las peticiones de los usuarios a través de la Vista, luego procesa los datos del usuario a través del Modelo que pasa los resultados a la Vista. La Vista y el Presentador están completamente separados, a diferencia de la Vista y Controlador que se comunican entre sí mediante una interfaz. El presentador tampoco maneja el tráfico de solicitudes entrantes como el controlador.

Patrón de diseño MVVM

- Soporta enlace de datos bidireccional entre la Vista y la VistaModelo.
- Esto permite la propagación automática de cambios, dentro del estado de la VistaModelo a la Vista.
- Generalmente la VistaModelo utiliza el patrón Observador para informar cambios de la VistaModelo al Modelo.



MVVM Pattern

Patrón de diseño MVVM

- **Modelo:** Representa una colección de clases que explica el modelo de negocio y el modelo de datos. Define las reglas del negocio para los datos, es decir como los datos pueden ser alterados y manipulados.
- **Vista:** Representa los componentes de la interfaz de usuario como CSS, JavaScript, HTML, etc. La vista muestra los datos que se reciben del controlador como resultado. Esto también cambia el modelo (s) en la interfaz de usuario.
- **VistaModelo:** Es responsable de mostrar métodos, comandos y otras funciones que ayudan a mantener el estado de la Vista, manipulando el Modelo como resultado de acciones en la Vista y desencadenar los eventos en la Vista en si misma.

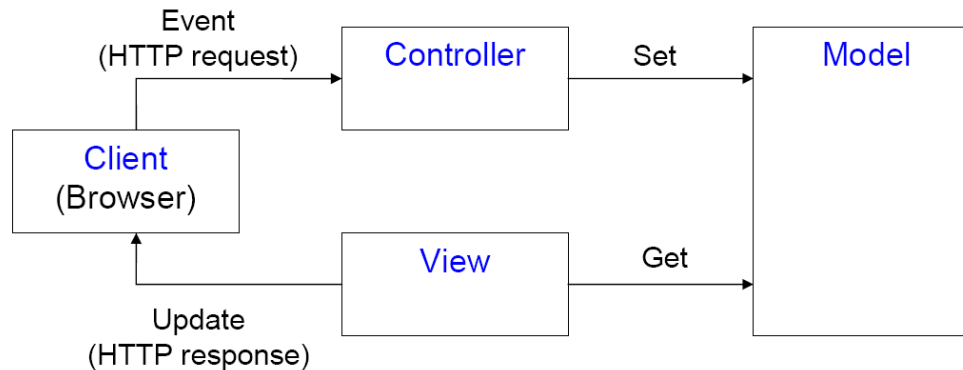
Arquitecturas Web: Especificaciones

- **Limitaciones tecnologicas**
 - HTTP
- **Amplia variedad de soluciones tecnicas**
 - Servidores de aplicación, proxies, firewalls, legacy applications
 - Dificultad para verificar la calidad
 - e.g., rendimiento dependen de varios componentes, como base de datos, ancho de banda de la red, procesador, memoria, codigo, ...
 - Dificultad para mejorar la calidad
 - e.g., el rendimiento del código no puede cambiar de forma sustancial el rendimiento general
- **Soluciones tecnicas no homogeneas e inmaduras**
 - Cortos ciclos de vida del producto
 - Falta de estandares impiden la integracion de componentes de diferentes fabricantes
 - Muchas soluciones son open source: continuidad de desarrollo, extensibilidad, ...
- **Acceso global a aplicaciones web**
 - Internacionalizacion, diferencias culturales

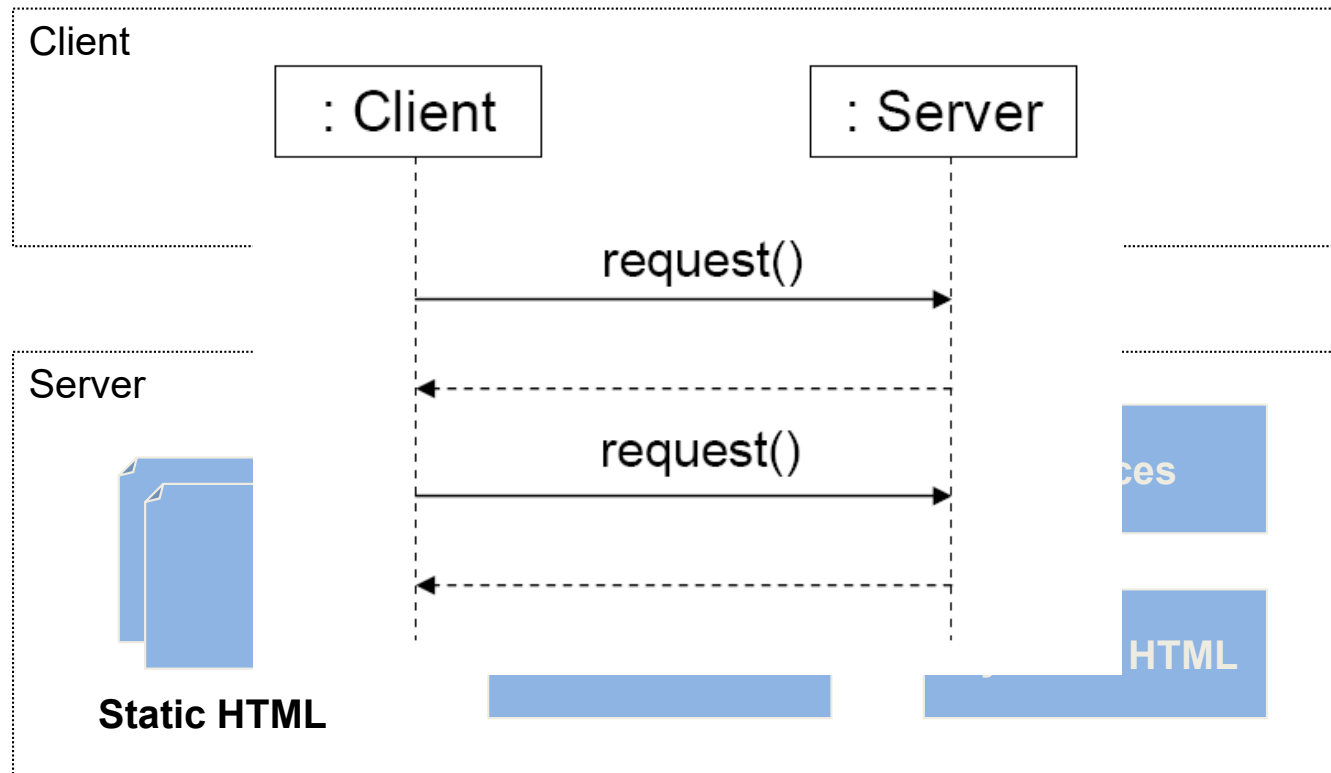
Modelo-Vista-Controlador 2 (MVC 2)

- **Adaptacion del MVC para la Web**

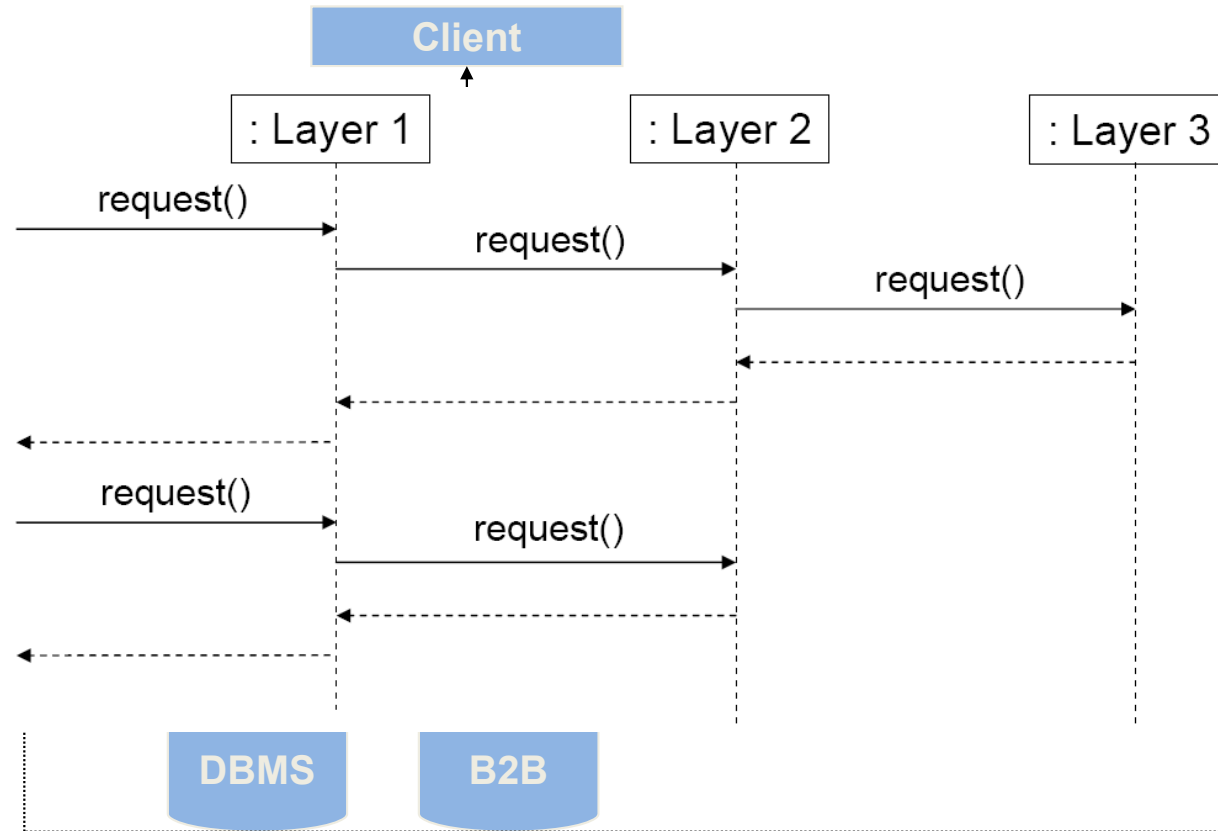
- Conexion sin estado entre el cliente y servidor
- Notificacion de cambios en la vista
- Volver a consultar el servidor para descubrir modificaciones del estado de la aplicación



Client/Server (2-Layer)



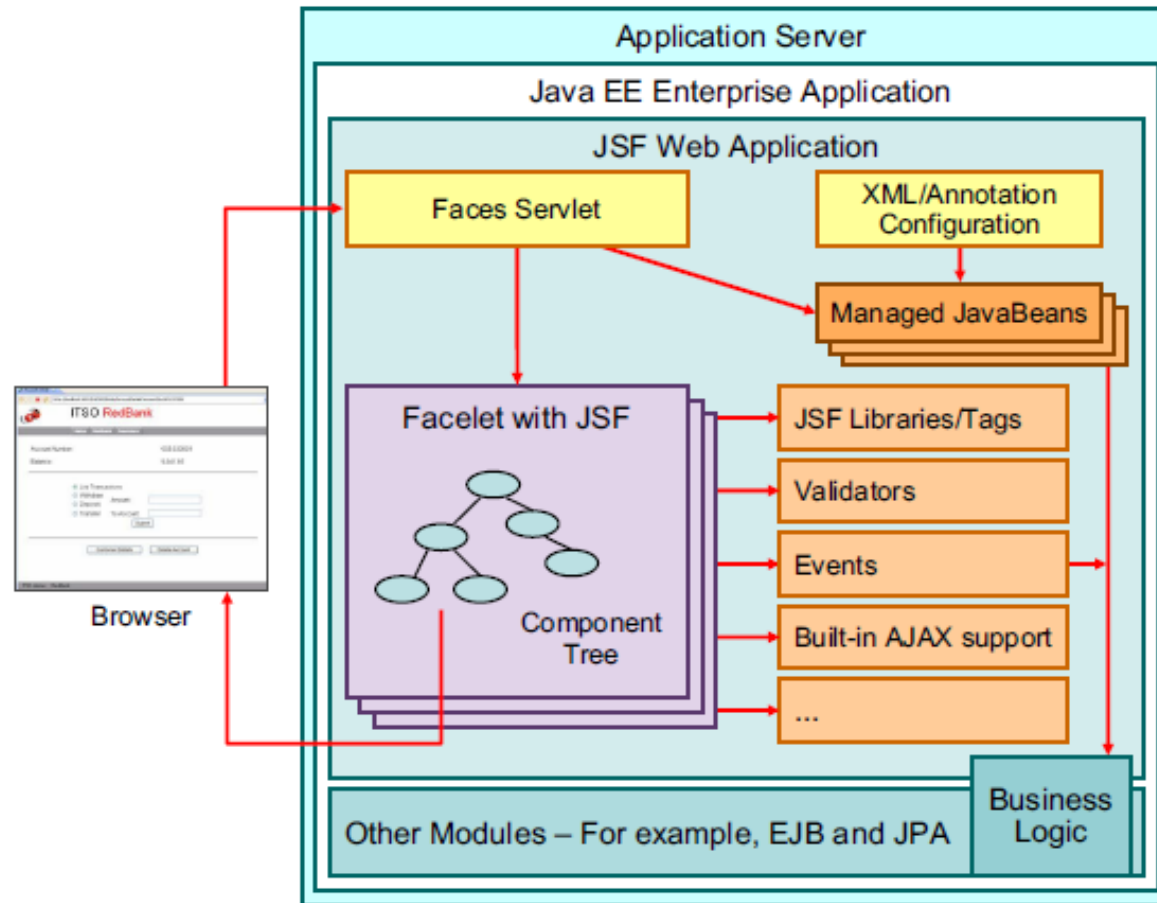
Arquitecturas de N-Capas



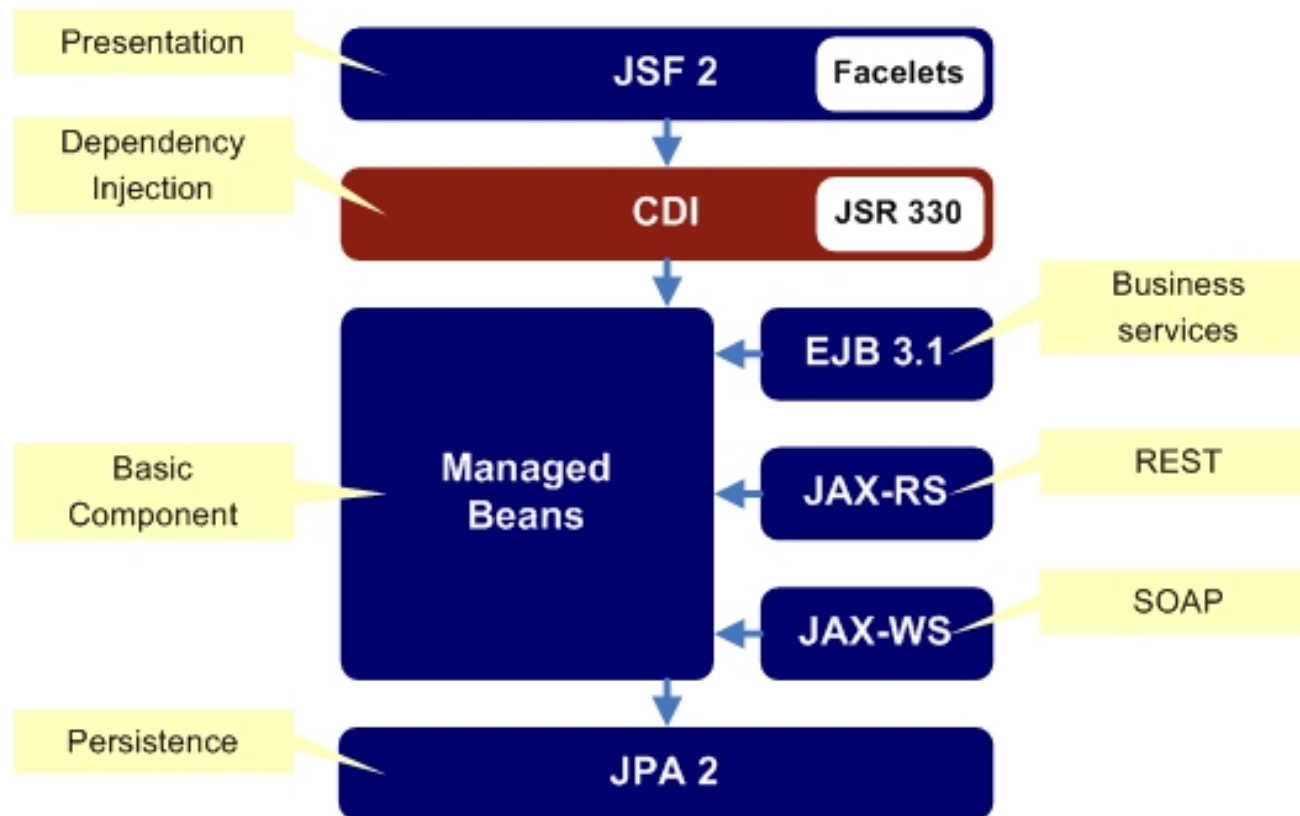
Porque una Arquitectura de N-Capas?

- **La separacion de servicios en la capa de negocios promueve la reutilizacion entre aplicaciones**
 - Pobre acoplamiento – cambios reducen el impacto en todo el sistema.
 - Mas mantenibilidad (en terminos de codigo)
 - Mas extensible (modular)
- **Ventajas y desventajas**
 - Complejidad innecesaria
 - Mas puntos de falla

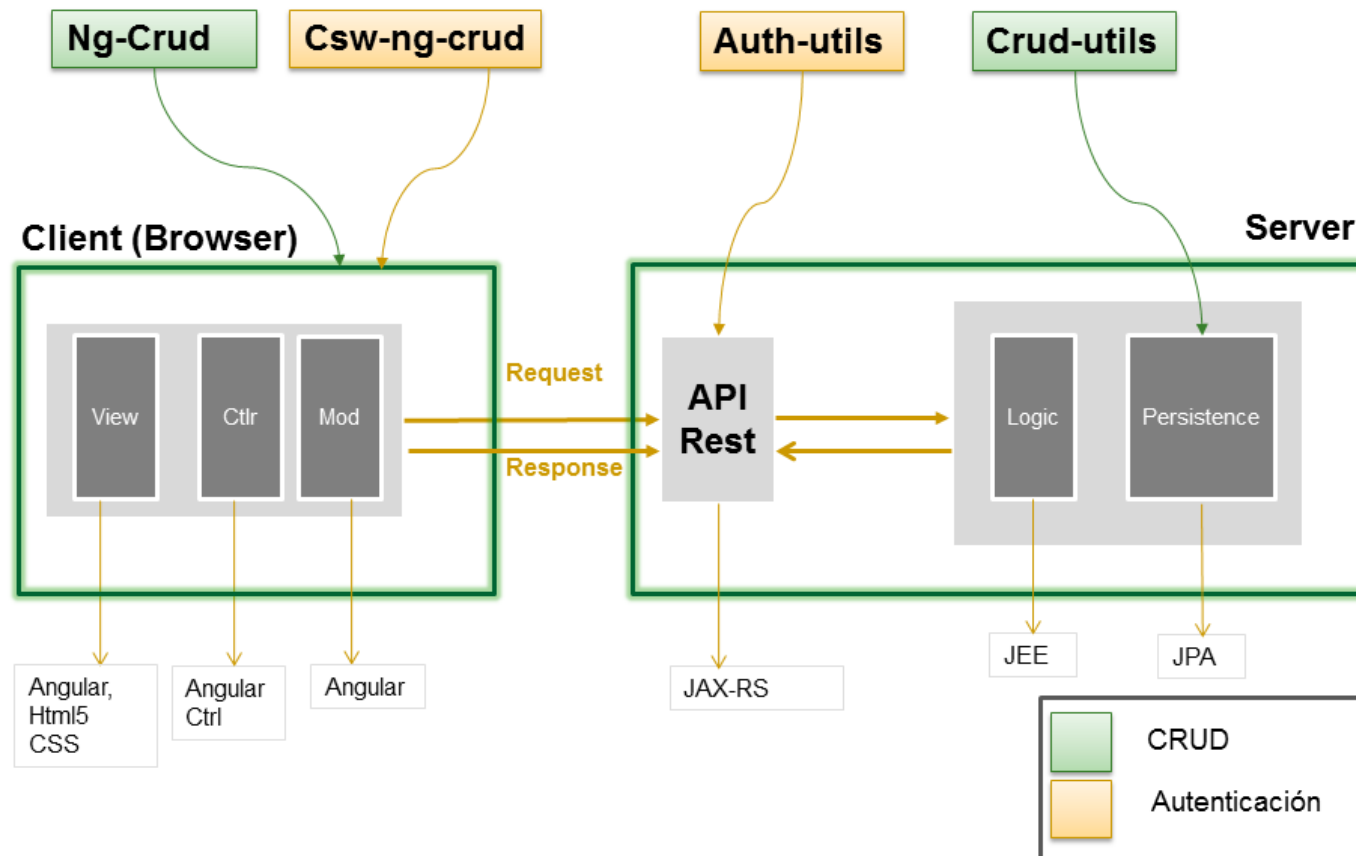
Arquitectura JSF



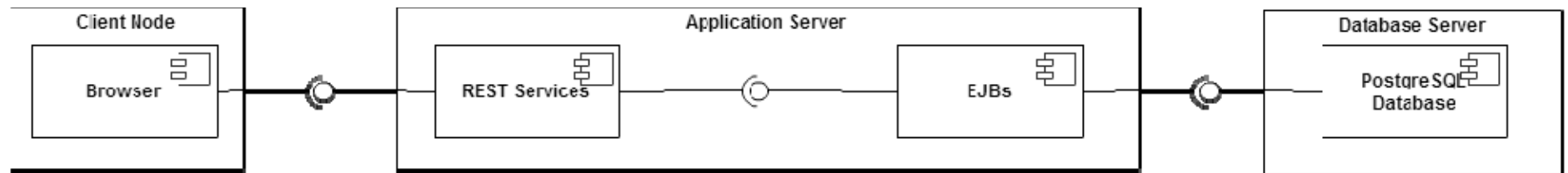
Middleware



Arquitectura: componentes y tecnología



Arquitectura – vista de despliegue



Tanto el Frontend como el Backend se despliegan en el mismo servidor de aplicaciones, el cual permitirá el acceso a la aplicación a través del protocolo HTTP, soportado por todos los navegadores web

FINALMENTE

Cosas a tener en cuenta (resumen)

- **Buen diseño de la arquitectura es crucial**
- **Puedes aprovechar patrones y frameworks**
 - Ambos tienen ventajas y desventajas
- **El diseño web esta restringido a la “infraestructura”**
- **MVC es un patron usado comunmente, en angular se esta usando MVVM**

Bibliografia

- **Lectura obligatoria**

- Kappel, G., Proll, B. Reich, S. & Retschitzegger, W. (2006). *Web Engineering*, Wiley & Sons. **4th Chapter**

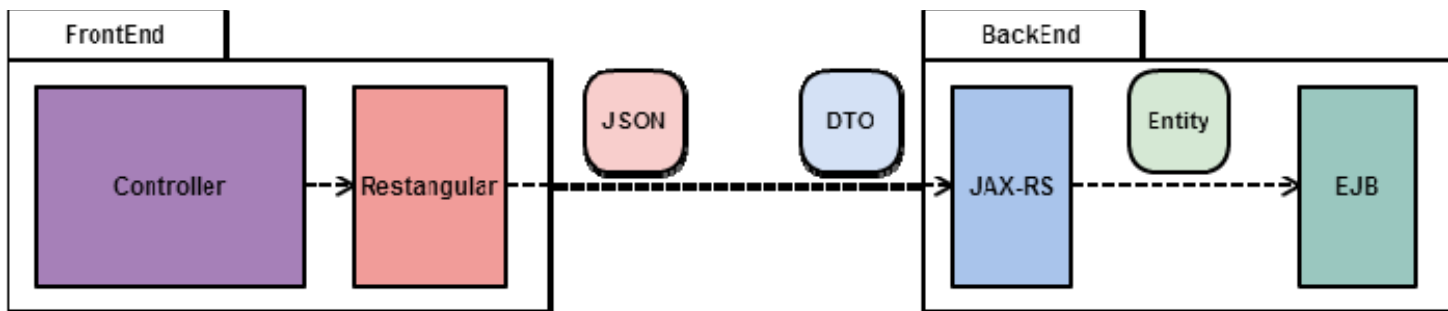
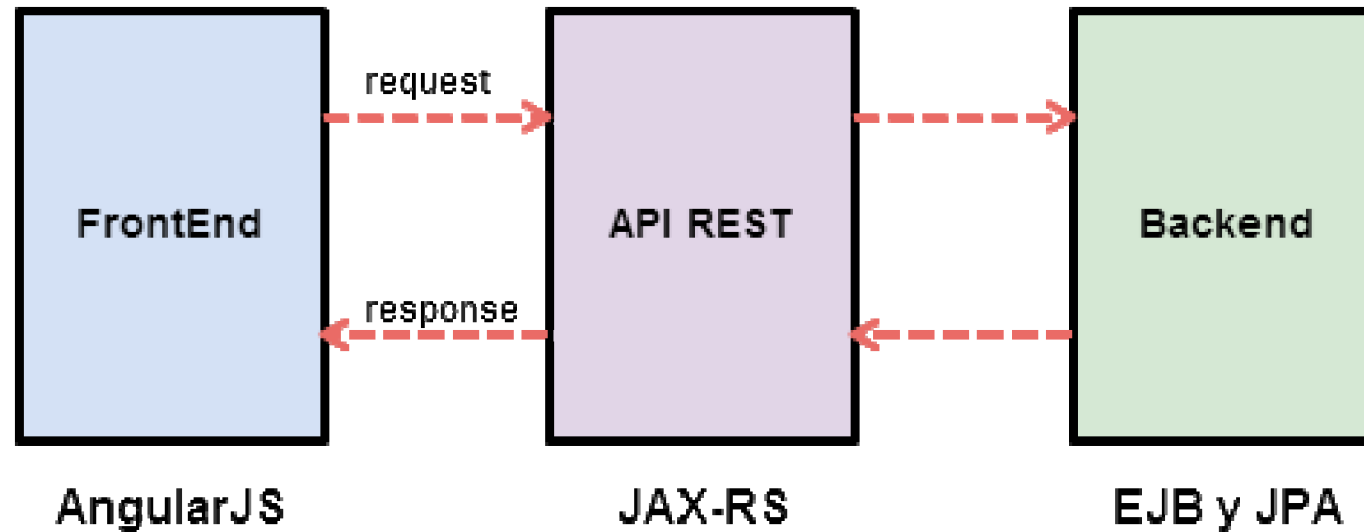
- **Web links**

- Model-View-Controller pattern <http://en.wikipedia.org/wiki/Model-View-Controller>

Preguntas?



Arquitectura: vista funcional (Frontend – Backend)



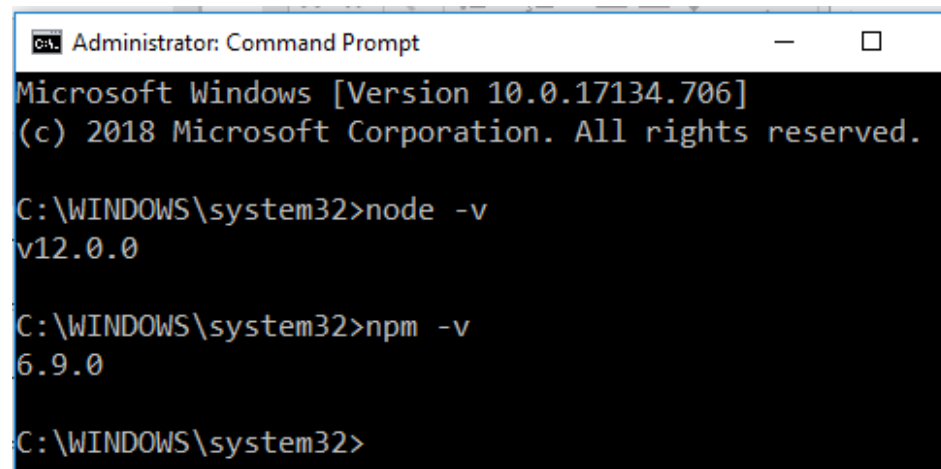
Preparando el entorno de desarrollo

- **Instalar:**

- NodeJS <https://nodejs.org/es/>

Verificar la instalación: `node -v`

Verificar la instalación npm: `npm -v`



```
Administrator: Command Prompt
Microsoft Windows [Version 10.0.17134.706]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>node -v
v12.0.0

C:\WINDOWS\system32>npm -v
6.9.0

C:\WINDOWS\system32>
```

- Chrome <https://www.google.com/chrome/>

Preparando el entorno de desarrollo

- **Instalar:**

- TypeScript <https://www.typescriptlang.org/>

```
Administrator: Command Prompt
C:\WINDOWS\system32>npm install -g typescript
C:\Users\Edwin\AppData\Roaming\npm\tsserver -> C:\Users\Edwin\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
C:\Users\Edwin\AppData\Roaming\npm\tsc -> C:\Users\Edwin\AppData\Roaming\npm\node_modules\typescript\bin\tsc+ typescript@3.4.5
updated 1 package in 7.789s

C:\WINDOWS\system32>tsc --version
Version 3.4.5

C:\WINDOWS\system32>
```

- Cliente angular <https://cli.angular.io/>

```
Administrator: Command Prompt
C:\WINDOWS\system32>npm install -g @angular/cli
C:\Users\Edwin\AppData\Roaming\npm\ng -> C:\Users\Edwin\AppData\Roaming\npm\node_modules\@angular\cli\bin\ng
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.8 (node_modules\@angular\cli\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.8: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"})

+ @angular/cli@7.3.8
added 111 packages from 92 contributors, removed 62 packages and updated 43 packages in 121.452s
```

```
Administrator: Command Prompt
C:\WINDOWS\system32>ng version

Angular CLI 7.3.8
Node: 12.0.0
OS: win32 x64
Angular:
...

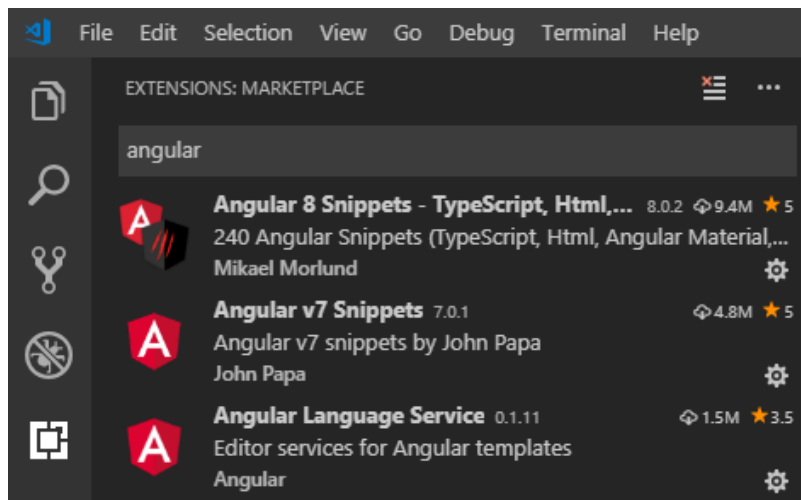
Package      Version
-----
@angular-devkit/architect 0.13.8
@angular-devkit/core       7.3.8
@angular-devkit/schematics 7.3.8
@schematics/angular        7.3.8
@schematics/update         0.13.8
rxjs                    6.3.3
typescript                3.2.4
```


Preparando el entorno de desarrollo

- Git <https://git-scm.com/>

```
MINGW64:/c/Users/Edwin  
  
Edwin@DESKTOP-5BS09F2 MINGW64 ~  
$ git config --global user.name "Edwin Valencia"  
  
Edwin@DESKTOP-5BS09F2 MINGW64 ~  
$ git config --global user.email evalencia@unc.edu.pe
```

- Instalar Visual Studio Code <https://code.visualstudio.com/> y los siguientes plugins:



- Angular 2 TypeScript Emmet
- Angular 8 Snippets - TypeScript, Html, Angular Material..
- Angular Language Service
- Angular v7 Snippets
- angular2-inline
- Bootstrap 4, Font awesome ..
- HTML Boilerplate
- IntelliSense for CSS class names in HTML
- css-grid-snippets
- JavaScript (ES6) code snippets
- JS-CSS-HTML Formatter
- Material Icon Theme
- Prettier - Code formatter
- Terminal
- TypeScript Hero
- TypeScript Importer
- Live Server
- open in browser
- file-icons