

Creando aplicaciones web con AngularJS NodeJS

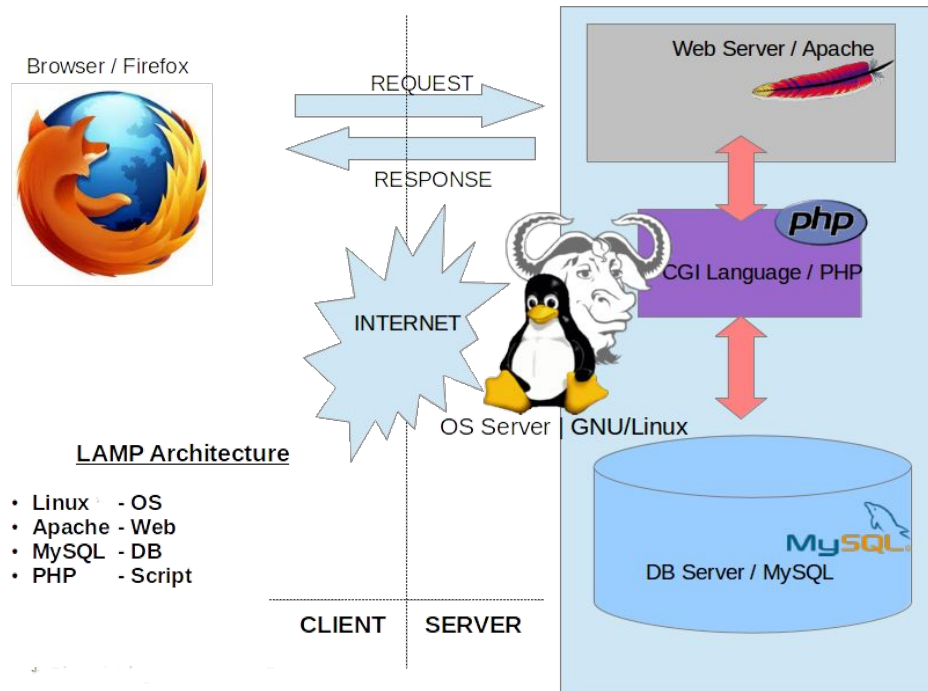
Juan Camilo Ibarra
John Alejandro Triana

Agenda

- Teoría
 - Pila de desarrollo web AMP
 - Desarrollo web con AngularJS, Express y NodeJS
- Práctica
 - Creación del proyecto
 - Arquitectura de una aplicación web
 - Aplicación: lista de tareas

Desarrollo web - AMP

- LAMP (Linux, Apache HTTP Server, MySQL, PHP)
- WAMP (Windows)
- DAMP (MacOSX)
- XAMP (Multiplataforma)
- XAMPP (XAMP+Perl)



Desarrollo web - AMP

Alternativas de MySQL

- PostgreSQL
- MariaDB (community-developed fork of MySQL)
- MongoDB (open-source NoSQL)



Desarrollo web - AMP

PHP and alternatives

- Perl
- Python
- Javascript



Desarrollo web con AngularJS, Express y NodeJS

MEAN



- DBMS No SQL
- Agil
- Escalable



- Framework para aplicaciones web
- Liviano



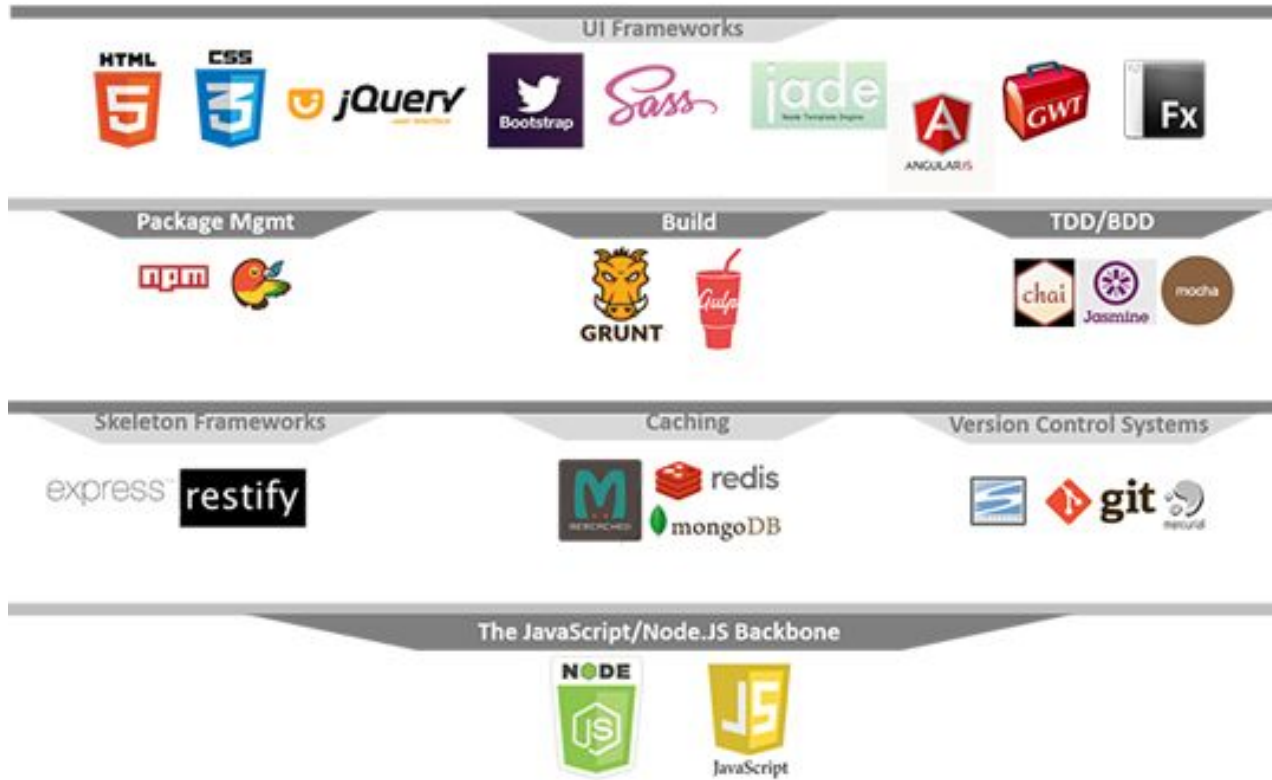
- Extension de HTML
- Ambiente de desarrollo ágil
- Implementa MVC



- Plataforma de desarrollo
- Ambiente de desarrollo ágil, escalable
- Asincrono

Desarrollo web con AngularJS, Express y NodeJS

MEAN Stack Expertise



Práctica: Manos a la obra!

1. Ingrese al GitHub de Imagine https://github.com/imaginebog/TallerNodeJS_AngularJS
2. En este github está el código completo de la solución del taller así como una wiki para desarrollarlo paso a paso.

Creación del proyecto

1. Crear una carpeta para el proyecto. El nombre no debe tener mayúsculas
2. Abrir la consola
3. Usar npm para inicializar nuestra aplicación

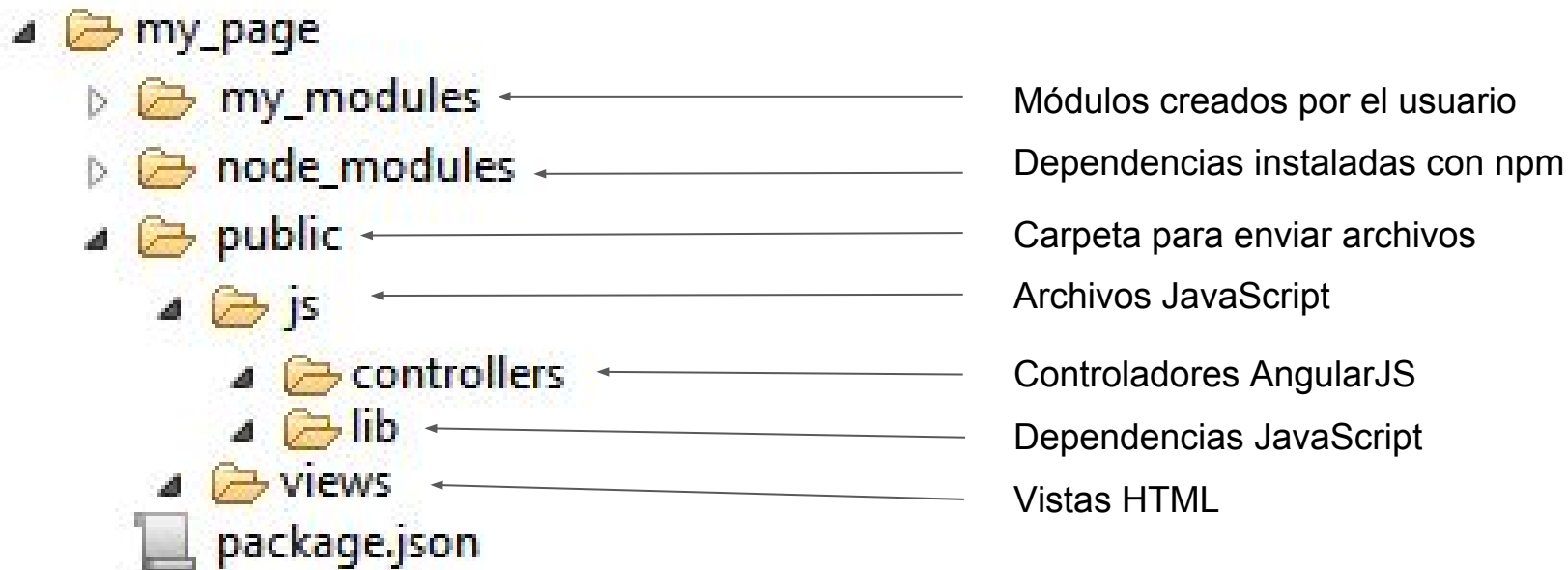
```
> npm init
```

4. Usar npm para instalar las dependencias Express y Socket.io

```
> npm install express socket.io --save
```

Este proceso genera un archivo package.json y la carpeta node_modules

5. Crear la siguiente jerarquía de carpetas



6. Crear el archivo index.js

Este archivo es el punto de entrada de la aplicación. En este vamos a especificar el servidor así como su forma de comunicación con el cliente. Para ello se debe:

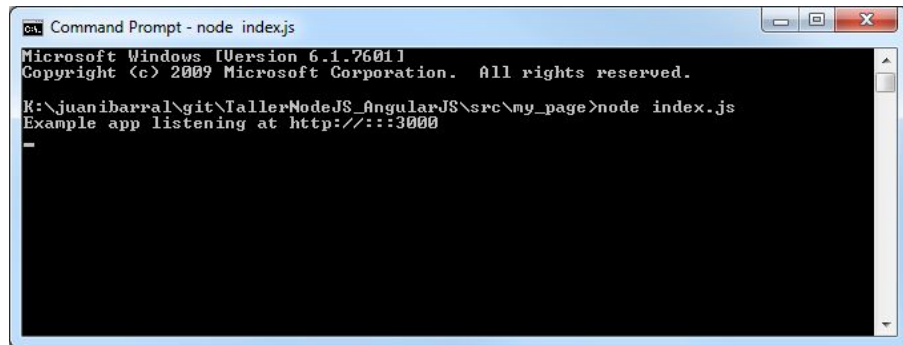
- a. Cargar la librería express a través de require
- b. Crear la aplicación express
- c. Especificar el punto para servir los archivos estáticos al cliente
- d. Especificar el enrutamiento
- e. Inicializar el servidor

7. Crear el archivo index.html

Este archivo va a ser el que se envía al cliente al tener una comunicación con el servidor

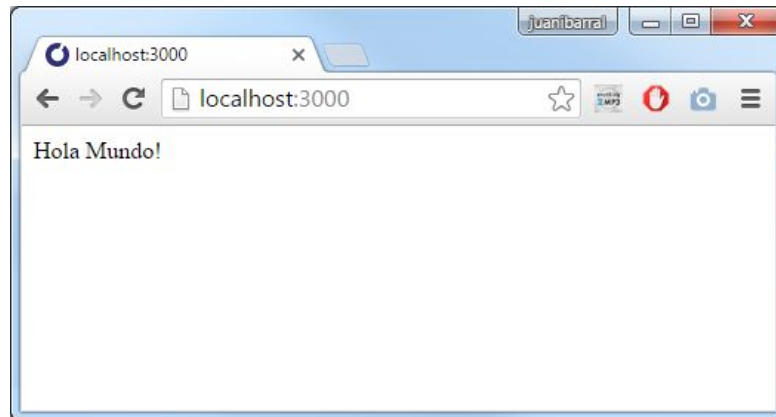
8. Iniciar el servidor a través de node

```
> node index.js
```



```
ca. Command Prompt - node index.js
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

K:\juanibarral\git\TallerNodeJS_AngularJS\src\my_page>node index.js
Example app listening at http://:::3000
-
```



Arquitectura de una aplicación



Aplicación: Lista de tareas

Requerimientos funcionales

- Agregar una nueva tarea a la lista de tareas
- Asignar un estado a la tarea: realizada, no realizada

Lista de Tareas

- ☐ Comprar leche
- ☐ Cita con Dr Perez

Requerimientos no funcionales

- Agregar la tarea a través de un cuadro de texto y un botón
- Mostrar la lista de tareas con sus respectivos estados
- Utilizar un checkbox para cambiar el estado de una tarea

Manejo de datos: database_connector.js

Módulo para administrar la lista de tareas:

- Agregar una tarea
- Actualizar una tarea
- Retornar la lista de todas las tareas

Utilizamos un objeto para guardar las tareas y utilizar el nombre de la tarea como llave de búsqueda.

Los métodos de este módulo utilizan un callback para manejar la secuencialidad del código

Comunicación por sockets

El cliente se comunicará con el servidor a través de sockets. Para ello hay que crear la dependencia dentro del archivo index.js y codificar el protocolo de comunicación.

- Mensaje para listar las tareas
- Mensaje para agregar una nueva tarea
- Mensaje para actualizar una tarea

Crear los listeners del WebSocket para responder a los mensajes del protocolo

Cliente: Controlador simple

Creamos un controlador simple para manejar las tareas:

- Crear los atributos y métodos para manejar las tareas: agregar, cambiar el estado, y actualizar la lista de tareas
- Conectar con el API de los WebSockets
- Crear los listeners para la implementación del protocolo de comunicación

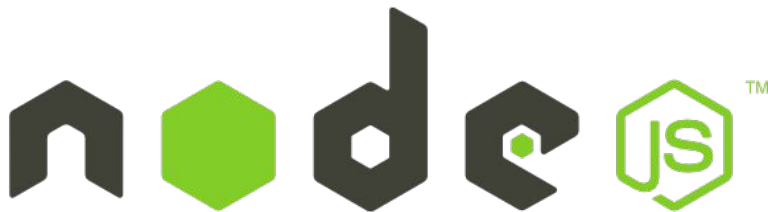
Angular permite una comunicación directa entre la vista y el controlador a través del objeto `$scope`.

Cliente: Creación de la vista

Actualizamos el archivo index.html para que utilice AngularJS.

- Los tags que comienzan con **ng-** son tags AngularJS
- **ng-app**: La aplicación que va a controlar todos los tags dentro del tag en el que se encuentra
- **ng-controller**: Controlador que va a administrar los llamados a los atributos dentro del scope del controlador. Estos llamados deben estar dentro del tag en donde se encuentra el controlador
- **ng-model**: Punto de encuentro entre la vista y el controlador a través de un atributo dentro del \$scope del controlador. (two-way data binding)
- **{{atributo}}**: Forma de tener información de las variables del \$scope (one-way data binding)

Gracias!



<https://nodejs.org/en/>



<https://angularjs.org/>