



JavaScript



# El proyecto Quiz y MVC

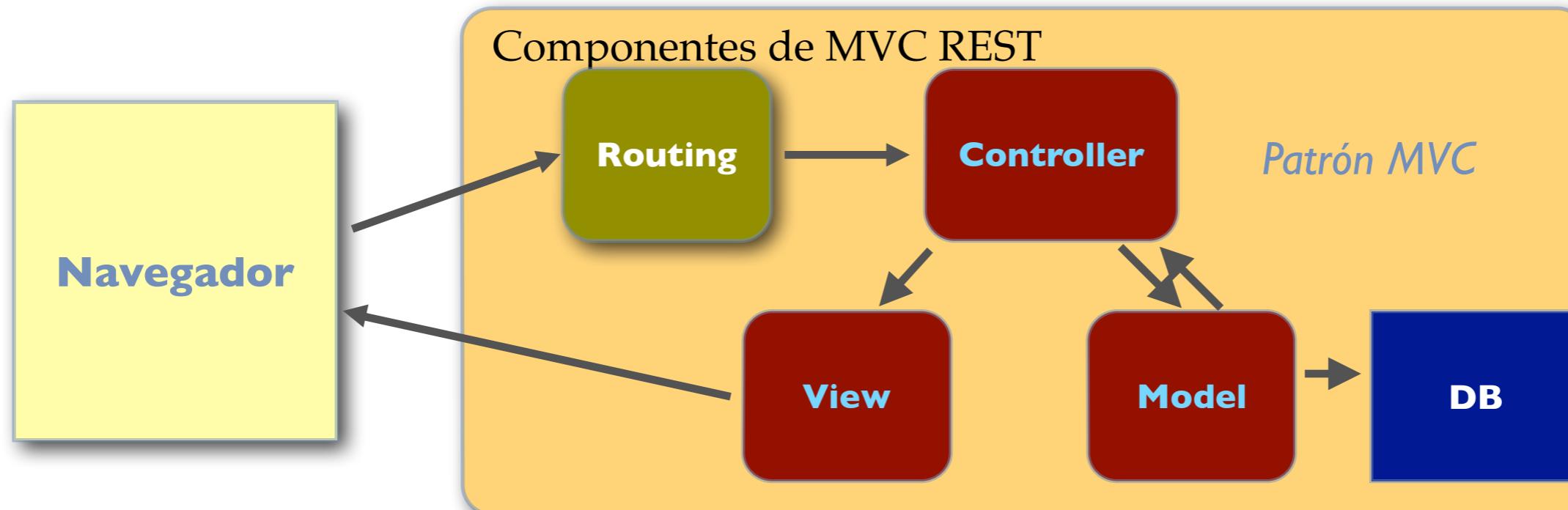
Juan Quemada, DIT - UPM

# Patrón MVC

(Adaptado a Web)

**MVC:** propuesto por **Trygve Reenskaug** en 1979 para estructurar aplicaciones basadas en ventanas

- ◆ **Modelo:** Datos de la aplicación y sus restricciones
  - En una BD, en un fichero (este ejemplo), ...
- ◆ **Vista:** Genera vistas de usuario
  - Prepara la página Web que se envía al cliente
- ◆ **Controlador:** orquesta la aplicación
  - Contiene la lógica de la aplicación
- ◆ **Routing:** asocia URLs con acciones del controlador



# MVC y la estructura de un equipo

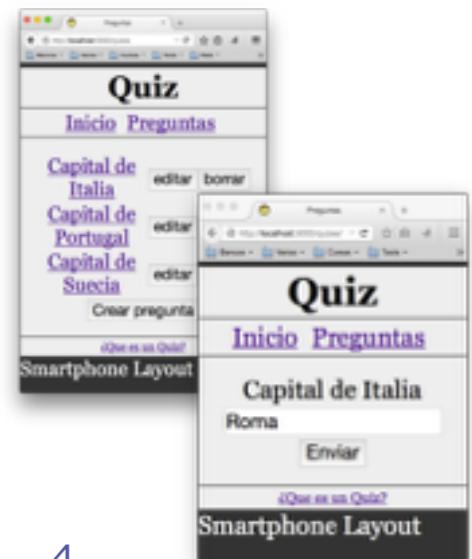
- ◆ MVC divide el trabajo de diseño entre distintos especialistas
  - **Modelo:** especialista en bases de datos (técnico)
  - **Vistas:** diseñador (creativo, artista)
  - **Controlador:** programador de la lógica de la aplicación (técnico)
  - **Router:** arquitecto diseñador del interfaz de la aplicación (técnico)
- ◆ Además existen otras especialidades
  - **Administrador:** administrador de sistemas en la nube (técnico)
  - **Diseñador de experiencia:** usabilidad, interacción, marca, .... (mixto)
  - **SEO (Search Engine Optimizatión):** posiciona portal en buscadores (Google, ...)
  - **Community Manager:** gestiona la relación con los usuarios
  - .....

# Proyecto Quiz

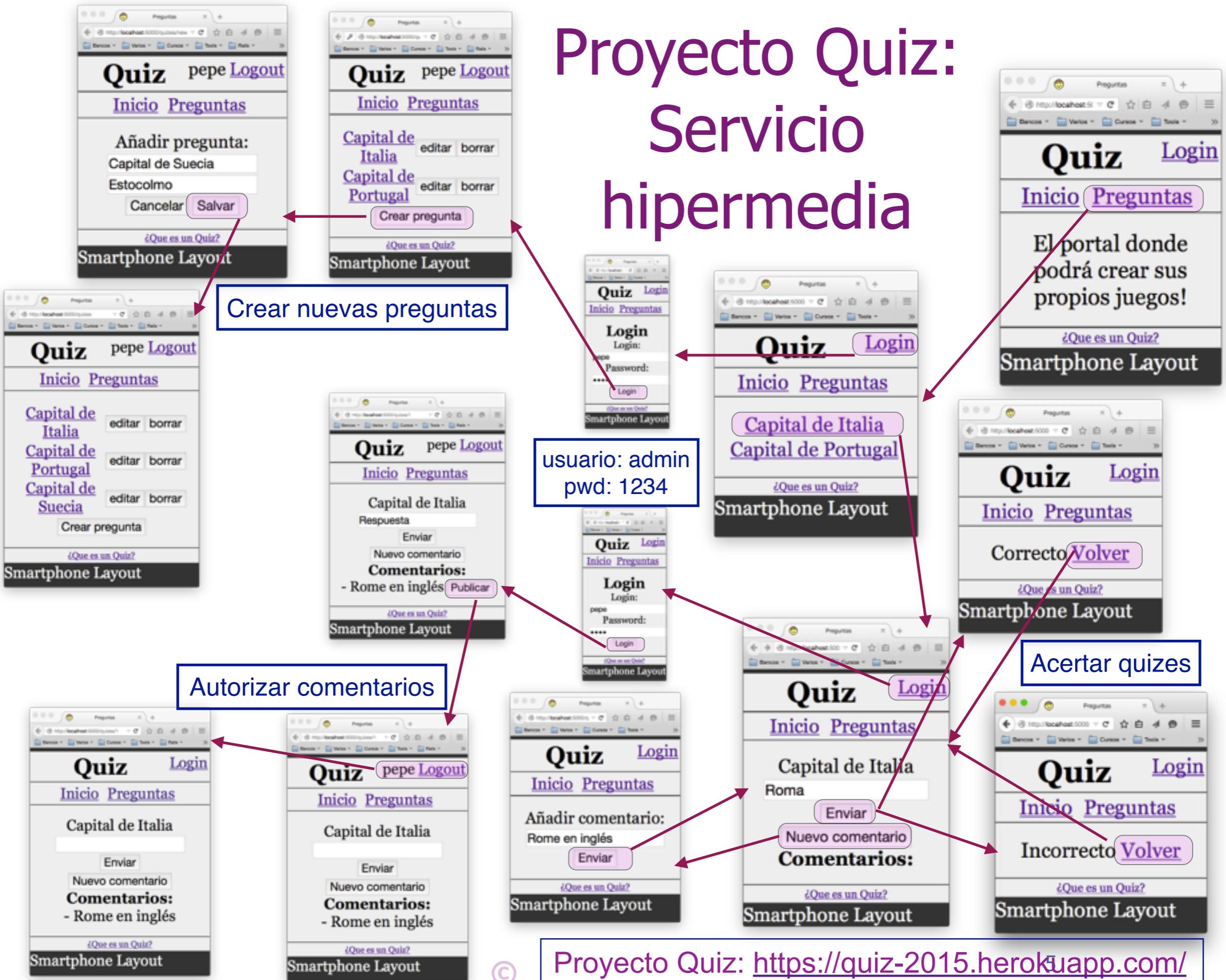
- ◆ **Paso 1:** Esqueleto del proyecto con express-generator
- ◆ **Paso 2:** Primera página y el favicon
- ◆ **Paso 3:** Primera pregunta
- ◆ **Paso 4:** Vistas parciales y marco
- ◆ **Paso 5:** CSS adaptable a móviles y PCs
- ◆ **Paso 6:** Despliegue en la nube (Heroku)
- ◆ **Paso 7:** La base de datos: sequelize.js y SQLite
- ◆ **Paso 8:** Desplegar en Heroku con Postgres
- ◆ **Paso 9:** Lista de preguntas
- ◆ **Paso 10:** Autoload de la DB
- ◆ **Paso 11:** Crear preguntas
- ◆ **Paso 12:** Validación de entradas
- ◆ **Paso 13:** Editar preguntas
- ◆ **Paso 14:** Borrar preguntas
- ◆ **Paso 15:** Crear comentario
- ◆ **Paso 16:** Autenticación y sesión
- ◆ **Paso 17:** Autorización
- ◆ **Paso 18:** Moderación de comentarios
- ◆ **Paso 19:** HTTPS - HTTP Seguro

**Objetivo:** Crear un pequeño portal Web con un juego de adivinanzas (quizes) usando MVC y vistas adaptables a móvil. Quiz ilustra también el uso de herramientas de gestión de proyectos.

El proyecto Quiz y sus versiones están en **GITHUB**  
<https://github.com/jquemada/quiz-2015/commits/master>  
y se ha desplegado y está operativo en **heroku**  
<https://quiz-2015.herokuapp.com/>



# Proyecto Quiz: Servicio hipermedia



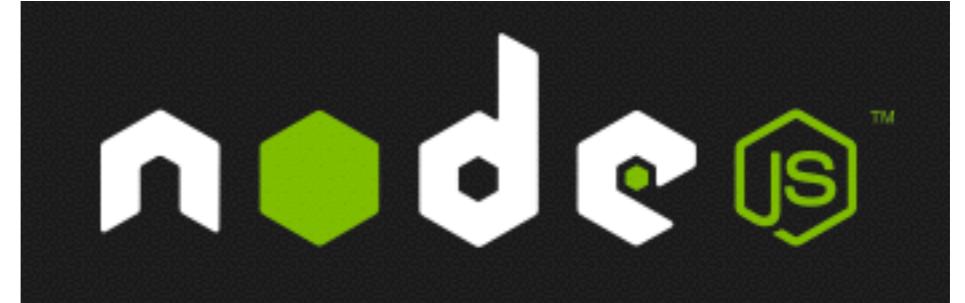
# Quiz: Interfaz REST, DB y MVC

id	pregunta	resp.		id	texto	publicar
1	Capital de Italia	Roma		1	Nuevo comentario	TRUE
2	Capital de Portugal	Lisboa		2	Fue Sintra en el pasado	FALSE
3	.....	.....		3	Rome en ingles	TRUE
4	.....	.....		4	.....	

Método	Ruta	Acciones y controladores	Vistas
<b>GESTION DE SESSION</b>			
<b>GET</b>	/login	sessionController. <b>new</b>	session/new.ejs
<b>POST</b>	/login	sessionController. <b>create</b>	
<b>GET</b>	/logout	sessionController. <b>destroy</b>	
<b>GESTION DE QUIZES (PREGUNTAS)</b>			
<b>GET</b>	/quizes	quizController. <b>index</b>	quizes/index.ejs
<b>GET</b>	/quizes/new	quizController. <b>new</b>	quizes/new.ejs
<b>GET</b>	/quizes/:id	quizController. <b>show</b>	quizes/show.ejs
<b>GET</b>	/quizes/:id/answer	quizController. <b>answer</b>	quizes/answer.ejs
<b>POST</b>	/quizes	quizController. <b>create</b>	
<b>GET</b>	/quizes/:id/edit	quizController. <b>edit</b>	quizes/edit.ejs
<b>PUT</b>	/quizes/:id	quizController. <b>update</b>	
<b>DELETE</b>	/quizes/:id	quizController. <b>destroy</b>	
<b>GESTION DE COMENTARIOS</b>			
<b>GET</b>	/quizes/:id/comments/new	commentController. <b>new</b>	comments/new.ejs
<b>POST</b>	/quizes/:id/comments	commentController. <b>create</b>	
<b>GET</b>	/quizes/:id/comments/:id/publish	commentController. <b>publish</b>	



JavaScript



# Quiz 1a : Esqueleto del proyecto con express-generador

Juan Quemada, DIT - UPM

# Paquetes node.js: npm

## ◆ Paquete

- programa empaquetado utilizable por terceras personas
  - ◆ comando del SO, programa independiente, librería, ..

## ◆ npm (Node Package System)

- Ecosistema de gestión de **paquetes de node.js**
  - ◆ Soporta la creación e instalación de paquetes
    - ◆ y sus dependencias (otros paquetes usados)
  - ◆ Tiene muchas otras funciones

## ◆ repositorio npm o npm registry : <https://www.npmjs.org>

- servidor con un repositorio de paquetes de libre acceso
  - ◆ Permite instalación, publicación, búsqueda, ....

## ◆ La comunidad node.js crece muy deprisa

- gracias a la sencillez y eficacia de **npm** y del **registry**

The screenshot shows the npm homepage with the following data:

- Total Packages:** 64 173
- Downloads:**
  - 1 720 179 downloads in the last day
  - 38 795 729 downloads in the last week
  - 129 263 936 downloads in the last month
- Patches welcome!**
- Any package can be installed by using `npm install`.**
- Add your programs to this index by using `npm publish`.**
- Recently Updated**
  - 0m `x-select`
  - 1m `hubot-scripts`
  - 2m `simplewebrtc`
  - 2m `easejs`
  - 4m `beyo-plugin-i18n`
  - 5m `gammautils`
  - 7m `segmentio-integrations`
  - 7m `mop`
  - 8m `datafilter`
  - 10m `cytoscape`
  - [More...](#)
- Most Depended Upon**
  - 5372 `underscore`
  - 4524 `async`
  - 3757 `request`
  - 2572 `commander`
  - 2550 `express`
  - 2491 `optimist`
  - 2380 `lodash`
  - 2057 `coffee-script`
  - 1919 `colors`
  - 1454 `mkdirp`
  - [More...](#)
- Most Starred**
  - 236 `express`
  - 121 `async`
  - 104 `request`
  - 100 `grunt`
  - [More...](#)
- Most Prolific Recently**
  - 51 `sindresorhus`
  - 44 `pnidem`
  - 41 `jonschlinkert`
  - 37 `dbashford`
  - [More...](#)

```

venus:s7 jq$ npm -h

Usage: npm <command>

where <command> is one of:
  add-user, adduser, apihelp, author, bin, bugs, c, cache,
  completion, config, ddp, dedupe, deprecate, docs, edit,
  explore, faq, find, find-dupes, get, help, help-search,
  home, i, info, init, install, isntall, la, link, list, ll,
  ln, login, ls, outdated, owner, pack, prefix, prune,
  publish, r, rb, rebuild, remove, restart, rm, root,
  run-script, s, se, search, set, show, shrinkwrap, star,
  start, stop, submodule, tag, test, tst, un, uninstall,
  unlink, unpublish, unstar, up, update, version, view,
  whoami

npm <cmd> -h      quick help on <cmd>
npm -l            display full usage info
npm faq           commonly asked questions
npm help <term>  search for help on <term>
npm help npm     involved overview

Specify configs in the ini-formatted file:
  /Users/jq/.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@1.1.63 /usr/local/lib/node_modules/npm
venus:s7 jq$ 

```

**npm install <paquete>** // instala un paquete del servidor

**npm init <paquete>** // inicializa paquete (fichero package.json)

**npm publish <paquete>** // publica un paquete en el servidor

.....

## Comando npm

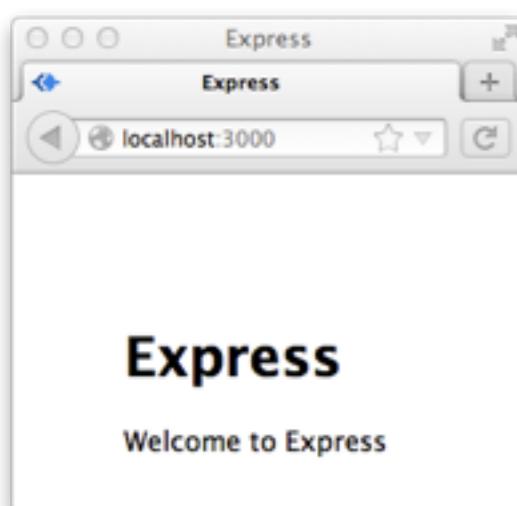
- ◆ **npm** es también el nombre de un comando de gestión de paquetes de node.js
  - Se utiliza fundamentalmente para instalar, descubrir, desarrollar o publicar paquetes node.js
- ◆ El comando npm se comunica con el servidor central cuando lo necesita
  - <https://www.npmjs.org>
- ◆ El comando npm entiende el formato de un paquete y lo procesa adecuadamente
  - tiene mucha funcionalidad
    - Doc: <https://www.npmjs.org/doc/>
    - La documentación también se puede obtener por consola como cualquier comando (tipo UNIX).

# Quiz 1: Creación del esqueleto

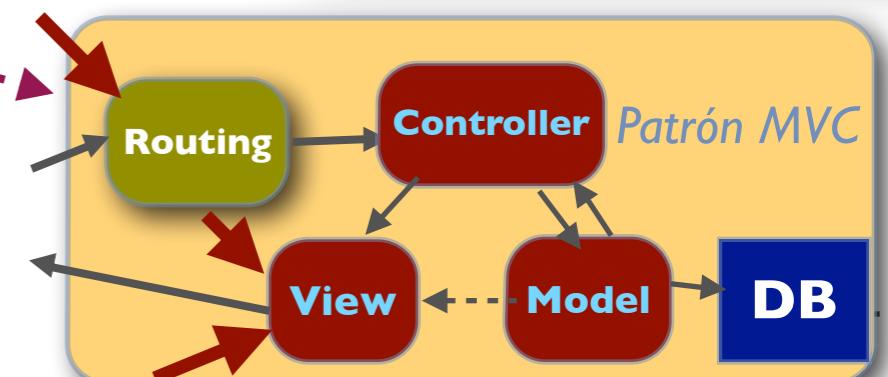
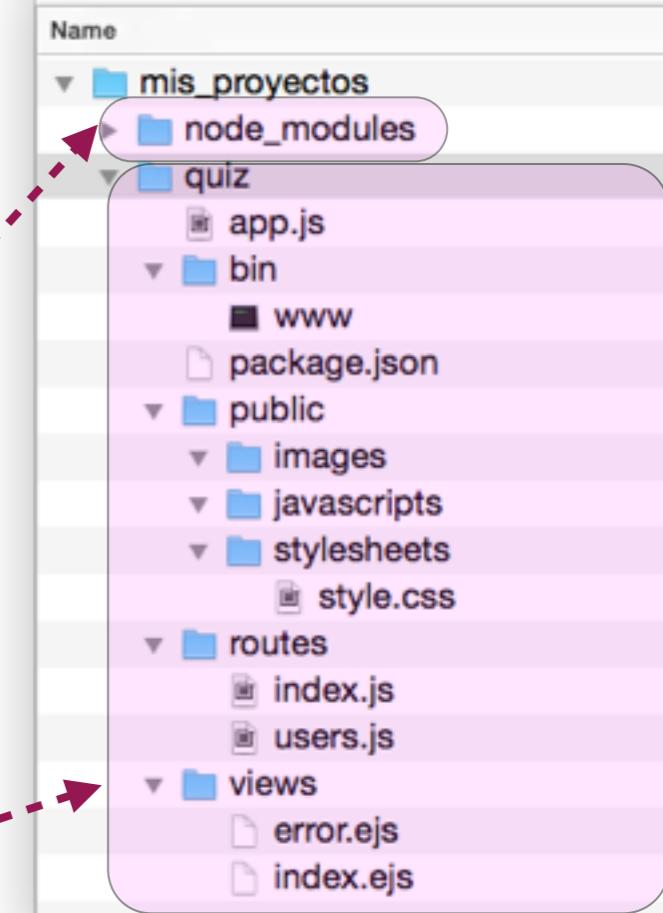
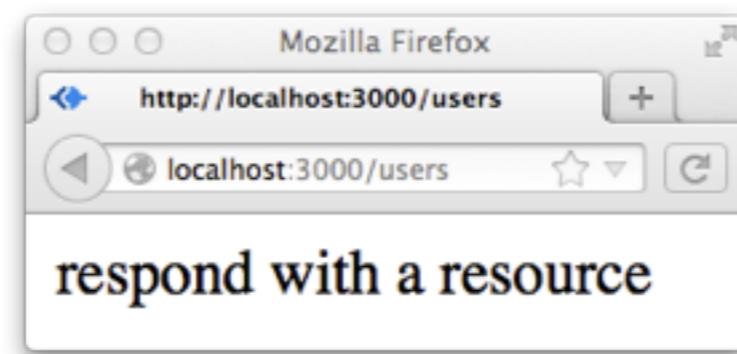
**Objetivo:** El paso del diseño Quiz 1 crea con el comando **express (generator)** el directorio del proyecto con un esqueleto de programas y directorios que iremos enriqueciendo en los siguientes pasos

- ◆ **Paso 1:** Instalar paquete **express-generator**
  - Generador de esqueletos de proyectos express
- ◆ **Paso 2:** generar el esqueleto del proyecto Quiz
  - Utilizando express-generator
- ◆ **Paso 3:** Guardar versión (commit) con git
  - Antes de guardarlo conviene ejecutarlo y probarlo

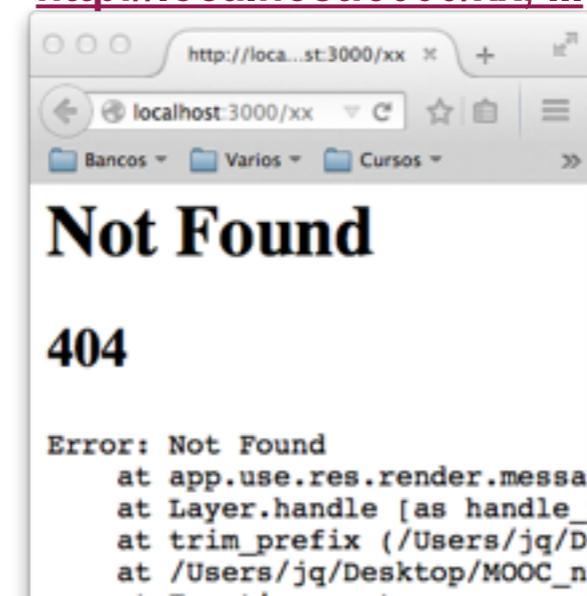
<http://localhost:3000/>



<http://localhost:3000/users>



[http://localhost:3000/xx, ...](http://localhost:3000/xx,...)



# express-generator

◆ **express-generator** es un paquete npm que

- Instala el comando **express**

- ◆ desde el servidor central [www.npmjs.org](https://www.npmjs.org)

◆ **express** se instala con “npm install” en el directorio

- **node\_modules/express-generator/bin/**

◆ El comando **express** crea proyectos express

- Se invoca desde **mis\_proyectos** con el path completo

- ◆ “**node\_modules/express-generator/bin/express .....**”

```
// Recomendamos crear un directorio para todos los
```

```
// proyectos generados con express-generator:
```

```
..$ mkdir mis_proyectos
```

```
..$ cd mis_proyectos
```

```
// Se instala la versión 4.9.0 de express-generator en el
```

```
// directorio node_modules/express-generator con
```

```
..$ npm install express-generator@4.9.0
```

```
// Quiz está probado solo con v.4.9.0 y no se
```

```
// garantiza que funcione con otras versiones.
```

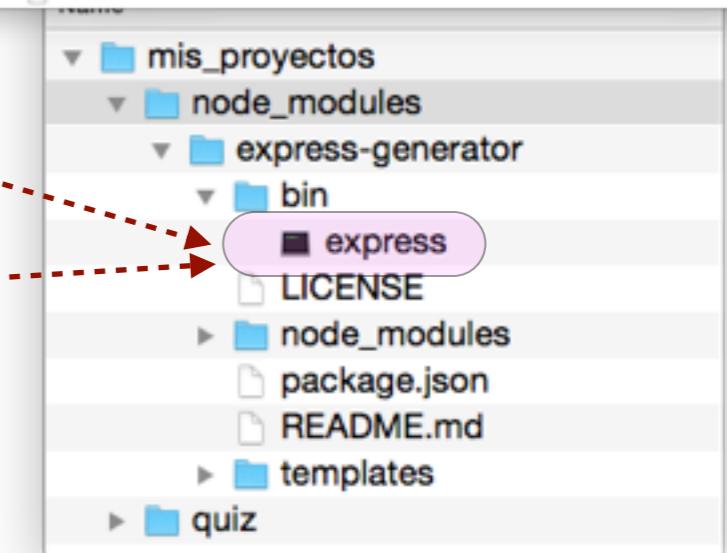
```
venus:s9 jq$ node_modules/express-generator/bin/express --help
```

Usage: express [options] [dir]

Options:

-h, --help	output usage information
-V, --version	output the version number
-e, --ejs	add ejs engine support (defaults to jade)
--hbs	add handlebars engine support
-H, --hogan	add hogan.js engine support
-c, --css <engine>	add stylesheet <engine> support (less stylus sass postcss)
-f, --force	force on non-empty directory

```
venus:s9 jq$
```



<https://www.npmjs.org>



Search Packages

**express-generator**

```
venus-5:s8 jq$ npm install express-generator@4.9.0
express-generator@4.9.0 node_modules/express-generator
├── commander@1.3.2 (keypress@0.1.0)
└── mkdirp@0.5.0 (minimist@0.0.8)
venus-5:s8 jq$
```

194 downloads in the last day

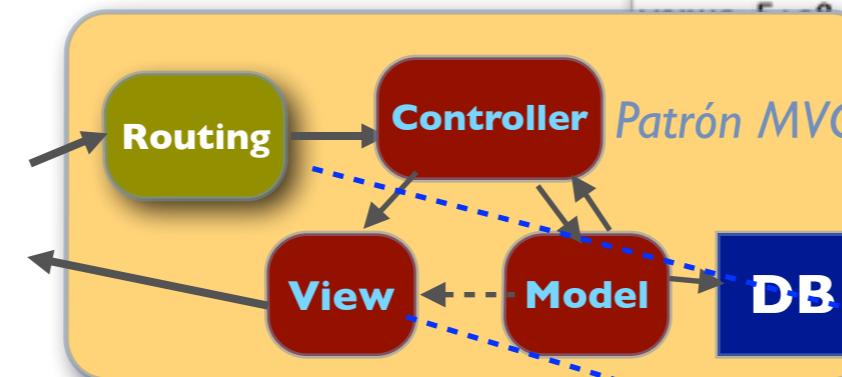
# Crear proyecto Quiz

## ◆ Al ejecutar el comando **express**

- Se crea el esqueleto del proyecto quiz
  - se debe invocar con el path completo al comando
  - Opción **--ejs**: instala el generador de vistas EJS
  - Parámetro **quiz**: directorio del proyecto/esqueleto
    - Donde se crea el esqueleto del proyecto

## ◆ Esqueleto de quiz

- Directorio donde albergar
  - todos los elementos básicos del proyecto
    - Aplicación
    - Vistas
    - Enrutador
    - ....



// Invocamos así el comando **express** desde el directorio **mis\_proyectos**

// la opción **--ejs** indica que se utilice el generador de vistas **EJS**  
// el parametro **quiz** indica el nombre del directorio de proyecto creado

..mis\_proyectos\$ node\_modules/express-generator/bin/express --ejs quiz

```
venus-5:s8 jq$ node_modules/express-generator/bin/express --ejs quiz
create : quiz
create : quiz/package.json
create : quiz/app.js
create : quiz/public
create : quiz/public/javascripts
create : quiz/public/stylesheets
create : quiz/public/stylesheets/style.css
create : quiz/routes
create : quiz/routes/index.js
create : quiz/routes/users.js
create : quiz/public/images
create : quiz/views
create : quiz/views/index.ejs
create : quiz/views/error.ejs
create : quiz/bin
create : quiz/bin/www

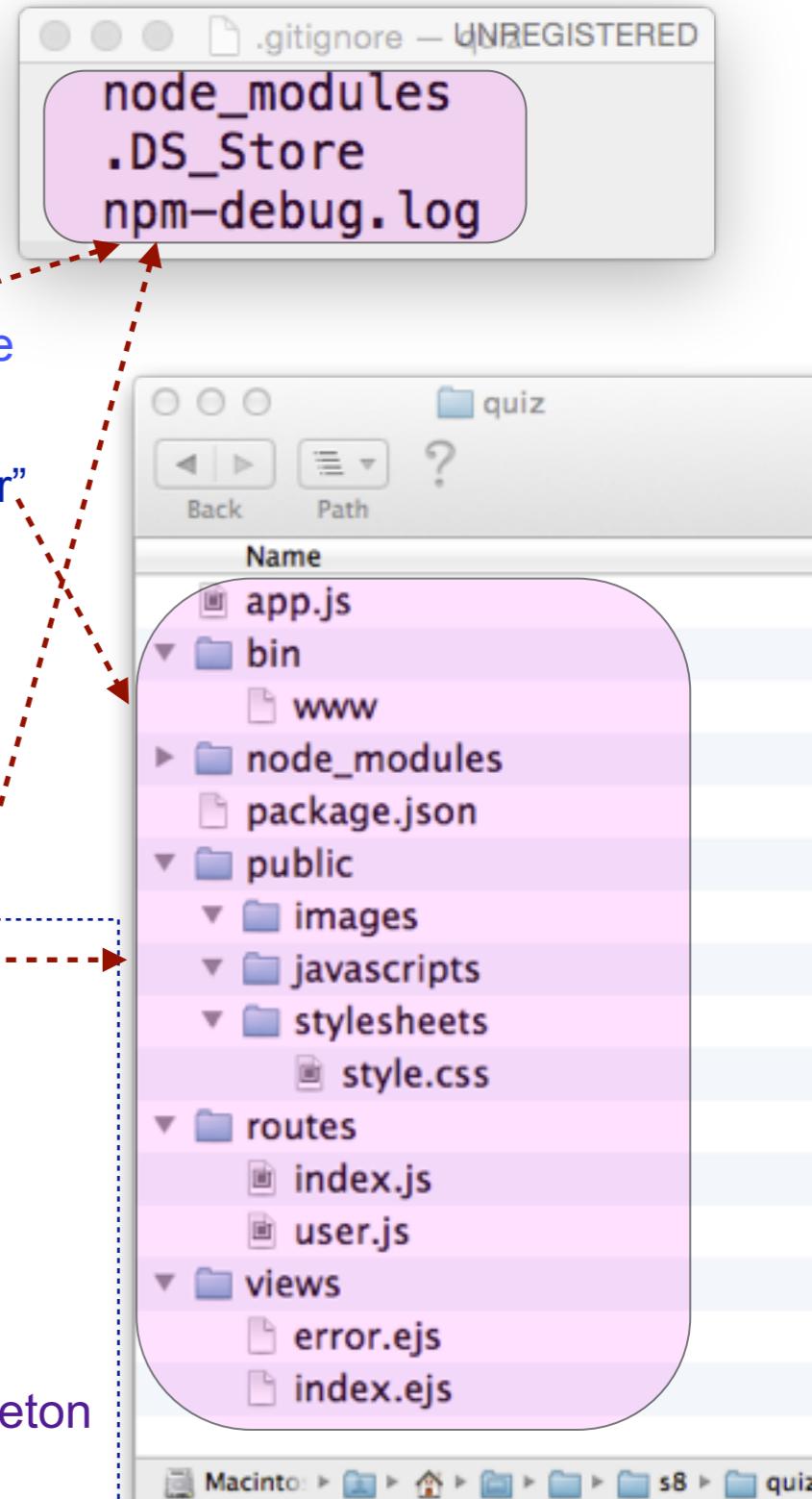
install dependencies:
$ cd quiz && npm install

run the app:
$ DEBUG=quiz ./bin/www
```

```
core
mis_proyectos
node_modules
express-generator
bin
express
LICENSE
package.json
README.md
templates
quiz
routes
index.js
users.js
views
error.ejs
index.ejs
```

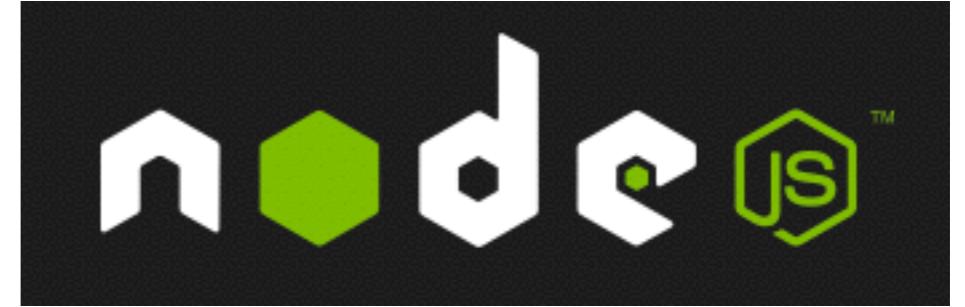
# Guardar versión GIT

- ◆ express-generator genera el esqueleto del proyecto
    - Guardamos este primer paso de Quiz en el repositorio git
      - ◆ Creamos el proyecto y el repositorio con: "git init"
      - ◆ Indicamos lo que no queremos guardar en ".gitignore"
        - En mi caso (MAC) el directorio node\_modules y el fichero .DS\_Store
      - ◆ Subimos los demás ficheros al índice con: "git add ."
      - ◆ Creamos la primera versión: git commit -m "esqueleto express-generator"
  - ◆ Ficheros de configuración: empiezan por punto, p.e. .gitignore
    - no se ven en las ventanas y para editarlos se puede:
      - ◆ Copiar a fichero visible, p.e. "cp .gitignore xx", editar y a la inversa
      - ◆ Editar con sublime text abriendo todo el directorio
- ```
..$ cd quiz // entramos en directorio quiz -----  
..$ git init // iniciamos la gestión del proyecto quiz con git  
          // Identificamos los ficheros que no queremos guardar en .gitignore  
..$ git add . // añadimos todo lo contenido en directorio al área de cambios  
          // generamos versión identificada como: express-generator skeleton  
..$ git commit -m "esqueleto express-generator"
```





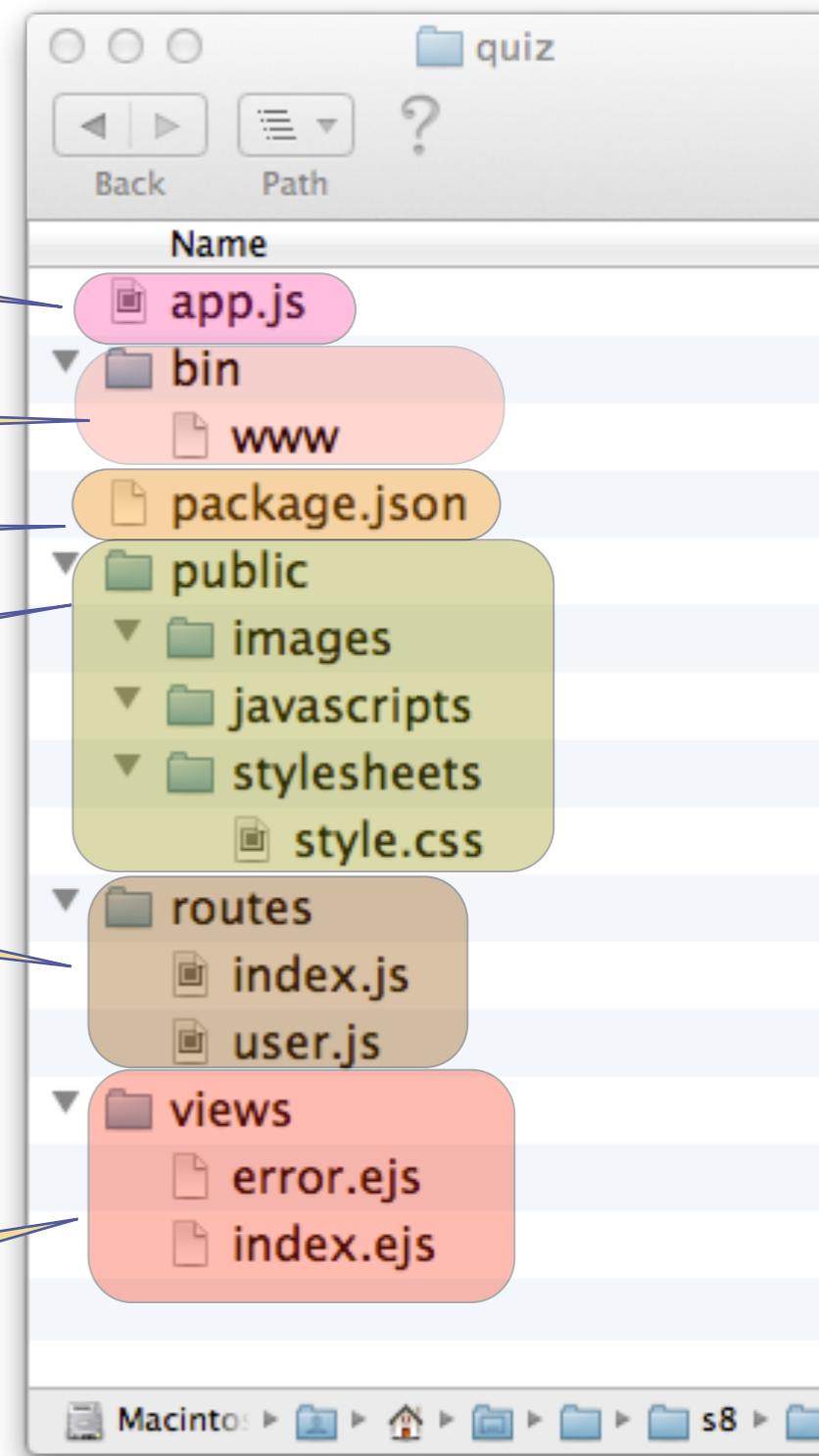
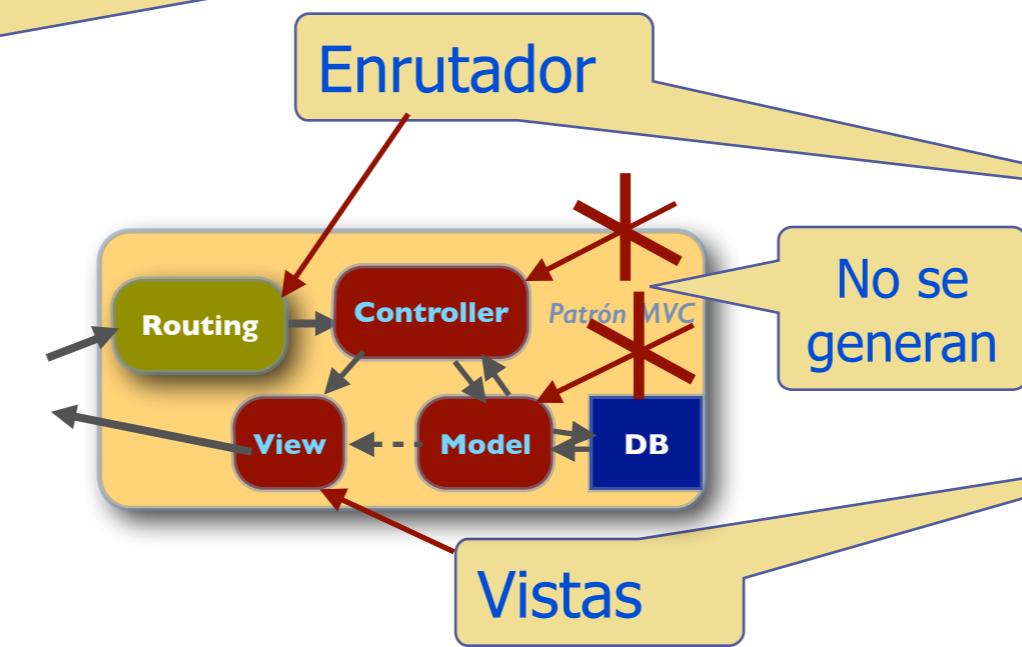
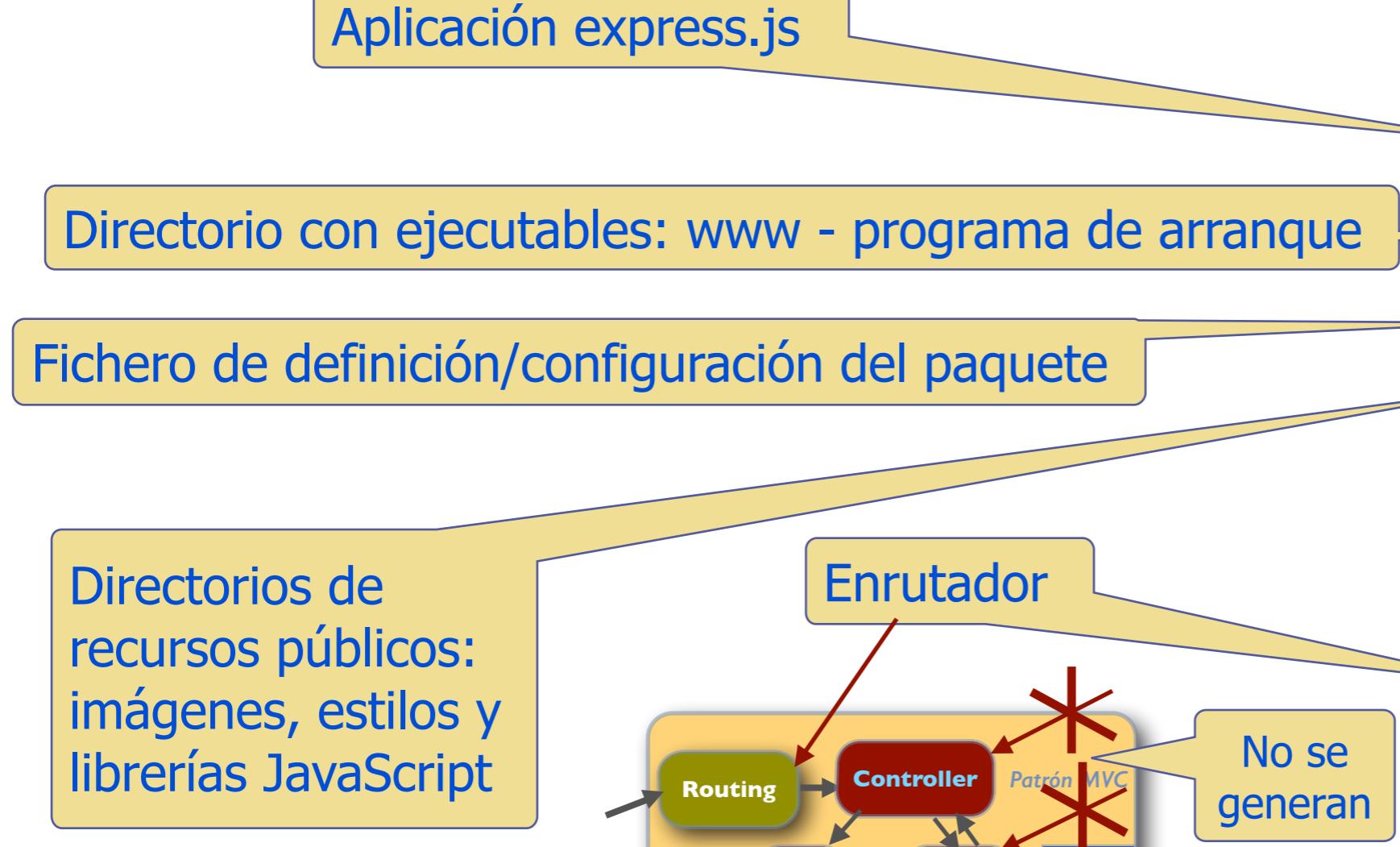
JavaScript



# Quiz 1b: El esqueleto del proyecto

Juan Quemada, DIT - UPM

# Ficheros creados por express-generador



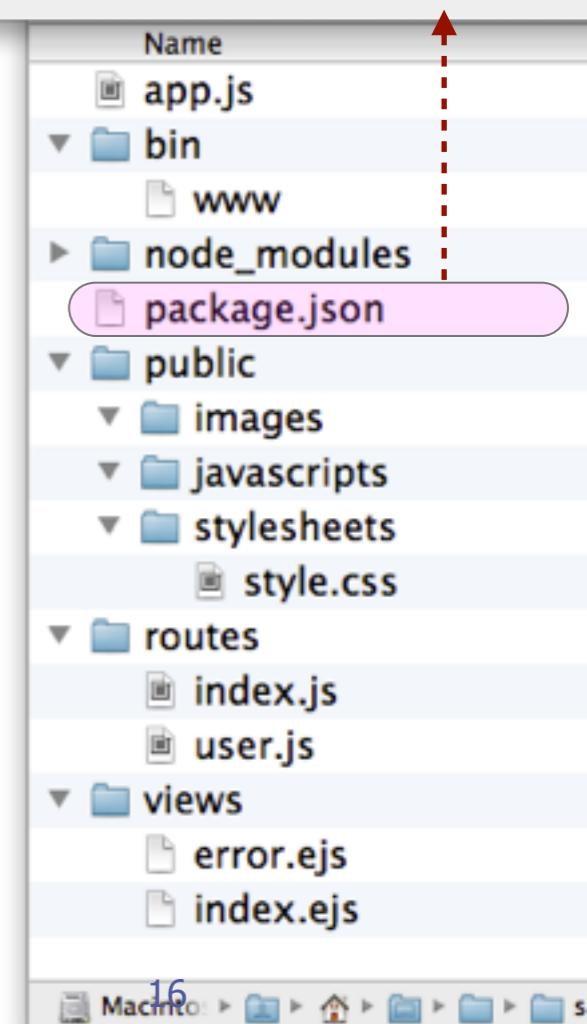
# package.json

- ◆ **package.son** es un fichero en formato JSON que define
  - las características del paquete npm de este proyecto
    - ◆ Doc: <https://www.npmjs.org/doc/files/package.json.html>
- ◆ **express-generator** genera solo los siguientes parámetros:
  - name, version, private, scripts y dependencies
- ◆ Los paquetes usados en el proyecto son las “dependencies”
  - Se instalan con: **npm install**
- ◆ Los scripts son los comandos del proyecto
  - Suelen incluir: arranque (start), pruebas (test), despliegue (deploy), ..

```
// package.json define a través de las dependencias (dependencies:)  
// los paquetes que nuestro proyecto necesita. Los instalamos así  
  
..$ npm install // instala todos los paquetes indicados en package.json  
// quiz/node_modules contiene el software instalado  
  
..$ npm start // invoca script de arranque (start) en package.json
```



```
{  
  "name": "quiz",  
  "version": "0.0.0",  
  "private": true,  
  "scripts": {  
    "start": "node ./bin/www"  
  },  
  "dependencies": {  
    "express": "~4.9.0",  
    "body-parser": "~1.8.1",  
    "cookie-parser": "~1.3.3",  
    "morgan": "~1.3.0",  
    "serve-favicon": "~2.1.3",  
    "debug": "~2.0.0",  
    "ejs": "~0.8.5"  
  }  
}
```



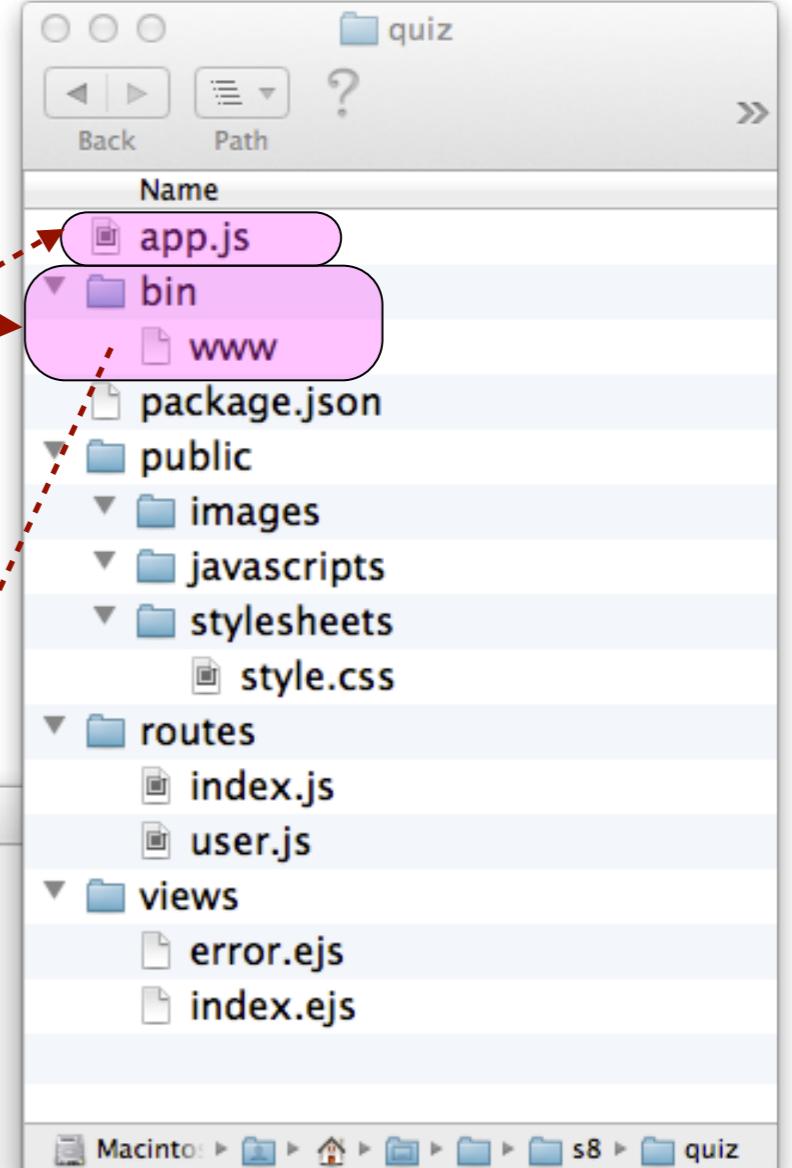
```
// arranque de la aplicación
```

```
..$ node bin/www
```

```
..$ npm start
```

# Programa de arranque

- ◆ El fichero **bin/www**
  - es el programa de arranque de la aplicación
- ◆ La aplicación está en el fichero **app.js**
  - Se importa como un modulo en **bin/www**
    - ◆ que arranca el servidor express



```
#!/usr/bin/env node
var debug = require('debug')('quiz');
var app = require('../app');

app.set('port', process.env.PORT || 3000);

var server = app.listen(app.get('port'), function() {
  debug('Express server listening on port ' + server.address().port);
});
```

# Ejecución de quiz

- ◆ **npm install** instala las dependencias definidas en package.json
  - son todos los paquetes necesarios para ejecutar express.js
- ◆ Con las dependencias instaladas se arranca el servidor:
  - Con el script de arranque (**start**) definido en package.json
    - ◆ **npm start** → “**start": "node ./bin/www"**
  - Invocando directamente bin/www con node
    - ◆ **node ./bin/www**

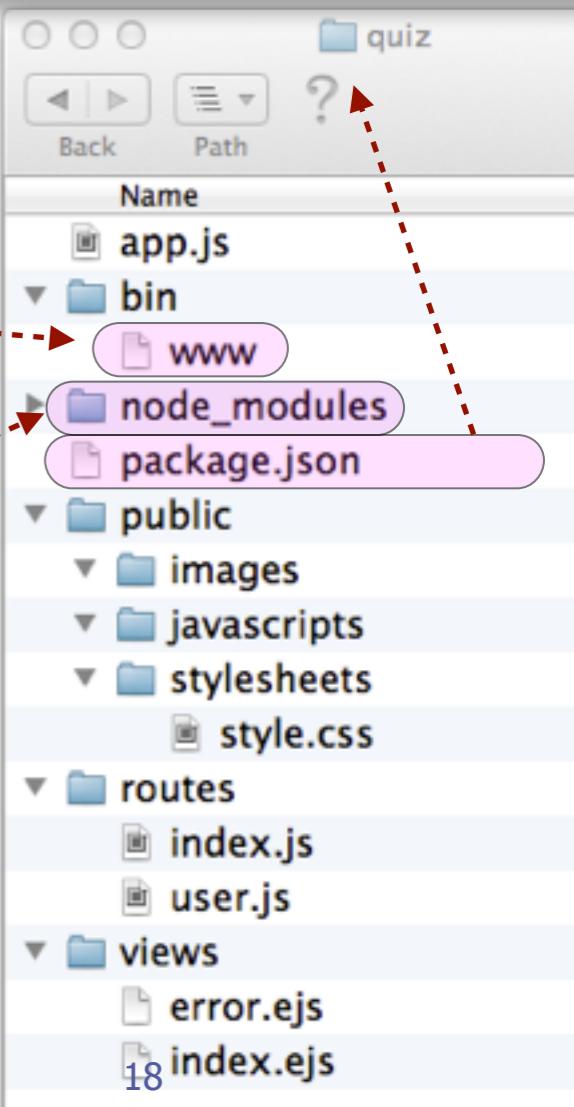
```
// package.json define a través de las dependencias (dependencies:) // los paquetes que nuestro proyecto necesita. Los instalamos así
```

```
..$ cd quiz          // entramos en directorio quiz

..$ npm install      // instala todos los paquetes indicados en package.json
                      // quiz/node_modules contiene el software instalado

..$ node bin/www    // invocación directa de bin/www con node
..$ npm start        // script de arranque a través de package.json
```

```
{  
  "name": "quiz",  
  "version": "0.0.0",  
  "private": true,  
  "scripts": {  
    "start": "node ./bin/www"  
  },  
  "dependencies": {  
    "express": "~4.9.0",  
    "body-parser": "~1.8.1",  
    "cookie-parser": "~1.3.3",  
    "morgan": "~1.3.0",  
    "serve-favicon": "~2.1.3",  
    "debug": "~2.0.0",  
    "ejs": "~0.8.5"  
  }  
}
```



# app.js I

```
app.js UNREGISTERED  
var express = require('express');           importar paquetes con middlewares  
var path = require('path');  
var favicon = require('serve-favicon');  
var logger = require('morgan');  
var cookieParser = require('cookie-parser');  
var bodyParser = require('body-parser');  
  
var routes = require('./routes/index');      importar enrutadores  
var users = require('./routes/users');  
  
var app = express();    crear aplicación  
  
// view engine setup  
app.set('views', path.join(__dirname, 'views')); instalar  
app.set('view engine', 'ejs');    generador de vistas EJS  
  
// uncomment after placing your favicon in /public  
//app.use(favicon(__dirname + '/public/favicon.ico'));  
app.use(logger('dev'));  
app.use(bodyParser.json());    instalar middlewares  
app.use(bodyParser.urlencoded({ extended: false }));  
app.use(cookieParser());  
app.use(express.static(path.join(__dirname, 'public')));  
  
app.use('/', routes);           instalar enrutadores  
app.use('/users', users);  
....
```

app.js: instala MWs con app.use().

app.js importa primero con require(...) los MWs que va a instalar.

Luego instala los MWs en el mismo orden en que deben ejecutarse cuando llegue una transacción HTTP.

Cada transacción HTTP ejecuta los MWs instalados en el mismo orden en que han sido instalados.

```
....  
app.use('/', routes);      asociar rutas a sus gestores  
app.use('/users', users);  
  
// undefined route, catch 404 and forward to error handler  
app.use(function(req, res, next) {  
  var err = new Error('Not Found');  resto de rutas:  
  err.status = 404;                genera error 404 de HTTP  
  next(err);  
});  
  
// error handlers  
  
// development error handler: will print stacktrace  
if (app.get('env') === 'development') {  
  app.use(function(err, req, res, next) {  gestión de  
    res.status(err.status || 500);          errores durante  
    res.render('error', {                  el desarrollo  
      message: err.message,  
      error: err // print err  
    });  
  });  
}  
  
// production error handler: no stacktraces leaked to user  
app.use(function(err, req, res, next) {  gestión de  
  res.status(err.status || 500);          errores de producción  
  res.render('error', {  
    message: err.message,  
    error: {} // don't print err (empty object)  
  });  
});  
exportar app para comando de arranque  
module.exports = app;
```

## app.js II

Se crean e instalan 3 MWs

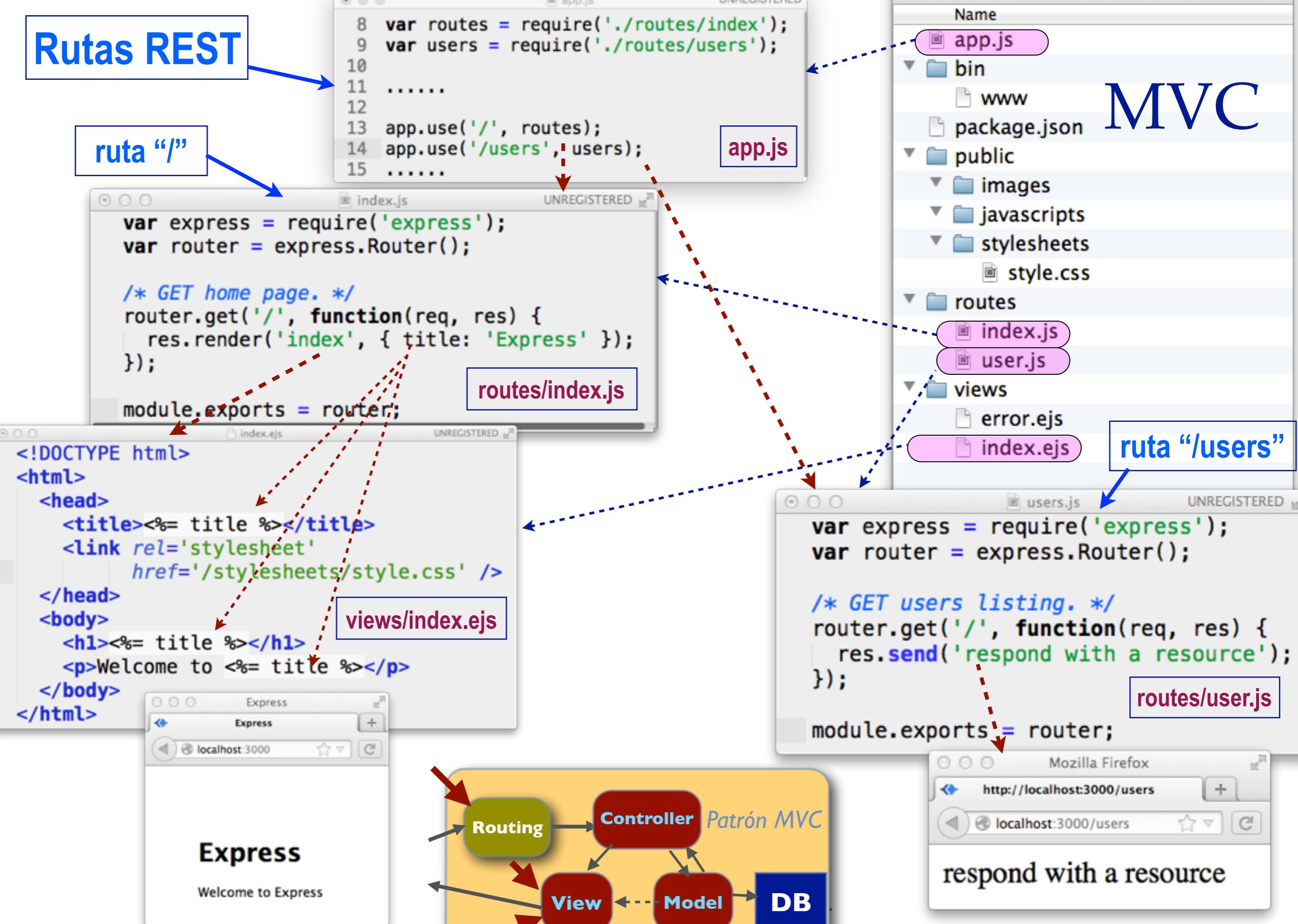
1) El primer MW atiende las rutas no atendidas por los MWs anteriores, generando un error HTTP 404 (recurso no encontrado).

2) MW gestor de errores solo para la fase de desarrollo del proyecto.

3) MW gestor de errores solo para la fase de despliegue del proyecto.

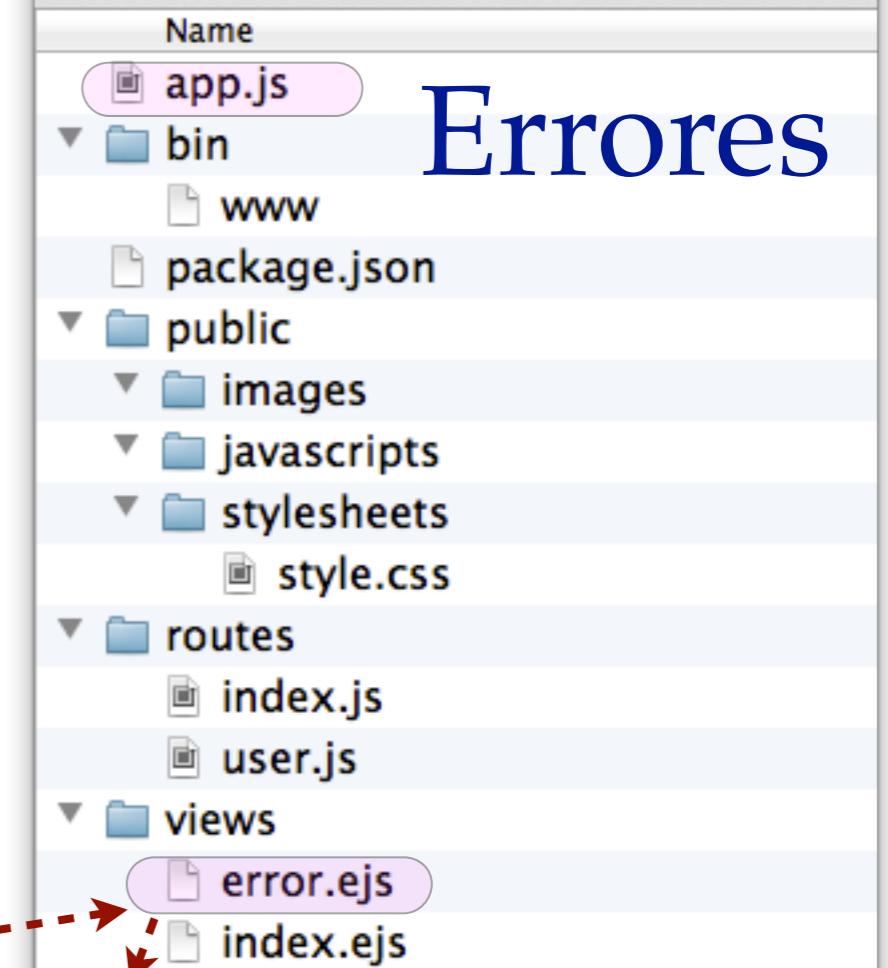
Por ultimo exporta la aplicación, de forma que pueda ser importada en bin/www

# Rutas REST



# Errores

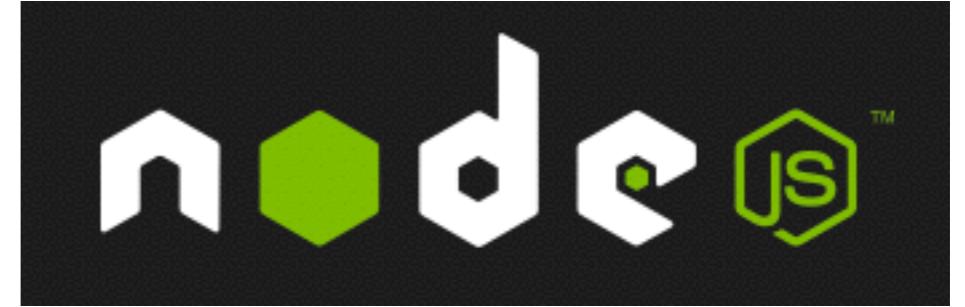
```
....  
app.use('/', routes);  
app.use('/users', users);  
  
// undefined route, catch 404 and forward to error handler  
app.use(function(req, res, next) {  
  var err = new Error('Not Found'); rutas indefinidas  
  err.status = 404; p.e. localhost:3000/xx  
  next(err);  
});  
  
// error handlers  
  
// development error handler: will print stacktrace  
if (app.get('env') === 'development') { desarrollo  
  app.use(function(err, req, res, next) {  
    res.status(err.status || 500);  
    res.render('error', {  
      message: err.message,  
      error: err // print err  
    });  
  });  
  
// production error handler: no stacktraces leaked to user  
app.use(function(err, req, res, next) { producción  
  res.status(err.status || 500);  
  res.render('error', {  
    message: err.message,  
    error: {} // don't print err (empty object)  
  });  
});  
  
module.exports = app;
```



```
<h1><%= message %></h1>  
<h2><%= error.status %></h2>  
<pre><%= error.stack %></pre>  
  
Not Found  
404  
Error: Not Found  
at app.use.res.render.message  
at Layer.handle [as handle_]  
at trim_prefix (/Users/jq/Desktop/MOOC_n...  
at /Users/jq/Desktop/MOOC_n...  
at Function.prototype.process
```



JavaScript



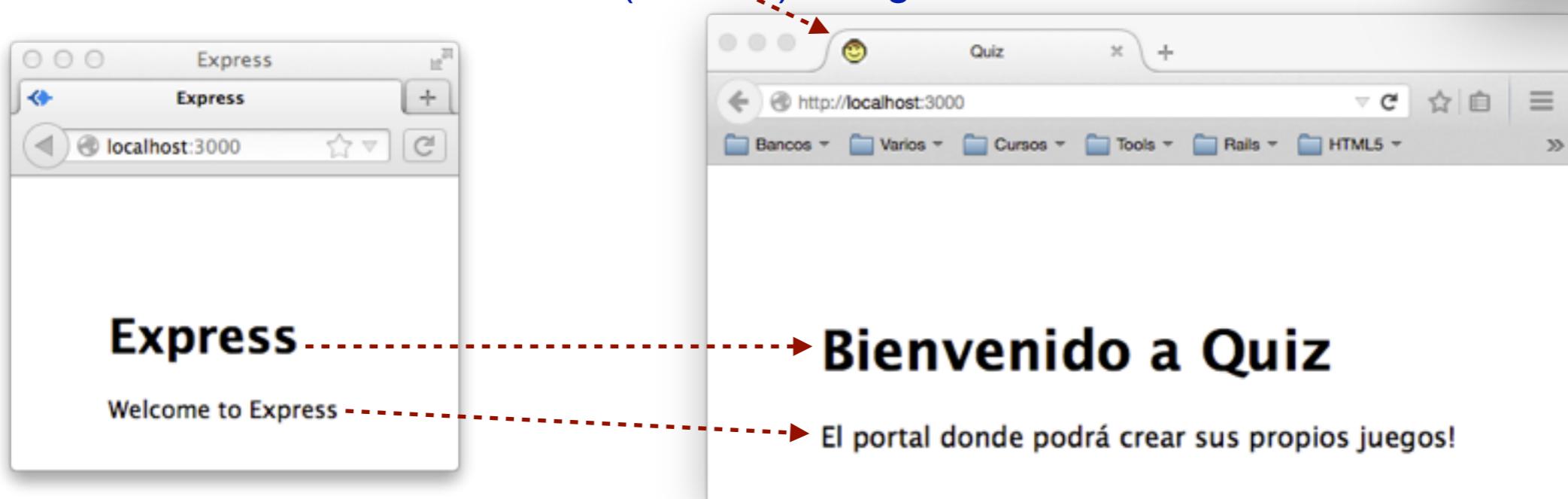
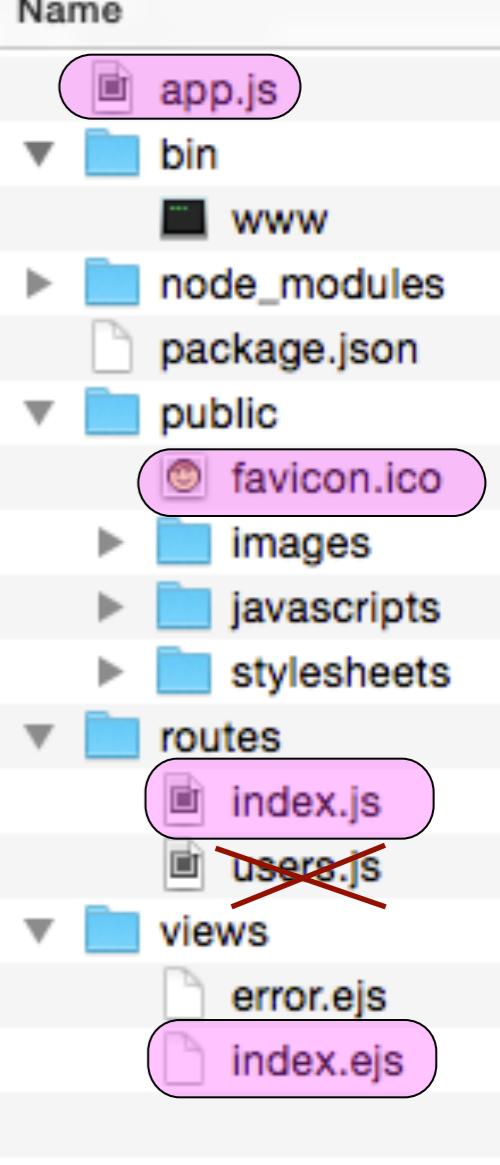
# Quiz 2: Primera página y favicon

Juan Quemada, DIT - UPM

# Quiz 2: Primera página y favicon

**Objetivo:** El paso del diseño Quiz 2 modificamos el esqueleto del proyecto para transformar la pagina (**views/index.ejs**) asociada a la ruta / en la página de entrada a nuestro portal Quiz. Además aprovechamos para **quitar la ruta /users** y **añadir un favicon**.

- ◆ **Paso 1:** Se transforma la página de entrada en: **GET /**
  - En página de bienvenida al portal
    - ◆ Modificar ficheros **routes/index.js** y **views/index.ejs**
- ◆ **Paso 2:** quitamos ruta **/users** creada por express-generator
  - borramos **routes/users.js** y modificamos **app.js**
- ◆ **Paso 3:** añadimos favicon
- ◆ **Paso 4:** Guardamos versión (**commit**) con git



# EJS: Embedded JavaScript

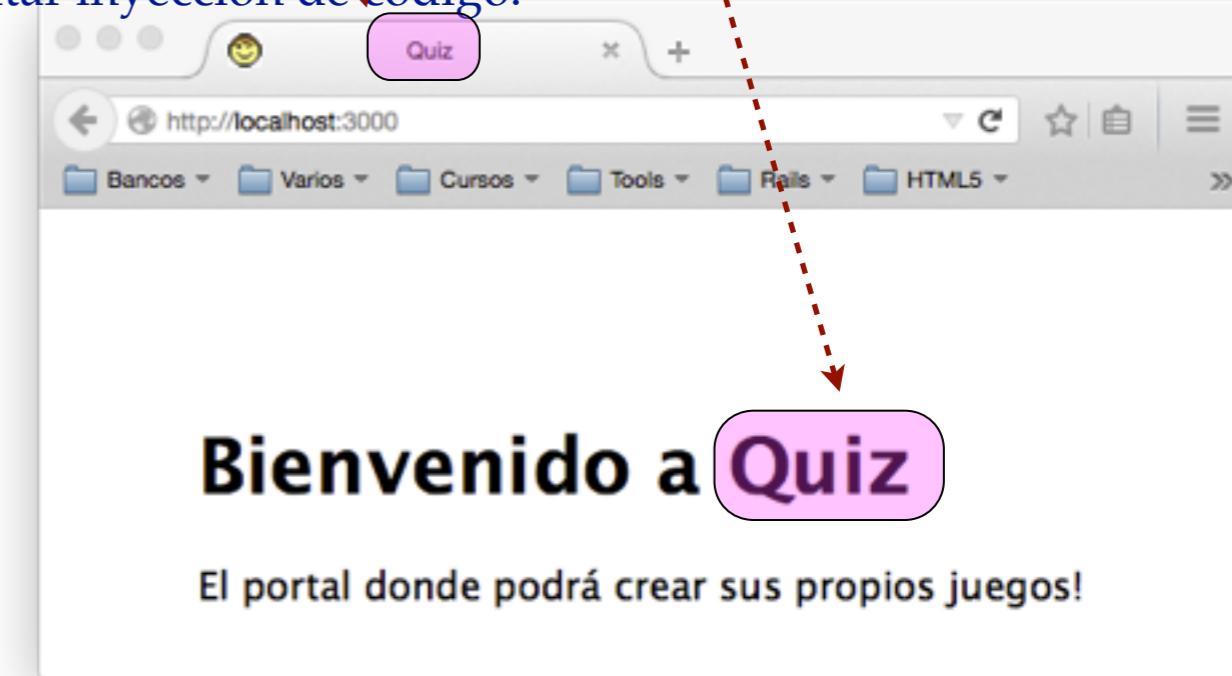
- EJS = Javascript embebido en HTML.
- Los ficheros de vistas EJS contienen texto HTML que incluye código JavaScript o comandos delimitados entre las marcas `<% código %>`.
  - código javascript entre las marcas `<% código %>`.
  - expresiones javascript entre las marcas `<%= expresión %>`.
    - El valor de las expresiones se incorpora al texto HTML.
    - Previamente se escapan los caracteres conflictivos para evitar inyección de código:
      - < se sustituye por `&lt;`
      - > se sustituye por `&gt;`
      - & se sustituye por `&amp;`
      - ...
  - expresiones javascript entre las marcas `<% - expresión %>`.
    - El valor de las expresiones se incorpora al texto HTML.
    - No se escapan los caracteres conflictivos.
  - inclusión de ficheros con `<% include path_a_fichero %>`.

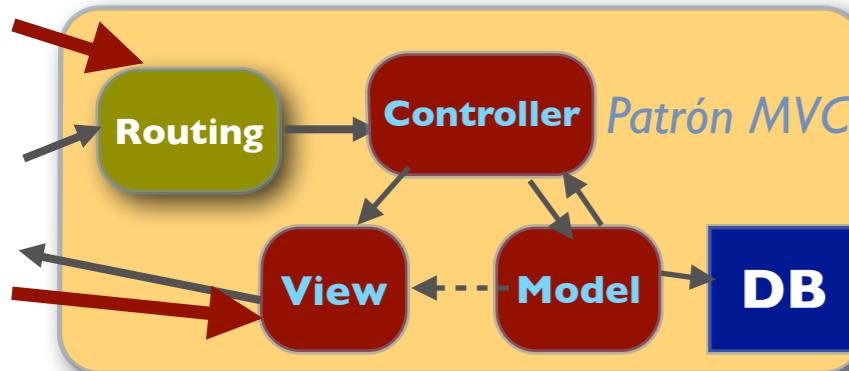
The screenshot shows a code editor window titled "index.ejs". The file contains the following EJS code:

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1>Bienvenido a <%= title %></h1>
    <p>El portal donde podrá crear sus propios juegos!</p>
  </body>
</html>
```

A pink rounded rectangle highlights the title placeholder `<%= title %>` both in the code and in the rendered output below. A dashed red arrow points from this highlighted area to the corresponding placeholder in the browser screenshot.

vista index.ejs





Las **vistas EJS** se generan en el objeto de respuesta **res** y se envían al cliente con

**res.render(vista, params)**

- **vista** es el nombre del fichero con la vista (con extensión **.ejs**)
- **params** es un objeto con parámetros a sustituir en la vista.

app.js

```
var routes = require('../routes/index');
var app = express();
.....
app.use('/', routes);
```

Ruta /

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function(req, res) {
  res.render('index', { title: 'Quiz' });
});

module.exports = router;
```

vista index.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1>Bienvenido a <%= title %></h1>
    <p>El portal donde podrá crear sus propios juegos!</p>
  </body>
</html>
```



Name

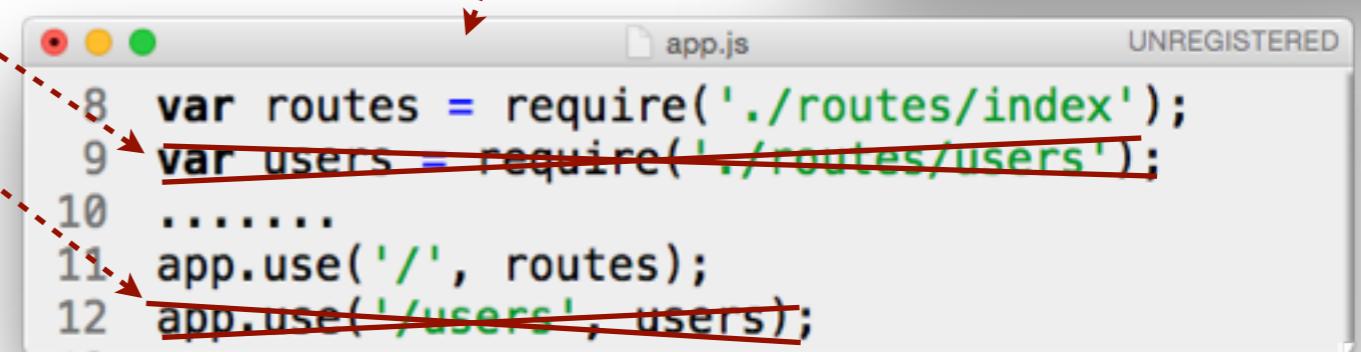
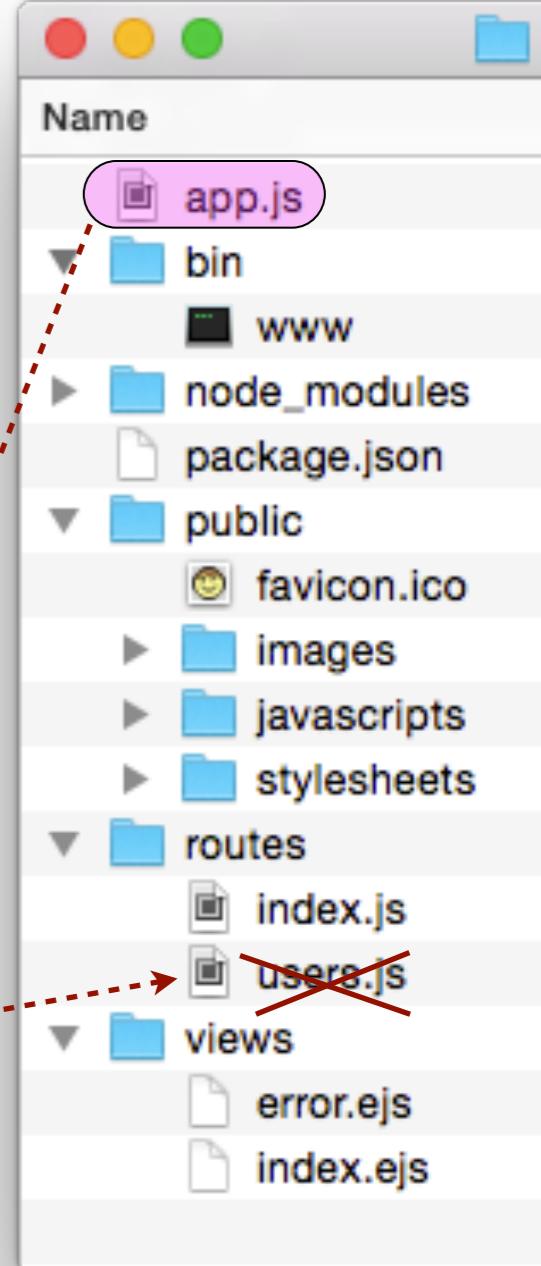
```

app.js
bin
www
node_modules
package.json
public
favicon.ico
images
javascripts
stylesheets
routes
index.js
users.js
views
error.ejs
index.ejs

```

# Quitar ruta /user

- ◆ express-generator genera **2 rutas**
  - La ruta “**/**” de acceso a la página de entrada
    - ◆ Esta ruta se reutiliza para la página principal
  - La ruta “**/users**” para la gestión de usuarios
    - ◆ Esta ruta **no se reutiliza** y la **borramos**
- ◆ Para borrar la ruta **/users** debemos
  - Borrar su fichero de rutas **routes/users.js**
  - Borrar en **app.js** la sentencia
    - ◆ **var users = require('./routes/users');**
  - Borrar en **app.js** la sentencia
    - ◆ **app.use('/users', users);**

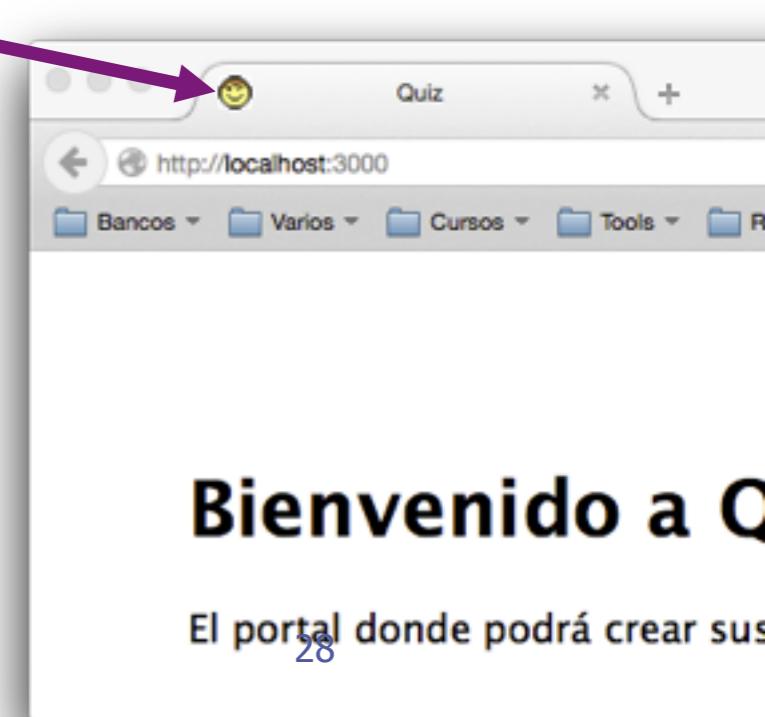


```
8 var routes = require('./routes/index');
9 var users = require('./routes/users');
10 .....
11 app.use('/', routes);
12 app.use('/users', users);
```

# Favicon

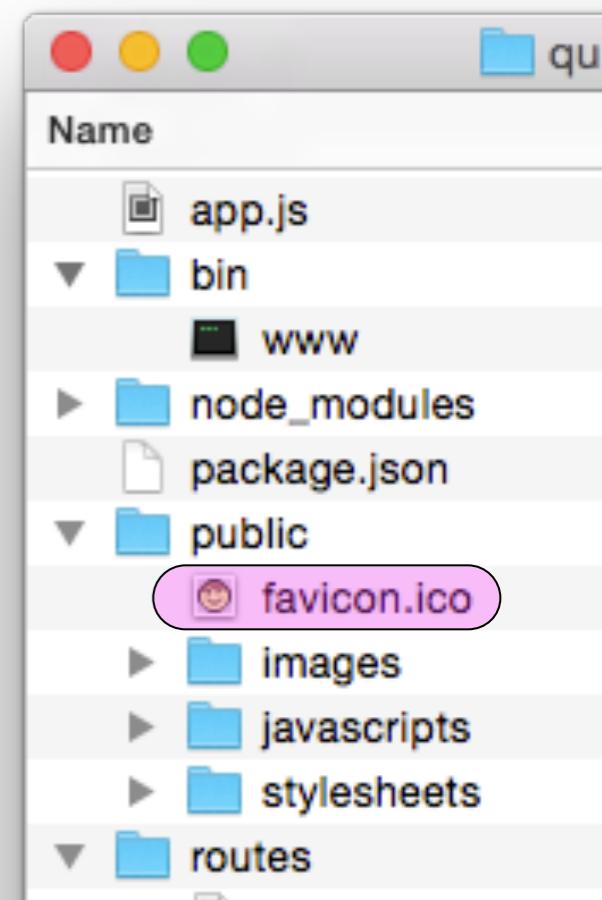


favicon.ico



- ◆ Se puede mostrar un icono en la pestaña de una página
  - Se denomina **favicon** (favourite icon) yes de 16x16 pixels
    - ◆ Se suele guardar en el fichero **favicon.ico**
    - Ejemplo de editor de favicon: <http://www.favicon.cc/>
- ◆ **app.js** incluye el MW **serve-favicon** que sirve un favicon
  - El MW se activa descomentando su instalación en app.js
    - ◆ Además hay que incluir el fichero “favicon.ico” en el directorio public

```
// uncomment after placing your favicon in /public
// app.use(favicon(__dirname + '/public/favicon.ico'));
app.use(favicon(__dirname + '/public/favicon.ico'));
app.use(logger('dev'));
```



- ◆ Hoy se recomienda incluir el favicon desde la página HTML
  - Con un enlace al ícono en la cabecera de la página
    - ◆ <link rel="icon" href="favicon.ico" type="image/x-icon" />

# Guardar versión GIT

◆ Para guardar los cambios debemos incluirlos en el índice (área de cambios)

- Con el comando: `git add .`

◆ Generamos una versión en la rama master

- Con el comando: `git commit -m "identificación de versión"`

◆ El repositorio `.git` contiene ahora las 2 primeras versiones

- podemos verlas con: `git log` o con `git log --oneline`

```
..$ git add . // añadimos todo lo contenido en directorio al area de cambios
```

```
// generamos versión identificada como: Primera página y favicon
```

```
..$ git commit -m "Primera página y favicon"
```

```
[master 984f3ee] Primera página y favicon
```

```
 5 files changed, 5 insertions(+), 15 deletions(-)
```

```
create mode 100644 public/favicon.ico
```

```
delete mode 100644 routes/users.js
```

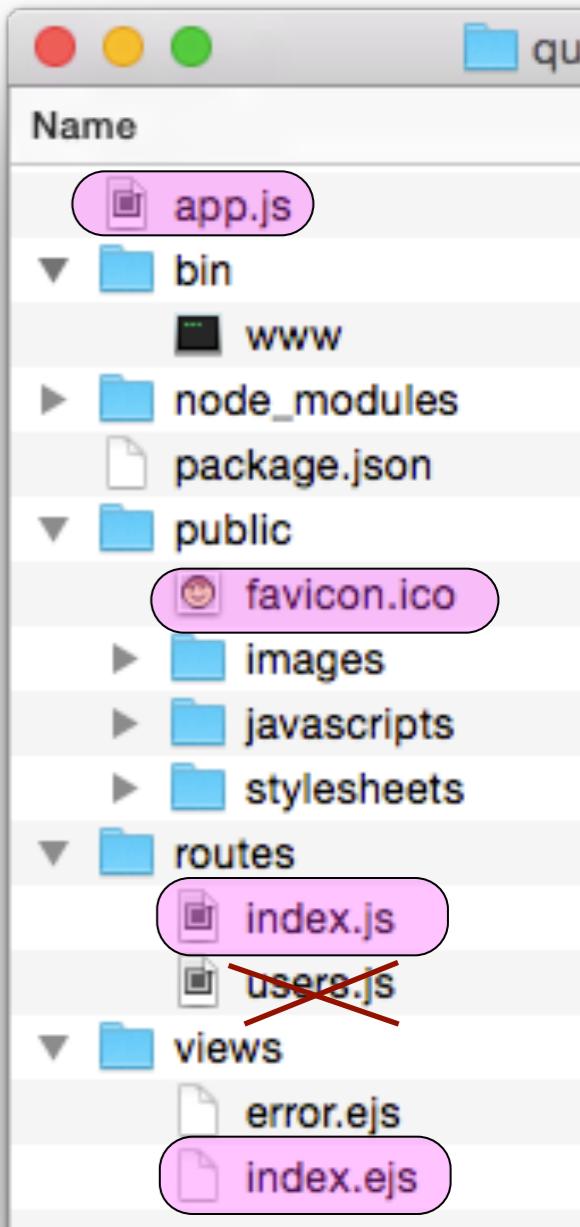
```
// listamos los commits en formato de una línea con
```

```
..$ git log --oneline
```

```
6fa18d0 Primera página y favicon
```

```
49b1f9c esqueleto express-generator
```

```
..$
```



```
// git status muestra el estado del índice o área de cambios

..git status // los cambios están sin indexar
Changes not staged for commit:
(use "git add/rm <file>..." to update what will be committed)
(use "git checkout -- <file>..." to discard changes in working directory)

    modified: app.js
    modified: routes/index.js
    deleted: routes/users.js
    modified: views/index.ejs

Untracked files:
(use "git add <file>..." to include in what will be committed)

    public/favicon.ico

no changes added to commit (use "git add" and/or "git commit -a")

..git add . // añadimos todos los cambios al índice

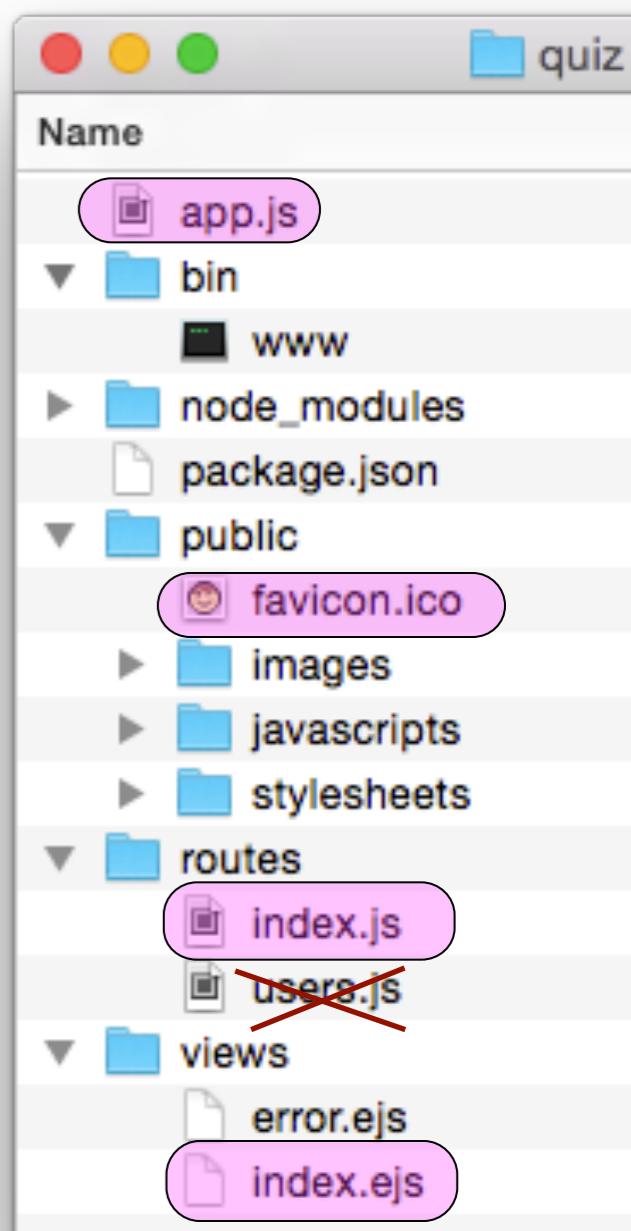
..git status // todos los cambios están en el índice
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)

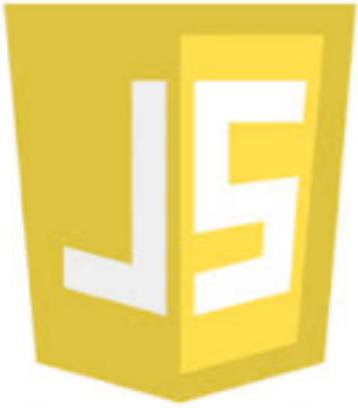
    modified: app.js
    new file: public/favicon.ico
    modified: routes/index.js
    deleted: routes/users.js
    modified: views/index.ejs

..git commit -m "Primera página y favicon" // generar versión
[favicon 83c5e74] Primera página y favicon
  5 files changed, 5 insertions(+), 15 deletions(-)
  create mode 100644 public/favicon.ico
  delete mode 100644 routes/users.js

..git status // la nueva versión no tiene cambios
nothing to commit, working directory clean
$
```

# git status: el índice





JavaScript



# Quiz 3: Primera pregunta

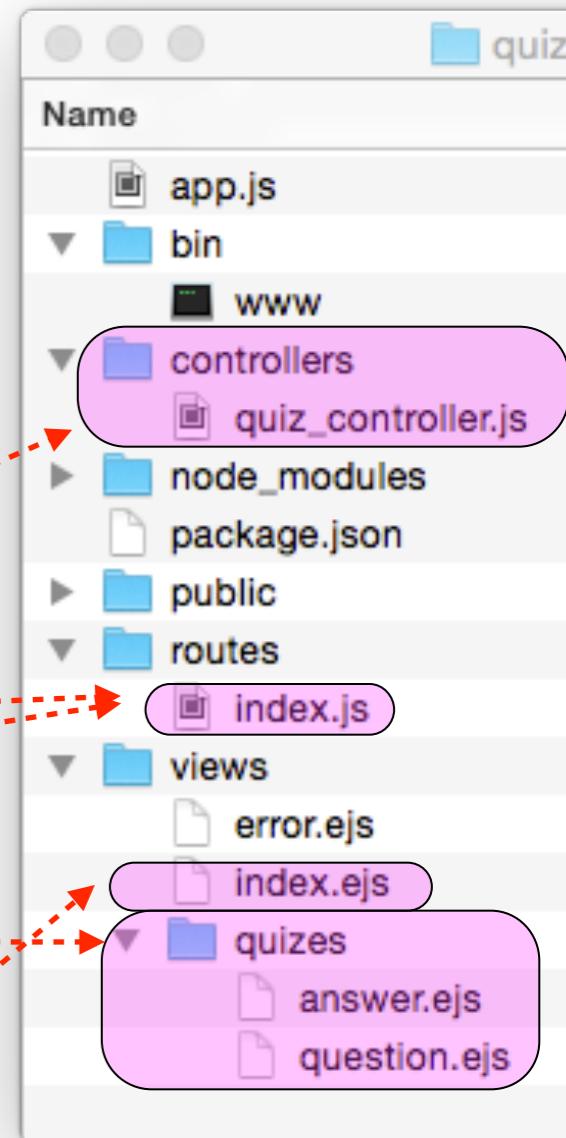
Juan Quemada, DIT - UPM

# Quiz 3: Primera pregunta

**Objetivo:** crear la primera pregunta “Capital de Italia” y su respuesta. Se debe crear para ello el directorio de controladores y quiz\_controller.js.

## ◆ Paso 1: Añadir MVC de pregunta y respuesta

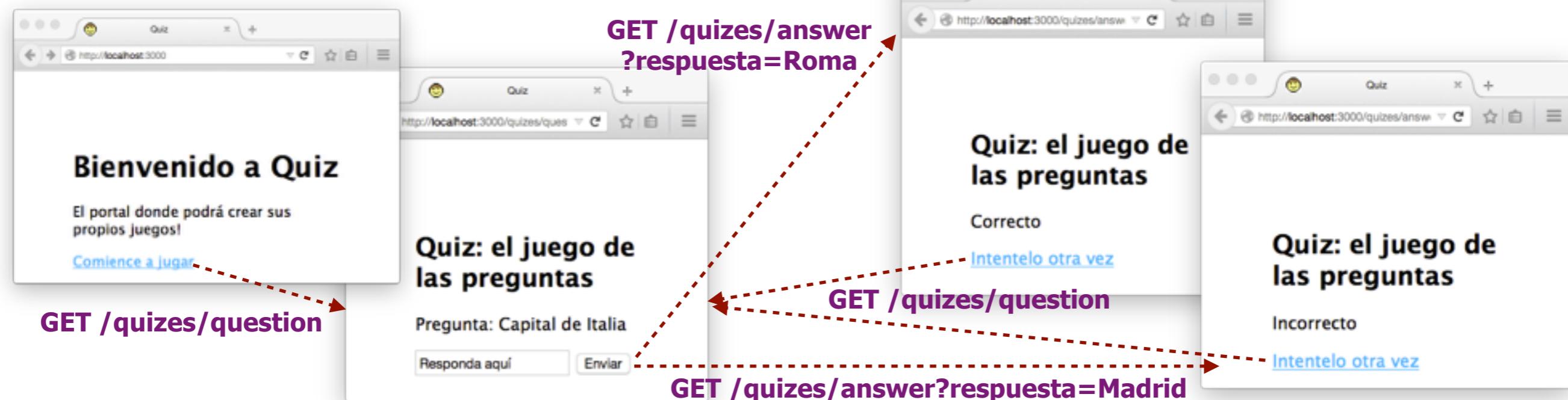
- a: Crear directorio **controllers** y el controlador **quiz\_controller.js**
- b: Importar **quiz\_controller.js** en **routes/index.js**
- c: Introducir nuevas rutas en el enrutador **routes/index.js**
  - Rutas: **GET /quizes/question** y **GET /quizes/answer**
- d: Crear directorio **view/quizes** y las vistas **question.ejs** y **answer.ejs**

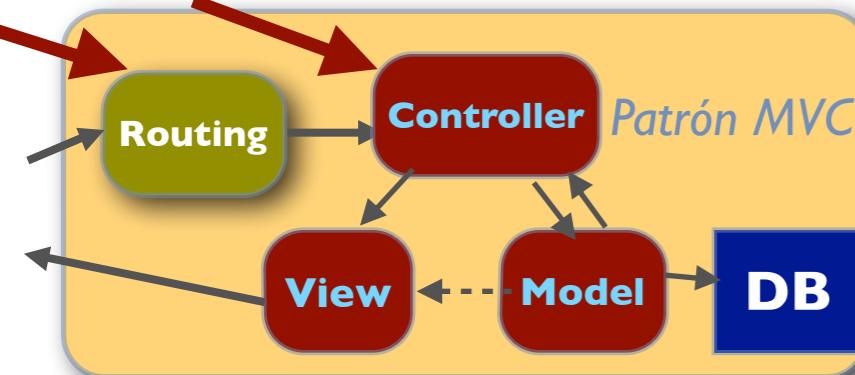


## ◆ Paso 2: Añadimos enlace a la pregunta en la página de entrada

- Así enlazamos la pregunta desde la página de entrada (hipermedia)

## ◆ Paso 3: Guardamos versión (**commit**) con git





```
app.js
```

```
UNREGISTERED
var routes = require('./routes/index');
var app = express();
.....
app.use('/', routes);
```

**app.js**

```
index.js
```

```
UNREGISTERED
var express = require('express');
var router = express.Router();

var quizController = require('../controllers/quiz_controller');

/* GET home page. */
router.get('/', function(req, res) {
  res.render('index', { title: 'Quiz' });
});

router.get('/quizes/question', quizController.question);
router.get('/quizes/answer', quizController.answer);

module.exports = router;
```

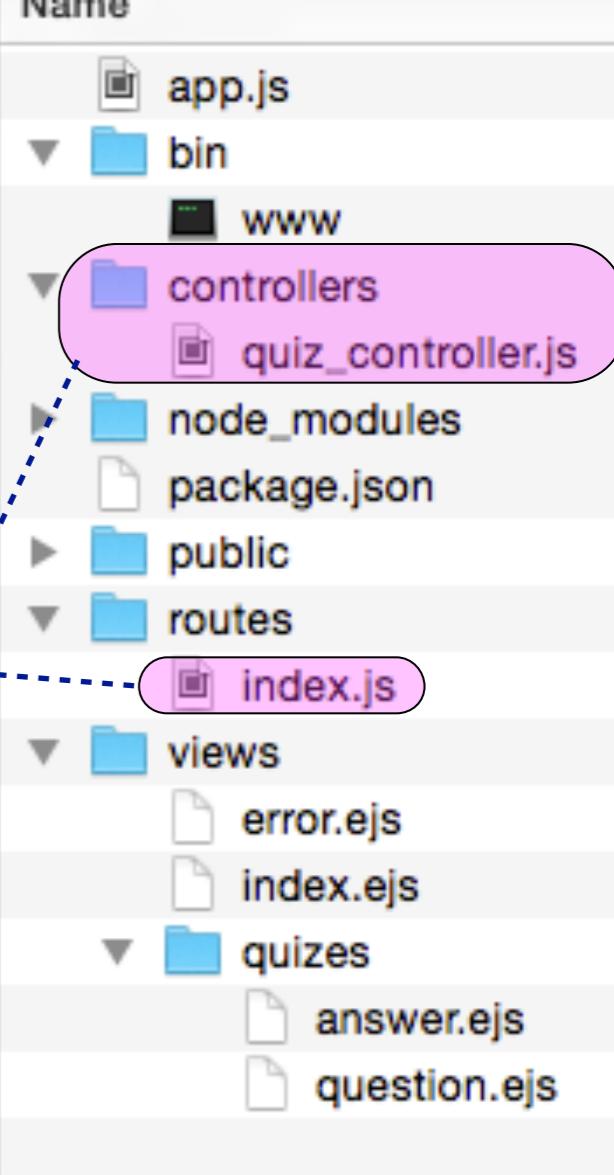
**Pasos 1b y 1c: rutas /quizes/....**

```
quz_controller.js
```

```
UNREGISTERED
// GET /quizes/question
exports.question = function(req, res) {
  res.render('quizes/question', {pregunta: 'Capital de Italia'});
};

// GET /quizes/answer
exports.answer = function(req, res) {
  if (req.query.respuesta === 'Roma'){
    res.render('quizes/answer', {respuesta: 'Correcto'});
  } else {
    res.render('quizes/answer', {respuesta: 'Incorrecto'});
  }
};
```

**Paso 1a: controllers/quiz\_controller.js**



**Quiz: el juego de las preguntas**

Pregunta: Capital de Italia

Responda aquí  Enviar

**Quiz: el juego de las preguntas**

Incorrecto

[Intentelo otra vez](#)

```
// GET /quizes/question
exports.question = function(req, res) {
  res.render('quizes/question', {pregunta: 'Capital de Italia'});
};

// GET /quizes/answer
exports.answer = function(req, res) {
  if (req.query.respuesta === 'Roma') {
    res.render('quizes/answer', {respuesta: 'Correcto'});
  } else {
    res.render('quizes/answer', {respuesta: 'Incorrecto'});
  }
};
```

### Paso 1a: controllers/quiz\_controller.js

```
www
  controllers
    quiz_controller.js
  node_modules
  package.json
  public
  routes
    index.js
  views
    error.ejs
    index.ejs
  quizzes
    answer.ejs
    question.ejs
```

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8"><title>Quiz</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h2>Quiz: el juego de las preguntas</h2>

    <form method="get" action="/quizes/answer">
      Pregunta: <%= pregunta %> <p>
      <input type="text" name="respuesta" value="Responda aquí"/>
      <input type="submit" value="Enviar">
    </form>
  </body>
</html>
```

### Paso 1d: views/quizes/question.ejs

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Quiz</title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h2>Quiz: el juego de las preguntas</h2>

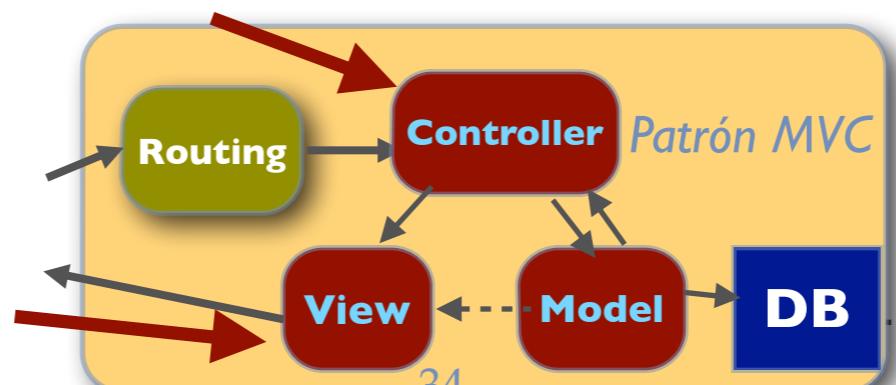
    <%= respuesta %><p>
    <a href="/quizes/question"> Intentelo otra vez </a>
  </body>
</html>
```

### Paso 1d: views/quizes/answer.ejs

Quiz: el juego de las preguntas

Pregunta Capital de Italia

Responda aquí Enviar



34

Quiz: el juego de las preguntas

Incorrecto

[Intentelo otra vez](#)

# Paso 2: Modificación de la página de entrada

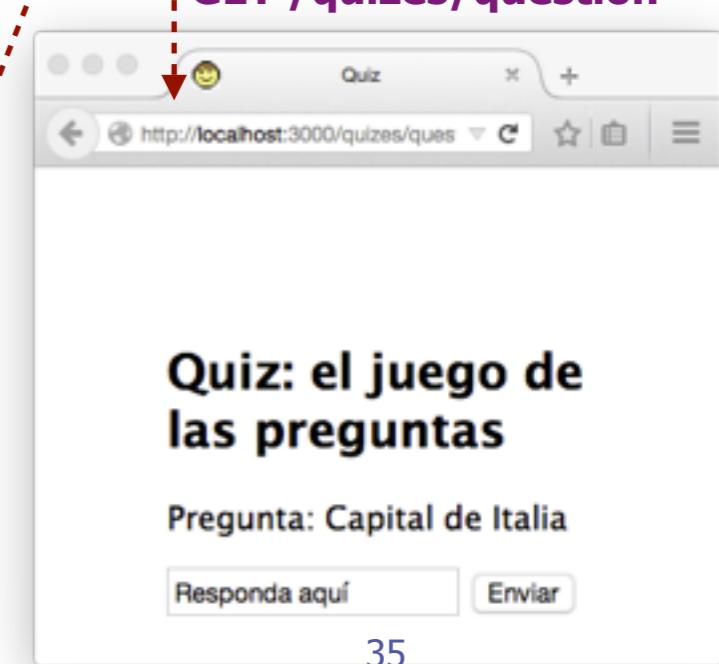
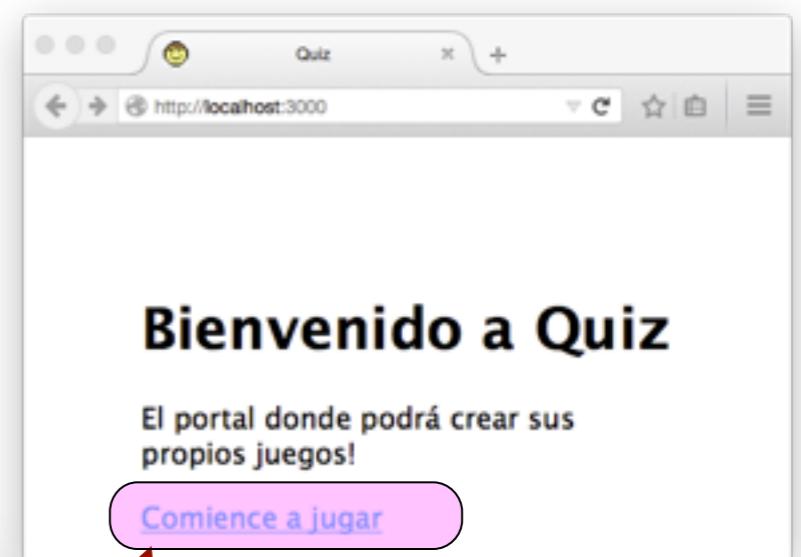
- ◆ Añadimos un enlace a la pregunta en la página de entrada

- En la vista **views/index.ejs**

- ◆ Así tenemos un servicio hypermedia

- Donde podemos navegar entre páginas
    - ◆ A través de los enlaces

```
<!DOCTYPE html>
<html>
  <head>
    <title><%= title %></title>
    <link rel='stylesheet' href='/stylesheets/style.css' />
  </head>
  <body>
    <h1>Bienvenido a <%= title %></h1>
    <p>El portal donde podrá crear sus propios juegos!</p>
    <p><a href="/quizes/question">Comience a jugar</a></p>
  </body>
</html>
```



# Guardar versión y corregirla

## ◆ Los cambios se pueden guardar con la opción **-a** de **git-commit**

- Opción -a equivale realizar antes “**git add .**”: **git commit -a -m “identificación de versión”**

## ◆ Si debemos corregir un error después de cerrar la versión

- La actualizamos con la opción “**--amend**” : **git commit --amend**

```
..$ git commit -a -m “Primera pregunta” // opción -a equivale a “git add .”
```

```
..$ git log --oneline // Hay tres versiones generadas
```

```
9600670 Primera pregunta
```

```
984f3ee Primera página y favicon
```

```
9971c09 esqueleto express-generator
```

```
..$ ..... // si descubrimos un error, lo corregimos y cuando hemos  
// que el error esta corregido y la versión es la correcta
```

```
..$ git commit --amend -m “Primera pregunta” // Actualizar versión con las  
// correcciones introducidas.
```

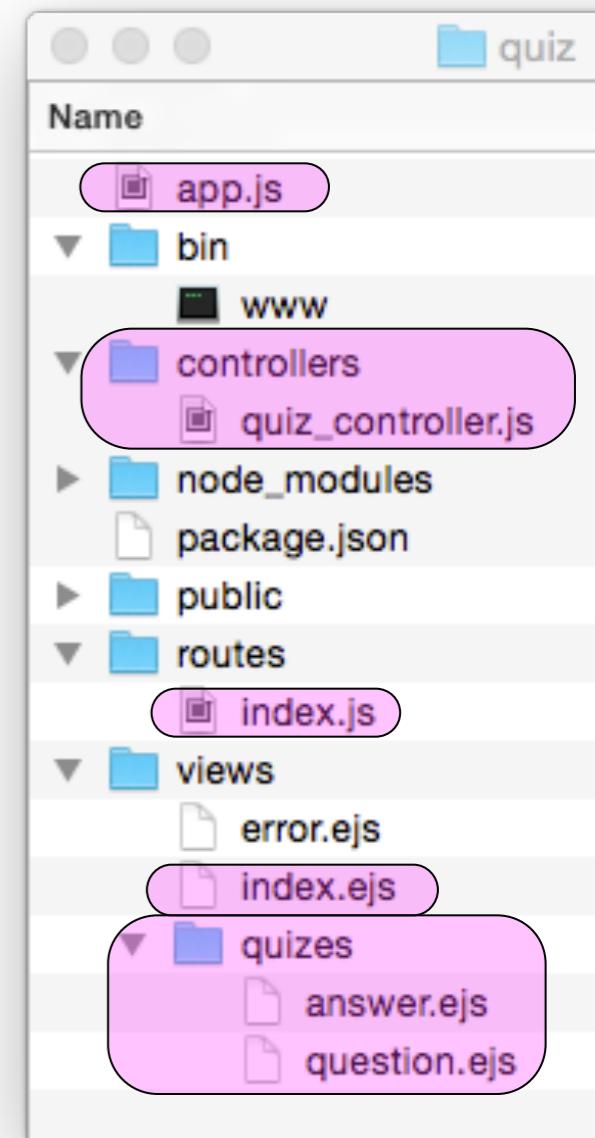
```
..$ git log --oneline // Siguen las mismas versiones
```

```
9f1ab23 Primera pregunta
```

```
984f3ee Primera página y favicon
```

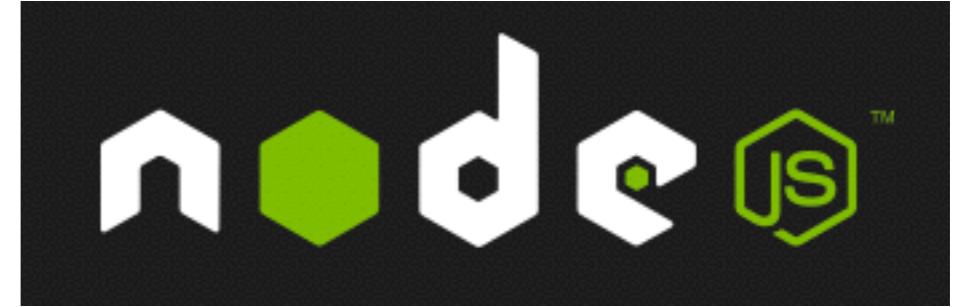
```
9971c09 esqueleto express-generator
```

```
..$
```





JavaScript



# Quiz 4: Marco de la aplicación

Juan Quemada, DIT - UPM

# Quiz 4: Marco de la aplicación

**Objetivo:** Introducir un marco (layout.ejs) común a todas las vistas para no tener replicar el mismo código HTML en todas las vistas utilizando el middleware **express-partials** que soporta vistas parciales.

## ◆ Paso 1: Instalar el paquete **express-partials**

- Añade vistas parciales y permite incluir un marco (layout) único

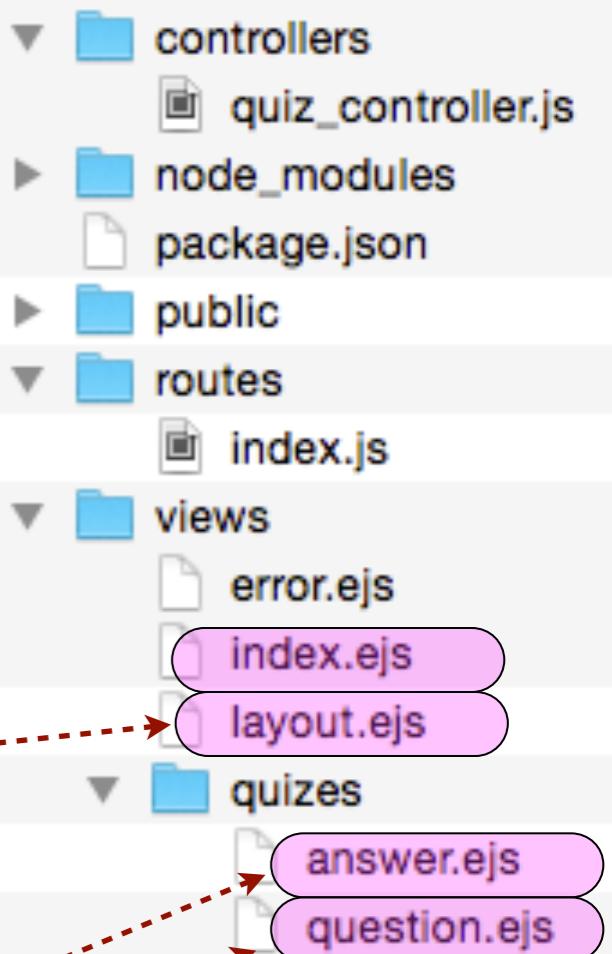
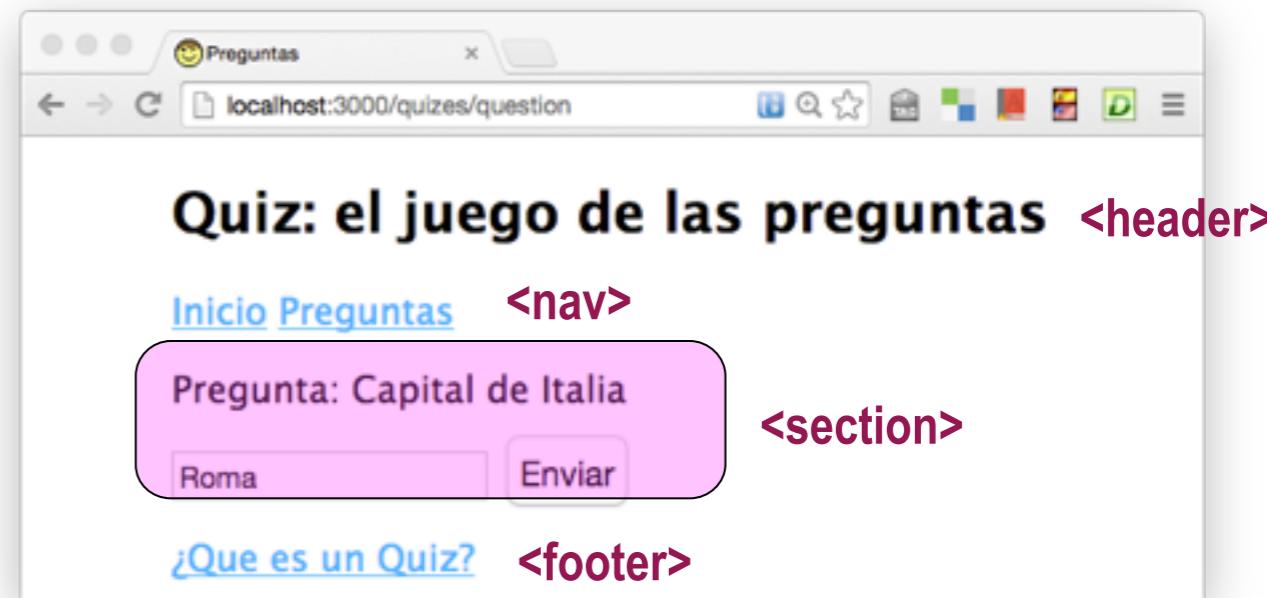
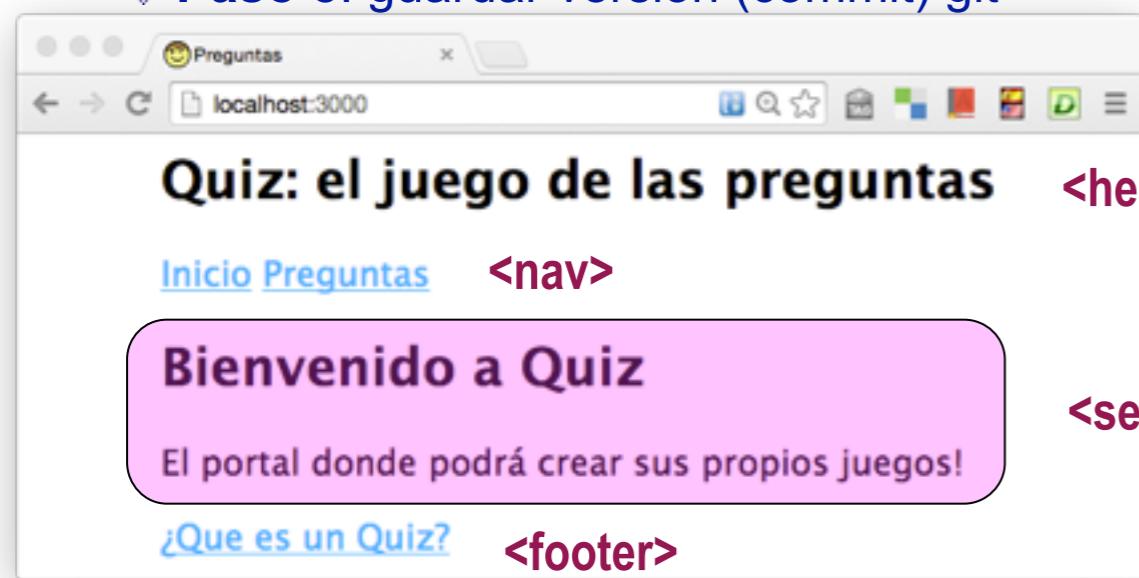
## ◆ Paso 2: Importar e instalar **express-partials** en **app.js**

## ◆ Paso 3: Añadir el marco único (**views/layout.ejs**) con las marcas HTML5

- Cabecera: **<header>**
- Navegación y enlaces: **<nav>**
- Vistas que cambian: **<section>**
- Pie de página: **<footer>**

## ◆ Paso 4: Se rehacen las vistas para renderizar en **<section>** solo el código que cambia

## ◆ Paso 5: guardar version (commit) git





## express-partials

Express 3.x Layout &amp; Partial support.

\$ npm install express-partials

Want to see pretty graphs? [Log in now!](#)

151 downloads in the last day

1 117 downloads in the last week

4 177 downloads in the last month

Last Published By slaskis

Version 0.2.0 last updated 3 months ago

Keywords

Repository <https://github.com/publicclass/express-partials>  
(undefined)Homepage <https://github.com/publicclass/express-partials>Bugs <https://github.com/publicclass/express-partials/issues>

Dependencies None

Dependents (11) authstarter, blogbyvista, express-device, mdwiki, node-upptime, node-web-scraping, nodemailer, ripple-core, turboexpress, waybo, yolo-server

```
{
  "name": "quiz",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "dependencies": {
    "body-parser": "~1.8.1",
    "cookie-parser": "~1.3.3",
    "debug": "~2.0.0",
    "ejs": "~0.8.5",
    "express": "~4.9.0",
    "express-partials": "^0.3.0",
    "morgan": "~1.3.0",
    "serve-favicon": "~2.1.3"
  }
}
```

# Paso 1: express-partials

## ◆ express-partials es un middleware

- Lo instalamos con la opción **--save**
  - ◆ Así se guarda la dependencia en **package.json**
  - ◆ **Sin opción —save:** no se guarda la dependencia

## ◆ package.json

- Debe incluir todas las dependencias los módulos usados
  - ◆ Así es posible reinstalar Quiz con: “**npm install**”
- Quiz está probado con: **express-partials@0.3.0**

## ◆ Una vez instalado tenemos que modificar

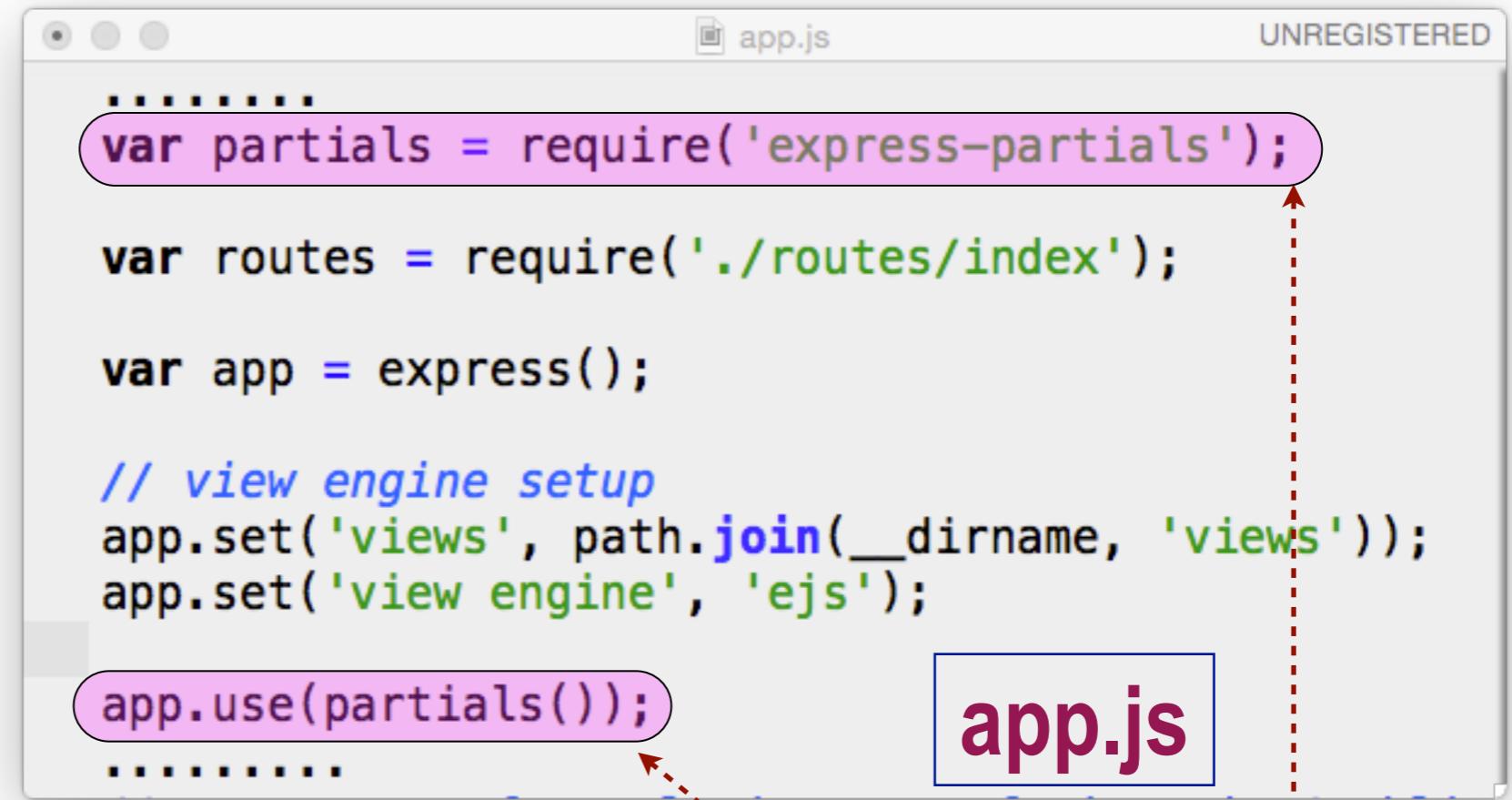
- **app.js** para incluir el middleware
- **las vistas** para incluir el marco (layout)

```
// instala express-partials en directorio app/node_modules
// opción --save guarda dependencia en package.json

// instala versión del proyecto en GITHUB y transparencias
..$ npm install --save express-partials@0.3.0

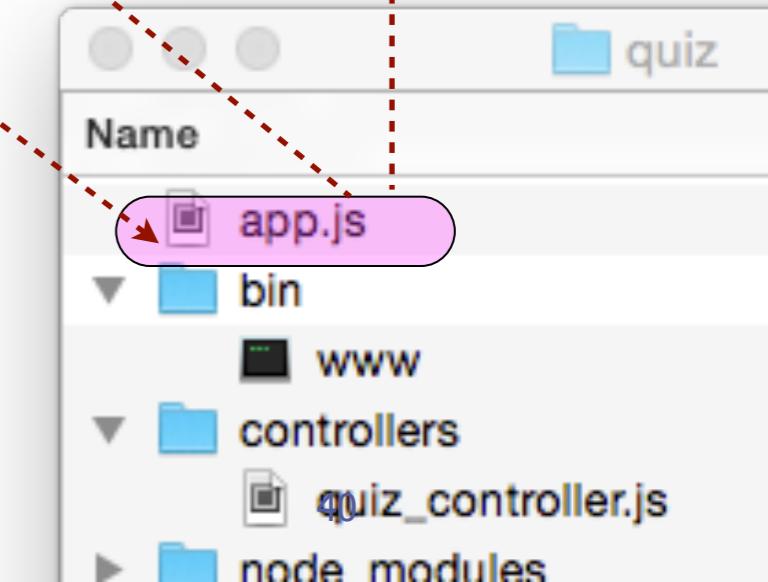
// instala última versión estable en www.npmjs.org
..$ npm install --save express-partials
```

## Paso 2: Instalar middleware

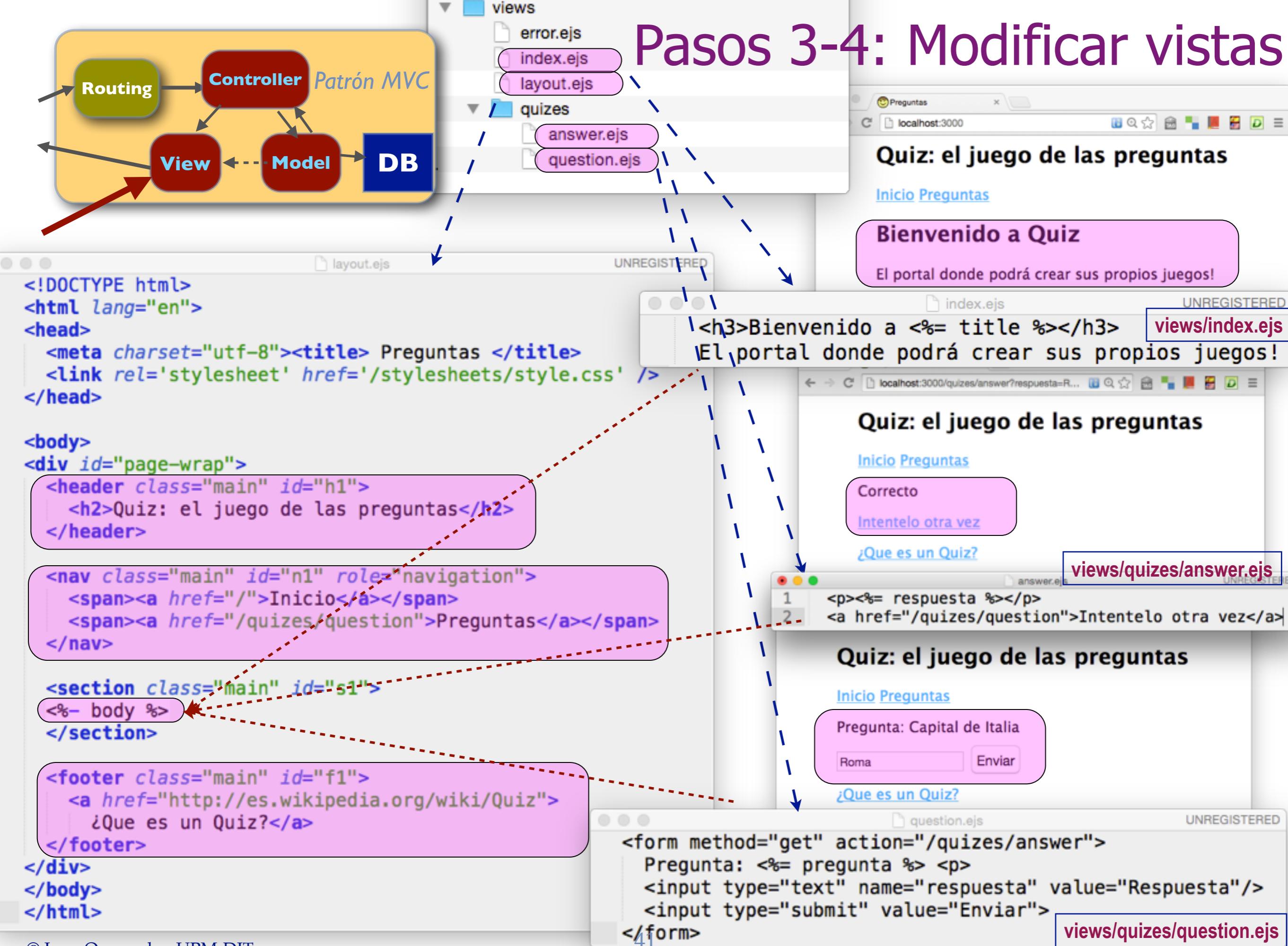


```
.....  
var partials = require('express-partials');  
  
var routes = require('./routes/index');  
  
var app = express();  
  
// view engine setup  
app.set('views', path.join(__dirname, 'views'));  
app.set('view engine', 'ejs');  
  
app.use(partials());  
.....
```

- ◆ El módulo **express-partials** importa una factoría que debe invocarse con () para generar el MW a instalar
  - Una vez instalado en el proyecto (transp. anterior)
    - ◆ Hay que **importarlo e instalarlo en app.js**

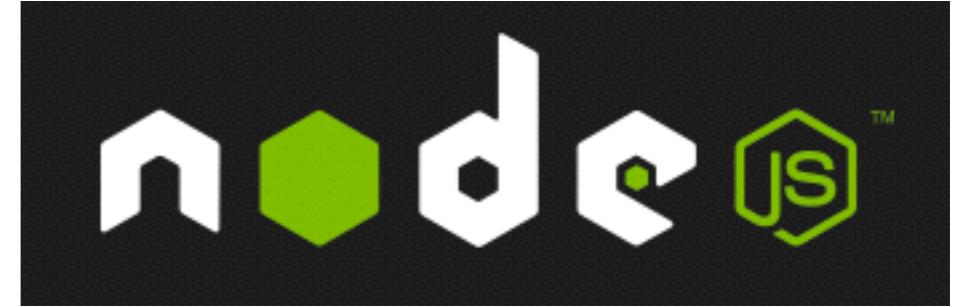


# Pasos 3-4: Modificar vistas





JavaScript



# Quiz 5: CSS adaptable a móviles

Juan Quemada, DIT - UPM

# Quiz 5: CSS adaptable a móviles

**Objetivo:** Introducir estilos CSS que adapten la visualización de las páginas de Quiz a PCs y móviles utilizando un estilo adaptativo y **media-queries**.

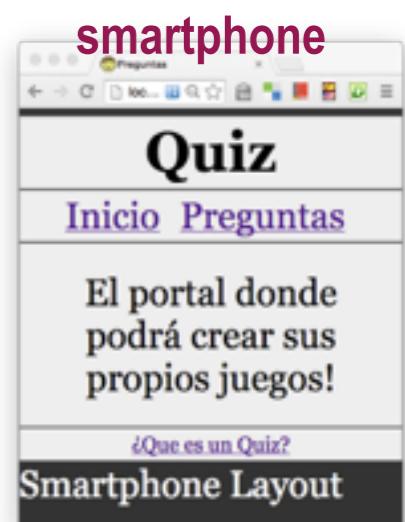
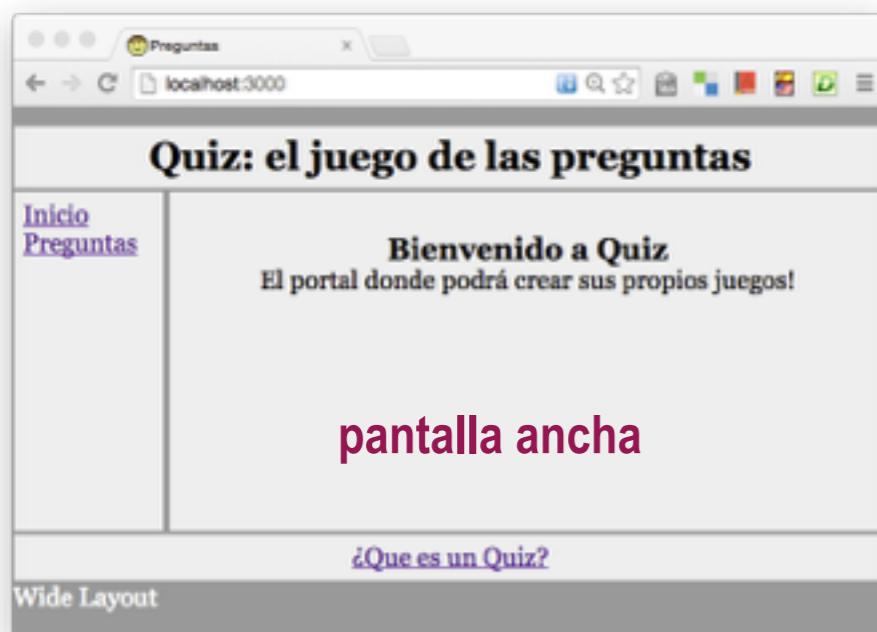
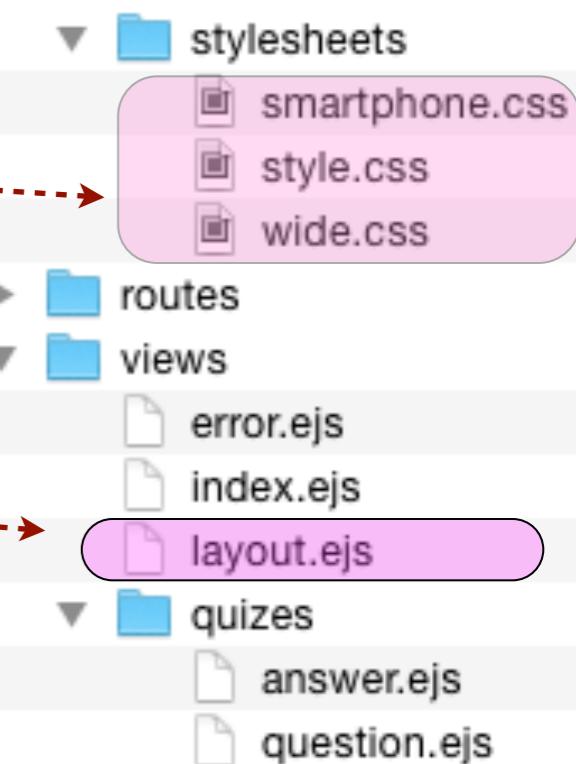
## ◆ Paso 1: Se definen los 3 estilos CSS a utilizar

- Estilo básico adaptable con facilidad
- Reglas CSS adicionales para pantalla ancha
- Reglas CSS adicionales para smartphone

## ◆ Paso 2: Se modifica el marco (**views/layout.ejs**)

- para que utilice cada estilo en la pantalla asociada
  - ◆ Se utilizan **media-queries** para determinar cuando aplicar cada estilo

## ◆ Paso 3: guardar version (commit) git



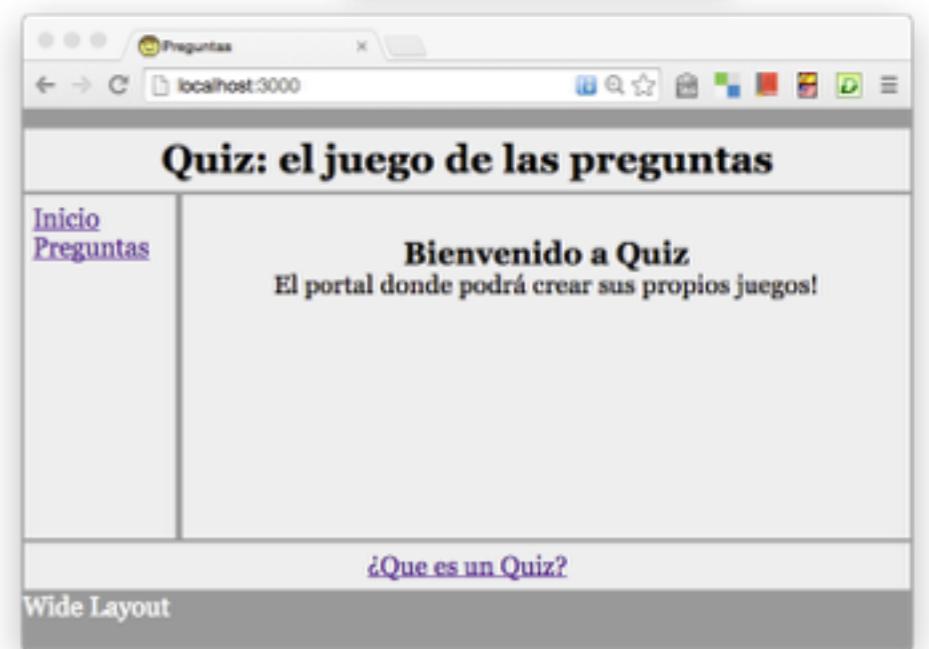
# Diseño Web adaptable



- ◆ El diseño adaptable (**responsive**) busca uniformizar el acceso
  - desde cualquier dispositivo, ya sea PC, móvil, tableta, smartTV, ....
    - optimizando la experiencia de usuario en cada uno de ellos
- ◆ El diseño adaptable utiliza una única definición HTML de interfaz o página
  - que se adapta al dispositivo con diferentes ficheros CSS
- ◆ La adaptación al tamaño de pantalla
  - Se puede realizar con **media queries** o con **JavaScript/DOM**
- ◆ Algunas referencias
  - Tutorial: <http://www.alistapart.com/articles/responsive-web-design/>
  - Ejemplos: <http://css-tricks.com/resolution-specific-stylesheets/>

# CSS adaptable

- ◆ Vamos a utilizar una única página HTML
  - con un estilo CSS básico
    - ◆ al que se añadirá con media queries
      - estilos CSS adicionales
        - para adaptar a PC y teléfono
- ◆ Media query para página pequeña (smartphone)
  - **media="only screen and (max-width: 480)"**
- ◆ Media query para pantalla ancha (PC)
  - **media= "only screen and (min-width: 530) and (min-device-width: 481)"**



# El atributo media de <link ..>

- ◆ El atributo **media** de la marca <link> de CSS3 detecta el dispositivo utilizado
  - [http://www.w3schools.com/tags/att\\_link\\_media.asp](http://www.w3schools.com/tags/att_link_media.asp)
- ◆ El atributo **media** condiciona la carga del fichero de estilos al dispositivo utilizado
  - **Smartphone** con pantalla menor de 480 px
    - ◆ **media="only screen and (max-device-width: 480)"**
  - **Pantalla ancha** con el navegador en una ventana de anchura mayor a 600 px
    - ◆ **media= "only screen and (max-width: 600) and (min-device-width: 481)"**
- ◆ Añadimos al fichero **layout.ejs** estilos con **media queries** de carga condicional

The screenshot shows a code editor window titled "layout.ejs" containing the following HTML code:

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8"><title> Quiz </title>
<link rel='stylesheet' type='text/css' href='/stylesheets/style.css' />
<link rel='stylesheet'
      type='text/css'
      media='only screen and (min-width: 530px) and (min-device-width: 481px)'
      href='/stylesheets/wide.css' />
<link rel='stylesheet'
      type='text/css'
      media='only screen and (max-width: 480px)'
      href='/stylesheets/smartphone.css' />
</head>
```

To the right of the code editor is a file browser window showing the project structure:

- UNREGISTERED
- layout.ejs (highlighted with a pink rounded rectangle)
- stylesheets (highlighted with a pink rounded rectangle)
  - smartphone.css
  - style.css
  - wide.css
- routes
- views
  - error.ejs
  - index.ejs
- quizzes
  - answer.ejs
  - question.ejs

A red arrow points from the "layout.ejs" entry in the file browser back towards the "layout.ejs" file in the code editor.

```

<!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8"><title> Preguntas </title>
  <link rel='stylesheet' href='/stylesheets/style.css' />
  <link rel='stylesheet'
    type='text/css'
    media='only screen and (min-width: 530px) and (min-device-width: 481px)'
    href='/stylesheets/wide.css' />
  <link rel='stylesheet'
    type='text/css'
    media='only screen and (max-width: 480px)'
    href='/stylesheets/smartphone.css' />
</head>

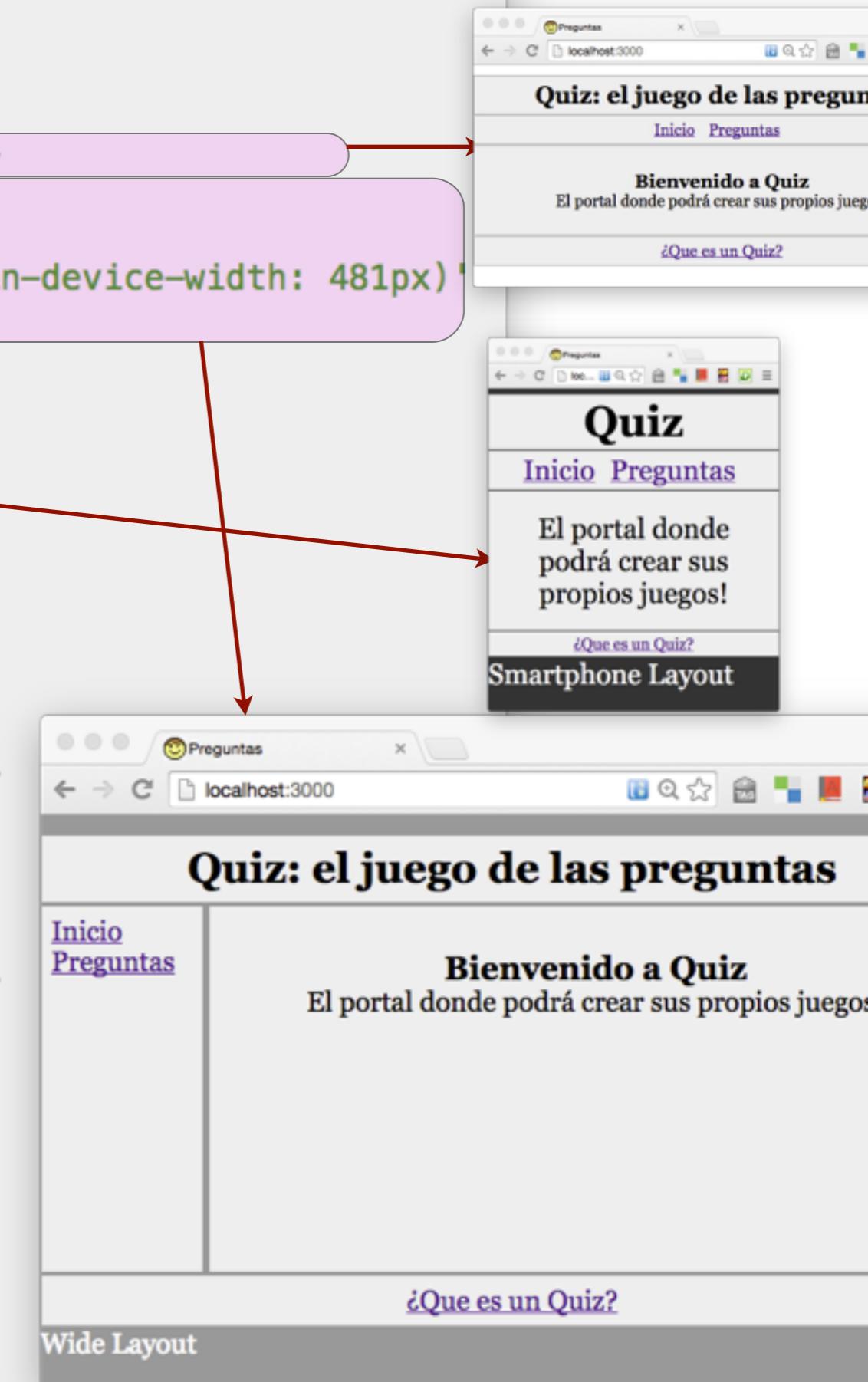
<body>
<div id="page-wrap">
  <header class="main" id="h1">
    <h2>Quiz<span>: el juego de las preguntas</span></h2>
  </header>

  <nav class="main" id="n1" role="navigation">
    <span><a href="/">Inicio</a></span>
    <span><a href="/quizes/question">Preguntas</a></span>
  </nav>

  <section class="main" id="s1">
    <div><%-- body %></div>
  </section>

  <footer class="main" id="f1">
    <a href="http://es.wikipedia.org/wiki/Quiz">
      ¿Que es un Quiz?</a>
  </footer>
</div>
</body>
</html>

```



Paso 2: layout.ejs

# Paso 1a: style.css

The diagram illustrates the relationship between the source code, the browser preview, and the CSS styles.

**HTML Structure:**

```
<body>
<div id="page-wrap">
  <header class="main" id="h1">
    <h2>Quiz<span>: el juego de las preguntas</span></h2>
  </header>

  <nav class="main" id="n1" role="navigation">
    <span><a href="/">Inicio</a></span>
    <span><a href="/quizes/question">Preguntas</a></span>
  </nav>

  <section class="main" id="s1">
    <div><% body %></div>
  </section>

  <footer class="main" id="f1">
    <a href="http://es.wikipedia.org/wiki/Quiz">
      ¿Que es un Quiz?
    </a>
  </footer>
</div>
</body>
</html>
```

**Browser Preview:**

localhost:3000

Quiz: el juego de las preguntas

Inicio Preguntas

Bienvenido a Quiz  
El portal donde podrá crear sus propios juegos!

¿Que es un Quiz?

**style.css:**

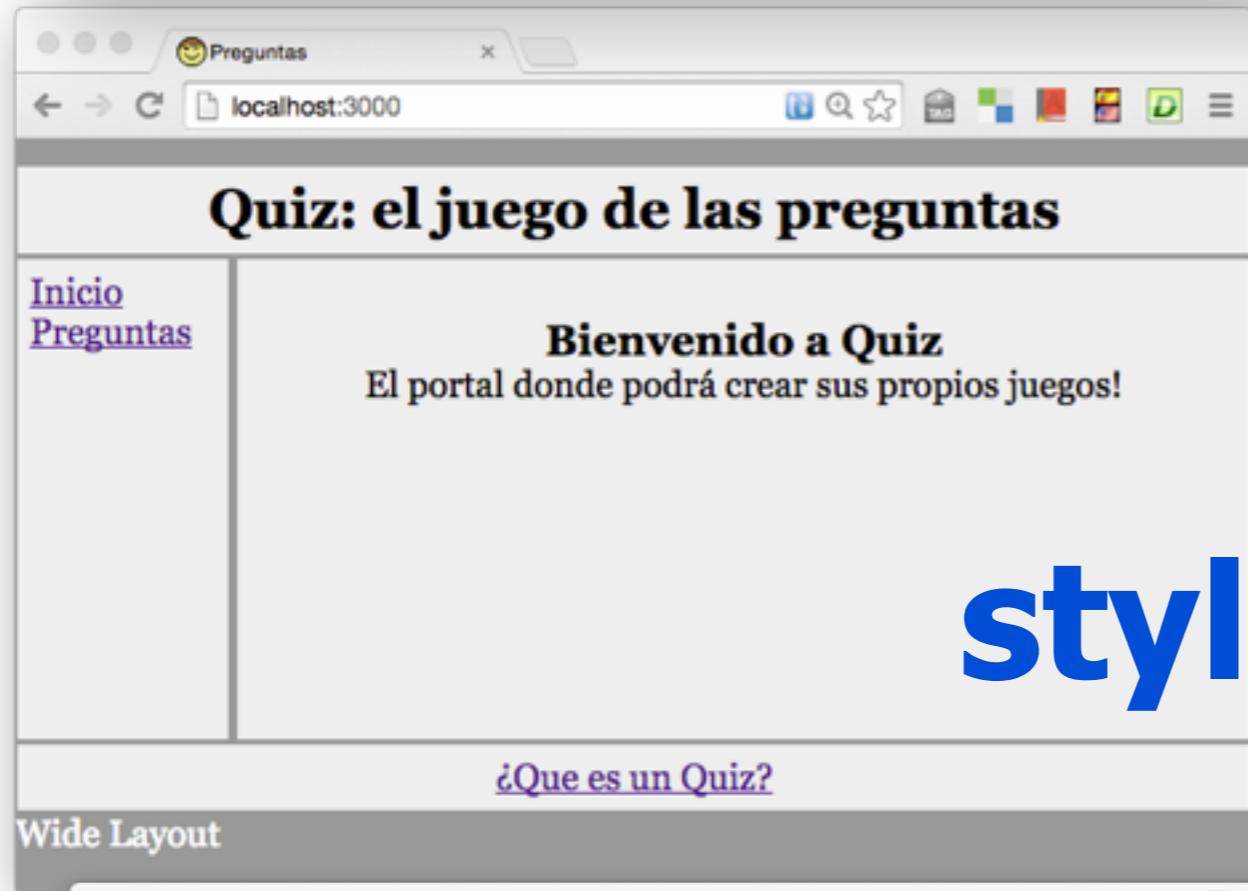
```
* { margin: 0; padding: 0; }
body { font: 16px Georgia, serif; }

#page-wrap { margin: 2% auto; color: white; }
#page-wrap > .main { border: 1px solid #999; padding: 1%; margin-bottom: 1px; color: black; }

section > div { padding: 20px; }
header, nav, section { background: #eee; text-align: center; }
footer { background: #eee; text-align: center; }
nav span { margin: 0px 10px 0px 0px; }
```

# Paso 1b: wide.css

```
wide.css  
UNREGISTERED  
body { background: #999; }  
  
#page-wrap:after { content: "Wide Layout"; }  
  
nav { float: left; width: 15%; height: 200px; }  
nav span { margin: 0px 100px 0px 0px; }  
section { float: right; width: 80%; height: 200px; }  
footer { clear: both; display: block; }
```



# style.css + wide.css

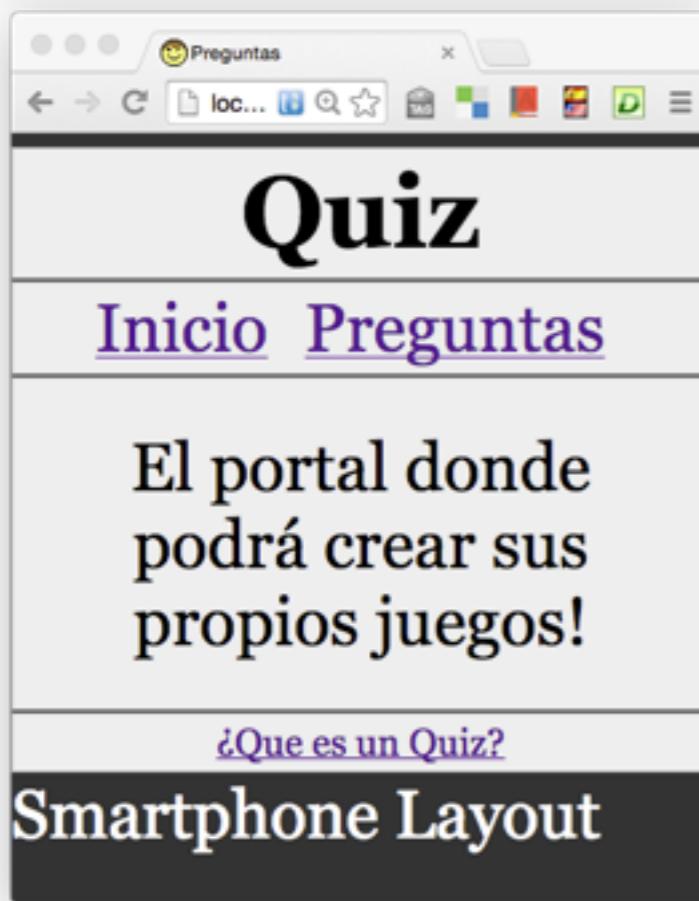
```
app.html  
UNREGISTERED  
<link rel='stylesheet'  
      media='only screen and (min-width: 530px) and (min-device-width: 481px)'  
      href='/css/wide.css' />
```

# Style.css + Smartphone.css

```
smartphone.css UNREGISTERED
body { background: #333; }

#page-wrap:after { content: "Smartphone Layout"; }

h2 span, section h3 { display: none; }
body { font-size: xx-large; }
footer, input { font-size: large; }
```



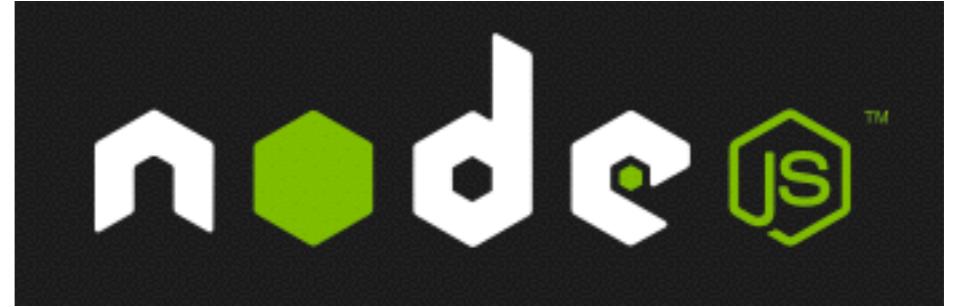
Paso 1c:  
smatphone.css



```
app.html UNREGISTERED
<link rel='stylesheet' media='only screen and (max-width: 480px)' href='/css/smartisan.css' />
```



JavaScript



# Quiz 6: Despliegue en Heroku

Juan Quemada, DIT - UPM



# Quiz 6: Despliegue en Heroku

**Objetivo:** desplegar Quiz en la nube para hacerla accesible a otros usuarios que se quieran conectar con un navegador desde cualquier punto de Internet.

Quiz es una aplicación de servidor que puede desplegarse en un servidor público. La vamos a desplegar en Heroku, que suministra un servicio PaaS (Platform as a Service) para node.js, con un despliegue muy sencillo y gratuito para aplicaciones con prestaciones limitadas.

Un servicio PaaS pone a nuestra disposición servidores virtuales con soporte para desplegar remotamente una aplicación de servidor en node.js.

Tutorial de despliegue con node.js:

<https://devcenter.heroku.com/articles/getting-started-with-nodejs>

- ◆ **Paso 1:** Crear una cuenta en Heroku: <http://www.heroku.com>
- ◆ **Paso 2:** Instalar Heroku Toolbelt
- ◆ **Paso 3:** Crear una aplicación en Heroku donde desplegar Quiz
- ◆ **Paso 4:** Adaptar la aplicación Quiz para despliegue en Heroku
- ◆ **Paso 5:** Versión git y despliegue de Quiz en Heroku

# Cuenta en Heroku y Heroku Toolbelt

- ◆ Heroku: servicio en la nube de tipo PaaS
  - Publica directamente aplicaciones en node.js, PHP, RoR, Python, ...
- ◆ Paso 1: debemos crear una cuenta en Heroku para publicar Quiz en la nube
  - <https://www.heroku.com/>
- ◆ Paso 2: instalar Heroku Toolbelt, las herramientas de gestión de Heroku
  - <https://toolbelt.heroku.com>
- ◆ Heroku Toolbelt instala en nuestro ordenador 2 comandos
  - **heroku** gestiona el despliegue de aplicaciones desde nuestro PC
  - **foreman** ejecuta localmente aplicaciones adaptadas a Heroku
  - Además necesita **git** para desplegar y si no está instalado debe instalarse



# Comandos heroku y foreman

```
venus-5:quiz jq$  
venus-5:quiz jq$ heroku --help  
Commands:  
foreman check          # Validate your application's Procfile  
foreman export FORMAT LOCATION # Export the application to another ...  
foreman help [COMMAND]    # Describe available commands or one...  
foreman run COMMAND [ARGS...] # Run a command using your applicati...  
foreman start [PROCESS]     # Start the application (or a specif...  
foreman version          # Display Foreman gem version  
  
Options:  
-f, [--procfile=PROCFILE] # Default: Procfile  
-d, [--root=ROOT]        # Default: Procfile directory  
  
venus-5:quiz jq$ █
```

```
venus-5:quiz jq$  
venus-5:quiz jq$ heroku --help  
Your version of git is 1.9.3. Which has serious security vulnerabilities.  
More information here: https://blog.heroku.com/archives/2014/12/23/update  
_your_git_clients_on_windows_and_os_x  
Usage: heroku COMMAND [--app APP] [command-specific-options]  
  
Primary help topics, type "heroku help TOPIC" for more details:  
  
addons      # manage addon resources  
apps        # manage apps (create, destroy)  
auth         # authentication (login, logout)  
config       # manage app config vars  
domains     # manage custom domains  
logs         # display logs for an app  
ps           # manage dynos (dynos, workers)  
releases    # manage app releases  
run          # run one-off commands (console, rake)  
sharing      # manage collaborators on an app  
  
Additional topics:  
  
certs        # manage ssl endpoints for an app  
drains       # display drains for an app  
features     # manage optional features  
fork         # clone an existing app  
git          # manage git for apps  
help         # list commands and display help  
keys         # manage authentication keys  
labs          # manage optional features  
maintenance # manage maintenance mode for an app  
members      # manage membership in organization accounts  
orgs         # manage organization accounts  
pg           # manage heroku-postgresql databases  
pgbackups   # manage backups of heroku postgresql databases  
plugins      # manage plugins to the heroku gem  
regions      # list available regions  
stack        # manage the stack for an app  
status       # check status of heroku platform  
twofactor    #  
update       # update the heroku client  
version      # display version  
  
venus-5:quiz jq$ █
```

# Aplicaciones Heroku

- ◆ Heroku es un **servicio de tipo PaaS** (Platform as a Service)
  - Permite instalar aplicaciones en node.js (y otros) que se ejecutan en la nube
    - ◆ El servicio es gratuito para aplicaciones con recursos limitados
    - Una sola maquina virtual (1 Dynos), 300 MBytes HD, ....
- ◆ **Aplicaciones:** se ejecutan en una o más máquinas virtuales, denominadas Dynos
  - **Dynos:** programa equivalente a una máquina servidora (ordenador físico)
    - ◆ Un Dynos se ejecuta en un contenedor Linux ligero y aislado de otros
- ◆ Las aplicaciones en Heroku se gestionan remotamente de 2 formas equivalentes
  - Con el comando “**heroku**” de Heroku Toolbelt
  - Accediendo al **Dashboard** de Heroku con un navegador
- ◆ Cada aplicación Heroku tiene un **repositorio remoto git asociado**
  - Para desplegar en la nube debemos enviar el código a dicho repositorio

# Paso 3: Crear aplicación Heroku con Toolbelt

```
// Después de crear una cuenta en Heroku e instalar Heroku Toolbelt en  
// nuestro ordenador local podemos crear y desplegar aplicaciones en Heroku
```

```
..$ heroku login // Nos conectamos a nuestra cuenta en Heroku con las mismas credenciales  
Enter your Heroku credentials.  
Email: jquemada@dit.upm.es  
Password (typing will be hidden):  
Authentication successful.
```



```
// Como la conexión debe ser segura, si no tenemos una clave pública creada, nos puede  
// preguntar además si la crea y la sube a Heroku (para que la comunicación sea segura)  
//           -> se debe contestar que si
```

```
..$ heroku create // crea una aplicación en heroku con un nombre aleatorio: pure-reaches-3917  
Creating pure-reaches-3917... done, stack is cedar-14  
https://pure-reaches-3917.herokuapp.com/ | https://git.heroku.com/pure-reaches-3917.git  
Git remote heroku added!
```

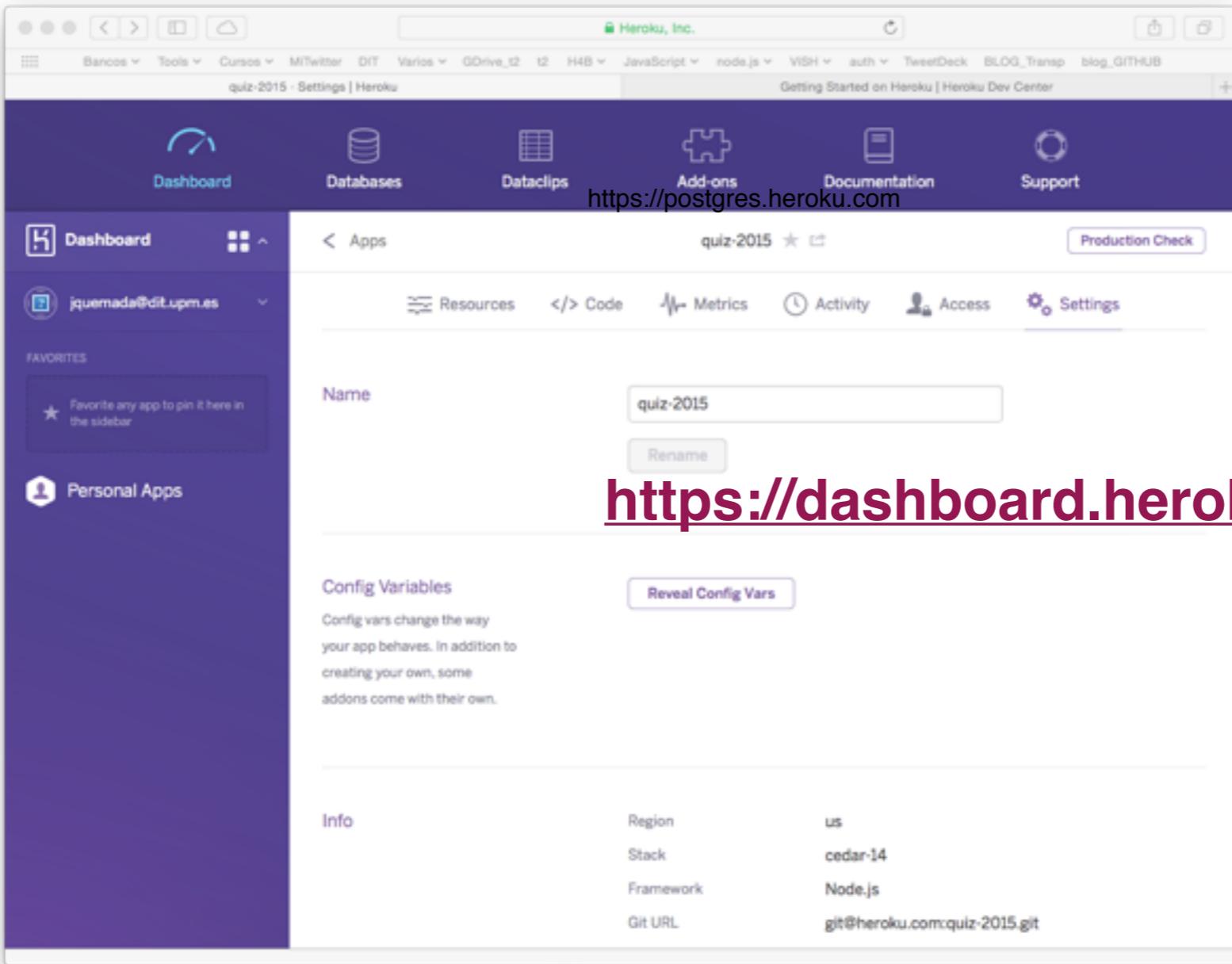
```
..$ heroku apps:rename quiz-2015
```

// Cambiamos el nombre/URL de la aplicación creada a uno propio nuestro  
// el nombre/URL pasará a ser: <https://quiz-2015-pepe.herokuapp.com/>  
// en vez del generado automáticamente: <https://pure-reaches-3917.herokuapp.com/>

```
..$ heroku open // abre el navegador en el URL https://quiz-2015.herokuapp.com/
```

# Heroku Dashboard

- ◆ El Dashboard es un interfaz Web interactivo de gestión de aplicaciones
  - Las aplicaciones también se pueden crear o modificar desde el Dashboard
    - ◆ Permite hacer muchas operaciones de Heroku interactivamente
    - ◆ Es más sencillo y claro de manejar para principiantes



The screenshot shows the Heroku Dashboard interface for the application 'quiz-2015'. The top navigation bar includes links for 'Dashboard', 'Databases', 'Dataclips', 'Add-ons', 'Documentation', and 'Support'. The main content area displays the application's name 'quiz-2015' and various management tabs: 'Resources', 'Code', 'Metrics', 'Activity', 'Access', and 'Settings'. The 'Settings' tab is active. In the 'Info' section, the application's details are listed:

Info	Region	us
Stack	cedar-14	
Framework	Node.js	
Git URL	git@heroku.com:quiz-2015.git	

**<https://dashboard.heroku.com>**



## FAVORITES



Favorite any app to pin it here in the sidebar



## Personal Apps

## Name

quiz-2015

nombre de la aplicación

## Config Variables

Config vars change the way your app behaves. In addition to creating your own, some addons come with their own.

## Info

Datos de la máquina virtual desplegada y del código instalado

Region

us

Stack

cedar-14

Framework

Node.js

Git URL

git@heroku.com:quiz-2015.git

Repo size

18.9KB

Slug size

10.1MB of 300MB

## Domains

Add your custom domains here then [point your DNS to Heroku](#).

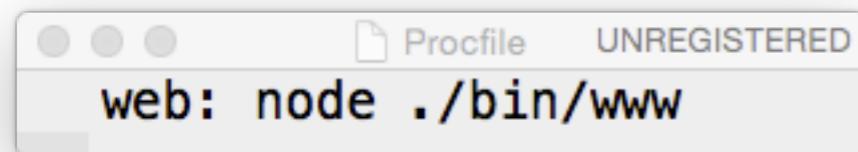
## Domain Names

quiz-2015.herokuapp.com

URL de la aplicación desplegada

# Paso 4: Adaptar Quiz a Heroku

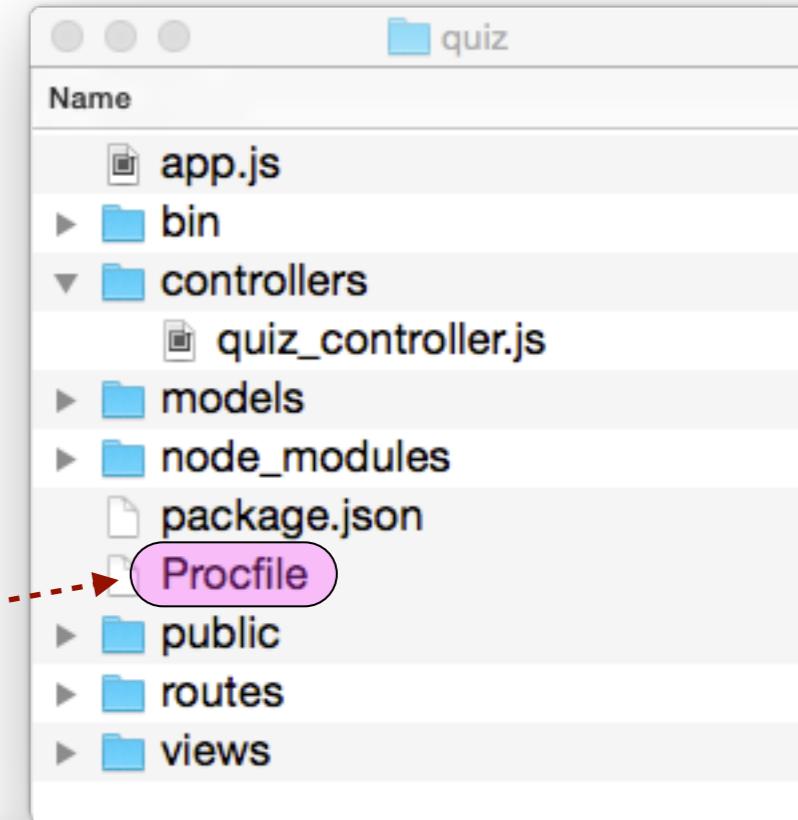
- ◆ El despliegue de Quiz en Heroku necesita
  - Crear el fichero Procfile y reestructurar package.json



```
Procfile UNREGISTERED
web: node ./bin/www
```

- ◆ **Procfile:** fichero de arranque en Heroku en directorio raíz
  - con el contenido “**web: node ./bin/www**”
    - ◆ Aplicación arranca ahora con “**foreman start**”
    - ◆ Creando un entorno similar al de heroku

- ◆ **package.json**
  - debe incluir un apartado “**engines**” que declara
    - ◆ la versión de engines a instalar en Heroku
    - ◆ Incluir versión de “**npm**” y “**node**” en “**engines**”



```
{
  "name": "quiz",
  "version": "0.0.0",
  "private": true,
  "scripts": {
    "start": "node ./bin/www"
  },
  "engines": {
    "node": "0.10.x",
    "npm": "1.4.x"
  },
  "dependencies": {
    "body-parser": "~1.8.1",
    "cookie-parser": "~1.3.3",
    "debug": "~2.0.0",
    "ejs": "~0.8.5",
    "express": "~4.9.0",
    "express-partial": "^0.3.0",
    "morgan": "~1.3.0",
    "serve-favicon": "~2.1.3",
  }
}
```

# Paso 5: versión y despliegue en Heroku

- ◆ Para desplegar en Heroku debemos subir al repositorio git de Heroku
  - El código de la aplicación a desplegar con “**git push heroku master**”
    - ◆ Para desplegar consolidamos la versión, normalmente en la rama master
- ◆ “**heroku create**” configura el repositorio remoto git como **heroku** en nuestro PC
  - El repositorio remoto **heroku** se puede configurar también con el comando
    - ◆ “**git remote add heroku URL**” -> URL dado en Dashboard
  - En el dashboard podemos ver también los datos del repositorio remoto git

```
..$ cd quiz      // entramos en directorio del proyecto quiz  
  
..$ git add .    // añadimos los cambios al índice  
  
                  // generamos versión “Despliegue en Heroku” en la rama master  
..$ git commit -m “Despliegue en Heroku”  
  
                  // Subir la última versión de la rama master a Heroku  
..$ git push heroku master // !!!Quiz ya esta desplegado!!!  
  
..$ heroku open // Lanzar el navegador conectado a la aplicación
```



# Final del tema