Servidor API REST con Node.js

Adolfo Sanz De Diego

Octubre 2013

# Acerca de

## El GUL

* El GUL es el **Grupo de Usuarios de Linux de la UC3M**.
* Grupo de personas con inquietudes en torno a la informática.
* Con la idea común de la utilización y promoción del **Software Libre**.
* Quedamos de vez en cuando y organizamos actividades sobre todo esto.
* El punto de unión es la **lista de correo** que está abierta a todo el mundo.

## ¿Dónde encontrarnos?

* Twitter: [@guluc3m](http://twitter.com/guluc3m)
* Lista: [gul@gul.uc3m.es](mailto:gul@gul.uc3m.es)
* Ftp: <ftp://ftp.gul.uc3m.es>
* Web: <http://www.gul.uc3m.es>
* Podcast: <http://holamundo.gul.es/>
* Blog: <http://planeta.gul.uc3m.es/>
* Linkedin: <http://www.linkedin.com/groups?gid=3451836>

## Adolfo Sanz De Diego



asanzdiego

## Mi CV resumido

* **Antiguo programador web JEE**
* Hoy en día:
  + **Profesor de FP de informática**:
  + - Hardware, Sistemas Operativos, Redes, Programación
  + **Formador Freelance**:
  + - Java, Android- JavaScript, jQuery- JSF, Spring, Hibernate y Groovy&Grails
  + Programo en mis tiempos libres.

## Hackalover

* **Meetup**: <http://www.meetup.com/Hackathon-Lovers/>



HackathonLovers

* **Twitter**: <http://twitter.com/HackathonLovers>



HackathonLovers

## ¿Donde encontrarme?

* Mi nick: **asanzdiego**
* AboutMe: [**http://about.me/asanzdiego**](http://about.me/asanzdiego)
  + [GitHub](http://github.com/asanzdiego/)
  + [Twitter](http://twitter.com/asanzdiego)
  + [Blog](http://asanzdiego.blogspot.com.es)
  + [LinkedIn](http://www.linkedin.com/in/asanzdiego)
  + [Google+](http://plus.google.com/118434794733307240711)

## Créditos

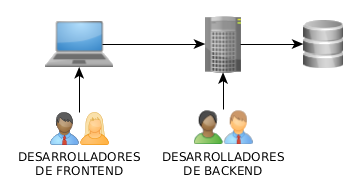
* Agradecimientos a Carlos Azustre ([@carlosazaustre](http://twitter.com/carlosazaustre))
  + [Cómo crear una API REST usando Node.JS](http://carlosazaustre.es/blog/como-crear-una-api-rest-usando-node-js/)
* Y a la empresa [redradix](http://redradix.com/)
  + [Curso de Node.js](http://redradix.com/courses/node)

## Licencia

* Estas **transparencias** están bajo una licencia:
  + [Creative Commons Reconocimiento-CompartirIgual 3.0](http://creativecommons.org/licenses/by-sa/3.0/es/)
* El **código fuente** de los programas están bajo una licencia:
  + [GPL 3.0](http://www.viti.es/gnu/licenses/gpl.html)

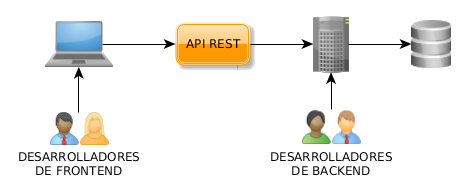
# APIs ¿Para qué?

## Aplicación estándar



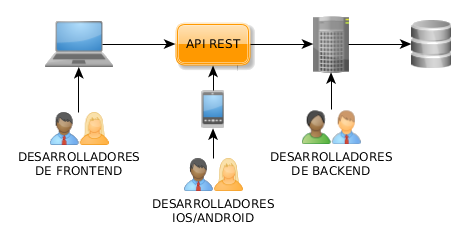
Esquema APIs RESTful 1

## Introducimos API



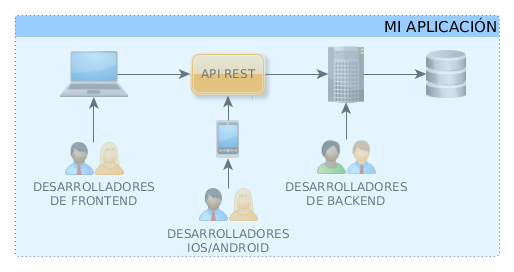
Esquema APIs RESTful 2

## Separación Roles



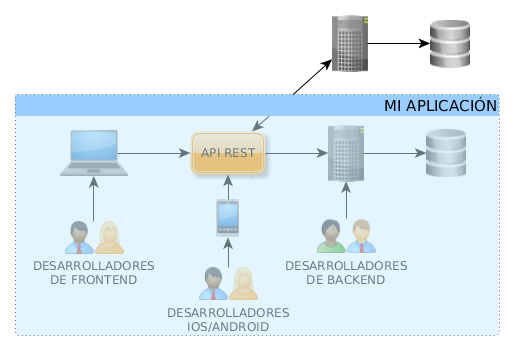
Esquema APIs RESTful 3

## ¿Y ahora qué?



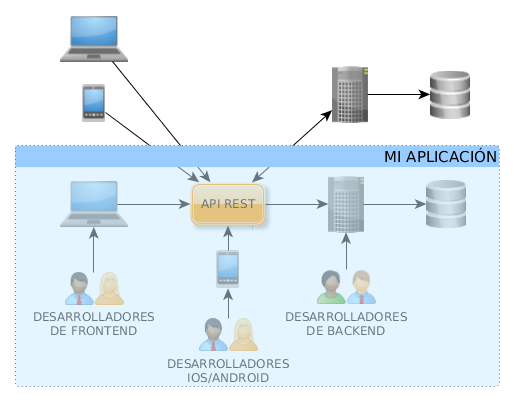
Esquema APIs RESTful 4

## Servicios externos



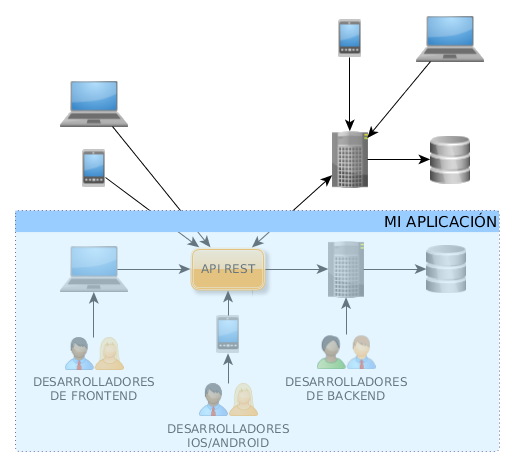
Esquema APIs RESTful 5

## Apps clientes



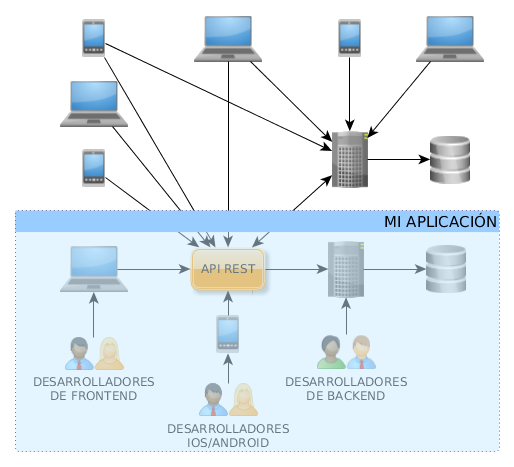
Esquema APIs RESTful 6

## Apps de servicios



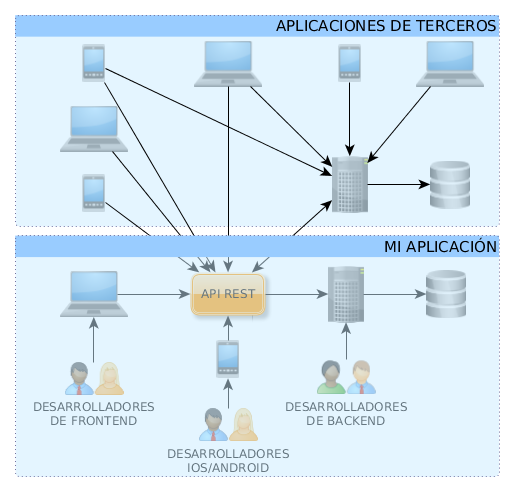
Esquema APIs RESTful 7

## Apps mixtas



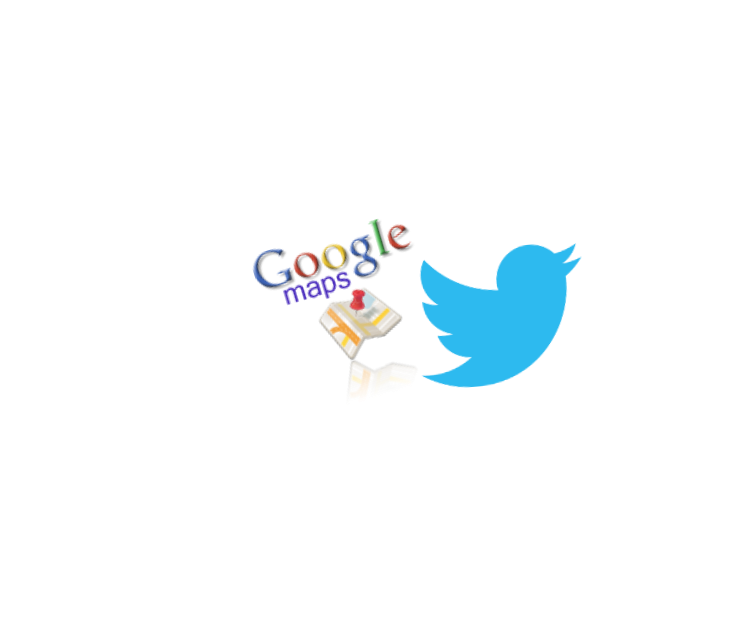
Esquema APIs RESTful 8

## Plataforma



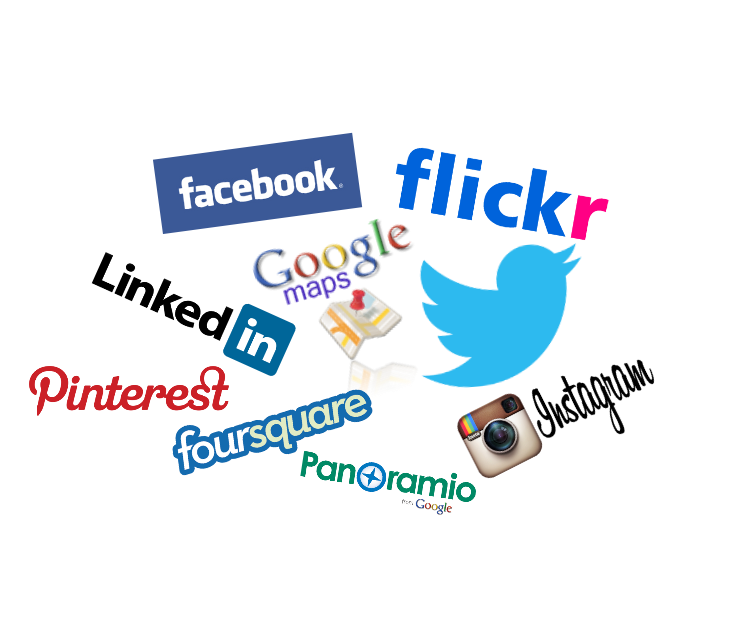
Esquema APIs RESTful 9

## ¿Quien expone APIs?



APIs 1

## ¿Quien expone APIs?



APIs 2

## ¿Quien expone APIs?



APIs 3

## Exponlas tú



Ejemlo Taza

## Exponlas tú



Ejemlo Taza Ece Homo

## Exponlas tú



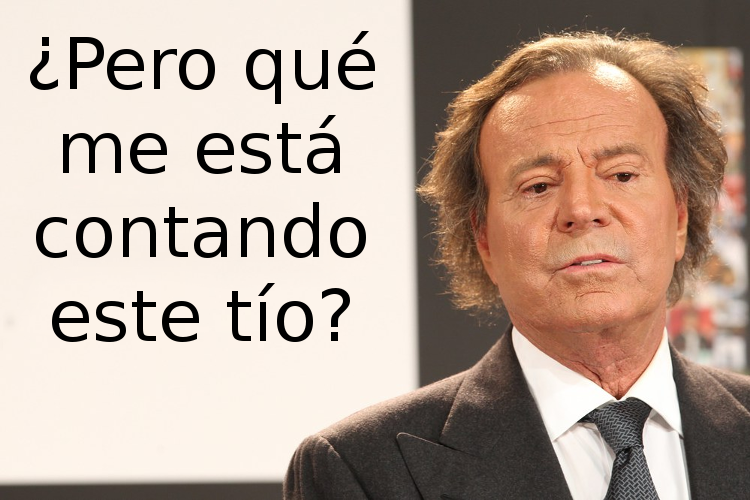
Ejemlo Taza Ece Homo Guay

# APIs RESTful

## ¿Qué es REST?

* REST (**Representational State Transfer**) es una técnica de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web.

## ¿Qué es REST?



Julio Iglesias WTF

## ¿Qué es REST?

* Una **URL** (Uniform Resource Locator) representa un **recurso**.
* Se puede acceder al recurso o modificarlo mediante los **métodos del protocolo HTTP**:
* - POST, GET, PUT, DELETE

## ¿Qué es REST?



Julio Iglesias SOLO ESO

## Ejemplo API

* **http://myhost.com/talk**
  + GET > Devuelve todas las charlas.
  + POST > Crear una nueva charla.
* **http://myhost.com/talk/123**
  + GET > Devuelve la charla con id=123
  + PUT > Actualiza la charla con id=123
  + DELETE > Borra la charla con id=123

## Manejo de errores

* **Se pueden utilizar los errores del protocolo HTTP**:
  + 200 Successful
  + 201 Created
  + 202 Accepted
  + 301 Moved Permanently
  + 400 Bad Request
  + 401 Unauthorised
  + 402 Payment Required
  + 403 Forbidden
  + 404 Not Found
  + 405 Method Not Allowed
  + 500 Internal Server Error
  + 501 Not Implemented

## ¿Por qué REST?

* Es **más sencillo** (tanto la API como la implementación).
* Es **más rápido** (peticiones más lijeras que se pueden cachear).
* Es **multiformato** (HTML, XML, JSON, etc.).
* Se complementa muy bien con **AJAX**.

## REST vs RESTful

* REST se refiere a un tipo de arquitectura de software
  + Se utiliza como **nombre**
  + Se utiliza como por ejemplo: success = éxito.
* Si un servicio web es REST**ful** indica que implementa dicha arquitectura.
  + Se utiliza como **adjetivo**
  + Se utiliza como por ejemplo: success**ful** = éxito**so**).

## REST vs RESTful

* A veces el **ful** se confunde con **full** = completo.
  + Y se refiere a los servicios web REST**full**
  + Aquellos que implementan una API contodos los métodos del protócolo HTTP.
  + Y se refiere a los servicios web REST (**sin el full**)
  + Aquellos que NO implementan una API contodos los métodos del protócolo HTTP.

# Node.js

## Introducción



Node.js

* **Node.js** permite programar en **Javascript del lado del servidor**
* Pensado para **un manejo de E/S orientada a eventos**.
* Pensado para **la creación de programas de red altamente escalables**.

## Ejecución

* Ejecución **concurrente**
  + Muchos tareas
* Pero **NO paralelo**
  + Una única hebra

## ¿Dónde usarlo?

* Cuando hay **mucha E/S**
  + y por tanto mucha CPU inactiva por tarea.
* Pero hay **muchos clientes**
  + que compensan esa inactividad de la CPU.

## Otros conceptos

* **npm** <http://npmjs.org/>:
  + es el gestor de paquetes de Node.js.
* **expressjs** <http://expressjs.com/>:
  + es una librería para Node.js de desarrollo web.
* **mongoosejs** <http://mongoosejs.com/>:
  + es una librería para Node.js de modelado de objetos de MongoDB <http://www.mongodb.org/>

## Primero pasos

* Instalar **Node.js**
  + <http://nodejs.org/download/>
* Instalar **MongoDB**
  + <http://docs.mongodb.org/manual/installation/>

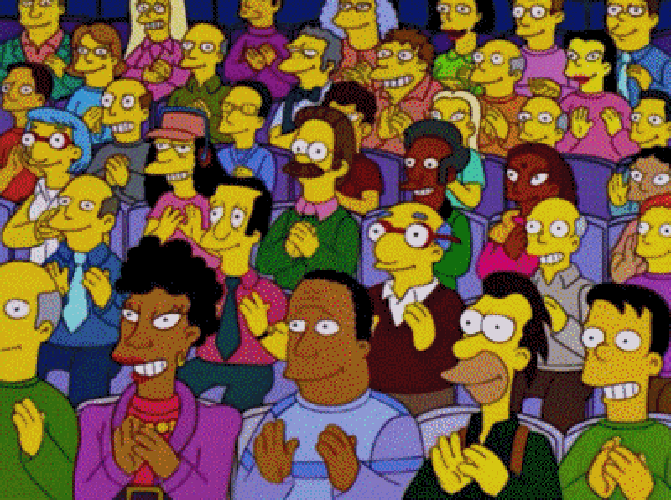
## Aburrido



Gato Aburrido

# Código

## Aplausos



Aplausos ANimados

## package.json

* Define las **dependencias** de nuestro proyecto.

{ "name" : "api-restful-nodejs-server", "version" : "0.0.1", "dependencies" : { "express" : "3.x", "mongoose" : "3.6.20" }}

## npm install

* Este comando instalará en la **carpeta node\_modules** las dependencias de nuestro proyecto.

npm install

## app.js

* Es el fichero principal.
* El nombre es lo de menos.
* A veces también se le suele llamar server.js
* Para ejecutar una aplicación de Node.js:

node app.js

## app.js

// modulos requeridosvar http, express, mongoose, app, server ...// configuramos appapp.configure(function () { // config...});// importamos las rutasvar routes = require('./routes/talkRoute')(app);// conectamos con la base de datosmongoose.connect('mongodb://localhost/gul', function(err, res) { // console.log('Connected to GUL MongoDB Database');});// arrancamos el servidorserver.listen(3000, function() { // console.log("Server running on http://localhost:3000");});

## Directorios

* Puedes usar la estructura de directorios que quieras.
* Yo he usado esta:
  + **models**:
  + **routes**:
  + **services**:

## models

* Directorio con los modelos que se van a guardar en base de datos.
* Yo creo un fichero js para cada colección.

## models/talkModel.js

// modulos requeridosvar mongoose = require('mongoose');var Schema = mongoose.Schema;// definimos el modelo 'talk' con sus restricciones// también podemos definir relaciones con otros modelos// aquí no se ha hecho por simplificarvar talkSchema = new Schema({ talkName: { type: String, required: true }, talkDate: { type: Date, required: true }, talkSpeaker: { type: String, required: true }, talkSpeakerMail: { type: String, required: true, match: /^\w+@[a-zA-Z\_]+?\.[a-zA-Z]{2,3}$/ }, talkPoints: { type: Number, required: true, default:0 }});// exportamos el modelomodule.exports = mongoose.model('Talk', talkSchema);

## routes

* Directorio con los mapeos de las rutas de la API RESTful.
* Aquí sólo gestiono la 'request' y el 'response'.
* Transformo la 'request' en un objeto 'options' y se le paso a un servicio.
* Lo que devuelva el servicio lo meto en el 'response'.
* Yo creo un fichero js para cada colección.

## routes/talkRoute.js

module.exports = function(app) { // importamos servicios var TalkService = require('../services/talkService.js'); var findTalks = function(req, res) { TalkService.findAllTalks({...}); }; var findTalk = function(req, res) { var talkId = req.params.talkId; TalkService.findTalkById({...}); }; var addTalk = function(req, res) {...}; var updateTalk = function(req, res) {...}; var deleteTalk = function(req, res) {...}; // mapeamos método y URL a una función app.get( '/talk', findTalks); app.get( '/talk/:talkId', findTalk); app.post( '/talk', addTalk); app.put( '/talk/:talkId', updateTalk); app.delete( '/talk/:talkId', deleteTalk);}

## services

* Aquí están los servicios que acceden a base de datos.
* Aquí no hay ni request ni response.
* Las funciones reciben un objeto 'options' con lo que necesita.
* Normalmente deben gestionar al menos un 'onSuccess' y un 'onError'.
* Hay funciones que además gestionan un 'onNotFound'.
* Un servicio puede llamar a otros servicios.
* Yo creo un fichero js para cada colección.

## services/talkService.js

//importamos el modelovar Talk = require('../models/talkModel.js');var findAllTalks = function(options) { Talk.find(function(error, talks) {...});};var findTalkById = function(options) { Talk.findById(options.talkId, function(error, talk) {...});};var saveTalk = function(options) {...};var findTalkByIdAndUpdate = function(options) { findTalkById({ ... saveTalk({...}); });};var findTalkByIdAndRemove = function(options) {...});};// exportamos los serviciosexports.findAllTalks = findAllTalks;exports.findTalkById = findTalkById;exports.saveTalk = saveTalk;exports.findTalkByIdAndUpdate = findTalkByIdAndUpdate;exports.findTalkByIdAndRemove = findTalkByIdAndRemove;

# Demo

# ¿Alguna pregunta?