Node.js & AWS

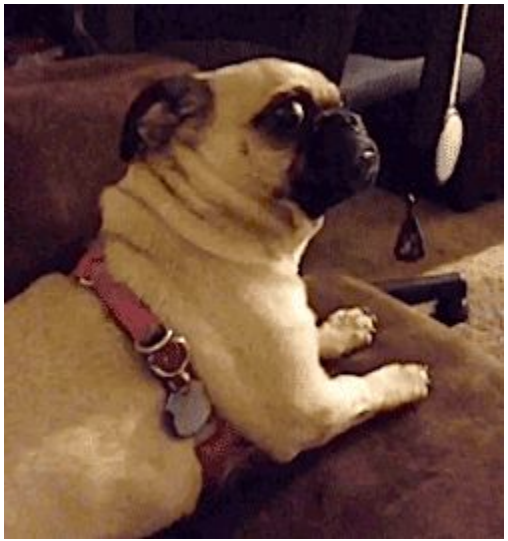
Workshop

Diego Varangot

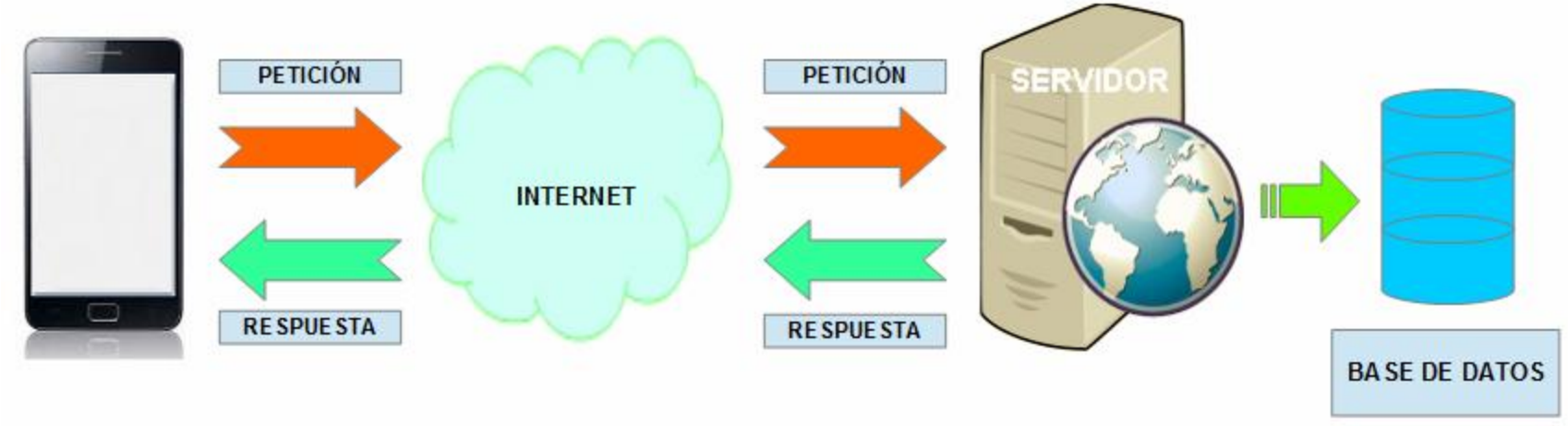
Daniela Marciukaitis

Cual es la finalidad del workshop??

- Crear una API en NodeJS
- Hacer un first deploy de la API usando AWS



Introducción

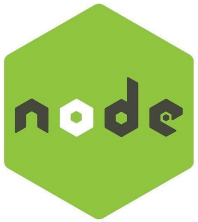


Agenda

- Que es NodeJS?
- ECMAScript, JavaScript, ES
- NVM
- Packages y NPM
- API
- Boilerplate
- Coding endpoints
- Coffee Break ☕

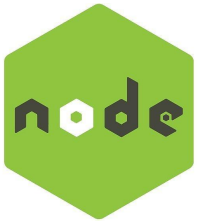
- Cloud Computing
- AWS
- EC2
- IAM
- S3
- Deploy with PM2

Qué es NodeJS?



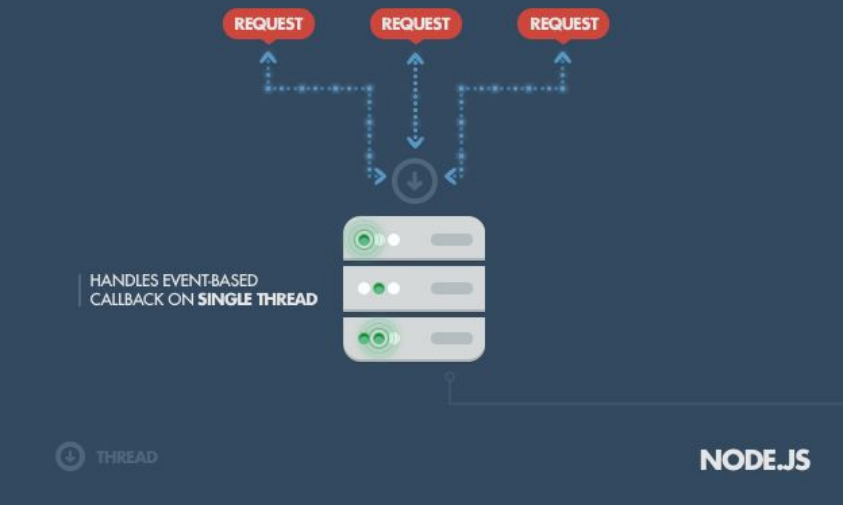
- Es un entorno en tiempo de ejecución multiplataforma
- Open source
- Basado en ECMAScript y en el motor V8 de Google
- Asíncrono
- Tiene un solo hilo de ejecución
- Es modularizable
- Soporta tanto base de datos relacionales como no relacionales
- I/O de datos en una arquitectura orientada a eventos
- Creado para ser útil en la creación de programas de red altamente escalables, como por ejemplo, servidores web

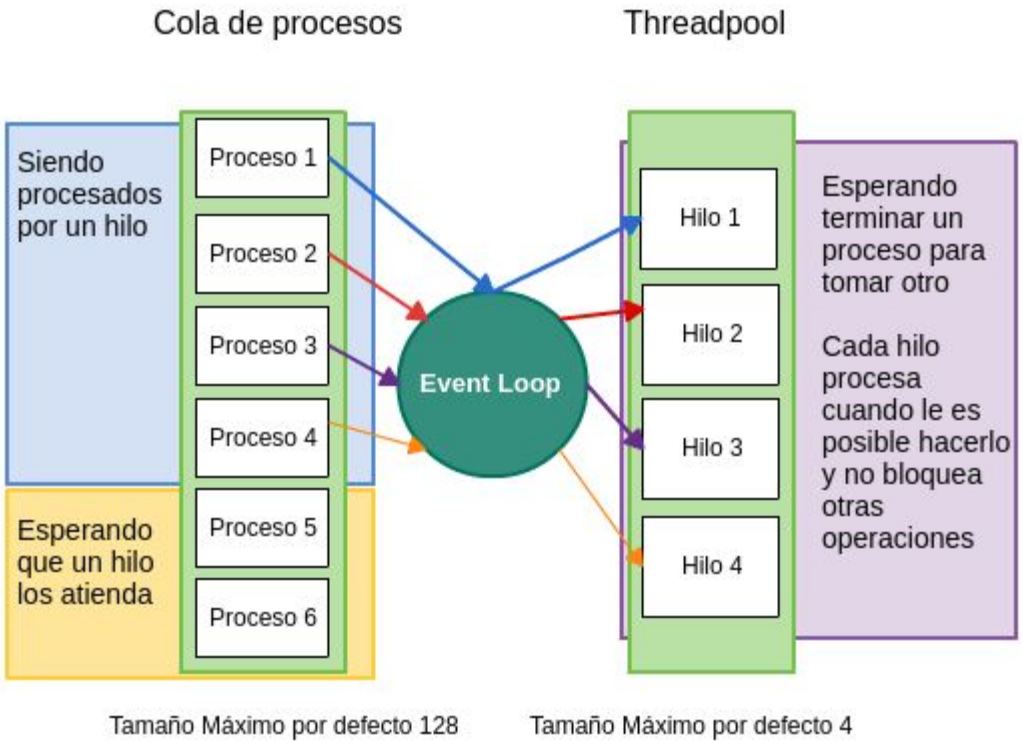
Qué es NodeJS?




```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```


REQUEST
REQUEST
REQUEST
REQUEST
REQUEST
SERVER CREATES NEW THREAD FROM LIMITED POOL OR WAITS FOR AVAILABLE THREAD
THREAD
WAITING
TRADITIONAL


REQUEST
REQUEST
REQUEST
HANDLES EVENT-BASED CALLBACK ON SINGLE THREAD
THREAD
NODE.JS


Cola de procesos
Threadpool
Siendo procesados por un hilo
Esperando que un hilo los atienda
Proceso 1
Proceso 2
Proceso 3
Proceso 4
Proceso 5
Proceso 6
Event Loop
Hilo 1
Hilo 2
Hilo 3
Hilo 4
Esperando terminar un proceso para tomar otro
Cada hilo procesa cuando le es posible hacerlo y no bloquea otras operaciones
Tamaño Máximo por defecto 128
Tamaño Máximo por defecto 4

JavaScript / ECMAScript / ES

A **JavaScript** se lo nombró así para aprovechar el éxito de JAVA en ese momento (1996). Luego Netscape presentó JavaScript en la **ECMA International** para estándares, lo que resultó en un nuevo estándar para lenguajes, conocido como **ECMAScript**.

Podemos decir que **ECMAScript** es un estándar mientras que JavaScript es la implementación más popular del estándar. JavaScript implementa **ECMAScript** y construye por encima de él.

ES es simplemente una abreviación de **ECMAScript**, y el número a continuación simplemente especifica la versión del estándar. Actualmente hay 8 versiones del estándar, la más reciente es ES2017 o ES8.

nvm

Es un manejador de versión de node, como desarrolladores es muy frecuente tener que cambiar de versiones de node ya sea por trabajar con diferentes proyectos, tanto como para actualizar la versión de un proyecto

https://node.green/

```
// Instalacion de una nueva version
nvm install 8

// Cambiar version de node
nvm use 8

// Ver version de node instaladas
nvm list
```

Node Package Manager

Es el manejador de paquetes por defecto para NodeJS, un entorno de ejecución para JavaScript.

npm init

```
{
  "name": "npmtest",
  "version": "0.0.0",
  "description": "Some NPM test.",
  "main": "index.js",
  "scripts": {
    "test": "make test"
  },
  "repository": {
    "type": "git",
    "url": "git://github.com/wolfeidau/npmtest.git"
  },
  "keywords": [
    "npm"
  ],
  "author": "Mark Wolfe <mark@wolfe.id.au>",
  "license": "MIT",
  "bugs": {
    "url": "https://github.com/wolfeidau/npmtest/issues"
  },
  "homepage": "https://github.com/wolfeidau/npmtest"
}
```

Yarn

- Es otro manejador de paquetes al igual que npm.
- Este fue creado por Facebook
- Es más rápido que npm
- También trabaja sobre el package.json

```
// inicializa el package.json
yarn init

// agregar paquete
yarn add [package]

//instalar dependencias
yarn install
```

Qué es una API REST?

- API (Application Programming Interfaces) es un conjunto de comandos, funciones y protocolos informáticos

- REST (Representational State Transfer) es un estilo de arquitectura para diseñar aplicaciones en red.
    - es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones
    - POST (crear), GET (leer y consultar), PUT (editar) y DELETE (eliminar)
    - es *stateless*

API endpoints

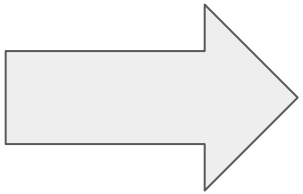
Repositorio:

https://bitbucket.org/dmarciukaitiscds/pets-api/src/boilerplate/

Postman requests:

https://www.getpostman.com/collections/8952e1f14545a099e6b3

```
//GET
/api/pets

//GET
/api/pets/:id

//POST
/api/pets

//PUT
/api/pets/:id

//DELETE
/api/pets/:id
```



Nosotros vamos a crear:

```
//POST
/api/pets/adopt/:id
```

Boilerplate

https://github.com/kunalkapadia/express-mongoose-es6-rest-api

- Node
- Express
- Mocha
- Mongoose
- JsonWebToken
- ES6
- Babel
- Code coverage
- npm & yarn

Mongo

- MongoDB es una base de datos orientada a documentos. En lugar de guardar los datos en registros, guarda los datos en documentos. Estos documentos son almacenados en BSON, que es una representación binaria de JSON.
- No es necesario seguir un esquema. Los documentos de una misma colección - concepto similar a una tabla -, pueden tener esquemas diferentes.

Express

Es un framework de node que nos permite manejar rutas, cookies, sesiones entre otras cosas

```
var express = require('express');
var app = express();

app.get('/', function (req, res) {
  res.send('Hello World!');
});

app.listen(3000, function () {
  console.log('Example app listening on port 3000!');
});
```

```
NOTE : sudo lsof -n -i :3000 | grep LISTEN
kill -9 processID
```

Autenticación

- Al ser una API Rest *stateless*, debemos de alguna forma identificar al usuario que está haciendo los requests.
- Al momento que el usuario se loguea, generamos un token con JWT y se le envía al frontend, de esta forma el frontend en cada request tendrá que enviarnos ese token así nosotros sabemos que quien hace la request es un usuario válido y logueado en nuestro sistema

Mongoose

Es un Object Document Mapper (ODM). Mongoose le permite definir objetos con un esquema que se asigna a un documento MongoDB

```
var mongoose = require('mongoose');

mongoose.connect('mongodb://localhost/mongoose_basics');

//--------------------------------
var userSchema = mongoose.Schema({
    firstName: String,
    lastName: String
});

//--------------------------------
var User = mongoose.model('User', userSchema);

User.find({
    firstName: "Juan"
}).exec(function(err, users) {
    if (err) throw err;

    console.log(users);
});
```

Transpiler

Es una herramienta que nos permite transformar nuestro código JS de última generación (ES2018, ES6, etc) a JS que cualquier navegador o versión de Node.js entienda.

```
const input = [1, 2, 3];
console.log(input.map(item => item + 1)); // [2, 3, 4]
```

Se transforma a :

```
var input = [1, 2, 3];
console.log(input.map(function (item) {
  return item + 1;
})); // [2, 3, 4]
```

Coding time

http://collabedit.com/wp3ch

A la vuelta continuamos con AWS y el deploy a una instancia EC2...

Cloud Computing

Dos conceptos muy importantes

- HaaS: hardware as a service
- SaaS: software as a service

Se proporcionan estos servicios por un tercero a través de internet, por lo general de una manera completamente transparente. Este nuevo término promete varias ventajas atractivas **para** las empresas y los usuarios finales.

AWS

Amazon Web Services es una colección de servicios de computación en la nube pública que en conjunto forman una plataforma de computación en la nube, ofrecidas a través de Internet por Amazon.com

Amazon Elastic Compute Cloud (EC2)

- Proporciona capacidad informática en la nube y de tamaño modificable
- Instance
- AMI (Amazon Machine Images)
- Security Groups

Identity and Access Management (IAM)

- Permite administrar el acceso a los servicios y recursos de AWS de manera segura
- Usuarios
- Roles

Simple Storage Service (S3)

- La capacidad para recopilar, almacenar y analizar sus datos de manera simple, segura y a gran escala
- Políticas de seguridad

Deploy time

Deploy Guide: Launch instance

1. Login en la consola de AWS => https://aws.amazon.com
2. Entrar a EC2
3. Launch image
4. Seleccionar Amazon Linux 2018.03.0
5. Seleccionar t2.micro (free tier)
6. Review and Launch
7. Security groups
8. Review and Launch
9. Wait for initialising to end

Deploy Guide: Install tools and packages

1. Seccionar la instancia de EC2
2. En “actions” seleccionar connect
3. Ir a la carpeta donde esta el pem y ejecutar:
   chmod 400 “your_file.pem”
4. Conectar usando el DNS público
5. Instalar GIT:
   sudo yum install git
6. Instalar NVM:
   curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh | bash
7. Instalar Node:
   nvm install 8.10
8. Instalar Yarn:
   npm install --global yarn
9. Instalar PM2:
   npm install --global pm2

Deploy Guide: ssh key exchange

1. Generar una clave pública local
   ssh-keygen -t rsa -b 4096 -C “your_email@example.com”
2. Abrir el archivo de claves públicas
   ~/.ssh/id_rsa.pub
3. Copiar la clave local
4. ir a Authorized keys en la instancia y agregar la clave
5. Generar una nueva clave en la instancia
6. Abrir el archivo de claves públicas
7. Copiarla
8. Ir al repositorio y agregar la key en las deploy keys

Deploy Guide: Deploy with PM2

1. pm2 deploy develop setup
2. pm2 deploy ecosystem.json develop