

**08/03 - Express**

Es un framework de desarrollo de aplicaciones web minimalista y flexible para Node JS:

NPM

21.28%

express

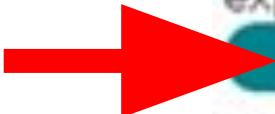
68.09%

Swig

8.51%

Javascript

2.13%



# En una aplicación NodeJS, el fichero package.json contiene:

La configuración y los metadatos de la aplicación

82.98%



El nombre del servidor

6.38%

Las dependencias

74.47%



Todos los ficheros JS de la aplicación

12.77%

# En NodeJS La función require se utiliza para:

Crear una nueva aplicación Node

2.04%

Definir módulos

14.29%

Eliminar modulos de la aplicación

0%

Importar modulos

83.67%



¿Cómo se denominan las funciones que se ejecutan cuando se recibe la petición en una app Node?

App (Aplicaciones)

8.7%

Handler(Manejadores)

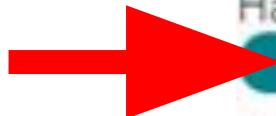
60.87%

Routing (Enrutamiento)

17.39%

Module (Módulos)

13.04%



# Forma de declarar un módulo en Node JS:

Received

`module.exports =function() {...}`



`module.imports =function() {...}`

`module.exports =(...)`



`module.require =function() {...}`

## Forma de incluir un módulo en Node JS:

```
module.require =function() [...]
```

33.33%

```
require("./routes/users.js")(app);
```

74.36%

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

66.67%

```
module.import = []
```

17.95%



**Para obtener los valores de los parámetros enviados en una petición POST en NodeJS se utiliza:**

req.body.<clave\_parámetro>

36.96%



req.params.<clave\_parámetro>

41.3%

req.query.<clave\_parámetro>

10.87%

req.form.query.<clave\_parámetro>

10.87%

**Para obtener los valores de los parámetros enviados en una petición GET en NodeJS se utiliza:**

req.body.<clave\_parámetro>

34.78%

req.params.<clave\_parámetro>

82.61%

req.query.<clave\_parámetro>

67.39%

req.form.query.<clave\_parámetro>

4.35%

Una plantilla Swig recibe un atributo con clave “nombre” ¿Como se insertaría el valor de ese atributo en un elemento H1?

<h1>[nombre]</h1>

17.39%

<h1>[[nombre]] </h1>

45.65%

<h1>\$[nombre] </h1>

28.26%

<h1>#[nombre] </h1>

9.7%



22/03

Proceso que permite comprobar si un usuario tiene permiso para acceder a ciertas URLs de una aplicación



## 1. Autorización

53.33%



## 2. Autenticación

33.33%



## 3. Enrutamiento

13.33%



## 4. Canal seguro

0%

En un módulo que permite cifrar y descifrar información en NodeJS

1. sha256



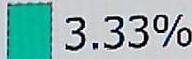
26.67%

✓ 2. crypto



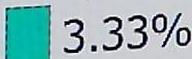
66.67%

3. createHmac



3.33%

4. require



3.33%

## Es una de las principales ventajas de MongoDB

1. Ideal para almacenar datos estructurados.



2. Muy potente en el manejo de transacciones.



3. No necesita definir un esquema previo.



4. Soporte flexible para la definición de relaciones entre entidades.



## En MongoDB, una colección:

1. Define la estructura de los documentos.  
 10%
2. Cada documento se almacena como una colección.  
 13.33%
- ✓ 3. Almacena documentos que pueden tener estructuras diferentes.

 76.67%

¿Cuál sería la mejor alternativa para utilizar un módulo desde otros modulos en una aplicación NodeJS?

1. Obtener el objeto/función siempre que vaya a utilizar el módulo.

23.33%

- 
2. Obtener el objeto/función una vez y enviarlo como parámetro a otros módulos.

46.67%

3. Obtener el objeto/función y almacenarlo en variables de la app.

30%

¿Cuál es la principal diferencia al definir un módulo como un objeto y no como una clase o función en una aplicación NodeJS?

- 1 Al incluir el módulo varias veces, todos retornan la referencia al mismo objeto.86.21%
- 2 Al incluir el módulo varias veces, todos retornan la referencia a diferentes objetos.13.79%
- 3 Al incluir el módulo varias veces, No hay diferencia al definir los módulos como clases, funciones u objetos.0%

05/04

# ¿Cuál es el objetivo principal de los servicios web?

- ✓ 1. Permitir la interoperabilidad entre aplicaciones.

 96.77%

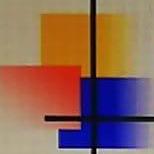
2. Permitir un acoplamiento fuerte entre aplicaciones.

 3.23%

3. Permitir la comunicación sólo entre aplicaciones desarrolladas en el mismo lenguaje.

0%

# ¿Porqué los servicios web son adecuado para entornos Web?

- 
1. Emplean tecnologías estándares como CORBA, RMI, DCOM, etc.

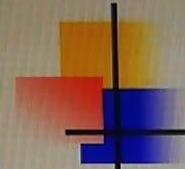
 2.63%

•
  - ✓ 2. Emplean tecnologías estándares como HTTP,SOAP, XML,JSON etc.

97.37%
  3. No son escalables y no permite la computación distribuida

0%

# ¿En qué consiste el principio HATEOAS de una API REST?



- ✓ 1. Los mensajes deben contener enlaces a otros recursos de la API.

32.26%

- 2. La API tiene una sintaxis universal para identificar los recursos (URI).

61.29%

- 3. Los recursos deben ser declarados como cacheables o no cacheables.

6.45%

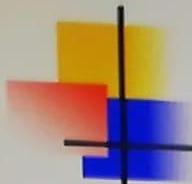
# ¿En qué filosofía se basa REST?



1. Se basa en la llamada a procedimientos/funciones remotas  
 27.03%
2. Se basa en que un mismo recurso solo puede representarse en único formato, ejemplo JSON.  
 2.7%
- ✓ 3. Se basa en una filosofía orientada a recursos  
 70.27%

12/04

Es un estándar basado en XML para el intercambio de información entre aplicaciones en entornos descentralizados y distribuidos



- ✓ 1. SOAP



2. WSDL



3. UDDI

0%

Se utiliza describir la funcionalidad que proporciona un servicio Web SOAP

1. ENVELOP

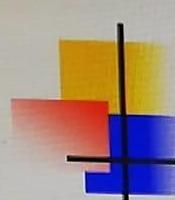
10.71%

✓ 2. WSDL

75%

3. UDDI

14.29%



Los servicios Web SOAP permiten el intercambio de información en formato:

1. JSON 6.67%
- ✓ 2. XML 40%
3. XML y JSON 53.33%

Sección en la que se añade información relativa a la autenticación o a un ID de transacción de un sobre SOAP

### 1. Body



13.79%

### ✓ 2. Header



48.28%

### 3. Envelope



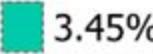
37.93%

07/05 Repaso final

## ¿Qué es NodeJS?

1. Un servidor web multiplataforma  
 3.7%
2. Un lenguaje de programación  
0%
3. Un Framework que permite ejecutar JS del lado del servidor  
 48.15%
- ✓ 4. Una plataforma de software que permite ejecutar JS del lado del servidor  
 48.15%

## ¿Cómo gestiona NodeJS las operaciones de I/O?

1. De forma síncrona y sin bloqueo  
 13.79%
2. De forma asíncrona y sin bloqueo  
 68.97%
3. De forma asíncrona y sólo de bloquean algunas operaciones  
 3.45%
4. De forma síncrona, asíncrona y con bloqueo  
 13.79%

## ¿Qué tipo de escalabilidad usa NodeJS para las aplicaciones?



1. Horizontal



2. Vertical



3. Transversal



4. Perpendicular

0%

## NodeJS mejora la concurrencia porque su arquitectura se basa:

1. Ejecución multihilo con un bucle de eventos 10.34%
2. Ejecución multihilo y un modelo I/O asíncrono 51.72%
- ✓ 3. Un solo hilo de ejecución con un bucle de eventos 37.93%
4. Un solo hilo de ejecución sin bucles de eventos 0%

## **En una aplicación NodeJS, el fichero package.json contiene:**

La configuración y los metadatos de la aplicación

80% 

El nombre del servidor

5%

Las dependencias

85% 

Todos los ficheros JS de la aplicación

0%

## Son características de una comunicación asíncrona:

Las operaciones no son bloqueantes

82.35% 

Las operaciones se ejecutan de forma secuencial

0% 

Las operaciones son bloqueantes

5.88% 

Se realizan mediante el sistema de callback(retrollamadas) y/o promesas

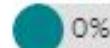
100% 

Es un framework de desarrollo de aplicaciones web minimalista y flexible para NodeJS:

1. NPM 15.38%
- ✓ 2. express 53.85%
3. Swig 30.77%
4. Javascript  
0%

## En una aplicación NodeJS la función require se utiliza para:

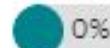
Crear una nueva aplicación Node



Definir módulos



Eliminar modulos de la aplicación



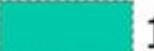
Importar modulos



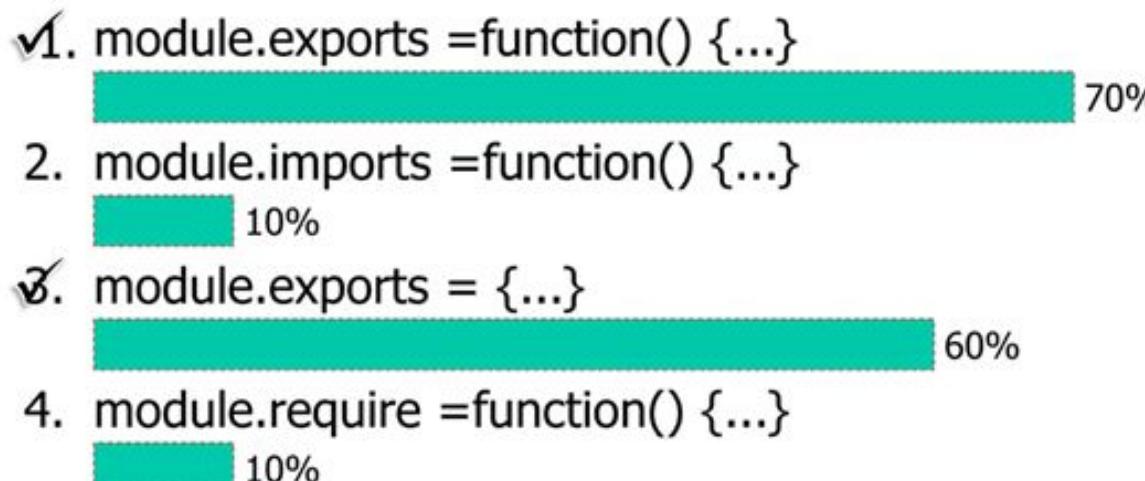
¿Cómo se denominan las funciones que se ejecutan cuando se recibe la petición en una app NodeJS?

1. App (Aplicaciones)  
 3.45%
2. Handler(Manejadores)  
 37.93%
3. Routing (Enrutamiento)  
 48.28%
4. Module (Módulos)  
 10.34%

Para obtener los valores de los parámetros enviados en una petición POST en NodeJS se utiliza:

- ✓ 1. req.body.<clave\_parámetro>  
 83.33%
- 2. req.params.<clave\_parámetro>  
 13.33%
- 3. req.query.<clave\_parámetro>  
 3.33%
- 4. req.form.query.<clave\_parámetro>  
0%

## Forma de declarar un módulo en NodeJS:



(% = Percentage of Voters)

## Forma de incluir un módulo en NodeJS:

1. module.require =function() {...}



✓ 2. require("./routes/users.js")(app);



✓ 3. var express = require('express');

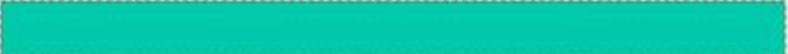


4. module.import = {}



(% = Percentage of Voters)

Para obtener los valores de los parámetros enviados en una petición GET en NodeJS se utiliza:

1. req.body.<clave\_parámetro>  
 15.79%
2. req.params.<clave\_parámetro>  
 94.74%
3. req.query.<clave\_parámetro>  
 78.95%
4. req.form.query.<clave\_parámetro>  
0%

(% = Percentage of Voters)

## ¿Cuál es el objetivo principal de los servicios web?

- 1. Permitir la interoperabilidad entre aplicaciones.

89.29%

- 2. Permitir un acoplamiento fuerte entre aplicaciones.

10.71%

- 3. Permitir la comunicación sólo entre aplicaciones desarrolladas en el mismo lenguaje.

0%

## ¿En qué consiste el principio HATEOAS de una API REST?

- ✓ 1. Los mensajes deben contener enlaces a otros recursos de la API.  
 37.04%
- 2. La API tiene una sintaxis universal para identificar los recursos (URI).  
 62.96%
- 3. Los recursos deben ser declarados como cacheables o no cacheables.  
0%

## ¿En qué filosofía se basa REST?

1. Se basa en la llamada a procedimientos/funciones remotas  
 26.32%
2. Se basa en que un mismo recurso solo puede representarse en único formato, ejemplo JSON.  
 26.32%
3. Se basa en una filosofía orientada a recursos  
 47.37%



Es un mecanismo que le permite a una aplicación realizar transferencias seguras de datos en dominios cruzados entre navegadores y servidores Web.

JSON



AJAX



CORS



## Una petición web a un recurso REST:

1. Debe retornar un código de respuesta estándar.

0%

2. Debe retornar una respuesta en formato estándar.



12%

3. Las respuestas 1 y 2 son correctas



84%

4. Las respuestas 1 y 2 son incorrectas



4%

El encabezado de respuesta que indica si los recursos de la respuesta pueden ser compartidos con una URL dada.

1. Access-Control-Allow-Credentials

 24.14%

2. Access-Control-Allow-Methods

 3.45%

- ✓ 3. Access-Control-Allow-Origin

 72.41%

Es un estándar basado en XML para el intercambio de información entre aplicaciones en entornos descentralizados y distribuidos

✓ 1 SOAP



2 WSDL



3 UDDI



Los servicios Web SOAP permiten el intercambio de información en formato

1. JSON



✓ 2. XML



3. HTML

0%

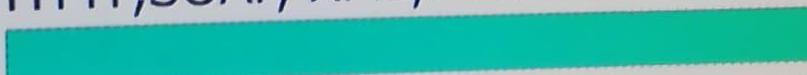
4. XML y JSON



Una plantilla Swig recibe un atributo con clave “nombre” ¿Como se insertaría el valor de ese atributo en un elemento H1?

1. <h1>{nombre}</h1>  
 6.67%
- ✓ 2. <h1>{{nombre}} </h1>  
 86.67%
3. <h1>\${nombre} </h1>  
 6.67%
4. <h1>#{nombre} </h1>  
0%

## ¿Porqué los servicios web son adecuado para entornos Web?

1. Emplean tecnologías estándares como CORBA, RMI, DCOM, etc.  
0%
- ✓ 2. Emplean tecnologías estándares como HTTP,SOAP, XML,JSON etc.  
 100%
3. No son escalables y no permite la computación distribuida  
0%

## ¿Qué es JSON?

- ✓ 1. Es un formato ligero de almacenamiento e Intercambio de datos.



- 2. Es un formato que sólo puede ser usado con JAVASCRIPT.



- 3. Es un formato menos eficiente que XML



Se utiliza describir la funcionalidad que proporciona un servicio Web SOAP

1 ENVELOP



23.53%

✓ 2 WSDL



70.59%

3 UDDI



5.88%

## Repaso

- En base al siguiente fragmento de código, si un cliente envía una petición a **localhost:8080/coche/borrar** ¿Qué respuesta retornará la aplicación?



```
var app = . . .
app.get('/coche/*', function (req, res) {
    res.send('1');
});
app.get('/coche/:id', function (req, res) {
    res.send('2');
});
app.get('/coche/borrar', function (req, res) {
    res.send('3');
});
```

## Repaso

- ¿Existe alguna diferencia entre obtener un atributo vía `req.body.<atributo>` o `req.query.<atributo>`? Explícalo brevemente.
  - `Req.body` -> parámetro en el cuerpo de la petición (valido para peticiones POST).
  - `Req.query` -> parámetro GET presente en la URL

## Repaso

- ¿Cómo gestiona NodeJS las operaciones de I/O?
  - Gestiona las operaciones de entrada y salida de forma **asíncrona y sin bloqueo**



## Repaso

- ¿Hay algún error de implementación en el siguiente fragmento de código?  
**(Error de implementación: uso incorrecto de los objetos o funciones que producirían un ERROR).**

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

app.get('/saludar', function() {
    res.send('saludar');
});
```

DONDE PONE app. ANTES PONÍA  
express.

## Repaso

- ¿De que dos formas se puede escalar una aplicación?
  - Vertical: Consiste en agregar más recursos a un solo nodo.
  - Horizontal: Consiste en agregar más nodos a un sistema

## Repaso

- Completa el siguiente fragmento de código, para que si un cliente realiza una petición a /prueba/<valor> la aplicación responda la cadena de texto <valor>

```
app.get("/prueba/:valor", function(req, res) {  
    var respuesta = req.params.valor;  
    res.send(respuesta);  
})
```