Master Frameworks/Librerías JS

# Qué son las promesas en JavaScript:

Una [Promise](https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Objetos_globales/Promise) (promesa en castellano) es un objeto que representa la terminación o el fracaso de una operación asíncrona.

let greet = new Promise((resolve, reject) => {

    setTimeout(() => {

        //

        let greeting = "Hi dude"

        if(greeting){

            resolve(greeting);

        }else{

            reject("No greeting available")

        }

    }, 2000)

})

greet.then(result => {

    alert.apply(result);

}).catch( err => {

    alert.err(err)

})

# Qué es MongoDB

ES una base de datos no relacional y noSQL, que permite almacenar documentos en archivos BSON (similar a JSON), en lugar de usar tablas.

## 1 NoSQL vs SQL

SQL:

* Relacionales
* Tablas
* Esquemas

NoSQL:

* No relacionales
* JavaScript
* Orientada a documentos (Json, Bson)
* Sencillez
* Velocidad
* Esquema libre

### Esquema tabla SQL

|  |  |  |
| --- | --- | --- |
|  | Columna | Columna |
| Fila | Registro | Registro |
| Fila | Registro | Registro |

### Base de datos > colección

|  |  |  |
| --- | --- | --- |
|  | Campo | Campo |
| Documento | Datos Doc. | Datos Doc. |
| Documento | Datos Doc. | Datos Doc. |

### Ejemplo

Usuarios://Colección

    {// documento

    // campo: datos

        nombre: 'Alberto',

        apellido: 'Perez',

        email: 'alberto@email.com'

    }

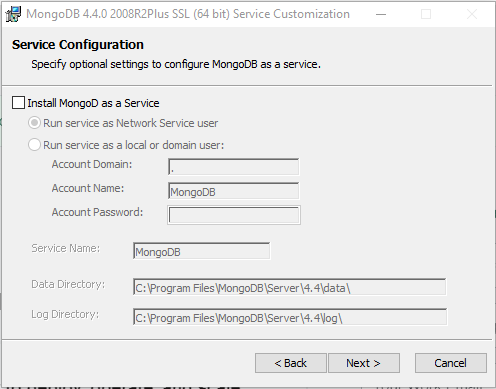
### Nomenclatura comparada :

Sql=>NoSQL

Tabla => Colecciones, Filas => Documentos, Columnas => Campos, Registros => Datos del documento

## 2 instalación de MongoDV

Siempre siguiente, pero en la pantalla:



Desmarcamos la casilla “Insaltalar mongo como servicio”, para que no tener ejecutado siempre el demonio de mongo, sino que lo arrancaremos cuando lo necesitemos.

Creamos una carpeta en c: llamada “data” y dentro de ella otra llamada “db” que será donde se guarde la configuración de MongoDb.

Para ejecutarlo debemos de ir al directorio donde lo hemos instalado : Server/4.4/bin/mongod.exe y de este modo se ejecuta en segundo plano

Y con mongo.exe se ejecuta la consola para poder trabajar con el.

### Crear una base de datos en Mongo

Cuando decimos use nombre\_bd, y esa bd no existe, mongo la crea de manera automática.

## > use pruebas\_1

## switched to db pruebas\_1

## > db.usuarios.save({nombre:'ALberto', apellido:'perez'})

## WriteResult({ "nInserted" : 1 })

De este modo creamos la bd pruebas\_1, con una colección llamada Usuarios y le pasamos los documentos.

## 3 administración visual

## <https://robomongo.org/>

Asegurarnos de descargar Robo3T y NO Studio 3T.

Una vez instalado, tenemos que conectarnos a nuestro MongoDb local, para ello, tenemos que dar a Create y en esta pantalla damos un nombre a la conexión y dejamos localhost y el puerto 27017.

# API RESTful con NodeJs y Mongoose

## Qué veremos:

1. Primeros pasos
2. Conexión a Mongo
3. Crear un servidor con NodeJs
4. Trabajar con un cliente RESTful
5. Modelos
6. Controladores
7. Rutas de NodeJs
8. CRUD (Create, Read, Update, Delete) de artículos
9. Subida de archivos
10. Hacer un buscador
11. Que es CORS y cómo trabajar con ello
12. ¡Más cosas!

## 1 primeros pasos

### Crear el proyecto:

Desde la consola y dentro del directorio donde queremos crear el proyecto, en mi caso :

PS C:\Users\Alber\Documents\Proyectos\Master-en-Frameworks-librerias-JavaScript-Aprende-Angular-React-Vue-> npm init

Nos pregunta el nombre del paquete y otros datos para configurar el proyecto

package name: (master-en-frameworks-librerias-javascript-aprende-angular-react-vue-) backend

version: (1.0.0)

description: Api RESTful del Master en Frameworks para JS

entry point: (promises.js) index.js

test command:

git repository: (https://github.com/Albertopga/Master-en-Frameworks-librerias-JavaScript-Aprende-Angular-React-Vue-.git)

keywords:

author: Alberto Perez

license: (ISC)

por último, nos muestra el fichero package.json que ha creado y si estamos conformes con ello:

About to write to C:\Users\Alber\Documents\Proyectos\Master-en-Frameworks-librerias-JavaScript-Aprende-Angular-React-Vue-\package.json:

{

"name": "backend",

"version": "1.0.0",

"description": "Api RESTful del Master en Frameworks para JS",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1"

},

"repository": {

"type": "git",

"url": "git+https://github.com/Albertopga/Master-en-Frameworks-librerias-JavaScript-Aprende-Angular-React-Vue-.git"

},

"author": "Alberto Perez",

"license": "ISC",

"bugs": {

"url": "https://github.com/Albertopga/Master-en-Frameworks-librerias-JavaScript-Aprende-Angular-React-Vue-/issues"

},

"homepage": "https://github.com/Albertopga/Master-en-Frameworks-librerias-JavaScript-Aprende-Angular-React-Vue-#readme"

}

Is this OK? (yes)

### Instalar dependencias

* expresJS, permite recibir peticiones http y retornar un resultado en JSON, crear la ruta… npm install --save express con—save nos lo guardará como dependencia del proyecto.
* Body-parser npm install --save body-parser cuando recibe un formulario lo que hace es convertir esos datos que le llegan, los convierte a Json directamente.
* Monggose npm install --mongoose da una interface para crear modelos de datos y poder trabajar con la base de datos.
* Connect-multyparty npm install --save connect-multiparty Librería para hacer uploads de filescon nodeJs
* Validator npm install --save validator nos ayuda a validar los datos recibidos
* Nodemon npm install –save-dev nodemon esta dependencia, instalado solo en desarrollo (-dev) nos permite reiniciar nodejs resfrescando así los resultados. Para utilizarlo, dentro del fichero package.json, añadimos la línea “ start”: “nodemon in index.js” dentro de scripts. Hacemos npm start y listo

## 2 ConectarNodeJs con mongodb

Primero debemos tener arrancado mongodb.

Importamos el módulo de mongoose y creamos la conexión:

let mongoose = require('mongoose')

//crear la conexion a mongodb por medio de una promesa

let url = 'mongodb://localhost:27017/api-rest-blog'

let options = {useNewUrlParser: true}

mongoose.connect(url, options).then(()=>{

    console.log('la conexión a la base de datos, se ha realizado correctamente!!!')

})

## 3 crear servidor con Node

Creamos un fichero app.js

## 4 Trabajar con un cliente RESTful

Para probar un api, vamos a utilizar Postman como cliente Rest, para manipular las peticiones que se hagan al api.

## 5 Modelo

Un modelo es una clase que nos da un molde para crear diferentes objetos. Que me dará métodos y funcionalidades para trabajar con el. Ayudan a crear la estructura correcta de la bbdd

## 6 Controlador