


## Algoritmos / Radix Sort

Radix Sort es un algoritmo de ordenación no comparativo que ordena enteros procesándolos por sus dígitos individuales, desde el dígito menos significativo hasta el más significativo. En cada iteración, los números se agrupan según el valor de un dígito específico, utilizando un sistema de "baldes" o "buckets". Es eficiente para ordenar números enteros, pero no es adecuado para ordenar objetos o cadenas alfabéticas.

[Mira la teoría aquí](#)

[o mira la ejecución, funcionamiento y explicación del código aquí](#)

js

 Copiar código

```
//
function getDigit(num, i) {
  // el valor absoluto permite que funcione con numeros negativos
  return Math.floor(Math.abs(num) / Math.pow(10, i)) % 10;
}

function getNumberLength(num) {
  if (num < 10) return 1;
  return Math.floor(Math.log10(Math.abs(num))) + 1;
}

function longestDigitCount(nums) {
  let maxDigits = 0;
  for (let i = 0; i < nums.length; i++) {
    maxDigits = Math.max(maxDigits, getNumberLength(nums[i]));
  }
  return maxDigits;
}

function radixSort(nums){
  let L = longestDigitCount(nums);
  for(let i = 0; i < L; i++){
    let digitBoxes = Array.from({length: 10}, () => []);
    for(let j = 0; j < nums.length; j++){
      let digit = getDigit(nums[j],i);
```

```
        digitBoxes[digit].push(nums[j]);
    }
    nums = [].concat(...digitBoxes);
}
return nums;
}

// Comprobación
const nums = [532, 123, 234, 643, 234, 1, 534, 654, 789, 432];
console.log("Array original:", nums);

const sortedNums = radixSort(nums);
console.log("Array ordenado:", sortedNums);
```

## Json Web Token / JWT



JWT (JSON Web Token) es un estándar abierto basado en JSON que permite transmitir información de forma segura entre partes como un objeto. Este token es comúnmente utilizado para autenticar usuarios y compartir información de forma segura entre el cliente y el servidor. Consiste en tres partes codificadas en Base64: encabezado, carga útil y firma. El servidor genera el token al autenticar al usuario, y el cliente lo almacena y lo envía en cada solicitud posterior para acceder a recursos protegidos, permitiendo así mantener la sesión del usuario sin necesidad de almacenar información en el servidor.


Primero crea un directorio

Las extensiones **Visual Studio Code** a usar son:

*Thunder Client y Postman*

**INICIANDO EL PROYECTO** en la ruta del directorio, en la terminal ejecuta


terminal

 Copiar código

```
// Inicializa un nuevo proyecto en node.js, da [enter] e ingresa los datos solicitados,  
// como version, nombre, descripción, entry point, etc.  
npm init  
  
// o usa npm init -y para crear el proyecto y que tome todas las opciones por defecto
```

Ahora instalamos un servidor, en este caso **EXPRESS**, y la librería de **JWT(JsonWebToken)**

terminal


 Copiar código

```
npm i express jsonwebtoken
```

Ahora en la raíz del proyecto crea un nuevo directorio llamado **config** y adicional crea un archivo llamado **app.js**

Dentro del directorio que acabas de crear llamado **config** crea un archivo llamado **keys.js**


## Estructura de carpetas

 Copiar código

```
config // New
  keys.js // New
node_modules
app.js // New
package-lock.json
package.json
```

En este proyecto no se va a trabajar con módulos, se va usar **REQUIRE**. Por ello en nuestro **app.js** configuramos nuestro servidor de la siguiente manera:

## app.js


 Copiar código

```
const express = require('express'); // Llamo la librería 'express'
const app = express(); // Se instancia la configuración de express para poder usar sus propiedades

// Configuramos el puerto de conexión
app.listen(5000, () => {
  console.log('El servidor esta conectado en http://localhost:5000 ');
});

// ejecuta la aplicación con ' nodemon app '
```

## app.js


 Copiar código

```
// Adicional mente podemos mostrar en pantalla algo como :
app.get('/', (req, res) => {
  res.send('Hola mundo');
});

// Guarda e ingresa a la ruta http://localhost:5000,
// (desde la terminal [ctrl + clic], o desde el navegador ingresando
// directamente la URL)
```

## Configurando las KEYS

keys.js


 Copiar código

```
module.exports = {  
  key: 'clave secreta2024';  
}
```

## Terminado de configurar 'app.js', primer parte

En esta primer parte configuramos que se este enviando por POST y se este generando el token, validando si los datos de ingreso son correctos

app.js

 Copiar código

```
const express = require('express'); // Llamo la librería 'express'  
const app = express(); // Se instancia la configuración de express para poder usar sus propiedades  
  
// Area de trabajo  
const jwt = require('jsonwebtoken');  
const keys = require('./config/keys');  
  
app.set('key', keys.key);  
app.use(express.urlencoded({extended: false}));  
app.use(express.json());  
  
app.post('/login', (req, res) => {  
  if (req.body.usuario == 'admin' && req.body.pass == 'asd1234') {  
    const payload = {  
      check: true  
    };  
    const token = jwt.sign(payload, app.get('key'), {  
      expiresIn: '3d'  
    });  
    res.json({msg: "Se encuentra logueado con el login", token: token})  
  } else {  
    res.json({msg: "El usuario y contraseña no son correctos"});  
  }  
});  
  
// Configuramos el puerto de conexión  
app.listen(5000, () => {  
  console.log('El servidor esta conectado en http://localhost:5000 ');  
});  
  
// ejecuta la aplicación con ' nodemon app ' si sale un error con que no reconoce el comando
```


```
// ejecuta npm install -g nodemon, para instalar nodemon de forma global.
```

12.04.2024 S24-M2

```
// Podemos mostrar en pantalla algo como :  
app.get('/', (req, res) => {  
  res.send('Hola mundo 😊');  
});
```

Probando desde Postman, enviado los siguientes datos:

POSTMAN

 Copiar código

```
{  
  "usuario" : "admin",  
  "pass" : "asd1234"  
}
```

NOTA

'**usuario**' y '**pass**' salen de nuestra condición en *app.js*. Solo si la CONDICIÓN SE CUMPLE se genera el "**token**".

Cuando los datos son correctos se genera el token:

POST http://localhost:5000/login

Body

```
{
  "usuario": "admin",
  "pass": "asd1234"
}
```

Status: 200 OK Time: 26 ms Size: 436 B

Body

```
{
  "msg": "Se encuentra logueado con el login",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJjaGVjaWVjaWV6dHJ1ZS1iaW0iOiJoxNzEzMjMjcyLCJleHAiOiE3MTM0OTQ8NzJ9.LOWSM3g1Az4n4V6AaRjICV1XS80hIkDPXUs7txU-7dU"
}
```

De lo contrario, cuando no son correctos, no se genera el token y se muestra el mensaje de error:

POST http://localhost:5000/login

Body

```
{
  "usuario": "admin",
  "pass": "pasworInvalid"
}
```

Status: 200 OK Time: 5 ms Size: 286 B

Body

```
{
  "msg": "El usuario y contraseña no son correctos"
}
```

Bryan Hernández | Telento Tech DWFSV2-42 | 2024

