Armado login: hashing

DigitalHouse >
Coding School

“

Cuando trabajamos con **datos sensibles** es fundamental almacenarlos **encriptados**, para **preservar** la información en caso de que un tercero acceda a ella.

”

¿Qué es un hash?

En informática, las funciones de hasheo nos permiten **encriptar** datos. Es decir, **transformar** un texto plano en una nueva serie de caracteres —con una longitud fija— imposible de descifrar para el ojo humano.

En este tipo de encriptación (de una sola vía) el dato encriptado no puede volver a desencriptarse.

Es por eso que estas funciones vienen acompañadas de dos características principales:

- La opción de **encriptar un dato**.
- La opción de **comparar** un dato entrante con un dato **hasheado** para verificar si coincide o no.

El paquete **bcryptjs** nos permite incorporar estas funciones en nuestro proyecto de Node.js.

Para usarlo hay que instalarlo a través de npm.

```
>_  npm i bcryptjs
```

.hashSync()

Es un **método** que trae el paquete `bcryptjs` que nos va a permitir encriptar datos. Recibe dos parámetros:

- El **dato** que queremos encriptar.
- La **sal** que le queremos añadir a la encriptación.

¿Qué es la sal?

Un pequeño dato añadido que hace que los hash sean significativamente más difíciles de romper. En este contexto se le suele pasar 10 o 12.

```
{}
const bcrypt = require('bcryptjs');
let passEncriptada = bcrypt.hashSync('monito123', 10);
```

.compareSync()

Es un **método** que trae el paquete `bcryptjs` que nos va a permitir **comparar** un texto plano contra un hash para saber si coinciden o no.

Este método **retorna** un **booleano** y recibe dos parámetros:

- El primero, el **texto plano**.
- El segundo, el **hash** con el que lo queremos comparar.

```
{}
let check = bcrypt.compareSync('monito123', passEncriptada);
console.log(check); // true
```


DigitalHouse>
Coding School