Armado login

DigitalHouse >
Coding School

Implementar session

- Instalar el módulo `express-session` con npm:

```
>_  npm i express-session
```

- Requerirlo en el entry point de la aplicación:

```
{}  const session = require('express-session');
```

- Configurarlo como middleware a nivel aplicación. Ejecutamos `session()` pasándole como argumento un objeto literal con la propiedad `secret` con un texto único aleatorio, que servirá para identificar nuestro sitio web.

```
{}  app.use(session( {secret: "Nuestro mensaje secreto"}));
```

Implementar session

- Al momento de querer **definir** y **almacenar** información, llamamos a la propiedad session del objeto request:

```
{}  req.session.colorFondo = 'Violeta';
```

- Para **leer** información de session:

```
{}  let colorFondo = req.session.colorFondo;
```

Toda lo información que almacenemos en la variable `session` estará disponible para usar en **todas** las vistas del sitio.

Implementar cookies

- Instalar el módulo `cookie-parser` con npm. (Con express-generator ya viene incluído este módulo)

```
>_  npm i cookie-parser
```

- Para **crear** una cookie y **guardar** información en ella, ejecutamos el método `cookie()` sobre el objeto `response`, pasándole dos argumentos:
  - El **nombre** que le queremos asignar a esa cookie.
  - El **valor** que tendrá.

```
{}  res.cookie('club', 'C. A. Tigre');
```

El nombre de la cookie que definimos será una propiedad de `cookies`.

Leyendo las cookies

- Para **leer** información de una cookie usamos el objeto `request`, llamando al objeto `cookies`, seguido del nombre de la cookie que definimos anteriormente:

```
{}  console.log(req.cookies.club);
```

.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