EXERCISES QUE TRABAJAREMOS EN EL CUE

- EXERCISE 1: GENERANDO UN TOKEN

EXERCISE 1: GENERANDO UN TOKEN

El objetivo de este ejercicio es el entendimiento de la generación de un TOKEN, así como verificar los conceptos básicos de los componentes de un JWT (Header, Playload y el Signature).

Principales pasos que se realizarán para visualizar los componentes de un JWT (To-Do):

1- Conceptos básicos de un Token y un JSON Web Token.

2- Generar un token aleatorio desde la terminal.

3- Generar un JWT desde la terminal de Linux.

CONCEPTOS BÁSICOS DE UN TOKEN Y UN JSON WEB TOKEN

- **¿QUÉ ES UN TOKEN?**

  Es una cadena alfanumérica con caracteres aparentemente aleatorios, como el siguiente:

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9eyJzdWIiOiIxMjM0NTY3OdkwIiwi
bmFtZSI6Ikp
```

  Estas cadenas de texto en sí pueden no expresar algún significado, pero el emisor o servidor que lo emitió si lo puede entender, y así validar al usuario que intenta acceder a la información, e incluso, puede tener datos adicionales.

- **¿QUÉ SON LOS JSON WEB TOKENS?**

  Son un tipo de token que contiene una estructura, la cual puede ser desencriptada por el servidor y de esta forma, autenticarnos como usuario en la aplicación.

Estructura de un JWT:

aaaaaaaaaaaaaa.bbbbbbbbbbbbbb.ccccccccccccc
HEADER PAYLOAD SIGNATURE

HEADER: la primera parte, donde se almacena generalmente el tipo de token, y el algoritmo de encriptamiento.

PAYLOAD: la segunda parte, que contiene los datos que identifican al usuario, como puede ser: su ID, nombre de usuario, entre otros.

FIRMA o **SIGNATURE:** la tercera parte, la cual se genera con las secciones anteriores, y sirve para validar que el contenido no haya sido alterado, partiendo de una clave secreta o token.

Ejemplo de **HEADER:**

```
{
   "alg":"HS256",
   "typ":"JWT"
}
```

Ejemplo de **PLAYLOAD:**

```
{
   "iss":"test.com",
   "exp":1470839345,
   "name":"Pedro Perez"
}
```

SIGNATURE:

```
        clave_secreta: "esta es mi clave"
        clave_secreta: bigsecretisveryhardtoguessbysneakypeopleright
```

En base hexadecimal es:

```
clave_secreta:
727968617264746f677565737362797 36e65616b7970656f706c6572696768
```

En este sentido, el JWT viene dado:

```
Y = Base64URLEncode(HEADER) + '.' + Base64URLEncode(PAYLOAD)
   JWT = Y + '.' + Base64URLEncode(HMACSHA256(Y))
```

GENERAR UN TOKEN ALEATORIO DESDE LA TERMINAL Y COLOCARLO TOKENS DE SEGURIDAD

Para generar un token o clave secreta verdaderamente aleatorio desde la terminal, lo podemos hacer de la siguiente manera:

```
$ node
Welcome to Node.js v12.16.2.
Type ".help" for more information.
> require("crypto").randomBytes(64).toString("hex")
'c10f110263f0526dea36879b300212903b84d272b22b55dd5d48aafb8cfd7d3be8a91a196a
a3d5fcdfc73a9400c6003a6c8a7601f961fdd56535897bf94bdf81'
```

GENERAR UN JWT DESDE LA TERMINAL DE LINUX

Creamos un directorio para crear el script y aperturar el Visual Studio Code:

```
mkdir generarJWT
cd generarJWT
code .
```

- **CONSTRUCCIÓN DEL HEADER:**

```
# Construcción del HEADER
jwt_header=$(echo -n '{"alg":"HS256","typ":"JWT"}' | base64 | sed
s/\+/-/g | sed 's/\//_/g' | sed -E s/=+$//)

echo "HEADER: "$jwt_header
```

SALIDA EN LA TERMINAL:

```
$ sh generateJWT.sh
HEADER: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
```

- **CONSTRUCCIÓN DEL PAYLOAD:**

```
# Construcción del payload
payload=$(echo -n '{"email":"jordan@example.com"}' | base64 | sed
s/\+/-/g sed 's/\//_/g' |  sed -E s/=+$//)
echo "\nPAYLOAD: "$payload
```

SALIDA EN LA TERMINAL:

```
$ sh generateJWT.sh
HEADER: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

PAYLOAD: eyJlbWFpbCI6ImpvcmRhbkBleGFtcGxlLmNvbSJ9
```

- **CLAVE SECRETA:**

```
clave_secreta='mi_clave_secreta'
echo -n "\nClave Secreta: "$clave_secreta
```

SALIDA EN LA TERMINAL:

```
$ sh generateJWT.sh
HEADER: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

PAYLOAD: eyJlbWFpbCI6ImpvcmRhbkBleGFtcGxlLmNvbSJ9

Clave Secreta: mi_clave_secreta
```

- **CLAVE SECRETA A HEXADECIMAL:**

```
# Convertir la clave secreta a hexadecimal (no base64)
Hexsecreta=$(echo -n "$clave_secreta" | xxd -p | tr -d '\n')
echo "\nClave en Hexadecimal: " $hexsecreta
```

- **FIRMAR LA CLAVE SECRETA:**

```
# Generar la firma hmac -- se debe notar que se esta pasando key como bytes hexadecimal
hmac_signatureHex=$(echo -n "${jwt_header}.${payload}" |  openssl dgst -sha256 -mac HMAC -macopt hexkey:$hexsecreta -binary | base64  | sed 's/\+/-/g' | sed 's/\//_/g' | sed -E 's/=+$//')
echo "\nFirma o SIGNATURE: "$hmac_signatureHex
```

SALIDA EN LA TERMINAL:

```
$ sh generateJWT.sh
HEADER: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

PAYLOAD: eyJlbWFpbCI6ImpvcmRhbkBleGFtcGxlLmNvbSJ9

Clave Secreta: mi_clave_secreta
Clave en Hexadecimal:  6d695f636c6176655f73656372657461
```

- **GENERAR LA FIRMA:**

```
$ sh generateJWT.sh
HEADER: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

PAYLOAD: eyJlbWFpbCI6ImpvcmRhbkBleGFtcGxlLmNvbSJ9

Clave Secreta: mi_clave_secreta
Clave en Hexadecimal:  6d695f636c6176655f73656372657461

Firma o SIGNATURE: 7vK90WXb6tB72TA_CaZhesoEuqR8IAQcat2Zj-lsI3Y
```

- **CONSTRUCCIÓN DEL JWT:**

```
# Creando el token completo
jwt="${jwt_header}.${payload}.${hmac_signatureHex}"
echo "\nJSON Web TOKEN (JWT): "$jwt
```

SALIDA EN LA TERMINAL:

```
$ sh generateJWT.sh
HEADER: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

PAYLOAD: eyJlbWFpbCI6ImpvcmRhbkBleGFtcGxlLmNvbSJ9

Clave Secreta: mi_clave_secreta
Clave en Hexadecimal:  6d695f636c6176655f73656372657461

Firma o SIGNATURE: 7vK90WXb6tB72TA_CaZhesoEuqR8IAQcat2Zj-lsI3Y
(base) luispc@FullStack-Dev:code$ sh 3_generateJWT.sh
HEADER: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9

PAYLOAD: eyJlbWFpbCI6ImpvcmRhbkBleGFtcGxlLmNvbSJ9

Clave Secreta: mi_clave_secreta
Clave en Hexadecimal: 6d695f636c6176655f73656372657461

Firma o SIGNATURE: 7vK90WXb6tB72TA_CaZhesoEuqR8IAQcat2Zj-lsI3Y
```

JSON WEB TOKEN (JWT):

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJlbWFpbCI6ImpvcmRhbkBleGFtcGxlL
mNvbSJ9.7vK90WXb6tB72TA_CaZhesoEuqR8IAQcat2Zj-lsI3Y
```

NOTA:

Para una mejor resolución del proyecto, es recomendable tener las librerías base64 y openssl.