## TEXT CLASS REVIEW

## TEMAS A TRATAR EN LA CUE:

- Servidor HTTP.
- Verbos o métodos HTTP.
- Componentes de una petición HTTP.
- ¿Qué es JSON?
- ¿Qué es una autenticación con JWT?
- Encriptación de datos.

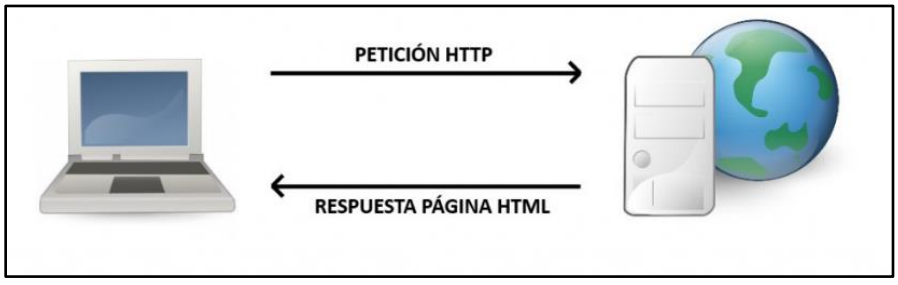
### ¿SERVIDOR HTTP?

`Hypertext Transfer Protocol`: El Protocolo de transferencia de hipertexto, es el protocolo de comunicación que permite las transferencias de información en la `World Wide Web`.

Comunicación cliente-servidor:

- Es el intercambio de información entre cliente y servidor.
- El servidor queda a la espera de alguna solicitud `HTTP`, ejecutada por el cliente, y proporciona una respuesta.

Cuando visitamos un sitio web, hacemos una solicitud `GET` de `HTTP`, y el servidor nos devuelve, por ejemplo, un `index.html` con el sitio web.



## VERBOS O MÉTODOS HTTP

`HTTP` define un conjunto de métodos de petición, para indicar la acción que se desea realizar en un recurso determinado.

Los más populares son: `GET, POST, PUT, DELETE`.

- **GET**: solicita una representación de un recurso específico. Las peticiones que usan el método `GET` sólo deben recuperar datos.
- **POST**: se utiliza para enviar una entidad a un recurso en específico, causando a menudo, un cambio en el estado o efectos secundarios en el servidor.
- **PUT**: reemplaza todas las representaciones actuales del recurso de destino, con la carga útil de la petición.
- **DELETE**: borra un recurso en específico.

## COMPONENTES DE UNA PETICIÓN HTTP

1. **Línea de inicio**

   Las peticiones `HTTP` son mensajes enviados por un cliente, para iniciar una acción en el servidor. Su línea de inicio está formada por tres elementos:

   - *Un método HTTP,* un verbo como: `GET, PUT o POST`, o un nombre como: `HEAD` (en-US) o OPTIONS (en-US), que describan la acción que se pide sea realizada. Por ejemplo: `GET` indica que un archivo ha de ser enviado hacia el cliente, o `POST` indica que hay datos que van a ser enviados hacia el servidor (creando o modificando un recurso, o generando un documento temporal para ser enviado).
   - *El objetivo de una petición,* que normalmente es una `URL`, o la dirección completa del protocolo, puerto y dominio, también suelen ser especificados por el contexto de la petición. Su formato varía según los distintos métodos `HTTP`.
     Puede ser:
     - Una dirección absoluta, seguida de un signo de cierre de interrogación '?', y un texto de consulta. Este es el formato más común, conocido como el formato original (origin form en inglés), y se usan en los métodos: `GET, POST, HEAD y OPTIONS`.

POST / HTTP 1.1
GET / background.png HTTP / 1.0
HEAD / test.html?query=alibaba HTTP / 1.1
OPTIONS / anypage.html HTTP / 1.0

- Una `URL` completa, conocido como el formato absoluto, usado mayormente con `GET` cuando se conecta a un proxy.
  GET http://developer.mozilla.org/en-US/docs/Web/HTTP/Messages HTTP/1.1
- El componente de autoridad de una `URL`, formado por el nombre del dominio, y opcionalmente el puerto (precedido por el símbolo ':'), se denomina a éste como el formato de autoridad. Únicamente se usa con `CONNECT`, cuando se establece un túnel `HTTP`.
  `CONNECT` devloper.mozilla.org:80 HTTP/1.1
- El formato de asterisco, se utiliza ('*') junto con las opciones: `OPTIONS`, representando al servidor entero en conjunto.
  OPTIONS * HTTP/1.1

- *La versión de HTTP*, la cual define la estructura de los mensajes, actuando como indicador de la versión que espera que se use para la respuesta.

## 2. Headers o Cabeceras

Las cabeceras `HTTP` de una petición, siguen la misma estructura que la de una cabecera `HTTP`: una cadena de caracteres, que no diferencia mayúsculas ni minúsculas; seguida por dos puntos (':'); y un valor cuya estructura depende de la cabecera. La cabecera completa, incluido el valor, ha de ser formada en una única línea, y puede ser bastante larga.

Hay bastantes cabeceras posibles. Estas se pueden clasificar en varios grupos:

- Cabeceras generales, ('General headers' en inglés), como Via (en-US), afectan al mensaje como una unidad completa.
- Cabeceras de petición, ('Request headers' en inglés), como: User-Agent, Accept-Type, modifican la petición especificándola en mayor detalle, como: Accept-Language (en-US), o dándole un contexto, como: Referer, o restringiéndola condicionalmente, como: If-None.

- Cabeceras de entidad, ('Entity headers' en inglés), como Content-Length, las cuales se aplican al cuerpo de la petición. Por supuesto, ésta no necesita ser transmitida si el mensaje no tiene cuerpo ('body' en inglés).


3. **Cuerpo**

Es la parte final de la petición. No todas las peticiones llevan uno: las que reclaman datos, como: `GET, HEAD, DELETE, o OPTIONS`, regularmente no necesitan ningún cuerpo. Algunas peticiones pueden enviarse al servidor, con el fin de actualizarlo; como es el caso de la petición `POST`, que contiene datos de un formulario `HTML`.

Los cuerpos pueden ser divididos en dos categorías:

- Cuerpos con un único dato, que consisten en un único archivo, definido por las dos cabeceras: `Content-Type` y `Content-Length`.
- Cuerpos con múltiples datos, que están formados por distintos contenidos, y generalmente están asociados con los formularios `HTML`.

## ¿QUÉ ES JSON?

Sus siglas en inglés `JavaScript Object Notation`, o sea, Notación de Objeto JavaScript. Se trata de un lenguaje usado para el intercambio de datos entre sistemas.

Básicamente, usa la misma notación o forma con la que se escriben los objetos JavaScript en el código, con algunas restricciones y añadidos extra.

Su utilidad es la de intercambiar datos, por eso, se conoce como lenguaje de intercambio de información, o lenguaje de transporte. Sirve para la comunicación entre servicios web (web services) y los clientes que los consumen, enviando y recibiendo la información en formato JSON.

### Características de JSON

- Es un lenguaje de modelador de datos.
- Consiste en pares "clave - valor".
- Los valores pueden ser cadenas, números o booleanos, así como otros objetos JSON, con cualquier nivel de anidación.
- Es un formato flexible, ligero y fácilmente transferible a través de las redes.

### Ventajas de JSON

- La lectura del código resulta fácil de entender, y la información es suficientemente expresiva para poder ser leída por personas, además de máquinas.
- El tamaño de los archivos que se transfieren es ligero.
- El código está basado en el lenguaje JavaScript, lo que es ideal para las aplicaciones web.
- Todos los lenguajes disponen de funciones para interpretar cadenas JSON, y convertir datos en cadenas válidas.
- Se escribe en archivos de texto plano con codificación UTF8, que es compatible con todos los sistemas.

**Sintaxis**

Las reglas sintácticas de Json son bastantes sencillas:

- Existen dos tipos de elementos:
  - Matrices (Arrays): son listas de valores separados por comas. Se escriben entre corchetes [ ].

```
[1, "pepe", 3.14, "Pepito Conejo"]
```

  - objetos (objects): son listas de parejas nombre / valor. Están separados por dos puntos (:), y las parejas están separadas por comas. Los objetos se escriben entre llaves {}, y los nombres de las parejas se escriben siempre entre comillas dobles.

```
{"nombre":"José Romero", "edad": 26, "carnet de conducir": true}
```

- Un documento JSON está formado por un único elemento (un objeto o matriz), y es incorrecto en las siguientes formas:

1.

```
[1, 2, 3] , ["a","b", "c"]
```

2.

```
"nombre":"José Romero"
```

- Tanto en los objetos como en los matices, el último elemento no puede ir seguido de una coma, y es incorrecto las siguientes formas:

1.
```
{"nombre":"José Romero",}
```

2.
```
{"nombre":"José Romero", "edad": 26, "carnet de conducir": true, }
```

- Los espacios en blanco, y los saltos de línea, no son significativos, es decir, puede haber cualquier número de espacios en blanco, o saltos de línea separando cualquier elemento o símbolo del documento.

```
[
    {
        "nombre": "José Romero",
        "edad": 26,
        "carnet de conducir": true
    },
    {
        "nombre": "Jocelyn Alvarado",
        "edad": 90,
        "carnet de conducir": false
    }
]
```

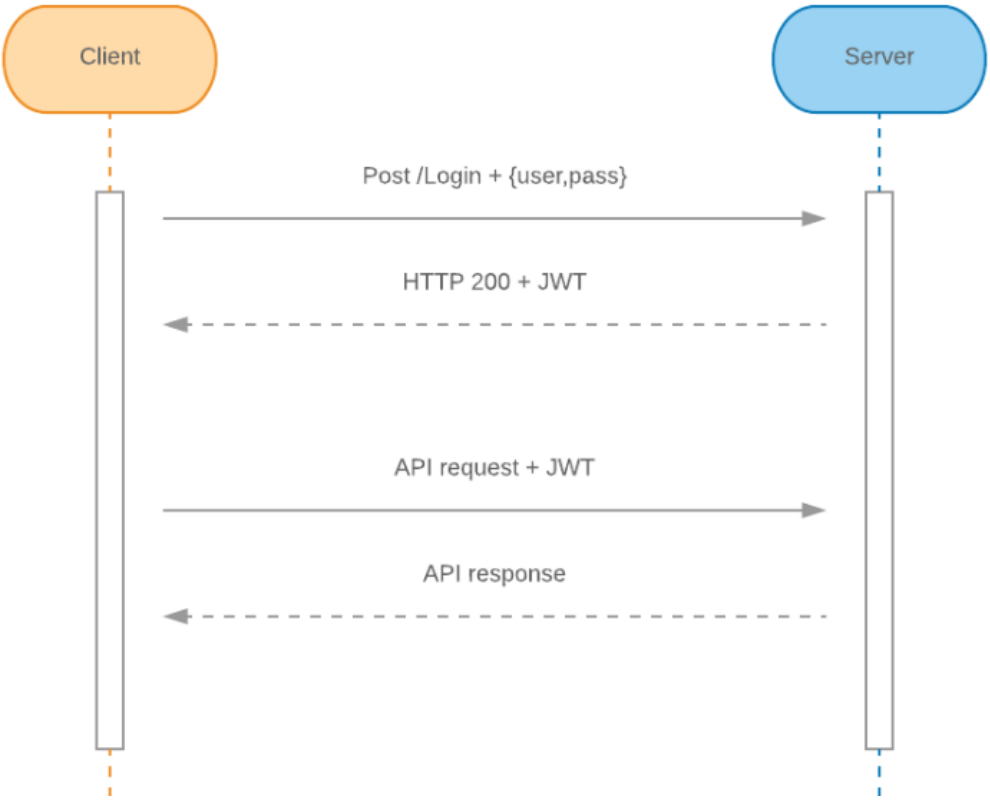
- Los valores (tanto en los objetos, como en las matrices) pueden ser:
    - Números: enteros, decimales o en notación exponencial. El separador decimal es el punto (.); un número negativo empieza por el signo menos (-); y el indicador de la notación exponencial, es e o E. Los números positivos no deben llevar signo, ni pueden empezar por varios ceros, o por un cero seguido de otra cifra.
    - Cadenas: se escriben entre comillas dobles. Los caracteres especiales, y los valores Unicode, se escriben con una contrabarra \ delante. Aquellos que siempre deben escribirse como caracteres especiales, son: \" (comillas); \\ (contrabarra); \b (retroceso); \f (salto de página); \n (salto de línea); \r (retorno de carro); \t (tabulador); y los caracteres Unicode (\u...). El carácter / (barra), puede escribirse como carácter /, o como carácter especial \/ (suele ser necesario cuando el contenido es código html, y la barra indica un cierre de etiquetas).
    - Los valores true, false y null: se escriben sin comillas.
    - Objetos y matrices: puede haber objetos y matrices dentro de éstos mismos, sin límite de anidamiento.
- Los ficheros JSON no pueden contener comentarios.

**¿QUÉ ES UNA AUTENTICACIÓN CON JWT?**

**JWT** es un sistema de autenticación, que se basa en el uso de tokens. El caso más común en el que se emplea, es para manejar la autenticación en aplicaciones móviles o web. Para esto, cuando el usuario se quiere autenticar, manda sus datos de inicio de la sesión al servidor, éste genera el **JWT,** y se lo manda a la aplicación cliente, luego, en cada petición, el cliente envía este token que el servidor usa, para verificar que el usuario esté correctamente autenticado y saber quién es.

El token está firmado, por lo que no se puede modificar. La página web lo guarda, y cuando el usuario accede a otra, o ejecuta una acción en la página que depende del servidor, lo envía.

En resumidas cuentas, se genera un token en el servidor, la página se lo guarda en la cookie o en memoria, y por cada petición a la API, se envía para comprobar si el usuario tiene permisos. De esta forma, se logran proteger llamadas a la API, sin necesidad de pasar usuario y contraseña en cada petición (pues solo se pasa el token).




**Composición del token**

Para nosotros, un Token JWT, será una cadena de texto compuesta por tres partes codificadas en Base64; cada una de éstas será separada por un punto (.), como se muestra a continuación:

**HEADER.PAYLOAD.SIGNATURE**

¿Qué significan estas tres partes?:

- Header: consta generalmente de dos valores, y proporciona información importante sobre el token. Contiene el tipo de token (propiedad typ), y el algoritmo de la firma (propiedad alg, y/o cifrado utilizados.

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

- Payload: contiene la información real que se transmitirá a la aplicación, como por ejemplo: un id, un name, entre otros. Aquí se definen algunos estándares, que determinan qué datos se transmiten y cómo. La información se proporciona como pares key/value (clave-valor); las claves se denominan claims en JWT. Hay tres tipos diferentes de claims: registrados, públicos y privados.

```
{
    "sub": "1234567890",
    "name": "John Doe",
    "iat": 1516239022
}
```

- Signature: es una firma que se genera usando los anteriores dos campos (header y pyload), en base64 y una key secreta (que solo se sepa en los servidores que creen o usen el **JWT**), para usar un algoritmo de encriptación. La forma de hacerlo (usando pseudo código), sería la siguiente:

```
key =  'secret'
unsignedToken = base64Encode(header) + '.' + base64Encode(payload)
signature = SHA256(key, unsignedToken)
```

- El token: aquí, las tres partes: head, payload y signatura, son concatenadas separadas por puntos.

```
key =  'secret'
unsignedToken = base64Encode(header) + '.' + base64Encode(payload)
signature = SHA256(key, unsignedToken)
token = unsignedToken + '.' + signature
```

**ENCRIPTACIÓN DE DATOS**

Es un método de codificación de datos (mensajes o archivos), de modo que, solo las partes autorizadas puedan leer la información o acceder a ella. La encriptación utiliza algoritmos complejos para codificar la información que se envía. Una vez recibidos, los datos se pueden descifrar con la clave proporcionada por el emisor del mensaje. La eficacia de su tecnología, está determinada por la fuerza del algoritmo, la longitud de la password, y la idoneidad del sistema de encriptación seleccionado.

**Términos importantes sobre encriptación**

- **Algoritmo**: también conocidos como cifrados, son las reglas o instrucciones del proceso de encriptación. Algunos ejemplos: Triple DES, RSA y AES.
- **Desencriptación**: proceso de convertir un texto encriptado ilegible, en información legible.
- **Claves**: una cadena aleatoria de bits, utilizada para encriptar o desencriptar información. Cada clave es única, y las más largas son más difíciles de averiguar. Sus longitudes habituales son 128 y 256 bits en el caso de las claves privadas, y 2048 bits en el caso de las claves públicas.

De esta forma, según la *password,* encontramos dos tipos de encriptación de archivos: simétrico o asimétrico. El **sistema de cifrado simétrico**, es aquel que utiliza una misma clave para cifrar y descifrar; mientras que, en la **encriptación de datos asimétrica,** se usan diferentes claves: una pública para cifrar, y una de carácter privado para descifrar, de forma que sea imposible deducir la contraseña privada a través de la pública.