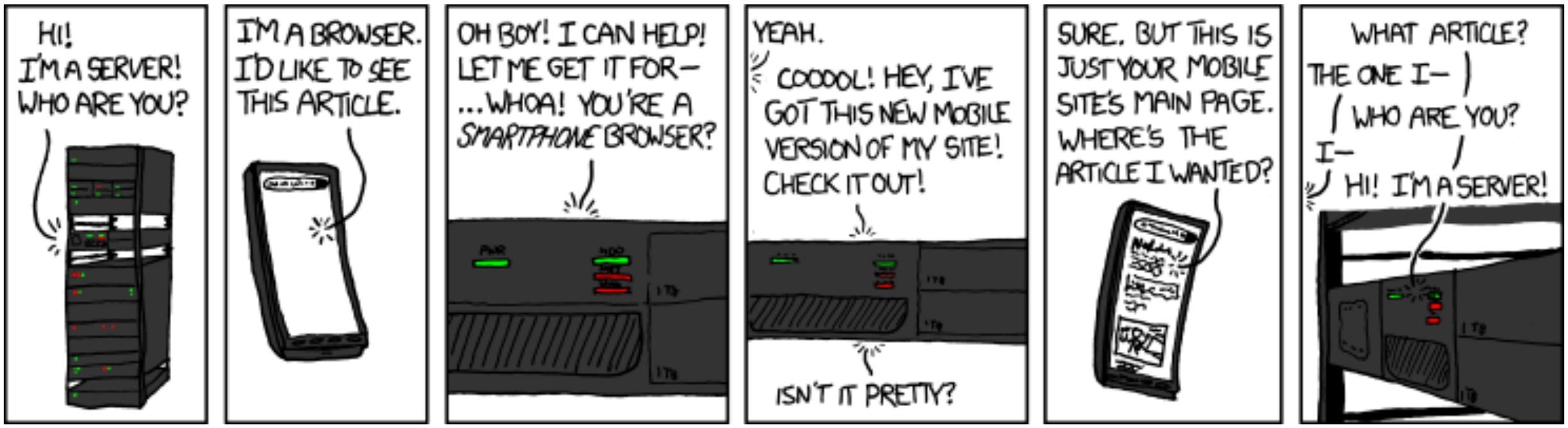
Tema 3: Autentificación en APIs web

Aplicaciones Distribuídas en Internet, curso 2024-25

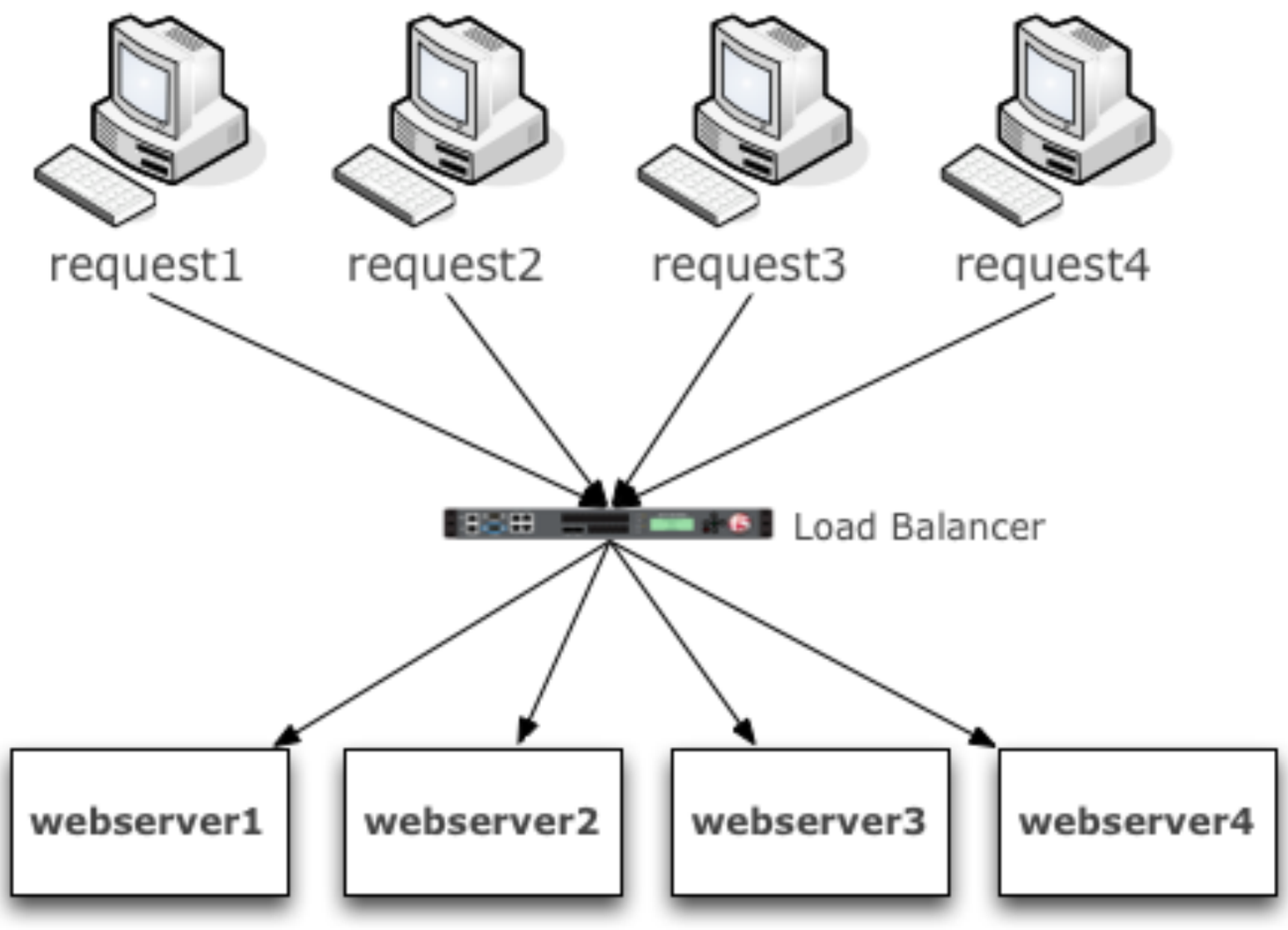
HTTP es un protocolo sin estado



Y sí, la experiencia de los usuarios es que el servidor les “recuerda” mientras navegan por el sitio, esto típicamente se ha hecho siempre a base de cookies, pero dejaremos esto de lado de momento…

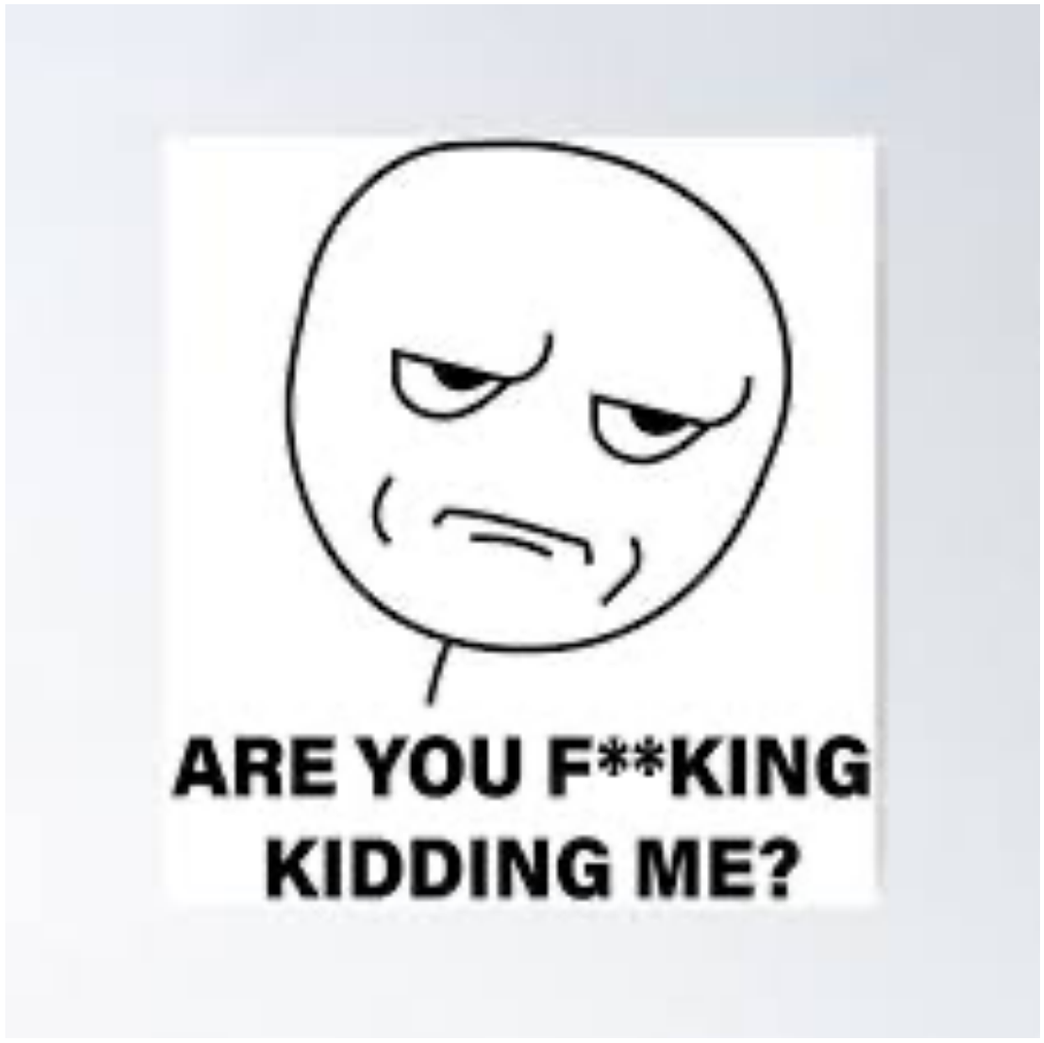
Ventajas de no tener estado

- Es mucho más fácil escalar una app con servidores *stateless*. Al cliente no le importa qué instancia sirva las peticiones, podemos redirigirlas, arrancar nuevos servidores, parar los que ya hay, etc.



Qué significa en realidad “sin estado”

- **El cliente es el responsable** de enviar el **contexto “relevante”** en cada petición/respuesta
- Por ejemplo alguna credencial de autenticación



Tokens

- Valores idealmente **únicos** e **imposibles de falsear** que identifican al cliente
- Normalmente se **obtienen a cambio** de otro mecanismo de autentificación (login+password, identificación biométrica, …)
- Para cualquier operación del API que sea restringida debemos **enviar el token en la petición HTTP**

Tokens en el mundo real (NYC Subway Tokens)

Tokens JWT (JSON Web Token)

- Es un Estándar de IETF. Hay implementación en multitud de lenguajes.
- Un Token JWT es una cadena en formato JSON formada por 3 partes:
  - **Cabecera**: indica el tipo de token y el algoritmo de firma. Se codifica en Base64. Ejemplo: `{"typ":"JWT", "alg":"HS256"}` (indica que esto es un "JWT" y se firmará con HMAC SHA-256)
  - **Payload**: lo que queremos almacenar en el token en formato JSON (p.ej. `{"login":"adi"}`) y codificado en Base64URL
  - **Firma**: se aplica un algoritmo de *hash* sobre la cabecera, el payload y una **clave secreta que solo conoce el servidor** y se pasa a Base64URL
- Las tres partes se concatenan con '.'

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJzYWx1ZG8iOiJob2xhIG11bmRvIn0=.pJPDpr
jxsouVfaaXau-Fyspj6rpKc7_hCui1RSaERAE
```

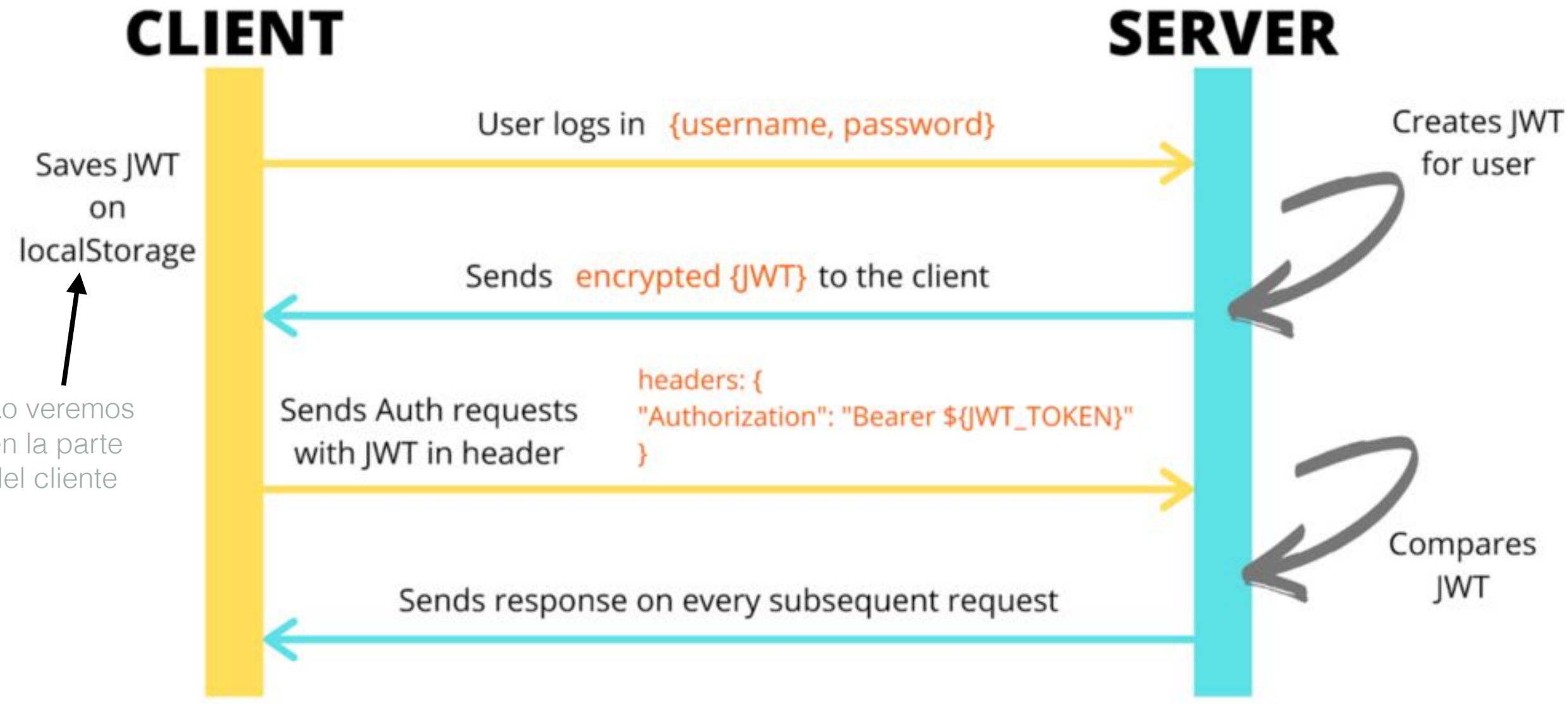
todo se transmite "en claro": Base64 es una codificación, no un cifrado. Por tanto habrá que usar HTTPS si no se quiere que el *payload* sea legible

Comprobar que un token es auténtico

- El servidor toma la **cabecera** y el ***payload*** (recordar que no están cifrados, solo en Base64) y la **clave secreta**, y **vuelve a aplicar el *hash***. Si no coincide con la firma, el token no es válido.

- En teoría no se puede generar un token si no se conoce la clave secreta, y esta no se puede averiguar a partir de un token auténtico (el *hash* no es invertible)

Flujo de uso típico de JWT



En el estándar no se especifica dónde debe enviar el servidor el JWT al cliente, pero sí que para autentificarse el cliente debe enviar al servidor el token en la cabecera HTTP “Authorization”

JWT en Firebase

- Cuando el usuario se autentifica (p.ej. `signInWithEmailAndPassword`) el objeto resultante contiene un **token JWT**
- Si hay un usuario autentificado, **en todas las peticiones se envía automáticamente el *token*** *(junto con muchos más datos del usuario)* aunque no exactamente según dice el estándar
- En las apps web el token se almacena en **indexedDB** (similar a localStorage pero “más sofisticado”, ya lo veremos)