Autoryzacja w aplikacjach webowych

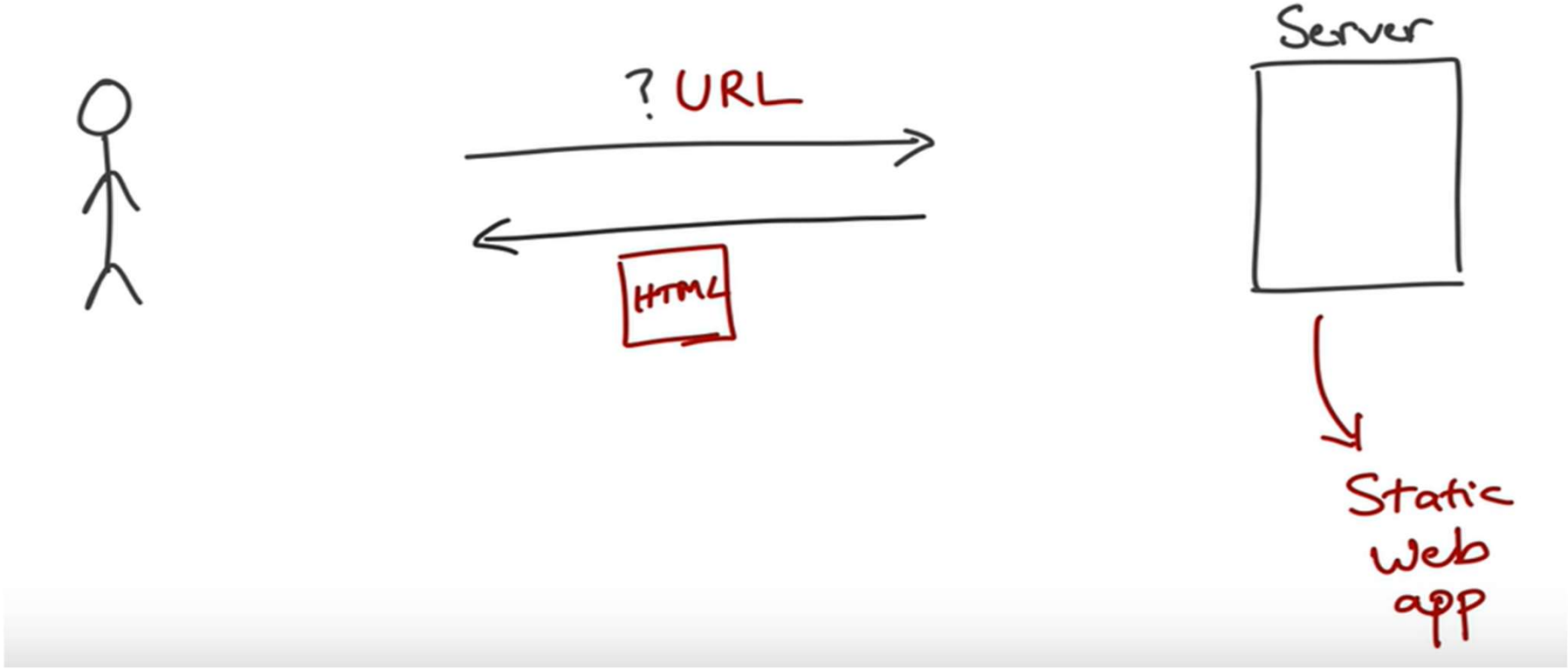
dr inż. Grzegorz Rogus

Strategie autoryzacji

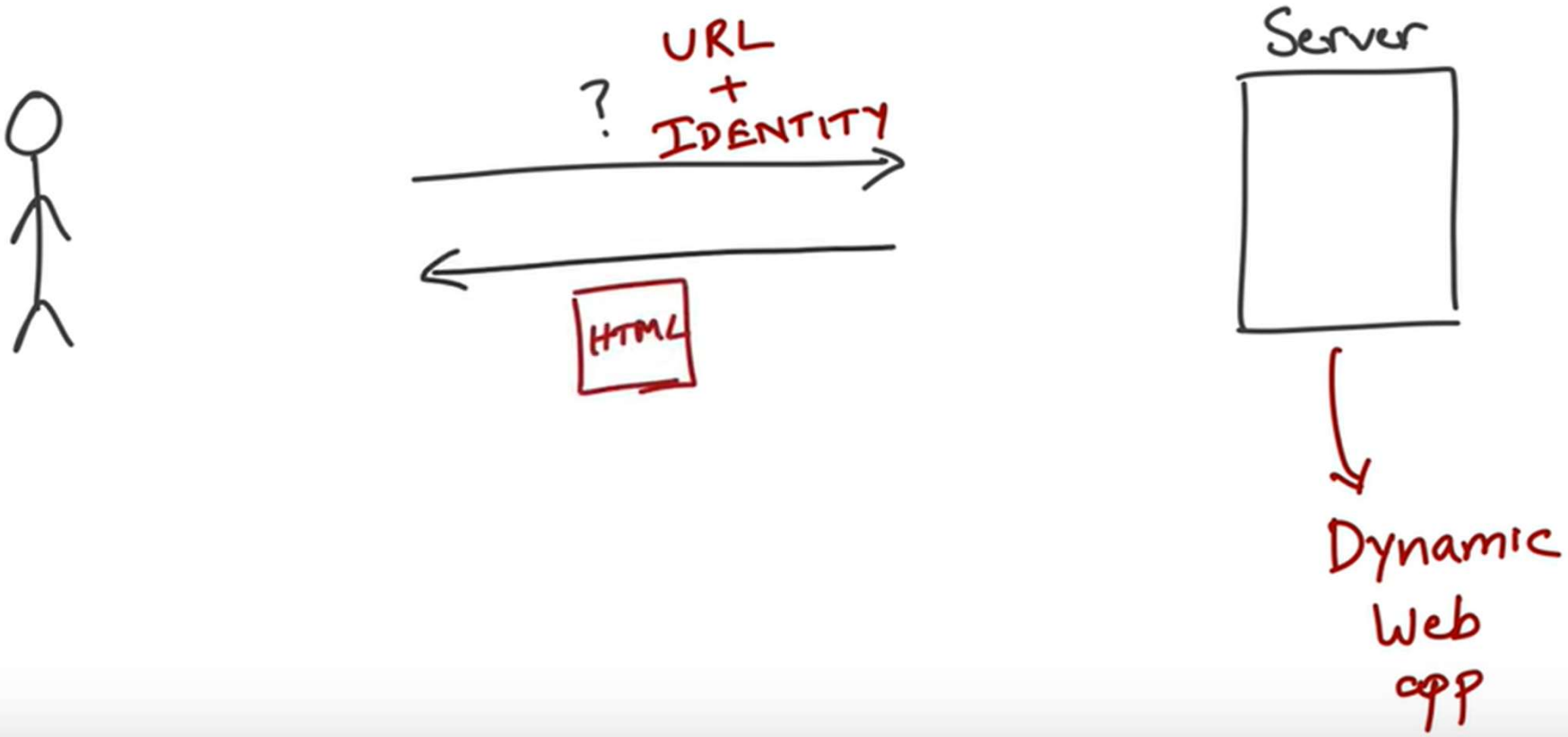
Przy użyciu tokenu

- Session token
- JSON web token

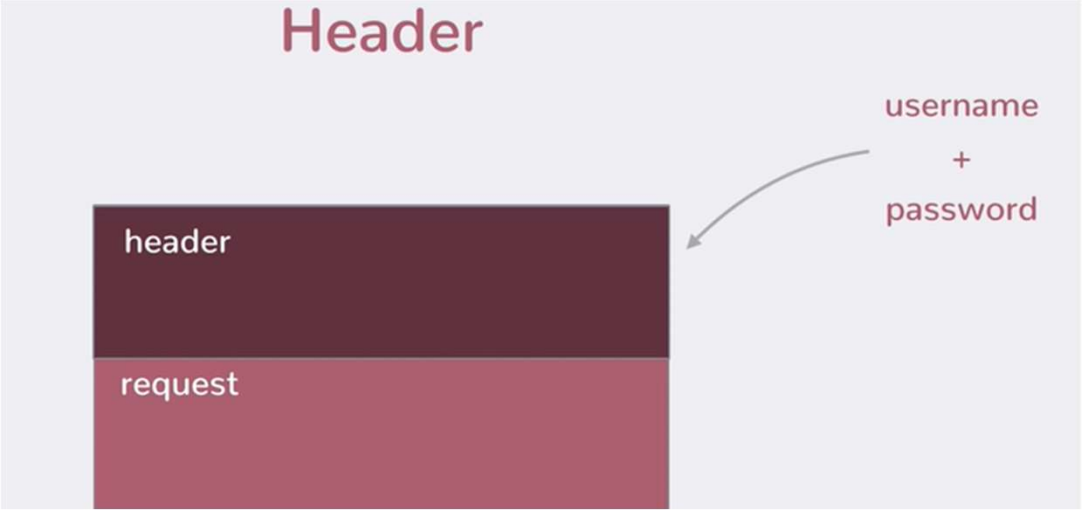
Client – serwer = strona statyczna



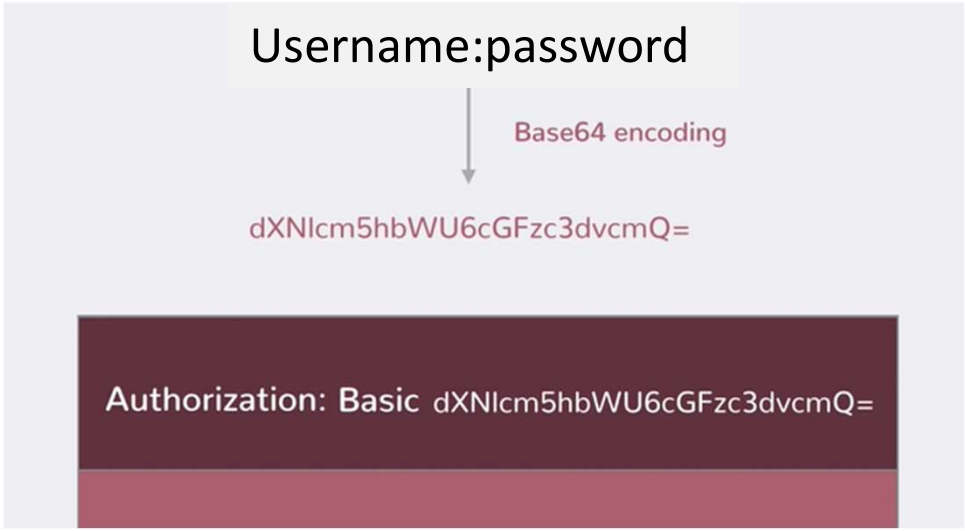
Client serwer – strona dynamiczna



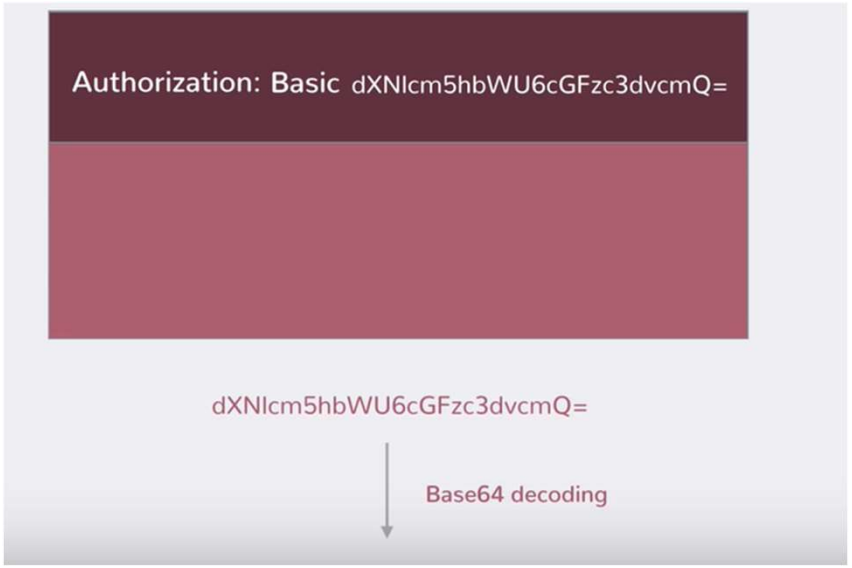
Basic Auth (Basic Access Authentication)



Basic Auth - po stronie klienta

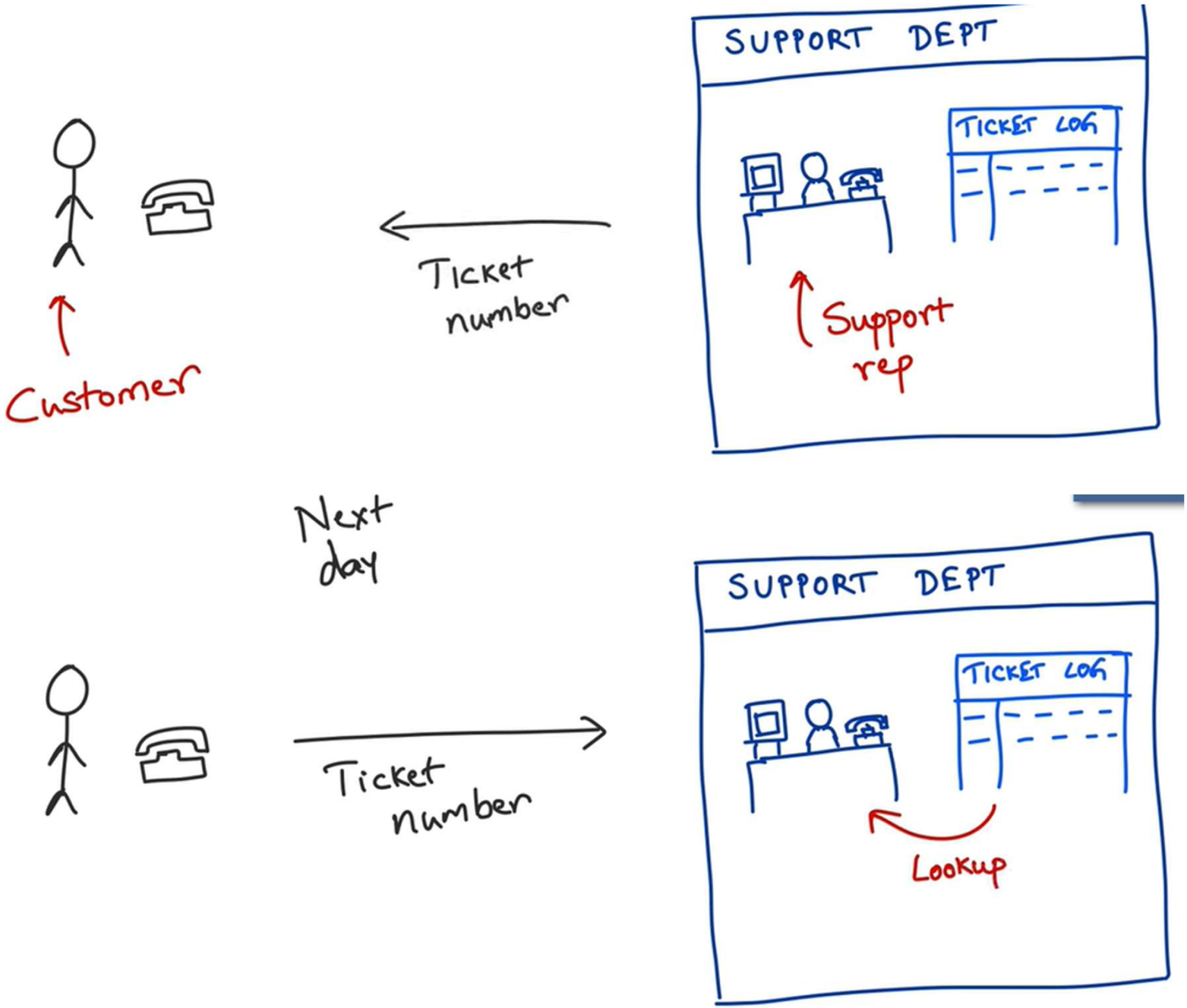


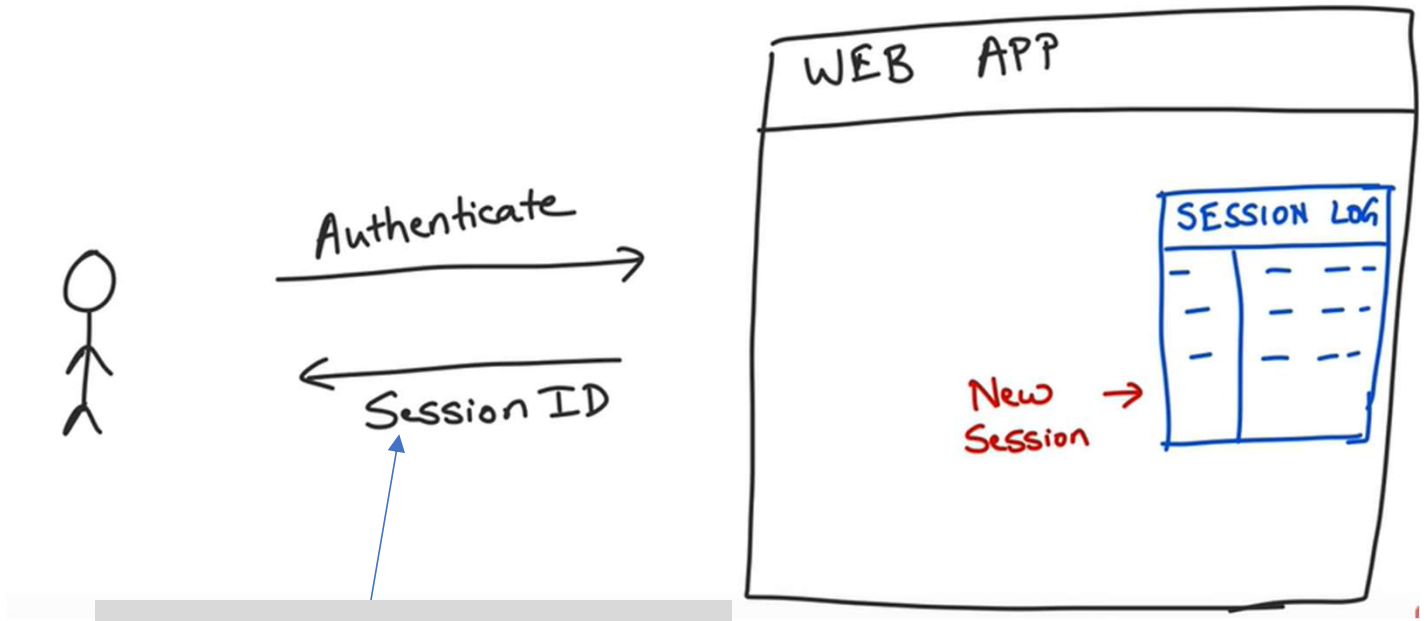
Basic Auth - po stronie servera



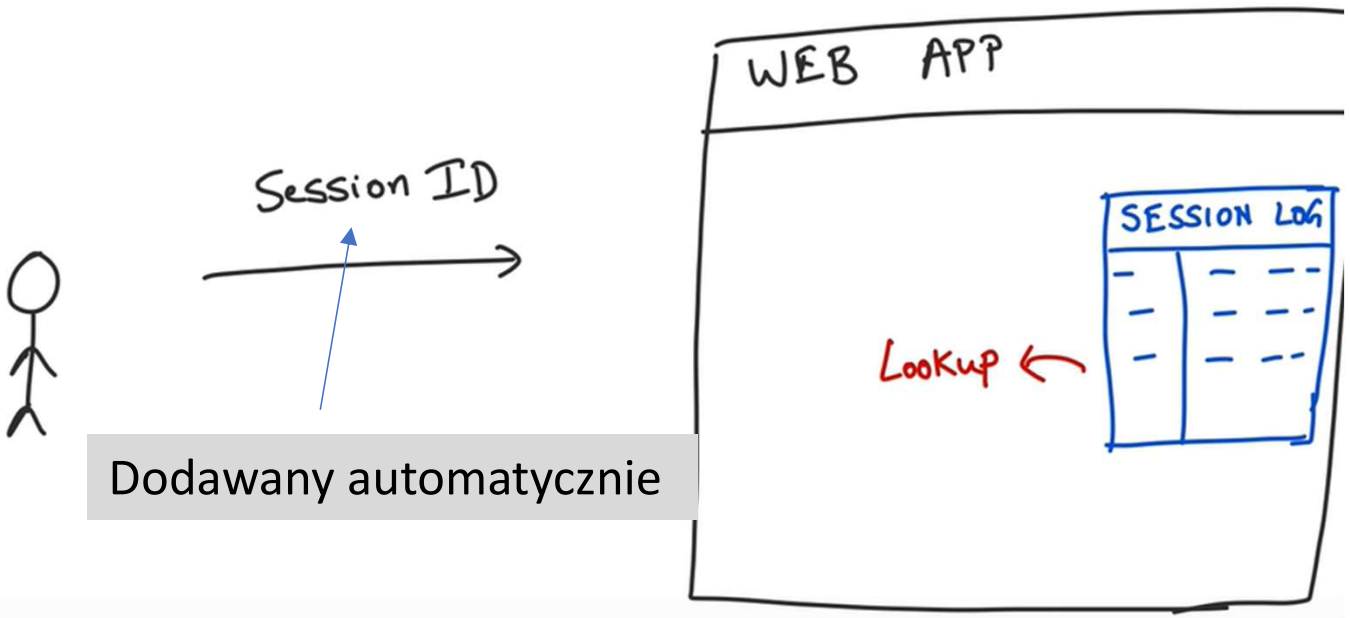
Username:password

Token sesyjny - koncepcja





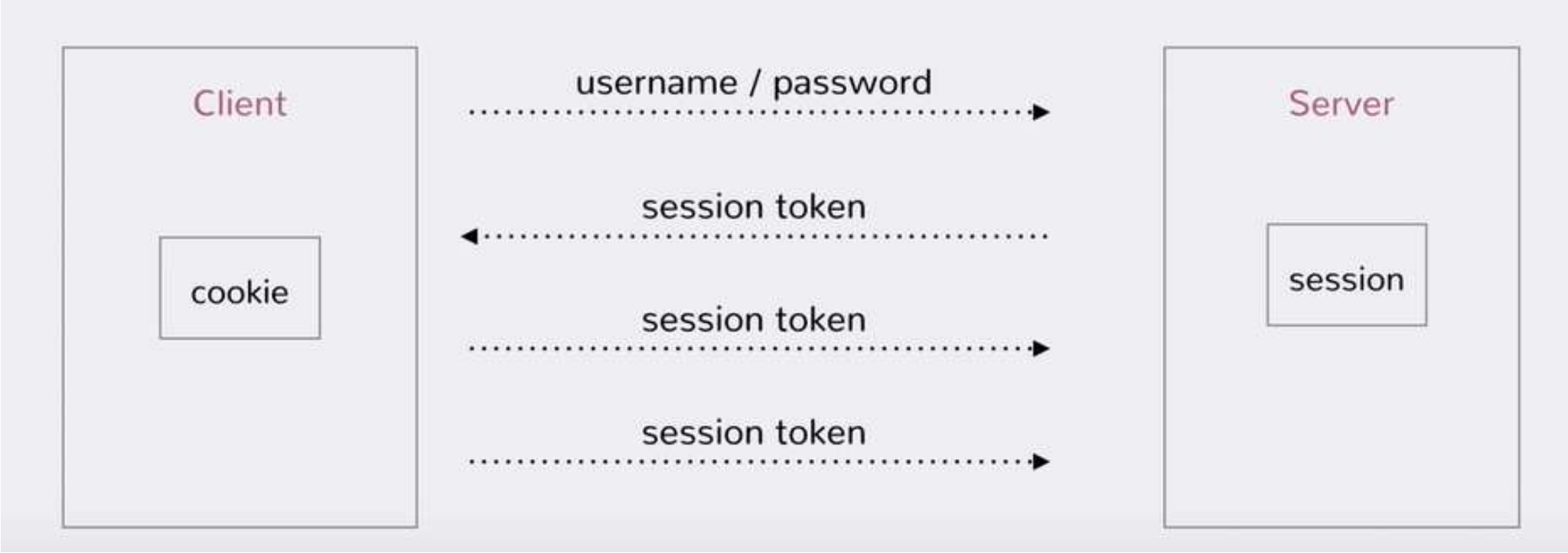
Zwraca jako ciasteczko



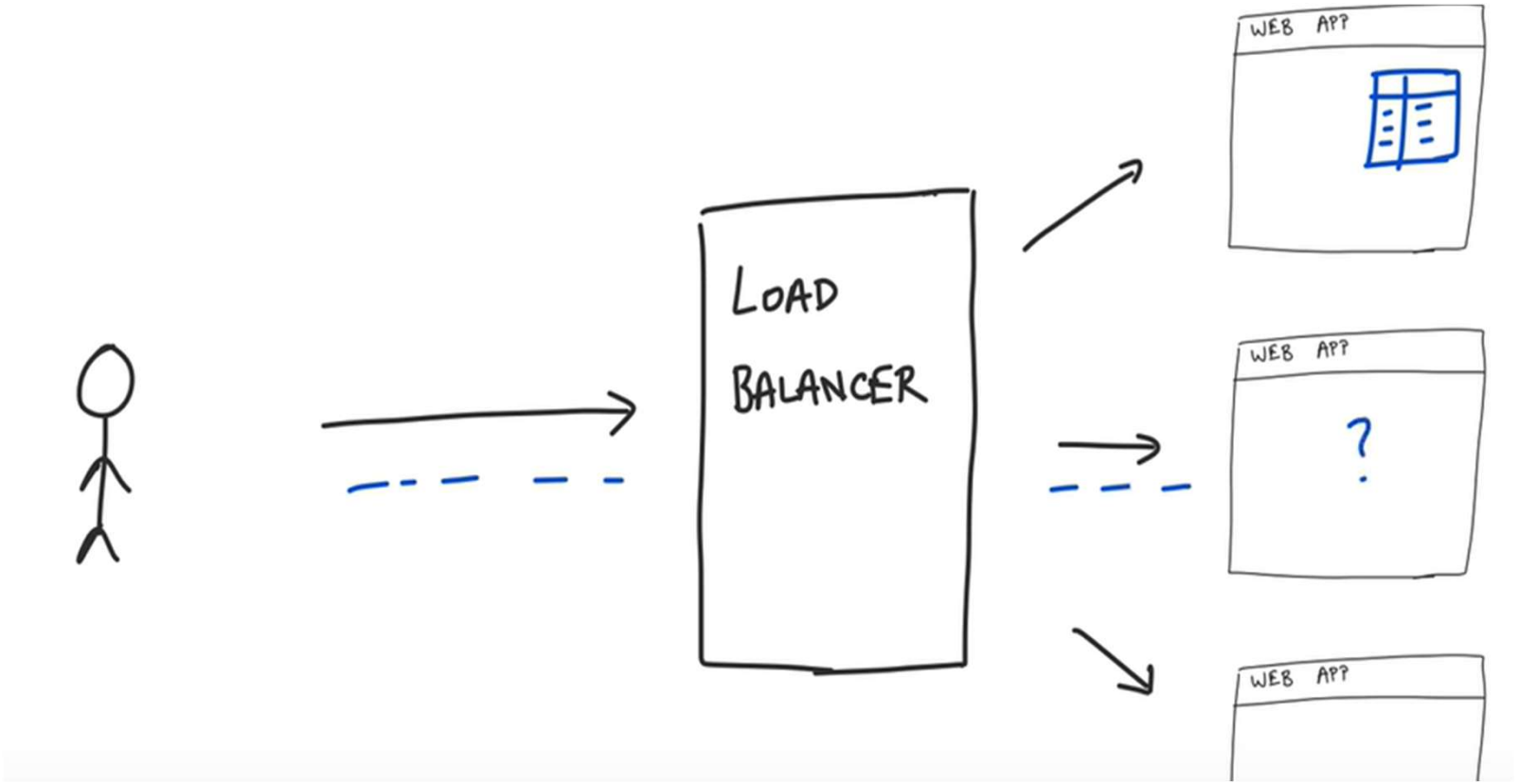
Dodawany automatycznie

Session ID + cookies (ciasteczka)
najpopularniejszy mechanizm do autoryzacji

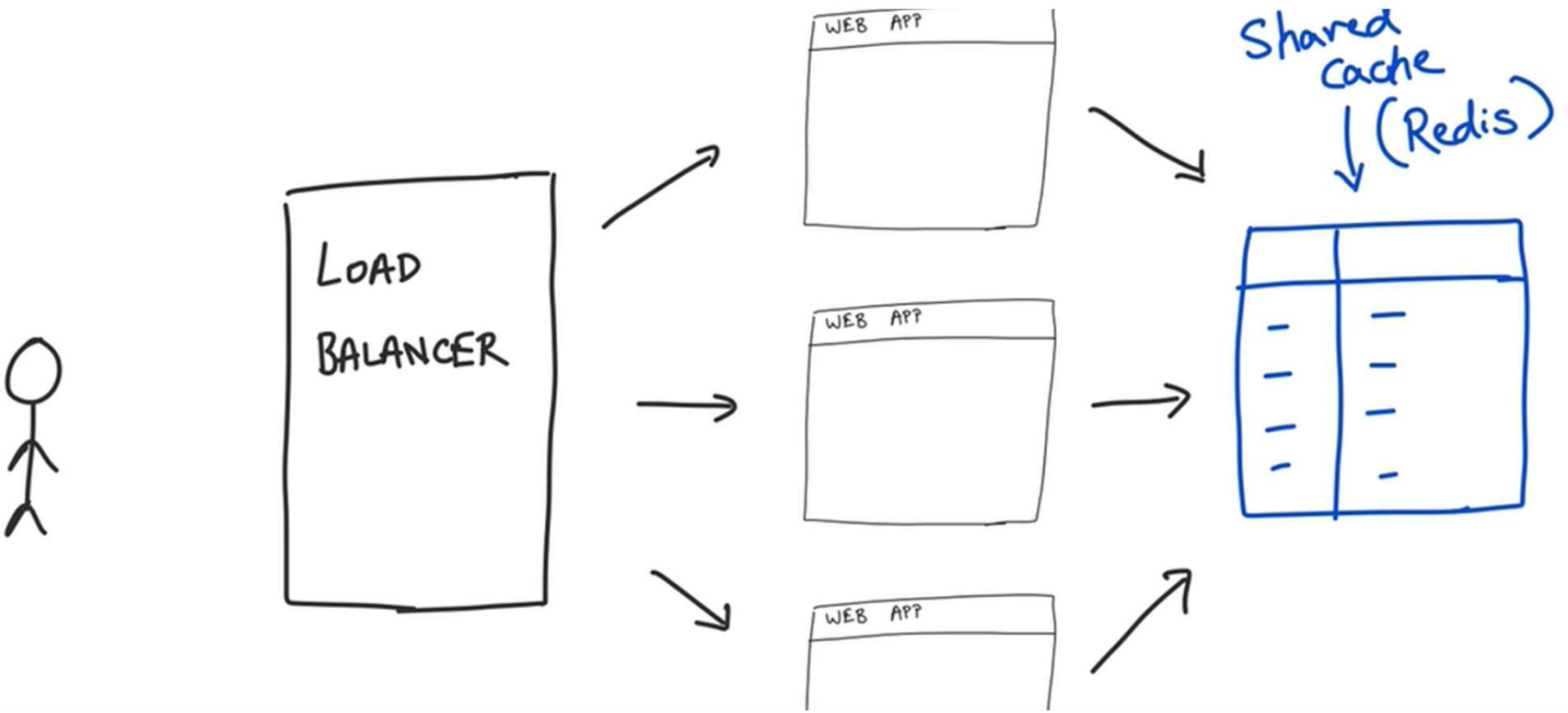
Klasyczna autentykacja oparta na mechanizmie sesji



Problem – architektura skalowalna

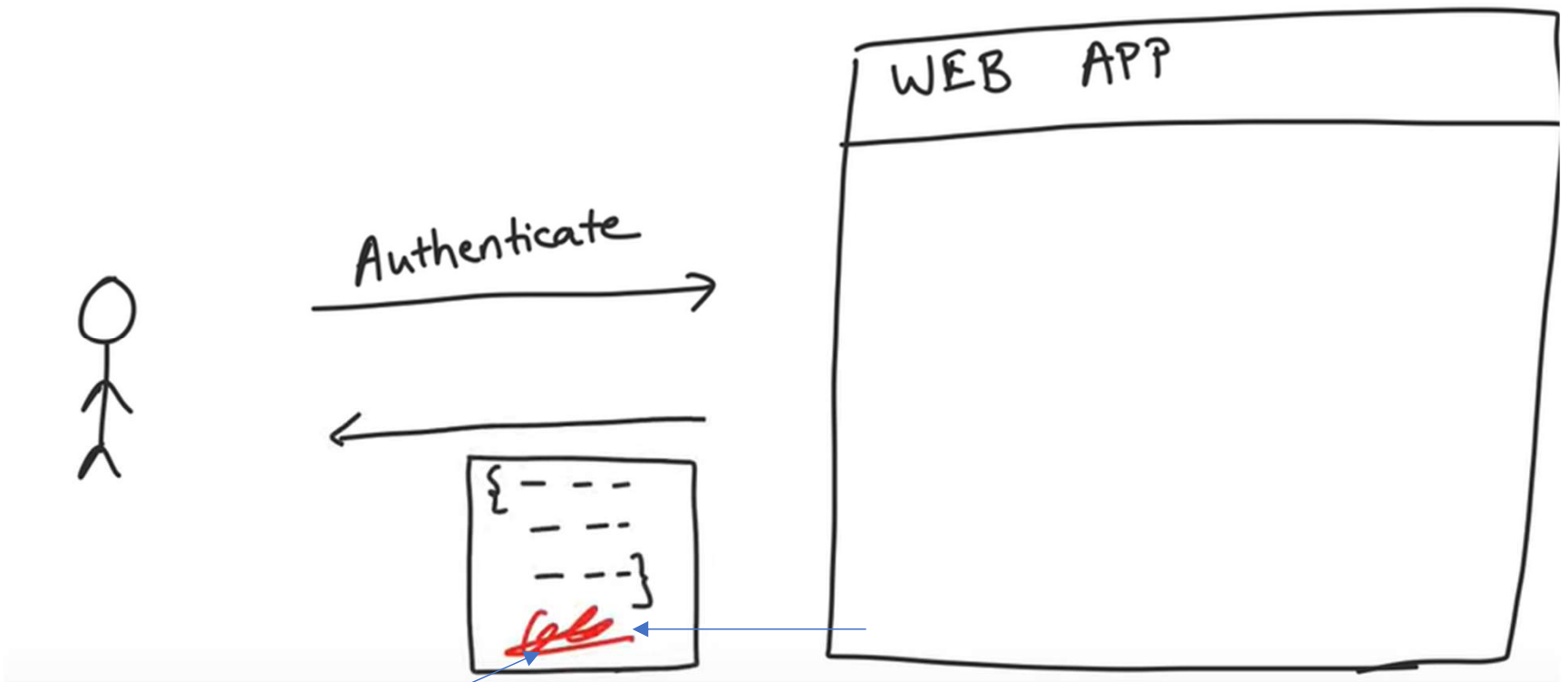


Rozwiązanie – zasób współdzielony



Ale nowe problemy -> pojedynczy punkt awarii -> współdzielona baza!!!

Inny sposob



Zweryfikowałem Cię jako użytkownika A

JWT

Token w formacie JSON - JSON Web Token

To zapytanie ma wszystko czego potrzebuje

JWT

- JSON Web Token czyli tytułowy JWT jest to standard (**RFC 7519**) definiujący przesyłanie informacji między stronami w postaci obiektu JSON.
- JWT gwarantuje, że przesyłane dane są prawdziwe (*nikt ich nie zmodyfikował po drodze*) i są przesłane przez ich właściciela *(pomijam fakt ukradnięcia tokenu i podszywania się pod właściciela).* Gwarancja ta wynika z faktu, że token jest podpisywany sygnaturą cyfrową (*używając* ***sekretów*** *lub pary klucza publicznego/prywatnego*).

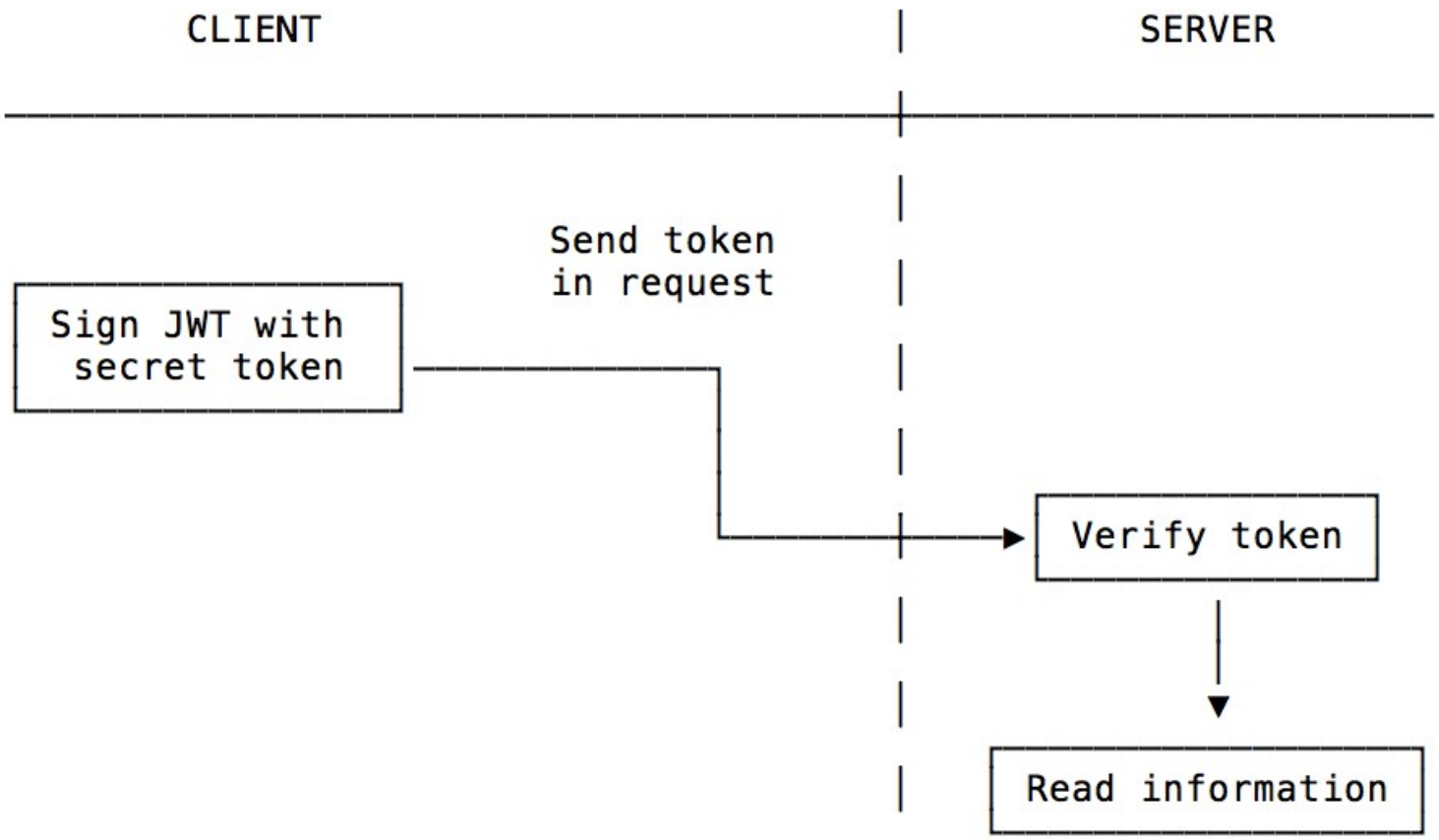
- Warto zapamiętać, JWT gwarantuje prawo własności danych (*data ownership*) ale **nie gwarantuje ich szyfrowania**! Dane przechowywane w tokenie mogą być widoczne dla każdego kto je przechwyci ponieważ **token jest serializowany a nie zaszyfrowany**. Z tego też powodu nie powinno umieszczać się tam danych, które są wrażliwe.

- SON Web Token z pewnością jest przydatny i należy z niego korzystać.

- Istnieją przypadki, gdzie jego użycie jest wręcz wskazane, np. **autoryzacja server-to-server** czy też **uwierzytelnianie API**.

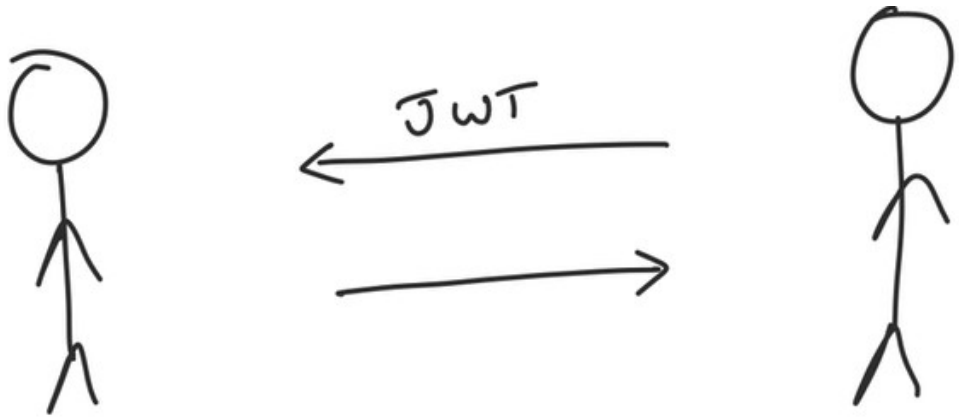
- Token może przechowywać dowolny rodzaj informacji o użytkowniku
- Serwer może ufać klientowi bo JWT jest podpisany a dzięki temu serwer nie musi uderzać do bazy danych po dodatkowe informacje

Uwierzytelnienia API (API Authentication)



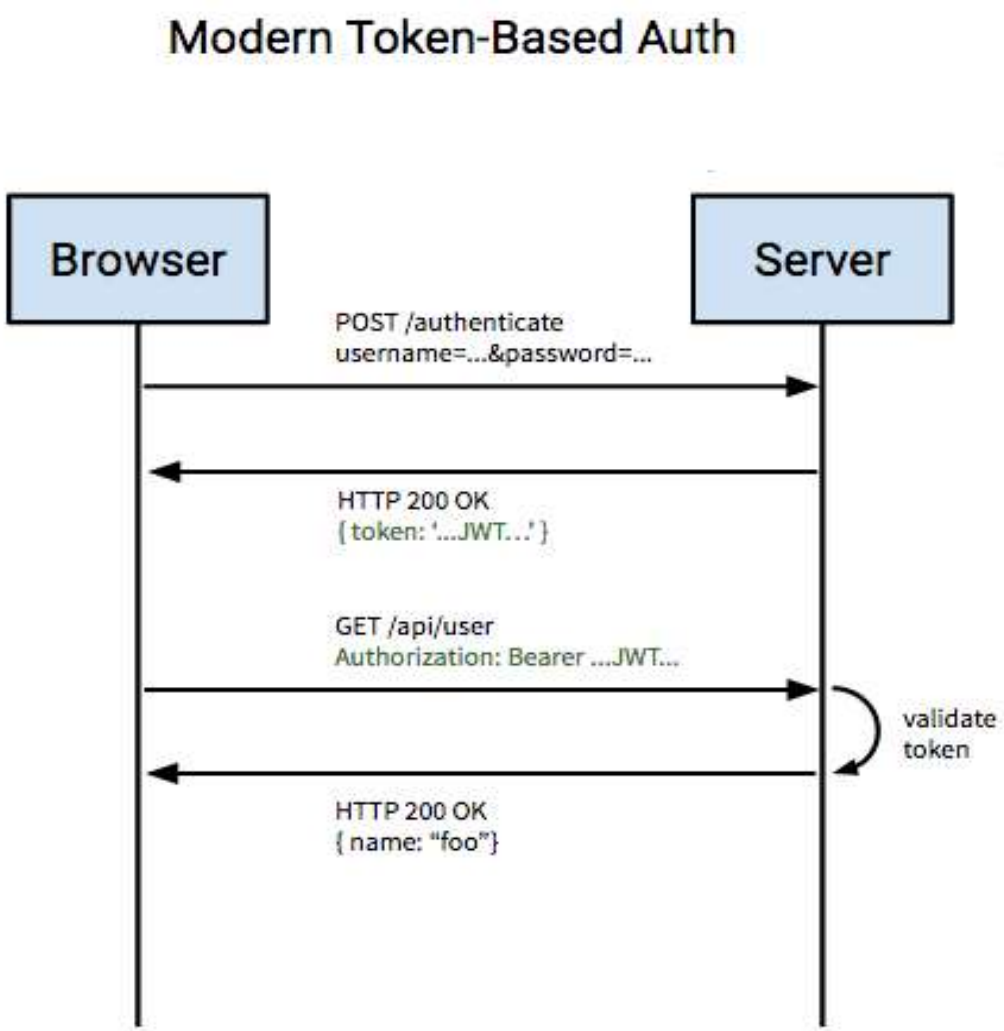
Strategie autoryzacji

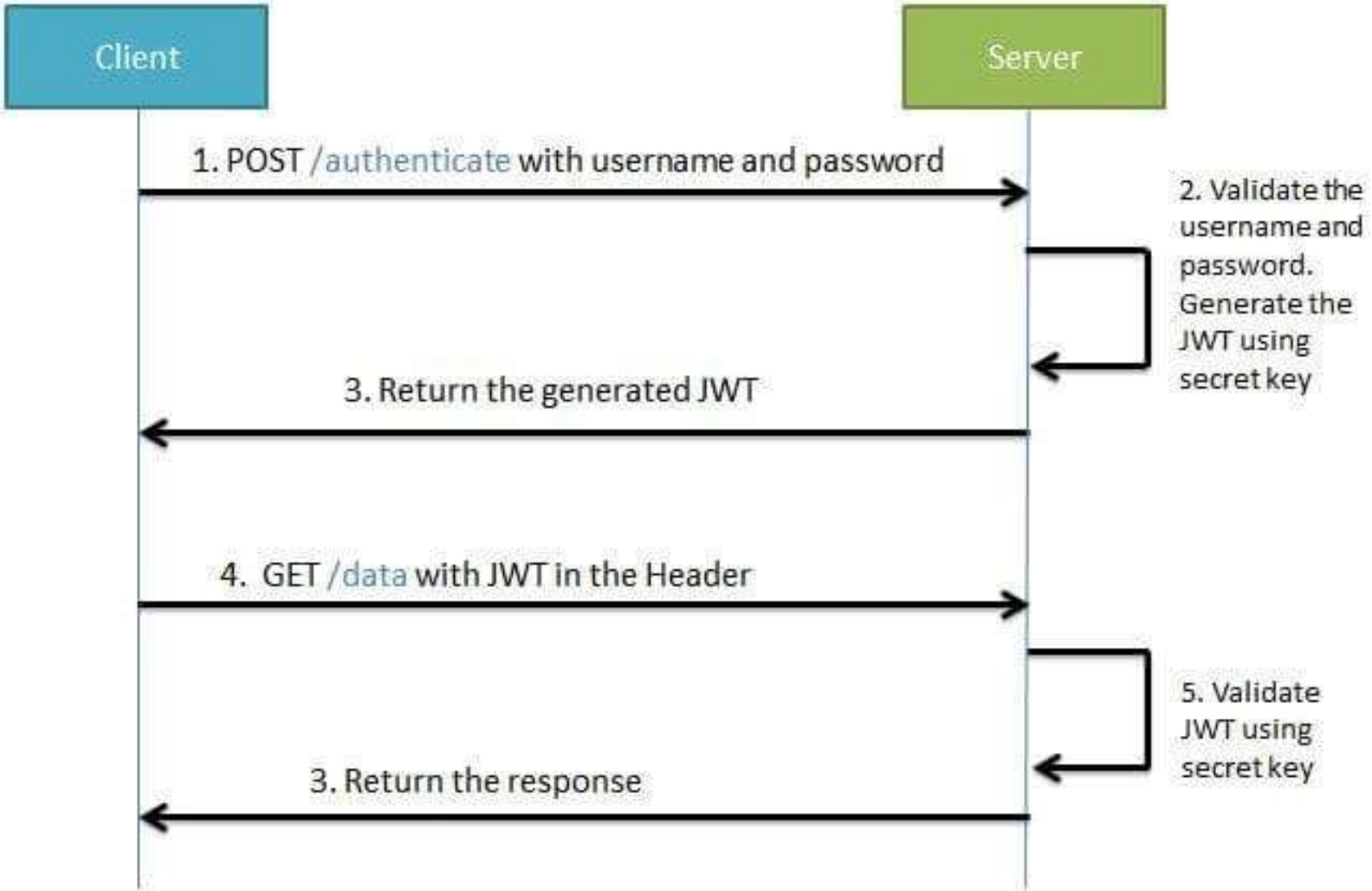
Session token -> token referencyjny opisujący stan aplikacji (odwołujący się do tokenu na serwerze)



JSON web token -> token zawierający wartość

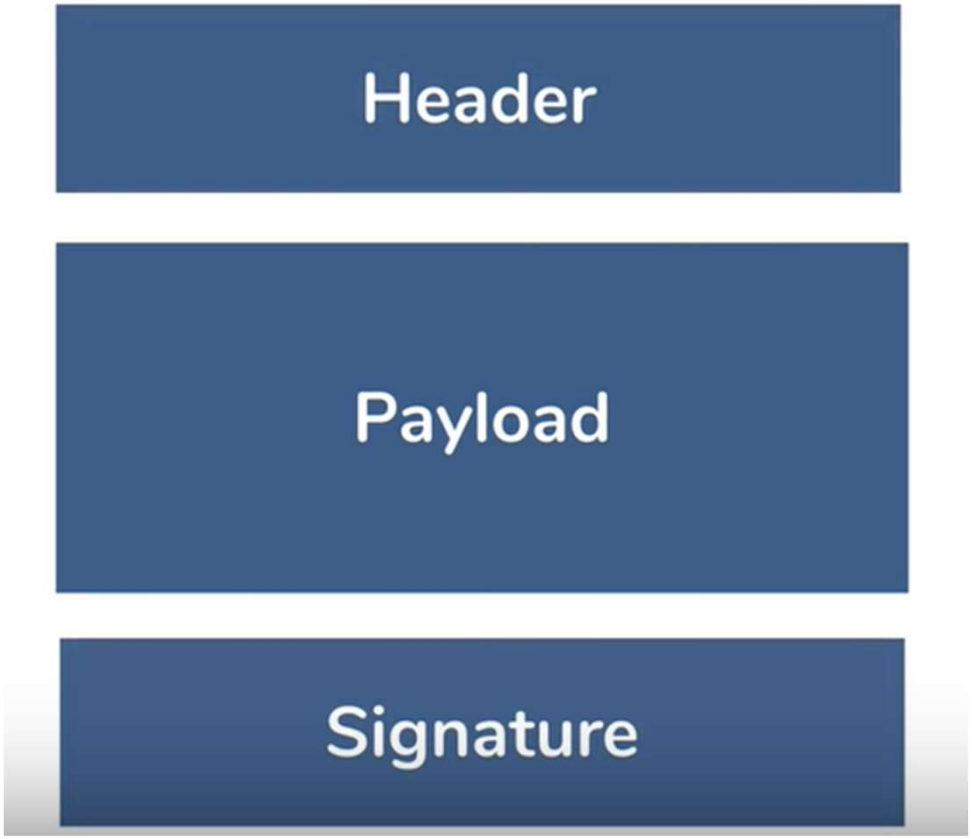
JWT – koncentracja na autoryzacji




Client
Server
1. POST /authenticate with username and password
2. Validate the username and password. Generate the JWT using secret key
3. Return the generated JWT
4. GET /data with JWT in the Header
5. Validate JWT using secret key
3. Return the response

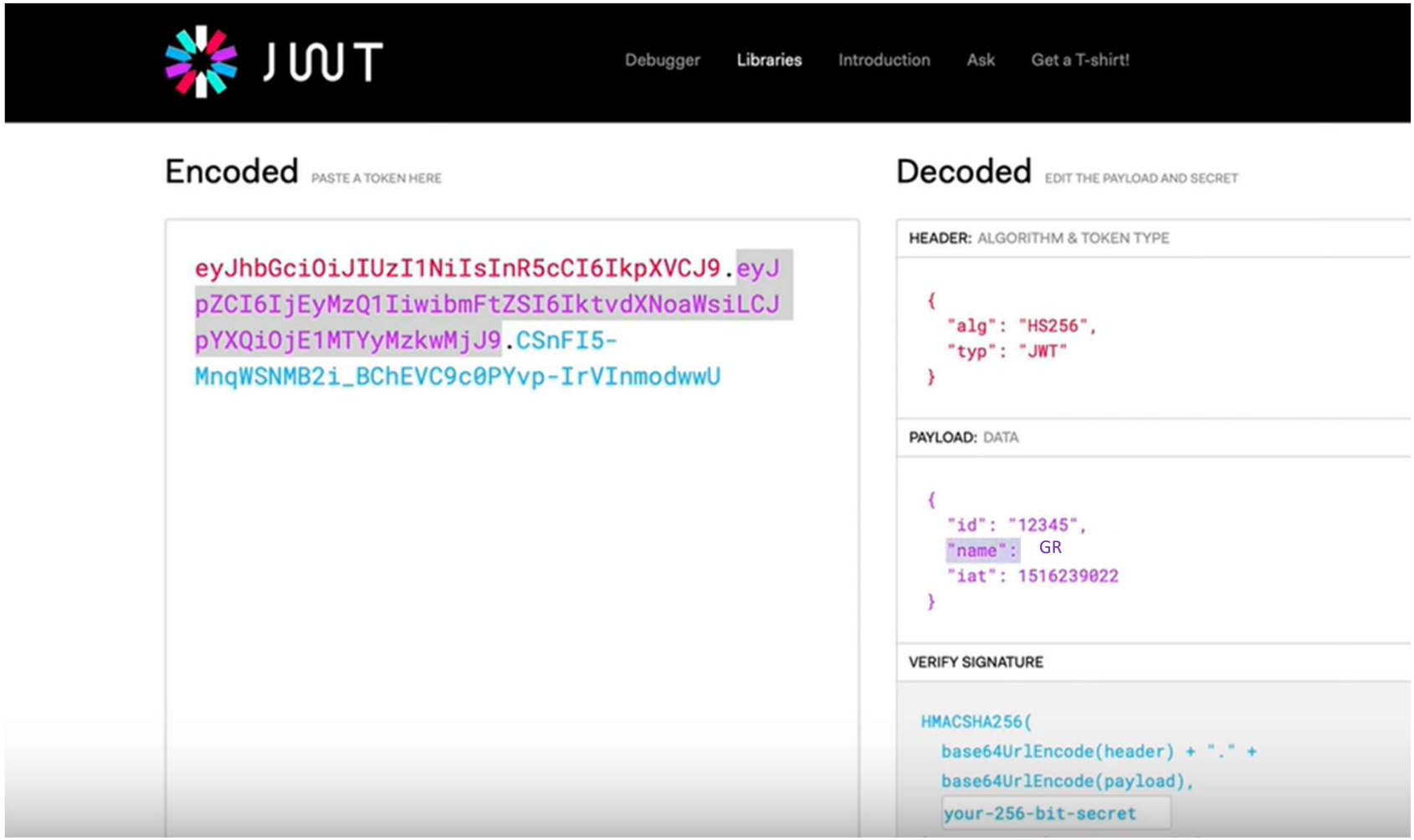
JWT – struktura JSON Web Token

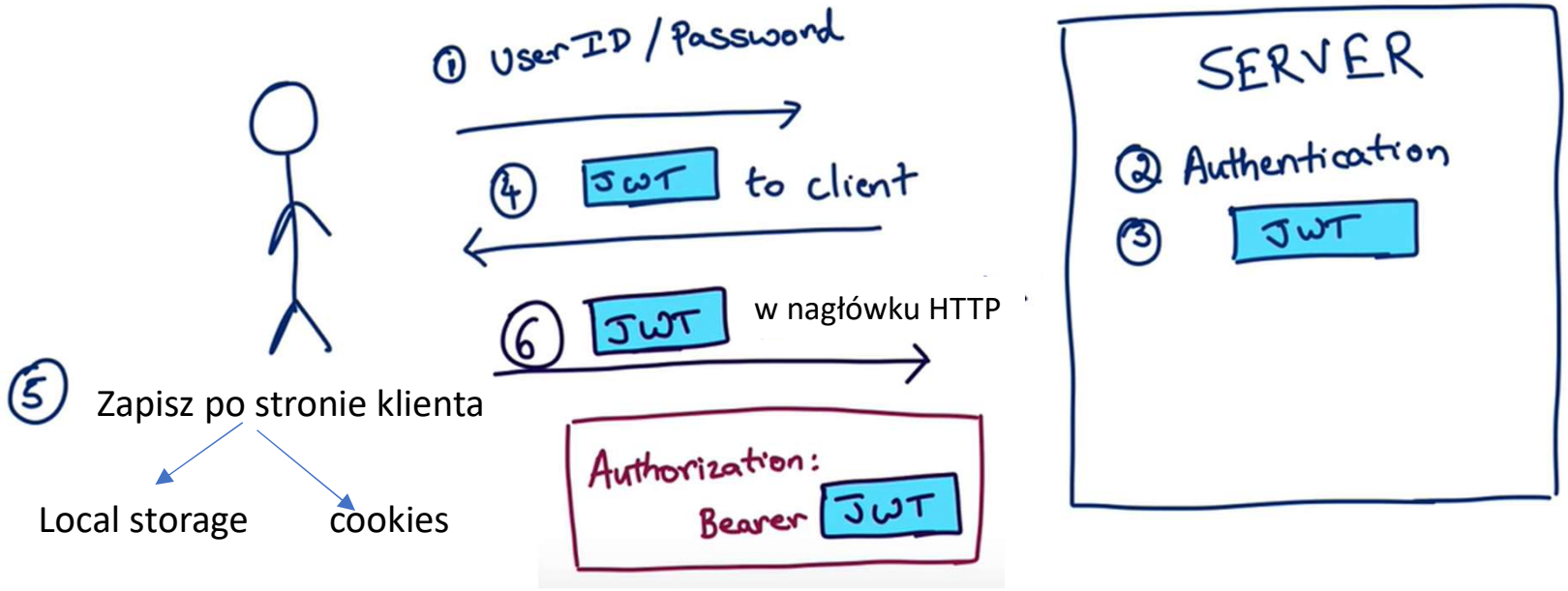
trzy części JWT



JWT przykład

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```


JWT
Debugger
Libraries
Introduction
Ask
Get a T-shirt!
Encoded
PASTE A TOKEN HERE
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6IjEyMzQ1IiwibmFtZSI6IktvdXNoaWsiLCJpYXQiOjE1MTYyMzkwMjJ9.CSnFI5-MnqWSNMB2i_BChEVC9c0PYvp-IrVInmodwwU
Decoded
EDIT THE PAYLOAD AND SECRET
HEADER: ALGORITHM & TOKEN TYPE
{
"alg": "HS256",
"typ": "JWT"
}
PAYLOAD: DATA
{
"id": "12345",
"name": GR
"iat": 1516239022
}
VERIFY SIGNATURE
HMACSHA256(
base64UrlEncode(header) + "." +
base64UrlEncode(payload),
your-256-bit-secret


① UserID / Password
④ JWT to client
⑥ JWT
w nagłówku HTTP
⑤ Zapisz po stronie klienta
Local storage
cookies
Authorization:
Bearer JWT
SERVER
② Authentication
③ JWT