Autentikáció és jogosultságkezelés

Cookie, session, JWT, token alapú auth, OAuth2, SSO alapjai

Bevezetés: Miért fontos az autentikáció?

- Ki vagy te? - **hitelesítés**
- Mit csinálhatsz? – **jogosultságkezelés**
- Webes alkalmazások alapbiztonsága

A hitelesítés és jogosultságkezelés **alapfeltétel** minden komoly rendszerben.

Autentikáció vs. Autorizáció

- Autentikáció (**Authentication**) - hitelesítés
- Autorizáció (**Authorization**) - jogosultságkezelés
- Ez két külön probléma
- Előbb azonosítunk, utána engedélyezünk.

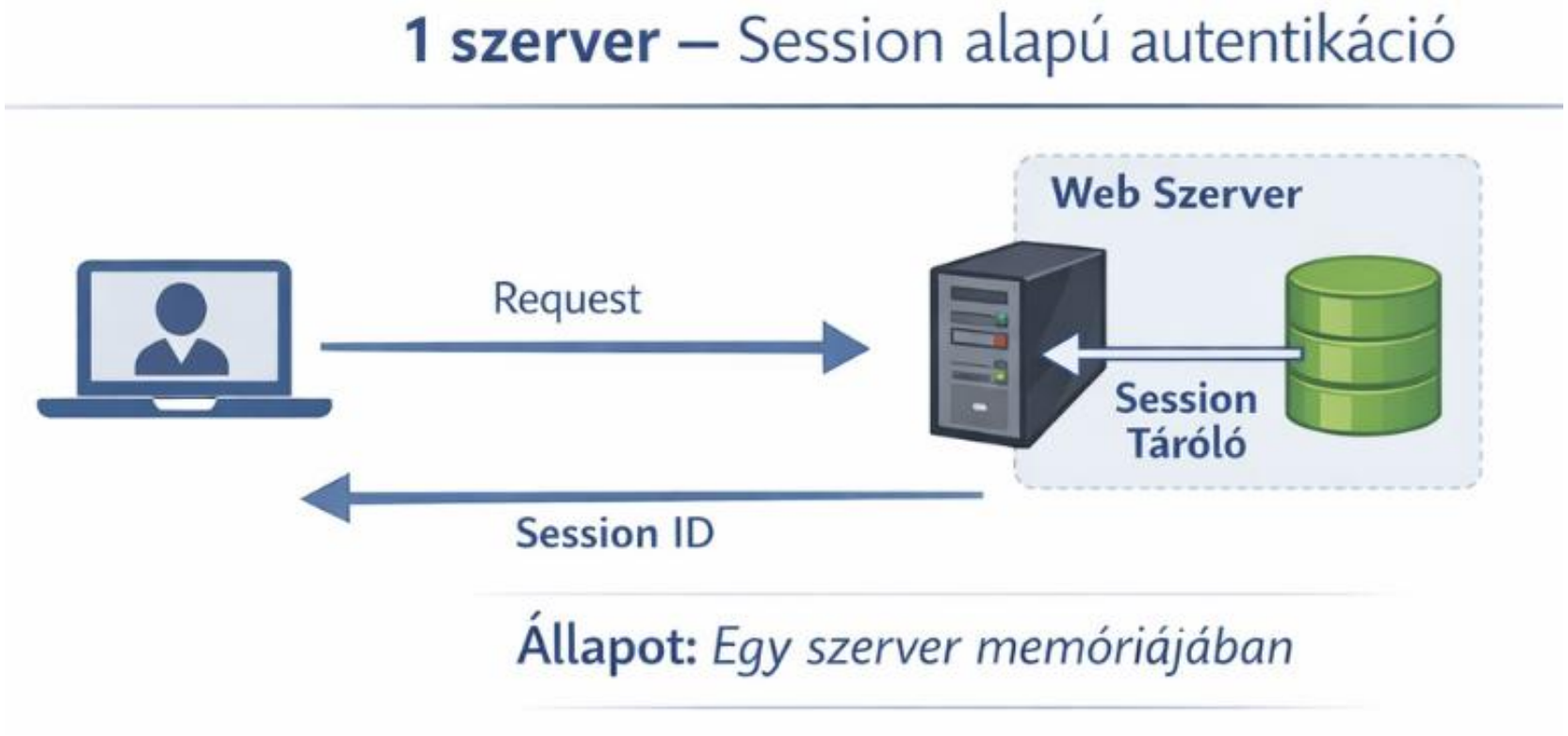
HTTP stateless működés problémája

- A HTTP protokoll **stateless**
- A szerver **nem tárol kliens-állapotot**
- Minden HTTP kérés **független és önálló**
- Bejelentkezési állapot **nem létezik alapból**
- **Állapotkezelés** bevezetése a stateless HTTP protokoll fölé

Session alapú autentikáció

- Szerver oldali **állapotkezelés**
- Session azonosító (**Session ID**)
- Kliens ↔ szerver kapcsolat „emlékezete”
- Klasszikus webalkalmazások megoldása

Session alapú autentikáció



Cookie szerepe az autentikációban

- **Cookie** = kliens oldali tároló
- A session ID tipikus hordozója
- Böngésző automatikusan küldi minden kéréskor
- Cookie ≠ autentikáció

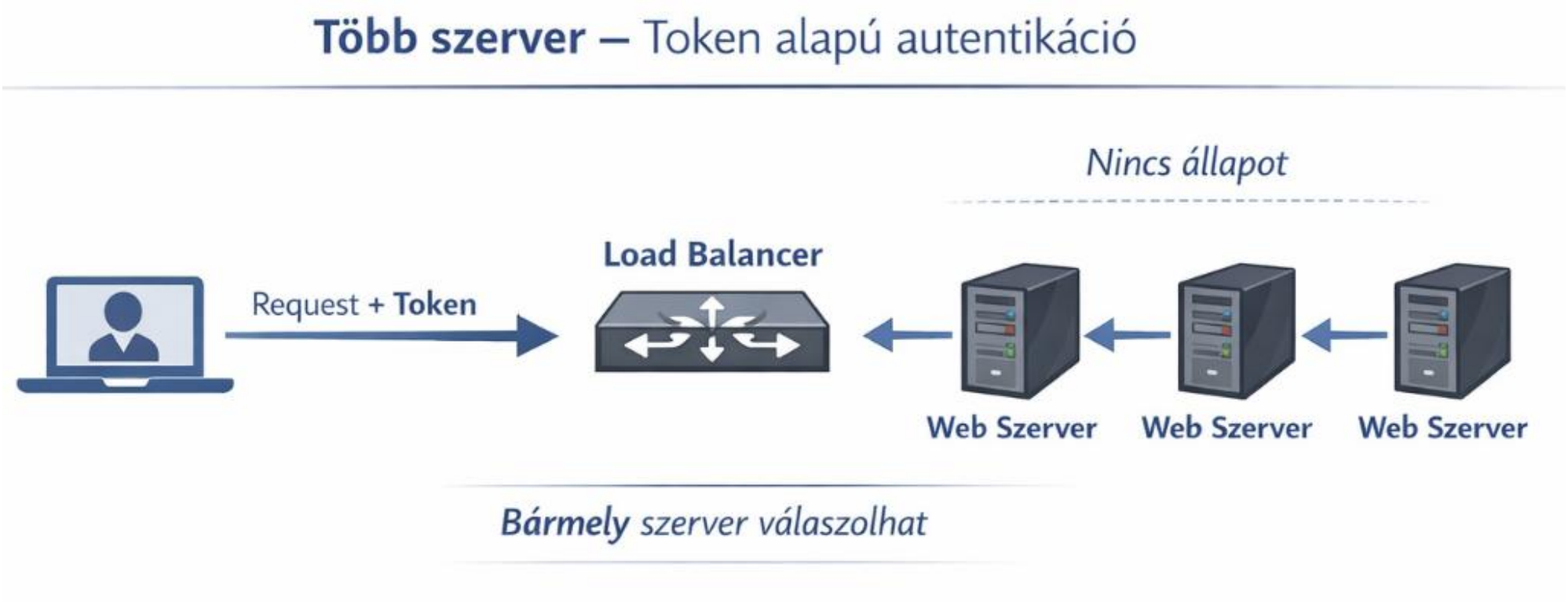
Session alapú autentikáció hátrányai

- Szerver oldali állapot
- **Monolit rendszer** (egy szerver process) esetén megfelelő
- **Skálázási** problémák
- **Load balancer** problémák: Több szerver = több gond
- Modern (microservice) architektúráknál nehézkes

Token alapú autentikáció alapjai

- **Stateless** megközelítés
- A **kliens hordozza** az állapotot
- **Token** minden kérés **fejlécében**
- **Skálázható** architektúrák alapja

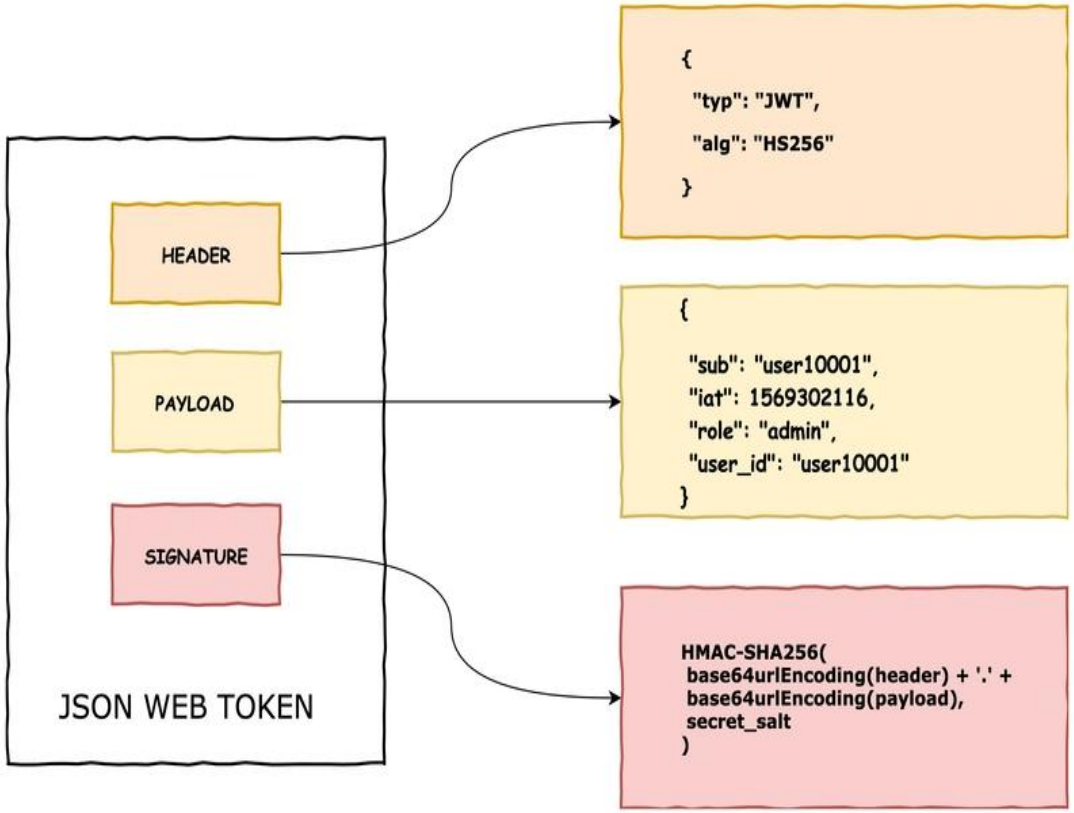
Token alapú autentikáció alapjai



JWT (JSON Web Token) felépítése

JWT három fő részből áll:

- Header
- Payload
- Signature
- Pontokkal elválasztva: xxxxx.yyyyy.zzzzz
- BASE64 encoding



JWT előnyei és kockázatai

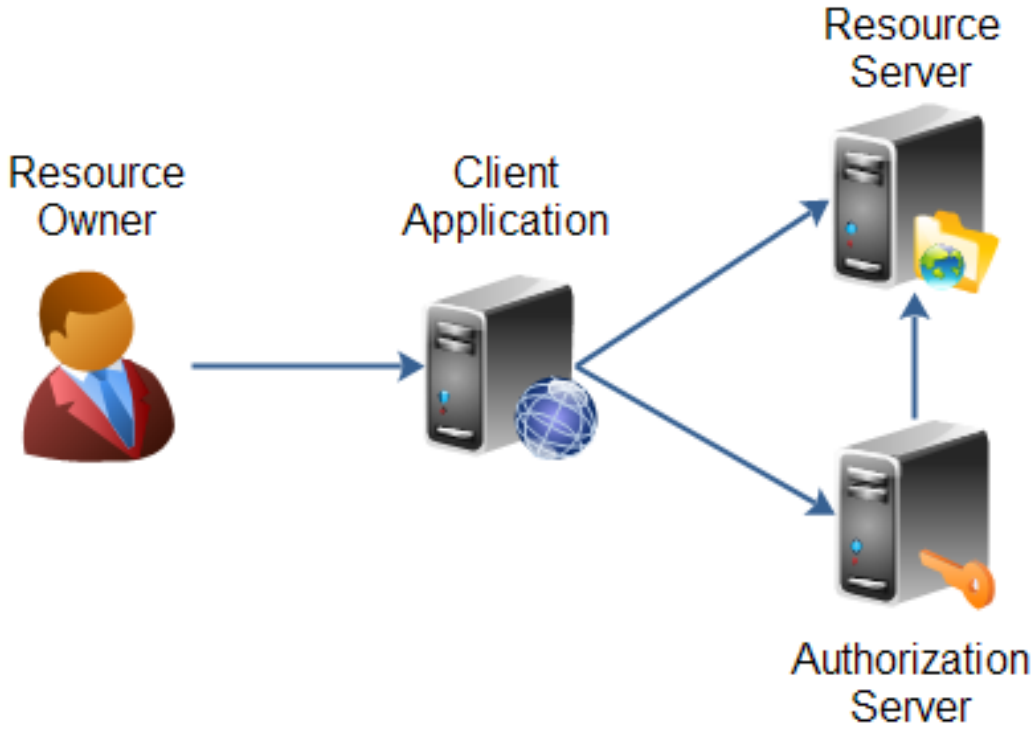
- **Stateless** működés
- **Jó skálázhatóság**
- **Gyors** ellenőrzés
- Biztonsági kockázatok helytelen használat esetén
    - ellopott token = hozzáférés
    - visszavonás nehéz
    - túl hosszú élettartam
    - túl sok adat

Session alapú vs. Token alapú autentikáció

Szempont	Session alapú autentikáció	Token alapú autentikáció
Állapot helye	Szerver oldalon	Kliens oldalon
HTTP modell	Stateful	Stateless
Azonosító	Session ID	Token (pl. JWT)
Szerver memóriaterhelés	Növekszik a felhasználók számával	Minimális
Skálázhatóság	Korlátozott	Kiváló
Több szerver kezelése	Sticky session vagy shared store szükséges	Természetes módon támogatott
Load balancer kompatibilitás	Problémás lehet	Teljes mértékben kompatibilis
Microservice környezet	Nehézkes	Ideális
API / mobil kliensek	Kényelmetlen	Kifejezetten ajánlott
Állapot elvesztése szerver kieséskor	Igen	Nem
Tipikus használat	Klasszikus weboldalak	REST API, SPA, mobil backend

OAuth2 alapelvek

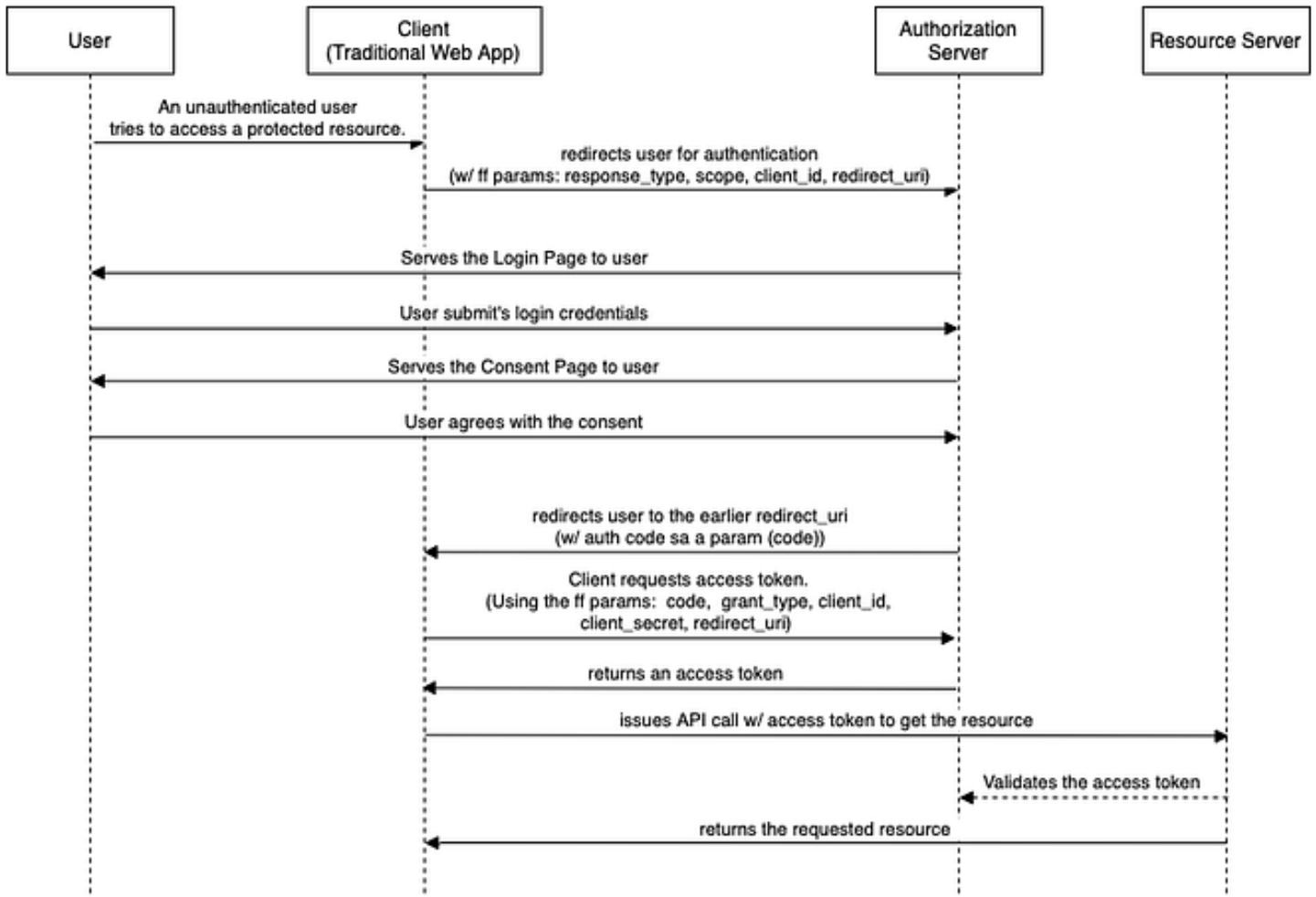
- **Delegált** hozzáférés
- **Token** alapú engedélyezés
- **Jelszó megosztása nélkül**
- Több szereplős modell
  - User (Resource Owner)
  - Client (application)
  - Authorization server
  - Resource server



OAuth2 flow-k röviden

- Authorization Code Flow
- Client Credentials Flow
- Implicit Flow (deprecated)
- Különböző problémák → különböző flow-k

Authorization Code Flow



SSO (Single Sign-On) alapjai

- Egy bejelentkezés
- Több alkalmazás
- Központi identitáskezelés
- OAuth2 / OpenID Connect alapokon

Összefoglalás

- **HTTP stateless** → állapotkezelés szükséges
- **Session** és **cookie**: klasszikus megoldás
- **Token** és **JWT**: modern, stateless megközelítés
- **OAuth2 és SSO**: központi, vállalati szintű megoldások

Döntési táblázat – mikor melyiket használjuk?

Helyzet / Igény	Ajánlott megoldás	Miért?
Egyszerű, klasszikus weboldal	Session + Cookie	Könnyű implementáció, szerver oldali kontroll
Egy szerver vagy kevés felhasználó	Session	Nem szükséges bonyolult architektúra
REST API / SPA frontend	JWT	Stateless, jól skálázható
Mobil alkalmazás backendje	JWT	Cookie-független, API-barát
Microservice architektúra	JWT	Nincs központi session state
Külső alkalmazás hozzáfér az adatokhoz	OAuth2	Delegált hozzáférés, nincs jelszómegosztás
Vállalati több rendszer/alkalmazás	SSO (OAuth2 + OIDC)	Egy bejelentkezés, központi identitás
Felhasználói kényelem + biztonság	SSO	Kevesebb jelszó, kisebb támadási felület