Appunti: HTTP - Parte 2

Recap: HTTP Request/Response

- **Struttura:**
  - **Request:** Request Line, Headers, Body
  - **Response:** Status Line, Headers, Body
- **Intermediari:** Client, Proxy, Gateway, Origin Server

HTTP/2

- **Obiettivo:** Migliorare le performance di HTTP/1.1 (RFC 7540, 2015).
- **Basato su:** SPDY (protocollo di Google).
- **Compatibilità:** Mantiene metodi, status code, headers di HTTP/1.1.
- **Novità:**
  - **Multiplexing:** Richieste e risposte multiple sulla stessa connessione, anche in ordine diverso.
  - **Push:** Il server può inviare più dati di quelli richiesti, anticipando richieste future.
  - **Binary Framing:** Protocollo binario, messaggi compressi con headers e payload separati. I flussi di dati sono suddivisi in frames.
  - **Compressione Header (HPACK - RFC 7541):** Riduce la dimensione degli header inviando solo le differenze rispetto alle richieste precedenti (riduzione stimata del 30%).

HTTP/3

- **Obiettivo:** Ulteriore miglioramento delle performance rispetto a HTTP/2 (RFC 9114, 2023).
- **Basato su:** QUIC (protocollo di Google) su UDP (non TCP/IP). QUIC è a livello di trasporto, HTTP a livello applicazione.
- **Compatibilità:** Mantiene metodi, status code, headers di HTTP/1.1 e HTTP/2.

Cookies

- **Concetto:** HTTP è stateless. I cookies sono informazioni scambiate tra server e client per mantenere uno stato tra le connessioni. Netscape li ha proposti, poi formalizzati in RFC 2109 e RFC 2965
- **Funzionamento:**
  1. Il server invia un cookie (dati arbitrari) al client con la prima risposta (header `Set-Cookie`).
  2. Il client memorizza il cookie e lo invia al server ad ogni richiesta successiva (header `Cookie`).
  3. Il server utilizza il cookie per identificare il client e ripristinare informazioni sulla sessione.
- **Analogia:** Tagliando di una lavanderia (blocco di dati opaco).
- **Headers:**
  - `Set-Cookie`: Inviato dal server nella risposta.
  - `Cookie`: Inviato dal client nelle richieste successive.
- **Struttura del Cookie (testuale):**
  - Nome e valore.
  - `Domain`: Dominio di validità.
  - `Path`: URI di validità.
  - `Max-Age` / `Expires`: Durata del cookie.
  - `Secure`: Richiede connessione sicura (HTTPS) per l'invio.
  - `Version`: Versione della specifica.
- **Tipi di Cookie:**

    - **Persistenti:** Lunga durata, per informazioni semi-permanenti (login, preferenze).
    - **Di sessione:** Breve durata, per raggruppare operazioni in sessioni di lavoro (cancellati alla chiusura del browser). Contengono un ID di sessione, le cui info sono sul server.
    - **Di terze parti:** Appartenenti a un dominio diverso, usati per pubblicità e tracciamento (possono essere disabilitati).

Autenticazione

- **Autenticazione:** Verifica dell'identità dell'utente.
- **Autorizzazione:** Verifica della possibilità dell'utente di accedere a una risorsa/eseguire un'operazione.
- **Obiettivo:** Evitare di ripetere l'autenticazione ad ogni richiesta (fino alla scadenza).
- **Approcci:**
    - **Session-based:** Il server memorizza un ID di sessione e informazioni sull'utente.
    - **Token-based:** Il client memorizza un token (ricevuto dal server) e lo invia ad ogni richiesta.
- **Implementazione:** Tramite headers HTTP e cookies.

Header WWW-Authenticate

- **Funzione:** Il server risponde con 401 (Unauthorized) e specifica i criteri di autenticazione.
- **Schemi:**
    - **Basic:** Informazioni di autorizzazione in chiaro (Base64) nell'header `Authorization` (deprecato).
    - **Digest:** Fingerprint della password (MD5) e stringa casuale (nonce) nell'header `Authorization` (per evitare replay attack).
    - **Bearer:** Token ("bearer token") nell'header `Authorization` (usato per autenticazione token-based).

Session-based Authentication

1. **Autenticazione:** Il server verifica l'identità dell'utente.
2. **Creazione Sessione:**
    - Il server genera un ID di sessione e associa le informazioni dell'utente.
    - Le informazioni sono memorizzate sul server.
    - L'ID di sessione viene inviato al client.
3. **Richieste Successive:**
    - Il client invia l'ID di sessione.
    - Il server verifica l'ID e la validità temporale.
4. **Implementazione:** Spesso tramite cookie di sessione (contengono l'ID univoco).
5. **Gestione Server-side:** Linguaggi server-side (PHP, Express.js) gestiscono automaticamente i cookie di sessione.

Token-based Authentication

1. **Autenticazione:** Il server verifica l'identità dell'utente.
2. **Creazione Token:**
    - Il server crea un token con un segreto e lo firma.
    - Il token viene inviato al client.
3. **Richieste Successive:**
    - Il client memorizza il token e lo include negli header (`Authorization` , schema `Bearer`).
    - Il server verifica la firma e le informazioni nel token.
4. **Vantaggio:** Più scalabile (non richiede memorizzazione sul server).

5. **Tipologia token:** Varia, il server può usare meccanismi diversi per firmarlo.

JWT (JSON Web Token) - RFC 7519

- **Standard:** Formato JSON per lo scambio di token di autenticazione e informazioni (claims).
- **Caratteristiche:**
  - Personalizzazione dei claims.
  - Diversi algoritmi di firma.
  - Basato su JSON Web Signature (JWS) e JSON Web Encryption (JWE).
  - Sintassi compatta e URL-safe.

Base64 (breve approfondimento)

- Codifica un flusso di dati in un sottoinsieme di 64 caratteri US-ASCII.
- Suddivide i dati in blocchi di 24 bit, poi in 4 blocchi di 6 bit, e li codifica secondo una tabella.
- Decodifica algoritmica, senza chiavi (NON crittografica).

Struttura JWT

1. **Header:** Tipo di token e algoritmo di cifratura (Base64).
2. **Payload:** Informazioni (claims) (Base64).
   - **Registered:** Predefiniti (iss, iat, exp, nbf).
   - **Public:** Dichiarati (IANA JSON Web Token Registry).
   - **Private:** Personalizzabili.
3. **Signature:** Firma del token (header e payload in Base64) con chiave segreta (simmetrica o asimmetrica). Protegge da manomissione ma non cifra il contenuto!

JWT: Debugger e Librerie

- **Debugger online:** https://jwt.io/#debugger-io
- **Librerie:** Disponibili per vari linguaggi (es. express-jwt per Express.js).

Express.js e Autenticazione

- **Middleware:** `passport.js` (flessibile, supporta varie strategie) o `express-jwt` (solo JWT).

CORS (Cross-Origin Resource Sharing)

- **Cross-site Vulnerability:** Codice malevolo in una pagina di un dominio protetto che trasmette informazioni a un altro dominio.
- **Politica dei Browser:** Rifiutare connessioni Javascript a domini diversi dalla pagina ospitante (stesso schema, dominio e porta).
- **Soluzione (CORS):** Basata sul metodo `OPTIONS` per indicare i domini ammessi.

CORS (W3C Rec 29/1/2013)

- **Tecnica:** Solo per connessioni Ajax.
- **Headers:**
  - **Richiesta:** `Origin` (dominio del contenuto).
  - **Risposta:** `Access-Control-Allow-Origin` (domini consentiti).
- **Preflight:** Verifica preliminare con `OPTIONS`.
- **Sessione CORS:**
  1. **Richiesta:** `GET` o `OPTIONS`, `Origin: www.dominio1.com`
  2. **Risposta:** `Status code: 200`, `Access-Control-Allow-Origin: http://www.dominio1.com` (o `*`)

Express.js e CORS

- **Middleware:** `cors`.

- **Funzioni:** Aggiunge header `Origin` e `Access-Control-Allow-Origin`.
- **Opzioni:** Abilitare tutti i domini, domini specifici, altre opzioni.

HTTP Caching

- **Tipi:** Client-side, server-side, proxy.
- **Obiettivi:**
  - Server-side: Riduce i tempi di computazione.
  - Altre: Riducono il carico di rete.
- **Meccanismi (HTTP/1.1):**
  - **Server-specified expiration:**
    - `Expires` header.
    - `max-age` directive in `Cache-Control`.
  - **Heuristic expiration:** La cache stima la durata in base ad altri header (es. `Last-Modified`).

Cache-Control

- **Direttive:**
  - `must-revalidate`: La risposta scaduta non può essere usata (riprendere dall'origin server, 504 se non disponibile).
  - `no-cache`: Richiesta sempre all'origin server.

Heuristic Expiration

- **Algoritmo:** Non fissato, dipende dall'implementazione.
- **Risposta:** 113 (heuristic expiration) se non valida con sicurezza.

Validazione delle Risorse in Cache

- **Obiettivo:** Verificare se una risorsa è ancora valida dopo la scadenza.
- **Metodi:**
  - `HEAD`: Richiede la data di ultima modifica (richiesta in più).
  - **Richiesta condizionale:**
    - Se modificata: Risposta normale.
    - Se non modificata: 304 (Not Modified) senza body.
- **Headers:** `If-Match`, `If-Modified-Since`, `If-Unmodified-Since`.