Sicurezza per gli Utenti

Presentazione Ing SW. 2024 - Thomas Arghittu

Concetti chiave

Autenticazione VS Autorizzazione

Processo di verifica dell'identità dell'utente.

CHI E' l'utente?

Processo di determinazione sulle possibilità dell'utente.

CHE COSA può fare l'utente?

Autenticazione a Multi Fattore (MFA)

Qualcosa che sai: Una password o PIN.

Qualcosa che possiedi: Un token fisico, un dispositivo mobile o un app di autenticazione.

Qualcosa che sei: Un'impronta digitale, il riconoscimento facciale o altre biometrie.

2FA (Two-Factor Authentication): Implementazione più comune, solitamente *password* e *codice temporaneo.*

Autenticazione Stateless - JSON Web Token (JWT)

Formato di token utilizzato per rappresentare in modo sicuro informazioni tra parti diverse.

Header: Contiene il tipo di token (*JWT*) e l'algoritmo di firma (ad esempio, *HS256* per *HMAC SHA-256*).

Payload: Contiene le informazioni (*claims*) codificate, come l'*ID utente*, il *ruolo* e altre informazioni personalizzate.

Signature: Firmata con una chiave segreta o una chiave privata per garantire che il token non sia stato alterato.

Autenticazione Stateful - Session

Metodo tradizionale di gestire l'autenticazione lato **server**.

Login utente → **Creazione** sessione → **Memorizzazione** ID sessione → **Invio** all'utente

Stateful: Il server deve mantenere lo stato della sessione per ogni utente.

Scadenza: Le sessioni hanno una durata limitata e possono essere invalidate.

Sicurezza: È essenziale proteggere gli ID delle sessioni per evitare attacchi di session hijacking.

Autenticazione Stateful - Cookies

Piccoli **file** di testo che vengono memorizzati nel **browser** dell'utente e inviati al server con ogni richiesta *HTTP* (per gestione di sessioni, autenticazione e tracciamento utenti).

Session Cookies: Utilizzati per mantenere la sessione di un utente attiva. Scadono alla chiusura del browser o dopo un periodo di inattività.

Persistent Cookies: Hanno una data di scadenza specifica e rimangono nel browser fino a tale scadenza. Utilizzati per ricordare l'accesso tra sessioni.

Single Sign-On (SSO)

Permette agli utenti di autenticarsi **una sola volta** e accedere a **più applicazioni** senza bisogno di ri-autenticarsi.

SAML

Protocollo XML per l'autenticazione e autorizzazione.

OAuth con OpenID Connect

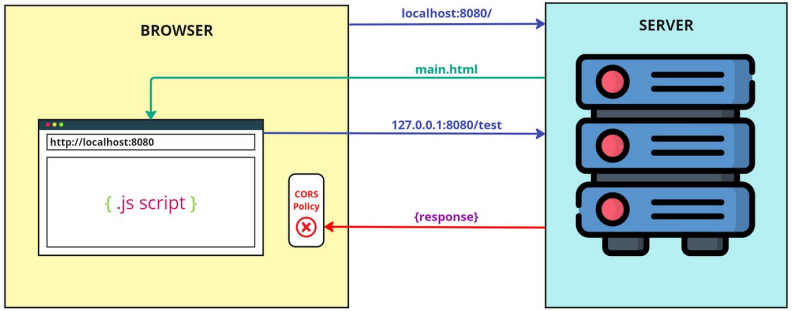
Protocollo di autorizzazione

Consente a terze parti di accedere a risorse protette senza condividere le credenziali dell'utente.

CORS (Cross-Origin Resource Sharing)

Meccanismo di sicurezza che **limita** come le risorse su un server possono essere richieste da **domini diversi**.

Proteggere le risorse del server da **richieste non autorizzate** provenienti da **fonti esterne**.



Soluzioni reali

Authelia

Piattaforma open-source per la gestione dell'autenticazione e autorizzazione.

Progettata per applicazioni web **self-hosted**, offre sicurezza e controllo senza dipendere da servizi esterni.

Si integra facilmente con un **reverse proxy** (come *Nginx* o *Traefik*) per proteggere le applicazioni.

Autenticazione a due fattori (2FA)

Single Sign-On (SSO)

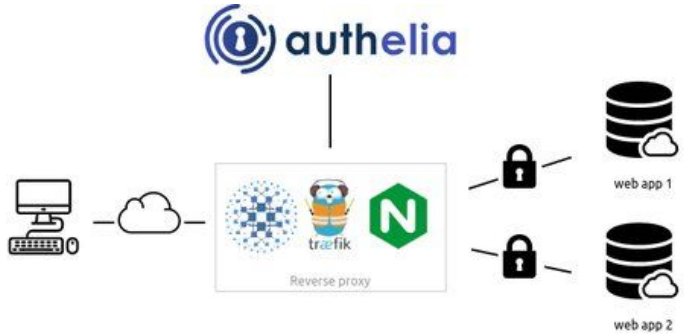
Gestione degli Accessi

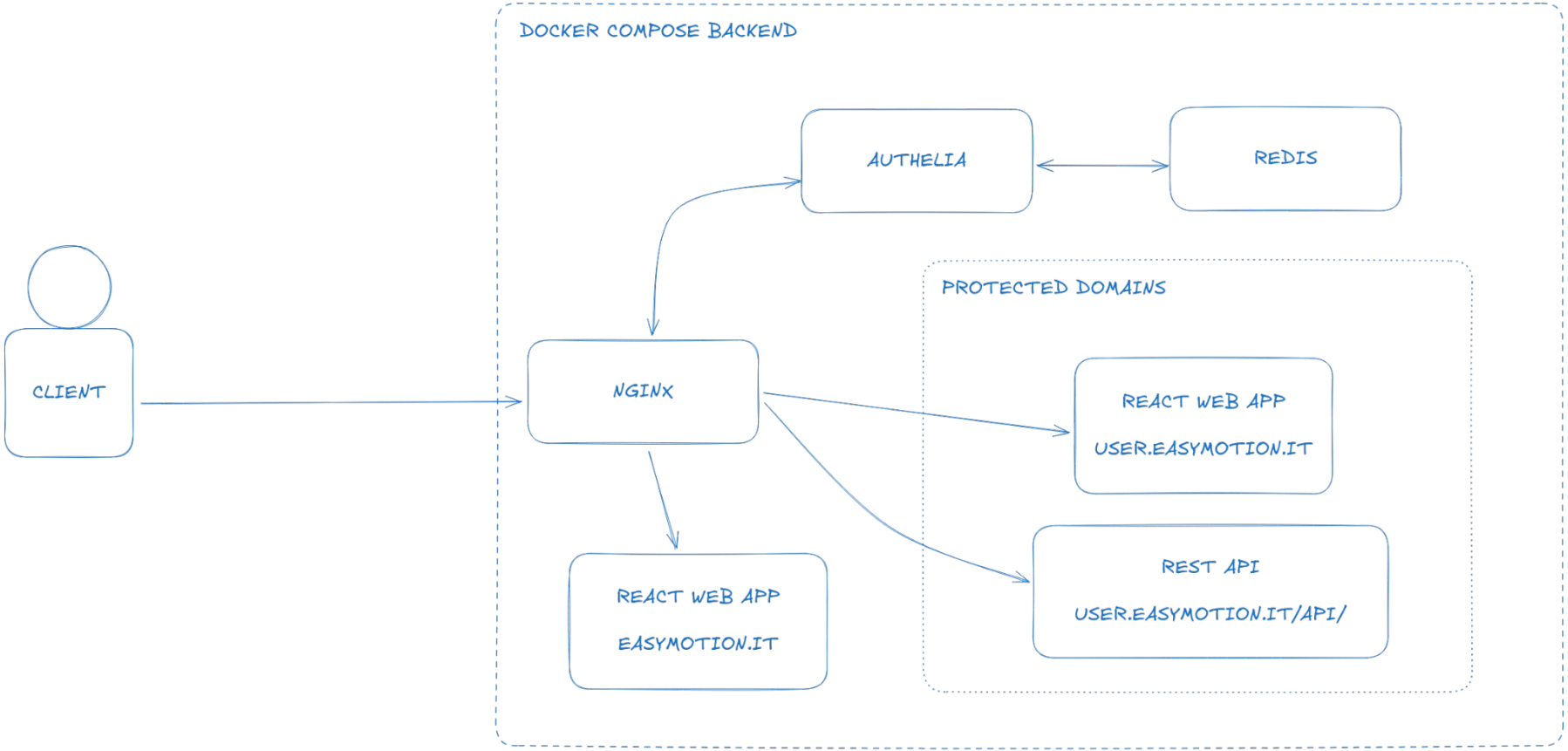
Authelia - Architettura

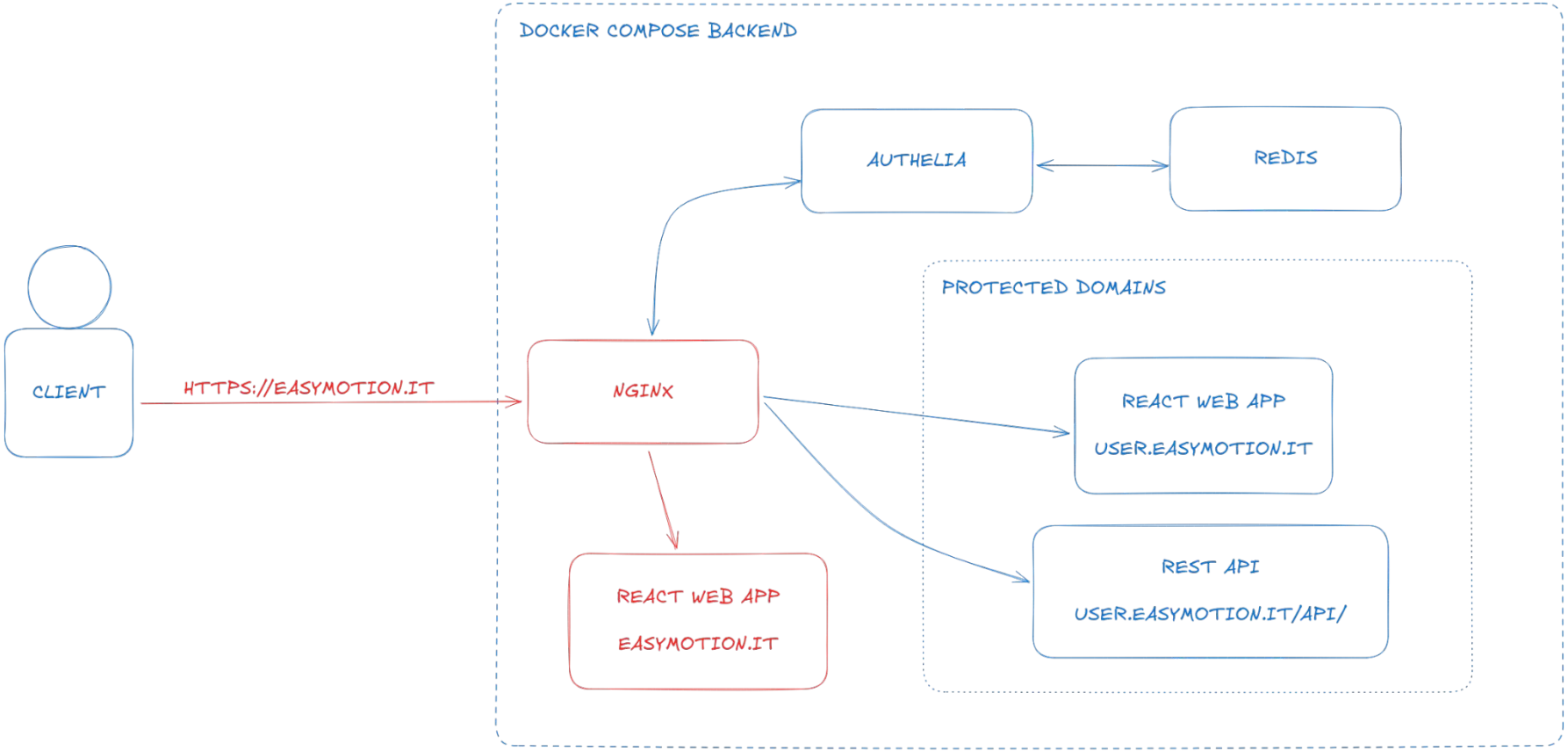
Authelia Server: Gestisce l'autenticazione, l'autorizzazione e le regole di accesso.

Reverse Proxy: *Nginx* o *Traefik* si interfacciano con Authelia per autenticare le richieste prima di raggiungere l'applicazione.

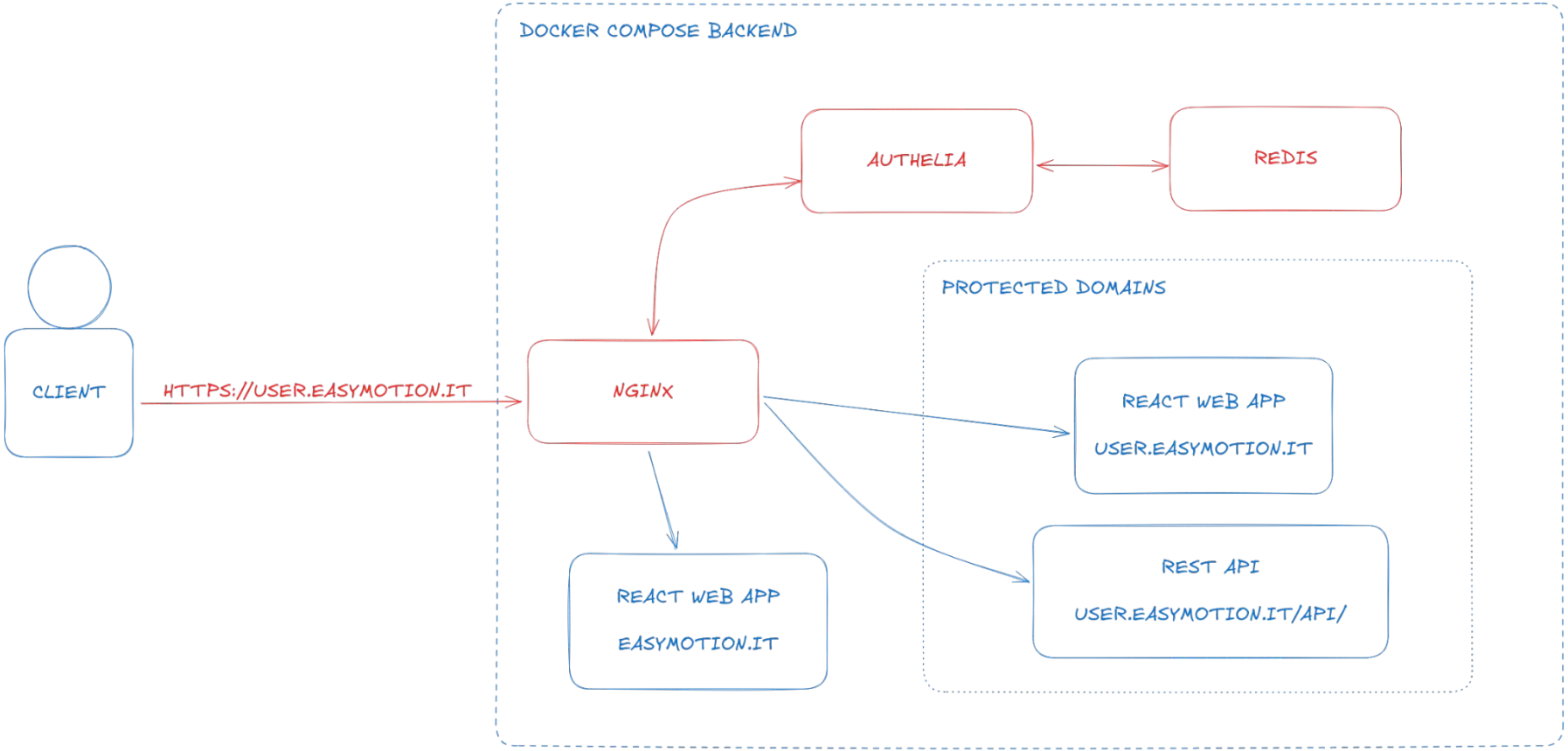
Database: Utilizzato per memorizzare utenti, configurazioni e sessioni (compatibile con *PostgreSQL*).



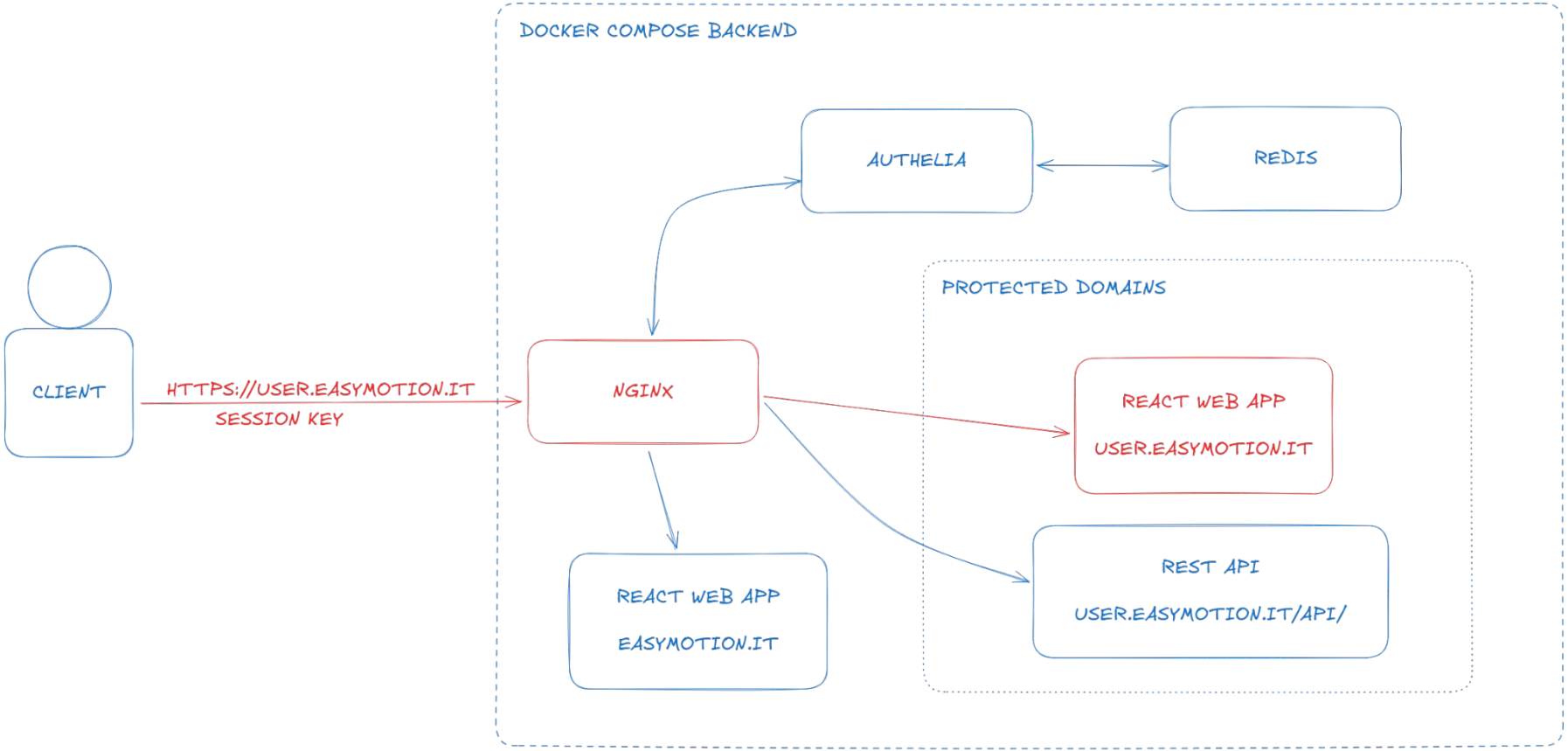

DOCKER COMPOSE BACKEND
AUTHELIA
REDIS
PROTECTED DOMAINS
CLIENT
NGINX
REACT WEB APP
USER.EASYMOTION.IT
REST API
USER.EASYMOTION.IT/API/
REACT WEB APP
EASYMOTION.IT



Browser ⟶ HTTPS ⟶ Nginx ⟶ Regola Nginx ⟶ (***Dominio non protetto***) ⟶ React App



Browser ⟶ HTTPS ⟶ Nginx ⟶ Regola Nginx ⟶ (***Dominio protetto***) ⟶ Authelia (Autenticazione)



Authelia (Login) ⟶ Sessione Creata in Redis ⟶ Nginx ⟶ React App o API Backend

Authelia - JWT

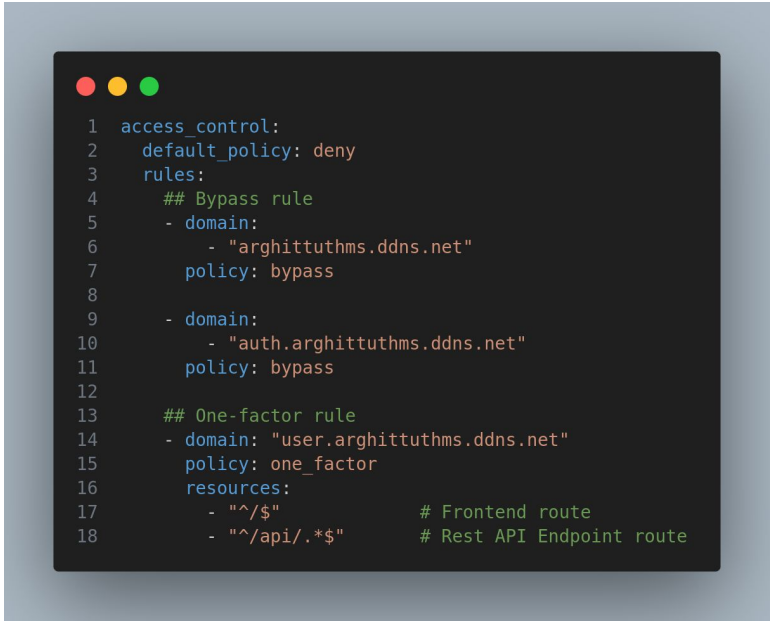
Authelia genera un JWT che contiene alcune informazioni essenziali sull'utente, come:

- **Identificatore** dell'utente (username o email).
- **Gruppi di appartenenza**, se configurati.
- **Timestamp** di scadenza (expiry timestamp).
- **Issuer** (chi ha generato il token).

Questi dati sono **firmati** (non criptati) per **garantire che non siano stati alterati**, ma possono essere letti in chiaro se si ha accesso al JWT.

Il JWT viene utilizzato principalmente come "token di sessione".

Authelia - Configuration.yml



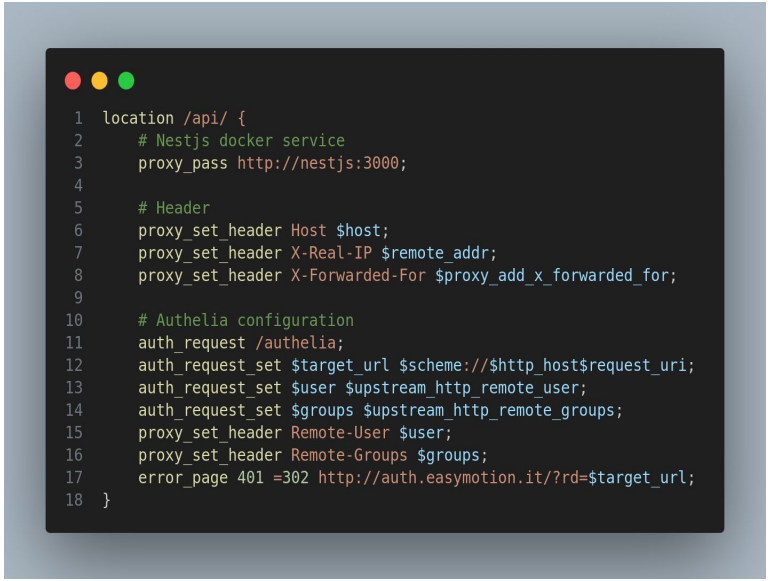
```yaml
access_control:
  default_policy: deny
  rules:
    ## Bypass rule
    - domain:
        - "arghittuthms.ddns.net"
      policy: bypass

    - domain:
        - "auth.arghittuthms.ddns.net"
      policy: bypass

    ## One-factor rule
    - domain: "user.arghittuthms.ddns.net"
      policy: one_factor
      resources:
        - "^/$"            # Frontend route
        - "^/api/.*$"      # Rest API Endpoint route
```

- Configurazione generale
  - *host, port, log, jwt secret, default redirect url*
- Autenticazione e **controllo accessi**
  - *totp, authentication backend*
- Gestione della sessione
  - *session, regulation*
- Storage e database
- Notifiche

NGINX - Configuration snippets



```
location /api/ {
    # Nestjs docker service
    proxy_pass http://nestjs:3000;

    # Header
    proxy_set_header Host $host;
    proxy_set_header X-Real-IP $remote_addr;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;

    # Authelia configuration
    auth_request /authelia;
    auth_request_set $target_url $scheme://$http_host$request_uri;
    auth_request_set $user $upstream_http_remote_user;
    auth_request_set $groups $upstream_http_remote_groups;
    proxy_set_header Remote-User $user;
    proxy_set_header Remote-Groups $groups;
    error_page 401 =302 http://auth.easymotion.it/?rd=$target_url;
}
```

Esempio: *https://user.easymotion.it/*

Regole di redirect:

- *location /*
- *location /api/*
- *location /authelia/*

Auth0

Piattaforma **SaaS** (*Software as a Service*) per la gestione dell'identità e dell'autenticazione.

Soluzione **Identity-as-a-Service** completa, semplifica l'implementazione di login sicuro per le applicazioni web e mobile.

Fornisce funzionalità di autenticazione, autorizzazione, Single Sign-On (SSO) e gestione degli utenti in un'unica piattaforma.

Autenticazione a due fattori (2FA)

Single Sign-On (SSO)

Login Social e Personalizzato

Gestione degli Utenti

Auth0 - Architettura

Auth0 Dashboard: Interfaccia web per la gestione degli utenti, applicazioni, ruoli e regole di accesso.

Auth0 APIs: Set completo di API per l'integrazione con le applicazioni, gestione degli utenti e la configurazione delle politiche di sicurezza.

Database Utenti: Utenti in database interni o esterni

Authelia VS Auth0 (PRO)

Scalabilità	Facilità di Implementazione
Personalizzazione	Scalabilità e Affidabilità
Sicurezza Avanzata	Personalizzazione Avanzata
Integrazione con Database Esistenti	Sicurezza Elevata

Authelia VS Auth0 (CONTRO)

Maggior curva di apprendimento iniziale

Gestione completa della sicurezza in “casa”

Infrastruttura backend più complessa

Vendor lock-in

Costi addizionali basati su numero utenti e numero richieste

Maggiori dipendenze a livello di backend / frontend