Studente: **Francesco Mancuso** Matricola: **942409**

Relazione di Sicurezza delle architetture orientate ai servizi

Sommario

[Obiettivo 3](#_Toc46844317)

[Authorization server 3](#_Toc46844318)

[Applicazione client 3](#_Toc46844319)

[Token di accesso 4](#_Toc46844320)

[Grant type 5](#_Toc46844321)

[Authorization Code 5](#_Toc46844322)

[Resource Owner Password Credentials 5](#_Toc46844323)

[Progetto 5](#_Toc46844324)

[Manuale d’uso 6](#_Toc46844325)

[Authorization Server 6](#_Toc46844326)

[Client 10](#_Toc46844327)

# Obiettivo

L’obiettivo di questo progetto è la realizzazione di un implementazione di un OAuth server semplificato. Essenzialmente consente l'emissione di un token di accesso da parte di un server autorizzativo ad un client di terze parti, previa approvazione dell'utente proprietario della risorsa cui si intende accedere. L’utente può accedere all’applicazione web loggandosi all’authorization server e richiedere la visione dei propri dati personali presentando un token di accesso valido precedentemente ottenuto. Per far ciò ho utilizzato una libreria open source scritta in php e consultabile al seguente link: “<https://github.com/bshaffer/oauth2-server-php>”. Questa libreria fornisce un insieme di classi php per l’implementazione di un server oauth. Nello specifico permette la creazione e gestione dei token, fornisce tutte le informazioni del client (es: client ID, secret client, ecc…), permette di abilitare i diversi tipi di grant type e fornisce la struttura del database sql. La documentazione dettagliata è visibile a questo indirizzo: <https://bshaffer.github.io/oauth2-server-php-docs/>.

# Authorization server

Nello specifico, l’authorization server da me creato permette di effettuare i seguenti passaggi:

1. Permette ad uno sviluppatore di iscriversi e di ricevere tutte le informazioni utili per realizzare un applicazione client che possa interagire con il server OAuth. Alcune di queste informazioni sono il Client ID, Client Secret e la chiave pubblica dell’authorization server.
2. Permette ad un utente generico di iscriversi e di inserire dati personali. Questi dati personali saranno accessibili dall’applicazione client dopo che l’utente avrà fornito la sua autorizzazione e avrà fornito il token di accesso.

# Applicazione client

Nello specifico, l’applicazione client da me creata permette di effettuare i seguenti passaggi:

1. Permette ad un utente di accedere attraverso l’authorization server, il quale fornirà al client un token di accesso. Ho inserito due modalità principali di accesso: Authorization Code Grant e Resource Owner Password Credentials Grant.
2. Se un utente decide di accedere all’applicazione client attraverso l’authorization server usando le sue credenziali, riceverà direttamente un token di accesso (se le credenziali sono corrette).
3. Se un utente decide di accedere all’applicazione client attraverso l’authorization server usando un’authorization code, verrà indirizzato alla pagina del server OAuth dove potrà scegliere se autorizzare o meno il client ad accedere ai suoi dati. Se l’utente autorizza, il client riceverà l’authorization code e lo riproporrà subito dopo (prima che scada) per ottenere un token di accesso. Questa parte del processo avviene in maniera nascosta all’utente.
4. Dopo che l’utente si sarà autenticato, verrà indirizzato alla sua pagina di profilo. In questa pagina sono forniti alcuni dati informativi utili al fine dell’esame: viene mostrato il token ricevuto dal server OAuth e mostrate le parti che lo compongono. Verrà spiegato in seguito nel dettaglio il token.
5. A questo punto l’utente può richiedere la risorsa all’authorization server. Questa risorsa è molto semplice, e rappresenta i dati personali dell’utente presenti nel database dell’authorization server inseriti al momento dell’iscrizione. Per ottenere queste informazioni il client invierà il token di accesso al server, il quale lo verificherà (controllando la scadenza e la firma digitale) e , se accettato, spedirà indietro i dati personali dell’utente.

# Token di accesso

Il token di accesso inviato dall’authorization server è nello specifico un JWT token che utilizza JSON Web Signatures e la crittografia a chiave pubblica per verificarne la validità. Il server OAuth firma il token con la sua chiave privata e il client può verificarlo con la chiave pubblica fornitagli dal server in fase di registrazione. Non solo il client può verificare la validità, ma anche lo stesso server può farlo quando riceve il token prima di inviare una certa risorsa.

Il formato del token è il seguente: HEADER.PAYLOAD.SIGNATURE

1. HEADER: contiene informazioni di configurazione come l’algoritmo usato per la firma (RSA a 2048 bit di base).
2. PAYLOAD: contiene i dati veri e propri del token, come un identificatore, l’id del client e dell’utente, il timestamp di scadenza e il timestamp di creazione.
3. SIGNATURE: contiene la firma del token con chiave privata del server attraverso un algoritmo a chiave pubblica (esempio RSA).

Nello specifico nel progetto ho implementato sia la verifica della firma del token da parte del server, sia la verifica da parte del client.

# Grant type

Come modalità di accesso all’authorization server ho implementato l’Authorization Code e il Resource Owner Password Credentials.

## Authorization Code

In questa modalità l’utente deve autorizzare esplicitamente l’applicazione client ad accedere ai sui dati personali. Nello specifico l’utente richiede l’accesso all’authorization server. Viene reindirizzato alla pagina sul server dove deve fornire l’autorizzazione. Una volta fornita, il server spedirà al client un codice che il client dovrà velocemente (prima della scadenza) rinviare al server per ottenere il token di accesso.

E’ importante ricordare che per usare questa modalità, l’utente dovrà essersi precedentemente loggato sull’authorization server.

## Resource Owner Password Credentials

In questa modalità l’utente inserisce direttamente le sue credenziali di accesso al server sull’applicazione client. E’ una modalità utilizzabile soltanto nei casi in cui il client sia molto sicuro.

# Progetto

Sia l’authorization server, sia l’applicazione client sono stati caricati su due indirizzi di altervista.

L’authorization server:

<https://myoauthdemo.altervista.org>

L’application client:

<https://myapplicationdemo.altervista.org>

L’intero codice sorgente è visionabile a questo indirizzo: <https://github.com/FraMan97/SoaSecurityProject>

## Manuale d’uso

### Authorization Server



Figura 1 - Questa immagine rappresenta l'homepage

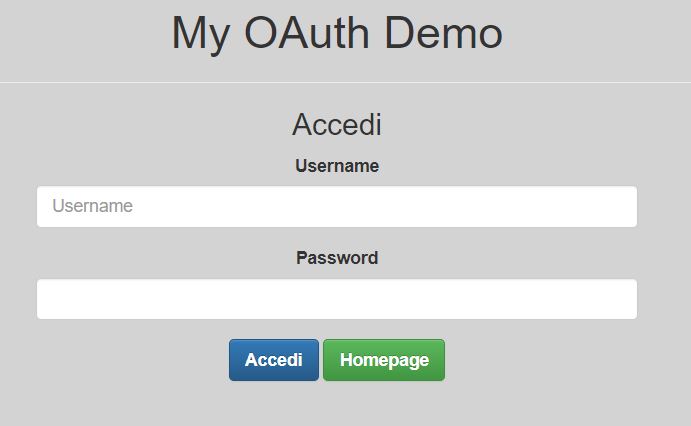


Figura 2 - Questa immagine rappresenta la pagina di login dell'utente all'authorization server



Figura 3 - Questa immagine rappresenta la pagina di registrazione dell'utente con alcuni dati riservati di esempio



Figura 4 - Questa immagine rappresenta la pagina di registrazione del client

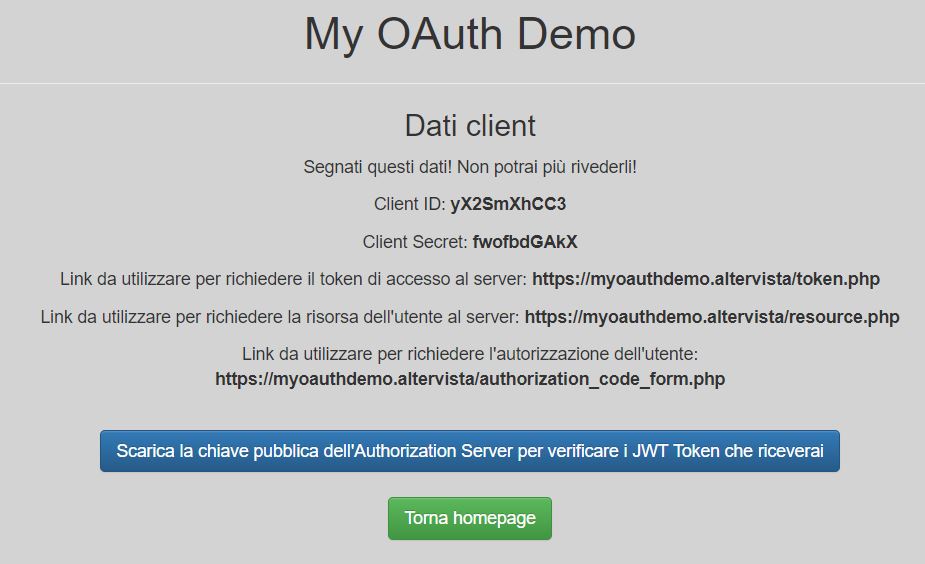


Figura 5 - Questa pagina rappresenta la pagina di riepilogazione dei dati del client

### Client

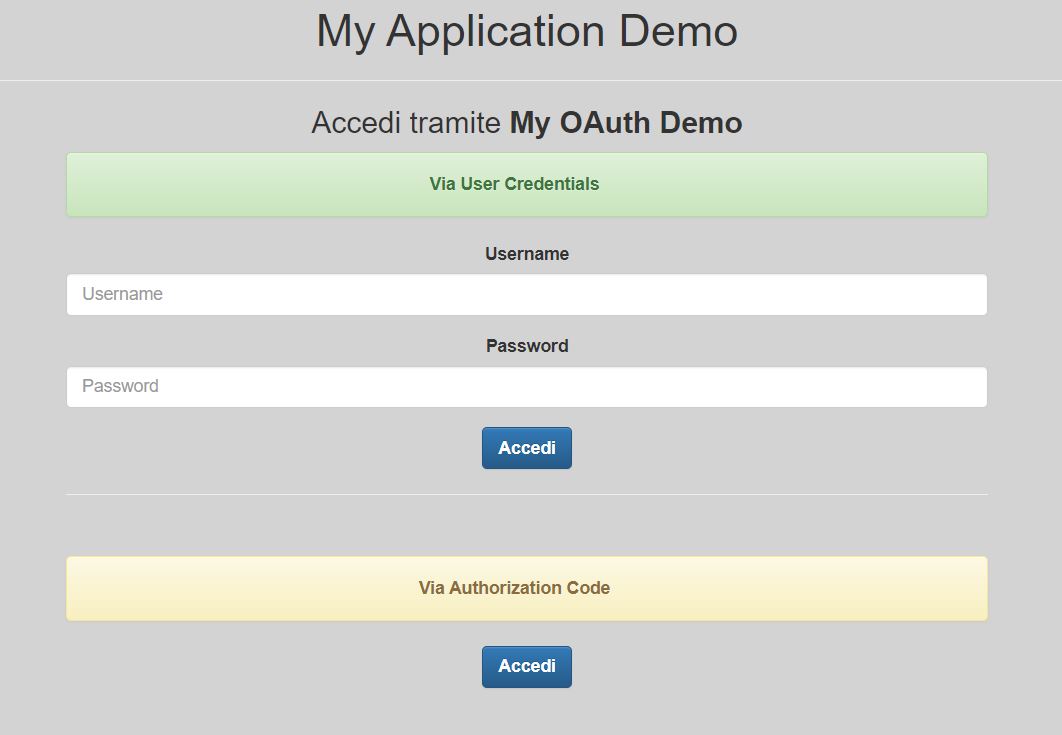


Figura 6 - Questa pagina rappresenta la pagina di login dell'utente sul client application



Figura 7 - Questa pagina rappresenta la pagina dove l'utente deve autorizzare il client ad accedere ai propri dati

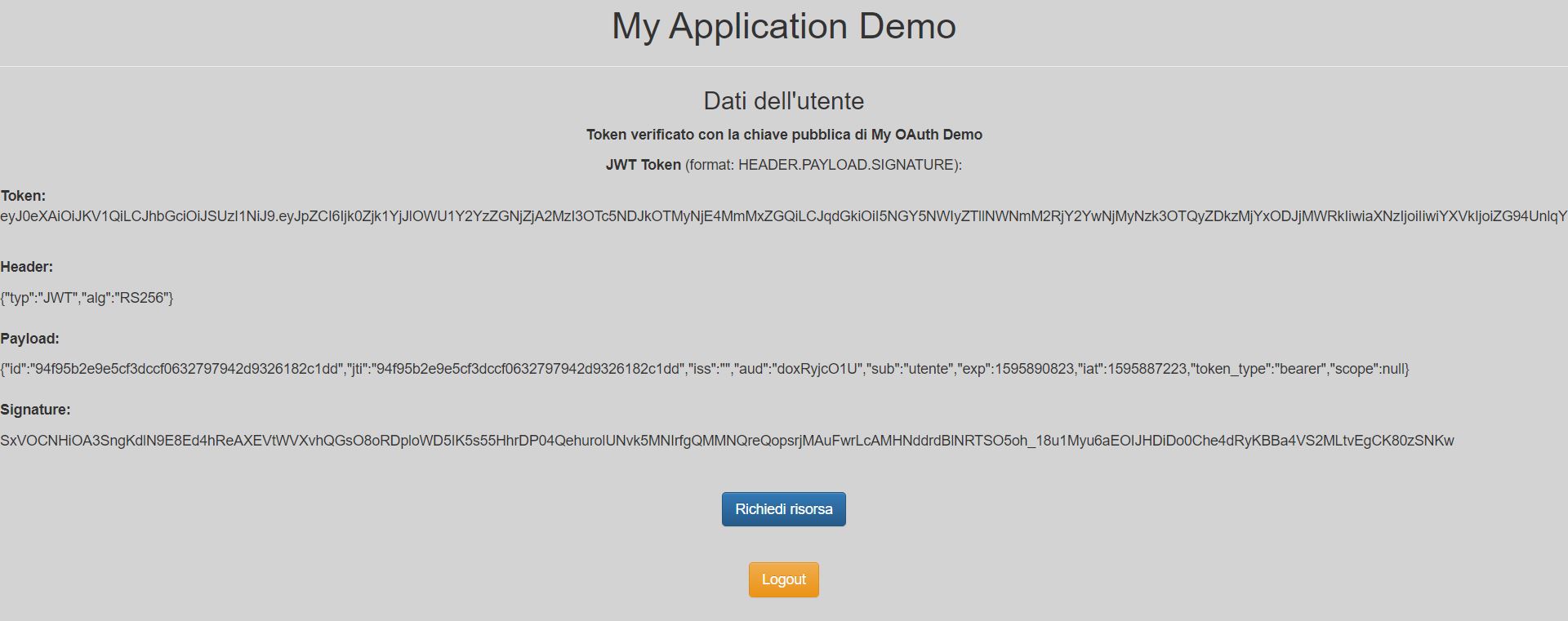


Figura 8 - Questa pagina rappresenta la pagina del client di accesso dell'utente. Contiene a titolo esplicativo il token ricevuto dal server

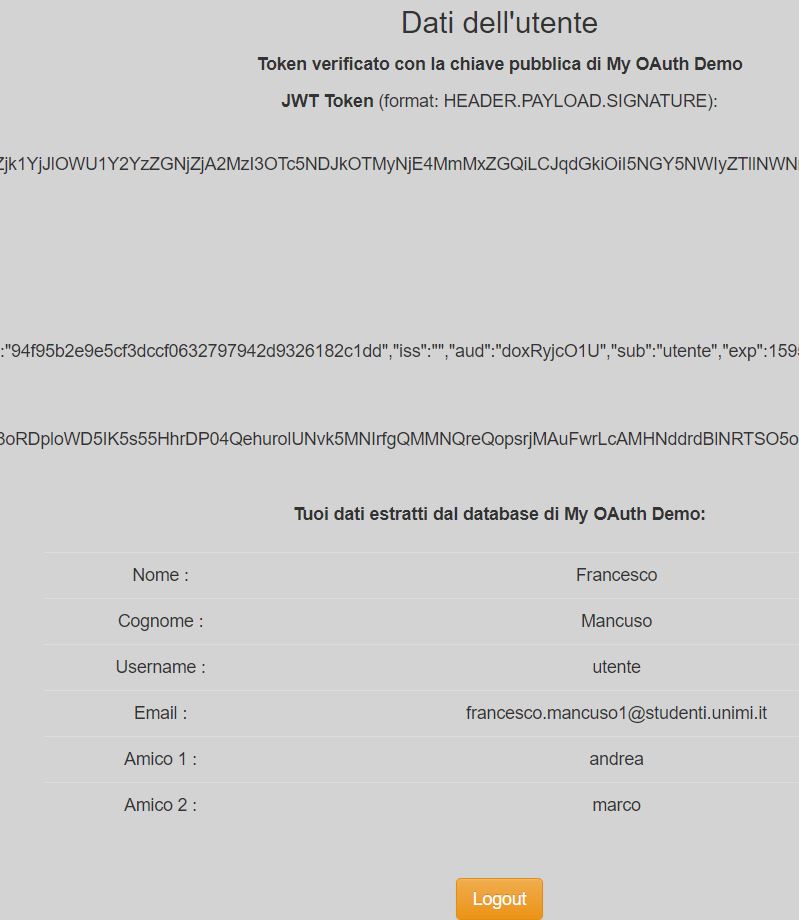


Figura 9 - Questa immagine contiene i dati dell'utente presi dall'authorization server una volta che il client ha fornito il token di accesso

Per visionare il progetto un utente è stato creato di default, ma è possibile crearne altri per fare delle prove.

Credenziali utente:

username: utente

password: 123456