

# KEYCLOAK IDENTITY PROVIDER



Bacaro  
Tech

CODE AND FUN



# IDENTITY PROVIDER SCRUM STORY

*As a user i want to login with  
email and password*

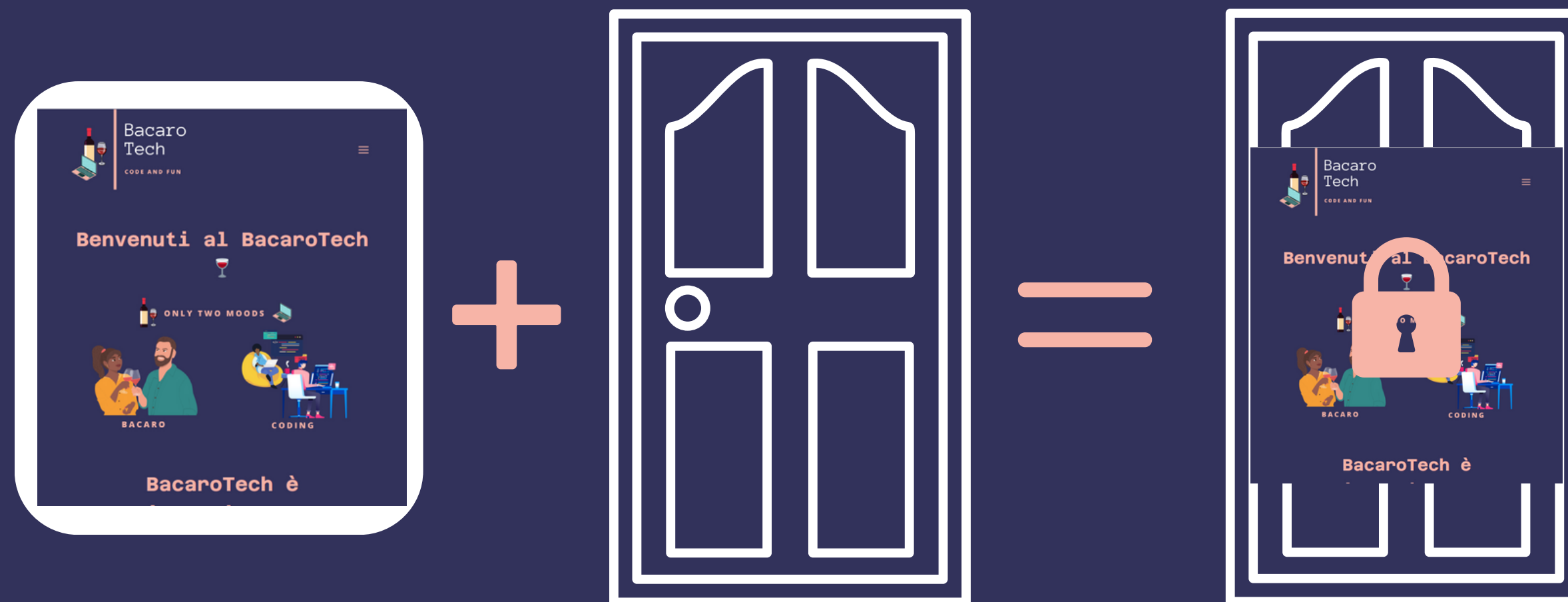


Bacaro  
Tech  
CODE AND FUN

# IDENTITY PROVIDER

## CASO D'USO

Ipotizziamo che abbiamo creato un applicativo web (sito) e che questo software richiede un controllo degli accessi (login)



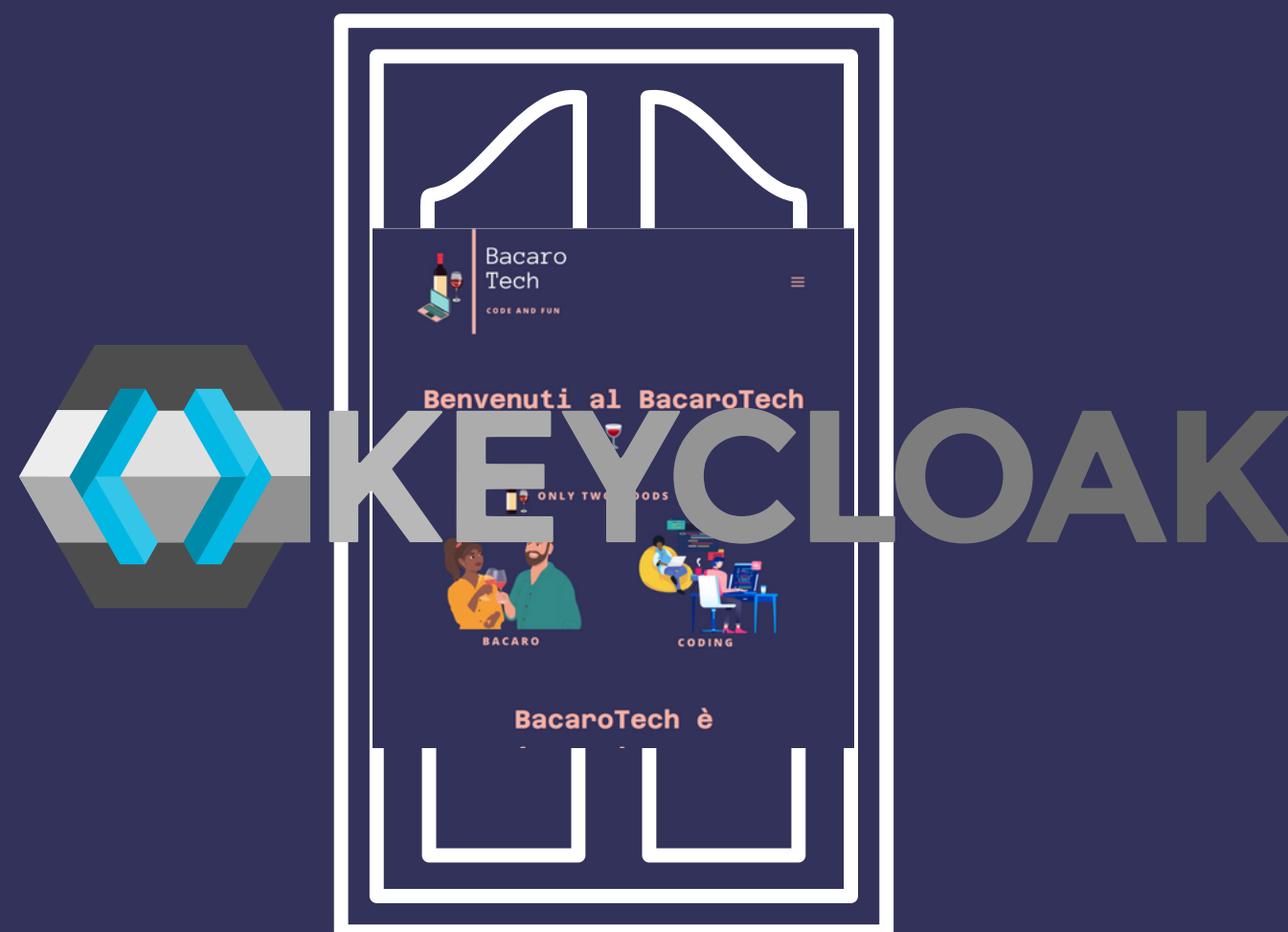


Bacaro  
Tech  
CODE AND FUN

# IDENTITY PROVIDER

## CASO D'USO

Per realizzare questa storia utilizziamo uno strumento che offre (server) questa funzionalità





Bacaro  
Tech  
CODE AND FUN

# IDENTITY PROVIDER

## CASO D'USO

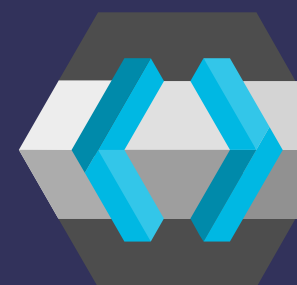
Keycloak utilizza i protocolli OAuth2 e il framework OpenID Connect (OIDC) per



+



=



KEYCLOAK

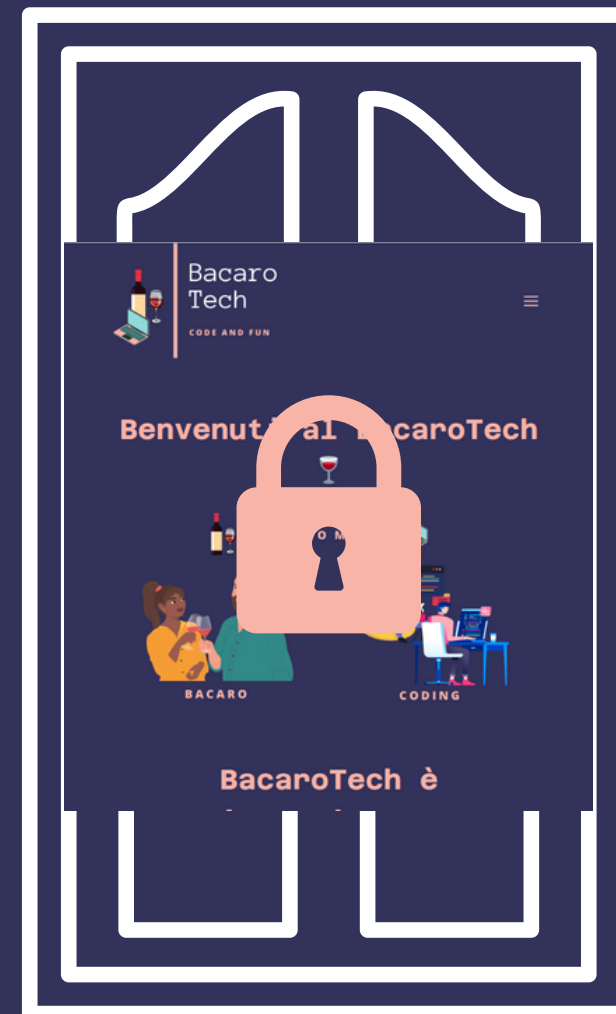


Bacaro  
Tech  
CODE AND FUN

# IDENTITY PROVIDER

## CASO D'USO

Abbiamo un sito web che richiede un controllo degli accessi: **login**





Bacaro  
Tech  
CODE AND FUN

# IDENTITY PROVIDER

## CASO D'USO



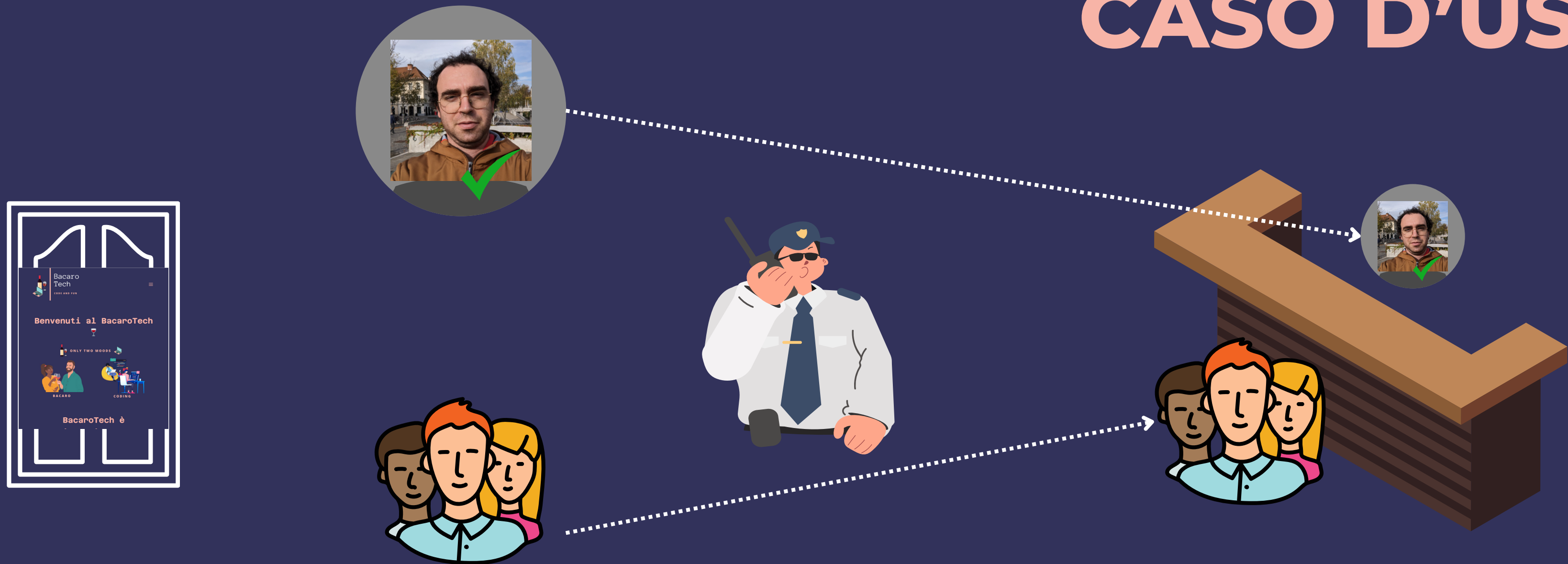
Deve poter distinguere se  
l'utente è riconosciuto dal  
nostro sistema



Bacaro  
Tech  
CODE AND FUN

# IDENTITY PROVIDER

## CASO D'USO



una volta che è stato riconosciuto, volete capire a quali risorse può accedere



# CHE COS'È KEYCLOAK



# CHE COS'È

ABBIAMO CHIESTO ALLE AI DI DARCI UNA  
DEFINIZIONE IN MENO DI 20 PAROLE

GPT

*Keycloak è un sistema open-source di gestione delle identità e degli accessi, facilitando l'autenticazione e l'autorizzazione in applicazioni e servizi.*

*Keycloak è una piattaforma open source per la gestione delle identità e degli accessi (IAM).*

*Keycloak semplifica l'autenticazione e l'autorizzazione per le applicazioni.*

BARD

p.s. bard ha voluto sottolineare il numero di parole usate nelle sue risposte...



Bacaro  
Tech  
CODE AND FUN

# CHE COS'È



## OPEN SOURCE

## IDENTITY AND ACCESS MANAGMENT

<https://www.keycloak.org/>

*....diffidate delle imitazioni, solo videocassette originali Walt Disney Home Video*

# CHE COS'È UN IAM

IAM sta per Identity Access Managment ed è un sistema informatico che ha lo scopo di gestire l'identità digitale degli utenti

**AUTENTICAZIONE**



**AUTORIZZAZIONE**



**IDENTITY ACCESS MANAGMENT**

*...ci sarebbero anche i ruoli, i gruppi e altre cose, ma è un workshop e ci prudono le mani*

# AUTHN AUTENTICAZIONE

Risponde alla domanda



are you



serve per identificare chi è l'utente che sta tentando di accedere alle nostre risorse



Bacaro  
Tech  
CODE AND FUN

# AUTHN AUTENTICAZIONE

Vado a trovare un  
amico e busso alla  
porta

*Sono Bacaro Tech*



*chi sei?*



# AUTHR AUTORIZZAZIONE

Risponde alla domanda

are you **allowed** to  
do this or not



Serve per capire se puoi fare quello che vuoi fare



Bacaro  
Tech  
CODE AND FUN

# AUTHR AUTORIZZAZIONE

*Posso  
passare?*



*Prego*







# CARATTERISTICHE KEYCLOAK

## **Single-Sign On**

Login once to multiple applications

## **Standard Protocols**

OpenID Connect, OAuth 2.0 and SAML 2.0

## **Centralized Management**

For admins and users

## **Adapters**

Secure applications and services easily

## **LDAP and Active Directory**

Connect to existing user directories

## **Social Login**

Easily enable social login

## **Identity Brokering**

OpenID Connect or SAML 2.0 IdPs

## **High Performance**

Lightweight, fast and scalable

## **Clustering**

For scalability and availability

## **Themes**

Customize look and feel

## **Extensible**

Customize through code

## **Password Policies**

Customize password policies

OAuth 2.0

OIDC

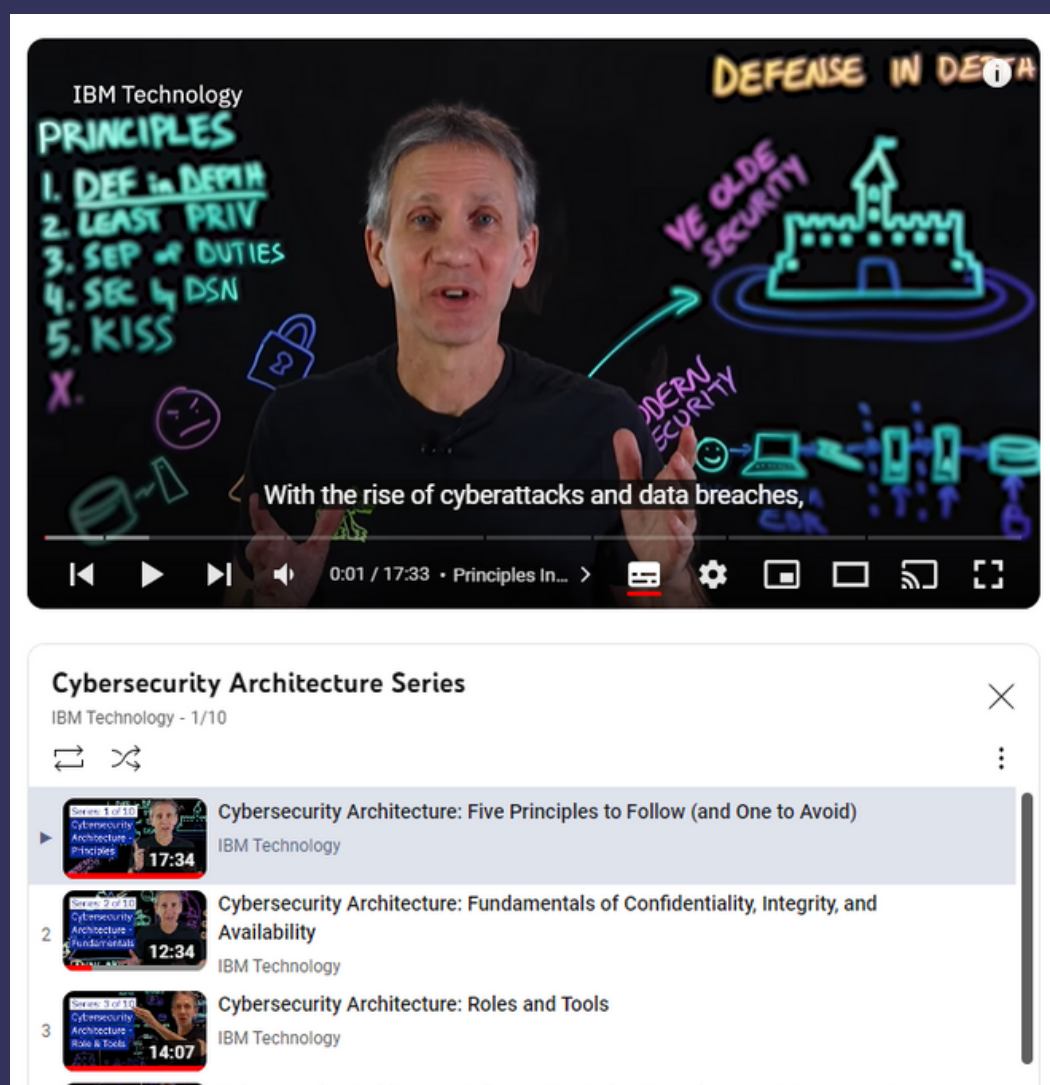
Users

Custom Themes



# CYBERSECURITY

Keycloak o altri Identity Provider rientrano in un altro consiglio della cybersecurity che è il concetto di **Security by design/by default**



Tra le fonti per  
approfondire questi  
argomenti suggeriamo  
questa playlist di IBM



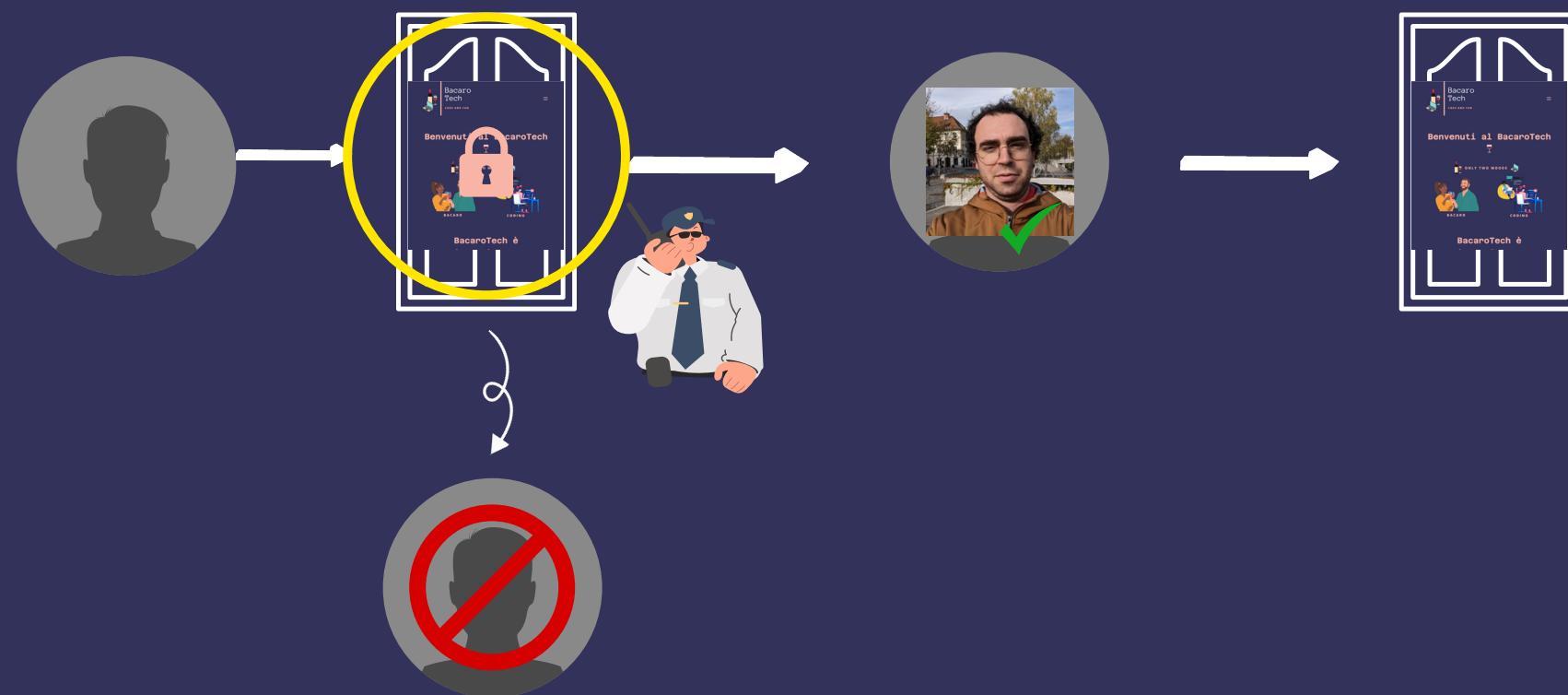
*...argomento troppo vasto per questa serata*

# COSA ANDIAMO A VEDERE OGGI?

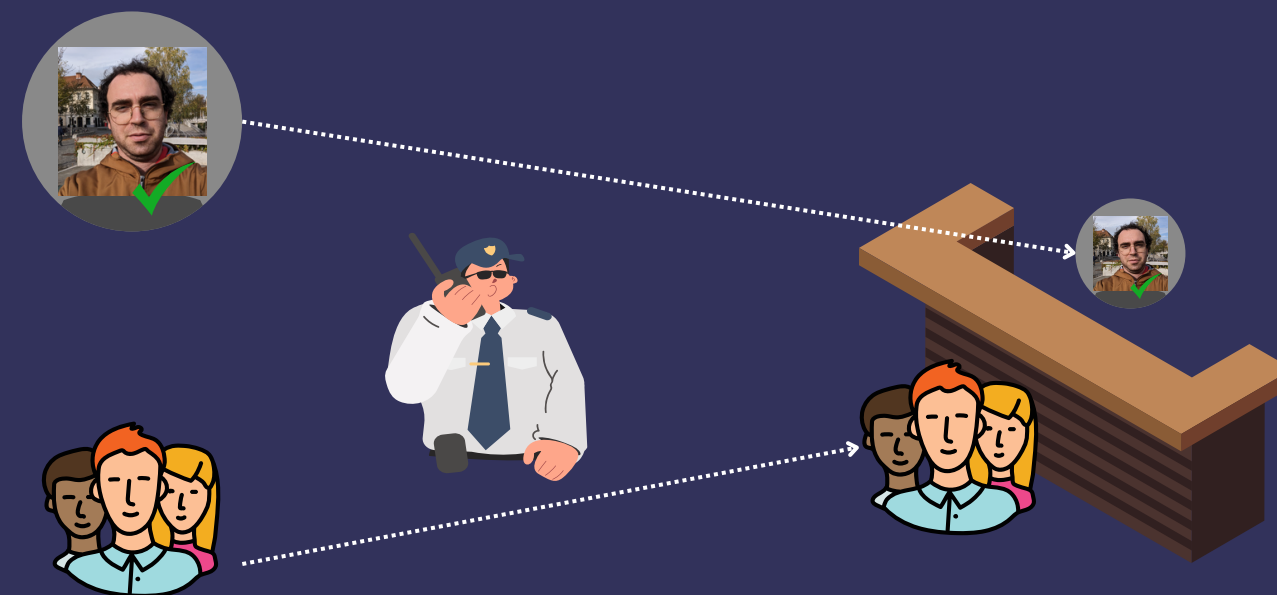
# WORKSHOP 1

Dato questo schema...

## AUTENTICAZIONE



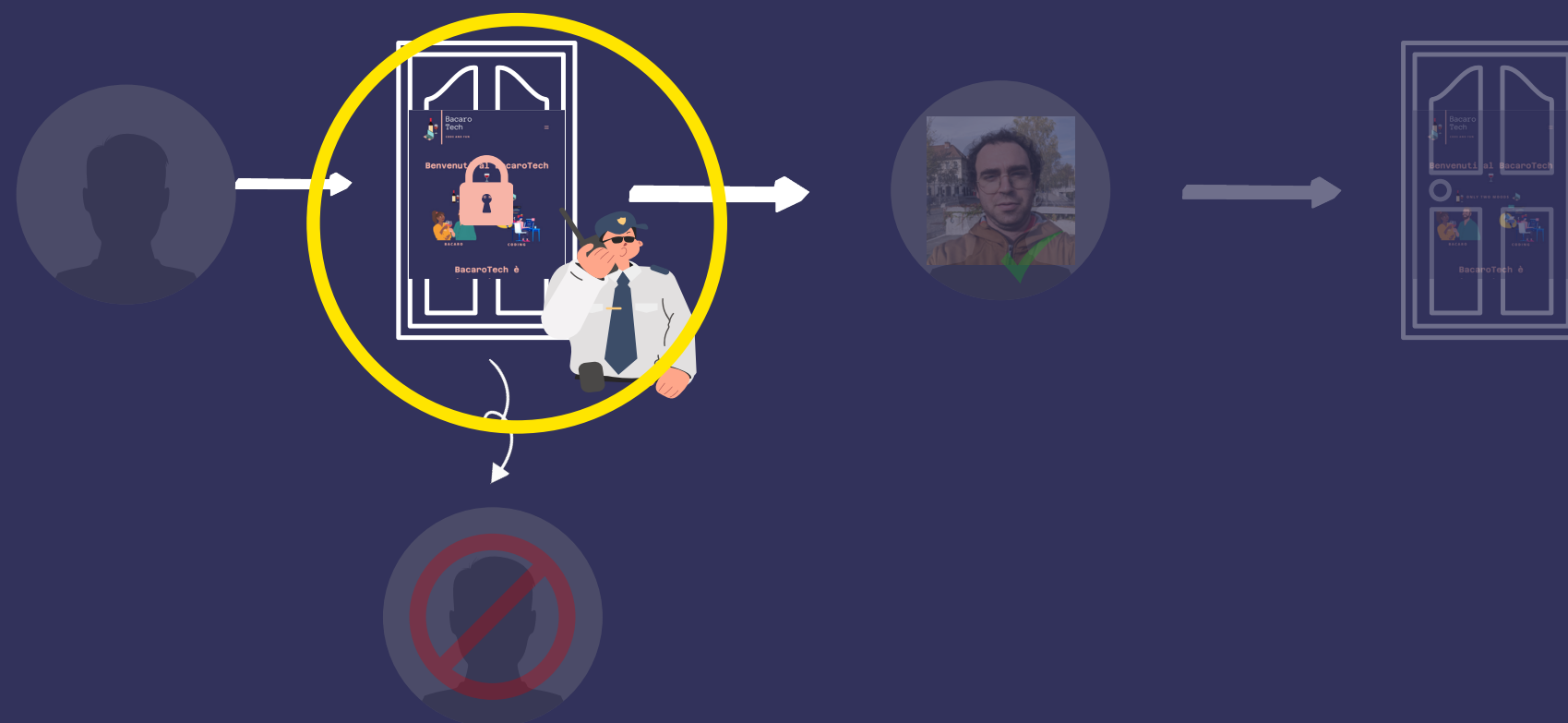
## AUTORIZZAZIONE



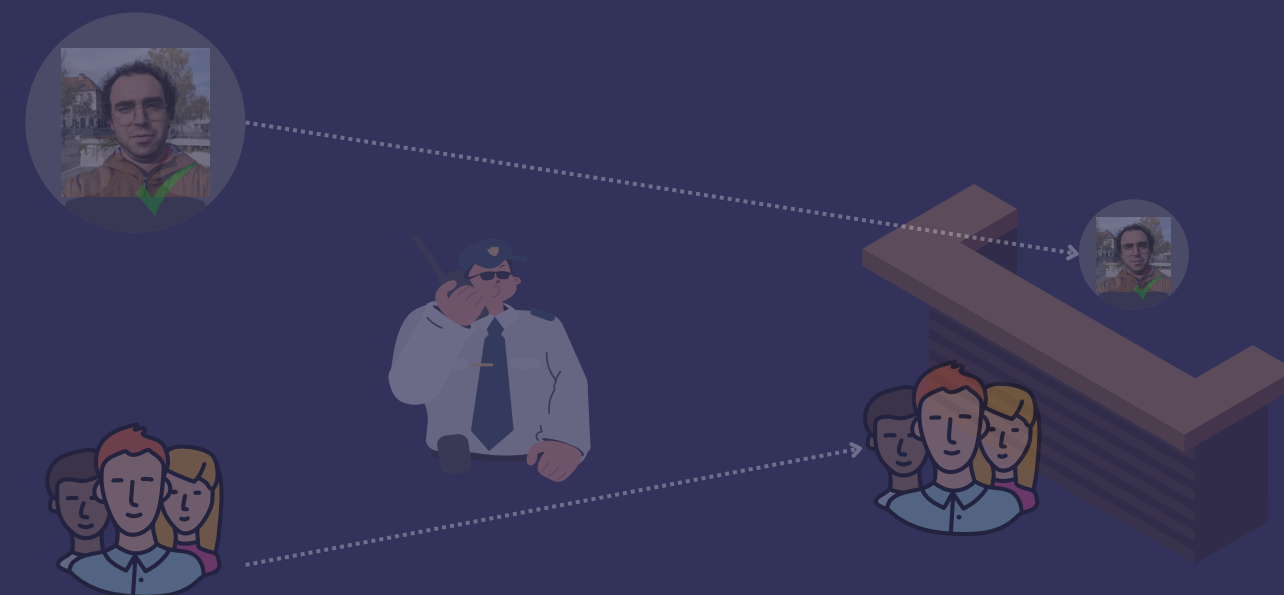
# WORKSHOP 1

Andremo a vedere il meccanismo di autenticazione

## AUTENTICAZIONE



## AUTORIZZAZIONE

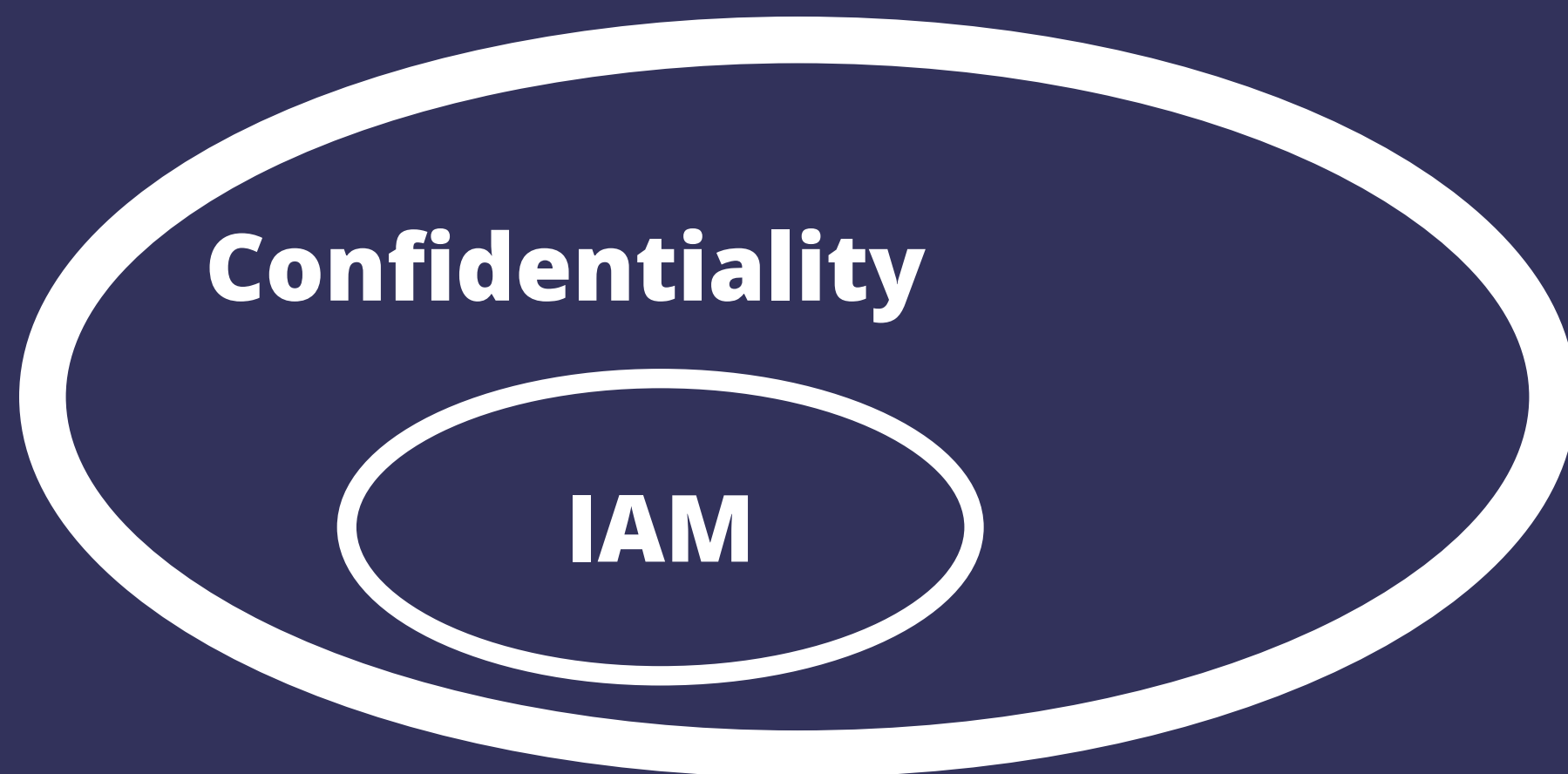


# COSA ANDIAMO A VEDERE OGGI?



# CONFIDENZIALITÀ CONFIDENTIALITY

L'access control è fa parte della confidentiality, che è uno dei tre pilastri della Cybersecurity (Integrity e Availability sono gli altri due )



**C**onfidentiality  
**I**ntegrity  
**A**vailability

*...argomento troppo vasto per questa serata... andiamo avanti...*



# OAUTH 2.0

<https://oauth.net/2>

**Visita il sito: sintetico e semplice**



OAUTH 2.0 è uno protocollo industriale standard di autorizzazione. Consente di accedere alle risorse senza rilevare le credenziali.



# OAUTH 2.0

## ACCESS TOKEN

Stringa che il client usa per fare richieste di accesso alle risorse

## OAUTH SCOPE

Autorizzazioni specifiche che un utente concede a un'applicazione

## REFRESH TOKEN

Stringa che il client usa per ottenere un nuovo token senza interazione dell'utente

## GRANT TYPE

Specifica il metodo attraverso il quale un'applicazione ottiene l'accesso a una risorsa



# GRANT TYPE

Ci sono diversi flussi di autorizzazione:

**AUTHORIZATION CODE**

**PKCE**

**CLIENT CREDENTIAL**

**DEVICE CODE**

**REFRESH TOKEN**

(Legacy)

**PASSWORD GRANT**

**IMPLICIT FLOW**



# GRANT TYPE

Ci sono diversi flussi di autorizzazione:

**AUTHORIZATION CODE**

**PKCE**

**CLIENT CREDENTIAL**

**DEVICE CODE**

**REFRESH TOKEN**

(Legacy)

**PASSWORD GRANT**

**IMPLICIT FLOW**



IL MODO PIÙ EFFICACE PER  
CAPIRE QUESTI TERMINI È CON LA  
PRATICA E KEYCLOAK CI AIUTERÀ  
IN QUESTO PERCORSO

# LAUNCH KEYCLOAK

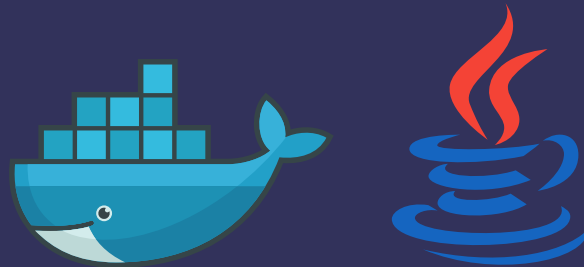
IT'S TIME TO CODING...

we ci serve keycloak



# LAUNCH KEYCLOAK

1



avviamo keycloak:  
Docker o OpenJDK

**http://localhost:8090** e login



2

3



creiamo il primo realm

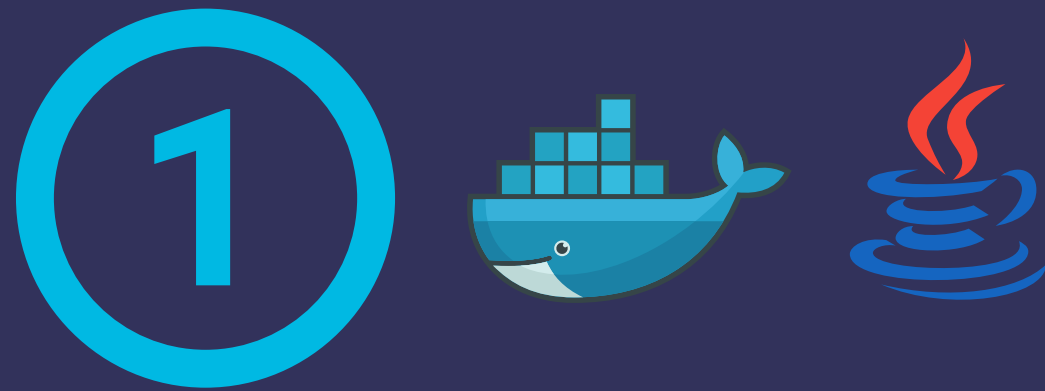
creiamo il client



4



# AVVIARE KEYCLOAK



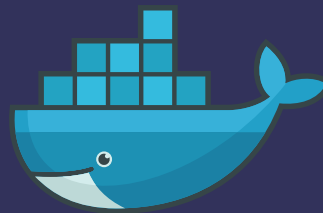
Ci sono diversi modi per avviare un server keycloak. Noi useremo due tecniche:

- Docker:
  - avviando il container: `docker run ...`
  - oppure tramite docker-compose: `docker compose up ...`
- OpenJDK:
  - si scaricano i compilati
  - si deve avere una JVM



# AVVIARE KEYCLOAK

1



Noi vi consigliamo di usare il docker-compose preparato.  
Il comando per avviare il compose è:

**DOCKER COMPOSE -F "DOCKER-COMPOSE.YML" UP -D --BUILD**

```
keycloak-workshop
> dpage/pgadmin4:6.20 keycloak-workshop-pgadmin-1 - U...
> postgres:15.2-alpine keycloak-workshop-postgresdb-1 - U...
> quay.io/keycloak/keycloak:22.0.1 keycloak-workshop-key...
```

Una volta avviato bisogna controllare che tutti i container siano partiti correttamente.





# PRIMA LOGIN



**`http://localhost:8090`**

Una volta avviato keycloak, la prima cosa da fare è andare alla pagina di configurazione. La pagina viene configurata all'avvio. Nel docker-compose abbiamo impostato la porta 8090

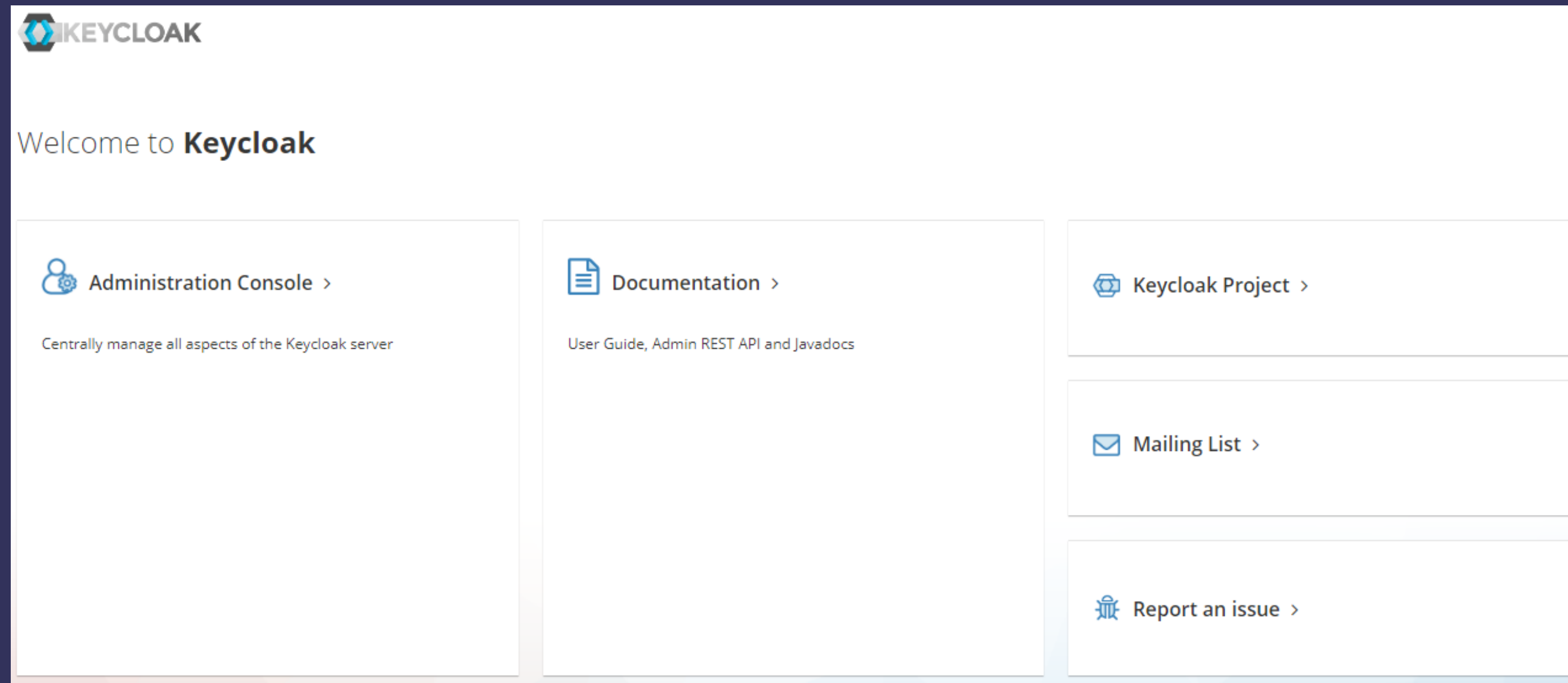


# PRIMA LOGIN

2



<http://localhost:8090>



3



è fondamentale creare un realm diverso da quello di default

# CREIAMO IL REALM

siru

### Create realm

A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can

Resource file

Drag a file here or browse to upload

Browse... Clear

1

Upload a JSON file

Realm name \*

Enabled

☒ On

Create

Cancel

# COS'È UN REALM

3

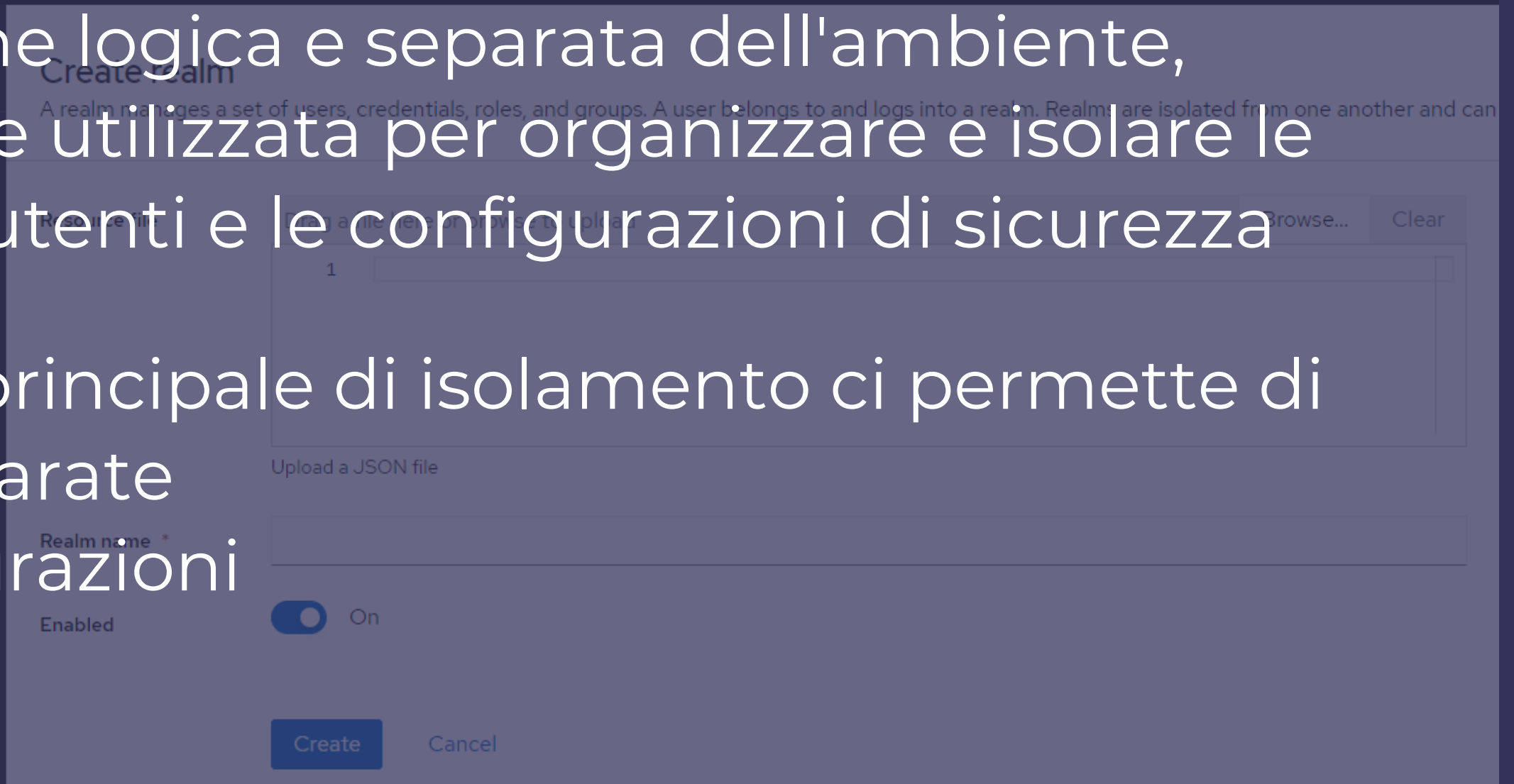


## REALM

Suddivisione logica e separata dell'ambiente, solitamente utilizzata per organizzare e isolare le risorse, gli utenti e le configurazioni di sicurezza

Funzione principale di isolamento ci permette di tenere separate

- configurazioni
- client
- utenti



The screenshot shows a 'Create Realm' dialog box. At the top, it says 'Create Realm' and 'A realm manages a set of users, credentials, roles, and groups. A user belongs to and logs into a realm. Realms are isolated from one another and can...'. Below this, there is a 'Resource ID' field with a 'Browse...' button and a 'Clear' button. A list of resources is shown below, with the first item labeled '1'. There is an 'Upload a JSON file' section with a text input field. Below that is a 'Realm name' field with an asterisk. At the bottom, there is an 'Enabled' toggle switch set to 'On', and 'Create' and 'Cancel' buttons.



# CREIAMO IL CLIENT



Il client è l'applicazione: web, server, mobile, altro che, tramite i vari protocolli, contatterà l'identity provider (IP) per l'autenticazione e l'autorizzazione.

Clients > Create client

## Create client

Clients are applications and services that can request authentication of a user.

1 General Settings

2 Capability config

3 Login settings

Client type ?

Client ID \* ?

Name ?

Description ?

Always display in UI ?

OpenID Connect

☐ Off

Next

Back

Cancel



# CREIAMO IL CLIENT

4



1 General Settings

2 Capability config

3 Login settings

Client authentication ☐ Off

Authorization ☐ Off

Authentication flow

- ☒ Standard flow
- ☐ Implicit flow
- ☐ OAuth 2.0 Device Authorization Grant
- ☐ OIDC CIBA Grant
- ☒ Direct access grants
- ☐ Service accounts roles

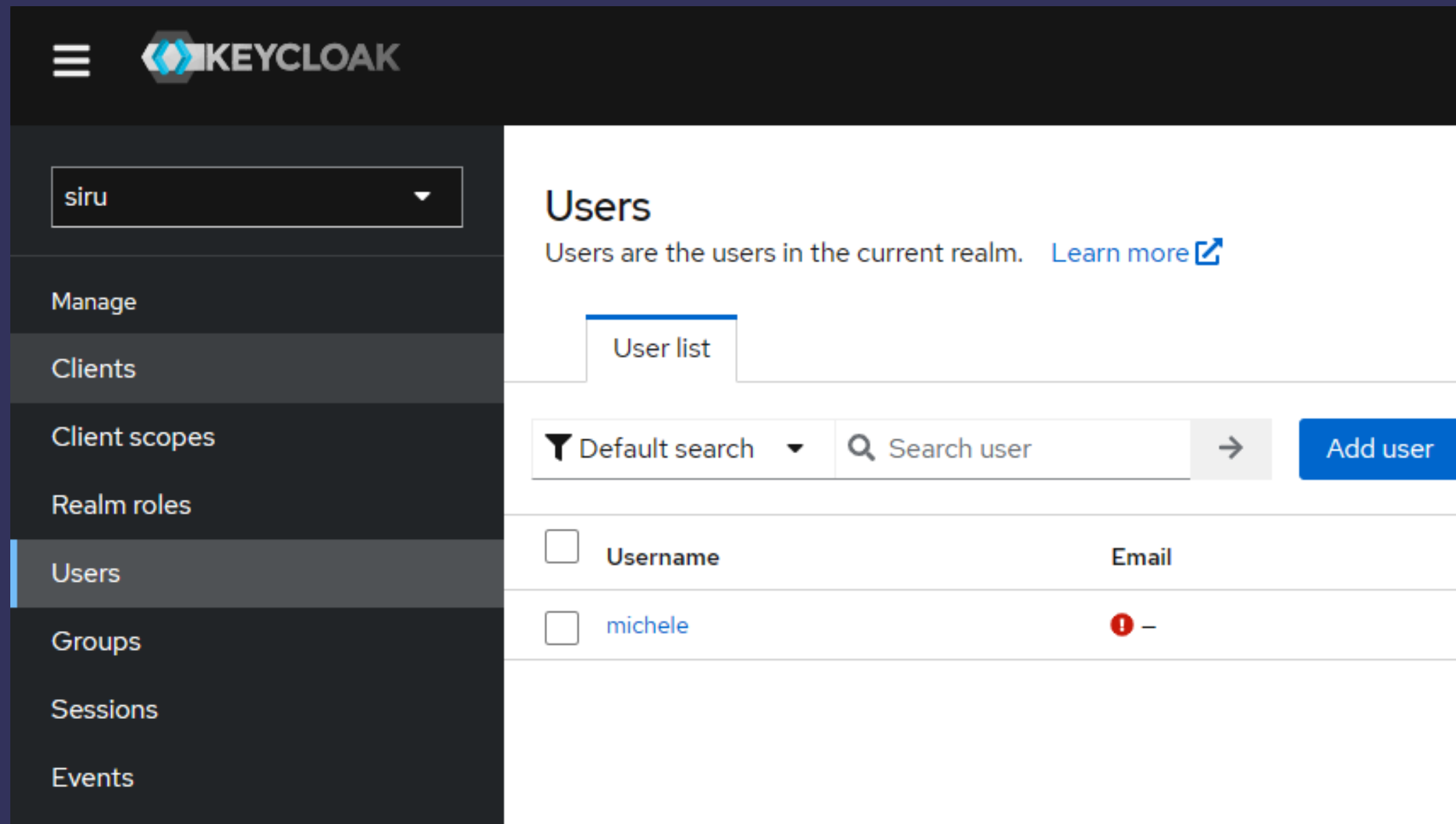
Creiamo il client attraverso gli step mantenendo le configurazioni di default.

Impareremo il significato delle varie configurazioni attraverso i workshop



# CREAZIONE UTENTE

Il primo step in assoluto è di creare un utente direttamente in Keycloak



Ora che abbiamo Keycloak, partiamo con il primo workshop...

# WORKSHOP NUMERO 1

## STANDARD FLOW





# WORKSHOP #1 STANDARD FLOW

La prima configurazione che proviamo è Authentication Flow: Standard Flow.  
In poche parole significa che per ottenere un token dovremo scambiare un codice, il code. Corrisponde al Grant-Type: Authorization-Code

## Capability config

Client authentication  
[?](#)

☐ Off

Authorization [?](#)

☐ Off

Authentication flow

☒ Standard flow [?](#)

☐ Direct access grants [?](#)

☐ Implicit flow [?](#)

☐ Service accounts roles [?](#)

☐ OAuth 2.0 Device Authorization Grant [?](#)

☐ OIDC CIBA Grant [?](#)



# STANDARD FLOW

Questo flusso prevede di ricevere il code in risposta dopo una redirect dall'identity provider. Dunque bisogna impostare l'url valido.

C'è il campo 'Valid redirect URIs' che è dove andiamo a configurare gli URL validi.

Valid redirect URIs ?	<input type="text" value="http://localhost:8081"/>
<input type="button" value="+ Add valid redirect URIs"/>	

<http://localhost:8081>



# STANDARD FLOW

Tramite un IDE o editor di testo, apriamo il file index.html, cerchiamo il tag script e mettiamo le configurazioni del nostro Identity Provider:

```
65 <script>
66 // #### CONFIGURAZIONE ####
67 // URL dell'endpoint di autorizzazione
68 const authorizationEndpoint = 'AUTH_ENDPOINT';
69 // ClientId
70 const clientId = 'CLIENT-ID';
71 // Url di redirect
72 const redirect = 'URL-CURRENT-PAGE';
73 // scope
74 const scope = 'openid profile email';
75 // Response Type
76 const responseTypeStandardFlow = 'code';
77
78 const responseTypeImplicitFlow = 'token';
```



# STANDARD FLOW

```
65 <script>
66   // #### CONFIGURAZIONE ####
67   // URL dell'endpoint di autorizzazione
68   const authorizationEndpoint = 'http://localhost:8090/realms/
   siru/protocol/openid-connect/auth';
69   // ClientId
70   const clientId = 'siru-fe';
71   // Url di redirect
72   const redirect = 'http://localhost:8081';
73   // scope
74   const scope = 'openid profile email';
75   // Response Type
76   const responseTypeStandardFlow = 'code';
77
78   const responseTypeImplicitFlow = 'token';
```



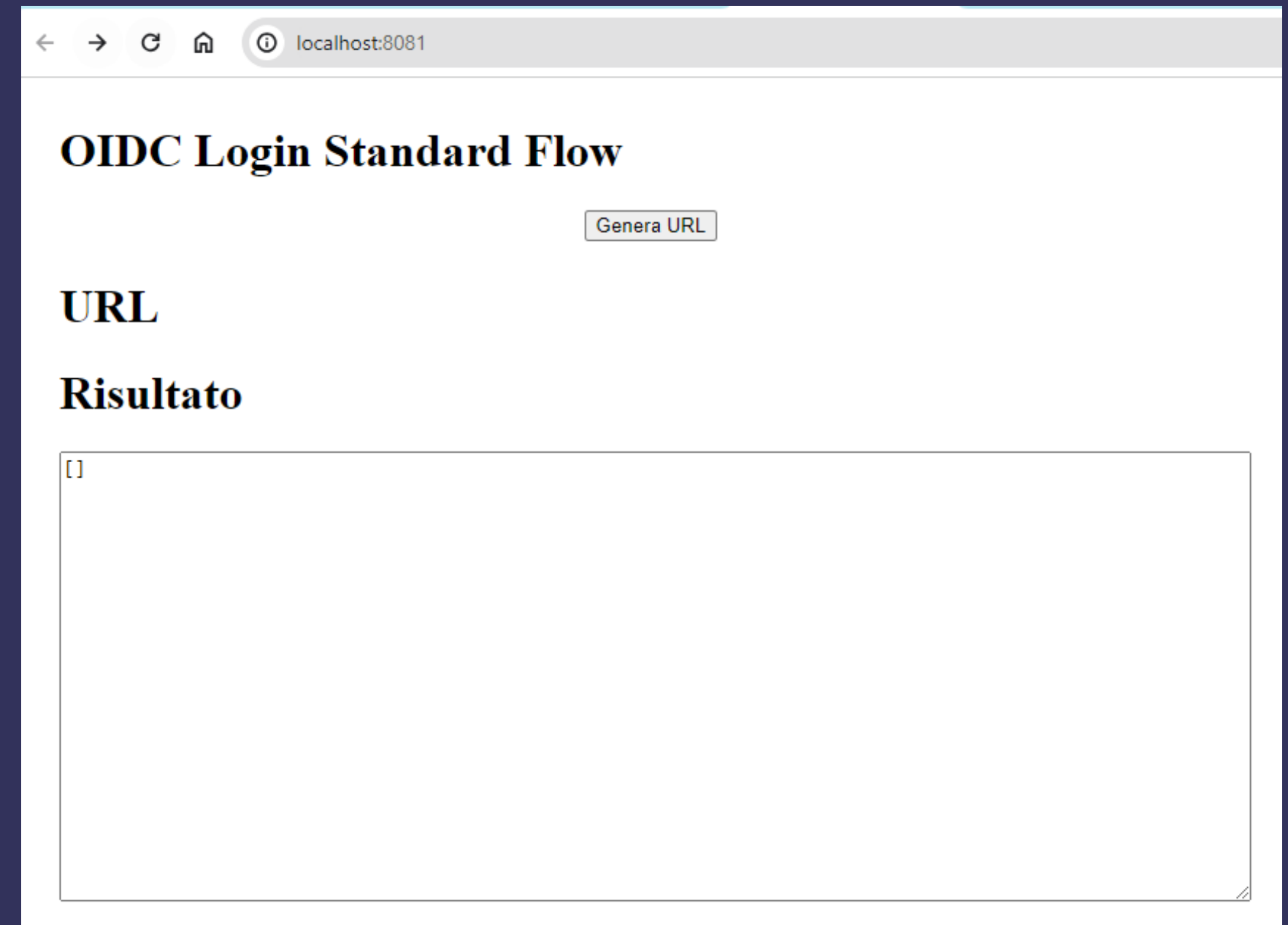
# STANDARD FLOW

Tramite il terminale, avviamo il lo script *npm start*

Si avvierà il tools *http-server* che è un server statico. Imposti la cartella e lui fornisce tutte le risorse dentro questa cartella.

Per vedere l'applicativo dobbiamo andare nel browser alla pagina

**<http://localhost:8081>**



La parte che ci interessa a noi è la OIDC Login Standard Flow



# STANDARD FLOW

Cliccando sul bottone **Generate URL** si creerà un URL. La pagina è pensata per poter ispezionare le sue componenti nella sezione URL

## URL DI LOGIN

Cliccando su **Login** verremo reindirizzati verso un'altra pagina con la login di Keycloak

### OIDC Login Standard Flow

Genera URL

[Login](#)

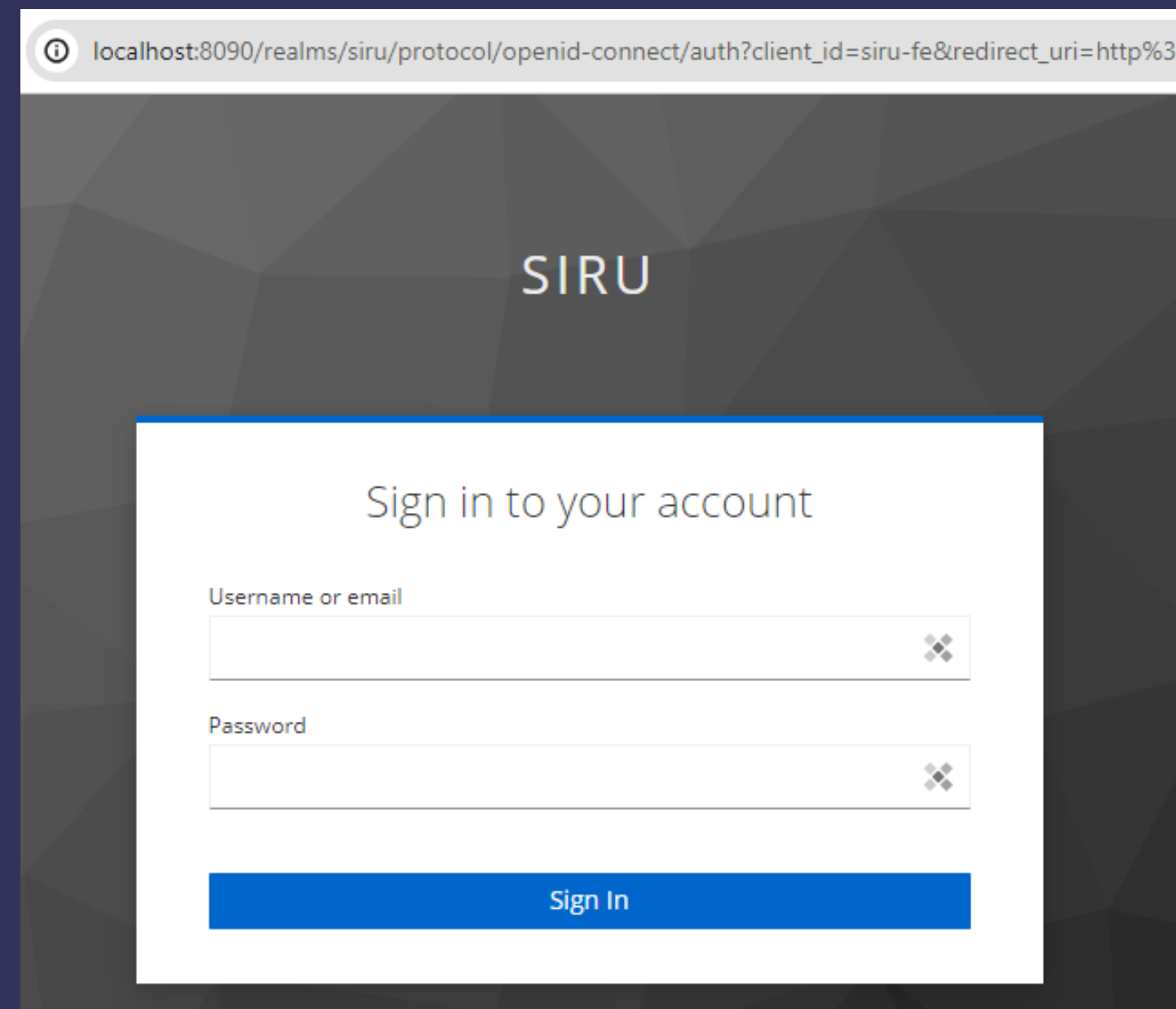
`http://localhost:8090/realms/siru/protocol/openid-connect/auth?client_id=siru-fe&redirect_uri=http%3A%2F%2Flocalhost%3A8081&response_type=code&scope=openid%20profile%20email&state=itXhZrUDbx6caRx4&nonce=PdfKZDmC6WbO4xQ1`

### URL

```
{  "authorizationEndpoint": "http://localhost:8090/realms/siru/protocol/openid-connect/auth",  "clientId": "siru-fe",  "redirectUri": "http%3A%2F%2Flocalhost%3A8081",  "scope": "openid%20profile%20email",  "responseType": "code",  "stateValue": "itXhZrUDbx6caRx4",  "nonceValue": "PdfKZDmC6WbO4xQ1"}
```

# STANDARD FLOW

Digitiamo username e password.



localhost:8090/realms/siru/protocol/openid-connect/auth?client\_id=siru-fe&redirect\_uri=http%3

SIRU

Sign in to your account

Username or email

Password

Sign In



# STANDARD FLOW

Una volta fatta la login ritorneremo sulla pagina <http://localhost:8081> e se facciamo attenzione vedremo dei parametri in più al nostro url.

## OIDC Login Standard Flow

Genera URL

### URL

### Risultato

Esempio:

[http://localhost:8081/?state=itXhzrUDbx6caRx4&session\\_state=458c1550-c38d-471d-81d3-9d69430f17c1&code=63d71981-bc8b-4179-a9a1-2b5a4ff6e948.458c1550-c38d-471d-81d3-9d69430f17c1.d637b91c-e09e-4bd0-97ee-0567f82d6a4d](http://localhost:8081/?state=itXhzrUDbx6caRx4&session_state=458c1550-c38d-471d-81d3-9d69430f17c1&code=63d71981-bc8b-4179-a9a1-2b5a4ff6e948.458c1550-c38d-471d-81d3-9d69430f17c1.d637b91c-e09e-4bd0-97ee-0567f82d6a4d)

```
{
  "key": "state",
  "value": "itXhzrUDbx6caRx4",
},
{
  "key": "session_state",
  "value": "458c1550-c38d-471d-81d3-9d69430f17c1",
},
{
  "key": "code",
  "value": "63d71981-bc8b-4179-a9a1-2b5a4ff6e948.458c1550-c38d-471d-81d3-9d69430f17c1.d637b91c-e09e-4bd0-97ee-0567f82d6a4d"
}
]
```





# STANDARD FLOW

Questi parametri si possono visualizzare meglio nella parte **Risultato**

Se non abbiamo avuto messaggi di errore il campo che ci serve per il prossimo step è **CODE**

Contengono dei messaggi e dei codici che ci serviranno per ottenere il token.

**Il primo passo è fatto, sei loggato!**

← → ↻ 🏠 ⓘ localhost:8081/?state=itXhzrUDbx6caRx4&session\_state=458c1550-c38d-471d-81d3-9d69430f17c1&code=63d71981-bc8b-4179-a9a1-2b5a4ff6e948.458c1550-c38d-471d-81d3-9d69430f17c1.d637b91c-e09e-4bd0-97ee-0567f82d6a4d

## OIDC Login Standard Flow

Genera URL

### URL

### Risultato


```
[
  {
    "key": "state",
    "value": "itXhzrUDbx6caRx4"
  },
  {
    "key": "session_state",
    "value": "458c1550-c38d-471d-81d3-9d69430f17c1"
  },
  {
    "key": "code",
    "value": "63d71981-bc8b-4179-a9a1-2b5a4ff6e948.458c1550-c38d-471d-81d3-9d69430f17c1.d637b91c-e09e-4bd0-97ee-0567f82d6a4d"
  }
]
```



# STANDARD FLOW

Ora dobbiamo ottenere un token. Per fare questo dobbiamo effettuare una chiamata POST mettendo dei parametri specifici.

Chiamiamo in soccorso **POSTMAN**.

1. Creiamo una nuova chiamata.
2. Impostiamo il metodo 
3. Incolliamo questo URL: `http://localhost:8090/realms/siru/protocol/openid-connect/token`
4. Impostiamo 'x-www-form-urlencoded' in body
5. Mettiamo i seguenti campi
  - a. `grant_type` = **authorization\_code**
  - b. `client_id` = il client-id
  - c. `code` = Il code che abbiamo ottenuto nel passaggio precedente
  - d. `redirect_uri` = l'url di redirect



# STANDARD FLOW

Ecco un esempio di campi compilati

POST

http://localhost:8090/realms/siru/protocol/openid-connect/token

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

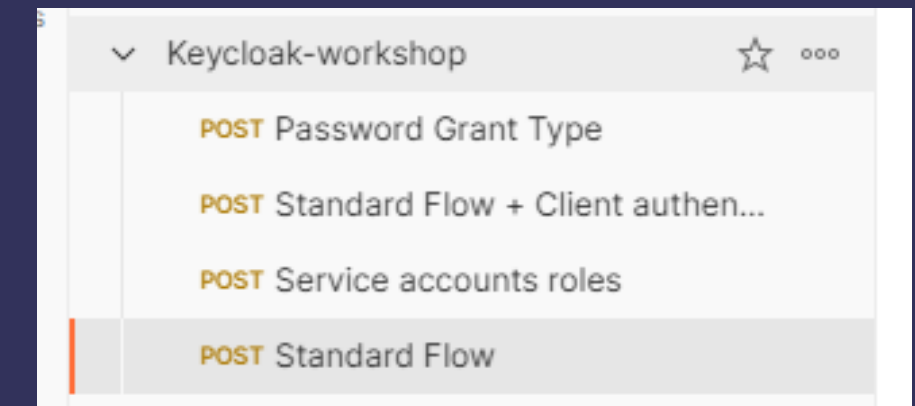
GraphQL

	Key	Value	Des...	...	Bulk Edit
<input checked="" type="checkbox"/>	grant_type	authorization_code			
<input checked="" type="checkbox"/>	client_id	siru-fe			
<input checked="" type="checkbox"/>	code	236e5441-f6f0-4971-9f13-e6b16b3dd9b0.458c1550-c38d-471d-81d3-9d69430f17c1....			
<div><div></div><div></div><div></div></div> <input checked="" type="checkbox"/>	redirect_uri	http://localhost:8081			<div></div>
	Key	Value	Description		



# STANDARD FLOW

Abbiamo preparato una collection POSTMAN da importare. Contiene le chiamate POST per ottenere il token. Quella che ci interessa è la: **Standard Flow**



In alternativa potete effettuare la chiamata da Linea Di Comando con **CURL**:

```
curl --location 'http://localhost:8090/realms/siru/protocol/openid-connect/token' \
--header 'Content-Type: application/x-www-form-urlencoded' \
--data-urlencode 'grant_type=authorization_code' \
--data-urlencode 'client_id=siru-fe' \
--data-urlencode 'code=cd8e2638-2303-448a-b339-03d11890c8a5.2822ce0d-35dc-45df-bb95-935e2990556c.d637b91c-e09e-4bd0-97ee-0567f82d6a4d' \
```

```
D:\BacaroProjects\keycloak-workshop>curl --location 'http://localhost:8090/realms/siru/protocol/openid-connect/token' --header 'Content-Type: application/x-www-form-urlencoded' --data-urlencode 'grant_type=authorization_code' --data-urlencode 'client_id=siru-fe' --data-urlencode 'code=236e5441-f6f0-4971-9f13-e6b16b3dd9b0.458c1550-c38d-471d-81d3-9d69430f17c1.d637b91c-e09e-4bd0-97ee-0567f82d6a4d' --data-urlencode 'redirect_uri=http://localhost:8081' [ ]
```



Se tutto è andato bene avremo ottenuto:

- access token
- refresh token
- token id

[illegible]

```
"expires_in": 300,  
"refresh_expires_in": 1792,  
"refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpZTEUiwiaw2lkIiA6IClYmQyN2Y5OS1mNmQ0LTQ3MzcwYjVhZS0yOGE4MDk2M2MyMGMiifQ.  
eyJleHAiOiE3MDM3NzI5NDM5ImldhdCI6MTcwMzc3MTE1MSwiaWwRdPjoiZTEwZDFjbmJtZjY1Yi00OTYyLTgwMzItNDZlZWZmODcxODk3IiwiaXNzIjoiaHR0cDovL2xvY2Fs  
aG9zdDo4MDkwL3JlYWxtcy9zaXJ1Iiwic3ViOiJoim2E0ZGZkOTgtN2NiNy00YThtIlTk1YjYtOTE4  
NTVhMDhmMzA0IiwidHlwIjoiUmVmcmVzaCIsImF6ICI6InNpcnUtZmUmLiJCub25jZSI6IkZzTklreThFRHFzMtM2UXoiLCJzZXNzaW9uX3N0YXRlIjoiINDU4YzE1NTAtYzM4  
ZC00NzFkLTgxZDMtOWQ2OTQzMGYxN2MxIiwic2NvcGUiOiJvcGVuaWQgcHJvZm1sZSB1bWVpbCIsInNpZCI6IjQ1OGMxNTUwLWMzOGQtNDcxZC04MWQzLTlkbnjk0MzBmMTdj  
MSJ9.DG-pfMZTf7pTKjSKXZKtJX11_L-Q7RTEn7h2toYan7U",  
"token_type": "Bearer",
```

```
"id_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IWFiaSlkUiwiwiaz2lkIA6ICjBUFpmVHNPMGLJ0ThyQUtCULZjLXdwHkwMDdsTTgtODhlNjNSNjFhUno4In0.  
eyJleHAiOjE3MDM3NzE0NTesImlhdCI6MTcwMzc3MTE1MSwiYXV0aF90aW11IjoxNzAzNzczNDazLCJqdGkiOiIwMjlhMTNhNy0xNWRILTRlNGUtYjZiYi0xYWUzYmQ3OTQ1  
ZTIiLCJpc3MiOiJodHRwOi8vbG9jYXRob3N0OjgwTAVcmVhbG1zL3NpenUlClJhdWQiOiJzaXJ1LWZlIiwic3ViIjoim2E0ZGZkOTgtN2NiNy00YThtLTk1YjYtOTE4NTVh  
MDhmMZA0IiwidHlwIjoisuQilCJhenAiOiJzaXJ1LWZlIiwibm9uY2UiOiJGc05Ja3k4RUxczEzNlF6Iiwic2Vzc2lvdDp5dGF0ZSI6IjQ1OGMxNTUwLWMzM0GQtNDcxZC04  
MWQzLTlkNjk0MzBmMTdjMSIsImF0X2hhc2giOiJJWlV0dFp5d09hU2lyNGR5akp5aUFRIiwiYWNyIjoimCIsInNpZCI6IjQ1OGMxNTUwLWMzM0GQtNDcxZC04MWQzLTlkNjk0  
MzBmMTdjMSIsImVtYWlsX3Zlcm1maWVkIjpmYXxzZWicHJlZmVycmVkdXVzZXJuYWI1IjoibWljagVsZSIsImdpdmVuX25hbWUiOiIiLCJmYW1pbHlfbmFtZSI6IiJ9.  
sScJ5EXBFVVqm056P3leQFEgoLgWUUoDaQq9rKaMDNCFXHBNqIiqBa0xSDQHCDjiFGv72z0gkELfTGQo-FmlQBWwIv6VKypZEgCKVQVoHtWusePEqnXYHVkdo7i0g0DIjBM  
D7YF8X2j0smTRZ_qJ59AZH6yLhm14r6BqU7Kb_kodz7518uKlAjZXBz8mUyoyKdq3DnUIpqPOs5reTx4VpU_JgqPVD6o5EiBBWM-X6VHAD_WsQU1IxqFpCEsYwf9QWuqkfBH  
vjyJYZjm0yqvISPfHaZdzkJMH2sKETBxF2Baxm72uoMsJ0CZRyh05_qbQywgSbcDSV1oq9omIW0IIa",
```





# STANDARD FLOW - CONSIDERAZIONI

## PRO

- Sicurezza Robusta
- Consenso dell'utente
- Standard Flow è supportato da OpenID Connect
- Flusso adatto per Applicazioni Web
- Supporto Multi-Fattore
- Riduzione del Rischio di Attacchi CSRF
- Integrazione con Servizi di Autorizzazione
- Con il PKCE si evita di utilizzare il secret

## CONTRO

- Complessità del Flusso
- Richiede una Registrazione Preventiva del Client
- Se presenti, Richiede la Gestione di Codici Segreti

# WORKSHOP NUMERO 1

## IMPLICIT FLOW



# WORKSHOP #1 IMPLICIT FLOW

La configurazione Implicit Flow corrisponde al OAuth2 di tipo Implicit Grant.

**Capability config**

**Client authentication** ☐ Off

**Authorization** ☐ Off

**Authentication flow**

- ☐ Standard flow
- ☒ Implicit flow
- ☐ OAuth 2.0 Device Authorization Grant
- ☐ OIDC CIBA Grant
- ☐ Direct access grants
- ☐ Service accounts roles

**Valid redirect URIs**   
[+ Add valid redirect URIs](#)

**Valid post logout redirect URIs**   
[+ Add valid post logout redirect U](#)

**Web origins**   
[+ Add web origins](#)

**Admin URL**

**Capability config**

**Client authentication** ☐ Off

**Authorization** ☐ Off

**Authentication flow** ☒ Standard flow

In keycloak è necessario impostare un URL di ritorno valido. Per farlo è necessario abilitare almeno una volta lo 'Standard Flow'. Sembra una cosa controintuitiva ma nell'esempio capiremo perché gli autori di Keycloak hanno fatto questo





# IMPLICIT FLOW

Torniamo alla pagina e vediamo il lato dedicato all'Implicit flow

Come nel caso precedente, generiamo l'URL, clicchiamo su Login e vediamo che dati ci ritorna

## OIDC Login Implicit Flow

Genera URL

URL

Risultato

[ ]

## OIDC Login Implicit Flow

Genera URL

[Login](#)

`http://localhost:8090/realms/siru/protocol/openid-connect/auth?client_id=siru-fe&redirect_uri=http%3A%2F%2Flocalhost%3A8081&response_type=token&scope=openid%20profile%20email`

URL

```
{
  "authorizationEndpoint": "http://localhost:8090/realms/siru/protocol/openid-connect/auth",
  "clientId": "siru-fe",
  "redirectUri": "http%3A%2F%2Flocalhost%3A8081",
  "scope": "openid%20profile%20email",
  "responseType": "token"
}
```



Questo è l'url di ritorno:

[illegible]

## Risultato

[illegible]



# IMPLICIT FLOW - CONSIDERAZIONI

Si può notare come l'implicit flow non necessita di una seconda chiamata per ottenere il token, ma ci viene restituito direttamente nell'URL.

Tuttavia, va notato che l'Implicit Flow è considerato obsoleto in OAuth 2.0 a favore del flusso Authorization Code.

## PRO

- Semplicità di implementazione
- Flusso Ottimizzato per Applicazioni Client-Side

## CONTRO

- Esposizione del Token nell'URL
- Flusso Ottimizzato per Applicazioni Client-Side
- Limitata Possibilità di Refresh Token
- Rischi di Attacchi Cross-Site Scripting (XSS) e Phishing
-

# WORKSHOP NUMERO 1

## DIRECT FLOW



# DIRECT FLOW

Ora proviamo l'autenticazione di tipo Direct Access Grants.  
Questo significa che lavoriamo con il Grant-Type di tipo Password

### Capability config

Client authentication

?

Off

Authorization

?

Off

Authentication flow

☐

Standard flow

?

☒

Direct access grants

?

☐

Implicit flow

?

☐

Service accounts roles

?

☐

OAuth 2.0 Device Authorization Grant

?

☐

OIDC CIBA Grant

?



# DIRECT FLOW


Per provare questo flusso è sufficiente fare una chiamata POST direttamente a Keycloak, senza passare per la login di Keycloak



1. Creiamo una nuova chiamata.
2. Impostiamo il metodo POST
3. Incolliamo questo URL: `http://localhost:8090/realms/siru/protocol/openid-connect/token`
4. Impostiamo 'x-www-form-urlencoded' in body
5. Mettiamo i seguenti campi
  - a. `grant_type` = **password**
  - b. `client_id` = il client-id
  - c. **username** = username dell'utente
  - d. **password** = la password dell'utente






# DIRECT FLOW

Ecco un esempio:


 Keycloak-workshop / Direct access grants - Password Grant Type


 Save 

POST 

http://localhost:8090/realms/siru/protocol/openid-connect/token

Send 

Params Authorization Headers (8) Body  Pre-request Script Tests Settings Cookies

☐ none

☐ form-data

☒ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

	Key	Value	Description	... Bulk Edit
<input checked="" type="checkbox"/>	grant_type	password		
<input checked="" type="checkbox"/>	client_id	siru-fe		
<input checked="" type="checkbox"/>	username	michele		
<input checked="" type="checkbox"/>	password	admin		
	Key	Value	Description	



# DIRECT FLOW - CONSIDERAZIONI

Le principali differenze sono:

- il grant\_type è diverso
- non serve il campo 'code' e quindi non è necessario ottenere il code
- si mettono direttamente username e password

**Questo può sembrare un vantaggio in termini di sviluppo ma è una pessima pratica soprattutto in Web Application di tipo SPA (Single Page Application)**

## PRO

- Semplicità per le applicazioni di prima parte
- Esperienza Utente Semplificata
- Pochi passaggi

## CONTRO

- Sicurezza ridotta
- Non supporto completo a OAuth2
- Conformità e Sicurezza compromesse



# WORKSHOP NUMERO 1

## SERVICE ACCOUNTS ROLES



# SERVICE ACCOUNTS ROLES

Ora proviamo l'autenticazione di tipo Service Accounts Roles  
Questo significa che lavoriamo con il Grant-Type di tipo client\_credentials

### Capability config

Client authentication ⓘ

☒ On

Authorization ⓘ

☐ Off

Authentication flow ⓘ

☐ Standard flow ⓘ

☐ Direct access grants ⓘ

☐ Implicit flow ⓘ

☒ Service accounts roles ⓘ

☐ OAuth 2.0 Device Authorization Grant ⓘ

☐ OIDC CIBA Grant ⓘ



# SERVICE ACCOUNTS ROLES

Per provare questo flusso è sufficiente fare una chiamata POST direttamente a Keycloak, senza passare per la login di Keycloak

1. Creiamo una nuova chiamata.
2. Impostiamo il metodo POST
3. Incolliamo questo URL: `http://localhost:8090/realms/siru/protocol/openid-connect/token`
4. Impostiamo 'x-www-form-urlencoded' in body
5. Mettiamo i seguenti campi
  - a. `grant_type` = **client\_credentials**
  - b. `client_id` = il client-id
  - c. **username** = username dell'utente
  - d. **password** = la password dell'utente
  - e. `client_secret` = il secret che si trova nella configurazione keycloak



# SERVICE ACCOUNTS ROLES

Ecco un esempio:

Keycloak-workshop / Service accounts roles

POST

http://localhost:8090/realms/siru/protocol/openid-connect/token

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

☐ none

☐ form-data

☒ x-www-form-urlencoded

☐ raw

☐ binary

☐ GraphQL

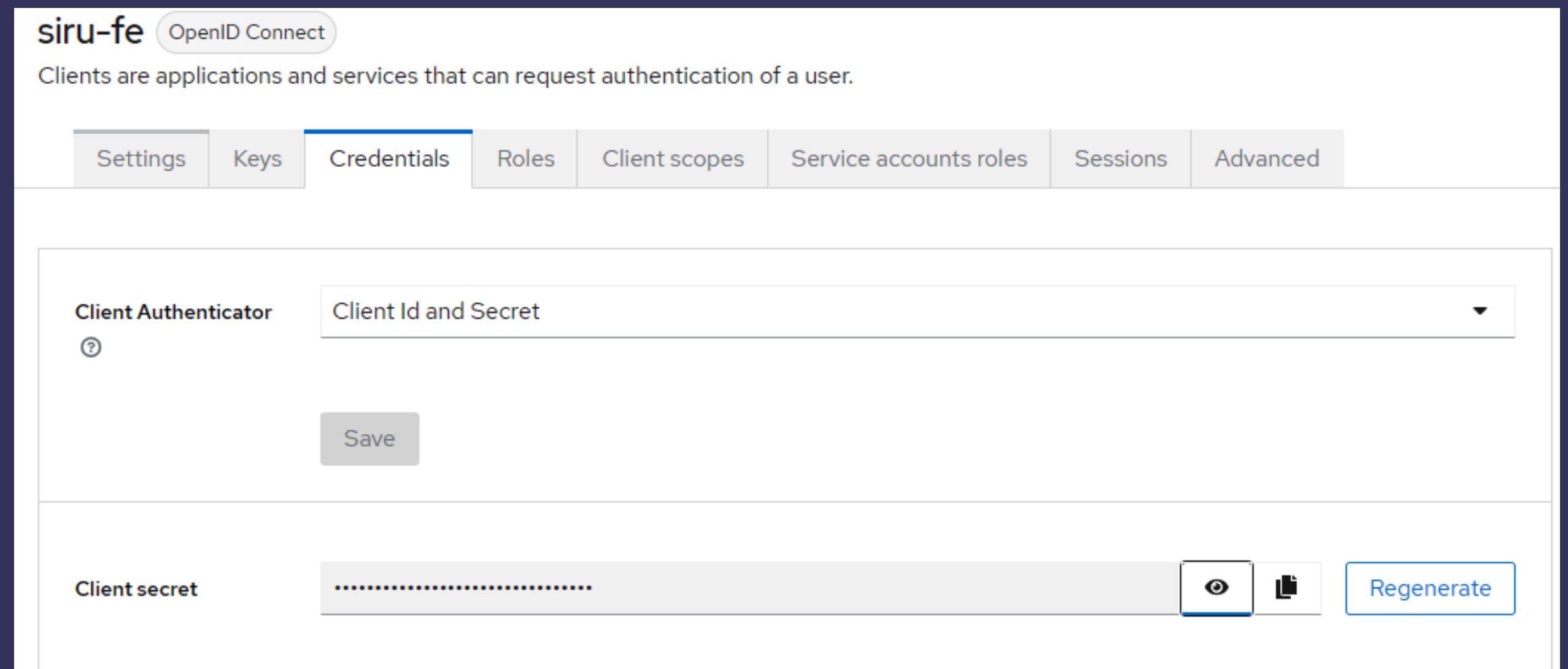
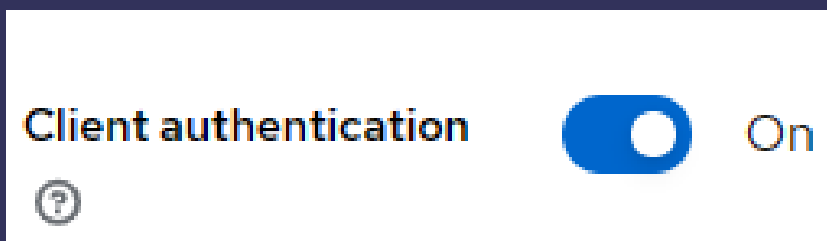
	Key	Value	Description	...	Bulk Edit
<input checked="" type="checkbox"/>	grant_type	client_credentials			
<input checked="" type="checkbox"/>	client_id	siru-fe			
<input checked="" type="checkbox"/>	username	michele			
<input checked="" type="checkbox"/>	password	admin			
<input checked="" type="checkbox"/>	client_secret	LUGudq80cZB3zA7LuZbEtGsT5qu8TyoH			
<input checked="" type="checkbox"/>	scope	openid profile email			
	Key	Value	Description		



# SERVICE ACCOUNTS ROLES

Per ottenere il client secret bisogna:

- Mettere a ON il Client Authentication
- Andare nella tab 'Credentials'
- Recuperare il client secret





# SERVICE ACCOUNTS ROLES CONSIDERAZIONI

Questa tipologia è spesso usata per accedere a risorse in modo automatico senza richiedere l'interazione diretta con un utente

Non avendo visto in modo approfondito la questione dei ruoli è ancora presto per trarre delle conclusioni.



# SERVICE ACCOUNTS ROLES RICAPITOLANDO

Grant Type  
Authorization Code

Grant Type  
Password

Authentication flow	<input checked="" type="checkbox"/> Standard flow ?	<input checked="" type="checkbox"/> Direct access grants ?
	<input checked="" type="checkbox"/> Implicit flow ?	<input checked="" type="checkbox"/> Service accounts roles ?

Grant Type  
Implicit Flow

Grant Type  
Client Credentials



Vi ringraziamo per l'attenzione e per  
ogni domanda siamo qui per  
rispondervi!

