***Keycloak***

1. ***Creare imagine si lansare in executie imagine:***

docker run -p 8080:8080 -e KEYCLOAK\_ADMIN=admin -e KEYCLOAK\_ADMIN\_PASSWORD=admin quay.io/keycloak/keycloak:23.0.1 start-dev

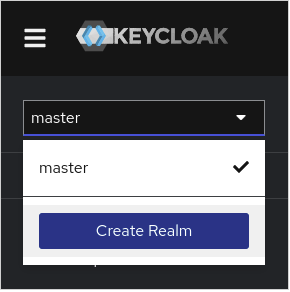
This command starts Keycloak exposed on the local port 8080 and creates an initial admin user with the username admin and password admin.

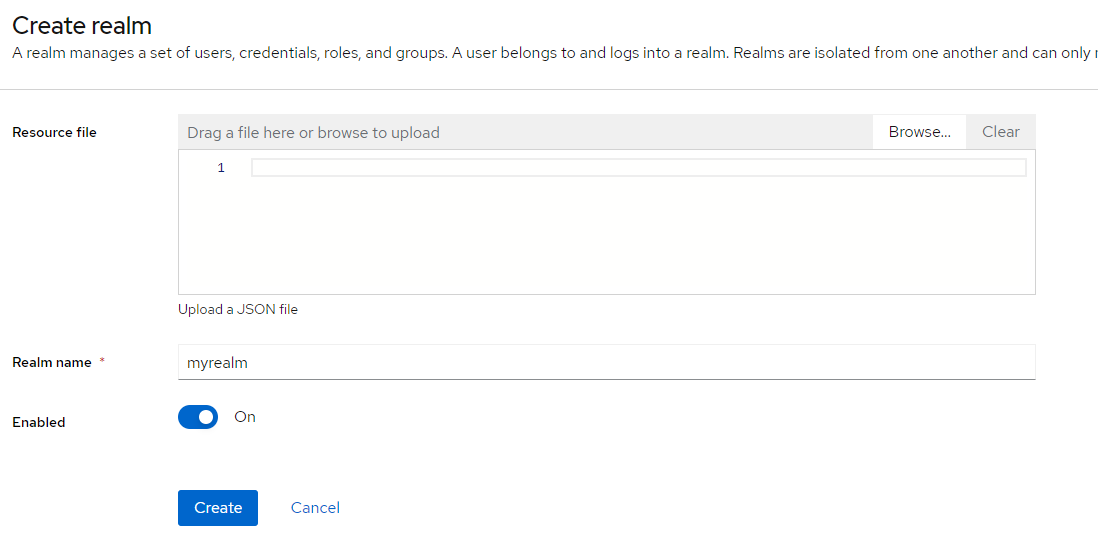
1. ***Accesam din browser dupa:*** <http://localhost:8080/admin/master/console/>
2. ***Ne conectam cu username-ul si parola setate anterior***
3. ***Create a realm:***

* Intr-un realm se aplica anumite politici de Securitate si gestionare a utilizatorilor
* Poate fi vazut ca un “spatiu” sau un “domeniu” care contine un set de utilizatori, roluri, grupuri si politici de Securitate, gestionand modul in care utilizatorii sunt autentificati si autorizati.
* Un realm separa un grup de utilizatori si resursele associate acestora de alte grupuri dintr-un sistem mai mare.
* Fiecare realm poate avea propriile sale setari pentru autentificare, autorizare, politici de Securitate si alte aspecte legate de gestionarea identitatii si accesului.
* De exemplu, intr-o organizatie mare poate avea realm-uri pentru diferite departamente, fiecare cu politicile sale de Securitate si setari de autentificare.
* Initial, Keycloak include un singur realm, numit master. Acest realm ar trebui folosit doar pentru managerierea Keycloak si nu pentru managerierea aplicatiilor. Master => administare Keycloak.
* Realm-ul master este utilizat pentru a gestiona toate aspectele serverului Keycloak, inclusiv crearea și configurarea altor realme, gestionarea utilizatorilor și rolurilor la nivel de server, și configurarea setărilor globale.
* În realm-ul master se configurează extensii sau provideri la nivel de server, cum ar fi provideri pentru autentificare, user federation etc.
* Realm-ul master permite administrarea setărilor globale legate de chei și certificate, care sunt critice pentru securitatea comunicațiilor și autentificării.

***Pentru a crea un alt realm:***

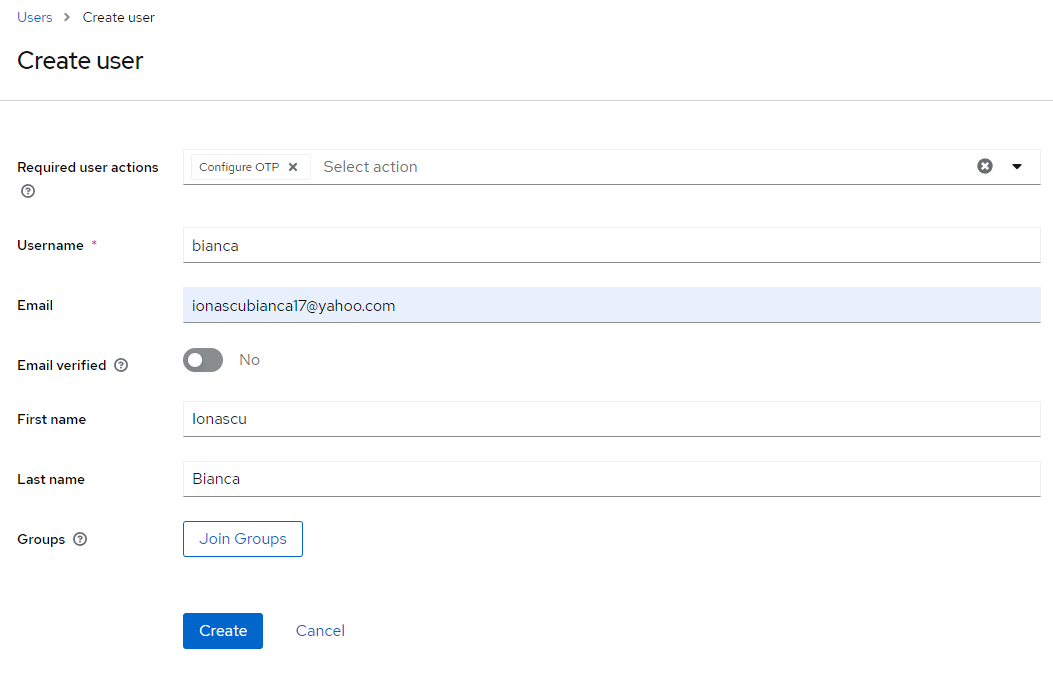
1. Din Admin Console o sa selectam Create Realm:





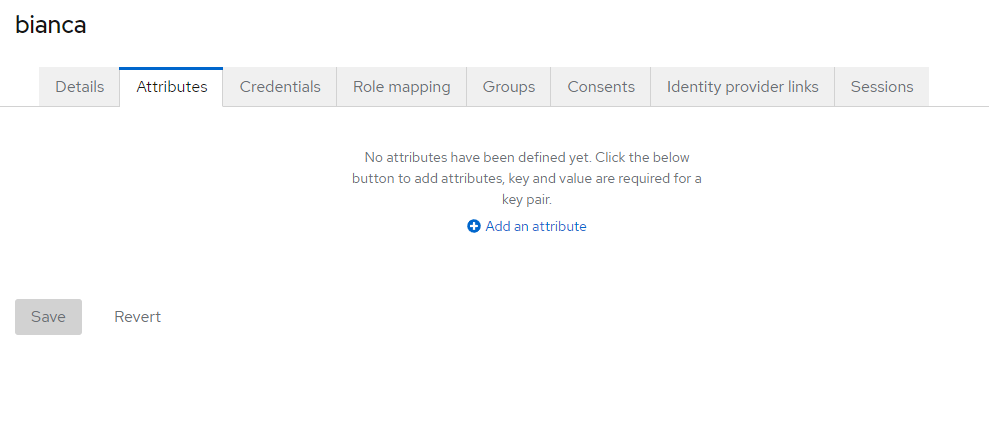
***Pentru a crea un user nou:***

Initial un realm nu are niciun user. Pentru a crea un nou user facem asa:

1. Ne mutam pe myrealm.
2. Accesam din bara din stanga Users si dupa Add user.
3. ***Required user actions:*** Acest câmp este folosit pentru a specifica anumite acțiuni pe care noul utilizator trebuie să le efectueze obligatoriu după prima autentificare. De exemplu, se poate cere utilizatorului să verifice adresa de e-mail, să schimbe parola, sau să configureze autentificarea cu doi factori. Aceste acțiuni sunt destinate să ajute la finalizarea profilului utilizatorului sau la îmbunătățirea securității contului.

Aici la required actions ar tebui utilizatorul meu sa isi aleaga al doilea factor pe care sa il foloseasca si sa isi configureze modul in care acesta le va folosi si le va sincroniza cu aplicatia mobila.

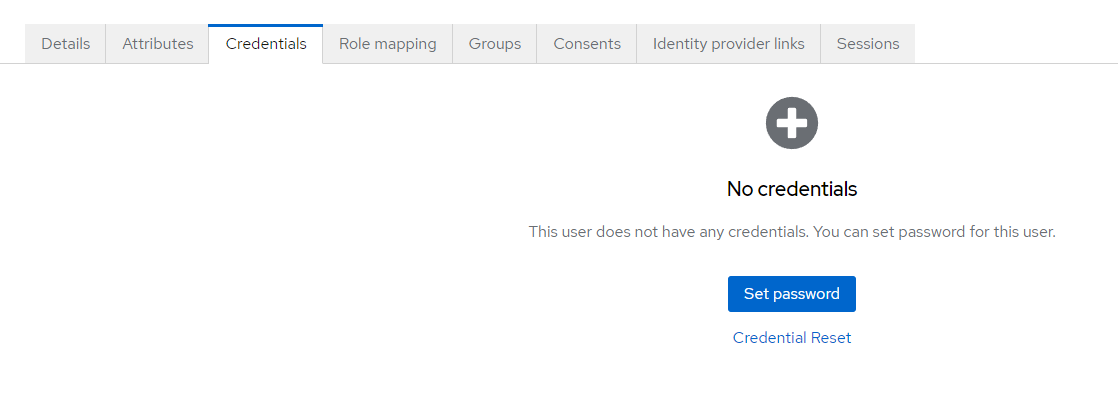
La User Details regasim:

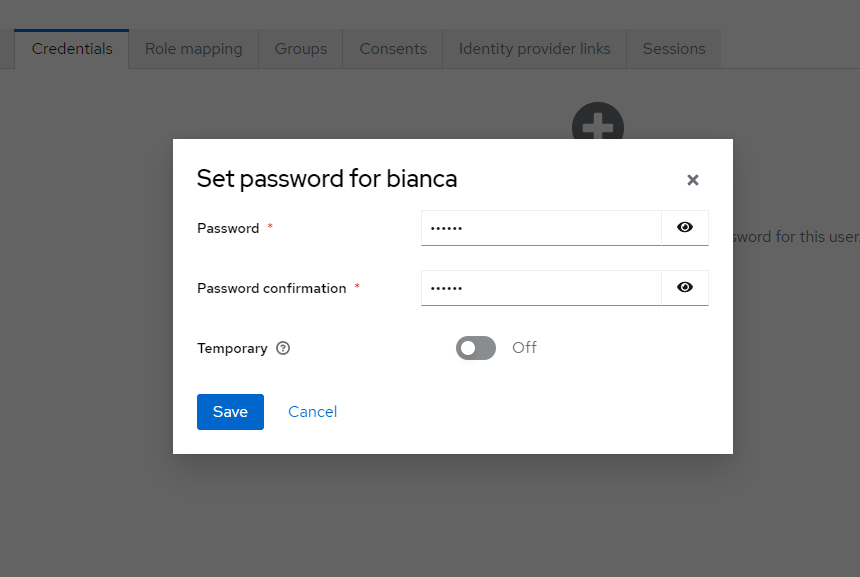
AAc

Aceasta sectiune se refera la atributele personalizate pe care le putem atribui unui utilizator. Atributele sunt perechi de cheie-valoare care pot stoca informatii suplimentare despre utilizator care nu sunt acoperite de campurile standard oferite de keycloak.

De exemplu, atributele pot include detalii cum ar fi:

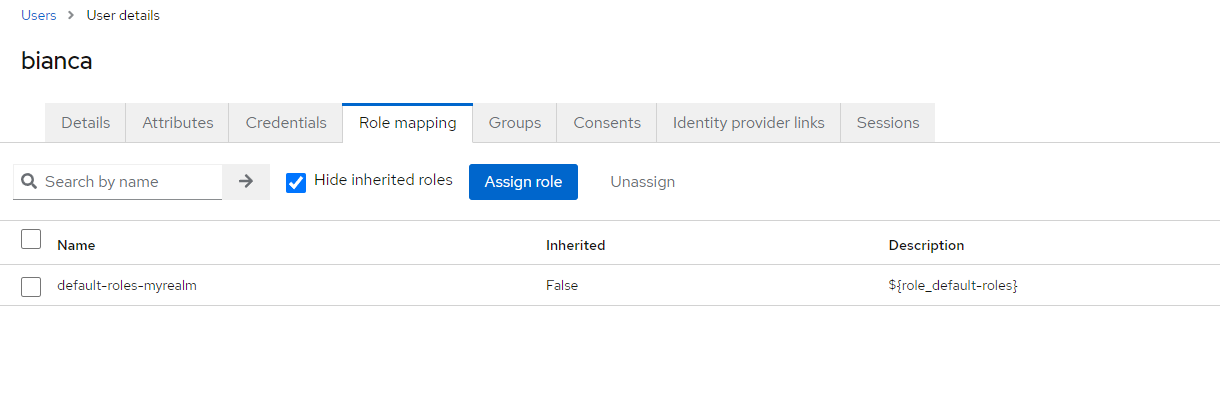
* Numărul de telefon
* Adresa de domiciliu sau de birou
* ID-ul angajatului
* Preferințe de limbă
* Date de expirare a contului
* Orice alt tip de metadate personalizate relevante pentru aplicație sau pentru procesele de business

User-ul are nevoie de o parola pentru a se loga.

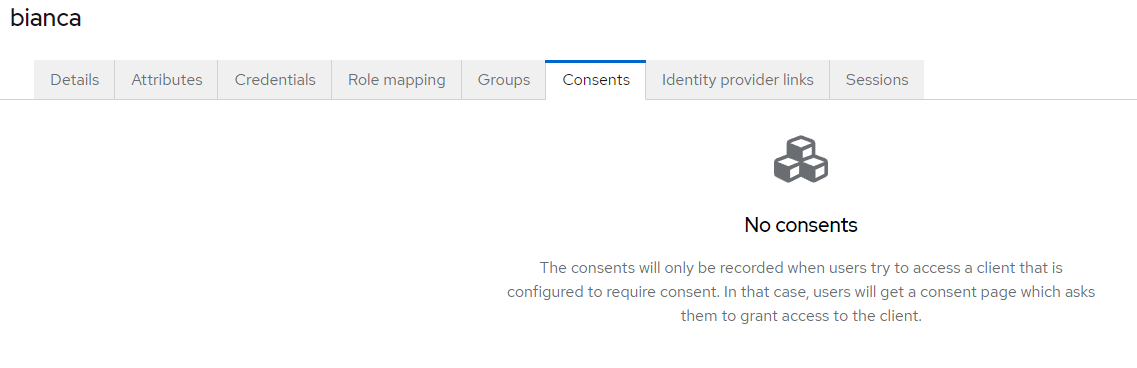


Parola: parola

Toggle **Temporary** to **Off** so that the user does not need to update this password at the first login.



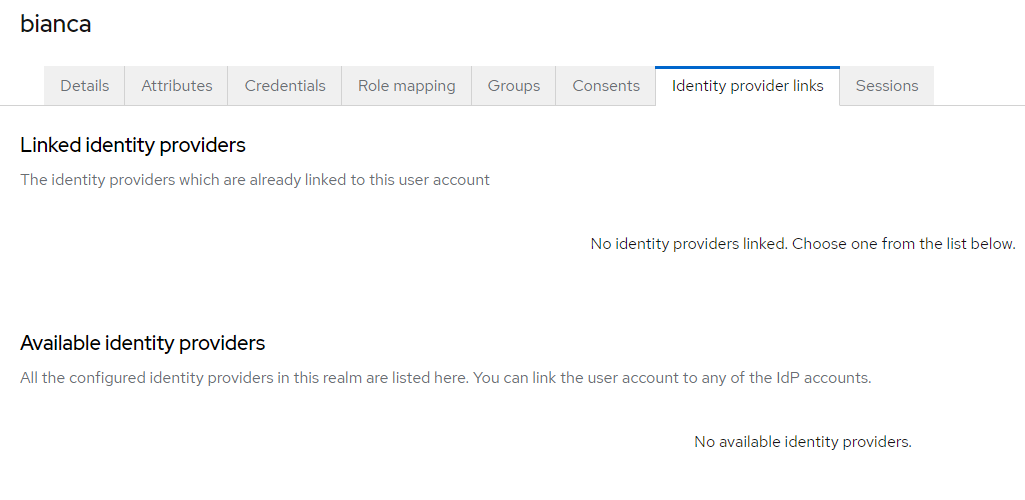
* se referă la procesul de atribuire a rolurilor unui utilizator. Rolurile sunt folosite pentru a defini permisiunile și accesul pe care un utilizator le are într-o aplicație sau într-un set de aplicații
* **Name:** Numele rolului atribuit utilizatorului. În exemplul dat, "default-roles-myrealm" indică un set de roluri implicite atribuite pentru realm-ul "myrealm".
* **Inherited:** Acest câmp indică dacă rolul a fost moștenit de la un grup sau dintr-o altă sursă. "False" înseamnă că acest rol a fost atribuit direct utilizatorului și nu este moștenit.
* **Description:** O descriere a rolului, care poate oferi mai multe detalii despre ce implică rolul. Se poate referi la un șablon sau la o variabilă care va fi înlocuită cu descrierea reală într-un context dinamic.
* **"Hide inherited roles"** este o opțiune care, când este activată, ascunde rolurile care nu sunt atribuite direct utilizatorului, ci sunt moștenite, de exemplu, prin apartenența la un grup.



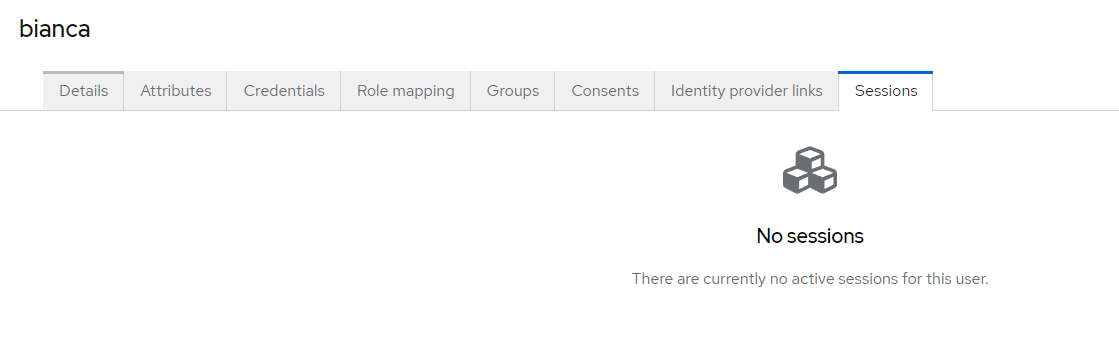
În imaginea prezentată din interfața Keycloak, secțiunea "Consents" se referă la consimțământurile acordate de utilizatori pentru diferite aplicații (cunoscute în Keycloak ca "clienți"). Consimțământul în acest context este permisiunea pe care un utilizator o dă pentru a permite unei aplicații să acceseze informațiile sale personale sau să efectueze acțiuni în numele său.

Keycloak permite configurarea aplicațiilor pentru a cere consimțământul utilizatorului pentru accesarea datelor lor. Când un utilizator se loghează într-o astfel de aplicație pentru prima dată, Keycloak prezintă o pagină de consimțământ care îi informează pe utilizatori despre informațiile și privilegiile pe care aplicația dorește să le acceseze și le cere să aprobe sau să respingă aceste cereri.

Vezi oauth2.0 userAgent si client.

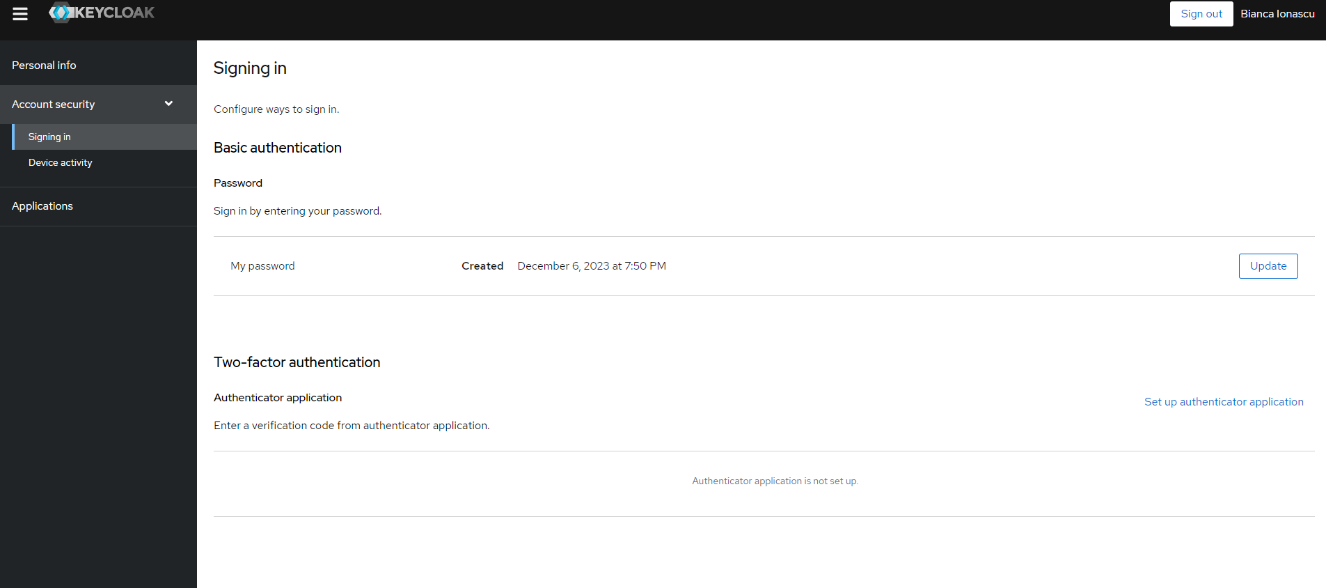


* În această secțiune a interfeței de utilizator Keycloak, "Identity provider links", este locul unde puteți gestiona legăturile dintre conturile utilizatorilor și furnizorii de identitate externi (IdP). Aceste legături sunt folosite pentru a permite utilizatorilor să se autentifice la Keycloak folosind credențiale de la diferiți furnizori de identitate, cum ar fi Google, Facebook, sau un sistem SAML sau OpenID Connect.
* *Pentru a permite autentificarea utilizatorului prin furnizori de identitate externi, un administrator ar trebui să configureze mai întâi unul sau mai mulți furnizori de identitate în cadrul realm-ului Keycloak. Odată ce aceștia sunt configurați, ei vor apărea în secțiunea "Available identity providers", și utilizatorii pot lega contul lor Keycloak de acești IdP pentru a utiliza funcția de Single Sign-On (SSO).*

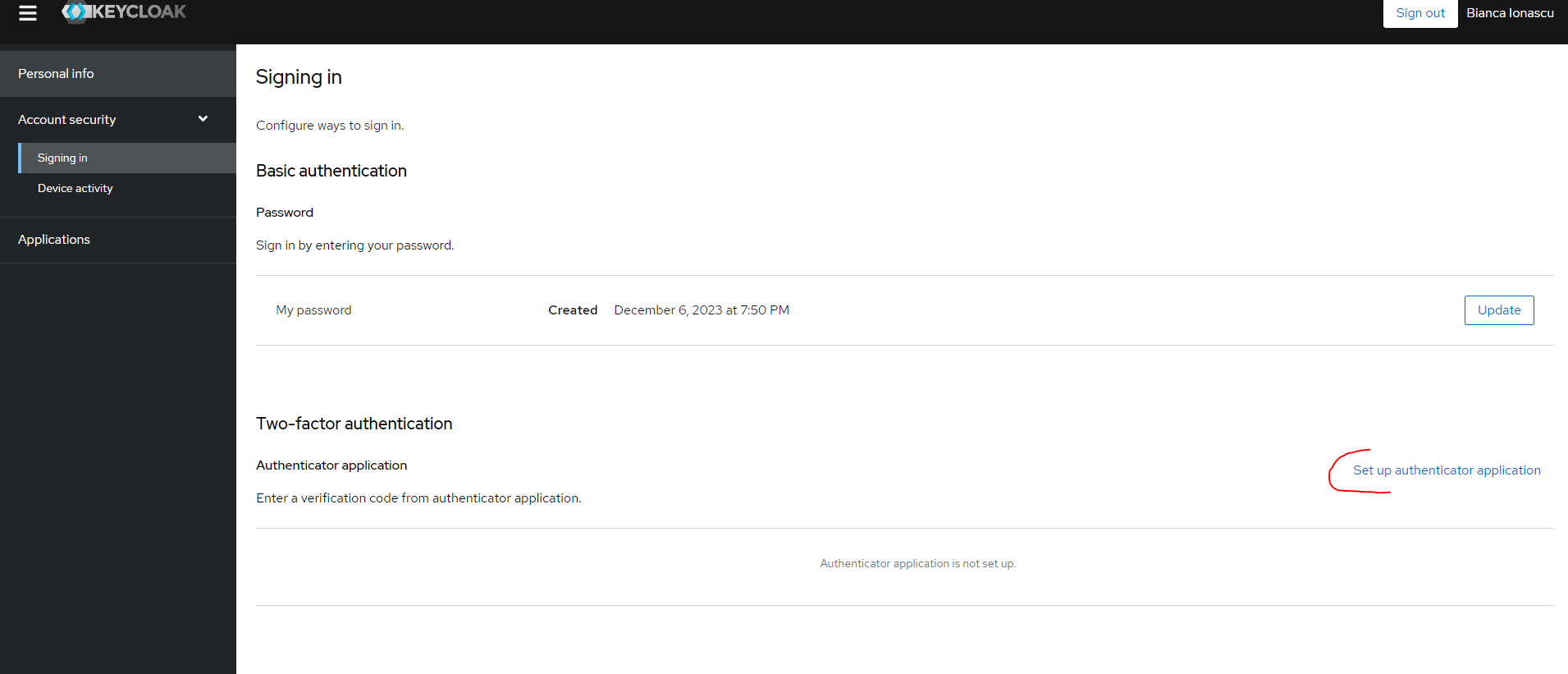


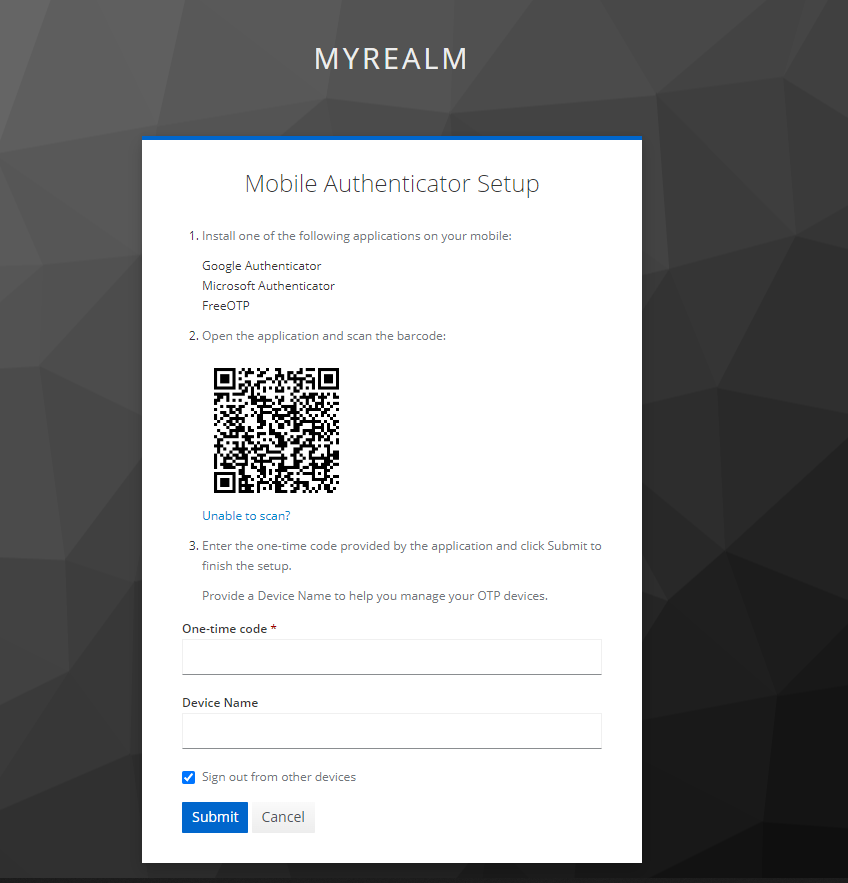
* În interfața Keycloak, tab-ul "Sessions" este destinat să ofere informații despre sesiunile active ale utilizatorului. Aceasta include sesiunile curente deschise când un utilizator este logat în aplicațiile securizate de Keycloak.

Pentru a ma conecta cu noul user creat am accesat acest link: <http://localhost:8080/realms/myrealm/account/#/>



As a user in the Account Console, you can manage your account including modifying your profile, adding two-factor authentication, and including identity provider accounts.



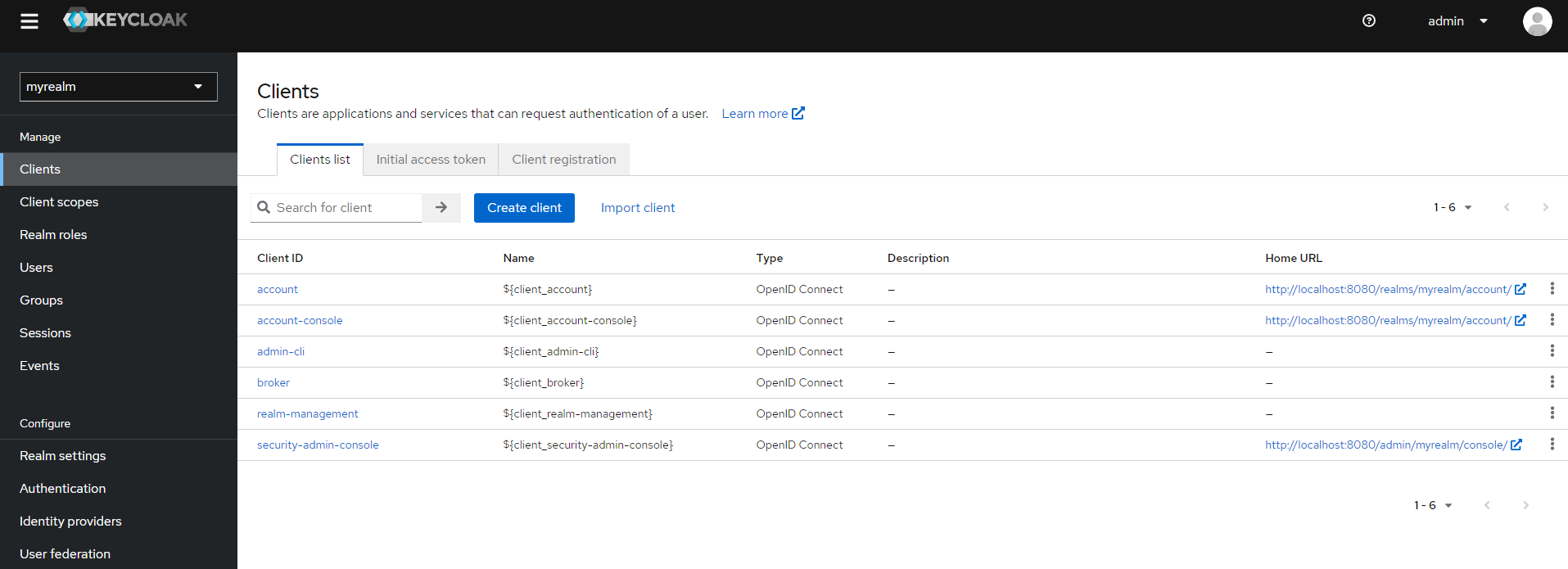


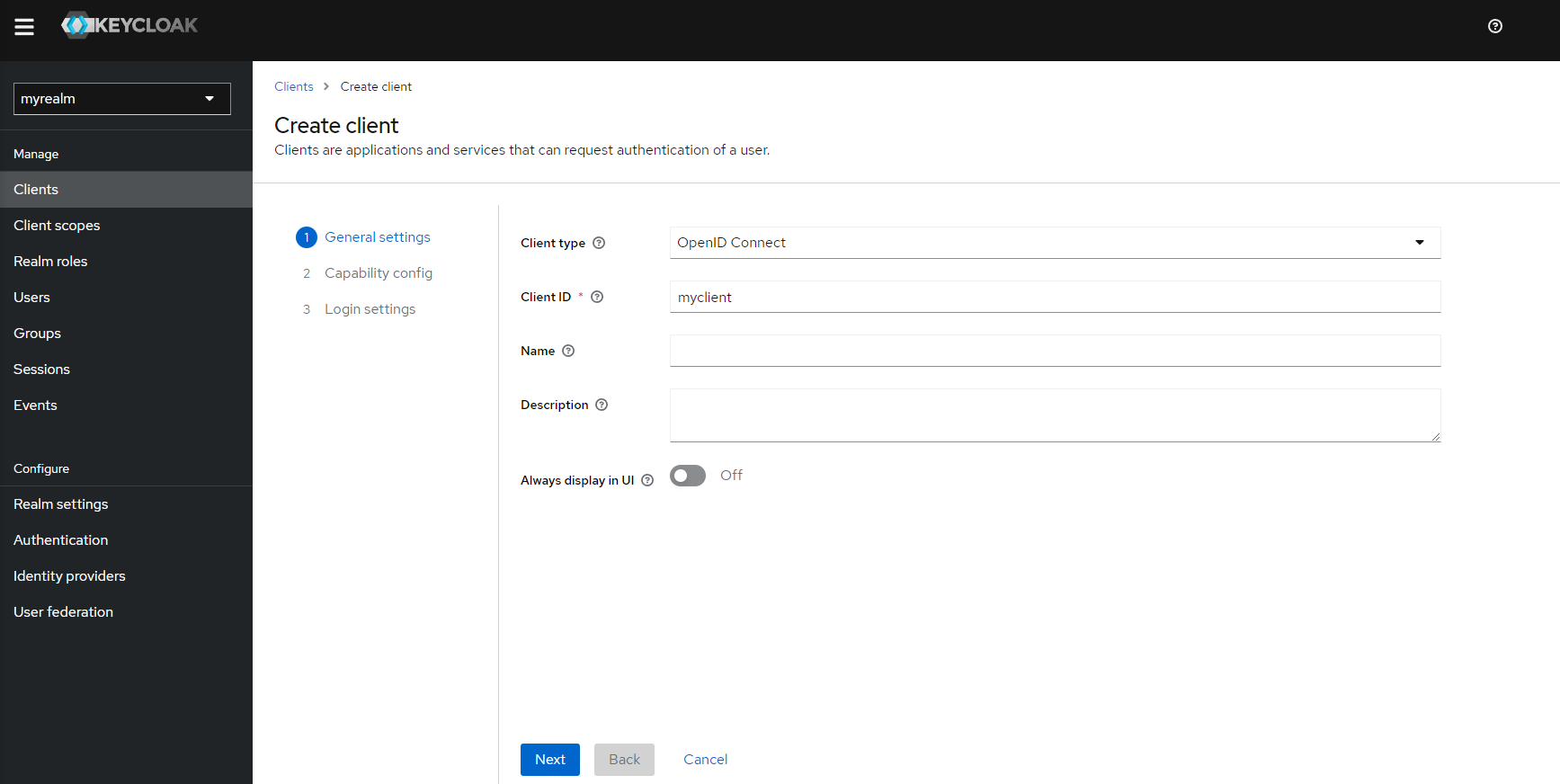
Aici imi inregistrez aplicatia mobila.

***Secure the first application***

1. Mai intai trebuie sa cream un client din Admin Console:

Alegem myrealm -> Clients -> Create client

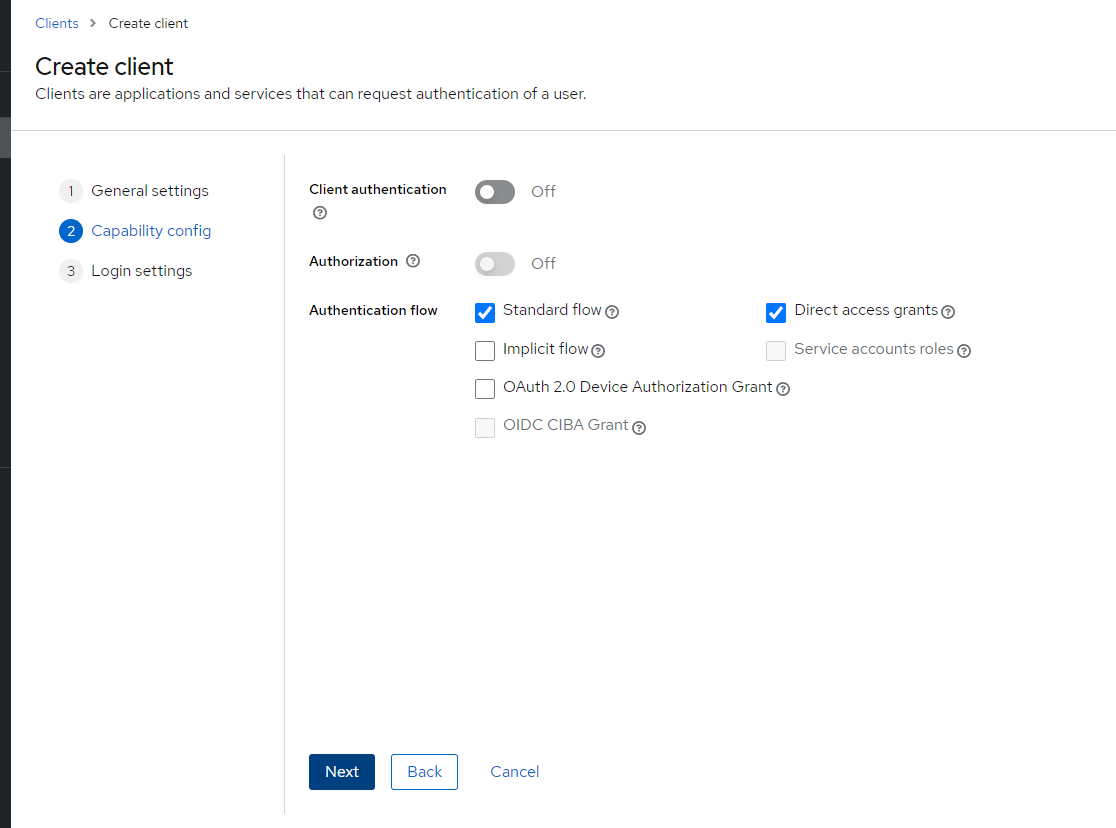




Cream un nou client in myrealm. Client = aplicatie, serviciu care va solicita autentificarea utilizatorului.

Campuri:

1. **Client type:** Tipul de client pe care îl configurați, care poate fi "OpenID Connect" (un protocol de autentificare bazat pe OAuth 2.0), "SAML" sau altele în funcție de nevoile dvs.
2. **Client ID**: Un identificator unic pentru clientul dvs., care va fi folosit în cadrul interacțiunilor cu Keycloak și aplicația client.
3. **Name:** Numele prietenos al clientului, care poate fi afișat utilizatorilor sau folosit în loguri.
4. **Description:** O descriere opțională a clientului, pentru a ajuta la identificarea scopului sau utilizării acestuia.
5. **Always display in UI:** Dacă este setat pe "On", clientul va fi afișat întotdeauna în interfața utilizatorului Keycloak



* În această secțiune, puteți configura diferite opțiuni care vor determina modul în care clientul va interacționa cu Keycloak pentru autentificarea și autorizarea utilizatorilor.
* **Client authentication:** Atunci când este activată, această opțiune necesită ca, clientul să se autentifice la Keycloak folosind un secret client sau o cheie publică și privată pentru a obține un token.
* **Authorization:** Dacă este activată, aceasta permite utilizarea serviciilor de autorizare Keycloak, unde Keycloak va gestiona nu doar autentificarea utilizatorilor, ci și ce resurse pot accesa acești utilizatori.
* **Authentication flow:**

--------- **Standard flow:** Este fluxul tradițional de autentificare în care utilizatorul este redirecționat la serverul de autentificare (Keycloak) pentru a se loga și este apoi redirecționat înapoi la aplicația client cu un cod de autorizare, care este schimbat pentru un token. => code authorization

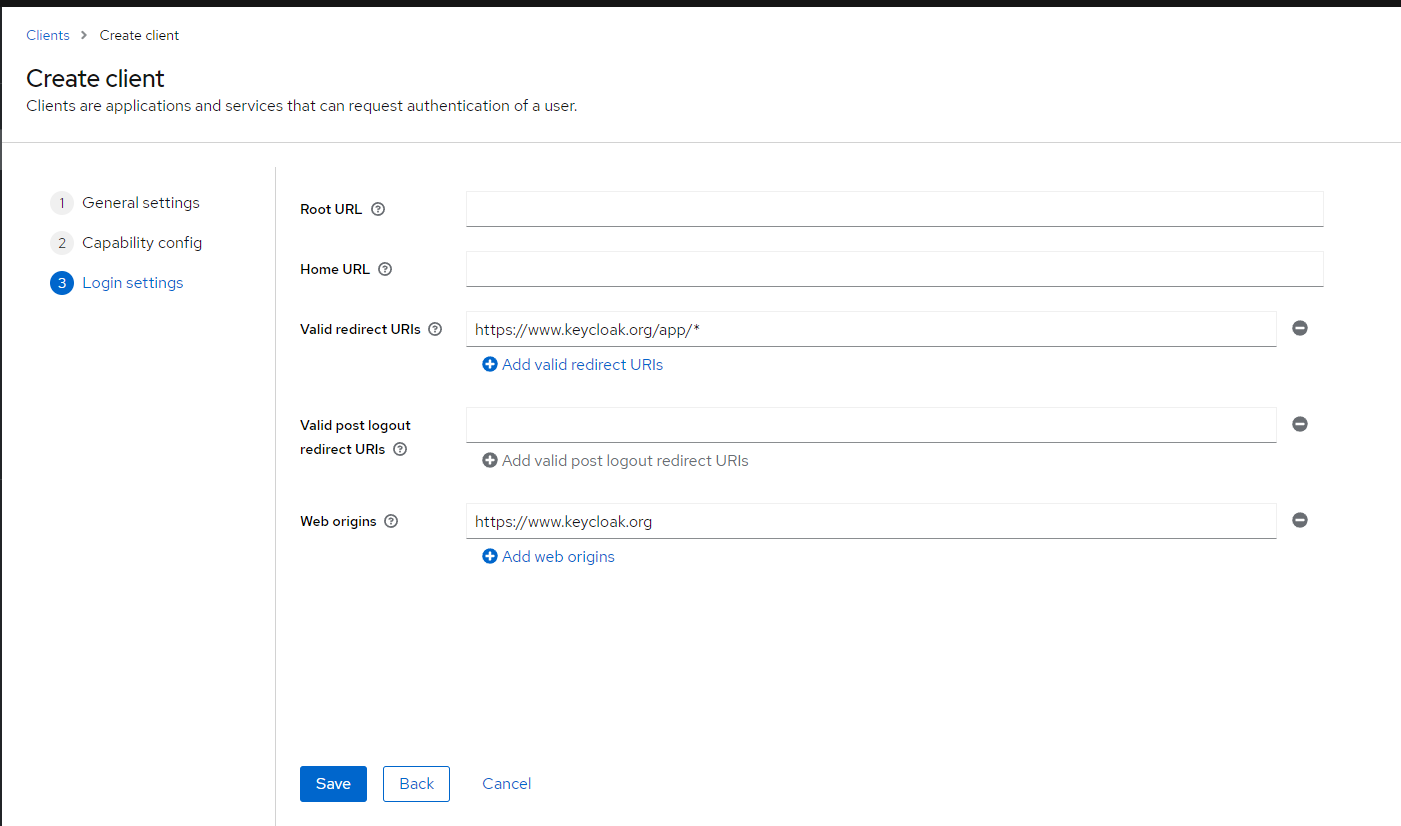
--------- **Direct access grants:** Permite aplicației client să trimită direct credențialele utilizatorului la Keycloak pentru a obține un token, fără să redirecționeze utilizatorul.

---------- **Implicit flow:** Un tip de flux OAuth 2.0, acum mai puțin folosit, care oferă acces direct la token-uri fără un cod intermediar de autorizare.

--------- **Service accounts roles:** Permite clientului să acceseze Keycloak cu un cont de serviciu propriu, care are propriile roluri asociate pentru accesul la API-uri.

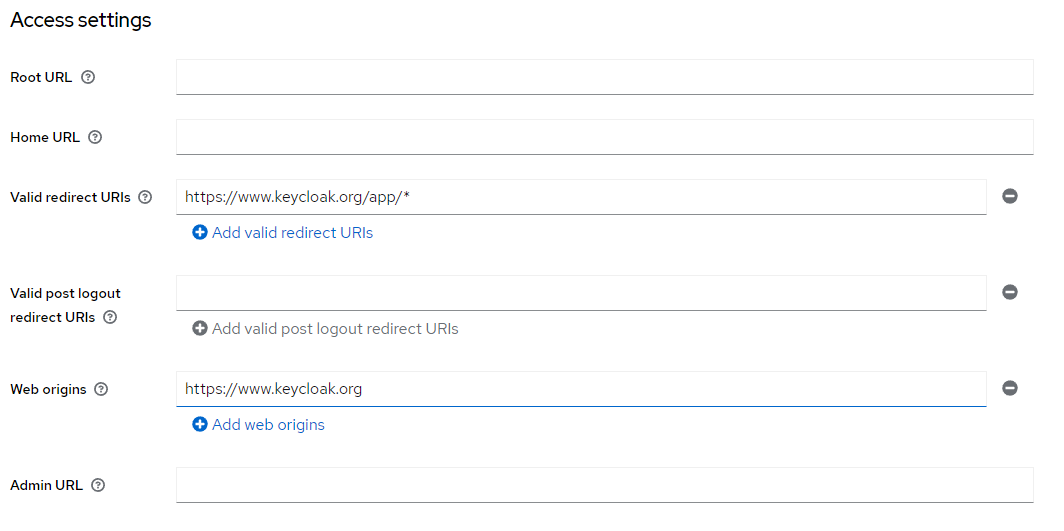
**-------- OAuth 2.0 Device Authorization Grant:** Un flux pentru dispozitivele care nu au capacitatea de a afișa un browser web, permițându-le să se autentifice într-un mod diferit.

**-------- OIDC CIBA Grant:** Client Initiated Backchannel Authentication este un flux de autentificare în care autentificarea utilizatorului se desfășoară pe un dispozitiv diferit de cel pe care se află clientul (de exemplu, autentificarea pe telefon pentru o aplicație pe TV).



Aici se configureaza detaliile legate de cum și unde se va redirecționa utilizatorul în timpul și după procesul de autentificare.

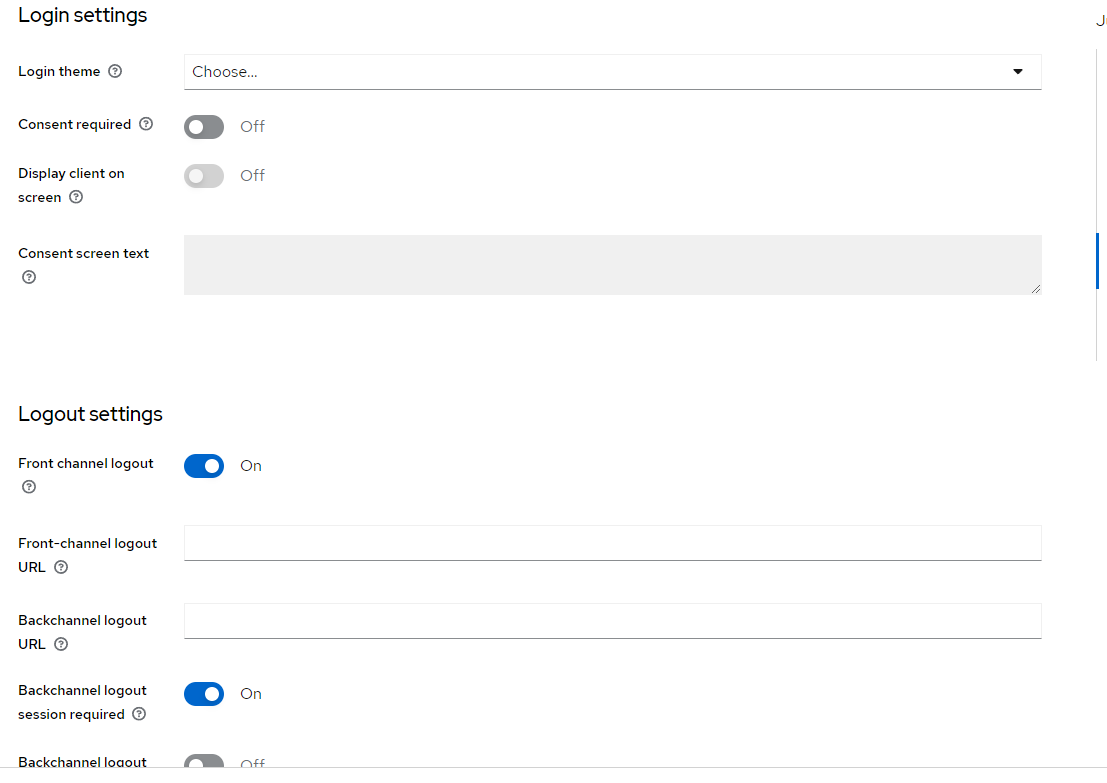
* **Root URL:** Baza URL-ului pe care clientul îl va folosi pentru construirea URL-urilor de redirecționare și post logout. Keycloak poate completa automat URL-urile de redirecționare și post logout dacă sunt furnizate într-un format relativ.
* **Home URL:** URL-ul unde utilizatorul poate fi redirecționat după autentificare, care este de obicei pagina principală a aplicației client.
* **Valid redirect URIs:** Lista de URI-uri la care Keycloak poate redirecționa utilizatorii după ce s-au autentificat cu succes. Acest lucru este folosit pentru a preveni atacurile de tip open redirector prin asigurarea că redirecționarea se face doar către locații de încredere. Wildcard-urile pot fi folosite pentru a specifica un set de URI-uri, cum ar fi în exemplul https://www.keycloak.org/app/\*.
* **Valid post logout redirect URIs:** Similar cu redirect URIs, acestea sunt locurile la care utilizatorul poate fi redirecționat după ce s-a deconectat de la aplicație.
* **Web origins:** Aceasta definește originile care sunt permise să facă cereri cross-origin către server folosind CORS (Cross-Origin Resource Sharing). Acest lucru este necesar pentru aplicațiile web care accesează Keycloak dintr-o pagină web prin AJAX.



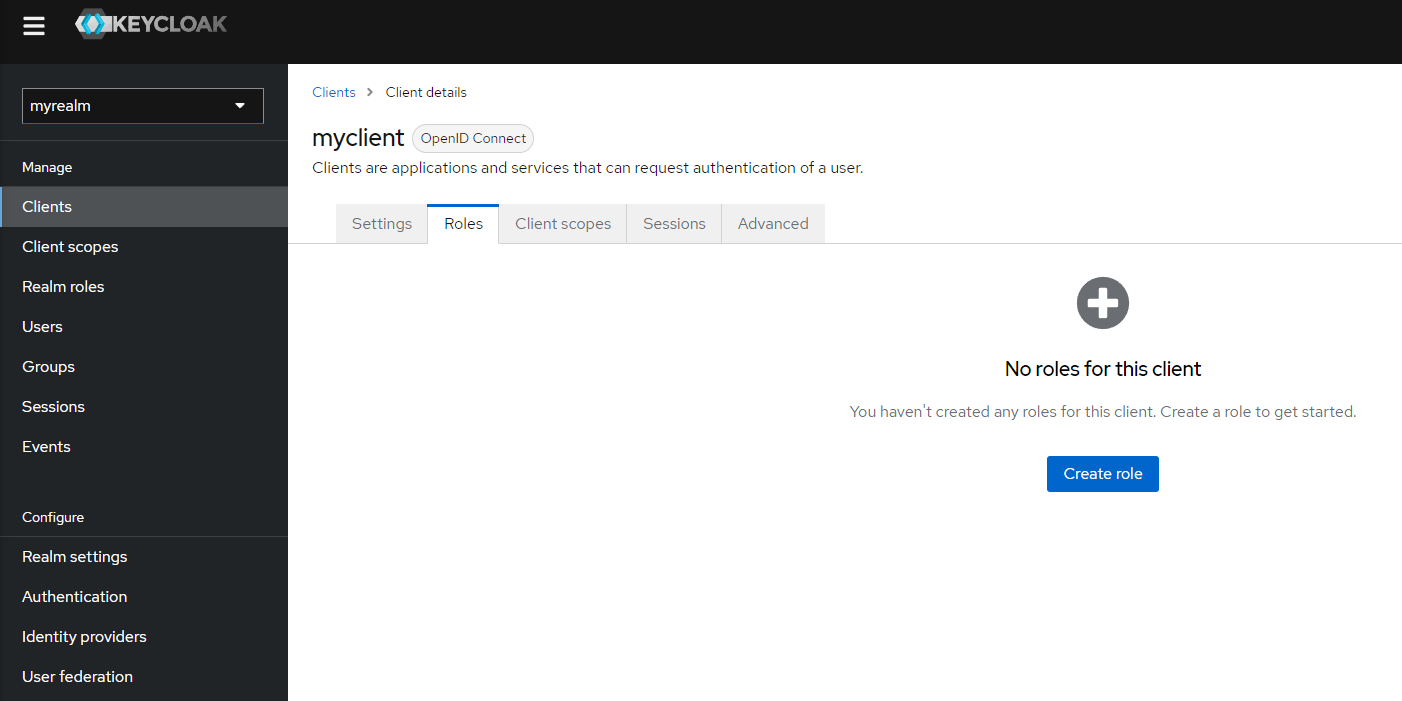
* În Keycloak, câmpul "Admin URL" este utilizat pentru a specifica URL-ul la care clientul poate primi cereri administrative din partea serverului Keycloak. Acest URL este folosit pentru operațiuni specifice de administrare și management ale clientului, cum ar fi:

1. Înregistrarea și de-înregistrarea instanțelor client atunci când acestea sunt deschise sau închise.
2. Push-ul actualizărilor de configurație către client, cum ar fi actualizarea rolurilor sau a altor metadate asociate cu clientul.
3. Trimiterea de cereri pentru invalidarea cache-urilor sau pentru a actualiza configurația la nivel de client.

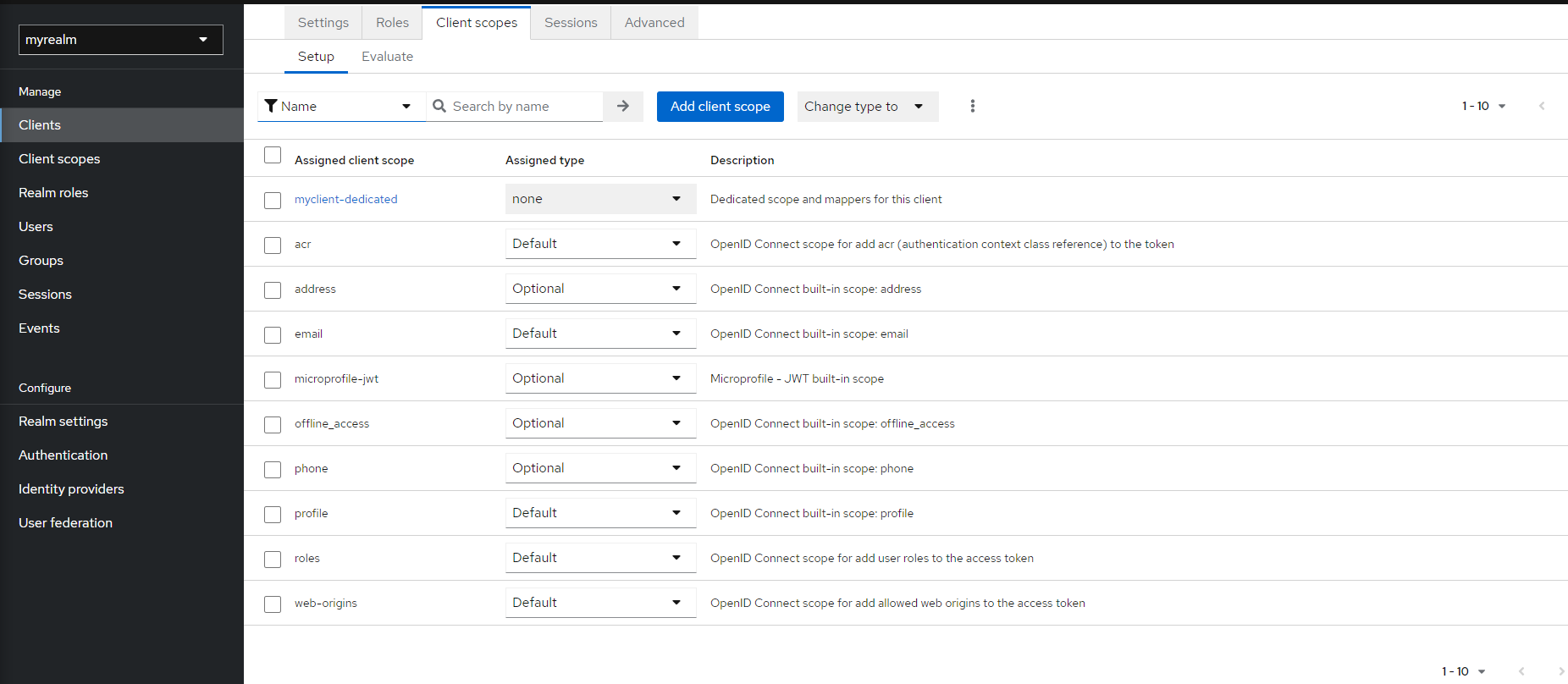
* URL-ul admin este, în general, un punct de sfârșit pe backend-ul aplicației client care este pregătit să gestioneze astfel de cereri venite de la serverul Keycloak. Acesta este folosit în scenarii unde aplicațiile client au nevoie să fie sincronizate în timp real cu schimbările de stare sau configurație de la serverul de identitate.



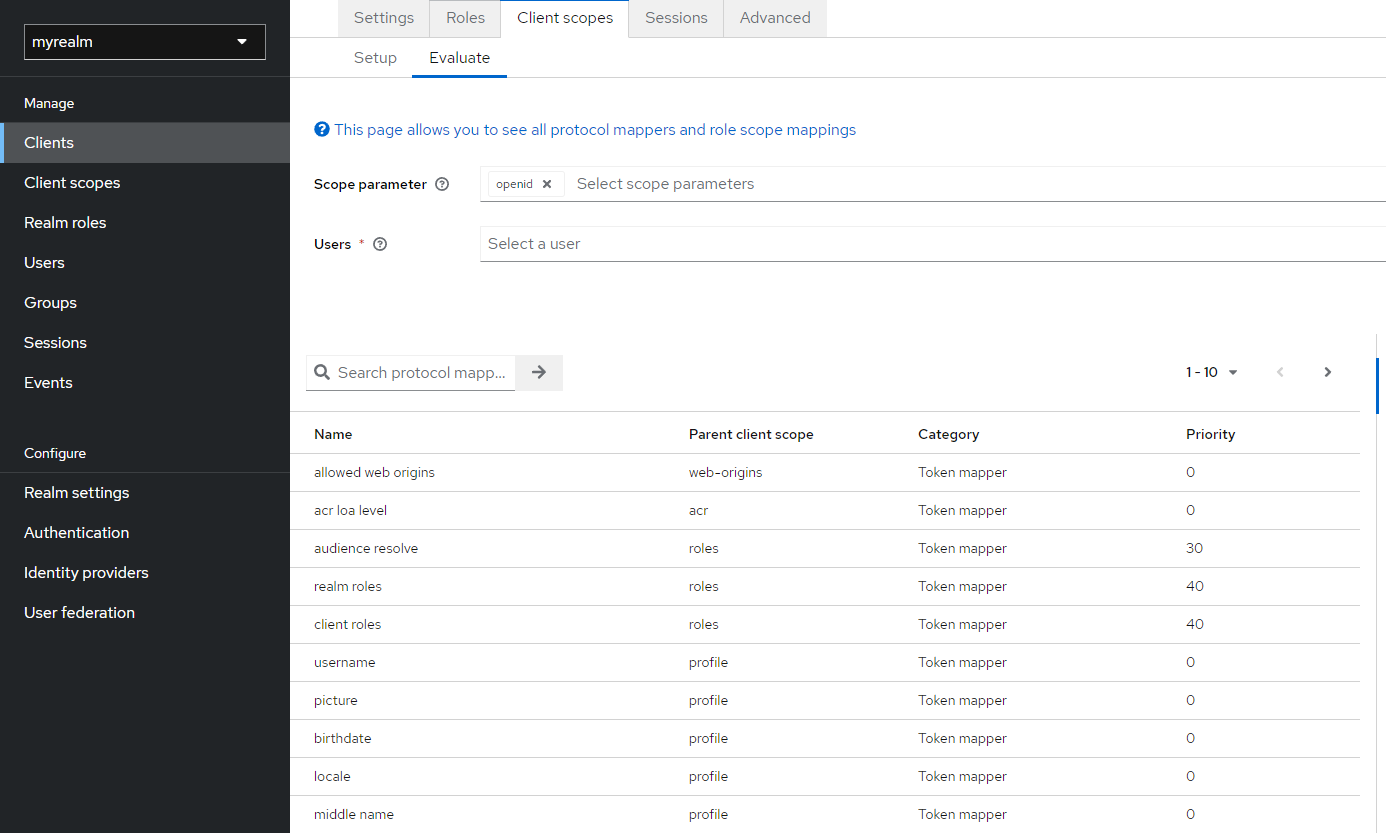
* **Consent required:** Când este activată, utilizatorii vor trebui să acorde consimțământ pentru acțiuni specifice, cum ar fi partajarea informațiilor lor cu clientul.
* **Display client on screen:** Dacă este activată, numele clientului va fi afișat pe ecranul de autentificare.
* **Consent screen text:** Textul personalizat care poate fi afișat pe ecranul de consimțământ atunci când utilizatorii trebuie să acorde permisiuni.
* **Front channel logout:** Când este activată, va fi trimis un semnal de logout la browserul utilizatorului pentru a-l deconecta de la client.
* **Front-channel logout URL:** URL-ul la care clientul va trimite cererea de logout prin browserul utilizatorului (front channel).
* **Backchannel logout URL:** URL-ul la care Keycloak va trimite cererea de logout direct la client fără a trece prin browserul utilizatorului, utilizat pentru a deconecta sesiunile server-side.
* **Backchannel logout session required:** Când este activată, specifică dacă pentru logout-ul prin backchannel este necesară o sesiune.



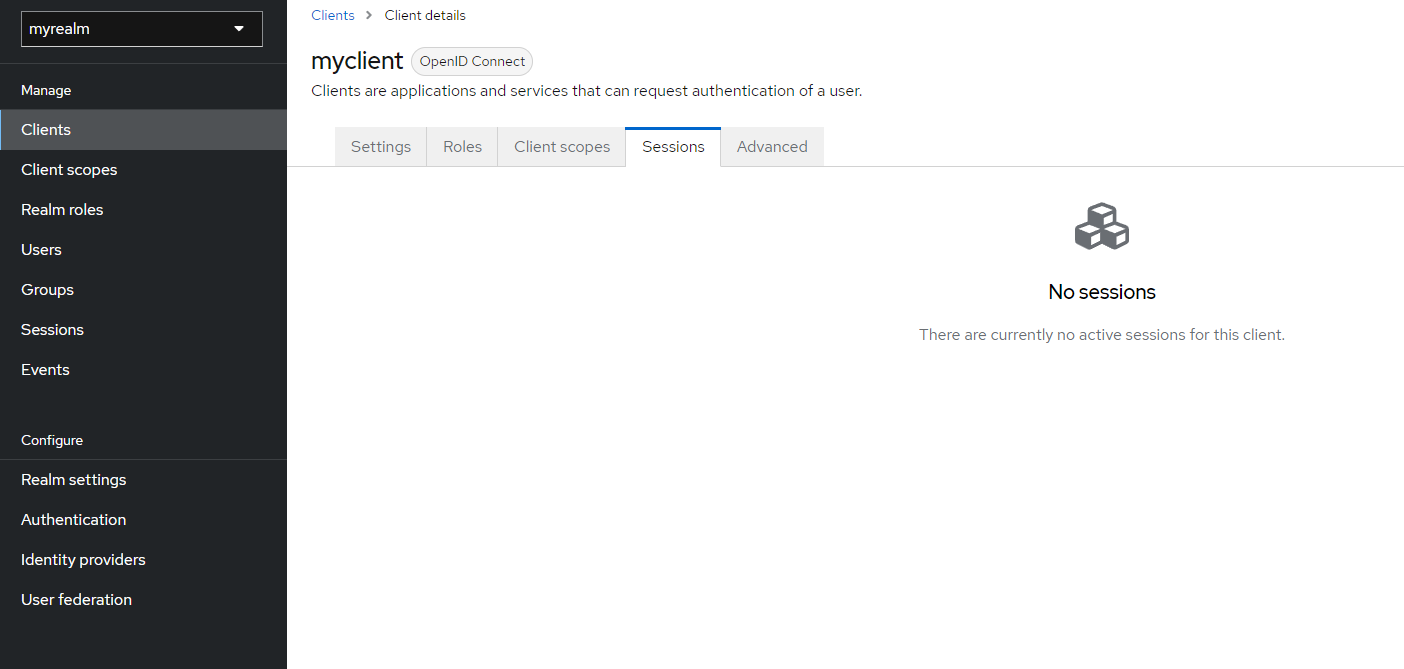
* Aici, este indicat că pentru clientul curent, denumit "myclient", nu au fost create încă roluri. În Keycloak, rolurile sunt utilizate pentru a defini permisiuni specifice pentru utilizatori sau pentru entități care accesează aplicația.
* De exemplu, puteți crea un rol numit "admin" pentru a permite accesul la panoul de administrare al unei aplicații sau un rol "user" pentru accesul la funcționalitățile de bază ale aplicației.



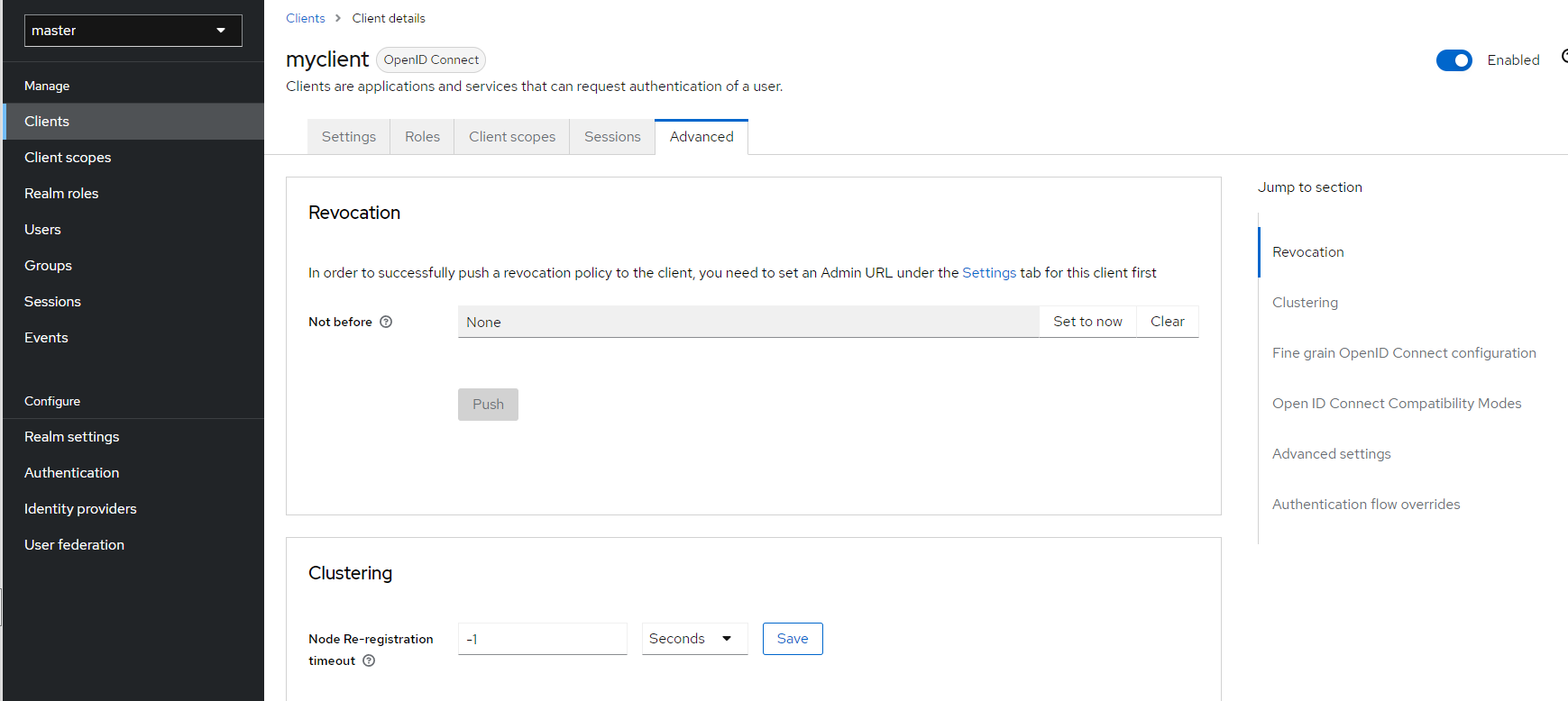
* Scopurile clientului (client scopes) sunt utilizate pentru a defini ce informații despre utilizatori sunt împărtășite cu aplicații client când un token de acces sau un token ID este emis de Keycloak
* **Name:** Numele scopului clientului, care poate fi folosit pentru a identifica setul de claims sau permisiuni care vor fi incluse în token.
* **Assigned type:** Tipul de atribuire a scopului, care poate fi "none", "default" sau "optional":
* **None:** Scopul nu este atribuit automat și trebuie selectat manual în timpul procesului de autentificare sau prin configurarea clientului.
* **Default:** Scopul este atribuit automat la toate token-urile de acces și ID emise pentru client.
* **Optional:** Scopul poate fi solicitat de client în timpul procesului de autentificare, dar nu este inclus automat.
* Description: O descriere a scopului clientului și a informațiilor pe care le va furniza.
* Scopurile clientului sunt utilizate în OpenID Connect și OAuth 2.0 pentru a specifica accesul la anumite resurse sau seturi de date (cum ar fi profilul utilizatorului, adresa de e-mail etc.). Acestea sunt critice pentru gestionarea datelor și permisiunilor într-un mod securizat și conform cu principiul minimului privilegiu.



* Imaginea arată ecranul "Evaluate" din secțiunea "Client Scopes" a Keycloak. Această pagină este folosită pentru a evalua și a testa ce informații sunt incluse în token-urile generate pentru un utilizator, în funcție de scopurile clientului și mapările de roluri atribuite.
* Categoria mapării, care este de obicei "Token mapper", ceea ce înseamnă că maparea adaugă informații la token-ul de acces sau la token-ul ID.
* Prioritatea mapării, care poate dicta ordinea în care mapările sunt evaluate și aplicate la token.

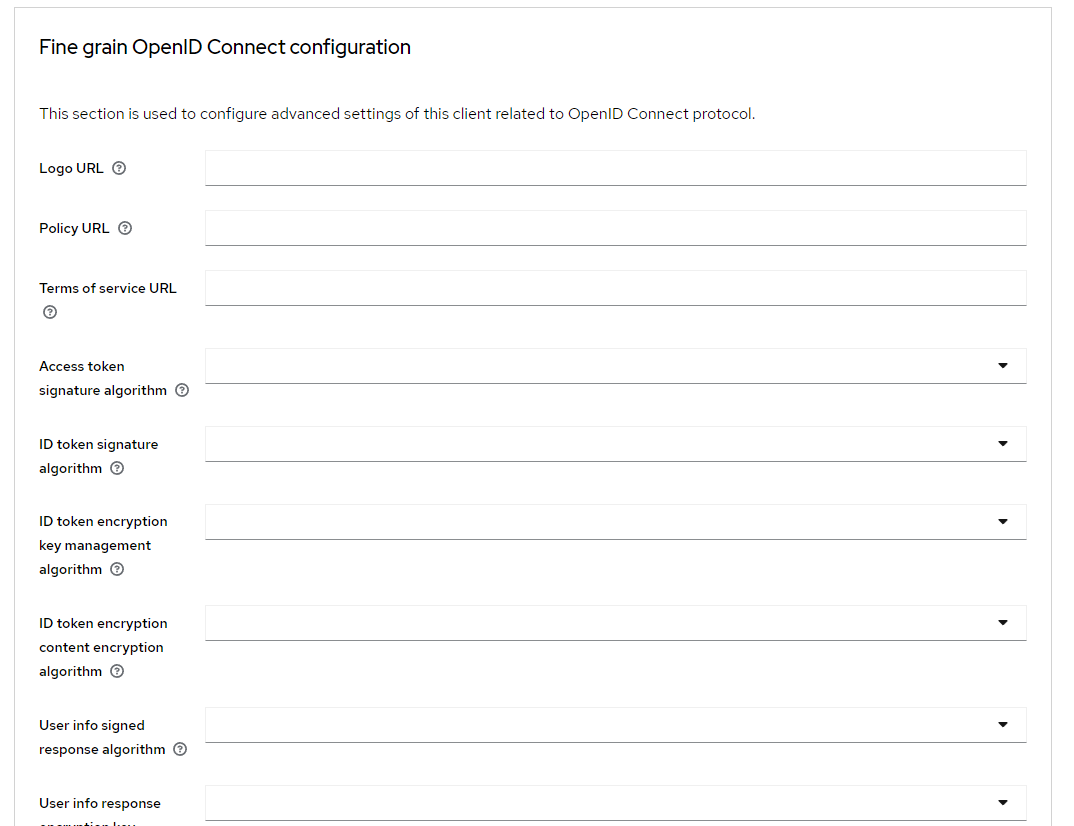


* Imaginea arată secțiunea "Sessions" pentru un client specific în cadrul interfeței de administrare Keycloak. Această secțiune este folosită pentru a vizualiza sesiunile active curente pentru clientul respectiv. Sesiunile active sunt instanțele în care utilizatorii sunt în prezent autentificați prin clientul respectiv.

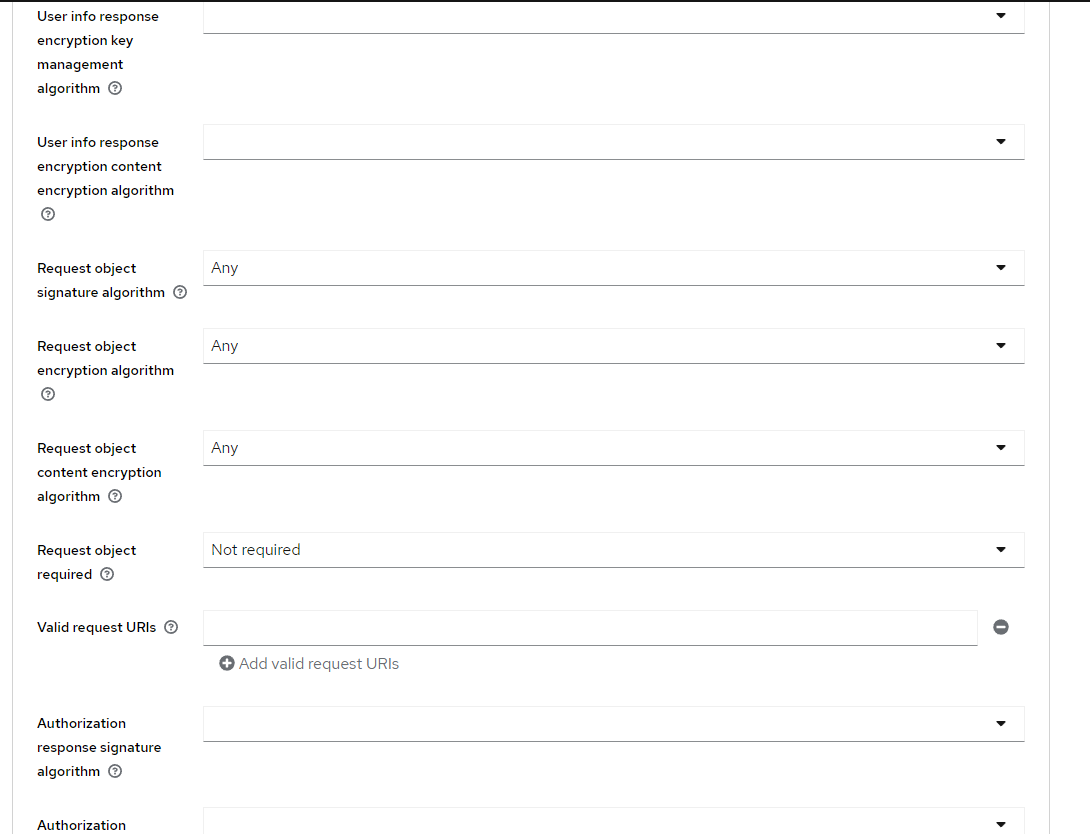


* tab-ul "Advanced" pentru un client OpenID Connect:

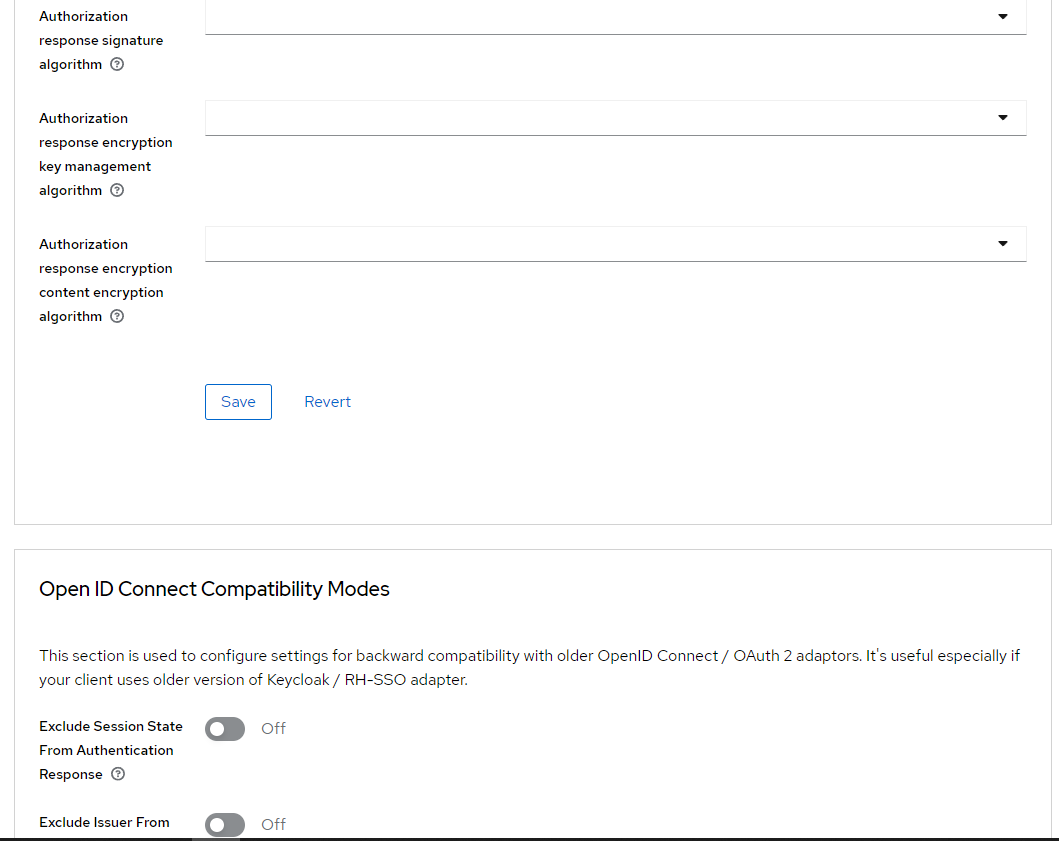
1. **Revocation** => Această secțiune este folosită pentru a gestiona politicile de revocare a token-urilor pentru client. Când setezi un timp pentru "Not before" (Nu înainte de), orice token emis înainte de această dată/timp va fi considerat invalid.
2. **Clustering =>** Aceste setări sunt legate de înregistrarea nodurilor în cazul în care Keycloak rulează în modul clusterizat. Node Re-registration timeout: Este un interval de timp în secunde după care nodurile înregistrate în cluster trebuie să se reînregistreze. O valoare negativă (-1) sugerează că acest mecanism este dezactivat.



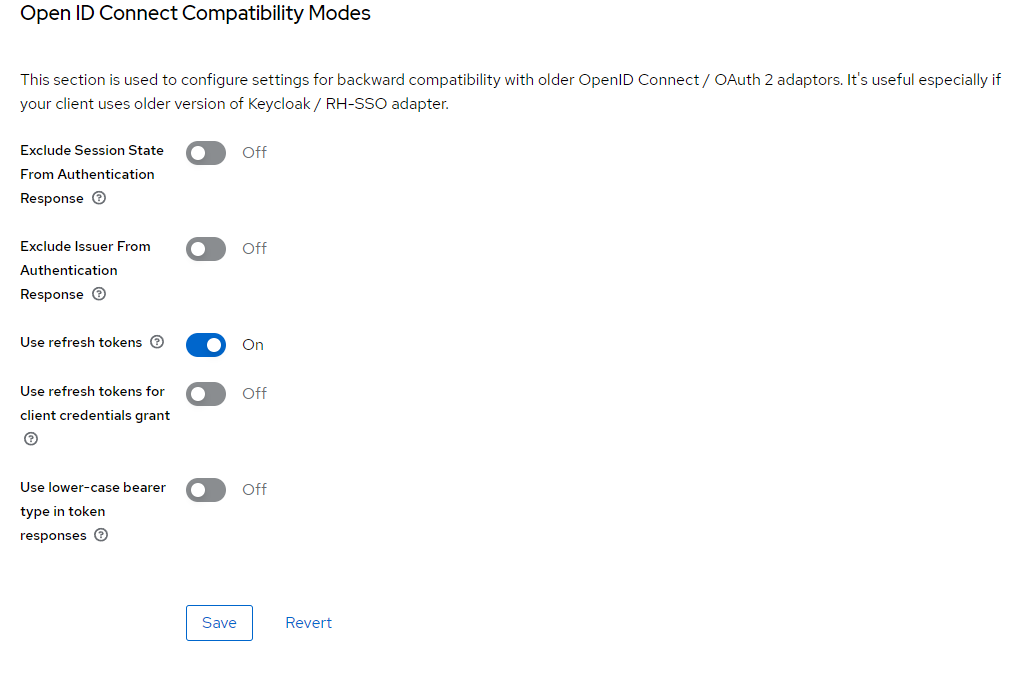
1. Logo URL: Adresa URL unde se poate găsi logo-ul care va fi folosit în paginile de login sau de consimțământ pentru clientul respectiv.
2. Policy URL: Adresa URL a politicii de confidențialitate a clientului, care poate fi afișată utilizatorilor.
3. Terms of Service URL: Adresa URL unde sunt descrise termenii de serviciu pentru client, care pot fi de asemenea afișați utilizatorilor.
4. Access Token Signature Algorithm: Algoritmul de semnătură folosit pentru semnarea tokenurilor de acces. De exemplu, RS256, HS256 etc.
5. ID Token Signature Algorithm: Algoritmul de semnătură folosit pentru semnarea ID tokenurilor.
6. ID Token Encryption Key Management Algorithm: Algoritmul folosit pentru gestionarea cheilor de criptare a ID tokenurilor. Aceasta este partea de "key management" din criptarea tokenului.
7. ID Token Encryption Content Encryption Algorithm: Algoritmul folosit pentru criptarea conținutului ID tokenurilor. Aceasta este partea de criptare a efectivă a conținutului tokenului.
8. User Info Signed Response Algorithm: Algoritmul folosit pentru semnarea răspunsurilor la cererile de informații ale utilizatorului (user info).
9. User Info Response Signature Algorithm: Similar cu "User Info Signed Response Algorithm", acesta specifică algoritmul folosit pentru semnarea răspunsurilor de informații ale utilizatorului.



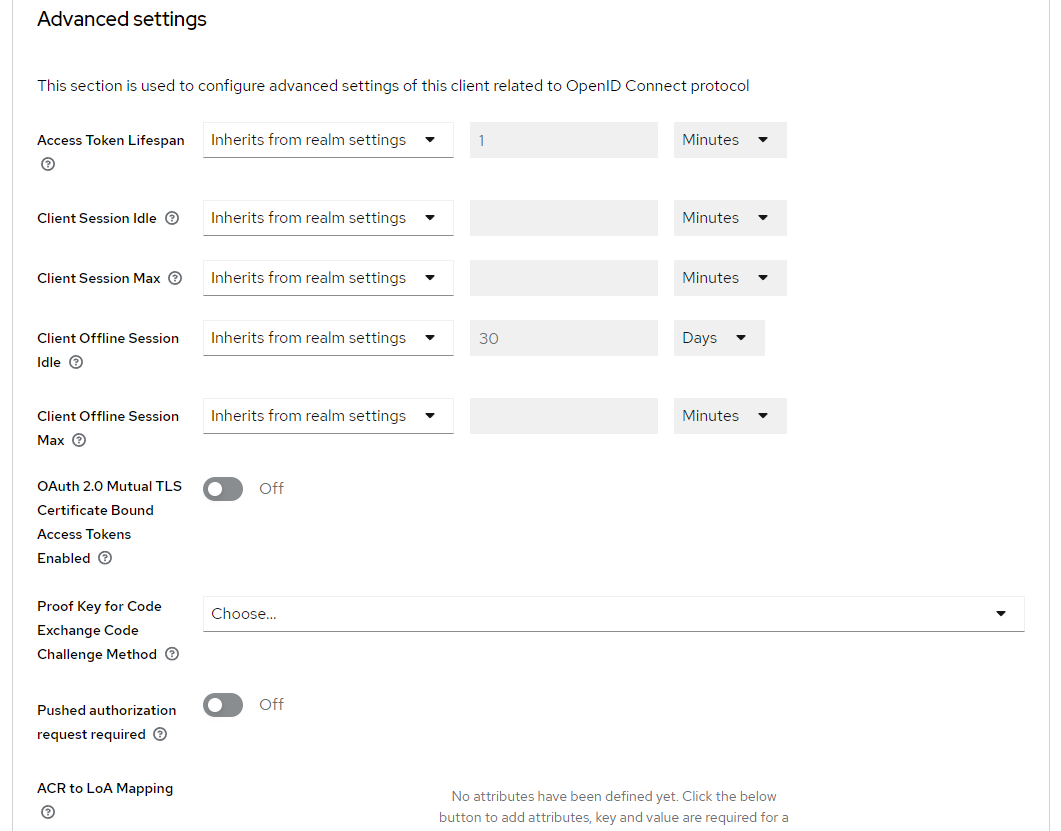
* User Info Response Encryption Key Management Algorithm: Algoritmul folosit pentru managementul cheii de criptare în răspunsurile cu informații ale utilizatorului (user info).
* User Info Response Encryption Content Encryption Algorithm: Algoritmul folosit pentru criptarea efectivă a conținutului răspunsurilor cu informații ale utilizatorului.
* Request Object Signature Algorithm: Algoritmul de semnătură pentru obiectele de solicitare (request objects). Acestea sunt JWT-uri trimise de clienți, care pot include unul sau mai multe Claims.
* Request Object Encryption Algorithm: Algoritmul folosit pentru criptarea obiectelor de solicitare.
* Request Object Content Encryption Algorithm: Algoritmul folosit pentru criptarea conținutului obiectelor de solicitare.
* Request Object Required: Indică dacă obiectele de solicitare sunt necesare pentru client. Dacă este setat la "Not required", clienții nu sunt obligați să trimită obiecte de solicitare.
* Valid Request URIs: URL-urile valide pe care Keycloak le va accepta pentru obiectele de solicitare. Aceasta este o măsură de securitate pentru a asigura că obiectele de solicitare vin de la surse de încredere.
* Authorization Response Signature Algorithm: Algoritmul folosit pentru semnarea răspunsurilor la autorizare. Aceasta se aplică la răspunsurile de autorizare trimise înapoi la client după un flux de autorizare de succes.



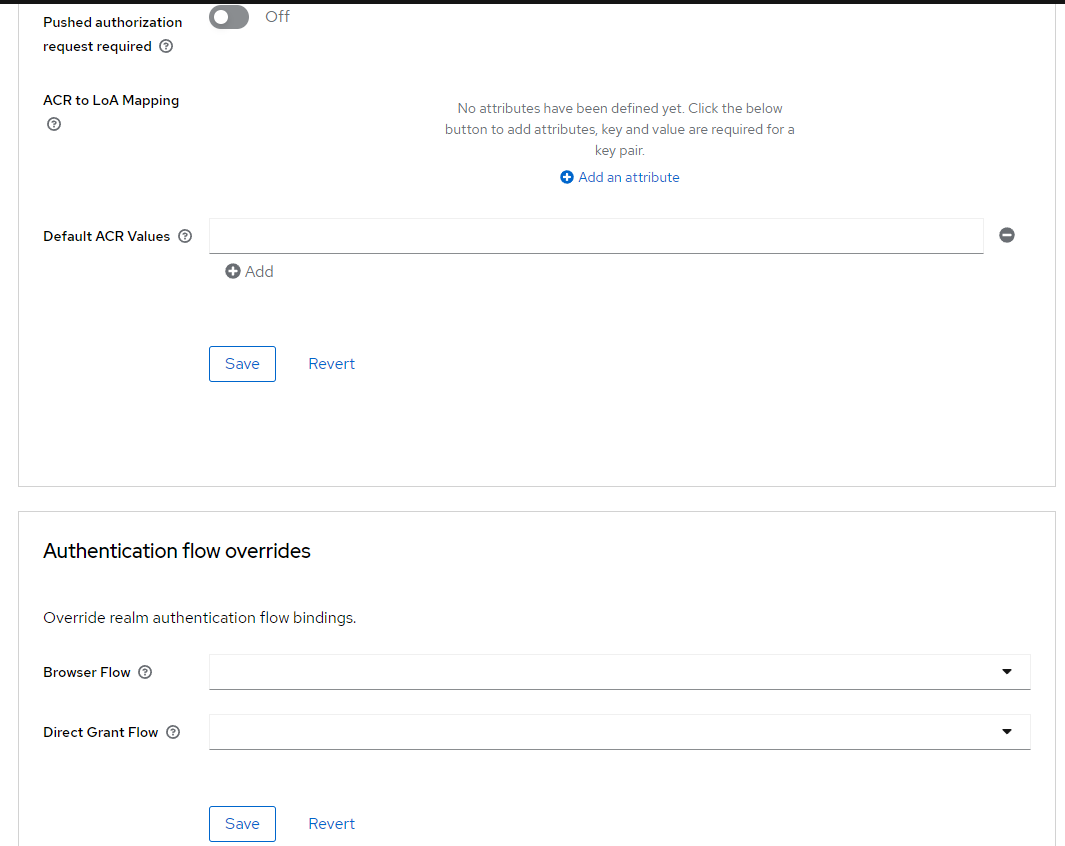
* Authorization Response Signature Algorithm: Algoritmul folosit pentru semnarea răspunsului de autorizare trimis de serverul Keycloak.
* Authorization Response Encryption Key Management Algorithm: Algoritmul de management al cheii folosit pentru criptarea răspunsului de autorizare.
* Authorization Response Encryption Content Encryption Algorithm: Algoritmul folosit pentru criptarea efectivă a conținutului răspunsului de autorizare.
* Exclude Session State From Authentication Response: Când este activat, acest comutator exclude starea sesiunii din răspunsul de autentificare. Aceasta poate fi necesară pentru compatibilitatea cu clienții care nu se așteaptă sau nu pot procesa acest parametru.
* Exclude Issuer From Authentication Response: Asemănător, când este activat, acest comutator exclude informațiile emitentului din răspunsul de autentificare. De asemenea, aceasta poate fi necesară pentru clienții mai vechi care nu se așteaptă la acest parametru.



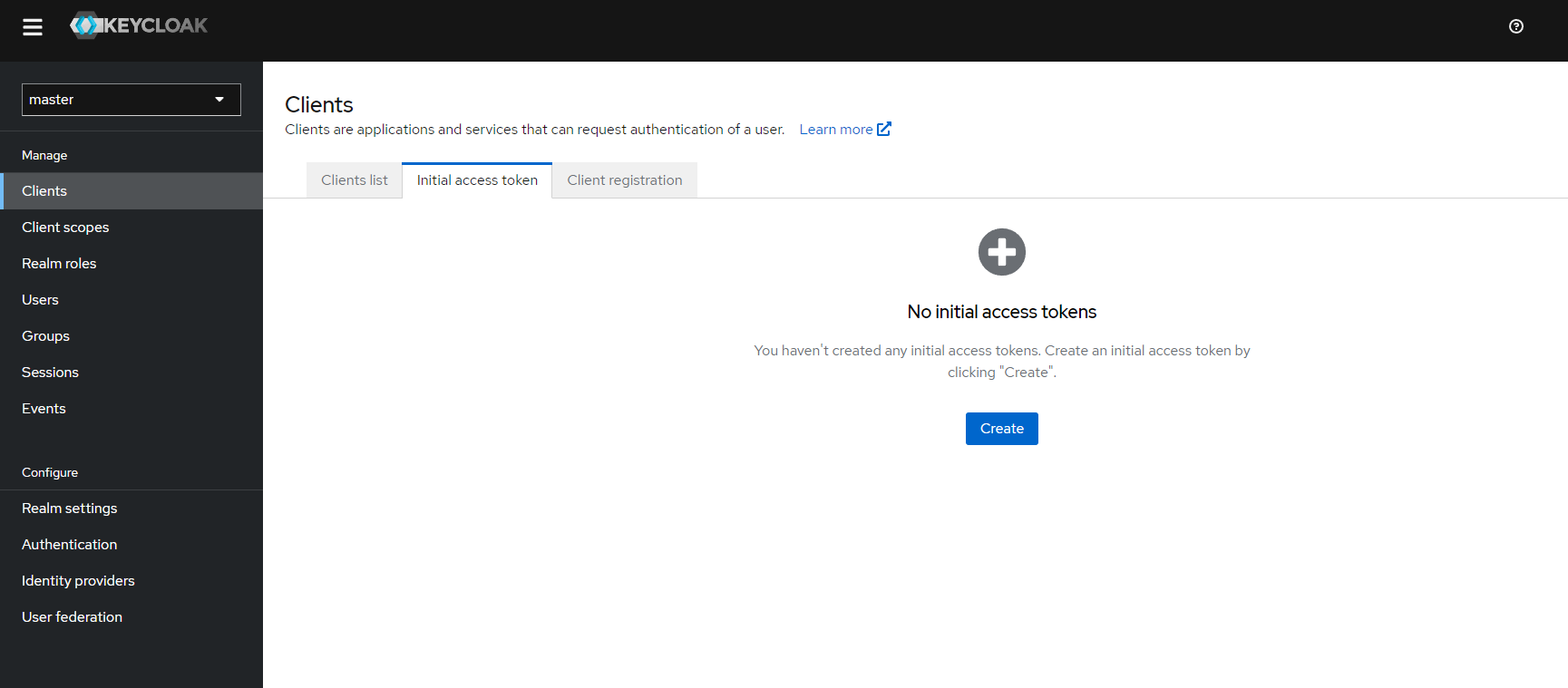
* Exclude Session State From Authentication Response: Dacă este activată, Keycloak nu va include starea sesiunii în răspunsurile de autentificare. Aceasta poate fi necesară pentru clienții care nu pot procesa acest parametru.
* Exclude Issuer From Authentication Response: Dacă este activată, informația despre emitentul tokenului (de obicei, URL-ul serverului Keycloak) nu va fi inclusă în răspunsurile de autentificare. Aceasta poate fi utilă dacă clienții nu se așteaptă la acest parametru.
* Use refresh tokens: Dacă este activată, permite utilizarea tokenurilor de reîmprospătare (refresh tokens). Acestea sunt folosite pentru a obține noi tokenuri de acces atunci când tokenul actual expiră, fără a necesita o nouă autentificare a utilizatorului.
* Use refresh tokens for client credentials grant: Specifică dacă tokenurile de reîmprospătare trebuie folosite și pentru fluxul de grant pentru acreditările clientului (client credentials grant). Acest flux este adesea folosit pentru aplicații server-to-server unde nu există o interacțiune a utilizatorului.
* Use lower-case bearer type in token responses: Unii clienți pot aștepta ca tipul purtătorului (bearer type) în răspunsurile tokenului să fie în litere mici. Activarea acestei opțiuni va asigura că răspunsurile tokenului folosesc "bearer" în litere mici, pentru compatibilitate cu astfel de clienți.



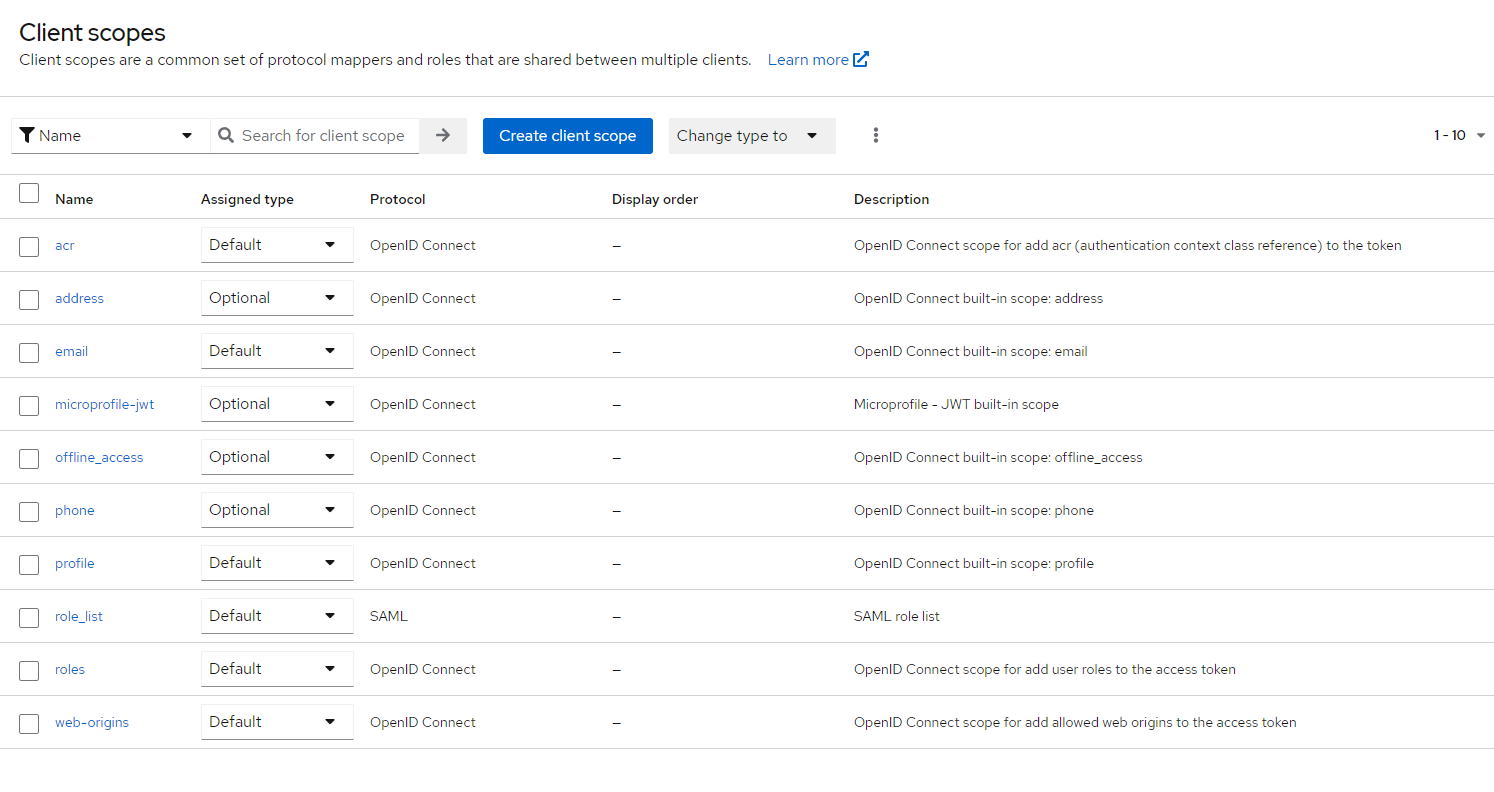
* Access Token Lifespan: Durata de viață a tokenului de acces. Aceasta poate fi setată să moștenească setările din realm sau poate fi configurată individual pentru client.
* Client Session Idle: Durata de timp în care o sesiune de client poate fi inactivă înainte de a fi închisă. De asemenea, aceasta poate moșteni setările din realm sau poate fi setată individual.
* Client Session Max: Durata maximă de viață a unei sesiuni de client indiferent de activitate. După această perioadă, sesiunea va fi închisă.
* Client Offline Session Idle: Durata de timp în care o sesiune offline de client poate fi inactivă înainte de a fi închisă. Sesiuile offline sunt utilizate pentru a permite reautentificarea fără interacțiunea utilizatorului.
* Client Offline Session Max: Durata maximă de viață a unei sesiuni offline de client.
* OAuth 2.0 Mutual TLS Certificate Bound Access Tokens Enabled: Această setare, când este activată, leagă tokenurile de acces la un certificat client TLS, ca parte a profilului de autentificare mutuală TLS pentru OAuth 2.0, ceea ce îmbunătățește securitatea.
* Proof Key for Code Exchange (PKCE) Code Challenge Method: Aceasta este o metodă de securizare a schimbului de coduri pentru autorizare în fluxurile OAuth, în special pentru aplicațiile client publice, cum ar fi aplicațiile mobile sau aplicațiile bazate pe spațiu de lucru.
* Pushed Authorization Request Required: Când este activată, clientul trebuie să folosească pushed authorization requests (PAR), care sunt o caracteristică de securitate suplimentară în care cererile de autorizare sunt trimise la server într-un mod care le face mai rezistente la anumite atacuri.
* ACR to LoA Mapping: ACR (Authentication Context Class Reference) la LoA (Level of Assurance) mapping permite maparea între contextul de autentificare și nivelul de asigurare. Acesta este un concept avansat utilizat în situațiile în care diferite niveluri de autentificare sunt necesare.



* Această secțiune permite suprascrierea fluxurilor de autentificare predefinite ale realm-ului pentru clientul specific. Poți specifica fluxuri de autentificare diferite pentru scenarii diferite, cum ar fi autentificarea prin browser sau autentificarea directă (direct grant).
* Browser Flow: Permite specificarea unui flux de autentificare personalizat atunci când utilizatorul se autentifică printr-un browser web.
* Direct Grant Flow: Permite specificarea unui flux de autentificare pentru grant-uri directe, care sunt folosite de obicei în scenarii non-browser, cum ar fi autentificarea dintr-o aplicație client sau un script.



* Clients list: Acesta este locul unde poți vedea toți clienții configurați în realm-ul actual.
* Initial access token: Aici poți crea tokenuri de acces inițiale, care sunt folosite pentru a înregistra clienți noi din afară interfetei de administrare, de obicei printr-o API sau un proces automatizat.
* Client registration: Permite înregistrarea clienților noi în realm. Acest proces poate fi efectuat manual prin interfața de utilizator sau poate fi automatizat.



* Imaginea arată pagina "Client Scopes" în consola de administrare Keycloak. Scopurile clientului (client scopes) sunt o colecție de mapper-e de protocol și roluri care sunt partajate între mai mulți clienți. Ele sunt utilizate pentru a defini cum sunt generat tokenurile și ce informații sunt incluse în ele.
* De exemplu, un client care are nevoie de acces la adresa de email a utilizatorului va include scopul "email" în cererea sa de autentificare. Keycloak va utiliza mapperele asociate acestui scop pentru a include adresa de email în tokenul de acces sau ID token-ul emis după autentificare.
* **KC\_HOSTNAME** = hostname=ul real sau adresa IP pe care Keycloak va fi accesibil in cluster

Am preluat din documentatia Keycloak-ului dockerfile-ul propus de acestia pentru configurarea simpla a unei imagini de Keycloak:

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

Dupa am creat cate un yaml pentru postgresSQL si pentru keycloak care se conecteaza la baza de date postgres: postgres.yaml si keycloak.yaml(keycloak.yaml l-am preluat din documentatie la care am adaugat si baza de date: https://raw.githubusercontent.com/keycloak/keycloak-quickstarts/latest/kubernetes/keycloak.yaml).

Dupa am rulat comanda: docker build -t biancadaniela/keycloak:1.0 . pentru a crea o imagine din acel dcokerfile pe care am incarcat-o pe docker hub si am pus-o in kecloak.yaml. Ca sa incarc pe docker hub am dat comanda:

1. Docker login => ca sa ma loghez la contul de docker hub
2. docker build -t biancadaniela/keycloak . => am creat imaginea
3. docker push biancadaniela/keycloak => i-am dat push pe docker hub

Dupa am rulat:

1. kubectl apply -f postgres.yaml
2. kubectl apply -f keycloak.yaml

To install custom providers, you just need to define a step to include the JAR file(s) into the /opt/keycloak/providers directory. This step must be placed before the line that RUNs the build command, as below:

1. # A example build step that downloads a JAR file from a URL and adds it to the providers directory
2. FROM quay.io/keycloak/keycloak:latest as builder
3. ...
4. # Add the provider JAR file to the providers directory
5. ADD --chown=keycloak:keycloak <MY\_PROVIDER\_JAR\_URL> /opt/keycloak/providers/myprovider.jar
6. ...
7. # Context: RUN the build command
8. RUN /opt/keycloak/bin/kc.sh build

**Root URL:** Acesta este URL-ul de bază al aplicației tale. Dacă aplicația ta este accesibilă la https://example.com/app, atunci acesta este Root URL-ul tău. Keycloak va folosi acest URL ca bază pentru a construi URL-uri de redirecționare dacă nu sunt specificate explicit.

**Home URL:** Acesta este URL-ul la care utilizatorii ar trebui să fie redirecționați după ce se autentifică, dacă nu este specificat un alt URL de redirecționare în timpul procesului de login. Acesta poate fi, de exemplu, pagina principală a aplicației tale.

**Valid Redirect URIs:** Aici trebuie să introduci toate URL-urile valide la care Keycloak este permis să redirecționeze utilizatorii după autentificare. De exemplu, https://example.com/app/\* ar permite redirecționările către orice sub-căi ale aplicației tale. Asigură-te că introduci wildcard-ul \* cu precauție pentru a evita vulnerabilități de securitate.

**Valid Post Logout Redirect URIs:** Acestea sunt URL-urile la care utilizatorii pot fi redirecționați după ce se deconectează de la aplicația ta. De exemplu, https://example.com/ ar putea fi pagina de start sau de bun venit a site-ului tău.

**Web Origins:** Aceasta este lista originilor web (scheme + hostname + port) care sunt permise să facă cereri CORS către serverul tău Keycloak. De obicei, aceasta va fi aceeași cu Root URL-ul, dar fără calea URL-ului.

Iată un exemplu generic de cum ar putea arăta aceste câmpuri pentru o aplicație locală:

**Root URL: http://localhost:8080**

**Home URL: http://localhost:8080/home**

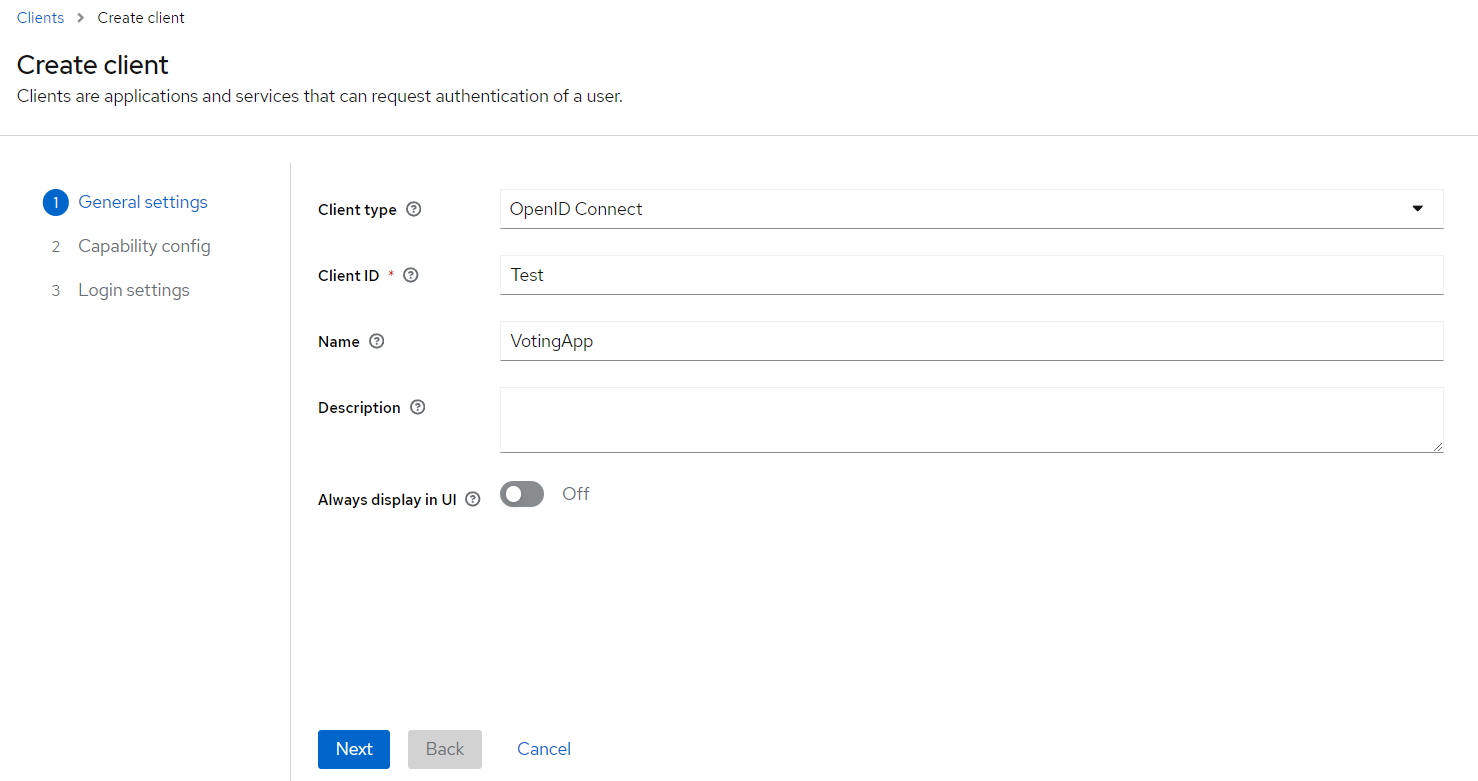
**Valid Redirect URIs: http://localhost:8080/\***

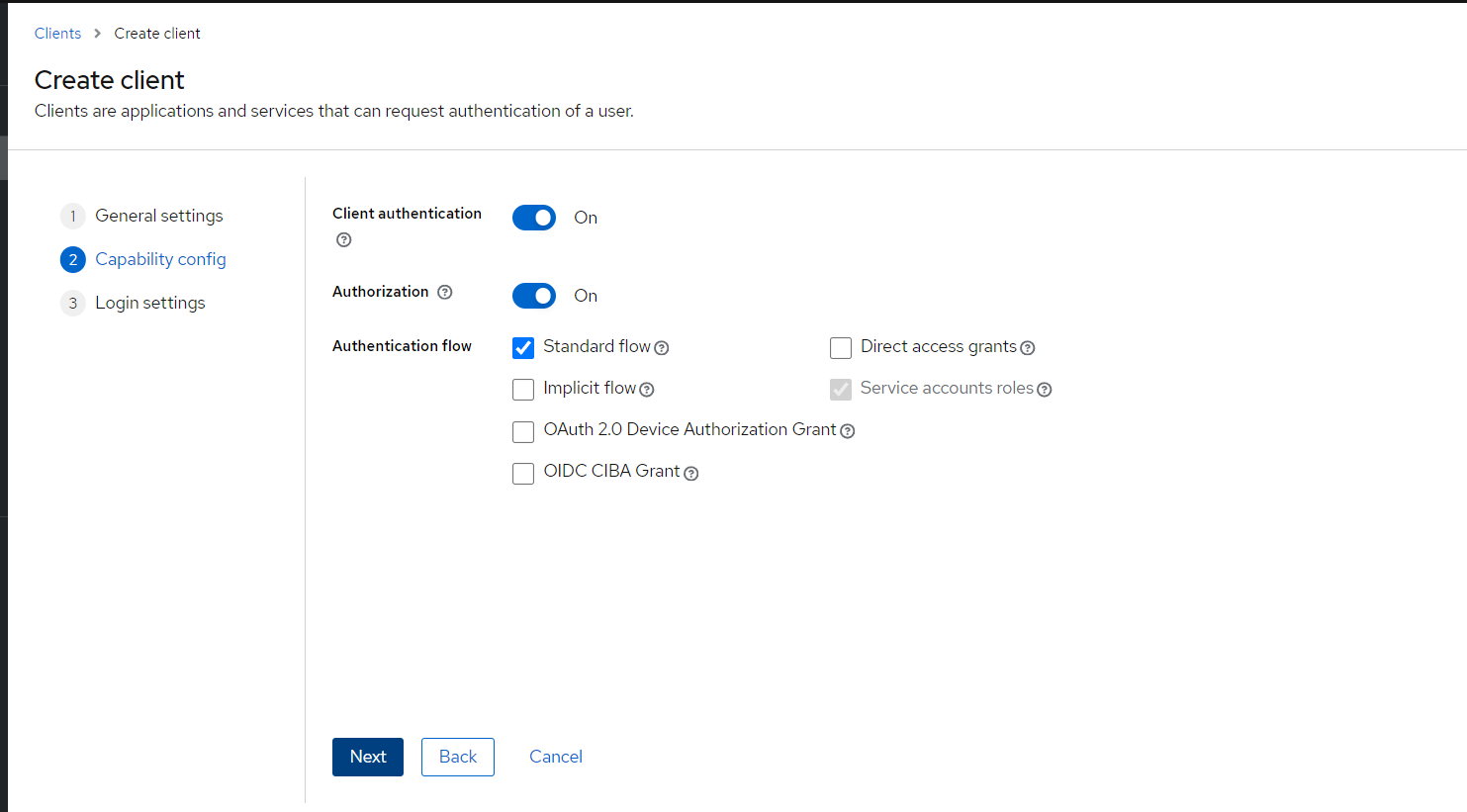
**Valid Post Logout Redirect URIs: http://localhost:8080/logout**

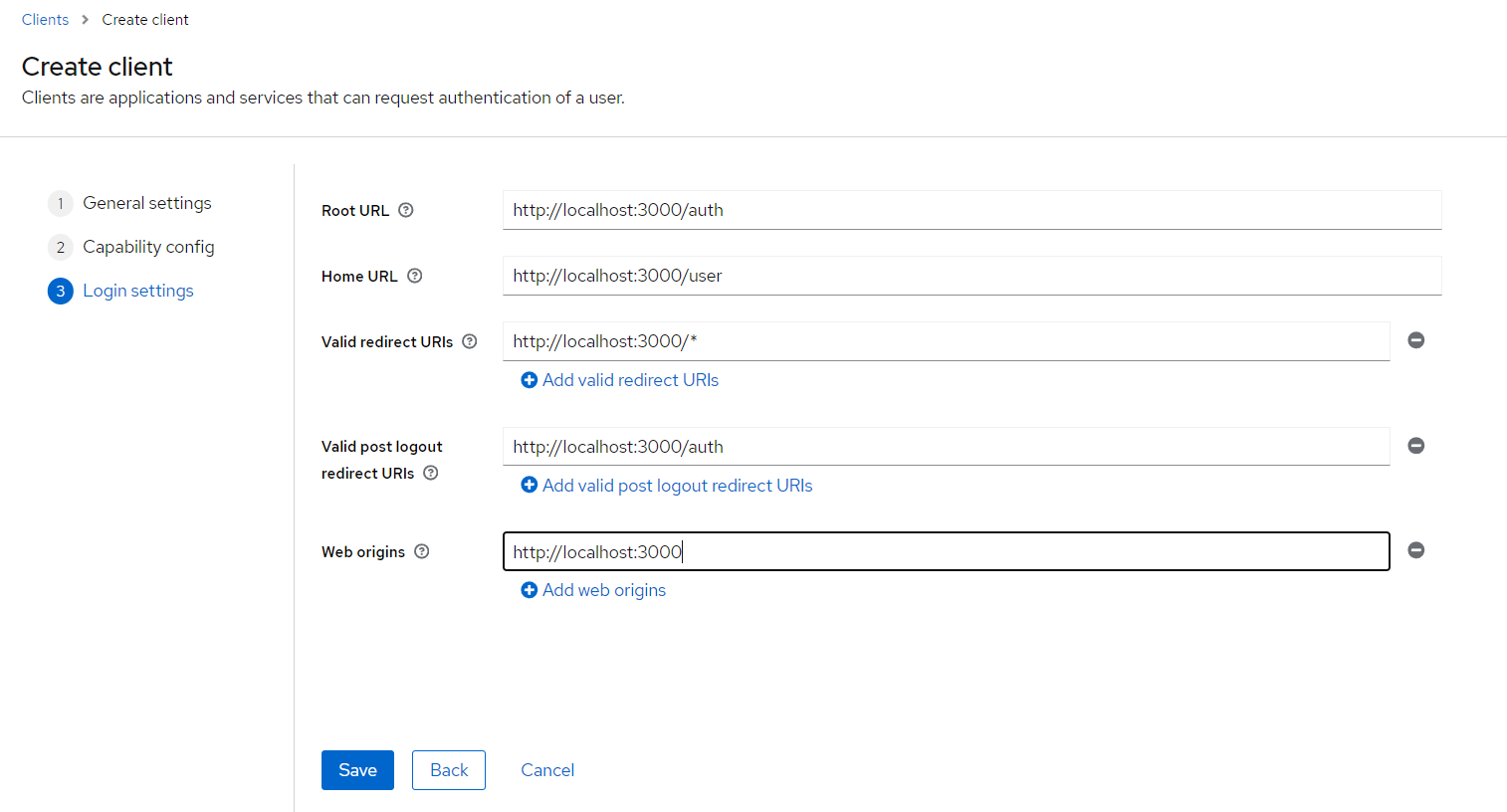
**Web Origins:** [**http://localhost:8080**](http://localhost:8080)

Ca sa imi adaug site-ul in Keycloak:

1. Am creat un client:







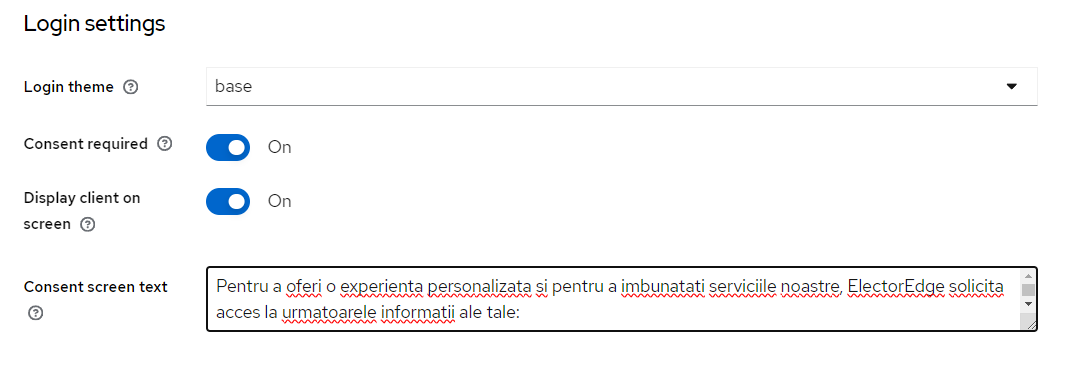
Aceste setări pe care le vezi sunt opțiuni în consola de administrare Keycloak care îți permit să personalizezi comportamentul ecranului de login pentru clientul tău OAuth:

**Login Theme:** Permite să selectezi tema pentru pagina de login. Keycloak vine cu câteva teme predefinite, cum ar fi "base", dar poți crea și teme personalizate. Aceasta afectează aspectul paginilor de login, înregistrare, și altele asemenea.

**Consent Required:** Dacă această opțiune este activată (On), utilizatorii vor trebui să dea consimțământul pentru a permite aplicației să acceseze informațiile lor înainte ca aceasta să primească un token. Aceasta este o parte a procesului de autorizare și poate fi necesară pentru aplicații care cer acces la informații sensibile ale utilizatorului.

**Display Client on Screen**: Dacă este activat, numele clientului (aplicației) va fi afișat pe ecranul de login. Acest lucru poate fi util pentru utilizatori pentru a recunoaște pentru ce aplicație își dau consimțământul.

**Consent Screen Text:** Aici poți introduce textul care va fi afișat pe ecranul de consimțământ, explicând utilizatorilor de ce și ce permisiuni solicită aplicația. Acesta este relevant doar dacă opțiunea "Consent Required" este activată.



**Front Channel Logout:** Este o setare care permite logout-ul utilizatorului din aplicația client printr-un redirect de browser. Când un utilizator se deconectează din Keycloak, acesta va fi redirecționat către URL-ul specificat aici pentru a încheia și sesiunea pe partea de client. Aceasta înseamnă că procesul de logout este inițiat de frontend-ul aplicației (ex. browser-ul web).

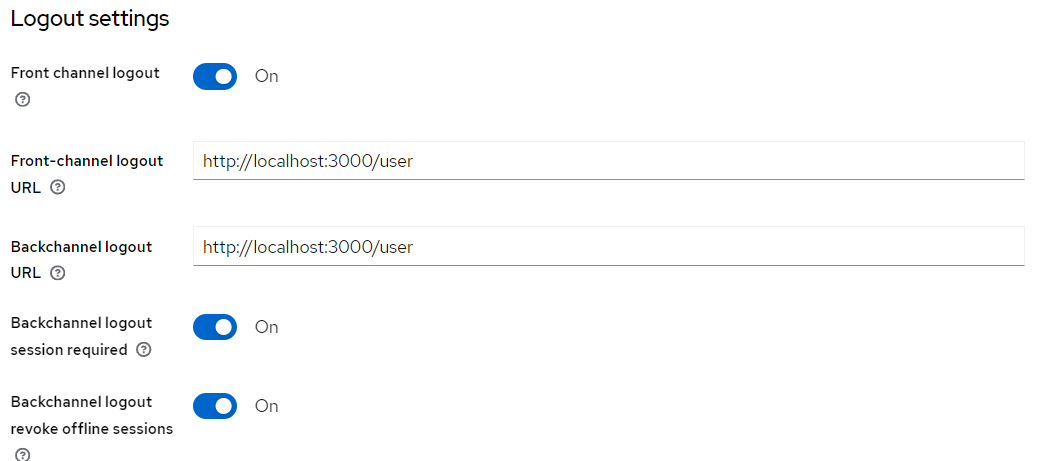
**Front-channel logout URL:** Aici se introduce URL-ul unde clientul (aplicația web) va primi cererea de logout. Keycloak va trimite un request la acest URL pentru a notifica clientul că utilizatorul a fost deconectat.

**Backchannel Logout:** Este un mecanism prin care serverul Keycloak trimite o cerere de logout direct la aplicația client fără a necesita interacțiunea utilizatorului printr-un browser. Aceasta este folosită pentru aplicații care pot gestiona cereri de logout pe partea de server.

**Backchannel logout URL:** Aici se introduce URL-ul endpoint-ului de backchannel logout al aplicației tale, unde Keycloak va trimite cereri de logout.

**Backchannel logout session required:** Dacă este activată, cererea de logout backchannel va include un ID de sesiune, ceea ce permite aplicației client să identifice sesiunea care trebuie încheiată.

**Backchannel logout revoke offline sessions:** Dacă această opțiune este activată, atunci când un utilizator se deconectează, toate sesiunile offline ale acestuia vor fi de asemenea revocate. Acest lucru este util pentru aplicații care suportă funcționalitatea de "remember me" și păstrează sesiuni offline.

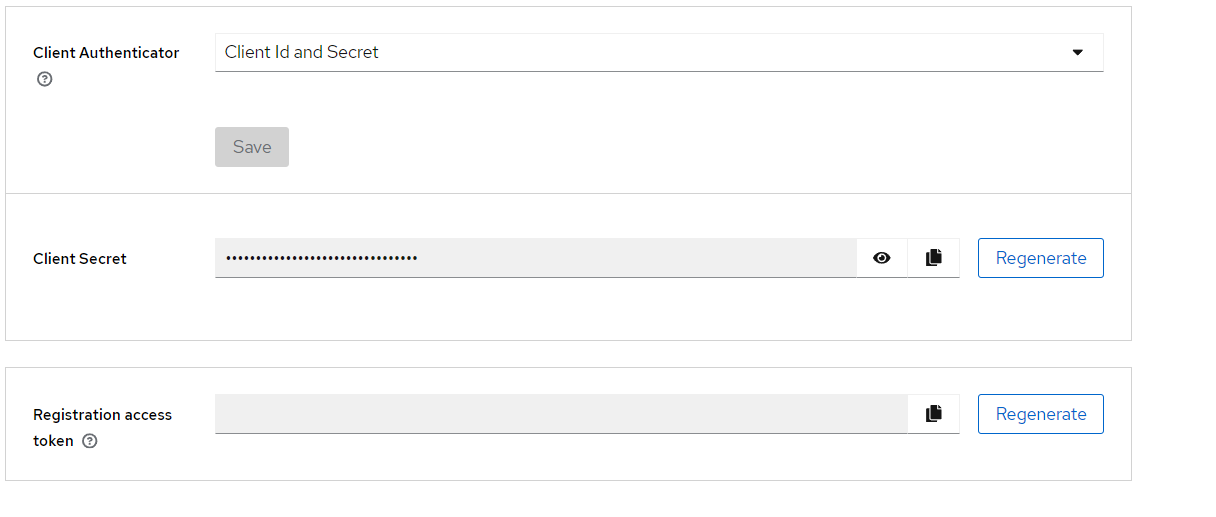


**Client Authenticator:** Acesta este mecanismul prin care clientul se va autentifica la serverul Keycloak. "Client ID and Secret" este o metodă comună, unde clientul folosește un ID unic și un secret (parolă) pentru a se identifica.

**Client Secret:** Este parola/cheia secretă utilizată de client pentru a se autentifica la Keycloak. Acest secret trebuie păstrat în siguranță și trebuie să fie cunoscut doar de aplicația client și de serverul Keycloak.

**Regenerate:** Butonul "Regenerate" îți permite să generezi un nou secret pentru client, în cazul în care cel vechi a fost compromis sau din orice alt motiv ai nevoie să schimbi secretul.

**Registration Access Token:** Acest token este folosit pentru a gestiona înregistrarea clientului printr-un API. Dacă este activat și un token este generat, acesta poate fi folosit pentru a modifica configurarea clientului prin apeluri API, în loc să se facă manual prin consola de administrare. De obicei, nu este nevoie să fie setat pentru majoritatea aplicațiilor.

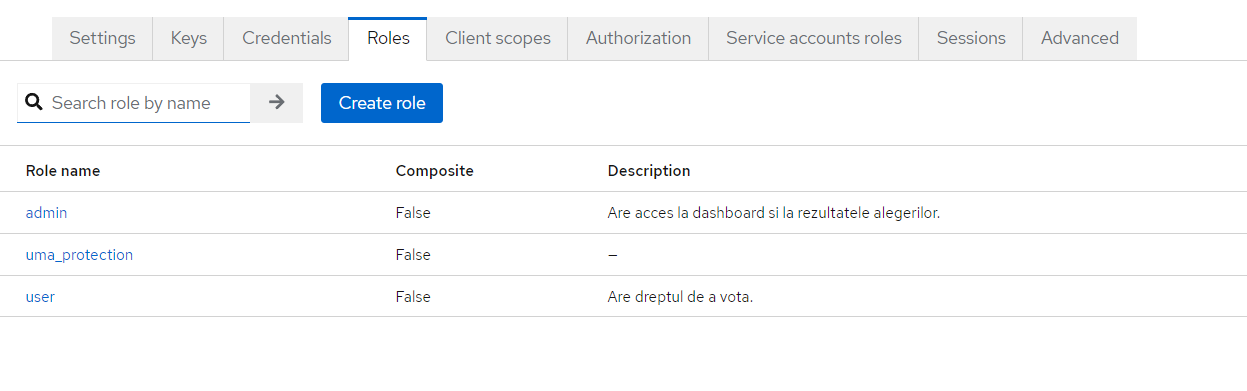


**Role name**: Este numele rolului pe care îl poți crea pentru client. Rolurile sunt folosite în politici de acces și decizii de autorizare. De exemplu, ai putea avea roluri precum "user", "admin", "editor" etc.

**Composite:** O opțiune "Composite" setată pe "True" înseamnă că rolul este un rol compozit, care poate include mai multe alte roluri. Dacă un rol este compozit, atunci un utilizator cu acel rol va moșteni implicit permisiunile tuturor rolurilor incluse.

**Description:** Aici poți adăuga o descriere pentru rol, explicând scopul sau permisiunile pe care le conferă.

Dacă în imaginea pe care ai încărcat-o este deja definit rolul "uma\_protection", nu va trebui să setezi nimic aici dacă nu ai nevoie să creezi roluri suplimentare sau să modifici rolurile existente. Rolul "uma\_protection" este de obicei asociat cu funcționalitatea User-Managed Access (UMA) din Keycloak, care este un standard pentru controlul accesului delegat.



**Import:** Butonul "Import" este folosit pentru a importa setările de autorizare, cum ar fi politicile, permisiunile și resursele, dintr-un fișier JSON sau dintr-o altă sursă configurabilă. Acest lucru este util pentru a configura rapid un client cu setări predefinite sau pentru a migra setările între medii diferite sau între clienți diferiți.

**Policy Enforcement Mode:**

**Enforcing:** În acest mod, politicile de autorizare pe care le configurezi sunt aplicate strict. Accesul este permis sau refuzat conform regulilor definite.

**Permissive:** Politicile sunt evaluate, dar nu sunt forțate. Dacă nu există o politică care să permită accesul, accesul nu va fi refuzat.

**Disabled:** Nu se aplică nicio politică de autorizare, accesul la resurse este permis fără restricții.

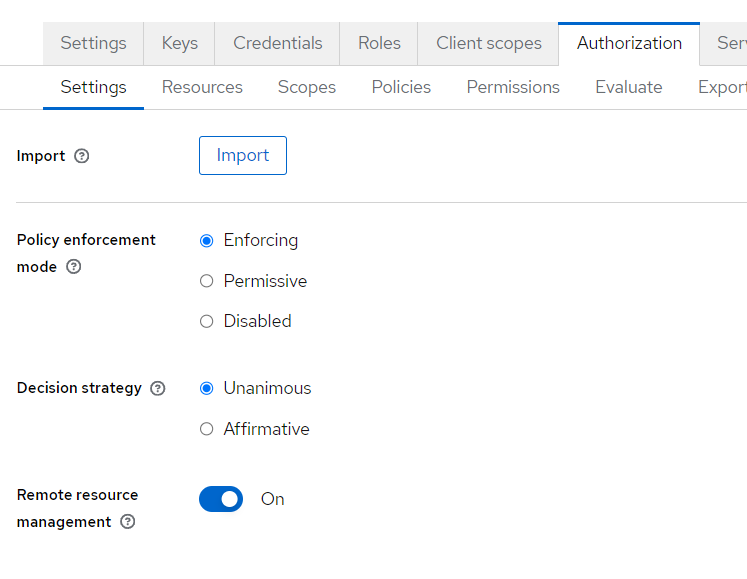
**Decision Strategy:**

**Unanimous:** Toate politicile asociate cu o permisiune trebuie să evalueze pozitiv pentru a acorda accesul.

**Affirmative:** Doar o singură politică din cele asociate trebuie să evalueze pozitiv pentru a acorda accesul.

**Remote Resource Management:**

**Activat (On):** Permite clientului să gestioneze resursele de la distanță printr-un API. De exemplu, poți avea o aplicație client care să creeze, să actualizeze sau să șteargă resurse în Keycloak.



**browser (Built-in):** Fluxul de autentificare utilizat pentru autentificarea bazată pe browser. Acesta este fluxul standard utilizat când un utilizator se loghează într-o aplicație web prin intermediul unui browser

**clients (Built-in):** Fluxul de autentificare pentru clienții aplicației, adică pentru software-ul care accesează Keycloak pentru a obține tokenuri.

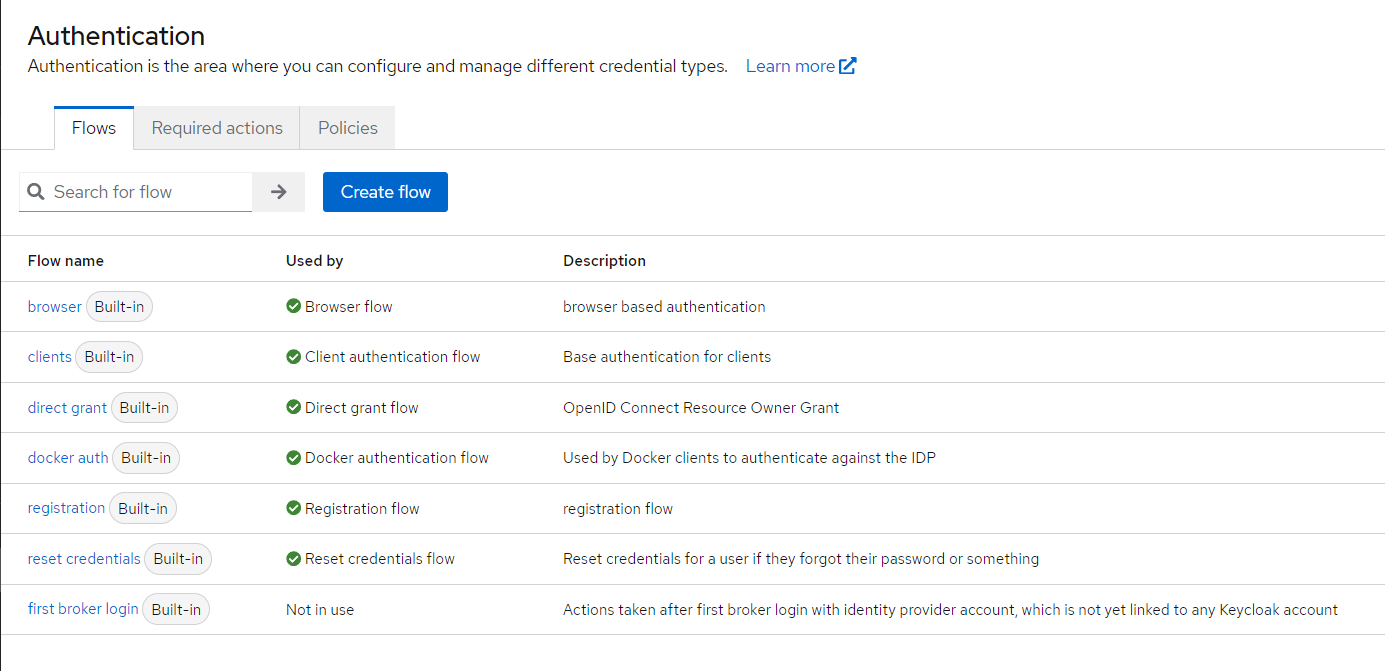
**direct grant (Built-in):** Un flux de autentificare care implementează protocolul OpenID Connect Resource Owner Password Credentials Grant, unde credențialele utilizatorului sunt trimise direct și autentificarea se face fără interacțiunea utilizatorului cu o pagină de logare.

**docker auth (Built-in):** Un flux de autentificare specializat utilizat de clienții Docker pentru a se autentifica împotriva furnizorului de identitate (IDP).

**registration (Built-in):** Fluxul de autentificare utilizat pentru înregistrarea unui nou utilizator.

**reset credentials (Built-in):** Acest flux este folosit când un utilizator a uitat parola sau are nevoie să-și reseteze credențialele.

**first broker login (Built-in):** Este folosit pentru acțiunile care sunt luate după prima autentificare a unui utilizator cu un furnizor de identitate terț (broker), care încă nu este legat de un cont Keycloak existent.



**Required:** Pasul trebuie completat cu succes pentru ca procesul de autentificare să continue. Dacă pasul eșuează, atunci întregul proces de autentificare va eșua.

**Alternative:** Mai multe pași pot fi marcați ca alternativi, ceea ce înseamnă că doar unul dintre pașii alternativi trebuie să reușească pentru a permite autentificarea să continue. Acest lucru este util atunci când dorești să oferi utilizatorilor mai multe metode de autentificare din care pot alege.

**Optional:** Pasul poate fi completat, dar nu este obligatoriu pentru succesul autentificării. Utilizatorii pot sări peste acest pas și să continue procesul de autentificare.

**Disabled:** Pasul este dezactivat și nu va fi executat în cadrul fluxului de autentificare.