

Metode de a

→ crea o auth și de a
terge o auth ⇒ Aceste metode
pelează UserModel's credential
manager care este responsabil
să știe unde să citească/
să scrie credentiale, de ex
local storage / federated storage.

Credential Input Validator

↓
permite keycloak
să știe că acest furnizor
poate fi folosit și
pt a valida o auth pt un
Authenticator

is Valid()

Credential Provider

! → validarea și verificarea
codurilor de acces

priza un parametru

generic care extinde
un Credential Model

credential_data

secret_data

json

Vault Provider

→ token FCM

EventListener Provider

⇒ pt a da și a asculta când este
trimis un token FCM și pt a-l stoca
in vault așa cum trebuie

Code Acces

serverul keycloak generează un cod
de acces pe care îl trimite mai departe
prin FCM.

codul este stocat temporar undeva
pt a putea face verificarea

Cache Infinispan

↳ este pentru distribuția uniformă a datelor din cache in
cluster

Cum ar trebui să fie stocat codul de acces?

⇒ Infinispan cache

→ este un sistem de caching și stocare a datelor in-memory ⇒ rapid și scalabil

- securitate:
- controlul accesului ⇒ permite configurarea controlului accesului pe baza de roluri, a.e. doar clienți autorizați să poată accesa și manipula cache-ul.
 - securitatea transportului ⇒ atunci când facem distribuția uniformă între clustere, Infinispan poate fi configurat să folosească SSL/TLS pt a cripta comunicarea între noduri, prevenind interceptarea datelor în tranziție
 - securitatea stocării: deși infinispan este un sistem de stocare in-memory, există opțiuni pt a persista datele în moduri securizate folosind criptarea

disc pt stocarea offline.

- criptarea datelor din cache
- securitatea clusterului ⇒ nodurile de încredere se pot alătura clusterului
- timp de expirare.

1. Credential Provider ⇒ este componenta care gestionează credențialele specifice, în acest caz codurile de acces. Aici se va implementa logica pt generarea, stocarea și validarea codurilor de acces. Acest provider va interacționa cu Infinispan cache sau cu un Vault provider pt a reține codurile de acces în mod sigur.

2. Authenticator ⇒ Acesta este responsabil pt procesul de autentificare propriu-zis, adică verificarea dacă codul de acces introdus de utilizator este valid. Authenticatorul va căuta codul de acces în locația de stocare temporară și va compara codul introdus de utilizator cu cel stocat. Dacă codul corespunde și este încă valid (nu a expirat), atunci utilizatorul va fi autentificat.

3. Vault Provider \Rightarrow (În cazul în care nu dorești să stocăm codurile de acces direct în cache (poate din motive de securitate suplimentare), ai putea să folosești un Vault Provider. Aceasta înseamnă interacționarea cu un sistem extern de gestionare a secretelor (de ex: HashiCorp Vault), unde codurile de acces pot fi stocate în mod securizat) ??? \Rightarrow trebuie criptare ai doar keycloak să poată cripta/decripta date

4. EventListener Provider \Rightarrow Această componentă ascultă și reacționează la diferite evenimente din cadrul keycloak. De ex: logare când un cod este generat, când expiră un cod \Rightarrow acțiune necesară, când un token FCM este trimis de către un utilizator să îl stocăm în vault.

5. Infinispan Cache \Rightarrow este sistemul de caching folosit de keycloak pt a stoca temporar codurile de acces. Infinispan oferă performanță înaltă și este bine integrat în keycloak, făcându-l o opțiune bună pt stocarea datelor care trebuie să fie rapid accesibile și care nu sunt persistente pe termen lung.

6. Secret Data \Rightarrow Acest model de date conține informații sensibile, cum ar fi codul de acces în sine. Secret Data ar trebui să fie stocat într-un mod securizat și să fie accesibil doar sistemului de auth pt validare.

7. Credential Data \Rightarrow Acest model de date conține metode legate de credențiale, cum ar fi data și ora la care codul de acces va expira. Aceasta ajută la det. dacă un cod de acces este încă valid atunci când este introdus de utilizator.

8. Credential Model \Rightarrow este folosit pt a reprezenta credențialele unui util. În acest caz, ar include SecretData (ex: cod acces) + Credential Data - legat de acces code de contul utilizatorului și este folosit în procesul de autentificare.

SecretData

\rightarrow cod acces

\rightarrow token FCM

Credential Data

\rightarrow data de expirare

\rightarrow data la care a fost emis

\rightarrow data când a fost ultima dată emis.

EventListener Provider

\rightarrow ex de utilizare: trimiteră unui token de la un util

Vault

(VS)

- securitate ridicată
- specializat în stocarea secretelor
- poate servi mai multe aplicații
- rotația și gestionarea ciclului de viață al secretelor
- accesul la secrete mai lent ⇒
⇒ apeluri în rețea și operații de criptare/decriptare

Infinispan

- oferă acces rapid la date
- setare timp de expirare
- utilizare în keycloak mai simplă,
deoarece este deja integrat și
folosit pt sesiuni și alte date temporare

Vault Provider

- tokenFAM

criptat ⇒
decriptare

key session

Infinispan

- code access

Deci fluxul va fi următorul:

- o să am Credential Data în core specific date normale aici (metadate) (2)
- o să am SecretData în core stocarea doar identificatorul unic al credentialelor. (1)
- Credential Model în core asociază un utilizator de o credință și setare type la credință prin care să fie recuperată de restul sistemului.