Android

Implementare

① Adăugare dependinte FCM ⇒ vezi document word.

② De implementat interfața serviciului

Firebase Messaging Service ⇒ avem clasa

MyFirebase Messaging Service ⇒ metode suprascrise:

→ onMessageReceived ⇒ logica pt gestionarea mesajelor push

→ onNewToken ⇒ logica pt trimiterea noului token FCM

Explicații flux:

→ Obținerea tokenului FCM în App Android:

- adăugare dependinte FCM in App (word)

Keycloak - interfețe de implementat

Implementare

① Authenticator ⇒ această interfață este utilizată în Keycloak pt a implementa logica de autentificare personalizată. Folosită pt a verifica dacă un utilizator are un token FCM asociat, și dacă nu are, pt a declanșa procesul de înregistrare a dispozitivului.

② VaultProvider ⇒ este utilizat pt stocarea securizată a datelor sensibile ⇒ stocarea tokenurilor FCM ⇒ sunt criptate și protejate datele
- stocare tokenul FCM și asociere cu un utilizator printr-un identificator unic (stocare cheie)
- securizare acces la vault (doar utilizatorii autorizați pot recupera sau modifica

- MyFirebaseMessagingService (clasa care extinde FirebaseMessagingService):
  - onNewToken(String token) ⇒ această metodă este utilizată atunci când un nou token este generat. FCM gestionează onNewToken în situații precum inițializarea aplicației sau reînnoirea tokenului. Pentru a afla tokenul FCM trebuie suprascrisă această metodă. În cadrul acestei funcții, FCM furnizează automat tokenul. Rolul meu e să implementez logica de trimitere a acestui token către serverul Keycloak (de fapt către un reverse proxy).
  - onMessageReceived(RemoteMessage remoteMessage) ⇒ această metodă este apelată atunci când aplicația primește o notificare push de la FCM. Aici ar trebui implementată logica pt tratarea mesajului ⇒ afișare mesaj pt user.

datele)

③ EventListenerProvider ⇒ această interfață permite modulului să asculte și să reacționeze la evenimente specifice din Keycloak (ca ar fi login, logout etc.)

→ în acest modul putem să folosim pt a declanșa trimiterea de notificări push pe baza unor evenimente specifice.

④ Protocol mapper ⇒ permite personalizarea tokenurilor JWT emise de keycloak adăugând câmpuri (claims) personalizate.

→ Putem folosi aceasta pt a include info suplimentare în tokenurile JWT care pot fi necesare aplicației tale, ca ar fi statusul activării notif. push pt user.

Trimiterea tokenului la Serverul Keycloak

→ Keycloak nu are un endpoint implicit pt gestionarea tokenurilor FCM, trb implementată o logică personalizată pt aceasta.

↳ crearea unui endpoint personalizat prin SPI

↳ folosire pt stocare secret (token) interfața VaultProviderFactory

↓

custom vault

→ trimiterea de la client (app) la server a tokenului ⇒ HTTPS și endpoint protejat împotriva accesului neautorizat.

FCM

Când utilizăm FCM în aplicația Android, sistemul FCM generează un token unic pt fiecare dispozitiv pe care este instalată aplicația. Acest token este esențial pt a identifica dispozitivul în

mod unic și pt a permite trimiterea de mesaje push către acel disp.

Cum funcționează?

① Integrarea FCM în App Android ⇒ configurare serviciu FCM conform docum. Firebase, sistemul FCM este pregătit să genereze tokenuri pt disp.

② Generare automată a tokenului ⇒ la prima lansare a aplicației pe un dispozitiv (sau în alte cazuri, cum ar fi reinstalarea aplicației, ștergerea datelor aplicației sau reînnoirea tokenului din alte motive), serv. FCM generează un token nou pt disp respectiv.

③ Metoda `onNewToken` ⇒ FCM apelează metoda onNewToken din clasa mea și furnizează noul token ca un string în argumentul metodei. Aceasta înseamnă că FCM "furnizează automat" tokenul (adică eu ca dezvoltator nu mai trb să solicit explicit acest token) ⇒ tot ce trb făcut este să suprascriu metoda pt a opta și gestiona tokenul primit ca argument.