MFA App - Progettazione

Daniel Eduardo Contro

6 agosto 2021

Indice

1 Diagrammi dei Package

Per quanto riguarda i diagrammi di package si è deciso di adottare, rispettando le imposizioni dei framework adottati, il principio *Common Reuse Principle*.
I principali package individuati sono:

- **services**: contiene i tipi per la gestione di servizi esterni all'applicazione;
- **models**: contiene i principali tipi utilizzati all'interno dell'applicazione per il mantenimento dello stato;
- **store**: contiene tutti i tipi che vanno a comporre lo store dell'applicazione, il package ha una profonda dipendenza dal framework di gestione dello stato *Vuex*;
- **modules**: contiene dei sottopackage ciascuno dei quali presenta le *Views* e i *ViewModel* e le *routes* necessarie per le macrofunzionalità dell'applicazione.

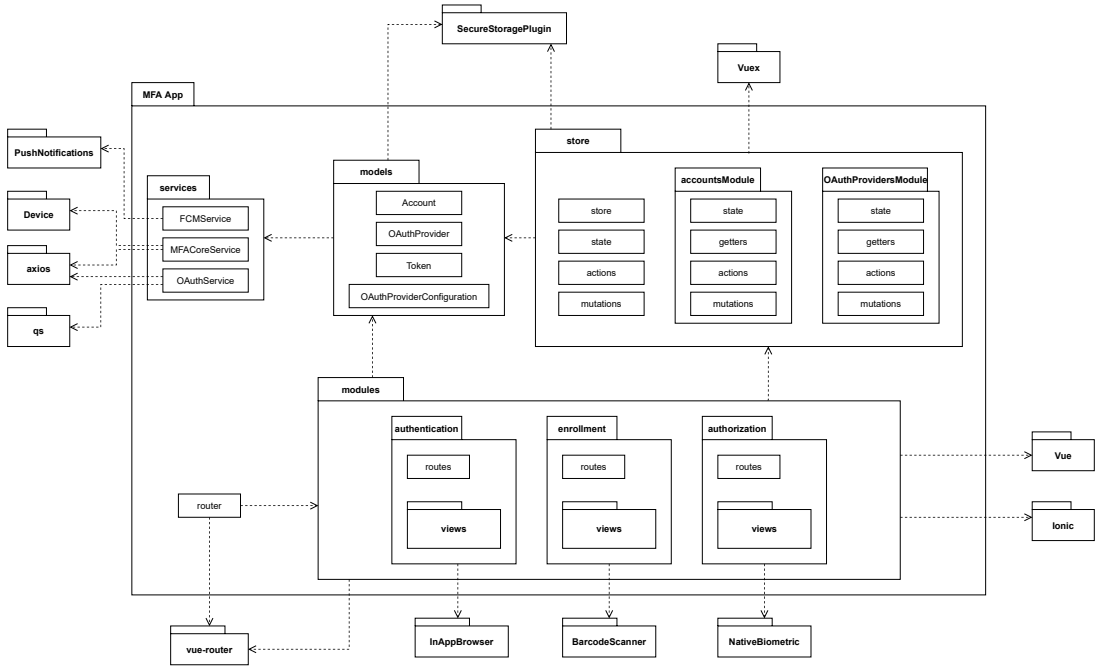


Figura 1: Diagramma di package di MFA App

1.1 Services

I tipi presenti all'interno del package si occupano di gestire la complessità dei sistemi esterni su cui l'applicazione dipende; in particolare:

- **MFACoreService** permette di effettuare le chiamate alle API del backend dell'applicazione;
- **FCMService** permette l'iscrizione e la gestione delle notifiche push basate su firebase;
- **OAuthService** permette di effetuare le diverse chiamate presenti in RFC6749 ad un provider di un servizio OAuth 2.0.

1.2 Models

All'interno del package sono presenti i tipi fondamentali dello stato dell'applicazione ossia **OAuthProvider**, che rappresenta un provider OAuth 2.0, **Account** rappresenta un account disponibile in uno degli *OAuthProvider* registrati. Sono poi presenti ulteriori tipi di supporto per questi principali tra cui i principali sono **Token** e **OAuthProviderConfiguration** che rappresentano rispettivamente un token OAuth 2.0 e la configurazione del provider OAuth 2.0.

1.3 Store

All'interno del package store sono presenti tutti gli oggetti per la gestione dello stato dell'applicazione; in particolare l'oggetto *store* di tipo **Store** di **Vuex** che contiene al suo interno i diversi moduli **OAuthProvidersModule** e **AccountsModule** per la gestione dei rispettivi oggetti.

1.4 Modules

All'interno del package sono presenti diversi sottopackage, ciascuno per una specifica macrofunzionalità dell'applicazione; tra i principali figurano **Enrollment**, **Authorization** e **Authentication**, all'interno dei quali vengono implementate le relative funzionalità.

2 Diagrammi delle classi

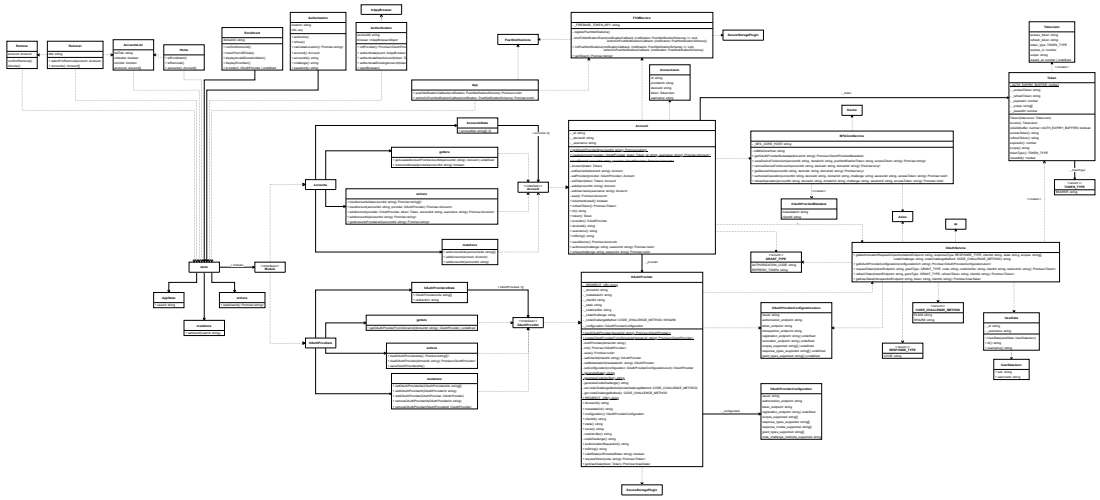
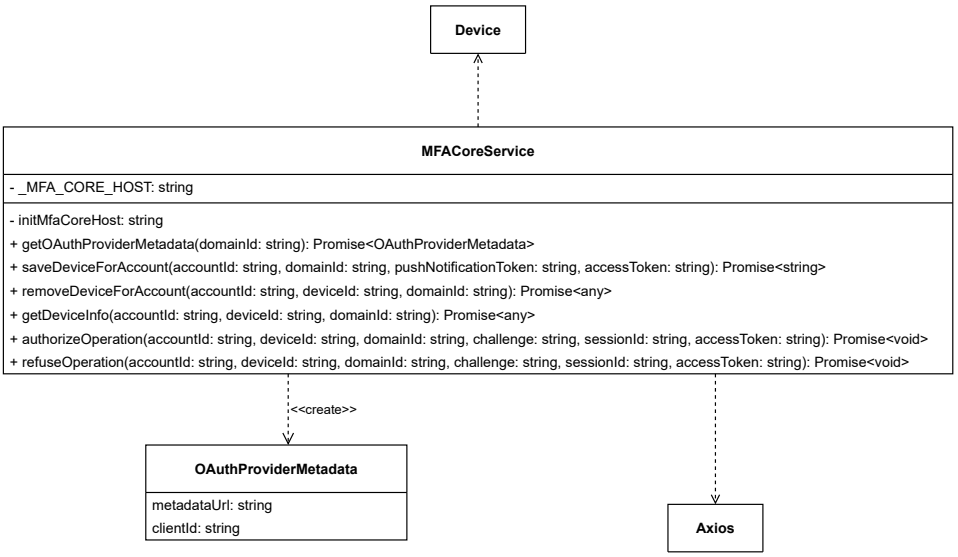


Figura 2: Diagramma delle classi di MFA App

2.1 Services

Di seguito vengono presentati i tipi che implementano e gestiscono la complessità di servizi esterni all'applicazione.

2.1.1 MFACoreService



Descrizione

Il tipo **MFACoreService** ha il compito di gestire le chiamate alle API del backend dell'applicazione quali:

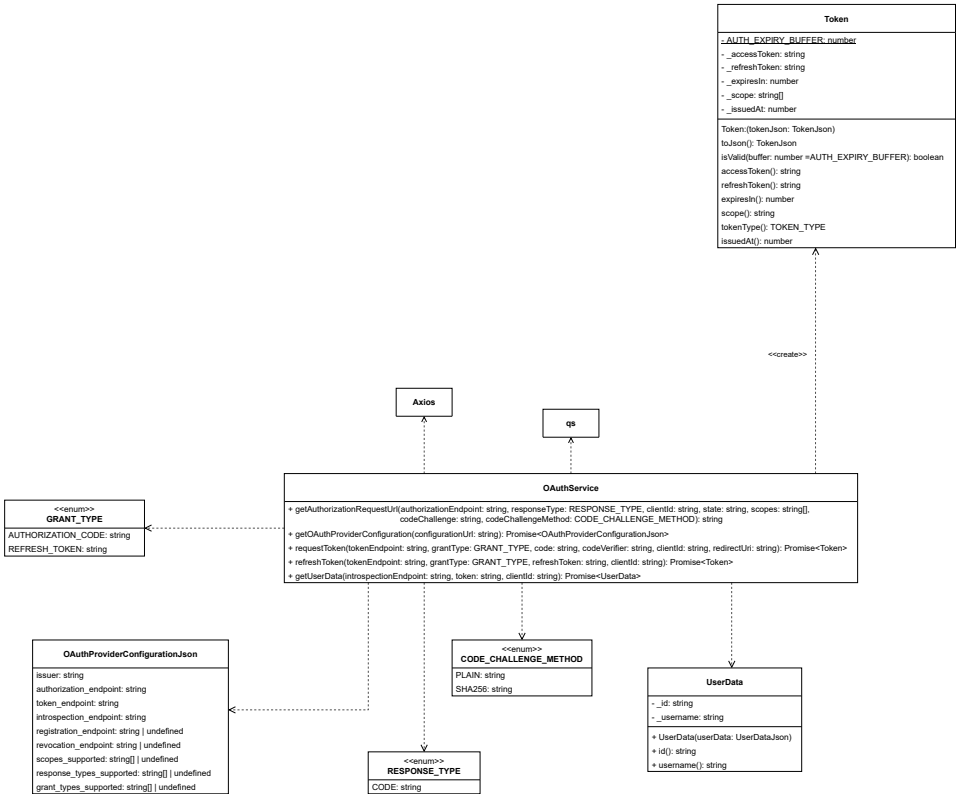
- la richiesta di enrollment di un dispositivo per un account autenticato;
- fornire il consenso ad un'operazione effettuata in uno degli account collegati;

- negare il consenso ad un'operazione effettuata in uno degli account collegati;
- disassociare uno degli account collegati dal dispositivo.

Dipendenze
Il tipo ha le seguenti dipendenze:

- axios;
- Device;
- OAuthProviderMetadata.

2.1.2 OAuthService



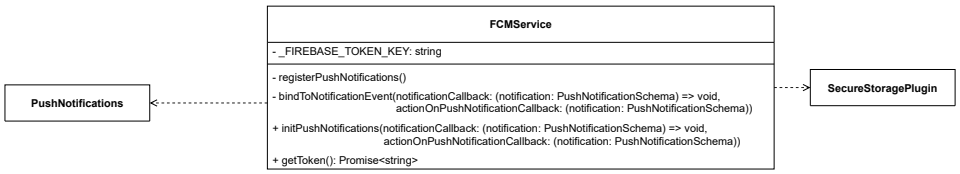
Descrizione
Il tipo **OAuthService** ha il compito di gestire le interazioni dell'applicazioni, la quale implementa un client OAuth 2.0, con il provider OAuth 2.0 degli account collegati; in particolare permette le seguenti funzionalità:

- ottenimento della configurazione del provider OAuth;
- richiesta di un *authorization code* con il grant_type PKCE;
- richiesta di un access_token con il grant_type PKCE;
- richiesta di refresh dell'access_token;
- ottenimento delle informazioni relative ad un account autenticato.

Dipendenze
Il tipo ha le seguenti dipendenze:

- axios;
- qs;
- OAuthProviderConfigurationJson;
- Token;
- TokenJson;
- RESPONSE_TYPE;
- GRANT_TYPE;
- CODE_CHALLENGE_METHOD;
- UserDataJson;
- UserData.

2.1.3 FCMService



Descrizione
Il tipo **FCMService** si occupa di gestire nella maniera opportuna le notifiche push ricevute dal servizio *Firebase Clod Messaging*; in particolare permette le seguenti funzionalità:

- registrazione al servizio di notifica;
- ricezione delle notifiche push;
- visualizzazione delle notifiche ricevute.

Dipendenze
Il tipo ha le seguenti dipendenze:

- PushNotifications;
- SecureStoragePlugin.

2.2 Models

2.2.1 OAuthProvider

Descrizione

Il tipo **OAuthProvider** permette la gestione e il salvataggio delle informazioni di un provider OAuth 2.0.

Dipendenze

Il tipo ha le seguenti dipendenze:

- SecureStoragePlugin;
- OAuthProviderConfiguration;

- OAuthProviderConfigurationJson;
- OAuthService.

2.2.2 Account

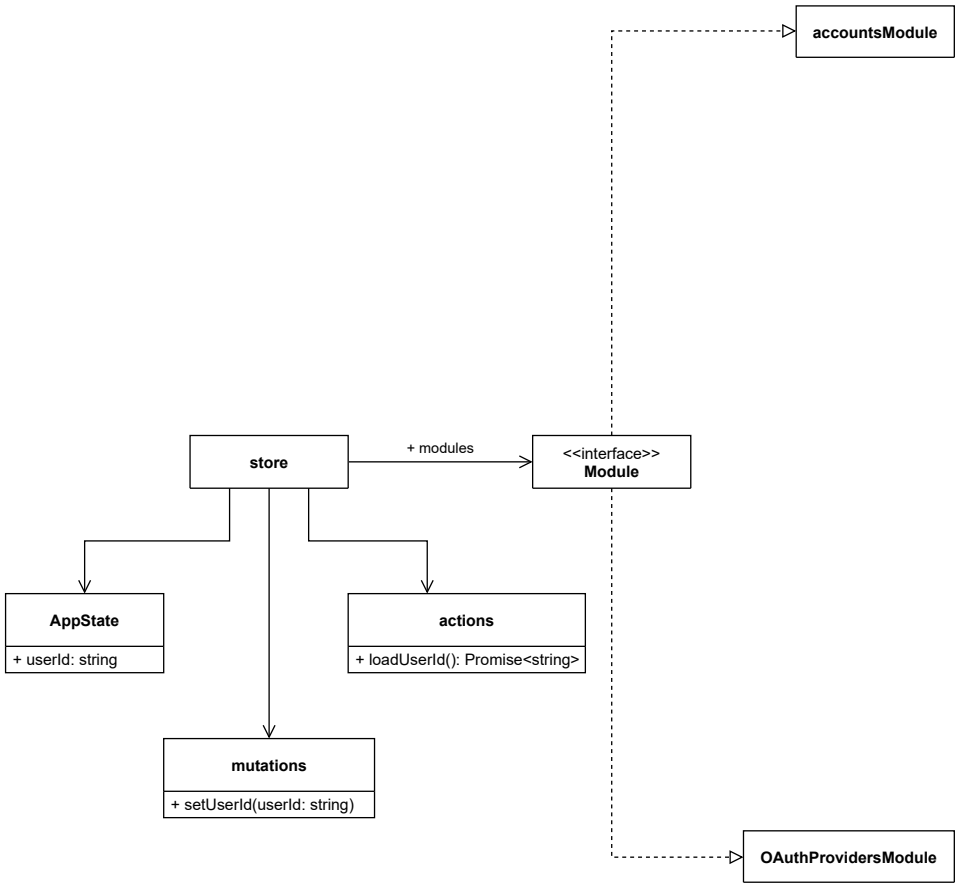
Account
- _id: string - _deviceId: string - _username: string
+ getOAuthProviderId(accountId: string): Promise<string> + createAccount(provider: OAuthProvider, token: Token, id: string, username: string): Promise<Account> + loadFromId(accountId: string, provider: OAuthProvider): Promise<Account> - Account(token: Token) - setDeviceId(deviceId: string): Account - setProvider(provider: OAuthProvider): Account - setToken(token: Token): Account - setId(accountId: string): Account - setUsername(username: string): Account - save(): Promise<Account> + isAuthenticated(): boolean + refreshToken(): Promise<Token> + id(): string + token(): Token + provider(): OAuthProvider + deviceId(): string + username(): string + toString(): string + saveDevice(): Promise<Account> + authorize(challenge: string, sessionId: string): Promise<void> + refuse(challenge: string, sessionId: string): Promise<void>

Descrizione
Il tipo **Account** rappresenta un account di un provider OAuth e permette la gestione delle sue informazioni e delle operazioni.

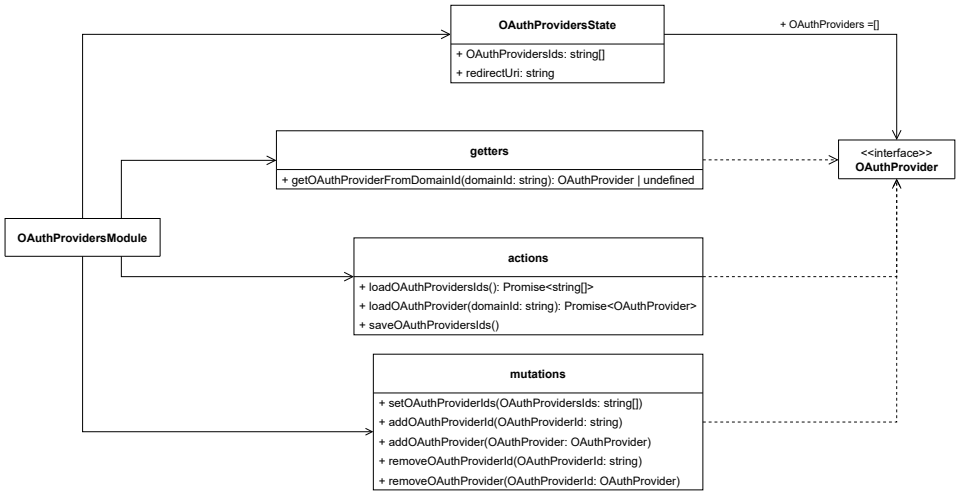
Dipendenze
Il tipo ha le seguenti dipendenze:

- FCMService;
- MFACoreService;
- OAuthService;
- AccountJson;
- Token;
- OAuthProvider.

2.3 Store



2.3.1 OAuthProvidersModule



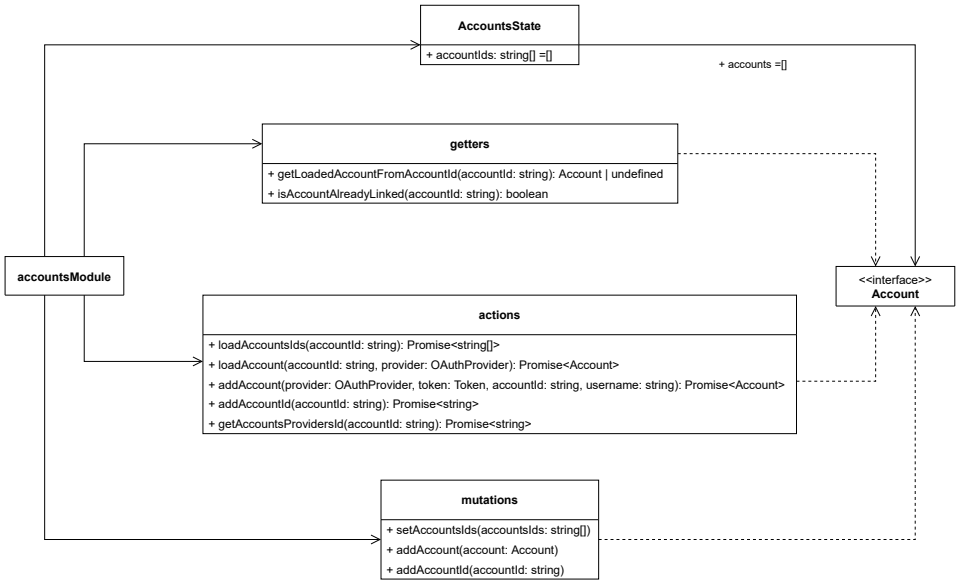
Descrizione

Il modulo **OAuthProvidersModule** si occupa della persistenza, del caricamento, del salvataggio e della gestione runtime dei providers OAuth nel dispositivo.

Dipendenze
Il tipo ha le seguenti dipendenze:

- OAuthProvider;
- SecureStoragePlugin;

2.3.2 AccountsModule



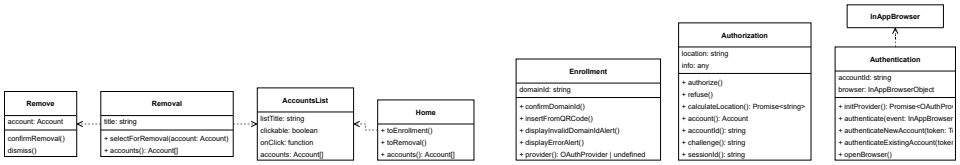
Descrizione
Il modulo **AccountsModule** si occupa della persistenza, del caricamento del salvataggio e della gestione runtime degli account collegati nel dispositivo.

Dipendenze

Il tipo ha le seguenti dipendenze:

- Account;
- OAuthProvider;
- SecureStoragePlugin;

2.4 Modules



Tra i moduli figurano diverse componenti Vue, ciascuna delle quali ha un compito ben preciso.

Home:
Il componente **Home** rappresenta la pagina principale dell'applicazione.

Enrollment:
Il componente **Enrollment** rappresenta la pagina per il salvataggio di un provider OAuth 2.0.

Authentication:
Il componente **Authentication** rappresenta la pagina per l'autenticazione dell'account desiderato ed il conseguente.

Authorization:
Il componente **Authorization** rappresenta la pagina per fornire il consenso ad un'operazione svolta in uno degli account collegati.

Removal:
Il componente **Removal** rappresenta la pagina per la rimozione di uno degli account collegati.

Remove:
Il componente **Remove** rappresenta la pagina per la conferma della rimozione di uno degli account collegati.

AccountsList:
Il componente **AccountsList** rappresenta un componente di utility per la visualizzazione della lista degli account collegati con il dispositivo.