

Bell Music

Documentazione



PROGRAMMAZIONE DI DISPOSITIVI MOBILI

Università degli Studi di Milano-Bicocca

Progetto a cura di:

Stefano Yecheng Hu 870084

INDICE

1. FUNZIONALITÀ	3
1.1 Tecnologie e linguaggi utilizzati	4
1.2 Flutter: pacchetti principali utilizzati (https://pub.dev/)	4
2. ARCHITETTURA.....	5
2.1 View (User Interface)	5
2.2 ViewModels	6
2.3 Servizi esterni.....	6
2.3.1 Firebase Authentication	6
2.3.2 Firebase Cloud Storage.....	7
2.3.3 Python RESTful API	7
3. DESIGN	8
3.1 Schema	8
3.2 Schema con UI	9
3.3 Spiegazione pagine.....	10
4. ESEMPI UTENTI	15

1. FUNZIONALITÀ

Bell Music permette di ascoltare musica attraverso l'utilizzo di YouTube. L'esperienza è priva di pubblicità assillanti.

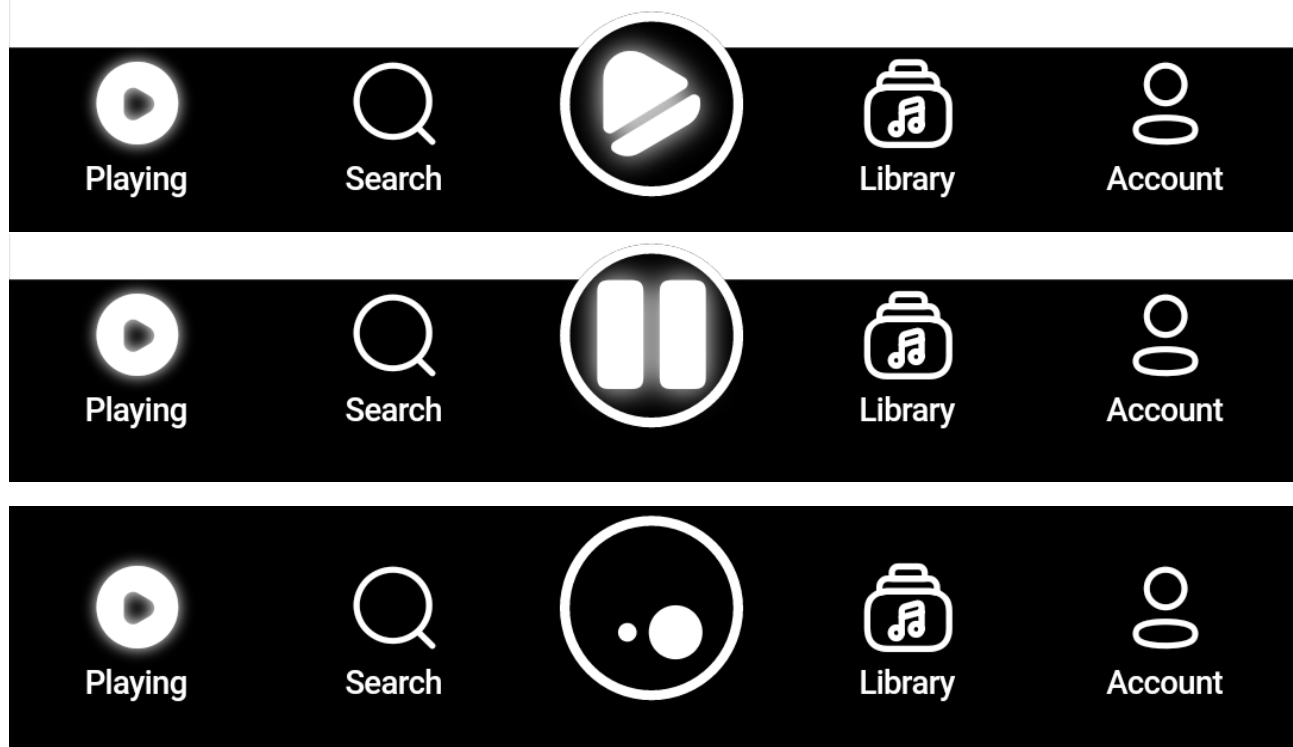
Visto l'impossibilità di utilizzare un API ufficiale di YouTube Music a causa della sua non esistenza, l'applicazione usa un modulo Python di terze parti

(<https://github.com/sigma67/ytmusicapi>) che fornisce le stesse informazioni di un API ufficiale, ma sotto forma di dictionaries.










L'app ha 4 tab:

1. **Playing:** mostra il media che si sta riproducendo al momento.
2. **Search:** permette di cercare elementi all'interno di YouTube Music con la possibilità di filtrare a seconda di cosa si vuole cercare.
3. **Library:** mostra gli elementi salvati all'interno dell'account dell'utente. Essi comprendono: liked songs, playlists, albums, songs, artists, subscriptions.
4. **Account:** mostra i dati dell'utente e permette il logout tramite l'apposito tasto.

Oltre ai 4 tab descritti precedentemente è presente un tasto centrale che racchiude il play, pause e, quando necessario, il cerchio di caricamento.



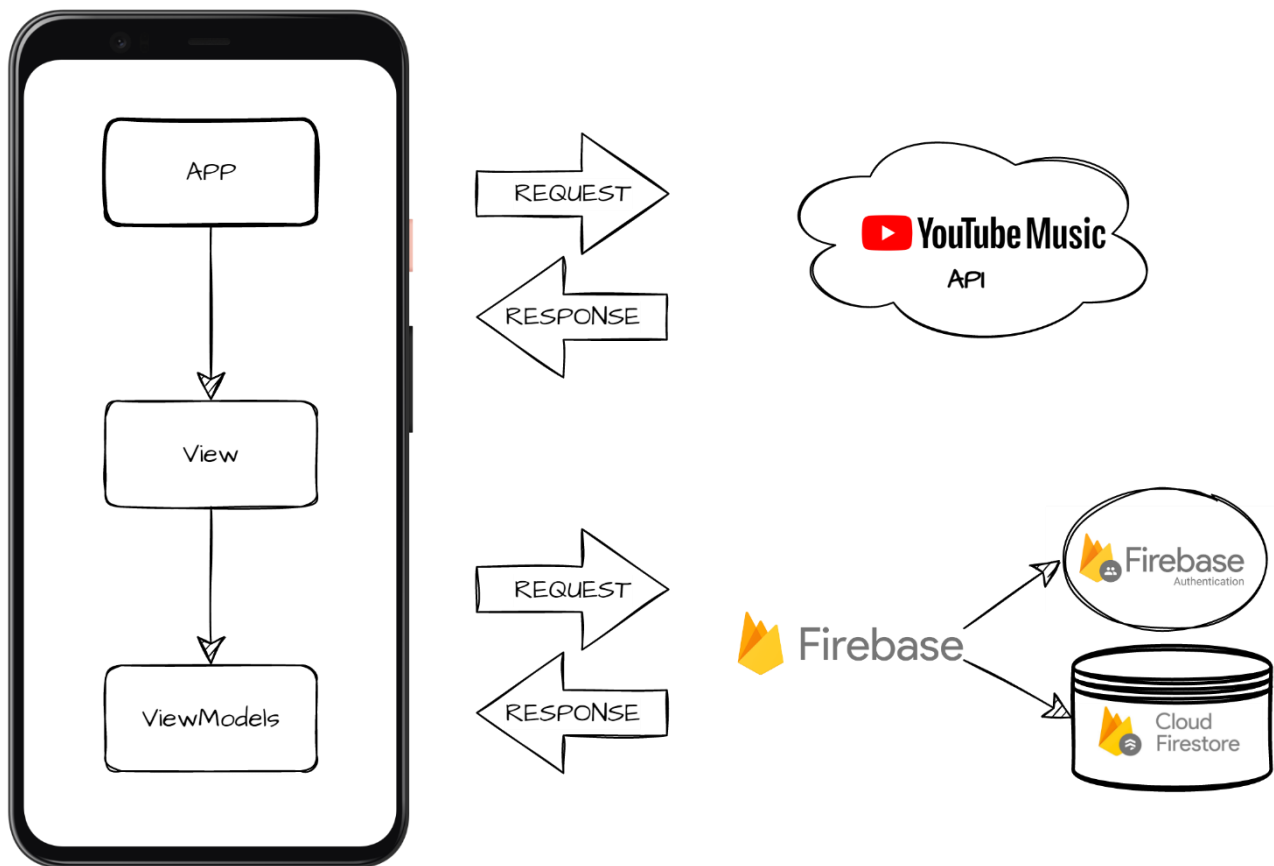
1.1 Tecnologie e linguaggi utilizzati

- **Framework:** Flutter 
- **Linguaggi:** Python  , Dart 
- **Database:** Firebase Cloud Store  Cloud Firestore
- **Autenticazione:** Firebase Authentication 
- **“IDE”:** Visual Studio Code 
- **Emulatore:** Android Studio 
- **Python:** Flask RESTful Flask  , ytmusicapi  , yt-dlp

1.2 Flutter: pacchetti principali utilizzati (<https://pub.dev/>)

- **Just-Audio:** gestione degli audio (https://pub.dev/packages/just_audio)
- **Dio:** richieste Http (<https://pub.dev/packages/dio>)
- **Provider:** condivisione delle risorse (<https://pub.dev/packages/provider>)

2. ARCHITETTURA



L'applicazione è costruita utilizzando il framework Flutter che può essere utilizzato per costruire app cross platform.

Essa utilizza il pattern architetturale MVVM. Quindi, le pagine che hanno la necessità di effettuare azioni al di fuori della costruzione della UI, per esempio richieste Http all'API, sono divise in due file differenti. Il primo rappresentante la View e il secondo il ViewModel.

Tutte le pagine di una applicazione costruita con Flutter sono una combinazione di Widget che possono essere Stateful o Stateless a seconda della necessità di cambiare lo stato della pagina. Quindi, i concetti di Fragment e Activity delle applicazioni native vengono mantenuti ma sottoforma di Widgets.

2.1 View (User Interface)

La User Interface è composta da varie Views dove è presente il codice che compone gli elementi grafici. Le Views implementate sono 9 pagine comprendenti sia i 4 tab principali che le pagine secondarie.

I tab principali sono:

- Media

- Search
- Library
- Account

Le pagine secondarie sono:

- Authenticate
- Album/Playlist
- Artist
- Artist Albums
- Library Content

2.2 ViewModels

I ViewModels contengono tutta la logica dietro la UI. Essi sono:

- AAPViewModel: condiviso da Artist e AlbumPlaylist perché le richieste Http sono molto simili. Questo ViewModel permette anche di ottenere gli album di un artista e di segnare un album/playlist come *liked/indifferent*.
- LibraryViewModel: permette di ottenere i contenuti della libreria.
- MediaViewModel: ottiene tutti i dati riguardanti il media in riproduzione tramite richieste Http. Può anche segnare un media come *liked/indifferent*.
- SearchViewModel: effettua ricerche e ritorna i risultati tramite richieste Http.

Tre ulteriori file degni di nota sono “database.dart”, “auth.dart” e “screen_navigator.dart”.

- Il primo racchiude tutti i metodi per il read/write dei dati su Firestore.
- Il secondo contiene tutti i metodi riguardanti il login e il logout.
- Il terzo contiene tutti i metodi per la navigazione tra le diverse pagine.

2.3 Servizi esterni

I servizi esterni utilizzati sono:

- Firebase Authentication per il login.
- Firebase Cloud Storage per il salvataggio dei dati.
- REST API locale scritto in Python utilizzando Flask-RESTful.

2.3.1 Firebase Authentication

L'applicazione utilizza Firebase Authentication per la parte di login e si affida alla presenza di un account Google. Questa scelta è stata fatta perché c'è la necessità di possedere un account YouTube che appartiene alla stessa azienda e non c'è bisogno di creare un nuovo account.

2.3.2 Firebase Cloud Storage

L'utilizzo di Firebase Cloud Storage serve per compensare tutte le funzionalità non fornite da ytmusicapi come il salvataggio della cronologia delle ricerche, il link dell'audio generato da yt-dlp e la queue della playlist in riproduzione.

Su Firestore è presente una sola collection chiamata *users* e al suo interno sono presenti dei documenti. Questi ultimi sono contrassegnati da un uid creato da Firebase Authentication durante la registrazione e assegnato ad ogni singolo utente.

Ogni documento ha al suo interno i dati citati precedentemente:

- **nowPlaying**
- **queue**
- **searchHistory**

2.3.3 Python RESTful API

Il server utilizza due moduli fondamentali oltre a Flask-RESTful:

1. ytmusicapi: <https://github.com/sigma67/ytmusicapi>
2. yt-dlp: <https://github.com/yt-dlp/yt-dlp>

Il primo serve ad ottenere dati da YouTube Music ritornandoli sottoforma di dictionaries e funziona come API di terze parti.

Il secondo serve per ottenere l'URL dell'audio a partire da un link YouTube.

L'applicazione Flutter invia richieste Http GET/POST all'API locale per ottenere i dati ottenuti dai due moduli elencati in precedenza.

Tuttavia, lo svantaggio di utilizzare un API di terze parti come *ytmusicapi* è che non c'è modo di ottenere una autenticazione automatica, ma bisogna fornire manualmente il cookie di autorizzazione. Questa azione ha i suoi svantaggi, però è l'unica opzione disponibile finché Google non fornisce un API ufficiale per YouTube Music.

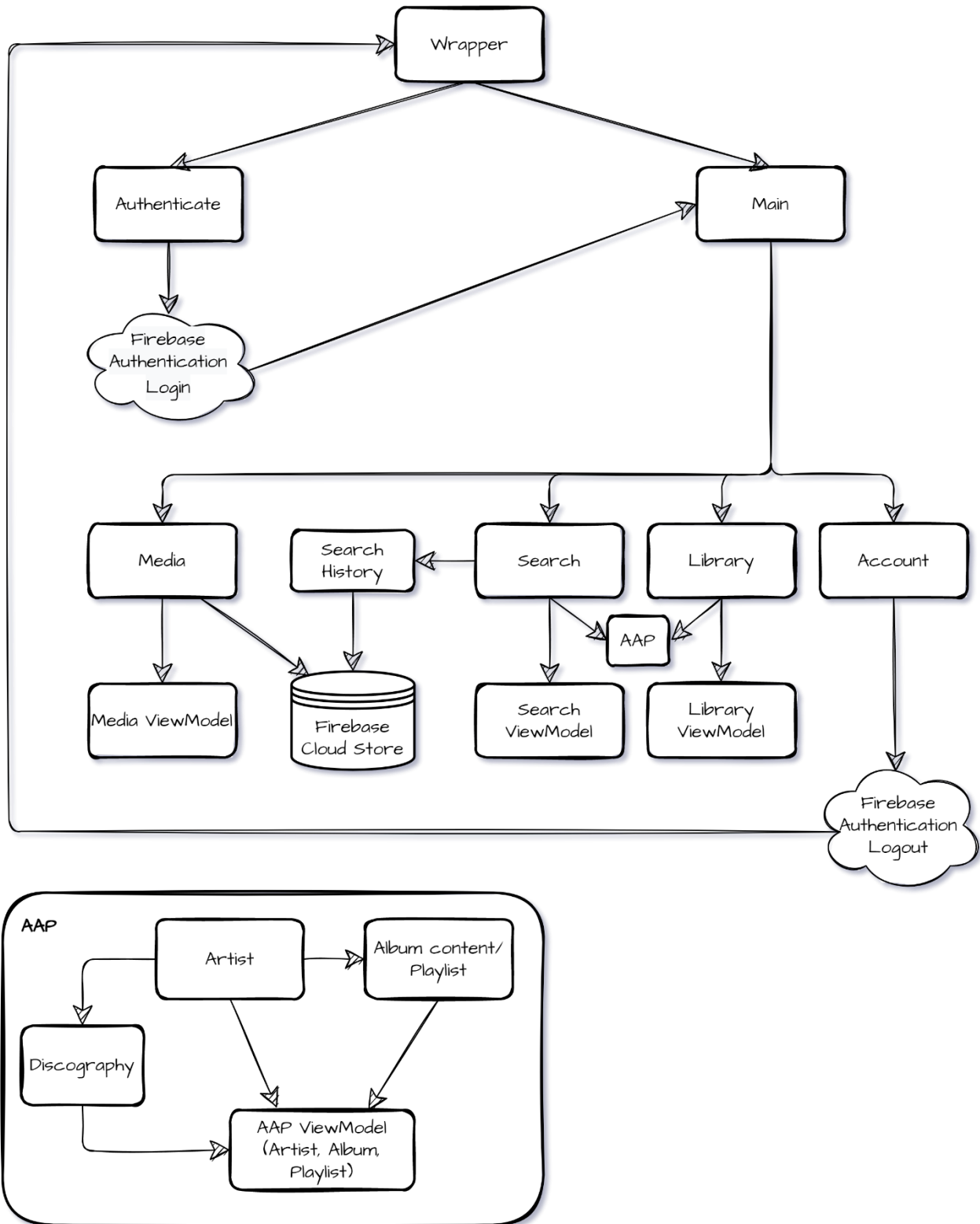
Detto questo, per accedere a tutte le funzionalità dell'applicazione, c'è la necessità di aggiungere il file `headers_auth.json` purché ytmusicapi funzioni correttamente. Questo file si può ottenere seguendo le istruzioni riportate sulla documentazione di ytmusicapi al link: “<https://ytmusicapi.readthedocs.io/en/latest/setup.html>”.

Attualmente il file si trova all'interno di “lib/python”.

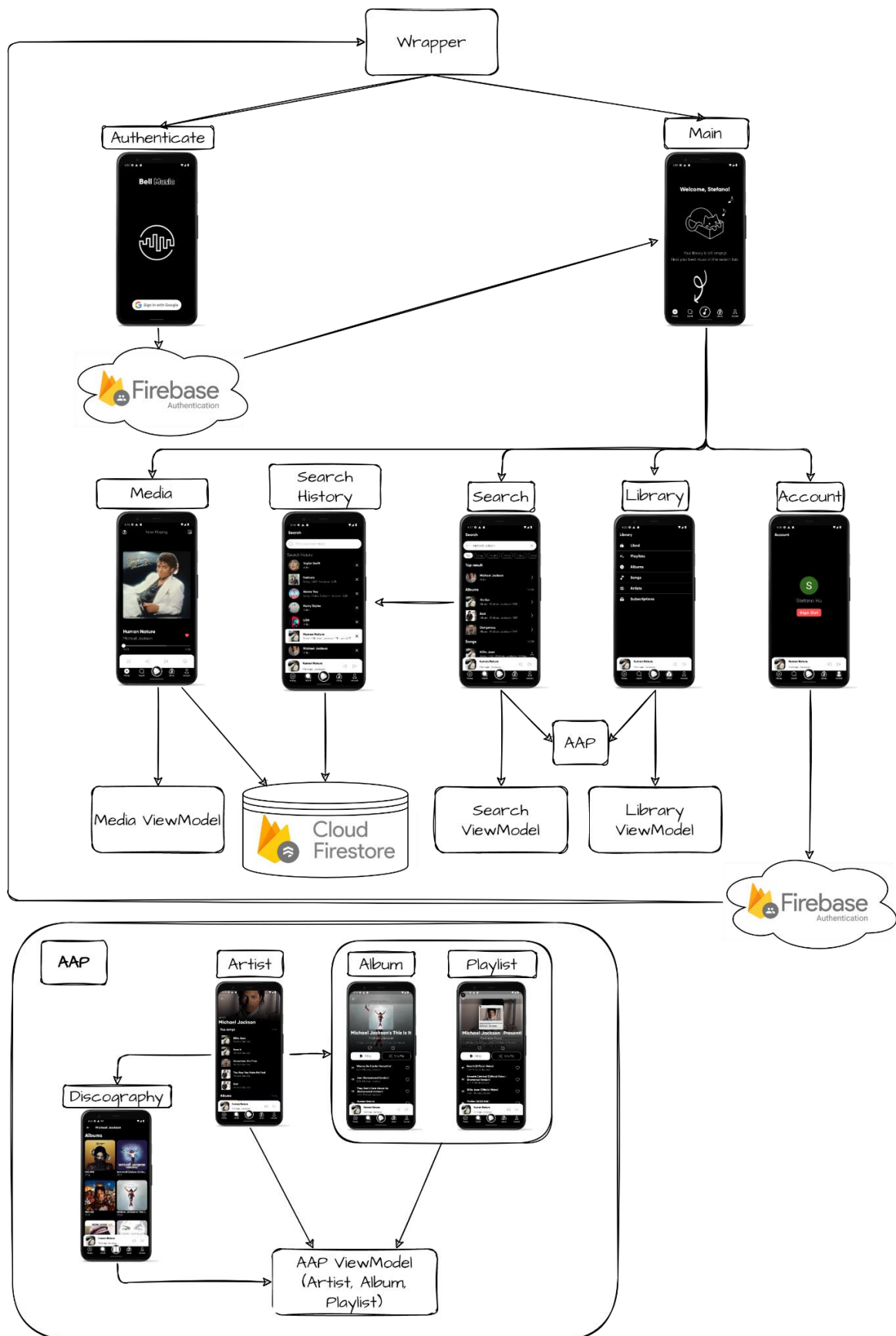
La mancanza di esso può generare errori.

3. DESIGN

3.1 Schema



3.2 Schema con UI



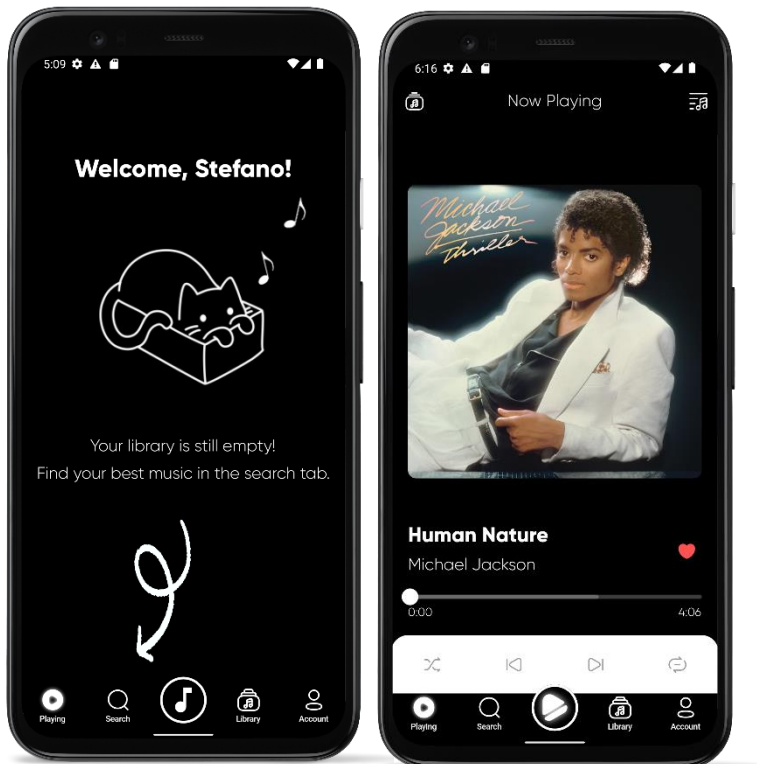
3.3 Spiegazione pagine

- **Wrapper:** controlla la presenza di un account loggato e reindirizza ad *Authenticate* se lo user è null e a *Main* se non è null.

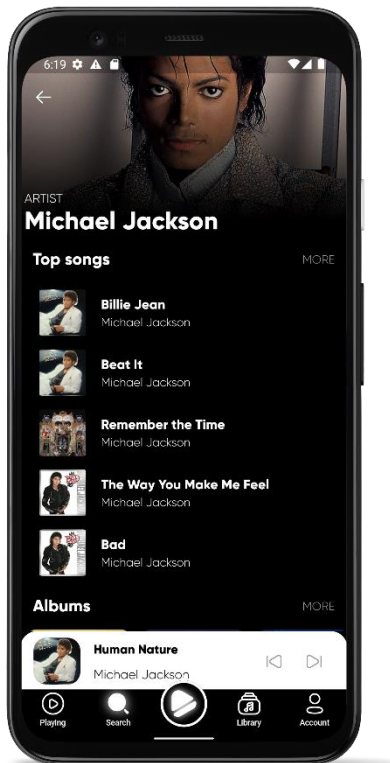


- **Authenticate:** mostra un tasto per fare il login con Google. Il login è gestito da Firebase Authentication e genera un uid che viene utilizzato su Firebase Cloud Store per dividere i dati degli utenti. Se non si possiede un account Google, il popup permette di creare un nuovo account.

- **Main:** contiene tutti i tab dell'applicazione e assegna ad essi le routes possibili.



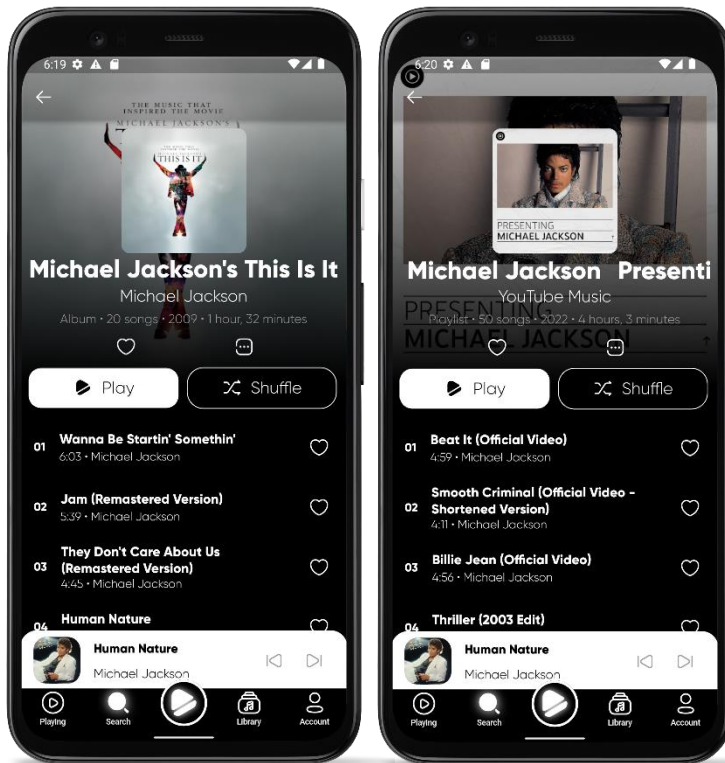
- **Media:** mostra il media in riproduzione e salva nel database i suoi dati insieme alla queue in corso. È possibile visualizzare la queue in riproduzione e le lyrics della canzone se sono disponibili attraverso i tasti appositi. Se non ci sono media in riproduzione, si visualizza una pagina che incita alla ricerca di qualcosa da riprodurre utilizzando il tab *Search*.



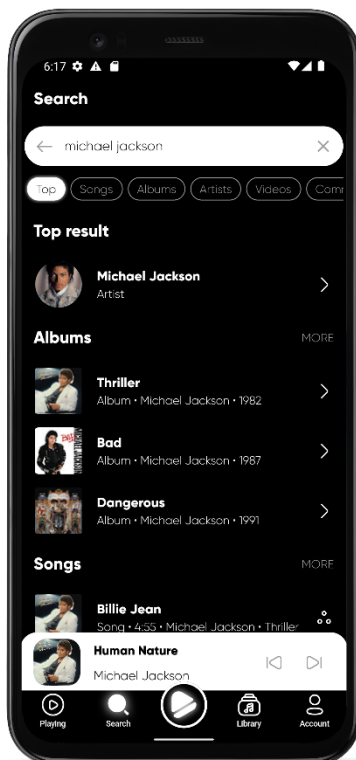
- **Artist:** mostra i dati dell'artista. È possibile navigare a delle pagine secondarie rappresentanti tutta la discografia.



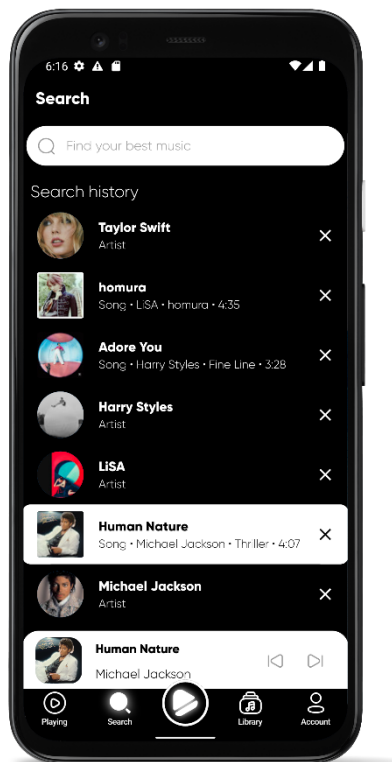
- **Discography:** mostra delle pagine secondarie dell'artista. Esse comprendono una lista degli album, dei singoli, delle canzoni e dei video.



- **Album/Playlist:** mostra le canzoni associate ad un album o una playlist.

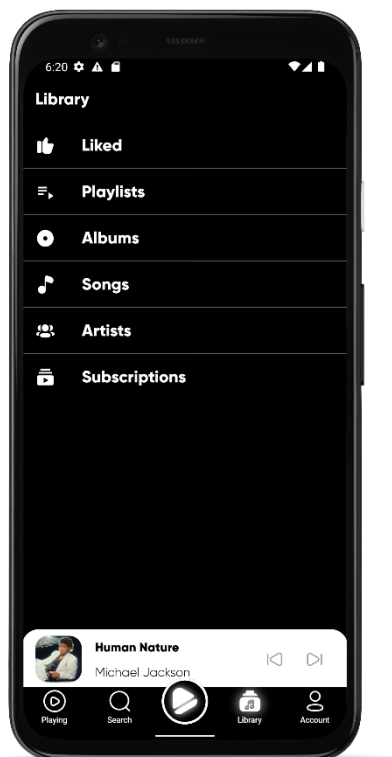


- **Search:** mostra la barra di ricerca che può essere utilizzata per cercare canzoni. Inoltre, sono presenti dei tasti per filtrare le ricerche.

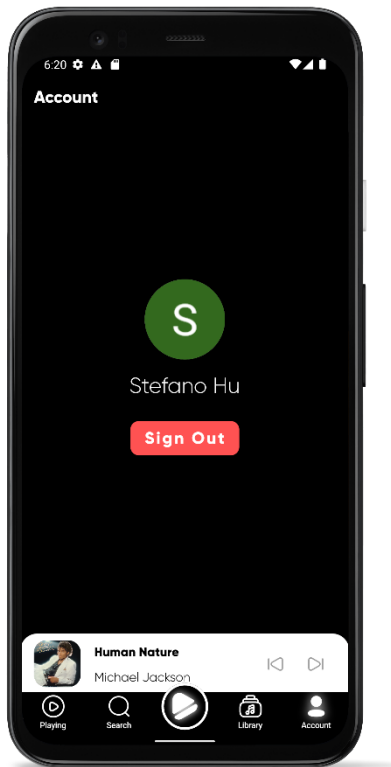


- **Search History:** se non si è effettuata una ricerca, allora viene mostrata la cronologia delle ricerche. Esse sono salvate all'interno di Firestore e possono essere eliminate tramite l'apposita icona.

Una nota da ricordare è che solo gli elementi cliccati delle ricerche precedenti vengono salvati.



- **Library:** mostra le diverse librerie generate per l'account in questione. Il tutto è gestito da YouTube e non ci sono elementi salvati su Firestore. Le librerie disponibili sono:
 - Liked
 - Playlists
 - Albums
 - Songs
 - Artists
 - Subscriptions



- **Account:** mostra l'immagine profilo e il nome dell'account loggato. È presente un tasto per effettuare il logout.

4. ESEMPI UTENTI

