ALMA MATER STUDIORUM - UNIVERSITA' DI BOLOGNA

CORSO DI LAUREA IN INFORMATICA PER IL MANAGEMENT

ANNO ACCADEMICO 2023/2024

DOCUMENTAZIONE PROGETTO LABORATORIO APPLICAZIONI MOBILI

"Your City is Listening To"

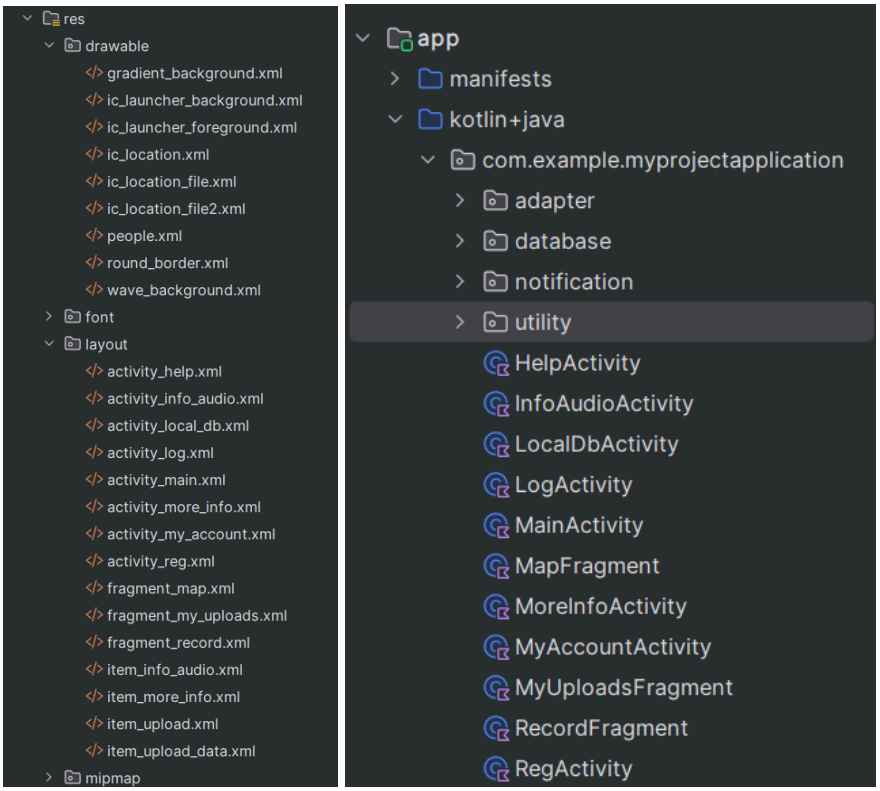
Lorenzo Maini

Matricola 0001020595
lorenzo.maini2@studio.unibo.it

IMPLEMENTAZIONE:

Il progetto consiste nello sviluppo di un'applicazione Android nativa, realizzata utilizzando il linguaggio di programmazione Kotlin.
Per la gestione delle chiamate API e l'interazione con il server, è stata impiegata la libreria Retrofit, che facilita la comunicazione con i servizi web RESTful, permettendo di inviare richieste HTTP e di ricevere risposte in modo semplice ed efficiente.
La persistenza dei dati locali è stata gestita attraverso Room, una libreria di gestione del database, che consente di creare e gestire il database SQLite in modo sicuro e ottimizzato.
L'intero sviluppo del progetto è stato documentato e aggiornato su una repository GitHub. La repository, inizialmente privata durante le fasi di sviluppo, è stata successivamente resa pubblica. Questa repository contiene il codice sorgente, la documentazione e tutte le risorse necessarie per comprendere il progetto.

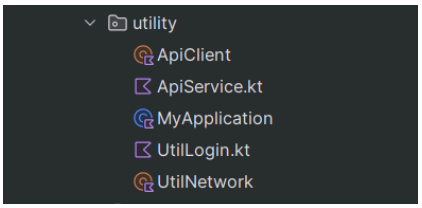
OVERVIEW DIRECTORY:



Il progetto è diviso nelle cartelle mostrate nello screen soprastante:
In “myprojectapplication” sono presenti i codici di tutte le activity ed i fragment che vanno a comporre l’applicazione. All’interno della cartella "Utility" sono presenti alcune classi con metodi utili a diverse parti del progetto. Nella directory “Notification” sono presenti due classi con all’interno la logica per creare le notifiche. Nella cartella “Database” sono presenti le

classi per la creazione e la gestione del database. Infine, nella cartella “Adapter” ci sono i codici dei vari adapter utili per le recycler view generate. Inoltre gli aspetti grafici principali sono curati in drawable e in layout.

UTILITY



ApiClient
Questo codice crea un singleton che fornisce un'istanza di Retrofit configurata per effettuare richieste HTTP ad un server specifico. Contiene l'URL di base del server API. L'istanza di Retrofit può essere utilizzata per creare servizi API che definiscono le chiamate HTTP da fare al server.

ApiService
Questo codice definisce una serie di classi dati e un'interfaccia per interagire con l’API tramite Retrofit. SignUpRequest rappresenta i dati di una richiesta di registrazione, SignUpResponse: Rappresenta i dati di risposta di una registrazione, TokenResponse: Rappresenta la risposta contenente un token di autenticazione, DeleteResponse: Rappresenta la risposta di una richiesta di cancellazione di un utente, ResponseMyUploads: Rappresenta un file caricato dall'utente, ResponseAllUploads: Rappresenta un file caricato da qualsiasi utente, ResponseMoreInfo: Rappresenta i dettagli di un file specifico, Tags: Rappresenta i tag di un file, come BPM e mood, ResponseHideFile, ResponseShowFile, ResponseDeleteFile: Rappresentano le risposte per nascondere, mostrare e cancellare un file, ResponseUpload: Rappresenta la risposta di un caricamento di un file. Infine l’Interfaccia ApiService definisce i metodi per le chiamate HTTP.

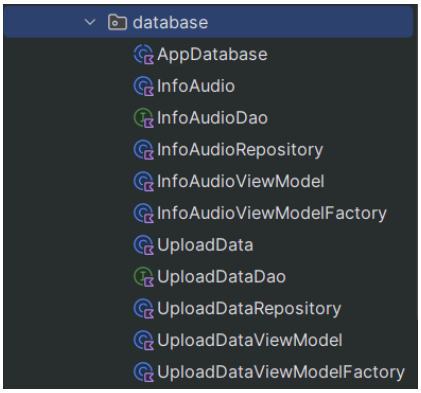
MyApplication
Questo codice Kotlin estende la classe Application di Android. Questa classe viene utilizzata per configurare l'applicazione al momento della sua creazione. In particolare, questo codice si concentra sulla creazione di un canale di notifica per le notifiche relative allo stato del WiFi, utile per fare l’upload anche in background.

UtilLogin
UtilLogin include un metodo forceLogin che accetta un parametro context. Questo metodo crea un Intent per avviare LogActivity, l'attività che gestisce il processo di login. Utilizzando il contesto fornito, il metodo avvia LogActivity forzando così l'utente a effettuare il login. Viene utilizzato in tutte le chiamate api nel caso in cui il token sia scaduto.

UtilNetwork
Questo codice definisce un oggetto Kotlin chiamato UtilNetwork, che contiene tre metodi per gestire la connettività di rete in un'applicazione Android. isNetworkAvailable è un metodo

che verifica se c'è una connessione internet disponibile. Utilizza il ConnectivityManager per ottenere le capacità di rete attive e controlla se la rete dispone della capacità di connessione a internet. checkConnection controlla se la connessione internet è disponibile utilizzando isNetworkAvailable. Se non c'è connessione, mostra un messaggio di avviso (Toast) e reindirizza l'utente a InfoAudioActivity, presumibilmente un'attività che fornisce informazioni audio offline. isWifiConnected controlla se il dispositivo è connesso a una rete WiFi. Anche qui utilizza il ConnectivityManager per ottenere le capacità di rete attive e verifica se la connessione avviene tramite WiFi.

DATABASE



In questa cartella viene definito il database che utilizza l'applicazione. E' formato da due entità: InfoAudio, che contiene le informazioni restituite dal backend dopo l'invio di un file audio registrato tramite il dispositivo che si sta utilizzando, e UploadData, che contiene l'username e le coordinate dei file audio che devono ancora essere caricati tramite l'api.

AppDatabase
AppDatabase gestisce la creazione e l'accesso al database Room, fornendo metodi per ottenere i DAO necessari per le operazioni di accesso ai dati. La classe è annotata con @Database, specificando le entità incluse e la versione del database. È una classe astratta che estende RoomDatabase. Include due metodi astratti che forniscono l'accesso ai DAO: infoAudioDao e uploadDataDao. Utilizza un pattern singleton per garantire che esista solo un'istanza di AppDatabase nell'applicazione.

InfoAudio
Questa classe rappresenta un'entità di database con la tabella chiamata infoAudio. Ogni record contiene informazioni audio, come latitudine, longitudine, BPM, danceability, loudness, mood, genere, strumento e il percorso del file audio. L'ID è la chiave primaria ed è generato automaticamente.

InfoAudioDao
È un'interfaccia che definisce le operazioni di accesso ai dati (DAO - Data Access Object) per la tabella infoAudio. Fornisce metodi per inserire, eliminare, e recuperare tutti i dati dalla tabella.

InfoAudioRepository
Questa classe agisce come un mediatore tra il DAO e il resto dell'applicazione. Gestisce le operazioni di accesso ai dati e offre un'API pulita per il ViewModel. Include metodi per ottenere tutti i dati audio, inserire nuovi dati, ed eliminare dati specifici o tutti i dati.

InfoAudioViewModel
È un ViewModel che gestisce i dati relativi a infoAudio per l'interfaccia utente. Utilizza il repository per eseguire operazioni di accesso ai dati. I metodi includono l'inserimento di dati, l'eliminazione di dati specifici o di tutti i dati, e l'ottenimento di tutti i dati.

InfoAudioViewModelFactory
Questa classe è la factory per creare istanze di InfoAudioViewModel. Viene utilizzata per inizializzare il ViewModel con un'istanza di InfoAudioRepository, garantendo che il ViewModel abbia accesso al repository necessario per le operazioni di accesso ai dati.

UploadData
Questa classe rappresenta un'entità di database con la tabella chiamata uploadData. Ogni record ha un nome utente, una latitudine e una longitudine come chiavi primarie, utilizzati per identificare univocamente un caricamento.

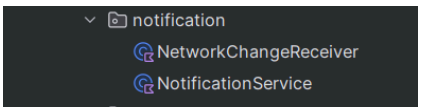
UploadDataDao
È un'interfaccia che definisce le operazioni di accesso ai dati (DAO - Data Access Object) per la tabella uploadData. Fornisce metodi per inserire, eliminare, e recuperare dati dalla tabella, sia in modo asincrono che sincrono.

UploadDataRepository
Questa classe agisce come un mediatore tra il DAO e il resto dell'applicazione. Gestisce le operazioni di accesso ai dati e offre un'API pulita per il ViewModel. Include metodi per ottenere tutti i dati caricati, inserire nuovi dati, eliminare dati specifici o tutti i dati.

UploadDataViewModel
È un ViewModel che gestisce i dati relativi a uploadData per l'interfaccia utente. Utilizza il repository per eseguire operazioni di accesso ai dati. I metodi includono l'inserimento di dati, l'eliminazione di dati specifici, l'eliminazione di tutti i dati e l'ottenimento di dati specifici o di tutti i dati.

UploadDataViewModelFactory
Si tratta della factory per creare istanze di UploadDataViewModel. Viene utilizzata per inizializzare il ViewModel con un'istanza di UploadDataRepository, garantendo che il ViewModel abbia accesso al repository necessario per le operazioni di accesso ai dati.

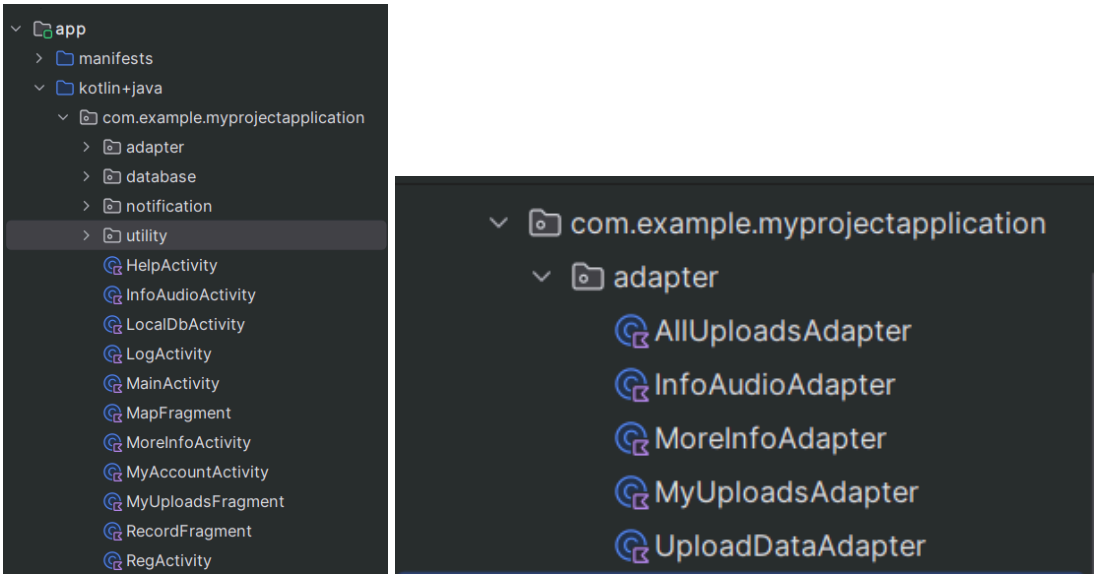
NOTIFICATION



NetworkChangeReceiver
Si tratta di un BroadcastReceiver che reagisce ai cambiamenti di connettività di rete. Quando viene rilevata una connessione WiFi, avvia una coroutine per verificare se ci sono dati da caricare. Se esistono, li carica e invia una notifica all'utente. il metodo *isWifiConnected* verifica se il dispositivo è connesso a una rete WiFi. *checkUploadDataAndNotify* controlla se ci sono dati da caricare nel database, se sì, avvia l'upload. *uploadFromDb* carica tutti i dati di upload presenti nel database e invia una notifica una volta completato. uploadFile carica un singolo file audio sul server utilizzando Retrofit. In caso di successo, inserisce le informazioni audio nel database InfoAudio e elimina i dati di upload dal database. *deleteUploadDataFromDb* elimina i dati di upload dal database UploadData. *insertInfoAudio* Inserisce nel database le informazioni audio ricevute dalla risposta dell'API dopo l'upload. *sendNotification* invia una notifica all'utente per informarlo che i file audio sono stati caricati con successo.

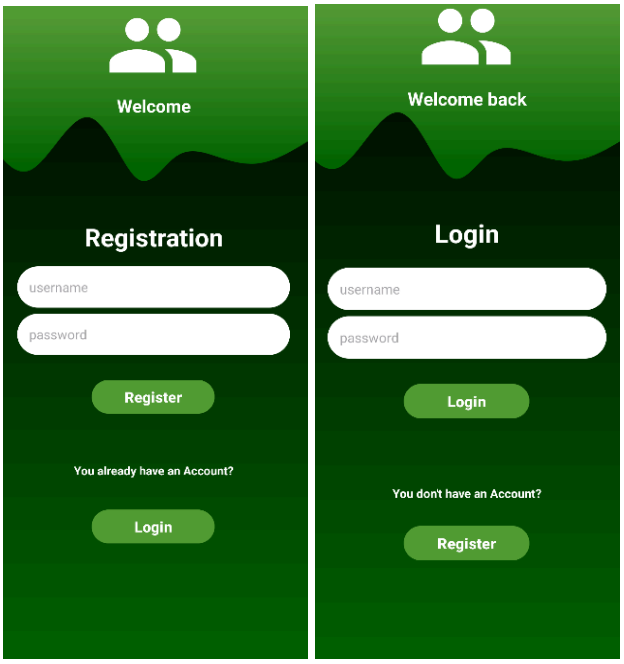
NotificationService
Questa classe è un servizio Android che ascolta i cambiamenti di stato della rete. Quando il servizio viene avviato, ottiene un token client dall'intento e registra un NetworkChangeReceiver per monitorare le modifiche alla connettività di rete. Quando il servizio viene distrutto, al receiver viene tolta la registrazione.

APPLICAZIONE: ACTIVITY, FRAGMENT, E ADAPTER



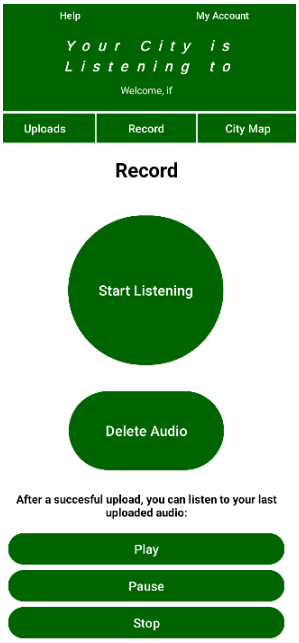
Nella cartella Adapter sono presenti i vari adapter, utili a creare le recycler view. Infine, nella cartella principale si possono trovare tutte le activity ed i fragment dell'applicazione. Di seguito una breve descrizione passo passo dell'uso della applicazione.

LOG ACTIVITY E REG ACTIVITY



Appena l'utente clicca sulla applicazione, viene mostrata la schermata di Login, dove l'utente può inserire username e password per accedere alla schermata principale. Appena viene creata l'activity, viene effettuato un controllo sulla connessione internet, e in caso quest'ultima sia assente, si viene reindirizzati all'activity InfoAudio, dove è possibile visualizzare i dati e ascoltare le tracce salvate sul database locale del dispositivo. Questo controllo viene effettuato più volte alla creazione delle varie activity e non solo. E' anche possibile registrare un nuovo account in caso l'utente attuale ne sia sprovvisto o voglia crearne un altro.

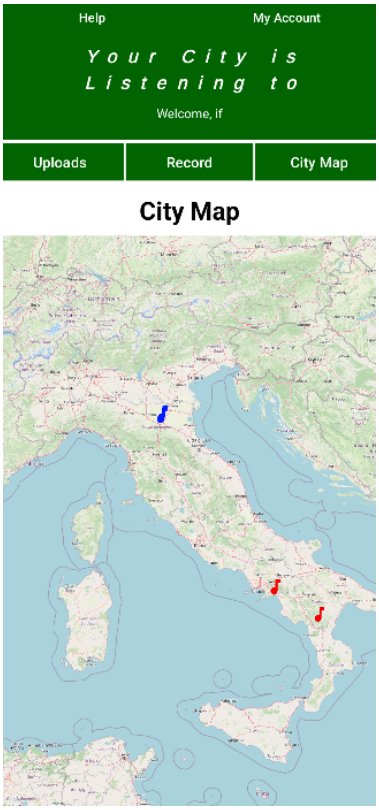
MAIN ACTIVITY - RECORD FRAGMENT



L'activity principale dell'app è la main activity, la quale a sua volta è divisa in tre fragment. Nel record fragment è possibile registrare un audio, interrompendone la registrazione con lo stesso tasto in un secondo momento, ed è anche possibile cancellarlo durante l'ascolto del dispositivo. Una volta concluso, in caso di connessione wifi presente verrà effettuato l'upload automatico tramite la chiamata API Upload, mentre in caso di utilizzo dei dati mobili verrà chiesto all'utente se vuole comunque fare l'upload, oppure se preferisce farlo più tardi. In quest'ultimo caso il file audio sarà salvato in locale, così come i dati saranno tenuti in un database interno chiamato "Ready for Upload", accessibile dalla schermata "Uploads", in attesa di una connessione wifi stabile per effettuare l'upload automatico, anche in

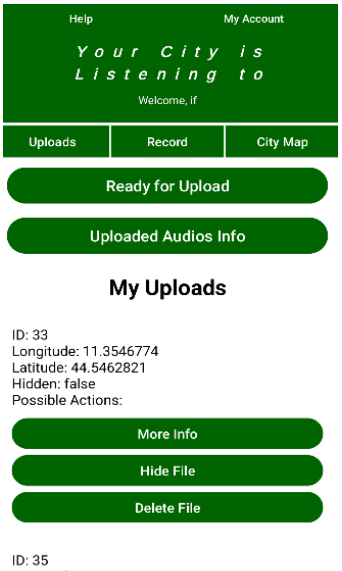
background. Subito dopo un salvataggio corretto di un audio, sarà possibile ascoltarlo con i tasti in basso “Play”, "Pause", e “Stop”. All’avvio di questa activity viene anche richiesto il permesso per usare la posizione corrente.

MAIN ACTIVITY - MAP FRAGMENT



In questa schermata ci troviamo nel “Map Fragment”, ovvero la parte dell’app dove è possibile visionare la mappa.
Appena creato il fragment viene effettuata la chiamata API Audio/All, e successivamente Audio/My, per distinguere a livello visivo i vari upload effettuati.
Gli upload degli utenti sono contrassegnati da una nota musicale rossa, mentre i propri upload sono contrassegnati da una nota blu. La propria posizione è segnalata da un cerchio verde.
Cliccando su ogni audio si verrà portati alla activity More Info di quel determinato file.
Per la realizzazione della mappa è stato utilizzato Open Street Maps.

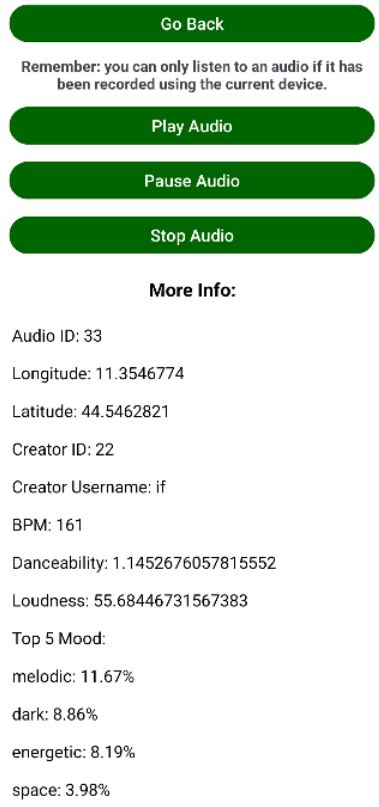
MAIN ACTIVITY - MY UPLOADS FRAGMENT



In questo fragment si dà la possibilità di vedere una panoramica di tutti gli upload effettuati sul backend, non solo i propri, e tutti i dati salvati nel database interno del dispositivo. Il tasto “Ready for Upload” porterà l’utente alla “LocalDb Activity”, mentre il tasto “Uploaded Audios Info” lo indirizzerà alla "Info Audio Activity”, che vedremo in seguito. Successivamente vengono popolate due recycler view con i vari upload: sotto la voce “My Uploads” saranno visibili le informazioni dei file mandati al backend dall’utente che ha effettuato il login, con la possibilità di

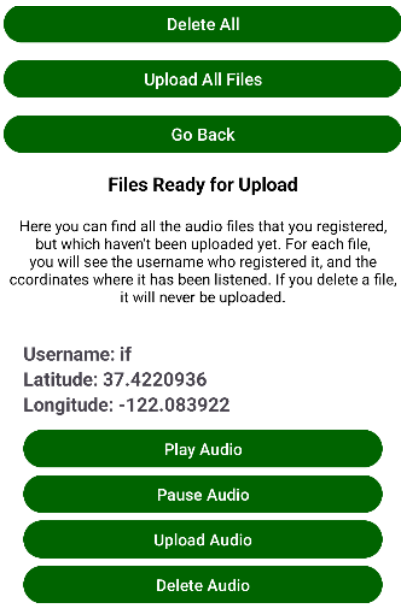
nasconderli, mostrarli, e cancellarli, mentre sotto la voce “All Uploads” sono visibili tutti i dati di tutti gli audio ricevuti dal backend. Le due liste vengono popolate con le chiamate API “audio/all” e “audio/my”. Infine, ogni file ha un bottone con scritto “More Info”, il quale porta all’activity more info dei vari audio che vedremo in seguito.

MORE INFO ACTIVITY



Quando questa activity viene creata, riceve tramite un intent il token corrente e l’id del file audio richiesto, per poi effettuare la chiamata API sul suddetto id, e rilevarne i dati restituiti dal backend (logica usata in ogni activity o fragment del progetto). A questo punto viene popolata una recycler view (sotto la scritta “More Info”) che mostra la risposta alla chiamata. E’ stato scelto, per facilitarne la leggibilità e l’immediatezza, di mostrare solo le cinque voci più compatibili sotto le categorie “Mood”, “Genre”, e “Instrument”, con la relativa percentuale in ordine crescente.
Inoltre sono presenti i tasti play, pause, e stop, per ascoltare l’audio in questione. In quanto i file sono salvati in locale, è possibile farlo solo se a registrarli è stato il dispositivo corrente. Qualora non fosse possibile viene stampato un Toast di errore.

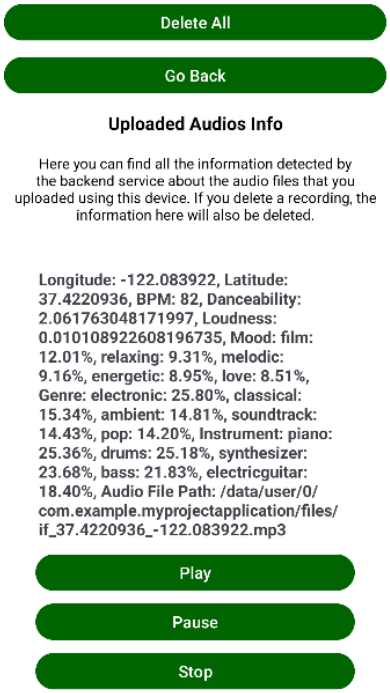
LOCALDB ACTIVITY



Questa activity è lo specchio dell’entità “Upload Data” del database interno all’applicazione. Qui, oltre ad una breve descrizione, vengono mostrati, tramite una recycler view, username e coordinate degli audio registrati ma non ancora caricati al backend. L’utente può quindi ascoltarli, e di conseguenza decidere se cancellarli oppure se eseguirne l’upload. A tal proposito, oltre ai comandi per il singolo audio, sono stati creati anche i tasti “Delete All” e “Upload All”, oltre che al consueto “Go Back” in alto alla pagina. Tutti i suddetti comandi controllano la presenza o meno del wifi, e in caso

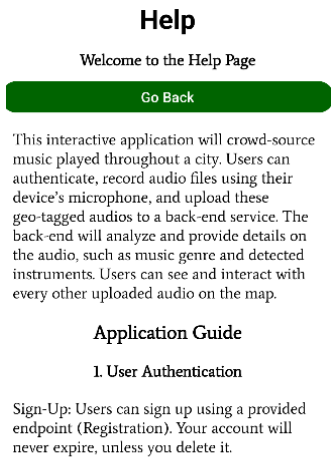
mandano un avviso all'utente per accertarsi che non faccia azioni involontarie.
Inoltre, è stato implementato l'upload in background seguendo la seguente logica: se esiste almeno un elemento in questo database, e c'è un cambio di connessione al wifi da assente a presente, viene eseguito l'upload al backend in automatico, e viene mandata una notifica all'utente.

INFOAUDIO ACTIVITY



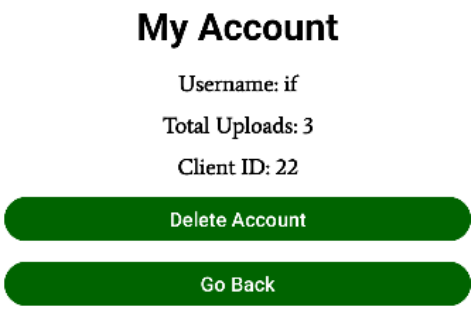
In questa activity viene popolata una recycler view con tutti i dati degli upload effettuati al backend tramite il dispositivo attualmente in uso. I dati vengono presi dall'entità Info Audio del database, la quale dà il nome all'activity, e vengono stampati dopo una breve scritta di descrizione della pagina.
E' anche possibile ascoltare gli audio caricati grazie ai tasti Play, Pause, e Stop.
In caso di connessione ad internet assente si viene indirizzati a questa activity, la quale stampa anche un messaggio che comunica all'utente l'accaduto, e viene disattivato il tasto "Go Back" se non è ancora presente una connessione stabile.

HELP ACTIVITY



Per facilitare l'uso della applicazione all'utente, è stata creata questa activity, facilmente accessibile dal testo in alto a sinistra della homepage. Si tratta di una semplice descrizione testuale di cosa l'app è in grado di fare, e come e farlo dal lato dell'utente.

MY ACCOUNT ACTIVITY



Per dare la possibilità all'utente di eliminare il proprio account, è stata creata l'activity "My Account". Oltre al tasto "Delete Account", è stata inserita la logica per calcolare quanti upload al backend ha effettuato l'utente, visibili poi dopo la voce "Total Uploads". Il contatore tiene in considerazione anche gli audio nascosti, ma non quelli nel "LocalDb" ancora da mandare. Infine sono visibili username e ClientID.

FUNZIONALITA’ EXTRA

My Account Activity: Contatore Total Uploads

Nella schermata di questa activity è stato implementato un contatore che sfrutta la chiamata API “audio/my” per contare quanti audio sono stati caricati dall’utente che sta utilizzando l’app.

Help Activity

E’ stata creata una activity allo scopo di migliorare la User Experience, ovvero la Help Activity, contenente diverse informazioni sulla piattaforma.

My Uploads Fragment: Lista degli audio con metadati

Per avere una migliore chiarezza su quanti e quali audio sono caricati sulla piattaforma, è stata creata questa activity che si appoggia su due recycler view. Vengono fatte le chiamate API “audio/my” e “audio/all” per trovare e poi mostrare i dati dei file, e i tasti aggiungono le varie funzionalità disponibili per l’utente.

InfoAudio Activity e LocalDb Activity: RecyclerView accessibili e comandi

Il database si basa su due entità: InfoAudio e UploadData. Per garantire una maggiore possibilità di interazione all’utente e una maggiore flessibilità dell’app, quest’ultimo ha totale controllo sui dati salvati in locale. Per farlo, può accedere alle due activity sopracitate, dove grazie a due recyclerview potrà visualizzare e gestire tutti i dati salvati localmente nel dispositivo, oltre che ascoltare gli audio registrati, anche offline.

Notification: Upload in Background

Quando si attua un cambio di connessione (da wifi assente a wifi presente), viene eseguito un controllo su UploadData, che contiene i dati degli audio ancora da mandare al backend. A questo punto, in caso di file presenti nel db, è stata creata una logica per eseguire subito l’upload in background, senza dover aspettare la conferma dell’utente. La notifica viene quindi mandata a caricamento completato, e il click su di essa porta l’utente nell’app.