Università Ca'Foscari Venezia

Musikè - Piattaforma di Streaming Audio

CT0006 - Basi di Dati

Gruppo PMG

Studente	Matricola
Paul Toffoloni	886733
Matteo Grazioso	884055
Gheorghe Grecu	884718

A.A. 2021/2022

Venezia, 3 Luglio 2022

Sommario

1 | Preambolo

Come esplicitato nel documento di specifica del progetto [2], l'obiettivo dello stesso è lo sviluppo di una web application che si interfaccia con un database relazionale. Il progetto è sviluppato in Python, utilizzando le librerie Flask e SQLAlchemy, e facendo uso del Relational Database Management System *PostgreSQL*.

É richiesto di curare il design e l'implementazione di una piattaforma di streaming audio simile a Spotify. Il progetto richiede di gestire solo i "metadati" associati alle canzoni, come l'artista, il titolo, l'album, l'anno di produzione, ecc.

Gli utenti devono poter creare le proprie playlist, mentre gli artisti devono avere accesso ad un'interfaccia di analitica relativa alle loro produzioni, che riporti le principali statistiche relative ai loro album ed alle relative canzoni. Gli artisti devono avere inoltre la possibilità di inserire nuovi album e canzoni.

2 | Introduzione

La scelta di sviluppare questo progetto e soprattutto di prendere in considerazione la medesima traccia è stata fatta per dei motivi ben precisi.

Questo progetto e l'idea di quello che c'è dietro a questa web application ci ha fatto sentire molto vicini ad una delle più grandi realtà al mondo.

Spotify è la più grande piattaforma di streaming online con ben 380 milioni di utenti in totale. Il nostro progetto consisteva nel creare qualcosa di molto simile e che si avvicinava a questa piattaforma. Quello che ci ha sorpreso e ci ha entusiasmati è che durante tutto il processo di sviluppo capivamo sempre più quanto complesso è Spotify e la maestosità di questo grande progetto che è portato avanti da un team numerosissimo. Oltre a capire le funzionalità e la complessità, ci siamo sentiti parte di quel grande team e ci ha avvicinato sempre di più al nostro futuro lavoro.

Lo sviluppo della traccia scelta può essere nettamente diviso in due fasi molti differenti tra loro. La prima parte è stata puramente concettuale dove è stato fondamentale analizzare le numerose idee del team, capire se era possibile realizzarle o meno ed assegnare una road-map per la sua realizzazione.

La seconda parte invece è stata caratterizzata dallo sviluppo e dal concretizzare tutte le nostre idee, collegandone il tutto.

L'obiettivo finale è stato quello di realizzare e finalizzare il nostro task principale, ottenere una nostra piattaforma che, per quanto distante possa essere da Spotify, perlomeno ci porti alla stessa idea e dia lo stesso feeling di questo grande progetto.

3 | Funzionalità principali

Musikè può essere suddivisa principalmente in 2 grandi aree. Una è quella riguardante l'artista e tutte le sue funzionalità mentre l'altra riguarda tutta l'area riguardante l'ascoltatore. L'utente, dopo aver fatto il login o registrazione con un apposito forum, entra nella sua homepage principale. In quest'ultima egli ha la possibilità di vedere una sequenza di brani che sono le "hit del momento". Oltre a questa funzionalità, tramite la quale cliccando sul brano può ovviamente ascoltarlo, l'utente ha anche una sua area personale dove può modificare l'e-mail, la password e tutti i suoi dati personali. E' necessario precisare che l'utente ha la possibilità di cambiare i suoi dati di accesso, in quanto è un ID univoco ad identificare l'account e non l'indirizzo e-mail. Oltre alle funzionalità di base, l'ascoltatore ha anche la possibilità di creare e modificare le proprie playlist e aggiungerci all'interno una serie di brani.

L'artista, dopo essersi registrato o fatto il login dall'apposito form, ha funzionalità ben diverse da quelle del listener. La cosa che non differisce da queste due tipologie di utenti è che entrambi possono modificare le proprie generalità e o dati personali. L'artista può aggiungere nuovi brani alla piattaforma. È fondamentale dire che sia gli album che le canzoni possono essere eliminate e tolte dalla piattaforma in qualsiasi momento. Oltre a tutto ciò, il producer ha accesso alla sua dashboard personale dove è possibile consultare tutte le statistiche sui suoi brani, tra cui numero di ascolti e il numero di ascoltatori mensili.

4 | Progettazione concettuale e logica della basi di dati

4.1 | Progettazione concettuale della basi di dati

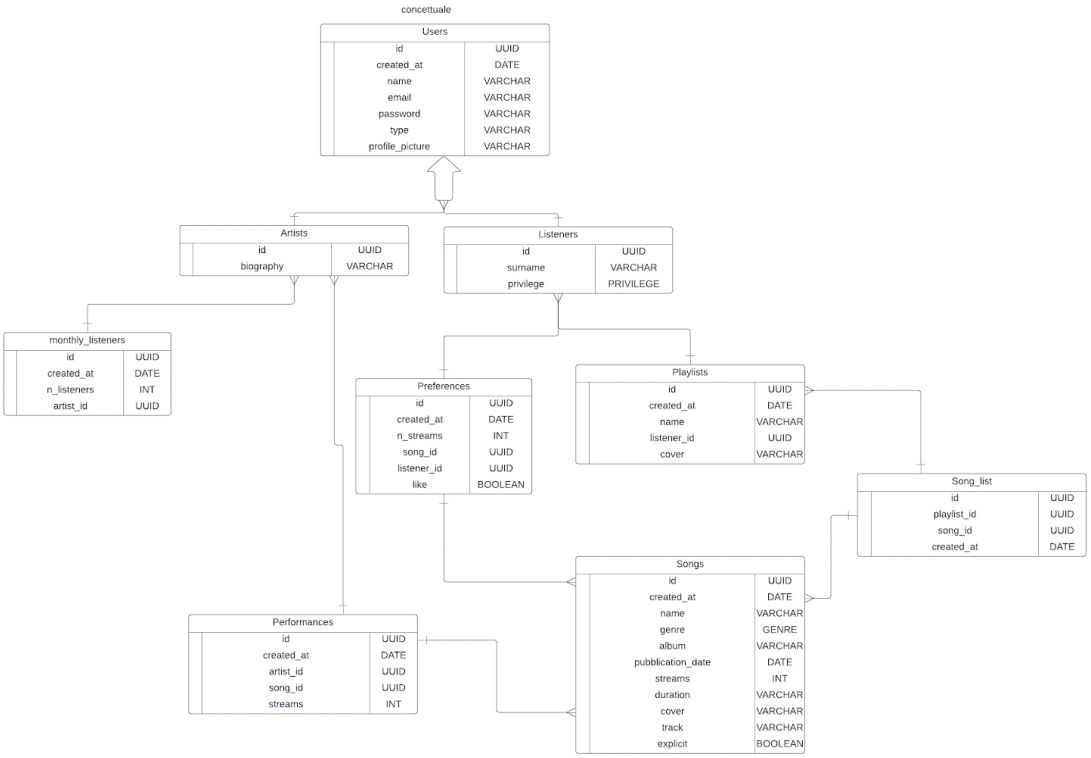

Figura 4.1: Progettazione concettuale della basi di dati

4.2 | Progettazione logica della basi di dati

Users (ID, created_at, name, email, password, type, profile_picture);
Artists (ID, biography, users.id*);
Listeners (ID, surname, privilege, users.id*);
Playlists (ID, created_at, name, listeners.id*, cover);
Songs (ID, created_at, name, genre, album, publication_date, streams, duration, cover, track, explicit);
Preferences (ID, created_at, n_streams, song.id*, listener.id*, like);
Song_lists (ID, created_at, playlist.id*, song.id*);
Performances (ID, created_at, artist.id*, song.id*, streams);
Monthly_listeners (ID, created_at, n_listeners, artists.id*);

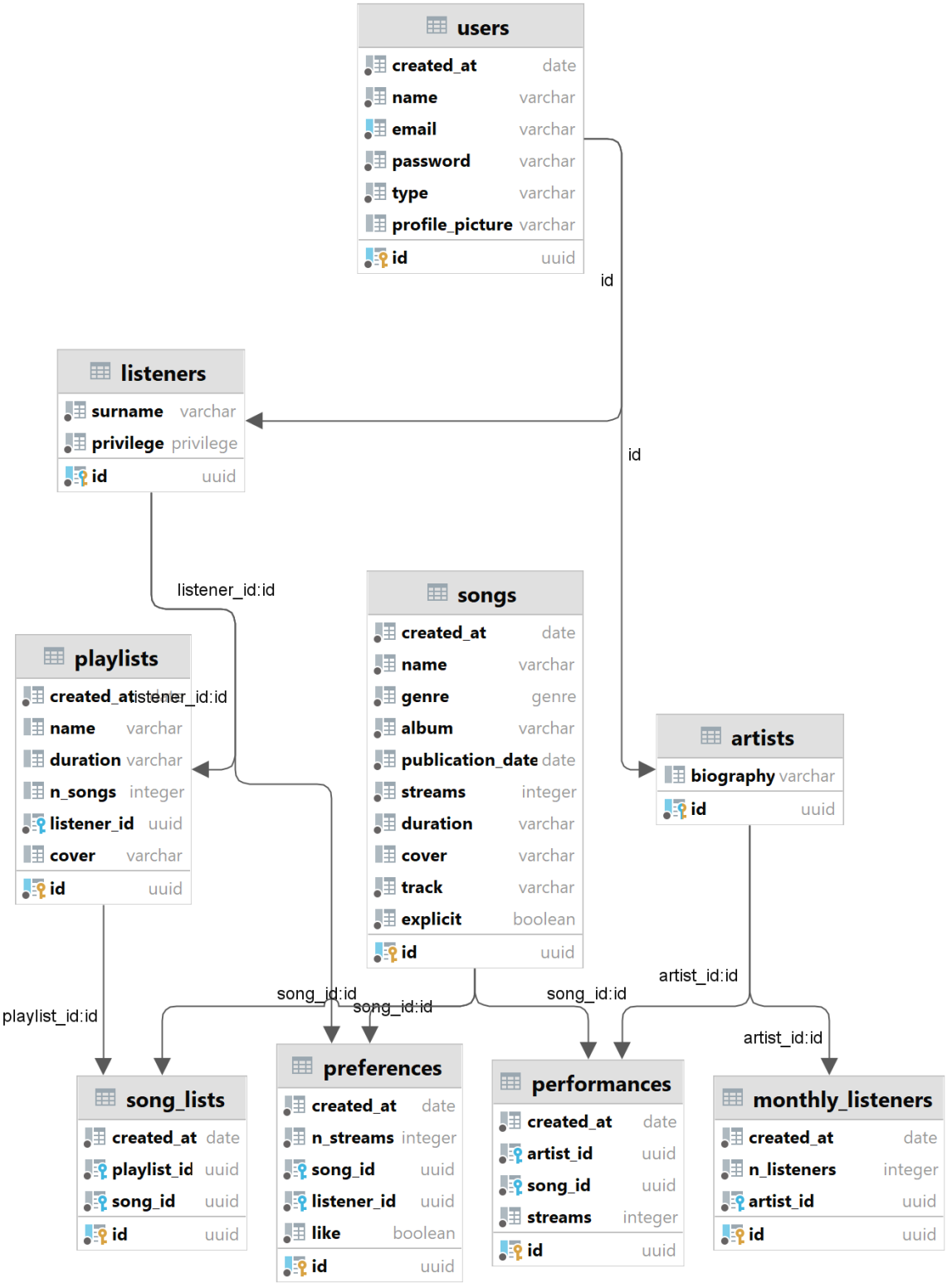

Figura 4.2: Progettazione logica della basi di dati

4.3 | Progettazione fisica della basi di dati

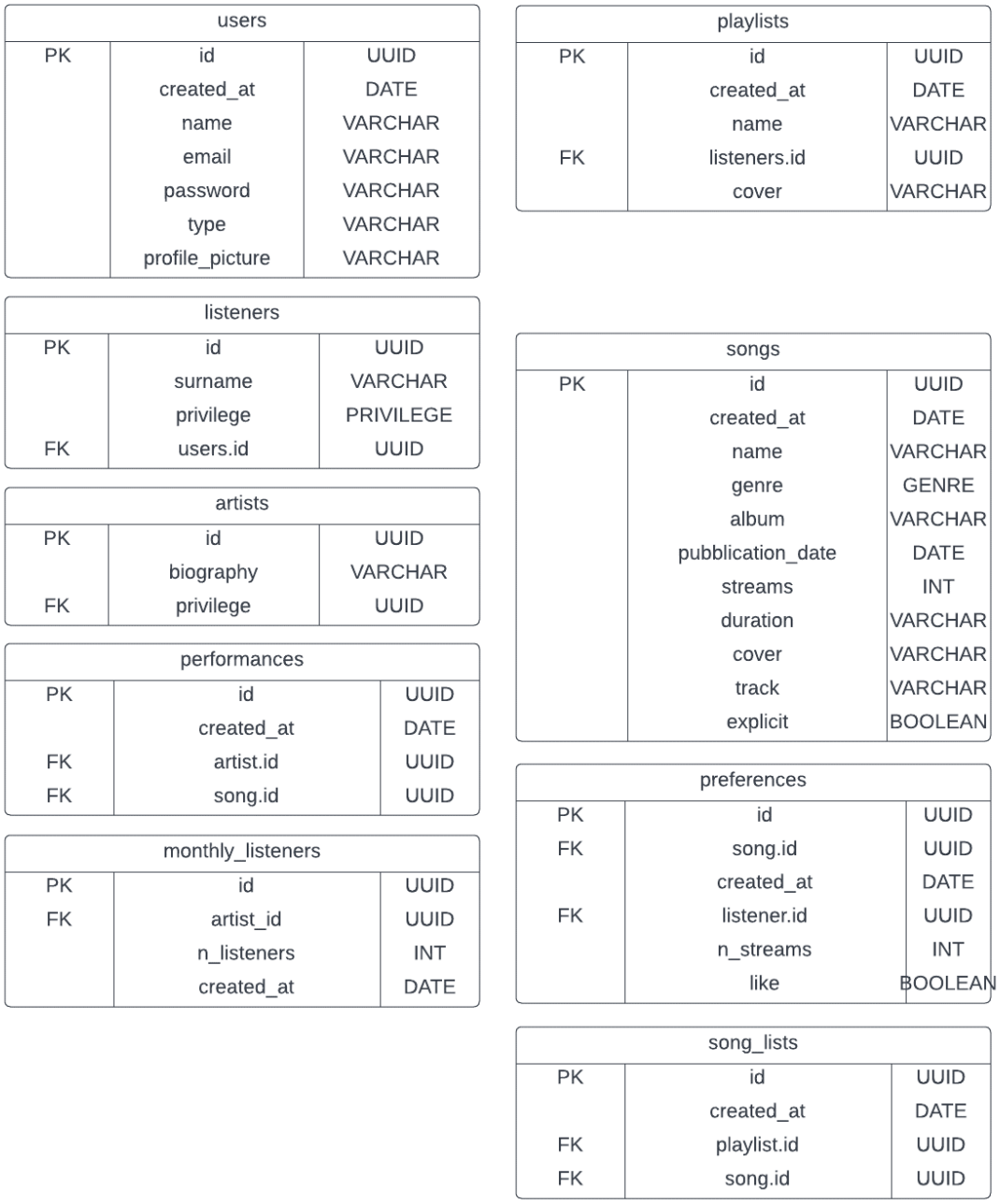

Figura 4.3: Progettazione fisica della basi di dati

Nella progettazione fisica, abbiamo riportato tutte le connessioni e le relazioni che ci sono tra le varie entità. È possibile osservare il tutto nella rappresentazione soprastante.

4.4 | Descrizione delle tabelle

Ogni tabella è caratterizzata da un ID univoco sotto forma di UUID v4 (Universally Unique Identifier, version 4)[4], ovvero un identificativo alfanumerico di 16 cifre generato casualmente da Flask al momento di creazione del record. Ogni istanza di qualsiasi tabella è inoltre caratterizzata da data e ora di creazione.

4.4.1 | Songs

Il brano rappresenta la traccia caricata all'interno della piattaforma. Ogni brano prevede un titolo, un album, un anno di pubblicazione, una durata e una copertina caricati dall'artista. Il brano potrà avere solo un valore per i campi Genere, Album, Anno di Pubblicazione e una Copertina. Il brano può avere uno o più artisti. Ogni brano viene caratterizzato a seconda del genere e dell'artista. È previsto inoltre anche un numero di like espressi dagli utenti registrati alla web-application. Il numero di like verrà utilizzato per gestire le preferenze e brani consigliati dall'utente. Ogni brano deve essere contrassegnato con l'etichetta explicit/not explicit che andrà utilizzata per gestire le restrizioni degli utenti minorenni; in particolare, gli utenti che non hanno raggiunto la maggiore età sono impossibilitati all'ascolto di contenuti explicit. Il numero di ascolti viene incrementato ad ogni riproduzione del brano.

4.4.2 | Artist

L'artista è dotato di un nome d'arte e di una sua biografia. Egli ha la possibilità di gestire i propri brani con una libreria personale. Può caricare nuovi brani nella piattaforma e/o eliminare quelli già presenti nella sua libreria. Può anche consultare una pagina dedicata alle statistiche. Queste ultime prevedono :

1. rappresentazione testuale degli ascoltatori del mese corrente con indicazione percentuale di incremento o decremento rispetto ai mesi precedenti;
2. rappresentazione tramite line graph dell'andamento degli ascolti nel corso dei mesi;
3. rappresentazione raggruppata tramite histogram degli ascolti relativi ad ogni proprio album;
4. rappresentazione dettagliata tramite histogram degli ascolti relativi ad ogni singolo album;

4.4.3 | Users

L'utente è identificato dalla sua e-mail con cui si è registrato alla piattaforma, oltre che dal suo ID univoco. Ne consegue che non si può verificare la circostanza in cui account abbia la stessa e-mail associata. L'utente in fase di registrazione può scegliere se attivare il piano Premium o rimanere nel piano Free. Se l'utente non esprime la scelta di attivare il piano Premium, verrà creato un account con il piano Free. Il piano Free prevede la riproduzione di un breve spot pubblicitario durante la riproduzione dei brani, con cadenza di una (1) riproduzione ogni 3 brani ascoltati. Il piano Premium non prevede la riproduzione della pubblicità durante l'ascolto dei brani. L'utente può in ogni momento sottoscrivere il piano Premium attraverso un pagamento; *[IN FASE DI SVILUPPO]*. L'utente in fase di registrazione dovrà inoltre inserire la propria data di nascita, dalla quale sarà poi calcolata l'età effettiva dell'utente. L'utente potrà creare e personalizzare le proprie playlist di brani, caratterizzate da un nome, aggiungendo nuovi brano o rimuovendo brani già inclusi.

4.4.4 | Playlists

Ogni playlist prevede un nome, data di creazione e ascoltatore. La playlist può avere solo un ascoltatore, il medesimo utente che l'ha creata. Non è pertanto prevista la condivisione intra-utenti delle playlist. La playlist sarà dotata dal numero di brani e della durata totale delle tracce musicali incluse. Ogni playlist non prevede un numero minimo di brani.

4.4.5 | Monthly_Listeners

Monthly listeners è una tabella usata per le statistiche per l'artista. Essendo composta dal numero di ascoltatori e dall'id dell'artista, ci permette di dare la statistica all'artista di quanti utenti stiano interagendo con i suoi brani mensilmente.

4.4.6 | Performances

Performances è l'entità che permette di associare ogni artista al proprio brano creato.

4.4.7 | Preferences

Con preferences andiamo a salvare ogni dato delle canzoni utili per una seconda fase di sviluppo. Con quanto appena detto si intende tutta la raccolta di informazioni per crearne poi la dashboard delle statistiche utilizzata dall'artista e per far sì che la web-applcation possa consigliare dei brani all'utente Listener.

4.4.8 | Songs_lists

Song lists è l'entità fondamentale per associare le canzoni alle playlist in cui le stesse canzoni sono contenute.

5 | Query principali

```
albums_with_streams = db.session.query(Song.album, func.sum(Song.streams)) \
        .join(Performance, Song.id == Performance.song_id and Performance.artist_id ==
    current_user.id) \
        .group_by(Song.album) \
        .order_by(Song.album) \
        .all()
```

Source code 1: Most streamed songs query

Questa query ritorna una lista di coppie in cui il primo elemento è il nome dell'album (che è una stringa) e il secondo elemento è un intero che rappresenta gli ascolti totali dell'album, che sono stati calcolati facendo la somma degli ascolti di ogni singola canzone di quell'album. I nomi sono giustamente ordinati in ordine alfabetico.

```
monthly_listeners = MonthlyListeners.query \
        .join(Artist, Artist.id == MonthlyListeners.artist_id) \
        .order_by(desc(MonthlyListeners.created_at)) \
        .all()
```

Source code 2: Monthly Listeners query

Questa query ritorna le tuple della tabella MonthlyListeners joinate con la tabella Artist e ordinate per ordine decrescente sulla data di creazione della tupla MonthlyListeners.

```
fetched_song = Song.query.join(SongList, Song.id == SongList.song_id) \
        .filter(SongList.playlist_id == playlist_id) \
        .order_by(Song.name).all()
```

Source code 3: Songs in playslist

Questa query ritorna tutte le canzoni appartenenti ad una determinata playlist ordinate in ordine alfabetico per il nome delle canzoni.

6 | Principali scelte progettuali

L'interfaccia di front-end è stata creata unendo le nostre competenze in ambito programmazione con l'aiuto delle librerie di Bootstrap. Grazie ai suoi componenti, abbiamo realizzato un interfaccia utente e gran parte della componente grafica di Musikè.

Per la parte strutturale, ovvero mettere in comunicazione il database SQL alla parte di back-end sviluppata tramite linguaggio Python, abbiamo utilizzato SQLAlchemy, perchè ci ha permesso di mettere in relazione il database relazione in SQL e farlo interagire al meglio con il nostro codice e il framework Flask [1], per il web development.
Abbiamo utilizzato inoltre la versione di SQLAlchemy ORM (Object-Relational Mapper) [3], un componente che ha lo scopo di rappresentare le relazioni del database come oggetti Python e che ci ha permesso di ovviare le difficoltà legate al problema di SQL Injection e Cross-Site Scripting (*XSS*).

Abbiamo fatto uso di PyCharm, in quanto riteniamo sia uno dei migliori IDE per la programmazione in Python, oltre a concederci la visione del DB nell'IDE stesso. Sono state fatte due scelte progettuali molto importanti per la realizzazione del progetto, che necessitano di attenzione. La prima è che è stato fatto uso dei *decorator* in modo tale da garantire l'unicità delle azioni da parte delle due tipologie utenti *listener* e *artist*. Questo per evitare che uno dei due utenti possa compiere azioni o modifiche non a lui permesse, in base alla politica di ruoli da noi attuata.

```
def listener_only(func):
    @wraps(func)
    def wrapper(*args, **kwargs):
        if current_user.type != 'listener':
            abort(403)
        return func(*args, **kwargs)
    return wrapper
```

Source code 4: listener_only

Come è possibile osservare nel codice sovrastante, è presente il controllo legato alla tipologia di user d tipo listener: questo ci permette così di bloccare azioni non permesse se il test risulta non superato.

Altro aspetto fondamentale da chiarire è che le classi ereditano dalla classe CustomModel. E' possibile osservarne l'implementazione qui sotto.

```
class CustomModel(db.Model):
    """
        Represents every model in the db.
    """
    __abstract__ = True
    id = Column(UUID(as_uuid=True),
                # In questo modo gli UUID di default vengono creati dal DBMS
                # usando la funzione `uuid_generate_v4()`
                server_default=func.public.uuid_generate_v4(),
                nullable=False, primary_key=True)
    created_at = Column(Date, nullable=False, default=datetime.datetime.now)
```

Source code 5: class CustomModel

7 | Sviluppi futuri

Il gruppo, oltre allo sviluppo di questa prima versione consistente della web-application, ha inoltre elaborato delle funzionalità che si potrebbero implementare in un prossimo futuro per una versione rivista ed estesa del proprio progetto.

Per quanto riguarda la gestione degli utenti, uno sviluppo futuro concerne nel concedere agli ascoltatori di scegliere il proprio piano tra *Free* e *Premium*. Se l'utente non esprime la scelta di attivare il piano *Premium*, verrà automaticamente inserito nel piano *Free*.

- Il piano "Free" prevede ad avere pubblicità durante la riproduzione di brani, con cadenza di 1 comparizione ogni 3 brani ascoltati.
- Il piano "Premium" non prevede la comparizione della pubblicità durante l'ascolto. Quando l'utente sottoscrive il piano "Premium" attraverso un pagamento e una conferma via email.

L'utente in fase di registrazione già da ora inserisce la propria data di nascita, dalla quale sarà poi calcolata l'età effettiva dell'utente. La maggiore età gli permetterà di visualizzare e ascoltare brani categorizzati *Explicit*. Si noti che già in questa versione i brani devono essere segnalati come *Explicit* o *not Explicit* in fase di creazione da parte dell'artista.

La piattaforma potrebbe inoltre consentire all'artista di aggiungere canzoni ad un proprio album solo quando tale album non è ancora stato pubblicato. Solo dopo la pubblicazione dell'album, gli ascoltatori potranno beneficiare dell'ascolto dei suoi brani e il numero di stream entrerà nel computo totale dell'album nonché dell'artista.

La piattaforma potrebbe altresì implementare un semplice meccanismo di raccomandazione, che sia in grado di suggerire nuovi canzoni agli utenti sulla base dei loro gusti.
In fase di brainstorming, il gruppo concorda di implementare tale politica come segue.
La preferenza viene espressa tramite una variabile Double che viene modificata ogni qual volta si faccia un'azione. Le azioni possono essere di vario tipo e vanno a modificare il valore della preferenza attraverso un servizio di queueing asincrono in modo da non sovraccaricare il database di richieste. Un sistema di preferenza si basa su due tipi di filtering:

- **Filtering basato sul contenuto:** si basa sulla reazione dell'utente rispetto a un determinato contenuto;
- **Filtering collaborativo:** non si basa sulle caratteristiche del contenuto di ciò che si ascolta, ma sulle tendenze dell'utente e di altri utenti simili, rispetto a cosa ascoltano di più e cosa piace in generale

Un elenco che mira ad essere esaustivo dei dati che verranno raccolti è il seguente:

- aggiunta/rimozione di un brano dalla playlist;
- comportamento di skip di una canzone;
- canzoni e artisti più ascoltati (per vedere gli artisti più ascoltati controlliamo il numero di ascolti per utente di una canzone e raggruppiamo per artista);
- canzoni condivise (quando si clicca sul tasto condividi, la web-application semplicemente creerà un link alla canzone del tipo *"https://musike.com/song/¡idcanzone¿"*;
- generi più ascoltati;
- quali brani o artisti sono più ascoltati;
- preferenze audio (bpm - battiti per minuto, spettro, tonalità);
- quali brani contengono altre playlist;
- like alla canzone.

Il gruppo si riserva, qualora si presenti l'occasione, di perseguire lo scopo di ampliare questo progetto e di implementare quanto specificato in questa sezione.

Riferimenti

[1] Miguel Grinberg. *Flask Web Development*. 2nd ed. O'Reilly Media, Inc., 2018.

[2] prof. Calzavara Stefano. *Basi di Dati Mod. 2 - Progetto A.A. 2021/2022*. 2022.

[3] *SQLAlchemy ORM*. https://docs.sqlalchemy.org/en/14/orm/. URL consultato il 29/04/2022.

[4] *Universally unique identifier, version 4 (random)*. https://en.wikipedia.org/wiki/Universally_unique_identifier#Version_4_(random). URL consultato il 29/04/2022.