Università degli studi di Salerno

Dipartimento di Informatica

Corso di Laurea in Informatica

Musimatica

"Transposition Midi Era"

Docenti:

prof. Roberto De Prisco

Studente

Choaib Goumri 0512118390

Repository GitHub *https://github.com/RedMulaCrackIn/TranspositionMidiEra.git*

03/02/25

INDICE

Panoramica del progetto

Trasposition Midi Era è un'applicazione avanzata progettata per la manipolazione e la trasposizione di file MIDI, con l'obiettivo principale di consentire agli utenti di modificare la tonalità delle tracce musicali digitali senza compromettere la qualità del suono o il ritmo originale. L'app è pensata per essere uno strumento versatile e preciso per musicisti, produttori musicali, arrangiatori e appassionati di musica che desiderano avere un controllo totale sulle proprie composizioni digitali.

Funzionalità e Obiettivi

Trasposition Midi Era si propone di semplificare il processo di trasposizione musicale, un aspetto spesso complesso nella produzione musicale. La trasposizione consiste nel cambiare la tonalità di un brano, ma senza alterare il suo andamento ritmico, la dinamica o la qualità timbrica. Ciò è particolarmente utile in contesti musicali in cui si desidera adattare una composizione a una voce, a uno strumento o a una specifica esigenza di arrangiamento, mantenendo al contempo l'integrità dell'opera.

Tecnologia Sottostante

Per garantire la massima precisione nella trasposizione delle note musicali e la gestione accurata delle varie sfumature sonore, Trasposition Midi Era utilizza la libreria **Music21**, una delle risorse più potenti e flessibili nel campo dell'analisi e della manipolazione dei dati musicali. Music21 è una libreria Python open-source ampiamente utilizzata in ambito accademico e professionale per lavorare con file MIDI, MusicXML, e altri formati musicali. Essa fornisce una vasta gamma di strumenti per l'analisi armonica, la generazione di partiture, la trasposizione automatica e la manipolazione avanzata delle tracce musicali.

La libreria Music21 è stata scelta per la sua capacità di gestire in modo preciso e accurato le trasposizioni, grazie alla sua profonda comprensione della teoria musicale, che consente di adattare le tracce MIDI a qualsiasi tonalità desiderata, mantenendo le proporzioni relative delle note e delle dinamiche, e intervenendo sulle armonie senza compromettere la qualità musicale.

Interfaccia Utente Intuitiva

Uno degli aspetti distintivi di Trasposition Midi Era è la sua interfaccia utente, progettata per essere semplice da usare anche per chi non ha una preparazione tecnica avanzata. Grazie a un design pulito e intuitivo, gli utenti possono caricare rapidamente i propri file MIDI, selezionare la tonalità di destinazione e applicare modifiche in pochi passaggi, senza doversi preoccupare di dettagli complessi o tecnicismi. L'interfaccia offre anche strumenti di anteprima, che consentono di ascoltare le modifiche prima di applicarle definitivamente, garantendo un processo di editing altamente controllato e personalizzabile.

Componenti Principali

1. **Interfaccia Grafica**: Presenta una griglia di generi musicali, pulsanti di interazione e un player MIDI.
2. **Motore di Trasposizione**: Algoritmo per il cambio di tonalità basato su **Music21**, che analizza e modifica le note MIDI rispettando le regole armoniche.
3. **Sistema di Riproduzione**: Player integrato per ascoltare le modifiche in tempo reale.

Guida all'Utilizzo

Selezione del Genere:

- L'utente accede alla pagina principale, seleziona un genere musicale cliccando su una delle carte di genere (Classico, Jazz, Blues).
- Una volta selezionato un genere, l'utente invia il modulo di generazione.

Generazione della Musica:

- Quando il modulo viene inviato, il server riceve la richiesta e genera una melodia di base.
- Successivamente, applica la trasformazione del genere scelto tramite le funzioni definite in `GENRE_FUNCTIONS`.
- I file MIDI risultanti (originale e processato) vengono salvati sul server.

Riproduzione

- L'utente viene reindirizzato alla pagina di riproduzione, dove può ascoltare le due tracce (originale e processata) usando un player MIDI basato su **Magenta.js**.

Architettura tecnica

1. Backend (Flask)

Il backend è sviluppato utilizzando il framework **Flask**, che è un framework web leggero per Python, e gestisce le logiche di server, routing e gestione dei file. Ecco una descrizione dei principali file e funzioni nel backend:

File Python - Flask

- **Routing:**
    - `@app.route('/')` – Questa route gestisce la pagina principale (`index.html`). È il punto di ingresso dell'utente nell'applicazione. In questa vista, l'utente può selezionare il genere musicale da una lista.
    - `@app.route('/generate', methods=['POST'])` – Questa route riceve i dati dal modulo di selezione del genere musicale, genera la composizione musicale e poi la salva in due versioni: originale e processata. Successivamente, l'utente viene rediretto alla pagina di riproduzione.
    - `@app.route('/generated/<filename>')` – Questa route serve i file MIDI generati (originali e modificati) che l'utente può scaricare o riprodurre direttamente nella pagina di riproduzione.

- **Logica di Generazione Musicale:**
    - La funzione `generate_base_melody()` crea una composizione musicale di base (una semplice melodia di 8 note in una tonalità di Do maggiore).
    - Il dizionario `GENRE_FUNCTIONS` contiene funzioni che applicano trasformazioni ai file musicali generati in base al genere selezionato:
        - **Classical**: Aggiunge dinamiche come "p" (piano) e "f" (forte).
        - **Jazz**: Applica un ritmo sincopato in stile jazz.
        - **Blues**: Applica una trasposizione e modifica alcune note (blue notes) tipiche del blues.

- **Salvataggio del MIDI:**
    - La funzione `save_midi(stream_obj, filename)` utilizza il metodo `write()` di Music21 per scrivere il file MIDI sul filesystem.

2. Generazione Musicale (Music21)

Music21 è una libreria potente per l'analisi e la manipolazione di musica in formato MIDI e MusicXML. Viene utilizzata per generare e trasformare la musica. Ecco come viene utilizzata nel progetto:

- **Creazione della Melodia di Base:**

    - La melodia di base è costruita con la funzione `generate_base_melody()`, che crea un oggetto `stream.Stream` contenente delle note MIDI.
    - Le note sono scelte casualmente da una lista di pitch (60-71, che corrispondono a note della scala di Do maggiore).
    - Ogni nota ha una durata che viene scelta casualmente tra 0.5 e 1.0 quarti di durata (mezzanote o note intere).
    - L'oggetto `stream.Stream` viene configurato con un tempo (4/4) e uno strumento (Piano).

- **Trasformazione per Genere Musicale:**

    - **Classical**: La melodia viene trasposta, e le dinamiche vengono aggiunte (le note nei tempi pari sono piano "p", quelle nei tempi dispari sono forte "f").
    - **Jazz**: Viene modificata la durata delle note per applicare il "swing" tipico del jazz. Le note sui tempi pari sono accorciate a 0.75 di durata (una croma), mentre quelle sui tempi dispari hanno una durata di 0.25 (una semicroma).
    - **Blues**: La melodia di base viene trasposta di 2 toni verso il basso, e le note vengono modificate per introdurre le "blue notes" (ad esempio, la nota Mi naturale diventa Mi bemolle).

---

3. Frontend (HTML, CSS, JavaScript)

Il frontend è costituito da HTML, CSS e JavaScript. Ecco il flusso di lavoro del frontend:

HTML (index.html, player.html)

- **index.html**:

    - Mostra una griglia di selezione dei generi musicali (Classico, Jazz, Blues). Ogni card di genere ha un'icona e una descrizione.
    - L'utente seleziona un genere e poi invia un modulo per generare la musica corrispondente.
    - Una volta che l'utente seleziona un genere, il valore viene inviato al backend come parte di una richiesta `POST` a `/generate`.

- **player.html**:

    - Mostra due tracce musicali (una originale e una processata), che sono i file MIDI generati dal backend.
    - Permette la riproduzione dei file MIDI con un player interattivo che usa JavaScript per caricare e riprodurre i file.
    - Ogni traccia ha un bottone di riproduzione che, quando clicco, carica il file MIDI associato e lo riproduce usando **Magenta.js**, una libreria che facilita la riproduzione e visualizzazione di dati musicali.

CSS (style.css):

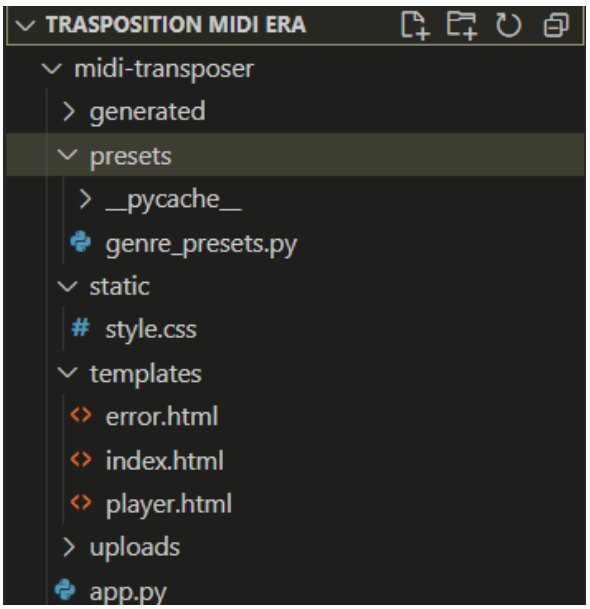
Viene usato per lo styling dell'interfaccia utente. Vengono applicati stili per la disposizione delle card dei generi, la visualizzazione del pulsante di generazione e il player.

JavaScript

- **Interazione dell'utente**: Gestisce la selezione del genere musicale. Quando l'utente clicca su una card di genere, il valore corrispondente viene salvato in un campo nascosto del modulo (`selectedGenre`) e l'utente può inviare il modulo per generare la musica.

- **Riproduzione MIDI**: Utilizza **Magenta.js** per caricare e riprodurre i file MIDI generati dal backend. Quando un utente preme il pulsante di riproduzione, viene fatto un `fetch()` al server per ottenere il file MIDI, che viene quindi convertito in un formato riproducibile dalla libreria.

Struttura delle Cartelle:



Dettagli Implementativi

1. Dipendenze e Moduli Utilizzati

Flask

Flask è un framework web leggero e flessibile scritto in Python, utilizzato per sviluppare applicazioni web. È considerato un framework "micro" perché fornisce solo gli strumenti di base per la creazione di web app, ma lascia la libertà di integrare librerie e funzionalità aggiuntive in base alle esigenze. Questo lo rende ideale per applicazioni di piccole e medie dimensioni, come quella in oggetto.

Flask offre funzionalità per la gestione delle route (ovvero le URL dell'app), la gestione delle richieste HTTP (GET, POST, ecc.) e la gestione dei template HTML per la visualizzazione dei contenuti. In questo caso, Flask è utilizzato per gestire l'interfaccia web che permette all'utente di selezionare il genere musicale desiderato e generare il file MIDI corrispondente.

Music21

music21 è una libreria Python che consente di analizzare, manipolare e generare musica in formato simbolico (ad esempio, note, durate, strumenti, accordi) e in vari formati come **MIDI**, **MusicXML** e **ABC notation**. La libreria è stata progettata per permettere l'analisi musicale, la creazione di composizioni automatiche e l'esportazione di queste composizioni in formati compatibili con strumenti musicali software e hardware.

Nel nostro codice, **music21** viene utilizzata per creare una melodia di base, manipolarla per adattarla a vari generi musicali e poi esportarla in formato MIDI.

Os

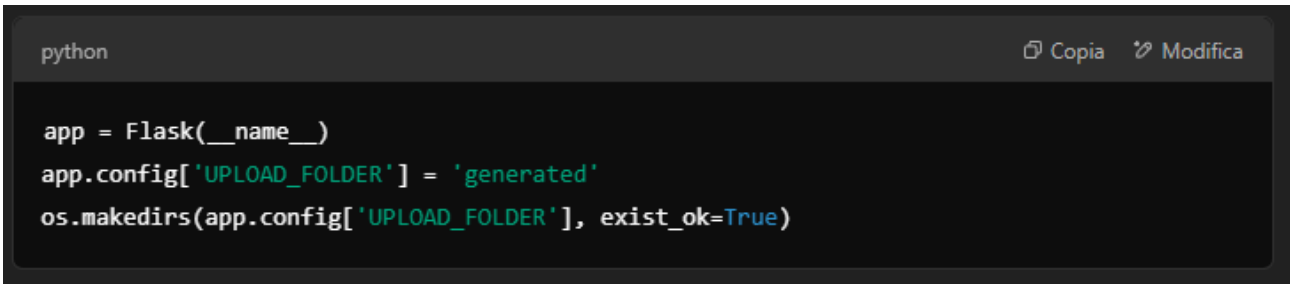
Il modulo **OS è** una libreria standard di Python che fornisce un'interfaccia per interagire con il sistema operativo. In questo caso, viene utilizzato per gestire i file, creare cartelle (come quella per salvare i file MIDI generati) e ottenere il percorso dei file. Ad esempio, **os.makedirs(app.config['UPLOAD_FOLDER'], exist_ok=True)** serve a creare la cartella in cui salvare i file generati, assicurandosi che la cartella venga creata solo se non esiste già.

Random

Il modulo **random** è una libreria standard che fornisce funzioni per generare numeri casuali. È utilizzato in vari punti del codice per generare note casuali per la melodia di base e per generare numeri casuali nei nomi dei file MIDI (per evitare conflitti di nomi).

2. Flask Application

Nel nostro caso, **Flask** è utilizzato per creare un server web che gestisce le richieste degli utenti. Gli utenti possono interagire con l'applicazione tramite un'interfaccia HTML. Flask permette di configurare facilmente il server, gestire la ricezione di dati (ad esempio, quando un utente invia un modulo), e fornire i file generati come risposte.



Flask(name): Qui viene inizializzata l'app Flask. L'argomento `__name__` è passato per determinare il contesto dell'applicazione.

app.config['UPLOAD_FOLDER'] = 'generated': Questa riga configura la cartella dove i file generati saranno salvati, ossia la cartella `generated`.

os.makedirs(...): Assicura che la cartella esista, altrimenti la crea.

3. Generazione della Melodia di Base

Cos'è una Melodia di Base

Una **melodia di base** è una sequenza di note che serve come struttura per una composizione musicale. In questo caso, la melodia di base è generata casualmente da un set predefinito di note appartenenti alla **scala di Do maggiore**. La melodia è creata come una sequenza di note che si susseguono, con durate variabili.

music21 viene utilizzato per generare questa melodia. L'oggetto `stream.Stream()` è una sequenza di elementi musicali che può contenere note, accordi, strumenti, misure, dinamiche, e altre informazioni musicali. La melodia di base è una sequenza di note (corrispondenti a valori MIDI) di durata casuale (mezzo tempo o tempo intero).

python Copia Modifica

```python
def generate_base_melody():
    s = stream.Stream()
    s.metadata = metadata.Metadata(title="AI Composition")
    s.append(meter.TimeSignature('4/4'))
    s.append(instrument.Piano())

    pitches = [60, 62, 64, 65, 67, 69, 71]  # Scala di Do maggiore
    for _ in range(8):
        n = note.Note(random.choice(pitches))
        n.duration.quarterLength = random.choice([0.5, 1.0])
        s.append(n)

    return s
```

stream.Stream(): Crea un flusso musicale che contiene la sequenza di note.

metadata.Metadata(title="AI Composition"): Aggiunge un titolo alla composizione.

meter.TimeSignature('4/4'): Aggiunge una firma di tempo 4/4 (quattro battiti per misura).

instrument.Piano(): Definisce l'istrumento, in questo caso il pianoforte.

random.choice(pitches): Seleziona casualmente una nota dalla lista di note in Do maggiore (60 è C4, 62 è D4, ecc.).

4. Applicazione dei Generi Musicali

Cosa significa applicare un genere musicale a una melodia?

Ogni genere musicale ha caratteristiche stilistiche che ne influenzano la composizione, come il ritmo, l'armonia e la melodia. Nel nostro caso, le funzioni `apply_classical`, `apply_jazz` e `apply_blues` applicano modifiche specifiche alla melodia di base per adattarla a ciascun genere musicale.

- **Classical**: Il genere classico solitamente presenta una struttura più rigida e delle dinamiche ben definite. Nel codice, vengono aggiunte dinamiche "p" (piano) e "f" (forte) in base all'offset delle note per simulare il comportamento dinamico tipico della musica classica.
- **Jazz**: La musica jazz è spesso caratterizzata da un ritmo "swing" e una maggiore libertà nell'interpretazione. Per questo motivo, nella versione jazz della melodia, le note vengono distribuite in modo da avere un ritmo che imita il "swing" (dove le note su un battito sono più lunghe rispetto a quelle su un battito debole).
- **Blues**: Il blues si distingue per l'uso di "blue notes" (note abbassate) che creano un'atmosfera emotiva e peculiare. La funzione `apply_blues` trasporta alcune note per creare questo effetto.

Funzione `apply_jazz`

python Copia Modifica

```python
def apply_jazz(original):
    jazz = stream.Stream()
    jazz.metadata = metadata.Metadata(title="Jazz Version")

    # Swing rhythm
    for n in original.recurse().notes:
        new_n = note.Note(n.pitch)
        new_n.duration.quarterLength = 0.75 if n.offset % 1 == 0 else 0.25
        jazz.append(new_n)

    return jazz
```

Swing rhythm: Le note su ogni battito (in offset pari) hanno una durata di 0.75, mentre le note sugli altri battiti sono più brevi (0.25), imitando il ritmo "swing" del jazz.

5. Gestione della Richiesta e Generazione dei File MIDI

Quando l'utente seleziona un genere musicale attraverso il modulo HTML, la funzione `generate_midi()` raccoglie la selezione, genera una melodia di base e la trasforma nel genere scelto. Successivamente, salva entrambe le versioni (originale e trasformata) come file MIDI e restituisce i nomi dei file in modo che l'utente possa scaricarli.

python Copia Modifica

```python
@app.route('/generate', methods=['POST'])
def generate_midi():
    try:
        genre = request.form.get('genre', '').lower()
        if genre not in GENRE_FUNCTIONS:
            return render_template('error.html', error="Seleziona un genere valido")

        # Genera e processa
        base_stream = generate_base_melody()
        processed_stream = GENRE_FUNCTIONS[genre](base_stream)

        # Salva i file
        original_filename = f"original_{random.randint(1000,9999)}.mid"
        processed_filename = f"processed_{random.randint(1000,9999)}.mid"

        save_midi(base_stream, original_filename)
        save_midi(processed_stream, processed_filename)

        return render_template('player.html',
                               original=original_filename,
                               processed=processed_filename,
                               genre=genre.capitalize())
```

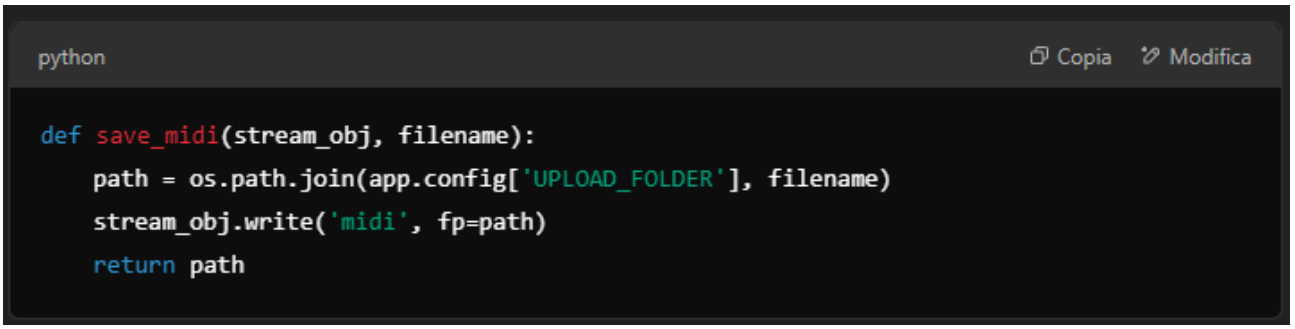
request.form.get('genre'): Ottiene il genere selezionato nel modulo HTML.

GENRE_FUNCTIONSgenre: Applica la funzione di elaborazione corrispondente al genere scelto.

save_midi(): Salva il flusso musicale in formato MIDI.

6. Gestione dei File MIDI

Il sistema salva i file MIDI nella cartella `generated` utilizzando la funzione `save_midi()`, che scrive il flusso musicale in un file MIDI e restituisce il percorso del file.

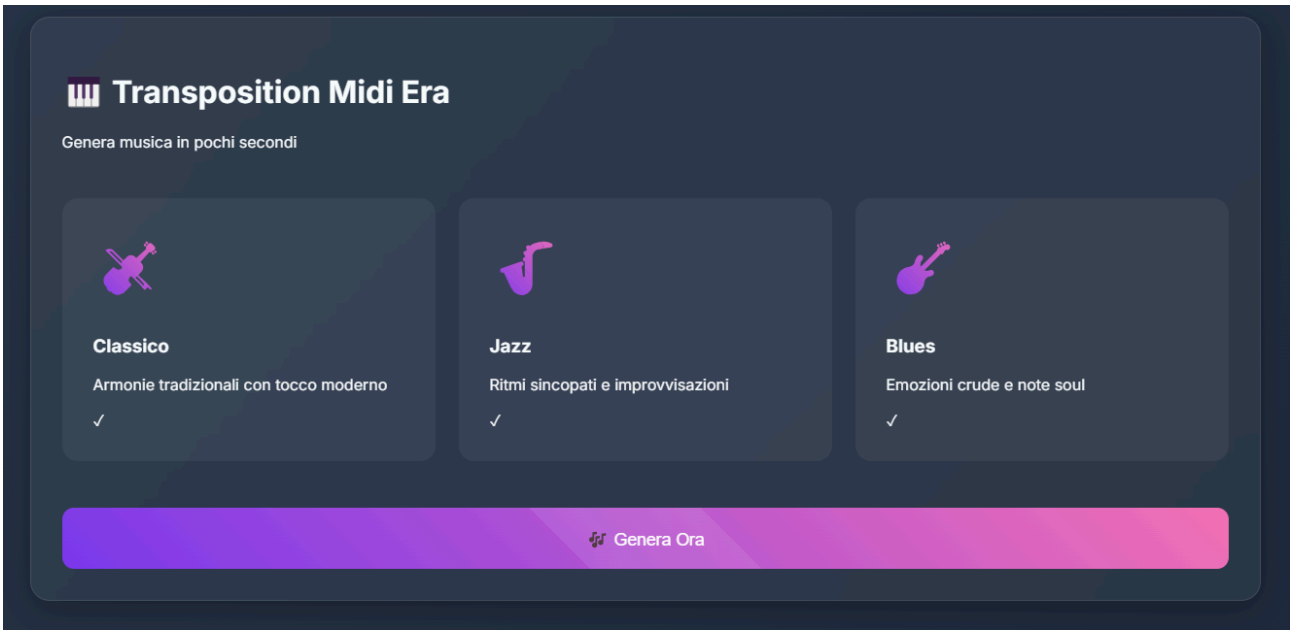


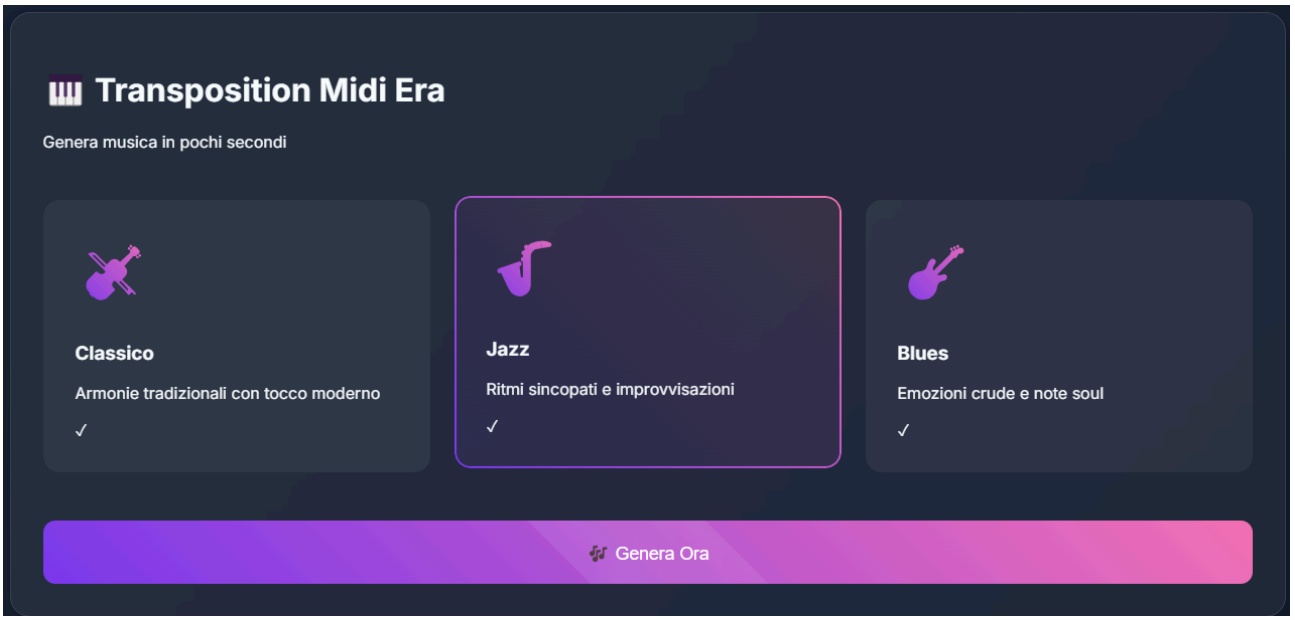
stream_obj.write('midi', fp=path): Esporta il flusso musicale come file MIDI nella cartella specificata.

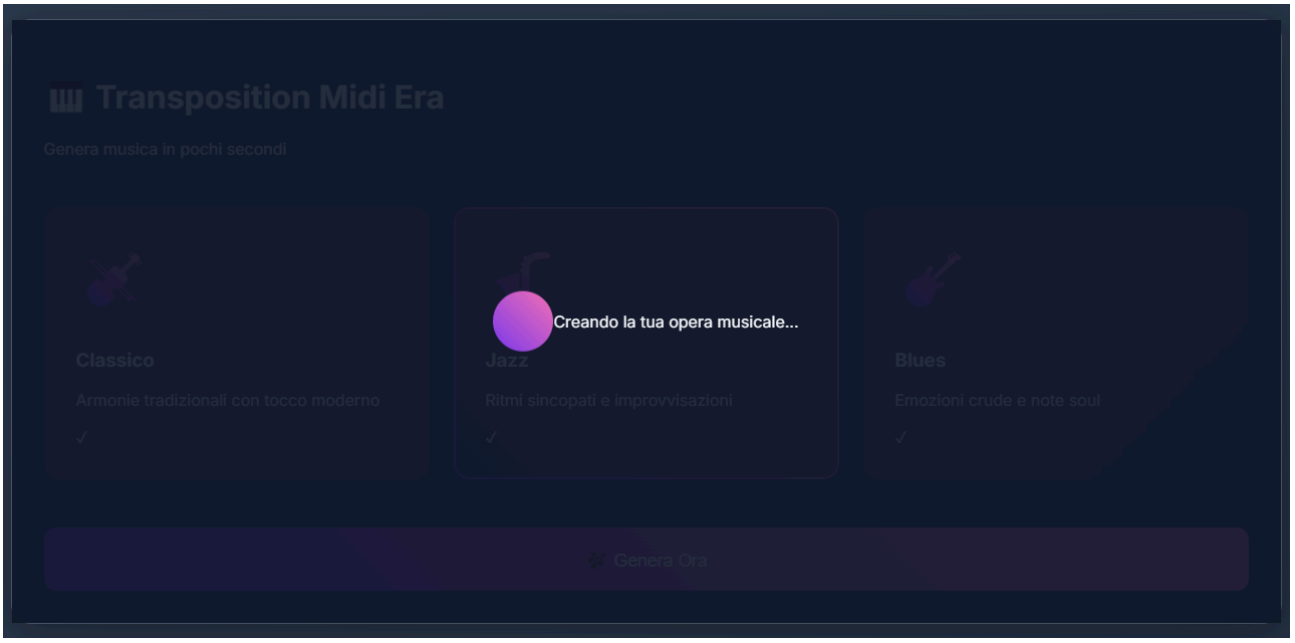
Sviluppi Futuri

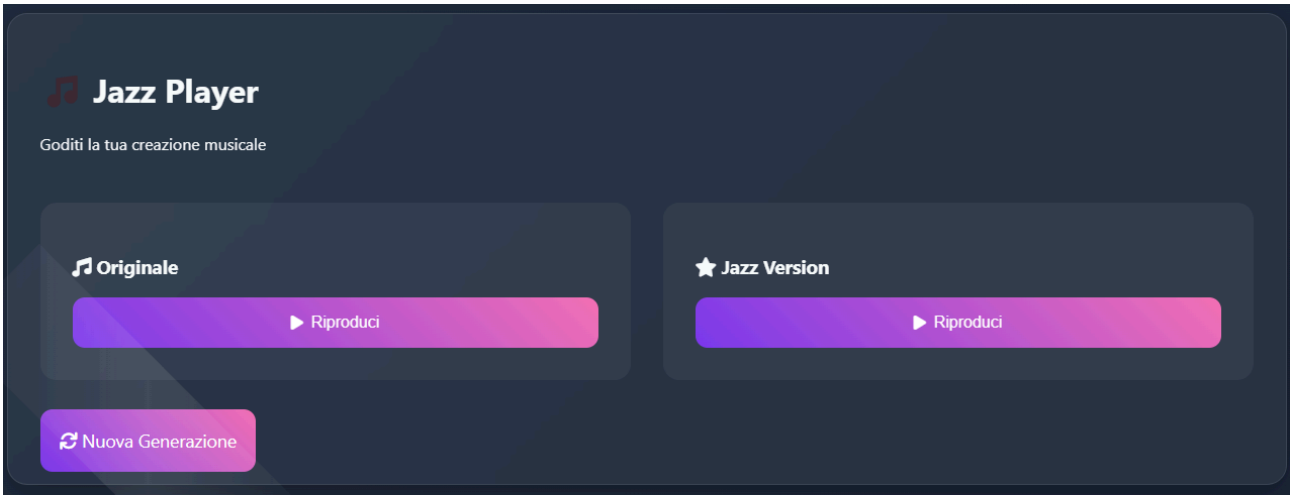
- **Integrazione con AI** per suggerire trasposizioni ottimali in base al genere musicale.
- **Supporto per più strumenti** con personalizzazione avanzata.
- **Conversione in altri formati audio** per una maggiore compatibilità.
- **Integrazione con piattaforme cloud** per il salvataggio automatico delle tracce.

Demo dell’applicazione sviluppata






Transposition Midi Era
Genera musica in pochi secondi
Classico
Armonie tradizionali con tocco moderno
Creando la tua opera musicale...
Jazz
Ritmi sincopati e improvvisazioni
Blues
Emozioni crude e note soul
Genera Ora


Jazz Player
Goditi la tua creazione musicale
Originale
Riproduci
Jazz Version
Riproduci
Nuova Generazione