

# Progetto Programmazione Web e Mobile

## Realizzazione di un'Applicazione Web

Il progetto era diviso in due parti **Frontend** and **Backend**

### Frontend

Il frontend è stato realizzato attraverso una combinazione di **HTML**, **CSS**, **JavaScript** e **Bootstrap** per la creazione di un'interfaccia web dinamica.

Le pagine html sono:

- La pagina **index.html** contiene le playlist pubbliche e una barra di ricerca per cercare un nome di playlist, un tag o una canzone al suo interno.
- La pagina **playlist.html** contiene le playlist di un utente. Un utente può creare nuovi playlist, modificare e cancellare le playlist.
- La pagina **search.html** contiene una barra di ricerca in cui l'utente può cercare le canzoni disponibili su Spotify utilizzando l'API di Spotify. Inoltre, visualizza informazioni dettagliate sulla canzone cercata e consente anche all'utente di aggiungere la canzone alla propria playlist.
- La pagina **profile.html** contiene le informazioni riguardanti l'utente, visualizzando i generi e gli artisti preferiti. L'utente ha la possibilità di modificare i propri dati, effettuare il logout e, inoltre, di cancellare il proprio account.
- La pagina **create.html** contiene i campi necessari per la creazione di una playlist. Ogni playlist è caratterizzata da un titolo, una descrizione e uno o più tag. Inoltre, l'utente ha la possibilità di impostare la playlist come privata o pubblica.
- La pagina **login.html** è un'interfaccia essenziale utilizzata per accedere all'applicazione. Presenta campi per inserire l'username e la password, offrendo anche la possibilità di registrarsi tramite un link dedicato alla registrazione.
- La pagina **registrati.html** contiene i campi necessari per la creazione di un utente. Un utente ha un Nome, Email e Password.

### Backend

Per gestire i dati e la logica lato server, ho utilizzato **MongoDB** ed **Express** insieme a **Node.js**. Inoltre, ho integrato **Swagger UI** per documentare e testare facilmente le API. Un elemento chiave di questo progetto è l'utilizzo **dell'API di Spotify**, che mi ha consentito di incorporare funzionalità legate alla piattaforma musicale direttamente nell'applicazione.

Il file principale per il backend è **index.js** dove sono definiti i ruoli dell'applicazione e fa partire il server.

### Database

Il database contiene due collections chiamate **users** e **playlist**.

AHMAD Hammad

La collection users ha documenti con questa struttura:

```
users = {  
  _id: ObjectId,  
  name: string,  
  password: string (hash of password string),  
  email: string,  
  generi: array di strings,  
  artist: array di strings  
}
```

```
{  
  "_id": ObjectId('655b4d23a3d0828d9f105072'),  
  "name": "hammad ahmad",  
  "email": "cd@gmail.com",  
  "password": "3d58301d074ddafde51a23f9cf18b4a6",  
  "artist": Array (3)  
    0: "Talha Anjum"  
    1: "Sfera Ebbasta"  
    2: "Eminem"  
  "generi": Array (6)  
    0: "black-metal"  
    1: "bossanova"  
    2: "acoustic"  
    3: "brazil"  
    4: "comedy"  
    5: "classical"  
}
```

La collection playlist ha documenti con questa struttura:

```
playlist = {  
  _id: ObjectId,  
  creator: string (ObjectId of the user),  
  title: string,  
  description: string,  
  tag: array di stringhe,  
  pubblica: booleano,  
  song: array di stringhe (id of the song),  
  like: array di stringhe (ObjectId of the user that liked the playlist)  
}
```

```
{  
  "_id": ObjectId('65646900821d6a308638845c'),  
  "creator": "655b4d23a3d0828d9f105072",  
  "title": "hammad new",  
  "description": "new",  
  "tag": Array (1)  
    0: "rock"  
  "pubblica": true  
  "like": Array (1)  
    0: "655b4d23a3d0828d9f105072"  
  "song": Array (1)  
    0: "6aD0Es1eyfVqpYZKJm9WkC"  
}
```

Le richieste HTTP per interagire con le operazioni definite nel file **index.js** possono essere effettuate utilizzando i seguenti nomi di endpoint:

### Operazioni GET:

#### 1. Ottieni Playlist

- Percorso: **/playlist/**
- Descrizione: Recupera informazioni su tutti i playlist pubblici

#### 2. Ottieni Utente per ID

- Percorso: **/users/{id}**
- Descrizione: Recupera informazioni su un utente attraverso il suo ID.

#### 3. Ottieni Tutti gli Utenti

- Percorso: **/users**
- Descrizione: Recupera una lista di tutti gli utenti.

**4. Ottieni Informazioni sulla Playlist per ID**

- Percorso: **/playlist/{id}/info**
- Descrizione: Recupera informazioni su una playlist attraverso il suo ID.

**5. Ottieni Playlist per ID**

- Percorso: **/playlist/{id}**
- Descrizione: Recupera informazioni su una playlist attraverso il suo ID.

**Operazioni POST:**

**1. Crea Utente**

- Percorso: **/users**
- Descrizione: Crea un nuovo utente.

**2. Crea Playlist**

- Percorso: **/playlist**
- Descrizione: Crea una nuova playlist.

**3. Login**

- Percorso: **/login**
- Descrizione: Effettua il login dell'utente.

**Operazioni PUT:**

**1. Aggiorna Utente per ID**

- Percorso: **/users/{id}**
- Descrizione: Aggiorna le informazioni su un utente attraverso il suo ID.

**2. Aggiorna Playlist per ID**

- Percorso: **/playlist/{id}**
- Descrizione: Aggiorna le informazioni su una playlist attraverso il suo ID.

**3. Aggiorna Preferenze Musicali dell'Utente per ID**

- Percorso: **/users/{id}/genere**
- Descrizione: Aggiorna le preferenze musicali di un utente attraverso il suo ID.

**4. Aggiungi like alla Playlist per ID**

- Percorso: **/like/{playlistid}**
- Descrizione: Aggiunge un like a una playlist attraverso il suo ID.

**5. Aggiungi Brano alla Playlist per ID**

- Percorso: **/playlist/{id}/song**
- Descrizione: Aggiunge un brano a una playlist attraverso il suo ID.

**6. Aggiorna Preferenze Artisti dell'Utente per ID**

- Percorso: **/users/{id}/artist**
- Descrizione: Aggiorna le preferenze degli artisti di un utente attraverso il suo ID.

**Operazioni DELETE:**

**1. Elimina Utente per ID**

- Percorso: **/users/{id}**
- Descrizione: Elimina un utente attraverso il suo ID.

**2. Elimina Playlist per ID**

- Percorso: **/playlist/{id}**
- Descrizione: Elimina una playlist attraverso il suo ID.

**3. Rimuovi Brano dalla Playlist per ID e Nome del Brano**

- Percorso: **/playlist/{id}/{song}**
- Descrizione: Rimuove un brano specifico da una playlist fornendo sia l'ID della playlist che il nome del brano.

**4. Elimina Preferenze Musicali dell'Utente per ID**

- Percorso: **/users/{id}/genere**
- Descrizione: Elimina le preferenze musicali di un utente attraverso il suo ID.

**5. Elimina Preferenze Artisti dell'Utente per ID**

- Percorso: **/users/{id}/artist**
- Descrizione: Elimina le preferenze degli artisti di un utente attraverso il suo ID.

## Swagger UI:

GET	/users/{id}
PUT	/users/{id}
DELETE	/users/{id}
GET	/users
POST	/users
POST	/login
GET	/
POST	/playlist
GET	/playlist
GET	/playlist/{id}/info
PUT	/playlist/{id}/song
PUT	/playlist/{id}/song
DELETE	/playlist/{id}/{song}
PUT	/playlist/{id}
DELETE	/playlist/{id}
GET	/playlist/{id}
PUT	/users/{id}/genere
DELETE	/users/{id}/genere
PUT	/users/{id}/artist
DELETE	/users/{id}/artist
PUT	/like/{playlistId}

## Generare Token

```
const BASE = "https://api.spotify.com/v1/"; //base for spotify api
const client_id = '13dba2d798ed4c35a07254e94aacf2bd'; //client id to generate token
const client_secret = '228fdf9d83bd47ecadea512beb2af31d'; //client secret
var url = "https://accounts.spotify.com/api/token"; //url to get token

// Retrieve an access token using client credentials
async function getToken() {
  const response = await fetch(url, {
    method: "POST",
    headers: {
      Authorization: "Basic " + btoa(`${client_id}:${client_secret}`),
      "Content-Type": "application/x-www-form-urlencoded",
    },
    body: new URLSearchParams({ grant_type: "client_credentials" }),
  });
  const tokenResponse = await response.json();
  const token = tokenResponse.access_token;
  sessionStorage.setItem("access", token);
  return;
}
```