Spotify API dashboard

Analisi e visualizzazioni di dati musicali

Bartolomei Daniel – Rover Simone

API vs Dati statici

API

- Dati aggiornati in tempo reale
- Dati generalmente affidabili
- Accesso previa autorizzazione
- Tempo di attesa per ottenimento dei dati (dipendente dal server)

Dati statici

- Facile ottenimento
- Facile accesso
- Tempo di ottenimento pressoché nullo
- Potenzialmente obsoleti
- Da verificare

Autorizzazione per l'accesso ad una API

- Per accedere ai dati forniti da una API è necessario autenticarsi tramite uno o più identificativi registrati. Generalmente si tratta di chiavi o token.
- Esistono diversi tipi di token a seconda dell'uso che si vuole fare dei dati o di chi li sta richiedendo.
- Occorre assicurarsi che il token in utilizzo sia valido e non scaduto.

Tipi di autorizzazione e token (Spotify)

App authorization

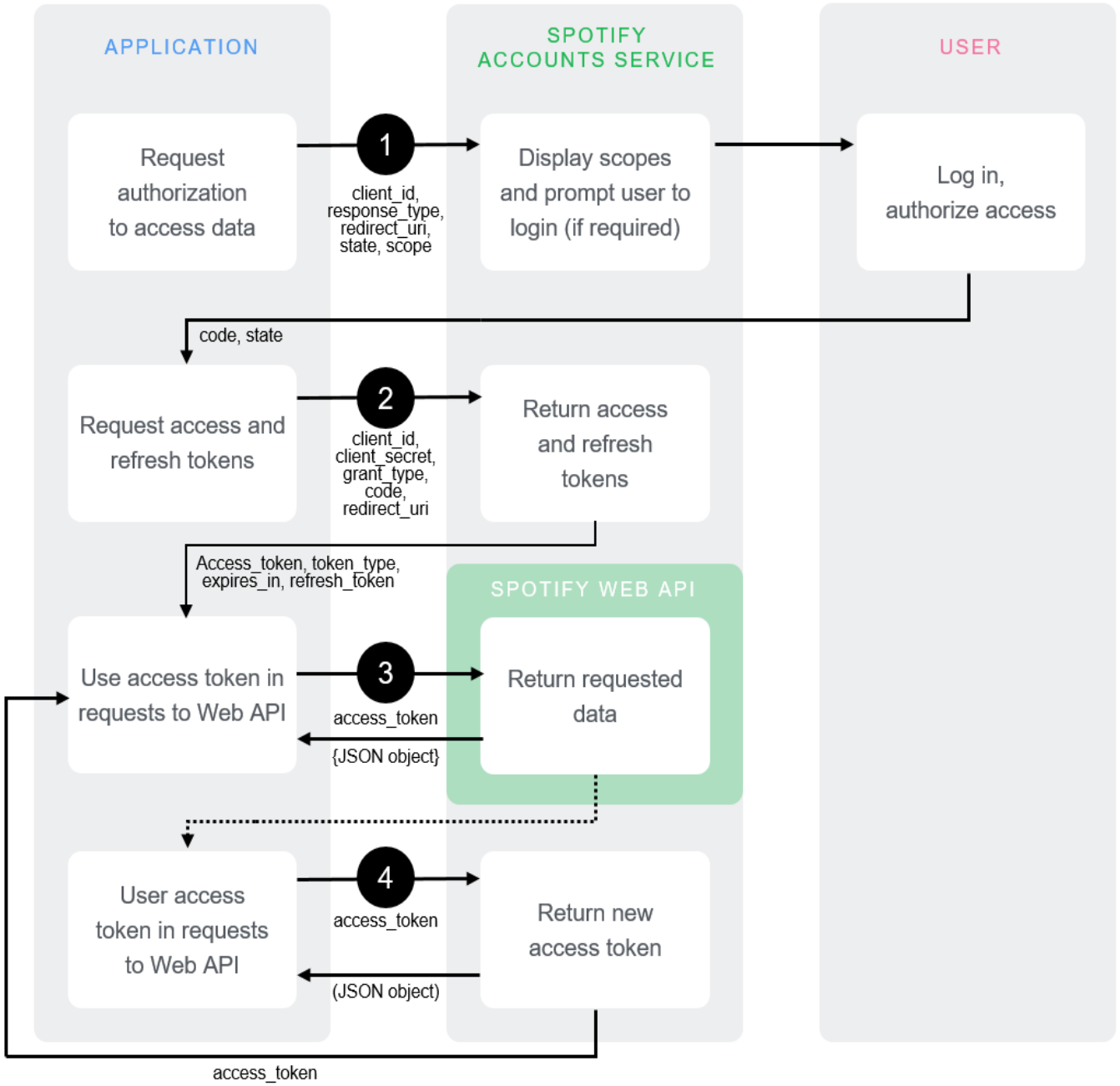
- Richiede secret-key (lato server)
- Token a scadenza fissata e non aggiornabile
- Non richiede autorizzazione dall'utente (ma non può accedere a dati privati)

User authorization

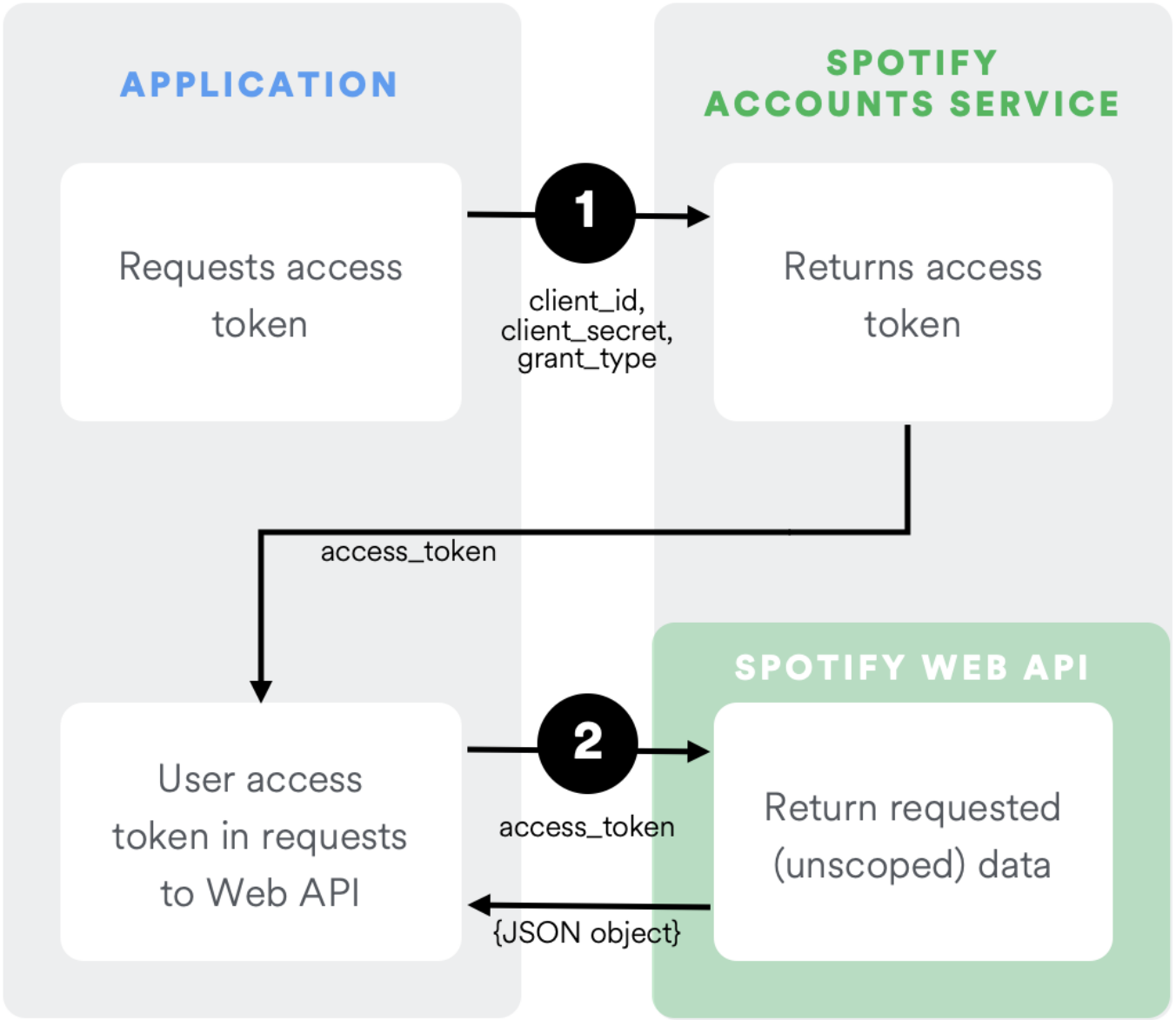
- Richiede autorizzazione utente (esplicita o implicita)
- Può accedere a dati privati tramite 'scopes'
- Scadenza aggiornabile

Documentazione ufficiale: Spotify auth guide

User auth token



App auth token



Ottenimento dell'access token

Una volta identificato il tipo di autorizzazione di cui si ha bisogno, si può procedere alla richiesta dell'**access token**, con il quale otterremo i dati dall'Application Programming Interface.

Abbiamo utilizzato una autorizzazione applicativa.

Di seguito è riportata la funzione che ci consente di ottenere l'access token.

Spotify richiede che 'clientid' e 'secretid' siano formattate in una stringa con ':' come separatore e codificata in base64.

```python
import requests as req
import base64


# returns the response for the access token (raw, not json)
def get_token_response(clientid, secretid):
    url = 'https://accounts.spotify.com/api/token'
    to_encode = '{0}:{1}'.format(clientid, secretid)
    to_bytes = to_encode.encode('ascii')
    base64bytes = base64.b64encode(to_bytes)
    base64message = base64bytes.decode('ascii')
    myheaders = {
        'Authorization': 'Basic {0}'.format(base64message)
    }
    mydata = {
        'grant_type': 'client_credentials'
    }
    resp = req.post(url, headers=myheaders, data=mydata)
    return resp
```

Ottenimento dei dati

- Utilizziamo l'access token (ed eventualmente altri parametri passati come query all'interno dell'URL) per ottenere i dati desiderati. Se la richiesta è andata a buon fine otterremo un codice di risposta HTTP 200, altrimenti il pacchetto conterrà un codice di errore.
- In caso di riuscita, i dati avranno formato JSON. E' possibile trasporre i JSON Objects ottenuti in un Dataframe tramite la funzione predisposta da Pandas: Pandas.json_normalize(json_object).
- In Python trattiamo un dato di tipo JSON come un dizionario. Ogni valore è accessibile specificando la relativa chiave.

```
playlist_name = 'Top 50'
sample = search_for_playlist(playlist_name, access_token)
print(sample)
```

```
<Response [200]>
```

```
def search_for_playlist(playlistname, accesstoken):
    baseurl = 'https://api.spotify.com/v1/search'
    query_name = str(playlistname).replace(' ', '+')
    query = '?q={0}&type=playlist'.format(query_name)
    url = baseurl + query
    my_headers = get_headers(accesstoken)
    resp = req.get(url, headers=my_headers)
    return resp
```

Risposta JSON

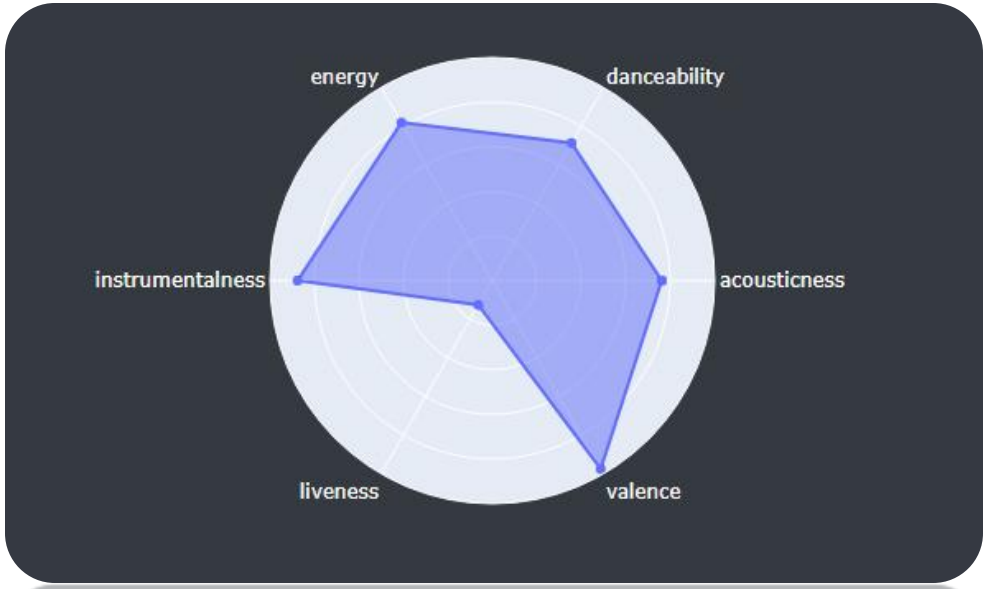
- Le '{' delimitano oggetti
- Le '[' delimitano liste
- A sinistra dei ':' ci sono le chiavi


```
{
    "artists": {
        "href": "https://api.spotify.com/v1/search?query=tania+bowra&offset
        "items": [
            {
                "external_urls": {
                    "spotify": "https://open.spotify.com/artist/08td7MxkoHQ
                },
                "genres": [],
                "href": "https://api.spotify.com/v1/artists/08td7MxkoHQkXnW
                "id": "08td7MxkoHQkXnWAYD8d6Q",
                "images": [
                    {
                        "height": 640,
                        "url": "https://i.scdn.co/image/f2798ddab0c7b76dc2d
                        "width": 640
                    },
```


Documentazione ufficiale: Spotify web API

Caratteristiche audio di Spotify

- Spotify classifica i brani in base a sette 'features': **danceability**, **energy**, **acousticness**, **instrumentalness**, **liveness**, speechiness, **valence**. Tutti questi sono valori decimali compresi tra 0 e 1. Dato che speechiness indica quanto una traccia ha al suo interno parole non cantate, è quasi sempre 0 e, quindi, la tralasceremo.



Plotly e Dash

- Plotly ci consente di visualizzare i dati ottenuti da Spotify. Abbiamo principalmente utilizzato Plotly.graph_objects poiché consente una stilizzazione maggiore rispetto a Plotly.express.
- Dash è una libreria creata appositamente per lavorare con Plotly. Incapsula tag e strutture HTML al fine di creare un layout di pagina e un file system del progetto. Consente di creare dashboard interattive.

Documentazione ufficiale: Plotly

Dash

- Ad un progetto corrisponde una applicazione dash. Ad ogni pagina dell'applicazione corrisponde un file python.
- Ogni file è suddiviso in due parti principali: layout e callbacks.
- Il layout contiene strutture HTML che saranno riempite con i dati opportuni.
- Un'interazione con la dashboard è tradotta in un callback di una funzione. Ognuna di queste prende come input valori presenti nel layout e, dopo una eventuale elaborazione, procede con la restituzione delle variabili desiderate.
- E' possibile restituire intere strutture HTML da una funzione di callback.

Documentazione ufficiale: Dash

Dash callbacks

- Ogni callback ha come parametro il valore di una proprietà di un elemento HTML presente nel layout. Deve essere inoltre specificato dove andare a collocare i dati restituiti dalla funzione.
- Per specificare questi parametri vengono usati dei particolari strumenti importanti direttamente da Dash.
- Output('component_id', 'component_attribute') specifica dove collocare i dati.
- Input/State('component_id', 'component_attribute') specifica i parametri in input.
- State non è un trigger per il callback, mentre Input sì.

```python

@app.callback(
    Output('page_layout', 'children'),
    [Input('url', 'pathname')]
)
def display_page(path):
    print(path)
    if path == '/apps/tendenze':
        return tendenze.layout
    elif path == '/apps/artist':
        return artist.layout
    elif path == '/apps/confronta':
        return confronta.layout
    elif path == '/':
        return homepage.layout
    else:
        error_layout = dbc.Jumbotron(
            [
                html.H1("404: Not Found", className="text-danger"),
                html.Hr(),
                html.P(f"The pathname {path} was not recognized...")
            ]
        )
        return error_layout

```

Callback per aggiornare la pagina visualizzata dentro index.py

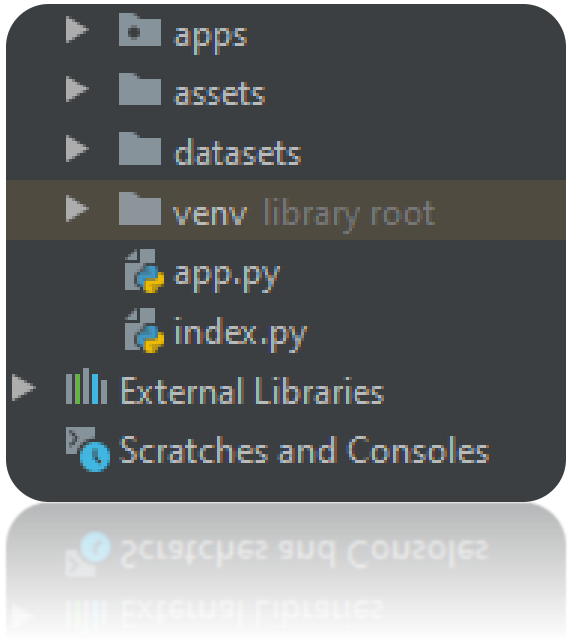
Pytrends

- Pytrends consente di estrapolare dati relativi alle ricerche. Lo abbiamo usato per creare la linea di trend per gli artisti e il trend in base alla nazione.

Documentazione ufficiale: Pytrends

File system del progetto

- 'apps': contiene i file python delle pagine della dashboard e un file __init__.py
- 'assets': contiene risorse di layout (immagini, navbar, …)
- 'datasets': contiene i file csv
- 'app.py': contiene le informazioni dell'applicazione che devono essere condivise con gli altri file python
- 'index.py': homepage della dashboard



Datasets esterni

- Spotify non consente di ottenere dati a livello generale (esempio: dataframe di tutti gli artisti registrati). Per ottenere dati non specifici utilizziamo sei datasets esterni costruiti utilizzando l'API di Spotify.
- Questo dataset contiene informazioni riguardanti i generi musicali e le loro informazioni.
- I dati sono aggiornati a dicembre 2020.

Fonte: Spotify datasets

Dati presentati sulla dashboard

Dati relativi ai generi musicali

- PCA delle caratteristiche audio
- Scatterpolar delle caratteristiche audio
- Caratteristiche audio nel tempo

Dati relativi alle playlist e album in tendenza

- Elenco
- Distribuzione caratteristiche audio
- PCA con playlist di appartenenza e popolarità dei brani

Dati relativi agli artisti

- Informazioni generali
- Media caratteristiche audio
- Informazioni dei correlati
- Popolarità album e interesse nel tempo
- Interesse in base alla nazione

Pagina per il confronto tra due artisti

- Confronto caratteristiche audio
- Confronto popolarità

Migliorie e aggiornamenti possibili

- Aggiungere l'utilizzo di uno user token per visualizzare dati realitivi alle canzoni degli utenti.
- Rendere automatico l'aggiornamento dei dataset.
- Visualizzare informazioni riguardanti le Categorie (vedi Spotify API doc).
- Aggiungere una funzionalità suggeriti in base alle caratteristiche audio delle tracce preferite di un utente.
- Possibile creazione di una playlist personalizzata in base alle preferenze di un utente.

Esiste già un sito che consente la visualizzazione dei brani più ascoltati da un utente: statsforspotify.com

Disclaimer

- Dato il disservizio accaduto l'8/6/2021, alcune feature della dashboard potrebbero non funzionare correttamente.