# ALMA MATER STUDIORUM – UNIVERSITA**''**DI BOLOGNA

DIPARTIMENTO DI SCIENZE STATISTICHE

“PAOLO FORTUNATI”

Corso di Laurea in Scienze Statistiche – Economia e Impresa

**Web App / Dashboard Analitica Analisi Dati API Spotify**

**(Informatica)**

|  |  |
| --- | --- |
| **Presentata da:** | **Relatore:** |
| Emanuele Pepe | Prof Claudio Sartori |
| Matricola: 922986 |  |

APPELLO III

ANNO ACCADEMICO 21/22

**INDICE**

[Abstract 5](#__RefHeading___Toc1160_2631891455)

[Introduzione 6](#__RefHeading___Toc352_568587729)

[API (Application Programming Interface) 7](#__RefHeading___Toc364_568587729)

[Introduzione 7](#__RefHeading___Toc366_568587729)

[Web API 7](#__RefHeading___Toc844_3232970131)

[Protocollo HTTP – Trasferimento Dati 8](#__RefHeading___Toc368_568587729)

[Metodi Richieste HTTP 9](#__RefHeading___Toc1162_2631891455)

[GET 9](#__RefHeading___Toc370_568587729)

[HEAD 9](#__RefHeading___Toc1164_2631891455)

[POST 10](#__RefHeading___Toc372_568587729)

[PUT 10](#__RefHeading___Toc374_568587729)

[DELETE 10](#__RefHeading___Toc1166_2631891455)

[CONNECT 10](#__RefHeading___Toc1168_2631891455)

[OPTIONS 10](#__RefHeading___Toc1170_2631891455)

[TRACE 10](#__RefHeading___Toc1172_2631891455)

[PATCH 11](#__RefHeading___Toc376_568587729)

[Modello REST 11](#__RefHeading___Toc380_568587729)

[Spotify Web API 12](#__RefHeading___Toc382_568587729)

[Introduzione 12](#__RefHeading___Toc384_568587729)

[Autenticazione e Autorizzazione (Authorization Code Flow) 12](#__RefHeading___Toc386_568587729)

[Raccolta Dati: Endpoints 13](#__RefHeading___Toc1165_1609035006)

[Get User's Saved Tracks 13](#__RefHeading___Toc390_568587729)

[Get Track's Audio Features 15](#__RefHeading___Toc392_568587729)

[Get Several Artists 18](#__RefHeading___Toc707_1710397322)

[R Shiny: Libreria Per Costruire Web App Interattive 20](#__RefHeading___Toc709_1710397322)

[Introduzione 20](#__RefHeading___Toc711_1710397322)

[UI 20](#__RefHeading___Toc713_1710397322)

[Server 20](#__RefHeading___Toc715_1710397322)

[Programmazione Reattiva 20](#__RefHeading___Toc717_1710397322)

[Analisi Dati 21](#__RefHeading___Toc400_568587729)

[Data Wrangling e Data Cleaning 21](#__RefHeading___Toc1104_2460335997)

[Note sulla scelta delle variabili 25](#__RefHeading___Toc1174_2631891455)

[Analisi Univariata (Distribuzioni, Visualizzazione) 27](#__RefHeading___Toc402_568587729)

[Analisi Multivariata (Correlazioni….) 27](#__RefHeading___Toc404_568587729)

[Cluster Analysis (k-means; Archetypal Analysis) 27](#__RefHeading___Toc406_568587729)

[PCA (Principal Components Analysis) 27](#__RefHeading___Toc408_568587729)

[Output 28](#__RefHeading___Toc959_1609035006)

[Dashboard Analitica 28](#__RefHeading___Toc719_1710397322)

[Insights Derivanti Dall](#__RefHeading___Toc721_1710397322)''[Analisi Dati 28](#__RefHeading___Toc721_1710397322)

[Conclusioni 29](#__RefHeading___Toc1106_2460335997)

[Bibliografia 30](#__RefHeading___Toc1108_2460335997)

[Appendice: Controllo Versione – GIT 32](#__RefHeading___Toc755_2703219746)

[Definizione 32](#__RefHeading___Toc757_2703219746)

[Utilizzo 32](#__RefHeading___Toc759_2703219746)

[GIT 32](#__RefHeading___Toc761_2703219746)

[GitHub 32](#__RefHeading___Toc763_2703219746)

# 

# Abstract

# Introduzione

# API (Application Programming Interface)

## Introduzione

Un’Application Programming Interface (API) è un sistema di procedure e protocolli con il quale due o più programmi informatici interagiscono e comunicano tra di loro.

La documentazione che descrive il sistema di procedure *specifica* un''API, mentre un sistema informatico che soddisfa tale sistema *implementa* o *espone* un’API.

Un’API è spesso organizzata in sezioni di istruzioni che offrono specifici servizi e svolgono determinate funzioni; queste sezioni vengono spesso chiamate *endpoints.*

Il termine API viene spesso utilizzato per riferirsi alle Web API, le quali consentono la comunicazione tra computer collegati tramite Internet. Esistono anche API per linguaggi di programmazione, librerie software, sistemi operativi e hardware.

In questo progetto di tesi ci concentreremo sulle API web.

## Web API

Un''API web è un'interfaccia di programmazione per un server web o un web browser (client). Si tratta di un concetto di sviluppo web ed è necessario distinguere lato client di un'applicazione web e lato server.

Un''API web lato server è un'interfaccia programmatica costituita da uno o più endpoint esposti con un sistema definito di messaggi richiesta-risposta, tipicamente espresso in JSON (JavaScript Object Notation)[[1]](#footnote-2) o XML (Extensible Markup Language)[[2]](#footnote-3), che viene esposto tramite il web, più comunemente per mezzo di un server web basato su HTTP.

Gli endpoint sono aspetti importanti dell'interazione con le API web lato server, in quanto specificano dove si trovano le risorse a cui possono accedere i software di terze parti. Di solito l'accesso avviene tramite un URI[[3]](#footnote-4) a cui vengono inviate le richieste HTTP (Hypertext Transfer Protocol, esaminato più in dettaglio in seguito) e da cui ci si aspetta la risposta. Le API Web possono essere pubbliche o private, in quest'ultimo caso richiedono un token di accesso.

Un''API web lato client è un'interfaccia programmatica per estendere le funzionalità di un browser web o di un altro client HTTP.

## Protocollo HTTP – Trasferimento Dati

L'Hypertext Transfer Protocol (HTTP) è un protocollo di livello applicativo della suite di protocolli di rete Internet TCP/IP[[4]](#footnote-5). È utilizzato come principale sistema per la trasmissione di informazioni sul web, tipicamente in un’architettura client-server. I dati vengono scambiati attraverso una sequenza di messaggi di richiesta-risposta.

Un client HTTP cerca inizialmente di connettersi a un server stabilendo una connessione (reale o virtuale). Un server HTTP in ascolto accetta la connessione e attende il messaggio di richiesta del client. A questo punto il client invia la richiesta al server. Una volta ricevuta la richiesta, il server invia un messaggio di risposta HTTP (contenente anche *header* e *body,* se necessario). Il body di questo messaggio è in genere la risorsa richiesta (dati), anche se può essere restituito un messaggio di errore o altre informazioni.

Le richieste inviate dal client al server consistono in: una *request line,* composta dal metodo di richiesta (analizzate in dettaglio in seguito), ; zero o più campi di intestazione della richiesta (*header*) con all’interno informazioni aggiuntive; una riga vuota, composta da un carattere *carriage return* e un carattere *line feed*; un *body* di messaggio opzionale.

## Metodi Richieste HTTP

Il protocollo HTTP definisce dei metodi per indicare l'azione da eseguire sulla risorsa web identificata nel server. Che cosa rappresenti questa risorsa, se dati preesistenti o dati generati dinamicamente, dipende dall'implementazione del server.

A seconda delle versioni del protocollo HTTP, più o meno recenti, sono identificati nove metodi principali: GET, HEAD, POST, PUT, DELETE, CONNECT, OPTIONS, TRACE e PATCH. Non c'è limite al numero di metodi che possono essere definiti. Questo consente di specificare metodi futuri senza interrompere l'infrastruttura esistente. Ogni client può utilizzare qualsiasi metodo e il server può essere configurato per supportare qualsiasi combinazione di metodi. Se un metodo è sconosciuto a un intermediario, questo sarà trattato come un metodo non sicuro.

### GET

Il metodo GET richiede che la risorsa di destinazione nel server trasferisca una rappresentazione completa del suo stato. Le richieste GET devono solo recuperare dati e non devono avere altri effetti, ovvero devono recuperare risorse senza apportare modifiche.

### HEAD

Il metodo HEAD richiede che la risorsa di destinazione trasferisca una rappresentazione del suo stato, come per una richiesta GET, ma senza i dati della rappresentazione racchiusi nel corpo della risposta. Questo è utile per recuperare i metadati della rappresentazione nell'intestazione della risposta, senza dover trasferire l'intera rappresentazione. Gli usi includono la verifica della disponibilità di una pagina o la ricerca della dimensione di un file.

### POST

Il metodo POST consente di inviare dati e informazioni al server. Richiede che il server elabori la rappresentazione allegata alla richiesta, i dati, secondo la semantica della risorsa di destinazione. Questo metodo viene utilizzato spesso per trasferire dati di autorizzazione o autenticazione in maniera sicura. Esploreremo questo aspetto in seguito per l'autenticazione con il server di Spotify.

### PUT

Il metodo PUT viene utilizzato per aggiornare la risorsa disponibile sul server. In genere, sostituisce tutto ciò che è contenuto nell'URL di destinazione con qualcos'altro. Si può usare per creare una nuova risorsa o sovrascriverne una esistente.

### DELETE

Il metodo DELETE richiede che la risorsa di destinazione cancelli il suo stato, ovvero elimini la risorsa presente all’URL specificato.

### CONNECT

Il metodo HTTP CONNECT avvia una comunicazione bidirezionale con la risorsa richiesta. Può essere utilizzato per aprire un *tunnel.* Questo metodo richiede di stabilire un tunnel TCP/IP verso il server di origine identificato dalla destinazione della richiesta.

### OPTIONS

Il metodo OPTIONS richiede che la risorsa di destinazione trasferisca i metodi HTTP che supporta. Richiede le opzioni di comunicazione consentite per un determinato URL o server.

### TRACE

Il metodo HTTP TRACE è progettato per scopi diagnostici. Se abilitato, il server web risponderà alle richieste che utilizzano il metodo TRACE riproducendo nella risposta l'esatta richiesta ricevuta. Questo metodo richiede che la risorsa di destinazione nel server trasferisca la richiesta ricevuta nel corpo della risposta. In questo modo un client può vedere quali (se ci sono) modifiche o aggiunte sono state fatte. Per questo motivo questo metodo ha scopi di diagnostica e controllo.

### PATCH

Il metodo PATCH è un metodo di richiesta in HTTP per apportare modifiche parziali a una risorsa esistente. Applica modifiche parziali a una risorsa e in questo modo questo modo si può risparmiare larghezza di banda aggiornando una parte di un file o di un documento senza doverlo trasferire interamente.

Tutti i server web generici devono implementare almeno i metodi GET e HEAD, mentre tutti gli altri metodi sono considerati opzionali dalla specificazione del server.

Per le finalità della stesura di questa tesi sono stati utilizzati solo i metodi GET e POST.

# Spotify Web API

## Introduzione

Spotify è un fornitore di servizi di streaming audio e media di proprietà svedese fondato il 23 aprile 2006 da Daniel Ek e Martin Lorentzon che offre musica registrata e podcast con diritti d'autore limitati,

Spotify espone un’API webattraverso la quale è possibile sviluppare un’applicazione web che recuperi il contenuto di Spotify.

Per accedere ai dati relativi a un utente attraverso l'API Web, un'applicazione deve essere autorizzata dall'utente ad accedere a quelle particolari informazioni.

L’API web di Spotify[[5]](#footnote-6) restituisce, tra le altre informazioni, meta-dati JSON su artisti musicali, album e brani direttamente dal catalogo dati di Spotify.

L''API web fornisce inoltre l'accesso ai dati relativi all'utente, come dati di registrazione, playlist salvate e azioni svolte durante l’utilizzo di Spotify; ad esempio, brani musicali salvati nella propria libreria o artisti “seguiti”. Tale accesso è abilitato tramite autorizzazione selettiva, da parte dell'utente.

L'indirizzo di base della Web API è https://api.spotify.com. L'API fornisce un set di endpoint, ognuno con il proprio percorso univoco. Come scritto in precedenza, per accedere ai dati privati ​​tramite l'API Web, come i profili utente e le playlist, un'applicazione deve ottenere l'autorizzazione dell'utente per accedere ai dati.

L'autorizzazione avviene tramite il servizio Account Spotify .

## Autenticazione e Autorizzazione (Authorization Code Flow)

La procedura di autorizzazione consta in diverse fasi e si struttura nel seguente modo.

Per prima cosa si ottiene un codice (Client ID) dal servizio di account Spotify, quindi si scambia quel codice con un token di accesso. Lo scambio da codice a token richiede una chiave segreta (Client Secret) e, per motivi di sicurezza, avviene tramite la comunicazione diretta da server a server.

Il primo passo è richiedere l'autorizzazione all'utente, in modo che la nostra app possa accedere alle risorse di Spotify per conto di quell'utente. Per fare ciò, la nostra applicazione deve compilare e inviare una richiesta GET /authorize all'endpoint con i seguenti parametri

## Raccolta Dati: Endpoints

Per la finalità di questa tesi, ho interrogato tre specifici endpoints della web API di Spotify. Saranno successivamente illustrati nel dettaglio le Query delle richieste ai singoli endpoint e i Body delle risposte a tali richieste.

Il linguaggio utilizzato per eseguire le richieste HTTP è stato R[[6]](#footnote-7), ed è stato possibile sfruttare un pacchetto presente in CRAN, *“*spotifyr”[[7]](#footnote-8)*,* un wrapper R per l'estrazione di dati dall'API web di Spotify.

Nel body delle risposte sono presenti i dati esposti negli specifici endpoint, che in questo caso riguardano i miei brani salvati in libreria, le caratteristiche audio di questi brani salvati e gli artisti, con i relativi attributi, dei brani. In un secondo momento questi dati verranno organizzati e utilizzati per condurre delle analisi statistiche.

I tre endpoints in questione sono: Get User's Saved Tracks; Get Track's Audio Features; Get Several Artists.

Di seguito i dettagli dei singoli endpoints.

### Get User's Saved Tracks

Restituisce un elenco dei brani salvati nella libreria "La tua musica" dell'utente Spotify corrente.

**Request.Query**

|  |  |
| --- | --- |
| Limit - integer | Il numero massimo di elementi da restituire. Valore predefinito: 20. Minimo: 1. Massimo: 50. |
| Market - string | Un codice paese ISO 3166-1 alpha-2. Se viene specificato un codice paese, verranno restituiti solo i contenuti disponibili in quel mercato.  Se nell'intestazione della richiesta è specificato un token di accesso utente valido, il Paese associato all'account utente avrà la priorità su questo parametro. |
| Offset - integer | L'indice del primo elemento da restituire. Predefinito: 0 (il primo elemento). Usare con limit per ottenere la serie successiva di elementi. |

**Response.Body**

|  |  |
| --- | --- |
| Href - string | Un link all'endpoint dell'API Web che restituisce il risultato completo della richiesta. |
| Items - array | Il contenuto richiesto |
| Limit - integer | Il numero massimo di elementi nella risposta (come impostato nella query o come predefinito). |
| Next – string or null | URL alla pagina successiva di elementi. (nullo se non presente) |
| Offset - integer | L'offset degli elementi restituiti (come impostato nella query o come predefinito) |
| Previous – string or null | URL alla pagina precedente di elementi. (nullo se non presente) |
| Total - integer | Il numero totale di articoli disponibili per la restituzione. |

### Get Track's Audio Features

Restituisce caratteristiche audio per più brani in base ai loro ID Spotify.

**Request.Query**

|  |  |
| --- | --- |
| Ids - string | Un elenco separato da virgole degli ID Spotify dei brani. Massimo: 100 ID. |

**Response.Body**

|  |  |
| --- | --- |
| Danceability - float | La “Danceability” è un indice che descrive quanto una canzone sia adatta per ballare sulla base di una combinazione di elementi musicali tra cui: tempo, stabilità del ritmo, forza del ritmo e regolarità generale. Un valore di 0.0 è meno ballabile e 1.0 è più ballabile. |
| Energy - float | L'energia è una misura che va da 0,0 a 1,0 e rappresenta una misura percettiva dell'intensità e dell'attività. In genere, i brani energetici sono veloci, forti e rumorosi. Ad esempio, il death metal ha un'energia elevata, mentre un preludio di Bach ha un punteggio basso su questa scala. Le caratteristiche percettive che contribuiscono a questo attributo includono il range dinamico del brano, il volume percepito, il timbro, la velocità di insorgenza e l'entropia generale. |
| Key - integer | La chiave (nota) in cui è composto il brano. Gli interi corrispondono alle tonalità utilizzate nella notazione standard della Classe di Altezze. Ad esempio, 0 = C, 1 = C♯/D♭, 2 = D e così via. Se non è stata rilevata alcuna tonalità, il valore è -1. |
| Loudness - float | La ‘Rumorosità''complessiva di una traccia in decibel (dB). I valori di rumorosità sono calcolati in media sull'intero brano e sono utili per confrontare il volume relativo dei brani. La rumorosità è la qualità di un suono che è principalmente correlata alla forza fisica (ampiezza). I valori sono tipicamente compresi tra -60 e 0 dB. |
| Mode - integer | Il modo indica la modalità (maggiore o minore) di un brano, ovvero il tipo di scala da cui deriva il suo contenuto melodico. Maggiore è rappresentato da 1 e minore da 0. |
| Speechiness - float | La "Speechiness" rileva la presenza di parole in una traccia. Più la registrazione è esclusivamente di tipo parlato (ad esempio, un talk show, un audiolibro, una poesia), più il valore dell'attributo si avvicina a 1,0. I valori superiori a 0,66 descrivono tracce che probabilmente sono composte interamente da parole parlate. I valori compresi tra 0,33 e 0,66 descrivono tracce che possono contenere sia musica che parlato, in sezioni o stratificati, compresi i casi di musica rap. I valori inferiori a 0,33 rappresentano molto probabilmente musica a basso contenuto o privi di testi |
| Acousticness - float | Una misura di confidenza da 0,0 a 1,0 che indica se il brano è acustico. 1,0 rappresenta un''elevata fiducia che il brano sia acustico. |
| Instrumentalness - float | E''una misura di probabilità che prevede se un brano non contiene voci. I suoni "ooh" e "aah" sono considerati strumentali in questo contesto. I brani rap o parlati sono chiaramente "vocali". Più il valore di strumentalità è vicino a 1, maggiore è la probabilità che il brano non contenga contenuti vocali. I valori superiori a 0,5 sono intesi come tracce strumentali, ma la fiducia è maggiore man mano che il valore si avvicina a 1,0. |
| Liveness - float | Rileva la presenza di un pubblico nella registrazione. Valori di "liveness" più alti rappresentano una maggiore probabilità che il brano sia stato eseguito dal vivo. Un valore superiore a 0,8 indica una forte probabilità che il brano sia dal vivo. |
| Valence - float | Una misura da 0,0 a 1,0 che descrive la positività musicale trasmessa da un brano. I brani con alta valenza suonano più positivi (ad esempio, felici, allegri, euforici), mentre i brani con bassa valenza suonano più negativi (ad esempio, tristi, depressi, arrabbiati). |
| Tempo - float | Il tempo complessivo stimato di un brano in battiti al minuto (BPM). Nella terminologia musicale, il tempo è la velocità o il ritmo di un determinato brano e deriva direttamente dalla durata media dei battiti. |
| Type - string | Il tipo di oggetto in risposta. In questo caso “audio-feature” |
| Id - string | L'ID Spotify del brano. |
| Uri - string | L'URI Spotify del brano. |
| Track\_href - string | Un link all'endpoint dell'API Web che fornisce tutti i dettagli della traccia. |
| Analysis\_url - string | Un URL per accedere all'analisi audio completa di questo brano. Per accedere a questi dati è necessario un token di accesso. |
| Duration\_ms - integer | La durata del brano in millisecondi. |
| Time\_signature - integer | Una stima dell'indicazione di tempo. L'indicazione di tempo (metro) è una convenzione di notazione per specificare quanti battiti ci sono in ogni battuta musicale (o misura). La firma temporale va da 3 a 7, indicando firme temporali da "3/4" a "7/4". |

### Get Several Artists

Restituisce informazioni dal catalogo Spotify per diversi artisti in base ai loro ID Spotify

**Request.Query**

|  |  |
| --- | --- |
| Ids - string | Un elenco separato da virgole degli ID Spotify degli artisti. Massimo: 50 ID. |

**Response.Body**

|  |  |
| --- | --- |
| Artists external\_urls - string | L'URL esterno noto di Spotify per questo artista. |
| Followers href - string | Questo valore sarà sempre impostato su null, poiché l'API Web non lo supporta al momento. |
| Followers total - integer | Il numero totale di follower |
| Genres – array | Un elenco dei generi a cui l'artista è associato. Se non ancora classificato, l'array è vuoto. |
| Href - string | Un link all'endpoint della Web API che fornisce tutti i dettagli dell'artista. |
| Id - string | L'ID Spotify dell'artista. |
| Images url - string | L'URL di origine dell'immagine dell'artista. |
| Images height - integer | L'altezza dell'immagine dell'artista in pixel. |
| Images width – integer | La larghezza dell'immagine dell'artista in pixel. |
| Name - string | Il nome dell''artista |
| ‍Popularity - integer | La popolarità dell'artista. Il valore sarà compreso tra 0 e 100, con 100 che rappresenta il valore più popolare. La popolarità dell'artista è calcolata in base alla popolarità di tutti i brani dell'artista. |
| ‍Type - string | Il tipo di oggetto in risposta. In questo caso “artist” |
| ‍Uri – string | L'URI di Spotify per l'artista. |

# 

# R Shiny: Libreria Per Costruire Web App Interattive

## Introduzione

R è un linguaggio di programmazione per l'analisi statistica e la visualizzazione grafica, supportato dall'R Core Team[[8]](#footnote-9) e dalla R Foundation for Statistical Computing[[9]](#footnote-10). Creato dagli statistici Ross Ihaka e Robert Gentleman, R è utilizzato da data miner, bioinformatici e statistici per l'analisi dei dati e lo sviluppo di software statistico.

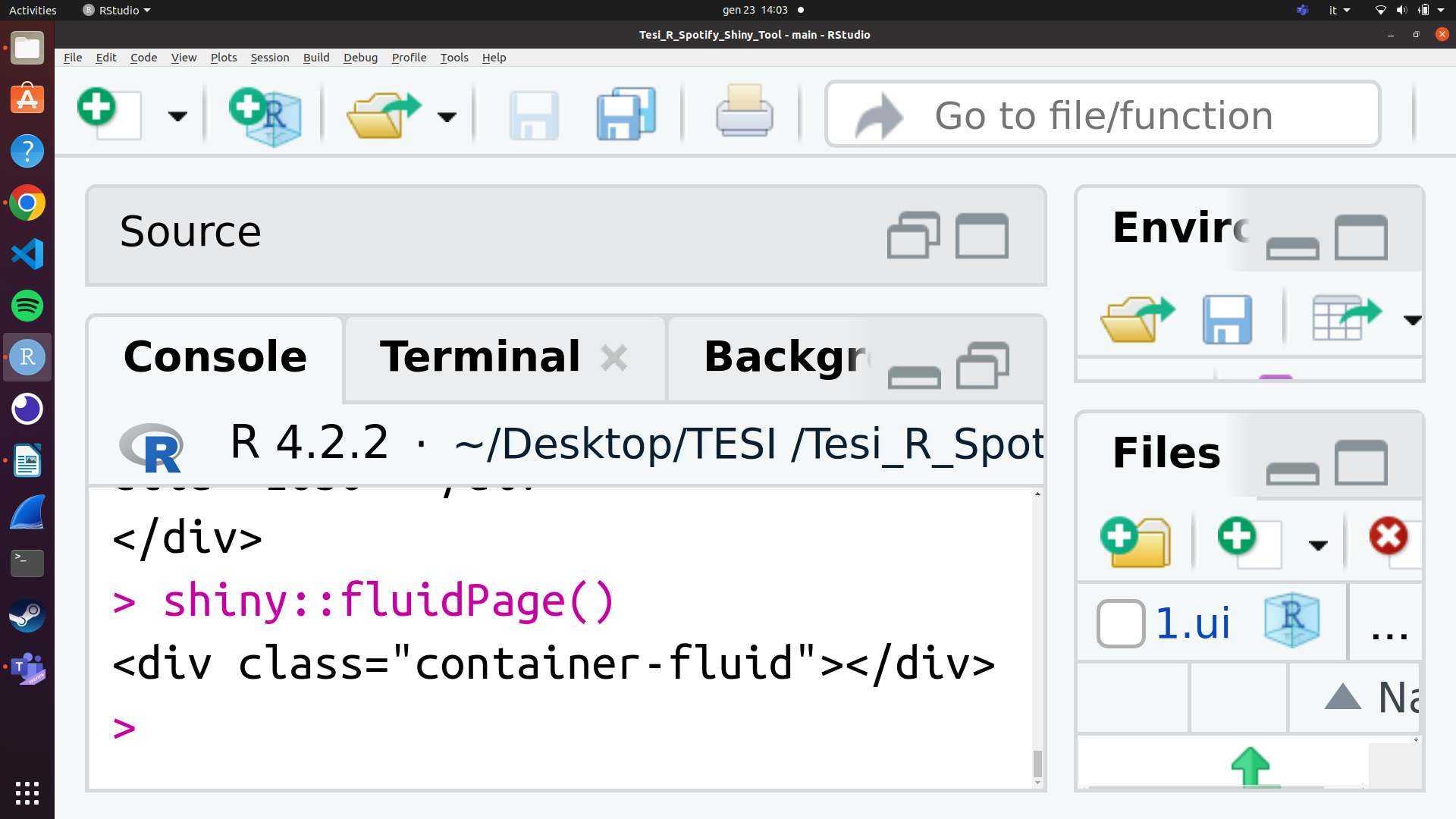
L'ambiente software ufficiale di R è un ambiente open-source di software libero all'interno del pacchetto GNU[[10]](#footnote-11), disponibile sotto la GNU General Public License[[11]](#footnote-12). R dispone di un'interfaccia a riga di comando. Sono inoltre disponibili diverse interfacce grafiche di terze parti, come RStudio[[12]](#footnote-13), un ambiente di sviluppo integrato.

Per aumentare le funzioni del linguaggio, R gli utenti hanno creato dei pacchetti. I pacchetti sono estensioni del linguaggio di programmazione R. I pacchetti R contengono codice, dati e documentazione in un formato di raccolta standardizzato che può essere installato dagli utenti di R, in genere tramite un repository software centralizzato come CRAN (Comprehensive R Archive Network)[[13]](#footnote-14).

I pacchetti R devono essere conformi ad una specifica rigida. Il *Writing R Extensions Manual[[14]](#footnote-15)* specifica una struttura di directory standard per il codice sorgente di R, i dati, la documentazione e i metadati dei pacchetti, che consente di installarli e caricarli utilizzando gli strumenti di gestione dei pacchetti integrati in R. I pacchetti distribuiti su CRAN devono soddisfare ulteriori standard.[[15]](#footnote-16)

Shiny è un pacchetto R scaricabile dalla repository CRAN, che consente di creare applicazioni web interattive. Le funzioni all’interno di questo pacchetto consentono di creare direttamente linguaggio HTML, CSS e Javascript.

Per esempio, se si esegue in console su R la funzione *fluidpage()* presente all’interno di Shiny, l’output sarà il codice HTML puro riportato di seguito:



Shiny fornisce un insieme di funzioni che generano l'HTML, il CSS e il JavaScript necessari per le attività di un’applicazione web e dunque permette di esporre tramite un browser web un progetto sviluppato in R.

Le applicazioni Shiny sono contenute in un singolo file script chiamato app.R. Lo script app.R vive in una cartella (per esempio, newdir/) e l'applicazione può essere eseguita con runApp("newdir").

app.R ha tre componenti:

* + - * + un oggetto interfaccia utente
        + una funzione server
        + una chiamata alla funzione shinyApp

L'oggetto interfaccia utente (ui) controlla il layout e l'aspetto dell'applicazione. La funzione server contiene le istruzioni di cui il computer ha bisogno per costruire l'applicazione. Infine, la funzione shinyApp crea gli oggetti dell'app Shiny a partire da una coppia esplicita UI/server.

Si noti inoltre che prima della versione 0.10.2, Shiny non supportava le app a file singolo e l'oggetto ui e la funzione server dovevano essere contenuti in script separati, chiamati rispettivamente ui.R e server.R. Questa funzionalità è ancora supportata in Shiny e per questo progetto verrà utilizzata la modalità a file separati per motivi di organizzazione del codice

Una caratteristica di Shiny e l’utilizzo di un paradigma di programmazione chiamato Programmazione Reattiva che verrà illustrato in dettaglio nelle sezioni successive.

Nel resto del capitolo verranno esaminati nel dettaglio le due componenti principali di un’applicazione Shiny (interfaccia utente e server) e verrà presentata la logica sottostante al paradigma della programmazione reattiva

## Interfaccia Utente - UI

## Server

## Programmazione Reattiva

# Analisi Dati

## Organizzazione e pulizia dati

La fase preliminare per poter condurre analisi statistiche rigorose su un dataset è l’organizzazione e la pulizia dei dati a disposizione. Per condurre queste operazioni preliminari e le successive analisi è stato utilizzato il software statistico R.

Dopo l’interrogazione dei tre endpoints della web API di Spotify descritti sopra, ho ottenuto tre differenti liste contenenti le informazioni relative alla mia musica salvata su Spotify.

Ottenute queste liste, l’obbiettivo è stato trasformare le tre liste in tre dataframe e successivamente unirli (merge) in un unico dataframe in cui ogni riga corrispondesse a un brano, e ogni colonna a una variabile relativa al singolo brano (i.e. id del brano, artista, una variabile per ogni caratteristica audio...etc)

La richiesta al primo endpoint (*Get User's Saved Tracks)* ha restituito una lista contenente la mia intera libreria aggiornata alla data della richiesta (6014 brani), con all’interno le informazioni presenti nel body della risposta di questo endpoint descritte nel capitolo precedente. Per poter ricondurre questa lista a un dataframe mi sono servito del pacchetto “data.table”[[16]](#footnote-17), disponibile su CRAN utilizzando la funzione rbindlist(). Questo mi ha restituito un dataframe di 6014 righe e 30 variabili, compresa l’id (chiave per il successivo merge), codice identificativo univoco per il singolo brano.

La richiesta al secondo endpoint (*Get Track's Audio Features*), ha ricevuto in input tutti gli id del dataframe ottenuto al punto precedente e ha restituito per ogni id tutte le informazioni contenute nel body della risposta di questo endpoint. La lista risultante è stata poi trasformata in dataframe sempre mediante la funzione rbindlist(). In questo modo ho ottenuto un dataframe di 6014 righe, corrispondenti al singolo brano identificato attraverso la variabile “id” e 17 ulteriori variabili, ognuna riferita a una caratteristica audio descritta nel dettaglio nel capitolo precedente nella sezione Response.Body dell’endpoint *Get Track's Audio Features.*

In questo modo ho ottenuto due dataframe distinti entrambi di 6014 righe e identificati da una colonna id contenente una stringa identificativa univoca del singolo brano. È stato dunque possibile eseguire un merge tra questi due dataset usando come chiave la colonna “id” ottenendo un dataframe unico di 6014 righe e 47 colonne.

Ai fini ultimi dell’analisi non erano necessarie tutte le colonne (variabili) di questo dataframe. Si è proceduto dunque alla selezione di un sottoinsieme delle colonne contenente solo le variabili rilevanti. Il dataframe così ottenuto conta 6014 righe e le seguenti 23 colonne:

* id
* danceability
* energy
* key
* loudness
* mode
* speechiness
* acousticness
* instrumentalness
* liveness
* valence
* tempo
* duration\_ms
* time\_signature
* added\_at
* track.artists
* track.explicit
* track.name
* track.popularity
* track.album.id
* track.album.name
* track.album.release\_date
* track.album.total\_tracks

In seguito, si è proceduto alla chiamata al terzo e ultimo endpoint, *Get Several Artists.* Quest'ultimo endpoint è stato interrogato con una chiamata GET per ottenere informazioni relative agli artisti e, nello specifico, al genere musicale composto da questi artisti, così da ottenere una proxy del genere musicale del brano associato all’artista. È infatti ad oggi non presente il genere musicale tra le informazioni del singolo brano, ma questa informazione è disponibile solo per artista.

La query di questa richiesta, come descritto nel capitolo precedente, ha ricevuto in input una lista di ids degli artisti. Per ottenere gli ids degli artisti ho isolato la colonna "track.artists" del dataframe descritto in precedenza e gli ho ulteriormente applicato la funzione unique() per non avere duplicati. In questo modo ho ottenuto un array di stringhe di 2237 ids di artisti univoci. Questo array è stato poi utilizzato nella query della richiesta all’endpoint *Get Several Artists.*

Questa richiesta ha ottenuto come risposta una lista, successivamente trasformata in un dataframe di 2237 righe corrispondenti ai singoli artisti e 12 colonne (variabili). Queste 12 variabili sono state poi ridotte a 5: "id"; "name"; "popularity"; "followers.total"; "main\_genre".

A questo punto dell’estrazione ho due dataframe: il primo con 6014 righe (brani) e 23 colonne descritte sopra; il secondo con 2237 (artisti) e 5 colonne.

Al fine di ottenere un unico dataframe contenente tutte le informazioni estratte e che avesse per riga i brani e per colonna le variabili riferite al brano ho eseguito un merge tra questi due dataframe.

La chiave per il primo è stata la variabile "track.artists". La chiave per il secondo dataframe è stata “id”.

Dopo una ricodifica dei nomi delle variabili per evitare problemi di ambiguità ho in questo modo ottenuto un unico dataframe di 6014 righe (brani) e le seguenti 33 variabili:

* track.id
* track.artists.id
* danceability
* energy
* key
* loudness
* mode
* speechiness
* acousticness
* instrumentalness
* liveness
* valence
* tempo
* duration\_ms
* time\_signature
* added\_at
* track.artists
* track.explicit
* track.name
* track.popularity
* track.album.id
* track.album.name
* track.album.release\_date
* track.album.total\_tracks
* track.artists.href
* track.artists.name
* track.artists.type
* track.artists.uri
* track.artists.external\_urls.spotify
* featuring
* popularity
* followers.total
* main\_genre

### Note sulla scelta delle variabili

Prima di procedere con la descrizione delle analisi statistiche condotte sul dataframe ottenuto è doveroso fare alcune considerazioni sulla scelta delle variabili oggetto di analisi.

Gli indici che descrivono le caratteristiche audio di un brano presenti nel body di risposta dell’endpoint *Get Track's Audio Features,* sono indici costruiti ed esposti direttamente da Spotify. Si intuisce che molte di queste variabili non sono quantità oggettive. Se da un lato la durata di un brano in millisecondi è una grandezza misurabile oggettivamente, questo non vale più per molte altre variabili prese in considerazione in questo studio. È doveroso sottolineare che non ho condotto alcuna analisi o studio preliminare per determinare quali siano le componenti e le proxy migliori per indicare tali fenomeni musicali. Ad esempio, sarebbe opportuno chiedersi cosa sia la “danceability” e quali siano le componenti che determinano questo fenomeno. Al fine di uno studio musicale approfondito sarebbe doveroso interrogarsi su quali siano le proxy migliori per descrivere le caratteristiche musicali di un brano. In questa tesi si è deciso di utilizzare le analisi già condotte da Spotify, affidandosi alle loro considerazioni e agli indici da loro proposti. La scelta è stata quella di confidare nella consolidata esperienza e negli studi approfonditi nel campo musicale di questa azienda.

Un’ulteriore nota sulla scelta delle variabili riguarda l’attribuzione del genere musicale al singolo brano. Come segnalato in precedenza, Spotify non riporta il genere musicale per il singolo brano. L’informazione a disposizione è un array di generi musicali relativo all’artista. Ad esempio, per l’artista Johann Sebastian Bach, per la variabile “genres”, Spotify restituisce il seguente array: [baroque, classical, early music, german baroque]. Questo array è ordinato per importanza di genere rispetto all’artista. Il primo elemento dell’array è il genere maggiormente suonato dall’artista, e i successivi a scalare in ordine decrescente.

Per poter selezionare il genere di un brano associato a un artista è stata adottata la seguente tecnica. È stato estratto il primo elemento dell’array per ogni artista ed è stato utilizzato come valore per una nuova variabile “main\_genre”; nel caso dell’esempio questo sarebbe stato *baroque.* Questo main\_genre è stato poi utilizzato come proxy del genere musicale del brano associato all’artista.

È doveroso segnalare che questa tecnica non è oggettiva nell’attribuzione del genere musicale al singolo brano e, come per le variabili precedentemente considerate, sarebbe opportuno fare analisi musicali ad hoc ed interrogarsi su quali siano le caratteristiche musicali che determinano l’appartenenza a uno specifico genere musicale. Per questo primo progetto di tesi si è deciso di operare come riportato sopra, ma sono noti i limiti di questa soluzione ed è possibile approfondire e modificare questa scelta in studi futuri.

## Analisi Univariata (Distribuzioni, Visualizzazione)

## Analisi Multivariata (Correlazioni….)

## Cluster Analysis (k-means; Archetypal Analysis)

## PCA (Principal Components Analysis)

# Output

## Dashboard Analitica

## Insights Derivanti all**''Analisi Dati**

# Conclusioni

# Bibliografia

* R Core Team (2022). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. <<https://www.R-project.org/>>. R version 4.2.2 Patched (2022-11-10 r83330)
* Thompson C, Antal D, Parry J, Phipps D, Wolff T (2021). \_spotifyr: R Wrapper for the 'Spotify''Web API\_. R package version 2.2.3, <<https://CRAN.R-project.org/package=spotifyr>>.
* Dowle M, Srinivasan A (2022). \_data.table: Extension of `data.frame`\_. R package version 1.14.6, <<https://CRAN.R-project.org/package=data.table>>.
* H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.
* Kassambara A (2022). \_ggpubr: 'ggplot2''Based Publication Ready Plots\_. R package version 0.5.0, <<https://CRAN.R-project.org/package=ggpubr>>.
* Kassambara A, Mundt F (2020). \_factoextra: Extract and Visualize the Results of Multivariate Data Analyses\_. R package version 1.0.7, <<https://CRAN.R-project.org/package=factoextra>>.
* Kunst J (2022). \_highcharter: A Wrapper for the 'Highcharts''Library\_. R package version 0.9.4, <<https://CRAN.R-project.org/package=highcharter>>.
* Taiyun Wei and Viliam Simko (2021). R package 'corrplot': Visualization of a Correlation Matrix (Version 0.92). <https://github.com/taiyun/corrplot>
* Wickham H, Girlich M (2022). \_tidyr: Tidy Messy Data\_. R package version 1.2.1, <<https://CRAN.R-project.org/package=tidyr>>.
* Firke S (2021). \_janitor: Simple Tools for Examining and Cleaning Dirty Data\_. R package version 2.1.0, <<https://CRAN.R-project.org/package=janitor>>.
* Manuel J. A. Eugster and Friedrich Leisch. From Spider-Man to Hero -- Archetypal Analysis in R. Journal of Statistical Software, 30(8), 1-23, 2009. <http://www.jstatsoft.org/v30/i08/>
* Chacon, S. (2014). *Pro Git* (2nd ed.). Apress.
* Spotify AB, Spotify, 2022, <<https://developer.spotify.com/>*>*
* Ross Ihaka and Robert Gentleman, 1993, <https://www.r-project.org/>

# Appendice: Controllo Versione – GIT

## Definizione

## Utilizzo

## GIT

## GitHub

Link repository GitHub

1. JSON è un formato per lo scambio di dati. Si basa su un sottoinsieme del linguaggio di programmazione JavaScript. https://www.json.org/json-en.html [↑](#footnote-ref-2)
2. XML è un linguaggio di markup e un formato di file per memorizzare, trasmettere e ricostruire dati arbitrari. Definisce un insieme di regole per la codifica dei documenti in un formato leggibile dall'uomo e dalla macchina https://www.w3.org/TR/xml/ [↑](#footnote-ref-3)
3. Un URI (Uniform Resource Identifier) è una sequenza unica di caratteri che identifica una risorsa logica o fisica utilizzata dalle tecnologie web. Gli URI possono essere utilizzati per identificare qualsiasi cosa, compresi oggetti del mondo reale, come persone e luoghi, concetti o risorse informative come pagine web e libri. [↑](#footnote-ref-4)
4. La suite di protocolli Internet, comunemente nota come TCP/IP, è un quadro di riferimento per organizzare l'insieme dei protocolli di comunicazione utilizzati in Internet I protocolli fondamentali della suite sono il Transmission Control Protocol (TCP), lo User Datagram Protocol (UDP) e l'Internet Protocol (IP). Nello sviluppo di questo modello di rete, le prime versioni erano conosciute come il modello del Dipartimento della Difesa (DoD) perché la ricerca e lo sviluppo erano finanziati dal Dipartimento della Difesa degli Stati Uniti attraverso il DARPA. [↑](#footnote-ref-5)
5. <https://developer.spotify.com/> [↑](#footnote-ref-6)
6. R Core Team (2022) [↑](#footnote-ref-7)
7. Thompson C, Antal D, Parry J, Phipps D, Wolff T (2021) [↑](#footnote-ref-8)
8. <https://www.r-project.org/index.html> [↑](#footnote-ref-9)
9. https://www.r-project.org/foundation/ [↑](#footnote-ref-10)
10. https://www.gnu.org/ [↑](#footnote-ref-11)
11. https://www.gnu.org/licenses/gpl-3.0.html [↑](#footnote-ref-12)
12. https://posit.co/ [↑](#footnote-ref-13)
13. https://cran.r-project.org/ [↑](#footnote-ref-14)
14. https://cran.r-project.org/doc/manuals/R-exts.html [↑](#footnote-ref-15)
15. https://cran.r-project.org/web/packages/policies.html [↑](#footnote-ref-16)
16. Dowle M, Srinivasan A (2022) [↑](#footnote-ref-17)