ANTEPROYECTO PI

Cancelo-Fernández-Senande
https://github.com/GEI-PI-614G010492223/aplicacion_django-cancelo_fernandez_senande

· Francisco Cancelo Fernández (f.cancelo@udc.es)
· Miguel Fernández Fernández (miguel.ffernandez@udc.es)
· Fabio Senande Torrado (f.torrado@udc.es)

1. Resumen

El objetivo es desarrollar una aplicación web en el framework Django y con Python como lenguaje de programación. Para ello usaremos los servicios de las APIs de Spotify, TicketMaster y Directions (Google) con el fin de que la aplicación cree una ruta óptima para asistir a conciertos de los artistas especificados mediante alguna de las funcionalidades proporcionadas. Se extraerán los artistas de la playlist seleccionada (Spotify), se buscarán sus próximos eventos (TicketMaster) y se decidirá la mejor ruta (Directions), decidiendo los conciertos a partir de variables como el presupuesto, la ubicación, rango de fechas, distancia…

2. Funcionalidades

- Login de usuario: Autorización a la app para usar sus datos personales de Spotify. El usuario es redirigido a la página de inicio de sesión de Spotify, donde introduce sus credenciales. Una vez se loguea, el usuario autorizará a nuestra app el permiso de lectura de datos musicales (artistas más escuchados…) propios del usuario. En caso contrario (el usuario no autoriza este uso) sus funcionalidades dentro de la app se verán limitadas a datos públicos.

- Buscar en top artistas: (*Solo para usuarios logueados) Selección de artistas según el top más escuchado personal del usuario. El usuario escoge esta opción y elige el rango de tiempo en el que quiere que se obtengan los datos: 1 mes, 6 meses o histórico (desde la creación de la cuenta). Se obtienen los artistas más escuchados en las fechas especificadas y se muestran en la lista. Sobre estos se aplican los filtros especificados por el usuario en la funcionalidad Filtros de búsqueda.

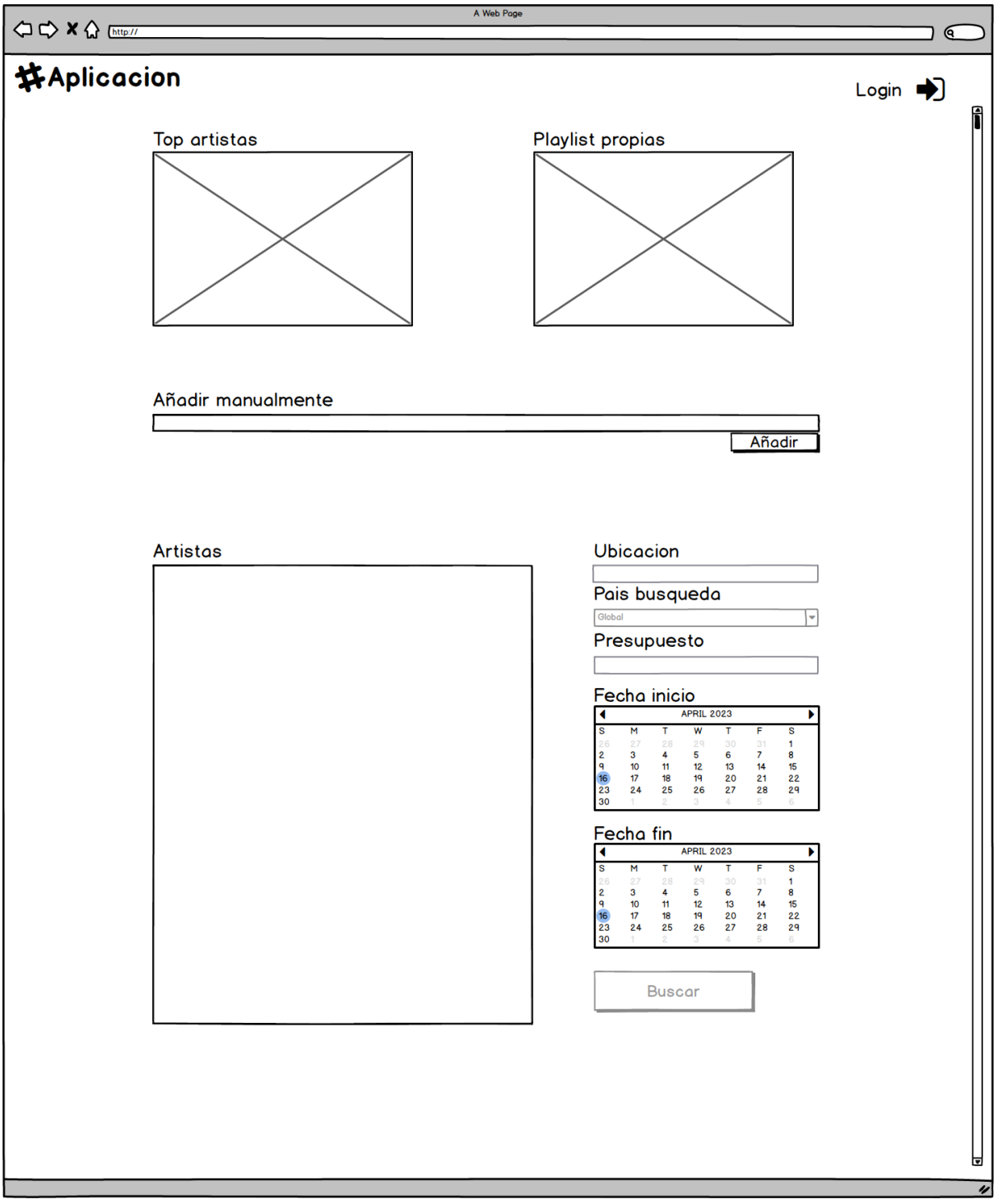
- Buscar en playlist propia: (*Solo para usuarios logueados) Selección de artistas según playlist específica. El usuario escoge esta opción y se abre un desplegable con todas las playlists propias y seguidas del usuario. El usuario elige alguna de estas playlists, se obtienen los artistas presentes en ella y se muestran en la lista. Sobre estos se aplican los filtros especificados por el usuario en la funcionalidad Filtros de búsqueda.

- Buscar artistas: Selección de artistas como búsqueda libre. El usuario escribe en el campo de buscar artista e introduce manualmente los nombres de los artistas que le interesan y se muestran en la lista.. Sobre estos se aplican los filtros especificados por el usuario en la funcionalidad Filtros de búsqueda.

- Iniciar búsqueda: Recogida de preferencias, se envía la lista de artistas y la información de los campos proporcionados para aplicar estos filtros a la BBDD. Mediante un formulario, el usuario introduce sus preferencias en los campos proporcionados. Estos campos son: "Presupuesto" (el límite para todos los conciertos), "Fecha de inicio", "Fecha de fin" (rango de fechas en las que el usuario puede o quiere ir), "País de búsqueda" (país en el que buscar conciertos) y "Ubicación" (lugar de residencia del usuario). Y se procesa para obtener la ruta óptima final y mostrarla al usuario.

- Eliminar artista lista: Se elimina el artista seleccionado de la lista (para no buscar conciertos de ese artista) mediante un botón "X" al lado de cada artista.

3. Mockups app



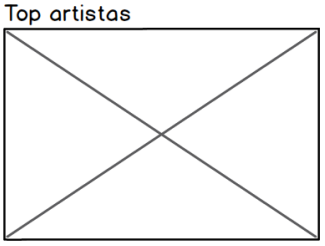
A Web Page

http://

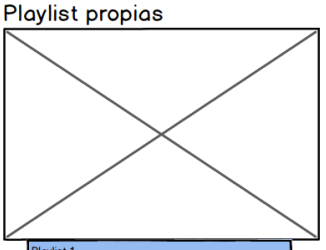
Aplicacion

Login

Top artistas



Playlist propias



- Playlist 1
- Playlist 2
- Playlist 3

Añadir manualmente

Añadir

Artistas

- Artista 1
- Artista 2
- Artista 3
- Artista 4
- Artista 5
- Artista 6
- Artista 7
- Artista 8
- Artista 9
- Artista 10

Ubicacion

A Coruña

Pais busqueda

España

Presupuesto

300

Fecha inicio

APRIL 2023

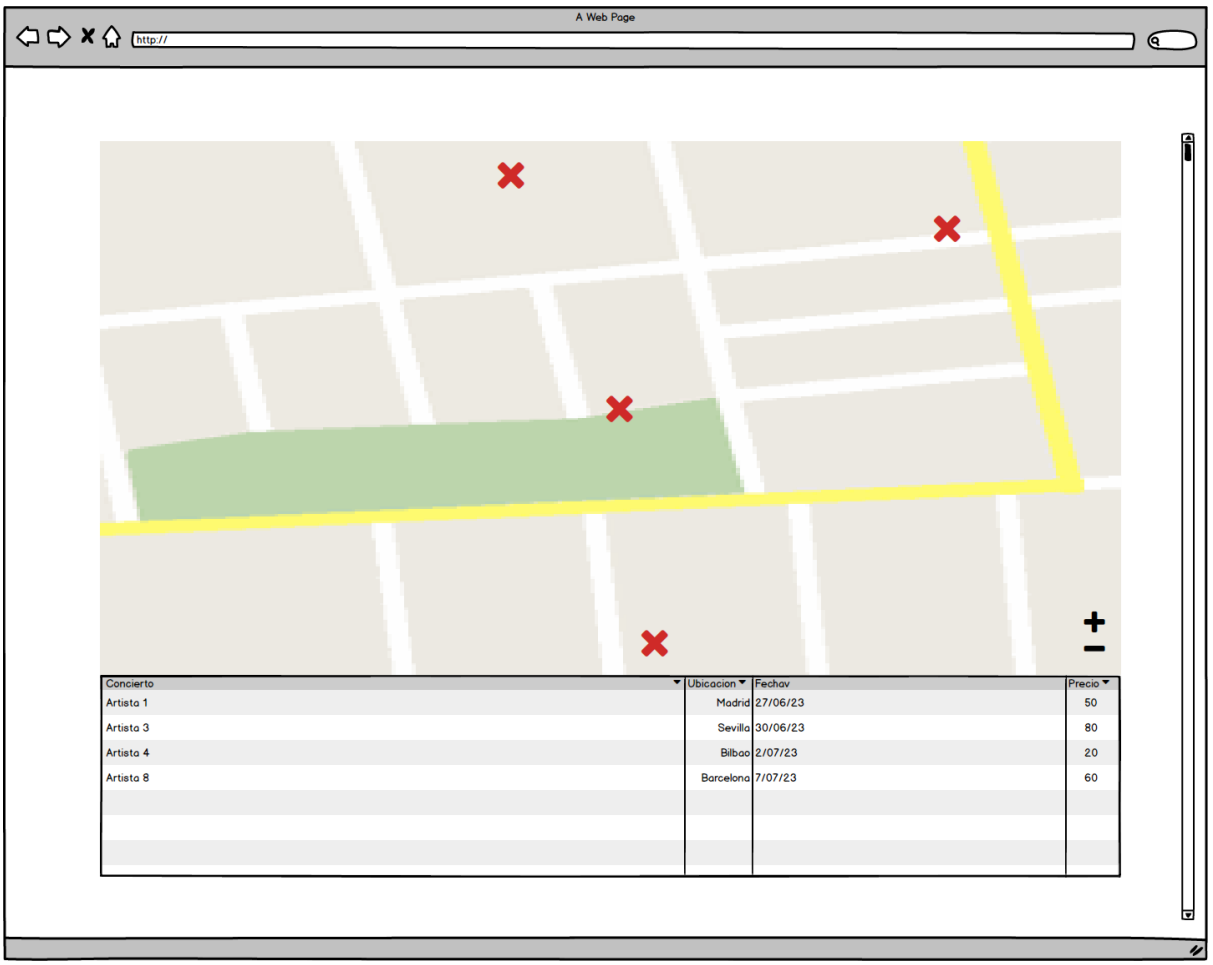
S	M	T	W	T	F	S
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Fecha fin

APRIL 2023

S	M	T	W	T	F	S
26	27	28	29	30	31	1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30	1	2	3	4	5	6

Buscar



4. Flujo de datos

1. **FLUJO DE AUTORIZACIÓN** (*Opcional en el caso de búsqueda manual): El usuario introduce sus credenciales de Spotify, autoriza a la aplicación y redirige al usuario a la página principal de nuestra app. La URL de vuelta trae un"code". Este "code" (GET) se intercambia por un "access_token" (POST) temporal con el que se realiza la búsqueda deseada.

2. **FLUJOS DE USO:**
    a. Petición: **Pulsa en "Buscar en top artistas"**
    Acción: Se llama a la API de Spotify (con el "access_token") solicitando el top de artistas de este usuario.
    Resultado: Se muestran en la lista los nombres de los artistas.

    b. Petición: **Pulsa en "Buscar en playlist propia"**
    Acción: Se llama a la API de Spotify solicitando las playlists propias del usuario (con el "access_token").
    Resultado: Se abre un desplegable con los nombres de las playlists.

c. Petición: **Elige en su playlist deseada.**
   Acción: Se llama a la API de Spotify solicitando los nombres de todos los artistas presentes en las canciones de esa playlist.
   Resultado: Se muestra en la lista los nombres de los artistas.

d. Petición: **Escribe en "Buscar artistas" el nombre de un artista.**
   Acción: Se busca el nombre de este artista en Spotify API.
   Resultado: Se muestra en la lista los nombres de los artistas que va introduciendo.

e. Petición: **Pulsa el botón “X” al lado del artista en la lista.**
   Acción: Se elimina de la lista el artista.
   Resultado: Se muestra la nueva lista actualizada.

f. Petición: **Escribe el presupuesto en el campo "Presupuesto" del formulario**.
   Acción: Se guarda el valor del campo como variable.
   Resultado: Se muestra el valor ingresado junto con los otros filtros.

g. Petición: **Selecciona las fechas de inicio y fin en el campo "Fechas" del formulario**.
   Acción: Se guarda el valor del campo como variable.
   Resultado: Se muestra el valor ingresado junto con los otros filtros.

h. Petición: **Escribe el país en el campo "País" del formulario.**
   Acción: Se guarda el valor del campo como variable.
   Resultado: Se muestra el valor ingresado junto con los otros filtros.

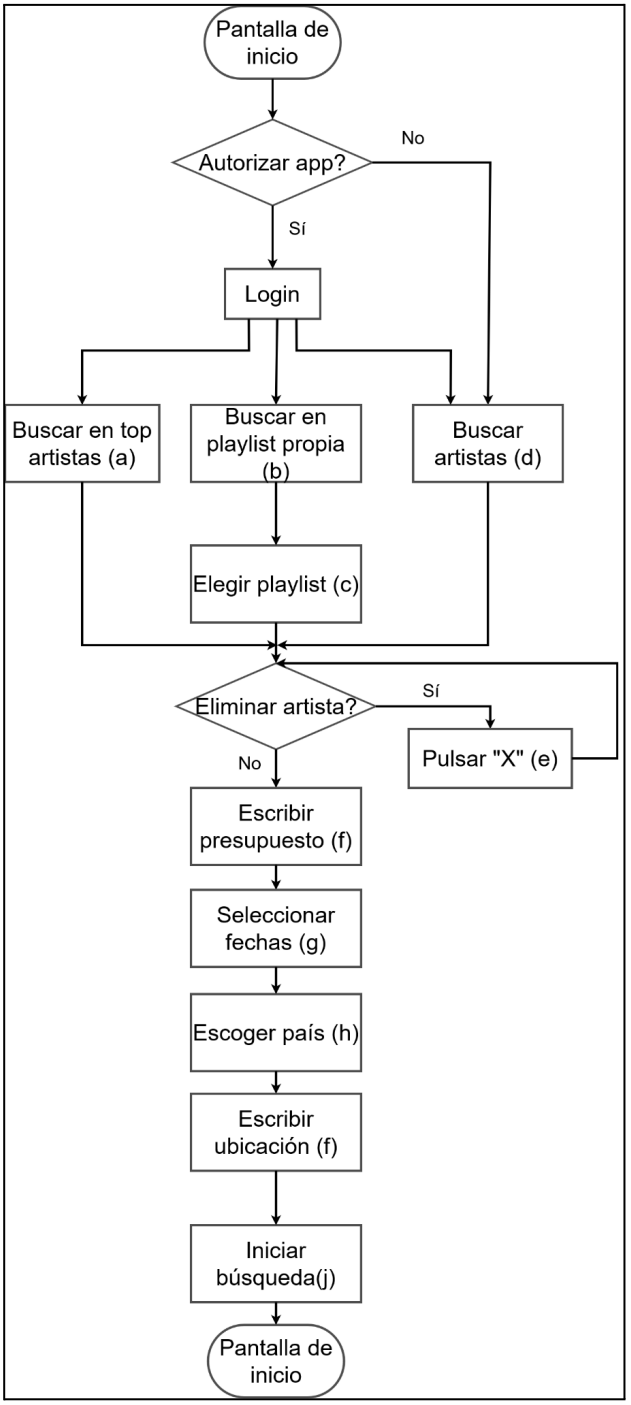
i. Petición: **Escribe su lugar de residencia en “Ubicación” del formulario.**
   Acción: Se guarda el valor del campo como variable.
   Resultado: Se muestra el valor ingresado junto con los otros filtros.

j. Petición: **Pulsa en "Iniciar búsqueda"**
   Acción:
   i. 1º Se llama a la API de Ticketmaster con los nombres de los artistas como entrada y devuelve los IDs de los artistas (dentro de Ticketmaster, si están dentro de su BBDD).
   ii. 2º Se llama a la API de Ticketmaster con estos IDs y se obtienen los distintos conciertos, de los cuales guardamos su nombre de concierto, nombre de artista, fecha, duración, dirección y precio.
   iii. 3º Se aplican los filtros de país y fecha para obtener aquellos que estén dentro del país y rango de fechas especificados.
   iv. 4º Se procesan los datos para obtener las distintas combinaciones de itinerarios posibles y viables filtrando (limitando) por aquellos que se mantienen dentro del presupuesto, y con una diferencia de duración (duración del evento + viaje) que tienen la mayor cantidad de conciertos posibles.

v. 5º Se llama a la API de Google Directions con las distintas combinaciones como entrada y devuelve datos como la distancia y duración como salida.

vi. 6º Se procesan los datos devueltos por Directions y se escoge el más óptimo teniendo en cuenta la distancia, el tiempo y el presupuesto.

Resultado: Se presenta un mapa con el recorrido de la ruta más óptima obtenida. Además se muestra el gasto final, nombre de los eventos, nombre de los artistas y fechas de los conciertos que se celebran en los distintos nodos del mapa.



5. APIs empleadas

- TicketMaster (https://developer.ticketmaster.com/products-and-docs/apis/discovery-api/v2/)
    - Uso de la API: Mediante esta API obtenemos los conciertos de los artistas deseados, junto con la información de fecha, ubicación, precios…
- Spotify (https://developer.spotify.com/documentation/web-api)
    - Uso de la API: Mediante esta API permitimos al usuario buscar los artistas presentes en su top personal o incluso en playlists propias y seguidas.
- Directions de Google Maps (https://developers.google.com/maps/documentation/directions?hl=es-419)
    - Uso de la API: Mediante esta API obtendremos las rutas, distancias y duración para ver todos los conciertos deseados de forma óptima.

6. Uso de Pandas

- Empleando la función *drop_duplicated()* de Pandas se eliminarán los conciertos y artistas duplicados que devolvería la API de TicketMaster.
- Empleamos la función *loc()* para filtrar y obtener aquellos conciertos cuya fecha esté dentro del rango de fechas especificado por el usuario y también utilizamos esta misma función para descartar, inicialmente, todos los conciertos cuyo precio sobrepasa el presupuesto indicado por el usuario.
- Pandas ordenará los eventos obtenidos de TicketMaster respecto a la fecha de inicio con *sort_values*.
- Seleccionaremos, mediante Pandas, subgrupos, usando *groupby()* de conciertos dividiendo según un rango de fechas prefijado o calculado (ej: 5 días, 1/5 del rango total de fechas…).
- Sobre estos subgrupos se escogerá el concierto más óptimo combinando: el precio del concierto (€) + la distancia al punto de origen (€ tras aplicar, *apply(),* un precio prefijado, por ejemplo el de la gasolina, a cada kilómetro). De esta forma tendríamos un único factor de decisión, el precio en €.
- El resto del subgrupo se descartan, y el artista cuyo concierto fue escogido se elimina de la base de datos (junto con sus conciertos), ya que el usuario no quiere volver a ver otro concierto de un mismo artista. Se usará *drop()*.
- Ahora el origen pasa a ser el nuevo concierto. Y sobre el siguiente subgrupo (con 1 día añadido por el propio viaje) se aplica el mismo proceso hasta que el precio total (¡el total de las entradas, no de los viajes!), calculado con *cumsum()* excede el presupuesto.
- Así obtendremos la lista final de conciertos. (Este método de selección del camino más óptimo puede sufrir cambios dependiendo de la eficiencia general y el retraso que puedan provocar las llamadas a la API de Google Directions)

7. Funcionalidades de la 1ª iteración

En la primera iteración tenemos planeado tener la funcionalidad más básica de búsqueda de artistas. Es decir, aquella que no necesita la API de Spotify ni el login del usuario ya que solo buscaríamos que se añadieran los nombres de los artistas deseados a la lista y se buscase la ruta más óptima entre los conciertos de esos artistas. Por lo tanto en esta primera iteración trabajaríamos con las APIs de Directions y Ticketmaster, y realizaríamos el flujo de datos principal: el que busca las combinaciones y calcula la ruta óptima tras pasar por los filtros deseados.

La presentación de los resultados y de la web en general puede variar mucho de lo descrito en los mockups ya que vamos a centrarnos principalmente en el objetivo principal. Este se define principalmente por estas 4 funcionalidades:

- Buscar artistas: Selección de artistas como búsqueda libre. El usuario escribe en el campo de buscar artista e introduce manualmente los nombres de los artistas que le interesan y se muestran en la lista. Sobre estos se aplican los filtros especificados por el usuario en la funcionalidad Filtros de búsqueda.

- Eliminar artista lista: Se elimina el artista seleccionado de la lista (para no buscar conciertos de ese artista) mediante un botón “X” al lado de cada artista.

- Iniciar búsqueda: Se envía la lista de artistas y la información de los campos proporcionados para aplicar estos filtros a la BBDD. Y se procesa para obtener la ruta óptima final y mostrarla al usuario.

8. Uso de librería *third-party*

Planeamos usar la librería de Scrapy para añadir una funcionalidad adicional a nuestra página web. Esta funcionalidad adicional puede variar dependiendo de un factor, el tiempo de carga y duración de la búsqueda. En esta etapa tan temprana no tenemos la certeza de si la cantidad (limitada o no) de artistas a buscar o si la cantidad de llamadas API retrasarán el proceso de búsqueda y optimización de caminos.

Nuestro plan es presentar noticias actuales (en tiempo real) durante el proceso de carga o/y, dependiendo de la duración de este, en la pantalla inicial (para lo cual el uso de peticiones AJAX (JavaScript) será necesario). Para ello utilizaremos Scrapy con el objetivo de buscar noticias en ciertos noticieros musicales como por ejemplo:
https://www.music-news.com, https://www.billboard.com/c/music/music-news …