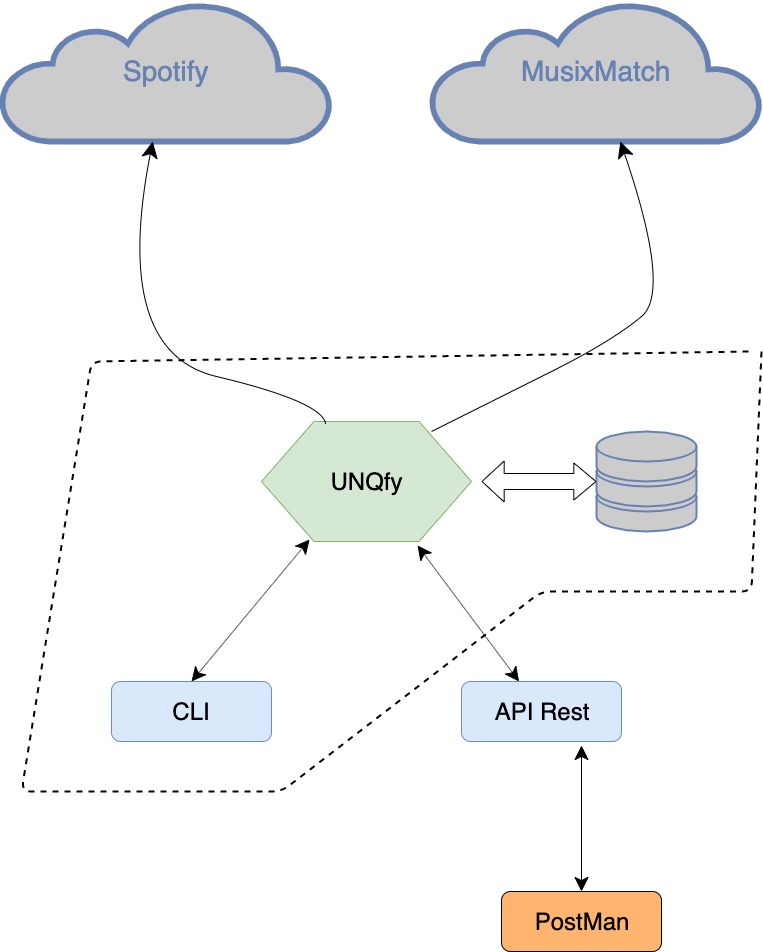
Visado 2 - REST / Asincrónismo

# Introducción

Basándose en su implementación de UNQfy (lo que está encerrado en líneas punteadas en gráfico) del visado anterior ahora ud. deberá agregar la siguiente funcionalidad:

1- Enriquecer su modelo con datos provenientes de Spotify y MusixMatch

2- Desarrollar una API rest para interactuar con UNQfy. Para testear su servicios RESTful usará test cases provistos



# Fase 1 - Enriquecer el modelo

## Todos los Albumes de un artista

En el modelo UNQfy debe ser posible preguntar por todos los albumes de un artista dado.Se debe reportar el título de los álbumes.

getAlbumsForArtist(artistName).

## Spotify

Implemente un comando en UNQfy llamado populateAlbumsForArtist( artistName) . Cuando UNQfy recibe este comando debe consultar los álbumes de dicho artista en Spotify, en base a los datos recibidos debe instanciar los álbumes correspondientes y asociarlos al artista, de manera tal que en la siguiente invocación a getAlbumsForArtist(artistName) se incluyan todos los álbumes recibidos de Spotify.

Debe implementar esta parte del enunciado utilizando requests (implementados usando callbacks o promesas).

Para conectarse a Spotify es necesario pasar por un proceso de autenticación basado en OAuth. Se provee un snippet de código para facilitar esta tarea.

La API de Spotify se encuentra documentada en:

<https://beta.developer.spotify.com/documentation/web-api/reference/>

En particular usted deberá utilizar el siguiente end point:

<https://beta.developer.spotify.com/documentation/web-api/reference/artists/get-artists-albums/>

G**enerar Credenciales para la API de Spotify:**

* Copiarse el archivo 'generateSpotifyCredentials.js' (provisto por la cátedra) en el mismo directorio donde se encuentra el código de su unqfy.
* Ejecutar: npm install --save express request
* Ejecutar: node generateSpotifyCredentials.js

Una vez ejecutado deberá:

* Hacer click en link impreso en la terminal
* Ese link lo llevará a loguearse en spotify (deberá crear una cuenta si no la tiene)
* Una vez logueado, volver a la terminal. Se habrá generado un archivo spotifyCreds.json con el access\_token

Para poder enviar un request autorizado a spotify deberá enviar el string 'BEARER ' + access\_token en el Header 'Authorization' en cada request. Utilizando request-promise:

const rp = require('request-promise');

const options = {

url: 'ALGUN\_ENDPOINT\_DE\_SPOTIFY'

headers: { Authorization: 'Bearer ' + 'ACCESS\_TOKEN' },

json: true,

};

rp.get(options).then((response) => //hacer algo con response);

## MusixMatch

MusixMatch es una aplicación (con una API Rest) que provee las letras (lyrics) de muchas canciones.

UNQfy ahora debe soportar pedirle la letra a un determinado track/tema.

Cuando un tema recibe el mensaje *getLyrics()* debe retornar el string con la letra de esa canción. Obviamente, la primera vez que se le envía ese mensaje a un track no tiene el lyric disponible, entonces debe ir a buscarlo a MusixMatch. Las siguientes invocaciones a getLyrics no deben volver a hacer el request, sino retornar el valor obtenido previamente.

MusixMatch utiliza *apikey* para validar la conexión. Utilice los pasos dados en la práctica 6 para obtener su API key.

La API de MusixMatch se encuentra documentada en este sitio.

<https://developer.musixmatch.com/documentation>

En particular, para resolver este ejercicio, ud utilizará el siguiente endpoint:

<https://developer.musixmatch.com/documentation/api-reference/track-lyrics-get>

Para obtener un track necesitará obtener su Id primero. Podrá obtener un listado de tracks que matchean con un string utilizando este Endpoint <https://developer.musixmatch.com/documentation/api-reference/track-search>

Nota: Tanto para obtener los albumes de un artista en Spotify, como para obtener la letra de un tema en MusixMatch, es necesario tener el *id*  correspondiente.

Para ello las búsquedas en ambos servicios retornan listas de objetos con sus IDs.

Por ej. si se quiere obtener los álbumes del artista Charly García, es necesario primero hacer una búsqueda de artista por el string "Charly García" como resultado obtendrá una serie de objetos con sus IDs (y otros campos) como:

* Charly García (4923943242)
* Charly García y Sui Generis (32454534532)
* Charly M. García (423094832048)

De todos esos seleccionaremos de manera automática siempre el primer artista (o tema según corresponda).

# Fase 2 - Desarrollo de la API Rest para UNQfy.

La API Rest de UNQfy debe permitir agregar, buscar y eliminar los objetos core del sistema: artistas, albums, tracks y playlists. **Para este visado, vamos a implementar operaciones CRUD de Artistas y álbumes.**

A continuación se describe la API. Debe respetar la definición para que los tests implementados con postman puedan ser utilizados.

## Artistas

|  |  |  |  |
| --- | --- | --- | --- |
| **Acción** | **Endpoint** | **Argumentos en Request** | **Respuesta** |
| Agregar un artista | POST /api/artists | **Body:**  {  'name': 'Guns n Roses' , 'country': 'USA'  } | Status Code: 201  {  "id": <artistId>  "name": 'Guns n' Roses"  "country":"USA",  "albums": []  } |
| Obtener un artista | GET /api/artists/<id> |  | Status Code: 200  {  "id": <artistId>  "name": 'Guns n' Roses"  "country":"USA",  "albums":[<album1>, <album2> ]  } |
| Actualizar un artista | PATCH /api/artists/<id> | **Body:**  {  'name': 'nuevo nombre' , 'country': 'ARG'  } | Status Code: 200  {  "id": <artistId>  "name": 'nuevo nombre"  "country":"ARG",  "albums":[<album1>, <album2> ]  } |
| Borrar un Artista | DELETE /api/artists/<id> |  | Status Code: 204 |
| Buscar artistas | GET /api/artists | **Query params**: name=<artistName> | Status Code: 200  [<artista1>, <artista2> ] |

## Albums

|  |  |  |  |
| --- | --- | --- | --- |
| **Acción** | **Endpoint** | **Argumentos en Request** | **Respuesta** |
| Agregar un album a un artista | POST /api/albums | **Body:**  {  artistId: artistId  'name': 'Appetite For Destruction' ,  year: 1987  } | Status Code: 201  {  "id": <albumId>  "name": 'Appetite For Destruction",  "year": 1987,  "tracks": []  } |
| Obtener un album | GET /api/albums/<id> |  | Status Code: 200  {  "id": <albumId>  "name": 'Appetite For Destruction",  "year": 1987,  "tracks": [],  } |
| Actualizar el año de un album | PATCH /api/albums/<id> | **Body:**  {  "year": 2020  } | Status Code: 200  {  "id": <albumId>  "name": 'Appetite For Destruction",  "year": 2020,  "tracks": []  } |
| Borrar un Album | DELETE /api/albums/<id> |  | Status Code: 204 |
| Buscar albumes | GET /api/albums | **Query**: name="<albumName>" | Status Code: 200  [<album1>, <album2> ] |

NOTA:

* Las búsquedas (de álbumes y artistas) **NO** deben ser sensibles a mayúsculas. Es decir, si existe el album con nombre “ALBUM 1” y se busca el string “aLbu” debe retornar el álbum “ALBUM 1”.
* Cuando se obtienen los artistas. El campo álbumes del artista debe tener una lista de objetos álbumes como los retornados en el Endpoint GET /albums/<albumId>
* Las búsquedas de artistas/álbumes por nombre, pueden no recibir el parámetro nombre y deberán retornar todos los álbumes/artistas.
* Para obtener las letras de una canción:
  + Deberá utilizar el método getLyrics implementado en la parte 1.
  + Asuma que la canción ya está creada en su UNQfy.

## Tracks

|  |  |  |  |
| --- | --- | --- | --- |
| **Acción** | **Endpoint** | **Argumentos en Request** | **Respuesta** |
| Buscar letra para una canción | GET /api/tracks/<trackId>/lyrics |  | Status Code: 200  {  Name: <trackName>,  lyrics: <lyricsDeMusixMatch>  } |

## Errores

Los código de error también forman parte de la API. Su implementación debe usar los códigos de error de la siguiente manera:

|  |  |  |
| --- | --- | --- |
| **Acción** | **Status Code Respuesta** | **Body Respuesta** |
| Se intenta agregar un artista/álbum duplicado | 409 | {  status: 409,  errorCode: "RESOURCE\_ALREADY\_EXISTS"  } |
| Se intenta agregar un álbum a un artista inexistente | 404 | {  status: 404,  errorCode: "RELATED\_RESOURCE\_NOT\_FOUND"  } |
| URL invalida/inexistente | 404 | {  status: 404,  errorCode: "RESOURCE\_NOT\_FOUND"  } |
| Borrar/Obtener/Actualizar un Artista/Album inexistente | 404 | {  status: 404,  errorCode: "RESOURCE\_NOT\_FOUND"  } |
| Se envía un Json invalido en el Body | 400 | {  status: 400,  errorCode: "BAD\_REQUEST"  } |
| Falta un parámetro en el JSON al agregar/modificar un artista/Álbum | 400 | {  status: 400,  errorCode: "BAD\_REQUEST"  } |
| Fallo inesperado | 500 | {  status: 500,  errorCode: "INTERNAL\_SERVER\_ERROR"  } |
| No existe la canción buscada en /api/tracks/<trackID>/lyrics | 404 | {  status: 404,  errorCode: "RESOURCE\_NOT\_FOUND"  } |

Notas/Aclaraciones:

* Recuerde persistir el objeto unqfy en las operaciones que modifican el mismo.
* Tenga en cuenta que para retornar un objeto de UNQfy como un JSON, probablemente quiera implementar el método toJSON en el objeto a devolver.

## Playlists

Deberán agregar a la API soporte para operaciones CRUD sobre las playlists:

|  |  |  |  |
| --- | --- | --- | --- |
| **Acción** | **Endpoint** | **Argumentos en Request** | **Respuesta** |
| Crea una playlist  Con máxima duración y tracks pertenecientes a alguno de los géneros | POST /api/playlists | **Body:**  {  'name': 'Nombre de la playlist',  'maxDuration': 3000,  'genres': ['genre1', 'genre2']  } | Status Code: 201  {  "id": <playlistId>  "name": 'nombre de la playlist",  "duration": 2700,  "tracks": [<track1>, <track2>]  } |
| Crear una playlist a partir de una lista de IDs de tracks | POST /api/playlists | **Body:**  {  'name': 'Nombre de la playlist',  'tracks': [<trackId1>, <trackId2>]  } | Status Code: 201  {  "id": <playlistId>  "name": 'nombre de la playlist",  "duration": 1200,  "tracks": [<track1>, <track2>]  } |
| Obtener una playlist | GET /api/playlists/<id> |  | Status Code: 200  {  "id": <playlistId>  "name": 'nombre de la playlist",  "duration": 2700,  "tracks": [<track1>, <track2>],  } |
| Borrar una Playlist | DELETE /api/playlists/<id> |  | Status Code: 204 |
| Buscar playlists | GET /api/playlists | **Query**: name="<playlistName>"&  durationLT=300  &durationGT=100 | Status Code: 200  [<playlist1>, <playlist2> ] |

**Nota:** para la búsqueda de playlists, debe de enviarse al menos un parámetro de los tres listados. En caso de mandar más de uno, se debe aplicar una operación AND entre todos los filtros.

* El parámetro durationLT filtra por playlists con duraciones menores a las especificadas
* El parámetro durationGT filtra por playlists con duraciones mayores a las especificadas

## Errores

Además de los errores ya manejados , deberá agregar:

|  |  |  |
| --- | --- | --- |
| **Acción** | **Status Code Respuesta** | **Body Respuesta** |
| Se intenta agregar un track inexistente a una playlist | 404 | {  status: 404,  errorCode: "RELATED\_RESOURCE\_NOT\_FOUND"  } |
| Borrar/Obtener/ una playlist inexistente | 404 | {  status: 404,  errorCode: "RESOURCE\_NOT\_FOUND"  } |
| Falta un parámetro en el JSON al agregar/ una Playlist | 400 | {  status: 400,  errorCode: "BAD\_REQUEST"  } |

## Usuarios y Listenings (bonus para llegar a nota 10)

Diseñe e implemente la API Rest para manipular usuarios y dar de alta "listenings".

Considere la consistencia del modelo a la hora de borrar un usuario del sistema.

Se evaluará:

1- Diseño de la API (uso de buenas prácticas)

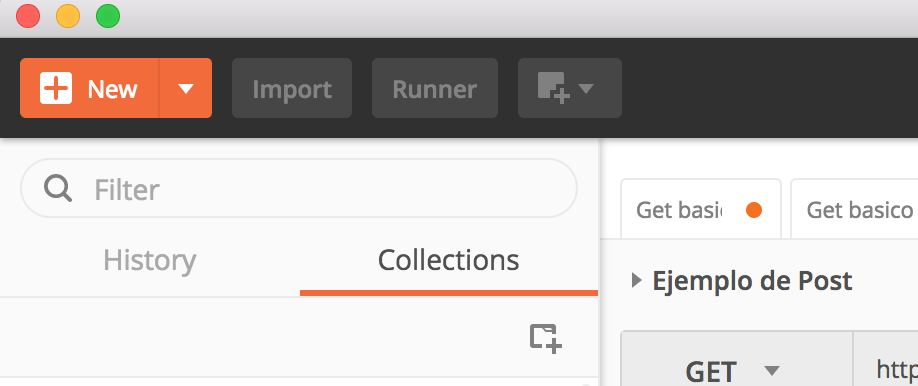
2- Completitud

3- Tests de postman siguiendo como modelo los entregados por la cátedra

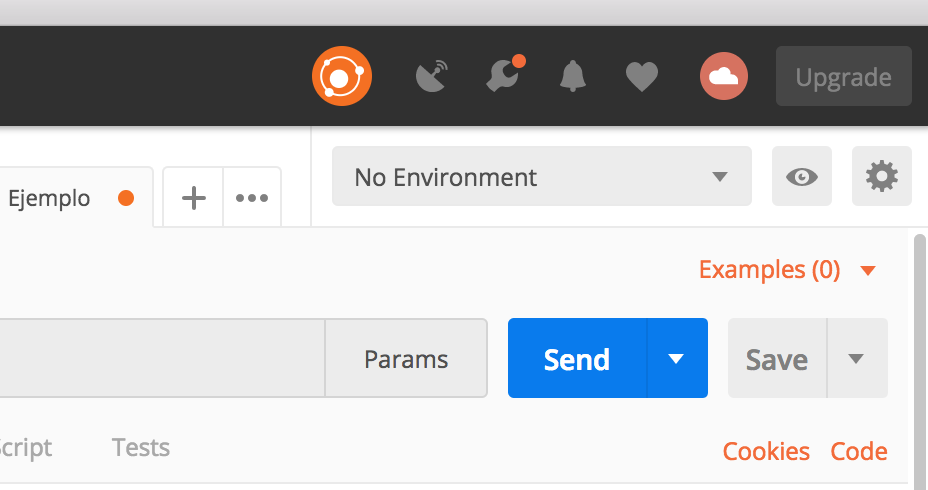
**CORRIENDO LOS TESTS DE INTEGRACIÓN**

Realice por única vez:

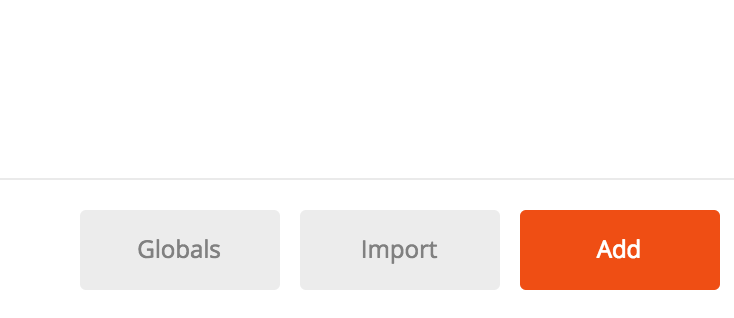
* Importe en postman la colección UNQfy.postman\_collection.json



* Importe en postman el ambiente UNQfy.postman\_environment.json.

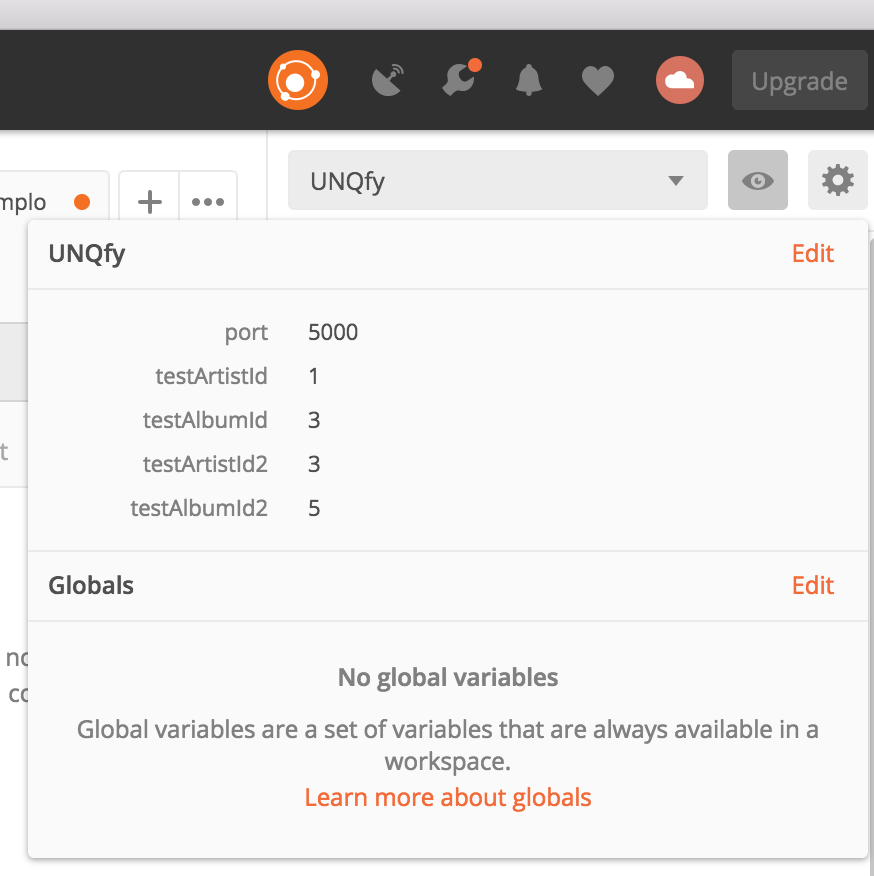


Click en el boton de stettings del environment



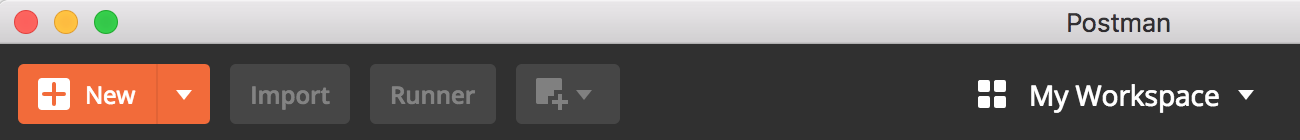
Botón import, seleccionar el archivo y luego seleccionar ese environment con como el activo.

* En el ambiente UNQfy importado deberá configurar el puerto donde tiene corriendo su servicio (variable port).

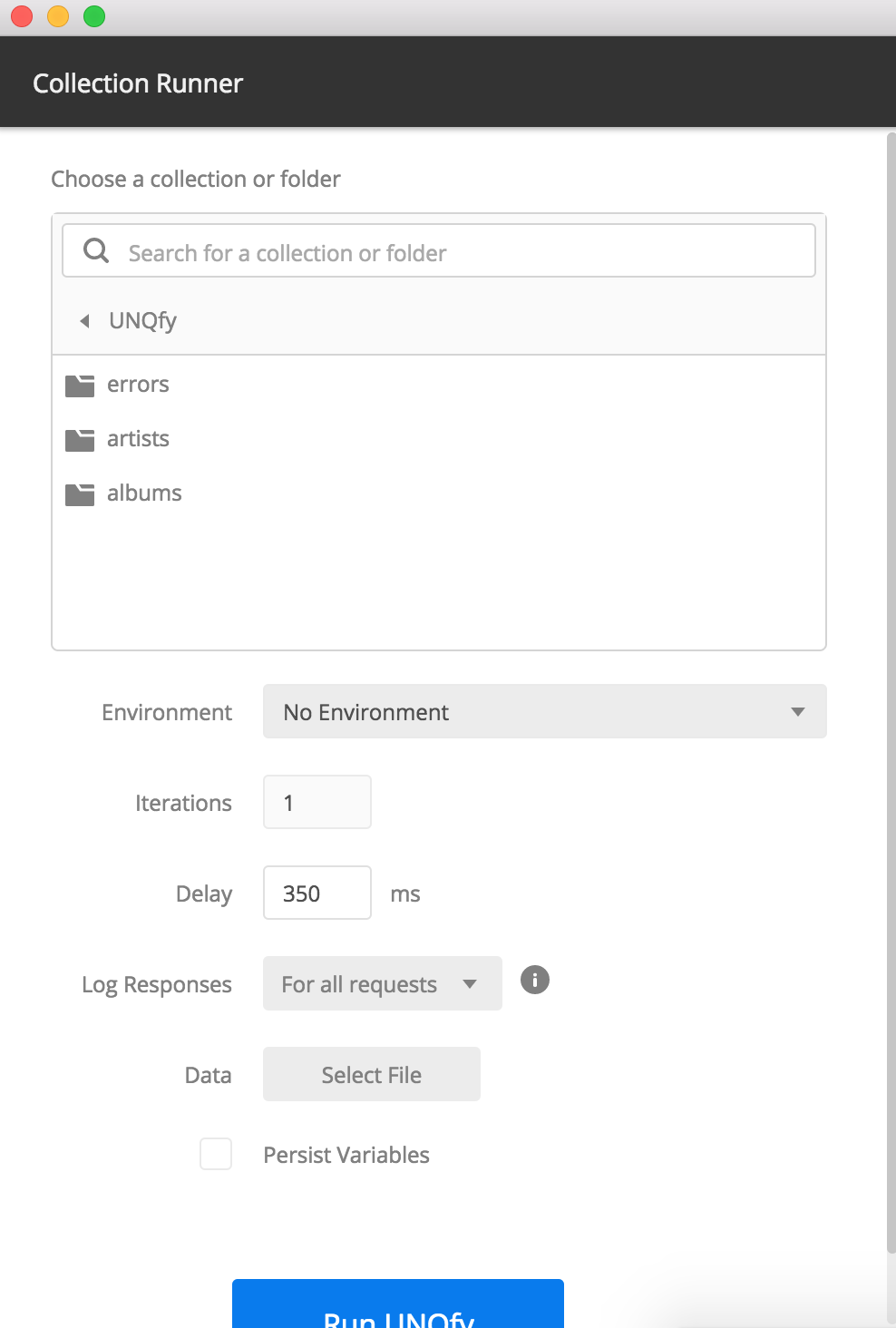


Cada vez que quiera correr los tests:

* Borre su archivo de estado en UNQfy y reinicie el servicio.
* Click en el boton “runner” en postman



* Elija la colección UNQfy.
* En el campo delay, ingrese “350 ms”.



NOTAS/Aclaraciones:

* Los tests modifican el estado de unqfy, es por eso que deberá borrar su estado antes de correr cada test.