

# UAIM

---

author: Wiktor Zawadzki

## Getting Started

---

Run below commands to start on your local machine

1. `git clone https://github.com/wiktoz/lab2.git`
2. `cd lab2`
3. `npm install`
4. `npm start`

## App Description

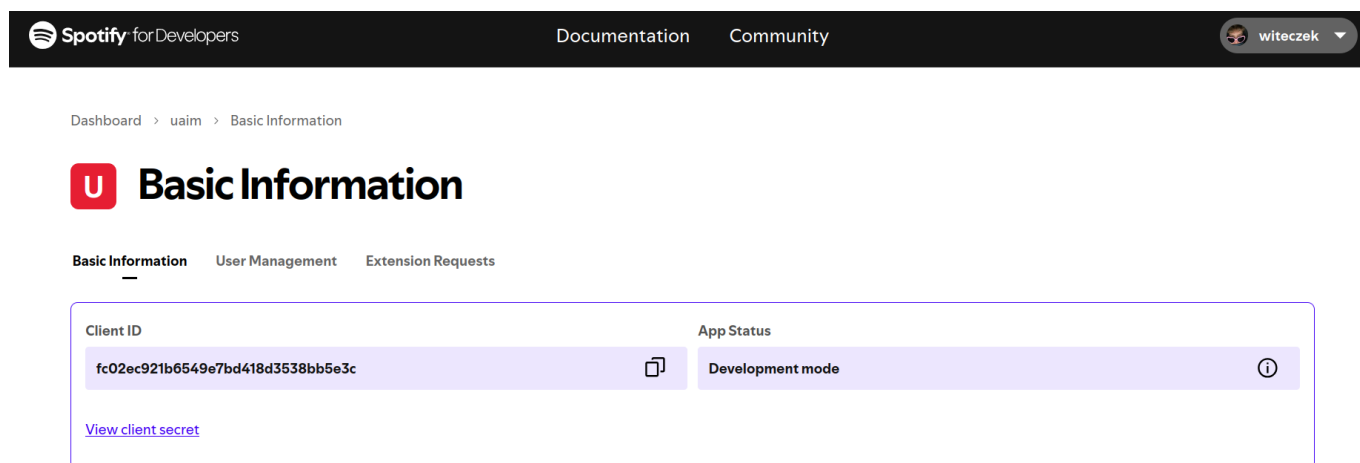
---

Application is presenting Top 50 Global Listened Songs. Data is fetched from Spotify.

API url: <http://api.spotify.com>

DOCS url: <http://developer.spotify.com>

In order to get access to the Spotify API we need to create account and get `client_id` and `client_secret`.



The screenshot shows the Spotify for Developers 'Basic Information' page. At the top, there's a navigation bar with the Spotify logo, 'for Developers', and links for 'Documentation' and 'Community'. A user profile 'witeczek' is in the top right. Below the navigation bar, a breadcrumb trail reads 'Dashboard > uaim > Basic Information'. The main heading is 'Basic Information' with a red 'U' icon. Underneath, there are tabs for 'Basic Information', 'User Management', and 'Extension Requests'. The 'Basic Information' tab is active. It contains two sections: 'Client ID' and 'App Status'. The 'Client ID' section shows the ID 'fc02ec921b6549e7bd418d3538bb5e3c' with a copy icon and a link to 'View client secret'. The 'App Status' section shows 'Development mode' with an information icon.

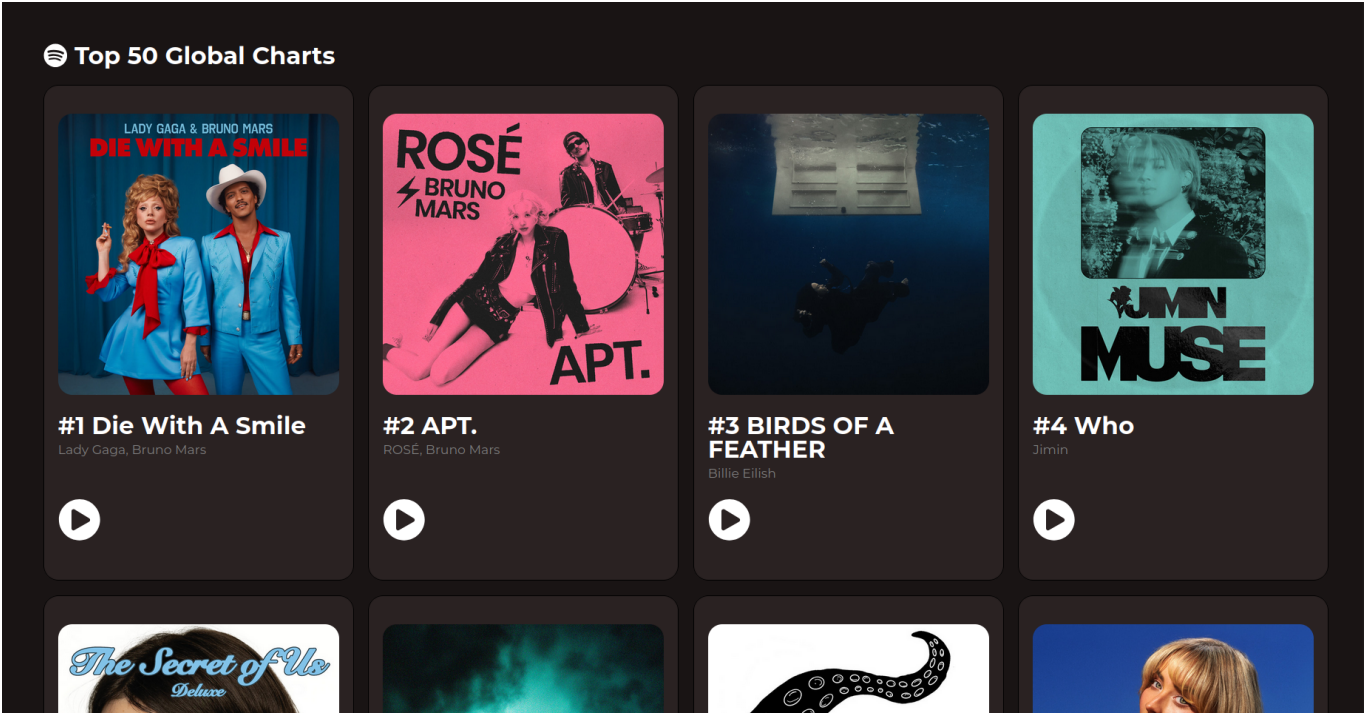
First action we need to perform to get access to the API is to send request to the `https://accounts.spotify.com/api/token` endpoint including `client_id` and `client_secret`. When credentials are correct we should obtain `access token` in response. From now we will include it in headers as `Authorization` header for every request we will make in the future.

In localStorage we also save token expiration time and re-fetch token automatically when it is expired.

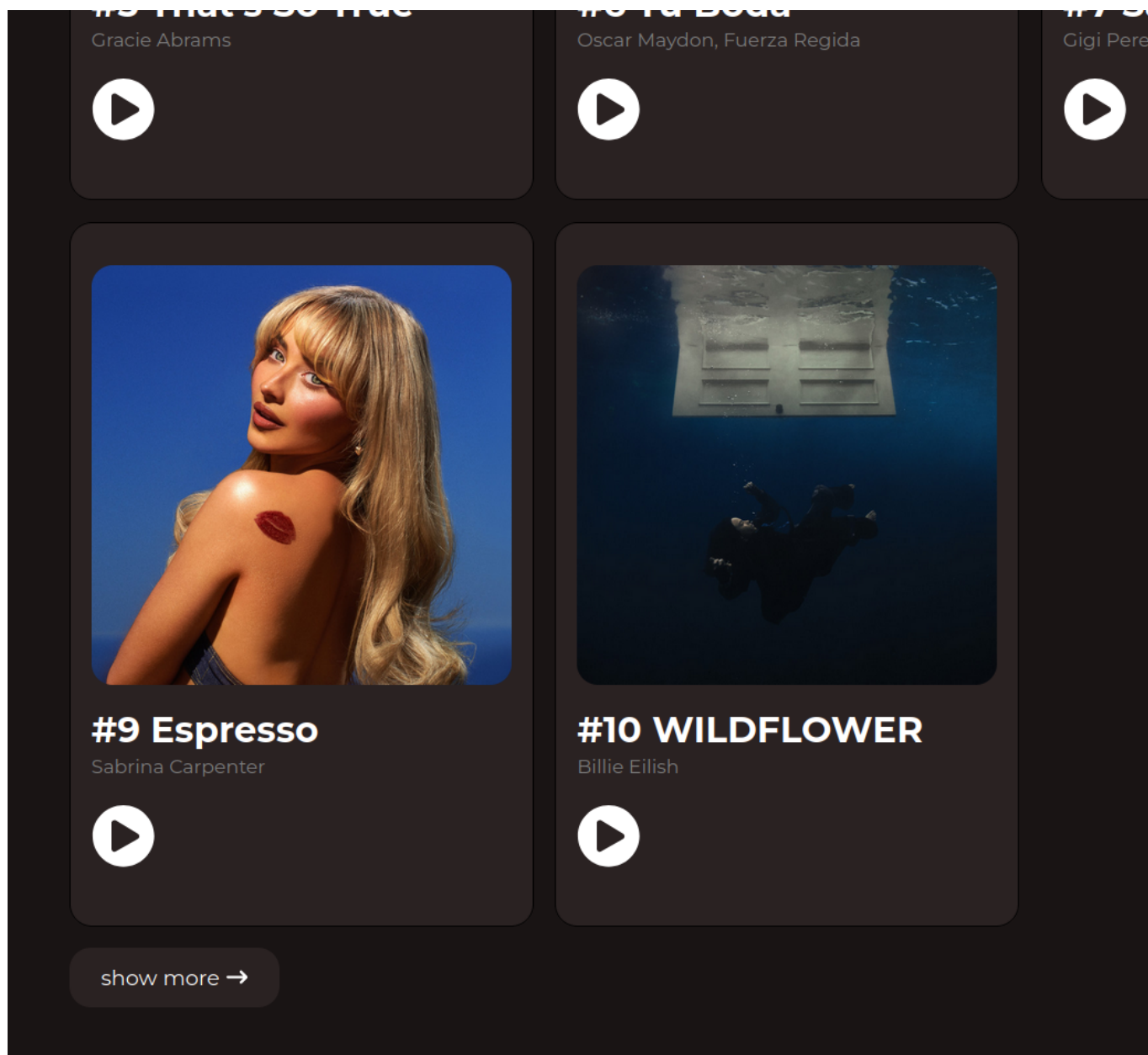
When we have a token we make request to fetch **Top 50 Global Charts** songs. The dataset is abundant therefore we remove unnecessary fields and save everything in localStorage. Songs can change their position in ranking for that reason I decided to include field containing **time to next fetch** (by default one hour).

Unfortunately Spotify does not provide option to fetch data in batches (e.g. using **offset** and **skip**) so I download whole dataset and render in React only 10 songs. Later on user can click **show more** to print more data. Many photos on the website has a huge impact on Largest Contentful Paint (LCP).

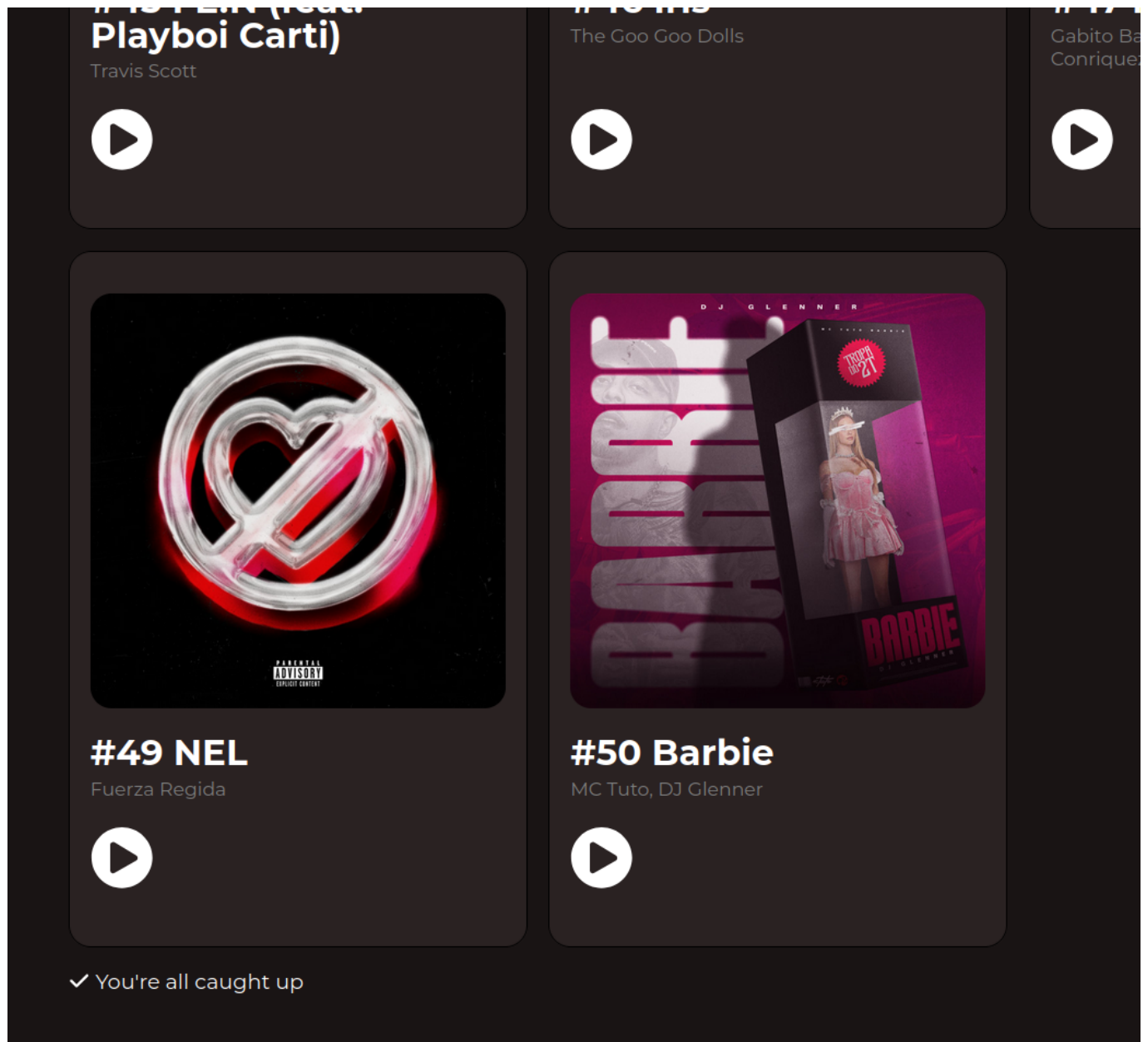
App View



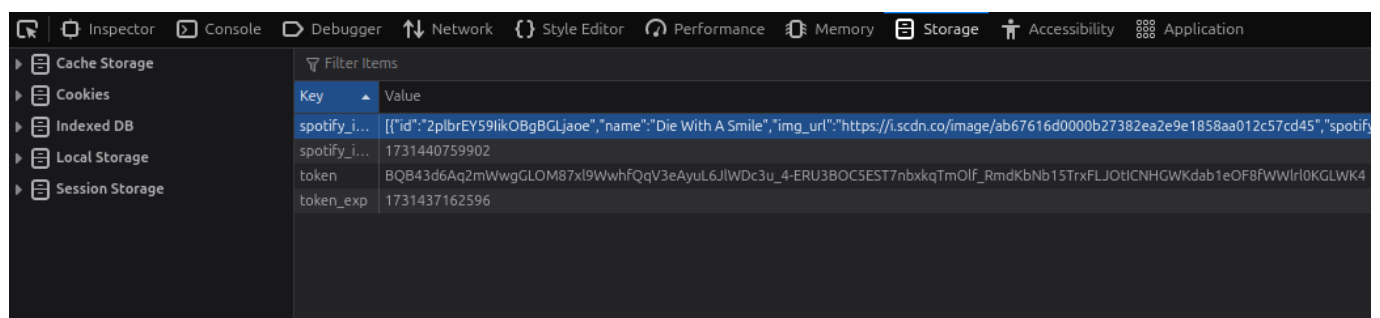
Show More



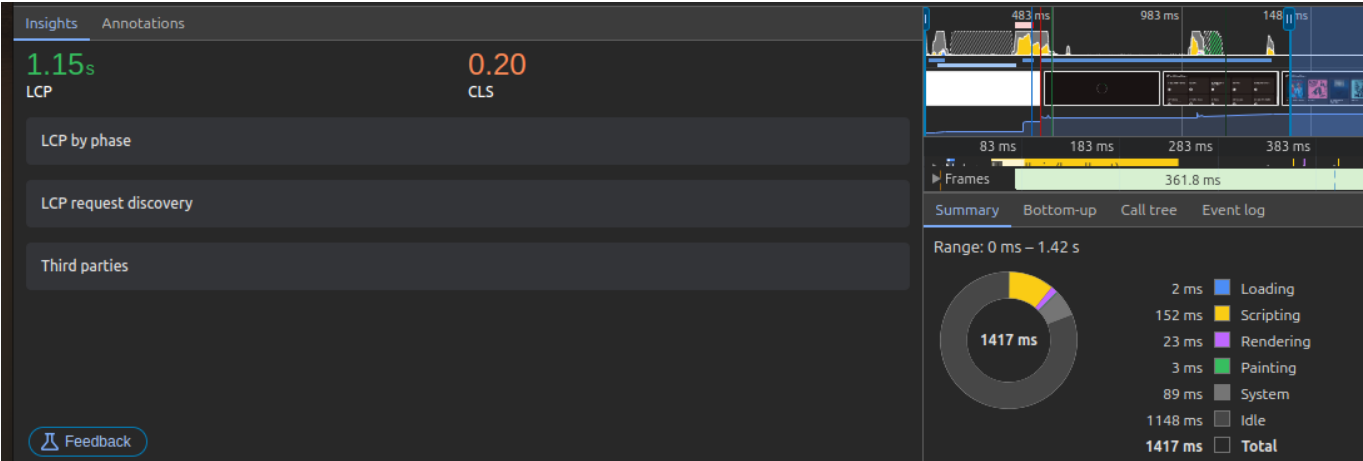
Information we expanded all possible songs



Data fetched from Spotify is reduced to the necessary fields and saved in localStorage. Moreover, there is a mechanism implemented to re-fetch data from Spotify to the localStorage every hour (by default - we can change that value).



First fetch - data from the API



Second fetch - data from localStorage. We got lower LCP time by a half.

