### 4. How does the application hook up to the Spotify API?

#### Register the app at Spotify

The app must be registered at Spotify (needs to be done once only).

My Spotify Developer account: <https://developer.spotify.com/my-applications/#!/applications>

App name: Jammming

#### 

#### Type of request

As a way of doing requests I intend to follow the steps as provided in [Course Unit 2, Requests 2](https://www.codecademy.com/courses/intermediate-javascript-requests/lessons/requests-ii/exercises/requests-intro-ii?action=lesson_resume&program_content_id=720c50d30543383fd8ee82ce13b3ef31&program_id=8e14e567a49f749ac096378873934906) step 12-17.

Boilerplate code for a GET-request is then:

async function getData() {

try {

let response = await fetch('https://api-to-call.com/endpoint');

if (response.ok) {

let jsonResponse = await response.json();

return jsonResponse;

}

throw new Error('Request failed!');

}

catch (error) {

console.log(error);

}

}

and for a POST-request:

async function getData() {

try {

let response = await fetch('https://api-to-call.com/endpoint', {

method: 'POST',

body: JSON.stringify({id: 200}) });

if (response.ok) {

let jsonResponse = await response.json();

return jsonResponse;

}

throw new Error('Request failed!');

}

catch(error) {

console.log(error);

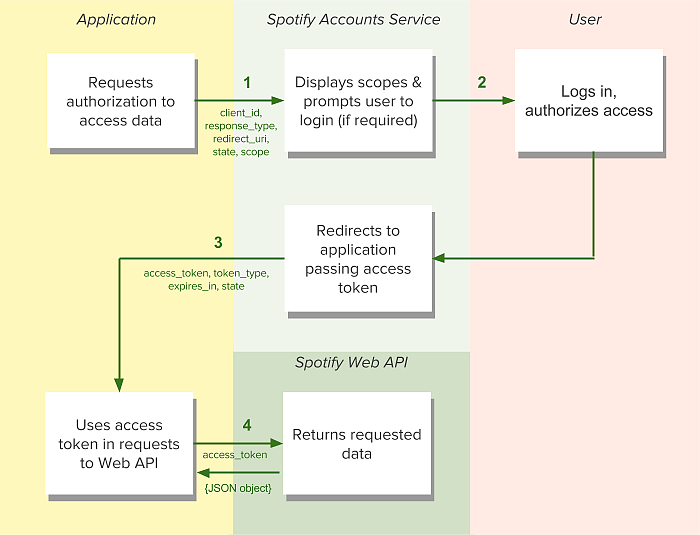
}

}

#### Authorization flow

In the project’s instructions Implicit Grant Flow (see [Course Unit 2, Article Authentication and OAth](https://www.codecademy.com/programs/8e14e567a49f749ac096378873934906/items/42276cdf573c1ac09591a0465853af34), step 9) is used, which doesn’t require the use of a client secret key, so I will stick to that.

The Implicit Grant Flow as implemented by Spotify is as follows [[1]](#footnote-0) :



The API documentation gives following example request for requesting authorization:

GET <https://accounts.spotify.com/authorize?client_id=5fe01282e94241328a84e7c5cc169164&redirect_uri=http:%2F%2Fexample.com%2Fcallback&scope=user-read-private%20user-read-email&response_type=token&state=123>

However in this project’s description only the required fields (client\_id, response\_type and redirect\_uri) are put in the query, as well as scope:

GET <https://accounts.spotify.com/authorize?client_id=CLIENT_ID&response_type=token&scope=playlist-modify-public&redirect_uri=REDIRECT_URI>

So I will stick to that.

Upon sending that request the user will be asked to login to his Spotify-account, if token has expired.

The Spotify API will then redirect to the redirect\_uri (as provided in the request) so for the development server that will be <https://localhost:3000> but that will need to be updated when published to the web. A token is sent with this response, that needs to be stored in the program to send along with every next request to the Spotify API. An example of a response will then be:

<https://localhost:3000#access_token=NwAExz...BV3O2Tk&token_type=Bearer&expires_in=3600&state=123> so ‘access\_token’ is the thing to extract from that response-message.

When access is denied the response will look like: <https://localhost:3000?error=access_denied&state=123> and then error is the thing we will be looking for.

In the next section I will describe how to use this token to make a query to the Spotify API.

#### The authorization request

Using async GET request format, the request for a token would look like:

Boilerplate:

async function getData() {

try {

let response = await fetch('https://api-to-call.com/endpoint');

if (response.ok) {

let jsonResponse = await response.json();

return jsonResponse;

}

throw new Error('Request failed!');

}

catch (error) {

console.log(error);

}

}

Actual code:

// data for accessing Spotify API

const clientId = ‘<put client ID here>’;

redirectUri = ‘http://localhost:3000’;

const url = `https://accounts.spotify.com/authorize?client\_id=${clientId}&response\_type=token&scope=playlist-modify-public&redirect\_uri=${redirectUri}`;

let accessToken;

async function getAccessToken() {

// copied this snippet from Yelp.js from project Ravenous Part4, but don’t understand this

if (accessToken) {

return new Promise(resolve => resolve(accessToken));

}

try {

let response = await fetch(url);

if (response.ok) {

let jsonResponse = await response.json();

accessToken = jsonResponse.access\_token;

return jsonResponse;

}

throw new Error('Request for token failed!');

}

catch (error) {

console.log(error);

}

}

1. "Understanding the Spotify Web API | Labs." 9 mrt.. 2015, <https://labs.spotify.com/2015/03/09/understanding-spotify-web-api/>. Geopend op 21 aug.. 2017. [↑](#footnote-ref-0)