P’OLOC PLAYZ Developer Guide

PREPARED FOR

Developers and collaborators of P’oloc Playz

PREPARED BY

EL ALLAM Valentin

FAUVET Jules

Table of contents

Overview

Our website is designed to allow users to search for videos on 3 different video platforms by using their respective API providing Embedded links to their videos along with useful and relevant information. The 3 platforms we are using are Youtube, DailyMotion and Vimeo.

Our users are given the opportunity to register with their email.

Once logged in, they are allowed to save videos from all sources to their favorite playlist, this playlist, along with their profile information, is stored on our distant Firebase server that is always up and available, no matter where you host the website from, be it from your PC or another distant server.

1. Project goals

We had to build a site that offered several features:

-Be able to search for videos from different video platforms

-Be able to register and Log in

-Be able to create a playlist of videos from all available platforms, and have the possibility to edit that playlist

2. Obstacles

We ran into a few technical problems along the way.

The most prominent one was the availability of the API services of the different video platforms, during development we ran out of daily API requests on youtube and Vimeo because we were sending a lot of requests for testing purposes.

We had to properly adapt our usage of bootstrap to fit it to our site.

3. Requests and API

a. Video platform APIs

Our site uses video platforms APIs to get a list of videos for a given query, along with firebase for the online database.

The APIs work as follows:

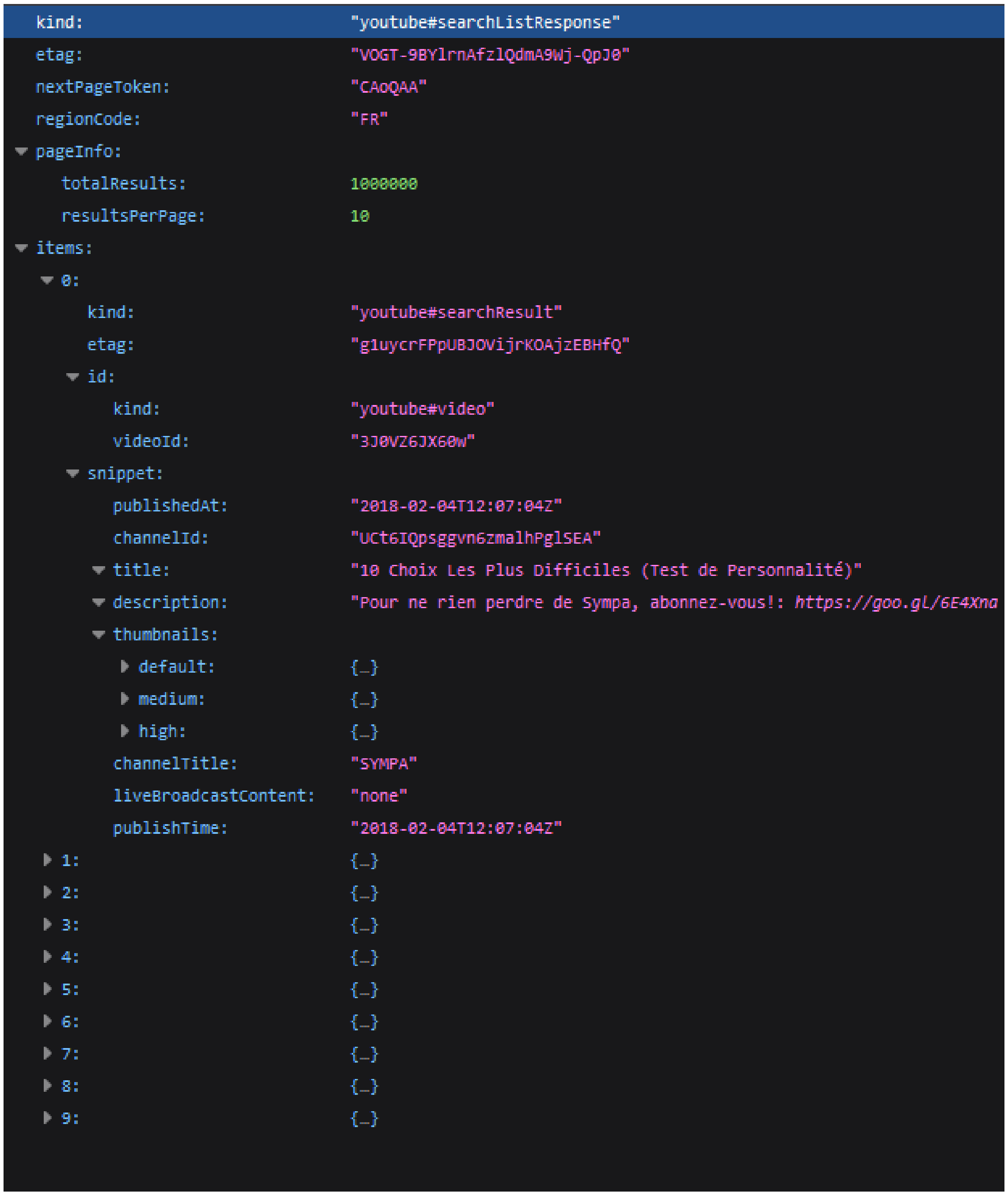
-we send a request to an API via a link that looks like this:

```
'https://content-youtube.googleapis.com/youtube/v3/search?part=snippet&q=' +
keyword + '&key='+ API_key+ '&maxResults=10'
```

The first part of the link refers to which video platform we’re using, then the keyword used for our query is added as well, along with the API key we generated from our video platform account and finally we used the `maxResults=10` argument to limit the amount of videos

Similar API requests go for DailyMotion and Vimeo.

These APIs, unless there's an error, returns us a JSON that looks like this:



```
  kind:                         "youtube#searchListResponse"
  etag:                         "VOGT-9BYlrnAfzlQdmA9Wj-QpJ0"
  nextPageToken:                "CAoQAA"
  regionCode:                   "FR"
▼ pageInfo:
    totalResults:               1000000
    resultsPerPage:             10
▼ items:
  ▼ 0:
      kind:                     "youtube#searchResult"
      etag:                     "g1uycrFPpUBJOVijrKOAjzEBHfQ"
    ▼ id:
        kind:                   "youtube#video"
        videoId:                "3J0VZ6JX60w"
    ▼ snippet:
        publishedAt:            "2018-02-04T12:07:04Z"
        channelId:              "UCt6IQpsggvn6zmalhPglSEA"
      ▼ title:                  "10 Choix Les Plus Difficiles (Test de Personnalité)"
      ▼ description:            "Pour ne rien perdre de Sympa, abonnez-vous!: https://goo.gl/6E4Xna
      ▼ thumbnails:
        ▶ default:              {…}
        ▶ medium:               {…}
        ▶ high:                 {…}
        channelTitle:           "SYMPA"
        liveBroadcastContent:   "none"
        publishTime:            "2018-02-04T12:07:04Z"
  ▶ 1:                          {…}
  ▶ 2:                          {…}
  ▶ 3:                          {…}
  ▶ 4:                          {…}
  ▶ 5:                          {…}
  ▶ 6:                          {…}
  ▶ 7:                          {…}
  ▶ 8:                          {…}
  ▶ 9:                          {…}
```

The relevant fields from the received JSON are then extracted and either displayed on the current page or saved in the backend for later use.

b. Firebase communication

Our Firebase database is used with requests that correspond to it.

Here’s an example of an GET HTTP request to a firebase database:

```
curl 'https://[PROJECT_ID].firebaseio.com/users/jack/name.json'
```

If the request is successful, we once again obtain a JSON that can be exploited via its field to be directly displayed on the current page or saved in the backend for later use. A successful request is indicated by a 200 OK HTTP status code. The response contains the data associated with the path in the GET HTTP request.

```
{ "first": "Jack", "last": "Sparrow" }
```

4. Hardware

For development on our side, we were using our personal computers running on Windows 10 to host the website with WAMP.

As for our database hosting, it was made possible through the servers of Google's Firebase, here are the characteristics and capabilities of their servers summed up in a table (a full version is available here):

Service	Free Spark plan
Phone authentication	10k/month
Stored data	1 GiB total
Network egress	10 GiB/month
Document writes	20K writes/day
Document reads	50K reads/day
Document deletes	20K deletes/day
Storage	10 GB
Data transfer	360 MB/day
Realtime Database	
Simultaneous connections	100
GB stored	1 GB
GB downloaded	10 GB/month
Cloud Storage	
GB stored	5 GB
GB downloaded	1 GB/day
Upload operations	20K/day
Download operations	50K/day

Virtual Device Tests	10 tests/day
Physical Device Tests	5 tests/day

5. Software

We are using different softwares and technologies for this project, mostly web technologies, they are listed below:

- HTML version 5
- Javascript version 1.5
- CSS
- All of the above with the Bootstrap version 3.3.4 framework
- WAMP version 3.2.6 64bit for windows 10
- Windows 10 operating system
- Visual Studio Code as IDE

Our site was tested for display on 16/9 screen ratio in both the latest Google Chrome and Mozilla Firefox builds.

6. Deployment

In order for you to properly deploy the site on your machine, you only have to deploy the website part of it, as the database part is a firebase project that is hosted on a server and readily available at all times.

If you wish to create the needed environment on your PC for a standard deployment, simply install WAMP in the default proposed directory, then paste the whole code from our GitHub directly into the folder “`C:\wamp64\www`”; then after successfully launching WAMP, you can access our site from the localhost at the following address : http://localhost/Urba_des_SI/index.html