Work Activity Monitor with Fitbit

Eduardo de Prado Cañal

INDEX

1.Introduction..3

2. Theoretical background.. 4
    2.1 OAuth 2.0 authentication.. 4
    2.2 Data HTTP requests... 6

1. Introduction

The main idea of this project is creating a tool to extract raw fat from a Fitbit monitor, with the purpose of process it and add custom features to the fitbit's applications that are totally generic for the regular user.

The name *Work Activity Monitor with Fitbit* comes with the idea of monitoring the activity of a fitbit's user when he/she is at work (or in the University, school... etc).
If you are a fitbit user, you can realize that everything revolves around the sport and how tracking your heart rate or steps can improve your performance while you are working out, but no one cares about how your inactivity during the hours you are on the work, sitting down on your chair, moving only your eyes, or maybe your fingers, can totally affect your health, your sports performance, or ever your efficiency on your tasks.

The user of this application will be able to check how looks him/her activity during the work hours, set and delete silence alarms on this time period to be more active, and compare the progressions.

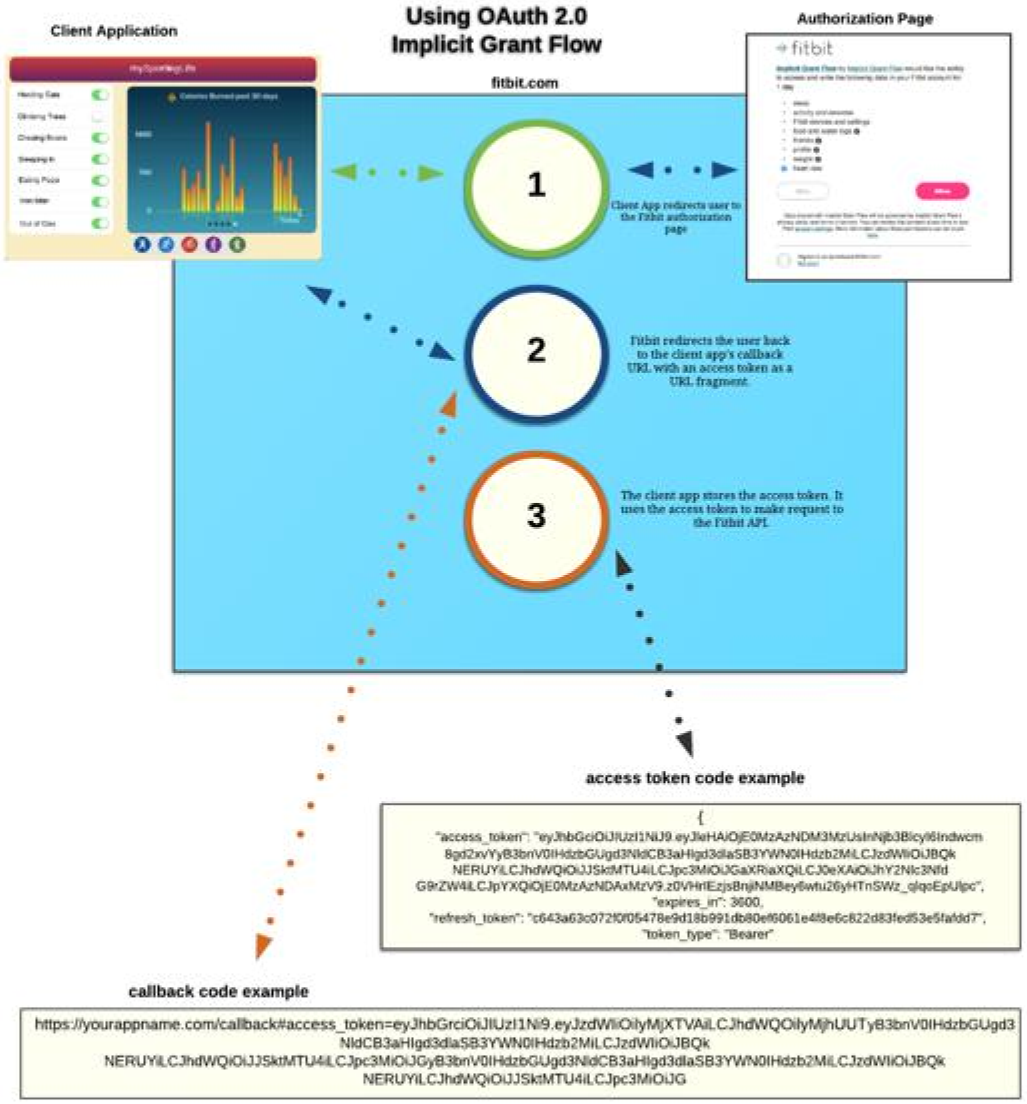
2. Theoretical background

2.1 OAuth 2.0 authentication

Fitbit uses OAuth 2.0 for user authorization and API authentication, this protocol requires the third application to obtain a token authorized by the user in order to be able to make the HTTP requests to obtain the desires fitbit data.

The first step we have to do is register the application on https://dev.fitbit.com/
This is necessary because fitbit provides you an "OAuth 2.0 client ID" that identifies the application and allows you to do the request of the token.

I used the Implicit Grant Flow to do the authentication.

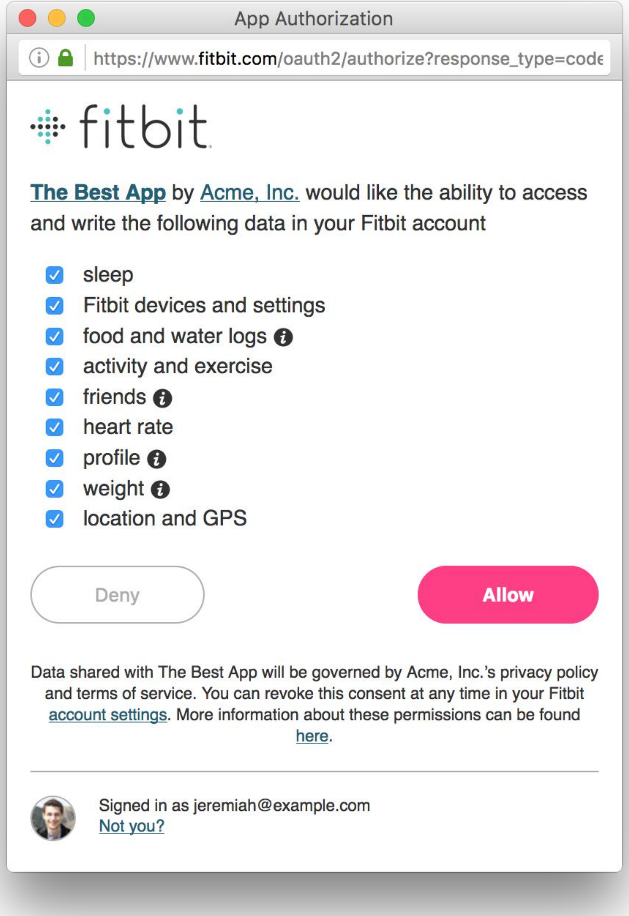

Client Application
Using OAuth 2.0
Implicit Grant Flow
fitbit.com
Authorization Page
fitbit
1
Client App redirects user to the Fitbit authorization page
2
Fitbit redirects the user back to the client app's callback URL with an access token as a URL fragment.
3
The client app stores the access token. It uses the access token to make request to the Fitbit API.
access token code example
callback code example

```
{
"access_token": "eyJhbGciOiJIUzI1NiJ9.eyJleHAiOjE0MzAzNDM3MzUsInNjb3BlcyI6Indwcm
8gd2xvYyB3bnV0IHdzbGUgd3NldCB3aHlgd3dlaSB3YWN0IHdzb2MiLCJzdWIiOiJBQk
NERUYiLCJhdWQiOiJJSktMTU4iLCJpc3MiOiJGaXRiaXQiLCJ0eXAiOiJhY2Nlc3Nfd
G9rZW4iLCJpYXQiOjE0MzAzNDAxMzV9.z0VHrIEzjsBnjiNMBey6wtu26yHTnSWz_qlqoEpUlpc",
"expires_in": 3600,
"refresh_token": "c643a63c072f0f05478e9d18b991db80ef6061e4f8e6c822d83fed53e5fafdd7",
"token_type": "Bearer"
```

```
https://yourappname.com/callback#access_token=eyJhbGrciOiJIUzI1Ni9.eyJzdWliOilyMjXTVAiLCJhdWQOilyMjhUUTyB3bnV0IHdzbGUgd3
NldCB3aHlgd3dlaSB3YWN0IHdzb2MiLCJzdWliOiJBQk
NERUYiLCJhdWQiOiJJSktMTU4iLCJpc3MiOiJGyB3bnV0IHdzbGUgd3NldCB3aHlgd3dlaSB3YWN0IHdzb2MiLCJzdWliOiJBQk
NERUYiLCJhdWQiOiJJSktMTU4iLCJpc3MiOiJG
```

As can be seen on the image, first of all the client application asks for the token.

After that the user have to authorize which data the application will have access.



The Best App by Acme, Inc. would like the ability to access and write the following data in your Fitbit account

- ☑ sleep
- ☑ Fitbit devices and settings
- ☑ food and water logs
- ☑ activity and exercise
- ☑ friends
- ☑ heart rate
- ☑ profile
- ☑ weight
- ☑ location and GPS

Data shared with The Best App will be governed by Acme, Inc.'s privacy policy and terms of service. You can revoke this consent at any time in your Fitbit account settings. More information about these permissions can be found here.

Signed in as jeremiah@example.com
Not you?

Then with the authorization, fitbit redirects the user to the application again and sends the token on the URL.
With this token, the application can do the requests to obtain all the allowed data.

The first problem is that this is thought for a web application, where all is stored in a web server. When the application is registered on fitbit's page they ask you to register the URL of the application, and just in case, is mandatory to fill a REDIRECT_URI field on the token request.
So when the redirection is done, your web application just have to read the url path to obtain the token.

To solve that I had to do the request of the token with the loopback direction as REDIRECTION_URI and using the function web() of matlab to open a native web browser, that allowed me to obtain the url with the token directly from the webrowser figure.

2.2 Data HTTP requests

To obtain the data from the fitbit’s BBDD the API allows the application to do HTTP requests with specific querys.

An example of HTTP requests to obtain the activities of a [user-id] on a specific [date] that returns a json structure:

```
  GET https://api.fitbit.com/1/user/[user-id]/activities/date/[date].json
```

To do the petition is necessary to add the header:
`Authorization: Bearer [TOKEN]`

So in order to do an HTTP requests with headers I used the library **`urlread2`**, that is a thirds-party library created to improve the native `urlread` library, and allow matlab applications to do more specific petitions.

After the data is received, it will use the `loadjson` function to transform the json data on a matlab object that the application will use to show the results.