API con autenticación

Objetivos

- Obtener el id y la clave para autenticarse a una API
- Conectarse a una API que requiere autenticación
- Utilizar la API del diccionario de oxford
- Estudiar otros tipos de autenticación

Introducción

Muchas APIs requieren de autenticación para poder acceder a sus servicios.

Para autenticarse con una API nuestro primer paso consiste en conseguir una clave para conectarse, las mismas APIs disponen de servicio de registro y al completarlo te entregan un id.

Consiguiendo la clave de la API de Oxford Dictionary

Para conseguir esta clave entraremos a: https://developer.oxforddictionaries.com y llenaremos el formulario de registro, también necesitaremos confirmar nuestro email.

En algunos casos se utilizan dos claves, un app_id y un app_key, en otros casos un key y un secret, independiente de cuantas claves el sistema siempre es similar.

Una vez obtenido el key (o los keys) podremos acceder a la API

Autenticando desde Postman

Existen distintos tipos de autenticación, la mayoría de los modelos consiste en pasar un token en los **header** del request.

```
require 'uri'
require 'net/http'

def request(url_requested)
   url = URI(url_requested)
```

```
    http = Net::HTTP.new(url.host, url.port)
    http.use_ssl = true
    http.verify_mode = OpenSSL::SSL::VERIFY_PEER

    request = Net::HTTP::Get.new(url)
    request['app_id'] = 'Tú_ID'
    request['app_key'] = 'Tu_llave_secreta'
    response = http.request(request)
    return JSON.parse(response.body)
end

word = "test"
request("https://od-api.oxforddictionaries.com:443/api/v1/entries/en/#{word}")
```

Veremos que el resultado muestra mucha información, podemos mostrarla de forma ordenada con `pp

```
pp request("https://od-api.oxforddictionaries.com:443/api/v1/entries/en/#{word}")
```

La pregunta importante es que información es la que queremos, por ahora nos conformaremos con la definición **definitions**, así que iremos del final hasta el principio.

- Definitions es el key de un hash `['definitions']`
- El hash está dentro de un arrego, es el primer elemento `[0]['definitions']`
- Este arreglo parte de un hash bajo la clave **senses** `['senses'][0]['definitions']`
- Senses es un key de entries `['entries']['senses'][0]['definitions']`
- El hash está dentro de un arreglo y es el primer elemento `[0]['entries']['senses'][0]['definitions']`
- El arreglo está asociado al key lexicalEntries `['lexicalEntries'][0]['entries']['senses'][0]['definitions']`
- El diccionario es el primer elemento de un arreglo `[0]['lexicalEntries'][0]['entries']['senses'][0]['definitions']`
- Todo está bajo la clave de results `['results'][0]['lexicalEntries'][0]['entries']['senses'][0]['definitions']`

```
result = request("https://od-api.oxforddictionaries.com:443/api/v1/entries/en/test")
result['results'][0]['lexicalEntries'][0]['entries'][0]['senses'][0]['definitions']
```

No siempre es tan complicado, muchas veces los datos están con uno o dos niveles de profundidad, pero por lo mismo este es un buen caso de estudio.

Otros tipos de autenticación

En este caso nos tocó ingresar la autenticación en el header, esta es una de las opciones mas frecuentes pero algunas APIs tienen diferentes modelos de autenticación.

- En algunos casos nos tocará ingresar la clave en el body.
- En algunos caso el token de autenticación cambiará después de cada request y tendremos que usar el último valor para rescatar el siguiente.

También existen otros modelos mas complejos como el OAuth2, el cual delega la autenticación a otros proveedores de servicios web como Facebook, Google, Github, etc para probar la identidad de una solicitud.

De todas formas lo importante siempre es leer con calma la documentación y encontrar las pistas necesarias para lograr la conexión y en el peor de los casos siempre existe la opción de comunicarse con los desarrolladores de la API para pedir ayuda o buscar otra API que logre el mismo propósito.

Cuidado con los token

Lo último y mas importante es que muchos servicios tienen límites de consumo y otros tanto manejan información delicada, los tokens, passwords y otros datos delicados entregados no deben ser subidos a repositorios públicos o compartidos con terceros.

¿Cómo podemos manejar los tokens?

Con ARGVS (vector de argumentos)

Hay varias estrategias, podemos utilizar una muy simple que consiste en dejar los datos sensibles fuera del programa, al cargar el programa le pasaremos como parámetro los datos necesarios.

Con variables de entorno

En Linux y OSX podemos utilizar variables de entorno.

Esta opción consiste en utilizar variables que se definen en el terminal y las podemos llamar desde nuestro programa.

Veamos un ejemplo básico, en el terminal escribiremos `export a=5`. Luego entraremos a IRB y escribiremos ENV['a'] y observaremos el valor 5.

Persistiendo las variables de entorno

Pero este valor solo quedará en la sesión, al abrir un terminal nuevo no lo tendremos. Para hacerlo persistente podemos escribir esta línea en el archivo `.bashrc` o `.bash_profile`. Este archivo que se encuentra en la carpeta de usuario. Lo podemos abrir con un editor de texto y en la última línea agregar nuestra variable.

```
export API_1_SECRET_KEY=23123u2139183
```

Finalmente, después de guardar, para recargar los cambios realizados dentro del archivo ocuparemos la siguiente línea:

```
source ~/.bash_profile
```

Si queremos probar si funcionó podemos hacerlo desde IRB con `puts ENV['API_1_SECRET_KEY']` o directamente desde el terminal con `echo $API_1_SECRET_KEY`. El signo peso es necesario en el echo para indicar que es una variable.