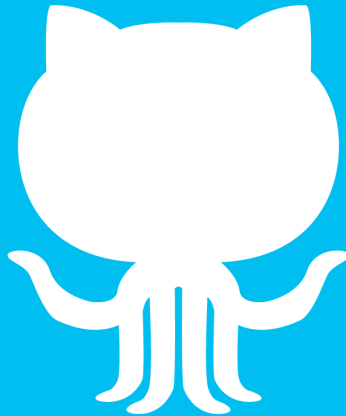


Consumo de API de GitHub



Consumir Web APIs es común en las aplicaciones web. En el tema de hoy vamos a estudiar como consumir el API de Github.

El concepto de real time o tiempo real se basa en usar un conjunto de tecnologías y herramientas que nos permitan abrir un canal de comunicación entre el cliente y el servidor y que éste se mantenga abierto en todo momento, entonces no tenemos que preocuparnos por estar preguntándole al servidor si tiene nueva información que entregarnos, sino que cuando el servidor tiene nuevos datos se los notifica al cliente y viceversa.

La documentación oficial la encuentras en este sitio web: <https://developer.github.com/v3/>

Sin embargo te recomendamos leer una página guía que ayuda mucho a entender mejor la API: <https://developer.github.com/guides/getting-started/>

En el tema anterior estudiamos cómo consumir un JSON externo mediante AJAX. Es muy útil saberlo porque el API de Github funciona con JSON.

GitHub Developer

[API](#)
[Blog](#)
[Early Access](#)
[Support](#)

API

[Reference](#)
[Webhooks](#)
[Guides](#)
[Libraries](#)

Overview

This describes the resources that make up the official GitHub API v3. If you have any problems or requests please contact [support](#).

- [i. Current Version](#)
- [ii. Schema](#)
- [iii. Parameters](#)
- [iv. Root Endpoint](#)
- [v. Client Errors](#)

- Overview
- Activity
- Gists
- Git Data
- Issues
- Migration
- Miscellaneous

URL de API de github

Las operaciones del api se realizan a través de la url <https://api.github.com>. Recordando el tema anterior, reemplazamos en el código base la url por la url de github.

```
$.ajax({
  url: 'https://api.github.com',
  complete: function(xhr) {
  }
});
```

Algo muy útil de la API de Github es que si hacemos una simple solicitud a esta url nos responde con las posibles urls que podemos utilizar. Mediante la función `console.log()` analicemos la respuesta:

```
▼ responseJSON: Object
  authorizations_url: "https://api.github.com/authorizations"
  code_search_url: "https://api.github.com/search/code?q={query}&page,per_page,sort,order"
  current_user_authorizations_html_url: "https://github.com/settings/connections/applications/{client_id}"
  current_user_repositories_url: "https://api.github.com/user/repos?type,page,per_page,sort"
  current_user_url: "https://api.github.com/user"
  emails_url: "https://api.github.com/user/emails"
  emojis_url: "https://api.github.com/emojis"
  events_url: "https://api.github.com/events"
  feeds_url: "https://api.github.com/feeds"
  followers_url: "https://api.github.com/user/followers"
  following_url: "https://api.github.com/user/following/{target}"
  gists_url: "https://api.github.com/gists/{gist_id}"
  hub_url: "https://api.github.com/hub"
  issue_search_url: "https://api.github.com/search/issues?q={query}&page,per_page,sort,order"
  issues_url: "https://api.github.com/issues"
  keys_url: "https://api.github.com/user/keys"
  notifications_url: "https://api.github.com/notifications"
  organization_repositories_url: "https://api.github.com/orgs/{org}/repos?type,page,per_page,sort"
  organization_url: "https://api.github.com/orgs/{org}"
  public_gists_url: "https://api.github.com/gists/public"
  rate_limit_url: "https://api.github.com/rate_limit"
  repository_search_url: "https://api.github.com/search/repositories?q={query}&page,per_page,sort,order"
  repository_url: "https://api.github.com/repos/{owner}/{repo}"
  starred_gists_url: "https://api.github.com/gists/starred"
  starred_url: "https://api.github.com/user/starred/{owner}/{repo}"
  team_url: "https://api.github.com/teams"
  user_organizations_url: "https://api.github.com/user/orgs"
  user_repositories_url: "https://api.github.com/users/{user}/repos?type,page,per_page,sort"
  user_search_url: "https://api.github.com/search/users?q={query}&page,per_page,sort,order"
  user_url: "https://api.github.com/users/{user}"
```

Encontramos que dentro del objeto `responseJSON` se encuentra la información relevante a la API.

Podríamos encontrar términos que no conozcamos y eso es bueno, así nos familiarizamos con Github. Pero de los que si reconocemos serían: usuarios, repositorios u organizaciones.

```
user_url: "https://api.github.com/users/{user}"

repository_url: "https://api.github.com/repos/{owner}/{repo}"

organization_url: "https://api.github.com/orgs/{org}"
```

Mostrar un usuario

Tomemos como ejemplo los usuarios. Consultemos al usuario 'defunkt' (provisto por la guía de Github): De acuerdo a la información anterior, la url sería así: <https://api.github.com/users/defunkt>

```
▼ responseJSON: Object
  avatar_url: "https://avatars.githubusercontent.com/u/2?v=3"
  bio: ""
  blog: "http://chriswanstrath.com/"
  company: "Github"
  created_at: "2007-10-20T05:24:19Z"
  email: "chris@github.com"
  events_url: "https://api.github.com/users/defunkt/events{/privacy}"
  followers: 15731
  followers_url: "https://api.github.com/users/defunkt/followers"
  following: 208
  following_url: "https://api.github.com/users/defunkt/following{/other_user}"
  gists_url: "https://api.github.com/users/defunkt/gists{/gist_id}"
  gravatar_id: ""
  hireable: true
  html_url: "https://github.com/defunkt"
  id: 2
  location: "San Francisco"
  login: "defunkt"
  name: "Chris Wanstrath"
  organizations_url: "https://api.github.com/users/defunkt/orgs"
  public_gists: 273
  public_repos: 107
  received_events_url: "https://api.github.com/users/defunkt/received_events"
  repos_url: "https://api.github.com/users/defunkt/repos"
  site_admin: true
  starred_url: "https://api.github.com/users/defunkt/starred{/owner}/{/repo}"
  subscriptions_url: "https://api.github.com/users/defunkt/subscriptions"
  type: "User"
  updated_at: "2016-09-13T19:41:49Z"
  url: "https://api.github.com/users/defunkt"
```

Obtener repositorios de un usuario

Ahora obtenemos más información interesante, por ejemplo, obtengamos los repositorios del usuario: la url se observa en el parámetro repos_url: <https://api.github.com/users/defunkt/repos>

```
▼ responseJSON: Array(30)
  ► 0: Object
  ► 1: Object
  ► 2: Object
  ► 3: Object
  ► 4: Object
  ► 5: Object
  ► 6: Object
  ► 7: Object
  ► 8: Object
  ► 9: Object
  ► 10: Object
  ► 11: Object
  ► 12: Object
  ► 13: Object
  ► 14: Object
  ► 15: Object
  ► 16: Object
  ► 17: Object
  ► 18: Object
  ► 19: Object
  ► 20: Object
  ► 21: Object
  ► 22: Object
  ► 23: Object
  ► 24: Object
  ► 25: Object
  ► 26: Object
  ► 27: Object
  ► 28: Object
  ► 29: Object
```

Obtenemos un arreglo de los repositorios.

Autenticación

Hasta este punto, podemos seguir explorando diferentes url y analizar la información que obtenemos. Sin embargo, ahora probemos algo básico para trabajar más con la API: Autenticación. La documentación oficial presenta diferentes métodos de autenticación, en este tema estudiaremos la forma básica. Un punto importante de la autenticación es que te aumenta el número posible de solicitudes por hora, de 60 a 5000.

Para autenticar, basta agregar el siguiente código como parámetro de la función ajax:

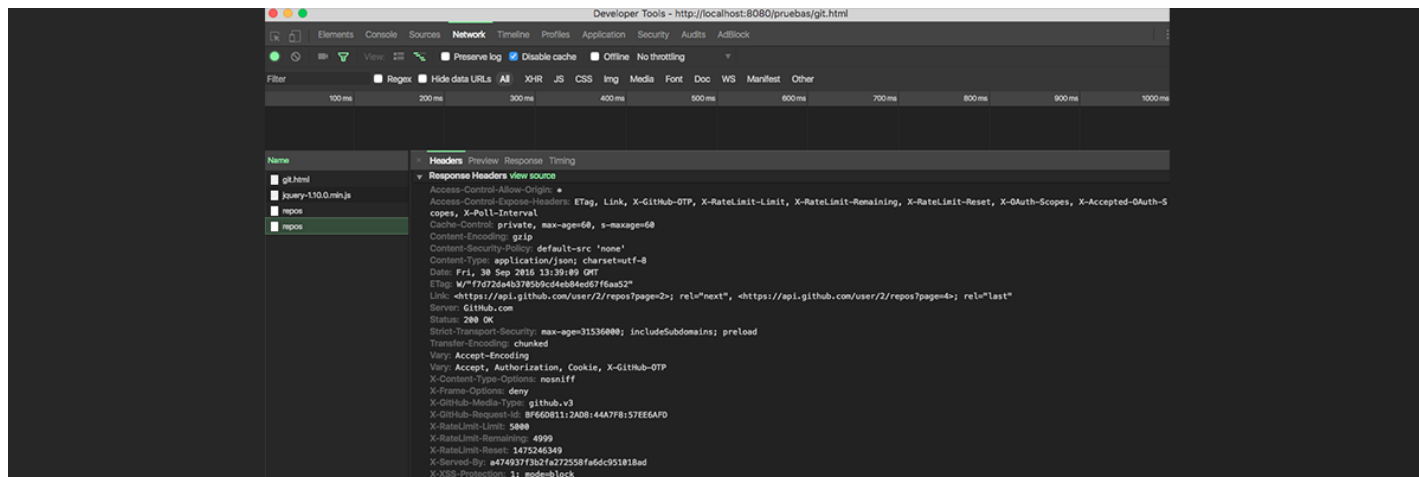
```
headers: {
  Authorization: 'Basic ' + btoa('<USER>:<PASSWORD>')
},
```

La función nativa de javascript 'btoa' convierte a base64 una cadena de caracteres.

El código completo sería así:

```
$.ajax({
  headers: {
    Authorization: 'Basic '+btoa('<USER>:<PASSWORD>')
  },
  url: "https://api.github.com",
  complete: function( data,ms,settings) {
    console.log(data,ms,settings)
  }
});
```

Recordemos que en la pestaña Network podemos obtener información importante de la respuesta.



Por ejemplo encontramos los parámetros:

- X-RateLimit-Limit:5000
- X-RateLimit-Remaining:4999
- Content-Type:application/json; charset=utf-8

Los parámetros que inician con X- son exclusivos de github. Vemos que X-RateLimit-Limit tiene asignado el valor 5000, lo que muestra que si nos autenticamos correctamente.

Crear un repositorio

Ahora que ya sabemos autenticarnos, podemos hacer operaciones más complejas, por ejemplo, creemos un repositorio.

Para crear un repositorio necesitamos hacer una operación extra. Por motivos de seguridad, es necesario otorgar permisos a un usuario para poder hacer determinadas operaciones utilizando la API, Github provee el uso de tokens para hacer estas operaciones, de tal modo que cada token puede o no hacerlas.

Primero debemos solicitar crear el token, por favor analiza el código:

```

var token_id,token;
$.ajax({
  url: 'https://api.github.com/authorizations',
  headers: {
    Authorization: 'Basic '+btoa('<USER>:<PASSWORD>')
  },
  type: 'POST',
  data: JSON.stringify({
    scopes:["repo"],
    note: 'Crear repositorio'
  }),
  complete: function(xhr,message,settings){
    if (xhr.status != 201) {
      console.log(xhr,message,settings);
      return;
    };
    console.log('Creado token');
    token = xhr.responseJSON.token;
    token_id = xhr.responseJSON.id;
    crearRepositorio();
  }
});

```

Lo primero que observamos es que la url usada es de autorizaciones, la primera en la lista cuando hicimos un llamado simple a la API. Seguidamente encontramos los headers de autenticación y luego encontramos el siguiente cambio. El parámetro type está definido como POST, esto se debe a que vamos a solicitar crear un token. Finalmente encontramos el parámetro data y es MUY IMPORTANTE que notes que el contenido de data es dentro de la función JSON.stringify, esto se debe a que la API solo funciona con JSON, por lo tanto debes hacer este procedimiento para que la API entienda las instrucciones. Dentro de data encontramos el parámetro scope, es un array que contiene los permisos que se van a otorgar, asignamos simplemente 'repo'. El parámetro note es opcional.

A continuación encontramos en la función complete una validación, recuerda que el status 201 significa que el recurso fue creado, por lo tanto si este no es el status, no podemos continuar.

Habrás notado que en la parte superior definimos unas variables globales token y token_id, ahora observa que sus valores son definidos dentro del complete del ajax, se usarán más adelante. Seguidamente encontramos la función crearRepositorio:

```

function crearRepositorio(){
  $.ajax({
    url: 'https://api.github.com/user/repos',
    type: 'POST',
    headers: {
      Authorization: 'token '+token
    },
    data: JSON.stringify({
      "name": "Hola-Mundo",
      "description": "Tu primer repositorio",
      "homepage": "https://github.com",
      "private": false,
      "has_issues": true,
      "has_wiki": true,
      "has_downloads": true
    }),
    complete: function(xhr,message,settings){
      if (xhr.status != 201) {
        console.log(xhr,message,settings);
        return;
      };
      console.log('Creado repositorio');
      borrarToken();
    }
  });
}

```

Primero, la URL, esta al encontramos en la documentación de Github: 


<https://developer.github.com/v3/repos/#create>. A continuación el type, nuevamente POST, porque vamos a crear un repositorio. Sin embargo, ahora la autenticación es diferente, es mediante token,

¿cuál token? El token que obtuvimos como respuesta en el llamado anterior. Seguidamente de la información, recordemos que debe estar como JSON y para eso se usa la función `JSON.stringify`. Finalmente, el `complete`, al igual que en el llamado ajax anterior, es necesario validar el status 201.

Como último paso luego de crear el repositorio y como ejercicio de este tema, borramos el token que creamos llamado la función `borrarToken`:

```
function borrarToken(){
$.ajax({
  url: 'https://api.github.com/authorizations/'+token_id,
  type: 'DELETE',
  headers: {
    Authorization: 'Basic '+btoa('<USER>:<PASSWORD>')
  },
  complete: function(xhr,message,settings){
    if (xhr.status !== 204) {
      console.log(xhr,message,settings);
      return;
    };
    console.log('Borrado token');
  }
});
}
```

Lo más notorio de este llamado ajax es la URL, observemos que cuenta con el `token_id` obtenido en respuesta de la primera función y el status de la función `complete` que es 204, que significa No Content.

Esperamos que con estos ejemplos te sientas libre de seguir explorando las demás operaciones que provee la API de Github. Nuevamente te recomendamos leer la guía que provee la página  <https://developer.github.com/guides/getting-started/> y nos vemos en el próximo tema. ¡Hasta pronto!
