Introducción

Este artículo describe cómo utilizar la API REST de GitHub con GitHub CLI, curl o JavaScript. Para obtener una guía de inicio rápido , consulte " Inicio rápido para la API REST de GitHub ".

Acerca de las solicitudes a la API REST

Esta sección describe los elementos que componen una solicitud de API:

- método HTTP
- Camino
- Encabezados
- Tipos de medios
- Autenticación
- Parámetros

Cada solicitud a la API REST incluye un método HTTP y una ruta. Dependiendo del punto final de la API REST, es posible que también necesite especificar encabezados de solicitud, información de autenticación, parámetros de consulta o parámetros del cuerpo.

La documentación de referencia de la API REST describe el método HTTP, la ruta y los parámetros para cada punto final. También muestra solicitudes y respuestas de ejemplo para cada punto final. Para obtener más información, consulte la documentación de referencia de REST .

método HTTP

El método HTTP de un punto final define el tipo de acción que realiza en un recurso determinado. Algunos métodos HTTP comunes son GET, POST, DELETE y PATCH. La documentación de referencia de la API REST proporciona el método HTTP para cada punto final.
Por ejemplo, el método HTTP para el punto final "Enumerar problemas del repositorio" es GET".
Siempre que sea posible, la API REST de GitHub se esfuerza por utilizar un método HTTP apropiado para cada acción.

- OBTENER: Se utiliza para recuperar recursos.
- POST: Se utiliza para crear recursos.
- PATCH: Se utiliza para actualizar las propiedades de los recursos.
- PUT: Se utiliza para reemplazar recursos o colecciones de recursos.
- ELIMINAR: Se utiliza para eliminar recursos.

Camino

Cada punto final tiene una ruta. La documentación de referencia de la API REST proporciona la ruta para cada punto final. Por ejemplo, la ruta para el punto final "Listar problemas del repositorio" es /repos/{owner}/{repo}/issues.
Las llaves {} en una ruta indican los parámetros de ruta que debe especificar. Los parámetros de ruta modifican la ruta del punto final y son obligatorios en su solicitud. Por ejemplo, los parámetros de ruta para el punto final "Listar problemas del repositorio" son {propietario} y {repo}. Para utilizar esta ruta en su solicitud de API, reemplace {repo} con el nombre del repositorio donde desea solicitar una lista de problemas y reemplace {owner} con el nombre de la cuenta propietaria del repositorio.

Encabezados

Los encabezados proporcionan información adicional sobre la solicitud y la respuesta deseada. A continuación se muestran algunos ejemplos de encabezados que puede usar en sus solicitudes a la API REST de GitHub. Para ver un ejemplo de una solicitud que utiliza encabezados, consulte " Realizar una solicitud ".

Acepto
La mayoría de los puntos finales de la API REST de GitHub especifican que debes pasar un encabezado Accept con un valor de application/ vnd.github+json . El valor del encabezado Aceptar es un tipo de medio. Para obtener más información sobre los tipos de medios, consulte " Tipos de medios ".
X-GitHub- Api -Versión
Debe utilizar este encabezado para especificar una versión de la API REST que utilizará en su solicitud. Para obtener más información, consulte " Versiones de API ".

Agente de usuario
Todas las solicitudes de API deben incluir un encabezado User-Agent válido. El encabezado User-Agent identifica el usuario o la aplicación que realiza la solicitud.
De forma predeterminada, GitHub CLI envía un encabezado User-Agent válido. Sin embargo, GitHub recomienda usar su nombre de usuario de GitHub, o el nombre de su aplicación, para el valor del encabezado User-Agent. Esto permite que GitHub se comunique con usted si hay problemas.
El siguiente es un ejemplo de User-Agent para una aplicación llamada Awesome- Octocat -App:
Agente de usuario: Awesome- Octocat -App
Se rechazarán las solicitudes sin encabezado User-Agent. Si proporciona un encabezado User-Agent no válido, recibirá una respuesta 403 Prohibida.

Tipos de medios

Puede especificar uno o más tipos de medios agregándolos al encabezado Aceptar de su solicitud. Para obtener más información sobre el encabezado Aceptar, consulte " Aceptar ". Los tipos de medios especifican el formato de los datos que desea consumir de la API. Los tipos de medios son específicos de los recursos, lo que les permite cambiar de forma independiente y admitir formatos que otros recursos no admiten. La documentación para cada punto final de la API REST de GitHub describirá los tipos de medios que admite. Para obtener más información, consulte la " Documentación de la API REST de GitHub ".

Los tipos de medios más comunes admitidos por la API REST de GitHub son application/ vnd.github+json y application/ json .
Hay tipos de medios personalizados que puede utilizar con algunos puntos finales. Por ejemplo, la API REST para administrar confirmaciones y solicitudes de extracción admite los tipos de medios diff, patch y sha . Algunos otros puntos finales utilizan los tipos de medios completo, sin formato, texto o html.
Todos los tipos de medios personalizados para GitHub tienen este aspecto: aplicación/ vnd.github.PARAM+json , donde PARAM es el nombre del tipo de medio. Por ejemplo, para especificar el tipo de medio sin formato, usaría application/ vnd.github.raw+json .
Para ver un ejemplo de una solicitud que utiliza tipos de medios, consulte " Realizar una solicitud ".

Autenticación

Muchos puntos finales requieren autenticación o devuelven información adicional si está autenticado. Además, puede realizar más solicitudes por hora cuando esté autenticado.

Aunque se puede acceder a algunos puntos finales de la API REST sin autenticación, la CLI de GitHub requiere que te autentiques antes de poder usar el subcomando api para realizar una solicitud de API. Utilice el subcomando auth login para autenticarse en GitHub. Para obtener más información, consulte " Realizar una solicitud ".

Parámetros

Muchos métodos API requieren o le permiten enviar información adicional en los parámetros de su solicitud. Hay algunos tipos diferentes de parámetros: parámetros de ruta, parámetros de cuerpo y parámetros de consulta.

Parámetros de ruta

Los parámetros de ruta modifican la ruta del punto final. Estos parámetros son obligatorios en su solicitud. Para obtener más información, consulte " Ruta ".

Parámetros del cuerpo

Los parámetros del cuerpo le permiten pasar datos adicionales a la API. Estos parámetros pueden ser opcionales u obligatorios, según el punto final. Por ejemplo, un parámetro de cuerpo puede permitirle especificar el título de un problema al crear un nuevo problema o

especificar ciertas configuraciones al habilitar o deshabilitar una función. La documentación para cada punto final de la API REST de GitHub describirá los parámetros del cuerpo que admite. Para obtener más información, consulte la " Documentación de la API REST de GitHub ".

Por ejemplo, el punto final "Crear un problema" requiere que especifique un título para el nuevo problema en su solicitud. También le permite especificar opcionalmente otra información, como texto para colocar en el cuerpo del problema, usuarios para asignar al nuevo problema o etiquetas para aplicar al nuevo problema. Para ver un ejemplo de una solicitud que utiliza parámetros del cuerpo, consulte " Realizar una solicitud ".

Debe autenticar su solicitud para pasar los parámetros del cuerpo. Para obtener más información, consulte " Autenticación ".

Parámetros de consulta

Los parámetros de consulta le permiten controlar qué datos se devuelven para una solicitud. Estos parámetros suelen ser opcionales. La documentación para cada punto final de la API REST de GitHub describirá cualquier parámetro de consulta que admita. Para obtener más información, consulte la " Documentación de la API REST de GitHub ".

Por ejemplo, el punto final "Listar eventos públicos" devuelve treinta números de forma predeterminada. Puede utilizar el parámetro de consulta per_page para devolver dos números en lugar de 30. Puede utilizar el parámetro de consulta de página para recuperar solo la primera página de resultados. Para ver un ejemplo de una solicitud que utiliza parámetros de consulta, consulte " Realizar una solicitud ".

hacer una solicitud

Esta sección demuestra cómo realizar una solicitud autenticada a la API REST de GitHub usando la CLI de GitHub.

1. Configuración

Instale GitHub CLI en macOS , Windows o Linux. Para obtener más información, consulte Instalación en el repositorio CLI de GitHub.

2. Autenticado

1. Autentíquese con GitHub ejecutando este comando desde su terminal.

   caparazón
   Señor inicio de sesión de autenticación
   Puede utilizar la opción --scopes para especificar qué ámbitos desea. Si desea autenticarse con un token que creó, puede usar la opción --with-token. Para obtener

más información, consulte la documentación de inicio de sesión de autenticación de GitHub CLI .

2. Siga las indicaciones en pantalla.

   GitHub CLI almacena automáticamente sus credenciales de Git cuando elige HTTPS como su protocolo preferido para las operaciones de Git y responde "sí" al mensaje que le pregunta si desea autenticarse en Git con sus credenciales de GitHub. Esto puede ser útil ya que le permite usar comandos de Git como git push y git pull sin necesidad de configurar un administrador de credenciales separado o usar SSH.

3. Elija un punto final para su solicitud

1. Elija un punto final al que realizar una solicitud. Puede explorar la documentación de la API REST de GitHub para descubrir puntos finales que puede utilizar para interactuar con GitHub.

2. Identifique el método HTTP y la ruta del punto final. Los enviará con su solicitud. Para obtener más información, consulte " Método HTTP " y " Ruta ".

   Por ejemplo, el punto final "Crear un problema" utiliza el método HTTP POST y la ruta /repos/{owner}/{repo}/issues.
3. Identifique los parámetros de ruta necesarios. Los parámetros de ruta requeridos aparecen entre llaves {} en la ruta del punto final. Reemplace cada marcador de posición de parámetro con el valor deseado. Para obtener más información, consulte " Ruta ".
   Por ejemplo, el punto final "Crear un problema" utiliza la ruta /repos/{owner}/{repo}/issues y los parámetros de ruta son {owner} y {repo}. Para utilizar esta ruta en su solicitud de API, reemplace {repo} con el nombre del repositorio donde desea crear una nueva incidencia y reemplace {owner} con el nombre de la cuenta propietaria del repositorio.

4. Realice una solicitud con GitHub CLI

Utilice el subcomando api de la CLI de GitHub para realizar su solicitud de API. Para obtener más información, consulte la documentación de la API de la CLI de GitHub .
En su solicitud, especifique las siguientes opciones y valores:

- **--método** seguido del método HTTP y la ruta del punto final. Para obtener más información, consulte " Método HTTP " y " Ruta ".

- **--encabezado** :

    - **Aceptar** : pasa el tipo de medio en un encabezado Aceptar. Para pasar varios tipos de medios en un encabezado Aceptar, separe los tipos de medios con una coma: Aceptar: aplicación/ vnd.github+ json,application /

vnd.github.diff . Para obtener más información, consulte " Aceptar " y " Tipos de medios ".
    - **X-GitHub- Api -Version** : pase la versión de API en un encabezado X-GitHub- Api -Version. Para obtener más información, consulte " X-GitHub- Api -Version ".
- **-f** o **-F** seguido de cualquier parámetro del cuerpo o parámetro de consulta en formato clave=valor. Utilice la opción -F para pasar un parámetro que sea un número, booleano o nulo. Utilice la opción -f para pasar parámetros de cadena. Algunos puntos finales utilizan parámetros de consulta que son matrices. Para enviar una matriz en la cadena de consulta, use el parámetro de consulta una vez por elemento de la matriz y agregue [] después del nombre del parámetro de consulta. Por ejemplo, para proporcionar una matriz de dos ID de repositorio, utilice -f repository_ids [ ]=REPOSITORY_A_ID -f repository_ids [ ]=REPOSITORY_B_ID.

  Si no necesita especificar ningún parámetro de cuerpo o parámetro de consulta en su solicitud, omita esta opción. Para obtener más información, consulte " Parámetros del cuerpo " y " Parámetros de consulta ". Para ver ejemplos, consulte " Solicitud de ejemplo utilizando parámetros de cuerpo " y " Solicitud de ejemplo utilizando parámetros de consulta ".

Solicitud de ejemplo

La siguiente solicitud de ejemplo utiliza el punto final "Obtener Octocat " para devolver el octocat como arte ASCII.

```
caparazón
Señor api --método GET / octocat \
--header 'Aceptar: aplicación/ vnd.github+json ' \
--encabezado "X-GitHub- Api -Versión: 2022-11-28"
```

Solicitud de ejemplo utilizando parámetros de consulta

El punto final "Listar eventos públicos" devuelve treinta números de forma predeterminada. El siguiente ejemplo utiliza el parámetro de consulta per_page para devolver dos números en lugar de 30, y el parámetro de consulta de página para recuperar solo la primera página de resultados.

```
caparazón
Señor api --método GET /eventos -F por_página =2 -F página=1
--header 'Aceptar: aplicación/ vnd.github+json ' \
```

Solicitud de ejemplo utilizando parámetros del cuerpo

El siguiente ejemplo utiliza el punto final "Crear un problema" para crear un nuevo problema en el repositorio octocat /Spoon-Knife. En la respuesta, busque el html_url de su problema y navegue hasta su problema en el navegador.

caparazón
Señor api --method POST /repos/ octocat /Spoon-Knife/issues \
--header "Aceptar: aplicación/ vnd.github+json " \
--header "X-GitHub- Api -Versión: 2022-11-28" \
-f title='Creado con la API REST' \
-f body='Este es un problema de prueba creado por la API REST' \

Usando la respuesta

Después de realizar una solicitud, la API devolverá el código de estado de la respuesta, los encabezados de la respuesta y, potencialmente, un cuerpo de la respuesta.

Acerca del código de respuesta y los encabezados

Cada solicitud devolverá un código de estado HTTP que indica el éxito de la respuesta. Para obtener más información sobre los códigos de respuesta, consulte la documentación del código de estado de respuesta HTTP de MDN .

Además, la respuesta incluirá encabezados que brindan más detalles sobre la respuesta. Los encabezados que comienzan con X- o x- están personalizados en GitHub. Por ejemplo, los encabezados x- ratelimit -remaining y x- ratelimit -reset le indican cuántas solicitudes puede realizar en un período de tiempo.
Para ver el código de estado y los encabezados, use la opción --include o --i cuando envíe su solicitud.
Por ejemplo, esta solicitud obtiene una lista de problemas en el repositorio octocat /Spoon-Knife:

Señor API
--header 'Aceptar: aplicación/ vnd.github+json ' \
--método GET /repos/ octocat /Spoon-Knife/issues \
-F por_página =2 --incluir
Y devuelve un código de respuesta y encabezados que se parecen a esto:

HTTP/2.0 200 correcto
Control-de-acceso-permitir-origen: *
Access-Control-Expose-Headers: ETag , Enlace, Ubicación, Reintentar después, X-RateLimit -Limit, X- RateLimit -Remaining, X- RateLimit -Used, X- RateLimit -Resource, X- RateLimit -Reset, X- OAuth-Scopes, X-Accepted-OAuth-Scopes, X-Poll-Interval, X-GitHub-Media-Type, X-GitHub-SSO, X-GitHub-Request-Id, obsolescencia, puesta en marcha
Control de caché: privado, edad máxima = 60, s- edad máxima = 60
Política-de-seguridad-de-contenido: predeterminado- src 'ninguno'
Tipo de contenido: aplicación/ json ; juego de caracteres = utf-8
Fecha: jueves, 4 de agosto de 2022 19:56:41 GMT
Etiqueta electrónica :
W/"a63dfbcfdb73621e9d2e89551edcf9856731ced534bd7f1e114a5da1f5f73418"

Enlace: <https://api.github.com/repositories/1300192/issues?per_page=1&page=2>; rel ="siguiente",
<https://api.github.com/repositories/1300192/issues?per_page=1&page=14817>; rel ="último"
Política de referencia: origen-cuando-origen-cruzado, origen-estricto-cuando-origen-cruzado
Servidor: GitHub.com
Estricta seguridad en el transporte: edad máxima = 31536000; incluirSubdominios ; precarga
Variar: Aceptar, Autorización, Cookie, Aceptar-Codificación, Aceptar, X-Solicitado-Con
Aceptado por X- Oauth -Scopes: repositorio
Opciones de tipo de contenido X: nosniff
Opciones de X-Frame: negar
X- Github - Api -Versión seleccionada: 2022-08-09
X- Github -Tipo de medio: github.v 3; formato = json
X- Github -Id. de solicitud: 1C73:26D4:E2E500:1EF78F4 :62EC2479
X- Oauth -Id. del cliente: 178c6fc778ccc68e1d6a
X- Oauth - Ámbitos: esencia, lectura: org , repositorio, flujo de trabajo
X- Límite de tasa - Límite: 15000
X- Límite de velocidad - Restante: 14996
X- Límite de velocidad - Restablecer: 1659645499
X- Límite de velocidad - Recurso: núcleo
X- Límite de tasa -Usado: 4
X- Xss -Protección: 0
En este ejemplo, el código de respuesta es 200, lo que indica una solicitud exitosa.

Acerca del cuerpo de respuesta

Muchos puntos finales devolverán un cuerpo de respuesta. A menos que se especifique lo contrario, el cuerpo de la respuesta está en formato JSON. Los campos en blanco se incluyen como nulos en lugar de omitirse. Todas las marcas de tiempo aparecen en hora UTC, formato ISO 8601: AAAA-MM-DDTHH: MM:SSZ .
A diferencia de la API GraphQL , donde usted especifica qué información desea, la API REST generalmente devuelve más información de la que necesita. Si lo desea, puede analizar la respuesta para extraer información específica.

Por ejemplo, puedes usar > para redirigir la respuesta a un archivo. En el siguiente ejemplo, reemplace REPO-OWNER con el nombre de la cuenta propietaria del repositorio y REPO-NAME con el nombre del repositorio.
caparazón
Señor API
--header 'Aceptar: aplicación/ vnd.github+json ' \
--método GET /repos/REPO-OWNER/REPO-NAME/issues \
-F por_página =2 > datos.json
Luego puedes usar jq para obtener el título y el ID del autor de cada número:

caparazón

jq '.[ ] | {título: .título, ID de autor : .user.id}' data.json
Los dos comandos anteriores devuelven algo como:

{
"title": "Actualizar index.html",
" ID de autor ": 10701255
}
{
"title": "Editar archivo de índice",
" ID de autor ": 53709285
}
Para obtener más información sobre jq , consulte la documentación de jq .

Representaciones detalladas versus resumidas

Una respuesta puede incluir todos los atributos de un recurso o solo un subconjunto de atributos, dependiendo de si recupera un recurso individual o una lista de recursos.

- Cuando recupera un *recurso individual* , como un repositorio específico, la respuesta normalmente incluirá todos los atributos de ese recurso. Esta es la representación "detallada" del recurso.
- Cuando recupera una *lista de recursos* , como una lista de múltiples repositorios, la respuesta solo incluirá un subconjunto de atributos para cada recurso. Esta es la representación "resumen" del recurso.

Tenga en cuenta que la autorización a veces influye en la cantidad de detalles incluidos en una representación.

La razón de esto es que algunos atributos son computacionalmente costosos para que los proporcione la API, por lo que GitHub excluye esos atributos de la representación resumida. Para obtener esos atributos, puede buscar la representación detallada.

La documentación proporciona una respuesta de ejemplo para cada método API. La respuesta de ejemplo ilustra todos los atributos que devuelve ese método.

hipermedia

Todos los recursos pueden tener una o más *_ propiedades de URL que enlacen a otros recursos. Están destinados a proporcionar URL explícitas para que los clientes API adecuados no necesiten construir URL por sí mismos. Se recomienda encarecidamente que los clientes API los utilicen. Hacerlo facilitará a los desarrolladores las futuras actualizaciones de la API. Se espera que todas las URL sean plantillas de URI RFC 6570 adecuadas .
Luego puedes expandir estas plantillas usando algo como la gema uri_template :

>> tmpl = URITemplate.new ('/notificaciones {? desde ,todos,participantes }')
>> tmpl.expandir

=> "/notificaciones"

>> tmpl.expandir todo: 1
=> "/ notificaciones?todos =1"

>> tmpl.expandir todo: 1, participando: 1
=> "/ notificaciones?todos =1&participando=1"

Versiones API
Obtenga más información sobre cómo especificar la versión de la API REST que se utilizará cada vez que se realice una solicitud de API REST.

Acerca del control de versiones desde la API

GitHub tiene versiones de la API REST. El número de versión de API se basa en la fecha en la que se publicó esa versión. Por ejemplo, la versión 2022-11-28 de la API se publicó el lunes 28 de noviembre de 2022.
Todos los cambios importantes se publicarán en una nueva versión de API. Los cambios importantes son cambios que pueden interrumpir una integración. Los cambios importante incluir :

- quitar uno operación llenar
- quitar un parámetro o cambiar su nombre
- quitar un campo de respuesta o cambiar su nombre
- agregar un nuevo parámetro obligatorio
- hacer que se requiera un parámetro opcional anterior
- cambiar el tipo de un parámetro o campo de respuesta
- quitar valores de enumeración
- agregar una nueva regla de validación a un parámetro existente
- cambiar los requisitos de autenticación o autorización

Todos los cambios aditivos (no importantes) estarán disponibles en todas las versiones de API compatibles. Los cambios aditivos son cambios que no deben interrumpir una integración. Los cambios aditivo incluir :

- agregación uno operación
- agregar un parámetro opcional
- agregar un encabezado de solicitud opcional
- agregar un campo de respuesta
- agregar un campo de respuesta
- agregación valores de enumeración

Cuando se publica una nueva versión de la API REST, la versión anterior de la API será compatible durante al menos 24 meses después del lanzamiento de la nueva versión de la API.

Especificación de una versión API

Debes usar el encabezado X-GitHub-Api- Version para especificar una versión de API. Por ejemplo:
curl --encabezado "X-GitHub-Api-Version:2022-11-28" https://api.github.com/zen
las solicitudes sin el encabezado X-GitHub-Api- Version se utilicen en la versión 2022-11-28 de forma predeterminada.
Si especifica una versión de API que ya no es compatible, recibirá un error 400.

Actualizar a una nueva versión de API

Antes de actualizar una nueva versión de la API REST, debe leer el registro de cambios de los cambios importantes que corresponden a la nueva versión de la API para comprender qué cambios importantes se incluyen y obtener más información sobre cómo actualizar esta versión específica de la API. Para obtener más información, consulte « Cambios importantes ».

Para actualizar la integración para especificar la nueva versión de API en el encabezado X-GitHub-Api -Version , también debe realizar los cambios necesarios para que la integración funcione con la nueva versión de API.
Una vez que vea la actualización de la integración, la prueba para verificar que funciona con la nueva versión de la API.

Biblioteca para la API REST
Puede utilizar las bibliotecas oficiales de Octokit y otras bibliotecas de terceros para ampliar y simplificar el uso de la API de GitHub.

Acerca de las bibliotecas

Las bibliotecas se pueden utilizar para ampliar y simplificar la forma en que la aplicación interactúa con la API de GitHub. Cada biblioteca proporciona código precompilado para un lenguaje de programación específico. Después de integrar una biblioteca en el proyecto, se pueden usar módulos de código precompilados para interactuar con la API de GitHub a través de un lenguaje de programación específico.

bibliotecas oficiales de Octokit para algunos idiomas. También existen bibliotecas de terceros que se pueden usar con la API de GitHub, que GitHub no mantiene.

Bibliotecas oficiales de GitHub

GitHub mantiene estas bibliotecas cliente oficiales para la API de GitHub. Estos repositorios son de código abierto y las contribuciones de la comunidad son bienvenidas.

Para obtener más información, consulte « Secuencias de comandos con la API REST y JavaScript » y « Secuencias de comandos con la API REST y Ruby ».

- JavaScript: octokit.js .
- Rubí: octokit.rb
- .NET: octokit.net
- Terraform: proveedor-terraform -github

Biblioteca de terceros

A continuación se muestran ejemplos de bibliotecas de terceros que se pueden utilizar para interactuar con la API de GitHub en varios lenguajes de programación.

Estos archivos de terceros no mantienen GitHub. Las bibliotecas de terceros se rigen por términos de servicio, directivas de privacidad y documentación externa.

Cierre

- Tentáculos: clj -commons/tentáculos

Dardo

- github.dart : SpinlockLabs / github.dart

Emacs Lisp

- gh.el : sigma/ gh.el

Ir

- ir- github : google/go -github

Haskell

- haskell-github : haskell-github / github

Java

- API de GitHub para Java, una representación orientada a objetos de la API de GitHub: org.kohsuke.github (de github -api)

- La API GitHub JCabi se basa en la API JSON para Java7 (JSR-353) y simplifica las pruebas con un código auxiliar de GitHub en el momento de la ejecución, e incluye la API completa: github.jcabi.com (web personal)

javascript

- NodeJS GitHub: pksunkara / octonode
- Contenedor Github.js en torno a la API de GitHub API: github-tools / github
- Biblioteca CoffeeScript basada en Promise para el navegador NodeJS : philschatz / github-client

rascar

- GitHub.jl : JuliaWeb / GitHub.jl

OCaml

- ocaml-github : espejismo/ ocaml-github

perla

- Pithub : más / Pithub
- Net::GitHub: fileland / perl -net- github

PHP

- API de PHP GitHub: KnpLabs / php-github-api
- GitHub Joomla! Paquete : joomla -framework/ github-api
- Puente de GitHub para Laravel : GrahamCampbell / Laravel -GitHub

PowerShell

- PowerShellForGitHub : microsoft / PowerShellForGitHub

Pitón

- gidgethub : gidgethub / gidgethub
- ghapi : rápido / ghapi
- PyGithub : PyGithub / PyGithub
- libsaas : tarima/ libsaas
- github3.py: sigmavirus24/github3.py
- agithub : mozilla / agithub

- github -flask: github -flask (sitio web oficial)
- githubkit : yanyongyu / githubkit
- octokit.py: khornberg/octokit.py

Rubí

- Gema API de GitHub: piotrmurach / github

Óxido

- Octocrab : XAMPPRocky / octocrab
- Octocat : octocat-rs / octocat-rs

Escala

- Github4s: 47 grados/github4s

caparazón

- ok.sh: blanqueamiento/ok.sh