Universidad de San Carlos de Guatemala

Matemática Aplicada 2

Proyecto #1

Renato Flores, 201709244

catedratico
Ing. José Saquimux

1 de octubre de 2020

Índice

Introducción

Las Google Cloud APIs son una gran parte de la plataforma cloud de Google. Con ayuda de estas APIs puedes facilmente agregar el poder de todos los servicios que provee google con tus aplicaciones, esto incluye tanto los servicios comunes utilizados por millones de usuarios al rededor del mundo como lo son Gmail, Calendar, Drive, Docs, etc. Como tambien los servicios menos famosos diseñados especificamente para su uso programatico como machine-learning-based APIs, Business Analytics, entre otros.
Las APIs de Google son gratis para su uso personal, y tambien tienen la opcion de integrarse con una empresa por medio del uso de la GSuite de dicha empresa.
Puedes accesar a estas APIs desde cualquier aplicacion que soporta las librerias cliente de Google. Entre ellas se encuentran:

- Android
- NodeJS
- Python
- Go
- Java
- PHP

APIs REST

Las APIs de Google siguen el modelo REST. Una API REST es un modo sencillo de acceder a servicios web sin excesivo procesado. Cuando se llama a una API RESTful, el servidor transferirá al cliente una representación del estado de recurso solicitado. En realidad, esto lo hacemos casi cada día. Si está buscando vídeos en YouTube, teclearás una palabra clave en su buscador, pulsarás Enter y obtendrás un listado de vídeos. Conceptualmente, así es como funciona una API REST. Estás buscando algo y obtienes una lista de resultados del servicio al que has lanzado la petición. Una API es un interfaz de programación de aplicaciones. Es un conjunto de reglas que permite a los programas comunicarse entre ellos. El desarrollador crea la API en el servidor y permite al cliente 'hablar' con ella. REST es lo que determina el 'aspecto' de la API. Son las reglas que siguen los desarrolladores cuando crean una API. Una de estas reglas determina que deberías ser capaz de obtener un 'trozo' de datos (un recurso), cuando invocas una URL determinada. Cada llamada a una URL se denomina una petición, mientras que los datos obtenidos se denomina repuesta.

2.1. Metodos de de una API REST

HTTP tiene cinco métodos que se usan normalmente en arquitecturas basadas en REST: POST, GET, PUT, PATCH y DELETE. En realidad se corresponden con crear, leer, actualizar y borrar, respectivamente. Conviene recordar que existen también otros métodos que se usan con menos frecuencia, como OPTIONS y HEAD.

2.1.1. GET

Este método se usa para recuperar (o leer) una representación de un recurso. Si todo va bien, GET devuelve una representación en XML o JSON y en código HTTP de respuesta 200 (OK). En caso de error, habitualmente devuelve un 404 (no encontrado) o 400 (petición incorrecta).

2.1.2. POST

Este método se usa a menudo para crear nuevos recursos. En particular, recursos subordinados, es decir subordinado a algún otro recurso (padre). Al crearse con éxito, devuelve un estado HTTP 201, devolviendo una cabecera con el enlace al recurso recién creado.

2.1.3. PUT

Se usa para actualizar y también crear un recurso (en el caso en el que el identificador del recurso lo elige el cliente en lugar del servidor). Esencialmente, PUT se lanza a una URL que contiene el valor de un recurso no existente. Una actualización con éxito devuelve un 200 (o 204 si no se devuelve ningún contenido). Si se usa PUT para crear, devuelve un HTTP 201 si se crea con éxito.

2.1.4. PATCH

Se usa para modificar capacidades. La petición PATCH sólo necesita los cambios del recurso, no el recurso completo. Es parecido a PUT, sin embargo el cuerpo contiene un conjunto de instrucciones que describen cómo un recurso que actualmente reside en el servidor debe ser modificado para generar una versión. Así pues, el cuerpo del PATCH no debería ser sólo la parte modificada del recurso, sino algún tipo de lenguaje como XML o JSON.

2.1.5. DELETE

Muy explicativo, se usa para borrar recursos identificados por una URL. Al borrar con éxito, devuelve un estado HTTP 200 (OK), junto con un cuerpo de respuesta.

2.2. Respuesta

Luego de realizar una llamada exitosa a alguna API de Google por medio de cualquiera de sus metodos, la mayoria de sus APIs retornan una respuesta JSON que generalmente sigue el siguiente esquema: Donde, la informacion que realmente nos interesa se encuentra en el objeto data. El status & status test, por lo general son siempre 200, .ºK". Ya que si en caso ocurriera un error o nuestra peticion fuera rechazada por Google, retorna un HTTP error code correspondiente y la estructura de la respuesta cambia totalmente.

2.3. Partes de una respuesta

Si bien se explico antes sobre la forma en que se recibe una respuesta estandar de Google, a continuacion se explica mas detalladamente, las partes de la respuesta.

2.3.1. Config

Contiene metadata sobre el tipo de solicitud, el servidor, la version y otra informacion sobre el tipo de solicitud realizada.

2.3.2. Data

La informacion que solicitamos.

2.3.3. Headers

Headers estandar de una respuesta HTTP.

2.3.4. Status

Codigo de status de la solicitud siempre sera 200 si esta es exitosa.

2.3.5. StatusText

Una pequeña descripcion del status de la respuesta, siempre sera .ºK"si esta es exitosa.

2.3.6. Request

Indica el resource path utilizado para realizar esta request.

Autenticacion

Si bien es cierto que las APIs de Google pueden ser accesadas desde basicamente cualquier dispositivo con una conexion a internet, su consumo en realidad no es tan simple. TODAS las apis de Google requieren de una autenticacion por parte del usuario para utilizarlas. Los procesos de autenticacion disponibles incluyen.

3.1. API Keys

Una API Key es la forma mas simple de autenticacion, ya que unicamente requieren que estas se incluyan como una QueryString en la request. Esto implica que cualquier dispositivo puede emplearlas, en teoria es posible utilizarlas directamente desde Arduino o cualquier dispositivo con conexion a internet.
Sin embargo, no son perfectas y tienen varias desventajas. A continuacion se listan ambas.

3.1.1. Ventajas

- Facil de generar
- Facil de utilizar
- Se pueden utilizar desde cualquier cliente

3.1.2. Desventajas

- Solo pueden accesar la informacion publica
- Solo pueden realizar operaciones de solo lectura
- No todas las APIs admiten autenticacion por API Keys.

3.2. OAuth2 personal authentication

El flujo general a seguir cuando se utiliza la verificacion personal por medio de OAuth2 tokens es:

1. Crear un proyecto de desarrollador de Google
2. Habilitar las APIs a utilizar por el proyecto
3. Crear un proyecto local en el lenguaje de eleccion.
4. Descargar e instalar las librerias para autenticacion con OAuth2
5. Descargar las credenciales de autenticacion de la consola de desarrolladores de Google
6. Utilizar la libreria de OAuth2 y las credenciales descargadas para comenzar el proceso de autenticacion
7. Aprobar el acceso de sus recursos a la aplicacion
8. Utilizar la API.

3.3. Service Accounts

La autenticacion por Service Accounts es muy similar a la autenticacion personal, sin embargo con este metodo en vez de utilizar credenciales personales para acceder directamente a los recursos de una cuenta, se emplean las credenciales de una cuenta especial creada especialmente para acceder a los recursos de un dominio entero comprado bajo una GSuite. Estas cuentas tienen direcciones de correo que pueden ser utilizadas para otorgar acceso a ciertos recursos como si se tratara de simples cuentas personales. Sin embargo, estas cuentas no tienen acceso a Gmail y no tienen contraseña, pues no estan dirigidas al uso por humanos. La idea es utilizarlas para autenticarse con Google a favor de un dominio entero y poder utilizar los recursos de dicho dominio. Estas cuentas se crean en la consola de desarrolladores de Google, y necesita de permisos elevados del administrador del dominio para ser creadas. Esta es la forma en la que las empresas integran los servicios de Google con sus aplicaciones o portales existentes. Un ejemplo de ello es nuestro dominio de la facultad de ingenieria.

Scopes

El Scope o Alcance en español, indica de forma especifica a que recurso (API) se tiene acceso y a que permitos (leer, editar) sobre los recursos de una cuenta se tiene acceso. Un Scope se indica como un URL y se incluye en el cuerpo de la autenticacion inicial con Google. Ejemplos de Scopes validos son:

- https://www.googleapis.com/auth/drive.readonly
- https://www.googleapis.com/auth/spreadsheets.readonly
- https://www.googleapis.com/auth/drive.file
- https://www.googleapis.com/auth/spreadsheets
- https://www.googleapis.com/auth/admin.directory.resource.calendar

Puede visitar el link: https://developers.google.com/identity/protocols/oauth2/scopes para una lista de todos los Scopes de todas las APIs disponibles de Google.

Google APIs Library

De manera general, Google provee una libreria demoniada "googleapis"que engloba todas las APIs disponibles por google. Estas deben ser instanciadas en el codigo de la aplicacion cliente y se deben crear objetos por cada API especifica que se desee utilizar. El flujo de trabajo general, es el siguiente:

a) Concectarse y Autenticarse

b) Crear el objeto cliente que representa la API

c) Identificar el recurso al que se desea acceder

d) Utilizar el metodo especifico

e) Enviar el body esperado de acuerdo al metodo

f) Enviar un callback

g) Procesar la informacion devuelta

(

Estructura de una Instancia de una Google API De manera general, un objeto instanciado de manera correcta que representa una Google API luego de haber sido autenticado se ve de la siguiente forma: La forma en que se agrupan los recuros y sus respectivas funciones puede encontrarse en la documentacion oficial de la API especifica de interes.
Es importante notar, que el Metodo no se refiere a un metodo REST estandar. La lista de metodos soportados por cada recurso puede encontrarse en la documentacion oficial.