



1º Encuentro Programadores HARBOUR
NOVELDA – 10/11/2017
por Cristóbal Navarro López (C)



APIs: REST

REFERENCIAS:

- <https://dosideas.com/noticias/java/314-introduccion-a-los-servicios-web-restful>
- <https://bbvaopen4u.com/es/actualidad/api-rest-que-es-y-cuales-son-sus-ventajas-en-el-desarrollo-de-proyectos>
- https://www.ibm.com/developerworks/webservices/library/ws-restful/index.html?ca=dgr-jw22RESTfulBasics&S_Tact=105AGX59&S_CMP=GRsitejw22
- https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional
- <https://www.adictosaltrabajo.com/tutoriales/spring-hateoas/>
- <https://www.genbetadev.com/desarrollo-aplicaciones-moviles/por-que-deberiamos-abandonar-rest-y-empezar-a-usar-graphql-en-nuestras-apis>



Indice:

1. Introducción

2. Presentando REST

3. Características de REST

- A. Protocolo cliente/servidor sin estado
- B. Operaciones más importantes disponibles
- C. Los objetos en REST siempre se manipulan a partir de la URI
- D. Interfaz uniforme
- E. Sistema de capas
- F. Uso de hypermedios



4. Ventajas que ofrece REST para el desarrollo (RESTful)

1. Separación entre el cliente y el servidor
2. Visibilidad, fiabilidad y escalabilidad
3. Es independiente del tipo de plataformas o lenguajes

5. Desarrollando un RESTful

1. Uso de los métodos HTTP
2. No mantiene estado
 - a. Responsabilidad del servidor
 - b. Responsabilidades del cliente de la aplicación
3. URIs con forma de árbol de directorios
4. Utilizar XML, JSON, o ambos

6. Conclusión



7. Es posible utilizar REST con Harbour?

- A. Crear un servidor REST con Harbour: HBHTTPD
- B. Consumir (cliente) un WS REST con Harbour: HBCURL

8. Clases for API Google

A. CLASS TGGoogle

- Métodos
- Datas

B. CLASS TGMail: Sample

C. Google API: Requerimientos Previos

- Panel de Control: Console Google
- Crear Credenciales

D. TIPOS DE APIS GOOGLE



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



- E. Documentación**
- F. Primeros Pasos**
- G. Authorize Request and Scope**



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Introducción:

En la página anterior he querido hacer referencia a algunas de las principales páginas web de las que he obtenido la información que contiene el presente documento y que me sirvieron inicialmente para introducirme en este apasionante tema.

Hay muchísima información tanto escrita como audiovisual en la web, por lo que cualquiera que esté interesado en ampliar y/o investigar este tema, no ha de tener ningún problema. El único problema es siempre el mismo: nos parece insuficiente lo que estudiamos, por si encontramos otro enfoque o algo novedoso (es a lo que estamos acostumbrados: no conformarnos con la visión que nos ofrece un determinado estudio del tema), y seguimos buscando y reuniendo información, lo que en algunos temas no nos ha ayudado a determinar el verdadero camino a seguir.

Este fue otro de los argumentos que me ayudaron a decidir a estudiar e investigar el REST: todos los documentos que leía, me llevaban a los mismos conceptos y definiciones.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Así que permitirme empezar por la conclusión, a modo de resumen, que deberíamos obtener al estudiar este tema:

De algún modo, REST es la vuelta a la Web antes de la aparición de los grandes servidores de aplicaciones, ya que hace énfasis en los primeros estándares de Internet, URI y HTTP.

XML sobre HTTP es una interfaz muy poderosa que permite que aplicaciones internas, como interfaces basadas en JavaScript Asíncrono + XML (AJAX) puedan conectarse, ubicar y consumir recursos. De hecho, es justamente esta gran combinación con AJAX lo que en mi opinión, produjo esta gran atención que tiene REST hoy en día.

Sus estrictas reglas en su desarrollo (para que sea el servicio desarrollado se defina como un verdadero RESTful) y utilización, me convencían cada vez más, a cada pequeño paso que daba, como un tema que ha de generar un profundo interés sin generar distintos puntos de vista acerca de su utilización.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



¿Es siempre la mejor opción?

REST ha aparecido como una alternativa para diseñar servicios web con menos dependencia en middleware propietario (software que asiste a una aplicación para interactuar o comunicarse con otras aplicaciones, o paquetes de programas, redes, hardware y/o sistemas operativos), por ejemplo, un servidor de aplicaciones, que utiliza SOAP y los servicios basados en WSDL, por lo que seguramente sigamos encontrando casos en los que esta opción siga siendo más adecuada.

¿Qué contiene este documento?

He intentado hacer un resumen de los conceptos básicos que hemos de conocer de este tema y poder relacionarlo con el lenguaje y las estructuras de programación que conocemos, entendiendo que no sólo podemos utilizar los lenguajes “standard” de uso en la web para su desarrollo e interacción.

(Al final veremos si he conseguido por lo menos vuestra aprobación, entendiendo que no soy ningún experto, sino un estudioso con muchas ganas de aprender).



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



El lanzamiento del sistema REST como protocolo de intercambio y manipulación de datos en los servicios de internet cambió por completo el desarrollo de software a partir del 2000, aunque en este momento no tuvo mucha atención. Ahora casi toda empresa o aplicación dispone de una API REST para su negocio.

REST cambió por completo la ingeniería de software. Este nuevo enfoque de desarrollo de proyectos y servicios web fue definido por **Roy Fielding**, el padre de la especificación HTTP y uno los referentes en todo lo relacionado con la Arquitectura de Redes, para usar la Web como una plataforma de Procesamiento Distribuido. En el campo de las APIs, REST (**Representational State Transfer - Transferencia del Estado Representacional**) es, a día de hoy, el corazón del desarrollo de servicios de aplicaciones.

En la actualidad no existe proyecto o aplicación que no disponga de una API REST para la creación de servicios profesionales a partir de ese software. Google, Twitter, YouTube, los sistemas de identificación con Facebook... hay cientos de empresas que generan negocio gracias a REST y las APIs REST. Sin ellas, todo el crecimiento en horizontal sería prácticamente imposible. Esto es así porque REST es el estándar más lógico, eficiente y habitual en la creación de APIs para servicios de Internet.



Presentando REST

Buscando una definición sencilla, REST es cualquier interfaz entre sistemas que use HTTP para obtener datos o generar operaciones sobre esos datos en todos los formatos posibles, como XML y JSON. Es una alternativa a otros protocolos estándar de intercambio de datos como SOAP (**Simple Object Access Protocol**), que disponen de una gran capacidad pero también mucha complejidad. A veces es preferible **una solución más sencilla de manipulación de datos como REST.**

REST define un set de principios arquitectónicos por los cuales se diseñan servicios web haciendo foco en los recursos del sistema, incluyendo cómo se accede al estado de dichos recursos y cómo se transfieren por HTTP hacia clientes escritos en diversos lenguajes. REST emergió en los últimos años como el modelo predominante para el diseño de servicios. De hecho, REST logró un impacto tan grande en la web que prácticamente logró desplazar a SOAP y las interfaces basadas en WSDL por tener un estilo bastante más simple de usar.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Importante diferenciar entre:

REST : es una arquitectura que se ejecuta sobre HTTP.

RESTful: hace referencia a un servicio web que implementa la arquitectura REST.

Un ejemplo bastante básico (de crear una aplicación RESTful), sería un proyecto donde implementamos un **CRUD** y con la arquitectura **MVC** (Modelo Vista-Controlador), el resultado de las respuestas serán devueltas en JSON, más no es obligatorio manejar sólo ese tipo **Content-type**, ya que se puede devolver HTML, texto, etc...

CRUD es el acrónimo de "Crear, Leer, Actualizar y Borrar" (del original en inglés: Create, Read, Update and Delete), que se usa para referirse a las funciones básicas en bases de datos o la capa de persistencia en un software.



Características de REST

- Protocolo cliente/servidor sin estado

Cada petición HTTP contiene toda la información necesaria para ejecutarla, lo que permite que ni cliente ni servidor necesiten recordar ningún estado previo para satisfacerla.

Aunque esto es así, algunas aplicaciones HTTP incorporan memoria caché, lo que no significa que sea lo más recomendable.

Se configura lo que se conoce como protocolo cliente-caché-servidor sin estado: existe la posibilidad de definir algunas respuestas a peticiones HTTP concretas como cacheables, con el objetivo de que el cliente pueda ejecutar en un futuro la misma respuesta para peticiones idénticas.



- **Operaciones más importantes disponibles**

Aunque no las únicas posibles, relacionadas con los datos en cualquier sistema

REST y la especificación HTTP son cuatro:

POST (crear)

GET (leer y consultar)

PUT (editar)

DELETE (eliminar)



- **Los objetos en REST siempre se manipulan a partir de la URI**

Es la URI y ningún otro elemento el identificador único de cada recurso de ese sistema REST.

La URI nos facilita acceder a la información, o para su modificación o borrado, o, por ejemplo, para compartir su ubicación exacta con terceros.

- **Interfaz uniforme**

Para la transferencia de datos en un sistema REST se aplican acciones concretas (POST, GET, PUT y DELETE) sobre los recursos, **siempre y cuando estén identificados con una URI.**

Esto facilita la existencia de una interfaz uniforme que sistematiza el proceso con la información.



- **Sistema de capas**

Arquitectura jerárquica entre los componentes. Cada una de estas capas lleva a cabo una funcionalidad dentro del sistema REST.

- **Uso de hipermedios**

Hipermedia es un término acuñado por Ted Nelson en 1965 y que es una extensión del concepto de hipertexto. Ese concepto llevado al desarrollo de páginas web es lo que permite que el usuario puede navegar por el conjunto de objetos a través de enlaces HTML.

En el caso de una API REST, el concepto de hipermedia **explica la capacidad de una interfaz de desarrollo de aplicaciones de proporcionar al cliente y al usuario los enlaces adecuados para ejecutar acciones concretas sobre los datos**



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Para cualquier API REST es obligatorio disponer del principio **HATEOAS**

Es decir, Hypermedia As The Engine Of Application State - Hipermedia como Motor del Estado de la Aplicación) **para ser una verdadera API REST.**

Este principio es el que define que cada vez que se hace una petición al servidor y éste devuelve una respuesta, parte de la información que **contendrá serán los hipervínculos de navegación asociada a otros recursos del cliente**

Veamos un ejemplo de la devolución de una petición a una API REST según el principio **HATEOAS** (en la información que devuelve, contiene enlaces (vínculos) a los recursos disponibles de los datos devueltos:



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Información y tutorial: <https://www.miscochesvendidos.com/>

```
{
  "id": 78,
  "nombre": "Juan",
  "apellido": "García",
  "coches": [
    {
      "coche":
"http://miservidor/concesionario/api/v1/clientes/78/coches/1033"
    },
    {
      "coche":
"http://miservidor/concesionario/api/v1/clientes/78/coches/3889"
    }
  ]
}
```



Ventajas que ofrece REST para el desarrollo

1. Separación entre el cliente y el servidor

El protocolo REST separa totalmente la interfaz de usuario, del servidor y del almacenamiento de datos (eso explica la facilidad y casi obligatoriedad de aplicar MVC al aplicar REST). Esto tiene algunas ventajas cuando se hacen desarrollos. Por ejemplo, mejora la portabilidad de la interfaz a otro tipo de plataformas, aumenta la escalabilidad de los proyectos y permite que los distintos componentes de los desarrollos puedan evolucionar de forma independiente.



2. Visibilidad, fiabilidad y escalabilidad

La separación entre cliente y servidor tiene una ventaja evidente y es que cualquier equipo de desarrollo puede escalar el producto sin excesivos problemas.

Se puede migrar a otros servidores o realizar todo tipo de cambios en la base de datos, siempre y cuando los datos de cada una de las peticiones se envíen de forma correcta.

Esta separación facilita tener en servidores distintos el front y el back y eso convierte a las aplicaciones en productos más flexibles a la hora de trabajar.



3. Es independiente del tipo de plataformas o lenguajes

La API REST siempre se adapta al tipo de sintaxis o plataformas con las que se estén trabajando, **lo que ofrece una gran libertad a la hora de cambiar o probar nuevos entornos dentro del desarrollo.**

Con una API REST se pueden tener servidores PHP, Java, Python o Node.js. Lo único que es indispensable es que las respuestas a las peticiones se hagan siempre en el lenguaje de intercambio de información usado, normalmente XML o JSON.



Desarrollando un RESTful

Hemos de tener en cuenta los principales principios básicos fundamentales que ha de contemplar en su desarrollo:

- Utilizar los métodos HTTP explícitamente.
- No mantener estado.
- Utilizar URIs con forma de carpetas y directorios.
- Utilizar en la transferencia de información: XML, JavaScript Object Notation (JSON), o ambos.

Intentemos explicar y desarrollar cada uno de estos principios en su forma adecuada de utilización.



1. Uso de los métodos HTTP

- se usa **POST** para crear un recurso en el servidor
- se usa **GET** para obtener un recurso
- se usa **PUT** para cambiar el estado de un recurso o actualizarlo
- se usa **DELETE** para eliminar un recurso

Un fallo de diseño que tienen algunas APIs web es el uso de métodos HTTP para otros propósitos:

Por ejemplo, la petición del URI en una solicitud HTTP GET, que, en general identifica a un recurso específico.

O la cadena de consulta en el URI incluye un conjunto de parámetros que definen el criterio de búsqueda que usará el servidor para encontrar un conjunto de recursos.

O usar el método HTTP GET para ejecutar algo transaccional en el servidor, por ejemplo, agregar registros a una base de datos. En estos casos, no se utiliza adecuadamente el URI de la petición HTTP, o al menos no se usa como REST define.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Si el API utiliza GET para invocar un procedimiento remoto, seguramente se verá algo como esto:

GET /addusuario?nombre=Juan HTTP/1.1

Si esta operación se procesa, producirá un cambio que será procesado por el servidor: añadir un usuario a la base de datos.

En cambio, esta definición, si cumple estrictamente los conceptos en los que se basa REST:

POST /usuarios HTTP/1.1

Host: miservidor

Content-type: application/xml

<usuario>

 <nombre>Juan</nombre>

</usuario>



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Más allá de la sintaxis, el problema de la utilización del GET en este caso, es que al ejecutar eliminaciones, modificaciones o creación de registros en la base de datos, o al cambiar el estado de los recursos de cualquier manera, provoca que las herramientas de caché web y los motores de búsqueda (crawlers) puedan realizar cambios de forma no intencionada en el servidor. Una forma simple de evitar este problema es mover los nombres y valores de los parámetros en la petición del URI a tags XML. Los tags resultantes, una representación en XML de la entidad a crear, pueden ser enviados en el cuerpo de un HTTP POST cuyo URI de petición es el nombre de la entidad.

Sin embargo, utilizar esta expresión, para obtener datos de la representación de un recurso (ya que ahora sabemos que se ubican en **/usuarios**), sería correcta:

GET **/usuarios/Juan** HTTP/1.1

Host: miservidor

Accept: application/xml

El uso del GET ha sido explícito, ya que se usa únicamente para recuperar datos.

GET es una operación que no debe tener efectos secundarios.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Veamos otro ejemplo al realizar una actualización de un registro a través del método HTTP GET.

El siguiente método GET intenta cambiar el "nombre" de un recurso. Si bien se puede usar la cadena de consulta para esta operación, evidentemente no es el uso apropiado y tiende a ser problemático en operaciones más complejas.

GET /updateusuario?nombre=Juan&NewNombre=Jose HTTP/1.1

Correcto según REST

PUT /usuarios/Juan HTTP/1.1

Host: miservidor

Content-Type: application/xml

```
<usuario>  
  <nombre>Jose</nombre>  
</usuario>
```



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



En un servicio web REST, las peticiones que se realicen al recurso que utilicen la URI anterior (**/usuarios/Juan**) van a generar un error estándar "404 Not Found".

La petición PUT que se ha mostrado es explícita, ya que apunta al recurso a ser actualizado identificándolo en el URI, y transfiere una nueva representación del recurso del cliente hacia el servidor en el cuerpo de la petición PUT, en vez de transferir los atributos del recurso como un conjunto de parámetros (nombre = valor) en el mismo URI de la petición.

Es ideal que, para mantener una interfaz general, y para que los clientes puedan ser explícitos en las operaciones que puedan realizar, los servicios web no deberían definir más métodos o procedimientos remotos (**en mi corta experiencia esto no me ha sido posible: LIST, etc**).

También como norma general, sólo debemos utilizar sustantivos en la definición de nuevos métodos y no verbos.



2. No mantiene estado

Los servicios web REST necesitan ser escalables para poder satisfacer una demanda en constante crecimiento. Se usan clusters de servidores con balanceadores de carga y alta disponibilidad, proxies, y gateways de manera de conformar una topología adecuada, que permita transferir peticiones de un equipo a otro para disminuir el tiempo total de respuesta de una invocación al servicio web.

El posible uso de servidores intermedios para mejorar la escalabilidad hace necesario que los clientes de **servicios web REST envíen peticiones completas e independientes**, es decir, se deben enviar peticiones que **incluyan todos los datos necesarios para cumplir la solicitud**, de manera que los componentes en los servidores intermedios puedan redireccionar y gestionar la carga sin mantener el estado localmente entre las peticiones.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Una solicitud completa e independiente hace que el servidor no tenga que recuperar ninguna información de contexto o estado al procesar la petición.

Una aplicación o cliente de servicio web REST debe incluir dentro del encabezado y del cuerpo HTTP de la petición todos los parámetros, contexto y datos que necesita el servidor para generar la respuesta. De esta manera, el no mantener estado mejora el rendimiento de los servicios web y simplifica el diseño e implementación de los componentes del servidor, ya que la ausencia de estado en el servidor elimina la necesidad de sincronizar los datos de la sesión con una aplicación externa.

Por ejemplo en una petición de página siguiente en un resultado multi-página, el servidor ha de guardar en algún lugar la página actual que ha solicitado el cliente. En un diseño con estado, el servidor ha de almacenar, e incrementa o disminuye el contador “pagina” que servirá de referencia a las siguientes peticiones. Hay muchos ejemplos de lo complicados que pueden resultar mantener los servicios web con estado, tanto en su análisis como en su desarrollo y mantenimiento ante posibles fallos.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Los servicios sin estado son mucho más simples de diseñar, escribir y distribuir a través de múltiples servidores. Un servicio sin estado no sólo funciona mejor, sino que además **mueve la responsabilidad de mantener el estado al cliente de la aplicación.**

En un servicio web REST, el servidor es responsable de generar las respuestas y proveer una interfaz que le permita al cliente mantener el estado de la aplicación por su cuenta. Por ejemplo, en el mismo ejemplo de una petición de datos en múltiples páginas, el cliente debería incluir el número de página a recuperar en vez de pedir "la siguiente" o "la anterior" al servidor.

Un servicio web sin estado genera una respuesta que se enlaza a la siguiente página del conjunto y le permite al cliente hacer todo lo que necesita para almacenar la página actual, de ser necesario.

Llegado este punto, intentemos aclarar de quién depende y cuáles son las responsabilidades tanto del servidor como del cliente.



Responsabilidad del servidor

- Generar respuestas que incluyen enlaces a otros recursos para permitirle a la aplicación navegar entre los recursos relacionados. Este tipo de respuestas tiene enlaces embebidos.
- Generar respuestas que indican si son susceptibles de caché o no, para mejorar el rendimiento al reducir la cantidad de peticiones para recursos duplicados, y para lograr eliminar algunas peticiones completamente. El servidor utiliza los atributos Cache-Control y Last-Modified de la cabecera en la respuesta HTTP para indicarlo.



Responsabilidades del cliente de la aplicación

- Utiliza el atributo Cache-Control del encabezado de la respuesta para determinar si debe cachear el recurso (es decir, hacer una copia local del mismo) o no. El cliente también lee el atributo Last-Modified y envía la fecha en el atributo If-Modified-Since del encabezado para preguntarle al servidor si el recurso cambió desde entonces.

Esto se conoce como GET Condicional, y ambos encabezados van de la mano con la respuesta del servidor 304 (No Modificado) y se omite el recurso que se había solicitado si no hubo cambios desde esa fecha. Una respuesta HTTP 304 significa que el cliente puede seguir usando la copia local de manera segura, evitando así realizar las peticiones GET hasta tanto el recurso no cambie.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



- Enviar peticiones completas que pueden ser ejecutadas en forma independiente a otras peticiones. Esto implica que el cliente hace uso completo de los encabezados HTTP tal como está especificado por la interfaz del servicio web, y envía las representaciones del recurso en el cuerpo de la petición. El cliente envía peticiones que hacen muy pocas presunciones sobre las peticiones anteriores, la existencia de una sesión en el servidor, la capacidad del servidor para agregarle contexto a una petición, o sobre el estado de la aplicación que se mantiene entre las peticiones.

Esta colaboración entre el cliente y el servicio es esencial para crear un servicio web REST sin estado.

Mejora el rendimiento, ya que ahorra ancho de banda y minimiza el estado de la aplicación en el servidor.



3. URIs con forma de árbol de directorios

Para el cliente, la URI determina qué tan intuitivo va a ser el web service REST, y si el servicio va a ser utilizado tal como fue pensado al momento de diseñarlo.

Las URI de los servicios web REST deben ser intuitivas, hasta el punto de que sea fácil adivinarlas. Pensemos en las URI como una interfaz auto-documentada que necesita de muy poca o ninguna explicación o referencia para que un desarrollador pueda comprender a lo que apunta, y a los recursos derivados relacionados.

Una forma de lograr este nivel de usabilidad es definir URIs con una estructura al estilo del árbol de directorios. Este tipo de URIs es jerárquica, con una única ruta raíz, y va abriendo ramas a través de las subrutas para exponer las áreas principales del servicio.

<http://www.harbourmagazine.org/encuentro1/temas/{tema}>



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Tomemos algunos recursos organizados por fecha, que son muy prácticos de organizar usando una sintaxis jerárquica.

El siguiente ejemplo es intuitivo porque está basado en reglas:

<http://www.harbourmagazine.org/encuentro1/{año}/{mes}/{dia}/{tema}>

Estos ejemplos nos permiten indicar algunos detalles a tener en cuenta en su definición:

- Ocultar la tecnología usada en el servidor que aparecería como extensión de archivos (.jsp, .php, .asp), de manera de poder portar la solución a otra tecnología sin cambiar las URI.
- Mantener todo en minúsculas.
- Sustituir los espacios con guiones o guiones bajos (uno u otro).
- Evitar el uso de cadenas de consulta.
- No usar un 404 Not Found, si la petición es una URI parcial, devolver una página o un recurso predeterminado como respuesta.



4. Utiliza en las peticiones y respuestas XML, JSON, o ambos

La representación de un recurso en general, refleja el estado actual del mismo y sus atributos al momento en que el cliente de la aplicación realiza la petición.

La representación del recurso son simples "fotos" en el tiempo. Esto podría ser una representación de un registro de la base de datos que consiste en la asociación entre columnas y tags XML, donde los valores de los elementos en el XML contienen los valores de las filas.

```
<encuentro1 fecha="{fecha}" tema="{tema}">
  <respuestas>
    <respuesta de="joseluis" href="/encuentro1/temas/{tema}/joseluis"/>
    <respuesta de="rafa" href="/encuentro1/temas/{tema}/rafa"/>
  </respuestas>
```



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Es aconsejable construir los servicios de manera que usen el atributo **HTTP Accept** del encabezado, en donde el valor de este campo es tipo **MIME**. De esta manera, los clientes pueden pedir por un contenido en particular que mejor pueden analizar. Algunos de los tipos MIME más usados para los servicios web REST son:

MIME-Type	Content-Type
JSON	application/json
XML	application/xml
XHTML	application/xhtml+xml

Esto permite que el servicio sea utilizado por distintos clientes escritos en diferentes lenguajes, corriendo en diversas plataformas y dispositivos. El uso de los tipos MIME y del encabezado HTTP Accept es un mecanismo conocido como **negociación de contenido**, el cual le permite a los clientes elegir qué formato de datos puedan leer, y minimiza el acoplamiento de datos entre el servicio y las aplicaciones que lo consumen.



Conclusión

Resulta muy flexible el poder exponer los recursos del sistema con un API REST, de manera que podemos ofrecer los datos a distintas aplicaciones, formateados en distintas maneras. REST ayuda a cumplir con los requerimientos de integración que son críticos para construir sistemas en donde los datos tienen que poder combinarse fácilmente y extenderse. Desde este punto de vista, los servicios REST se convierten en algo mucho más grande.

Una vez desarrollados y entendidos los fundamentos de una REST y la forma de definir y utilizar desde el punto de vista del cliente un RESTful, mi interés se centró, antes de intentar crear mi propio servicio web basado en REST, en conocer su uso como cliente, para ver sus pros y contras y entender bien en la práctica su funcionamiento. Es por lo que me interesé en el estudio de las APIs de Google, ya que además me permitían utilizar algunos productos que me serían muy prácticos como Vision, y a las que me referiré en este documento.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Pero, no todo son ventajas en el uso de REST. Según me he ido adentrando en el uso de diversas API REST, me he encontrado con un problema, que quiero entender, pueden producirse también por la forma en la que estos APIs son diseñados.

En la mayoría de los casos, cuando hacemos una request, no recibimos simplemente la información que necesitamos, sino todo el conjunto de datos relativos al resource alojado en esa URI. Algo complicado de gestionar y costoso en recursos cuando el 90% de los datos procedentes de cada resource/endpoint son innecesarios. Por ejemplo, si solo queremos obtener el nombre de una persona al hacer la consulta podemos recibir datos como, por ejemplo, su fecha nacimiento o su profesión. Datos innecesarios que desde el lado cliente no podemos controlar y entorpecen la obtención de información necesaria.

También, En **RESTful utilizamos una URI para leer o escribir un único recurso**, ya sea, por ejemplo, un producto, una persona, un post o un comentario. Si necesitamos trabajar con múltiples recursos como un listado de posts con un listado de comentarios, necesitamos manejar múltiples **endpoint** y encadenar distintas llamadas, a veces de forma secuencial. Es en las apps donde recae la responsabilidad de unir y mezclar todas las informaciones de cada recurso.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



¿Cuántas veces os habéis encontrado con la tediosa tarea de “pintar” una pantalla con la información procedente de 4 o 5 endpoint diferente? Por ejemplo, solicitar los ids de los comentarios de un post concreto para luego pintarlos en la misma pantalla junto a él. Aquí por ejemplo, necesitamos un request al detalle del post, a las estadísticas del post, al listado de comentarios, etc... y, después, juntar toda esa información. Como ejemplo, solicitar los mails de gmail, y lo que te devuelve es el id, entre otros datos. A partir de ahí, si deseas un listado del contenido de cada mail, has de realizar otro request (uno por cada id), para formalizar el informe.

REST es un claro caso de éxito, pero su concepción CRUD basada en resources, verbos y códigos de respuesta HTTP denotan sus limitaciones y su inflexibilidad. Otro punto débil con el que me he encontrado es el **versionado de una API REST, que no es una tarea trivial a la hora de ser diseñado el web service**, aunque no tan complicado a la hora de ser consumido.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



GraphQL es una de las alternativas que han surgido para solucionar la mayor parte de estos problemas descritos con API REST.

El primer borrador del RFC (aún en desarrollo) que especifica GraphQL fue creado por Facebook. Tal como ellos mismo explican llevan usándolo y perfeccionándolo **desde 2012** en sus apps móviles. Y no son los únicos, también **Github, Pinterest o Shopify** se han unido al carro. Todos los interesados en implementar GraphQL pueden colaborar en su definición, es Open Source.

Su gran ventaja: Con REST, no podemos elegir los datos que queremos recibir en el JSON/Payload de respuestas, en cambio, en GraphQL podemos elegir exactamente lo que necesitamos.

Pero, creo que con un buen análisis previo en el diseño y desarrollo de nuestro RESTful, podemos aliviar de forma muy importante estos problemas que menciono.



Es posible utilizar REST con Harbour?

Por supuesto que SI, ya que como sabemos (o no, ya que yo particularmente cada vez que le dedico un poco de tiempo a ampliar mis conocimientos sobre Harbour, descubro una nueva posibilidad de aplicación), pocos temas no pueden ser abordados con nuestra joya, directamente, o gracias a la multitud de contribs de las que disponemos.

Hay que diferenciar, tal y como he intentado explicar anteriormente, por un lado, la utilización tomando como punto de vista del servidor y por otra parte, acometiendo su utilización desde el punto de vista del cliente.

- **Crear un servidor REST con Harbour:**

Tenemos un muy buen ejemplo en las contribs\hbhttpd, con un ejemplo práctico de ver un RESTful funcionando.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



- Consumir (como cliente) un WS REST con Harbour:

1. Se puede utilizar CURL (contrib/hbcurl), junto a SSL de ser necesario.
2. Utilizar ActiveX.

Aunque no es el objetivo de este documento, no quiero dejar de comentar mi profunda admiración, tanto por la variedad de resultados que podemos obtener, como por el código utilizado en su desarrollo, el contenido de la contrib/hbhttpd para el desarrollo de un servidor web con Harbour.

He realizado desarrollos con esta herramienta, y he visto la multitud de posibilidades que tiene su aplicación, tanto en el tema que nos ocupa como en otros temas más sencillos, pero no menos prácticos.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Sin embargo, sí deseo comentar que hay muchas modificaciones sobre el mismo por parte de otros compañeros de Harbour, por ejemplo, para permitir utilizar **CORS** sin bibliotecas adicionales o herramientas (aunque pocos han sido publicados).

* CORS: https://en.wikipedia.org/wiki/Cross-origin_resource_sharing

Y un magnífico ejemplo de desarrollo en:

<https://github.com/tfonrouge/harbour-vszakats/blob/master/contrib/hbhttpd/examples/angularHeroesHTTP.prg>

/*

+ Simple HTTP Server as REST service provider for the AngularJS tutorial: <https://angular.io/tutorial>

- * define supported methods

- * implements required headers to allow CORS (<https://enable-cors.org>)

- * uses simple fast plain dbf's

- * can be easily configured to use another Harbour available database(?):

 - * MongoDB : <https://github.com/tfonrouge/hbmongoc>

 - * SQL : available on the Harbour main stream

(C) 2017 tfonrouge@gmail.com

*/

* **MongoDB es una base de datos Open Source NoSQL orientada a documentos tipo JSON**



Google API: Requerimientos Previos:

- Ir a la página web:
console.developers.google.com
- Crear un proyecto
- Activar APIs (Añadir todas las APIs que puedes necesitar)
- Ir a “CREDENCIALES”, en el panel izquierdo y pulsa en “Crear Credenciales” button
- Ir a IDs de cliente de OAuth 2.0
Descargar el fichero client.json



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Panel Control Console

Panel de control - Cristóbal

Es seguro | <https://console.developers.google.com/apis/dashboard?project=cristobal01-154518&duration=PT1H>

Aplicaciones Scintilla Fivewin Traductor de Google Log in — Bitbucket Personal Acceso Cliente Qt Windows 10 Facebook Gestión de Partes Temp Canales Google Drive

Google APIs Cristóbal01

API Administrador de ...

Panel de control [+ HABILITAR API](#)

Panel de control

Biblioteca

Credenciales

API	Solicitudes	Errores	Proporción de errores	Latencia, mediana	Latencia (98%)	
Admin SDK	—	—	—	—	—	Inhabilitar
Analytics API	—	—	—	—	—	Inhabilitar
Contacts API	—	—	—	—	—	Inhabilitar
Gmail API	—	—	—	—	—	Inhabilitar
Google Calendar API	—	—	—	—	—	Inhabilitar
Google Cloud Translation API	—	—	—	—	—	Inhabilitar
Google Drive API	—	—	—	—	—	Inhabilitar ⚙
Google Places API Web Service Private API	—	—	—	—	—	Inhabilitar
Google Sheets API	—	—	—	—	—	Inhabilitar
Groups Migration API	—	—	—	—	—	Inhabilitar
Groups Settings API	—	—	—	—	—	Inhabilitar
Tasks API	—	—	—	—	—	Inhabilitar



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Crear Credenciales

The screenshot shows the Google API Console interface. The left sidebar has a menu with 'API Administrador de ...', 'Panel de control', 'Biblioteca', and 'Credenciales' (selected). The main content area is titled 'Credenciales' and includes tabs for 'Credenciales', 'Pantalla de autorización de OAuth', and 'Verificación de dominio'. There are buttons for 'Crear credenciales' and 'Eliminar'. Below this, a section titled 'Claves de API' contains a table with one entry: 'Clave de API' with a creation date of '7 ene. 2017' and a restriction of 'Ninguna'. The 'ID de cliente de OAuth 2.0' section contains a table with one entry: 'cristobal01-' with a creation date of '3 ene. 2017' and a type of 'Otro'. At the bottom, there is a section for 'Claves de cuenta de servicio' and a link to 'Administra las cuentas de servicio'.

Nombre	Fecha de creación	Restricción	Clave
Clave de API	7 ene. 2017	Ninguna	[Redacted]

Nombre	Fecha de creación	Tipo	ID de cliente
cristobal01-	3 ene. 2017	Otro	[Redacted]



TIPOS DE APIS GOOGLE

- Con Autenticación REST API (Free use)
- Con Autenticación REST API CLOUD (requiere Registro en CLOUD Productos)
- No Autenticación (APIS Free y uso limitado)



Documentación

Google pone a nuestra disposición una gran cantidad de información acerca de cada producto y las APIs disponibles para su uso.

La mayoría de APIs tienen ejemplos en distintos lenguajes de su uso REST API

Por ejemplo en:

<https://developers.google.com/drive/v3/web/about-sdk>

O buscar en vuestro navegador “google drive REST API”



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Traductor de Google x Google Drive REST API x

Es seguro | <https://developers.google.com/drive/v3/web/about-sdk>

Aplicaciones Scintilla Fivewin Traductor de Google Log in — Bitbucket Personal Acceso Cliente Qt Windows 10 Facebook Gestión de Partes Temp Canales Google Drive

Google Drive APIs > REST

Buscar TODOS LOS PRODUCTOS

GUIDES REFERENCIA MUESTRAS ASISTENCIA SWITCH TO V2 ENVIAR COMENTARIOS

Google Drive REST API Overview

☆☆☆☆☆

The Drive platform gives you a group of APIs along with client libraries, language-specific examples, and documentation to help you develop apps that integrate with Drive.

The core functionality of Drive apps is to [download](#) and [upload](#) files in Google Drive. However, the Drive platform provides a lot more than just storage. This page describes some of that functionality and points you to resources for building it into your app.

★ **Note:** With the Drive platform, you'll use a model based on file IDs — rather than a traditional folder hierarchy — when working with Google Drive files and folders. For more information, see [Work with Folders](#) and [Manage File Metadata](#).

Search for files

You can [search for files](#) using the [files.list](#) method of the Drive API.

Drive automatically indexes the content of most common file types (e.g. .html, .xml, .txt, ...) for search as soon as they are uploaded. Also, Drive uses OCR to find text in images or PDF files and automated object recognition technology to examine and index identifiable objects, people and places. For other resources such as drawings or other unique file types that are not automatically indexed by Google, [apps can provide their own indexable text](#) when they save files to Drive.

Overview

- Quickstarts
- Concepts
 - Authorization
 - Files and Folders
 - Collections
 - Team Drives
 - Permissions
 - Change Logs
 - Drive Apps
- Manage Files and Folders
 - Upload Files
 - Search Files
 - Manage File Metadata
 - Download Files
 - Work with Folders
 - Store Application Data
 - Add Custom File Properties
- Enable Collaboration
- Detect Changes and Revisions
- Integrate with the Drive UI

Contenido

- Search for files
- Distribute and market through Google Drive
- Share and collaborate
- Create and open files using the Google Picker
- Use shortcuts
- Export and convert Google docs



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Traductor de Google x Google Drive REST API x Get Started with the Cal x

Es seguro | <https://developers.google.com/google-apps/calendar/overview>

Aplicaciones Scintilla Fivewin Traductor de Google Log in — Bitbucket Personal Acceso Cliente Qt Windows 10 Facebook Gestión de Partes Temp Canales Google Drive

Google Calendar API

Buscar TODOS LOS PRODUCTOS

INICIO GUÍAS REFERENCIA MUESTRAS ASISTENCIA ENVIAR COMENTARIOS

Get Started

Understand the Calendar API
Overview
Calendars and Events
Sharing and Attendees
Reminders and Notification
Google for Work Features

Use the Calendar API
Authorize Requests
Create Events
Recurring Events
Synchronize Resources
Get Push Notifications
Get Versioned Resources
Extended Properties
Pagination
Batch Requests
Improve Performance
Handle API Errors

Get Started with the Calendar API

☆☆☆☆☆

The Calendar API lets you display, create and modify calendar events as well as work with many other calendar-related objects, such as calendars or access controls. This document describes how to use RESTful calls and client libraries for various programming languages (Java, PHP, .NET, JavaScript, NodeJs, Ruby, Python, Go, Android, iOS).

Here are a few ideas for getting started with the Google Calendar API.

APIs Explorer

To play around and see what the API can do, without writing any code, visit the [APIs Explorer](#). Try using `calendar.events.list`, with your Google email as the calendar ID. (You'll need to enable [OAuth 2.0](#) authorization.)

Quickstarts

For a step-by-step tutorial explaining how to get up and running, follow the instructions in one of our quickstart guides:

- [.NET](#)
- [Android](#)

Contenido
APIs Explorer
Quickstarts
Download Client Libraries
Reference



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Traductor de Google x Google Drive REST API x API Reference | Google x

Es seguro | <https://developers.google.com/google-apps/calendar/v3/reference/>

Aplicaciones Scintilla Fivewin Traductor de Google Log in — Bitbucket Personal Acceso Cliente Qt Windows 10 Facebook Gestión de Partes Temp Canales Google Drive

31 Google Calendar API

INICIO GUÍAS REFERENCIA MUESTRAS ASISTENCIA

ENVIAR COMENTARIOS

API Reference

☆☆☆☆☆

This API reference is organized by resource type. Each resource type has one or more data representations and one or more methods.

Resource types

Acl

For Acl Resource details, see the [resource representation](#) page.

Method	HTTP request	Description
URIs relative to https://www.googleapis.com/calendar/v3 , unless otherwise noted		
delete	DELETE <code>/calendars/<i>calendarId</i>/acl/<i>ruleId</i></code>	Deletes an access control rule.
get	GET <code>/calendars/<i>calendarId</i>/acl/<i>ruleId</i></code>	Returns an access control rule.
insert	POST <code>/calendars/<i>calendarId</i>/acl</code>	Creates an access control rule.



Primeros Pasos

- Solicitar la Autorización de uso
(Esta acción es válida y necesaria para todos productos)
- Buscar el “SCOPE” apropiado para cada producto y propósito de uso
Ejemplo para Calendar:

Scope	Meaning
https://www.googleapis.com/auth/calendar	read/write access to Calendars
https://www.googleapis.com/auth/calendar.readonly	read-only access to Calendars



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Authorize Request and Scope

Tructor de Google x Google Drive REST API x Authorizing Requests to x

Es seguro | <https://developers.google.com/google-apps/calendar/auth>

Aplicaciones Scintilla Fivewin Traductor de Google Log in — Bitbucket Personal Acceso Cliente Qt Windows 10 Facebook Gestión de Partes Temp Canales Google Drive

31 Google Calendar API

INICIO GUÍAS REFERENCIA MUESTRAS ASISTENCIA

ENVIAR COMENTARIOS

Get Started

Understand the Calendar API

Overview

Calendars and Events

Sharing and Attendees

Reminders and Notification

Google for Work Features

Use the Calendar API

[Authorize Requests](#)

Create Events

Recurring Events

Synchronize Resources

Get Push Notifications

Get Versioned Resources

Extended Properties

Pagination

Batch Requests

Improve Performance

Handle API Errors

Use Calendar Gadgets

3. When your application needs access to user data, it asks Google for a particular **scope** of access.
4. Google displays a **consent screen** to the user, asking them to authorize your application to request some of their data.
5. If the user approves, then Google gives your application a short-lived **access token**.
6. Your application requests user data, attaching the access token to the request.
7. If Google determines that your request and the token are valid, it returns the requested data.

Some flows include additional steps, such as using **refresh tokens** to acquire new access tokens. For detailed information about flows for various types of applications, see Google's [OAuth 2.0 documentation](#).

Here's the OAuth 2.0 scope information for the Google Calendar API:

Scope	Meaning
https://www.googleapis.com/auth/calendar	read/write access to Calendars
https://www.googleapis.com/auth/calendar.readonly	read-only access to Calendars

To request access using OAuth 2.0, your application needs the scope information, as well as information that Google supplies when you register your application (such as the client ID and the client secret).

★ **Tip:** The Google APIs client libraries can handle some of the authorization process for you. They are available for a variety of programming languages; check the [page with libraries and samples](#) for more details.

Contenido

About authorization protocols

[Authorizing requests with OAuth 2.0](#)

Perform Google Apps Domain-Wide Delegation of Authority



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Qué métodos podemos utilizar ? (Eso depende de cada API a usar)

Traductor de Google x Google Drive REST API x API Reference | Google x

Es seguro | <https://developers.google.com/google-apps/calendar/v3/reference/>

Aplicaciones Scintilla Fivewin Traductor de Google Log in — Bitbucket Personal Acceso Cliente Qt Windows 10 Facebook Gestión de Partes Temp Canales Google Drive

31 Google Calendar API

INICIO GUÍAS REFERENCIA MUESTRAS ASISTENCIA

ENVIAR COMENTARIOS

Calendars

For Calendars Resource details, see the [resource representation](#) page.

Method	HTTP request	Description
clear	POST <code>/calendars/<i>calendarId</i>/clear</code>	Clears a primary calendar. This operation deletes all events associated with the primary calendar of an account.
delete	DELETE <code>/calendars/<i>calendarId</i></code>	Deletes a secondary calendar. Use <code>calendars.clear</code> for clearing all events on primary calendars.
get	GET <code>/calendars/<i>calendarId</i></code>	Returns metadata for a calendar.
insert	POST <code>/calendars</code>	Creates a secondary calendar.
patch	PATCH <code>/calendars/<i>calendarId</i></code>	Updates metadata for a calendar. This method supports patch semantics.
update	PUT <code>/calendars/<i>calendarId</i></code>	Updates metadata for a calendar.

URIs relative to `https://www.googleapis.com/calendar/v3`, unless otherwise noted

Channels

For Channels Resource details, see the [resource representation](#) page.

Resource Summary

- › Acl
- › CalendarList
- › Calendars
- › Channels
- › Colors
- › Events
- › Freebusy
- › Settings

Contenido

- Resource types
- Acl
- CalendarList
- Calendars
- Channels
- Colors
- Events
- Freebusy
- Settings



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Traductor de Google x Google Drive REST API x API Reference | Google x

Es seguro | <https://developers.google.com/google-apps/calendar/v3/reference/>

Aplicaciones Scintilla Fivewin Traductor de Google Log in — Bitbucket Personal Acceso Cliente Qt Windows 10 Facebook Gestión de Partes Temp Canales Google Drive

31 Google Calendar API

INICIO GUÍAS REFERENCIA MUESTRAS ASISTENCIA

ENVIAR COMENTARIOS

CalendarList

For CalendarList Resource details, see the [resource representation](#) page.

Method	HTTP request	Description
URIs relative to https://www.googleapis.com/calendar/v3/ , unless otherwise noted		
delete	DELETE <code>/users/me/calendarList/<i>calendarId</i></code>	Deletes an entry on the user's calendar list.
get	GET <code>/users/me/calendarList/<i>calendarId</i></code>	Returns an entry on the user's calendar list.
insert	POST <code>/users/me/calendarList</code>	Adds an entry to the user's calendar list.
list	GET <code>/users/me/calendarList</code>	Returns entries on the user's calendar list.
patch	PATCH <code>/users/me/calendarList/<i>calendarId</i></code>	Updates an entry on the user's calendar list. This method supports patch semantics.
update	PUT <code>/users/me/calendarList/<i>calendarId</i></code>	Updates an entry on the user's calendar list.
watch	POST <code>/users/me/calendarList/watch</code>	Watch for changes to CalendarList resources.

Resource Summary

- ▶ Acl
- ▶ CalendarList
- ▶ Calendars
- ▶ Channels
- ▶ Colors
- ▶ Events
- ▶ Freebusy
- ▶ Settings

Contenido

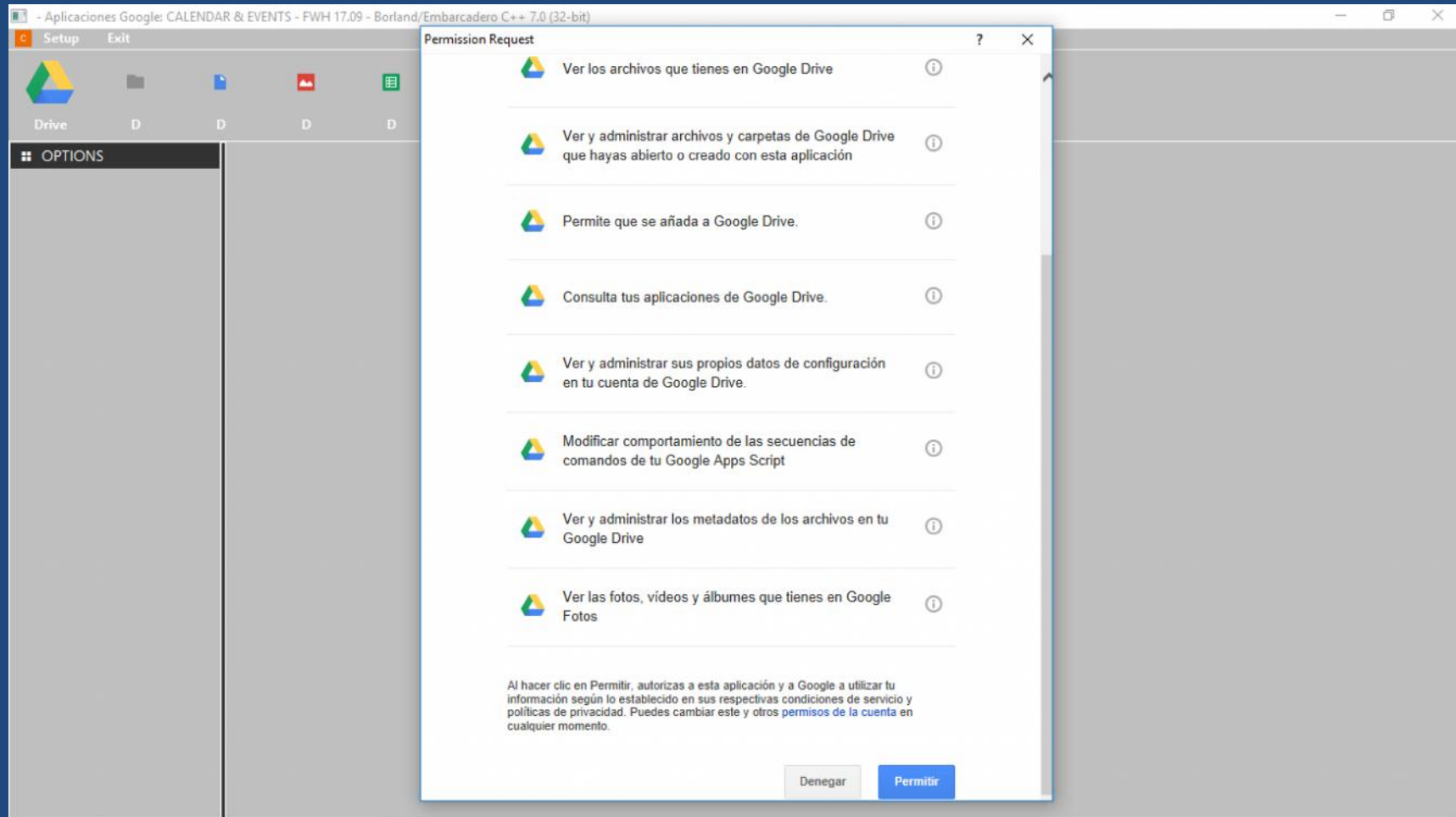
- Resource types
- Acl
- [CalendarList](#)
- Calendars
- Channels
- Colors
- Events
- Freebusy
- Settings



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)





1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



- Aplicaciones Google: CALENDAR & EVENTS - FWH 17.09 - Borland/Embarcadero C++ 7.0 (32-bit)

Setup Exit

D D D D D D D D

OPTIONS

- UPLOAD FILES
- DOWNLOAD FILES
- DELETE FILE
- EXPORT FILE TO
- REFRESH DRIVE
- SETUP DRIVE
- INFO FILE
- OPEN FILE
- CREATE NEW FOLDER
- DELETE FOLDER

Google Drive Files Tree Google Drive Files

A	NAME	ID	MIMETYPE	PARENT	SIZE
	Folder1	0B4maZB0JeqtELURKLWVhVG1FUEE	application/vnd.google-apps.folder	0AlmaZB0JeqtEUK9PVA	0
	vc98.zop	0B4maZB0JeqtEMEtNWjQtbfqTEg1RW1Ra3hINFFNZFgxX0dV	application/octet-stream	0AlmaZB0JeqtEUK9PVA	13076812
	Programas	0B4maZB0JeqtEakROXzVxTW5sOW8	application/vnd.google-apps.folder	0AlmaZB0JeqtEUK9PVA	0
	OTRA_CARPETA	0B4maZB0JeqtEcFd4bDJKX21vTUE	application/vnd.google-apps.folder	0AlmaZB0JeqtEUK9PVA	0
	barlib.png	0B4maZB0JeqtEWIVWdGFFN2M3aGQtenc0NkU3cUxzZERYUlo4	image/png	0AlmaZB0JeqtEUK9PVA	77557
	HojaGoogle1	1AJLdWtmhaXRB8pUvvlclWz9Xwkanl6zAsO9_JVcEI4Q	application/vnd.google-apps.spreadsheet	0AlmaZB0JeqtEUK9PVA	0
	defs.zip	0B4maZB0JeqtETjJUbePsS1dkU0puM0ZheDd0SszQ3bzhYWGJ3	application/zip	0AlmaZB0JeqtEUK9PVA	1373
	Title132.png	0B4maZB0JeqtEVS1EVjN0SGQ5Wkh6Vi0yWXJiazd4QmV6T2IR	image/png	0AlmaZB0JeqtEUK9PVA	77480



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



- Aplicaciones Google: CALENDAR & EVENTS - FWH 17.09 - Borland/Embarcadero C++ 7.0 (32-bit)

Setup Exit

Google Drive Files Tree Google Drive Files

A	Parent Fold	NAME	ID	MIMETYPE	PARENT	SIZE
		customer.dbf	0B4maZB0JeqtEY1laXy1ZbiRQREU	application/octet-stream	0B4maZB0JeqtELURKLWVhV	101887
		CUSTOMER.cdx	0B4maZB0JeqtEX2VaZIFZWkojdU0	image/x-coreldraw	0B4maZB0JeqtELURKLWVhV	60416
		atis4win.prg	0B4maZB0JeqtEUWVF3J1c2IPa2RBLU52Q0RxaC1nRG8wcUdV	text/plain	0B4maZB0JeqtELURKLWVhV	119838
		test.prg	0B4maZB0JeqtEbWJRY0JGc0pxOTA	application/octet-stream	0B4maZB0JeqtELURKLWVhV	7373

OPTIONS

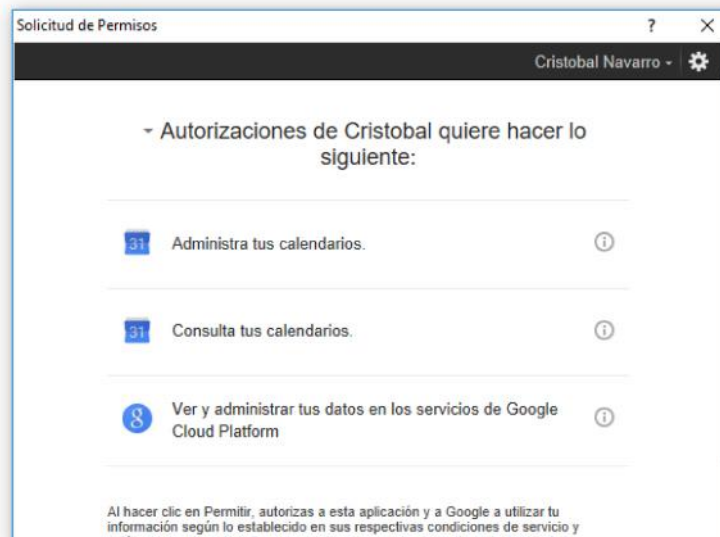
- UPLOAD FILES
- DOWNLOAD FILES
- DELETE FILE
- EXPORT FILE TO
- REFRESH DRIVE
- SETUP DRIVE
- INFO FILE
- OPEN FILE
- CREATE NEW FOLDER
- DELETE FOLDER



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)





1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Exit

Sheets

Docs

Forms

Presentations

Drive

Gmail

Calendar

Tasks

Contacts

Groups

Maps

Translate

Sites

Cloud Print

Photos

Firebase

ACTIONS

31 Add Calendar

Delete Calendar

Refresh List Calendar

Exit Calendar

navarro.cristobal@gmail.com

Calendar

November 2017

MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY	SATURDAY	SUNDAY
		1	2	3	4	5
6	7	8	9	10 Salir hacia Novelda	11 Conferencia	12 Conferencia
13	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30			

navarro.cristobal@gmail.com

romyrna@gmail.com

Otro Calendario

Contacts

Festivos en España



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Exit

Sheets

Docs

Forms

Presentations

Drive

Gmail

Calendar

Tasks

Contacts

Groups

Maps

Translate

Sites

Cloud Print

Photos

Firebase

ACTIONS

31 Add Calendar

Delete Calendar

Refresh List Calendar

Exit Calendar

navarro.cristobal@gmail.com

romyrna@gmail.com

Otro Calendario

Contacts

Festivos en España

navarro.cristobal@gmail.com

Calendar

ID	SUMMARY	DESCRIPTION	CREATED	UPDATED	START	END
14rhfm6gsh53s705ratk5rkk8f	Salir hacia Novelda	Not Implemented	2017-11-04	2017-11-04	2017-11-10	2017-11-11
1rmipvrtu95bhheg0udor9pn44	Prueba con Meses consecutivos	Not Implemented	2017-02-13	2017-02-13	2017-02-16	2017-02-16
32cibr4ib418q81t6u07k5oooi	Conferencia	Not Implemented	2017-11-04	2017-11-04	2017-11-12	2017-11-13
5u72u1gpsaudiusnd4t1rgkkr8	Prueba desde EXE	Descripcion evento	2017-07-04	2017-07-04	2017-07-14	2017-07-14
5ubdpok092u4rie3amoeg3t5pc	Prueba desde EXE	Descripcion evento	2017-01-14	2017-01-14	2017-01-24	2017-01-24
77tgqoa2fjvqbd7m42nh9c1463	Conferencia	Not Implemented	2017-11-04	2017-11-04	2017-11-11	2017-11-12
8cn71nl2aupsdirmfavmsiallo	Prueba desde EXE	Descripcion evento	2017-07-04	2017-07-04	2017-07-14	2017-07-14
8s3cjrggcs6krvo73q1cp5t14c	Otra Prueba con Fechas	Not Implemented	2017-02-11	2017-02-11	2017-02-12	2017-02-13
irbh1m88copv25k62hkvni30no	Evento con intervalo de	Descripcion del Evento con intervalo de	2017-02-11	2017-02-11	2017-02-15	2017-02-15
tdnk8ssp3vql5cj0qmr6vi3ovo	Prueba desde EXE	Descripcion evento	2017-01-15	2017-01-15	2017-01-25	2017-01-25
tn8ttm2vit8ns4pt5h7izv8k6s	Mañana ir a Madrid	Not Implemented	2017-01-14	2017-01-14	2017-01-15	2017-01-16
ue8r8eip7jie1f36uln4i3qhtg	Evento de prueba-01	Descripcion del evento prueba-01	2017-01-14	2017-01-14	2017-01-14	2017-01-15
_6tlnaqrle5p6cpb4dhmj4phpegsml	Vuelo a MADRID (IB 3191)	Si quieres ver información detallada sobre lc	2017-06-29	2017-07-07	2017-07-07	2017-07-07
_6tlnaqrle5p6cpb4dhmj4phpeghmi	Vuelo a Múnich (UX 1517)	Si quieres ver información detallada sobre lc	2017-05-09	2017-07-02	2017-07-02	2017-07-02



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



FE - FIVEDITS - Ver.: 171029 (32 bits) - Usuario: C - File: \Fwh\FwhTeam\samples\gcalend.prg

Fichero Proyecto Editar Buscar Ejecutar Transformar Visualizar Herramientas Historico Ayuda

CALENG.PRG CALGOOGLE.PRG GOOGLECAL.PRG DRIVEG.PRG GVISION.PRG TGVISION.PRG TGOOGLE.PRG TGCALEND.PRG TGEVECAL.PRG TGCALLIS.PRG FUNCTIONS PROJECTS DATABASES RECENTS

TGMAIL.PRG GGMAIL.PRG TGSHEET.PRG TGTASKS.PRG TGTRANSL.PRG TGMESAGE.PRG TRANSLATE.PRG MAPS1.PRG MAPS4.PRG GCALEND.PRG

```
43 Function CalendList()  
44  
45 local oCalendList  
46 local lSw := .F.  
47  
48 oCalendList := TGCalendarList():New()  
49 lSw := oCalendList:Activate()  
50 if lSw  
51 ? "Activado"  
52 oCalendList:ListCalendars()  
53 XBrowse( oCalendList:aCalendList )  
54 endif  
55  
56 Return nil  
57  
58 //-----  
59  
60 Function EventList()  
61  
62 local oEvents  
63 local lSw  
64  
65 oEvents := TGEEventCalc():New()  
66 lSw := oEvents:Activate()  
67 if lSw  
68 ? "Activado"  
69 oEvents:ListCalendars()  
70 oEvents:nCalendAct := 1 // Seleccionamos el primer calendario de la lista  
71 oEvents:ListEvent()  
72 XBrowse( oEvents:aEventLists )  
73 endif  
74
```

Functions

Functions	Type
Calend()	FUNCTION
CalendList()	FUNCTION
EventList()	FUNCTION
Main()	FUNCTION

LIST FUNCTION TREE FUNCTION MARKS USER

Source Code Editor and Projects Zoom: 100% Config Default Harbour Borland 7.00 Annotations Actives: YES File changed: NO Row: 00043 / 00078 Col: 001 / 023 Num Caps Ins



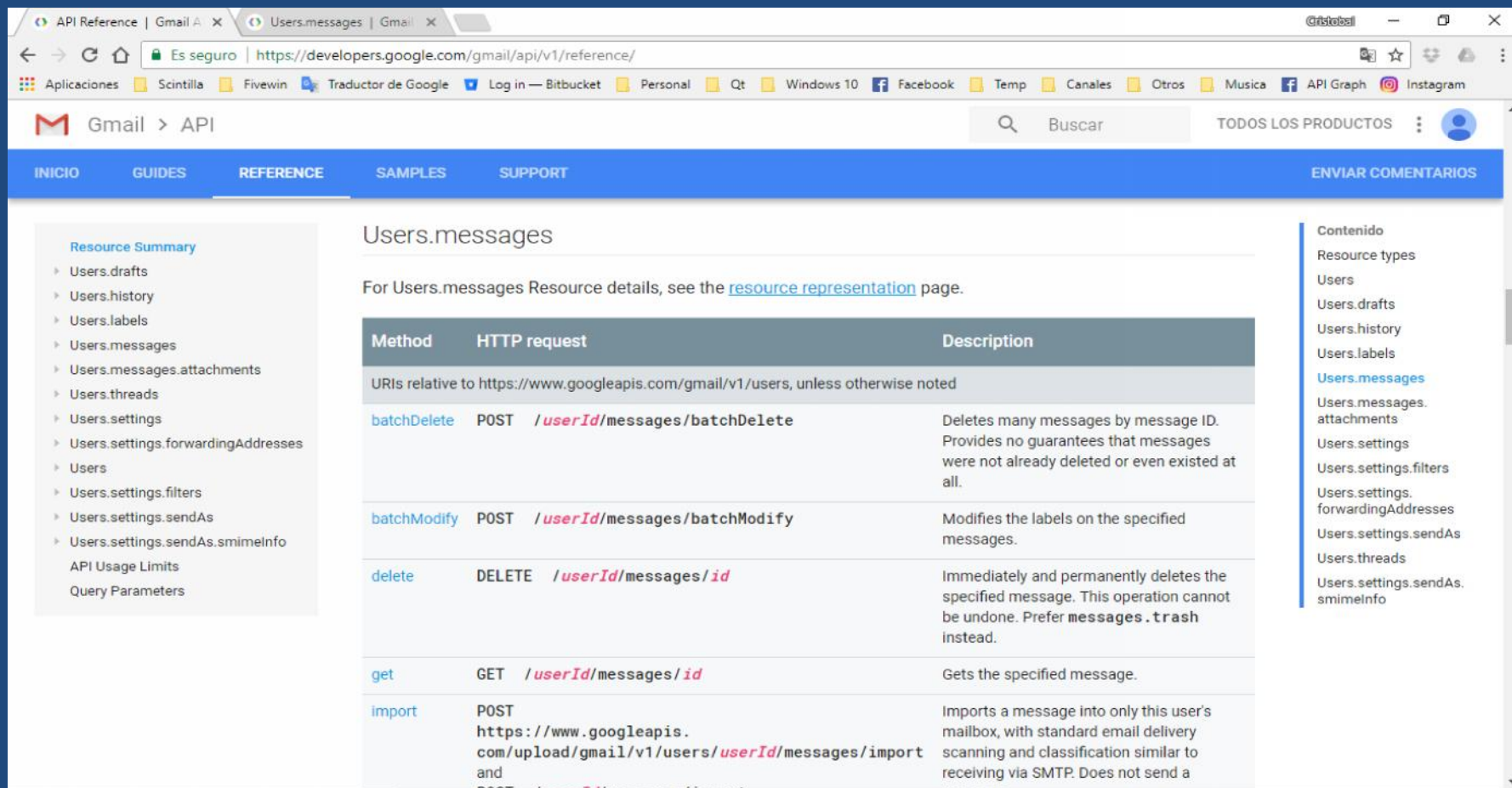
1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



El desarrollo de los **METHODS** está realizado según los métodos que utiliza el API



The screenshot shows the Gmail API Reference page for the `Users.messages` resource. The page is titled "Users.messages" and includes a navigation sidebar on the left with links to various API resources. The main content area displays a table of methods for this resource, including `batchDelete`, `batchModify`, `delete`, `get`, and `import`. Each method entry includes its HTTP request (method and URL) and a description of its function.

Method	HTTP request	Description
URIs relative to <code>https://www.googleapis.com/gmail/v1/users</code> , unless otherwise noted		
<code>batchDelete</code>	POST <code>/userId/messages/batchDelete</code>	Deletes many messages by message ID. Provides no guarantees that messages were not already deleted or even existed at all.
<code>batchModify</code>	POST <code>/userId/messages/batchModify</code>	Modifies the labels on the specified messages.
<code>delete</code>	DELETE <code>/userId/messages/id</code>	Immediately and permanently deletes the specified message. This operation cannot be undone. Prefer <code>messages.trash</code> instead.
<code>get</code>	GET <code>/userId/messages/id</code>	Gets the specified message.
<code>import</code>	POST <code>https://www.googleapis.com/upload/gmail/v1/users/userId/messages/import</code> and <code>POST /userId/messages/import</code>	Imports a message into only this user's mailbox, with standard email delivery scanning and classification similar to receiving via SMTP. Does not send a message.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Y las **DATAS**, las que sus recursos necesitan para evitar el uso de variable externas a la clase

The screenshot shows the Google Developers API Reference page for the Gmail `Users.messages` resource. The browser address bar shows the URL `https://developers.google.com/gmail/api/v1/reference/users/messages#resource`. The page has a blue header with navigation links: INICIO, GUIDES, REFERENCE (selected), SAMPLES, and SUPPORT. On the right of the header is a search bar and a link to 'ENVIAR COMENTARIOS'. A left sidebar contains a 'Resource Summary' with a tree view showing the hierarchy: `Users.messages` (selected), `Overview`, `delete`, `get`, `insert`, `list`, `modify`, `send`, `trash`, `untrash`, `import`, `batchDelete`, `batchModify`, `Users.messages.attachments`, `Users.threads`, `Users.settings`, `Users.settings.forwardingAddresses`. The main content area is titled 'Resource representations' and contains the text 'An email message.' followed by a JSON representation of the resource. A right sidebar shows a 'Contenido' section with links to 'Resource representations' (selected) and 'Methods'. The JSON code is as follows:

```
{
  "id": string,
  "threadId": string,
  "labelIds": [
    string
  ],
  "snippet": string,
  "historyId": unsigned long,
  "internalDate": long,
  "payload": {
    "partId": string,
    "mimeType": string,
    "filename": string,
    "headers": [
      {
        "name": string,
        "value": string
      }
    ],
    "body": users.messages.attachments Resource,
    "parts": [
      (MessagePart)
    ]
  }
}
```




Clases for API Google

- TGGOOGLE
 - TGCalendar
 - TGCalendarList
 - TGEventCalc
 - TGDrive
 - TGMail
 - TGTasks
 - TGTranslate
 - TGVision (CLOUD)



CLASS TGGoogle

METHODS:

METHOD New(IDbg) CONSTRUCTOR
METHOD Activate(nT, nL, nH, cTit)
METHOD Authorize()
METHOD EventShell(Event, aParams, pParams, oWnd, cWndTitle)
METHOD GetTokens(cStr)
METHOD GetTokensAuth(cStr)
METHOD Init()
METHOD End()
METHOD HMethod(cUrl, cParams, cContentType, cAuth, cType)
METHOD AddAction(cStrUrl, cStrFormData, cAppli, cType)
METHOD ListAction(cStrUrl, cStrFormData)
METHOD ViewAcces(oObj, nH)



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



DATAS:

DATA cKeyAuth
DATA strResponseText
DATA cClientId
DATA cKey_Id
DATA cSecretId
DATA cRedirect
DATA cRedirect2
DATA cProject
DATA cAuthUri
DATA cTokenUri
DATA cCertXUri
DATA cURLAuth
DATA strToken
DATA strRefreshToken
DATA cId_Token



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



DATA cExpire
DATA cType_Token
DATA cUser_Id
DATA aMethods
DATA cResponse
DATA id
DATA cSummary
DATA bView
DATA IDebug
DATA IMail INIT .F.
DATA IProfile INIT .F.
DATA IDrive INIT .F.
DATA IAddDrive INIT .F.
DATA ICalendar INIT .F.
DATA ITasks INIT .F.
DATA ISheets INIT .F.
DATA IDoc INIT .F.



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



DATA IForm	INIT .F.
DATA IPresen	INIT .F.
DATA IContac	INIT .F.
DATA IGroup	INIT .F.
DATA IMap	INIT .F.
DATA ITrans	INIT .F.
DATA ISite	INIT .F.
DATA ICloud	INIT .F.
DATA IPhoto	INIT .F.
DATA IFire	INIT .F.
CLASSDATA cRevokeUri	
CLASSDATA cScope	
CLASSDATE cBaseUrl	
CLASSDATA myEmail	
CLASSDATA myPassword	
CLASSDATA cJsonFile	



CLASS TGMail FROM TGGoogle

```
DATA aThreads    INIT {}  
DATA aMessages   INIT {}  
DATA aDrafts     INIT {}  
DATA aLabels     INIT {}  
DATA aHistorys   INIT {}  
DATA cLabel      INIT ""  
DATA cMessage    INIT ""  
DATA cld_Message INIT ""  
DATA cld_Thread  INIT ""  
DATA cld_History INIT ""  
DATA cld_Labels  INIT ""  
DATA cNextPageToken  
DATA nResultSize INIT 0
```



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



METHOD New() CONSTRUCTOR

METHOD End()

METHOD Action(cId, nPos, cType)

METHOD Get(cId, nPos)

METHOD Delete(cId, nPos)

METHOD Trash(cId, nPos)

METHOD UnTrash(cId, nPos)

METHOD List(cType, cLabel, cId, nMaxResults, lAdd)

METHOD ListDrafts(nMaxResults) INLINE ::List("drafts", , , nMaxResults,)

METHOD ListLabels(nMaxResults) INLINE ::List("labels", , , nMaxResults,)

METHOD ListMessages(nMaxResults, cLabel) INLINE ::List("messages", cLabel, , nMaxResults,)

METHOD ListThreads(nMaxResults) INLINE ::List("threads", , , nMaxResults,)

METHOD ListHistorys(nMaxResults, cId) INLINE ::List("history", , cId, nMaxResults,)



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



METHOD ChatMessages(nMaxResults)	INLINE ::ListMessages(nMaxResults, "CHAT")
METHOD DraftMessages(nMaxResults)	INLINE ::ListMessages(nMaxResults, "DRAFT")
METHOD InboxMessages(nMaxResults)	INLINE ::ListMessages(nMaxResults, "INBOX")
METHOD SentMessages(nMaxResults)	INLINE ::ListMessages(nMaxResults, "SENT")
METHOD SpamMessages(nMaxResults)	INLINE ::ListMessages(nMaxResults, "SPAM")
METHOD TrahsMessages(nMaxResults)	INLINE ::ListMessages(nMaxResults, "TRASH")
METHOD UnReadMessages(nMaxResults)	INLINE ::ListMessages(nMaxResults, "UNREAD")

ENDCLASS



CLASS TGCalendar FROM TGGoogle

DATA aCalend INIT {}

METHOD New() CONSTRUCTOR

METHOD End()

METHOD AddCalendar(cNew)

METHOD ClearCalendar(cId)

METHOD DelCalendar(cId)

METHOD GetCalendar(cId)

METHOD GetColors()

ENDCLASS



CLASS TGCalendarList FROM TGCalendar

```
DATA aCalendList INIT {}  
DATA nCalendAct INIT 1  
  
METHOD New() CONSTRUCTOR  
METHOD Activate( nT, nL, nH, cTit )  
METHOD End()  
METHOD ListCalendars()  
METHOD HashListCalend( cStr )
```

ENDCLASS



CLASS TGEventCalc FROM TGCalendarList

DATA dDateI	INIT Ctod(" / / ")
DATA dDateF	INIT Ctod(" / / ")
DATA dDateC	INIT Ctod(" / / ")
DATA dDateU	INIT Ctod(" / / ")
DATA cEvent	INIT Space(120)
DATA cDescrip	INIT Space(120)
DATA cProjectE	INIT Space(80)
DATA cLoc	INIT Space(80)
DATA cTimeI	INIT " : : "
DATA cTimeF	INIT " : : "
DATA cTimeC	INIT " : : "
DATA cTimeU	INIT " : : "
DATA cHtmlLink	INIT Space(120)



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



```
DATA cCalUId      INIT Space( 40 )  
DATA cStatus      INIT Space( 10 )  
DATA cMyEmail     INIT Space( 80 )  
DATA cDisplayName INIT Space( 80 )  
DATA cEmailOrg    INIT Space( 80 )  
DATA cDisplayOrg  INIT Space( 80 )  
DATA cTimeZoneI  INIT Space( 40 )  
DATA cTimeZoneF  INIT Space( 40 )  
DATA aEventLists  INIT {}
```



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



```
METHOD New() CONSTRUCTOR  
METHOD End()  
METHOD AddEvent()  
METHOD DelEvent( cld, IMsg )  
METHOD EditEvent()  
METHOD HashListEvents( llni )  
METHOD Init()  
METHOD ListEvent( llni )  
METHOD ListAllEvents()  
METHOD MoveEvent( cld, nDest, IMsg )  
METHOD SearchEvent( cld, IMsg )  
//METHOD UnDelete( cld, IMsg )
```

ENDCLASS



CLASS TGDrive FROM TGGoogle

```
DATA aFilesLists    INIT {}  
DATA IVer2          INIT .T.  
DATA IViewIcon      INIT .T.  
DATA IAddDrive       INIT .T.  
DATA cMime           INIT ""
```

```
METHOD New( IDbg, IV2 ) CONSTRUCTOR  
METHOD AddFolderG( cFolder )  
METHOD DelFileFolderG( cFolder, IFold, ITrash )  
METHOD DownLoadFileG( uFile, cMime )  
METHOD End()  
METHOD ExportFile( uFile, uFormat )  
METHOD UpLoadFileG( uFile, nSize )
```



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



```
METHOD HashItemFiles( cStr )
METHOD HashInfoFiles( cStr )
METHOD HashListFiles( cStr, llni )
METHOD IconFile( uFile )
METHOD InfoFile( uFile )
METHOD ListAllFiles()
METHOD ListFiles( llni )
METHOD ListFolderFiles( uFold )
METHOD ListOnlyFiles()
METHOD ListOnlyFolders()
METHOD ListRootFiles( llni )      INLINE ::ListFolderFiles( llni, "root" )
METHOD ListTrashFiles()
METHOD MimeTypes( cExt )
METHOD OpenFile( uFile, cMime )
METHOD ParentFile( uFile )
```

ENDCLASS



CLASS TGMail: Sample

```
#include "Fivewin.ch"
```

```
Function Main()
```

```
    local oMail
```

```
    local lSw := .F.
```

```
    oMail := TGMail():New()
```

```
    lSw := oMail:Activate()
```

```
if lSw
```

```
    oMail:ListThreads()
```

```
    oMail:ListMessages(,)  
    XBrowse( oMail:aMessages )
```

```
    oMail:ListDrafts()  
    oMail:ListLabels()  
    XBrowse( oMail:aLabels )
```

```
    oMail:Get( , 10 )  
    ? oMail:cMessage
```

```
    XBrowse( oMail:cId_Labels )  
    oMail:ListHistorys()  
    XBrowse( oMail:aHistorys )
```

```
endif  
Return nil
```




1º Encuentro Programadores HARBOUR
NOVELDA – 10/11/2017
por Cristóbal Navarro López (C)



**oMail:SpamMessages()
XBrowse(oMail:aMessages)**

**oMail:TrahsMessages()
XBrowse(oMail:aMessages)**

**oMail:UnReadMessages()
XBrowse(oMail:aMessages)**

**oMail:ChatMessages()
XBrowse(oMail:aMessages)**

**oMail:DraftMessages()
XBrowse(oMail:aMessages)**

**oMail:SentMessages()
XBrowse(oMail:aMessages)**

**oMail:InboxMessages()
XBrowse(oMail:aMessages)**



CLASS TGCalendar: Sample

```
#include "Fivewin.ch"
```

```
Function Main()
```

```
    //Calend()    // No es necesario, solo como ejemplo de creacion de objeto calend  
    CalendList()  
    EventList()
```

```
Return nil
```

```
//-----//
```



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Function Calend()

local oCalend

local ISw := .F.

oCalend := TGCalendar():New()

ISw := oCalend:Activate()

if ISw

 ? "Activado"

 oCalend:GetColors()

 ? oCalend:cResponse

else

 ? oCalend:cError

endif

Return nil

//-----//



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Function CalendList()

local oCalendList

local ISw := .F.

oCalendList := TGCalendarList():New()

ISw := oCalendList:Activate()

if ISw

? "Activado"

oCalendList:ListCalendars()

XBrowse(oCalendList:aCalendList)

endif

Return nil

//-----//



1º Encuentro Programadores HARBOUR

NOVELDA – 10/11/2017

por Cristóbal Navarro López (C)



Function EventList()

local oEvents

local lSw

oEvents := TGEEventCalc():New()

lSw := oEvents:Activate()

if lSw

 ? "Activado"

 oEvents:ListCalendars()

 oEvents:nCalendAct := 1 // Seleccionamos el primer calendario de la lista

 oEvents:ListEvent()

 XBrowse(oEvents:aEventLists)

endif

Return nil

//-----//



1º Encuentro Programadores HARBOUR
NOVELDA – 10/11/2017
por Cristóbal Navarro López (C)



SALUDOS A TODOS

ESPERO HAYA SIDO INTERESANTE