UNIDAD 2 – APIs

1. UNIDAD 2: Introducción

 Bienvenido a Backend

En esta unidad estudiaremos uno de los conceptos más usados en el desarrollo de cualquier tipo de aplicación: API o Interfaz de Programación de Aplicaciones. Podrás explorar su funcionamiento y realizar ejemplos, así como comprender qué es un servicio web y cómo se usa un API REST.

 Puntos de aprendizaje

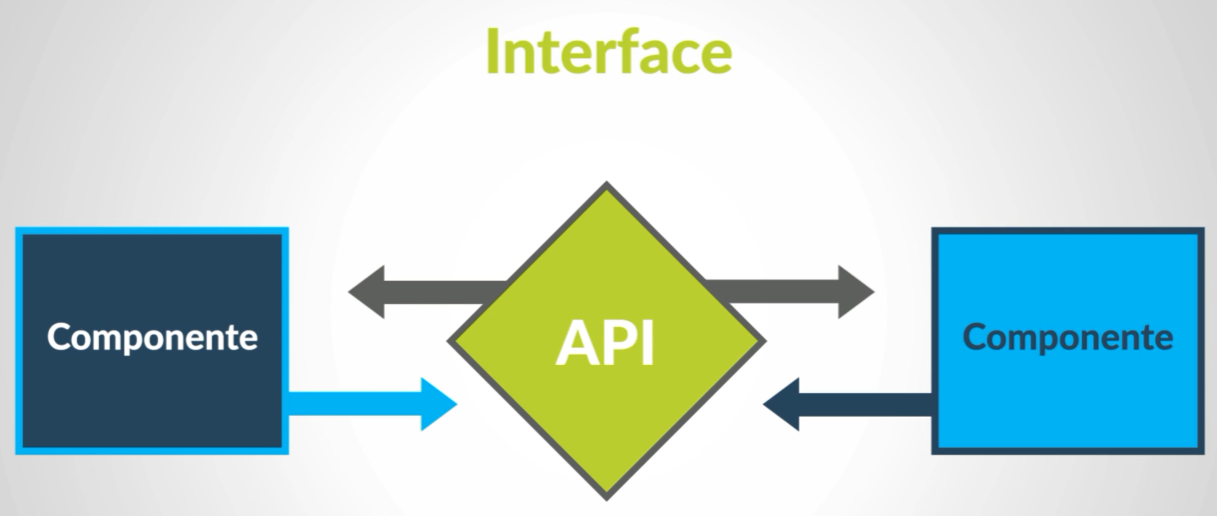
Unidad 2: API's  
¡Aquí comienza la Unidad 2: API's!

* Los objetivos del Programa que corresponden a esta Unidad son:
  + Usar metodologías sincrónicas y asincrónicas para gestionar datos desde el servidor.
* Los objetivos del Curso 6 que corresponden a esta Unidad son:
  + Describir el funcionamiento de la ejecución sincrónica y asincrónica
  + Definir qué es una API y cómo funcionan los Servicios Web
  + Implementar AJAX

1. Lección 1: Qué es una API?
   1. Qué es una API y para qué sirve

Application Programming Interface

Intermediario entre dos componentes de software



Web API – Cualquier interfaz de software en la web, móvil o nube

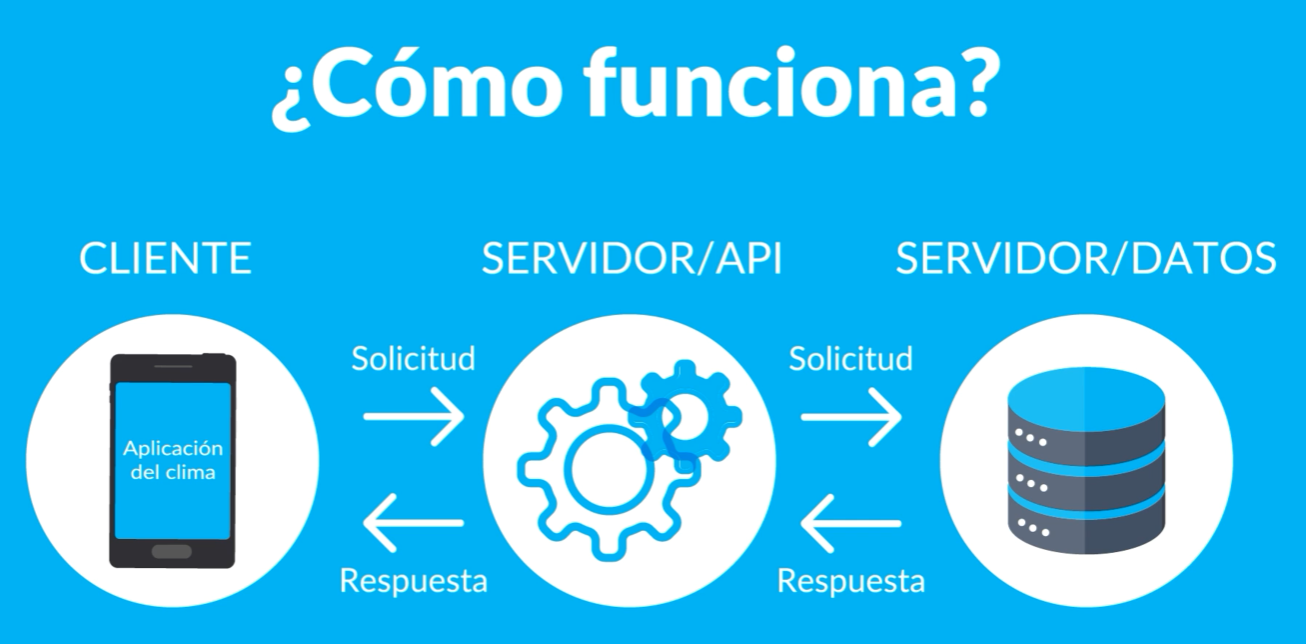
Ventajas:

* Mayor agilidad – cuando se usan algunas API de terceros que se adaptan a las necesidades propias.
* Separación de intereses – Ofrece muchas ventajas como mejor organización de código, manejo, mantenimiento y documentación, entre otros.
* Seguridad – Permite controlar los datos que se compartirán entre componentes.
* Acceso fácil a funcionalidades.
  1. Como funciona una API

Cuando dos componentes se conectan a través de una API, decimos que están integrados.

Servidor – Componente que proveen los datos. La API es otro componente que está corriendo en el servidor

Cliente – Es la parte que sabe que los datos están disponibles a través de una API y que los puede manipular.



Seguridad: los datos del teléfono nunca están expuestos al servidor, del mismo el servidor nunca está expuesto al teléfono.

* 1. Ejemplos de APIs en lenguajes de programación

Ejemplos de APIs



Diariamente nuevas herramientas de desarrollo web son lanzadas al mercado. Como desarrolladores, para mantenernos actualizados debemos explorar constantemente nuevas tecnologías que mejorarán la calidad del código que entregamos. Por ende, existen miles de APIs que han sido diseñadas por compañías para ahorrar tiempo, resolver problemas de desarrollo o aumentar la popularidad de sus servicios. A continuación mostramos una lista de las APIs más populares y para qué sirve cada una:

1. Google Maps APIs

Estas APIs permiten incrustar Google Maps en sitios web. Están diseñadas para trabajar en dispositivos móviles y también para aplicaciones en navegadores de equipos de escritorio. Además, proporcionan datos geográficos como geocodificación, indicaciones, elevación, sitio e información de zonas horarias, entre otras. Un caso básico de su uso es el de dibujar lugares como marcadores en un mapa aunque nos permite también dibujar rutas hacia múltiples puntos. Usos más avanzados incluyen la creación de interacciones como hacer clic en un resultado de una búsqueda y resaltar el lugar en el mapa.  
**Proveedor:** Google  
**Disponible en:**<https://cloud.google.com/maps-platform/>

2. Twitter API:

Esta API provee acceso programático para leer y escribir datos de Twitter. Esto incluye actualizar líneas de tiempo, datos de status e información de usuario, crear un nuevo Tweet, y más. La API de busqueda le da la posibilidad a los desarrolladores de hacer búsquedas entre los Tweets más populares de tal forma que se pueda tener datos acerca de tendencias.  
**Proveedor:** Twitter  
**Disponible en:**<https://developer.twitter.com/en/docs>

3. Facebook API:

La API de Facebook permite construir aplicaciones con funcionalidades de Facebook integradas. Incluir este tipo de tecnologías es una buena alternativa pues hace que tu sitio sea más social al usar las conecciones y la información de perfil de los usuarios, mostrando contenidos acorde con ésta información. La API de facebook tiene mucho por ofrecer, sin embargo podemos destacar dos funcionalidades. La primera es el inicio de sesión con Facebook para aplicaciones que es una forma cómoda, rápida y segura de iniciar sesión en una aplicación o un sitio web; la segunda es la facilidad a la hora de compartir contenidos ya que puedes agregar métodos rápidos y sencillos para que los usuarios publiquen contenido en Facebook directamente desde tu sitio web.  
**Proveedor:** Facebook  
**Disponible en:**<https://developers.facebook.com/>

4. Yahoo Weather API:

Con la API de clima de Yahoo puedes obtener información actualizada acerca del clima en tu ubicación. Estos datos pueden ser usados como un RSS feed en un lector de feeds o pueden ser integrados en aplicaciones de terceros. Con la API de Yahoo puedes obtener información de cualquier ubicación, incluyendo pronósticos de 5 dias, vientos, atmósfera, condiciones astronómicas y más. También puedes consultar el clima por el nombre de la ciudad o latitud/longitud.  
**Proveedor:** Yahoo  
**Disponible en:**<https://developer.yahoo.com/weather/>

YouTube APIs:

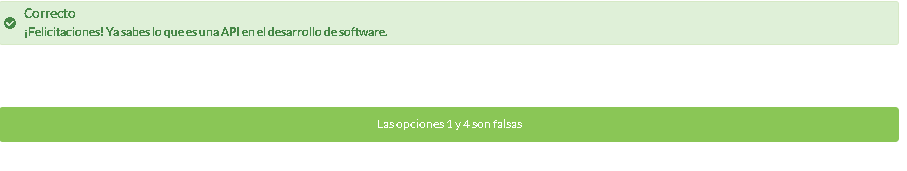
Las APIs de Youtube hacen posible integrar en tu sitio web muchas de las operaciones disponibles en el sitio web de YouTube. Esta API de datos permite buscar videos, y ver contenidos relacionados a ellos. Así, al autenticarse como usuario permitirá subir videos, crear y manejar listas de reproducción, entre otras. Además, la API de analíticas devuelve datos estadísticos de vistas, métricas de popularidad y más, para videos y canales de YouTube.  
**Proveedor:** Google  
**Disponible en:**<https://developers.google.com/youtube/>

* 1. Actividad

 Tiempo de pensar

En esta lección pudiste observar un nuevo concepto: API o Application Programming Interface, el cual es un intermediario entre dos programas, que provee información desde un servidor hasta un cliente. Además, pudiste conocer algunos de los ejemplos más comunes como la API de Google Maps, la API de Facebook o la de Twitter.  
  
Instrucciones:  
  
A continuación, verás una serie de afirmaciones sobre las APIs. Léelas e identifica cuáles son verdaderas y cuáles falsas, luego elige la opción correcta.  
  
Afirmaciones:

* Las APIs se diseñaron con el objetivo de crear la estructura base para un proyecto.
* Las APIs permiten la separación de intereses en el desarrollo de software.
* Las APIs permiten exponer funcionalidades del back-end.
* Las APIs exponen funcionalidades del front-end.



1. Lección 2: Introducción a los Servicios Web
   1. Qué es un servicio web

¿Qué es un servicio web?



Existen múltiples definiciones sobre lo que son los servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son y lo que implican. *En esta ocasión haremos un estudio del concepto de Servicios web y para qué sirven.*

Podemos definir Servicio web como una colección de protocolos abiertos y estándares usados para intercambiar datos entre aplicaciones o sistemas. Los servicios web pueden ser usados por aplicaciones de software que estén escritas en diferentes lenguajes de programación y que corran en varias plataformas, para intercambiar datos entre redes de computadores como la Internet, de manera similar a la comunicación entre procesos en un solo computador. Esta interoperabilidad, por ejemplo, entre Java y Python o entre aplicaciones Windows y Linux se da debido a los estándares abiertos. Una definición más concreta de **servicio web**es la siguiente: *“aplicación que expone su funcionalidad a través de una interfaz de programación de aplicaciones API”.*

Recordemos que en la lección anterior aprendimos que una API hace posible la comunicación aplicación-aplicación. Aquí llegamos a un punto clave: *los servicios web usan la web para ejecutar integración aplicación-aplicación.*

Así, entender el concepto de servicio es la clave para entender los servicios Web. *Un servicio es una pieza de software que trabaja para otro software.*En muchas circunstancias un servicio corre en un servidor esperando a que una aplicación lo llame y le pida hacer algún trabajo. En muchos casos, lo servicios no proveen ningún tipo de interfaz humana y la única forma de acceder al servicio es a través de su API.

Un servicio web es un **recurso** que está diseñado para ser consumido por un software en vez de humanos. Ahora, un recurso web es cualquier tipo de objeto de información como una imágen, una cuenta de correo, un documento, una aplicación, entre otros, a los que puedes acceder a través de la web.

En esto radica la diferencia entre un **servicio web** y un **sitio web.** Un sitio web representa un grupo de recursos web diseñados para ser accedidos por humanos, en cambio los servicios web son un grupo de recursos diseñados para ser accedidos por aplicaciones. El término servicio implica algo especial sobre diseño de aplicaciones. Se refiere a algo conocido como **arquitectura orientada a los servicios** del inglés service-oriented architecture **(SOA).**

Una de las características más importantes del **SOA**es la separación de la interfaz de la implementación. Un servicio expone su funcionalidad a través de su interfaz y ésta, a su vez, esconde los trabajos de la aplicación. Por ende, la aplicación cliente no necesita entender cómo el servicio hace su trabajo, todo lo que debe saber es cómo usar la interfaz.

A manera de ilustración tomemos el ejemplo de una lavadora.  
Una lavadora tiene una interfaz que es fácil de usar, de tal forma que no necesitas saber cómo funciona por dentro el motor o los circuitos internos, pues sólo necesitas saber cómo usar la interfaz.

¿Para qué sirven?

Los servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar.

* 1. Cómo funcionan

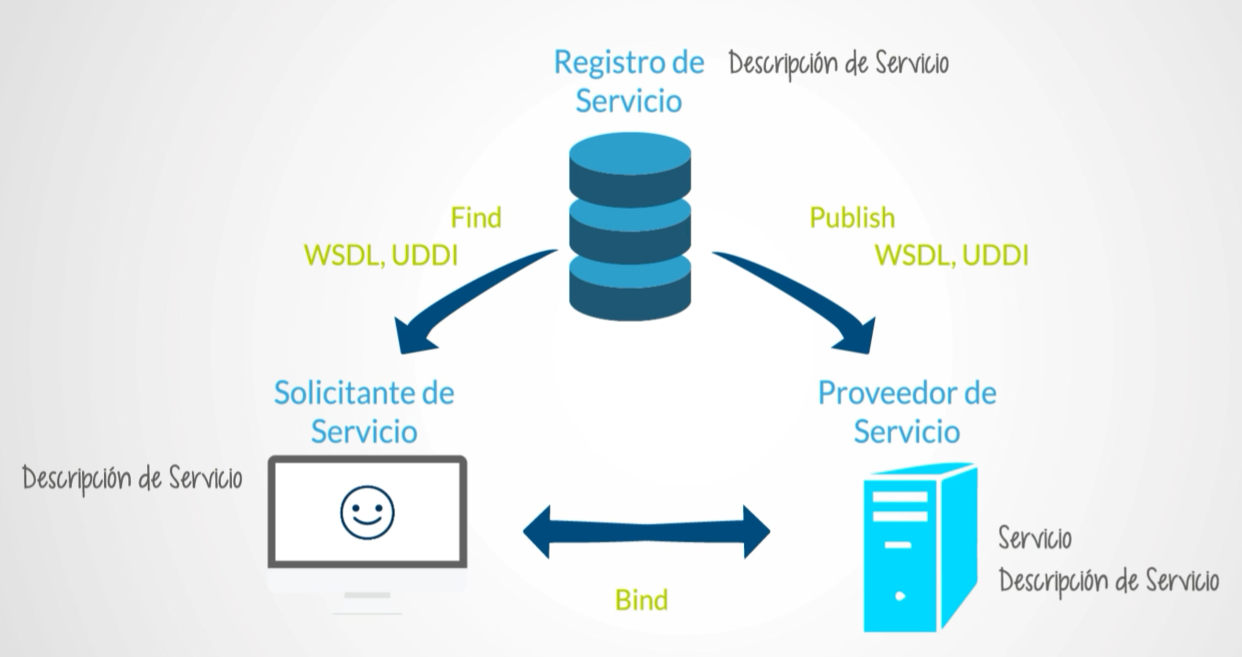
Se dan entre 3 roles:

* Proveedor (service provider) 🡪 Plataforma que aloja el servicio
* Solicitante (service requester) 🡪Es el que interactúa con el servicio, puede ser manejado por una persona desde un navegador u otro servicio web
* Registro (service registry) 🡪 Opcional 🡪 Es un registro de descripción del servicio

Descripción del servicio 🡪 contiene los detalles de la interfaz y la implementación del servicio. (service description)

Para que una aplicación utilice un servicio web, debe realizarse 3 comportamientos:

* Publich – Publicación de la descripción del servicio
* Find – Búsqueda de la descripción del servicio
* Bind – Invocación o enlace con los servicios con base en su descripción



5 tipos de tecnologías organizadas en capaz

* Descubrimiento: Proporciona el mecanismo par a que los consumidores del servicio busquen la descripción del servicio de los proveedores (Ej: UDDI Universal Description Discovery and Integration – Directorio donde es posible publicar los servicios web, permitiendo que los usuarios de servicio puedan obtener toda la información necesaria para la innovación y ejecución del servicio web)
* Descripción: Contiene la información necesaria para que el solicitante del servicio pueda solicitar y usar del servicio web (WSDL – Web Service Description Language – Es el estándar por defecto para proveer ese tipo de descripciones)
* Empaquetamiento – Es necesaria para que los datos puedan moverse por la red y todas las partes puedan entenderlos (SOAP – Simple Object Access Protocol – es una forma de empaquetamiento muy común construida en XML)
* Transporte – incluye varias tecnologías que habilitan la comunicación directa,aplicación-aplicación. Estas tecnologías incluyen TCP, HTTP y SMTP
* Red – Es exactamente la misma que la capa de red del modelo de red TCP/IP. Proporciona comunicación, direccionamiento y enrutamiento
  1. Tipos de Servicios (SOAP, REST)

**SOAP**

SOAP - Simple Object Access Protocol – Que es una forma estandarizada de empaquetamiento para mensajes compartidos por aplicaciones. Son documentos XML

SOAP Envelop – Sobre

Header – Opcional – información relevante

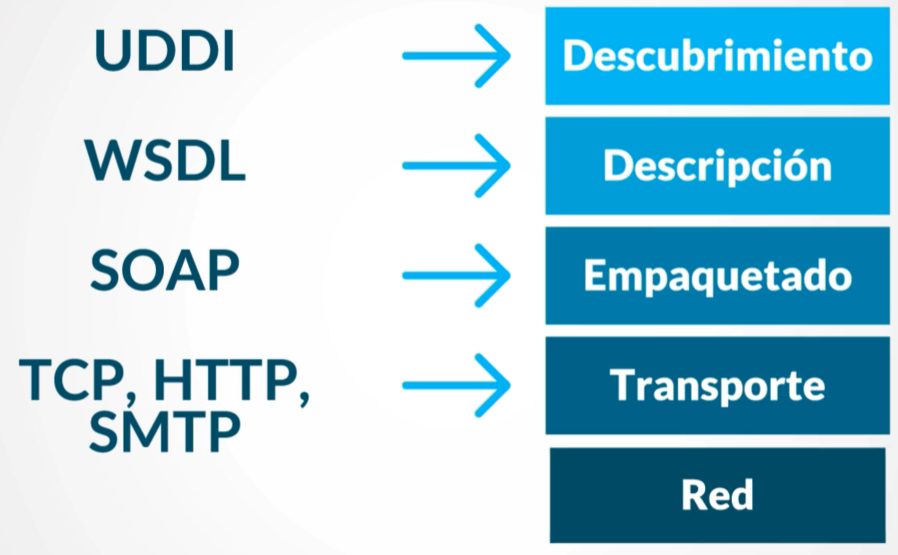
Body – Requerido (Cuerpo) – contiene información de la llamada y la respuesta

Fault – Contiene información de errores y estado

Una característica de los mensajes SOAP es que son independientes de los sistemas operativos y pueden ser transportados por los protocolos que funcionan en internet: SMTP, MIME, HTTP

Un mensaje en SOAP posee las siguientes características:

* Los mensajes deben ser transportados sobre SOAP
* La descripción del servicio debe ser hecha en WSDL
* Debe utilizar UDDI para el descubrimiento



**REST**

REST – Representational State Transfer – Estilo de arquitectura de software que define principios de diseño sobre cómo transmitir datos a través de una interfaz estandarizada como HTTP.

No es entendido como un estándar sino como un estilo de arquitectura

REST – no contiene una capa adicional de empaquetamiento del mensaje y se basa en recursos.

Un recurso – puede ser cualquier cosa expuesta en la web, desde un documento hasta un proceso de negocio. Es un objeto con tipo, datos asociados, relaciones con otros recursos. Similar un POO (Programación Orientada a Objetos), con la diferencia de que el recurso solo tiene unos métodos estándares definidos: HTTP GET, POST, PUT y DELETE

Todos los recursos deben ser asociados con una URI (Uniform Resource Identifier) – Cade de caracteres que identifica a un recurso por ubicación, nombre o ambos.

o La interacción con la API requiere del uso correcto de los métodos o verbos HTTP (GET, POST, PUT y DELETE)

permite diferentes tipos de formatos para representar recursos como texto plano HTML, XML y JSON.

Cuando deseas publicar una API compleja puedes usar SOAP

Cuando se requiere algo liviano y transacciones sencillas, cuando se va a hacer un servicio sin estado o se requiere mejorar el desempeño usando cache, sería más apropiado usar REST

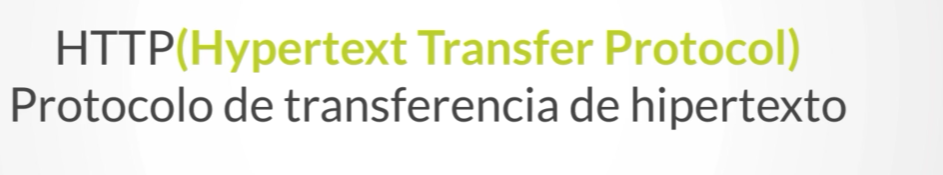
* 1. Actividad

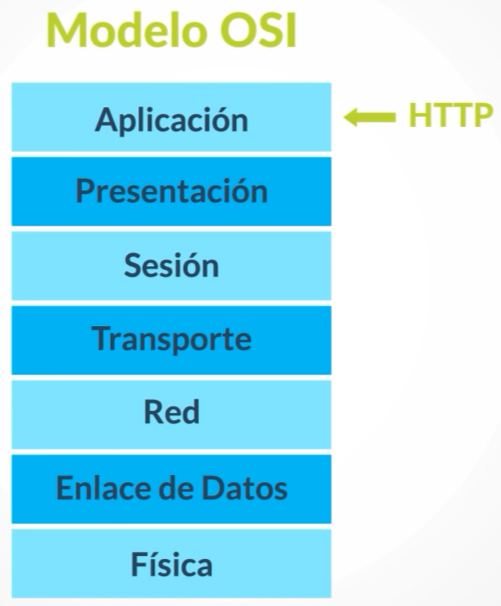
Tiempo de pensar

En los recursos de esta lección estudiamos el concepto de servicio web y su funcionamiento, así como explorar los tipos de servicio SOAP y REST. Es primordial identificar los escenarios en los que es recomendable usar un tipo de servicio u otro ya que esto permitirá un mejor rendimiento y aprovechamiento de los recursos en la API.  
  
Instrucciones:  
  
De las siguientes afirmaciones sobre los servicios web, selecciona tres que sean correctas.

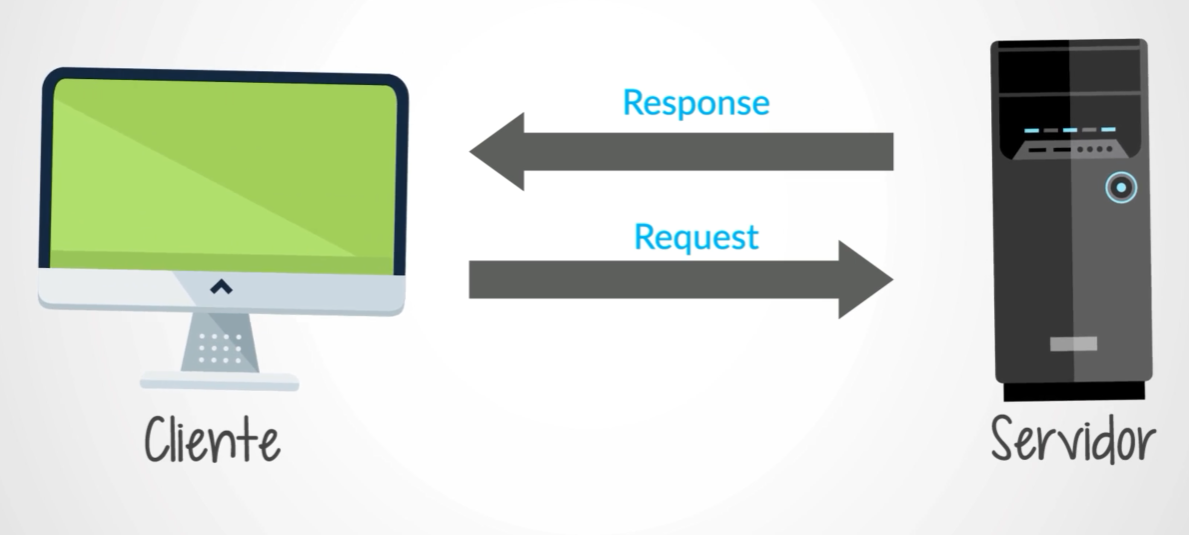


1. Lección 3: Estructura APIs REST
   1. ¿Qué es una petición HTTP?

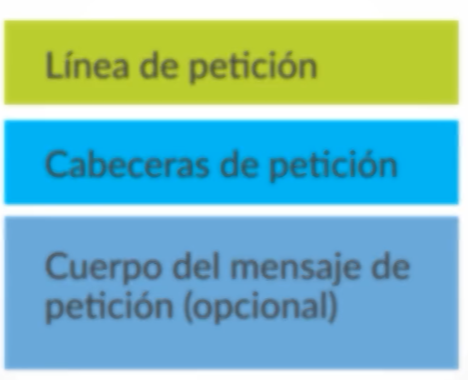




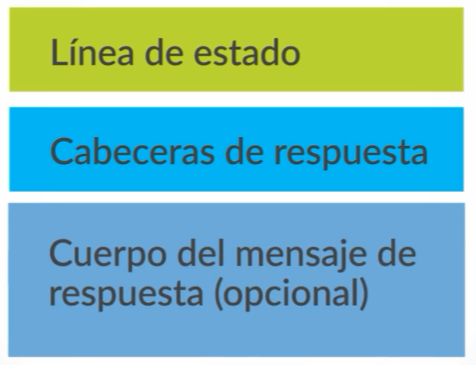




Una petición está formada por:



Una respuesta está formada por:



Línea de petición – (request-line). Tres campos

* Request-method-name 🡪 El nombre de petición
* Request-URI 🡪 identificador del recurso
* HTTP-version 🡪 La versión del HTTP

**Request-method-name**

El método le dice al servidor que tipo de acción el cliente quiere que tome (método es comúnmente denominado verbo). Existen varios métodos, los más comunes son: GET, POST,PUT y DELETE

GET – Se usa para devolver información de un método identificado con una URI, solo devuelven datos

POST- le solicita al servidor crear un nuevo recurso

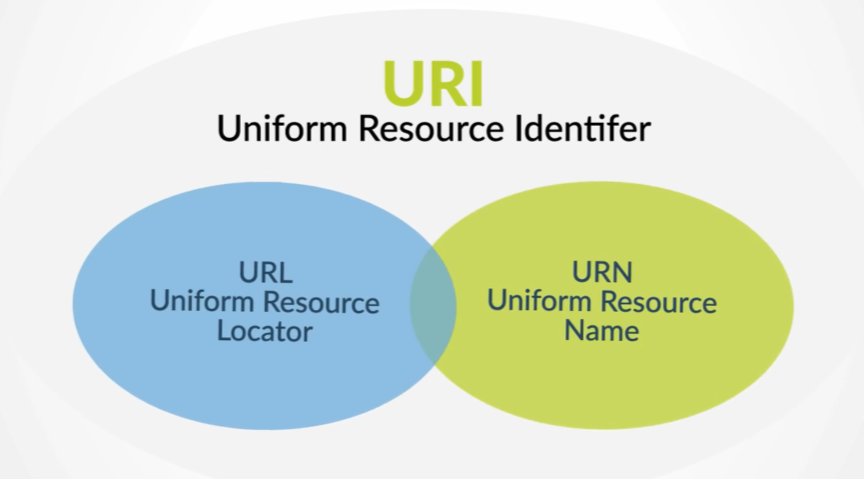
PUT- Para editar o actualizar un recurso se usa PUT

DELETE – este método le pide al servidor borrar un recurso

**Request-URI**

Recurso de identificador uniforme – que identifica al recurso al que se le hará la petición. Este URI se podrá especificar de forma absoluta o mediante una ruta relativa.

El URI está formado por



URL – especifica donde está disponible el recurso y el mecanismo de devolverlo.



**HTTP-version**

Especifica la versión del protocolo HTTP Usado

* HTTP/1.0
* HTTP/1.1

Ejemplo:



* 1. Cabeceras (Autorización, Autenticación)

Cabeceras son importantes en las peticiones y respuestas HTTP



El uso de las cabeceras enviadas en transacciones HTTP le dan al protocolo gran flexibilidad. Estos campos permiten enviar información descriptiva en una transacción permitiendo encriptación, autenticación o identificación de usuarios.

Las cabeceras son un bloque de datos que preceden a los datos sí se desean enviar.

Cabeceras 🡪 Metainformación, es información acerca de información

Las cabeceras son pares: Nombre-valor y pueden especificar múltiples valores separados por coma.



**Ejemplo de cabeceras de request:**

User-agent : Información del navegador, información del sistema operativo.

Accept-encoding: algoritmo de codificación admite 🡪 puede ser usado para incrementar el rendimiento con conexiones lentas, ya que permite la compresión

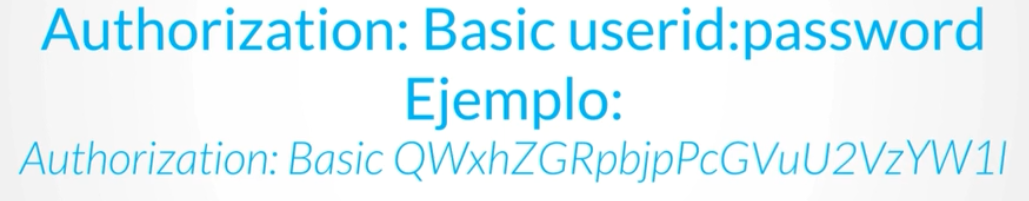
Accept-language: especifica el lenguaje que prefiere el cliente

Cookie: envía las cookies almacenadas para ese navegador, nombre=valor;nombre2=valor2;

Authorization: Proporciona la autorización del cliente para acceder datos en una URI (scheme credentials)

HTTP/1.0: describe la autorización basic:password – codificado en base64

Ej:



**Ejemplo de cabeceras de response:**

Server: contiene el nombre y el número de versión del servidor

Location: define la ubicación exacta del recurso que fue identificado con la URI de la petición. Es usada para redirigir al receptor a una ubicación diferente para completar la operación .

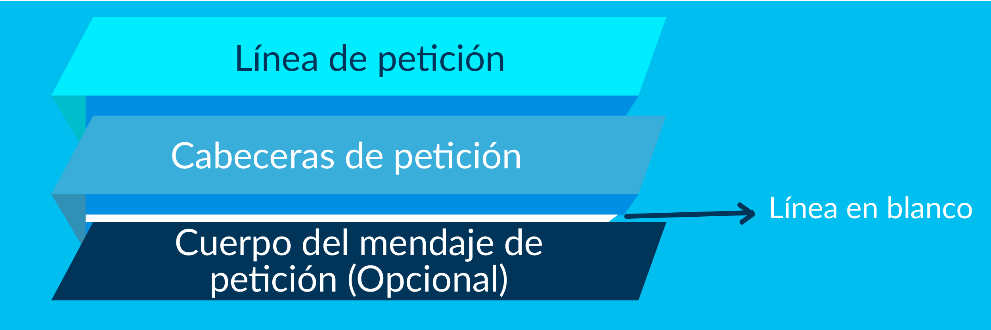
www-Authenticate: especifica el tipo de autenticación requerida:

Ej:



* 1. Cuerpo de la petición

Hemos estudiado que un mensaje de petición HTTP se compone de las partes que se ilustran a continuación:



El cuerpo del mensaje es una parte opcional en un mensaje HTTP. Se debe incluir una línea en blanco que separa las cabeceras del cuerpo. Usualmente se incluyen las cabeceras **content-type** y **content-length** que especifican la naturaleza del cuerpo asociado.  
El cuerpo del mensaje es el que lleva los datos de la petición que el cliente quiere enviar al servidor incluyendo los datos de formulario. A diferencia del método, la URL o las cabeceras donde el protocolo HTTP requiere una estructura rígida, el cuerpo permite al cliente enviar cualquier información que necesite.  
En el siguiente ejemplo muestra un mensaje de petición http con todas sus partes. Allí la URL **/prueba.php** será usada para procesar los datos enviados y por consiguiente una respuesta será enviada. Observa que la cabecera **content-type**le dice al servidor que los datos enviados es un texto XML y la cabecera indica content-length que indica el tamaño de los datos puestos en el cuerpo del mensaje, en este caso son 86 bytes.

POST /prueba.php HTTP/1.1

User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)

Host: www.example.com

Content-Type: text/xml; charset=utf-8

Content-Length: 86

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Connection: Keep-Alive

<?xml version="1.0" encoding="utf-8"?>

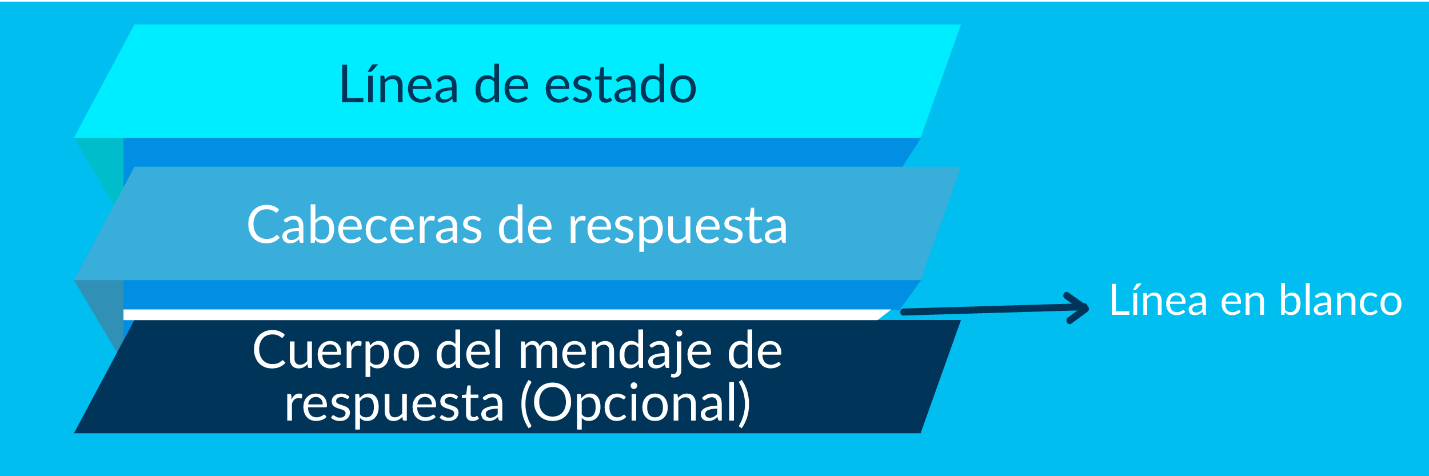
<string xmlns="http://test.com/">string</string >

En el ejemplo anterior podemos observar las partes de un mensaje que hemos estudiado hasta el momento:

* La primera línea: la línea de petición
* Cabeceras
* Una línea en blanco
* Cuerpo del mensaje
  1. Códigos de Estado HTTP

Códigos de estado HTTP

La siguiente figura ilustra los componentes de un mensaje de respuesta HTTP:



La primera línea del mensaje de respuesta es la de estado y contiene el código de estado de la respuesta, el cual es generado por el servidor para indicar el resultado de la petición.  
El siguiente ejemplo muestra un mensaje de respuesta:

HTTP/1.1 200 OK

Date: Mon, 27 Jul 2009 12:28:53 GMT

Server: Apache/2.2.14 (Win32)

Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT

Content-Length: 88

Content-Type: text/html

Connection: Closed

<html>

<body>

<h1>Hello, World!</h1<

</body>

</html>

En la línea de estado HTTP/1.1 200 OK observamos las siguientes partes:

* Versión HTTP: HTTP/1.1
* Codigo de estado: 200
* Comentario descriptivo de estado: OK

El código de estado es un número de 3 dígitos y cada uno tiene un significado único.  
Existen varios tipos de códigos, continuación un listado de ellos con algunos de los códigos de estado más usados:

**1. Informativos: 1xx**

Indica que la petición se ha recibido y el servidor está continuando con el proceso.  
100 Continue: El cliente debería continuar con su petición. El servidor debe enviar una respuesta final luego que la petición ha sido completada.  
101 Switching Protocols: El servidor acepta el cambio de protocolo propuesto por el navegador.

**2. Éxito: 2xx**

Este tipo de código de estado indica que la petición del cliente fue exitosamente recibida, entendida, aceptada y servida.  
200 OK: La petición ha sido completada exitosamente. Es el código de estado estándar para peticiones exitosas.  
201 Created: La petición ha sido completada y resultó en la creación de un nuevo recurso.  
204 No Content: El servidor ha completado la petición con éxito pero no necesita enviar cuerpo en el mensaje de respuesta. La respuesta 204 no debe incluir cuerpo del mensaje y por lo tanto debe finalizar con la línea en vacía después de las cabeceras.

**3. Redireccionamiento: 3xx**

Indica que el cliente debe ejecutar una acción adicional para poder completar la petición.  
301 Move Permanently: El recurso pedido ha sido movido permanentemente a una nueva ubicación. La url de la nueva ubicación está dada por la cabecera de respuesta llamada Location.  
02 Found & Redirect (Move Temporarily): El recurso solicitado ha sido movido a una nueva ubicación pero sólo por tiempo limitado.  
304 Not Modified: El servidor notifica que el recurso pedido no ha sido modificado. Por lo general, el cliente hace una petición con una cabecera como If-Modified-Since para indicarle al servidor la fecha con la cual comparar. Un mensaje de respuesta con código 304 no debe contener cuerpo.

**4. Error del cliente: 4xx**

Este tipo de código de estado se utiliza para casos cuando al parecer el cliente ha errado en la petición.  
400 Bad Request: El servidor no puede interpretar o entender la petición.  
401 Unauthorized: El recurso solicitado está protegido, y requiere de las credenciales cliente. El cliente debe volver a hacer la petición con sus credenciales (usuario/contraseña).  
403 Forbidden: El servidor no puede proveer el recurso solicitado independientemente de la identidad del cliente.  
404 Not Found: El recurso solicitado no pudo ser encontrado en el servidor.  
409 Conflict: La petición no se ha podido completar porque se ha producido un conflicto con el recurso solicitado. El caso más habitual es el de las peticiones de tipo PUT que intentan modificar un recurso que a su vez ya ha sido modificado por otro lado.

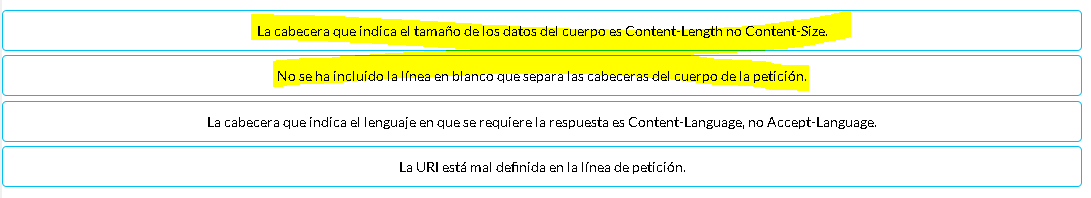
**5. Error del servidor: 5xx**

Indica que el servidor ha errado o es incapaz de realizar una petición aparentemente válida.  
500 Internal Server Error: El servidor encontró una condición inesperada que le impidió completar la petición.  
502 Bad Gateway: El servidor actuando como gateway o proxy, recibió una respuesta inválida del servidor de flujo ascendente(upstream server).  
504 Gateway Timeout: El servidor actuando como gateway o proxy, no recibió a tiempo respuesta del servidor de flujo ascendente(upstream server).

* 1. Actividad

 Tiempo de pensar

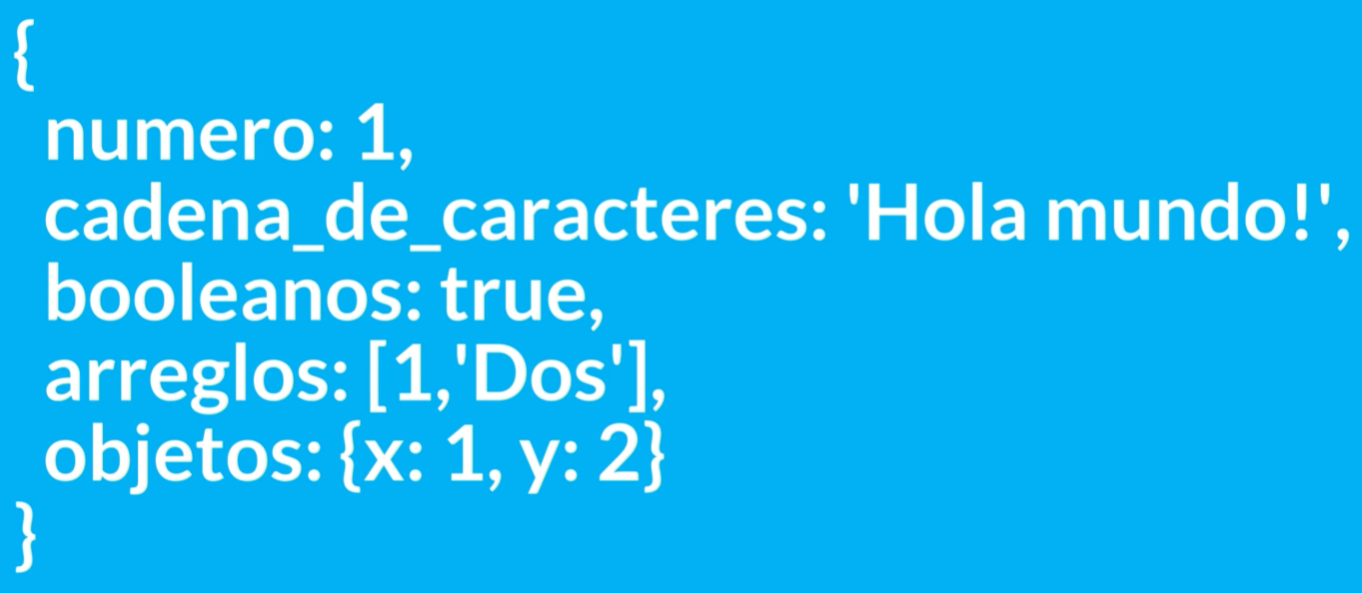
En esta lección comprendimos el concepto de API REST, profundizando en las peticiones HTTP, su estructura y sintaxis. Dicha estructura es muy importante puesto que permite que los sistemas cliente y servidor, en una interacción con una API REST, se comuniquen e intercambien la información necesaria.  
  
Instrucciones:  
  
Observa con mucha atención la siguiente petición HTTP. Luego identifica dos errores en la definición de la petición y elige las opciones correctas:  
  
GET /docs/prueba.html HTTP/1.1  
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)  
Host: www.example.com  
Content-Type: application/x-www-form-urlencoded  
Content-Size:11  
Accept-Language: en-us  
Accept-Encoding: gzip, deflate  
Connection: Keep-Alive  
IdLibro=678&area=ciencias



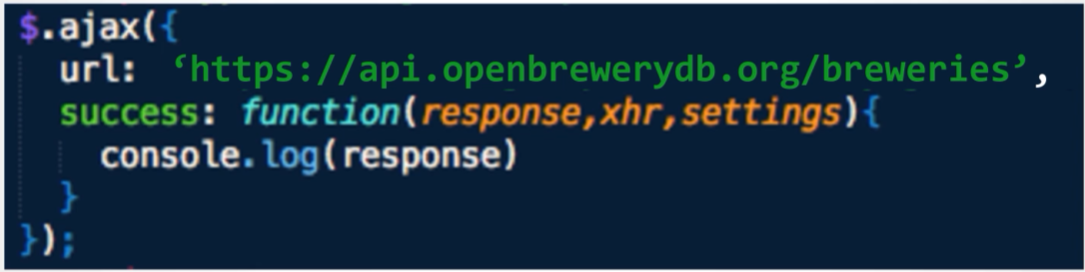
1. Lección 4: Consumir APIs externos
   1. Consumiendo un JSON Externo

JSON y AJAX

Formato JSON – Se forma con llave-valor(key-value) separados por dos puntos



AJAX – Para interactuar con APIs externos normalmente trabajan con JSON



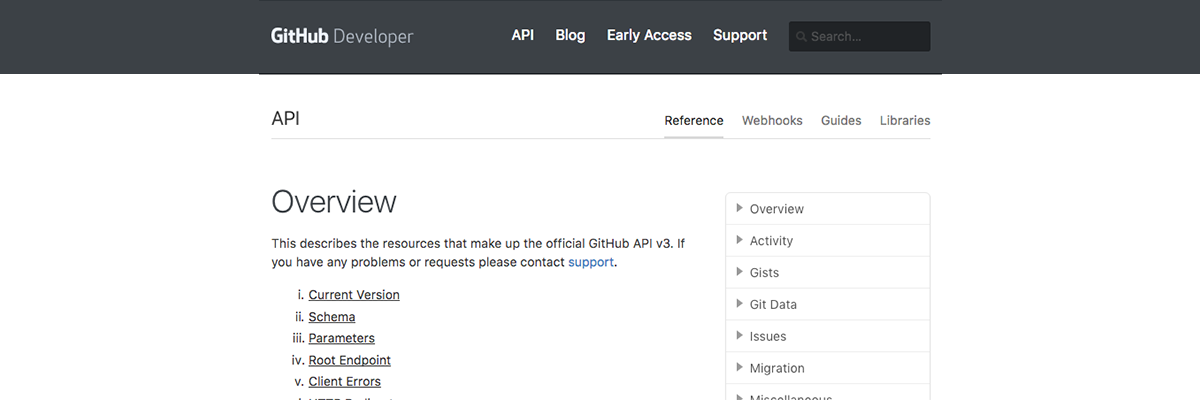
Si es cadena de caracteres podemos usar la función nativa, para convertirlo a un objeto



* 1. Consumiendo la API de Github

Consumo de API de GitHub

Consumir Web APIs es común en las aplicaciones web. En el tema de hoy vamos a estudiar como consumir el API de Github.  
  
El concepto de real time o tiempo real se basa en usar un conjunto de tecnologías y herramientas que nos permitan abrir un canal de comunicación entre el cliente y el servidor y que éste se mantenga abierto en todo momento, entonces no tenemos que preocuparnos por estar preguntándole al servidor si tiene nueva información que entregarnos, sino que cuando el servidor tiene nuevos datos se los notifica al cliente y viceversa.  
  
La documentación oficial la encuentras en este sitio web: <https://developer.github.com/v3/>  
  
Sin embargo te recomendamos leer una página guía que ayuda mucho a entender mejor la API: <https://developer.github.com/guides/getting-started/>  
  
En el tema anterior estudiamos cómo consumir un JSON externo mediante AJAX. Es muy útil saberlo porque el API de Github funciona con JSON.



**URL de API de github**

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



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



Encontramos que dentro del objeto responseJSON se encuentra la información relevante a la API.  
  
Podríamos encontrar términos que no conozcamos y eso es bueno, así nos familiarizamos con Github. Pero de los que si reconocemos serían: usuarios, repositorios u organizaciones.



**Mostrar un usuario**

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



**Obtener repositorios de un usuario**

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



Obtenemos un arreglo de los repositorios.

**Autenticación**

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



La función nativa de javascript ‘btoa’ convierte a base64 una cadena de caracteres.  
El código completo sería así:



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



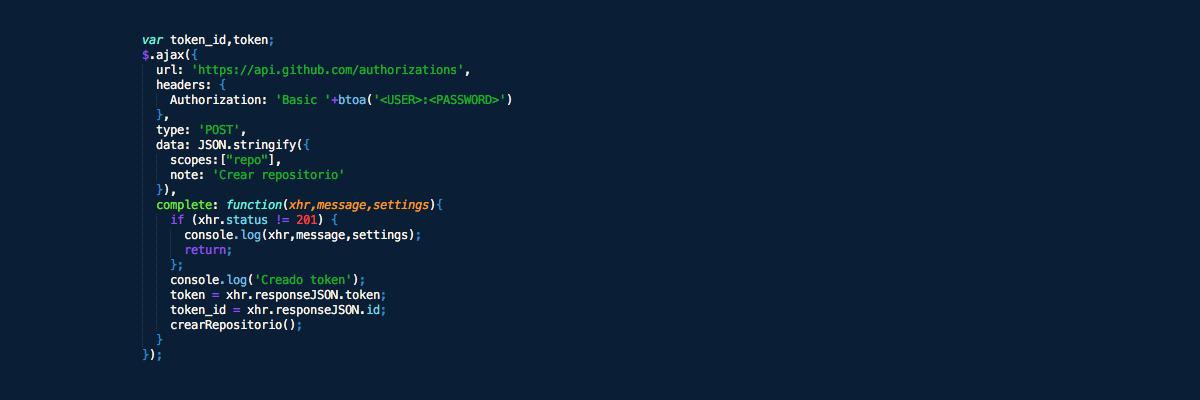
Por ejemplo encontramos los parámetros:

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

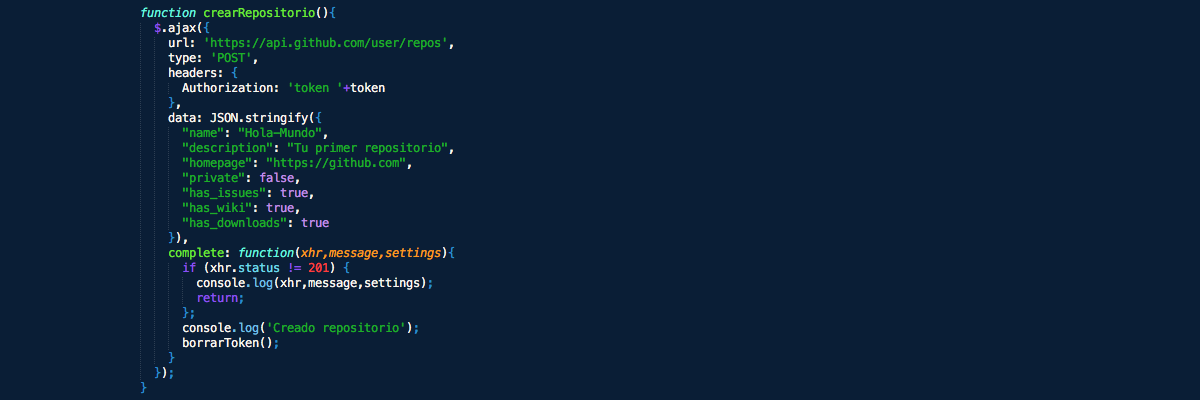
Los parámetros que inician con X- son exclusivos de github. Vemos que X-RateLimit-Limit tiene asignado el valor 5000, lo que muestra que si nos autenticamos correctamente.

**Crear un repositorio**

Ahora que ya sabemos autenticarnos, podemos hacer operaciones más complejas, por ejemplo, creemos un repositorio.  
  
Para crear un repositorio necesitamos hacer una operación extra. Por motivos de seguridad, es necesario otorgar permisos a un usuario para poder hacer determinadas operaciones utilizando la API, Github provee el uso de tokens para hacer estas operaciones, de tal modo que cada token puede o no hacerlas.  
  
Primero debemos solicitar crear el token, por favor analiza el código:



Lo primero que observamos es que la url usada es de autorizaciones, la primera en la lista cuando hicimos un llamado simple a la API. Seguidamente encontramos los headers de autenticación y luego encontramos el siguiente cambio. El parámetro type está definido como POST, esto se debe a que vamos a solicitar crear un token. Finalmente encontramos el parámetro data y es MUY IMPORTANTE que notes que el contenido de data es dentro de la función JSON.stringify, esto se debe a que la API solo funciona con JSON, por lo tanto, debes hacer este procedimiento para que la API entienda las instrucciones. Dentro de data encontramos el parámetro scope, es un array que contiene los permisos que se van a otorgar, asignamos simplemente ‘repo’. El parámetro note es opcional.  
  
A continuación encontramos en la función complete una validación, recuerda que el status 201 significa que el recurso fue creado, por lo tanto, si este no es el status, no podemos continuar.  
  
Habrás notado que en la parte superior definimos unas variables globales token y token\_id, ahora observa que sus valores son definidos dentro del complete del ajax, se usarán más adelante. Seguidamente encontramos la función crearRepositorio:



Primero, la URL, esta al encontramos en la documentación de Github: [https://developer.github.com/v3/repos/#create.](https://developer.github.com/v3/repos/#create) A continuación, el type, nuevamente POST, porque vamos a crear un repositorio. Sin embargo, ahora la autenticación es diferente, es mediante token, ¿cuál token? El token que obtuvimos cómo respuesta en el llamado anterior. Seguidamente de la información, recordemos que debe estar cómo JSON y para eso se usa la función JSON.stringify. Finalmente, el complete, al igual que en el llamado ajax anterior, es necesario validar el status 201.  
  
Cómo último paso luego de crear el repositorio y cómo ejercicio de este tema, borramos el token que creamos llamado la función borrarToken:



Lo más notorio de este llamado ajax es la URL, observemos que cuenta con el token\_id obtenido en respuesta de la primera función y el status de la función complete que es 204, que significa No Content.  
  
Esperamos que con estos ejemplos te sientas libre de seguir explorando las demás operaciones que provee la API de Github. Nuevamente te recomendamos leer la guía que provee la página <https://developer.github.com/guides/getting-started/> y nos vemos en el próximo tema. ¡Hasta pronto!

* 1. Actividad

 Tiempo de pensar

Gracias a los recursos que hemos estudiado conocimos cómo consumir APIs externas mediante peticiones Ajax, gestionando el envío de peticiones a una URL específica y definiendo acciones al recibir la respuesta del servidor de la API.  
  
Instrucciones:  
  
A continuación verás el código de una petición Ajax realizada a la API de Github, con el propósito de crear un nuevo repositorio. Sin embargo, existe un error en el código que debes identificar y seleccionar el número de línea en que se encuentra dicho error.

JavaScript

$.ajax({

url: 'https://api.github.com/authorizations',

headers: {

Authorization: 'Basic '+btoa(':')

},

type: 'GET',

data: JSON.stringify({

scopes:["repo"],

note: 'Crear repositorio'

}),

complete: function(xhr,message,settings){

if(xhr.status != 204){

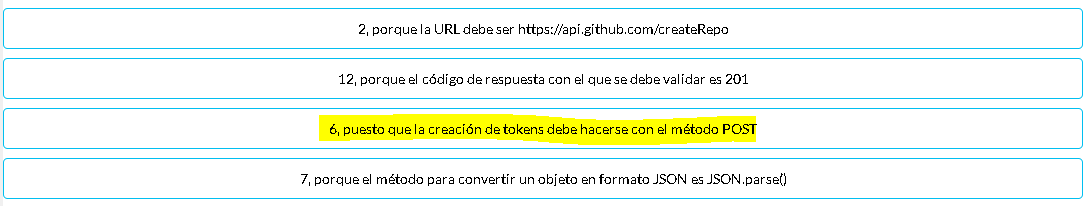
console.log(xhr, message, settings);

return;

};

}

});



* 1. Ejercicio de Codificación

HORA DE  
codificar

30 MINUTOS

Una API es una herramienta que facilita en gran medida el desarrollo de sistemas de información, puesto que establece un medio de comunicación adecuado para la integración de diversas partes de un sistema o incluso de varios proyectos entre sí.  
  
Existen gran cantidad de APIs libres de uso para que los desarrolladores puedan hacer uso de la información que proveen. Sin embargo, en la gran mayoría de sitios que proveen este tipo de APIs, requieren un proceso de autenticación con el fin de identificar quién está haciendo uso de su API y verificar que sea para propósitos aceptables y no para efectos de ataques o violación del reglamento propio de cada portal web.  
  
En este ejercicio debes consumir la API de Marvel, la cual contiene información sobre personajes de comics. Para hacerlo debes seguir las instrucciones del sitio oficial de la API para cumplir con todos los requisitos de autenticación y así acceder a ella dentro de una página web propia, en la que se mostrará la información.

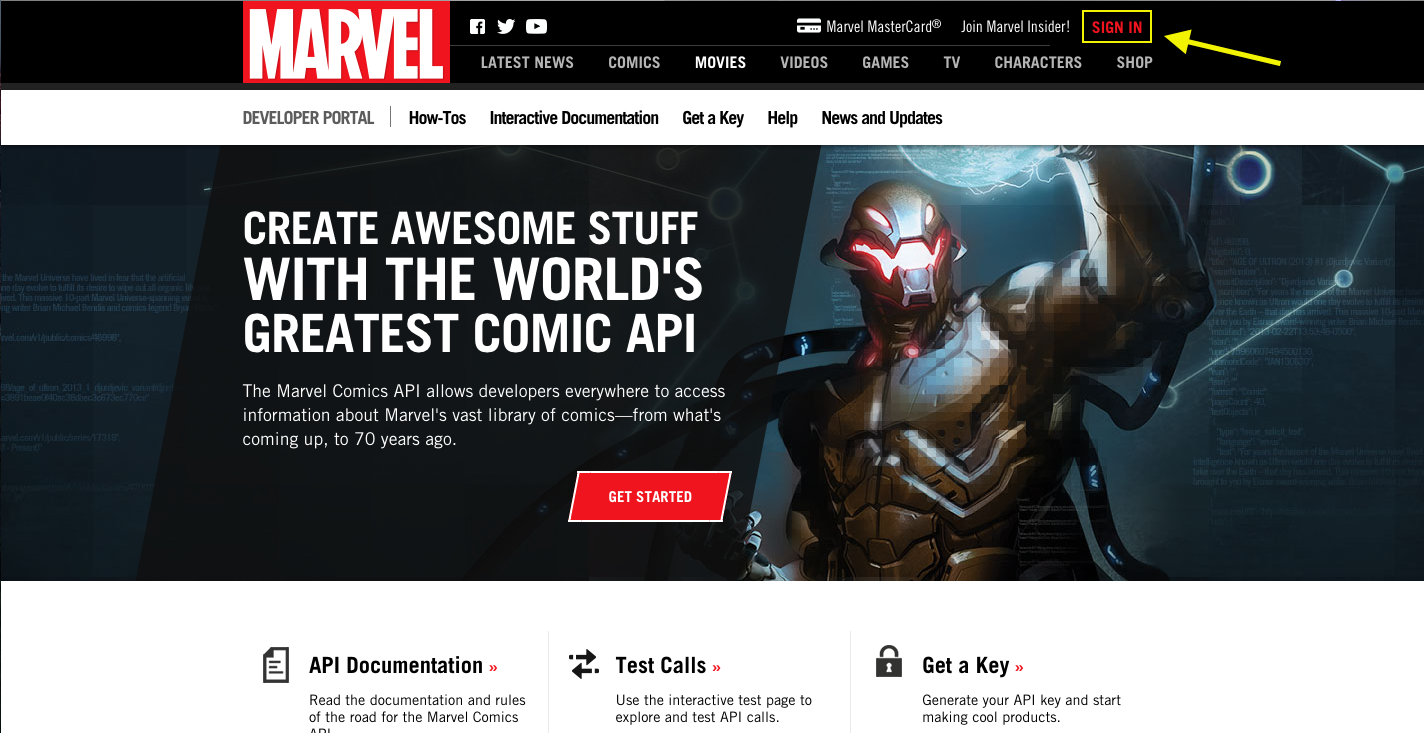
**Contenido**

Para el desarrollo de este ejercicio debes descargar el archivo base llamado codBase.zip. En él encontrarás una página HTML con las bases gráficas del sistema. A partir de los elementos que allí se encuentran, debes seguir las siguientes instrucciones para completar el ejercicio.

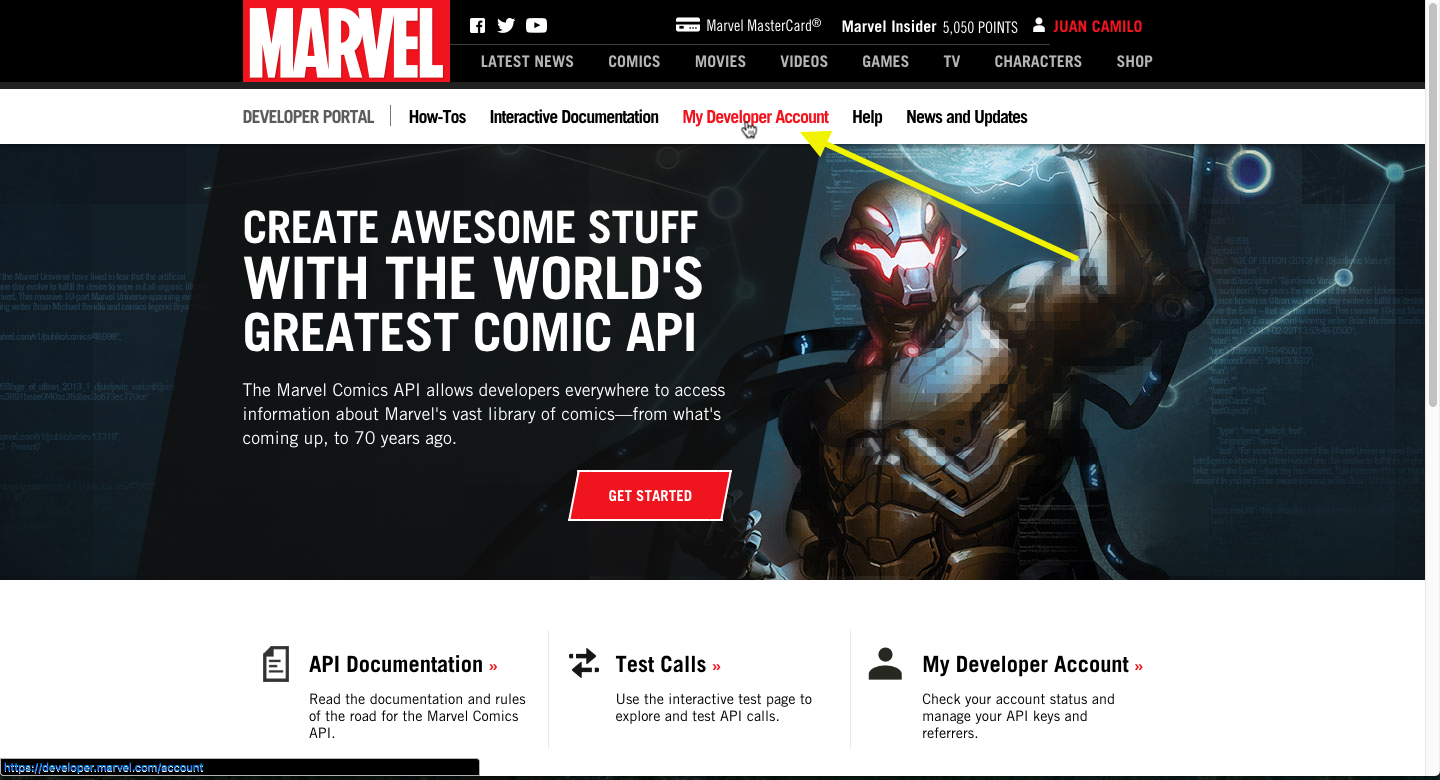
**Instrucciones**

Ingresa a la página oficial de la API en la siguiente url: <http://developer.marvel.com>

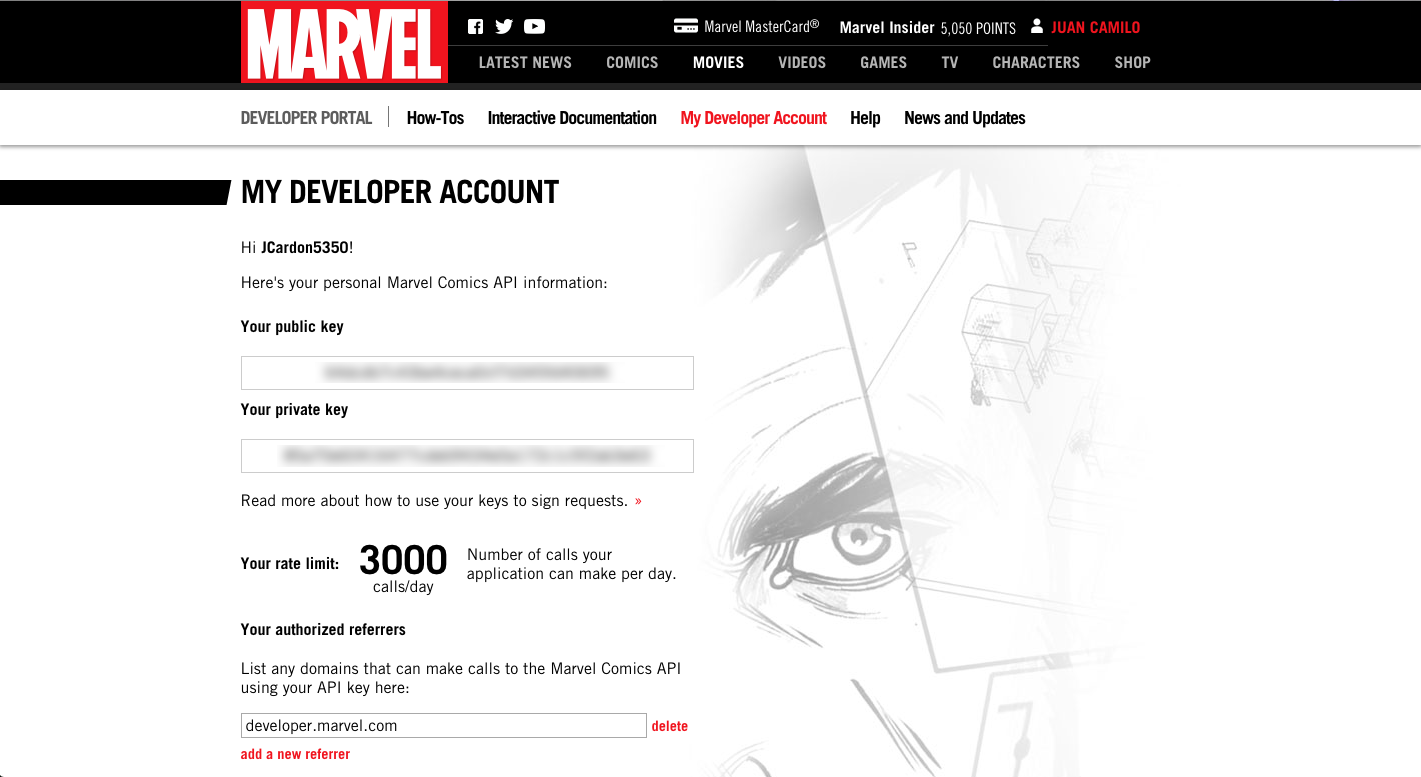
Crea una nueva cuenta en este sitio web. Lo puedes hacer con tu cuenta de Facebook, Google o con un correo electrónico y un nombre de usuario. Luego de aceptar los términos y condiciones de la página debes revisar la bandeja de entrada del correo con el cual afiliaste tu cuenta y hacer click en el enlace de verificación.



Ingresa nuevamente a la página www.developer.marvel.com y después de iniciar sesión con la cuenta recién creada, hacer click en la sección “My Developer Account”.



En esta página puedes observar dos contraseñas o llaves que son asignadas a tu usuario. La primera es una llave pública y la segunda privada. De acuerdo con las instrucciones de la página debes mantener estos datos reservados y no publicarlos o compartirlos.



Crea el hash de la petición al API. Para esto ingresa a la página http://7thspace.com/webmaster\_tools/online\_md5\_encoder.html la cual es un codificador md5 (puedes usar la página indicada o cualquier otro sistema que codifique en md5). La cadena de caracteres que debes codificar debe ser un valor de TimeStamp o valor ts que puede ser un número que elijas al azar; sin embargo debes recordarlo porque más adelante debes usar el mismo número. A continuación la llave privada y por último la llave pública. Ejemplo:  
  
Si ts = 3, llave privada = abcd y llave pública = 1234; la cadena a codificar sería: 3abcd1234. Una vez codificada esta cadena, el valor obtenido corresponde al valor hash que usaremos en próximas instrucciones.

Crea en el archivo app.js una función que realice una petición ajax al enviarse el formulario en index.html, consultando en la API el personaje que coincida con el nombre ingresado en el campo de texto. Dicha petición debe ser de tipo ‘GET’ y la URL de la petición se debe componer de la URL base, los directorios o lugares que se van a acceder, luego las variables de autenticación ts, llave pública (apikey) y hash, y por último el nombre del personaje a buscar con el parámetro name; de la siguiente manera.

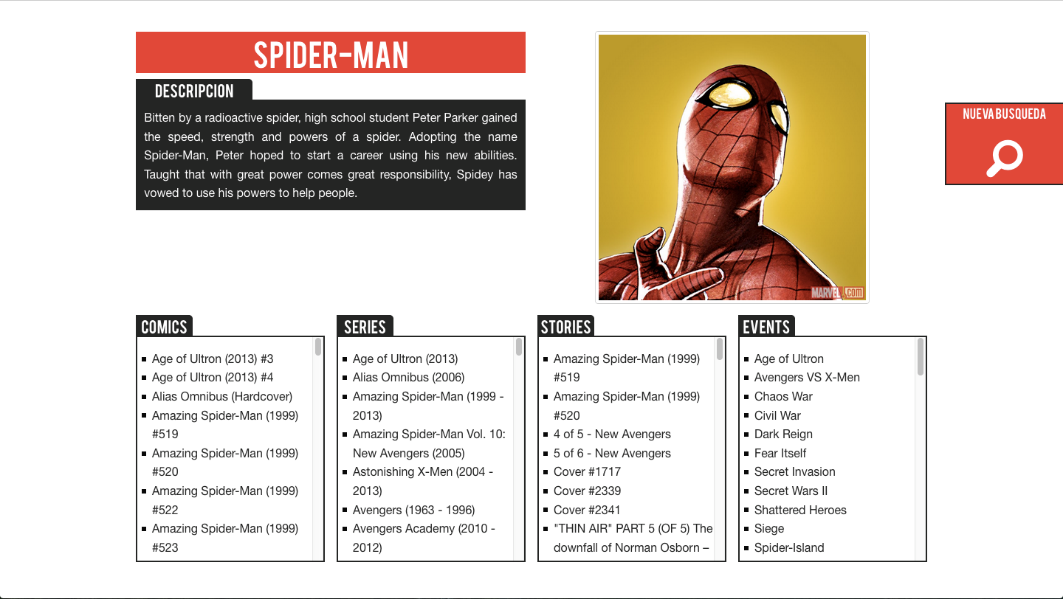
URL base→ https://gateway.marvel.com

Directorio de personajes → /v1/public/characters

Variables de autenticación → ts=1, apikey=1234, hash=iu3424u23

Personaje a buscar → name=Spider-Man

Entiende la estructura del JSON que envía la API como respuesta y actualiza, usando la función “appedHTML” incluida en el código base, los campos de la página con el nombre del personaje en el elemento de la clase nombre; la descripción en el elemento con el id descripción; la imágen thumbnail en el elemento con el id image; llenar las listas con los ids comics, series, stories y events con los nombres de los valores correspondientes en la respuesta para dicho personaje.



**Buenas Prácticas**

Cada uno de los portales web que ofrecen APIs de libre acceso, cuentan con un procedimiento particular para la autenticación de los usuarios que van a consumir dicha API. Por lo tanto es de vital importancia que antes de realizar cualquier petición, leas exhaustivamente la documentación ofrecida por los creadores de la API, para obtener los resultados esperados de la operación.

1. UNIDAD 2: Prueba

|  |  |
| --- | --- |
| **Comenzado en** | Friday, 18 de September de 2020, 12:11 |
| **Estado** | Terminados |
| **Finalizado en** | Friday, 18 de September de 2020, 12:22 |
| **Tiempo empleado** | 10 mins 27 segundos |
| **Puntos** | 9/10 |
| **Calificación** | **90** de un total de 100 |
| **Comentario de retroalimentación** | Buen trabajo! |

Top of Form

### **Pregunta**1

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Qué grupo de códigos de cabecera indican éxito en la petición?

Seleccione una:

a. 1XX

b. 2XX

c. 3XX

d. 4XX

#### Retroalimentación

Los códigos de cabecera que retorna el servidor cuando la petición fue exitosa empiezan con el número 2.

La respuesta correcta es: 2XX

### **Pregunta**2

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Cuál de las siguientes es un tipo de arquitectura orientada a los servicios?

Seleccione una:

a. SOAP

b. API

c. REST

d. SOA

#### Retroalimentación

La arquitectura orientada a servicios o Service Oriented Architecture es conocida también como SOA.

La respuesta correcta es: SOA

### **Pregunta**3

Incorrecta

Puntúa 0 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Qué código de respuesta se obtiene al ejecutar el siguiente código? (Reemplazando USER y PASSWORD por valores válidos):

$.ajax({

url: ‘https://api.github.com/authorizations’,

headers: {

Authorization: ‘Basic’ +btoa(‘<USER>:<PASSWORD>’)

},

type: ‘POST’,

data: {

scopes:[“repo”],

note: ‘Crear repositorio’

},

Complete: function(xhr,message,settings){

console.log(xhr.status)

}

})};

Seleccione una:

a. 200

b. 201

c. 204

d. 400

#### Retroalimentación

Se obtiene un código de respuesta 400: Bad Request: La API de Github solo acepta formato JSON y la data no se envió en este formato por lo tanto el servidor no puede entender la petición

La respuesta correcta es: 400

### **Pregunta**4

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Selecciona el término que complete correctamente la siguiente afirmación: \_\_\_\_\_\_ se utiliza para describir los recursos consumibles de un Web Service.

Seleccione una:

a. XML

b. REST

c. API

d. WSDL

#### Retroalimentación

La descripción de los recursos consumibles de un web service está a cargo del Web Service Description Language.

La respuesta correcta es: WSDL

### **Pregunta**5

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Qué tecnología funciona como un intermediario entre dos componentes de software?

Seleccione una:

a. SOA

b. API

c. JSON

d. SOAP

#### Retroalimentación

Una Application Programing Interface sirve para interconectar sistemas de información facilitando el transporte de datos y haciendo más fácil la interacción entre varias partes de la arquitectura de un producto de software.

La respuesta correcta es: API

### **Pregunta**6

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Cómo se diferencian las cabeceras de petición del cuerpo del mensaje?

Seleccione una:

a. Una línea de estado

b. Un prefijo que las identifica

c. Una línea en blanco

d. Una línea de petición

#### Retroalimentación

Para iniciar el cuerpo del mensaje HTTP se debe primero dejar una línea en blanco después de las cabeceras de la petición.

La respuesta correcta es: Una línea en blanco

### **Pregunta**7

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Indica si la siguiente afirmación es verdadera o falsa: El URI en una línea de petición se podrá especificar de forma absoluta o mediante una ruta relativa.

Seleccione una:

a. Verdadero

b. Falso

#### Retroalimentación

En una línea de petición el URI puede ser una ruta absoluta o relativa

La respuesta correcta es: Verdadero

### **Pregunta**8

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Indica si la siguiente afirmación es falsa o verdadera: Todas las APIs cuentan con el mismo sistema de autenticación.

Seleccione una:

a. Falso

b. Verdadero

#### Retroalimentación

Cada API cuenta con su sistema de autenticación por medio de llaves, claves, usuarios registrados, etc.

La respuesta correcta es: Falso

### **Pregunta**9

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

¿Qué se debe hacer antes de consultar una API y desarrollar el código que muestre los resultados en una página web?

Seleccione una:

a. Comprender la estructura de la respuesta recibida del servidor, para acceder a sus datos adecuadamente.

b. Definir los tipos de datos que se van a mostrar en la página para mostrar adecuadamente números o texto.

c. Buscar una API que muestre la información en tu página web automáticamente

d. No es posible mostrar datos en una página web a partir del consumo de una API

#### Retroalimentación

HyperText Transfer Protocol es uno de los protocolos más usados en la actualidad.

La respuesta correcta es: Comprender la estructura de la respuesta recibida del servidor, para acceder a sus datos adecuadamente.

### **Pregunta**10

Correcta

Puntúa 1 sobre 1

Señalar con bandera la pregunta

#### Texto de la pregunta

Estás trabajando en una aplicación y el browser te muestra un error que inicia por 5, sin saber qué es exactamente ya sabes que el error tiene que ver con:

Seleccione una:

a. Web services

b. Tu código Javascript

c. El servidor

d. Redireccionamiento

#### Retroalimentación

Los códigos de error del servidor inician por 5

La respuesta correcta es: El servidor

Bottom of Form