Cisco Networking Academy

Cisco Devnet

Módulo #4: Comprensión y Uso de un API

Autor:Paco Aldarias
francisco.aldarias@aulamentor.es

Fecha:
18-10-2021

¿Qué aprenderá en este módulo?

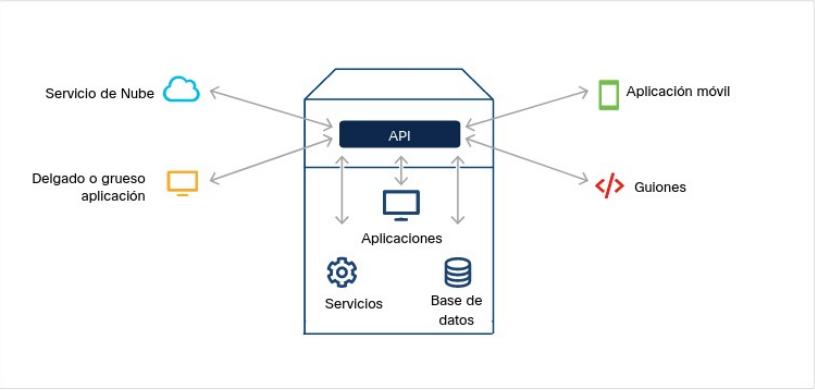
Título del módulo: Comprender y usar API Objetivo del

módulo: Crear solicitudes de API REST a través de HTTPS para integrar servicios de forma segura.

Título del tema	Objetivo del tema
Introducir las APIs	Explique el uso de API.
Estilos de Diseño de la API	Compare los estilos de diseño síncronos y asíncronos de las API.
Estilo de Arquitectura de la API	Describa los estilos de arquitectura de API comunes.
Introducción a la API REST	Explique las funciones de las API de REST.
Autenticando una API REST	Cree solicitudes de API de REST sobre HTTPS para integrar servicios de manera segura.
Límites de tasa de API	Explique el propósito de los límites de velocidad de las API.
Trabajar con Webhooks	Explique el uso de Webhooks.
Solución de problemas de llamadas a la API	Explique cómo solucionar problemas de API de REST.

1.- Introducción

Ejemplo de diferentes tipos de integraciones de API.



Una API es una interfaz de programación de aplicaciones (del inglés API: Application Programming Interface).

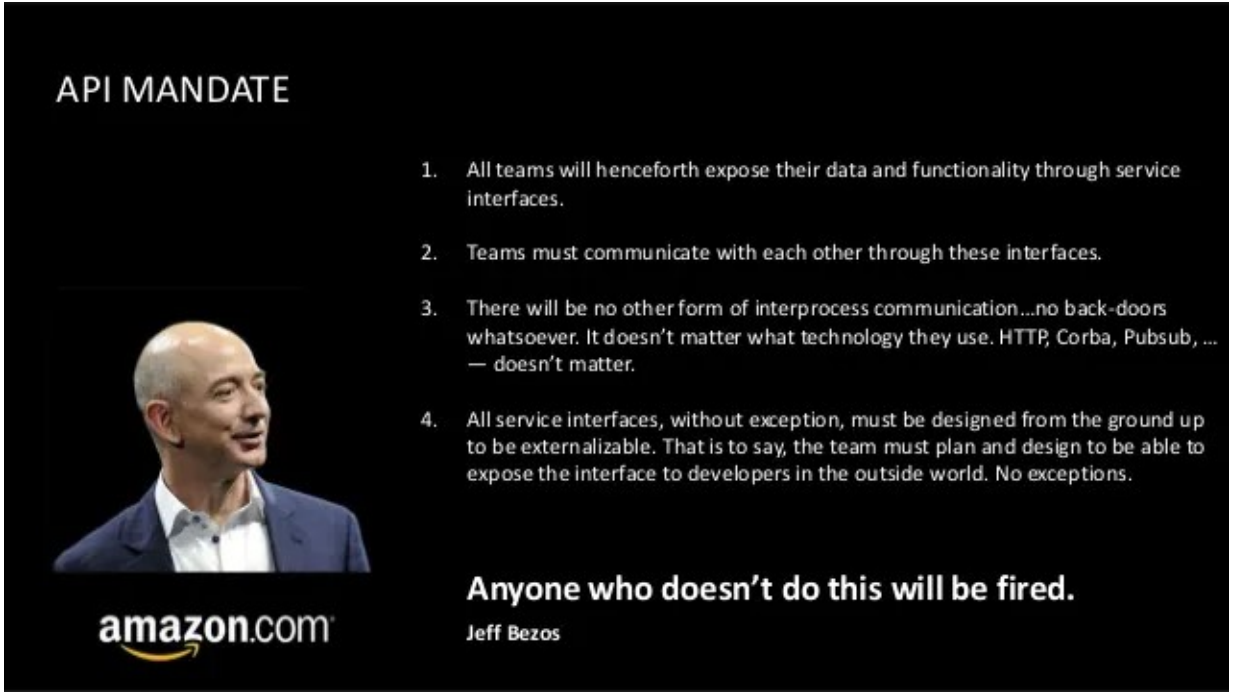
Una API permite que una pieza de software hable con otra. Una API es análoga a una toma de corriente.

Una API puede usar interacciones basadas en web o protocolos de comunicación comunes, y también puede usar sus propios estándares patentados.

Ejemplo API: Google

Ejemplo http://google.com/search?q=devnet

1.- Introducción



Henceforth: de ahora en adelante
Whatever: cualquier cosa
From ground up: desde cero.
Fired: despedido

1.- Introducción

1.2 Porque usar API

Las API generalmente se construyen para ser consumidas mediante programación por otras aplicaciones, por lo que se llaman interfaces de programación de aplicaciones. Pero no hay veces que puedes ser usadas por humanos. Casos uso:

- Tareas automatizadas que simplifican el trabajo por ejemplo conjunto de tareas a realizar a final de mes en una empresa.
- Integrar datos de una aplicación con otra como por ejemplo servicios de pago.
- Integrar funcionalidades entre apps diferentes. Ejemplo optimizar viajes entre uber y google maps.

Si conoces algún API WEB interesante ponla en el foro del tema

1.- Introducción

1.2 Porque usar API. Mashup (aplicación híbrida)

- En desarrollo web, una mashup es una forma de integración y reutilización.
- Ocurre cuando una aplicación web es usada o llamada desde otra aplicación, con el fin de reutilizar su contenido o funcionalidad.
- El uso en otra(s) fuente(s), para crear nuevos servicios simples, visualizado en una única interfaz gráfica diferente. Por ejemplo, se pueden combinar las direcciones y fotografías de las ramas de una biblioteca con un mapa de Google Maps

2.- Estilo de diseño de API

Las API se pueden entregar de una de dos maneras: sincrónicamente o asincrónicamente.

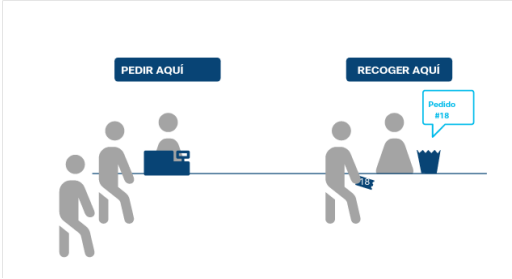
Necesita saber la diferencia entre los dos, porque la aplicación que consume la API administra la respuesta de manera diferente dependiendo del diseño de la API.

Las API **sincronas** generalmente están diseñadas para ser sincrónicas cuando los datos de la solicitud están disponibles fácilmente, como cuando los datos se almacenan en una base de datos o en la memoria interna. El servidor puede obtener instantáneamente estos datos y responder de inmediato.

Las API **asíncronas** proporcionan una respuesta para indicar que la solicitud se ha recibido, pero esa respuesta no tiene ningún dato real. El servidor procesa la solicitud, que puede llevar tiempo, y envía una notificación (o desencadena una devolución de llamada) con los datos después de que se haya procesado la solicitud

APIs Sincrónicas

APIs Asincrónicas



3.- Estilos de Arquitecturas de API

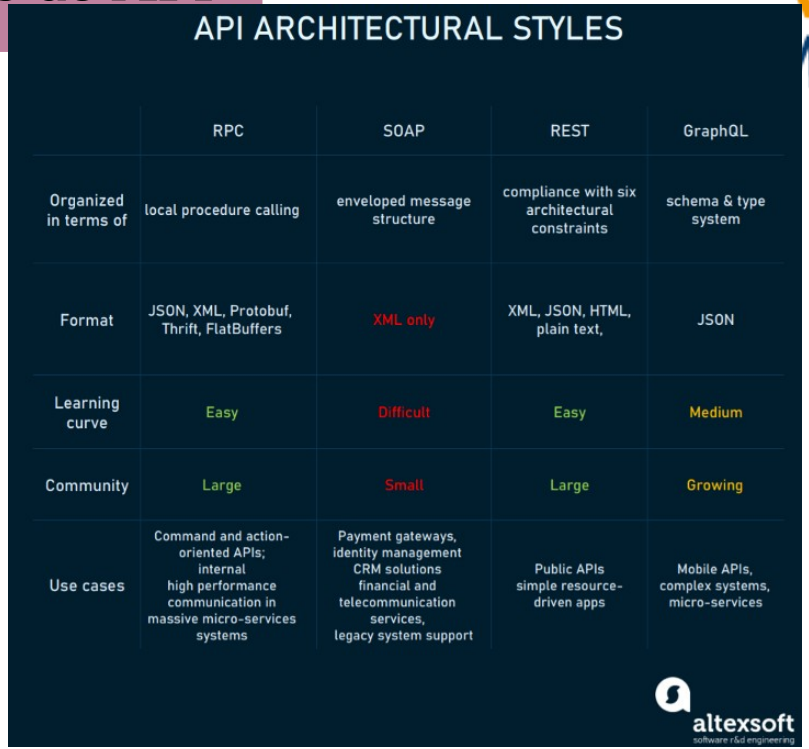
La aplicación define cómo interactúan terceros con ella, lo que significa que no hay una forma «estándar» de crear una API.

Sin embargo, aunque una aplicación técnicamente puede exponer una interfaz al azar, la mejor práctica es seguir estándares, protocolos y estilos arquitectónicos específicos.

Esto hace que sea mucho más fácil para los consumidores de la API aprender y entender la API, porque los conceptos ya estarán familiarizados.

Los tres tipos más populares de estilos arquitectónicos de API son:

- RPC,
- SOAP
- REST



CISCO DevNet Associate v1.0

3.- Arquitecturas de API

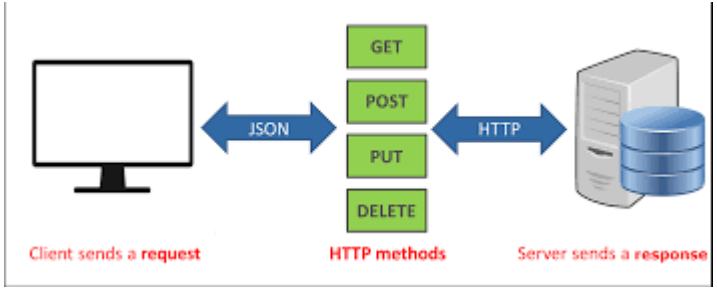
3.4 REST

La transferencia de estado representacional (en inglés representational state transfer) o REST es un estilo de arquitectura software para sistemas hipermedia distribuidos como la World Wide Web

Restricciones:

1. Cliente-Servidor
2. Sin estado
3. Caché
4. Interfaz uniforme
5. Sistema por capas
6. Código bajo demanda

Estas seis restricciones se pueden aplicar a cualquier protocolo, y cuando se aplican, a menudo se oye que es RESTFUL.



3.- Arquitecturas de API

3.4 REST (continua)

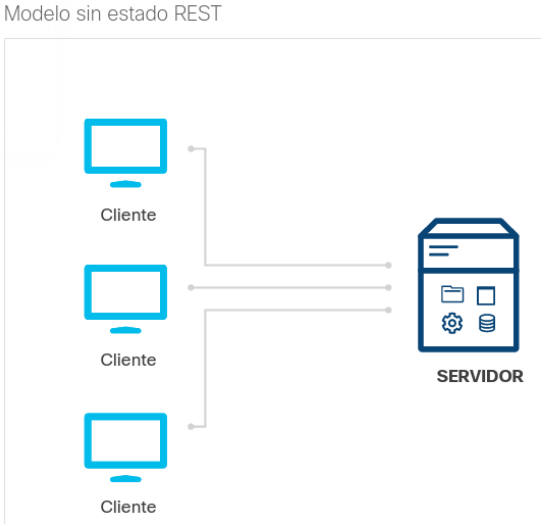
Modelo Cliente - Servidor Rest

El cliente y el servidor deben ser independientes entre sí, permitiendo que el cliente se construya para múltiples plataformas y simplificando los componentes del lado del servidor.

3.- Arquitecturas de API

3.4 REST (continua)

Sin estado



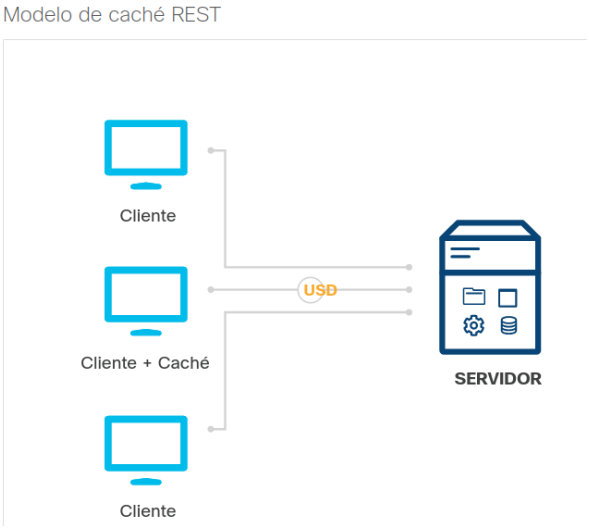
Las solicitudes del cliente al servidor deben contener toda la información que el servidor necesita para realizar la solicitud. El servidor no puede contener estados de sesión.

CISCO DevNet Associate v1.0

3.- Arquitecturas de API

3.4 REST (continua)

Cache



Las respuestas del servidor deben indicar si la respuesta se puede almacenar en caché o no en caché. Si se puede almacenar en caché, el cliente puede utilizar los datos de la respuesta para solicitudes posteriores.

CISCO DevNet Associate v1.0

3.- Arquitecturas de API

3.4 REST (continua)

Interface uniforme

La interfaz entre el cliente y el servidor debe cumplir con estos cuatro principios:

Identificación de recursos: Un recurso puede ser cualquier información, como un documento, una imagen, una persona, una colección de otros recursos, etc. Por ejemplo, en la solicitud para cambiar una contraseña para un usuario, el usuario individual debe ser identificado.

Manipulación de recursos a través de representaciones: el cliente recibe una representación del recurso del servidor. Esta representación debe contener suficientes datos o metadatos para que el cliente pueda manipular el recurso. Por ejemplo, una solicitud destinada a rellenar el perfil de un usuario debe incluir la información del perfil.

Mensajes autodescriptivos: cada mensaje debe contener toda la información para que el destinatario procese el mensaje. Ejemplos de información pueden ser: El tipo de protocolo, El formato de datos del mensaje, La operación solicitdada

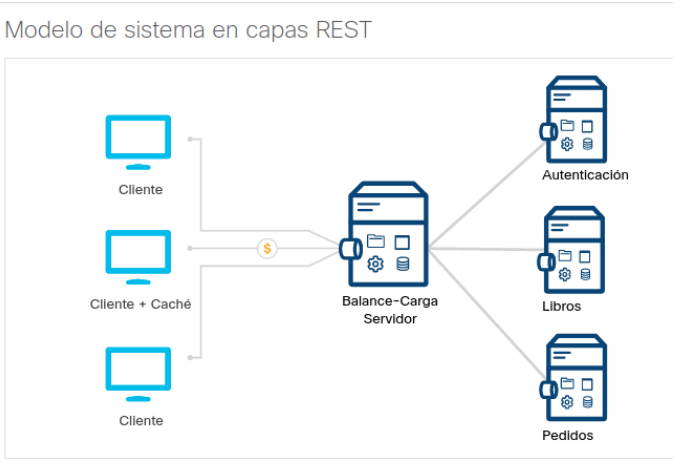
Hypermedia como motor del estado de la aplicación: los datos enviados por el servidor deben incluir acciones y recursos adicionales disponibles para que el cliente acceda a información adicional sobre el recurso.

3.- Arquitecturas de API

3.4 REST (continua)

Sistema por capas

El sistema se compone de diferentes capas jerárquicas en las que cada capa proporciona servicios sólo a la capa que está encima.



CISCO DevNet Associate v1.0

4.- Introducció a API REST

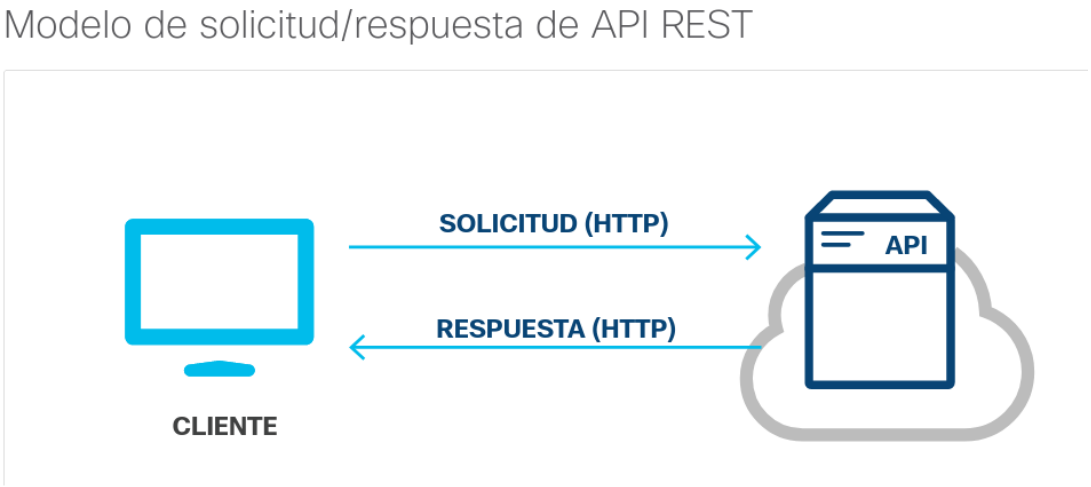
4.1. API de servicio web REST

Una API de servicio web REST (API REST) es una interfaz de programación que se comunica a través de HTTP mientras se adhiere a los principios del estilo arquitectónico REST.

Utiliza los principios del estilo arquitectónico REST.

Debido a que las API REST se comunican a través de HTTP, utilizan los mismos conceptos que el protocolo HTTP:

- Solicitudes/respuestas HTTP
- Verbos HTTP
- Código de estado HTTP
- Encabezados/cuerpo HTTP



4.- API de servicio web REST

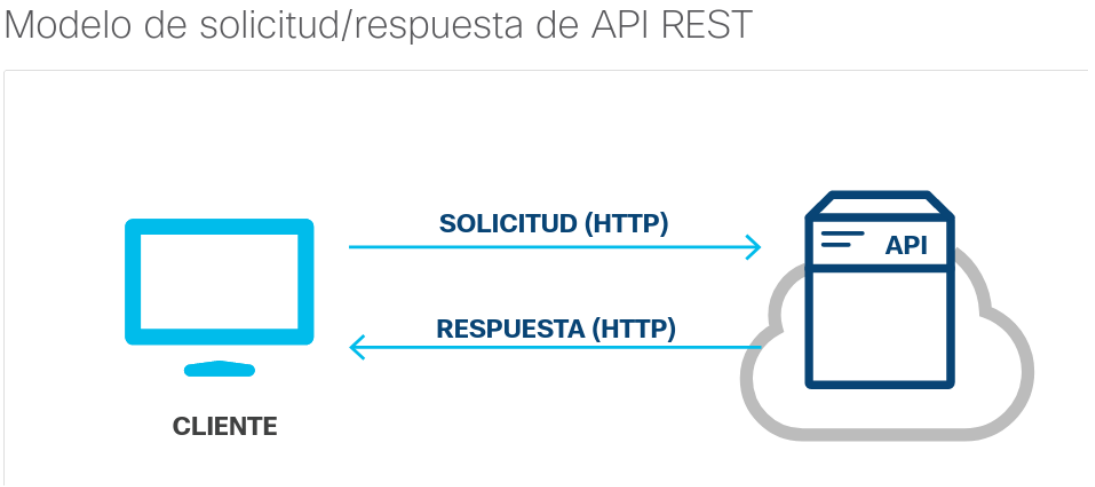
4.2. Solicitudes de API REST

Solicitudes de API REST

Las solicitudes de API REST son esencialmente solicitudes HTTP que siguen los principios REST. Esta solicitudes son una forma de que una aplicación (cliente pida al servidor que realice una función. Debido a que e una API, estas funciones están predefinidas por el servidor y deben seguir la especificación proporcionada

Las solicitudes de API REST se componen de cuatro componentes principales:

- Identificador uniforme de recursos (URI)
- Método HTTP:
- Encabezado
- Cuerpo



4.- API de servicio web REST

4.2. Solicitudes de API REST(continua)

Identificador uniforme de recursos (URI)

El identificador uniforme de recursos (URI) se conoce a veces como localizador uniforme de recursos (URL). El URI identifica qué recurso quiere manipular el cliente. Una solicitud REST debe identificar el recurso solicitado; la identificación de recursos para una API REST suele ser parte del URI.

Un URI tiene esencialmente el mismo formato que la URL que utiliza en un navegador para ir a una página web. La sintaxis consta de los siguientes componentes en este orden particular: Esquema, Autoridad, Ruta, Consulta

Al unir los componentes, un URI se verá así: `esquema: [//authority] [/path] [?consulta]`

Diferentes componentes de un identificador uniforme de recursos (URI)

4.- API de servicio web REST

4.2. Solicitudes de API REST(continua)

Método HTTP

Las API REST utilizan los métodos HTTP estándar, también conocidos como verbos HTTP, como una forma de decirle al servicio web qué acción se solicita para el recurso dado. No hay un estándar que defina qué método HTTP se asigna a qué acción, pero la asignación sugerida se ve así:

Método HTTP:	Acción	Descripción
POST	Crear (Create)	Cree un nuevo objeto o recurso.
GET	leer	Recuperar detalles de recursos del sistema.
PUT	Actualizar	Reemplazar o actualizar un recurso existente.
PATCH	Actualización parcial	Actualice algunos detalles de un recurso existente.
DELETE	Eliminar (Delete)	Quite un recurso del sistema.

4.- API de servicio web REST

4.3. Respuestas de API REST

Las respuestas de la API REST son esencialmente respuestas HTTP. Estas respuestas comunican los resultados de la solicitud HTTP de un cliente.

La respuesta puede contener los datos solicitados, indicar que el servidor ha recibido su solicitud, o incluso informar al cliente que ha habido un problema con su solicitud.

Las respuestas de la API REST son similares a las solicitudes, pero se componen de tres componentes principales: Estado HTTP, Encabezado y Cuerpo

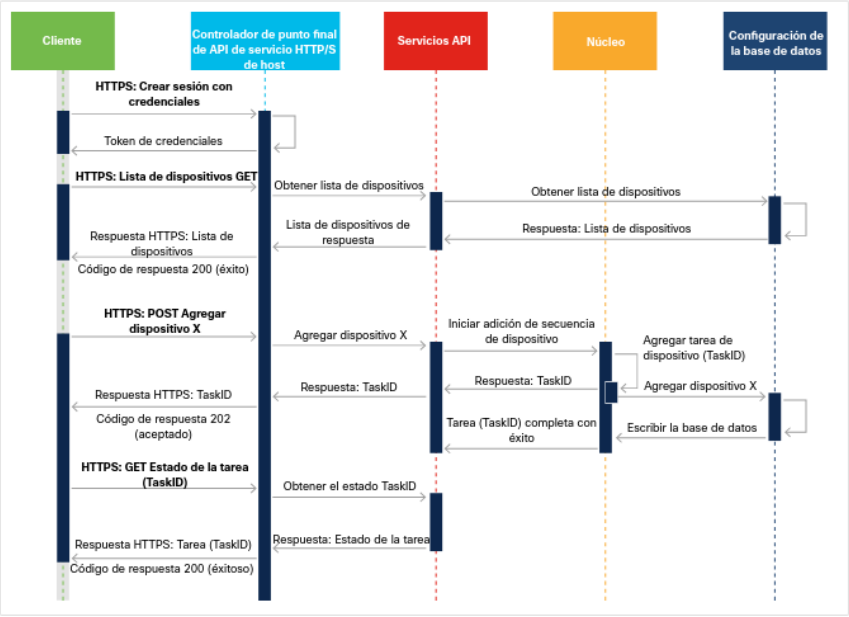
Código común de estados HTTP

Código de Estado HTTP	Mensaje de estado	Descripción
200	Aceptada	La solicitud se realizó correctamente y normalmente contiene una carga útil (cuerpo)
201	Creada	Se cumplió la solicitud y se creó el recurso solicitado
202	Aceptada	La solicitud ha sido aceptada para su procesamiento y está en proceso
400	Solicitud no válida	La solicitud no se procesará debido a un error con la solicitud
401	No autorizado	La solicitud no tiene credenciales de autenticación válidas para realizar la solicitud
403	Prohibida	La solicitud ha sido entendida pero ha sido rechazada por el servidor
404	No se encontró	No se puede cumplir la solicitud porque la ruta de acceso del recurso de la solicitud no se encontró en el servidor
500	Error del servidor interno	No se puede cumplir la solicitud debido a un error del servidor
503	El servicio no está disponible	No se puede cumplir la solicitud porque actualmente el servidor no puede manejar la solicitud

4.- API de servicio web REST

4.4. Uso diagrama secuencia con API REST

Diagrama de secuencia de petición/respuesta de API



Los diagramas de secuencia se utilizan para explicar una secuencia de intercambios o eventos. Proporcionan un escenario de un conjunto ordenado de eventos.

5.- Autenticando la API REST

5.1 Autenticación de API

“Autenticar” y “autentificar” pueden emplearse como sinónimos con el significado de 'acreditar o dar fe de que un hecho o un documento es verdadero o auténtico'.

Por razones de seguridad, la mayoría de las API REST requieren autenticación para que los usuarios aleatorios no puedan crear, actualizar o eliminar información de forma incorrecta o maliciosa, ni acceder a información que no debería ser pública.

Sin autenticación, una API REST permite a cualquier persona acceder a las características y servicios del sistema que se han expuesto a través de la interfaz. Exigir autenticación es el mismo concepto que requerir una credencial de nombre de usuario/contraseña para acceder a la página de administración de una aplicación.

5.- Autenticando la API REST

5.2 Autenticación vs Autorización

Autenticación

La autenticación demuestra la identidad del usuario

La **autenticación** es el acto de verificar la identidad del usuario. El usuario está demostrando que es quien dicen ser. Por ejemplo, cuando vas al aeropuerto, tienes que mostrar tu identificación emitida por el gobierno o usar biometría para demostrar que eres la persona que afirmas ser.

Autorización es el usuario que prueba que tiene los permisos para realizar la acción solicitada en ese recurso. Por ejemplo, cuando vas a un concierto, todo lo que necesitas para mostrar es tu entrada para demostrar que estás permitido entrar. No necesariamente tienes que demostrar tu identidad.

5.- Autenticando la API REST

5.3 Mecanismos de atenticación

Los tipos comunes de mecanismos de autenticación incluyen Basic, Bearer y API Key.

En una solicitud de API REST, la información de autenticación se proporcionará en el encabezado:

1) Autorización básica <username>:<password>
2) Autorización: Portadora <token>
3) Autorización: apiKey <API Key>. Es que el que usar Google Maps.

5.- Autenticando la API REST

5.3 Mecanismos de autorización

La autorización abierta, también conocida como OAuth, combina la autenticación con la autorización.

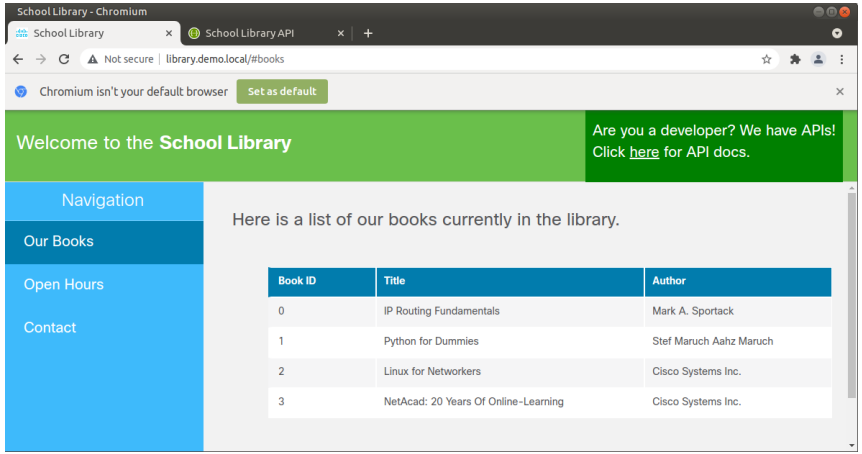
OAuth fue desarrollado como una solución para mecanismos de autenticación inseguros. Es el que usa github.

Con una mayor seguridad en comparación con las otras opciones, suele ser la forma recomendada de autenticación/autorización para las API REST.

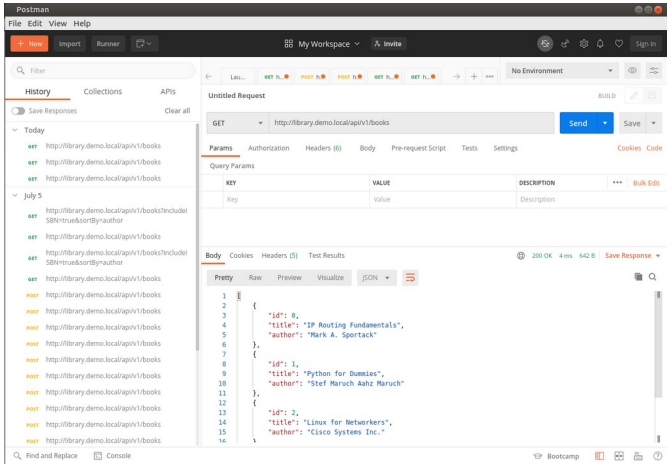
Este proceso de obtención del token se denomina flujo. A continuación, la aplicación utiliza este token en la API REST como autenticación al portador. El servicio web para la API REST comprueba entonces el servidor de autorización para asegurarse de que el token es válido y que el solicitante está autorizado para realizar la solicitud.

Lab. 4.5.5 Explore API REST con Simulador API Y Postman

API REST



POST MAN



Usar Máquina Virtual donde se encuentra una aplicación que implementa la School library API . Probaremos comando API REST GET, POST, DELETE, ETC.
- Aplicación WEB con API docs
- POSTMAN
- CURL
- PYTHON

DevNet Associate

6.- Limites de rangos API

6.1 Qué son los limites de tarifa

Las API REST permiten construir interacciones complejas.

El uso de un límite de velocidad de API es una forma de que un servicio web controle el número de solicitudes que un usuario o aplicación puede realizar por unidad de tiempo definida. Implementar límites de velocidad es una práctica recomendada para las API públicas y no restringidas. Rango de límite de ayudas:

- evitar una sobrecarga del servidor por demasiadas solicitudes a la vez
- proporcionar un mejor servicio y tiempo de respuesta a todos los usuarios
- proteger en contra de un ataque de denegación de servicio (DoS)

Los consumidores de la API deben entender las diferentes implementaciones de algoritmos para limitar la velocidad para evitar alcanzar los límites, pero las aplicaciones también deben manejar con gracia situaciones en las que se superan esos límites.

7.- Trabajar con Webhooks

6.1 Qué es un webhook

Un webhook es una devolución de llamada HTTP, o un HTTP POST, a una URL especificada que notifica a su aplicación cuando se ha producido una actividad o «evento» en particular en uno de sus recursos en la plataforma.

El concepto es simple. Piensa en preguntarle a alguien «dime de inmediato si X sucede». Ese «alguien» es el proveedor de webhook, y tú eres la aplicación.

Puede crear un webhook para que los equipos de Cisco Webex le notifiquen cada vez que se publiquen nuevos mensajes en una sala en particular. De esta forma, en lugar de hacer llamadas repetidas a la API de Teams para determinar si se ha publicado un nuevo mensaje, el webhook le notifica automáticamente cada mensaje. En este caso, Cisco Webex Teams es el proveedor webhook.