sensedia

Guía rápida de

Buenas prácticas y usos para las API

Su socio en un mundo conectado

Resumen:

Introducción

Las buenas prácticas en el uso de estado HTTP son fundamentales para asegurar una buena experiencia de consumo de las API por parte de los desarrolladores. Nuestra especialista en el área de Developer Experience, Luciana Bandeira, ha preparado un contenido que reúne toda la información esencial que su operación debe conocer.

Inclusive, estas buenas prácticas también se pueden aplicar en el **Portal de Desarrolladores**, apoyando y guiando al desarrollador en su jornada de consumo de las APIs.

¿Por qué tener un Portal de Desarrolladores optimizado?

El Portal del Desarrollador es una herramienta esencial para las estrategias de APIs abiertas o restringidas. Es un portal enfocado en compartir y difundir la documentación necesaria sobre el uso de las APIs, y un canal exclusivo y apto para la creación y administración de APPs.

Internamente, segregamos la información y formatos en dos partes:

Documentaciones funcionales: Flujos básicos, API Guide, Status Code. Se desarrolla por el consultor que está trabajando en el cliente o quien ha desarrollado la API.

Documentaciones técnicas: Publicando la API el módulo lee automáticamente el archivo JSON y arma dinámicamente los recursos, ejemplos de llamadas, etc. (datos que aparecen en la página API Browser).

Uno de los ítems que hemos visto a diario y que nos ha facilitado la comunicación y entendimiento para el desarrollo en el consumo de sus APIs es el correcto uso de **HTTP Status.**

La estandarización de los códigos de retorno que dará la API ayudará a su desarrollador y su integración, además de auxiliarlo a extraer métricas de manera más precisa.

Así mismo, poner a disposición esta información en su Portal de Desarrolladores hará que todos los que consuman estas llamadas conozcan estos estándares y la representación de retorno.

Códigos de Error del Cliente

Código HTTP	Descripción
400	Solicitud mal elaborada.
401	Solicitud necesita autenticación.
403	Solicitud rechazada.
404	Recurso no encontrado.
405	Método no permitido.
406	Contenido o Content-Type inválido.
408	Tiempo finalizado para la solicitud.
409	Recurso ya registrado.
413	Solicitud excede el tamaño máximo permitido.
415	Tipo de medio no soportado (falta de informar el Content-Type correcto).
422	Excepciones de negocio.
429	Solicitud excede la cantidad máxima de llamadas permitidas para la API.

Uso correcto de estado HTTP

Ya sabemos que muchos desarrolladores están muy atentos a todos los procesos de uso de las APIs. Estos estándares de código son conocidos en todo el mundo, y tornar esta información disponible junto con una explicación/ representación de lo que significará cada retorno dentro de su integración ayudará al desarrollador a comprender mejor los errores de uso.

Además, pone a disposición su propia integración para que estos consumos sigan estos estándares determinados desde el principio, garantizando incluso un mejor control y seguimiento de tales errores.

Vamos a suponer que en tu primera API disponible para el consumo, hayas seguido estos estándares de códigos, pero por alguna razón otro equipo puso a disposición una nueva API sin seguir también estos estándares. Esto puede generar divergencias en el uso por parte de los desarrolladores y también tendrá un impacto negativo en sus métricas.

Usando como ejemplo este escenario de estandarización, podemos suponer, por ejemplo, que tu desarrollador realiza un POST para ingresar información, pero que esta información ya ha sido registrada. Dejar como predeterminado solo un retorno, por ejemplo, a través de HTTP Status 400 y el mensaje de error que expone este escenario no mostrará métricas en este caso, pero será una descripción general de las diversas razones que pueden causar el 'bad request'.

Por lo tanto, se puede recomendar, por ejemplo, utilizar **HTTP Status** 409 (conflict) o 422 (unprocessable entity). De esta forma, el desarrollador

pronto sabrá que ese recurso en particular ya ha sido registrado y cuando tú, como propietario de la API, extraigas métricas, podrás distinguir este caso entre otros posibles 4XX de forma más fácil y asertiva.

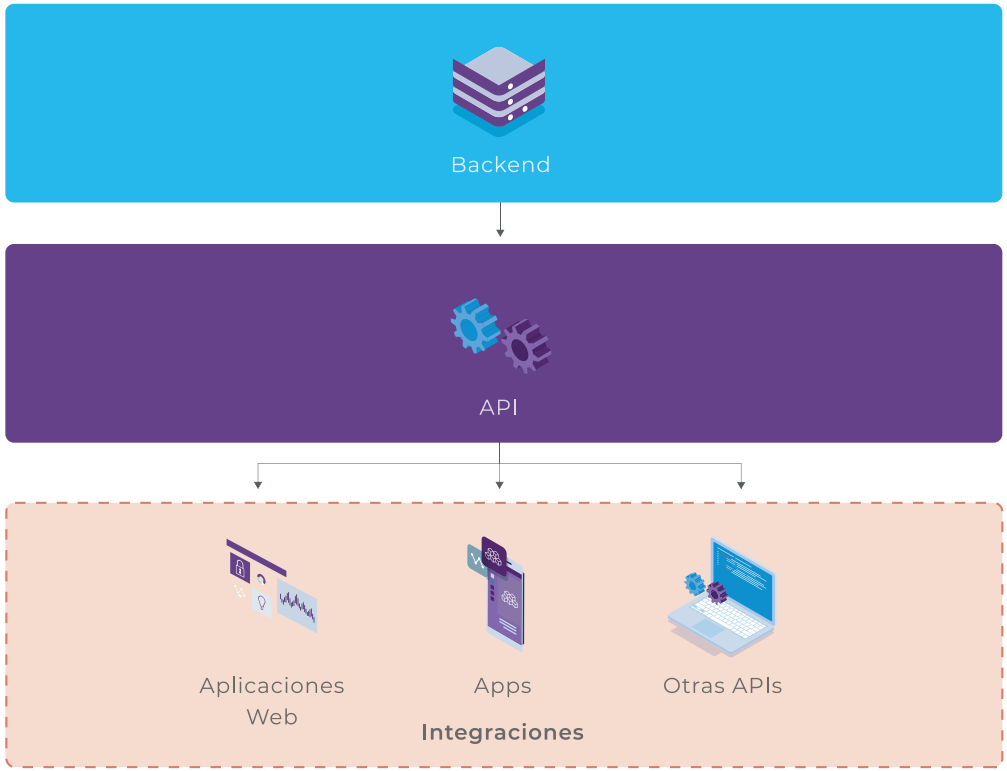
Como hemos visto, existen varios factores muy interesantes a los que se deben prestar atención para las buenas prácticas en el uso de APIs y también para alimentar Portales de Desarrolladores.

Poner a disposición una API va más allá de simplemente abrir una integración, es necesario pensar en su experiencia de consumo, y cuanto más clara y objetiva sea la comunicación, mejor se utilizará su API.

APIs y sus métodos

Como ya debes saber, la **API** es una comunicación simplificada y directa entre el desarrollador y su empresa, de forma segura y fácil de usar. Los métodos de uso de las APIs pueden ser en formas de pago, marketplaces, beneficios, comunicación y tráfico entre sistemas internos y en **otros diversos medios.**

Su comprensión se basa especialmente en el siguiente ejemplo ilustrativo:



OK.. ¿Y entonces?

A pesar de ser un tema muy claro para todos los que ya conocen y utilizan las APIs, las personas novatas en este tema (pero que quieren conocer y aprender a usar) pueden acabar teniendo algunas dudas al respecto.

Las operaciones siguen los estándares del protocolo HTTP para procesar llamadas y definen la acción esperada para cada uno de los métodos de uso de las APIs.

Métodos de uso de las APIs:

Get (Consulta/Búsqueda de datos)

Este método permite buscar informaciones/datos de acuerdo con recurso y endpoint ingresado. También puede (y se debe) utilizar este método junto con filtros y paginación del recurso. A través de este método el backend retornará las informaciones consultadas.

Post (Envío/Inserción)

Crea/ejecuta la carga de una nueva información a la base. Es necesario enviar la información en el estándar JSON requerido en el cuerpo de la solicitud.

Put y Patch (Actualización de datos)

Se utiliza para actualizar o editar información que ya existe en la base de datos. Algunas APIs contemplan, o no, el uso del patch (indicado para actualizaciones parciales), ya el uso de put es conocido de una forma general y es el responsable de cambiar/actualizar una información existente.

Delete (Borrar datos)

Como su nombre lo indica, es el método responsable de borrar una información. Este es un punto de atención donde solo se considera dentro de la API, si es para la regla del negocio se necesita (y está permitido) excluir información de la base. Algunas empresas no permiten este uso, ya que es una acción que borrará definitivamente el registro determinado.

Paginación y filtros

Paginación de las APIs

Siempre que sea necesario el retorno de más detalles sobre alguno de los atributos del recurso deseado, se puede utilizar el parámetro de paginación.

Este uso es necesario para evitar que la consulta ejecutada en la API se sobrecargue, provocando time-out al utilizar determinada solicitud. En general, todos los servicios que retornan una gran cantidad de datos siempre deben usar la paginación para devolver los registros.

Junto con la paginación, será necesario indicar la cantidad de información que se debe retornar en cada página. Con eso, se recomienda junto usar el parámetro de tamaño y tener este valor de tamaño inferior a 200, dependiendo del tipo de consulta y la información que se retornará en caso contrario.

El uso de esta información junto con la consulta ejecutada hace que la solicitud sea más rápida de leer y de regreso del backend, ocasionando así una experiencia más agradable y ágil para el desarrollador que está consumiendo la API.

En otras palabras, mejora la experiencia del usuario y evita el procesamiento de la información de forma muy extensa, con la posibilidad

de ocurrir fallas e incluso la interrupción de esta lectura (ocasionando el time-out, donde el backend no devuelve la información en el tiempo hábil necesario).

Un ejemplo práctico de cómo ingresar esta información en el endpoint es a través de page=0&size=100.

Vale la pena recordar que estas nomenclaturas de campo de parámetro pueden cambiar según la implementación de su API (puedes encontrar estos campos como Page y Size; Offset y Limit; Page y PageSize; entre otros).

Utilizando filtros

Otro gran apoyo para optimizar las consultas de información es el uso de filtros.

Poner a disposición este uso dentro de tu integración también es de gran ayuda cuando se trata de una mejor experiencia de uso de las APIs.

A través de este recurso, será posible que el desarrollador limite la información que desea que le retornen, optimizando las solicitudes ejecutadas para específicamente lo que se necesita.

Suponiendo que en la solicitud solo desea que se retorne la información que cumple con el estado de aprobado, puede ingresar en su solicitud el filtro status=approved (recordando que estas nomenclaturas dependerán de la implementación de su API), delimitando las devoluciones solo para este escenario.

Se pueden realizar muchas otras acciones, que en ocasiones incluso pueden parecer triviales, para ayudar al desarrollador que está implementando el consumo de APIs a utilizarlo correctamente, evitando así una sobrecarga de llamadas innecesarias e incorrectas además de no comprender los flujos y acciones, e incluso las buenas prácticas en el uso del estado HTTP.

A continuación, enumeramos 10 pasos que también pueden ayudarte en este proceso:

1. Extracción de métricas

La extracción de métricas de uso de las solicitudes ejecutadas te permitirá saber exactamente el volumen de uso y qué porcentaje de Errores X Éxitos, además de identificar los principales infractores de APPs/desarrollador, operación y tipo de error y, en consecuencia, tomar medidas para mejorar la experiencia y uso.

2. Alineación constante de errores

Realizar accionamientos al desarrollador, junto con evidencias y orientaciones de ajustes a tratar, hará que los escenarios de error se alineen y proporcionará información y guías de corrección para tratar la causa raíz. Suponiendo que tu desarrollador comete errores debido a alguna falla en el proceso, es posible alinearse lo antes posible junto con la evidencia y las pautas para la corrección interna en el desarrollo.

3. Seguimiento de las llamadas ejecutadas de error

Además de extraer e identificar los errores generados en el uso de APIs, también es importante realizar un seguimiento constante. De ese modo, será una buena acción para identificar la causa/motivo de tales errores y también evaluar si el desarrollador ha realizado las correcciones necesarias. Esta acción también ayudará en el ítem 2, donde será posible recolectar las evidencias para alinear con el desarrollador y supervisar la usabilidad.

4. Portal de Desarrolladores con documentaciones e insumos sobre API

El Portal de Desarrolladores es la entrada del desarrollador que desea consumir las APIs disponibles y a través del mismo deben estar disponibles las informaciones y guías de utilización.

5. Documentación clara y elaborada en el Portal de Desarrolladores

Esto respaldará todo el proceso de onboarding y hará que la comprensión sea más clara. Un Portal de Desarrolladores de fácil comunicación es aquel que contiene tanto lenguaje técnico así como también de negocio, para obtener una gama más amplia de información.

6. Inclusión de buenas prácticas en el Portal de Desarrolladores sobre el uso de API, reglas de negocio, posibles estados HTTP.

Algunas de las informaciones que vemos que se necesitan incluir en el Portal ayudarán al desarrollador a comprender las reglas de uso, como hemos visto anteriormente, de manera más objetiva y así poder incluir reglas y comprensión en el propio desarrollo.

7. Mensaje de retorno con información clara y explicativa del error

Esto facilitará que el desarrollador comprenda cuándo ocurre un error. Exponer un mensaje amigable cuando ocurre un error, además de estandarizar el estado HTTP, como ya se ha mencionado, ayudará a identificar errores y causas más fácilmente al desarrollador.

8. Desarrollo de reconocimiento de retornos por parte del desarrollador

Por ejemplo, que el desarrollador comience a identificar el error a través del estado http + mensaje de error expuesto y tomar las acciones necesarias, evitando así que el escenario continúe repetidamente.

9. Swagger con calidad

Un swagger bien detallado con la descripción de los campos, ejemplo de valores y todos los parámetros de consulta, ayudará a los desarrolladores y reducirá considerablemente el tiempo de "descubrimiento" de uso.

10. Entorno de pruebas

Para simular los procesos que se ejecutarán/utilizarán en producción. Así,

el desarrollador podrá realizar las pruebas necesarias y varias simulaciones, sin impacto al uso en el entorno de producción.

Concluyendo

Estos son algunos puntos que se deben tener en cuenta con relación a la experiencia que están teniendo los desarrolladores al consumir sus APIs, ya sean abiertas o privadas. Este material tuvo como objetivo enumerar los puntos principales para asegurar las buenas prácticas de las APIs y sus usos. Si deseas obtener más información sobre el tema, accede a nuestra página de contenido de Sensedia: sensedia.com/blog o comunícate directamente con nosotros.

Para que sigas estudiando hemos traído otras referencias de contenido

Developer Experience (DX) – Impulsa el uso de sus APIs.

¿Por qué es imporante tener un Portal de Desarrolladores para el negocio?

API Care: Qué es el seguimiento activo y por qué es importante para tus estrategias digitales.

Sobre a Sensedia

Fundada en 2007, Sensedia es un especialista en API con presencia en Latino América y Europa. Trabaja con líderes del mercado en diversos sectores y sus soluciones permiten a los clientes ampliar sus negocios digitales.

Es uno de los principales “pure players” de API en el mundo, centrado únicamente en su plataforma de gestión de API y en los servicios profesionales y de estrategia en torno al ciclo de vida completo de la gestión de API.

Sensedia está reconocida por Gartner en su Cuadrante Mágico como Visionaria y por Forrester en su Wave como Strong Performer. También es reconocida como una de las mejores soluciones de Open Banking en el mundo.

¿Le gustaría conocer nuestras soluciones?

Visítenos www.sensedia.com

sensedia

Su socio en un mundo **conectado**